TUGAS AKHIR - KI141502

ANALISIS KINERJA ANTHOCNET PADA LINGKUNGAN VANETS

DEWI KARTIKA
NRP 5113100008

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

Dosen Pembimbing II
Prof. Ir. Supeno Djanali M.Sc, Ph.D.

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya, 2018

[Halaman ini sengaja dikosongkan]

TUGAS AKHIR – KI141330

ANALISIS KINERJA ANTHOCNET PADA LINGKUNGAN VANETS

DEWI KARTIKA P
NRP 5113100008

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

Dosen Pembimbing II
Prof.Ir. Supeno Djanali, M.Sc. Ph.D.

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2018

[Halaman ini sengaja dikosongkan]

UNDERGRADUATE THESIS – KI141502

PERFORMANCE ANALYSIS OF ANTHOCNET ON VANETS ENVIRONMENT

DEWI KARTIKA P
NRP 5113100008

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

Second Advisor
Prof.Ir. Supeno Djanali, M.Sc. Ph.D.

Department of Informaticn
Faculty of Information Technology and Communication
Tenth November Institute of Technology
Surabaya 2018

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

ANALISIS KINERJA ANTHOCNET PADA LINGKUNGAN VANETS

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Arsitektur dan Jaringan Komputer
Program Studi S1 Departemen Informatika
Fakultas Teknologi Informasi dan Komunikasi
Insttitut Teknologi Sepuluh November

Oleh:
DEWI KARTIKA PRASETYAWATI
NRP. 5113100008

Disetujui oleh Pembimbing Tugas Akhir:

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

2. Prof. Ir. Supeno Djanali, M.Sc., Ph.D. .....
   NIP. 194806191973011001 (Pembimbing II)

SURABAYA
JANUARI, 2018

vii

[Halaman ini sengaja dikosongkan]

ANALISIS KINERJA ANTHOCNET PADA LINGKUNGAN PADA VANETS

Nama : Dewi Kartika Prasetyawati
NRP : 5113100008
Jurusan : Informatika FTIK-ITS
Pembimbing 1 : Dr.Eng. Radityo Anggoro, S.Kom., M.Sc.
Pembimbing 2 : Prof. Ir. Supeno Djanali, M.Sc., Ph.D.

Abstrak

AODV merupakan salah satu routing protocol yang popular karena proses route discovery yang efisien. Namun jika diterapkan pada lingkungan VANET yang memiliki topologi dinamis dan berubah dengan cepat, AODV kurang mampu menjaga kestabilan dalam komunikasi antar kendaraan. Hal ini tentu disebabkan oleh AODV yang pada dasarnya diciptakan untuk lingkungan VANET.

AntHocNet merupakan jalan terbaru untuk pembelajaran penyesuaian tabel routing pada jaringan komunikasi. AntHocNet membagikan mobile agent berbasiskan algoritma Ant Colony Optimization yang digunakan untuk adanya informasi mengenai jaringan kendaraan, seperti posisi kendaraan dan kecepatan, agar desain algoritma semut dapat menunjukkan performa yang baik pada jaringan yang dinamis. Pada Tugas Akhir ini, AntHocNet menunjukkan performa yang baik pada VANET.

Dari hasil uji yang dilakukan, AntHoceNet memiliki performa yang lebih baik dibandingkan dengan AODV. Seperti peningkatan nilai rata-rata packet delivery ratio, delay dan routing overhead.

Kata kunci: AODV, AntHocNet, NS-2, dan VANETs

[Halaman ini sengaja dikosongkan]

PERFORMANCE ANALISYS OF ANTHOCNET ON VANETS ENVIRONMENT

Nama	**: Dewi Kartika Prasetyawati**
NRP	**: 51131000008**
Jurusan	**: Informatika FTIK-ITS**
Pembimbing 1	**: Dr.Eng.Radityo Anggoro, S.Kom., M.Sc.**
Pembimbing 2	**: Prof. Ir. Supeno Djanali, M.Sc., Ph.D.**

Abstract

AODV is one of the popular routing protocol because of its efficiency. In route discovery process. However if applied in VANET environment which topology is dynamic and changing rapidly, AODV is less able to maintain is stability in vehicle-to-vehicle communication. This is certainly due to AODV which is basically created for MANET environment.

AntHocNet is a novel approach to the adaptive learning of routing tables in communications networks. AntHocNet is a distributed, mobile agents based on Ant Colony Optimization Algorithm which uses information available in vehicular networks such as the vehicles position and speed in order to design an ant based algorithm that performs well in the dynamics of such networks. This final project implements this method to showed that AntHocNet performs good in VANET.

The Simulation results show that AntHocNet is able to achieve better routing performance in packet delivery ratio, average end-to-end delay, and routing overhead with increasing vehicle density compared to AODV.

Kata kunci: AODV, AntHocNet, NS-2, VANETs

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

Segala puji bagi Allah SWT, yang telah melimpahkan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan pembuatan laporan tugas akhir dengan judul "Analisis Kinerja AntHocNet pada Lingkungan VANETs "yang merupakan salah satu syarat untuk mendapatkan gelar Sarjana Komputer Jurusan Informatika ITS.

Selesainya Tugas Akhir ini tidak terlepas dari bantuan dan dukungan beberapa pihak, sehingga pada kesempatan ini penulis mengucapkan terima kasih kepada :

1. Tuhan Yang Esa yang telah memberikan hidayah-Nya sehingga dalam pengerjaan Tugas Akhir ini dapat dikerjakan dengan baik.
2. Keluarga yang senantiasa memberikan doa dan dukungan hingga saat ini.
3. Bapak Dr.Eng. Radityo Anggoro, S.Kom, M.Sc. dan Prof. Ir. Supeno Djanali M.Sc., Ph.D. selaku dosen pembimbing yang memberikan nasehat, memberikan motivasi dan tentunya bimbingan itu sendiri dalam menyelesaikan Tugas Akhir ini.
4. Bapak Prof. Ir. Supeno Djanali M.Sc., Ph.D. selaku dosen wali yang telah memberikan arahan dan nasehat dalam dunia akademika di Teknik Informatika.
5. Seluruh dosen dan karyawan Teknik Informatika yang telah memberikan ilmu dan pengalaman kepada penulis selama menjadi civitas akademika Teknik Informatika.
6. Teman dan sahabat penulis yang tidak dapat disebutkan semuanya yang selalu membantu, dan bertukar ilmu dan berjuang bersama dengan penulis.

Penulis juga menyadari bahwa banyak kekurangan sehingga dengan kerendahan hati penulis mengharapkan kritik dan saran untuk perbaikan kedepan.

Surabaya, Januari 2018

[Halaman ini sengaja dikosongkan]

DAFTAR ISI

LEMBAR PENGESAHAN ..... **vii**

Abstrak ..... **ix**

Abstract ..... **x**

1. PENDAHULUAN **1**
1.1 Latar Belakang 1
1.2 Rumusan Masalah. 2
1.3 Batasan Masalah 2
1.4 Tujuan 3
1.5 Manfaat 3
1.6 Metodologi. 3
1.7 Sistematika Penulisan. 4

2. TINJAUAN PUSTAKA **7**
2.1 Ant Colony Optimization (ACO) 7
2.2 AntHocNet 10
2.3 AdHoc OnDemand Distace Vector(AODV) 12
2.4 Simulation of Urban Mobility (SUMO) 16
2.5 OpenStreetMap 17
2.6 JOSM 18
2.7 AWK 18
2.8 Network Simulator 2 (NS-2) 18
2.8.1 Installasi NS-2 19
2.8.2 Penggunaan Skrip OTcl 21
2.8.3 NS-2 TraceFile. 21
2.9 Patching AntHocNet 23

3. PERANCANGAN **27**
3.1 Deskripsi Umum 27
3.2 Perancangan Skenario Grid 28
3.3 Perancangan Skenario Riil 28
3.4 Perancangan Simulasi pada NS-2 30
3.5 Perancangan Metrik Analisis 30

3.5.1 Packet Delivery Ratio. 30
3.5.2 Average End-to-End Delay. 31
3.5.3 Routing Overhead 31

4. IMPLEMENTASI.....33
4.1 Implementasi Skenario Grid 33
4.2 Implementasi Skenario Riil 34
4.3 Implementasi Metrik Analisis 37
4.3.1 Implementasi Packet Delivery Ratio 38
4.3.2 Implementasi Average End-to-End Delay 38
4.3.3 Implementasi Routing Overhead.....39
4.4 Implementasi Simulasi pada AntHocNet.....39
4.4.1 Implementasi Struktur Paket Hello 41
4.4.2 Implementasi Perhitungan Pheromone 41
4.4.3 Implementasi Update Pheromone 42
4.4.4 Implementasi Pheromone Table 43
4.4.5 Implementasi Route Discovery pada AntHocnet 44
4.5 Implementasi Simulasi pada NS-2.....45

5. UJI COBA DAN EVALUASI 49
5.1 Lingkungan Uji Coba.....49
5.2 Hasil Uji Coba Skenario Grid.....50
5.3 Hasil Uji Coba Skenario Riil 53

6. PENUTUP.....59
6.1 Kesimpulan 59
6.2 Saran.....59

7. DAFTAR PUSTAKA.....62

8. LAMPIRAN.....62
8.1 Kode Implementasi Average End to end Delay 62
8.2 Kode Implementasi Routing OverHead 62
8.3 Kode Implementasi Packet Delivery Ratio.....63
8.4 Kode Implementasi TCL NS-2.....64
8.5 Kode Implementasi Request Table pada AntHocNet 67

8.6 Kode Implementasi Perhitungan Pheromone.....67
8.7 Kode Implementasi Update Pheromone 68

9. BIODATA PENULIS..... 71

DAFTAR GAMBAR

Gambar 2.1. Contoh Vanets dalam kehidupan 8
Gambar 2.2. Stigmery Semut 11
Gambar 2.3. Pengiriman Route Request (RREQ) 17
Gambar 2.4. Pengiriman Route Replay (RREP) 17
Gambar 2.5. Mekanisme Route Discovery.....18
Gambar 2.6. Hasil Pengubahan ls.h.....23
Gambar 2.7. Hasil Pengubahan Makefile.in.....23
Gambar 2.8. Contoh skrip lingkungan simulasi 24
Gambar 2.9. Link Patch Protokol AntHocnet 25
Gambar 3.1. Rancangan Simulasi 27
Gambar 3.2. Alur Pembuatan Peta Grid 29
Gambar 3.3. Alur Pembuatan Peta Riil 30
Gambar 4.1. Sintax netgenerate 33
Gambar 4.2. Hasil netgenerate 34
Gambar 4.3. Sintaks Pembuatan Titik Asal dan Tujuan 34
Gambar 4.4. Perintah Untuk Membuat Rute 34
Gambar 4.5 Isi file sumo.cfg 34
Gambar 4.6. Hasil Simulasi Lalu Lintas.....35
Gambar 4.7. Pemilihan Lokasi dari OpenStreetMap.....36
Gambar 4.8. Pengambilan Peta dari OpenStreetMap 36
Gambar 4.9. Penyuntingan Peta dengan JOSM 37
Gambar 4.10. Hasil gan Peta dengan JOSM 37
Gambar 4.11. Perintah untuk Konversi osm ke net.xml.....37
Gambar 4.12. Perintah untuk menjalankan delay.awk 39
Gambar 4.13. Penambahan Atribut pada ant_packet.h 40
Gambar 4.14. Penambahan Atribut pada ant_types.h 40
Gambar 4.15. Penambahan perhitungan formula pada ant_packet.cc 41
Gambar 4.16. Implementasi Phermone Update 42
Gambar 4.17. Implementasi Pheromone Table 43
Gambar 4.18. Potongan Skrip pengaturan Node 45
Gambar 5.1. Grafik PDR pada Skenario Grid 51
Gambar 5.2. Grafik RO pada Skenario Grid 52
Gambar 5.3. Grafik Delay pada Skenario Grid 53
Gambar 5.4. Grafik PDR pada Skenario Riil 54
Gambar 5.5. Grafik RO pada Skenario Riil 55

Gambar 5.6. Grafik Delay pada Skenario Riil 56

DAFTAR TABEL

Tabel 4.1. Parameter Pengaturan Node 49
Tabel 5.1. Spesifikasi Perangkat 49
Tabel 5.2. Parameter Lingkungan Simulasi 50
Tabel 5.3. Hasil Perhitungan Rata – Rata PDR pada Skenario Grid 52
Tabel 5.4. Hasil Perhitungan Rata – Rata RO pada Skenario Grid 53
Tabel 5.5. Hasil Perhitungan Rata – Rata Delay pada Skenario Grid 54
Tabel 5.6. Hasil Perhitungan Rata – Rata PDR pada Skenario Riil 55
Tabel 5.7. Hasil Perhitungan Rata –Rata RO pada Skenario Riil 56
Tabel 5.8. Hasil Perhitungan Rata - Rata Delay pada Skenrio Riil 57

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

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

1.1. Latar Belakang

Perkembangan teknologi dibidang komunikasi data sangat pesat sehingga memungkinkan adanya perangkat yang bersifat *ad-hoc*. Dengan adanya teknologi tersebut, komunikasi data antar perangkat bisa dilakukan meskipun dalam posisi diam.

Jaringan *Ad-Hoc* adalah jaringan yang menggunakan sedikit infrastrukur terutama untuk berbagai jenis komunikasi nirkabel. Jaringan ini dikategorikan dalam dua jenis yaitu *Mobile Ad-Hoc Network (MANET)* dan *Vehicular Ad-Hoc Network (VANET)* digunakan untuk kendaraan. Banyak protokol routing untuk MANET yang telah dirancang. Protokol utama yang terlibat adalah *routing protocol Ad-Hoc On-Demand Distance Vector (AODV), Destination Sequence Distance Vector (DSDV)* dan *Link State Geografis Routing Protocol (LGSR),* kinerja dari masing-masing protokol telah dievaluasi dalam hal *Throughtput, Average End-to-End Delay, Packet Delivery Ratio, Routing Overhead* dll.

Penilaian kinerja protokol LSGR, AODV dan DSDV menggunakan Network Simulator (NS-2). Dengan mempertimbangkan kinerja masing-masing protokol yang telah diamati, LSGR adalah protocol yang tampil lebih baik dari pada DSDV dan AODV pada jaringan *Ad-Hoc.*

Protokol routing pada jaringan *Ad-Hoc* menjadi suatu permasalahan yang menantang untuk diteliti semenjak sebuah node bisa bergerak secara bebas. Pada jaringan *Ad-Hoc* terdapat dua tipe protokol routing, yaitu :

1. Proaktif : *Destination Sequenced Distance Vector* (DSDV), *Cluster Switch Gateway Routing* (CSGR), *Wireless Routing Protocol* (WRP), *Optimized Linkstate Routing*(OLSR).

2. Reaktif : *Dynamic Source Routing* (DSR), *Ad hoc On-Demand Distance Vector* (AODV), *Temporally Ordered Routing Protocol Algorithm* (TORA), *Associativy Based Routing* (ABR), *Signal Stability Routing* (SSR).

