TESIS

OPTIMASI KINERJA PROTOKOL AODV PADA SKENARIO *VEHICLE TO VEHICLE COMMUNICATION* DENGAN *STATIC INTERSECTION NODE*

Johan Ericka W.P
NRP. 5112201009

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

Waskitho Wibisono, S.Kom., M.Eng., Ph.D

PROGRAM MAGISTER
BIDANG KEAHLIAN KOMPUTASI BERBASIS JARINGAN
JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2016

[Halaman ini sengaja dikosongkan]

TESIS

OPTIMASI KINERJA PROTOKOL AODV PADA SKENARIO *VEHICLE TO VEHICLE COMMUNICATION* DENGAN *STATIC INTERSECTION NODE*

Johan Ericka W.P
NRP. 5112201009

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

Waskitho Wibisono, S.Kom., M.Eng., Ph.D

PROGRAM MAGISTER
BIDANG KEAHLIAN KOMPUTASI BERBASIS JARINGAN
JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2016

[Halaman ini sengaja dikosongkan]

THESIS

OPTIMIZING AODV PERFORMANCE IN VEHICLE TO VEHICLE COMMUNICATON USING STATIC INTERSECTION NODE

Johan Ericka W.P
NRP. 5112201009

SUPERVISOR
Dr.Eng. Radityo Anggoro, S.Kom., M.Sc.
Waskitho Wibisono, S.Kom., M.Eng., Ph.D

MAGISTER PROGRAMME
INFORMATICS ENGINEERING DEPARTMENT
FACULTY OF INFORMATION TECHNOLOGY
SEPULUH NOPEMBER INSTITUTE OF TECHNOLOGY
SURABAYA
2016

[Halaman ini sengaja dikosongkan]

Tesis disusun untuk memenuhi salah satu syarat memperoleh gelar
Magister Komputer (M.Kom)
di
Institut Teknologi Sepuluh Nopember Surabaya
Oleh:
JOHAN ERICKA WAHYU PRAKASA
NRP. 5112201009

Dengan judul:
Optimasi Kinerja Protokol AODV Pada Skenario *Vehicle To Vehicle Communication*
Dengan *Static Intersection Node*

Tanggal Ujian : 21 Juni 2016
Periode Wisuda : 2015 Genap

Disetujui Oleh:

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

Waskitho Wibisono, S.Kom, M.Eng, Ph.D
NIP. 197410222000031001 ..... (Pembimbing 2)

Prof.Ir.Supeno Djanali, MSc,PhD
NIP. 194806191973011001 ..... (Penguji 1)

Tohari Ahmad, S.Kom, MIT, Ph.D
NIP. 197505252003121002 ..... (Penguji 2)

Henning Titi Ciptaningtyas, S. Kom, M. Kom
NIP. 198407082010122004 ..... (Penguji 3)

Direktur Program Pascasarjana,

Prof. Ir. Djauhar Manfaat, M.Sc, Ph. D
NIP. 196012021987011001

[Halaman ini sengaja dikosongkan]

OPTIMASI KINERJA PROTOKOL AODV PADA SKENARIO *VEHICLE TO VEHICLECOMMUNICATION* DENGAN *STATIC INTERSECTION NODE*

Nama Mahasiswa	:	Johan Ericka
NRP	:	5112201009
Jurusan	:	Teknik Informatika, FTIF-ITS
Dosen Pembimbing 1	:	Dr.Eng. Radityo Anggoro, S.Kom., M.Sc.
Dosen Pembimbing 2	:	Waskitho Wibisono, S.Kom., M.Eng., PhD

ABSTRAK

VANET (*Vehicular Adhoc Network*) merupakan teknologi yang dikembangkan dari MANET (*Mobile Adhoc Network*) dimana teknologi ini memungkinkan terjadinya komunikasi antar kendaraan atau *Vehicle to Vehicle Communication*tanpa adanya infrastruktur jaringan yang tetap. Komunikasi dapat terjadi dengan cara saling mengirimkan data antar kendaraan sehingga data dapat sampai ke kendaraan tujuan. Maka dibutuhkan protokol *routing* untuk mengetahui jalur pengiriman data yang dalam penelitian ini menggunakan protokol *routing* AODV.

Untuk meningkatkan performa protokol *routing*, maka pada penelitian ini ditambahkan *SIN* (*Static Intersection Node*). *Static Intersection Node* adalah *RSU* (*Road Side Unit*) yang diletakkan di persimpangan jalan (intersection). Fungsi dari *Static Intersection Node* pada penelitian ini adalah sebagai *repeater* untuk membantu mengirimkan paket data ke kendaraan lain yang berada disekitarnya sehingga dapat meningkatkan *Packet Delivery Ratio* serta meminimalkan *Packet Loss* dan *End to End Delay*. Fokus utama penelitian ini adalah untuk menentukan posisi *Static Intersection Node* yang paling ideal sekaligus jumlah *Static Intersection Node* yang paling efektif.

Kata kunci : ***vanet,* aodv, *static intersection node***

[Halaman ini sengaja dikosongkan]

OPTIMIZING AODV PERFORMANCE IN VEHICLE TO VEHICLE COMMUNICATON USTATIC INTERSECTION STATIC NODE

Student's Name	:	Johan Ericka
Student's ID	:	5112201009
Department	:	Informatics Engineering, FTIF-ITS
First Advisor	:	Dr.Eng. Radityo Anggoro, S.Kom., M.Sc.
Second Advisor	:	Waskitho Wibisono, S.Kom., M.Eng., PhD

ABSTRACT

VANET (Vehicular Adhoc Network) is a relatively new computer network concept that enabling vehicle to vehicle communication without static network infrastructure. Vehicle to vehicle communication achieved by relaying data through the nodes between sender and receiver node even with no direct connection between them. To achieve this, the routing protocol take the most important role in VANET. In this research will use AODV as the routing protocol

This research will enhance AODV proformance by adding some Static Intersection Node in the map. Static Intersection Node is a Road Side Unit placed in the intersection. The Static Intersection Node will help to relay the data packet to its destination.

This research will focus to find the best Static Intersection Node position so the Packet Delivery Ratio will be better than not using Static Intersection Node. And also to minimize Packet Loss and End to End Delay between sender and receiver.

Keywords: VANET ,AODV, static intersection node

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

Segala puji bagi Alloh SWT yang telah memberikan rahmatnya sehingga penulis dapat menyelesaikan tesis yang berjudul "Optimasi Kinerja Protokol AODV Pada Skenario Vehicle To Vehicle Communication Dengan Static Intersection Node". Tesis ini disusun untuk memenuhi sebagian persyaratan guna memperoleh gelar sarjana Magister Komputer di Institut Teknologi Sepuluh Nopember.

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

1. Alloh SWT Tuhan semesta alam yang maha mengatur segala sesuatu. Hanya karena berkat rahmat dan hidayahnya serta ijin nya akhirnya tesis ini dapat terselesaikan.
2. Para dosen pembimbing tesis Bapak Dr.Eng. Radityo Anggoro, S.Kom, M.Sc yang telah bersedia membimbing hingga tesis ini selesai serta Bapak Waskitho Wibisono, S.Kom, M.Eng, Ph.D, selaku Ketua Program Studi Pascasarjana Teknik Informatika Institut Teknologi Sepuluh Nopember sekaligus sebagai pembimbing tesis yang terus memberikan support untuk segera menyelesaikan tesis ini.
3. Orang tua penulis, yang telah memberikan dukungan mental dan spiritual yang sangat berharga selama penulis menuntut ilmu di Institut Teknologi Sepuluh Nopember ini.
4. Istri dan anak penulis, yang telah memberikan dukungan, pengertian dan do'a selama penulis menuntut ilmu di Institut Teknologi Sepuluh Nopember ini.
5. Bapak Ibu Dosen pengajar di Program Pascasarjana Teknik Informatika Institut Teknologi Sepuluh Nopember, yang telah mengajarkan banyak ilmunya.
6. Teman-teman Pascasarjana FTIF angkatan 2012, atas bantuan, semangat, do'a, pengalaman, dan kenangan-kenangan berharga selama penulis menuntut ilmu di Institut Teknologi Sepuluh Nopember ini. Terutama keluarga besar

Sekte NCC 2012, yang telah menyemangati untuk terus berjuang sampai akhir.

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

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

Malang, 27Juni 2016

Johan Ericka Wahyu Prakasa

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

[Halaman ini sengaja di kosongkan]

DAFTAR GRAFIK

[Halaman ini sengaja di kosongkan]

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dari penelitian ini dapat disimpulkan bahwa keberadaan *Static Intersection Node* pada peta terbukti meningkatkan performa protokol AODV dalam melakukan pengiriman data. Hal tersebut dibuktikan dengan hasil penelitian yang menunjukkan adanya peningkatan *Packet Delivery Ratio* rata – rata sebesar 21,5% setelah diletakkan *Static Intersection Node* pada beberapa titik optimal pada peta. Demikian juga dengan *Packet Loss Ratio* juga mengalami penurunan rata – rata sebesar 7,25% serta *End to End Delay* juga mengalami penurunan rata – rata sebesar 1,04 detik yang menunjukkan bahwa paket data semakin cepat diterima oleh node *receiver*.

Berdasarkan hasil dari penelitian ini dapat disimpulkan bahwa keberadaan *Static Intersection Node* pada peta dapat memberikan kontribusi pada perbaikan performa protokol AODV. Pada penelitian ini *Static Intersection Node* diletakkan pada ruas jalan yang memiliki kepadatan kendaraan rendah dan hasil penelitian menunjukkan terdapat perbaikan performa protokol AODV dengan jumlah *Static Intersection Node* yang minimal. Maka dapat disimpulkan bahwa penentuan posisi *Static Intersection Node* pada peta berpengaruh terhadap perbaikan performa protokol AODV.

5.2 Saran

Penelitian ini masih dapat dikembangkan lebih lanjut dengan melakukan pengiriman data dari node sumber dan tujuan yang bergerak. Serta dapat juga digunakan protokol routing lain (misal pada kelompok proaktif) atau dengan jumlah dan jenis kendaraan serta kecepatan yang bervariasi untuk mendapatkan data yang lebih valid.

[Halaman ini sengaja di kosongkan]

DAFTAR PUSTAKA

Perkins, C.E.; Royer, E.M., "Ad-hoc on-demand distance vector routing," in *Mobile Computing Systems and Applications, 1999. Proceedings. WMCSA '99. Second IEEE Workshop on* , vol., no., pp.90-100, 25-26 Feb 1999

Nakamura, M., Kitani, T., Sun, W., Shibata, N., Yasumoto, K. and Ito, M. 2010. A method for improving data delivery efficiency in delay tolerant vanet with scheduled routes of cars. pp. 1--5.

Menouar, H., Filali, F. and Lenardi, M. 2006. A survey and qualitative analysis of MAC protocols for vehicular ad hoc networks. Wireless Communications, IEEE, 13 (5), pp. 30--35.

Narendra Mohan Mittal, Dr. Prem Chand Vashist, 2014. Performance Evaluation of AODV and DSR Routing Protocols for Vehicular Ad-hoc Networks (VANETs), International Journal of Emerging Technology and Advanced Engineering vol. 4 issue 6 pp 522 – 530

Yong Ding, Chen Wang, Li Xiao, 2007. A Static-Node Assisted Adaptive Routing Protocol in Vehicular Networks, VANET '07 Proceedings of the fourth ACM international workshop on Vehicular ad hoc networks Pages 59 – 68

Anggoro, Radityo, 2008. Static Intersection Node-based Multicast Protocol in VANET, Thesis, Computer Science and Information Engineering Department in National Taiwan University of Science and Technology.

Festag, A. Hessler, A. Baldessari, R., Le., L., Zhang, W., & Westhoff, D 2008 Vehicle-To-Vehicle And Road-Side Sensor Communication For Enhanced Road Safety, 9th International Conference On Intelligent Tutoring Systems (ITS 2008) IEEE Press

Lakshmi Ramachandran, Sangheethaa Sukumaran, Surya Rani Sunny, 2013 An Intersection Based Traffic Aware Routing With Low Overhead inVANET, International Journal of Digital Information and Wireless Communications (IJDIWC) 3(2): 190-196

Sok-Ian Sou and Ozan K. Tonguz, 2011 Enhancing VANET Connectivity Through Roadside Units on Highways, IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, VOL. 60, NO. 8, OCTOBER 2011

Evellyn S. Cavalcante, Andre L. L. Aquino, Gisele L. Pappa, 2012, Roadside Unit Deployment for Information Disseminationin a VANET: An Evolutionary Approach, GECCO'12 Companion, July 7–11, 2012

Khaleel Mershad, Hassan Artail, Mario Gerla, 2012 ROAMER: Roadside Units as message routers in VANETs, Ad Hoc Networks Vol. 10 (3) 2012 pp. 479 - 492

Christian Lochert, Björn Scheuermann, Christian Wewetzer, Andreas Luebke, Martin Mauve, 2008, Data Aggregation and Roadside Unit Placement for a VANET Traffic Information System, VANET '08 Proceedings of the fifth ACM international workshop on VehiculAr Inter-NETworking Pages 58-65

Jeonghee Chi, Yeongwon Jo, Hyunsun Park, Taehyeon Hwang and Soyoung Park, 2013 An Effective ROAD SIDE UNIT Allocation Strategy for Maximizing Vehicular Network Connectivity, International Journal of Control and Automation Vol. 6 No.4 August, 2013 pp. 259 – 270

BIODATA PENULIS

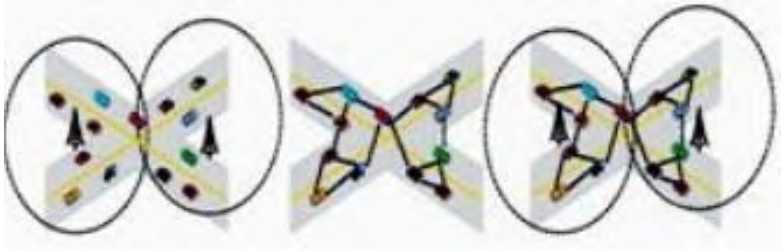
Penulis adalah anak pertama dari dua bersaudara yang lahir di Malang pada tanggal 13Desember 1983. Pendidikan sekolah dasarnya ditempuh di Sekolah Dasar Negeri Sawojajar 3 Kota Malang, kemudian dilanjutkan ke SLTPN 8 Malang dan SMKTelkom Malang jurusan Teknik Informatika. Pada tahun 2006 penulis berhasil mendapatkan gelar Sarjana Komputer (S.Kom) dari Sekolah Tinggi Informatika & Komputer Indonesia (STIKI) Malang. Pada tahun 2009, penulis diminta untuk kembali menjadi tenaga pengajar di STIKI Malang. Karena minat penulis pada bidang jaringan komputer cukup tinggi maka sejak tahun 2012, penulis diangkat menjadi Koordinator Inftrastruktur Jaringan & TI. Dimana dengan jabatan ini penulis bertanggung jawab penuh terhadap jaringan komputer serta semua server di STIKI Malang. Karena hal inilah ketika penulis mendapatkan kesempatan untuk melanjutkan pendidikan pascasarjana di jurusan Teknik Informatika Institut Teknologi Sepuluh Nopember (ITS) Surabaya pada tahun 2012 melalui beasiswa BPPS-DN (Beasiswa Program Pasca Sarjana Dalam Negeri) penulis memutuskan untuk megambil konsentasi pada bidang Komputasi Berbasis Jaringan.Korespondensi dengan penulis dapat dilakukan melalui email pada alamat johanericka@gmail.com.

BAB I
PENDAHULUAN

1.1 Latar Belakang

VANET (*Vehicular Adhoc Network*) adalah salah satu bidang penelitian yang sedang berkembang. Teknologi ini merupakan pengembangan dari MANET (*Mobile Adhoc Network*) yang memungkinkan terjadinya pengiriman data antar kendaraan dan dari kendaraan ke *Road Side Unit* meskipun tidak terdapat infrastruktur jaringan yang tetap diantara pengirim dan penerima (Nidhi et al, 2012). Data dikirimkan antar kendaraan atau dari kendaraan ke *Road Side Unit* secara bertahap sampai ke kendaraan atau *Road Side Unit* yang dituju. Dalam hal ini komunikasi data dilakukan secara nirkabel dengan memanfaatkan protokol *Dedicated Short Range Communication* (DSRC). Protokol DSRC ini menggunakan sebagian spektrum dari frekuensi 5,9 Ghz untuk mengirimkan data.

Secara umum, topologi jaringan VANET terbagi atas 3 kelompok yaitu *Pure Cellular, Pure Adhoc* dan *Hybrid* seperti yang ditampilkan pada Gambar 1. 0-1. Menurut penelitian yang telah di lakukan (Kumar, et al 2011) *Pure cellular* adalah jaringan yang dibentuk oleh *Wireless LAN* dan 3G yang dapat digunakan sebagai jalur ke internet. *Pure Adhoc* adalah jaringan yang dibentuk antar *vehicle*, sedangkan *Hybrid* adalah jaringan yang mengkombinasikan *Pure Cellular* dan *Pure Adhoc.*



Gambar 1. 1 *Pure Cellular*, *Pure Adhoc* dan *Hybrid Network* (Kumar, et al 2011)

Dikatakan *Ad Hoc* karena jaringan dapat tercipta dan hilang setiap saat. Hal ini dikaren posisi kendaraan yang selalu berubah. Maka topologi yang tercipta selalu berubah karena mobilitas kendaraan tersebut.

Salah satu kunci dari keberhasilan pengiriman data pada VANET adalah algoritma pemilihan jalur yang merupakan tugas protokol *routing*. Protokol routing yang banyak digunakan pada penelitian di lingkungan VANET adalah AODV (*Adhoc On Demand Distance Vector*) dimana merupakan salah satu protokol *routing* reaktif. Pada protokol ini untuk mengetahui jalur menuju penerima atau yang sering disebut dengan *Path Discovery*, menggunakan mekanisme *Route Request* dan *Route Reply* (Perkins et al, 1999). Mekanisme *Route Request* digunakan untuk mencari jalur menuju penerima sedangkan *Route Reply* digunakan untuk mengirimkan informasi jalur yang telah ditempun oleh *Route Reply* agar sampai kepada penerima. Sehingga kendaraan pengirim dapat mengetahui kendaraan / jalur mana saja yang dilalui oleh data yang dikirimkan.

Karena proses pengiriman data bergantung pada kendaraan yang menjadi perantara pengiriman, maka tingkat keberhasilan pengiriman paket data bervariasi. Untuk meningkatkan keberhasilan pengiriman paket data, maka pada penelitian ini memanfaatan *Static Intersection Node*. *Static Intersection Node* adalah *Road Side Unit* yang berada di persimpangan jalan. Tujuan dari keberadaan *Static Intersection Node* adalah untuk meningkatkan performa protokol AODV dalam menentukan jalur pengiriman data.

1.2 Rumusan Permasalahan

Pengiriman data pada lingkungan VANET memiliki tantangan utama yaitu topologi jaringan yang dinamis / selalu berubah, sehingga proses pengiriman data baru dilaksanakan setelah ditemukan jalur dari pengirim ke penerima. Maka rumusan masalah yang di bahas pada penelitian ini adalah sebagai berikut :

1. Sejauh mana *Static Intersection Node* dapat meningkatkan performa protokol *routing* AODV pada lingkungan VANET ?
2. Bagaimana menentukan lokasi *Static Intersection Node* yang optimal pada peta

1.3 Batasan Masalah

Agar penelitian yang dilakukan dapat fokus, maka dibuat beberapa batasan masalah antara lain sebagai berikut :

1. Proses pengujian menggunakan Network Simulator versi 2.
2. Protokol routing yang diuji coba adalah AODV.
3. Mobilitas kendaraan dibuat menggunakan SUMO.
4. Ukuran kendaraan seragam (sedan).
5. Kecepatan kendaraan maksimal 8 m/s atau 30 Km/jam
6. Peta uji coba menggunakan peta nyata yang didapat dari OpenStreetMap

1.4 Tujuan

Adapun tujuan yang dicapai pada penelitian ini antara lain sebagai berikut

1. Terjadinya peningkatan rasio keberhasilan pengiriman data dengan adanya *Static Intersection Node* pada peta.
2. Mengetahui posisi optimal *Static Intersection Node* pada peta.

1.5 Manfaat

Penelitian ini bermanfaat untuk mengetahui peranan *Static Intersection Node* dalam peningkatan keberhasilan pengiriman paket pada lingkungan VANET.

1.6 Kontribusi Penelitian

Penelitian ini memiliki kontribusi sebagai berikut :

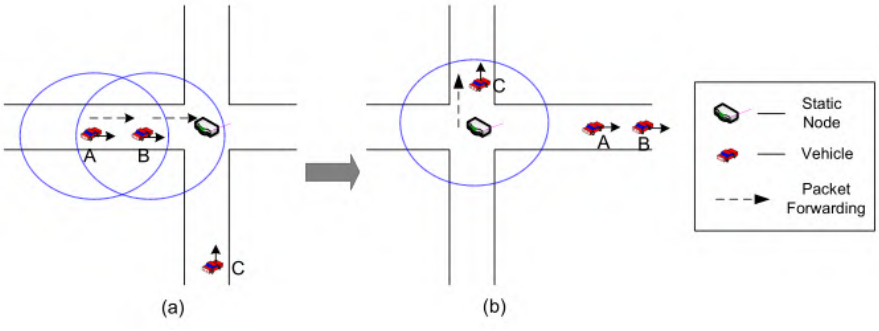
1. Mengukur peningkatan performa pada lingkungan VANET tanpa adanya *Static Intersection Node* dan dengan adanya *Static Intersection Node*.
2. Mengetahui posisi *Static Intersection Node* yang optimal pada peta.

[Halaman ini sengaja di kosongkan]

BAB II
KAJIAN PUSTAKA

Beberapa penelitian VANET telah dilakukan dengan menggunakan *Static Intersection Node* sebagai salah satu komponen pendukung VANET. Salah satunya adalah penelitian tentang *hybrid communication* untuk mencegah terjadinya kecelakaan (Festag, A et al, 2008). Penelitian ini mengusulkan ketika sebuah kendaraan masuk ke dalam jangkauan *Static Intersection Node* maka kendaaraan tersebut mendapatkan data terbaru untuk wilayah tersebut (cuaca, halangan dll) sehingga setelah dilakukan pemrosesan data dapat diketahui kondisi jalan di depan nya. Maka ketika terdapat potensi bahaya berdasarkan dari data yang telah diolah, maka pengemudi bisa mendapatkan peringatan dini. Penelitian ini memiliki konsep yang baik namun memiliki kelemahan dalam hal pemrosesan data. Jumlah data berbanding lurus dengan kecepatan pemrosesannya. Sehingga ketika jumlah data yang diterima terlalu banyak maka proses yang dibutuhkan untuk mengolah data tersebut juga semakin lama.

Penelitian lain yang menggunakan *Road Side Unit* untuk membantu mengirimkan paket ke *node* lain juga telah dilakukan (Ding, Y. et al 2007). Penelitian ini mengusulkan sebuah protokol baru untuk mengirim data ke *node* yang berada jauh dari *node* pengirim yang dinam *Static-node assisted Adaptive data Dissemination protocol for Vehicular Networks* (SADV). Inti dari penelitian ini adalah membuat data berhenti sementara di *Road Side Unit* sampai ditemukan jalur yang paling baik untuk mengirimkan data ke node tujuan seperti pada Gambar 2.1. *Road Side Unit* atau *Static Node* diposisikan di setiap persimpangan. Dengan mempertimbangkan posisi kendaraan yang akan melalui persimpangan tesebut maka *Road Side Unit* akan memberikan data kepada kendaraan yang diperkirakan menuju ke arah *receiver*. Dengan demikian diharapkan data dapat lebih cepat untuk mencapai *receiver*. Namun permasalahan akan terjadi ketika tidak terdapat kendaraan yang menuju ke arah *receiver*. Untuk mencegah buffer overflow maka diberlakukan mekanisme timeout terhadap data yang berada di memoy *Road Side Unit*.



