TUGAS AKHIR – IF184802

ANALISIS KINERJA IEEE WLAN 802.11A DAN 802.11P PADA OPTIMIZED LINK STATE ROUTING PROTOCOL (OLSR) DI LINGKUNGAN MOBILE AD HOC NETWORK (MANET)

Irsyad Iswanda Putra
NRP 05111340000185

Dosen Pembimbing
Dr.Eng. RADITYO ANGGORO, S.Kom., M.Sc.

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

TUGAS AKHIR – IF184802

ANALISIS KINERJA IEEE WLAN 802.11A DAN 802.11P PADA OPTIMIZED LINK STATE ROUTING PROTOCOL (OLSR) DI LINGKUNGAN MOBILE AD HOC NETWORK (MANET)

Irsyad Iswanda Putra
NRP 05111340000185

Dosen Pembimbing
Dr.Eng. RADITYO ANGGORO, S.Kom., M.Sc.

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

[Halaman ini sengaja dikosongkan]

UNDERGRADUATE THESIS – IF184802

ANALISIS KINERJA IEEE WLAN 802.11A DAN 802.11P PADA OPTIMIZED LINK STATE ROUTING PROTOCOL (OLSR) DI LINGKUNGAN MOBILE AD HOC NETWORK (MANET)

Irsyad Iswanda Putra
NRP 05111340000185

Advisor
Dr.Eng. RADITYO ANGGORO, S.Kom., M.Sc.

DEPARTMENT OF INFORMATICS
Faculty of Information Technology and Communication
Institut Teknologi Sepuluh Nopember
Surabaya 2018

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

ANALISIS KINERJA IEEE WLAN 802.11A DAN 802.11P PADA OPTIMIZED LINK STATE ROUTING PROTOCOL (OLSR) DI LINGKUNGAN MOBILE AD HOC NETWORK (MANET)

TUGAS AKHIR

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

Oleh :
Irsyad Iswanda Putra
NRP : 05111340000185

Disetujui oleh Dosen Pembimbing Tugas Akhir :

Dr.Eng. RADITYO ANGGORO, S.Kom., M.Sc.
NIP: 198410162008121002 ..... (pembimbing 1)

SURABAYA
DESEMBER 2018

[Halaman ini sengaja dikosongkan]

ANALISIS KINERJA IEEE WLAN 802.11A DAN 802.11P PADA OPTIMIZED LINK STATE ROUTING PROTOCOL (OLSR) DI LINGKUNGAN MOBILE AD HOC NETWORK (MANET)

Nama Mahasiswa : Irsyad Iswanda Putra
NRP : 05111340000185
Jurusan : Informatika FTIK-ITS
Dosen Pembimbing 1 : Dr.Eng. Radityo Anggoro, S.Kom., M.Sc.
Dosen Pembimbing 2 : -

Abstrak

Di era modern saat ini, teknologi nirkabel memungkinkan perangkat bergerak dapat berkomunikasi secara langsung dengan perangkat bergerak lainnya tanpa adanya bantuan dari jaringan infrastruktur. Jaringan ini disebut sebagai Mobile Ad Hoc network (MANET). Dalam MANET, setiap node bergerak secara bebas, sehingga jaringan dapat mengalami perubahan topologi dengan cepat. Karena node dalam MANET memiliki jarak transmisi yang terbatas, beberapa node tidak bisa berkomunikasi secara langsung dengan node lainnya. Implementasi MANET di dunia nyata masih sulit untuk dilakukan, sehingga banyak penelitian dilakukan dengan membuat simulasi MANET menggunakan network simulator.

Terdapat beberapa metode routing pada jaringan MANET, salah satunya adalah routing protocol Optimized Link State Routing Protocol (OLSR). Uji coba dilakukan dengan membuat pola traffic koneksi dan pola pergerakan node yang kemudiaan disimulasikan dengan menggunakan script tcl OLSR.