Pada jaringan multi jalur adalah membuat algoritma *routing* yang dapat mencari jalur terpendek dan efisien, terutama saat menghadapi perubahan topologi atau perubahan nilai (*cost*) pada jalur. Dalam hal ini perubahan topologi jaringan bisa disebabkan antara lain karena adanya kenaikan atau penurunan kapasitas *bandwidth*, perubahan kondisi beban data, dan kondisi saat terjadi kegagalan pada jalur utama. Untuk itu diperlukan kinerja protokol *routing* yang efisien dalam memberikan solusi jalur alternatif terbaik.[1]

Beberapa penelitian sebelumnya terkait dengan optimasi *routing* tersebut dengan menggunakan metode *Ant Colony* telah dilakukan. Masalah – masalah pemilihan jalur terpendek dan terbaik seperti pada *Travelling Saleman Problem* dan *Data Network Routing* telah dicoba diselesaikan dengan metode *Ant Colony*. Dengan menggunakan metode *Ant Colony*, mencoba membandingkan metode *routing metaheuristic* dengan metode *routing* konvensional (RIP dan OSPF). Hasil perbandingan menunjukkan bahwa metode *Ant Colony* lebih baik dalam kecepatan penyampaian data walaupun dalam hal *delay* masih kurang bagus dibandingkan dengan yang lain. [1]

1.2. Rumusan Masalah

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

1. Bagaimana cara mengimplementasikan protokol AntHocNet pada NS-2?
2. Bagaimana perbedaan kinerja anatara AODV dengan AntHocNet pada skenario grid dan skenario riil?

1.3. Batasan Masalah

Permasalahan yang dibahas pada Tugas Akhir ini memiliki beberapa batasan, yaitu :

1. Software Simulator yang digunakan adalah NS-2 versi 2.35
2. Area simulasi dibuat dengan SUMO, berukuran 2600 m x 2600 m untuk skenario grid dan 2600 m x 2600 m untuk skenario riil.
3. Skenario dibedakan berdasarkan jumlah node, 100,150, 200, 250, 300, dan 350

1.4. Tujuan

Tujuan dari pembuatan Tugas Akhir ini adalah untuk melakukan analisis perbandingan performa antara routing protokol AODV dengan AntHocNet.

1.5. Manfaat

Dengan dibuatnya Tugas Akhir ini maka akan dapat memberikan informasi mengenai perbandingan performa antara routing protokol AntHocNet berbasis *Ant Colony Optimization (ACO)* dengan *Ad-Hoc On-Demand Distance Vector (AODV).* Dan tentunya menjadi acuan untuk penelitian yang lebih mendalam mengenai performa gabungan AODV dengan metode yang berbeda untuk menghasilkan routing protokol dengan performa yang baik.

1.6. Metodologi

1. Penyusunan Proposal Tugas Akhir

   Tahap awal untuk memulai pengerjaan Tugas Akhir adalah penyusunan proposal Tugas Akhir. Pada proposal tersebut dijelaskan secara garis besar tentang alur pembuatan sistem.

2. Studi Literatur

   Tahap ini dilakukan studi literature mengenai alat dan metode yang digunakan. Literatur yang dipelajari dan digunakan meliputi buku referensi, artikel, jurnal dan dokumentasi dari internet yang terpercaya.

3. Implementasi Protokol

   Tahap ini meliputi perancangan sistem berdasarkan dari perangkat lunak yang ada. Tahap ini merupakan tahap yang paling penting dimana bentuk awal aplikasi yang akan diimplementasikan didefinisikan. Pada tahapan ini dibuat *prototype* sistem, yang merupakan rancangan dasar dari sistem yang akan dibuat. Serta dilakukan desain suatu sistem dan desain proses-proses yang ada.

4. Uji Coba dan Evaluasi

   Tahap ini dilakukan proses uji coba terhadap aplikasi yang telah dibuat. Pengujian dan evaluasi akan dilakukan dengan melihat kesesuaian dengan perencanaan. Tahap ini dimaksudkan juga untuk mengevaluasi jalannya sistem, mencari masalah yang mungkin timbul dan mengadakan perbaikan jika terdapat kesalahan.

5. Penyusunan Buku Tugas Akhir

   Tahap ini disusun buku sebagai dokumentasi dari pelaksanaan Tugas Akhir.

1.7. Sistematika Penulisan

Buku Tugas Akhir ini disusun dengan sistematika penulisan sebagai berikut :

Bab I PENDAHULUAN

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

Bab II TINJAUAN PUSTAKA

Bab ini berisi penjelasan detail mengenai dasar-dasar penunjang untuk mendukung pembuatan Tugas Akhir ini.

Bab III PERANCANGAN

Bab ini berisi perancangan metode yang nantinya akan diimplementasikan dan dilakukan uji coba.

Bab IV IMPLEMENTASI

Bab ini membahas implementasi dari desain yang telah dibuat pada bab sebelumnya. Penjelasan berupa imlpementasi mobilitas vehicular, konfigurasi sistem dan skrip analisis yang digunakan untuk menguji performa routing protokol.

Bab V UJI COBA DAN EVALUASI

Bab ini menjelaskan tahap pengujian sistem dan pengujian performa dalam skenario mobilitas vehicular yang dibuat.

Bab VI PENUTUP

Bab ini menyampaikan kesimpulan dari hasil uji coba yang dilakukan terhadap rumusan masalah yang ada dan saran untuk pengembangan yang lebih lanjut.

[Halaman ini sengaja dikosongkan]

BAB II
TINJAUAN PUSTAKA

Bab ini berisi penjelasan teori-teori yang berkaitan dengan pengimplementasian perangkat lunak. Penjelasan ini bertujuan untuk memberikan gambaran secara umum terhadap routing protokol, alat, serta definisi yang digunakan dalam pembuatan Tugas Akhir.

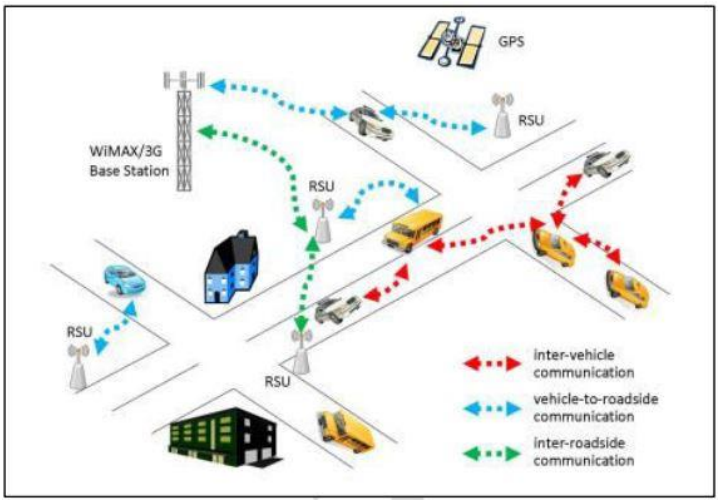
2.1. Vehicular Adhoc Network

Vehicular Adhoc Network merupakan sekumpulan *node* yang bergerak secara dinamis sebagai pengembangan dari *Mobile Adhoc Network* yang mempertimbangkan semua kendaraan dalam jaringan sebagai *node* tersebut untuk berkomunikasi dengan kendaraan lainnya pada radius tertentu. Pada VANETs, *node* yang bergerak tergantung pada *adhoc routing protocol* untuk menentukan bagaimana proses pengiriman data dari *node* asal ke *node* tujuan[1]

Adhoc routing protocol diklasifikasikan menjadi dua kategori, yaitu *Topology-Based routing, Geographic Posistion-Based Routing* dan *Hybrid Routing Protocol*. Meskipun menggunakan *routing protocol* yang sama, VANETs memiliki karakter yang berbeda dengan MANET karena memilki batasan pergerakan dan kecepatan yang tinggi dengan menciptakan keunikan karakteristik dari VANETs[2]

Ada dua jenis *node* yang terhubung pada VANETs, yaitu RSU (*Road-Side Unit*) dan OBU (*On-Board Unit*). RSU merupakan node yang terpasang pada beberapa bagian jalan yang terhubung dengan jaringan *backbone* untuk memberikan informasi-informasi penting kepada OBU. Kedua jenis *node* ini mengakibatkan ada dua tipe komunikasi yang memungkinkan terjadi, yaitu antara kendaraan dengan RSU.

Contoh informasi yang dapat diberikan oleh RSU adalah batas kecepatan, status lampu lalu lintas, keberadaan infrastruktur penting dan lain-lain. Selain itu, RSU dapat memberikan jaringan internet kepada OBU yang terhubung.



Gambar 2.1 Contoh Vanets dalam kehidupan.

Vanet memiliki banyak manfaat, diantaranya sebagai berikut :

1. Untuk menghindar tabrakan antara dua kendaraan atau lebih dengan mempertimbangkan jarak dan kecepatan dengan menggunakan sistem pengereman mendadak.
2. Untuk mendeteksi keadaan yang berbahaya, misalnya kondisi jalan yang rusak, salju tebel, banjir, jalan yang diperbaiki, diblokir maupun jalan yang licin.
3. Digunakan sebagai sinyal pengiriman untuk meminta bantuan secara otomatis pada saat terjadi kecelakaan.
4. Mendeteksi pengemudi yang tidak menaati peraturan lalu lintas, seperti menelpon dijalan, tidak menggunakan sabuk keselamatan.

2.2. Ant Colony Optimization (ACO)

Ant Colony Optimization (ACO) adalah suatu metodologi yang dikemukakan pada tahun 1991 oleh Macro Dorigo. *Ant System* telah diterapkan dibanyak permasalahan, seperti optimasi kombinatorial. Salah satu contohnya adalah *Travelling Salesman*

Problem, Quadratic Assigment Problem, Jobschedulling Vehicle Routing, Graph Coloring, dan *Network Routing* [3]

Algoritma *Ant Colony Optimization* merupakan bagian dari *Swan Intelligence*. *Ant colony optimization* telah berhasil di aplikasikan pada jumlah angka besar yang berbeda, permasalahan yang sering diselesaikan yaitu *traveling salesman*, *quadratic assignment*, penjadwalan, routing kendaraan, routing pada jaringan telekomunikasi. Koloni serangga memiliki kapabilitas untuk menyelesaikan permasalahan optimisasi tersebut.

Pada algoritma *Ant Colony*, konsep dasarnya adalah terinspirasi dari kebiasaan semut yang alami. Ketika semut mencari makan, mereka memulai dari sarang mereka dan bergerak secara acak menuju ke daerah makanan. *Ants* menggunakan senyawa kimia dengan konsentrasi yang banyak yang disebut dengan *pheromone* untuk mengenali sistem sinyal. Ketika berjalan, ants mengeluarkan kuantitas *pheromone*, menandai rute yang dipilih agar mereka mengikuti substansi percobaan. Ketika *ants* mengikuti *intersection*, hal ini akan menentukan lintasan mana yang akan diikuti. Konsentrasi *pheromone* ditentukan pada lintasan merupakan indikasi kegunaan dari lintasan tersebut. *Ant* akan memilih lintasan dengan probabilitas yang tinggi untuk diikuti dan disitulah kualitas *pheromone* menjadi kuat[4]

Konsentrasi *pheromone* akan berkurang dikarenakan faktor difusi. Proses ini merupakan proses yang memiliki karakter yang positif karena *feedback loop*, dimana probability yang ant pilih memberikan lintasan yang kuat seiring bertambahanya *ant* yang memilih lintasan tersebut. *Ant* yang mengambil jalur terpendek akan menemukan tujuan dengan cepat. Ketka mereka kembali ke sarang, *ant* kembali memilih lintasan yang harus dipilih. Setelah waktu yang cukup lama, konsentrasi *pheromone* akan menjadi sangat tinggi dan akan berulang pada *ant* yang lain. Dan akhirnya *ant* tersebut menemukan lintasan yang paling pendek. Semua proses tersebut dapat dijadikan metode untuk mencari jaringan dengan lintasan terpendek. Dan juga, ant memiliki kapabilitas untuk beradaptasi dengan lingkungan dan dapat menemukan lintasan terpendek baru dengan cepat. Proses ini sangat disarankan untuk *vehicular adhoc network* dengan jaringan yang dinamis.

Dalam informatika biologi termasuk aplikasi dari *Ant colony optimization* untuk melipatkan protein. Yang terdiri dari penemuan bentuk fungsi dan penyesuaian dari protein ke dalam dua atau tiga bentuk dimensi. Rangkaian blok perkalian menjadi perhatian dari beberapa rangkaian protein atau rangkaian DNA agar dapat menemukan kesamaan diantara mereka.setelah selesai, agar dapat ditentukan perbedaan protein yang sama datang dari perbedaaan spesies. Informasi ini mendukung kesimpulan dari pohon phylogenetic[5]

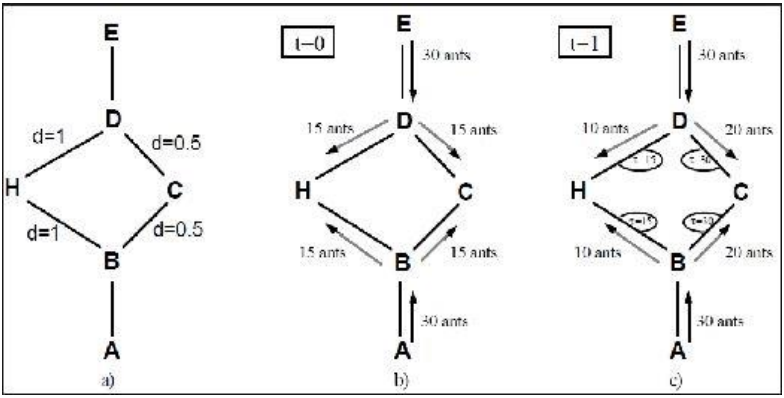
Algoritma *Ant Colony Optimization* sekarang ini berada di metode penyelesaian *state-of-the-art*, rangkaian permasalahan pengiriman, masalah penggajian yang terdapat penjadwalan proyek, masalah jadwal buka toko, dan permasalahan 2D and 3D *hydrophobic polar protein*.

Secara alamiah koloni semut mampu menemukan rute terpendek dalam perjalanan dari sarang ke tempat-tempat sumber makanan. Semut dapat bekerja sama dengan koloninya dan bertukar informasi secara tidak langsung yang disebut dengan *Stigmery*. Pada saat melakukan suatu rute perjalanan, semut meletakkan sejumlah informasi pada daerah yang dilaluinya yaitu *Pheromon*e.

Pheromone adalah zat yang dikeluarkan oleh semut. Semut berikutnya yang melalui jalur tersebut akan mengidentifikasikan *pheromone* yang diletakkan oleh semut sebelumnya dan memutuskan dengan probabilitas yang tinggi untuk mengikutinya dan menguatkan jalur yang dipilihnya dengan *Pheromone* miliknya[5]

Karakteristik ACO adalah sebagai berikut :

1. Menggunakan interaksi agen *(ant)* dimana masing-masing ant hanya mampu melakukan tugas sederhana untuk menghasilkan solusi.
2. Menggunakan informasi yang diperoleh dari iterasi sebelumnya berupa *pheromone* untuk menentukan hasil pada iterasi selanjutnya.
3. Terdapat mekanisme *positive feedback* untuk menambahkan *pheromone* pada suatu *node*.
4. *Evaporasi pheromone* untuk mengurangi *pheromone*.