Gambar 2. 1 Protokol *Static-node assisted Adaptive data Dissemination protocol for Vehicular Networks* (SADV) (Ding, Y. et al 2007)

Gambar 2.1 di atas mengilustrasikan paket data dikirimkan dari node A ke node B. Karena node B telah memasuki *range Road Side Unit* maka data diteruskan ke *Road Side Unit*. Diasumsikan jalur menuju node tujuan telah diketahui sebelumnya, maka *Road Side Unit* meneruskan data ke node C yang lebih cepat mencapai tujuan. Dengan menggunakan teknik ini terdapat keuntungan sekaligus kerugian dimana ketika terdapat node yang bergerak menuju arah node yang dituju oleh data maka data lebih cepat sampai ke tujuan. Namun sebaliknya ketika tidak terdapat node yang searah dengan node tujuan data maka data *expired* di *Road Side Unit* yang menyebabkan data harus di kirim ulang.

Protokol routing yang memanfaatkan persimpangan jalan juga telah diteliti (Ramachandran, L. et al 2013). *Shortest Path Based Traffic Aware Routing* (STAR) adalah sebuah protokol yang dikembangkan untuk memaksimalkan penggunaan persimpangan jalan dimana terdapat lampu lalulintas. STAR memanfaatkan kendaraan yang sedang berhenti di lampu merah untuk memastikan *end to end connectivity*. Untuk mengetahui hubungan antar kendaraan maka STAR mengun teknik *broadcast hello message*. Namun teknik ini juga dapat menyebabkan *broadcast storm*. Untuk mengatasi masalah tersebut, pada penelitian ini diusulkan untuk menggunakan teknik *Red Light First Forwarding* (RLFF). Ketika data sampai di persimpangan jalan, maka kendaraan yang sedang berhenti karena lampu merah yang meneruskan data tersebut sehingga mengurangi *broadcast storm*.

Konsep ini memiliki kelebihan yaitu pada penggunaan lampu lalu lintas sebagai *Road Side Unit* yang membantu meneruskan paket data. Namun karena lampu lalu lintas menyala bergantian maka terdapat kemungkinan untuk paket data *expired* ketika menunggu node yang memiliki jalur ke node tujuan.

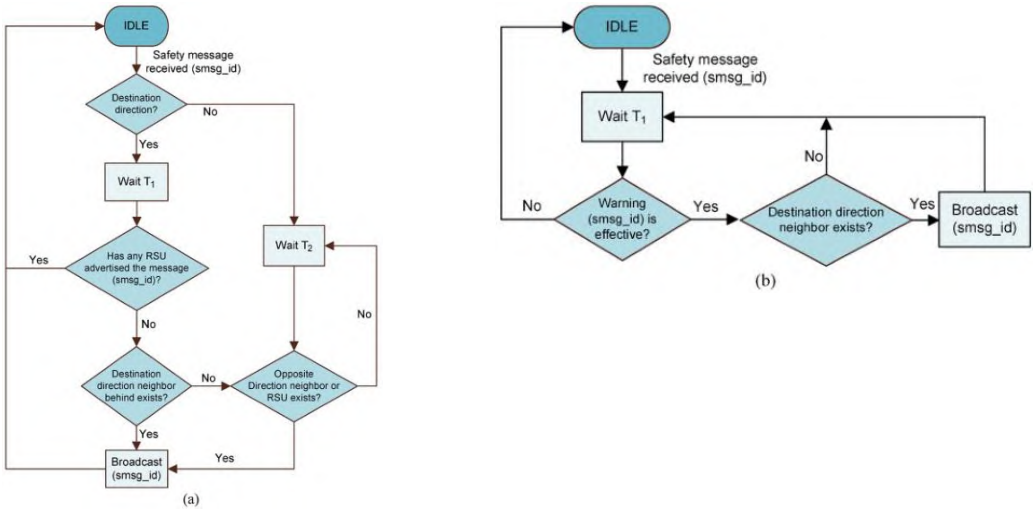
Penelitian lain tentang *Road Side Unit* sebagai *message router* yang dilakukan (Mershad, K. et al , 2012) mengusulkan sebuah mekanisme pengiriman data untuk kasus dimana antara *node* pengirim dan penerima terpisah dengan jarak yang cukup jauh sehingga pengiriman data antar *node* dianggap kurang efektif. ROAMER (*Roadside Unit As Message Routers in VANETs*) memanfaatkan *hybrid Road Side Unit* dimana sebagian *Road Side Unit* terhubung antara satu sama lain untuk membentuk *backbone* dan sebagian lainnya terhubung ke internet sebagai *gateway*. Dan kesemua *Road Side Unit* dapat saling terhubung membentuk jaringan *mesh*. Setiap *Road Side Unit* menerima *hello message* dari setiap kendaraan yang berada di dalam jangkauannya. Ketika salah satu kendaraan mengirimkan data ke kendaraan lainnya maka data dikirimkan ke *Road Side Unit* terdekat. *Pseudocode* algoritma ROAMER dapat dilihat pada Gambar 2.2 :

```
Vehicle-to-RSU Algorithm
Input: Packet P.
Source: vehicle S.
Destination: RSU R.

start
    Step1: if (R is within S range) {
               S sends P directly to R.
               Go to end.
           }
    else {
        Step2:  S defines total path (Pa) between itself and R.
        Step3:  S determines the set Sn of neighbors that are nearer from it to R.
        Step4:  if (Sn = ø) {
                    Go to Delay routine.
                }
                else {
                    S sends P to k neighbors in Sn.
                    S drops P.
                }
        }
end

Delay routine:
    After Delay ends {
        Go to Step1
    }
```

Gambar 2. 2 *Pseudocode* algoritma ROAMER (Mershad, K. et al , 2012)

Dari *pseudocode* pada Gambar 2.2 di atas dapat diketahui bahwa apabila kendaraan pengirim tidak berada dalam jangkauan *Road Side Unit*, maka data dikirimkan ke kendaraan lain sehingga sampai ke *Road Side Unit*. Setelah data berada di *Road Side Unit* maka data di kirimkan antar *Road Side Unit* sehingga data diterima oleh *Road Side Unit* yang memiliki akses ke kendaraan tujuan data. Dengan demikian dapat mengurangi jumlah *hop* yang harus di tempuh oleh data untuk sampai di tujuan.

Selain digunakan di persimpangan, penelitian terhadap penggunaan *Road Side Unit* di jalan bebas hambatan juga telah dilakukan (Sou, Sok-Ian et al, 2011). Pada penelitian ini, *Road Side Unit* digunakan sebagai *message broadcaster* ketika terjadi *emergency situation*. Ketika sebuah kendaraan mengalami *emergency situation*, maka kendaraan tersebut mengirimkan *emergency message* kepada *Road Side Unit* terdekat. Apabila sebuah *Road Side Unit* mendapatkan *emergency message* maka *Road Side Unit* tersebut melakukan *emergency message broadcast* kepada setiap kendaraan yang searah dengan kendaraan yang mengalami *emergency situation*. Hal ini dapat diketahui dari arah kendaraan yang berada dalam jangkauan *Road Side Unit*. Apabila kendaraan yang mengalami *emergency message* tidak berada di dalam jangkauan *Road Side Unit*, maka *emergency message* dikirimkan ke kendaraan lain yang berjalan berlawanan arah sampai menemukan *rehealing node*. *Rehealing node* dapat berupa *Road Side Unit* atau *node* lain yang menjadi *head of cluster*. Dengan demikian dapat mengurangi jumlah hop yang harus dilalui data. *Flowchart* untuk pengiriman *emergency message* dari kendaraan ke *Road Side Unit* dapat dilihat pada Gambar 2.3 berikut ini :



Gambar 2. 3 (a) Broadcast *emergency message* pada kendaraan (b) Broadcast *emergency message* pada *Road Side Unit* (Sou, Sok-Ian et al, 2011)

Dari Gambar 2.3 di atas dapat diketahui kendaraan yang menerima *broadcast emergency message* pertama melakukan pengecekan apakah *id* kendaraan yang berada dalam data tersebut berada didepan nya (searah). Jika benar maka kendaraan menunggu apakah ada data yang sama dari *Road Side Unit*. Jika tidak maka data dikirimkan ke kendaraan lain yang berada dalam jangkauannya. Jika tidak maka data dikirimkan ke *Road Side Unit*. Sedangkan pada sisi *Road Side Unit* ketika menerima data *emerency message* pertama melakukan pengecekan terlebih dahulu apakah pesan tersebut telah *expired* atau masih aktif. Apabila masih aktif maka berikutnya melakukan pengecekan apakah ada kendaraan yang dapat menerima data tersebut. Jika ada maka data dikirimkan ke kendaraan, jika tidak maka *Road Side Unit* menunggu selama beberapa saat sampai terdapat kendaraan yang dapat menerima data.

2.1 Vehicular Ad hoc Network (VANET)

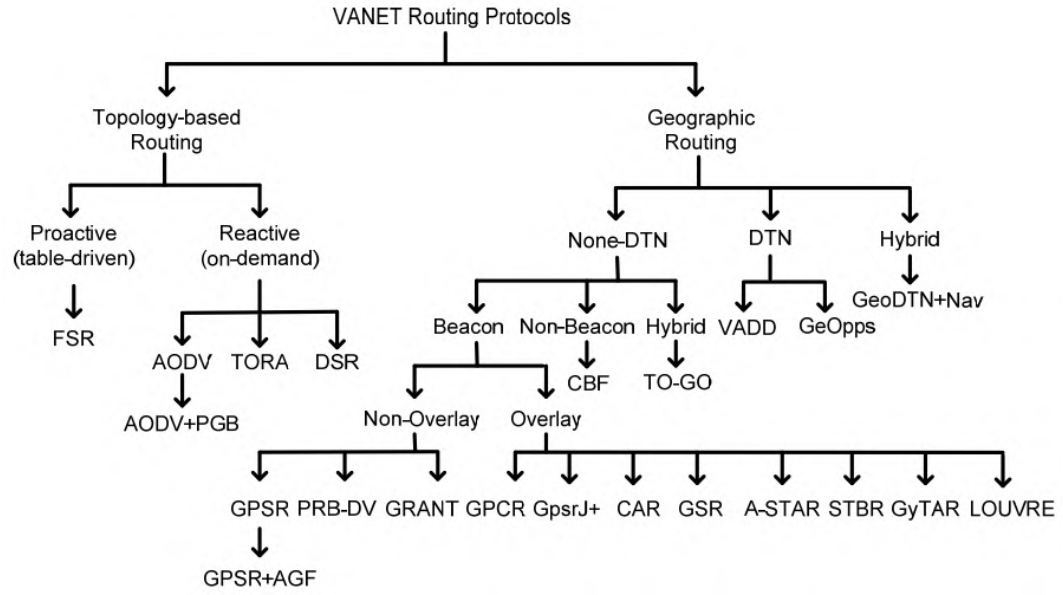
VANET (Vehicular Ad-hoc Network) merupakan pengembangan dari *MANET (Mobile Ad-hoc Network)* dimana implementasinya difokuskan pada *node* (*vehicle*). *VANET* membentuk jaringan *multi-hop* antar *node* yang dapat digunakan untuk mengirimkan data kepada *node* lain ataupun *static intersection node* (Nakamura et al, 2010). Karena *VANET* merupakan pengembangan dari *MANET* dimana model jaringan ini tidak memiliki infrastuktur maka dimungkinkan setiap *node* didalam jaringan ini untuk bergerak. Yang membed VANET dengan MANET adalah mobilitas pada VANET terbatas pada jalan (*road*) yang telah ditentukan dari peta (*map*). Mobilitas ini dapat mengakibatkan perubahan susunan *node* Hal ini yang menjadi salah satu permasalahan dalam pengiriman paket data di jaringan *VANET*.

Karena *VANET* digunakan pada kendaraan serta diasumsikan memiliki perangkat GPS, maka informasi kecepatan, jalan, arah hingga mobilitas setiap *node* dapat diprediksi / tidak acak. Hal ini dapat membantu mempermudah pengiriman paket data (Menouar et al, 2007). Namun karena tidak adanya infrastruktur maka pengiriman data harus dilakukan secara *multi-hop*. Untuk itu diperlukan mekanisme untuk menentukan jalur pengiriman data. Mekanisme tersebut terdapat pada protokol *routing*. Protokol *routing* adalah sebuah protokol yang menentukan pencarian jalur dari pengirim ke penerima sebelum terjadi pengiriman data. Karena topologi pada lingkungan VANET bersifat dinamis / selalu berubah maka tidak dapat menggunakan protokol *routing* pada jaringan dengan topologi statis yang ada pada saat ini.

2.2 Protokol Routing pada VANET

Karena pada VANET tidak memiliki topologi yang statis, maka data dikirimkan secara estafet oleh *node* diantara *node source* dan *node destination*. Dibutuhkan algoritma *routing* yang mampu mencari jalur pengiriman data pada topologi yang dinamis. Para peneliti telah banyak mengembangkan algoritma *routing* di VANET dengan berbagai kelebihannya. Menurut penelitian yang dilakukan oleh (K.C. Lee, et al 2010) terdapat beberapa pengelompokan protokol

routing yang dikembangkan pada lingkungan vanet seperti yang terlihat pada Gambar 2.4 berikut ini :



Gambar 2. 4 VANET *routing protocol* (K.C. Lee, et al 2010)

Secara umum protokol *routing* pada VANET yang masuk dalam kategori *topology based* terbagi menjadi 3 bagian yaitu proaktif, reaktif dan hybrid. Protokol *routing* proaktif selalu berusaha untuk mengetahui informasi *node* di sekitarnya dengan terus memperbarui tabelnya. Kelebihan dari teknik ini adalah setiap *node* dapat mengetahui jalur menuju setiap *node* sehingga ketika ada permintaan untuk mengirimkan data kepada sebuah *node* maka node pengirim sudah mengetahui jalurnya. Namun karena pada VANET setiap node bergerak yang memungkinkan terjadinya perubahan topologi maka dengan menggunakan topologi ini, setiap *node* terus memperbarui informasi di dalam tabelnya sesuai informasi dari *node* lain. Di sisi lain protokol *routing* yang termasuk kelompok reaktif hanya mengetahui *node* di sekitarnya saja, sehingga tidak dibutuhkan untuk melakukan pengecekan *node* yang terlalu banyak. *Node* pengirim mengirimkan paket *route discovery* untuk mengetahui jalur menuju *node* penerima. Sedangkan dalam kelompok *hybrid* memanfaatkan kelebihan dari kedua kelompok protokol *routing* yang sekaligus meminimalisir efek negatif dari kedua kelompok protokol routing.

2.3 AODV (*Ad hoc On Demand Distance Vector*)

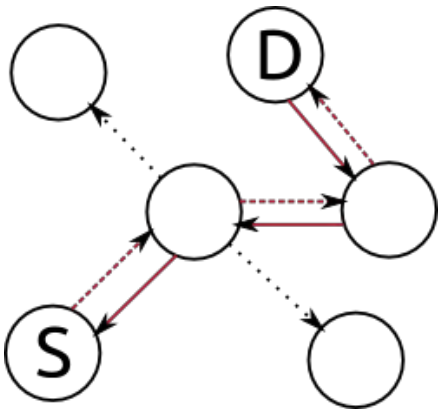
Adhoc On Demand Vector Routing adalah salah satu routing protokol reaktif di VANET. Pada bulan November 2001 *draft* protokol AODV pertama kali di publikasikan oleh peneliti pada kelompok kerja MANET (Mobile Adhoc Network) pada komunitas IEFT (*Internet Engineering Task Force*). Dan akhirnya pada bulan Juli 2003 *draft* protokol AODV telah di publikasikan melalui RFC3561 (C. Perkins, et al 2003). AODV termasuk dalam kelompok protokol *Distance Vector* dimana setiap node mengetahui node tetangganya beserta jarak nya. Secara umum tabel routing AODV dapat dilihat pada tabel 2.1 berikut ini :

Tabel 2. 1 *Distance Vector table*

Destination	**Cost**	**Next Hop**
A	1	A
B	0	B
C	∞	-
D	1	D
E	∞	-

Untuk mencari jalur menuju *node destination*, AODV menggunakan teknik *Route Request (RREQ), Route Reply (RREP)* dan *Route Error (RERR)*. *Route Request (RREQ)* digunakan ketika *node source* hendak mencari jalur untuk mengirimkan data kepada *node destination*. Sedangkan *Route Reply* digunakan untuk mengirimkan rute / jalur yang telah dibuat kembali kepada *node source* ketika *node destination* telah ditemukan. Hal ini terjadi karena pada protokol AODV, setiap *node* hanya mengetahui *node* tetangganya melalui mekanisme *Hello Message*. *Hello message* adalah sebuah paket RREP yang memiliki *Time To Live* (TTL) = 1 yang berarti hanya dapat diterima oleh node tetangganya. Selain untuk mengetahui node tetangganya, *Hello Message* juga dapat digunakan untuk mengidentifikasi kerus *link*. Hal ini dapat dilakukan dengan mengirimkan *Hello Message* secara berkala sehingga dapat diketahui ketika adanya *link* yang rusak.

Ketika terjadi *broken link*, sebuah node dapat mencoba untuk memperbaiki *link* ke node tersebut dengan cara mencari jalur lain menuju ke node tersebut dengan mengirimkan kembali RREQ. Apabila upaya ini gagal maka node mengirimkan RERR untuk menginformasikan kepada node lain bahwa terdapat node yang tidak dapat diakses. Maka ketika sebuah *node* mengirimkan data ke *node* yang lain, dikirimkan paket *Route Request (RREQ)* untuk mengetahui jalur menuju *node destination*. Setiap *node* yang mendapatkan paket *Route Request (RREQ)* memeriksa apakah *node destination* berada di daftar tetangganya. Apabila ada maka *node* tersebut mengirimkan paket *Route Reply (RREP)* baik kepada *node source* maupun *node destination*. Apabila *node destination* tidak terdapat di daftar tetangganya maka *node* tersebut menerukan paket *Route Request (RREQ)* ke *node* tetangganya. Begitu seterusnya sampai ditemukan *node* yang menjadi tetangga *node destination*. Secara singkat, cara kerja protokol AODV dapat dilihat dari Gambar 2.5 berikut ini :



Gambar 2. 5 *AODV* RREQ & RREP (www.des-testbed.net)

Karena *node source* tidak memiliki informasi jalur menuju *node destination* (AODV hanya mengetahui node tetangganya saja melalui *Hello Message*) maka paket RREQ dikirimkan ke *node* tetangganya.

Pada jaringan VANET kemungkinan terjadinya *link breakage* cukup tinggi dikaren mobilitas masing – masing *node*. Ketika sebuah *node* mengetahui bahwa *node* tetangganya sudah tidak dapat diakses (keluar dari jangkauan) maka *node*

tersebut mengirimkan RERR (*Route Error*) yang berisi daftar *node* yang tidak dapat diakses. *Node* lain yang menerima paket RERR memeriksa di tabelnya apakah terdapat *node* yang tidak dapat diakses. Jika benar maka tabel di update sesuai dengan informasi terbaru kemudian menyebarkan kembali paket RERR. Paket RREQ berisi informasi *Src. Node, Dst. Node, Lifespan* dan ID. *Node* tetangga yang menerima paket ini memeriksa apakah *Dst. Node* yang dituju terdapat di daftar tetangganya. Apabila tidak ada maka paket RREQ ini dikirimkan kembali ke *node* tetangganya. Apabila *Dst. Node* terdapat di daftar tetangganya maka node mengirimkan paket RREP yang berisi *Dst. Node, Src. Node, Hop Count* dan *ID*.

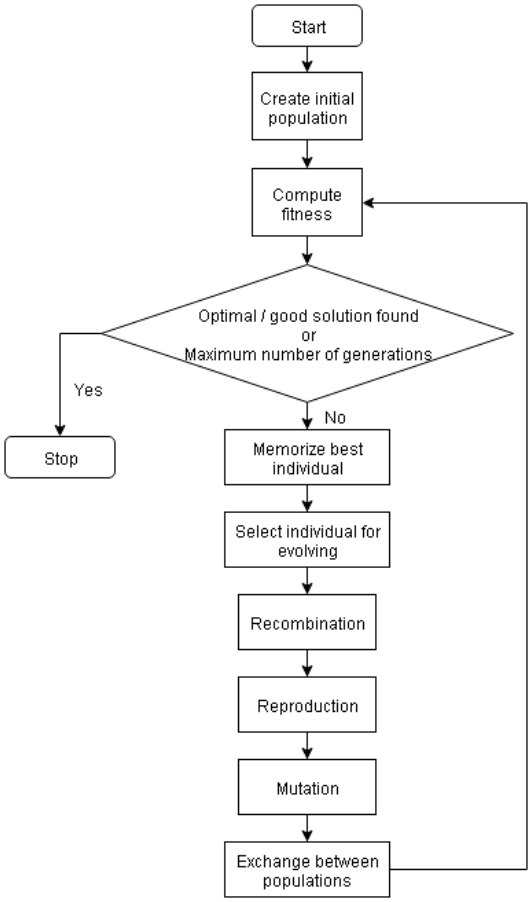
2.3 *Static Intersection Node*

Selain *Vehicle to Vehicle Communication,* VANET juga mendukung untuk melakukan *Vehicle to Infrastructure Communication (V2I)*. Infrastruktur dapat berupa *Access Point* atau infrastruktur jaringan lain misal 3G atau HSDPA yang disebut dengan *RSU (Road Side Unit)* (Calvacante, et al 2012). Pada beberapa penelitian menyebut infrastruktur sebagai *Road Side Unit*. Salah satu permasalahan utama pada VANET adalah topologi yang dinamis sehingga keberhasilan pengiriman data sangat bergantung kepada posisi *node*. Untuk meningkatkan performa protokol *routing*, salah satu cara yang dapat dilakukan adalah dengan menambahkan *Road Side Unit.* Keberadaan *Road Side Unit* dapat membantu *node* untuk mengirimkan paket data baik ke *node* yang berada di sekitarnya maupun ke *Road Side Unit* lainnya. Komunikasi yang terjadi antar *node* maupun *node* dengan *Road Side Unit* disebut dengan komunikasi *hybrid*. Pengembangan dari *Road Side Unit* adalah *Static Intersection Node*. *Static Intersection Node* merupakan *Road Side Unit* yang berada pada *intersection*.

2.3.1 Penentuan Lokasi Static Intersection Node

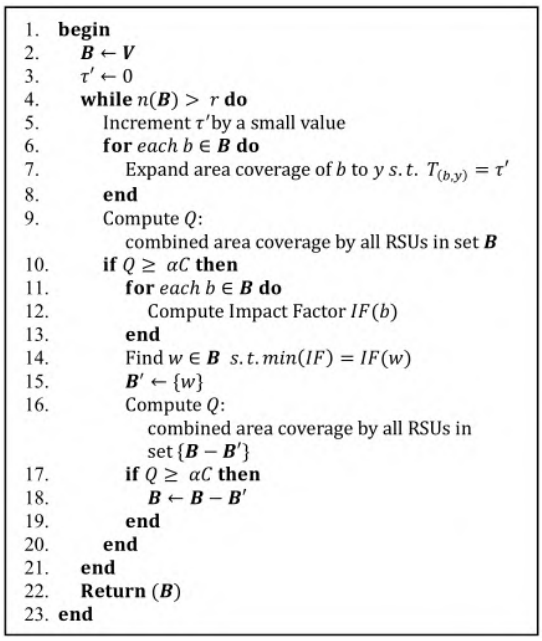
Pada kondisi nyata dimana banyak terdapat persimpangan jalan tentunya kurang efektif apabila meletakkan setiap *Static Intersection Node* di setiap persimpangan jalan. Maka harus dilakukan pemilihan terhadap lokasi *Static Intersection Node* yang paling optimal. Beberapa penelitian telah dilakukan dalam hal ini salah satu diantaranya adalah penelitian (Christian Lochert et al, 2008). Pada

penelitian ini menggunakan algoritma genetika untuk menentukan lokasi *Supporting Unit* (*Road Side Unit*). Langkah – langkah yang dilakukan pada penelitian ini dapat dilihat pada Gambar 2.6 berikut ini :



Gambar 2. 6 *Flowchart* penentuan posisi SU (Christian Lochert et al, 2008)

Pada penelitian lain (Baber Aslam, et al 2012) untuk mementukan posisi *Road Side Unit* yang terbaik digunakan algoritma *Binary Integer Programming* (BIP) yang pada dasarnya menggunakan teknik *branch and bound* serta algoritma *Balloon Expansion Heuristics* (BEH). Dengan menggunakan algoritma BIP & BEH para peneliti dapat merekomendasikan posisi *Road Side Unit* yang terbaik berdasarkan *pseudocode* seperti pada Gambar 2.7 berikut ini :



```
1.  begin
2.    B ← V
3.    τ′ ← 0
4.    while n(B) > r do
5.      Increment τ′by a small value
6.      for each b ∈ B do
7.        Expand area coverage of b to y s.t. T(b,y) = τ′
8.      end
9.      Compute Q:
          combined area coverage by all RSUs in set B
10.     if Q ≥ αC then
11.       for each b ∈ B do
12.         Compute Impact Factor IF(b)
13.       end
14.       Find w ∈ B s.t. min(IF) = IF(w)
15.       B′ ← {w}
16.       Compute Q:
            combined area coverage by all RSUs in
            set {B − B′}
17.       if Q ≥ αC then
18.         B ← B − B′
19.       end
20.     end
21.   end
22.   Return (B)
23. end
```

Gambar 2. 7 *Psedocode Ballon Expansion Heuristics* (Baber Aslam, et al 2012)

Sedangkan pada penelitian (Jeonghee Chi, et al 2013) penentuan posisi *Road Side Unit* melalui beberapa kriteria seperti *intersection priority* dan *intersection connectivity*. Kemudian diukur *Intersection priority* pada masing – masing *intersection*. *Intersection priority* dapat di hitung dari kepadatan kendaraan serta lokasi persimpangan secara geografis. Sedangkan *intersection connectivity* dapat dihitung berdasarkan jumlah kendaraan yang melalui kedua persimpangan. Pada penelitian ini *Road Side Unit* diletakkan di setiap *intersection* berdasarkan informasi *intersection priority*. Informasi *intersection priority* didapatkan dari kondisi nyata di lapangan dengan mempertimbangkan lokasi sebuah *intersection*. Apabila sebuah intersection berada di daerah yang padat seperti pasar atau pusat kota maka diasumsikan banyak kendaraan yang melalui jalan tersebut atau dapat dikat padat. Kemudian jumlah *Road Side Unit* dikurangi berdasarkan kriteria lain seperti *intersection connectivity* sampai ditemukan *Road Side Unit* yang benar – benar optimal.

