TUGAS AKHIR - KI141502

PENINGKATAN KINERJA *RELIABLE MULTIPOINT RELAY* PADA OLSR DI VANET

IRFAN NAUFAL PRAWIRO SAMUDRO
NRP 5112100066

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

Dosen Pembimbing II
Ir. Muchammad Husni, M.Kom

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

[Halaman ini sengaja dikosongkan]

TUGAS AKHIR – KI1502

PENINGKATAN KINERJA *RELIABLE MULTIPOINT RELAY* PADA OLSR DI VANET

IRFAN NAUFAL PRAWIRO SAMUDRO
NRP 5112100066

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

Dosen Pembimbing II
Ir. Muchammad Husni, M.Kom

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

[Halaman ini sengaja dikosongkan]

UNDERGRADUATE THESIS – KI1502

PERFORMANCE IMPROVEMENT ON RELIABLE MULTIPLE RELAY OF OLSR IN VANET

IRFAN NAUFAL PRAWIRO SAMUDRO
NRP 5112100066

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

Supervisor II
Ir. Muchammad Husni, M.Kom

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA2017

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

PENINGKATAN KINERJA *RELIABLE MULTIPOINT RELAY* PADA OLSR DI VANET

TUGAS AKHIR

Diajukan Untuk 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
IRFAN NAUFAL PRAWIRO SAMUDRO
NRP : 5112 100 066

Disetujui oleh Dosen Pembimbing Tugas Akhir

1. Dr.Eng. Radityo Anggoro, S.Kom., M.Sc. .....
   NIP:198410162008121002 (Pembimbing 1)
2. Ir. Muchammad Husni ,M.Kom. .....
   NIP:196002211984031001 (Pembimbing 2)

SURABAYA
JUNI, 2017

[Halaman ini sengaja dikosongkan]

PENINGKATAN KINERJA *RELIABLE MULTIPOINT RELAY* PADA OLSR DI VANET

Nama Mahasiswa	**: IRFAN NAUFAL PRAWIRO SAMUDRO**
NRP	**: 5112100064**
Jurusan	**: Teknik Informatika FTIF-ITS**
Dosen Pembimbing 1	**: Dr.Eng. Radityo Anggoro, S.Kom., M.Sc**
Dosen Pembimbing 2	**: Ir. Muchammad Husni, M.Kom.**

Abstrak

Saat ini perkembangan teknologi Vehicular Ad Hoc Networks (VANETs) sudah semakin pesat. Mulai dari penggunaan pada kendaraan, maupun beberapa peralatan yang lain. Namun sampai saat ini penggunaan VANETs masih terkendala dengan metode routing mana yang paling efektif dan paling efisien. Salah satu metode routing yang paling sering digunakan adalah metode routing Optimize Link State Routing (OLSR) Namun terdapat kelemahan pada metode ini, yakni Topology Control(TC) akan meningkat jika dalam kondisi mobilitas tinggi hingga akan terjadi broadcast storm.

Peningkatan algoritma Prefered Group Broadcasting (PGB) pada protokol OLSR bertujuan untuk mengatasi permasalahan broadcast storm pada proses Topology Control(TC). Hasil menunjukkan bahwa protokol OLSR+PGB (optimasi) membuat nilai dari jumlah paket dari protokol yang dikirim lebih sedikit dari jumlah paket dari protokol OLSR maupun OLSR+PGB.

Kata kunci:OLSR, PGB, VANET, routing protokol

[Halaman ini sengaja dikosongkan]

PERFORMANCE IMPROVEMENT ON RELIABLE MULTIPLE RELAY OF OLSR IN VANET

Student's Name	**: IRFAN NAUFAL PRAWIRO SAMUDRO**
Student's ID	**: 5112100066**
Department	**: Teknik Informatika FTIF-ITS**
First Advisor	**: Dr.Eng. Radityo Anggoro, S.Kom., M.Sc**
Second Advisor	**: Ir. Muchammad Husni, M.Kom.**

Abstract

Nowadays, Vehicular Ad Hoc Networks (VANETs) Technology has been developed incredibly fast. Starting from the implementation on vehicles and the other tools. Unfortunately, there is a problem in using VANETs, which is choosing the most effective and efficient routing method. One of the routing method that is commonly used is Optimize Link State Routing (OLSR) routing method. But there are some weaknesses found in this method, that is the value of TC will increase if it is running in high mobility condition until it occurs broadcaststorm.

The Optimization of Prefered Group Broadcasting(PGB) algorithm on OLSR protocol is designed to solve the problem of broadcast storm in Topology Control(TC) process. The result show that the protocol of OLSR+PGB (optimize) has less number of sent Packet than OLSR protocol and OLSR+PGB.

Keyword:OLSR, PGB, VANET, routing protokol

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

Alhamdulillahirabbil'alamin, segala puji bagi Allah SWT,yang telah melimpahkan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul **"***PENINGKATAN KINERJA RELIABLE MULTIPOINT RELAY PADA OLSR DI VANET***".** Shalawat serta salam selalu senantiasa saya tujukan kepada Rasulullah SAW, selaku inspirator dunia nomor satu saat ini, hingga akhir zaman.

Pengerjaan Tugas Akhir ini merupakan salah satu dari sekian banyak kesempatan yang saya dapatkan, untuk mendapatkan ilmu dan pengalaman berharga selama saya berada di kampus Teknik Informatika ITS ini. Dengan pengerjaan Tugas Akhir ini, saya menjadi semakin bisa untuk memanajemen waktu dan memanajemen diri sendiri, sehingga Tugas Akhir ini dapat selesai tepat waktu.

Selesainya Tugas Akhir ini tidak lepas dari bantuan dan dukungan beberapa pihak. Sehingga pada kesempatan ini penulis mengucapkan syukur dan terima kasih kepada:

1. Allah SWT dan Nabi Muhammad SAW.
2. Ibu, Ibu, Ibu, Ayah dan Adik yang selalu memberikan do'a, dukungan, serta motivasi, sehingga penulis selalu termotivasi untuk menyelesaikan Tugas Akhir.
3. Bapak Dr.Eng. Radityo Anggoro, S.Kom., M.Sc, selaku pembimbing I yang selalu menyemangati dan memotivasi dengan ilmu-ilmu yang diluar dugaan saya.
4. Bapak Ir.Muchammad Husni, M.Kom selaku pembimbing II yang telah mengoreksi buku ini dengan cermat.
5. Hanafi,Kamali,Wimba atas suntikan semangatnya tiap hari.
6. Randy,Raga,Indra dan Ode, dan semua anak bimbing Pak Onggo atas bantuan dalam pengerjaan TA.
7. Mbak Iriyanti yang telah membantu dalam pengerjaan Buku Tugas Akhir ini.
8. Serta semua pihak yang yang telah turut membantu penulis dalam menyelesaikan Tugas Akhir ini.

Penulis menyadari bahwa Tugas Akhir ini masih memiliki banyak kekurangan. Sehingga dengan kerendahan hati, penulis mengharapkan kritik dan saran dari pembaca untuk perbaikan ke depannya.

Surabaya, Juni 2017

DAFTAR ISI

LEMBAR PENGESAHAN vii
Abstrak ix
Abstract xi
KATA PENGANTAR xiii
DAFTAR ISI xv
DAFTAR GAMBAR xix
DAFTAR TABEL xxiii
BAB I PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Rumusan Masalah 2
1.3 Batasan Masalah 2
1.4 Tujuan 2
1.5 Manfaat 2
1.6 Metodologi 2
1.7 Sistematika Penulisan Laporan Tugas Akhir 4
BAB II TINJAUAN PUSTAKA 7
2.1 VANET (*Vehicular Ad hoc Network*) 7
2.2 OLSR(*Optimized Link State Routing)* 8
2.3 PGB (*Preferred Group Broadcasting*) 9
2.4 OpenStreetMap 10
2.5 JOSM 11
2.6 TCL 11
2.7 AWK 12
2.8 NS2 dan Proses Instalasi NS2 12
BAB III PERANCANGAN 21
3.1 Deskripsi Umum 21
3.2Perancangan Skenario 22
3.2.1Pembuatan Peta Grid 22
3.2.2Pembuatan Peta Riil Surabaya 23
3.3 Perancangan Sumulasi pada NS2 24
3.4 Perancangan Protokol OLSR+PGB 24
3.5 Perancangan Metrik Analisis 27
3.5.1 *Packet Delivery Ratio* (PDR) 27

3.5.2 *End to end delay (E2E)*.....28
3.5.3 *Topology Control* (TC).....28
BAB IV IMPLEMENTASI.....29
4.1 Lingkungan Implementasi.....29
4.1.1 Lingkungan Perangkat Lunak.....29
4.1.2Lingkungan Perangkat Keras.....29
4.2 Implementasi Skenario.....29
4.2.1 Skenario Grid.....30
4.2.2 Skenario Riil.....33
4.3 Implementasi Algoritma PGB Pada OLSR.....34
4.3.1 Modifikasi Packet.h.....34
4.2.2 Modifikasi Wireless-phy.cc.....35
4.3.2 Modifikasi Wireless-phy.h.....37
4.3.3 Modifikasi Scheduler.cc.....37
4.3.4 Modifikasi scheduler.h.....39
4.3.4 Modifikasi OLSR.cc.....39
4.4 Implementasi Metrik Analisis.....41
4.4.1Implementasi *Packet Delivery Ratio*.....41
4.4.2Implementasi *Average End-to-End Delay*.....42
4.5 Implementasi Skenario Simulasi pada NS2.....43
BAB V UJI COBA DAN EVALUASI.....45
5.1 Lingkungan Uji Coba.....45
5.2 Hasil Uji Coba.....46
5.2.1 Hasil Uji Coba Grid.....46
5.2.1.1 Skenario Grid Kecepatan 10 m/s.....46
5.2.1.2 Skenario Grid Kecepatan 15 m/s.....49
5.2.1.3 Skenario Grid Kecepatan 20 m/s.....52
5.2.2 Hasil Uji Coba Peta Riil Surabaya.....54
5.2.2.1 Skenario *Real* Pada Kecepatan 10 m/s.....55
5.2.2.2 Skenario *Real* Pada Kecepatan 15 m/s.....58
5.2.2.3 Skenario Real Pada Kecepatan 20 m/s.....61
BAB VI KESIMPULAN DAN SARAN.....65
6.1 Kesimpulan.....65
6.2 Saran.....66
DAFTAR PUSTAKA.....67

BAB VII LAMPIRAN 69
A 1. Kode Skenario NS-2 69
A 2*Script Pattern* Skenario 71
A 3Kode awk Perhitungan *Packet Delivery Ratio* 72
A 4Kode awk Perhitungan *End-to-End Delay* 73
BIODATA PENULIS 75

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

Gambar2-1 Pembagian grup pada PGB 10
Gambar 2-2 Perintah untuk dependensi NS2 13
Gambar 2-3 Laman unduh untuk *berkas* NS2 13
Gambar 2-4 Perintah untukmengekstrak berkas NS2 14
Gambar 2-5 Perintah untuk melakukan edit pada ls.h 14
Gambar 2-6Contoh tampilan ls.h yang diedit 14
Gambar 2-7 Perintah untuk melakukan edit pada Makefile.in 14
Gambar 2-8 Tampilan dalam berkas Makefile.in 15
Gambar 2-9 Perintah install NS2 15
Gambar 2-10 Konfigurasi *Path* pada berkas .bashrc 16
Gambar 2-11 laman unduh file NS2.35 17
Gambar 2-12 cara ekstraksi file NS2.35 17
Gambar 2-13 masuk ke folder NS2.35 17
Gambar 2-14 laman unduh file patch OLSR 17
Gambar 2-15 proses patching OLSR ke dalam NS2 18
Gambar 2-16 proses instalasi 18
Gambar 2-17 masuk ke folder NS2 18
Gambar 2-18 membuat file untuk eksekusi 18
Gambar 2-19 cara menjalankan NS 19
Gambar3-1 Diagram rancangan simulasi 21
Gambar 3-2 Alur pembuatan peta grid 23
Gambar 3-3 Alur pembuatan peta riil Surabaya 24
Gambar 3-4 Proses *route discovery* pada OLSR 25
Gambar 3-5.*Flow chart OLSR+PGB* 27
Gambar 4-1 Perintah untuk membuat peta 30
Gambar 4-2 Peta hasil dari netgenerate 31
Gambar 4-3 Perintah untuk membuat *node* 31
Gambar 4-4 Perintah untuk membuat rute 31
Gambar 4-5 Contoh isi berkas sumocfg 32
Gambar 4-6 Perintah untuk mengkonversi peta menjadi xml 32
Gambar 4-7 Perintah untuk mengkonversi xml ke format tcl 32
Gambar 4-8 Proses capture peta Surabaya dengan OpenStreetMap 33

Gambar 4-9 Perintah konversi berkas osm menjadi format SUMO 34
Gambar 4-10 Modifikasi paket.h 35
Gambar 4-11Modifikasi wireless-phy.cc 36
Gambar 4-12Modifikasi fungsi sendUp 37
Gambar 4-13Modifikasi wireless-phy.h 37
Gambar 4-14Modifikasi pada fungsi schedulerDel 38
Gambar 4-15Modifikasi pada fungsi schedulex 38
Gambar 4-16Modifikasi scheduler.h 39
Gambar 4-17Modifikasi OLSR.cc 40
Gambar4-18 Modifikasi fungsi recv_OLSR 41
Gambar 4-19 Perintah untuk menjalankan skrip AWK penghitungan PDR 42
Gambar 4-20 Keluaran dari Skrip pdr.awk 42
Gambar 4-21 Perintah untuk menjalankan skrip AWK penghitungan endt to end delay 42
Gambar 4-22Keluaran dari Skrip endtoend.awk 43
Gambar 4-23 Perintah Untuk Menjalankan Skenario NS-2 43
Gambar 5-1 Grafik PDR dengan kecepatan 10 m/s 46
Gambar 5-2Grafik end to end kecepatan 10 m/s 47
Gambar 5-3 Grafik topology control kecepatan 10 m/s 48
Gambar 5-4 Grafik PDR dengan kecepatan 15 m/s 49
Gambar 5-5 Grafik end to end delay dengan kecepatan 15 m/s 50
Gambar 5-6 Grafik topology control dengan kecepatan 15 m/s.. 51
Gambar 5-7 Grafik PDR dengan kecepatan 20 m/s 52
Gambar 5-8 Grafik end to end delay dengan kecepatan 20 m/s. 53
Gambar 5-9Grafik topology control dengan kecepatan 20 m/s... 54
Gambar 5-10 Grafik PDR dengan kecepatan 10m/s 55
Gambar 5-11 grafik end to end delay dengan kecepatan 10 m/s 56
Gambar 5-12 grafik topology control dengan kecepatan 10 m/s 56
Gambar 5-13 grafik PDR dengan kecepatan 15 m/s 58
Gambar 5-14 grafik end to end delay dengan kecepatan 15 m/s 59
Gambar 5-15 grafik topology control dengan kecepatan 15 m/s 60
Gambar 5-16 grafik PDR dengan kecepatan 20 m/s 61
Gambar 5-17 grafik end to end delay dengan kecepatan 20 m/s 62

Gambar 5-18 grafik topology control dengan kecepatan 20 m/s 63

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

Tabel 5.1 Spesifikasi laptop yang digunakan 45
Tabel 5.2 Parameter Pengujian 45

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

1.1 Latar Belakang

Saat ini perkembangan Teknologi *wireless* telah didukung oleh berbagai perangkat *mobile*. Sehingga memungkinkan suatu perangkat mobile mengirimkan data ke perangkat lainnya. Perangkat *mobile* disini mencakup *netbook, tablet, smartphone* dan berbagai perangkat yang terhubung menggunakan teknologi. Dengan adanya kemampuan pengiriman data antara perangkat mobile tersebut, Maka memungkinkan perangkat-perangkat tersebut saling terhubung satu sama lain untuk saling bertukar data pada jaringan MANET (*Mobile Ad hoc Network*) ataupun VANET (*Vehicular Ad-hoc Networks*).