Proses simulasi dijalankan menggunakan konfigurasi tambahan dari IEEE WLAN 802.11a dan IEEE WLAN 802.11p. Proses tersebut akan menghasilkan file output berupa trace file (.tr). Trace file hasil dari simulasi akan dianalisis untuk menghitung Packet Delivery Ratio (PDR), Routing Overhead

(RO), dan End-to-End Delay (E2E) dari setiap ip yang di bandingkan.
***Kata kunci**: IEEE WLAN 802.11a, IEEE WLAN 802.11p, MANET, NS-2, OLSR.*

ANALYSIS IEEE WLAN 802.11A AND 802.11P WITH OPTIMIZED LINK STATE ROUTING PROTOCOL (OLSR) ON MOBILE AD HOC NETWORK (MANET)

Student Name	: Irsyad Iswanda Putra
Student ID	: 05111340000185
Major	: Informatics FTIK-ITS
Advisor 1	: Dr.Eng. Radityo Anggoro, S.Kom., M.Sc.
Advisor 2	: -

Abstract

In today's modern era, wireless technology allows mobile devices to communicate directly with other mobile devices without any help of infrastructure networks. This network is called Mobile Ad Hoc network (MANET). In MANET, each node moves freely, so the topology network can changes quickly. Because of nodes in MANET have a limited transmission distance, some nodes cannot communicate directly with other nodes. MANET implementation in the real world is still difficult to do, so much research is done by making MANET simulations using a simulator network.

There are several routing methods in the MANET network, one of which is the Optimized Link State Routing Protocol (OLSR) routing protocol. The trial is done by making a pattern of connection traffic and node movement patterns which are then simulated using the OLSR tcl script.

The simulation process is run using additional configurations from IEEE WLAN 802.11a and IEEE WLAN 802.11p. The process will produce an output file in the form of a trace file (.tr). The trace file results from the simulation will be analyzed to calculate the Packet Delivery Ratio (PDR), Overhead Routing (RO), and End-to-End Delay (E2E) from each ip compared.

Keyword: *IEEE WLAN 802.11a, IEEE WLAN 802.11p, MANET, NS-2, OLSR.*

KATA PENGANTAR

Alhamdulillahirabbil'alamin, segala puji bagi Allah Subhanahu Wa Ta'ala, yang telah melimpahkan rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang Berjudul **Analisis Kinerja IEEE WLAN 802.11a dan 802.11p pada *Optmized Link State Routing Protocol* (OLSR) di lingkungan *Mobile Ad-hoc Network* (MANET).**

Tugas akhir ini dilakukan untuk memenuhi salah satu syarat memperoleh gelar Sarjana Komputer di Departemen Informatika Fakultas Teknologi Informasi dan Komunikasi Institut Teknologi Sepuluh Nopember.

Penulis mengucapkan terima kasih kepada semua pihak yang telah memberikan dukungan baik secara langsung maupun tidak langsung selama proses pengerjaan tugas akhir ini hingga selesai, antara lain:

1. Allah SWT atas limpahan nikmat dan anugerah-Nya yang telah di berikan kepada penulis.
2. Keluarga besar penulis yang senantiasa memberikan dukungan kepada penulis.
3. Bapak Dr. Eng Radityo Anggoro S.Kom., M,Sc. selaku dosen pembimbing pertama yang telah bersedia meluangkan waktu untuk membimbing dan memotivasi penulis serta memberi arahan dalam mengerjakan tugas akhir.
4. Bapak/Ibu dosen, staf dan karyawan Jurusan Teknik Informatika ITS yang telah banyak memberikan dukungan, ilmu dan bimbingan yang tak ternilai harganya bagi penulis.
5. Agung Teguh Setyadi, Rezky Budi Prasetyo, Fauzan Adim, Ahmad Fauzi Triyanto, dan Ahmad Zaki yang selalu membantu penulis dalam mengerjakan tugas akhir ini.