Gambar 2.2. Stigmery semut

Keterangan :

1. Graf dengan jarak D – H = H – B = 1, dan D – C = C – B = 0.5. E adalah sarang dan A adalah sumber makanan.
2. Pada saat t = 0 belum terdapat jejak, semut memilih untuk bergerak kearah kanan atau ke kiri.
3. Pada saat t = 1, jejak *Pheromone* lebih kuat pada jarak yang lebih pendek, sehingga lebih banyak semut yang memilih jalur tersebut untuk dilewati.

Persamaan yang digunakan untuk menghitung setiap rute :

$$P_{Update} = \frac{(P_{old} + \Delta P)}{1 + \Delta P} \qquad (1)$$

Keterangan :

1. P adalah *pheromone* yang terdapat pada jalur.
2. ΔP adalah pertambahan *pheromone.*

Persamaan yang digunakan untuk menghitung setiap rute :

$$P_{new} = \frac{Nn}{Nh} + \frac{Lc}{100} + \frac{1}{T} \qquad (2)$$

Keterangan :

1. Nn adalah *No Node*
2. Nh adalah *No Hop*
3. Lc adalah *Link Capacity*
4. T adalah *Time*

2.3. AntHocNet

Adhoc wireless multi-hop network dibangun dengan *node wireless* yang dirancang dengan tipe *adhoc*. Setiap *node* dapat meneruskan paket dan juga bertindak sebagai *node* sumber. *Adhoc wireless multi-hop network* (AHWMNs) tidak memiliki infrastruktur yang tetap. *Node* dapat bertindak sebagai *node* sumber atau *node* tujuan maupun penerus (*router*). Setiap *node* merupakan *node independent* dan tidak memiliki koordinator terpusat. [6]

Sebuah *node* dapat meninggalkan jaringan setiap waktu. Pada protokol jaringan *adhoc wireless multi-hop network* terbagi menjadi *reactive, proactive* dan *hybrid routing protokol*. *Reactive routing protocol* dapat menemukan rute ketika *node* sumber membutuhkan untuk mengirimkan data ke *node* tujuan. Rute akan diselesai dibuat ketika paket tersebar dijaringan. *Proactive routing protocol* memelihara informasi routing terakhir pada setiap *node*. AntHocNet adalah *bio-inspire routing protocol* yang merupakan *routing protocol hybrid. Routing protokol hybrid* merupakan routing kombinasi antara *route reactive setup* dan *proactive path maintenance.*

Routing protokol yang terinspirasi dari alam dapat menyelesaikan permasalahan routing pada *adhoc wireless multi-hop network* maupun vanets. Dengan berbasiskan pada algoritma *Ant Colony*. Pada waktu *ants* mencari makan ketika *ants* menemukan sumber makanan, mereka kembali ke tempat dimana mereka memulai. *Ants* melepaskan senyawa kimia yang disebut dengan *pheromone* pada jalur.

Dengan menggunakan rute yang berbeda, beberapa *ants* akan menjelajahi ke tempat makanan yang sama. *ants* akan menjelajahi lintasan yang akan meninggalkan *pheromone* lebih pada

setiap lintasan untuk membantu *ants* yang lain berpindah pada lintasan. Semakin banyak *ants* yang terpengaruh oleh *pheromone* maka semakin kuat pula jaringan lintasan tersebut untuk sering dilalui oleh *ants*.

Ants melakukan komunikasi secara tidak langsung menggunakan fenomena yang disebut *stigmery*. *Stigmery* merupakan agen yang akan meninggalkan sinyal kepada yang lain dan agen yang lain akan merasakannya. Tipe komunikasi ini merupakan komunikasi lokal yang merupakan agen sederhana untuk melakukan interaksi tanpa informasi menyeluruh.

AntHocNet merupakan algortima *hybrid* yang terdiri dari dua elemen yaitu *reactive dan proactive*. *AntHocNet* menyimpan informasi routing hanya ketika *node* sumber dan *node* tujuan masuk sesi komunikasi dan proaktif disini maksudnya adalah melakukan perawatan dan memperbaiki informasi mengenai lintasan yang sedang ada ketika sedang berjalan. Informasi routing tersimpan pada tabel *pheromone*. Paket control dan paket data dikirimkan ke *stochastic way* mengunakan tabel tersebut [6]

Pada proses *reactive rute setup* ketika *node* sumber akan melakukan koneksi dengan *node* tujuan, akan mengecek table routing pada alamat *node* tujuan. Jika tidak bisa maka dilakukan penyebaran pesan yang membanjiri keseluruhan *node* yang ada dijaringan. Hal tersebut disebut dengan *reactive forward ants*. *Node* lanjutan yang menerima pesan hasil *broadcast* akan disebarkan jika bukan termasuk *node* tujuan begitu pula sebaliknya akan *unicast* pesan tersebut.

Reactive forward ants akan menjaga daftar *node* yang telah selesai digunakan. Ketika *node* tujuan sudah menerima pesan maka dilanjutkan dengan mengonversi ke *reactive backward ant*. Diikuti oleh lintasan yang sama dimana *forward ant* telah sedang digunakan dan juga mempunyai kualitas informasi mengenai setiap lintasan *link* dan perbaruan pada setiap *node* pertengahan dan *node* sumber.

Pada *proactive route maintenance* terdapat dua fase yaitu *pheromone diffusion* dan *proactive ant sampling*. Penyebaran *pheromone* pada *node* secara periodik membroadcast informasi dengan kualitas yang baik. Dengan menggunakan semua informasi *node* akan menghitung tabel *pheromone* yang baru dan dilanjutkan

dengan membroadcastkannya. *Pheromone* ini disebut dengan *virtual pheromone*.

Pada sampling *proactive ant*, semua *node* sumber dari sesi komunikasi akan secara periodik menyebarkan *proactive forward ant*. *Ant* tersebut akan memilih *node* selanjutnya secara acak dan membangun jalur baru. Untuk pemilihan *node* selanjutnya, *ant* mempertimbangkan antara *regular* dan *virtual pheromone* dan *ant* meninggalkan lintasan sebelumnya dan mengikuti lintasan yang ada pada fase *pheromone diffusion.* Setelah mencapai tujuan, *proactive forward ant* berubah menjadi *proactive backward ant* dan kembali ke sumber.

Link failure diatur oleh penggunaan mekanisme *reactive,* seperti *local route repair* dan *warning messages*. Pada mekanisme *local route repair, forward ant* dikirim dari *node* yang gagal ke *node* sumber dari sesi komunikasi tersebut. *Local route repair* akan membatasi ketelitian jumlah *hops*. Ketika batas jumlah *hops* telah tercapai, maka *ant* tersebut akan dihapus. Pesan peringatan *unicast* akan dikirim dari *node* yang gagal ke *node* sumber. Setelah penyebaran *ant* yang dihapus juga jika paket yang diterima pada lintasan yang rusak maka pesan akan digunakan untuk menyediakan pemberitahuan lintasan gagal [5]

2.4. Ad-hoc On-Demand Distance Vector (AODV)

AODV menawarkan adaptasi yang cepat terhadap kondisi link yang dinamis, proses yang rendah, rendahnya rendahnya penggunaan jaringan dan menentukan pilihan rute ke tujuan dalam jaringan *adhoc*. AODV memungkinkan *node mobile* untuk mendapatkan rute ke tujuan baru dengan cepat. [7]

Ad hoc On-Demand Distance Vector (AODV) adalah sebuah protocol *routing* yang paling banyak diteliti pada lingkungan *adhoc. .Ad hoc On-Demand Distance Vector* (AODV) adalah *distance vector routing protocol* yang termasuk dalam klasifikasi reaktif *routing protocol*, yang hanya me-*request* sebuah rute saat dibutuhkan [2]. Ciri Utama dari AODV adalah menjaga *timer-based state* pada

setiap *node* sesuai dengan penggunaan tabel *routing*. Tabel routing akan kadaluarsa jika jarang digunakan.

AODV memerlukan setiap *node* untuk menjaga table *routing* yang berisi *field* :

1. *Desitination IP Address* : berisi alamat IP dari *node* tujuan yang digunakan untuk menetukan rute.
2. *Destination Sequence Number* : *Destination Sequence Number* bekerjasama untuk menentukan rute.
3. *Next Hop* : 'Loncatan' (*hop*) berikutnya, bisa berupa tujuan atau *node*.
4. *Hop Count* : Jumlah *hop* dari alamat IP sumber sampai ke alamat IP tujuan.
5. *Lifetime* : Waktu dalam milidetik yang digunakan untuk *node* menerima RREP.
6. *Routing Flags* : Status sebuah rute: *up* (valid), *down* (tidak valid) atau sedang dipakai.

AODV memiliki *Route Discovery* dan *Route Maintenance*. *Route Discovery* berupa *Route Request* (RREQ) dan *Route Reply* (RREP). Sedangkan *Route Maintenance* berupa *Data*, *Route Update* dan *Route Error* (RRER).

Route Discovery adalah suatu mekanisme pada protocol yang berfungsi untuk melakukan pencarian *Path* (jalur) secara dinamis dalam jaringan *ad hoc*, baik secara langsung didalam range transmisi ataupun dengan melewati beberapa *Node Intermediate* [5].

Route Request (RREQ) adalah paket yang dikirmkan ke *node* sumber (*Source*) ke jaringan untuk menemukan rute ke *node* tujuan. RREQ berisi *Source Destination* (SrcID), *Destination Identifier* (DestID), S*ource Sequence Number* (SrcSeqNum), *Destination Sequence Number* (DestSeqNum), *Broadcast Identifier* (BcastID), dan *Time to Live* (TTL).

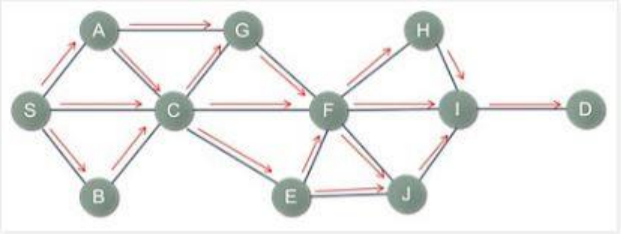
Route Reply (RREP) adalah balasan dari hasil respon karena telah menerima RREQ. RREP akan dikirm kembali ke *node* asalnya jika *Destination Sequence Number* sama atau lebih besar dari yang dispesifikasikan di RREQ.

Ketika sebuah *node* sumber (*Source*) ingin mengirim beberapa data ke *node* tujuan (*Destination*) dimulai dari pencarian rute. Penemuan jalur (*Path Discovery*) atau *Route Discovery* melakukan *Broadcast*. Paket yang di *Broadcast* yaitu paket RREQ ke jaringan. *Node Intermediate* ketika menerima Paket RREQ melihat ke dalam *cache*-nya untuk mengetahui apakah telah ada beberapa rute tujuan, jika ada maka akan membalas ke pengirim dengan paket RREP (balasan rute), dimana mengandung rute.

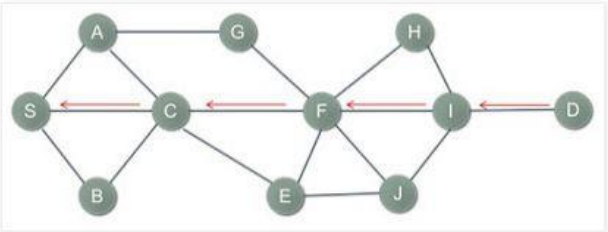
Jika *Node Intermediate* tidak memiliki rute, maka melakukan *broadcast* permintaaan ulang setelah menambahkan alamat ke dalam rute sumber. Ketika *query* mencapai *node* tujuan, yang menemukan *node* yang dipesan pada urutan *hop* dalam paket RREQ dan menggunakan itu untuk memberikan paket RREP ke pengirim.

Sebuah *link* ke *hop* berikutnya tidak dapat ditemukan atau dideteksi dengan metode penemuan rute, maka link tersebut akan disimpulkan telah putus dan *Route Error* (RRER) akan disebarkan ke *node* terdekat. Dengan demikian sebubah *node* bisa menghentikan pengiriman data melalui rute ini atau meminta rute baru dengan menyebarkan RREQ kembali, seperti pada pada Gambar 2.2.

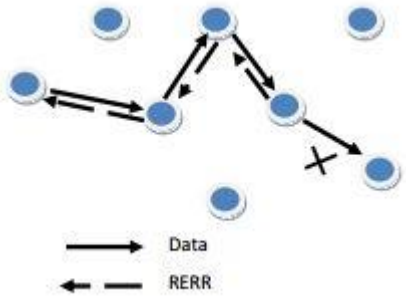
Kelemahan dari AODV adalah lintasan bisa menjadi tidak valid setiap waktu karena node dijaringan ini adalah mobile dan tidak pasti ketika lintasan akan menjadi tidak valid. Ketika jumlah node meningkat pada jaringan ini makan AODV menignkatkan banyak bandwidth dan juga membuat delay tambahan. AODV mungkin akan mengalami delay ketika sedang membangun lintasan, dan link failure memungkinkan mengenalkanroute discorey yang lainnya, dengan mengenerat delay yang berlebih dan mengonsumsi bandwidth sesuai dengan jaringan yang meningkat. Seperti yang kita ketahui bahwa Vanet memilki jaringan yang besar dan delay pada jaringan ini dapat menyebabkan masalah serius dan juga membutuhkan banyak bandwidth.



Gambar 2.3. Pengiriman Route Request (RREQ)



Gambar 2.4. Pengiriman Route Reply (RREP)



Gambar 2.5. Mekanisme Route Discovery

2.5. Simulation of Urban Mobility (SUMO)

Simulation of Urban Mobility atau SUMO merupakan sebuah program simulasi lalu lintas. SUMO dikembangkan sejak tahun 2000 dengan tujuan untuk mengakomodasi penelitian-penelitian yang melibatkan pergerakan kendaraan dijalan raya, terutama dalam daerah dengan penduduk yang padat (*urban*). Publikasi referensi terbaru tentang SUMO ditulis oleh Krajzewice et al. pada tahun 2012.

SUMO terdiri dari beberapa bagian yang dapat membantu pemuatan simulasi lalu lintas pada tahap-tahap yang berbeda. Berikut penjelasan fungsi tools yang digunakan untuk pembuatan Tugas Akhir ini :

- netgenerate
  netgenerate merupakan tool yang berfungsi untuk membuat peta berbentuk seperti *grid, spider,* dan bahkan *random network.* Sebelum proses netgenerate, pengguna dapat menentukan kecepatan maksimum jalan dan membuat *traffic light* pada peta. Hasil dari netgenerate ini berupa file dengan extensi .net.xml. Pada Tugas Akhir ini netgenerate digunakan untuk membuat peta skenario *grid*. [8]
- netconvert
  netconvert merupakan program CLI yang berfungsi untuk melakukan konversi dari peta seperti OpenStreetMap menjadi format *native* SUMO. Pada Tugas Akhir ini penulis menggunakan *netconvert* untuk mengkonversi peta dari OpenStreetMap. [9]
- randomTrips.py
  tool dalam SUMO yang digunakan untuk membuat rute acak yang akan dilalui oleh kendaraan dalam simulasi.
- route2trips.py
  tool yang digunakan untuk membuat detail perjalanan setiap kendaraan berdasarkan output dari randomTrips.py
- duarouter

tool yang digunakan untuk melakukan perhitungan rute berdasarkan definisi yang diberikan dan memperbaiki kerusakan rute.