Untuk menentukan lokasi optimal Static Intersection Node pada penelitian ini akan menggunakan algoritma Dijkstra dimana menurut penelitian (Santoso, et al 2010) algoritma Dijkstra merupakanakan algoritma yang paling cocok untuk menentukan jalur terpendek pada peta. Penelitian ini membandingkan algortima

Disjktra, A* (A Star) serta Ant Colony untuk menentukan jalur optimal pada studi kasus peta kota Surabaya. Hasil dari penelitian menunjukkan bahwa algoritma Dijkstra paling cocok digunakan untuk pencarian rute optimal pada peta.

Sedangkan pada penelitian ini algoritma Dijkstra akan digunakan untuk mencari jalur dari pengirim ke penerima. Nilai *weight* yang digunakan diambil dari nilai kepadatan kendaraan pada setiap ruas jalan. Karena pada lingkungan VANET paket data akan diteruskan oleh kendaraan, maka *Static Intersection Node* akan dibutuhkan pada ruas jalan dengan kepadatan kendaraan yang rendah. Dengan demikian dapat meningkatkan performa pengiriman data.

[Halaman ini sengaja di kosongkan]

BAB III
METODOLOGI PENELITIAN

Langkah-langkah penelitian yang dilakukan pada penelitian ini adalah sebagai berikut:

1. Perumusan Masalah
2. Studi Literatur
3. Desain Sistem
4. Desain Algoritma dan Implementasi
5. Pengujian dan Evaluasi
6. Analisis Hasil
7. Penyusunan Buku Tesis

3.1 Perumusan Masalah

Masalah yang diangkat pada penelitian ini adalah bagaimana meningkatkan performa protokol AODV untuk pengiriman data antar *node* pada lingkungan VANET. Setiap *node / vehicle* pada lingkungan VANET memiliki perangkat nirkabel sehingga dimungkinkan untuk saling menerima & mengirimkan data meskipun tidak memiliki koneksi secara langsung. Karena setiap *node* bergerak, maka sangat dimungkinkan terjadi perubahan susunan *node* yang dilalui oleh data. Ketika jalur data terputus maka *node* pengirim menghentikan transmisinya sampai menemukan jalur baru. Kecepatan menemukan jalur bergantung kepada beberapa faktor diantaranya kepadatan *node*, jarak transmisi nirkabel setiap *node*, arah mobilitas *node* dan lain sebagainya. Dari permasalahan tersebut maka pada penelitian ini digunakan *Static Intersection Node* untuk meningkatkan performa protokol AODV dalam melakukan pencarian jalur pengiriman data sebelum dilakukan proses pengiriman data. *Static Intersection Node* adalah *Road Side Unit* yang berada di persimpangan jalan. Dengan adanya *Static Intersection Node* diharapkan dapat membantu mempercepat proses *route discovery* serta meningkatkan rasio keberhasilan pengiriman data.

Karena pada kondisi nyata cukup banyak persimpangan, akan kurang efektif apabila meletakkan *Static Intersection Node* di setiap persimpangan. Maka perlu di tentukan posisi *Static Intersection Node* yang optimal. Permasalahan berikutnya yaitu bagaimana menentukan posisi *Static Intersection Node* yang optimal. Pada penelitian ini diusulkan menggunakan algoritma Dijkstra untuk mencari jalur yang memiliki tingkat kepadatan paling rendah antara pengirim dan penerima. Kemudian *Static Intersection Node* di pasang pada persimpangan di jalur tersebut. Dengan demikian diharapkan dapat meminimalisir jumlah *Static Intersectin Node* yang dipasang namun tetap mengoptimalkan *Packet Delivery Ratio*.

3.2 Studi Literatur

Pada tahap studi literatur, dikaji berbagai referensi yang berkaitan dengan metode optimasi protokol AODV yang sudah diteliti. Selain itu dikaji pula hal-hal yang mendukung penelitian ini seperti peletakan posisi *Static Intersection Node* sehingga optimal serta implementasi pada simulator.

3.3 Desain Sistem

Dalam penelitian yang dilakukan ini digunakan beberapa tahapan antara lain sebagai berikut :

a. Penentuan lingkungan uji coba

   Penelitian ini dilakukan dengan menggunakan beberapa simulator. Adapun simulator yang digunakan adalah SUMO (*Simulation for Urban Mobility*) untuk membuat simulasi mobilitas node dan NS 2 (*Network Simulator* 2) untuk mensimulasikan pengiriman data.

b. Penentuan parameter uji coba

   Parameter uji coba adalah batasan – batasan yang digunakan pada penelitian ini sehingga fokus serta faktor – faktor apa saja yang digunakan sebagai acuan dalam penelitian ini.

c. Pemilihan skenario uji coba

   Agar menghasilkan output yang valid, maka pada penelitian ini dilakukan beberapa skenario uji coba sehingga semua data yang dibutuhkan telah didapatkan.

d. Pengukuran hasil uji coba

   Pengukuran hasil uji coba dilakukan untuk mengukur data yang dihasilkan dari skenario uji coba yang telah dilakukan.

e. Analisis hasil uji coba

   Analisis hasil uji coba bertujuan untuk menelaah hasil yang telah diperoleh dari proses pengujian apakah data telah sesuai dengan hipotesis awal ataukah masih belum.

f. Evaluasi

   Evaliasi dilakukan untuk me-*review* hasil pengujian, kelemahan – kelemahan yang ditemukan serta saran dan masukan untuk penelitian selanjutnya.

3.4 Desain Algoritma dan Implementasi

Desain algoritma pada penelitian ini dimulai dengan melakukan perancangan lingkungan penelitian. Adapun langkah – langkah yang dilakukan adalah sebagai berikut :

1. Membangun peta jalan yang dilalui kendaraan
   a. Menentukan peta yang digunakan (diutamakan yang memiliki banyak persimpangan)
   b. Mengunduh peta dari Open Street Map
   c. Menyimpan peta ke dalam format .osm untuk diproses lebih lanjut
2. Membuat mobilitas kendaraan
   a. Mengubah format peta sehingga kompatibel dengan simulator SUMO
   b. Membuat mobilitas kendaraan dengan jumlah kendaraan 25, 50, 75 dan 100 kendaraan

3. Merancang pengiriman data
    a. Menentukan variabel data yang di kirimkan diantaranya sebagai berikut :
        - Jenis paket data
        - Ukuran paket data
        - Jumlah paket data
        - Node pengirim dan node penerima
        - Waktu pengiriman
    b. Menentukan variabel media pengiriman data
        - Radius transmisi
        - Protokol MAC
        - Tipe kanal
        - Protokol routing

4. Percobaan pengiriman data pada peta 1
    a. Percobaan pengiriman data pada peta 1 tanpa *Static Intersection Node*
        - Pengiriman data dilakukan pada peta 1 dengan menggunakan 4 mobilitas kendaraan yang berbeda.
    b. Percobaan pengiriman data pada peta 1 dengan *Static Intersection Node* pada setiap *intersection*
        - Mendaftar koordinat setiap *intersection*
        - Meletakkan *Static Intersection Node* pada setiap *intersection*
        - Melakukan percobaan pengiriman data pada peta 1 dengan menggunakan 4 mobilitas kendaraan yang berbeda.
    c. Percobaan pengiriman data pada peta 1 dengan *Static Intersection Node* pada persimpangan jalan tertentu
        - Menggunakan algoritma Dijkstra untuk menentukan jalur dari pengirim ke penerima
        - Variabel *weight* didapatkan dari tingkat kepadatan kendaraan
        - Algoritma Dijkstra di gunakan untuk mencari jalur dari *node* pengirim ke *node* penerima yang paling rendah kepadatan kendaraannya.
        - Dilakukan percobaan pengiriman data dengan menggunakan 4 mobilitas kendaraan yang berbeda.

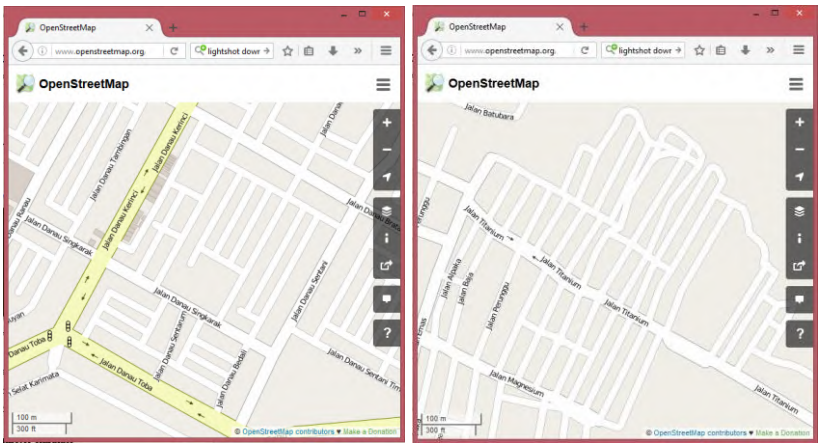
5. Percobaan pengiriman data pada peta 2
    a. Percobaan pengiriman data pada peta 2 tanpa *Static Intersection Node*
        - Pengiriman data dilakukan pada peta 2 dengan menggunakan 4 mobilitas kendaraan yang berbeda.
    b. Percobaan pengiriman data pada peta 2 dengan *Static Intersection Node* pada setiap *intersection*
        - Mendaftar koordinat setiap *intersection*
        - Meletakkan *Static Intersection Node* pada setiap *intersection*
        - Melakukan percobaan pengiriman data pada peta 2 dengan menggunakan 4 mobilitas kendaraan yang berbeda.
    c. Percobaan pengiriman data pada peta 2 dengan *Static Intersection Node* pada persimpangan jalan tertentu
        - Menggunakan algoritma Dijkstra untuk menentukan jalur dari pengirim ke penerima
        - Variabel *weight* didapatkan dari tingkat kepadatan kendaraan
        - Algoritma Dijkstra di gunakan untuk mencari jalur dari *node* pengirim ke *node* penerima yang paling rendah kepadatan kendaraannya.
        - Dilakukan percobaan pengiriman data dengan menggunakan 4 mobilitas kendaraan yang berbeda.

6. Analisa hasil pengiriman data pada peta 1 dan peta 2

    Setelah dilakukan simulasi pengiriman data pada peta 1 sesuai skenario percobaan diatas, maka langkah berikutnya adalah menganalisa hasil dari simulasi tersebut. Adapun parameter yang dijadian acuan adalah *Packet Delivery Ratio, Packet Loss* serta *End to End Delay* dengan menggunakan script AWK. *Packet Delivery Ratio* merepresentasikan jumlah data yan berhasil diterima oleh node tujuan, *Packet Loss* merepresentasikan jumlah data yang gagal sampai ke tujuan serta *End to End Delay* merepresentasikan waktu yang dibutuhkan paket data untuk sampai ke tujuan.Hasil dari percobaan tersebut yang akan menunjukkan efektifitas penggunaan *Static Intersection Node* pada peta sesuai dengan tujuan penelitian.

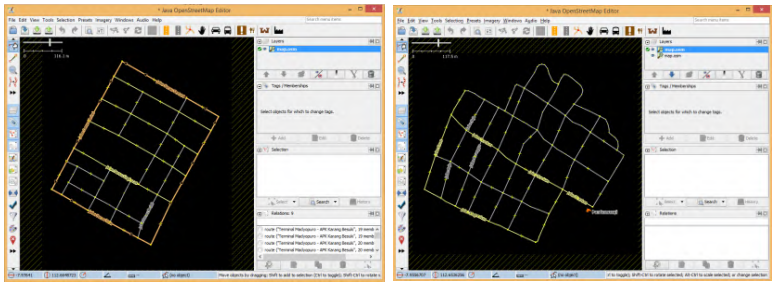
3.4.1 Membangun Lingkungan Pengujian

Langkah pertama yang harus dilakukan sebelum melakukan pengujian adalah membangun lingkungan untuk pengujian. Pada penelitian ini digunakan peta nyata kota Malang yang di unduh dari website http://openstreetmap.org seperti yang di tunjukkan pada Gambar 3.1 berikut ini :



Gambar 3. 1 Peta nyata pada website http://openstreetmap.org

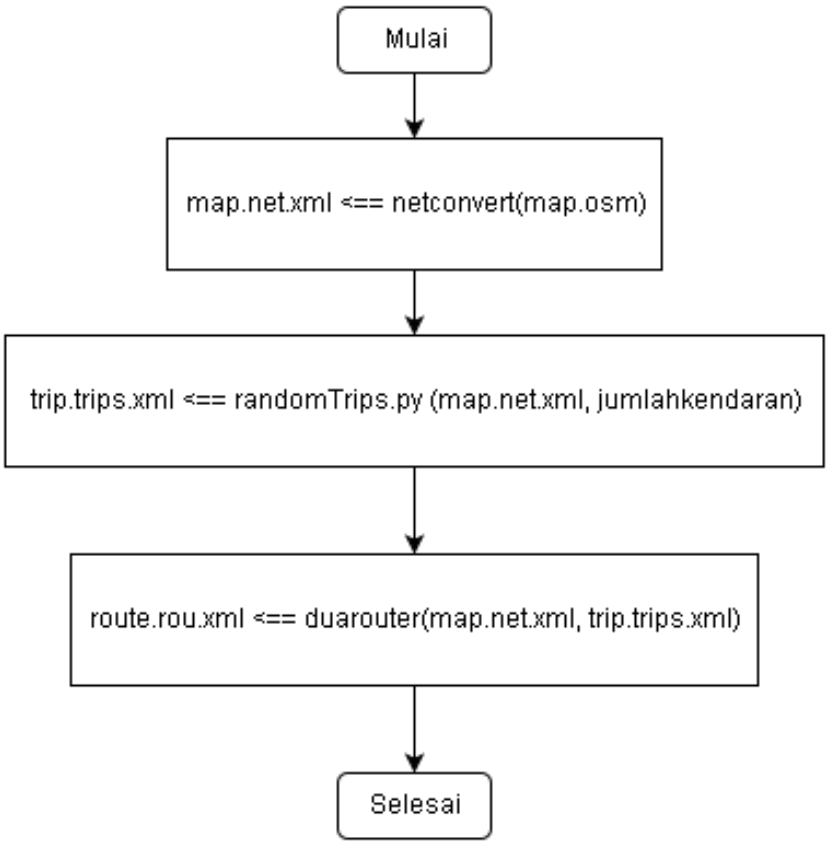
Setelah lokasi peta ditentukan, langkah berikutnya adalah mengunduh peta tersebut menggunakan aplikasi Java Open Street Map seperti yang tidunjukkan pada Gambar 3.2 berikut ini. Dengan menggunakan aplikasi Java Open Street Map peta di konversi ke dalam format .osm (open street map). Selain mengkonversi format, aplikasi Java Open Street Map ini juga dapat digunakan untuk mengubah parameter jalan misal jenis jalan, nama jalan, kecepatan maksimal, arah jalan dan lain sebagainya.



Gambar 3. 2 Peta 1 & Peta 2 pada editor JOSM

3.4.2 Mobilitas Kendaraan

Setelah menyimpan peta ke dalam format .osm maka berikutnya adalah membuat mobilitas kendaraan pada peta tersebut dengan menggunakan program SUMO (*Simulation for Urban Mobility*). Adapun langkah – langkah untuk membuat mobilitas kendaraan ditunjukkan pada Gambar 3.3 berikut ini :



Gambar 3. 3 Langkah – langkah pembuatan mobilitas kendaraan

Berikut penjelasan dari langkah – langkah tersebut :

1. Mengubah format peta dari .osm menjadi .net.xml dengan menggunakan program *netconvert*
2. Membuat mobilitas kendaraan secara acak pada peta menggunakan program *randomTrips.py*

3. Menggabungkan mobilitas kendaraan ke dalam peta menggunakan program *duarouter*
4. Membuat skenario mobilitas menggunakan program SUMO (hanya diperlukan apabila hasil mobilitas hendak ditampilkan secara visual pada simulator SUMO)
5. Mengubah skenario mobilitas ke dalam format Network Simulator 2 menggunakan program *traceExporter* (diperlukan agar mobilitas kendaraan dapat dibaca pada format Network Simulator 2).

Proses di atas menghasilkan mobilitas kendaraan secara acak pada peta yang telah ditentukan. Agar hasil percobaan dapat valid, maka dibuat 4 mobilitas kendaraan secara acak dengan jumlah kendaraan yang berbeda (25, 50, 75 dan 100). Selanjutnya mobilitas tersebut digunakan untuk percobaan pengiriman data pada *Network Simulator*.

3.4.3 Pengiriman Data

Setelah mobilitas kendaraan dibuat, maka berikutnya dibuat pengiriman data yang dilakukan di *Network Simulator* 2. Karena topik pada penelitian ini adalah VANET maka pengiriman data yang dilakukan dengan memanfaatkan mobilitas kendaraan yang telah dibuat sebelumnya. Terdapat 4 mobilitas kendaran yang berbeda pada masing – masing peta dengan arah serta jumlah kendaraan yan berbeda – beda (25, 50, 75 dan 100 kendaraan). Data yang dikirimkan berupa UDP (*User Datagram* Protocol) melalui paket CBR (Constant Bit Rate) yang dikirimkan 1 paket per detik. Waktu simulasi di tentukan 100ms. Sehingga dapat diukur berapa paket data yang sampai ke tujuan maupun berapa paket data yang gagal diterima oleh *node* tujuan serta rata – rata waktu yang dibutuhkan data sampai ke *node* tujuan. Adapun beberapa parameter yang digunakan pada *Network Simulator* 2 antara lain sebagai berikut :

Tabel 3. 1 Parameter-parameter simulasi

No.	Parameter	Spesifikasi
1	Network simulator	NS-2.34
2	Routing Protocol	AODV
3	Waktu simulasi	100 detik
4	Area simulasi	350m x 350m 700m x 700m
5	Banyak kendaraan	25, 50, 75, 100
6	Radius transmisi	100 m
7	Kecepatan maksimal	8 m/s (30 km/j)
8	Tipe data	UDP (CBR)
9	Jumlah data	1 paket / detik
10	Protokol MAC	IEEE 802.11
11	Peta	Peta nyata
12	Tipe kanal	*Wireless channel*

3.4.4 Pemilihan posisi *Static Intersection Node*

Karena *Static Intersection Node* merupakan *Road Side Unit* yang berada di persimpangan jalan, maka posisi *Static Intersecton Node* terikat pada posisi persimpangan jalan (tidak dapat diletakkan secara bebas seperti *Road Side Unit*). Salah satu skenario pada penelitian ini adalah melakukan pengiriman data pada peta dimana terdapat *Static Intersection Node*. Namun karena meletakkan *Static Intersection Node* pada setiap persimpangan jalan menjadi kurang efektif baik dari segi fungsi maupun biaya, maka perlu dilakukan penelitian untuk menentukan posisi *Static Intersection Node* yang optimal.

Pada penelitian ini menggunakan kepadatan kendaraan sebagai salah satu parameter penentu posisi *Static Intersection Node.* Algoritma yang digunakan adalah algoritma Dijkstra untuk menentukan jalur dari *node* pengirim ke *node* penerima. Dimana jalur yang dipilih adalah jalur dengan tingkat kepadatan rendah sehingga diperlukan adanya *Static Intersection Node* pada setiap persimpangannya. Dipilihnya jalur dengan tingkat kepadatan rendah dengan asumsi jalur dengan

tingkat kepadatan tinggi tidak memerlukan *Static Intersection Node* untuk dapat mengirimkan data, melainkan dapat dilakukan oleh kendaraan yang melalui di jalan tersebut.

Algoritma Dijkstra dipilih karena selain prosesnya yang sederhana sehingga tidak membutuhkan sumberdaya komputasi yang besar juga merupakan salah satu algoritma yang tepat untuk kasus *path finding*. Hal ini membuat algoritma Dijkstra cocok digunakan pada lingkungan VANET yang memiliki sumberdaya komputasi terbatas. Adapun algoritma Dijkstra yang digunakan pada penelitian ini adalah untuk mencari jalur dari *node* pengirim ke *node* penerima berdasarkan pada tingkat kepadatan pada setiap ruas jalan. Secara umum langkah - langkah untuk menentukan jalur dari *node* pengirim ke *node* penerima pada penelitian ini adalah sebagai berikut :

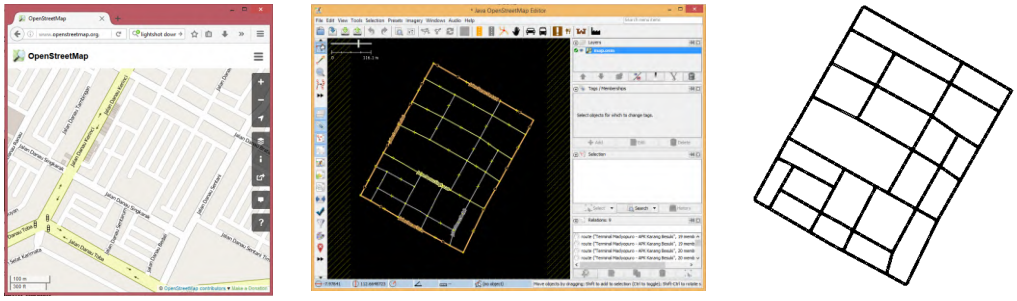
1. Daftar kepadatan kendaraan di setiap ruas jalan pada peta.
2. Daftar setiap *intersection* yang saling terhubung.
3. Temukan jalur dari sumber ke tujuan melalui jalan yang tingkat kepadatannya paling rendah menggunakan algoritma Dijkstra

3.5 Pengujian dan Evaluasi

Penelitian ini melakukan pengiriman data dari sebuah *node* yang berada pada salah satu sisi peta kepada *node* lain yang berada pada sisi peta yang lain tanpa adanya hubungan secara langsung antara kedua node tersebut. Data yang dikirim adalah data CBR (*Constant Bit Rate*) yang menggunakan protokol UDP (*User Datagram Protocol*). Sedangkan protokol *routing* yang digunakan adalah AODV (*Adhoc On demand Distance Vector*). Untuk meningkatkan performa protokol AODV dalam melakukan pengiriman data maka pada penelitian ini ditambahkan *Static Intersection Node* yang di letakkan di persimpangan jalan. Diharapkan potokol AODV dapat memanfaatkan *Static Intersection Node* untuk meningkatkan *Packet Delivery Ratio* serta mengurangi *Packet Loss* serta *End to End delay*.