6. Julio Anthony Leonard, Nugroho Wicaksono, Muhammad Fahmi Purnomo, Muhammad Luthfie La Roeha, Daniel Bintar, Daniel Fablius, John Stephanus Peter yang memberikan semangat kepada penulis.
7. Teman Teman angkatan 2013 dan angkatan 2014 yang sudah membantu penulis selama di kampus.
8. Serta pihak-pihak lain yang tidak dapat disebutkan disini yang telah banyak membantu penulis dalam penyusunan Tugas Akhir ini.

Penulis telah berusaha sebaik-baiknya dalam menyusun Tugas Akhir ini, namun penulis mohon maaf apabila terdapat kekurangan, kesalahan maupun kelalaian yang telah penulis lakukan. Kritik dan saran yang membangun dapat disampaikan sebagai bahan perbaikan selanjutnya.

Surabaya, 4 Januari 2018
Penulis

Irsyad Iswanda Putra

DAFTAR ISI

DAFTAR GAMBAR

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

DAFTAR KODE SUMBER

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

Pada bab ini akan 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 zaman membuat teknologi informasi berkembang dengan sangat pesat. Di era modern saat ini, teknologi nirkabel memungkinkan perangkat bergerak seperti *handphone*, *tablet* dapat berkomunikasi secara langsung dengan perangkat bergerak lainnya dengan tanpa adanya bantuan dari jaringan infrastruktur. Jaringan ini disebut sebagai *Mobile Ad Hoc network (MANET).* Teknologi jaringan nirkabel (*wireless*) dapat membuat beberapa perangkat *mobile* tersebut yang berada di dalam suatu area dimana fasilitas nirkabel tersebut bisa saling terkoneksi dan saling menjangkau akan sangat mungkin membentuk sebuah jaringan yang bersifat sementara dan jaringan semacam ini disebut sebagai *Mobile Ad Hoc Network* (MANET). Dalam teknologi MANET tidak bisa dilepaskan dengan proses pengiriman paket data. Proses pencarian rute untuk mengirimkan data dari sumber ke tujuan disebut dengan routing. Dalam proses *routing*, *data* yang berasal dari *node* sumber dikirimkan ke *node*s lain hingga mencapai *node* tujuan. Jalur *routing* mengandung beberapa *hop* dan setiap *node* berfungsi sebagai *router* untuk menentukan ke arah mana tujuan atau rute yang akan mereka pilih. Dalam menentukan setiap jalur *routing* pada MANET terdapat tiga jenis *routing protocol*. Salah satunya adalah *routing protocol* proactive. Contoh dari *routing protocol* ini adalah OLSR. OLSR bersifat menyebarkan seluruh informasi rute tabel ke setiap *node* yang ada. Apabila terjadi perubahan maka informasi perubahannya akan disebarkan ke seluruh *node*.

Implementasi pada lingkungan MANET dapat dilakukan dengan menggunakan simulasi sehingga penelitian ini dapat dilakukan untuk mempelajari sistem dengan baik. Simulasi dilakukan dengan menggunakan *Network Simulator* 2 (NS-2). NS-2 yang digunakan pada penelitian ini adalah versi 2.35. Penelitian yang akan dilakukan adalah menganalisis kinerja dari IEEE WLAN 802.11a dan IEEE WLAN 802.11p yang menggunakan *routing protocol* OLSR berdasarkan *End-to-End* (E2E), *Packet Ratio Delivery* (PDR), *Routing Overhead* (RO).

IEEE 802.11a adalah sebuah teknologi jaringan nirkabel yang merupakan pengembangan lebih lanjut dari standar IEEE 802.11 yang pertama. Standar 802.11a menggunakan protokol inti yang sama dengan standar 802.11, beroperasi pada frekuensi 5 GHz, dan menggunakan OFDM (*Orthogonal Frequency - Division Multiplexing*) dengan kecepatan *transfer* data hingga 54Mbit/s. Kecepatan *transfer* data dikurangi menjadi 48, 36, 24, 18, 12, 9 dan 6 Mbit/s jika diperlukan. 802.11a memiliki 12/13 *non-overlapping channel*. Sedangkan, IEEE 802.11p adalah pengembangan dari IEEE 802.11a dengan menambahkan *physical layer* dan MAC layer yang dapat meningkatkan sistem operasi dan aplikasi keselamatan dengan memberikan tingkat *latency* rendah. IEEE 802.11p sendiri beroperasi pada frekuensi 5.9 GHz dengan menggunakan sistem multiplexing OFDM (Orthogonal Frequency Division Multiplexing). Yang berarti IEEE 802.11p dan IEEE 802.11a masih menggunakan metode pentransmisian yang sama. IEEE 802.11p dapat mencapai kecepatan pentransmisian data antara 6-27Mbps. IEEE 802.11p terdiri dari 7 channel pada frekuenesi 10 MHz dan 6 *service channel* pada frekuensi 5.9 GHz.