Pada penelitian kali ini akan digunakan jaringan VANET, karena VANET merupakan jenis khusus dari MANET karena memiliki mobilitas yang sangat tinggi Namun dalam implementasinya VANETs tersebut akan membentuk banyak *route* yang menghubungkan tiap *node* yang akan membuat beban routing menjadi besar, selain itu VANETs juga dapat mengalami terjadinya *Storm Broadcasting*. Untuk mengatasi permasalahan tersebut solusi yang dapat ditawarkan adalah dengan menggunakan teknik routing OLSR (*Optimized Link State Routing Protocol*). OLSR adalah optimasi protokol *routing* IP untuk jaringan *mobile ad hoc*, Yang juga dapat digunakan pada jaringan nirkabel. OLSR adalah *link-state routing* protokol proaktif, menggunakan path *forwarding hop* terpendek untuk mengirimkan data, Sehingga dapat mengurangi beban *routing* dan dapat mencegah terjadinya *Storm Broadcasting* .Namun sebelum dapat melakukan *routing* OLSR akan dilakukan penghitungan kekuatan signal dari *node* yang ada dengan menggunakan metode *broadcast* protokol PGB (*Preferred Group Broadcasting*). Dengan meningkatkan kinerja PGB dan metode routing OLSR maka didapatkan satu rute pengiriman paket data yang lebih efisien, Mampu memperkecil beban *routing* yang diperlukan dan

akan menghasilkan metode routing baru yaitu OLSR+PGB. Metode *routing*OLSR+PGB akan diuji pada simulator NS2 agar paket data yang dikirim hanya melewati sedikit node dan mendapatkan hasil yang optimal.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam tugas akhir ini dapat dipaparkan sebagai berikut:

1. Bagaimana meningkatkan kinerja metode PGB kedalam protokol routing OLSR pada jaringan VANETs dengan menggunakan NS2?
2. Bagaimana mencegah terjadinya *storm broadcasting* pada OLSR?.

1.3 Batasan Masalah

Permasalahan yang dibahas dalam tugas akhir ini memiliki beberapa batasan antara lain:

1. Simulator yang digunakan adalah simulator NS2.
2. Metode routing yang digunakan adalah metode OLSR.
3. Metode yang digunakan untuk mengatasi *Broadcast Storm problem* adalah PGB(modifikasi).

1.4 Tujuan

Tujuan dari Tugas Akhir ini adalah Meningkatkan kinerja Protokol PGB kedalam metode *routing*OLSR untuk mencegah terjadinya *Storm Broadcasting*.

1.5 Manfaat

Dengan mencegah terjadinya *Storm Broadcasting*, diharapkan akan mampu mengurangi jumlah TC, *end-to-end Delay* dan meningkatkan PDR (*Packet Delivery Ratio*).

1.6 Metodologi

1. Penyusunan proposal tugas akhir

   Penyusunan Proposal Tugas Akhir ini merupakan tahap awal dalam proses pengerjaan Tugas Akhir. Dalam

proposal ini akan dilakukan peningkatan kinerja algoritma PGB (*Preferred Group Broadcasting*) pada metode routing OLSR (*Optimized Link State Routing Protocol*).

2. Studi literatur

   Tugas Akhir ini menggunakan literatur paper dan buku. Paper yang digunakan adalah "*An Evaluation of Inter-Vehicle Ad Hoc Networks Based on Riilistic Vehicular Traces*","Peningkatan Kinerja *Protokol Optimized Link State Routing* pada lingkungan *Vehicular ADHOC Network* dengan menggunakan prediksi Mobilitas dan *Multipath routing*". Paper tersebut menjelaskan mengenai implementasi algoritma *broadcasting Preferred Group Broadcasting* pada metode routing *Ad hoc On-Demand Distance Vector* dan cara untuk meningkatkan kinerja protokol pada OLSR menggunakan prediksi mobilitas.

3. Implementasi

   Pada tahap ini dilakukan implementasi dari metode PGB (*Preferred Group Broadcasting*) dan metode routing OLSR (*Optimized Link State Routing Protocol*) untuk mengatasi masalah *broadcasting strom* dengan menggunakan simulator NS2.

4. Pengujian dan evaluasi

   Pengujian dilakukan dengan menjalankan skenario simulasi dengan beberapa paremeter pada NS2 dengan metode *routing* OLSR + PGB(setelah melakukan peningkatan kinerja). Evaluasi dilakukan dengan melihat hasil dari implementasi.Parameter yang digunakan adalah:

   1. PDR (*packet delivery ratio*)

      PDR adalah jumlah paket data yag diterima dibagi dengan jumlah paket data yang dikirimkan

   2. *End-to-end delay*

      End-to-end delay adalah waktu kedatangan paket data pada destination dikurangi waktu paket data yang dikirimkan oleh source

3. *Topology Control*
   Topology Control merupakan jenis kontrol pada protocol OLSR yang dapat dikirim ke seluruh jaringan secara *broadcasting*.

5. Penyusunan Buku Tugas Akhir
   Pada tahap ini dilakukan penyusunan laporan yang menjelaskan dasar teori dan metode yang digunakan dalam tugas akhir ini serta hasil dari implementasi aplikasi perangkat lunak yang telah dibuat.

1.7 Sistematika Penulisan Laporan Tugas Akhir

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

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 Dasar Teori
Bab ini berisi penjelasan secara detail mengenai dasar-dasar penunjang dan teori-teori yang digunakan untuk mendukung pembuatan Tugas Akhir ini.

Bab III Perancangan Perangkat Lunak
Bab ini berisi tentang desain sistem yang disajikan dalam bentuk *pseudocode*

Bab IV Implementasi
Bab ini membahas implementasi dari desain yang telah dibuat pada bab sebelumnya. Penjelasan berupa code 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.

[Halaman ini sengaja dikosongkan]

BAB II
TINJAUAN PUSTAKA

Bab ini berisi penjelasan teori-teori yang berkaitan dengan rancangan alat yang diajukan pada pengimplementasian program. Penjelasan ini bertujuan untuk memberikan gambaran secara umum terhadap alat yang dirancang dan berguna sebagai penunjang dalam pengembangan perangkat lunak.

2.1 VANET (*Vehicular Ad hoc Network*)

Mobile ad hoc network (MANET) adalah sekumpulan mobile *node*s yang secara dinamis lokasinya sering berubah sedemikian rupa hingga interkoneksi antar *node*s mampu mengatasi perubahan yang terjadi secara terus menerus. *Routing protocol* digunakan untuk menemukan rute antar *node*s agar komunikasi antar jaringan bisa terfasilitasi. Tujuan utama dalam *ad-hoc network routing protocol* adalah membetulkan dan meng-efisisensikan rute yang sudah ada diantara pasangan *node*s sehingga pesan bisa dikirim dengan waktu yang lebih singkat. Konstruksi route didesain dengan *overhead* dan konsumsi *bandwith* minimal[2].

VANET berasal dari MANET. Namun VANET dikhususkan pada penggunaan di kendaraan. Sebuah jaringan terorganisir yang dibentuk dengan menghubungkan kendaraan dan RSU (*Roadside Unit*) disebut *Vehicular Ad Hoc Network* (VANET), dan RSU lebih lanjut terhubung ke jaringan *backbone* berkecepatan tinggi melalui koneksi jaringan. Kepentingan peningkatan baru-baru ini telah diajukan pada aplikasi melalui V2V (*Vehicle to Vehicle*) dan V2I (*Vehicle to Infrastructure*) komunikasi, bertujuan untuk meningkatkan keselamatan mengemudi dan manajemen lalu lintas sementara menyediakan driver dan penumpang dengan akses Internet. Dalam vanets, RSUdapat memberikan bantuan dalam menemukan fasilitas seperti restoran dan pompa bensin, dan melakukan *broadcast* pesan yang terkait seperti

(maksimum kurva kecepatan) pemberitahuan untuk memberikan pengendara informasi. Sebagai contoh, sebuah kendaraan dapat berkomunikasi dengan lampu lalu lintas cahaya melalui V2I komunikasi, dan lampu lalu lintas dapat menunjukkan ke kendaraan ketika keadaan lampu ke kuning atau merah. Ini dapat berfungsi sebagai tanda pemberitahuan kepada pengemudi, dan akan sangat membantu para pengendara ketika mereka sedang berkendara selama kondisi cuaca musim dingin atau di daerah asing. Hal ini dapat mengurangi terjadinya kecelakaan. Melalui komunikasi V2V, pengendara bisa mendapatkan informasi yang lebih baik dan mengambil tindakan

Awal untuk menanggapi situasi yang abnormal. Untuk mencapai hal ini, suatu OBU (*On-Boards Unit*) secara teratur menyiarkan pesan yang terkait dengan informasi dari posisi pengendara, waktu saat ini, arah mengemudi, kecepatan, status rem, sudut kemudi, 6 lampu sen, percepatan / perlambatan, kondisi lalu lintas.

2.2 OLSR(*Optimized Link State Routing)*

*Optimized Link State Routing(*OLSR) adalah sebuah *protocol routing* untuk IP yang di optimisasi untuk jaringan *mobile ad-hoc*, yang juga dapat digunakan pada jaringan *wireless ad-hoc*. OLSR adalah sebuah *protocol link-state routing* yang *proactif*, yang artinya dia akan selalu mengupdate tabel routing secara otomatis tanpa ada permintaan terlebih dahulu

Perubahan topologi pada suatu jaringan dapat menyebabkan luapan informasi topologi yang berujung overhead pada semua node yang ada di jaringan. Untuk mencegah terjadinya *over head* digunakan teknik *Multi Point Relay* (MPR). Tujuan utama dari MPR adalah untuk mengurangi luapan dengan cara memilih beberapa node yang bertindak sebagai MPR. Sehingga hanya node MPR saja yang

akan meneruskan pesan yang diterima dengan kata lain juga dapat membuat rute menjadi semakin pendek.

OLSR menggunakan 2 jenis kontrol, yaitu pesan "HELLO" dan "*Topology Control* (TC)". Pesan HELLO digunakan untuk menemukan info tentang kondisi link dan node tetangga. Pesan HELLO juga digunakan untuk memilih MPR *selection set*. Tugas dari MPR *selection set* adalah memilih *node* tetangga untuk bertindak sebagai node MPR. Sehingga dengan adanya pesan hello ini *node* pengirim dapat menentukan *node* MPR-nya. Pesan hello hanya dapat dikirim sejauh 1*hop*, tetepi pesan TC dapat dikirim ke seluruh jaringan secara broadcasting. Fungsi dari pesan TC ini adalah untuk menyebar informasi tentang *node* tetanga mana saja yang telah digunakan sebagain *node* MPR. *Node* TC hanya disebar satu periodik dan hanya n*ode* MPR yang dapat meneruskan pesan TC tersebut

2.3 PGB (*Preferred Group Broadcasting*)

Preferred Group Broadcasting atau disingkat dengan PGB adalah sebuah *broadcasting mechanism* yang bertujuan untuk mereduksi *broadcast overhead* [4]. PGB akan mengurangi tranmisi yang redundan untuk menemukan *route* yang stabil dengan kemampuan *auto-correct* [1]. PGB membuat *node* penerima menentukan apakah akan melakukan *broadcast* ataukah membiarkan saja. PGB akan membuat tiga grup yaitu *Preferred group* (PG), *IN group*, dan *OUT group*. PG adalah kumpulan *node* yang akan melakukan *broadcasting*. *IN group* adalah *group* yang memiliki sinyal yang lebih kuat dari pada PG dan *OUT group* adalah grup yang memiliki sinyal yang lebih lemah dari pada PG. Ilustrasi pembagian grup bisa dilihat pada Gambar2-1[1].

Pembagian ini bertujuan untuk menentukan besaran *delay* yang akan digunakan kepada masing-masing paket. Setiap grup akan memiliki *delay* masing-masing bergantung pada posisi dan *signal strength* yang diterima oleh *node* penerima

dari *node* pengirim. Penghitungan *delay* setiap grup berbeda sesuai dengan posisi mereka.

Persamaan untuk penghitungan delay bisa dilihat pada

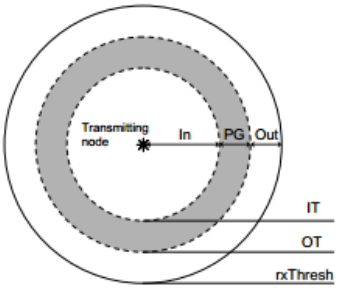
$$\boldsymbol{hoTime = T_i + jitterT_i} \quad (2.1)$$

$$\boldsymbol{\Delta P_i = \left||rxTh - rxP|-f_i\right|} \quad (2.2)$$

$$\boldsymbol{T_i = \Delta P_i . dt_i + minT_i} \quad (2.3)$$

$$jitterT_i \in [0, dt_i)$$

Pada Tugas Akhir ini algoritma PGB akan digunakan untuk mengatasi masalah *broadcast storm* dalam proses *route discovery*. Metode yang digunakan yaitu dengan mengatur delay pada setiap paket yang akan dikirim. Hal ini memanfaatkan sifat dari NS2 dimana setiap paket atau setiap aktivitas yang dilakukan akan dianggap sebagai sebuah even.



Gambar2-1 Pembagian grup pada PGB[1]

2.4 OpenStreetMap

Open Street Map (OSM) adalah proyek nirlaba yang dilakukan oleh *OpenStreetMap Foundation* yang berada di *United Kingdom*. *OpenStreetMap Project* berbasis di *OpenStreetMap.org* adalah upaya pemetaan seluruh dunia yang mencakup lebih dari dua juta relawan di seluruh dunia. Sifat dari *openstreetmap* adalah *Open* Data sehingga setiap data yang terdapat pada *OpensStreetMap* dapat digunakan

dan diubah secara bebas untuk keperluan apapun termasuk keperluan navigasi[6].

Melalui *Open Data Commons Open Database License* 1.0, kontributor OSM dapat memiliki, memodifikasi, dan membagikan data peta secara luas. Terdapat beragam jenis peta digital yang tersedia di internet, namun sebagian besar memiliki keterbatasan secara legal maupun teknis. Hal ini membuat masyarakat, pemerintah, peneliti dan akademisi, inovator, dan banyak pihak lainnya tidak dapat menggunakan data yang tersedia di dalam peta tersebut secara bebas. Di sisi lain, baik peta dasar OSM maupun data yang tersedia di dalamnya dapat diunduh secara gratis dan terbuka, untuk kemudian digunakan dan diredistribusikan kembali.Pada Tugas akhir ini, *OpenStreetMap* digunakan untuk membentuk peta Surabaya.

2.5 JOSM

JOSM (Java *OpenStreetMap Editor*) adalah alat penyunting untuk data hasil dari *OpenStreetMap*. JOSM tidak membutuhkan koneksi internet. Penggunaan JOSM membutuhkan installasi Java sebelumnya. Pada tugas akhir ini JOSM digunakan untuk mengedit petayang didapat dari *OpenStreetMap*[7].

2.6 TCL

TCL *(Tool Command Language)* atau sering diucapkan sebagai "*tickle*" adalah bahasa pemrograman yang diciptakan oleh John Ousterhout, pendiri Electric Cloud.Inc (bersama John Graham Cumming) dan seorang professor komputer di Universitas Stanford. Sebelumnya Ousterhout adalah professor komputer di Berkeley California, disanalah dia pertama sekali menciptakan bahasa pemrograman Tcl dan *platform* Tk sebagai *widgettoolkit* independen untuk mengembangkan aplikasi *desktop*. Pada saat ini penggunaan

Tcl sebagai alat untuk melakukan pengecekan/ujicoba secara otomatis pada *hardware* ataupun *software* cukup popular

2.7 AWK

Awk adalah bahasa pemrograman yang digunakan untuk melakukan manipulasi data dan membuat laporan[8]. Awk adalah sebuah pemrograman *filter* untuk teks, seperti halnya perintah "grep". Awk dapat digunakan untuk mencari bentuk/model dalam sebuah berkas teks. Awk juga dapat mengganti bentuk satu *teks* kedalam bentuk *teks* lain. Awk dapat juga digunakan untuk melakukan proses aritmatika seperti yang dilakukan oleh perintah "expr". Awk adalah sebuah pemrograman seperti pada shell atau C yang memiliki karakteristik yaitu sebagai *tool* yang cocok untuk *jobs* juga sebagai pelengkap (*complicated*) untuk *filter standard*. Pada tugas akhir ini AWK digunakan untuk membuat *script* untuk menghitung PDR, *delay*, dan *routing overhead* dari hasil *trace* NS-2.

2.8 NS2 dan Proses Instalasi NS2

NS adalah simulator diskrit yang digunakan pada penelitian jaringan. NS memberikan dukungan substansial untuk simulasi TCP, *routing*, dan protokol *multicast* melalui jaringan kabel dan nirkabel (lokal dan satelit) [9]. NS@ adalah varian dari produk NS dan penerus dari versi sebelumnya yaitu NS1. NS2 dibuat dengan menggunakan bahasa pemrograman C++ dan untuk melakukan konfigurasi menggunakan O Tcl.