- sumo
  program yang melakukan simulasi lalu lintas berdasarkan data-data yang didapatkan dari netgenerate (scenario *grid*) atau netconvert dan randmTrips.py. hasil simulasi dapat di-export ke sebuah file untuk di konversikan menjadi format lain.
- sumo-gui
  GUI yang digunakan untuk melihat simulasi yang dilakukan oleh SUMO secara grafis.
- traceExporter.py
  tool yang bertujuan untuk mengkonversi output dari sumo menjadi format yang dapat digunakan pada simulator lain. Pada Tugas Akhir ini penulis menggunakan traceExporter.py untuk mengkonversi data menjadi format.tcl yang dapat digunakan pada NS-2.

2.6. OpenStreetMap

Pengembangan OSM diinspirasi oleh kesuksesan Wikipedia pengaruh dari peta *proprietary* di UK dan tempat lain. Sejak diluncurkan hingga sekarang OSM telah berkembang hingga memiliki lebih dari dua juta pengguna yang terregistrasi yang dapat mengambil data menggunakan survey manual, poranti GPS, *aerial photography* dan sumber bebas lainnya. [10]

Data yang terdapat pada OSM berada dalam lisensi *Open Database License* sehingga data dari OSM dapat dengan bebas digunakan oleh semua orang. Pada Tugas Akhir ini penulis menggunakan data yang tersedia pada OpenStreetMap untuk membuat skenario lalu lintas dengan peta Surabaya.

2.7. JOSM

JOSM (*Java OpenStreetMap Editor*) adalah sebuah alat yang digunakan untuk menyunting data yang didapatkan dari

OpenStreetMap. Aplikasi JOSM dapat diunduh pada alamat https:josm.openstreetmap.de/. Penulis menggunakan aplikasi ini untuk menyunting dan merapikan potongan peta yang diunduh dari OpenStreetMap.

2.8. AWK

AWK merupakan sebuah *Domain Spesific Language* (DSL) yang didesain untuk *text processing* dan biasanya digunakan sebagai alat ekstraksi data dan *reporting.* AWK bersifat data-driven yang berisikan kumpulan perintah yang akan dijalankan pada data tekstual baik seara langsung apda file atau digunakan sebagai bagian dari pipeline. Pada Tugas Akhir ini penulis menggunakan AWK untuk memproses data yang dihasilkan dari simulasi pada NS-2 dan mendapatkan analisis mengenai *packet delivery ratio, end-to-end delay, routing overhead* dan lain-lain.

2.9. Network Simulator 2 (NS-2)

NS-2 merupakan sebuah *discrete event simulator* yang didesain untuk membantu penelitian pada bidang jaringan komputer. Versi terbaru dari NS-2 adalah ns-2.35 yang dirilis pada tanggal 4 November 2011. Dalam membuat sebuah simulasi, NS-2 menggunakan Bahasa C++ dan OTcl.

Bahasa C++ digunakan untuk mengimplementasikan bagian-bagian jaringan yang akan disimulasikan. Sedangkan OTcl digunakan untuk menulis skenario simulasi jaringan. NS-2 mendukung sistem operasi GNU/Linux, FreeBSD, OSX dan Solaris. NS-2 juga dapat dijalankan pada sistem operasi Windows dengan menggunakan Cygwin.

2.9.1. Installasi NS-2

Sebelum melakukan instalasi NS-2 hal pertama yang harus dilakukan adalah melakukan installasi dependensi yang dibutuhkan. Salah satu dari dependensi dari NS-2 adalah GCC versi 4.4. Gambar

2.5. menunjukkan dependensi NS-2 beserta GCC 4.4 dan cara meng-install-nya pada distribusi Debian dan turunannya.

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

Gambar 2.6. Dependensi dari NS-2

Setelah semua dependensi ter-install, selanjutnya unduh packages NS-2. Jika proses pengunduhan sudah selesai, lakukan ekstraksi file tersebut dengan command seperti pada script dibawah ini..

```
tar –xvsf ns-allinone-2.35.tar.gz
```

Lakukan navigasi menuju folder “linkstate” yang terdapat pada direktori ns-allinone-2.35/ns-2.35/linkstate. Kemudian buka file yang bernama “ls.h” dan cari baris 137 pada kode tersebut. Setelah tambahkan ”this->” sebelum erase. Screenshot dari perubahan yang dilakukan pada file tersebut dapat dilihat pada script dibawah ini..

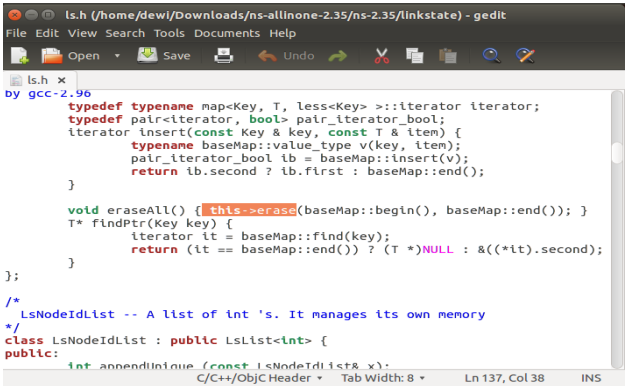
```
cd ~/ns-allinone-2.35/ns-2.35/linkstate
```

Setelah itu lakukan navigasi selanjutnya pada folder Otcl-1.14 yang terdapat pada direktori ns-allinone-2.35/ns-2.35/otcl-1.14/. Kemudian lakukan pengubahan pada file bernama Makefile.in. Buka file Makefile.in, ubah cc = @cc@ dengan cc= gcc-4.4. Screenshot dari perubahan yang dilakukan pada file tersebut dapat dilihat pada Gambar 2.7.
Berikut adalah perintah untuk membuka file Makefile.in.

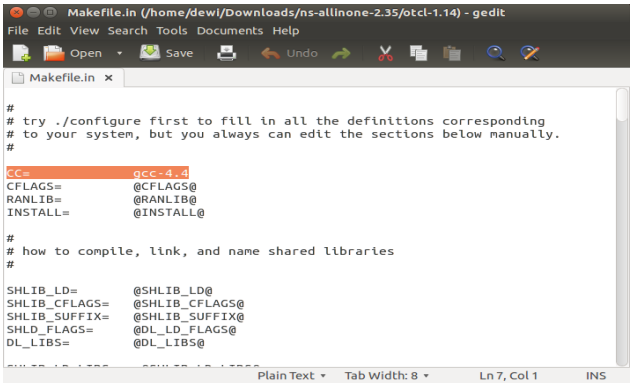
```
cd ~/ns-allinone-2.35/otcl-1.14/Makefile.in
```

Ketikkan perintah sudo make ./install seperti pada script dibawah ini untuk melakukan installasi ns2.

```
sudo make ./install
```



Gambar 2.7. Hasil Pengubahan ls.h



Gambar 2.8. Hasil Pengubahan Makefile.in

2.9.2. Penggunaan skrip OTcl

OTcl merupakan ekstensi *object oriented* dari Bahasa pemrograman Tcl. Bahasa OTcl digunakan sebagai Bahasa scripting pada NS2 untuk mengatur lingkungan dan skenario simulasi. Setiap class yang terdapat pada OTcl memiliki *binding* pada kode C++. Hal ini memungkinkan pembuatan skenario simulasi tanpa perlu menggunakan Bahasa C++ secara langsung. Gambar 2.10

menunjukkan potongan kode OTcl pada NS-2 untuk melakukan pengaturan lingkungan simulasi.

Baris satu hingga baris sembilan digunakan untuk mengatur lingkungan simulasi. Kemudian baris sebelas merupakan objek simulator yang akan menjalankan simulasi VANET diinstansiasi. Baris duabelas menginstansiasi koneksi wireles yang akan digunakan oleh setiap *node*. Pada baris tiga belas, semua variable pengaturan pada *node* dimasukkan. Ketika simulasi berjalan, seluruh *node* akan menggunakan pengaturan yang sama.

2.9.3. NS-2 Trace File

Trace file adalah file hasil simulasi dari sebuah skenario pada NS-2. Isi dari sebuah *trace file* adalah catatan dari setiap paket yang dikirim dan diterima oleh setiap *node* dalam simulasi. Setiap jenis paket pada jaringan memiliki pola penulisan tersendiri sehingga dapat dibedakan satu sama lain dan membantu memudahkan analisis terhadap hasil simulasi.

Pada Tugas Akhir ini penulis menggunakan routing protokol AODV dan AntHocNet. Pada AODV paket yang digunakan adalah HELLO, RREP, RRER, RREQ dan paket data. Paket routing protokol ditandai dengan tulisan “RTR” pada kolom keempat. Kolom ketujuh menunjukkan informasi paket routing protokol. Kolom kedelapan menunjukkan ukuran dari paket routing control.

Pola penting lainnya adalah paket yang dikirm selalu bertuliskan “s” dan paket yang diterima selalu bertuliskan “r” pada kolom pertama. Kolom kedua adalah waktu (dalam detik) ketika *event* tersebut terjadi.

Dengan mengetahui pola yang terdapat pada *trace file*, analisis hasil simulasi dapat dilakukan. Berikut adalah potongan kode pengaturan lingkungan simulasi VANET.

2.10. Patching AntHocNet

Protokol AntHocNet pada dasarnya belum terdapat dalam *source code* NS-2 yang telah diunduh, oleh karena itu perlu

dilakukan *patching* sebelum installasi NS-2. Pada Tugas Akhir ini, penulis menggunakan *patch* AntHocNet. *Patch* diunduh pada tautan :

```
https://drive.google.com/file/d/0B7S255p3kFXNv2ctNFctd3JsZGs/view
```

Gambar.2.9. Link Patch Protokol AntHocNet

```
set val(chan) Channel/WirelessChannel ;
set val(prop) Propagation/TwoRayGround ;
set val(netif) Phy/WirelessPhy ;
set val(mac) Mac/802_11 ;
set val(ifq) Queue/DropTail/PriQueue ;
set val(ll) LL ;
set val(ant) Antenna/OmniAntenna ;
set val(ifqlen) 50 ;
set ns_ [new Simulator]
 set topo           [new Topography]
 $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)
-channel  [new $val(chan)]
-agentTrace ON
-routerTrace ON
-macTrace OFF
-movementTrace OFF
-topoInstance $topo
```

Gambar 2.10. Contoh skrip lingkungan simulasi

File yang terunduh bernama Anthocnet_ns235.patch. Letakkan file tersebut pada direktori ns-allinone-2.35. Kemudian lakukan *patch* dengan perintah :

```
patch -p0 < Anthocnet_ns235.patch
```

Tahap selanjutnya adalah melakukan proses installasi NS-2. Pastikan lokasi direktori berada pada ns-allinone-2.35. Jika sudah berada pada direktori tersebut, lakukan proses installasi dengan perintah :

```
sudo make ./install
```

Aplikasi NS-2 yang telah dilakukan installasi diatas merupakan aplikasi *stand alone*, sehingga perlu dilakukan perintah dibawah ini agar dapat menjalankan perintah pada terminal. Perintah tersebut adalah :

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

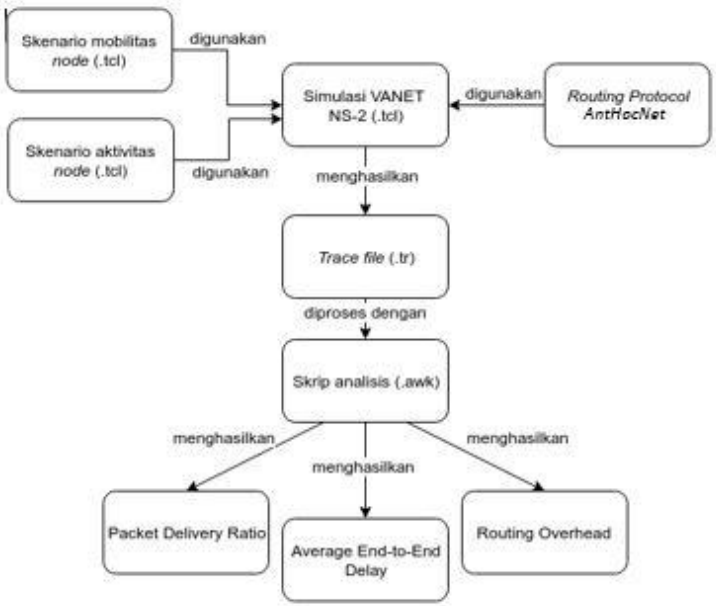
[Halaman ini sengaja dikosongkan]

BAB III
PERANCANGAN

Perancangan merupakan bagian yang sangat penting dari implementasi sistem. Bab ini secara khusus akan menjelaskan rancangan sistem yang dibuat pada Tugas Akhir. Bagian yang akan dijelaskan pada bab ini berawal dari deskripsi umum hingga perancangan skenario, alur dan implementasinya.

3.1. Deskripsi Umum

Pada Tugas Akhir ini akan dilakukan implementasi dan analisis dari routing protokol AntHocNet pada NS-2. Diagram rancangan simulasi dapat dilihat pada Gambar 3.1.



Gambar 3.1. Rancangan simulasi

Dalam Tugas Akhir ini, terdapat dua jenis skenario yang digunakan sebagian perbandingan pengukuran, yaitu peta berbentuk grid dan peta riil dari lingkungan lalu lintas kota Surabaya. Skenario *grid* dibuat dengan bantuan aplikasi SUMO. Skenario riil yang didasarkan pada peta lalu lintas kota Surabaya diambil dari OpenStreetMap dan dirapikan menggunakan aplikasi JOSM. Setelah file peta sudah terbentuk, dilakukan simulasi lalu lintas dengan SUMO. Hasil simulasi SUMO digunakan untuk simulasi protokol AntHocNet pada NS-2. Kemudian hasil simulasi NS-2 dianalisis dengan menggunakan script AWK untuk menghitung metrik analisis berupa *packet delivery ratio, average end-to-end delay, dan routing overhead.* Perhitungan *metric* analisis bertujuan untuk mengukur performa dari protokol AODV dan AntHocNet.

3.2. Perancangan Skenario *Grid*

Pembuatan peta grid diawali dengan menentukan panjang jalan dan jumlah *vertex*. Secara default, peta *grid* akan berbentuk segi empat. Alur pembuatan peta *grid* dapat dilihat pada Gambar 3.2.

Panjang jalan dan jumlah *vertex* yang sudah ditentukan akan dimasukkan sebagai argumen untuk setiap netgenerate.

Secara opsional, batas kecepatan untuk setiap jalan juga dapat ditentukan melalui argument netgenerate. Kemudian hasil dari netgenerate digunakan sebagai argumen randomTrips.py program randomTrips.py berfungsi untuk mendefinisikan rute perjalanan dari seluruh node. Keluaran dari randomTrips.py diproses oleh duarouter untuk memperbaiki masalah konektivitas rute (jika ada). Setelah itu dilakukan simulasi lalulintas dengan SUMO.