Berikut dijelaskan tentang skenario pengujian yang dilakukan pada penelitian ini :

3.5.1 Peta 1



(a) (b) (c)

Gambar 3. 4 Pemrosesan peta 1

Pada Gambar 3.4 di atas dapat diketahui tahapan pemrosesan peta dari mulai memilih peta yang digunakan dari Open Street Map (Gambar 3.4.a) kemudian dilanjutkan dengan melakukan penyesuaian peta pada aplikasi Java Open Street Map (Gambar 3.4.b) sampai hasil akhir peta setelah di proses oleh SUMO (Gambar 3.4.c)

Berikut informasi teknis peta tersebut :

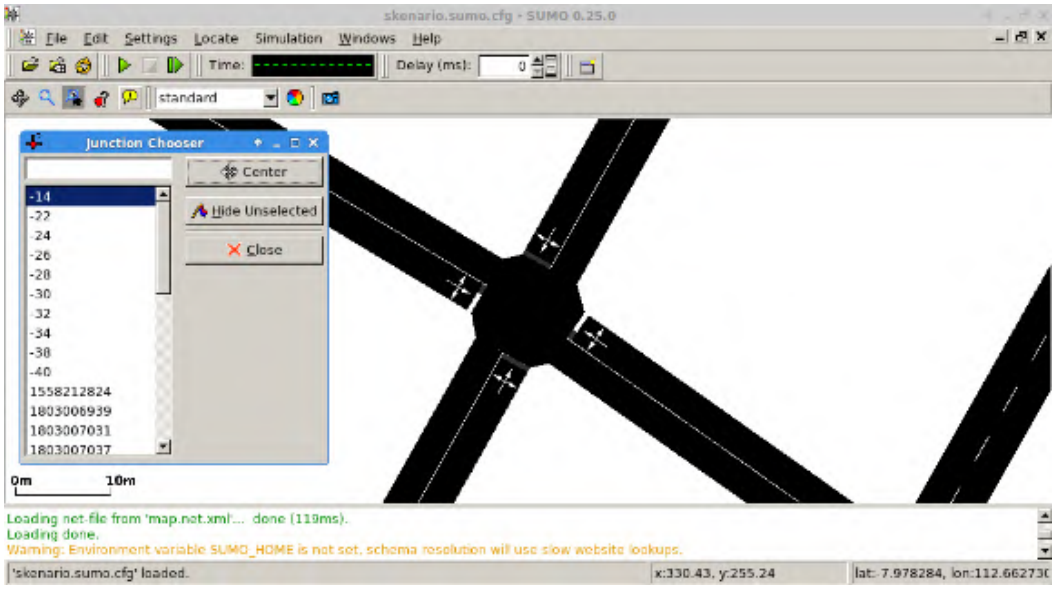
- Peta 1
- Luas area 350m x 350m
- Jumlah persimpangan jalan 31
- Jumlah ruas jalan 54

Pada peta di atas dilakukan percobaan pengiriman data dengan menggunakan skenario sebagai berikut :

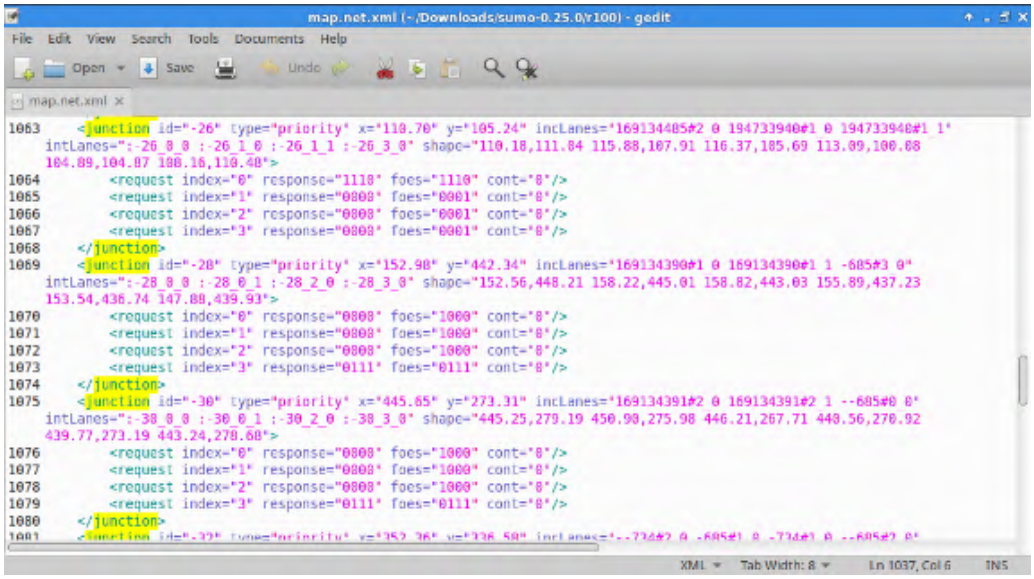
- Skenario 1 Tanpa *Static Intersection Node*
- Skenario 2 Dengan *Static Intersection Node* pada setiap persimpangan
- Skenario 3 Dengan *Static Intersection Node* pada persimpangan tertentu

Untuk mendapatkan data yang valid maka pada setiap skenario diatas akan dilakukan percobaan dengan 4 mobilitas kendaraan yang berbeda yang telah disiapkan sebelumnya.

Agar dapat meletakkan *Static Intersection Node* untuk percobaan berikutnya, maka perlu diketahui terlebih dahulu koordinat masing – masing *intersection* yang ada pada peta. Cara untuk mengetahui koordinat setiap *intersection* yang ada pada peta, dapat menggunakan fitur *Junction* pada SUMO seperti yang terlihat pada Gambar 3.5 atau langsung melihat dari isi file map.net.xml seperti yang terlihat pada Gambar 3.6 berikut ini :

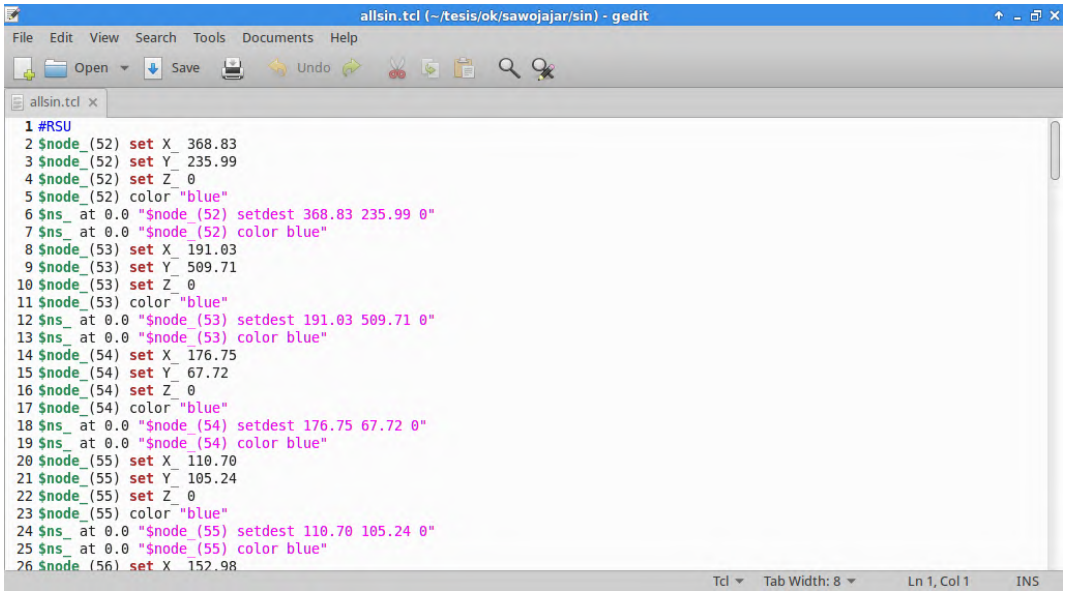


Gambar 3. 5 Koordinat *intersections* pada peta



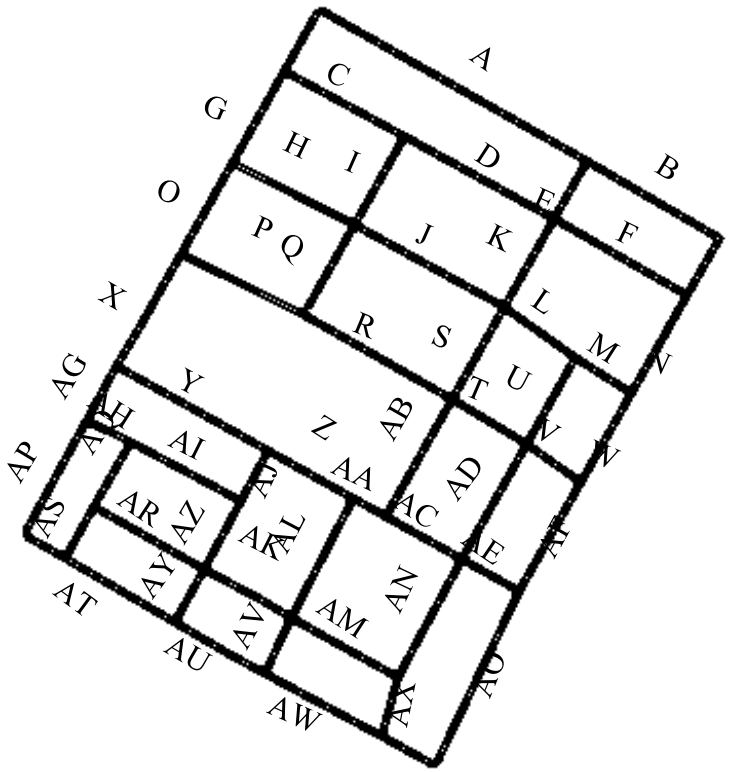
Gambar 3. 6 Koordinat *intersections* pada file map.net.xml

Setelah di ketahui koordinat masing – masing *intersection* pada peta, selanjutnya meletakkan *Static Intersection Node* pada koordinat tersebut. Untuk memudahkan proses maka pada penelitian ini *Static Intersection Node* diletakkan pada file tersendiri yang dinamakan sin.tcl seperti yang tampak pada Gambar 3.7 berikut ini



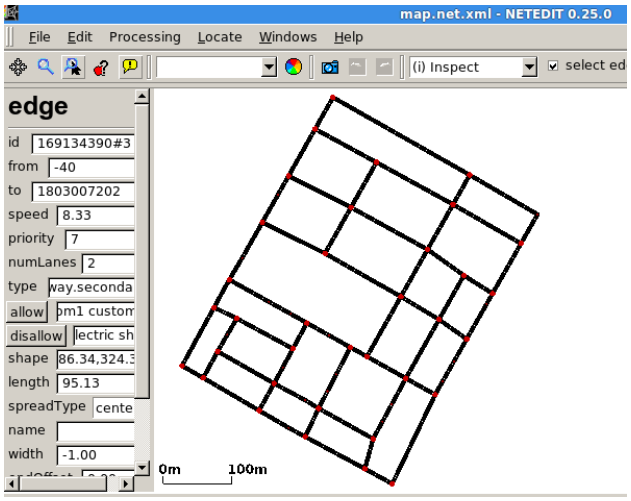
Gambar 3. 7 Koodinat *Static Intersection Node*

Setelah mengetahui posisi setiap *Static Intersection Node* pada peta maka berikutnya dilakukan pengurangan jumlah *Static Intersection Node* namun dengan tetap memaksimalkan *Packet Delivery Ratio*. Pada penelitian ini diusulkan menggunakan algoritma Dijkstra untuk mengurangi jumlah *node* pada peta. Algoritma Dijkstra digunakan untuk menentukan jalur dengan tingkat kepadatan yang paling rendah antara *node* pengirim dan *node* penerima. Dengan asumsi jalan yang membutuhkan *Static Intersection Node* adalah jalan dengan kepadatan kendaraan yang rendah karena pada jalan dengan kepadatan kendaraan yang tinggi data dapat dikirimkan antar kendaraan. Sehingga kurang efektif apabila meletakkan *Static Intersection Node* pada jalan yang tingkat kepadatannya tinggi. Maka diperlukan untuk mengetahui tingkat kepadatan kendaraan pada setiap ruas jalan yang ada pada peta. Untuk memudahkan pengitungan tingkat kepadatan di masing – masing ruas jalan, maka diberikan nama pada setiap ruas jalan seperti pada Gambar 3.8 berikut :



Gambar 3. 8 Penamaan ruas jalan pada peta 1

Setelah diberikan penamaan pada setiap ruas jalan, maka langkah selanjutnya adalah mengukur panjang setiap ruas jalan yang ada pada peta. Pengukuran dapat dilakukan melalui program Net Edit pada SUMO seperti yang tampak pada Gambar 3.9 berikut ini :



Gambar 3. 9 Tampilan peta 1 pada editor Net Edit

Berikut tabel yang beisi nama beserta panjang ruas jalan pada peta 1

Tabel 3. 2 Panjang setiap ruas jalan pada peta 1

Road	Length (m)	Road	Length (m)	Road	Length (m)
A	225.69	S	77.76	AK	73.25
B	122.89	T	63.88	AL	98.87
C	100.22	U	72.69	AM	91.35
D	125.45	V	48.61	AN	98.73
E	48.69	W	75.17	AO	142.09
F	112.50	X	95.13	AP	95.57
G	77.37	Y	127.54	AQ	54.69
H	100.33	Z	71.49	AR	93.09
I	74.35	AA	27.30	AS	43.29
J	125.39	AB	98.89	AT	93.15
K	74.35	AC	64.05	AU	75.96
L	64.09	AD	96.90	AV	45.03

M	48.67	AE	48.14	AW	97.30
N	82.71	AF	91.80	AX	44.75
O	76.61	AG	45.50	AY	44.39
P	100.39	AH	36.15	AZ	56.88
Q	75.92	AI	91.66		
R	125.34	AJ	41.98		

Setelah mengetahui panjang setiap ruas jalan pada peta maka langkah berikutnya adalah menentukan jalur dari *node* pengirim ke *node* penerima melalui jalur dengan kepadatan yang paling rendah. Hal ini dilakukan dengan memasukkan informasi kepadatan setiap ruas jalan di atas serta daftar *intersection* yang saling terhubung ke dalam program Dijkstra seperti pada Gambar 3.10 berikut ini :

```
# add to evaluated
while i_can_do_it:
    for node in EDGES[CURRENT]:
        if node in OPENNED:
            distance = EDGES[CURRENT][node]
            if (node not in EVALUATED) or (EVALUATED[node][0] > distance):
                EVALUATED[node] = (EVALUATED[CURRENT][0]+distance, CURRENT)
    print CURRENT, EVALUATED, '\n'
    # cari node mana yang harus di-close
    min_distance = -1
    min_node = None
    for node in EVALUATED:
        if node in OPENNED:
            if min_distance == -1 or min_distance > EVALUATED[node][0]:
                min_distance, min_node = EVALUATED[node][0], node
```

Gambar 3. 10 Algoritma Dijkstra

Setelah mengetahui panjang setiap ruas jalan pada peta maka berikutnya dapat di hitung tingkat kepadatan pada setiap ruas jalan tersebut. Tingkat kepadatan diukur melalui jumlah unit kendaraan dibagi dengan panjang ruas jalan dengan menggunakan rumus berikut : (1)

$$kepadatan = \frac{jumlah\ kendaraan}{panjang\ ruas\ jalan}$$

Sehingga pada masing – masing skenario mobilitas yang berbeda didapatkan data yang di tuliskan pada tabel – tabel berikut ini :

Tabel 3. 3 Kepadatan kendaraan pada peta 1 mobilitas 1

Road	Mobility1	Density1	Road	Mobility1	Density1
A	1	0.0044	AA	17	0.6227
B	2	0.0163	AB	10	0.1011
C	4	0.0399	AC	14	0.2186
D	7	0.0558	AD	5	0.0516
E	5	0.1027	AE	10	0.2077
F	5	0.0444	AF	12	0.1307
G	2	0.0258	AG	5	0.1099
H	3	0.0299	AH	3	0.0830
I	3	0.0403	AI	3	0.0327
J	5	0.0399	AJ	10	0.2382
K	10	0.1345	AK	7	0.0956
L	6	0.0936	AL	9	0.0910
M	6	0.1233	AM	3	0.0328
N	5	0.0605	AN	6	0.0608
O	4	0.0522	AO	5	0.0352
P	7	0.0697	AP	5	0.0523
Q	5	0.0659	AQ	3	0.0549
R	7	0.0558	AR	2	0.0215
S	6	0.0772	AS	3	0.0693
T	7	0.1096	AT	6	0.0644
U	6	0.0825	AU	3	0.0395
V	9	0.1851	AV	4	0.0888
W	7	0.0931	AW	3	0.0308
X	8	0.0841	AX	3	0.0670
Y	6	0.0470	AY	6	0.1352
Z	16	0.2238	AZ	6	0.1055

Tabel 3. 4 Kepadatan kendaraan pada peta 1 mobilitas 2

Road	Mobility2	Density2	Road	Mobility2	Density2
A	4	0.0177	AA	23	0.8425
B	1	0.0081	AB	8	0.0809
C	6	0.0599	AC	18	0.2810
D	9	0.0717	AD	3	0.0310
E	3	0.0616	AE	11	0.2285
F	9	0.0800	AF	10	0.1089
G	8	0.1034	AG	7	0.1538

H	3	0.0299	AH	7	0.1936
I	1	0.0134	AI	5	0.0545
J	3	0.0239	AJ	11	0.2620
K	2	0.0269	AK	6	0.0819
L	4	0.0624	AL	7	0.0708
M	2	0.0411	AM	5	0.0547
N	12	0.1451	AN	4	0.0405
O	8	0.1044	AO	3	0.0211
P	3	0.0299	AP	4	0.0419
Q	3	0.0395	AQ	7	0.1280
R	3	0.0239	AR	6	0.0645
S	2	0.0257	AS	1	0.0231
T	4	0.0626	AT	5	0.0537
U	2	0.0275	AU	5	0.0658
V	2	0.0411	AV	7	0.1555
W	12	0.1596	AW	2	0.0206
X	12	0.1261	AX	4	0.0894
Y	10	0.0784	AY	5	0.1126
Z	18	0.2518	AZ	5	0.0879

Tabel 3. 5 Kepadatan kendaraan pada peta 1 mobilitas 3

Road	Mobility3	Density3	Road	Mobility3	Density3
A	1	0.0044	AA	16	0.5861
B	1	0.0081	AB	3	0.0303
C	6	0.0599	AC	13	0.2030
D	6	0.0478	AD	3	0.0310
E	2	0.0411	AE	9	0.1870
F	4	0.0356	AF	10	0.1089
G	6	0.0775	AG	7	0.1538
H	9	0.0897	AH	4	0.1107
I	4	0.0538	AI	6	0.0655
J	5	0.0399	AJ	12	0.2859
K	6	0.0807	AK	3	0.0410
L	10	0.1560	AL	3	0.0303
M	8	0.1644	AM	4	0.0438
N	2	0.0242	AN	4	0.0405
O	12	0.1566	AO	5	0.0352
P	7	0.0697	AP	9	0.0942
Q	6	0.0790	AQ	7	0.1280

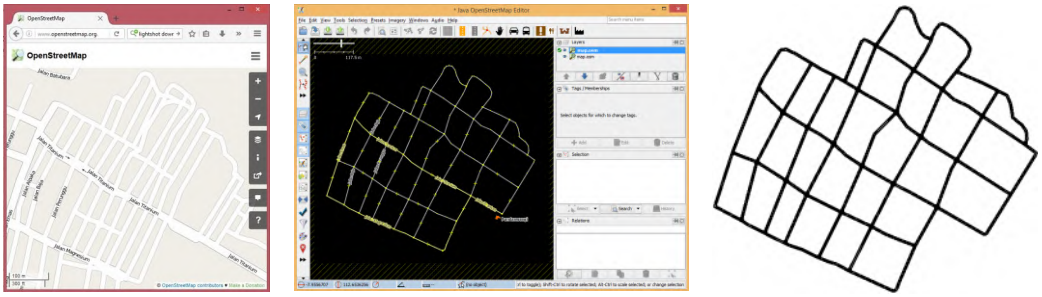
R	9	0.0718	AR	3	0.0322
S	3	0.0386	AS	5	0.1155
T	8	0.1252	AT	9	0.0966
U	2	0.0275	AU	6	0.0790
V	6	0.1234	AV	4	0.0888
W	9	0.1197	AW	3	0.0308
X	14	0.1472	AX	4	0.0894
Y	9	0.0706	AY	6	0.1352
Z	16	0.2238	AZ	9	0.1582

Tabel 3. 6 Kepadatan kendaraan pada peta 1 mobilitas 4

Road	Mobility4	Density4	Road	Mobility4	Density4
A	6	0.0266	AA	18	0.6593
B	4	0.0325	AB	5	0.0506
C	6	0.0599	AC	17	0.2654
D	4	0.0319	AD	5	0.0516
E	10	0.2054	AE	13	0.2700
F	11	0.0978	AF	14	0.1525
G	4	0.0517	AG	9	0.1978
H	7	0.0698	AH	6	0.1660
I	3	0.0403	AI	5	0.0545
J	5	0.0399	AJ	8	0.1906
K	4	0.0538	AK	3	0.0410
L	9	0.1404	AL	3	0.0303
M	5	0.1027	AM	5	0.0547
N	12	0.1451	AN	8	0.0810
O	9	0.1175	AO	9	0.0633
P	4	0.0398	AP	7	0.0732
Q	1	0.0132	AQ	5	0.0914
R	6	0.0479	AR	3	0.0322
S	5	0.0643	AS	5	0.1155
T	4	0.0626	AT	4	0.0429
U	7	0.0963	AU	3	0.0395
V	5	0.1029	AV	1	0.0222
W	13	0.1729	AW	5	0.0514
X	9	0.0946	AX	6	0.1341
Y	8	0.0627	AY	3	0.0676
Z	16	0.2238	AZ	5	0.0879

Percobaan pengiriman data dilakukan pada setiap skenario untuk kemudian di evaluasi hasilnya.

3.5.2 Peta 2



(a) (b) (c)

Gambar 3. 11 Pemrosesan peta 2

Pada Gambar 3.11 di atas dapat diketahui tahapan pemrosesan peta dari mulai memilih peta yang digunakan dari Open Street Map (Gambar 3.11.a) kemudian dilanjutkan dengan melakukan penyesuaian peta pada aplikasi Java Open Street Map (Gambar 3.11.b) sampai hasil akhir peta setelah di proses oleh SUMO (Gambar 3.11.c)

Berikut informasi teknis peta 2:

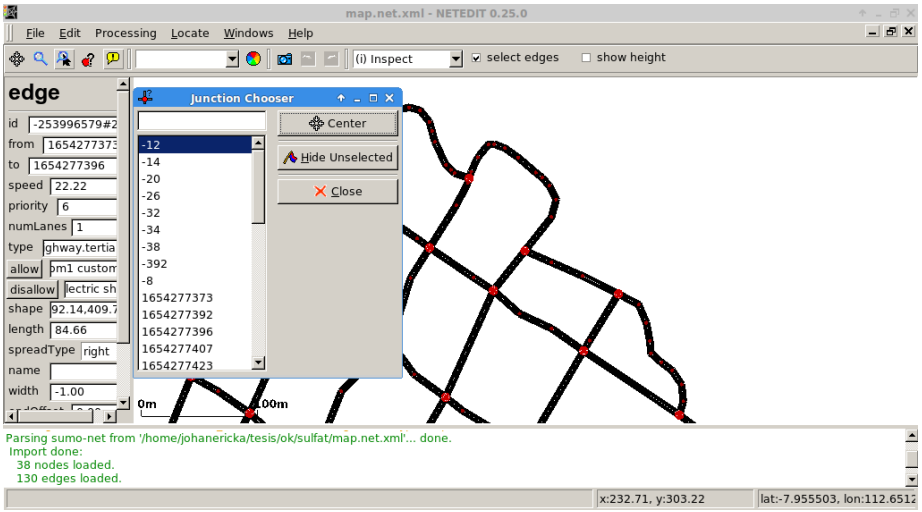
- Luas area 700m x 700m
- Jumlah persimpangan jalan 38
- Jumlah ruas jalan 89

Pada peta di atas dilakukan percobaan pengiriman data dengan menggunakan skenario sebagai berikut :

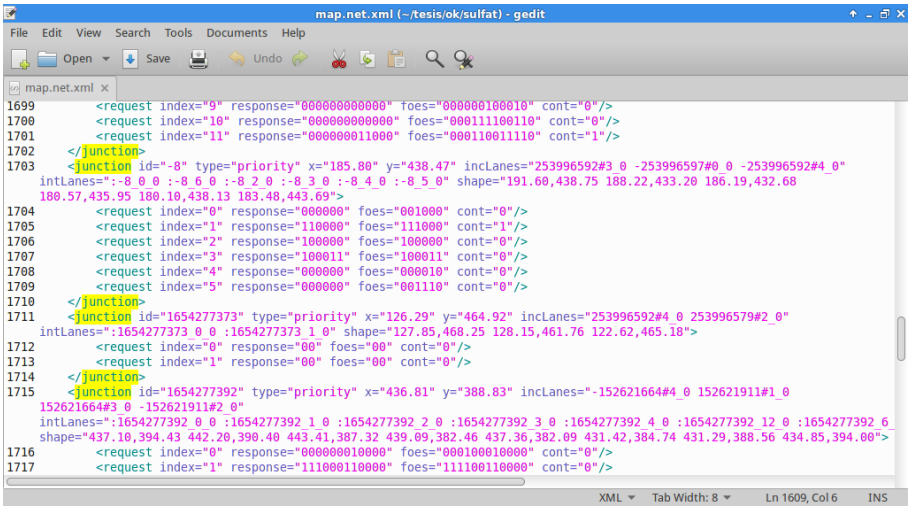
- Skenario 1 Tanpa *Static Intersection Node*
- Skenario 2 Dengan *Static Intersection Node* pada setiap persimpangan
- Skenario 3 Dengan *Static Intersection Node* pada persimpangan tertentu

Masing – masing skenario dilakukan 4 kali percobaan dengan mobilitas kendaraan yang berbeda – beda. Mobilitas kendaraan di hasilkan dari program randomTrips.py yang ada pada SUMO.