1.2. Rumusan Permasalahan

Rumusan masalah yang diangkat dalam Tugas Akhir ini adalah bagaimana hasil analisis perbandingan kinerja IEEE WLAN 802.11a dan 802.11p pada *routing protocol* OLSR di lingkungan MANET berdasarkan parameter *End-to-End* (E2E), *Packet Ratio Delivery* (PDR), *Routing Overhead* (RO).

1.3. Batasan Permasalahan

Beberapa batasan masalah yang terdapat dalam tugas akhir ini dapat dipaparkan sebagai berikut:

1. Simulasi pengujian *routing protocol* menggunakan *Network Simulator 2* versi 2.35.
2. *Routing protocol* menggunakan OLSR.
3. Ujicoba dilakukan pada topologi manet.
4. MAC *protocol* yang akan dianalisis adalah IEEE WLAN 802.11a dan IEEE WLAN 802.11p.
5. Analisis Kinerja Didasarkan pada *End-to-End* (E2E), *Packet Ratio Delivery* (PDR), *Routing Overhead* (RO).

1.4. Tujuan

Tujuan dari pembuatan Tugas Akhir ini adalah untuk memberikan hasil perbandingan kinerja IEEE WLAN 802.11a dengan IEEE WLAN 802.11p pada *routing protocol* OLSR di lingkungan MANET dengan menggunakan aplikasi *Network Simulator* 2 (NS-2) berdasarkan *End-to-End* (E2E), *Packet Ratio Delivery* (PDR), *Routing Overhead* (RO).

1.5. Manfaat

Dengan dibuatnya Tugas Akhir ini akan memberikan sebuah manfaat dalam menentukan mana yang lebih optimal antara IEEE WLAN 802.11a atau IEEE WLAN 802.11p.

1.6. Metodologi

Tugas Akhir ini menggunakan beberapa tahapan dalam proses pengerjaannya. Metodologi yang dilakukan dalam pengerjaan Tugas Akhir ini terdiri atas beberapa tahapan yang dipaparkan sebagai berikut.

1. Penyusunan proposal Tugas Akhir

Pada tahap ini dilakukan penyusunan proposal Tugas Akhir yang berisi tentang deskripsi umum rancangan Tugas Akhir yang akan dibuat. Penyusunan ini terdiri dari menentukan judul Tugas Akhir, latar belakang, rumusan masalah, batasan masalah, tujuan Tugas Akhir, manfaat Tugas Akhir, tinjauan pustaka, ringkasan Tugas Akhir, metodologi, serta rencana jadwal kegiatan pengerjaan Tugas Akhir.

2. Studi literatur

Tahap studi literatur merupakan tahap pembelajaran dan pengumpulan informasi yang digunakan untuk mengimplementasikan Tugas Akhir. Tahap ini diawali dengan pengumpulan literatur, diskusi, eksplorasi teknologi dan pustaka, serta pemahaman dasar teori yang digunakan pada topik Tugas Akhir. Literatur-literatur yang dimaksud adalah MANET, NS-2, OLSR, IEEE WLAN 802.11a, dan IEEE 802.11p.