Hasil dari simulasi tersebut diekspor kedalam format yang bisa diproses oleh NS-2 melalui traceExport.py. hasilnya berupa file yang berisi mobilitas dari setiap node (mobility.tcl) dan informasi lifetime dari setiap node (activity.tcl).

3.3. Perancangan Skenario Riil

Perancangan scenario riil diawali dengan pemilihan daerah yang akan digunakan sebagai model untuk simulasi. Setelah

mendapatkan daerah yang diinginkan, unduh daerah tersebut disitus OpenStreetMap melalui fitur export

Kemudian potongan peta tersebut dirapikan dengan menggunakan program JOSM. Agar bisa semirip mungkin dengan dunia nyata, lakukan pengaturan pada interval lampu lalu lintas. Kemudian hapus jalan yang terputus dari potongan peta tersebut sehingga menjadi daerah yang tertutup.

Gambar 3.2.. Rancangan simulasi

Setelah potongan peta dirapikan, buat sebuah file type file yang mendefinisikan spesifikasi batasan simulasi lalu lintas, seperti batas kecepetan pada kelas jalan tertentu, dan lain-lain. Kemudian peta dikonversikan dengan netconvert berdasarkan *typefile* yang telah dibuat.

Hasil konversi tersebut digunakan untuk membuat file rute pergerakan kendaraan melalui randomTrips.py dan routeTrips.py. Kemudian file peta yang telah dikonversi menggunakan tools

netconvert yang menghasilkan file berekstensi .net.xml dan file yang berisi rute digunakan untuk simulasi SUMO.

Gambar 3.3. Alur Pembuatan Peta Riil

3.4. Perancangan Simulasi pada NS-2

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

3.5. Perancangan Metrik Analisis

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

3.5.1. Packet Delivery Ratio

Packet Delivery Ratio adalah perbandingan dari jumlah paket data yang dikirim dengan paket data yang diterima. *Packet Delivery Ratio* dihitung menggunakan persamaan seperti pada Persamaan 2, dimana *recv* adalah jumlah paket data yang diterima dan *sent* adalah jumlah paket data yang dikirim.

$$PDR = \frac{Data_{recv}}{Data_{sent}} \qquad (3)$$

3.5.2. Average End-to-End Delay

Average end-to-end delay adalah waktu rata-rata dari setiap paket ketika sampai ditujuan. Semua paket, termasuk *delay* yang diakibatkan oleh paket routing, juga akan diperhitungkan. Paket yang akan dimasukkan ke dalam perhitungan hanya paket yang berhasil sampai ditujuan. *Average end-to-end delay* dihitung menggunakan persamaan seperti pada Persamaan 3. , dimana i adalah nomor paket ketika paket i dikirim, $t_{sent}[i]$ adalah waktu ketika paket i diterima, dan pktCount adalah jumlah paket yang berhasil dikirim sampai ditujuan.

$$Delay = \frac{\sum_{i=0}^{n} t_{recv}[i] - t_{sent}[i]}{pktCount} \qquad (4)$$

3.5.3. Routing Overhead

Routing Overhead adalah jumlah paket routing kontrol yang ditransmisikan selama simulasi terjadi. Paket kontrol yag dihitung adalah jumlah AGT dan RTR. *Routing Overhead* adalah jumlah paket routing diperlukan untuk komunikasi jaringan.

[Halaman ini sengaja dikosongkan]

BAB IV
IMPLEMENTASI

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

4.1. Implementasi Skenario Grid

Skenario *grid* dibuat melalui *tool* netgenerate dari SUMO. Skenario *grid* dibuat dengan panjang jalan 200 dan luas peta 2600m x 2600m. Jumlah titik persimpangan antara jalan vertikal dan horizontal sebanyak 6 titik x 6 titik. Kecepatan kendaraan yang diperbolehkan diatur sebesar 15 m/s.

Untuk membuat peta *grid* dengan spesifikasi tersebut, digunakan perintah seperti Gambar 4.1.

```
netgenerate –grid –grid.number=6 –grid.length=200 –
default.speed=15 –tls.guess=1 –o map.net.xml
```

Gambar 4.1. Sintax netgenerate

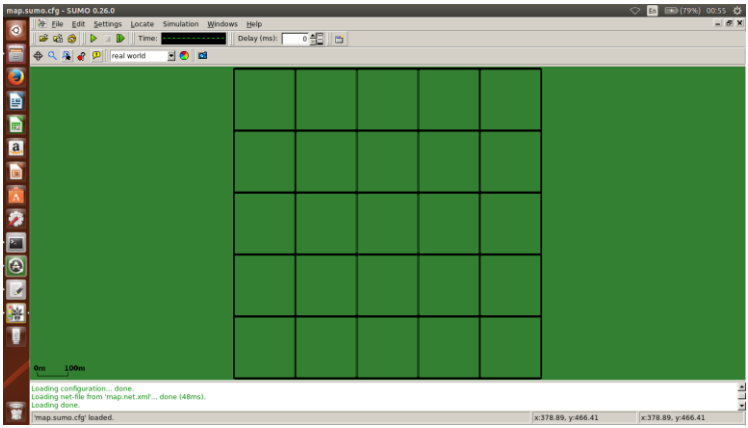
Setelah peta terbentuk, dilakukan pembuatan titik asal dan titik tujuan untuk setiap kendaraan secara acak melalui modul randomTrips.py seperti pada Gambar 4.2.

Opsi –e diisi dengan jumlah kendaraan yang diinginkan, sedangkan –intermediate diisi dengan nilai yang besar agar setiap kendaraan memiliki banyak rute alternative sehingga setiap kendaraan dapat dipastikan aktif hingga simulasi mobilitas berakhir.

Setelah titik akhir dan titik asal didefinisikan, maka dilakukan pembuatan rute yang akan digunakan oleh kendaraan menggunakan perintah pada Gambar 4.4.

```
python $SUMO_HOME/tools/randomTrips.py –seed=$RANDOM –fringe-
factor 5.0 –e num_nodes –intermediate=$RANDOM –n map.net.xml –
r map.passenger.rou.xml –vehicle-class passenger –trip-
atributes 'departLane="best"'
```

Gambar 4.2. Sintaks pembuatan titik asal dan tujuan



Gambar 4.3.. Hasil netgenerate

```
duarouter -n $SAVEDIR/map.net.xml -t $SAVEDIR/trips.trips.xml -o $SAVEDIR/route.rou.xml –ignore-errors –repair
```

Gambar 4.4. Perintah untuk membuat Rute

Selanjutnya dilakukan pembuatan file.sumo.cfg yang akan digunakan sebagai argument perintah sumo. Gambar 4.5 menunjukkan isi dari file.sumo.cfg.

File sumo.cfg disimpan pada direktori yang sama dengan .net..xml dan .trips.xml. File .sumo.cfg digunakan untuk mendefinisikan lokasi file .net.xml dan .trips.xml serta durasi simulasi. Untuk melihat visualisasi simulasi lalu lintas, file .sumo.cfg dapat dibuka melalui sumo-gui. Kemudian lakukan simulasi lalu lintas dengan perintah seperti pada script berikut.

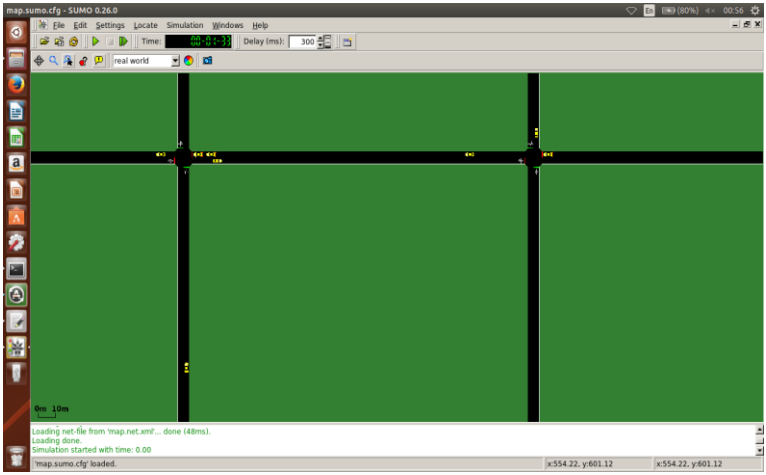
Agar dapat digunakan di NS-2, keluaran dari perintah sumo harus dikonversikan ke format yang dapat dipahami oleh NS-2 melalui perintah pada Gambar 4.7.

```
python $SUMO_HOME/tools/traceExporter.py –fcd-input simulation-result.xml –ns2mobility-output mobility.tcl –ns2config-output config.tcl
```

Gambar 4.7. Sintaks untuk konversi keluaran dari sumo

```
      sumo-gui -c map.sumo.cfg -fcd-output simulation.xml
<configuration>
       <input>
              <net-file value="map.net.xml"/>
              <route-files value="route.rou.xml"/>
       </input>
       <time>
              <begin value="0"/>
              <end value="360"/>
       </time>
</configuration>
```

Gambar 4.5. Isi file sumo.cfg

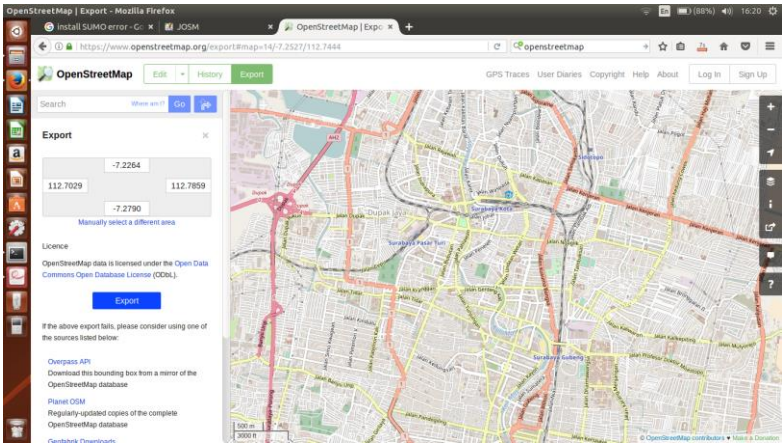


Gambar.4.6. Hasil Simulasi Lalu Lintas

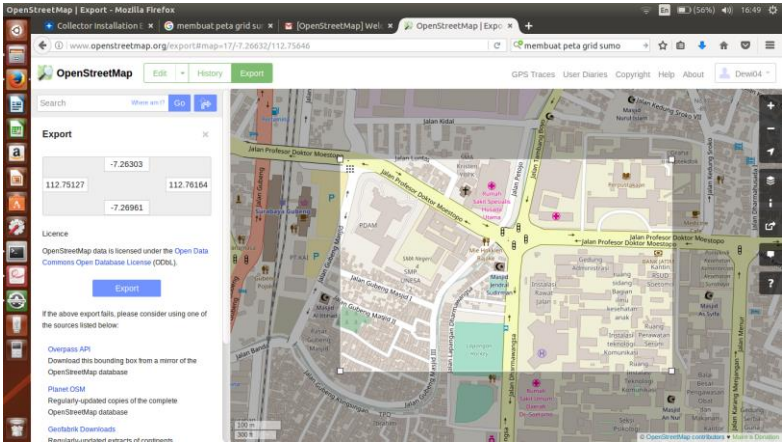
4.2. Implementasi Skenario Riil

Skenario riil menggunakan bagian peta wilayah kota Surabaya yang diambil dari OpenStreetMap. Peta diambil dengan cara membuat area seleksi wilayah kemudian diekspor dalam bentuk file .osm melalui browser seperti pada Gambar 4.8. Peta yang telah diekspor dari OpenStreetMap kemudian diubah dengan

menggunakan program JOSM. Tujuan dari penyuntingan adalah menghapus jalan yang tidak digunakan.

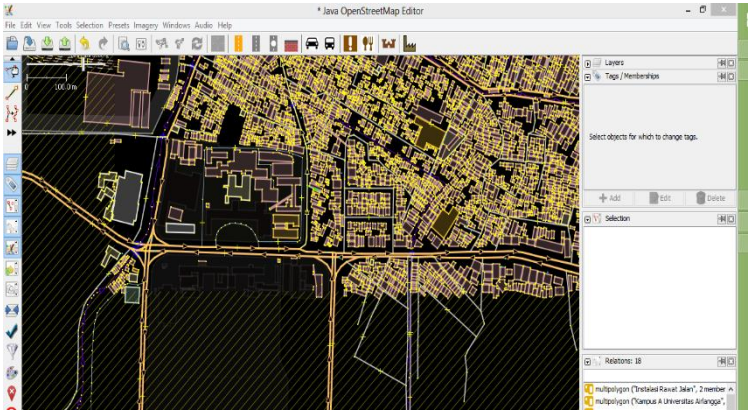


Gambar 4.8. Pemilihan Lokasi dari OpenStreetMap

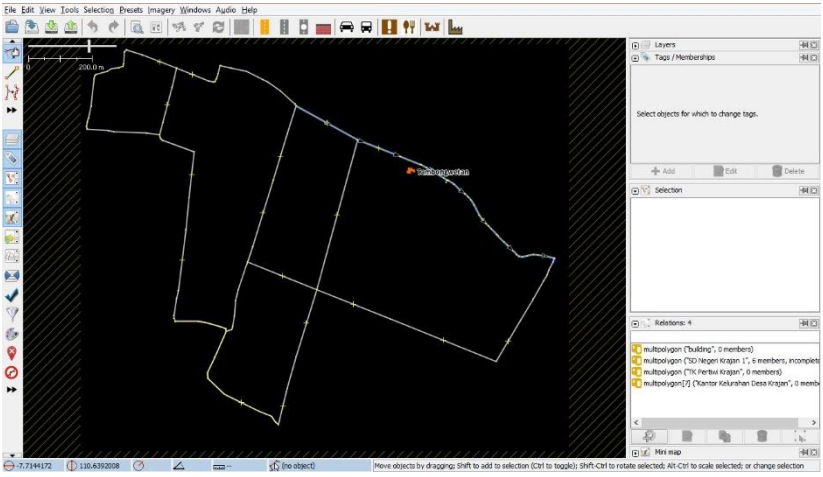


Gambar 4.9. Pengambilan Peta dari OpenStreetMap

Beberapa jalan baru juga ditambahkan agar tidak ada jalan yang buntu dan kepadatan jalan tetap stabil sehingga tidak ada daerah pada peta yang jarang dikunjungi oleh kendaraan. Screenshot dari proses penyuntingan dapat dilihat pada Gambar 4.11.



Gambar 4.10. Penyuntingan Peta dengan JOSM



Gambar 4.11. Hasil Penyuntingan Peta dengan JOSM

Setelah proses penyuntingan peta selesai, peta tersebut dikonversikan ke dalam format .net.xml menggunakan tool netconvert.

```
netconvert –osm-files map.osm –o map.net.xml –remove-
edges.isolated –type-files osmNetconvert.typ.xml
```

Gambar 4.12. Perintah untuk konversi .osm ke .net.xml

Setelah peta terbentuk, langkah selanjutnya sama seperti tahapan membuat skenario grid mulai dari pembuatan titik asal dan titik tujuan dengan randomTrips.py sampai konversi ke dalam format yang dapat digunakan di NS-2.