Agar dapat meletakkan *Static Intersection Node* untuk percobaan berikutnya, maka perlu diketahui terlebih dahulu koordinat masing – masing *intersection* yang ada pada peta. Cara untuk mengetahui koordinat setiap *intersection* yang ada pada peta, dapat menggunakan fitur Junction pada netedit seperti yang terlihat pada Gambar 3.12 atau langsung melihat dari file map.net.xml seperti yang terlihat pada Gambar 3.13 berikut ini :

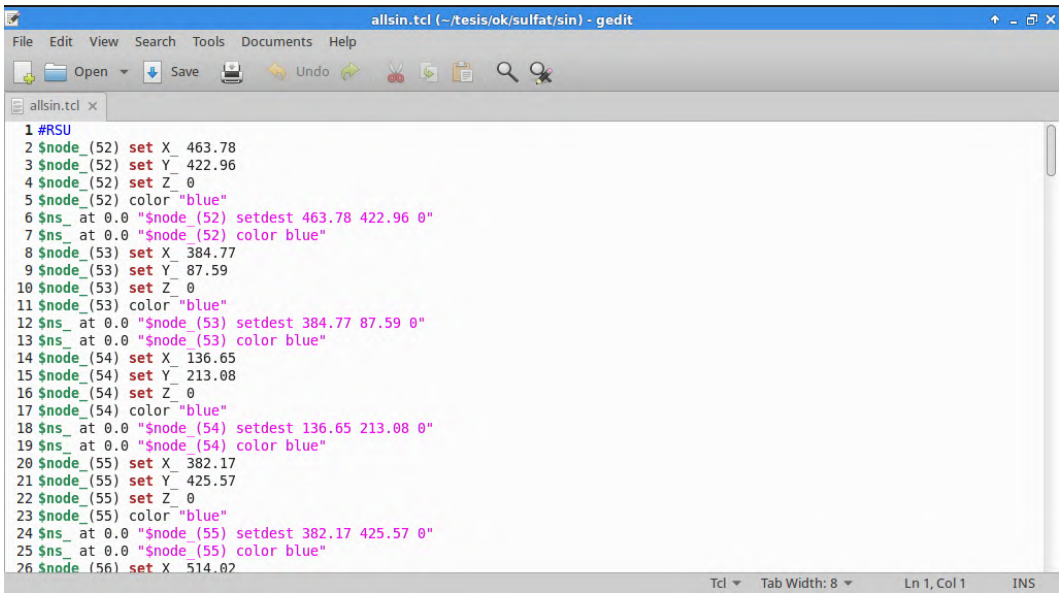


Gambar 3. 12 Koordinat *intersections* pada Net Edit



Gambar 3. 13 Koordinat *intersections* pada file map.net.xml

Setelah di ketahui koordinat masing – masing *intersection* pada peta, maka selanjutnya membuat *Static Intersection Node* pada koordinat tersebut. Untuk memudahkan proses maka pada penelitian ini *Static Intersection Node* diletakkan pada file tersendiri yang dinam sin.tcl seperti yang tampak pada Gambar 3.14 berikut ini :



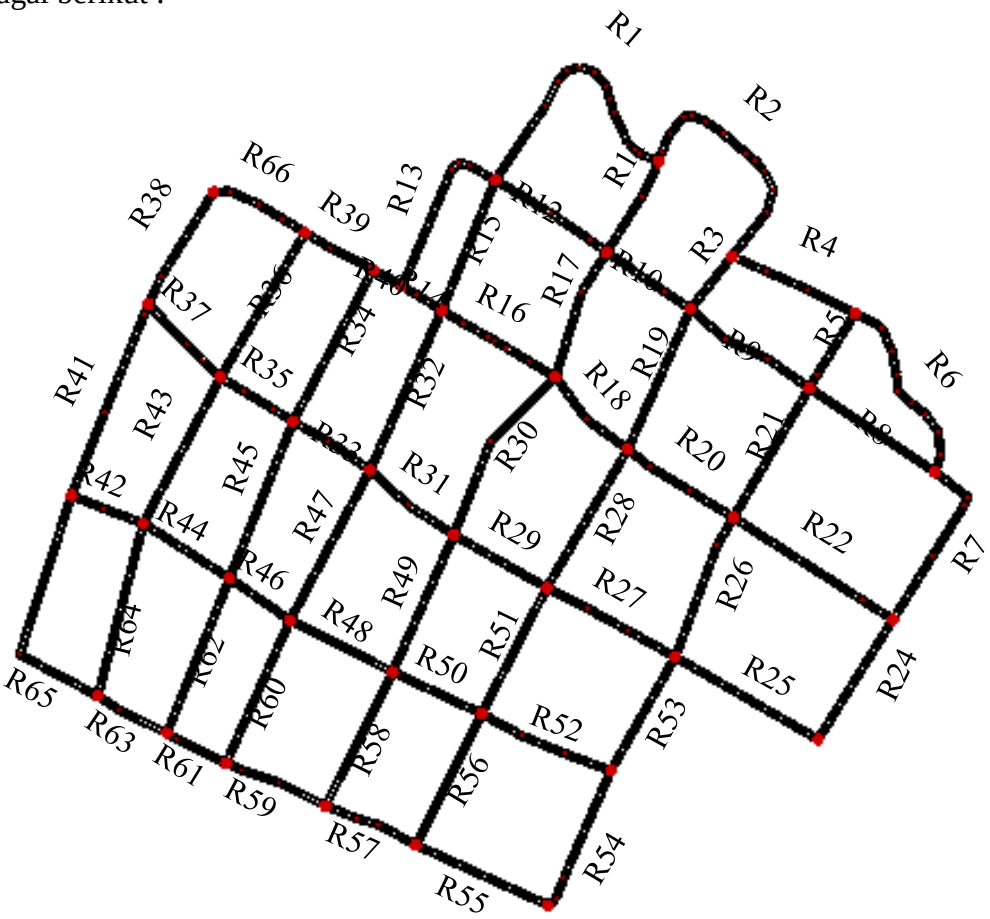
Gambar 3. 14 Koodinat *Static Intersection Node*

Static Intersection Node dimulai dari node nomor 52 karena node nomor 0 sampai 49 digunakan untuk kendaraan (*vehicle*). Sedangkan *node* nomor 50 dan 51 digunakan sebagai *node* pengirim dan penerima. Untuk membed *node* yang merepresentasikan kendaraan, pengirim & penerima serta *Static Intersection Node* maka dilakukan perbedaan warna sebagai berikut :

- Node berwarna hitam merepresentasikan kendaraan (*vehicle*)
- Node berwarna merah merepresentasikan *node* pengirim & penerima
- Node berwarna biru merepresentasikan *Static Intersection Node*

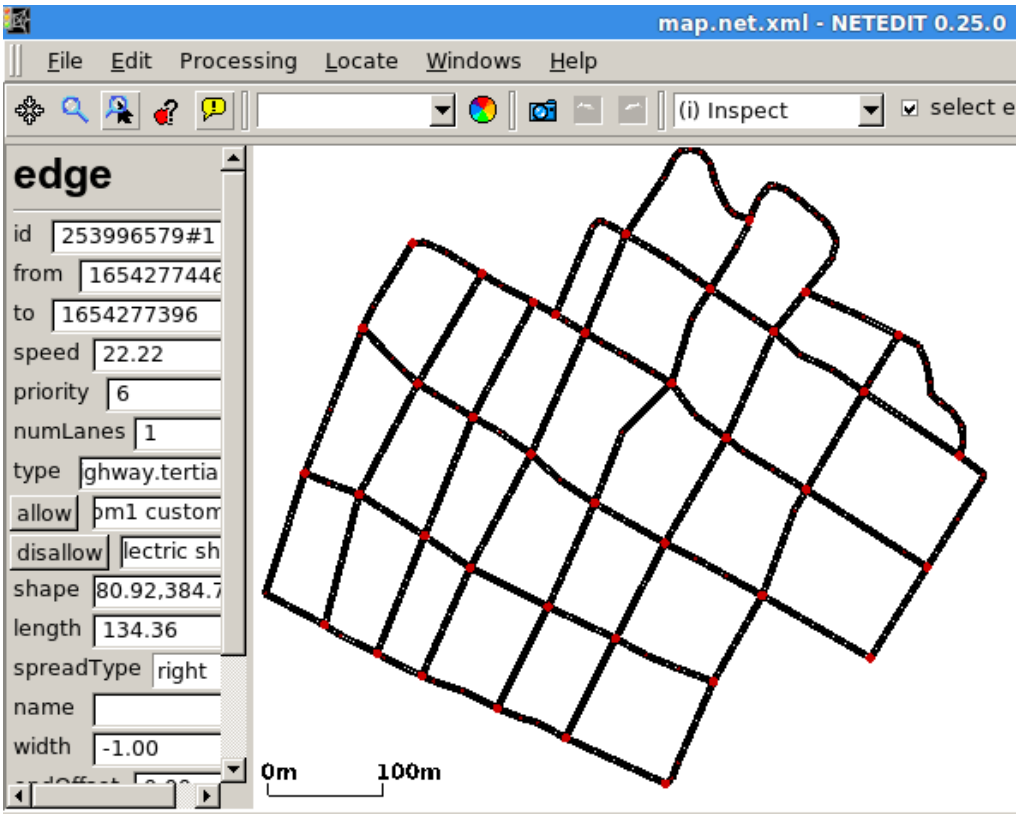
Setelah mengetahui posisi setiap *Static Intersection Node* pada peta maka berikutnya dilakukan pengurangan jumlah *Static Intersection Node* namun dengan tetap memaksimalkan *Packet Delivery Ratio*.

Pada penelitian ini diusulkan menggunakan algoritma Dijkstra untuk mengurangi jumlah *node* pada peta. Algoritma Dijkstra digunakan untuk menentukan jalur dengan tingkat kepadatan yang paling rendah antara *node* pengirim dan *node* penerima. Dengan asumsi jalan yang membutuhkan *Static Intersection Node* adalah jalan dengan kepadatan kendaraan yang rendah karena pada jalan dengan kepadatan kendaraan yang tinggi, data dapat dikirimkan antar kendaraan. Sehingga kurang efektif apabila meletakkan *Static Intersection Node* pada jalan yang tingkat kepadatannya tinggi. Maka diperlukan untuk mengetahui tingkat kepadatan kendaraan pada setiap ruas jalan yang ada pada peta. Untuk memudahkan pengitungan tingkat kepadatan di masing – masing ruas jalan, maka diberikan nama pada setiap ruas jalan seperti yang tampak pada Gambar 1.15 sebagai berikut :



Gambar 3. 15 Penamaan ruas jalan pada peta 2

Setelah diberikan penamaan pada setiap ruas jalan, maka langkah selanjutnya adalah mengukur panjang setiap ruas jalan yang ada pada peta. Pengukuran dapat dilakukan melalui program Net Edit seperti yang tampak pada Gambar 3.16 berikut ini :



Gambar 3. 16 Net Edit Peta 2

Berikut daftar setiap ruas jalan beserta panjang nya yang didapat dari peta pada Gambar di atas :

Tabel 3. 7 Panjang setiap ruas jalan pada peta 2

Road	Length (m)	Road	Length (m)	Road	Length (m)
R1	107.19	R24	92.11	R46	47.96
R2	78.34	R25	107.41	R47	110.91
R3	43.50	R26	98.38	R48	74.87
R4	88.26	R27	95.04	R49	97.75
R5	57.22	R28	104.67	R50	63.95
R6	116.18	R29	69.54	R51	92.40
R7	99.82	R30	122.42	R52	91.88
R8	98.77	R31	69.03	R53	85.28
R9	92.97	R32	114.23	R54	96.73
R10	65.84	R33	55.34	R55	94.44
R11	68.54	R34	111.85	R56	95.62
R12	86.45	R35	55.34	R57	64.04
R13	91.84	R36	109.22	R58	97.65
R14	30.05	R37	67.00	R59	70.41
R15	91.61	R38	84.66	R60	102.00
R16	85.97	R39	50.53	R61	43.26
R17	87.88	R40	21.82	R62	108.93
R18	66.49	R41	134.36	R63	51.10
R19	100.32	R42	50.20	R64	116.36
R20	81.92	R43	107.68	R65	131.76
R21	97.82	R44	66.54	R66	65.12
R22	123.30	R45	110.27		

Langkah berikutnya adalah menentukan jalur dari *node* pengirim ke *node* penerima melalui jalur dengan kepadatan yang paling rendah atau dengan kata lain paling sepi. Dengan asumsi ruas jalan dengan kepadatan kendaraan tinggi tidak memerlukan *Static Intersection Node* karena kendaraan yang berada pada ruas jalan tersebut dapat mengirimkan data. Hal ini dilakukan dengan memasukkan informasi kepadatan setiap ruas jalan di atas serta daftar *intersection* yang saling terhubung ke dalam program Dijkstra. Sedikit modifikasi dilakukan pada program Dijkstra untuk memilih jalan dengan kepadatan terendah. Berikut kode program Dijkstra untuk mengetahui jalur dengan kepadatan terendah pada peta yang menghubungkan *node* pengirim dan *node* penerima dapat dilihat pada Gambar 3.17 berikut ini :

```
# add to evaluated
while i_can_do_it:
    for node in EDGES[CURRENT]:
        if node in OPENNED:
            distance = EDGES[CURRENT][node]
            if (node not in EVALUATED) or (EVALUATED[node][0] > distance):
                EVALUATED[node] = (EVALUATED[CURRENT][0]+distance, CURRENT)
    print CURRENT, EVALUATED, '\n'
    # cari node mana yang harus di-close
    min_distance = -1
    min_node = None
    for node in EVALUATED:
        if node in OPENNED:
            if min_distance == -1 or min_distance > EVALUATED[node][0]:
                min_distance, min_node = EVALUATED[node][0], node
    if min_node != None:
        OPENNED.remove(min_node)
        CURRENT = min_node
    else:
        i_can_do_it = False

current = END
while current != START:
    print current
    current = EVALUATED[current][1]
```

Gambar 3. 17 Algoritma Dijkstra

Setelah mengetahui panjang setiap ruas jalan pada peta maka berikutnya dapat di hitung tingkat kepadatan pada setiap ruas jalan tersebut. Tingkat kepadatan diukur dari jumlah unit kendaraan yang berada pada ruas jalan dibagi panjang ruas jalan tersebut dengan menggunakan rumus berikut :

(2)

$$kepadatan = \frac{jumlah\ kendaraan}{panjang\ ruas\ jalan}$$

Sehingga pada 4 skenario mobilitas yang berbeda didapatkan data yang di tuliskan pada tabel – tabel berikut ini :

Tabel 3. 8 Kepadatan kendaraan pada Peta 2 mobilitas 1

road	mobility1	density1	road	mobility1	density1	road	mobility1	density1
R1	1	0.0093	R24	5	0.0543	R46	3	0.0626
R2	1	0.0128	R25	6	0.0559	R47	5	0.0451
R3	3	0.0690	R26	9	0.0915	R48	4	0.0534
R4	3	0.0340	R27	10	0.1052	R49	8	0.0818
R5	5	0.0874	R28	8	0.0764	R50	1	0.0156
R6	0.1	0.0009	R29	17	0.2445	R51	13	0.1407
R7	5	0.0501	R30	3	0.0245	R52	1	0.0109

R8	4	0.0405	R31	20	0.2897	R53	7	0.0821
R9	2	0.0215	R32	4	0.0350	R54	4	0.0414
R10	3	0.0456	R33	16	0.2891	R55	6	0.0635
R11	1	0.0146	R34	7	0.0626	R56	5	0.0523
R12	1	0.0116	R35	11	0.1988	R57	9	0.1405
R13	1	0.0109	R36	6	0.0549	R58	9	0.0922
R14	5	0.1664	R37	3	0.0448	R59	6	0.0852
R15	1	0.0109	R38	8	0.0945	R60	2	0.0196
R16	3	0.0349	R39	5	0.0990	R61	6	0.1387
R17	3	0.0341	R40	7	0.3208	R62	1	0.0092
R18	4	0.0602	R41	8	0.0595	R63	7	0.1370
R19	5	0.0498	R42	3	0.0598	R64	0.1	0.0009
R20	1	0.0122	R43	3	0.0279	R65	6	0.0455
R21	9	0.0920	R44	5	0.0751	R66	7	0.1075
R22	3	0.0243	R45	5	0.0453			

Tabel 3. 9 Kepadatan kendaraan pada Peta 2 mobilitas 2

Road	Mobility2	Density2	Road	Mobility2	Density2	Road	Mobility2	Density2
R1	3	0.0280	R24	4	0.0434	R46	5	0.1043
R2	2	0.0255	R25	7	0.0652	R47	2	0.0180
R3	4	0.0920	R26	3	0.0305	R48	5	0.0668
R4	1	0.0113	R27	8	0.0842	R49	2	0.0205
R5	2	0.0350	R28	7	0.0669	R50	4	0.0625
R6	1	0.0086	R29	9	0.1294	R51	7	0.0758
R7	2	0.0200	R30	5	0.0408	R52	5	0.0544
R8	1	0.0101	R31	9	0.1304	R53	7	0.0821
R9	1	0.0108	R32	2	0.0175	R54	5	0.0517
R10	2	0.0304	R33	10	0.1807	R55	5	0.0529
R11	5	0.0730	R34	4	0.0358	R56	3	0.0314
R12	4	0.0463	R35	8	0.1446	R57	9	0.1405
R13	2	0.0218	R36	1	0.0092	R58	1	0.0102
R14	5	0.1664	R37	6	0.0896	R59	10	0.1420
R15	5	0.0546	R38	3	0.0354	R60	5	0.0490
R16	2	0.0233	R39	2	0.0396	R61	9	0.2080
R17	7	0.0797	R40	4	0.1833	R62	6	0.0551
R18	5	0.0752	R41	4	0.0298	R63	6	0.1174
R19	8	0.0797	R42	3	0.0598	R64	2	0.0172
R20	4	0.0488	R43	3	0.0279	R65	4	0.0304
R21	1	0.0102	R44	2	0.0301	R66	1	0.0154
R22	4	0.0324	R45	5	0.0453			

Tabel 3. 10 Kepadatan kendaraan pada Peta 2 mobilitas 3

Road	Mobility3	Density3	Road	Mobility3	Density3	Road	Mobility3	Density3
R1	2	0.0187	R24	6	0.0487	R46	4	0.0834
R2	4	0.0511	R25	4	0.0434	R47	4	0.0361
R3	6	0.1379	R26	0.1	0.0009	R48	4	0.0534
R4	3	0.0340	R27	13	0.1321	R49	9	0.0921
R5	4	0.0699	R28	11	0.1157	R50	1	0.0156
R6	0.1	0.0009	R29	4	0.0382	R51	10	0.1082
R7	3	0.0301	R30	16	0.2301	R52	1	0.0109
R8	4	0.0405	R31	10	0.0817	R53	2	0.0235
R9	3	0.0323	R32	23	0.3332	R54	1	0.0103
R10	4	0.0608	R33	2	0.0175	R55	2	0.0212
R11	8	0.1167	R34	21	0.3795	R56	4	0.0418
R12	0.1	0.0012	R35	8	0.0715	R57	1	0.0156
R13	0.1	0.0011	R36	12	0.2168	R58	6	0.0614
R14	4	0.1331	R37	5	0.0458	R59	4	0.0568
R15	2	0.0218	R38	6	0.0896	R60	0.1	0.0010
R16	4	0.0465	R39	2	0.0236	R61	4	0.0925
R17	10	0.1138	R40	4	0.0792	R62	1	0.0092
R18	5	0.0752	R41	4	0.1833	R63	2	0.0391
R19	5	0.0498	R42	5	0.0372	R64	1	0.0086
R20	5	0.0610	R43	1	0.0199	R65	2	0.0152
R21	6	0.0613	R44	4	0.0371	R66	0.1	0.0015
R22	2	0.0187	R45	3	0.0451			

Tabel 3. 11 Kepadatan kendaraan pada Peta 2 mobilitas 4

Road	Mobility4	Density4	Road	Mobility4	Density4	Road	Mobility4	Density4
R1	2	0.0187	R24	2	0.0217	R46	6	0.1251
R2	1	0.0128	R25	1	0.0093	R47	6	0.0541
R3	3	0.0690	R26	8	0.0813	R48	2	0.0267
R4	3	0.0340	R27	7	0.0737	R49	5	0.0512
R5	3	0.0524	R28	9	0.0860	R50	3	0.0469
R6	0.1	0.0009	R29	11	0.1582	R51	11	0.1190
R7	2	0.0200	R30	1	0.0082	R52	1	0.0109
R8	2	0.0202	R31	10	0.1449	R53	4	0.0469
R9	2	0.0215	R32	7	0.0613	R54	4	0.0414
R10	2	0.0304	R33	12	0.2168	R55	4	0.0424
R11	3	0.0438	R34	4	0.0358	R56	5	0.0523

R12	4	0.0463	R35	14	0.2530	R57	4	0.0625
R13	2	0.0218	R36	4	0.0366	R58	4	0.0410
R14	8	0.2662	R37	3	0.0448	R59	4	0.0568
R15	2	0.0218	R38	4	0.0472	R60	2	0.0196
R16	4	0.0465	R39	4	0.0792	R61	4	0.0925
R17	5	0.0569	R40	7	0.3208	R62	5	0.0459
R18	7	0.1053	R41	3	0.0223	R63	2	0.0391
R19	4	0.0399	R42	6	0.1195	R64	0.1	0.0009
R20	6	0.0732	R43	7	0.0650	R65	1	0.0076
R21	5	0.0511	R44	6	0.0902	R66	3	0.0461
R22	5	0.0406	R45	10	0.0907			

Percobaan pengiriman data dilakukan pada setiap skenario di atas untuk kemudian di evaluasi hasilnya.