3. Implementasi

Tahap ini awalnya dilakukan dengan instalasi program ns-2 versi 2.35. Lalu men-*download patch* khusus untuk OLSR dan melakukan instalasi di dalam *folder* ns-2.35. Setelah itu menetapkan *File traffic Connection Pattern* dengan membuat *file* cbrgen.tcl dan men-*generate random movement* dari *node* di jaringan *wireless.*

4. Pengujian dan Evaluasi

Pada tahap ini dilakukan uji coba terhadap sistem yang sudah dibuat. Pengujian dan evaluasi akan diketahui berdasarkan nilai dari *Packet Delivery Ratio* (PDR), *Routing Overhead* (RO), dan *End-to-End Delay*.

5. Penyusunan Buku Tugas Akhir

Pada tahap ini dilakukan pendokumentasian dan pelaporan dari seluruh konsep, dasar teori, implementasi, proses yang telah dilakukan, dan hasil-hasil yang telah didapatkan selama pengerjaan Tugas Akhir.

1.7. Sistematika Penulisan

Berikut adalah sistematika penulisan buku tugas akhir ini:

Bab I Pendahuluan

Bab ini berisi latar belakang masalah, tujuan dan manfaat pembuatan Tugas Akhir, permasalahan, batasan masalah, metodologi yang digunakan, dan sistematika penyusunan Tugas Akhir.

Bab II Tinjauan Pustaka

Bab ini membahas beberapa teori penunjang yang berhubungan dengan pokok pembahasan dan mendasari pembuatan Tugas Akhir ini.

Bab III Analisis dan Perancangan Sistem

Bab ini membahas mengenai perancangan metode yang nantinya akan diimplementasikan dan dilakukan pengujian dari aplikasi yang dibangun.

Bab IV Implementasi

Bab ini berisi implementasi dari perancangan sistem yang sudah dilakukan pada tahap perancangan. Penjelasan berupa skenario *node node* pada jaringan *wireless*, konfigurasi sistem dan skrip analisis yang digunakan untuk menguji performa *routing protocol*

Bab V Pengujian dan Evaluasi

Bab ini membahas pengujian dari sistem dan performa dalam skenario mobilitas *ad hoc* yang dibuat dalam *network simulator*.

Bab VI Kesimpulan dan Saran

Bab ini berisi kesimpulan dari hasil pengujian yang dilakukan. Bab ini membahas saran-saran untuk pengembangan sistem lebih lanjut.

Daftar Pustaka

Merupakan daftar referensi yang digunakan untuk mengembangkan Tugas Akhir.

Lampiran

Merupakan bab tambahan yang berisi daftar istilah yang penting pada aplikasi ini.

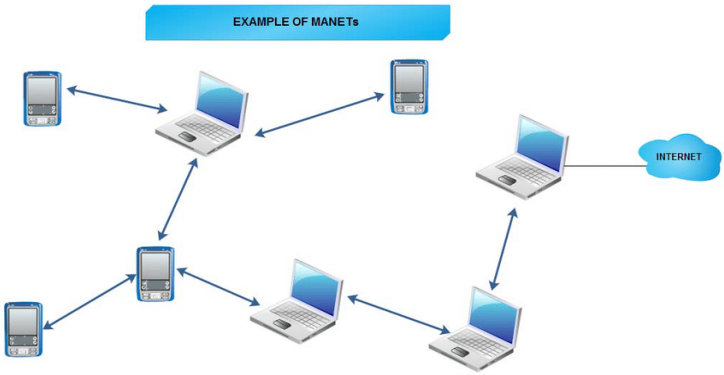
BAB II
TINJAUAN PUSTAKA

Bab ini menjelaskan tentang tinjauan pustaka yang menjadi dasar pembuatan Tugas Akhir. Beberapa teori, pustaka, dan teknologi yang mendasari pengerjaan Tugas Akhir ini. Serta memberi gambaran terhadap alat, *routing protocol* serta definisi yang digunakan dalam pembuatan Tugas Akhir.