NS2 telah digunakan dalam banyak penelitihan, telah tervalidasi dan terverifikasi pada banyak paper internasional. NS2 dikembangkan oleh University of California Berkeley dan USC ISI sebagai bagian dari projek Virtual INternet Testbed (VINT). Network Simulator pertama kali dibangun sebagai varian dari *REAL Network Simulator* pada tahun 1989 di University of California Berkeley (UCB). Pada tahun 1995

pembangunan *Network Simulator* didukung oleh *Defense Advanced Research Project Agency* (DARPA) melalui VINT *Project*, yaitu sebuah tim riset gabungan yang beranggotakan tenaga-tenaga ahli dari beberapa instansi ternama. NS juga bersifat *open source* dibawah Gnu Public License (GPL), sehingga NS dapat di-download dan digunakan secara gratis melalui web site NS yaitu http://www.isi.edu/nsnam/. Sifat *open source* juga mengakibatkan pengembangan NS menjadi lebih dinamis.Struktur NS. Ns dibangun menggunakan metoda *object oriented* dengan bahasa C++ dan OTcl (variant object oriented dari Tcl).

2.8.1 Instalasi NS2.35 Normal

Sebelum melakukan instalasi NS2, terlebih dahulu dilakukan instalasi dependensi untuk mendukung proses instalasi. Salah satu dependensi yang dibutuhkan yakni gcc-4.3.

```
sudo apt-get install gcc-4.4 build-essential
autoconf automake libxmu-dev
```

Gambar 2-2 Perintah untuk dependensi NS2

Namun karena gcc-4.3 sudah tidak ada maka bisa diganti dengan gcc-4.4. berikut potongan *command* diterminal untuk melakukan instalasi dependensi NS2. Setelah instalasi semua dependensi lengkap silahkan unduh *berkas* NS2 pada laman yang tertera pada Gambar 2-3

```
https://sourceforge.net/projects/nsnam/fil
es/latest/download
```

Gambar 2-3 Laman unduh untuk *berkas* NS2

Setelah semua selesai terunduh buat direktori sebagai tempat *berkas* NS2 nanti dan pindahkan *berkas* hasil unduhan tadi ke dalam direktori tersebut. Disini direktori yang dibuat bernama “workplace”

```
$ cd workplace
$ tar xjf ns-allinone-2.35.tar.gz
```

Gambar 2-4 Perintah untukmengekstrak berkas NS2

Sebelum dilakukan beberapa modifikasi pada dua berkas yakni berkas “ls.h” pada direktori “linkstate” dan berkas “Makefile.in” pada direktori “otcl-1.13”.Berikut langkah untuk melakukan edit berkas ls.h pada Gambar 2-5

```
$ cd ns-allinone-2.35/ns-2.35/linkstate
$ gedit ls.h
```

Gambar 2-5 Perintah untuk melakukan edit pada ls.h

Setelah tampil kotak dialog gedit silahkan menuju baris ke 137 dan lakukan perubahan pada kata “erase” ubah menjadi “this -> erase”. Contoh perubahan bisa dilihat pada Gambar 2-6

```
  void eraseAll() { this->erase(baseMap::begin(), baseMap::end()); }
  T* findPtr(Key key) {
      iterator it = baseMap::find(key);
      return (it == baseMap::end()) ? (T *)NULL : &((*it).second);
  }
;
```

Gambar 2-6Contoh tampilan ls.h yang diedit

Setelah itu, ubah berkas “Makefile.in” pada direktori “otcl-1.13” untuk memberikan informasi pada NS2 mengenai versi gcc yang digunakan. Berikut perintahnya.

```
Sudo gedit ns-allinone-2.34/otcl-
1.13/Makefile.in
```

Gambar 2-7 Perintah untuk melakukan edit pada Makefile.in

Setelah kotak dialog tampil, silahkan ganti CC= @CC@ menjadi CC=gcc-4.4 seperti Gambar 2-8. Setelah proses edit kedua berkas selesai, maka dilakukan proses instalasi dengan

perintah pada Gambar 2-9. Proses instalasi memakan waktu sekitar 10 – 15 menit.Langkah selanjutnya yang harus dilakukan adalah melakukan konfigurasi pada *environment path* melalui berkas ".bashrc". Pada berkas tersebut harus ditambahkan pengaturan mengenai *path* yang akan digunakan oleh NS2.

```
CC=         gcc-4.4
CFLAGS=     @CFLAGS@
RANLIB=     @RANLIB@
INSTALL=    @INSTALL@

#
# how to compile, link, and name shared libraries
#

SHLIB_LD=   @SHLIB_LD@
SHLIB_CFLAGS=   @SHLIB_CFLAGS@
SHLIB_SUFFIX=   @SHLIB_SUFFIX@
SHLD_FLAGS= @DL_LD_FLAGS@
DL_LIBS=    @DL_LIBS@

SHLIB_LD_LIBS  = @SHLIB_LD_LIBS@
```

Gambar 2-8 Tampilan dalam berkas Makefile.in

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

Gambar 2-9 Perintah install NS2

Perintah untuk pengaturan pada berkas ".bashrc" dapat dilihat pada Gambar 2-10 Konfigurasi *Path* pada berkas .bashrc Pada baris "home/irfan/workplace/" bisa diganti sesuai dengan letak direktori yang digunakan untuk meletakkan berkas dari NS2. Misal jika berkas diletakkan pada "home/abc/ns2/" maka konfigurasi harus diganti dengan "/home/abc/ns2/ns-allinone-2.35/otcl-1.14".

```
# LD_LIBRARY_PATH
OTCL_LIB=/home/irfan/workplace/ns-allinone-
2.35/otcl-1.14/
NS2_LIB=/home/irfan/workplace/ns-allinone-
2.35/lib/
USR_Local_LIB=/usr/local/lib/
export
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$OTCL_LIB:$NS2_
LIB:$USR_Local_LIB
# TCL_LIBRARY
TCL_LIB=/home/irfan/workplace/ns-allinone-
2.35/tcl8.5.10/library/
USR_LIB=/usr/lib/
export
TCL_LIBRARY=$TCL_LIBRARY:$TCL_LIB:$USR_LIB
# PATH
XGRAPH=/home/irfan/workplace/ns-allinone-
2.35/xgraph-12.2/:/home/irfan/workplace/ns-
allinone-2.35/bin/:/home/irfan/workplace/ns-
allinone-
2.35/tcl8.5.10/unix/:/home/irfan/workplace/ns-
allinone-2.35/tk8.5.10/unix/
NS=/home/irfan/workplace/ns-allinone-2.35/ns-
2.35/
NAM=/home/irfan/workplace/ns-allinone-2.35/nam-
1.15/
export PATH=$PATH:$XGRAPH:$NS:$NAM
```

Gambar 2-10 Konfigurasi *Path* pada berkas .bashrc

2.8.2 Instalasi NS2.35 Dengan Patch OLSR

Pada NS2.35 standar,tidak tersedia *protocol*OLSR didalamnya,maka untuk dapat melakukan implementasi OLSR pada NS2.35 harus dilakukan proses *patch*. Sebelum melakukan *patching* NS2.35 , terlebih dahulu dilakukan instalasi *dependensi* yang dibutuhkan seperti saat kita melakukan instalasi *dependensi* untuk NS2.35 normal sebelumnya

Setelah semua *dependensi* berhasil terinstal,yang perlu dilakukan adalah mengunduh file NS2.35 pada laman berikut.

```
https://sourceforge.net/projects/nsnam/files/latest/download
```

Gambar 2-11 laman unduh file NS2.35

Setelah file berhasil terunduh ,*ekstrak* file tersebut sehingga menjadi sebuah folder dengan nama "ns-allinone-2.35" dengan cara seperti berikut.

```
$tar xvf ns-allinone – 2.35_gcc482.tar.gz
```

Gambar 2-12 cara ekstraksi file NS2.35

Lalu masuk kedalam direktori folder NS2.35 dengan cara menuliskan perintah seperti berikut.

```
$cd ns-allinone-2.35/
```

Gambar 2-13 masuk ke folder NS2.35

Langkah selanjutnya adalah mengunduh file *patch* yang akan digunakan untuk memasukkan protokol OLSR kedalam NS2-35 , file *patch* tersebut dapat diunduh dalam lama berikut.

```
https://drive.google.com/file/d/0B7S255p3kFXNeVZhWFVVZlJnUEU/view
```

Gambar 2-14 laman unduh file patch OLSR

Setelah file berhasil terunduh ,masukkan file tersebut ke dalam folder “ns-allinone-2.35” dan letakkan proses *patching* seperti berikut.

```
$ patch -p0 < umOLSR-ns235_v1.0-2014.patch
```

Gambar 2-15 proses patching OLSR ke dalam NS2

Setelah proses *patching* selesai, lakukan proses instalasi dengan cara berikut

```
$ ./install
```

Gambar 2-16 proses instalasi

Tunggu beberapa saat sampai proses instalasi selesai , setelah proses instalasi selesai dan berhasil dilakukan, masuk ke folder”ns-2.35” dengan cara seperti berikut.

```
$cd ns-2.35
```

Gambar 2-17 masuk ke folder NS2

Proses selanjutnya adalah membuat file untuk eksekusi dalam menjalankan protokol kita sesuai dengan *patching* yang telah dilakukan. Lakukan cara seperti berikut.

```
$cp ns ns-OLSR
$sudo cp ns-OLSR /usr/local/bin/
```

Gambar 2-18 membuat file untuk eksekusi

Tujuan membuat file “ns-OLSR” adalah untuk digunakan dalam menjalankan protokol yang akan kita gunakann. Jika

pada NS2.35 normal menggunakan perinta “ns nama file” maka menjadi “ns-OLSR nama file”.

Untuk memastikan apakah proses *patching* kita berhasil kita bisa mencoba melakukan pengujian protokol OLSR dengan menulisakan perintah.

```
$ns-OLSR <namaFile.tcl>
```

Gambar 2-19 cara menjalankan NS

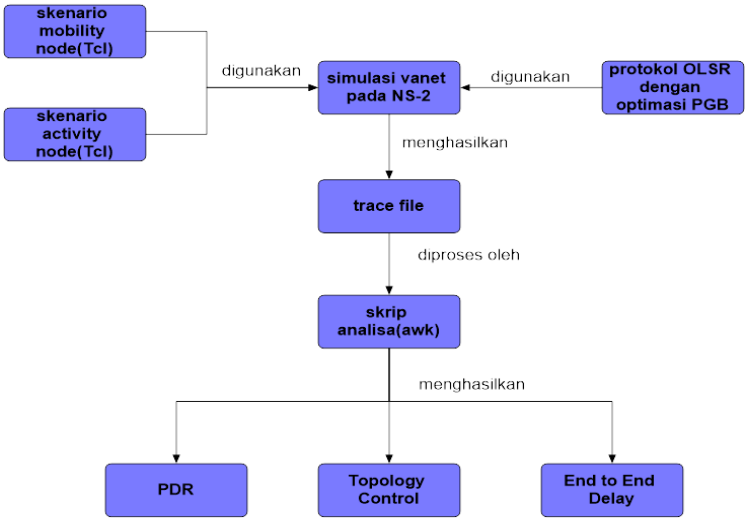
[Halaman ini sengaja dikosongkan]

BAB III
PERANCANGAN

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

3.1 Deskripsi Umum

Pada Tugas Akhir ini akan dilakukan analisis pengaruh peningkatan algoritma PGB pada protokol OLSR di jaringan VANET. Dimana metode routing OLSR akan menerapkan algoritma PGB pada proses meneruskan TC. Ilustrasi rancangan sistem bisa dilihat pada Gambar 3.1



Gambar3-1 Diagram rancangan simulasi

PGB pada protokol OLSR di jaringan VANET. Dimana metode *routing* OLSR akan menerapkan Proses pengujian

melibatkan dua jenis skenario yang berbeda yakni skenario dengan menggunakan peta *grid* dan skenario dengan menggunakan peta *real* Surabaya. Perancangan dengan peta berbentuk grid menggunakan alat bantu berupa SUMO sedangkan perancangan dengan menggunakan peta *real* Surabaya menggunakan alat bantu SUMO, peta digital *OpenStreetMap*, dan aplikasi JOSM.

Simulasi pada NS2 dilakukan dengan melakukan beberapa modifikasi pada modul OLSR, *Scheduler*, *Packet*, dan pada *Wireless-phy*. Setiap hasil dari simulasi akan dihitung PDR, *end to end delay* , dan *Topology Control*. Dari hasil tersebut dianalisis pengaruh dari algoritma PGB pada protocol OLSR.

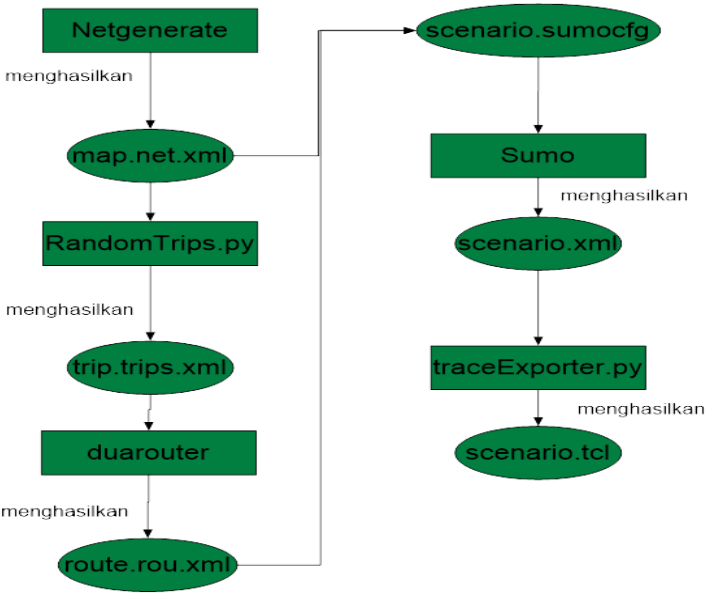
3.2Perancangan Skenario

Perencangan skenario simulasi pada VANET dibagi menjadi dua yakni skenario menggunakan peta berbentuk *grid* dan peta *real* Surabaya. Peta *grid* berupa peta yang hanya terdapat jalan dan saling berpotongan hingga membentuk persegi. Peta dengan bentuk *grid* dipilih sebagai tes awal simulasi karena bentuknya yang menggambarkan kondisi yang sebenarnya dan kondisi peta yang seimbang dan stabil. Sedangkan untuk peta riil menggunakan peta dari Surabaya. Peta riil didapat dengan cara mengimpor dari OpenStreetMap.

3.2.1Pembuatan Peta Grid

Peta *grid* untuk pengujian pada tugas ini berukuran 1000 m x 1000 m atau 1 km^2 dan jumlah garis *horizontal* dan *vertical* masing-masing sebanyak 6 (enam). Ilustrasi dapat dilihat pada Gambar 3.2. Pembuatan peta konfigurasi menggunakan *tools* netgenerate yang menghasilkan peta lengkap dengan jalan dan *traffic light*. Peta yang dibuat berjumlah tiga buah dengan variasi kecepatan 10 m/s, 15 m/s, dan 20 m/s. Hasil peta yang telah dibuat akan ditambah dengan *node* beserta *node* awal dan tujuan oleh modul randomTrips.py. Sedang untuk arah setiap *node* akan dibuat secara acak oleh modul duarouter. Kedua berkas yakni peta

grid dan pergerakan disimulasikan menjadi satu di dalam sebuah konfigurasi simulasi dan dengan menggunakan *tools* sumo dan modul traceExporter menghasilkan skenario yang dapat digunakan pada NS2.