4.3. Implementasi Metrik Analisis

Hasil dari simulasi skenario dalam NS-2 adalah sebuah *trace file*. *Trace file* digunakan sebagai bahan analisis untuk pengukuran performa dari protokol yang disimulasikan. Dalam Tugas Akhir ini, terdapat tiga *metric* yang akan mejadi parameter analisis, yaitu *packet delivery ratio, average end-to-end delay, dan routing overhead.*

4.3.1. Implementasi Packet Delivery Ratio

Packet Delivery Ratio didapatkan dengan cara menghitung setiap baris terjadinya *event* pengiriman dan penerimaan paket data yang dikirim melalui agen pada *trace file*. Pada lampiran di tunjukkan cara menyaring data yang dibutuhkan untuk perhitungan PDR.

Pada skrip tersebut dilakukan penyaringan pada setiap baris yang mengandung string AGT karena event tersebut berhubungan dengan paket data. Paket yang dikirim dan diterima dibedakan dari kolom pertama pada baris yang telah disaring. Setelah pembacaan setiap baris diselesaikan, selanjutnya dilakukan perhitungan PDR dengan persamaan (3).

Contoh perintah yang digunakan untuk memanggil skrip AWK untuk menganalisis *trace file* dan contoh keluarannya dapat dilihat pada skrip berikut.

```
awk –f pdr.awk simple.tr
```

Berikut adalah tampilan hasil pdr.

```
s : 890 r : 879 ratio : 0.9876
```

4.3.2. Implementasi Average End-to-End Delay

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

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

Contoh perintah untuk memanggil skrip AWK untuk menganalisa *trace file* dan contoh keluarannya dapat dilihat pada script dibawah ini.

```
awk –f delay.awk simple-ant.tr
```

Berikut adalah tampilan hasil output end to end delay.

```
Delay : 344.66
```

4.3.3. Implementasi Routing Overhead

Routing Overhead didapatkan dengan cara menyaring setiap baris yang mengandung string request, reply,dan error. Setiap ditemukannya string tersebut, dilakukan *increment* untuk menghitung

jumlah paket routing yang tersebar dijaringan. Kode implementasi dari perhitungan *routing overhead* dapat dilihat pada lampiran.

```
awk –f ro.awk simple.tr
```

Berikut adalah hasil tampilan Routing Overhead.

```
Overhead : 5598
```

4.4 Implementasi Protokol AntHocNet

Kode implementasi dari protokol AntHocNet pada NS-2 versi 2.35 berada pada direktori ns-2.35/anthocnet. Daftar kode sumber yang harus diperhatikan pada direktori tersebut adalah sebagai berikut :

- Implementasi struktur paket *Hello*
- Implementasi perhitungan *pheromone*
- Implementasi *update pheromone*
- Implementasi *pheromone table*
- Implementasi *Route Discovery* AntHocNet

Pada bagian ini, penulis akan menjelaskan langkah-langkah dalam implementasi protokol AntHocNet.

4.4.1 Implementasi Struktur Paket Hello

Pada implementasi routing protokol AntHocNet di NS-2 juga menggunakan paket HELLO sama seperti halnya AODV. Pada protokol AntHocNet, terdapat *field* yang harus ada di dalam struktur paket HELLO, yaitu *pheromone*, koordinat x dan koordinat y. Untuk itu dibutuhkannya atribut *pheromone* pada ant_packet.h seperti pada Gambar 4.13.

```
class AntBackPacket: public AntBasicPacket
{
        protected:

        AntTimeEntry**   visitedNodesHist;   //   this   will
simulate a stack for ant
        int sizeNodesHist ; // number of nodes in list
```

```
        nsaddr_t prevHop; // previous node
        int hops; // count numbers of hops to current node
        double prevSINR; // Hop time estimated at the mac
layer of the previous node
        double pheromone; // value of pheromone

        public:

        AntBackPacket()
{
        prevHop = -1;
        hops = 0;
        prevSINR = 0;
        pheromone = 0;
}
```

Gambar 4.13. Penambahan atribut pada ant_packet.h

Penambahan atribut nilai *pheromone* juga dilakukan pada file ant_types.h. penambahan tersebut berupa phValue seperti pada gambar 4.14.

```
class AntNode {
private:
        int node_;
        double phValue_;

public:
        AntNode(nsaddr_t node, double phValue) {
                this->node_ = node;
                this->phValue_ = phValue;
        }

        nsaddr_t node() const {
                return node_;
        }
        double phValue() const {
                return phValue_;
        }
};
```

Gambar 4.14. Penambahan atribut pada ant_types.h

Pengiriman paket Hello terdiri dari pebuatan paket, pengisian atribut paket, dan penjadwalan *event* pengiriman dalam NS-2. Pada AntHocNet, paket Hello digunakan untuk mengirim informasi mengenai pheromone dari *node* pengirim.

Informasi *pheromone* dari *node* yang akan mengirim paket Hello didapatkan dari kelas AntBackPacket.

4.4.2 Implementasi Perhitungan Pheromone

Tahap perhitungan *pheromone* membutuhkan informasi – ant_packet.cc. Perhitungan *pheromone* membutuhkan informasi berupa prevSINR yang merupakan estimasi *hop time*, *hops* yang merupakan jumlah *hops* pada *node* tertentu dan waktu. Jika informasi prevHops dan waktu telah didapatkan, tahap selanjutnya adalah mencari nilai simpleSINR, yaitu dengan melakukan pembagian terhadap prevSINR dengan nilai 1000. Implementasi dapat dilihat seperti pada Gambar 4.15.

```
AntTimeEntry* entry = visitedNodes[sizeNodes_-1];

 prevHop = entry->node();
 hops++;
 double time = entry->time();
 prevSINR += time;

 double simpleSINR = prevSINR / 1000;

// calculate pheromone to used to destination
 int hTime = AntHocNetUtils::HOP_TIME;
 pheromone = pow( ((hops*hTime + simpleSINR) / 2) , -1);

// add visited nodes history
if (visitedNodesHist == NULL) {
        visitedNodesHist                                        =
(AntTimeEntry**)malloc(sizeof(AntTimeEntry*) * 100);
        sizeNodesHist_ = 0;
}

visitedNodesHist[sizeNodesHist_++] = new AntTimeEntry(entry-
>node(), entry->time());
        // get next neighbor
        sizeNodes_--;
        entry = visitedNodes[sizeNodes_-1];

#ifdef DEBUG
        fprintf(stdout, "AntBackPacket::findNextHop - seqnum:
%d:%d >> sddr (%d) == prev (%d) >> time (%f) -- pheromone
(%f)    --    vizited    size    %d    next    %d    \n",    this-
>getSorceAddress(),    this->getSeqNum(),    sddr,    prevHop,
prevSINR, pheromone, sizeNodes_, entry->node());
```

```
for (int i=0; i< sizeNodes_; i++) {
 AntTimeEntry* entry = visitedNodes[i];
 fprintf(stdout, " ---- entry %d => time %d \n", entry-
>node(), entry->time());
}

return entry->node();
```

Gambar 4.15. Penambahan perhitungan formula pada ant_packet.cc

4.4.3 Implementasi Update Pheromone

Tujuan dari *pheromone update* adalah menambahkan nilai *pheromone* dengan solusi terbaru dan mengurangi nilai *pheromone* yang buruk. Nilai *pheromone* akan berubah karena terjadi perhitungan pada *pheromone evaporation* dan berkurangnya tingkat *pheromone* karena pemilihan solusi terbaik. Implementasi dari *pheromone update* dapat dilihat pada Gambar 4.16.

```
nsaddr_t neighbor = ah->getSorceAddress();
AntNode** dests = ah->getDestinations();
int size = ah->sizeDests();

for (int i=0; i< size; i++) {
        AntNode* node = dests[i];
        nsaddr_t destination = node->node();
        double       phValue       =       pheromoneTable()-
>getPheromoneVirtual(destination, neighbor);
        double phUpdate = node->phValue();
        double  pIncress  =  this->incressPheromone(phValue,
phUpdate);
        pheromoneTable()->setPheromoneVirtual(destination,
neighbor, pIncress)}

        return pheromoneTable();
```

Gambar 4.16. Implementasi Pheromone Update

4.4.4 Implementasi Pheromone Table

Implementasi *request table* digunakan untuk mengurangi *routing overhead* sebagaimana yang digunakan pada protokol – protokol yang lain. File yang mengimplementasikan *request table*

adalah ahn_ant_nest.h, ahn_ant_nest.cc. *Request table* digunakan untuk mengurangi *routing overhead* pada proses pencarian rute. *Request table* bersifat lokal, dimana setiap *node* memiliki *request table* masing – masing sehingga setiap *node* satu dengan *node* yang lain memiliki *request table* yang berbeda.

Request table pada protokol antHocNet berisi daftar *node-node* inisiator yang telah mengirimkan *forwardant* dan *backant* kepada *node* tersebut beserta idnya. Berikut adalah implementasi dari *request table* pada protokol anthocnet seperti pada Gambar 4.17.

```
class AntNest
{
friend class AntHocNet;
private:
        PheromoneTable* pheromone_rt_;
        MapAntPQueue mapQueue;

        AntHistoryList history; // history of ant packet to
send by node
nsaddr_t addr_; // address of the agent
        u_int8_t  ant_seq_num_;  //  sequence  number  for  ant
packets

        PheromoneTable* pheromoneTable() const;

        double evaporatePheromone(double phValue);
        double   incressPheromone(double   phValue,   double
phUpdate);

public:
        AntNest(nsaddr_t addr)
{
        pheromone_rt_ = new PheromoneTable();
        ant_seq_num_ = 0;
        addr_ = addr;
}
```

Gambar 4.17. Implementasi Pheromone Table

4.4.5 Implementasi Route Discovery pada AntHocnet

Pada fase ini rute baru akan dibuat dengan bantuan forwardant dan backwardant. Forwardant adalah agen yang membentuk jalur pheromone ke node sumber, sedangkan backwardant membentuk jalur pheromone menuju node tujuan.

Forwardant dan backwardant adalah paket kecil dengan urutan nomor yang unik. Sehingga node dapat membedakan paket berdasarkan nomor urutan dan alamat sumber daro forwardant dan backwardant.

Sebuah forwardant akan di broadcast oleh pengirim dan akan disebarkan ke node tetangga. Node yang menerima forwardant akan membuat catatan pada routing table mengenai alamat tujuan, hop selanjutnya, dan nilai pheromonenya. Dimana node menerjemahkan alamat sebelumnya sebagai next hop dan menghitung nilai pheromone berdasarkan jumlah hop yang dibutuhkan forwardant untuk mencapai tujuan node ini. Forwardant melewati node yang sama akan dideteksi melalui nomor urutannya maka informasinya akan diekstrak dan langsung dihancurkan. Selanjutnya membuat backwardant untuk dikirm ke node pengirim. Backwardant mempunyai tugas yang sama dengan forwardant, yaitu membuat jalur menuju node. Ketika node pengirim menerima backwardant dari node tujuan, maka jalur terbentuk dan paket data dapat dikirimkan.

4.5 Implementasi Simulasi pada NS-*2*

Untuk melakukan simulasi VANET pada NS-2, dibutuhkan sebuah file OTcl yang berisi deskripsi dari lingkungan simulasi. File tersebut berisikan pengaturan untuk setiap *node* dan beberapa *event* yang perlu diatur agar berjalan pada waktu tertentu. Contoh potongan skrip pengaturan *node* pada dapat dilihat pada Gambar 4.13.

Penjelasan dari pengaturan node dapat dilihat pada Tabel 4.1 pengaturan lainnya yang dilakukan pada file tersebut antara lain lokasi penyimpanan *trace file*, lokasi mobilitas *node*, konfigurasi *node* sumber dan *node* tujuan, dan konfigurasi *event* pengiriman pada data. Kode implementasi skenario yang digunakan pada Tugas Akhir ini dapat dilihat pada Lampiran.

Setelah simulasi selesai akan ada keluaran berupa *tracefile* hasil simulasi yang akan digunakan untuk analisis. Isi dari *tracefile* adalah catatan seluruh *event* yang dari setiap paket yang tersebar didalam lingkungan simulasi.

ns-anthocnet ant.tcl

```
set val(chan) Channel/WirelessChannel;
set val(prop) Propagation/TwoRayGround;
set val(netif) Phy/WirelessPhy;
set val(mac) Mac/802.11;
set val(ifq) Queue/DropTail/PriQueue;
set val(ll) LL;
set val(ant) Antenna/OmniAntenna;
set val(ifqlen) 50;
set val(rp) AntHocNet;
set val(nn) 350;
set val(stop) 360;

set val(cbrsize) 512;
set val(cbrrate) 2KB;
set val(cbrinterval) 1;

set val(x) 2600;
set val(y) 2600;

set ns_ [new Simulator]
set topo [new Topography]
$topo load_flatgrid $val(x) $val(y)

$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)
–channel [new $val(chan)] –agentTrace ON –routerTrace ON –
macTrace OFF –movementTrace OFF –topoInstance $topo
```

Gambar 4.18. Potongan Skrip Pengaturan Node

Tabel 4.1. Parameter Pengaturan Node

Parameter	Value	Penjelasan
llType	LL	Menggunakan link layer standart
macType	Mac/802.11	Menggunakan tipe MAC 802.11 kerena komunikasi data bersifat wireless
ifqType	Queue/ DropTail/ PriQueue	Menggunakan priority queue sebagai antrian paket dan paket yang akan dihapus saat antrian penuh adalah paket yang paling baru

ifqLen	50	Jumlah maksimal paket pada antrian
antType	Antenna/ OmniAntenna	Jenis antenna yang digunakan adalah Omni Antenna
propType	Propagation/ TwoRayGround	Tipe propagasi sinyal wireless adalah two-ray ground
phyType	Phy/ WirelessPhy	Komunikasi menggunakan media nirkabel
topoInstance	$topo	Topologi yang digunakan untuk menjalankan skenario
agentTrace	ON	Mengaktifkan pencatatan aktifitas dari agent routing protokol
routerTrace	ON	Mengaktikan pencatatan pada aktifitas routing protokol
macTrace	OFF	Matikan pencatatan MAC layer pada *trace file*
movementTrace	OFF	Matikan pencatatan pergerakan *node*
channel	Channel/Wireless Chaannel	Kanal komunikasi yang digunakan

[Halaman ini sengaja dikosongkan]

BAB V
UJI COBA DAN EVALUASI

Pada bab ini akan dibahas mengenai uji coba dan evaluasi hasil simulasi dari skenario NS-2 yang telah dibuat.

5.1. Lingkungan Uji Coba

Spesifikasi perangkat keras yang digunakan pada Tugas Akhir ini dapat dilihat pada Tabel 5.1.

Tabel 5.1. Spesifikasi perangkat.