3.6 Analisis Hasil

Hal-hal yang menjadi fokus utama dalam analisa hasil adalah poin-poin berikut ini:

- *Packet Delivery Ratio* yaitu rasio paket data yang berhasil diterima oleh node tujuan yang bisa didapatkan melalui rumus sebagai berikut :

$$PDR = \frac{packet\ received}{packet\ sent} \times 100\% \quad (3)$$

- *Average End to End Delay* yaitu rata – rata waktu yang dibutuhkan untuk data sampai di tujuan yang bisa didapatkan melalui rumus sebagai berikut

$$Delay = \frac{\sum_{i \le sent}^{i=0} t_{received[i]} - t_{sent[i]}}{packet\ sent} \quad (4)$$

- *Packet Loss* yaitu jumlah paket yang tidak sampai di tujuan yang bisa didapatkan melalui rumus sebagai berikut

$$PL = \frac{packet\ sent - packet\ received}{packet\ sent} \times 100\% \quad (5)$$

Analisa hasil tersebut dilakukan dengan menggunakan script AWK. Berikut script AWK untuk mendapatkan *Packet Delivery Ratio, Average End to End Delay* serta *Packet Loss* :

```
#initial
total_pkt_sent=0;
total_pkt_recvd=0;
total_pkt_drop=0;
pkt_delivery_ratio = 0;

#menghitung packet sent
if((pkt_type == "cbr") && (state == "s") && (level=="AGT")) {
        for(i=0;i<133;i++) {
                if(i == node_id) {
                packet_sent[i] = packet_sent[i] + 1; }}

#menghitung packet received
}else if((pkt_type == "cbr") && (state == "r") && (level=="AGT")) {
        for(i=0;i<133;i++) {
                if(i == node_id) {
                packet_recvd[i] = packet_recvd[i] + 1; }}

#menghitung packet dropped
}else if((pkt_type == "cbr") && (state == "d")) {
        for(i=0;i<133;i++) {
                if(i == node_id) {
                packet_drop[i] = packet_drop[i] + 1; }}

#total paket
total_pkt_sent = total_pkt_sent + packet_sent[i];
total_pkt_recvd = total_pkt_recvd + packet_recvd[i];
total_pkt_drop = total_pkt_drop + packet_drop[i];

#Packet Delivery Ratio
pkt_delivery_ratio = (total_pkt_recvd/total_pkt_sent)*100;
printf("Packet Delivery Ratio : %.2f%\n",pkt_delivery_ratio);

#packet Loss
printf("Total Packets Dropped :%d\n",total_pkt_drop);

# Average End to End Delay
if ( start_time[packet_id] == 0 )  { start_time[packet_id] = time; }
if (( state == "s") &&  ( pkt_type == "cbr" ) && ( level == "AGT" ))  {
start_time[packet_id] = time; }
 if (( state == "r") &&  ( pkt_type == "cbr" ) && ( level == "AGT" )) {
end_time[packet_id] = time;  }
 else {  end_time[packet_id] = -1;  }

# End to End Delay
for ( i in end_time ) {
 start = start_time[i];
 end = end_time[i];
 packet_duration = end - start;
 if ( packet_duration > 0 )  { sum += packet_duration; recvnum++; }
}
delay=sum/recvnum;
printf("Average End to End Delay     :%.9f ms\n", delay);
```

Gambar 3. 18 Kode Program menghitung *Packet Delivery Ratio, Packet Loss, End to End Delay*

BAB IV
HASIL & PEMBAHASAN

Pada bab ini diuraikan implementasi dari skenario pengujian serta evaluasi hasil pengujian dengan melakukan analisa terhadap data hasil pengujian. Dari rencana penelitian yang telah di jelaskan pada bab sebelumnya, berikut disajikan hasil dari penelitian ini.

4.1 Pengujian Peta 1

Pengujian dilakukan dengan cara mengirimkan data dari *node* pengirim dari sisi bawah peta ke *node* penerima pada sisi atas peta. Jarak antara kedua *node* kurang lebih 350m tanpa adanya koneksi langsung antara kedua *node*. Berikut data – data teknis yang digunakan pada skenario ini :

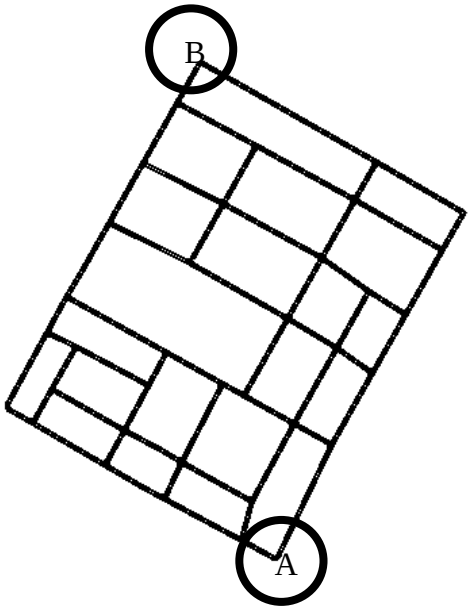
Tabel 4. 1 Data Skenario 1

Peta	Peta 1
Luas Area	350m x 350m
Jumlah persimpangan	31
Jumlah ruas jalan	54
Jumlah kendaraan	25, 50, 75, 100
Mobilitas kendaraan	Acak
Lama waktu pengujian	100 detik
Jenis data dikirimkan	CBR (UDP)
Jumlah data dikirimkan	1 data / detik
Protokol Routing	AODV
Radius transmisi radio	100m
Kecepatan kendaraan	8 m/s (30 km/j)

Pada peta 2 ini dilakukan percobaan pengiriman data pada 3 skenario yaitu sebagai berikut :

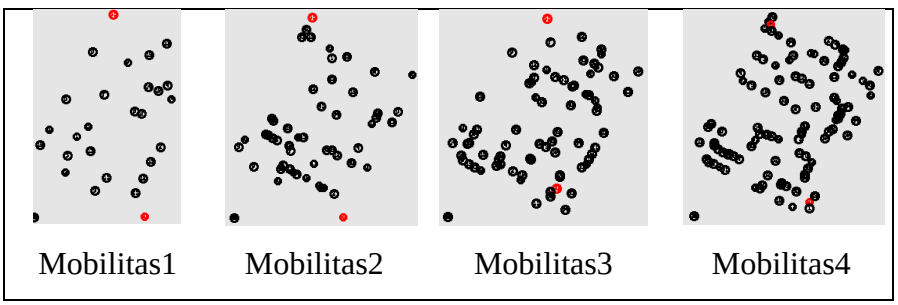
4.1.1 Skenario 1 Tanpa *Static Intersection Node*

Pengiriman data dilakukan dari *node* pada posisi A ke *node* pada posisi B dimana tidak terdapat koneksi secara langsung dari *node* A ke *node* B seperti tampak pada Gambar 4.1 berikut ini :



Gambar 4. 1 Posisi node Pengirim & Penerima pada Peta 1

Pada skenario ini data dikirimkan dari *node* A ke *node* B melalui kendaraan yang berada di antara *node* A dan *node* B. Tampilan visual dari posisi kendaraan di peta pada detik ke 10 di setiap merbilitas dapat dilihat pada Gambar 4.2 berikut ini :



Gambar 4. 2Percobaan skenario 1 tanpa *Static Intersection Node* pada peta 1

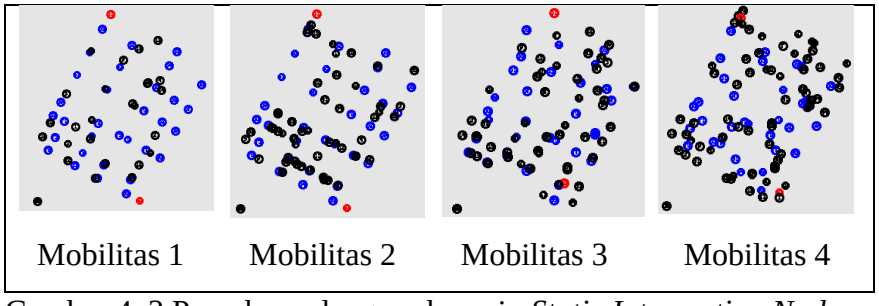
Dari Gambar 4.2 di atas dapat dilihat posisi *node* A & *node* B (berwarna merah) serta kendaraan yang melintas diantaranya (berwarna hitam). Gambar tersebut diambil dari tampilan Network Animator pada detik ke 10. Percobaan pengiriman data dilakukan dengan mengirimkan 1 paket / detik selama 100 detik (100 paket dikirimkan). Hasil yang didapatkan dapat dilihat pada tabel 4.2 berikut ini :

Tabel 4. 2 Hasil Percobaan skenario 1 pada peta 1

	Packet Delivery Ratio (%)	**Packet Loss (%)**	**End to End Delay (s)**
Mobilitas 1	14,00%	85%	2,32
Mobilitas 2	21,00%	77%	1,97
Mobilitas 3	77,00%	46%	0,52
Mobilitas 4	40,00%	70%	3,51
Rata - Rata	**38,00%**	**70%**	**2,08**

4.1.2 Dengan *Static Intersection Node* pada setiap persimpangan jalan

Percobaan berikutnya adalah dengan menambahkan *Static Intersection Node* pada setiap persimpangan yang ada pada peta. Hal ini dilakukan untuk membuktikan bahwa penggunaan *Static Intersection Node* berkontribusi terhadap optimasi protokol AODV dalam melakukan pengiriman paket data. Hasil penelitian tersebut secara visual dapat dilihat pada Gambar 4.3 berikut ini. Gambar berikut merupakan tampilan visual dari Network Animator pada detik ke 10 di setiap skenario :



Gambar 4. 3 Percobaan dengan skenario *Static Intersection Node*

Tampak pada percobaan di atas terdapat *Static Intersection Node* yang berfungsi sebagai *repeater* (berwarna biru) diletakkan pada setiap persimpangan jalan yang ada pada peta. Hasil percobaan pengiriman data pada skenario tersebut dapat dilihat pada tabel 4.3 berikut ini :

Tabel 4. 3 Hasil Percobaan Dengan *SIN* di semua *intersection* pada peta 1

	Packet Delivery Ratio (%)	**Packet Loss (%)**	**End to End Delay (s)**
Mobilitas 1	82,00%	43%	0,85
Mobilitas 2	88,00%	29%	2,26
Mobilitas 3	71,00%	43%	1,94
Mobilitas 4	75,00%	52%	0,31
Rata - Rata	**79,00%**	**42%**	**1,34**

4.1.3 Skenario Dengan *Static Intersection Node* pada *intersection* tertentu

Pada skenario ini digunakan algoritma Dijkstra untuk menentukan jalur dengan intensitas kendaraan terendah dari *node* A ke *node* B. Penentuan jalur dengan kepadatan kendaran terendah dilakukan sebanyak 4 kali dengan mobilitas kendaraan yang berbeda – beda. Hasil pemilihan jalur dapat dilihat pada tabel 4.4 berikut ini :

Tabel 4. 4 Hasil pemilihan jalur menggunakan Dijkstra

Mobilitas 1	**Mobilitas 2**	**Mobilitas 3**	**Mobilitas 4**
Road G	Road G	Road C	Road G
Road O	Road O	Road D	Road O
Road X	Road X	Road F	Road X
Road Y	Road Y	Road N	Road Y
Road Z	Road Z	Road W	Road Z
Road AA	Road AA	Road AF	Road AA

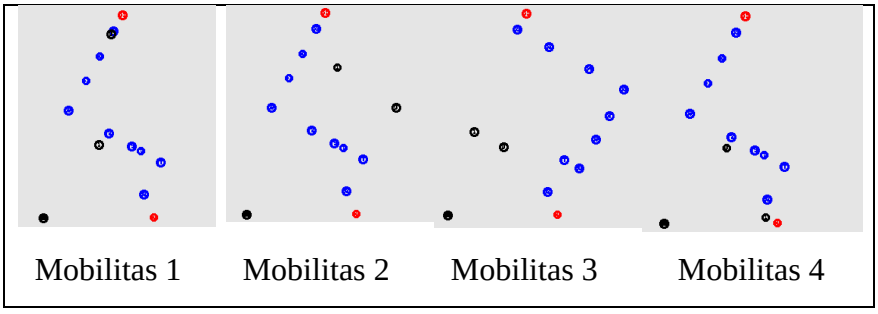
Road AC	Road AC	Road AE	Road AC
Road AN	Road AN	Road AN	Road AN

Dari hasil pemilihan jalan pada tabel 4.4 di atas berikutnya dapat ditentukan posisi *Static Intersection Node* yaitu pada setiap *instersection* yang dilalui jalur tersebut. *Intersection* yang terpilih dapat dilihat pada tabel 4.5 berikut ini :

Tabel 4. 5 Daftar *intersection* terpilih pada peta 1

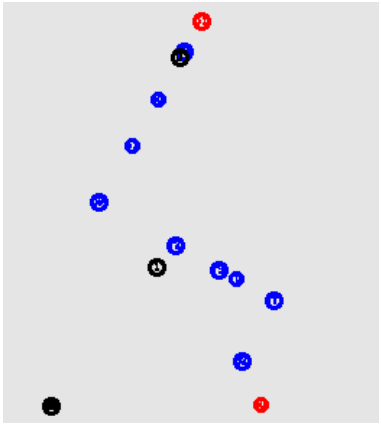
Mobilitas 1	Mobilitas 2	Mobilitas 3	Mobilitas 4
53	53	53	53
56	86	62	86
61	61	64	61
69	69	66	69
72	72	57	72
73	73	59	73
76	76	80	76
77	77	77	77
82	82	82	82

Kemudian diletakkan *Static Intersection Node* pada setiap *intersection* di atas. Hasil secara visual pada *Network Animator* dapat dilihat pada Gambar 4.4 berikut ini



Gambar 4. 4 Posisi *Static Intersection Node* hasil Dijkstra skenario 1

Dari ke 4 mobilitas tersebut diambil rata – rata posisi *Static Intersection Node* yang digunakan sehingga didapatkan posisi *Static Intersection Node* yang maksimal. Ditemukan 9 posisi *Static Intersection Node* yang paling optimal (sering digunakan) pada peta seperti tampak pada Gambar 4.5 berikut ini. Kemudian dilakukan pecobaan pengiriman data dengan 4 skenario mobilitas kendaraan yan berbeda :



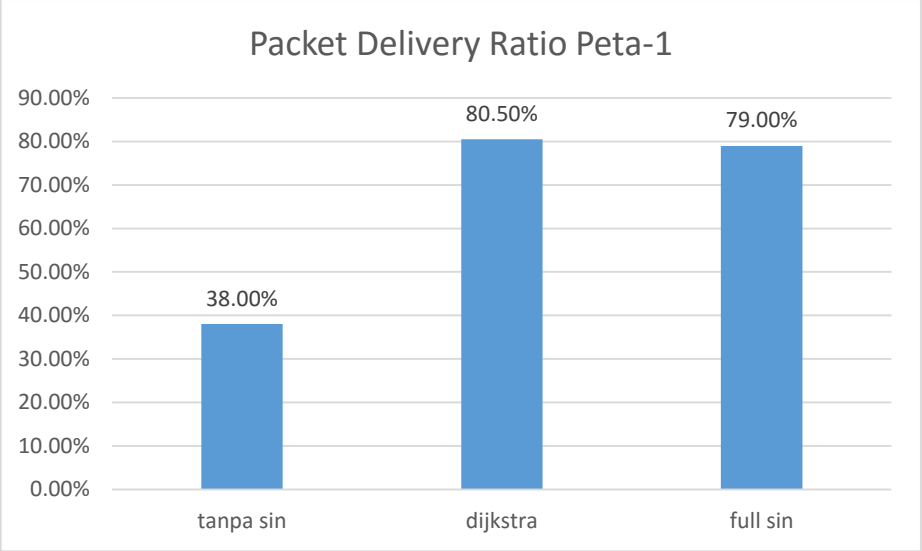
Gambar 4. 5 Posisi *Static Intersection Node* optimal peta 1

Tampak pada Gambar 4.5 di atas lokasi *Static Intersection Node* (warna biru) pada peta. Jumlah *Static Intersection Node* dikurangi dari total 32 titik *Static Intersection Node* menjadi hanya 9 titik *Static Intersection Node* yang optimal.

Tabel 4. 6 Rata – Rata Hasil Percobaan dengan *SIN* di *intersection* optimal pada peta 1

	Packet Delivery Ratio (%)	**Packet Loss (%)**	**End to End Delay (s)**
Mobility1	79,00%	37%	0,93
Mobility2	75,00%	56%	2,79
Mobility3	84,00%	38%	0,27
Mobility4	84,00%	37%	2,27
Rata - Rata	**80,50%**	**42%**	**1,56**

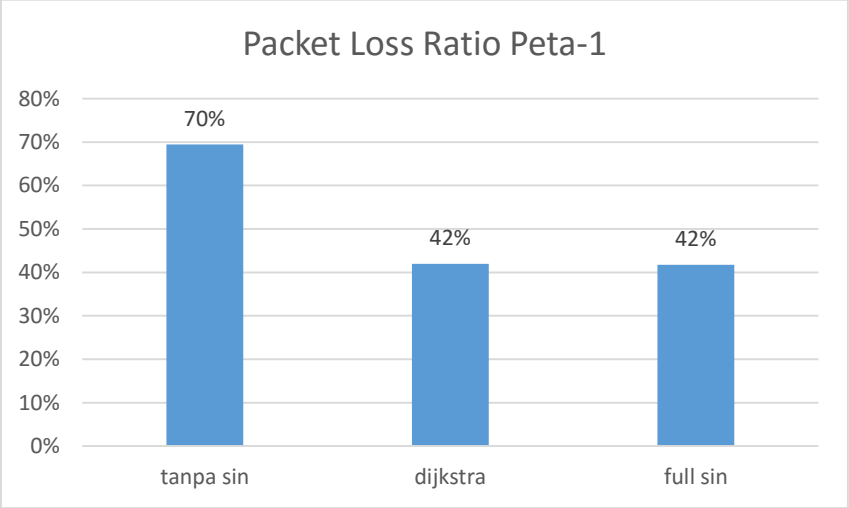
Hasil akhir percobaan pengiriman data yang dilakukan pada Peta 1 menggunakan *Static Intersection Node* di setiap *intersection* dibandingkan dengan *Static Intersection Node* pada *intersection* tertentu serta tanpa *Static Intersection Node* pada peta dapat dilihat pada Grafik 4.1 berikut ini :



Grafik 4. 1 Rata – Rata *Packet Delivery Ratio* pada peta 1

Dari grafik 4.1 di atas dapat dilihat bahwa dengan menambahkan *Static Intersection Node* pada peta dapat meningkatkan *Packet Delivery Ratio*. Pada grafik tersebut juga dapat dilihat bahwa peletakan *Static Intersection Node* pada *intersection* yang memiliki tingkat kepadatan kendaraan rendah juga memiliki kontribusi yang signifikan dalam meningkatkan *Packet Delivery Ratio* jika dibandingkan dengan skenario yang tanpa menggunakan *Static Intersection Node*.

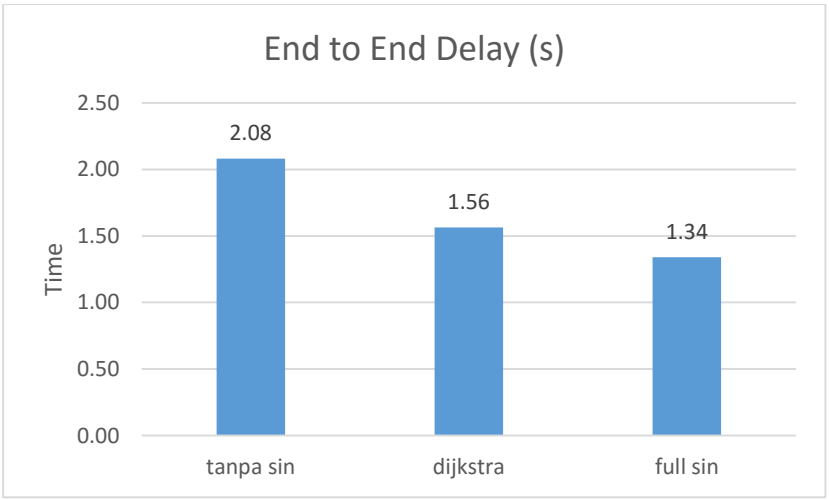
Sedangkan untuk *Packet Loss* yaitu jumlah paket data yang dikirimkan dari *node* A namun gagal diterima oleh *node* B dapat dilihat pada grafik 4.2 berikut ini :



Grafik 4. 2 Rata – rata *Packet Loss* pada peta 1

Dari grafik 4.2 di atas dapat diketahui bahwa peletakan *Static Intersection Node* pada peta dapat menurunkan *Packet Loss* yaitu jumlah paket data yang gagal diterima oleh *node* B. Penyebab terjadinya *Packet Loss* cukup beragam selain perubahan topologi jaringan karena mobilitas kendaraan, juga dapat disebabkan karena memory pada kendaraan yang dilalui paket data penuh sehingga paket data harus di *drop*. Selain itu juga dapat dipengaruhi dari *bandwidth* yang terbatas sehingga tidak semua paket dapat terkirim ke kendaraan berikutnya.

Untuk *End to End Delay* yaitu rata – rata waktu yang dibutuhkan paket data sejak dikirimkan sampai ke tujuan dapat dilihat pada grafik 4.3 berikut ini :



Grafik 4. 3 Rata – Rata *End to End Delay* pada peta 1

Pada grafik 4.3 di atas dapat diketahui bahwa *Static Intersection Node* memiliki kontribusi yang cukup signifikan dalam menekan *End to End Delay* yang artinya semakin kecil nilai *End to End Delay* maka semakin cepat data sampai ke tujuan.