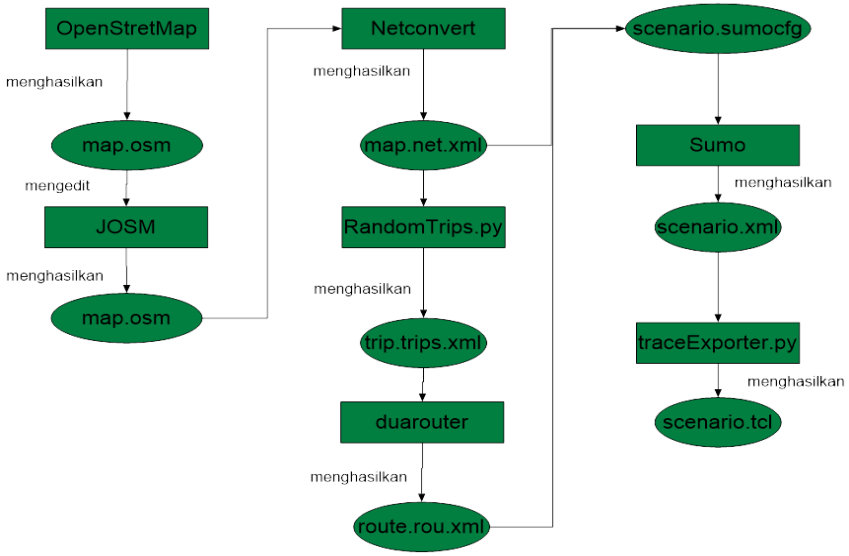
Gambar 3-2 Alur pembuatan peta grid

3.2.2Pembuatan Peta Riil Surabaya

Pembuatan peta riil Surabaya menggunakan peta yang diambil dari OpenStreetMap. Kemudian diimpor kedalam berkas berformat *.osm. Setelah itu akan dilakukan proses edit pada aplikasi JOSM. Pada aplikasi ini, akan dilakukan penghapusan objek yang diarasa tidak diperlukan dan penambahan *traffic light* pada peta. Setelah peta dirasa sudah cukup maka akan dilakukan proses pengkonversian dengan menggunakan *tools* netconvert hingga menghasilkan berkas dengan ekstensi *.net.xml. Alur proses secara lengkap bisa dilihat pada Gambar 3-3

3.3 Perancangan Sumulasi pada NS2

Simulasi VANET pada NS-2 dilakukan dengan menggunakan skenario mobilitas dan digabungkan dengan skrip TCL yang berisikan konfigurasi mengenai lingkungan simulasi.



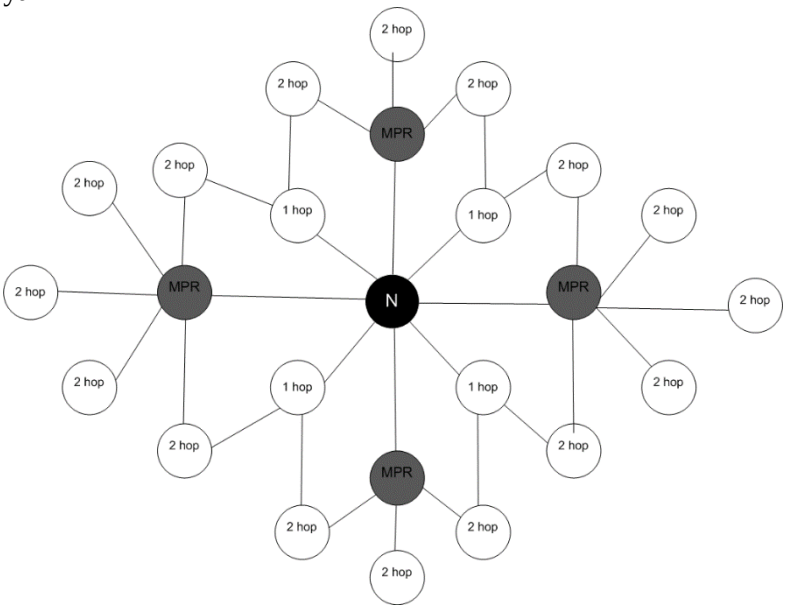
Gambar 3-3 Alur pembuatan peta riil Surabaya

3.4 Perancangan Protokol OLSR+PGB

Permasalahan yang sering muncul pada protokol OLSR yaitu kondisi ketika sebuah *node* menerima sebuah paket bertipe TC maka *node* tersebut akan melakukan *forwarding* paket TC kepada semua *node* tetangganya. Sehingga akan terjadi *broadcast storm* paket TC seperti pada Gambar 3-4 Proses *route discovery* pada

Setiap *node* dapat memperoleh lebih dari satu TC yang sama namun dari sumber yang berbeda. Hal ini mengakibatkan *traffic* menjadi tinggi dan kemungkinan *collision* semakin tinggi. Sehingga diperlukan algoritma baru untuk mengatasi masalah tersebut

Algoritma *Preferred Group Broadcasting* (PGB) mengubah prinsip pada proses *forwarding* TC pada protokol OLSR dimana hanya *node-node* tertentu yang akan melakukan proses *forwarding* TC bukan semua *node* dengan cara membatasi *node* yang berhak melakukan *rebroadcasting* hanya pada *node* yang berada pada area tertentu. PGB membagi area menjadi tiga bagian yakni



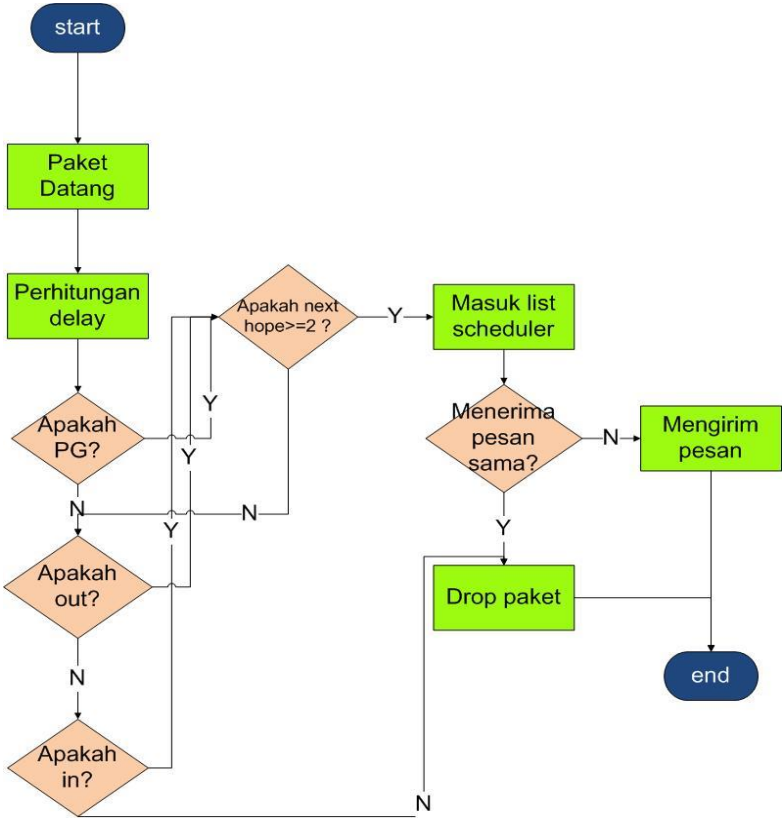
Gambar 3-4 Proses *route discovery* pada OLSR

- Area dalam (*Inner*/IN) = *node* dengan signal lebih kuat dari PG
- Area *Preferred group* (PG) = *node* yang akan digunakan untuk melakukan *rebroadcasting*
- Area luar (*Outer*/OUT) = *node* dengan signal lebih lemah dari PG

Proses pengelompokan tersebut berdasarkan *signal strength* yang diterima oleh masing-masing *node*. Pengelompokan *signal*

strength menggunakan *thresholds* yang yang telah ada[3]. Pengelompokan juga berguna menentukan nilai delay yang akan diberikan kepada masing-masing *node*. Delay akan digunakan sebagai pengatur jadwal pada *scheduler*. Setiap TC paket yang diterima akan diberikan delay sesuai dengan posisi *node* asal yang mengirimkan terhadap penerima. Delay paling sedikit berasal dari *node* pada posisi PG(*prefered Group)*, sedangkan delay terlama berasal dari *node* pada posisi IN.

Ketika ada paket datang pertama-tama dilakukan perhitungan delay apakah paket TC masuk area PG. Jika ya paket TC apakah memiliki 2 hop atau tidak. Jika ya maka akan masuk kedalam antrian *scheduler* jika tidak maka akan masuk area kedua yaitu OUT jika tidak masuk juga maka akan ke area IN jika tidak masuk area IN maka paket akan di *drop*. Setelah masuk area *scheduler node* akan di cek apakah pernah menerima paket yang sama jika ya maka paket akan di *drop* jika tidak maka pesan akan diteruskan. Proses penentuan node akan terus berlanjut seperti itu. Alur proses penentuan dapat dilihat pada Gambar 3-5.



Gambar 3-5.*Flow chart OLSR+PGB*

3.5 Perancangan Metrik Analisis

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

3.5.1 *Packet Delivery Ratio* (PDR)

Packet delivery ratio adalah persentase jumlah antara paket yang dikirim dan diterima. Penghitungan melibatkan perbandingan dari paket yang dikirim dan diterima. Rumus

untuk menghitung PDR dapat dilihat pada persamaan dibawah ini dimana $received$ adalah banyaknya paket data yang diterima dan *sent* adalah banyaknya paket data yang dikirimkan.

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

3.5.2 *End to end delay (E2E)*

End to end delay merupakan sebutan untuk waktu rata-rata yang dibutuhkan sebuah paket untuk sampai pada *node* tujuan. Hal tersebut juga dipengaruhi oleh keterlambatan yang disebabkan oleh proses penemuan rute/jalan untuk mengirimkan paket data dari *node* awal ke node tujuan. Lamanya waktu yang dibutuhkan sebuah paket untuk dapat terkirim dari *node* awal ke *node* tujuan.

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

3.5.3 *Topology Control* (TC)

Topology Control.merupakan salah satu jenis protokol pada protocol OLSR yang dapat dikirim ke seluruh jaringan secara *broadcasting*. Fungsi dari pesan TC ini adalah untuk menyebar informasi tentang *node* tetangga mana saja yang telah digunakan sebagai *node* MPR. *Node* TC hanya disebar satu periodik dan hanya *node* MPR yang dapat meneruskan pesan TC tersebut. Jumlah dari pesan TC dapat dihitung melalui *trace* file yang didapatkan setelah pengujian protokol dijalankan. Pesan TC yang dihitung hanya pesan TC yang bersasal dari *node* yang melakukan aktifitas *sending* paket dat

BAB IV
IMPLEMENTASI

Pada bab ini akan dibahas mengenai implementasi yang dilakukan berdasarkan rancangan yang telah dijabarkan pada bab sebelumnya. Sebelum penjelasan implementasi akan ditunjukkan terlebih dahulu lingkungan untuk melakukan implementasi.

4.1 Lingkungan Implementasi

Pada implementasi sistem, digunakan beberapa perangkat pendukung sebagai berikut:

4.1.1 Lingkungan Perangkat Lunak

Adapun perangkat lunak yang digunakan untuk pengembangan sistem sebagai berikut:

- Sistem operasi Ubuntu 14.04 64 bit
- Google Chrome versi 50.0.2661.102 m (64-bit) untukmengoperasikan web OpenSteetMap, dan
- JOSM versi 9979 sebagai editor peta hasil ekspor dariOpenStreetMap.

4.1.2Lingkungan Perangkat Keras

Spesifikasi perangkat keras yang digunakan sebagai lingkungan implementasi perangkat lunak Tugas Akhir adalah sebagai berikut:

- P*rocessor* Intel(R) Pentium(R) CPU B960@ 2.2GHz,
- RAM sebesar 4 GB DDR3, dan
- Media penyimpanan sebesar 50 GB.

4.2 Implementasi Skenario

Pada subbab ini akan dijelaskan implementasi yang telah dilakukan pada sistem. Implementasi skenario dibagi menjadi dua

bagian yakni skenario dengan peta grid dan skenario dengan peta riil Surabaya.

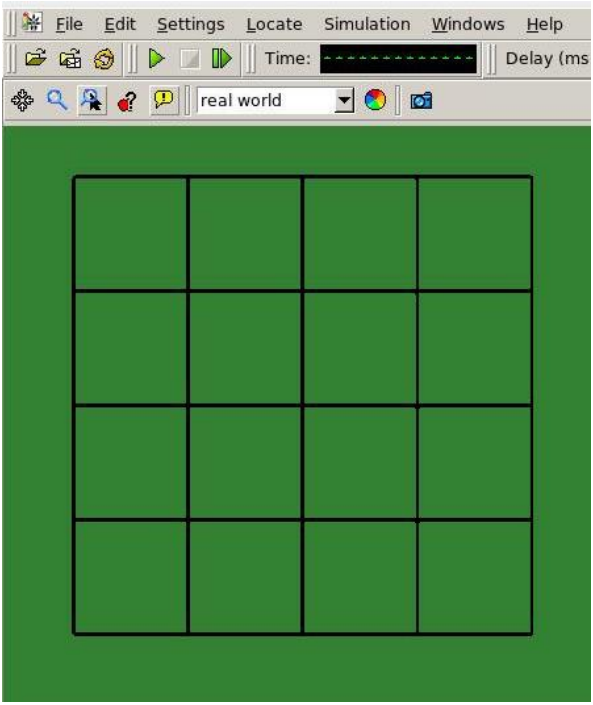
4.2.1 Skenario Grid

Dalam implementasi skenario grid, dibuat peta melalui *tools* netgenerate dari aplikasi SUMO dengan luas 1000 m x 1000 m dan jumlah garis vertikal dan horizontal sebanyak lima buah. Setiap petak berukuran 100 m x 100 m dan terdapat 5 titik x 5 titik serta terdapat 9 perempatan dengan kecepatan 10 m/s, 15 m/s, 20 m/s. Proses pembuatan peta grid pada terminal di Ubuntu dengan menggunakan netgenerate menggunakan perintah seperti pada Gamabar 4-1

```
netgenerate --grid --grid.number=5 --
grid.length=100 --default.speed=10 --
tls.guess=1 --output-file=map5-6.net.xml
```

Gambar 4-1 Perintah untuk membuat peta

Hasil peta jika dibuka dengan sumo-gui akan terlihat seperti pada Gambar 4-2.Setelah petak terbentuk, maka akan dilakukan pembuatan *node* beserta tujuan dan asal secara random dengan menggunakan modul randomTrips.py. Perintah untuk modul randomTrips.py bisa dilihat pada Gambar 4-3.Gambar 4-2 Hasil dari proses tersebut akan dilanjutkan oleh *tools* duarouter dimana setiap *node* akan diberikan arah gerak. Algoritma yang digunakan secara normal adalah algoritma Djikstra. Setelah proses tersebut selesai akan menghasilkan berkas berekstensi *.xml. Perintah untuk proses duarouter bisa dilihat pada Gambar 4-4 Perintah untuk membuat rute. Lalu hasil dari proses netgenerate dan hasil dari proses duarouter akan digabungkan menjadi satu pada berkas berekstensi sumocfg. *Script* untuk menggabungkan bisa dilihat pada Gambar 4-5.



Gambar 4-2 Peta hasil dari netgenerate

```
randomTrips.py -n map5-6.net.xml -e 100 -l --trip-
attributes="departLane=\"best\"
departSpeed=\"max\" departPos=\"random_free\"" -o
trip.trips5-6.xml
```

Gambar 4-3 Perintah untuk membuat *node*

```
duarouter --trip-files trips.trips.xml --net-file
kapal.xml --output-file result.rou.xml --ignore-
errors --repair
```

Gambar 4-4 Perintah untuk membuat rute

Script yang telah dibuat sebelumnya akan dijalankan oleh sumo untuk menggabungkan berkas peta dan berkas pergerakan nya. Hal yang perlu diperhatikan yakni berkas map.net.xml dan

route.rou.xml harus disimpan dalam direktori yang sama dengan berkas .sumocfg. Untuk melakukan pengecekan, berkas sumocfg bisa dibuka melalui tools sumo-gui.