Komponen	Spesifikasi
CPU	Intel®Core™i5-4200U@2.30GHz × 8
Sistem Operasi	Linux Ubuntu 14.04 LTS 64-bit
Memori	8GB PC3-12800 DDR3L SDRAM 1600MHz
Penyimpanan	1 TB

Adapun versi perangkat lunak yang digunakan dalam Tugas Akhir ini, sebagai berikut :

- SUMO-0.26.0 untuk pembuatan skenario mobilitas VANET.
- JOSM-12039 untuk penyuntingan peta OpenStreetMap.
- Network Simulator-2.35 untuk simulasi skenario VANET.

Parameter lingkungan uji coba yang digunakan pada NS-2 dapat dilihat pada Tabel 5.2.

Pengujian dilakukan dengan menjalankan skenario melalui NS-2. Hasil dari skenario yang sudah dijalankan berupa *tracefile* yang akan dianalisis dengan skrip AWK.

Tabel 5.2. Parameter lingkungan simulasi

No.	Parameter	Spesifikasi
1	Network Simulator	NS-2.35
2	Routing Protokol	AODV dan AntHocNet
3	Waktu Simulasi	900 detik
4	Area Simulasi	2600 m x 2600 m (grid) 2600 m x 2600 m (riil)
5	Jumlah Kendaraan	100, 150, 200, 250, 300,

		350, 400
6	Radius Transmisi	250 m
7	Kecepatan maksimum (grid)	15 m/s
8	Agent	Constant Bit Rate
9	Source /Destination	Stasionary
10	Ukuran Paket	600 Bytes
11	Packet Rate	2kB/s
12	Packet Interval	1 paket/detik
13	Protokol MAC	IEEE 802.11
14	Model Propagasi	Two-Ray Ground

5.2. Hasil Uji Coba Skenario Grid

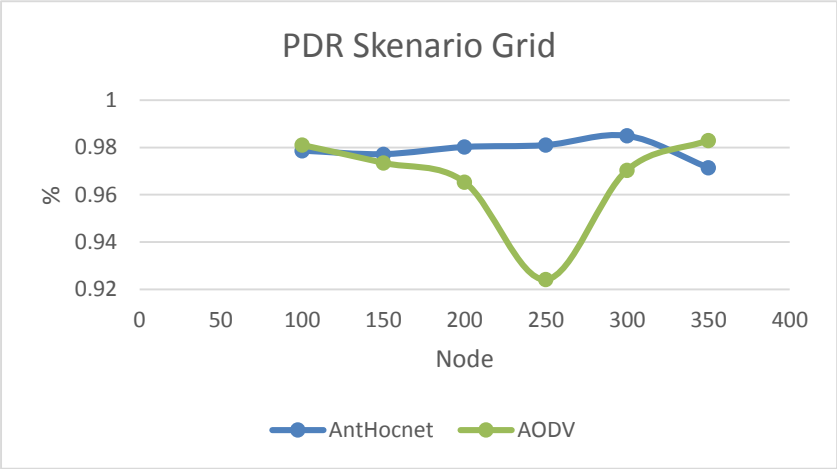
Uji coba skenario dilakukan pada mobilitas random pada peta *grid* dengan luas area 2600 m x 2600 m dengan jumlah *node* sebanyak 100, 150, 200, 250, 300, dan 350. Kecepatan maksimum dari setiap *node* adalah 15 m/s. hasil analisis dengan metric *packet delivery ratio* (PDR), average *end-to-end delay*, dan *routing overhead* dapat dilihat pada Tabel 5.3., 5.4. dan 5.5. Berikut adalah tabel yang menunjukkan hasil *average* PDR skenario grid pada AntHocNet dan AODV.

Tabel 5.3.Hasil Perhitungan Rata –Rata PDR pada Skenario Grid

Node	AntHocnet	AODV	Perbedaan
100	0.978	0.981	0.002
150	0.977	0.973	0.003
200	0.980	0.965	0.014
250	0.981	0.924	0.056
300	0.984	0.970	0.014
350	0.971	0.983	0.011

Dari data diatas, dapat dilihat bahwa AntHocNet memiliki rata-rata PDR lebih baik dibandingkan dengan AODV. Perbedaan PDR pada AntHocnet terhadap AODV tidak begitu terlihat. Pada

jumah *node* 100, AntHocNet memiliki PDR 0.2% lebih buruk dari dibandingkan dengan AODV karena pada AntHocNet lebih mengutamakan penyebaran *pheromone* untuk mengenali setiap node.



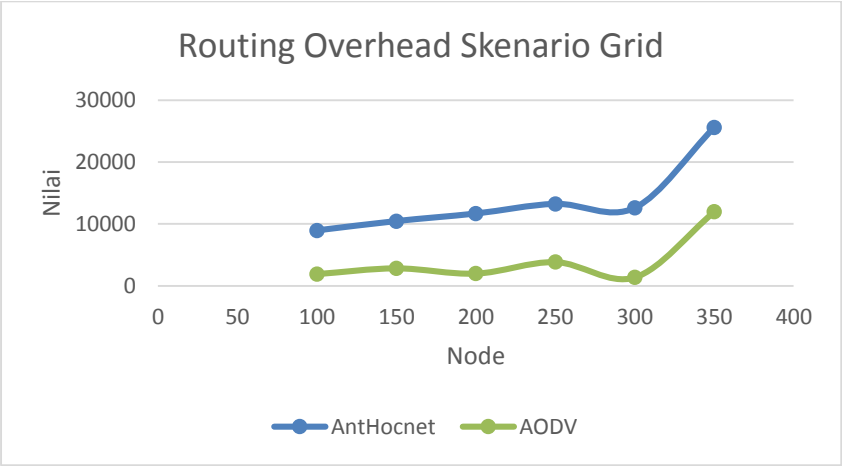
Gambar 5.1. Grafik PDR pada Grid

Pada jumah *node* 150, AntHocNet memiliki PDR 0.3 % lebih baik dari dibandingkan dengan AODV. Pada jumah *node* 200, AntHocNet memiliki PDR 1.48 % lebih baik dari dibandingkan dengan AODV. Pada jumah *node* 250, AntHocNet memiliki PDR 5.69 % lebih baik dari dibandingkan dengan AODV. Pada jumah *node* 300, AntHocNet memiliki PDR 1.4 % lebih baik dari dibandingkan dengan AODV. Pada jumah *node* 350, AntHocNet memiliki PDR 1.1 % lebih rendah dari dibandingkan dengan AODV.

Pada Tabel 5.4 menunjukkan hasil *routing overhead* pada protocol AntHocNet dan AODV. *Routing overhead* pada AntHocNet lebih banyak daripada AODV. Akan tetapi pada AODV terjadi kenaikan yang signifikan pada *node* 350. Sedangkan protokol AntHocNet tetap stabil sekalipun nilai *routing overhead* lebih banyak dibandingkan dengan AODV.

Tabel 5.4 Hasil Perhitungan Rata-Rata RO pada Skenario Grid

Node	AntHocnet	AODV	Perbedaan
100	8936	1909	7027
150	12332	2840	9492
200	14956	1991	12965
250	13236	3848	9388
300	12594	1379	11215
350	25608	11971	13637



Gambar 5.2. Grafik RO pada Grid

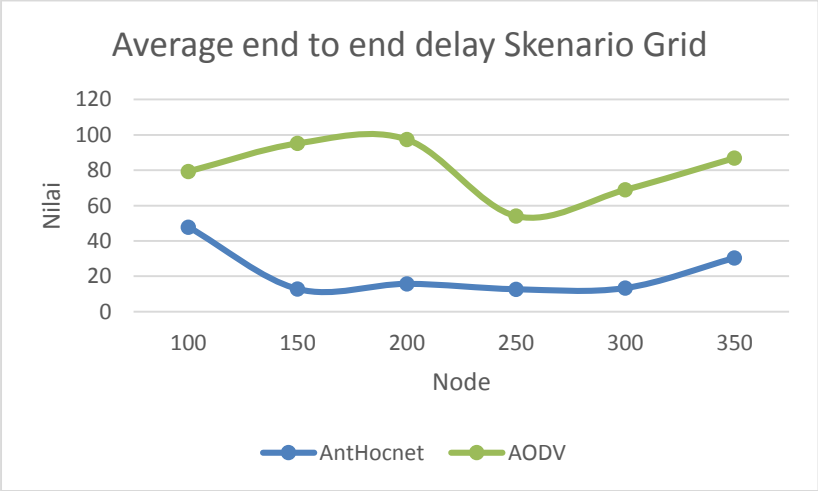
Pada AntHocNet nilai *routing overhead* akan bertambah sesuai dengan jumlah *node* yang dijalankan. Semakin banyak *node*, maka semakin tinggi pula *routing overheadnya*. Pada Tabel 5.5 menunjukkan hasil *average end to end delay* pada protokol AntHocNet dan AODV.

Pada hasil uji coba skenario AntHocNet menunjukkan bahwa delay pada routing AntHocNet lebih rendah dibandingkan dengan AODV meskipun pada awal routing menunjukkan nilai delay yang tinggi akan tetapi pada nilai selanjutnya lebih stabil dan lebih baik.

Average End-to-end delay pada protokol AntHocNet akan tinggi nilainya ketika di awal. Sedangkan untuk nilai selanjutnya akan semakin kecil, karena nilai *pheromone* yang besar. Dan akan naik lagi ketika jumlah *node* yang banyak seperti yang tertera pada tabel.

Tabel 5.5. Hasil Perhitungan Rata-Rata Delay pada Skenario Grid

Node	AntHocnet	AODV	Perbedaan
100	47.831	31.513	16.32
150	12.839	82.351	66.27
200	15.719	81.667	36.49
250	12.668	41.387	28.72
300	13.345	55.567	42.22
350	30.342	56.528	26.28



Gambar 5.3. Grafik Delay pada Grid

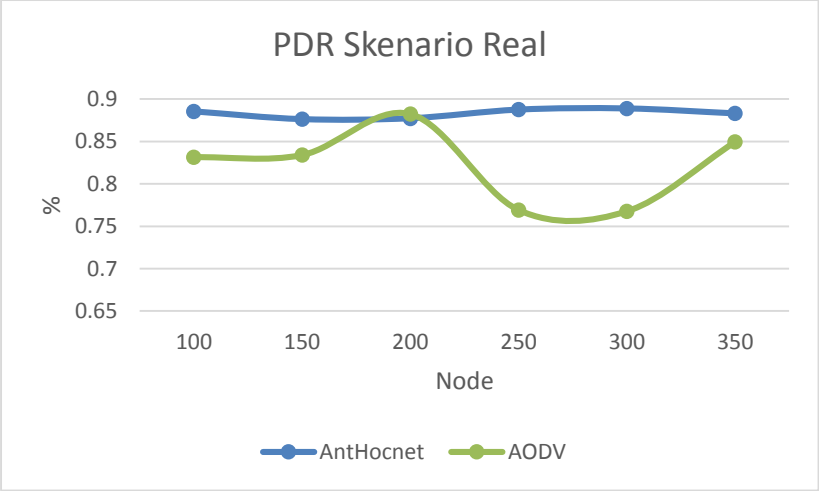
5.3. Hasil Uji Coba Skenario Riil

Uji coba skenario dilakukan pada *mobilitas random* pada peta riil dengan luas area 2600 m x 2600 m dengan jumlah *node*

sebanyak 100, 150, 200, 250, 300, dan 350. Kecepatan maksimum dari setiap *node* adalah 15 m/s. hasil analisis dengan *metric packet delivery ratio* (PDR), *end-to-end delay*, *dan routing overhead* dapat dilihat pada Tabel 5.6.

Tabel 5.6. Hasil Perhitungan Rata-Rata PDR pada Skenario Riil

Node	AntHocnet	AODV	Perbedaan
100	0.885	0.831	0.053
150	0.876	0.833	0.042
200	0.877	0.882	0.005
250	0.887	0.769	0.118
300	0.888	0.767	0.121
350	0.882	0.849	0.033



Gambar 5.4. Grafik PDR pada Skenario Riil

Dari data diatas, dapat dilihat bahwa AntHocNet memiliki rata-rata PDR lebih baik dibandingkan dengan AODV. Perbedaan PDR pada AntHocnet terhadap AODV tidak begitu besar. Pada

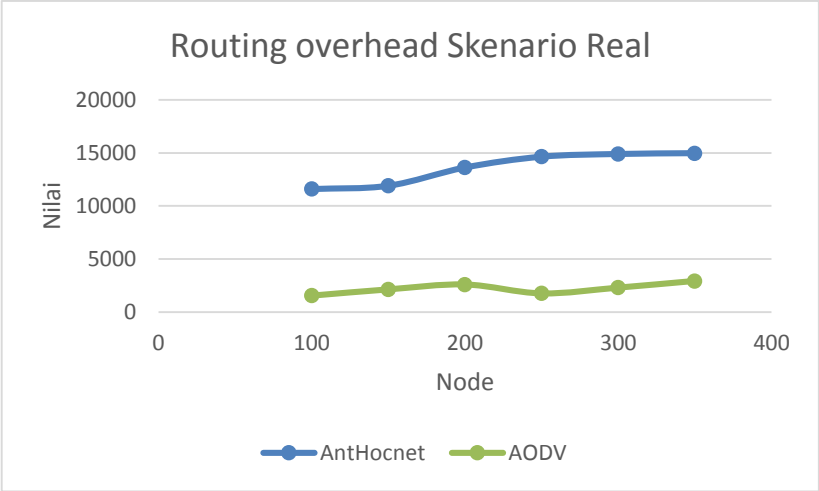
jumah *node* 100, AntHocNet memiliki PDR 5.3 % lebih baik dari dibandingkan dengan AODV. Pada jumah *node* 150, AntHocNet memiliki PDR 4.2 % lebih baik dari dibandingkan dengan AODV. Pada jumah *node* 200, AntHocNet memiliki PDR 0.5 % lebih baik dari dibandingkan dengan AODV. Pada jumah *node* 250, AntHocNet memiliki PDR 11.8 % lebih rendah dari dibandingkan dengan AODV. Pada jumah *node* 300, AntHocNet memiliki PDR 12.1 % lebih rendah dari dibandingkan dengan AODV. Pada jumah *node* 350, AntHocNet memiliki PDR 3.34 % lebih baik dari dibandingkan dengan AODV. Tabel 5.7 berikut menunjukkan hasil average routing overhead pada scenario real .

Tabel 5.7. Hasil Perhitungan Rata-Rata RO pada Skenario Riil

Node	AntHocnet	AODV	Perbedaan
100	11604	1549	10055
150	11913	2132	9781
200	13627	2589	11038
250	14646	1761	12885
300	14902	2297	12605
350	14977	2918	12059

Pada Tabel 5.7 menunjukkan hasil *routing overhead* pada protokol AntHocNet dan AODV. Pada Tabel tersebut menunjukkan informasi bahwa *routing overhead* pada AntHocNet, dimana semakin banyak *node,* maka semakin tinggi *routing overhead.* Berbeda dengan *routing overhead* pada protokol AODV yang cenderung konsisten.

Pada *routing overhead* AODV peningkatan nilai routing overhead bersesuaian dengan dengan banyaknya node. Semakin tinggi node maka semakin tinggi nilai routing overhead. Pada dasarnya tingkat routing overhead pada routing AntHocNet juga menigkat seiring dengan semakin banyaknya node. Akan tetapi juga dipengaruhi oleh nilai mobilitas yang ada. Tabel 5.8 berikut menunjukkan hasil average end to end delay pada scenario real.