4.2 Skenario Pengujian Peta 2

Pengujian dilakukan dengan cara mengirimkan data dari *node* pengirim dari sisi kiri peta ke *node* penerima pada sisi kanan peta. Jarak antara kedua *node* kurang lebih 700m tanpa adanya koneksi langsung antara kedua *node*. Berikut data – data teknis yang digunakan pada skenario ini :

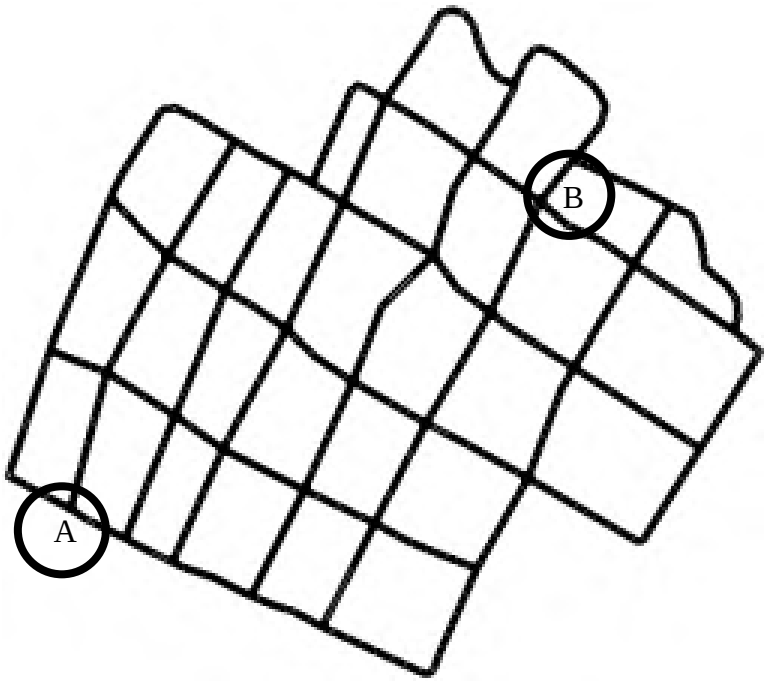
Tabel 4. 7 Data Skenario 2

Peta	Peta 2
Luas Area	700m x 700m
Jumlah persimpangan	38
Jumlah ruas jalan	66
Jumlah kendaraan	25, 50, 75, 100
Mobilitas kendaraan	Acak
Lama waktu pengujian	100 detik
Jenis data dikirimkan	CBR (UDP)
Jumlah data dikirimkan	1 data / detik
Protokol Routing	AODV
Radius transmisi radio	100m
Kecepatan kendaraan maksimal	8 m/s (30 km/j)

Pada skenario ini dilakukan percobaan pengiriman data pada 3 skenario yaitu sebagai berikut :

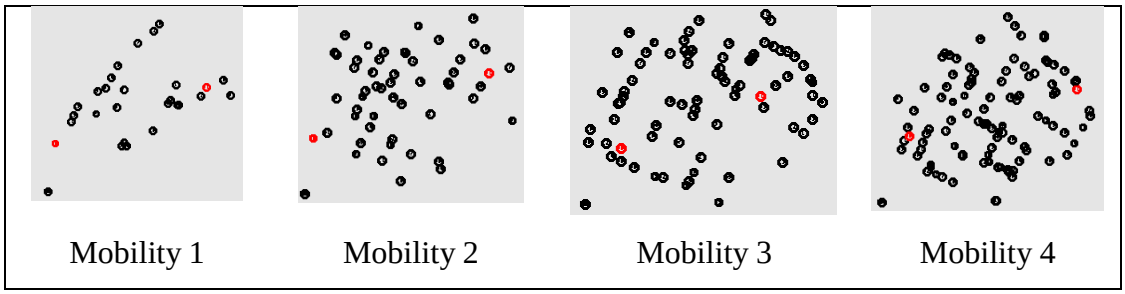
4.2.1 Skenario 1 Tanpa Static Intersection Node

Pengiriman data dilakukan dari *node* pada posisi A ke *node* pada posisi B dimana tidak terdapat koneksi secara langsung dari *node* A ke *node* B seperti tampak pada Gambar 4.6 berikut ini :



Gambar 4. 6 Posisi node Pengirim (A) & Penerima (B) pada peta 2

Pada skenario ini data dikirimkan dari *node* A ke *node* B melalui kendaraan yang berada di antara *node* A dan *node* B seperti yang tampak pada Gambar 4.7 berikut ini :



Gambar 4. 7 Skenario tanpa *Static Intersection Node* peta 2

Dari Gambar 4.7 di atas dapat dilihat posisi *node* A & *node* B (berwarna merah) serta kendaraan yang melintas diantaranya (berwarna hitam). Gambar tersebut diambil dari tampilan Network Animator pada detik ke 10. Percobaan pengiriman

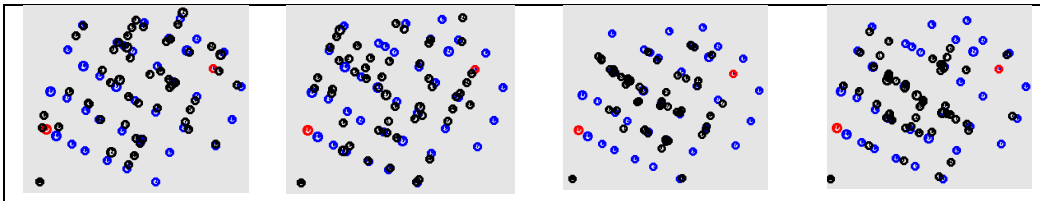
data dilakukan dengan mengirimkan 1 paket / detik selama 100 detik (100 paket dikirimkan). Hasil yang didapatkan dapat dilihat pada tabel 4.7 berikut ini :

Tabel 4. 8 Rata – Rata Hasil Percobaan tanpa *SIN* pada peta 2

	Packet Delivery Ratio (%)	**Packet Loss (%)**	**End to End Delay (s)**
Mobilitas 1	7%	83%	2,77
Mobilitas 2	86%	26%	0,90
Mobilitas 3	82%	36%	0,67
Mobilitas 4	84%	39%	0,52
Rata - Rata	**43,50%**	**64%**	**1,91**

4.2.2 Dengan *Static Intersection Node* pada setiap persimpangan jalan

Percobaan berikutnya adalah dengan menambahkan *Static Intersection Node* pada setiap persimpangan yang ada pada peta. Hal ini dilakukan untuk membuktikan bahwa penggunaan *Static Intersection Node* berkontribusi terhadap optimasi protokol AODV dalam melakukan pengiriman paket data. Hasil penelitian tersebut secara visual dapat dilihat pada Gambar 4.3 berikut ini. Gambar berikut merupakan tampilan visual dari Network Animator pada detik ke 10 di setiap mobilitas :



Mobilitas 1 Mobilitas 2 Mobilitas 3 Mobilitas 4

Gambar 4. 8 Percobaan dengan *Static Intersection Node* pada peta 2

Tampak pada percobaan di atas terdapat *Static Intersection Node* yang berfungsi sebagai *repeater* (berwarna biru) diletakkan pada setiap persimpangan jalan yang ada pada peta. Gambar di atas diambil dari skenario 1, skenario 2 dan skenario 3 pada detik ke 10. Hasil percobaan pengiriman data pada skenario tersebut dapat dilihat pada tabel 4.8 berikut ini :

Tabel 4. 9 Hasil Percobaan Dengan *SIN* di semua intersection pada peta 2

	Packet Delivery Ratio (%)	**Packet Loss (%)**	**End to End Delay (s)**
Mobilitas 1	100%	1%	0,04
Mobilitas 2	76%	41%	0,40
Mobilitas 3	78%	37%	0,79
Mobilitas 4	77%	41%	0,76
Rata - Rata	**82,75%**	**30%**	**0,50**

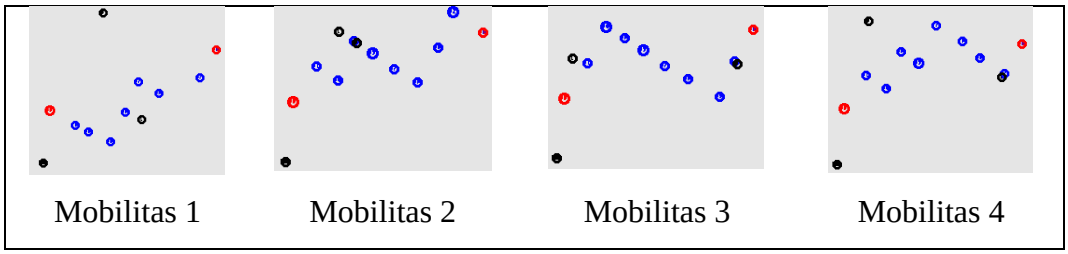
4.2.3 Dengan *Static Intersection Node* pada persimpangan jalan tertentu

Pada skenario ini digunakan algoritma Dijkstra untuk menentukan jalur dengan intensitas kendaraan terendah dari *node* A ke *node* B. Kemudian ditambahkan *Static Intersection Node* pada setiap persimpangan yang ada pada jalan tersebut. Diasumsikan dengan meletakkan *Static Intersection Node* pada persimpangan dimana tingkat kepadatan kendaraan yang rendah dapat memberikan kontribusi terhadap proses pengiriman data. Hasil pemilihan jalur dapat dilihat pada tabel 4.10 berikut ini :

Tabel 4. 10 Hasil pemilihan jalur menggunakan Dijkstra pada peta 2

Skenario1	**Skenario2**	**Skenario3**	**Skenario4**	**Skenario5**
Road R21	Road R8	Road R21	Road R5	Road R21
Road R26	Road R7	Road R26	Road R4	Road R20
Road R27	Road R22	Road R27	Road R3	Road R18
Road R29	Road R20	Road R29	Road R19	Road R30
Road R31	Road R28	Road R49	Road R28	Road R49
Road R33	Road R29	Road R58	Road R29	Road R58
Road R45	Road R31	Road R59	Road R31	Road R59
Road R44	Road R33	Road R61	Road R33	Road R61
Road R42	Road R45		Road R45	
	Road R62		Road R62	

Berikut tampilan visual dari Network Animator untuk posisi *Static Intersection Node* hasil dari algoritma Dijkstra :



Gambar 4. 9 Percobaan dengan *SIN* hasil algoritma Dijkstra pada peta 2

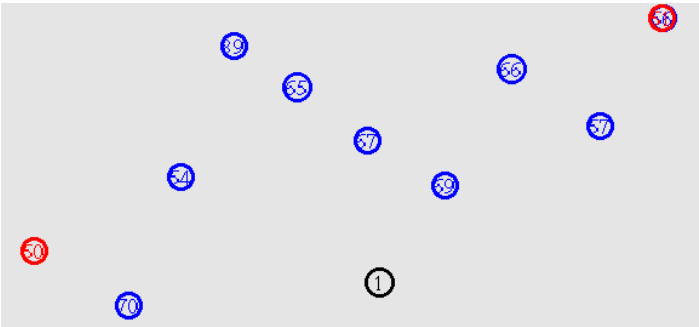
Dari hasil pemilihan jalan pada tabel 4.10 di atas berikutnya dapat ditentukan posisi *Static Intersection Node* yaitu pada setiap *instersection* yang dilalui jalur tersebut. *Intersection* yang terpilih dapat dilihat pada tabel 4.11 berikut ini :

Tabel 4. 11 Daftar *intersection* terpilih pada peta 2

Mobilitas 1	Mobilitas 2	Mobilitas 3	Mobilitas 4
56	56	56	56
57	80	57	78
83	58	83	52
69	57	69	62
67	66	67	66
65	69	59	69
89	67	86	67
54	65	88	65
81	89	70	89
68	54		54
	70		70

Dari ke 4 mobilitas tersebut diambil rata – rata posisi *Static Intersection Node* yang digunakan sehingga didapatkan posisi *Static Intersection Node* yang maksimal. Ditemukan 5 posisi *Static Intersection Node* yang paling optimal (sering

digunakan) pada peta seperti tampak pada Gambar 4.10 berikut ini. Kemudian dilakukan pecobaan pengiriman data dengan 4 mobilitas kendaraan yan berbeda :



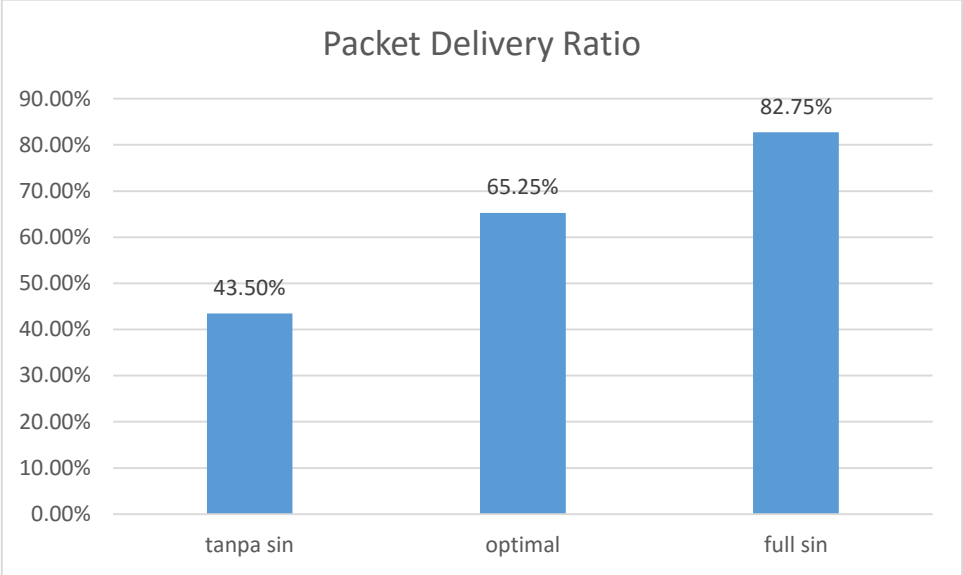
Gambar 4. 10 Posisi *Static Intersection Node* optimal pada peta 2

Tampak pada Gambar 4.10 di atas lokasi *Static Intersection Node* (warna biru) pada peta. Jumlah *Static Intersection Node* dikurangi dari total 32 titik *Static Intersection Node* menjadi hanya 8 titik *Static Intersection Node* yang optimal.

Tabel 4. 12 Rata – Rata Hasil Percobaan dengan *SIN* di *intersection* optimal pada peta 2

	Packet Delivery Ratio (%)	**Packet Loss (%)**	**End to End Delay (s)**
Skenario1	64.00%	48 %	2.83 s
Skenario2	84.00%	43 %	0.19 s
Skenario3	83.00%	34 %	0.93 s
Skenario4	65.00%	58 %	1.41 s
Skenario5	79.00%	46 %	1.05 s
Rata - Rata	**75.00%**	**46 %**	**1.28 s**

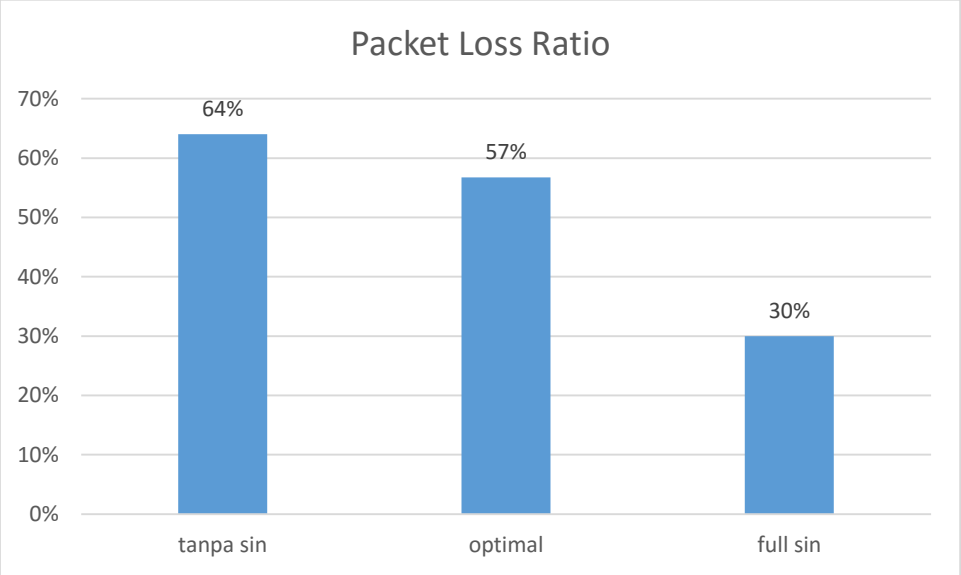
Data hasil ketiga percobaan tersebut dalam bentuk grafik dapat dilihat pada grafik – grafik berikut ini :



Grafik 4. 4 Rata – Rata *Packet Delivery Ratio* pada peta 2

Dari Grafik 4.4 di atas dapat dilihat bahwa dengan menambahkan *Static Intersection Node* pada peta dapat meningkatkan *Packet Delivery Ratio*, yang artinya semakin banyak paket data yang berhasil di kirimkan. Pada grafik tersebut juga dapat dilihat bahwa peletakan *Static Intersection Node* pada *intersection* yang memiliki tingkat kepadatan kendaraan rendah juga memiliki kontribusi yang cukup signifikan dalam meningkatkan *Packet Delivery Ratio* jika dibandingkan dengan skenario yang tanpa menggunakan *Static Intersection Node*.

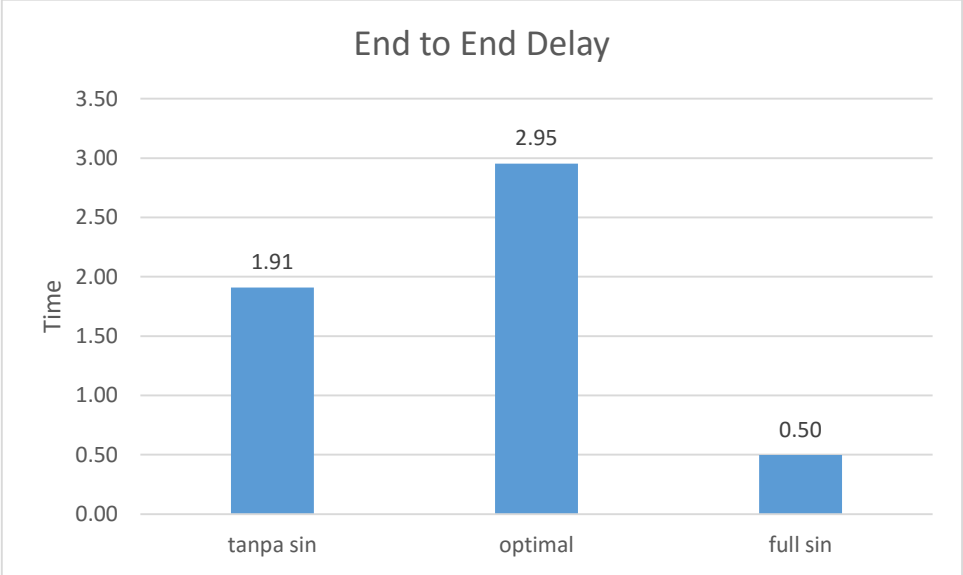
Faktor lain yang menjadi parameter pada penelitian ini adalah *Packet Loss Ratio*. *Packet Loss Ratio* adalah rasio jumlah paket yang gagal di terima oleh *receiver* dibandingkan dengan keseluruhan paket data yang dikirimkan. Data tersebut disajikan pada Grafik 4.5 berikut ini :



Grafik 4. 5 Rata – rata *Packet Loss* pada peta 2

Dari grafik 4.5 di atas dapat diketahui bahwa peletakan *Static Intersection Node* pada peta dapat menurunkan *Packet Loss* yaitu jumlah paket data yang gagal diterima oleh *node* B. Penyebab terjadinya *Packet Loss* cukup beragam selain perubahan topologi jaringan karena mobilitas kendaraan, juga dapat disebabkan karena *buffer* pada memori kendaraan / *Static Intersection Node* yang dilalui paket data penuh sehingga tidak dapat menampung paket baru sehingga ada paket data yang harus di *drop*. Selain itu juga dapat dipengaruhi dari *bandwidth* yang terbatas sehingga tidak semua paket dapat terkirim ke kendaraan berikutnya.

Parameter lain yang diukur pada penelitian ini adalah *End to End Delay* yaitu rata – rata waktu yang dibutuhkan oleh paket data untuk sampai kepada *receiver*. Data *End to End Delay* pada peta 2 ini dapat dilihat pada Grafik 4.6 berikut ini :



Grafik 4. 6 Rata – Rata *End to End Delay* pada peta *2*

Pada Grafik 4.6 di atas dapat diketahui bahwa *Static Intersection Node* memiliki kontribusi yang cukup signifikan dalam menekan *End to End Delay* yang artinya semakin kecil nilai *End to End Delay* maka semakin cepat data sampai ke tujuan.

[Halaman ini sengaja di kosongkan]

LAMPIRAN

Tabel panjang setiap ruas jalan pada peta 1 beserta ID ruas jalan

ROAD	ID	LENGTH	ROAD	ID	LENGTH
A	169134391#5	225,69	AA	169134388#2	27,3
B	169134391#4	122,89	AB	--734#0	98,89
C	-704#0	100,22	AC	-169134388#1	64,05
D	-704#1	125,45	AD	--440#1	96,9
E	-734#3	48,69	AE	-169134388#0	48,14
F	-704#2	112,5	AF	169134391#1	91,8
G	169134390#1	77,37	AG	169134390#4	45,5
H	-685#3	100,33	AH	450	36,15
I	-442#0	74,35	AI	169134396	91,66
J	-685#2	125,39	AJ	-169134485#0	41,98
K	--734#2	74,35	AK	169134438#1	73,25
L	--685#1	64,09	AL	-169134487#0	98,87
M	--685#0	48,67	AM	169134438#2	91,35
N	169134391#3	82,71	AN	169134499#0	98,73
O	169134390#2	76,61	AO	169134391#0	142,09
P	--448#0	100,39	AP	169134390#5	95,57
Q	-442#1	75,92	AQ	-169134468#0	54,69
R	--448#1	125,34	AR	-169134438#0	93,09
S	--734#1	77,76	AS	-169134468#1	43,29
T	--448#2	63,88	AT	194733940#1	93,15
U	-440#0	72,69	AU	194733940#2	75,96
V	-448#3	48,61	AV	169134487#1	45,03
W	169134391#2	75,17	AW	194733940#3	97,3
X	169134390#3	95,13	AX	-169134499#1	44,75
Y	169134388#4	127,54	AY	169134487#1	44,39
Z	169134388#3	71,49	AZ	169134485#1	56,88

Tabel kepadatan kendaraan pada peta 1 mobilitas 1 (25 kendaraan)

ROAD	LENGTH	Jml. KEND	DENSITY	ROAD	LENGTH	Jml. KEND	DENSITY
A	225,69	1,0	0,0044	AA	27,30	7,0	0,2564
B	122,89	0,1	0,0008	AB	98,89	1,0	0,0101
C	100,22	2,0	0,0200	AC	64,05	5,0	0,0781
D	125,45	3,0	0,0239	AD	96,90	1,0	0,0103
E	48,69	1,0	0,0205	AE	48,14	6,0	0,1246
F	112,5	3,0	0,0267	AF	91,80	5,0	0,0545
G	77,37	4,0	0,0517	AG	45,50	1,0	0,0220

H	100,33	6,0	0,0598	AH	36,15	0,1	0,0028
I	74,35	2,0	0,0269	AI	91,66	2,0	0,0218
J	125,39	5,0	0,0399	AJ	41,98	3,0	0,0715
K	74,35	1,0	0,0134	AK	73,25	1,0	0,0137
L	64,09	2,0	0,0312	AL	98,87	1,0	0,0101
M	48,67	1,0	0,0205	AM	91,35	0,1	0,0011
N	82,71	3,0	0,0363	AN	98,73	3,0	0,0304
O	76,61	4,0	0,0522	AO	142,09	0,1	0,0007
P	100,39	1,0	0,0100	AP	95,57	3,0	0,0314
Q	75,92	3,0	0,0395	AQ	54,69	1,0	0,0183
R	125,34	3,0	0,0239	AR	93,09	3,0	0,0322
S	77,76	1,0	0,0129	AS	43,29	0,1	0,0023
T	63,88	4,0	0,0626	AT	93,15	6,0	0,0644
U	72,69	2,0	0,0275	AU	75,96	2,0	0,0263
V	48,61	2,0	0,0411	AV	45,03	3,0	0,0666
W	75,17	5,0	0,0665	AW	97,30	1,0	0,0103
X	95,13	3,0	0,0315	AX	44,75	1,0	0,0223
Y	127,54	3,0	0,0235	AY	44,39	3,0	0,0676
Z	71,49	6,0	0,0839	AZ	56,88	4,0	0,0703

Tabel kepadatan kendaraan pada peta 1 mobilitas 2 (50 kendaraan)

ROAD	LENGTH	Jml. KEND	DENSITY	ROAD	LENGTH	Jml. KEND	DENSITY
A	225,69	1,0	0,0044	AA	27,30	22,0	0,8059
B	122,89	4,0	0,0325	AB	98,89	3,0	0,0303
C	100,22	9,0	0,0898	AC	64,05	16,0	0,2498
D	125,45	7,0	0,0558	AD	96,90	1,0	0,0103
E	48,69	6,0	0,1232	AE	48,14	17,0	0,3531
F	112,5	9,0	0,0800	AF	91,80	15,0	0,1634
G	77,37	7,0	0,0905	AG	45,50	6,0	0,1319
H	100,33	5,0	0,0498	AH	36,15	0,1	0,0028
I	74,35	3,0	0,0403	AI	91,66	5,0	0,0545
J	125,39	6,0	0,0479	AJ	41,98	11,0	0,2620
K	74,35	2,0	0,0269	AK	73,25	8,0	0,1092
L	64,09	4,0	0,0624	AL	98,87	8,0	0,0809
M	48,67	2,0	0,0411	AM	91,35	5,0	0,0547
N	82,71	11,0	0,1330	AN	98,73	5,0	0,0506
O	76,61	4,0	0,0522	AO	142,09	0,1	0,0007
P	100,39	5,0	0,0498	AP	95,57	7,0	0,0732
Q	75,92	3,0	0,0395	AQ	54,69	1,0	0,0183
R	125,34	4,0	0,0319	AR	93,09	0,1	0,0011
S	77,76	2,0	0,0257	AS	43,29	2,0	0,0462

T	63,88	5,0	0,0783	AT	93,15	6,0	0,0644
U	72,69	6,0	0,0825	AU	75,96	3,0	0,0395
V	48,61	5,0	0,1029	AV	45,03	4,0	0,0888
W	75,17	12,0	0,1596	AW	97,30	2,0	0,0206
X	95,13	6,0	0,0631	AX	44,75	3,0	0,0670
Y	127,54	7,0	0,0549	AY	44,39	4,0	0,0901
Z	71,49	16,0	0,2238	AZ	56,88	8,0	0,1406

Tabel kepadatan kendaraan pada peta 1 mobilitas 3 (75 kendaraan)