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration
      xmlns:xsi="http://www.w3.org/2001/XMLSch
ema-instance"
      xsi:noNamespaceSchemaLocation="http://su
mo.dlr.de/xsd/sumoConfiguration.xsd">
      <input>
            <net-file value="map5-6.net.xml"/>
            <route-files
value="route.rou.xml"/>
      </input>
      <time>
            <begin value="0"/>
            <end value="20"/>
      </time>
```

Gambar 4-5 Contoh isi berkas sumocfg

Untuk dikonversi menjadi sebuah berkas xml yang berisi peta grid dan pergerakan kendaraan dijalankan perintah

```
sumo -c jajal.sumocfg --fcd-output scan.xml
```

Gambar 4-6 Perintah untuk mengkonversi peta menjadi xml

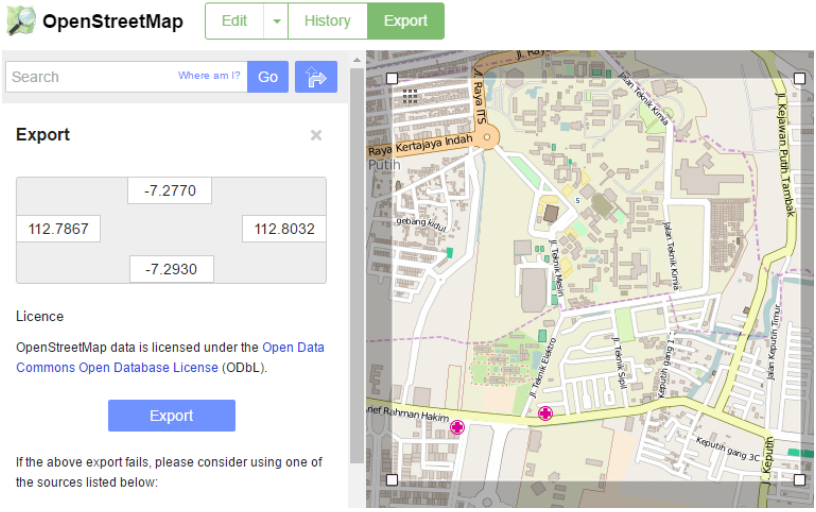
Berkas scenario.xml kemudian dikonversi menjadi berkas pergerakan dalam format mobilitas NS-2 dengan bantuan modul traceExporter.py dengan perintah seperti pada

```
traceExporter.py    --fcd-input=scan.xml    --
ns2mobility-output=ns25 6.tcl
```

Gambar 4-7 Perintah untuk mengkonversi xml ke format tcl

4.2.2 Skenario Riil

Pada dasarnya, pembuatan skenario dengan peta riil hampir sama dengan pembuatan skenario pada peta grid. Yang menjadi pembeda hanyalah pada bagian awalnya saja. Jika pada peta grid peta langsung dibuat secara otomatis oleh netgenerate, pada peta riil, berkas peta terlebih dahulu diambil dari OpenStreetMap dengan cara menandai wilayah kemudian diekspor dan secara otomatis akan terunduh berkas berekstensi .osm melalui browser. Contoh proses *capture* bisa dilihat pada Gambar 4-8.



Gambar 4-8 Proses capture peta Surabaya dengan OpenStreetMap

Hasil dari proses tersebut kemudian diedit menggunakan JOSM. JOSM berfungsi untuk mengisikan lampu merah pada tiap perempatan. Jalan yang tidak digunakan dihapus serta bentuk dan ukuran peta dibuat supaya hampir sama dengan peta grid. Beberapa jalan baru juga ditambahkan agar kepadatan jalan tetap stabil sehingga tidak akan ada daerah pada peta yang jarang

dikunjungi kendaraan. Jalan baru tersebut juga berfungsi agar bentuk peta mendekati persegi agar mirip dengan skenario grid.

Setelah dirasa cukup, lalu berkas dikonversi agar berbentuk .net.xml. Konversi dilakukan dengan bantuan netconvert dari SUMO. Perintah yang dijalankan untuk mengkonversi bisa dilihat pada Gambar 4-9.

```
netconvert --osm-files map-1.osm --output-file
map.net.xml
```

Gambar 4-9 Perintah konversi berkas osm menjadi format SUMO

Langkah selanjutnya sama persis dengan langkah pembuatan berkas skenario peta grid sampai berhasil dikonnversi kedalam format pergerakan NS-2.

4.3 Implementasi Algoritma PGB Pada OLSR

Protokol OLSR + PGB adalah turunan dari protocol OLSR. Untuk melakukan modifikasi pada *routing protocol* OLSR agar dapat menerapkan algoritma PGB harus ada perubahan pada beberapa bagian antara lain:

- Header dari paket
- Penentuan *signal strength* untuk memperhitungkan delay
- Penanganan paket TC
- Penanganan ketika terjadi *hop splitting* dan *hop marging*

Maka pertama-tama harus dilakukan proses pengeditan dulu pada beberapa berkas di NS-2. Terdapat 7 berkas yang diedit yakni `packet.h`, `OLSR.cc`, `OLSR.h`, `scheduler.cc`, `scheduler.h`, `wireless-phy.cc`,dan `wireless-phy.h`.

4.3.1 Modifikasi Packet.h

Packet.h merupakan *class* yang digunakan sebagai acuan untuk isi dari semua paket di semua kelas. Baik pada protokol OLSR maupun protokol yang lain. Dalam Tugas Akhir ini, pada packet.h akan ditambahkan variabel untuk menampung nilai *delay*

pada paket tersebut. Penambahan dilakukan pada `struct hdr_cmn` dikarenakan *struct* tersebut merupakan *header* bagi paket yang akan dikirim. Penambahan seperti pada Gambar 4-10

```
struct hdr_cmn {
       enum dir_t { DOWN= -1, NONE= 0, UP= 1 };
       packet_t ptype_;    // packet type (see above)
       int    size_;       // simulated packet size
       int    uid_;        // unique id
       int    error_;             // error flag
       int     errbitcnt_;     // # of corrupted bits
jahn
       int     fecsize_;
       double ts_;          // timestamp: for q-delay
measurement
       int    iface_;             //         receiving
interface (label)
       dir_t  direction_;   //    direction:    0=none,
1=up, -1=down
       double dly;          //variable delay
```

Gambar 4-10 Modifikasi paket.h

4.2.2 Modifikasi Wireless-phy.cc

Dalam tugas akhir ini,Pada file *wireless-phy* akan ditambahkan class baru untuk melakukan kalkulasi *delay* yang akan diberikan dalam suatu *header* paket, *class* tersebut diberi nama *call()*

Inti dari class *call()* adalah menentukan kekuatan sinyal sebuah node, selain itu juga dibuat batas atas dan batas bawah dari area luar “OT” dan area dalam “IT” yang nantinya akan digunakan proses mengeset nilai *delay* toleran yaitu nilai random yang ditambahkan dengan waktu sebuah *node* dalam menerima sebuah paket, yang digunakan untuk menentukan bahwa sebuah *node* berada dalam area PGB. Jika sebuah *node* mempunyai nilai *delay* toleran yang telah ditentukan, Maka node tersebut berada pada arae PGB. Dari pembuatan calss *call()* ini, akan didapatkan posisi dari sebuah node. Apakah berada pada area IN, OUT atau PGB

Selain itu pada fungsi *SendUp* juga akan dilakukan peneditan untuk melakukan pengecekan kekuatan sinyal.Pada NS2 disimpan pada variabel *Pr* ,pengecekan disini berhubungan dengan *class call()*,karena saat melakukan pengecekan dibutuhkan pemanggilan nilai *delay.*

```
double
WirelessPhy::cal(double  pr,double rxTh,double  txP){
if (pr>OT && pr<IT){
              double fot=OT;
              double delP1 =rxTh-pr;

              if (delP1<0) delP1=delP1*-1;
              double delP0 = delP1-fot;
              if (delP0<0) delP0=delP0*-1;
              double delT = mxT1-mnT1;
              double delFot=IT-OT;
              random=Random::uniform(0.0001, 0.0009);
              double T1=(delP0*delT/delFot)+mnT1;
              double waktu=T1+random;
              return waktu;
       }

              return waktu;
       }

else
       {
              return 0;
       }
}
```

Gambar 4-11Modifikasi wireless-phy.cc

```
p->txinfo_.RxPr = Pr;
      p->txinfo_.CPThresh = CPThresh_;
      struct hdr_cmn *ch = HDR_CMN(p);
      ch->dly=cal(Pr,RXThresh_,Pt_);
```

Gambar 4-12Modifikasi fungsi sendUp

Pada fungsi `WirelessPhy::cal()` dilakukan penghitungan delay berdasarkan kekuatan signal yang ada. Selain itu juga diberikan nilai random untuk mencegah terjadinya *broadcast* paket secara bersamaan dari dua atau lebih *node* yang memiliki nilai kekuatan sinyal yang sama.

4.3.2 Modifikasi Wireless-phy.h

Karena melakukan penambahan fungsi baru yakni `WirelessPhy::cal()`pada wireless-phy.cc maka harus dilakukan perubahan pada wireless-phy.h untuk melakukan pendefinisian dari fungsi `WirelessPhy::cal()` karena struktur dari NS-2 menggunakan *class*. Perubahan kode bisa dilihat pada Gambar 4-13

```
void node_on();
void node_off();
double   cal(double     pr,double   rxTh,double
```

Gambar 4-13Modifikasi wireless-phy.h

4.3.3 Modifikasi Scheduler.cc

Scheduler.cc merupakan sebuah file yang berfungsi sebagai tempat dimana semua kegiatan atau *event* sebuah pengiriman disimpan. Pada modifikasi kali ini akan dilakukan sebuah perubahan pada class *scheduler::schedulex()* yang berfungsi untuk mengirim paket data hanya ke *node* MPR yang berada pada area PGB, jika tidak ditemukan *node* MPR pada area PGB maka pencarian dilakukan pada area IN jiika pada area ini tidak ditemukan juga maka dilanjutkan ke area OUT. Selain itu

juga dibuat class dengan nama *scheduler::scheduleDel()* yang berfungsi menangani broadcast data yang sama dari node lain

```
bool
Scheduler::scheduleDel(scheduler_uid_t uid){
	Event* p = lookup(uid);
	if (p != 0) {
		/*XXX make sure it really is an
atevent*/
		cancel(p);
		//AtEvent* ae = (AtEvent*)p;
		//delete ae;
		return true; } return false;}
```

Gambar 4-14Modifikasi pada fungsi schedulerDel

```
int
Scheduler::schedulex(Handler* h, Event* e, double
delay)
{
	int uuuu=0;
		if (!h) {
		fprintf(stderr,
			"Scheduler: attempt to schedule
an event with a NULL handler."
			"  Don't DO that at time %f\n",
clock_);
		abort();
	};
	if (e->uid_ > 0) {
		printf("Scheduler: Event UID not
valid!\n\n");
		abort();
	}
	if (delay < 0) {
		fprintf(stderr,
			"warning: ns
Scheduler::schedule: scheduling event\n\t"
```

Gambar 4-15Modifikasi pada fungsi schedulex

4.3.4 Modifikasi Scheduler.h

Karena melakukan penambahan fungsi baru yakni `Scheduler::schedulex()` dan `Scheduler::scheduleDel` pada schedule.cc maka harus dilakukan perubahan pada schedule.h untuk melakukan pendefinisian dari fungsi `Scheduler::schedulex()` dan `Scheduler::scheduleDel` karena struktur dari NS-2 menggunakan *class*. Perubahan kode bisa dilihat pada Gambar 4-16.

```
class Scheduler : public TclObject {
public:
       static Scheduler& instance() {
              return (*instance_);              //
general access to scheduler
       }
       void   schedule(Handler*,   Event*,   double
delay);       // sched later event
       int   schedulex(Handler*,   Event*,   double
delay);       // tambah
       bool scheduleDel(scheduler uid t uid);
```

Gambar 4-16Modifikasi scheduler.h

4.3.4 Modifikasi OLSR.cc

OLSR.cc merupakan file utama pada protokol OLSR yang ada di NS, file ini yanga kan mmenaggil semua file yang berhubungan dengan protokol *OLSR.* Pada modifikasi kali ini dilakukan perubahan pada *OLSR::recv_OLSR* tujuan perubahan tersebut adalah untuk mekanisme penerusan paket data. Paket data hanya akan dikirimkan pada node yang berada di area PGB selain itu juga didalam PGB itu dilakukan selsksi hanya untuk menentukan node yang akan dikrim dengan teknik MPR.Gambar 4-17

```
struct hdr_ip* ih   = HDR_IP(p);
       OLSR_pkt* op        = PKT_OLSR(p);
       struct hdr_cmn *ch = HDR_CMN(p);
       assert(ih->sport() == RT_PORT);
       assert(ih->dport() == RT_PORT);


       if   (op->pkt_len()   <OLSR_PKT_HDR_SIZE   +
OLSR_MSG_HDR_SIZE) {
              Packet::free(p);
              return;
       }

       assert(op->count   >=   0   &&   op->count   <=
OLSR_MAX_MSGS);
       for (int i = 0; i < op->count; i++) {
              OLSR_msg& msg = op->msg(i);


              if (msg.ttl() <= 0 || msg.orig_addr()
== ra_addr())
                     continue;


              bool do_forwarding = true;
if (*it == ra_addr() && ch->dly < 0.01f)  {
       if (ch->dly >0.0001f){
       do_forwarding = true;}
              do_forwarding = false;
                     break;}}}
       rtable_computation();
       Packet::free(p);
}

              OLSR_dup_tuple*      duplicated      =
state_.find_dup_tuple(msg.orig_addr(),
msg.msg_seq_num());
```

Gambar 4-17Modifikasi OLSR.cc

Sedang pada fungsi `OLSR::recvRequest` bisa dilihat pada Gambar4-18

```
 if (id_lookup(rq->rq_src, rq->rq_bcast_id)) {
       //tambah
       printf(" -- Drop");
       int       yy=id_lookupx(rq->rq_src,       rq-
>rq_bcast_id,rq->rq_dst,rq->rq_hop_count);
       if(yy>-1){
              printf(" with uid= %d ",yy);
       if(Scheduler::instance().scheduleDel(yy)){
                     printf(" Droped ");
              }else{
                     printf(" Fail ");}  }
    printf("\n");
uidx=forwardx((aodv_rt_entry*) 0, p, DELAY);
   id_update(rq->rq_src,     rq->rq_bcast_id,     rq-
>rq_dst, rq->rq_hop_count, uidx);
```

Gambar4-18 Modifikasi fungsi recv_OLSR

4.4 Implementasi Metrik Analisis

Hasil dari simulasi skenario dalam NS-2 adalah sebuah trace berkas. Trace berkas digunakan sebagai bahan analisis untuk pengukuran performa dari protokol yang disimulasikan. Dalam Tugas Akhir ini, terdapat empat metrik yang akan menjadi parameter analisis, yaitu *packet delivery ratio*, *average end-to-end delay*, *dan topology control*

4.4.1Implementasi *Packet Delivery Ratio*

Packet delivery ratio didapatkan dengan cara menghitung setiap baris terjadinya *event* pengiriman dan penerimaan paket data yang dikirim melalui agen pada trace berkas. Pada lampiran A 3 ditunjukkan cara menyaring data yang dibutuhkan untuk perhitungan PDR. Pada skrip tersebut dilakukan penyaringan pada setiap baris yang mengandung *string* AGT karena *event* tersebut berhubungan dengan paket data. Paket yang dikirim dan diterima dibedakan dari kolom pertama pada baris yang telah

disaring. Setelah pembacaan setiap baris selesai dilakukan, selanjutnya dilakukan perhitungan PDR dengan persamaan 3.1.

Contoh perintah untuk memanggil skrip AWK untuk menganalisis trace berkas dan contoh keluarannya dapat dilihat pada Gambar 4-19 dan Gambar 4-20.

```
awk -f pdr.awk s-OLSRAlun.tr
```

Gambar 4-19 Perintah untuk menjalankan skrip pdr.awk

```
Sent: 150 Recv: 109 Ratio: 0.7267
```

Gambar 4-20 Keluaran dari Skrip pdr.awk

4.4.2Implementasi *Average End-to-End Delay*

Dalam pembacaan baris trace berkas untuk perhitungan *average end-to-end delay* terdapat lima kolom yang harus diperhatikan, yaitu kolom pertama yang berisi penanda event pengiriman atau penerimaan, kolom kedua yang berisi waktu terjadinya event, kolom keempat yang berisi informasi layer komunikasi paket, kolom keenam yang berisi ID paket, dan tipe paket pada kolom ketujuh.

Delay dari setiap paket dihitung dengan cara mengurangi waktu penerimaan dengan waktu pengiriman berdasarkan ID paket. Hasil pengurangan waktu dari masing-masing paket dijumlahkan dan dibagi dengan jumlah paket CBR yang ID-nya terlibat dalam perhitungan pengurangan waktu.Kode implementasi dari perhitungan *average end-to-end delay* dapat dilihat pada lampiran A 4.

Contoh perintah untuk memanggil skrip AWK untuk menganalisis trace berkas dan contoh keluarannya dapat dilihat pada Gambar 4-22 danGambar 4-23.