2.1. *Mobile Ad Hoc Network* (MANET)

Mobile Ad Hoc Network (MANET) adalah sekumpulan *mobile node* yang mana proses pertukaran informasinya melalui media transmisi nirkabel/*wireless* bersifat dinamis. Setiap *mobile host* dalam MANET bebas untuk bergerak ke segala arah. Di dalam jaringan MANET terdapat dua *node* (*mobile host*) atau lebih yang dapat berkomunikasi dengan *node* lainnya namun masih berada dalam jangkauan *node* tersebut. Selain itu, *node* juga dapat berfungsi sebagai penghubung antara *node* yang satu dengan *node* yang lainnya.

Jaringan *ad hoc* dapat bekerja dengan infrastruktur berupa *wireless* dengan cara berkomunikasi *secara mobile network*, serta untuk proses routingnya menggunakan *multihop* informasi jadi setiap informasi akan dikirimkan dan disimpan terlebih dahulu dan diteruskan ke *node* tujuan melalui perantara. Namun dari sisi keamanan tentunya sangat terbatas jika dibandingkan dengan *network* yang menggunaan kabel. Karakteristik dari *ad hoc* ini pun selalu berpindah-pindah dikarenakan *node* selalu bergerak tanpa diprediksi.



Gambar 2.1 Contoh MANET

Routing protocol yang tersedia dalam MANET terbagi menjadi tiga kategori yaitu:

• *Proactive routing*

Jenis *routing* ini bersifat menyebarkan seluruh informasi rute tabel ke setiap *node* yang ada. Apabila terjadi perubahan maka informasi perubahannya akan disebarkan ke seluruh *node*. Contoh dari tipe *routing* ini adalah *Optimized Link State Routing* (OLSR).

• *Reactive routing*

Cara kerja dari *routing* ini adalah apabila *node* sumber mau mengirimkan paket maka *node* sumber akan mengirimkan *route request*. Setelah sampai di *node* tujuan maka akan di kirim *node reply* ke *node* sumber. Contoh dari *routing* ini adalah *Dynamic Source Routing* (DSR), *ad-hoc on Demand Vector* (AODV).

• *Hybrid Routing*

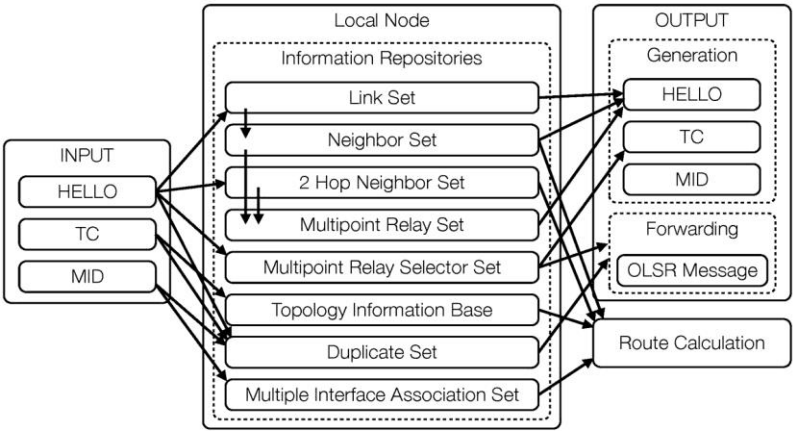
Jenis *routing* ini merupakan gabungan dari *Proactive routing* dan *Reactive routing*. Contoh dari *routing* ini adalah *Zone Routing Protocol* (ZRP).

2.2. *Optimized Link State Routing Protocol (OLSR)*

OLSR merupakan salah satu protokol di jaringan MANET (*Mobile Ad-hoc Network*) yang bersifat *proactive*. OLSR memiliki inisiatif untuk menemukan rute dan bertujuan untuk menjaga konsistensi dan informasi perutean setiap *node* dalam jaringan MANET. *Protocol* OLSR ini mewarisi sifat kestabilan dari *link state algorithm.* Berdasarkan sifat proaktifnya, *protocol* ini dapat menyediakan rute dengan segera apabila dibutuhkan. Dalam sebuah *link state protocol* yang murni, setiap *node* tetangga dideklarasikan dan dibanjiri dengan paket informasi yang akan memenuhi seluruh jaringan.