ROAD	LENGTH	Jml. KEND	DENSITY	ROAD	LENGTH	Jml. KEND	DENSITY
A	225,69	3,0	0,0133	AA	27,3	21,0	0,7692
B	122,89	3,0	0,0244	AB	98,89	4,0	0,0404
C	100,22	3,0	0,0299	AC	64,05	13,0	0,2030
D	125,45	6,0	0,0478	AD	96,9	1,0	0,0103
E	48,69	6,0	0,1232	AE	48,14	13,0	0,2700
F	112,5	10,0	0,0889	AF	91,8	19,0	0,2070
G	77,37	4,0	0,0517	AG	45,5	8,0	0,1758
H	100,33	4,0	0,0399	AH	36,15	3,0	0,0830
I	74,35	5,0	0,0672	AI	91,66	3,0	0,0327
J	125,39	10,0	0,0798	AJ	41,98	8,0	0,1906
K	74,35	6,0	0,0807	AK	73,25	9,0	0,1229
L	64,09	9,0	0,1404	AL	98,87	7,0	0,0708
M	48,67	0,1	0,0021	AM	91,35	4,0	0,0438
N	82,71	8,0	0,0967	AN	98,73	6,0	0,0608
O	76,61	6,0	0,0783	AO	142,09	3,0	0,0211
P	100,39	9,0	0,0897	AP	95,57	7,0	0,0732
Q	75,92	9,0	0,1185	AQ	54,69	3,0	0,0549
R	125,34	9,0	0,0718	AR	93,09	1,0	0,0107
S	77,76	2,0	0,0257	AS	43,29	2,0	0,0462
T	63,88	5,0	0,0783	AT	93,15	8,0	0,0859
U	72,69	10,0	0,1376	AU	75,96	4,0	0,0527
V	48,61	10,0	0,2057	AV	45,03	5,0	0,1110
W	75,17	12,0	0,1596	AW	97,3	4,0	0,0411
X	95,13	12,0	0,1261	AX	44,75	3,0	0,0670
Y	127,54	7,0	0,0549	AY	44,39	4,0	0,0901
Z	71,49	15,0	0,2098	AZ	56,88	14,0	0,2461

Tabel kepadatan kendaraan pada peta 1 mobilitas 4 (100 kendaraan)

ROAD	LENGTH	Jml. KEND	DENSITY	ROAD	LENGTH	Jml. KEND	DENSITY
A	225,69	7,0	0,0310	AA	27,3	32,0	1,1722
B	122,89	2,0	0,0163	AB	98,89	11,0	0,1112
C	100,22	10,0	0,0998	AC	64,05	18,0	0,2810
D	125,45	13,0	0,1036	AD	96,9	2,0	0,0206
E	48,69	9,0	0,1848	AE	48,14	15,0	0,3116
F	112,50	8,0	0,0711	AF	91,80	19,0	0,2070
G	77,37	13,0	0,1680	AG	45,50	11,0	0,2418
H	100,33	13,0	0,1296	AH	36,15	6,0	0,1660
I	74,35	6,0	0,0807	AI	91,66	8,0	0,0873
J	125,39	10,0	0,0798	AJ	41,98	13,0	0,3097
K	74,35	8,0	0,1076	AK	73,25	5,0	0,0683
L	64,09	9,0	0,1404	AL	98,87	4,0	0,0405
M	48,67	3,0	0,0616	AM	91,35	11,0	0,1204
N	82,71	11,0	0,1330	AN	98,73	9,0	0,0912
O	76,61	19,0	0,2480	AO	142,09	7,0	0,0493
P	100,39	5,0	0,0498	AP	95,57	6,0	0,0628
Q	75,92	8,0	0,1054	AQ	54,69	6,0	0,1097
R	125,34	6,0	0,0479	AR	93,09	6,0	0,0645
S	77,76	6,0	0,0772	AS	43,29	3,0	0,0693
T	63,88	8,0	0,1252	AT	93,15	7,0	0,0751
U	72,69	15,0	0,2064	AU	75,96	8,0	0,1053
V	48,61	16,0	0,3292	AV	45,03	9,0	0,1999
W	75,17	16,0	0,2129	AW	97,30	10,0	0,1028
X	95,13	19,0	0,1997	AX	44,75	7,0	0,1564
Y	127,54	20,0	0,1568	AY	44,39	9,0	0,2027
Z	71,49	25,0	0,3497	AZ	56,88	18,0	0,3165

Tabel panjang setiap ruas jalan pada peta 2 beserta ID ruas jalan

ROAD	**ID**	**LENGTH**	**ROAD**	**ID**	**LENGTH**
R1	152621886#0	107,19	R34	253996602#0	111,85
R2	152621664#5	78,34	R35	152621675#1	55,34
R3	152621664#4	43,5	R36	253996597#0	109,22
R4	152621883#0	88,26	R37	152621675#0	67
R5	152621905#2	57,22	R38	253996579#2	84,66
R6	152621883#1	116,18	R39	253996592#3	50,53
R7	252125753#0	99,82	R40	253996592#2	21,82
R8	152621911#0	98,77	R41	253996579#1	134,36
R9	152621911#1	92,97	R42	253996584	50,20

R10	152621911#2	65,84	R43	253996597#1	107,68
R11	152621664#6	68,54	R44	152621902#3	66,54
R12	152621911#3	86,45	R45	253996602#1	110,27
R13	152621911#4	91,84	R46	152621902#2	47,96
R14	253996592#1	30,05	R47	253996615#1	110,91
R15	152621886#1	91,61	R48	152621902#1	74,87
R16	253996592#0	85,97	R49	152621664#9	97,75
R17	152621664#7	87,88	R50	152621902#0	63,95
R18	152621908#2	66,49	R51	152621664#1	92,40
R19	152621664#3	100,32	R52	305	91,88
R20	152621908#1	81,92	R53	303	85,28
R21	152621905#1	97,82	R54	252125746	96,73
R22	152621908#0	123,3	R55	299	94,44
R24	252125753#1	92,11	R56	152621664#0	95,62
R25	301	107,41	R57	253996589#3	64,04
R26	152621905#0	98,38	R58	152621664#10	97,65
R27	152621658#1	95,04	R59	253996589#2	70,41
R28	152621664#2	104,67	R60	253996615#0	102
R29	152621658#0	69,54	R61	253996589#1	43,26
R30	152621664#8	122,42	R62	253996602#2	108,93
R31	152621675#3	69,03	R63	253996589#0	51,1
R32	253996615#2	114,23	R64	253996598	116,36
R33	152621675#2	55,34	R65	253996579#0	131,76
			R66	253996592#4	65,12

Tabel kepadatan kendaraan pada peta 2 mobilitas 1 (25 kendaraan)

ROAD	KENDARAAN	DENSITY	ROAD	KENDARAAN	DENSITY
R1	3	0,0280	R34	3	0,0268
R2	0,1	0,0013	R35	5	0,0904
R3	1	0,0230	R36	0,1	0,0009
R4	1	0,0113	R37	2	0,0299
R5	1	0,0175	R38	0,1	0,0012
R6	2	0,0172	R39	2	0,0396
R7	1	0,0100	R40	0,1	0,0046
R8	0,1	0,0010	R41	1	0,0074
R9	1	0,0108	R42	3	0,0598
R10	3	0,0456	R43	3	0,0279
R11	2	0,0292	R44	2	0,0301
R12	1	0,0116	R45	2	0,0181
R13	1	0,0109	R46	3	0,0626
R14	2	0,0666	R47	2	0,0180

R15	1	0,0109	R48	2	0,0267
R16	1	0,0116	R49	4	0,0409
R17	2	0,0228	R50	3	0,0469
R18	1	0,0150	R51	5	0,0541
R19	2	0,0199	R52	1	0,0109
R20	4	0,0488	R53	6	0,0704
R21	2	0,0204	R54	5	0,0517
R22	2	0,0162	R55	4	0,0424
R24	3	0,0326	R56	4	0,0418
R25	2	0,0186	R57	3	0,0468
R26	3	0,0305	R58	3	0,0307
R27	6	0,0631	R59	1	0,0142
R28	5	0,0478	R60	0,1	0,0010
R29	5	0,0719	R61	1	0,0231
R30	3	0,0245	R62	1	0,0092
R31	6	0,0869	R63	0,1	0,0020
R32	5	0,0438	R64	1	0,0086
R33	6	0,1084	R65	0,1	0,0008
			R66	0,1	0,0015

Tabel kepadatan kendaraan pada peta 2 mobilitas 2 (50 kendaraan)

ROAD	KENDARAAN	DENSITY	ROAD	KENDARAAN	DENSITY
R1	0,1	0,0009	R34	4	0,0358
R2	2	0,0255	R35	7	0,1265
R3	3	0,0690	R36	6	0,0549
R4	4	0,0453	R37	5	0,0746
R5	3	0,0524	R38	7	0,0827
R6	1	0,0086	R39	7	0,1385
R7	2	0,0200	R40	8	0,3666
R8	2	0,0202	R41	2	0,0149
R9	2	0,0215	R42	2	0,0398
R10	5	0,0759	R43	1	0,0093
R11	1	0,0146	R44	4	0,0601
R12	3	0,0347	R45	7	0,0635
R13	0,1	0,0011	R46	4	0,0834
R14	8	0,2662	R47	3	0,0270
R15	4	0,0437	R48	4	0,0534
R16	9	0,1047	R49	8	0,0818
R17	6	0,0683	R50	2	0,0313
R18	5	0,0752	R51	5	0,0541
R19	5	0,0498	R52	2	0,0218

R20	6	0,0732	R53	2	0,0235
R21	6	0,0613	R54	2	0,0207
R22	1	0,0081	R55	2	0,0212
R24	3	0,0326	R56	3	0,0314
R25	4	0,0372	R57	4	0,0625
R26	4	0,0407	R58	4	0,0410
R27	5	0,0526	R59	4	0,0568
R28	2	0,0191	R60	4	0,0392
R29	7	0,1007	R61	3	0,0693
R30	8	0,0653	R62	3	0,0275
R31	16	0,2318	R63	3	0,0587
R32	7	0,0613	R64	1	0,0086
R33	8	0,1446	R65	2	0,0152
			R66	6	0,0921

Tabel kepadatan kendaraan pada peta 2 mobilitas 3 (75 kendaraan)

ROAD	KENDARAAN	DENSITY	ROAD	KENDARAAN	DENSITY
R1	2	0,0187	R34	4	0,0358
R2	2	0,0255	R35	14	0,2530
R3	9	0,2069	R36	6	0,0549
R4	9	0,1020	R37	7	0,1045
R5	7	0,1223	R38	4	0,0472
R6	6	0,0516	R39	7	0,1385
R7	5	0,0501	R40	10	0,4583
R8	4	0,0405	R41	4	0,0298
R9	5	0,0538	R42	4	0,0797
R10	7	0,1063	R43	10	0,0929
R11	2	0,0292	R44	6	0,0902
R12	4	0,0463	R45	7	0,0635
R13	3	0,0327	R46	2	0,0417
R14	8	0,2662	R47	6	0,0541
R15	3	0,0327	R48	3	0,0401
R16	9	0,1047	R49	11	0,1125
R17	7	0,0797	R50	6	0,0938
R18	8	0,1203	R51	20	0,2165
R19	8	0,0797	R52	3	0,0327
R20	3	0,0366	R53	3	0,0352
R21	10	0,1022	R54	4	0,0414
R22	4	0,0324	R55	4	0,0424
R24	6	0,0651	R56	10	0,1046
R25	5	0,0466	R57	6	0,0937

R26	9	0,0915	R58	7	0,0717
R27	11	0,1157	R59	8	0,1136
R28	11	0,1051	R60	2	0,0196
R29	21	0,3020	R61	6	0,1387
R30	5	0,0408	R62	3	0,0275
R31	22	0,3187	R63	5	0,0978
R32	8	0,0700	R64	3	0,0258
R33	18	0,3253	R65	2	0,0152
			R66	5	0,0768

Tabel kepadatan kendaraan pada peta 2 mobilitas 4 (100 kendaraan)

ROAD	KENDARAAN	DENSITY	ROAD	KENDARAAN	DENSITY
R1	3	0,0280	R34	6	0,0536
R2	2	0,0255	R35	21	0,3795
R3	5	0,1149	R36	4	0,0366
R4	7	0,0793	R37	6	0,0896
R5	5	0,0874	R38	3	0,0354
R6	3	0,0258	R39	8	0,1583
R7	6	0,0601	R40	9	0,4125
R8	4	0,0405	R41	5	0,0372
R9	6	0,0645	R42	3	0,0598
R10	9	0,1367	R43	7	0,0650
R11	3	0,0438	R44	9	0,1353
R12	2	0,0231	R45	9	0,0816
R13	3	0,0327	R46	11	0,2294
R14	11	0,3661	R47	10	0,0902
R15	9	0,0982	R48	14	0,1870
R16	9	0,1047	R49	8	0,0818
R17	11	0,1252	R50	12	0,1876
R18	13	0,1955	R51	20	0,2165
R19	10	0,0997	R52	15	0,1633
R20	7	0,0854	R53	12	0,1407
R21	8	0,0818	R54	9	0,0930
R22	7	0,0568	R55	7	0,0741
R24	12	0,1303	R56	9	0,0941
R25	12	0,1117	R57	8	0,1249
R26	9	0,0915	R58	9	0,0922
R27	27	0,2841	R59	12	0,1704
R28	18	0,1720	R60	3	0,0294
R29	35	0,5033	R61	12	0,2774
R30	7	0,0572	R62	10	0,0918

R31	33	0,4781	R63	1	0,0196
R32	16	0,1401	R64	6	0,0516
R33	28	0,5060	R65	2	0,0152
			R66	5	0,0768

Kode program untuk membuat mobilitas

```
netconvert --osm-files map.osm -o map.net.xml --no-turnarounds
python ../tools/randomTrips.py -n map.net.xml -e 1 -o trip.trips.xml
-p 0.04
duarouter -n map.net.xml -t trip.trips.xml -o route.rou.xml -s 500
--remove-loops --repair --ignore-errors
sumo -c skenario.sumo.cfg --fcd-output skenario.xml
python   ../tools/traceExporter.py   --fcd-input   skenario.xml   --
ns2mobility-output mobility.tcl
```

Kode program hasil mobilitas (mobility.tcl)

```
$node_(0) set X_ 184.77
$node_(0) set Y_ 501.99
$node_(0) set Z_ 0
$ns_ at 0.0 "$node_(0) setdest 184.77 501.99 0.00"
$node_(1) set X_ 151.29
$node_(1) set Y_ 200.27
$node_(1) set Z_ 0
$ns_ at 0.0 "$node_(1) setdest 151.29 200.27 0.00"
$ns_ at 1.0 "$node_(0) setdest 184.06 500.73 1.44"
$ns_ at 1.0 "$node_(1) setdest 149.81 201.07 1.67"
$node_(2) set X_ 327.55
$node_(2) set Y_ 159.34
$node_(2) set Z_ 0
$ns_ at 1.0 "$node_(2) setdest 327.55 159.34 0.00"
$node_(3) set X_ 118.31
$node_(3) set Y_ 98.89
$node_(3) set Z_ 0
$ns_ at 1.0 "$node_(3) setdest 118.31 98.89 0.00"
$node_(4) set X_ 382.66
$node_(4) set Y_ 393.5
$node_(4) set Z_ 0
$ns_ at 1.0 "$node_(4) setdest 382.66 393.5 0.00"
$node_(5) set X_ 124.3
$node_(5) set Y_ 150.47
$node_(5) set Z_ 0
$ns_ at 1.0 "$node_(5) setdest 124.3 150.47 0.00"
$node_(6) set X_ 109.1
$node_(6) set Y_ 368.02
$node_(6) set Z_ 0
$ns_ at 1.0 "$node_(6) setdest 109.1 368.02 0.00"
$node_(7) set X_ 123.54
$node_(7) set Y_ 369.7
$node_(7) set Z_ 0
$ns_ at 1.0 "$node_(7) setdest 123.54 369.7 0.00"
$node_(8) set X_ 212.93
$node_(8) set Y_ 324.65
$node_(8) set Z_ 0
```

Kode program koordinat *Static Intersection Node*

```
#RSU
$node_(52) set X_ 368.83
$node_(52) set Y_ 235.99
$node_(52) set Z_ 0
$node_(52) color "blue"
$ns_ at 0.0 "$node_(52) setdest 368.83 235.99 0"
$ns_ at 0.0 "$node_(52) color blue"
$node_(53) set X_ 191.03
$node_(53) set Y_ 509.71
$node_(53) set Z_ 0
$node_(53) color "blue"
$ns_ at 0.0 "$node_(53) setdest 191.03 509.71 0"
$ns_ at 0.0 "$node_(53) color blue"
$node_(54) set X_ 176.75
$node_(54) set Y_ 67.72
$node_(54) set Z_ 0
$node_(54) color "blue"
$ns_ at 0.0 "$node_(54) setdest 176.75 67.72 0"
$ns_ at 0.0 "$node_(54) color blue"
$node_(55) set X_ 110.70
$node_(55) set Y_ 105.24
$node_(55) set Z_ 0
$node_(55) color "blue"
$ns_ at 0.0 "$node_(55) setdest 110.70 105.24 0"
$ns_ at 0.0 "$node_(55) color blue"
$node_(56) set X_ 152.98
$node_(56) set Y_ 442.34
$node_(56) set Z_ 0
$node_(56) color "blue"
$ns_ at 0.0 "$node_(56) setdest 152.98 442.34 0"
$ns_ at 0.0 "$node_(56) color blue"
$node_(57) set X_ 445.65
$node_(57) set Y_ 273.31
$node_(57) set Z_ 0
$node_(57) color "blue"
$ns_ at 0.0 "$node_(57) setdest 445.65 273.31 0"
$ns_ at 0.0 "$node_(57) color blue"
$node_(58) set X_ 352.36
$node_(58) set Y_ 336.58
$node_(58) set Z_ 0
$node_(58) color "blue"
$ns_ at 0.0 "$node_(58) setdest 352.36 336.58 0"
$ns_ at 0.0 "$node_(58) color blue"
$node_(59) set X_ 408.53
$node_(59) set Y_ 207.94
$node_(59) set Z_ 0
$node_(59) color "blue"
$ns_ at 0.0 "$node_(59) setdest 408.53 207.94 0"
$ns_ at 0.0 "$node_(59) color blue"
$node_(60) set X_ 45.92
$node_(60) set Y_ 253.30
$node_(60) set Z_ 0
$node_(60) color "blue"
```

Kode program untuk mengirimkan paket data

```
#preset
set val(chan)           Channel/WirelessChannel    ;#Channel Type
set val(prop)            Propagation/TwoRayGround   ;# radio-propagation model
set val(netif)           Phy/WirelessPhy            ;# network interface type
set val(mac)             Mac/802_11                 ;# MAC type
set val(ifq)             Queue/DropTail/PriQueue    ;# interface queue type
set val(ll)              LL                         ;# link layer type
set val(ant)             Antenna/OmniAntenna        ;# antenna model
set val(ifqlen)          50                         ;# max packet in ifq
set val(nn)              150                        ;# number of mobilenodes
set val(rp)              AODV                       ;# routing protocol
set val(x)               600                        ;# X dimension of topography
set val(y)               600                        ;# Y dimension of topography
set val(stop)            100                    ;# time of simulation end

#simulatornya
set ns_ [new Simulator]

set f [open 100.tr w]
$ns_ trace-all $f
set nf [open 100.nam w]
$ns_ namtrace-all-wireless $nf $val(x) $val(y)

$ns_ use-newtrace

# set up topography object
set topo       [new Topography]

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

#
# Create God
#

create-god $val(nn)

set chan_1_ [new $val(chan)]

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

```
#limiting wifi range
Phy/WirelessPhy set CPThresh_ 10.0
Phy/WirelessPhy set CSThresh_ 1.42681e-08 ;#100m
Phy/WirelessPhy set RXThresh_ 1.42681e-08 ;#100m
Phy/WirelessPhy set bandwidth_ 10M
Phy/WirelessPhy set Pt_ 0.281838
Phy/WirelessPhy set freq_ 9.14e+08
Phy/WirelessPhy set L_ 1

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

#initial position
for {set i 0} {$i < $val(nn) } { incr i } {
            $ns_ initial_node_pos $node_($i) 20}


# load mobility
source "../mobility/100/mobility.tcl"

#load sin position
#source "../sin/djsin.tcl"

#sender position
$node_(50) set X_ 301
$node_(50) set Y_ 2
$node_(50) set Z_ 0
$ns_ at 0.0 "$node_(50) setdest 301 2 0"
$node_(50) color "red"
$ns_ at 0.0 "$node_(50) color red"

#receiver position
$node_(51) set X_ 216.48
$node_(51) set Y_ 554.10
$node_(51) set Z_ 0
$ns_ at 0.0 "$node_(51) setdest 216.48 554.10 0"
$node_(51) color "red"
$ns_ at 0.0 "$node_(51) color red"

# Set a UDP connection
set UDP [new Agent/UDP]
set null [new Agent/Null]
$ns_ attach-agent $node_(50) $UDP
$ns_ attach-agent $node_(51) $null
$ns_ connect $UDP $null
$UDP set fid_ 2

# Set a CBR over UDP
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $UDP
$cbr set type_ cbr
$cbr set packetSize_ 100
$cbr set interval_ 1
$ns_ at 0.0 "$cbr start"
```

Kode program untuk memeriksa hasil pengiriman data

```
BEGIN {
print("\n\n******** Network Statistics ********\n");

# Change array size from 50 to any number of nodes for which u are doing simulation.
#  i.e.  change  values  of  arrays  packet_sent,  packet_drop,  packet_recvd,
packet_forwarded, energy_left,
packet_sent[133] = 100;
packet_drop[133] = 100;
packet_recvd[133] = 100;
packet_forwarded[133] = 100;

# Change energy assigned to initial node (as per your simulation tcl file)
# Initial Energy assigned to each node in Joules

energy_left[133] = 100;

total_pkt_sent=0;
total_pkt_recvd=0;
total_pkt_drop=0;
total_pkt_forwarded=0;
pkt_delivery_ratio = 0;
total_hop_count = 0;
avg_hop_count = 0;
overhead = 0;
start = 0.000000000;
end = 0.000000000;
packet_duration = 0.0000000000;
recvnum = 0;
delay = 0.000000000;
sum = 0.000000000;
i=0;
total_energy_consumed = 0.000000;
}

{
state       =     $1;
time        =     $3;

# For energy consumption statistics see trace file
node_num    =     $5;
energy_level      =     $7;


node_id     =     $9;
level             =     $19;
pkt_type    =     $35;
packet_id   =     $41;
no_of_forwards    =     $49;

# In for loop change values from 50 to number of nodes that u specify for your
simulation
```

```
if((pkt_type == "cbr") && (state == "s") && (level=="AGT")) {
      for(i=0;i<133;i++) {
            if(i == node_id) {
            packet_sent[i] = packet_sent[i] + 1; }
}
}else if((pkt_type == "cbr") && (state == "r") && (level=="AGT")) {
      for(i=0;i<133;i++) {
            if(i == node_id) {
            packet_recvd[i] = packet_recvd[i] + 1; }
}
}else if((pkt_type == "cbr") && (state == "d")) {
      for(i=0;i<133;i++) {
            if(i == node_id) {
            packet_drop[i] = packet_drop[i] + 1; }
}
}else if((pkt_type == "cbr") && (state == "f")) {
      for(i=0;i<133;i++) {
            if(i == node_id) {
            packet_forwarded[i] = packet_forwarded[i] + 1; }
}
}

# To calculate total hop counts
if ((state == "r") && (level == "RTR") && (pkt_type == "cbr")) { total_hop_count
= total_hop_count + no_of_forwards; }

# Routing Overhead
if ((state == "s" || state == "f") && (level == "RTR") && (pkt_type == "message"))
{ overhead = overhead + 1; }

# Calculating Average End to End Delay

if ( start_time[packet_id] == 0 )  { start_time[packet_id] = time; }

if (( state == "s") &&   ( pkt_type == "cbr" ) && ( level == "AGT" ))   {
start_time[packet_id] = time; }

 if (( state == "r") &&   ( pkt_type == "cbr" ) && ( level == "AGT" )) {
end_time[packet_id] = time;  }
 else {  end_time[packet_id] = -1;  }

# To Calculate Average Energy Consumption

# Change number of nodes in this for loop also

if(state == "N") {
      for(i=0;i<133;i++) {
            if(i == node_num) {
                              energy_left[i] = energy_left[i] - (energy_left[i] -
energy_level);
                  }

                    }
}

}
```