```
awk -f endtoend.awk s-OLSRAlun.tr
```

Gambar 4-21 Perintah untuk menjalankan skrip end to end.awk

```
Average End-to-End Delay     = 224.818 ms
```

Gambar 4-22Keluaran dari Skrip endtoend.awk

4.5 Implementasi Skenario Simulasi pada NS2

Untuk melakukan simulasi VANET pada NS-2, dibutuhkan sebuah berkas OTcl yang berisi deskripsi dari lingkungan simulasi. Berkas tersebut berisikan pengaturan untuk setiap *node* dan beberapa even yang perlu diatur agar berjalan pada waktu tertentu. Pengaturan lainnya yang dilakukan pada berkas tersebut antara lain lokasi penyimpanan trace berkas, lokasi berkas mobilitas *node*, konfigurasi *node* sumber dan *node* tujuan, dan konfigurasi event pengiriman paket data. Kode implementasi skenario yang digunakan pada Tugas Akhir ini dapat dilihat pada lampiran A 1 dan lampiran untuk *pattern* skenario. Skenario simulasi dijalankan dengan perintah pada Gambar 4-23. Setelah simulasi selesai akan ada keluaran berupa *trace* berkas hasil simulasi yang akan digunakan untuk analisis. Isi dari trace berkas adalah catatan seluruh *event* yang dari setiap paket yang tersebar di dalam lingkungan simulasi.