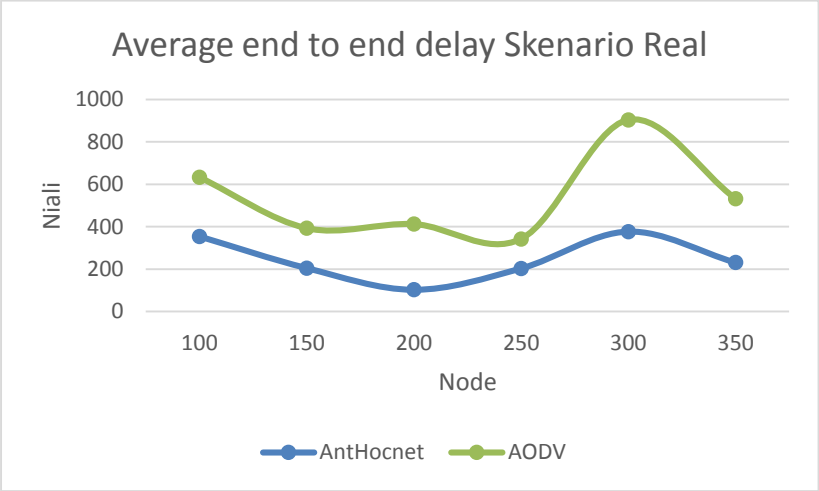


Gambar 5.5. Grafik RO pada Skenario Riil

Tabel 5.8. Hasil Perhitungan Rata-Rata Delay pada Skenario Riil

Node	AntHocnet	AODV	Perbedaan
100	354.19	278.79	75.37
150	204.85	187.79	17.06
200	102.49	309.43	206.94
250	202.89	139.14	63.74
300	376.84	526.72	149.88
350	230.74	301.15	70.41

Pada Tabel 5.8 menunjukkan hasil perhitungan rata –rata delay protokol AntHocNet dan AODV pada skenario riil. Pada skenario riil, *end-to-end delay* protokol AntHocNet lebih besar karena setiap *node* harus dilalui. Average End-to-end delay pada protokol AntHocNet akan tinggi nilainya ketika di awal. Sedangkan untuk nilai selanjutnya akan semakin kecil, karena nilai *pheromone* yang besar. Dan akan naik lagi ketika jumlah *node* yang banyak seperti yang tertera pada tabel.



Gambar 5.6. Grafik Delay pada Skenario Riil

Pada skenario *grid* merupakan skenario dengan bentuk yang stabil dan teratur, sedangkan pada skenario riil merupakan skenario daerah surabaya yang sesuai dengan keadaan lalu lintas. Nilai skenario *grid* untuk *end-to-end delay* pada protokol AntHocNet tidak selamanya memberikan hasil *end-to-end delay* yang baik karena bisa jadi jarak yang ditempuh lebih jauh daripada menggunakan protokol AODV. Selain itu, proses *route discovery* dari AntHocNet juga meningkatkan *end-to-end delay* karena setiap *node* harus disinggahi.

[Halaman ini sengaja dikosongkan]

BAB VI
PENUTUP

Pada bab ini akan diberikan kesimpulan yang dambil selama pengerjaan Tugas Akhir ini serta saran - saran tentang pengembangan yang dapat dilakukan terhadap Tugas Akhir ini.

6.1. Kesimpulan

Kesimpulan yang diperoleh pada uji coba dan evaluasi adalah sebagai berikut :

1. Hasil PDR protokol AntHocNet pada skenario *grid* dan skenario riil lebih baik daripada protokol AODV. *Packet delivery ratio* protokol Anthocnet pada skenario *grid* peningkatan dan penurunan dari 0.02 % hingga 5% dan pada skenario riil meningkat dari 0.08 % hingga 10%. *Packet delivery ratio* protokol AntHocNet dan AODV meningkat 1% seiring dengan bertambahnya *node* dalam jaringan.
2. Hasil routing overhead pada routing protokol AntHocnet lebih buruk dengan peningkatan nilai overhead mulai 7027 hingga 13637 pada skenario grid. Sedangkan pada skenario real peningkatan nilai overhead mulai dengan nilai 9781 hingga 12885. Hal tersebut karena kondisi lintasan mobility yang berbeda yang cenderung tidak stabil.
3. Hasil berbeda pada *average end to end delay* untuk skenario *grid* dan skenario riil. Pada skenario grid nilai *end to end delay* lebih rendah hingga 16.32 % lebih baik dari dibandingkan skenario riil.
4. Hasil simulasi menunjukkan skalabilitas dari performa AntHocNet ketika dibandingkan dengan AODV. Performa akan baik dengan semakin banyaknya node. Penggunaan routing AntHocNet dapat meningkatkan kapasitas performa komunikasi anatar link yang terpercaya terutama untuk *Vehicle Adhoc Network*. AntHocNet memiliki performa yang lebih stabil pada *node* yang banyak.

6.2. Saran

Saran yang dapat diberikan dari hasil uji coba dan evaluasi adalah sebagai berikut :

1. Masih diperlukan perbaikan pada pengujian AntHocnet sehingga performa menjadi semakin terlihat keunggulannya dibandingkan dengan AODV.
2. Studi lebih lanjut mengenai pengaruh jumlah paket terhadap kinerja protokol.
3. Modifikasi AntHocNet akan sangat membantu untuk memperbaiki kualitas routing protokol berbasis *Ant Colony Optimization.*

DAFTAR PUSTAKA

[1] R. Bastian, IMPLEMENTASI SOURCE ROUTE PADA PROTOKOL GPSR DENGAN BANTAUAN INTERSECTION NODE UNTUK MENINGKATKAN RELIABILITAS PENGIRIMAN DATA DI VANETs. .

[2] R. ARIEF, "IMPLEMENTASI ROUTING PROTOKOL AODV DENGAN PREDIKSI PERGERAKAN KENDARAAN DALAM VANETS."

[3] Gianni Di Caro, Frederick Ducatelle, dan Luca Maria Gambardella, "AntHocNet: an adaptive nature-inspired algorithm for routing in mobile ad hoc networks."

[4] Gianni Di Caro, Frederick Ducatelle, dan Luca Maria Gambardella, "AntHocNet: an Ant-Based Hybrid Routing Algorithm for Mobile Ad Hoc Networks."

[5] Frederick Ducatelle, Gianni A. Di Caro, dan Luca M. Gambardella, "Analysis of the different components of the AntHocNet routing algorithm."

[6] P. Y.lakshmi dan R. P.Chenna, "Analysis ofAntHocNet and AODV Performance using NS-2."

[7] V. Kumar, A. S. Baghel, dan P. Mishra, "Performance evaluation of DSDV, AODV and LSGR protocol in ad-hoc networks," dalam 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT), 2016, hlm. 4261–4266.

[8] "NETGENERATE," http://sumo.drl.de/wiki/NETGENERATE. .

[9] "NETCONVERT," http://sumo.drl.de/wiki/NETCONVERT. .

[10] "OpenStreetMap," https://www.openstreetmap.org/. .

[Halaman ini sengaja dikosongkan]

LAMPIRAN A 1

1. Kode Implementasi Average End-to-End Delay

```
BEGIN {
        seqno = -1; count = 0;
}
{
        if ($4=="AGT" && $1=="s" && seqno<$6){
                seqno = $6;
        }
        if ($4=="AGT" && $1=="s"){
                start_time[$6]=$2;
        }else if (($7=="cbr") && ($1=="r")){
                end_time[$6]=$2;
        }else if ($1=="D" && $7=="cbr"){
                end_time[$6]=-1;
        }
}
END {
        for (i=0;i<=seqno;i++)
        {
                if(end_time[i]>0){
                 delay[i]=end_time[i]-start_time[i];count++;
                }
                else{
                 delay[i];
                }
        }
        for (i=0;i<=seqno;i++){
                if(delay[i]>0){
                 n_to_n_delay=n_to_n_delay + delay[i]
                }
        }
        n_to_n_delay = n_to_n_delay/count;
        printf("Delay =%.2f\n", n_to_n_delay*1000);
}
```

2. Kode Implementasi Routing Overhead

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

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

$0 ~/^r.* AGT/ {
```

```
        recvLine ++;
}

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

END {
        printf ("Overhead : %d\n", (sendLine + recvLine +
errLine));
}
```

3. Kode Implementasi Packet Delivery Ratio

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

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

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

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

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

4. Kode Implementasi TCL NS-2

```
# Define options
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)      350 ;# number of mobilenodes
set val(rp)       AntHocNet ;# routing protocol
set val(x)        2600    ;# X dimension of topography
set val(y)        2600    ;# Y dimension of topography
set val(stop)    900     ;# time of simulation end
set opt(energymodel)    EnergyModel             ;# Energy
mode on
set opt(initialenergy)     10000                ;#
Initial energy in Joules

set val(cbrsize) 600;
set val(cbrrate) 2KB;
set val(cbrinterval) 1;


set ns_    [new Simulator]
set tracefd       [open simple-ant.tr w]

set namtrace      [open simple-wrls.nam w]

$ns_ trace-all $tracefd
$ns_ namtrace-all-wireless $namtrace $val(x) $val(y)

set topo       [new Topography]
$topo load_flatgrid $val(x) $val(y)

set god_ [create-god $val(nn)]

# configure the nodes
$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 OFF \
                 -energyModel $opt(energymodel) \
                 -idlePower 1.0 \
                 -rxPower 0.01 \
                 -txPower 0.01 \
                 -sleepPower 0.000001 \
                 -transitionPower 0.2 \
                 -transitionTime 0.005 \
```

```
                -initialEnergy $opt(initialenergy) \
                -movementTrace OFF

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

source /home/dewi/Downloads/R100/mobility.tcl

$ns_ at 15.0 "$topo load_flatgrid $val(x) $val(y)"

set udp [new Agent/UDP]
$udp set class_ 2
$ns_ attach-agent $node_(3) $udp

set null [new Agent/Null]
$ns_ attach-agent $node_(22) $null
$ns_ connect $udp $null

set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set type_ CBR
$cbr set packet_size_ $val(cbrsize)
$cbr set rate_ $val(cbrrate)
$cbr set interval_ $val(cbrinterval)
$ns_ at 10.0 "$cbr start"
$ns_ at 900.01 "$cbr stop"

# Define node initial position in nam
for {set i 0} {$i < $val(nn)} { incr i } {
# 30 defines the node size for nam
$ns_ initial_node_pos $node_($i) 20
}

# Telling nodes when the simulation ends
for {set i 0} {$i < $val(nn) } { incr i } {
    $ns_ at $val(stop) "$node_($i) reset";
}

# ending nam and the simulation
$ns_ at $val(stop) "$ns_ nam-end-wireless $val(stop)"
$ns_ at $val(stop) "stop"
$ns_ at 900.01 "puts \"end simulation\" ; $ns_ halt"
proc stop {} {
    global ns_ tracefd namtrace
    $ns_ flush-trace
    close $tracefd
    close $namtrace
}

$ns  run
```

5. Kode Implementasi Request Table pada AntHocNet

```
friend class AntHocNet;
private:
        PheromoneTable* pheromone_rt_;
        MapAntPQueue mapQueue;
        AntHistoryList history; // history of ant packet to
        send by node

        nsaddr_t addr_; // address of the agent
        u_int8_t ant_seq_num_; // sequence number for ant
        packets
        PheromoneTable* pheromoneTable() const;

        double evaporatePheromone(double phValue);
        double incressPheromone(double phValue, double
        phUpdate);

public:
        AntNest(nsaddr_t addr)
{
        pheromone_rt_ = new PheromoneTable();
        ant_seq_num_ = 0;
        addr_ = addr;
}
        ~AntNest()
{
        delete pheromone_rt_;
}
```

6. Kode Implementasi Perhitungan Pheromone

```
nsaddr_t AntBackPacket::findNextHop(nsaddr_t sddr)
{
        AntTimeEntry* entry = visitedNodes[sizeNodes_-1];
        //fprintf(stdout, "AntBackPacket:: entry* %d\n",
entry);
        prevHop = entry->node();
        hops++;
        // time to back ant send "-" time to ant forward
pass to node
        double time = entry->time();
        prevSINR += time;

        double simpleSINR = prevSINR / 1000;

        // calculate pheromone to used to destination
        int hTime = AntHocNetUtils::HOP_TIME;
```

```
        pheromone = pow( ((hops*hTime + simpleSINR) / 2)
, -1);

        // add visited nodes history
        if (visitedNodesHist == NULL) {
                visitedNodesHist                        =
(AntTimeEntry**)malloc(sizeof(AntTimeEntry*) * 100);
                sizeNodesHist_ = 0;
        }

        visitedNodesHist[sizeNodesHist_++]      =      new
AntTimeEntry(entry->node(), entry->time());
        // get next neighbor
        sizeNodes_--;
        entry = visitedNodes[sizeNodes_-1];

#ifdef DEBUG
        fprintf(stdout,   "AntBackPacket::findNextHop   -
seqnum: %d:%d >> sddr (%d) == prev (%d) >> time (%f) --
pheromone (%f) -- vizited size %d next %d \n", this-
>getSorceAddress(), this->getSeqNum(),
                        sddr,      prevHop,      prevSINR,
pheromone, sizeNodes_, entry->node());

        for (int i=0; i< sizeNodes_; i++) {
                AntTimeEntry* entry = visitedNodes[i];
                fprintf(stdout, " ---- entry %d => time
%d \n", entry->node(), entry->time());
        }
#endif //DEBUG

        return entry->node();
}
```

7. Kode Implementasi Update Pheromone

```
// update virtual pheromone
nsaddr_t neighbor = ah->getSorceAddress();
AntNode** dests = ah->getDestinations();
int size = ah->sizeDests();

for (int i=0; i< size; i++) {
        AntNode* node = dests[i];
        nsaddr_t destination = node->node();
        double      phValue      =      pheromoneTable()-
>getPheromoneVirtual(destination, neighbor);
        double phUpdate = node->phValue();
        double pIncress = this->incressPheromone(phValue,
phUpdate);
```

```
        pheromoneTable()-
>setPheromoneVirtual(destination, neighbor, pIncress);}
```

[Halaman ini sengaja dikosongkan]

BIODATA PENULIS

Dewi Kartika Prasetyawati, lahir pada tanggal 31 Maret 1995 di Klaten, Jawa Tengah. Penulis menempuh pendidikan mulai dari SD Negeri 02 Kadirejo (2001-2007), SMP Muhammadiyah 1 Klaten (2007-2010), SMA Negeri 1 Karanganom (2010-2013). Dan saat ini penulis sedang menempuh pendidikan perguruan tinggi di Institut Teknologi Sepuluh November Sepuluh November Surabaya jurusan Informatika Fakultas Teknologi Informasi angkatan tahun 2013.

Penulis memiliki bidang minat Arsitektur dan Jaringan Komputer (AJK) dengan fokus Keamanan Jaringan dan Routing. Komunikasi dengan penulis dengan senang hati dilayani dan dapat melalui email langsung ke: **dewi13@mhs.if.its.ac.id.**