```
Ns-OLSR[nama_file].tcl
```

Gambar 4-23 Perintah Untuk Menjalankan Skenario NS-2

[Halaman ini sengaja dikosongkan]

BAB V
UJI COBA DAN EVALUASI

Pada bab ini, dibahas uji coba dan evaluasi dari skenario NS2 yang telah dibuat.

5.1 Lingkungan Uji Coba

Uji coba dilakukan pada laptop dengan sistem operasi Ubuntu Linux dengan NS2 telah terpasang didalamnya. Untuk spesifikasi laptop dapat dilihat pada

Tabel 5.1 Spesifikasi laptop yang digunakan

Komponen	Spesifikasi
CPU	P*rocessor* Intel(R) Pentium(R) CPU @ 2.2GHz,
Sistem Operasi	Ubuntu 14.04 64 bit
Memori	4 GB DDR3
Grafis	Intel HD 4000
Penyimpanan	500 GB HDD

Pengujian dilakukan dengan menjalankan program yang sudah dibuat pada NS2 tadi. Parameter pengujian bisa dilihat pada Tabel 5.2.

Tabel 5.2 Parameter Pengujian

No	Parameters	Spesifikasi
1	Netwok Simulator	NS-2, 2.35
2	Routing Protocol	OLSR + PGB dan OLSR(modified)
3	Waktu simulasi	200 ms
4	Area simulasi	1000 m x 1000 m untuk grid, 1500 x 1500 untuk riil
5	Banyak kendaraan	30,50,70,90
6	Radius Tranmisi	250 m
7	Kecepatan maksimal	10 m/s, 15 m/s, 20 m/s
8	Tipe data	Constant Bit Rate (CBR)

No	Parameters	Spesifikasi
9	Ukuran paket data	512 kb
10	Protokol	MAC dan IEEE 802.11p
11	Mode propagasi	Two way ground
12	Mobility model	Peta grid dan peta real Surabaya
13	Tipe kanal	Wireless channel

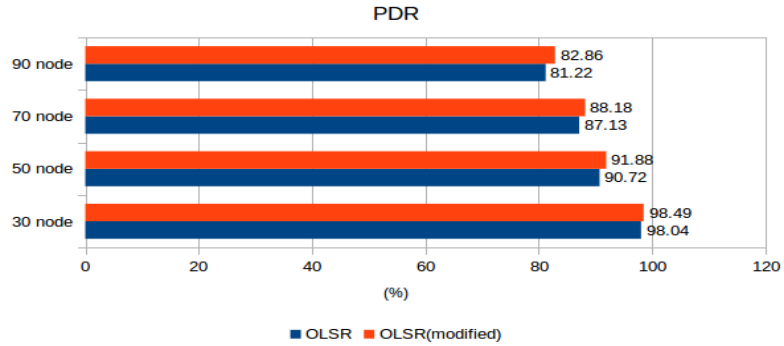
5.2 Hasil Uji Coba

Hasil uji coba dari skenario grid dan skenario Surabaya nantinya akan terdiri dari 3 indikator yaitu PDR,*end to end delay* dan *Topology Control*(TC). Pada PDR jika hasil semakin tinggi nilainya maka akan semakin bagus .Tetapi pada *end to end delay* dan *topology control* sebaliknya semakin kecil nilai yang dihasilkan maka akan semakin bagus. Hasil uji coba dapat dilihat sebagai berikut :

5.2.1 Hasil Uji Coba Grid

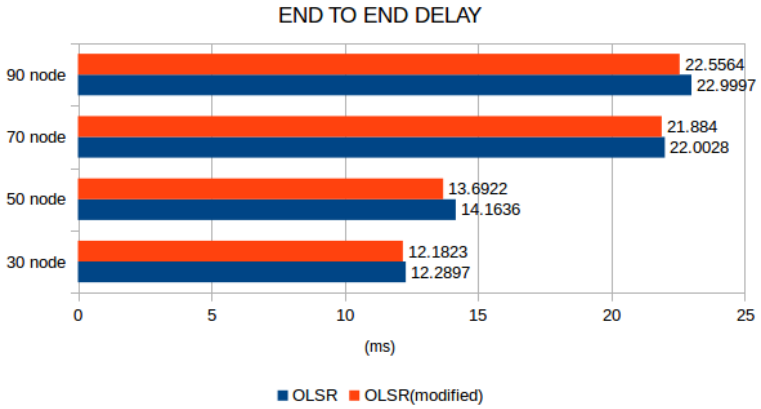
Hasil pengujian data PDR,*end to end delay* dan *topology control* pada skenario *grid* dilakukan pad area dengan luas 1000x1000 meter dan *node* sebanyak 20,50,70 dan 90 pada 3 kecepatan yang berbeda yaitu pada kecepatan maksimal 10 m/s ,15 m/s dan 20 m/s.

5.2.1.1 Skenario Grid Kecepatan 10 m/s



Gambar 5-1 Grafik PDR dengan kecepatan 10 m/s

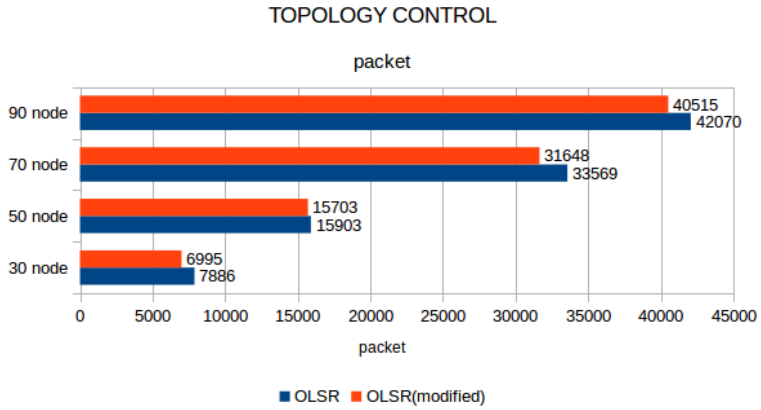
Dari hasil grafik diatas kita bisa lihat bahwa pada saat skenario menggunakan node sebanyak 30 node dengan protokol OLSR yang dimodifikasi menunjukkan nilai PDR sebesar 98,49 % sedangkan protokol OLSR yang biasa menunjukkan nilai PDR 98,04 % dan pada skenario 50 node protokol OLSR yang dimodifikasi menunjukkan nilai PDR 91,88 % sedangkan yang menggunakan protokol OLSR menunjukkan nilai PDR 90,72% begitu juga dengan skenario 70 dan 90 node protokol OLSR yang telah dimodifikasi menunjukkan nilai PDR yang lebih bagus dibandingkan dengan protokol OLSR biasa



Gambar 5-2Grafik end to end kecepatan 10 m/s

Dari hasil grafik diatas kita bisa lihat bahwa pada saat skenario menggunakan node sebanyak 30 node dengan protokol OLSR yang dimodifikasi menunjukkan waktu *end to end delay* sebesar 12.1823 ms sedangkan protokol OLSR yang biasa menunjukkan waktu *end to end delay* 12,2897 ms dan pada skenario 50 node protokol OLSR yang dimodifikasi menunjukkan waktu *end to end delay* 13,6922 ms sedangkan yang menggunakan protokol OLSR menunjukkan waktu *end to end delay* 14.1636 ms begitu juga dengan skenario 70 dan 90 node

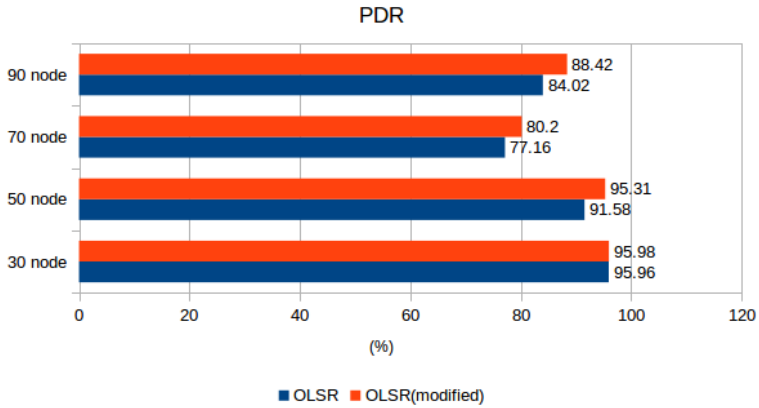
protokol OLSR yang telah dimodifikasi menunjukkan waktu *end to end delay* yang lebih kecil dibandingkan dengan protokol OLSR biasa



Gambar 5-3 Grafik topology control kecepatan 10 m/s

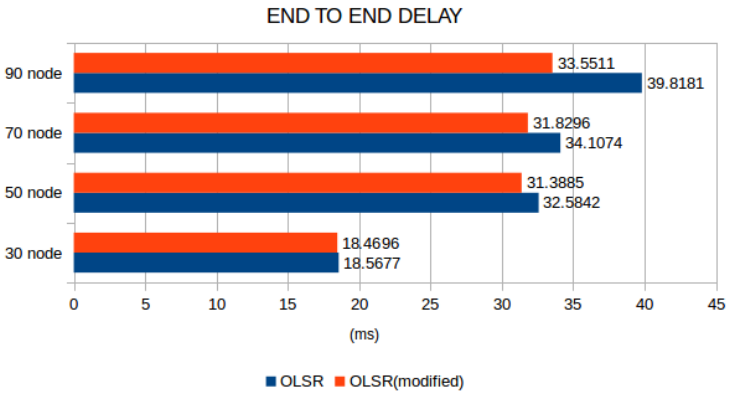
Dari hasil grafik diatas kita bisa lihat bahwa pada saat skenario menggunakan node sebanyak 30 node dengan protokol OLSR yang dimodifikasi menunjukkan nilai TC sebesar 6995 paket sedangkan protokol OLSR yang biasa menunjukkan nilai TC 7886 paket dan pada skenario 50 node protokol OLSR yang dioptimasi menunjukkan nilai TC 15703 paket sedangkan yang menggunakan protokol OLSR menunjukkan TC 1590 paket begitu juga dengan skenario 70 dan 90 node protokol OLSR yang telah dioptimasi menunjukkan nilai TC yang lebih bagus dibandingkan dengan protokol OLSR biasa. Hal ini menunjukkan bahwa protokol OLSR yang dimodifikasi dapat mengurangi terjadinya *strorm broadcasting*

5.2.1.2 Skenario Grid Kecepatan 15 m/s



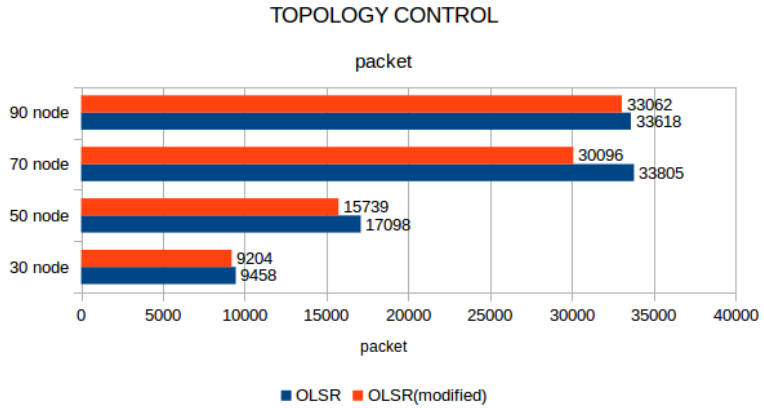
Gambar 5-4 Grafik PDR dengan kecepatan 15 m/s

Dari hasil grafik diatas kita bisa lihat bahwa pada saat skenario menggunakan node sebanyak 30 node dengan protokol OLSR yang dimodifikasi menunjukkan nilai PDR sebesar 95,98 % sedangkan protokol OLSR yang biasa menunjukkan nilai PDR 95,96 % dan pada skenario 50 node protokol OLSR yang dimodifikasi menunjukkan nilai PDR 95,31 % sedangkan yang menggunakan protokol OLSR menunjukkan nilai PDR 91,58% begitu juga dengan skenario 70 dan 90 node protokol OLSR yang telah dimodifikasi menunjukkan nilai PDR yang lebih bagus dibandingkan dengan protokol OLSR biasa



Gambar 5-5 Grafik end to end delay dengan kecepatan 15 m/s

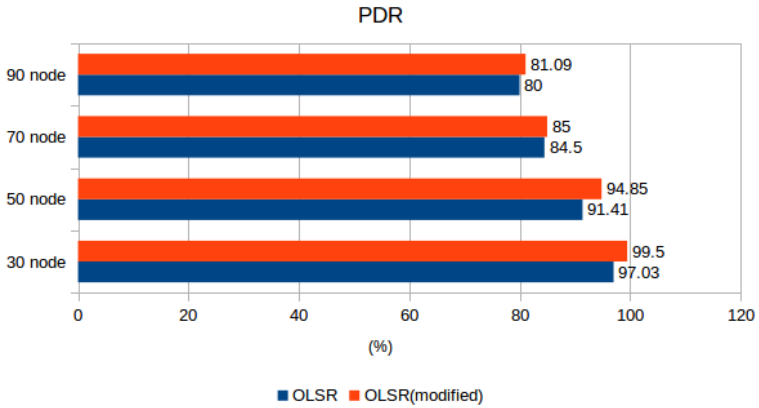
Dari hasil grafik diatas kita bisa lihat bahwa pada saat skenario menggunakan node sebanyak 30 node dengan protokol OLSR yang dimodifikasi menunjukkan waktu *end to end delay* sebesar 18,4696 ms sedangkan protokol OLSR yang biasa menunjukkan waktu *end to end delay* 18,5677 ms dan pada skenario 50 node protokol OLSR yang dimodifikasi menunjukkan waktu *end to end delay* 31,3885 ms sedangkan yang menggunakan protokol OLSR menunjukkan waktu *end to end delay* 32,5842 ms begitu juga dengan skenario 70 dan 90 node protokol OLSR yang telah dimodifikasi menunjukkan waktu *end to end delay* yang lebih kecil dibandingkan dengan protokol OLSR biasa



Gambar 5-6 Grafik topology control dengan kecepatan 15 m/s

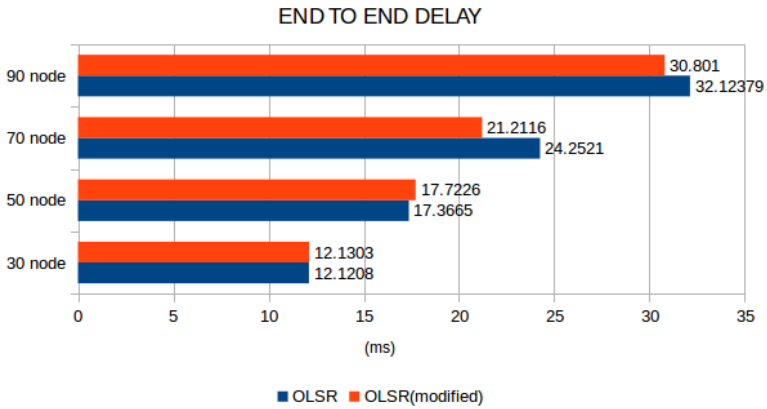
Dari hasil grafik diatas kita bisa lihat bahwa pada saat skenario menggunakan node sebanyak 30 node dengan protokol OLSR yang dimodifikasi menunjukkan nilai TC sebesar 9204 paket sedangkan protokol OLSR yang biasa menunjukkan nilai TC 9458 paket dan pada skenario 50 node protokol OLSR yang dimodifikasi menunjukkan nilai TC 15739 paket sedangkan yang menggunakan protokol OLSR menunjukkan TC 17098 paket begitu juga dengan skenario 70 dan 90 node protokol OLSR yang telah dimodifikasi menunjukkan nilai TC yang lebih bagus dibandingkan dengan protokol OLSR biasa. Hal ini menunjukkan bahwa protokol OLSR yang dimodifikasi dapat mengurangi terjadinya *strorm broadcasting*

5.2.1.3 Skenario Grid Kecepatan 20 m/s



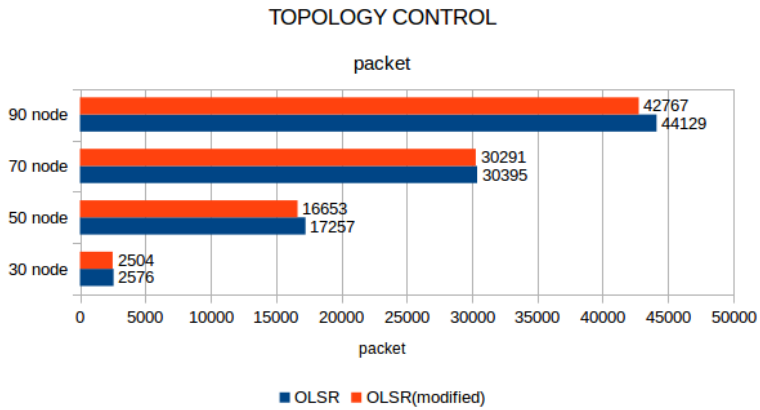
Gambar 5-7 Grafik PDR dengan kecepatan 20 m/s

Dari hasil grafik diatas kita bisa lihat bahwa pada saat skenario menggunakan node sebanyak 30 node dengan protokol OLSR yang dimodifikasi menunjukkan nilai PDR sebesar 99,5% sedangkan protokol OLSR yang biasa menunjukkan nilai PDR sebesar 97,03% dan pada skenario 50 node protokol OLSR yang dimodifikasi menunjukkan nilai PDR sebesar 94,85% sedangkan yang menggunakan protokol OLSR menunjukkan PDR sebesar 91,41% begitu juga dengan skenario 70 dan 90 node protokol OLSR yang telah dimodifikasi menunjukkan nilai PDR yang lebih bagus dibandingkan dengan protokol OLSR biasa.



Gambar 5-8 Grafik end to end delay dengan kecepatan 20 m/s

Dari hasil grafik diatas kita bisa lihat bahwa pada saat skenario menggunakan node sebanyak 30 node dengan protokol OLSR yang dimodifikasi menunjukkan waktu *end to end delay* sebesar 12.1303 ms sedangkan protokol OLSR yang biasa menunjukkan waktu *end to end delay* 12.1208 ms dan pada skenario 50 node protokol OLSR yang dimodifikasi menunjukkan waktu *end to end delay* 17.7226 ms sedangkan yang menggunakan protokol OLSR menunjukkan waktu end to end delay 17.3665 ms tetapi dengan skenario 70 dan 90 node protokol OLSR yang telah dimodifikasi menunjukkan waktu *end to end delay* yang lebih kecil dibandingkan dengan protokol OLSR biasa.pada node 30 dan 50 protokol OLSR yang dioptimasi menunjukkan hasil yang lebih besar hal itu karena pada skenario node yang lebih kecil pada kecapatan maksimal 20 m/s protokol yang dimodifikasi membutuhkan waktu lebih lama untuk memeriksa node manakah saja yang meneruskan ke node tetangga yang kuat sedangkan node protokol yang biasa tidak melakukan itu



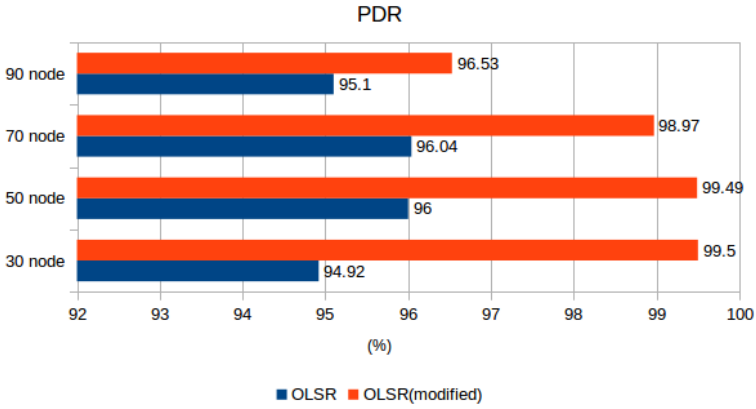
Gambar 5-9Grafik topology control dengan kecepatan 20 m/s

Dari hasil grafik diatas kita bisa lihat bahwa pada saat skenario menggunakan node sebanyak 30 node dengan protokol OLSR yang dimodifikasi menunjukkan nilai TC sebesar 2504 paket sedangkan protokol OLSR yang biasa menunjukkan nilai TC 2576 paket dan pada skenario 50 node protokol OLSR yang dimodifikasi menunjukkan nilai TC 16653 paket sedangkan yang menggunakan protokol OLSR menunjukkan TC 17257 paket begitu juga dengan skenario 70 dan 90 node protokol OLSR yang telah dimodifikasi menunjukkan nilai TC yang lebih bagus dibandingkan dengan protokol OLSR biasa. Hal ini menunjukkan bahwa protokol OLSR yang dimodifikasi dapat mengurangi terjadinya *strorm broadcasting*

5.2.2 Hasil Uji Coba Peta Riil Surabaya

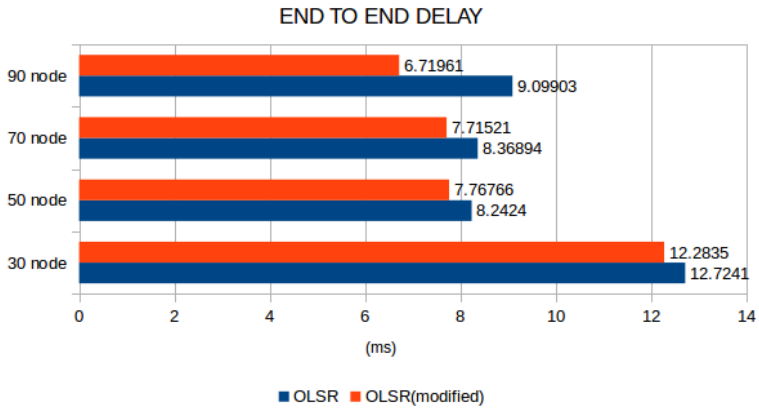
Hasil pengujian dat PDR,*end to end delay* dan *topology control* pada skenario *real* dilakukan pada area dengan luas 1500 X 1500 meter dan node sebanyak 30,50,70,90 pada 3 kecepatan berbeda,yaitu pada kecepatan 10 m/s,15m/s dan 20 m/s

5.2.2.1 Skenario *Real* Pada Kecepatan 10 m/s



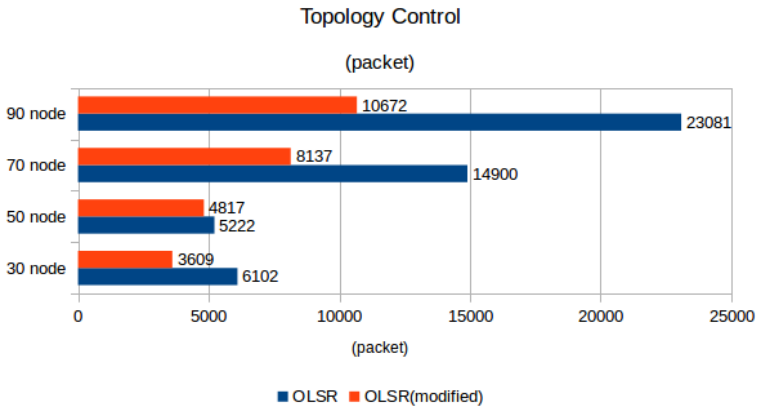
Gambar 5-10 Grafik PDR dengan kecepatan 10m/s

Dari hasil grafik diatas kita bisa lihat bahwa pada saat skenario menggunakan node sebanyak 30 node dengan protokol OLSR yang dimodifikasi menunjukkan nilai PDR sebesar 99,5% sedangkan protokol OLSR yang biasa menunjukkan nilai PDR sebesar 94,92% dan pada skenario 50 node protokol OLSR yang dimodifikasi menunjukkan nilai PDR sebesar 99,49% sedangkan yang menggunakan protokol OLSR menunjukkan PDR sebesar 96 % begitu juga dengan skenario 70 dan 90 node protokol OLSR yang telah dimodifikasi menunjukkan nilai PDR yang lebih bagus dibandingkan dengan protokol OLSR biasa



Gambar 5-11 grafik end to end delay dengan kecepatan 10 m/s

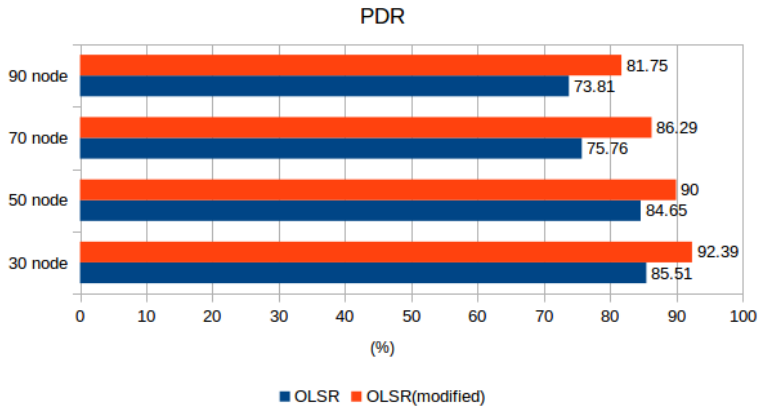
Dari hasil grafik diatas kita bisa lihat bahwa pada saat skenario menggunakan node sebanyak 30 node dengan protokol OLSR yang dimodifikasi menunjukkan waktu *end to end delay* sebesar 18,4696 ms sedangkan protokol OLSR yang biasa menunjukkan waktu *end to end delay* 18,5677 ms dan pada skenario 50 node protokol OLSR yang dimodifikasi menunjukkan waktu *end to end delay* 31,3885 ms sedangkan yang menggunakan protokol OLSR menunjukkan waktu *end to end delay* 32,5842 ms begitu juga dengan skenario 70 dan 90 node protokol OLSR yang telah dimodifikasi menunjukkan waktu *end to end delay* yang lebih kecil dibandingkan dengan protokol OLSR biasa



Gambar 5-12 grafik topology control dengan kecepatan 10 m/s

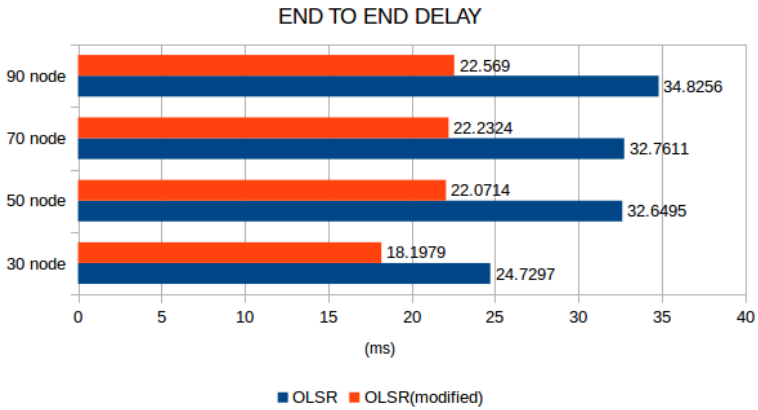
Dari hasil grafik diatas kita bisa lihat bahwa pada saat skenario menggunakan node sebanyak 30 node dengan protokol OLSR yang dioptimasi menunjukkan nilai TC sebesar 3609 paket sedangkan protokol OLSR yang biasa menunjukkan nilai TC 6102 paket dan pada skenario 50 node protokol OLSR yang dimodifikasi menunjukkan nilai TC 4817 paket sedangkan yang menggunakan protokol OLSR menunjukkan TC 5222 paket begitu juga dengan skenario 70 dan 90 node protokol OLSR yang telah dimodifikasi menunjukkan nilai TC yang lebih bagus dibandingkan dengan protokol OLSR biasa. Hal ini menunjukkan bahwa protokol OLSR yang dimodifikasi dapat mengurangi terjadinya *strorm broadcasting*

5.2.2.2 Skenario *Real* Pada Kecepatan 15 m/s



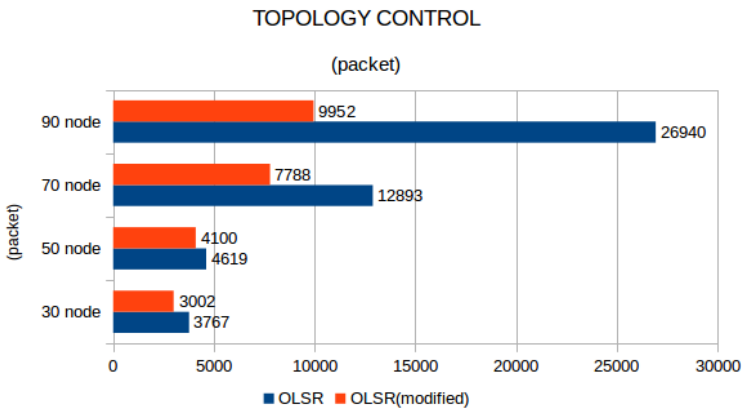
Gambar 5-13 grafik PDR dengan kecepatan 15 m/s

Dari hasil grafik diatas kita bisa lihat bahwa pada saat skenario menggunakan node sebanyak 30 node dengan protokol OLSR yang dimodifikasi menunjukkan nilai PDR sebesar 99,5% sedangkan protokol OLSR yang biasa menunjukkan nilai PDR sebesar 94,92% dan pada skenario 50 node protokol OLSR yang dimodifikasi menunjukkan nilai PDR sebesar 99,49% sedangkan yang menggunakan protokol OLSR menunjukkan PDR sebesar 96 % begitu juga dengan skenario 70 dan 90 node protokol OLSR yang telah dimodifikasi menunjukkan nilai PDR yang lebih bagus dibandingkan dengan protokol OLSR biasa



Gambar 5-14 grafik end to end delay dengan kecepatan 15 m/s

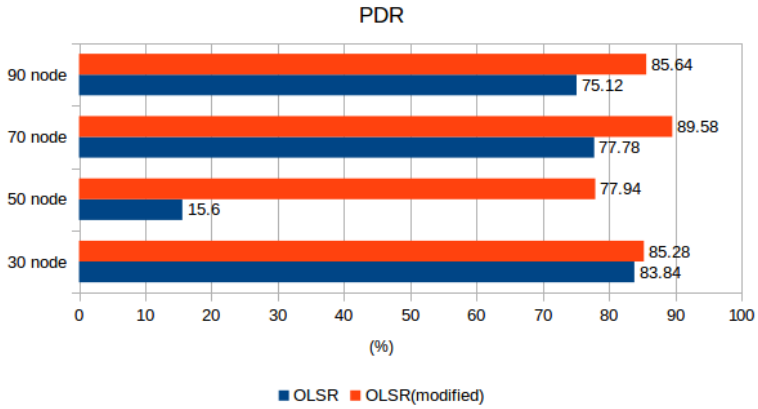
Dari hasil grafik diatas kita bisa lihat bahwa pada saat skenario menggunakan node sebanyak 30 node dengan protokol OLSR yang dimodifikasi menunjukkan waktu *end to end delay* sebesar 18,1979 ms sedangkan protokol OLSR yang biasa menunjukkan waktu *end to end delay* 24.7297 ms dan pada skenario 50 node protokol OLSR yang dimodifikasi menunjukkan waktu *end to end delay* 22,0714 ms sedangkan yang menggunakan protokol OLSR menunjukkan waktu end to end delay 32,6495 ms begitu juga dengan skenario 70 dan 90 node protokol OLSR yang telah dimodifikasi menunjukkan waktu *end to end delay* yang lebih kecil dibandingkan dengan protokol OLSR biasa



Gambar 5-15 grafik topology control dengan kecepatan 15 m/s

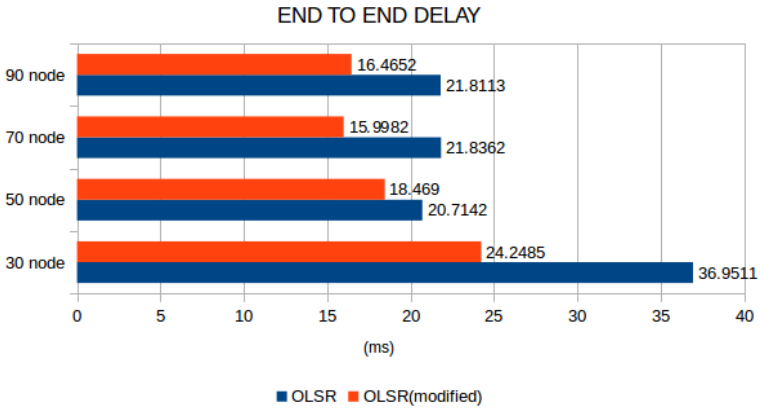
Dari hasil grafik diatas kita bisa lihat bahwa pada saat skenario menggunakan node sebanyak 30 node dengan protokol OLSR yang dimodifikasi menunjukkan nilai TC sebesar 3002 paket sedangkan protokol OLSR yang biasa menunjukkan nilai TC 3767 paket dan pada skenario 50 node protokol OLSR yang dimodifikasi menunjukkan nilai TC 4100 paket sedangkan yang menggunakan protokol OLSR menunjukkan TC 4619 paket begitu juga dengan skenario 70 dan 90 node protokol OLSR yang telah dimodifikasi menunjukkan nilai TC yang lebih bagus dibandingkan dengan protokol OLSR biasa. Hal ini menunjukkan bahwa protokol OLSR yang dimodifikasi dapat mengurangi terjadinya *strorm broadcasting*

5.2.2.3 Skenario Real Pada Kecepatan 20 m/s



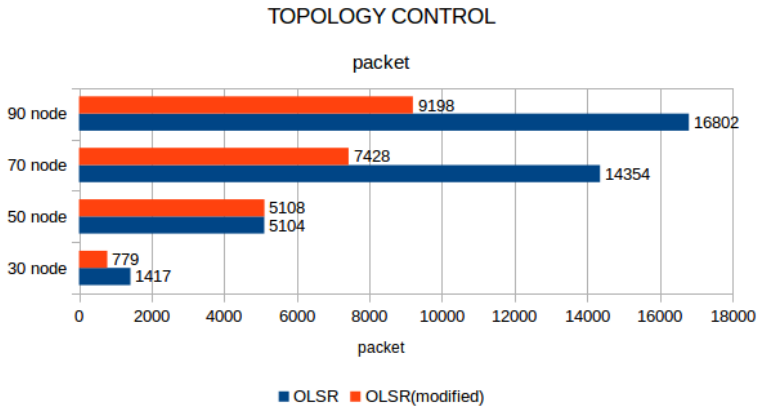
Gambar 5-16 grafik PDR dengan kecepatan 20 m/s

Dari hasil grafik diatas kita bisa lihat bahwa pada saat skenario menggunakan node sebanyak 30 node dengan protokol OLSR yang dimodifikasi menunjukkan nilai PDR sebesar 85,28% sedangkan protokol OLSR yang biasa menunjukkan nilai PDR sebesar 83,84% dan pada skenario 50 node protokol OLSR yang dimodifikasi menunjukkan nilai PDR sebesar 77,94% sedangkan yang menggunakan protokol OLSR menunjukkan PDR sebesar 15,6 % begitu juga dengan skenario 70 dan 90 node protokol OLSR yang telah dimodifikasi menunjukkan nilai PDR yang lebih bagus dibandingkan dengan protokol OLSR biasa



Gambar 5-17 grafik end to end delay dengan kecepatan 20 m/s

Dari hasil grafik diatas kita bisa lihat bahwa pada saat skenario menggunakan node sebanyak 30 node dengan protokol OLSR yang dimodifikasi menunjukkan waktu *end to end delay* sebesar 24,2485 ms sedangkan protokol OLSR yang biasa menunjukkan waktu *end to end delay* 36,9511 ms dan pada skenario 50 node protokol OLSR yang dimodifikasi menunjukkan waktu end to end delay 18,469 ms sedangkan yang menggunakan protokol OLSR menunjukkan waktu *end to end delay* 20,7142 ms begitu juga dengan skenario 70 dan 90 node protokol OLSR yang telah dimodifikasi menunjukkan waktu *end to end delay* yang lebih kecil dibandingkan dengan protokol OLSR biasa



Gambar 5-18 grafik topology control dengan kecepatan 20 m/s

Dari hasil grafik diatas kita bisa lihat bahwa pada saat skenario menggunakan node sebanyak 30 node dengan protokol OLSR yang dimodifikasi menunjukkan nilai TC sebesar 779 paket sedangkan protokol OLSR yang biasa menunjukkan nilai TC 1417 paket dan pada skenario 50 node protokol OLSR yang dioptimasi menunjukkan nilai TC 5108 paket sedangkan yang menggunakan protokol OLSR menunjukkan TC 5104 paket begitu juga dengan skenario 70 dan 90 node protokol OLSR yang telah dimodifikasi menunjukkan nilai TC yang lebih bagus dibandingkan dengan protokol OLSR biasa. Hal ini menunjukkan bahwa protokol OLSR yang dimodifikasi dapat mengurangi terjadinya *strorm broadcasting*. Pada node 50 terjadi peningkatan node jika dibandingkan OLSR biasa hal itu dikarenakan ada node yang mati sehingga meningkatkan jumlah node yang meneruskan pesan.

[Halaman ini sengaja dikosongkan]

BAB VI
KESIMPULAN DAN SARAN

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

6.1 Kesimpulan

1. Hasil pengujian PDR baik pada seknario *real* maupun *grid* menunjukkan bahwa protokol OLSR yang di modifikasi menghasilkan hasil yang lebih baik daripada protokol OLSR biasa.Dari penghitungan PDR skenario *real* menunjukkan nilai rata – ratayang lebih besar yaitu 90,28 % untuk OLSR yang dimodifikasi sedangkan untuk OLSR standar memiliki nilai rata-rata 79,51%dan skenario *grid* menunjukkan nilai rata-rata 90,15%. untuk OLSR yang dimodifikasi sedangkan OLSR standar menunjukan nilai rata rata 88,23%.
2. Hasil pengujian *end to end delay* menunjukkan secara keseluruhan bahwa protokol OLSR yang dimodifikasi menghasilkan nilai yang lebih kecil dibandingkan dengan protokol OLSR biasa.Hal itu ditunjukan dari hasil perhitungan *end to end delay* skenario rill OLSR yang dimodifikasi menunjukan nilai sebesar 16.2281 ms sedangkan pada OLSR biasa menunjukan nilai sebesar 22.594 ms dan pada skenario grid OLSR yang dimodifikasi menunjukan nilai sebesar 22.2849 ms sedangkan pada OLSR biasa menunjukan nilai sebesar 23.533 ms. Hal itu menunjukkan bahwa protokol OLSR yang dimodifikasi berhasil menurunkan waktu yang dibutuhkan untuk sampai node tujuan. Tetapi pada skenario *grid* dengan kecepatan 20 m/s dan node 50 menunjukkan protokol OLSR lebih bagus dibandingkan dengan yang dimodifikasi hal itu karena pada node

tersebut terdapat pola node yang beraturan sehingga waktu yang dibutuhkan untuk pengoperasian lebih cepat

3. Hasil pengujian *Topology Control* pada skenario grid maupun *real* menunjukkan bahwa protokol OLSR yang dimodifikasi menunjukkan jumlah paket yang lebih sedikit jika dibandingkan protokol OLSR yang biasa.Hal itu ditunjukan dari penghitungan *Topology Control* skenario *real* OLSR yang dimodifikasi menunjukkan banyak paket yang dikirim rata-rata adalah 6216 paket sedangkan pada OLSR biasa menunjukan banyak paket yang dikirim rata-rata adalah 11266 paket.Dan pada skenario *grid* menunjukkan jumlah paket yang dikirim rata-rata 22931 paket sedangkan pada OLSR biasa menunjukan banyak paket yang dikirim rata-rata adalah 23980 paket.Hal itu menunjukkan bahwa Protokol OLSR yang dimodifikasi ini cocok untuk skenario real yang bersifat lebih dinamis.Hal itu dikarenakan pada protokol OLSR biasa setelah dilakukan metode PGB node yang berada di area PGB akan dikirimi dan meneruskan node ke node selanjutnya teteapi pada OLSR yang dimodifikasi setelah dilakukan metode PGB node yang berada di area PGB akan dicluster lagi sehingga node yang memiliki kekuatan yang bagus dan memiliki node tetangga yang dapat menjangkau banyak node yang dikirimi dan boleh meneruskan pesan. Sehingga dengan semakin sedikit nilai *Topology control* maka dapat mengurangi kemungkinan terjadinya *Storm Broadcasting*

6.2 Saran

Perlu diadakan penelitian lebih lanjut untuk optimasi waktu *end to end delay* dan penurunan jumlah paket yang dikirm agar lebih kecil kemungkinan terjadinya *storm broadcasting*

DAFTAR PUSTAKA

[1] D. P. I. I. Ismail and M. H. F. Ja'afar, "Mobile Ad l Network Overview," in *Asia-Pacific Conference On App Electromagnetics Proceedings*, Malaka, 2007.

[2] M. Sangari and Dr.K.Baskaran, "A Comprehensive Survey Efficient Routing Protocols And Simulation Tools VANET," *International Journal of Computer Science Information Technologies*, , vol. 5 (3), pp. 2729-2737, 2014

[3] V. Naumov, R. Baumann and T. Gross, "An Evaluation Inter-Vehicle Ad Hoc Networks Based on Realistic Vehic Traces," *MobiHoc '06 Proceedings of the 7th A international symposium on Mobile ad hoc networking computing*, pp. 108-119, 2006.

[4] OpenStreetMap Foundation, "OpenStreetMa OpenStreetMap Foundation, [Online]. Availa https://www.openstreetmap.org. [Accessed 20 June 2016].

[5] OpenstreetMap, "JOSM,", 13 June 2016. [Online]. Availa http://openstreetmap.org/JOSM. [Accessed 20 June 2016].

[6] JavaOpen, "AWK,", 10 June 2016. [Online]. Availa https://javaopen.org [Accessed 20 June 2016].

[7] K. Fall and K. Varadhan, "The Manual (formerly Notes Documentation)," UC Berkeley, LBL, USC/ISI, and Xe PARC., 5 November 2011. [Online]. Availa http://www.isi.edu/nsnam/ns/doc/index.html. [Accessed Desember 2015].

[8] SUMO, "SUMO User Documentation," SUMO, 6 June 2([Online]. Availa http://sumo.dlr.de/SUMO_User_Documentation. [Accessed

June 2016].

[9] R. F. Sari, A. Syarif and B. Budiardjo, "Analisis Kinerja Protokol Routing Ad Hoc On-Demand Distance Vector (AODV) Pada Jaringan Ad Hoc Hybrid: Perbandingan Hasil Simulasi dengan NS-2 Dan Implementasi Pada Testbed dengan PDA," *MAKARA, TEKNOLOGI, VOLUME 12, NO. 1*, pp. 7-18, 2008.

[10] R. A. Siregar and W. Wibisono, PENINGKATAN KINERJA PROTOKOL OPTIMIZED LINK STATE ROUTING PADA LINGKUNGAN VEHICULAR ADHOC NETWORK DENGAN MENGGUNAKAN PREDIKSI MOBILITAS DAN MULTIPATH ROUTING, Surabaya: Fakultas Teknologi Informasi Jurusan Teknik Informatika, 2016.463.

[11] N. Sorrecy, "OLSR OPTIMIZE LINK STATE ROUTING," Network Sorrecy, 2012. [Online]. [Accessed 28 1 2017].

[12] m. Gruteser and M. T. Moreno, SIGMobile, 24 September 2010. [Online]. Available: https://www.sigmobile.org/workshops/vanet2010/index.html. [Accessed 24 1 2017].

[13] t. issariyakul and h. ekram, introduction to network second edition, ottawa: springer, 2006.

[14] U. agrwal and s. Moika, "Comparative and Behavioral Study of Various Routing Protocols in VANET," vol. 3, no. 10, 2013.

LAMPIRAN

A 1. Kode Skenario NS-2

```
set val(chan)        Channel/WirelessChannel;
set val(prop)        Propagation/TwoRayGround;
set val(netif)             Phy/WirelessPhy;
set val(mac)         Mac/802_11    ;
set val(ifq)         Queue/DropTail/PriQueue;
set val(ll)          LL            ;
set val(ant)         Antenna/OmniAntenna;
set val(ifqlen)            50            ;
set val(nn)          50                  ;
set val(rp)          OLSR                ;
set val(energymodel)       EnergyModel   ;
set val(initialenergy)     100           ;
set val(lm)          "off"               ;
set val(x)        1000                   ;
set val(y)           1000                ;
set val(stop)        200                 ;
set val(cp)              "traffic1"      ;
set val(sc)              "ns2.tcl"
set ns_       [new Simulator]
set tracefd [open s-aodv2.tr w]
set namtrace [open s-aodv2.nam w]
#$ns_ use-newtrace
$ns_ trace-all $tracefd
$ns_ namtrace-all-wireless $namtrace $val(x)
$val(y)
Agent/AODV set num_nodes $val(nn)

# Setting up Topography Object

set topo [new Topography]
$topo load_flatgrid $val(x) $val(y)
create-god $val(nn)

set chan_1_ [new $val(chan)]

# Configure the nodes
```

```
$ns_ node-config    -adhocRouting $val(rp) \
                    -llType $val(ll) \
                    -macType $val(mac) \
                    -channel $chan_1_ \
                    -ifqType $val(ifq) \
                    -ifqLen $val(ifqlen) \
                    -antType $val(ant) \
                    -propType $val(prop) \
                    -phyType $val(netif) \
                    -topoInstance $topo \
                    -agentTrace ON \
                    -routerTrace ON \
                    -macTrace OFF \
                    -movementTrace ON \

for {set i 0} {$i < $val(nn)} {incr i} {
set node_($i) [$ns_ node]
}

# Provide Initial Location of Mobile Nodes
puts "Loading connection pattern..."
source $val(cp)
puts "Loading scenario file..."

source $val(sc)

for {set i 0} {$i < $val(nn)} {incr i} {
$ns_ initial_node_pos $node_($i) 30
}
for {set i 0} {$i < $val(nn)} { incr i} {
$ns_ at $val(stop) "$node_($i) reset"
}
$ns_ at 200.01 "puts \"end simulation\" ; $ns_
halt"
proc stop {} {
global ns_ tracefd namtrace
$ns_ flush-trace
close $tracefd
close $namtrace
#exec nam aodv.nam &
}
$ns  run
```

A 2*Script Pattern* Skenario

```
#
# nodes: 50, max conn: 1, send rate: 1, seed: 1
#
#
# 1 connecting to 2 at time 2.5568388786897245
#
set udp_(0) [new Agent/UDP]
$ns_ attach-agent $node_(1) $udp_(0)
set null_(0) [new Agent/Null]
$ns_ attach-agent $node_(29) $null_(0)
set cbr_(0) [new Application/Traffic/CBR]
$cbr_(0) set packetSize_ 512
$cbr_(0) set interval_ 1
$cbr_(0) set random_ 1
$cbr_(0) set maxpkts_ 10000
$cbr_(0) attach-agent $udp_(0)
$ns_ connect $udp_(0) $null_(0)
$ns_ at 2.5568388786897245 "$cbr_(0) start"
#
#Total sources/connections: 1/1
#
```

A 3Kode awk Perhitungan *Packet Delivery Ratio*

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

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

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

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

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

A 4Kode awk Perhitungan *End-to-End Delay*

```
BEGIN {

    seqno = -1;

    count = 0;}
{
    if($4 == "AGT" && $1 == "s" && seqno < $6) {

          seqno = $6;

    }
    #end-to-end delay

    if($4 == "AGT" && $1 == "s") {

          start_time[$6] = $2;

    } else if(($7 == "cbr") && ($1 == "r")) {

        end_time[$6] = $2;

    } else if($1 == "D" && $7 == "cbr") {

          end_time[$6] = -1;

    }

}

END {

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

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

BIODATA PENULIS

Irfan Naufal P. S. dilahirkan pada tanggal 16 September 1994 di Surabaya. Pada tahun 2012. Penulis menemupuh pendidikan mualia dari SDN Asemrowo 8 Surabaya (2000-2006), SMPN5 Surabaya (2006-2009), SMAN7 Surabaya. Sesudah lulus dari SMAN 7 Surabaya melanjutkan menimba ilmu di jurusan Teknik Informatika Fakultas Teknologi Informasi Institut Teknologi Sepuluh Nopember Surabaya

Selama kuliah di teknik informatika ITS, Penulis mengambil rumpun mata kuliah Arsitektur dan Jaringan Komputer (AJK). Komunikasi dengan penulis dapat melalui email : **irfannaufalprawirosamudro@gmail.com.**