Langkah pertama dari optimasi tersebut adalah mengurangi ukuran dari paket *control*. Dibandingkan dengan membanjiri paket *control* tersebut pada setiap jalur, OLSR lebih memilih sejumlah jalur dengan *node* tetangga yang disebut dengan *multipoint relay selector*. Langkah kedua, OLSR meminimalisir pembanjiran paket *control* pada jaringan dengan menggunakan MPR untuk menghantarkan paket-paket tersebut. Teknik ini akan mengurangi secara sognifikan jumlah dari transmisi ulang yang akan membanjiri jaringan dengan prosedur *broadcast*. *Protocol* ini dirancang untuk bekerja pada kondisi yang selalu bergerak serta tidak memerlukan adanya pengaturan secara terpusat. Setiap *node* mengirimkan paket kontrolnya masing-masing secara periodik sehingga dapat mentoleransi terjadinya *loss* dari beberapa paket pada saat-saat tertentu akibat gangguan transmisi lainnya. Setiap paket *control* yang dikirimkan akan diberikan *sequence number* (nomor urut) yang dapat menandakan tingkat baru tidak paket tersebut.

OLSR menggunakan *multihop routing* dimana setiap *node* menggunakan informasi *routing* tersebut dalam mengantarkan sebuah paket informasi. Sehingga walaupun sebuah *node* bergerak ataupun berpindah tempat maka pesan yang dikirimkan padanya akan tetap dapat diterima.



Gambar 2.2 Ilustrasi skenario OLSR

2.2.1. Tahapan Kerja OLSR

Secara umum langkah-langkah kerja dalam OLSR dapat diurutkan sebagai berikut:

a. *Link Sensing* (pendeteksian hubungan).

Link sensing dilakukan dengan mengirimkan pesan HELLO secara periodik dan berkesinambungan. Hasil dari *link sensing* adalah *local link set* yang menyimpan informasi hubugan antara *interface* yang ada pada *node* tersebut dengan *nodes* tetangganya.

b. *Neighbour detection* (pendeteksian *node* tetangga)

Node mengirim pesan HELLO akan menerima informasi alamat-alamat dari *node* tetangganya beserta *link* statusnya.

c. MPR *selection*

MPR *selection* (memilih MPR) melalui pesan HELLO *node* utama akan menentukan sejumlah *node* tetangga untuk dipilih sebagai *multipoint relay* (MPR) yang bertugas meneruskan paket-paket *control* ke dalam jaringan.

d. Pengiriman TC (*Topology Control*) *messages*

TC *Messages* dikirimkan untuk memberikan informasi *routing* kepada setiap *node* yang ada pada jaringan yang akan digunakan untuk penentuan jalur.

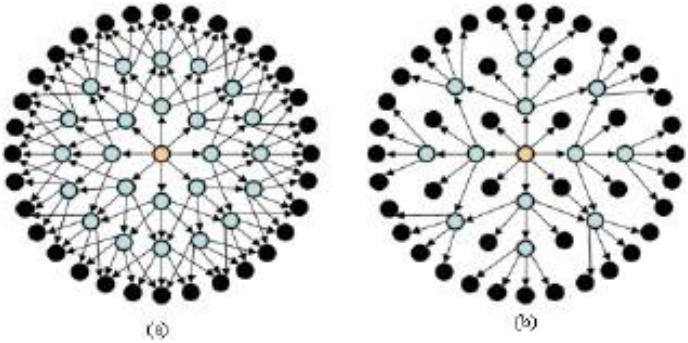
e. *Route calculation* (perhitungan jalur)

Berdasarkan informasi rute yang didapat dari paket-paket *control* seperti HELLO dan TC maka setiap *node* akan memiliki *routing table* yang berisi informasi rute yang dapat dilalui untuk dapat mengirimkan data ke *nodes* lainnya yang ada pada jaringan.

2.2.1. Pemilihan MPR

Tujuan dari *Multipoint Relay* (MPR) adalah meminimalisir penggunaan *overhead* dari pesan *broadcast* pada jaringan dengan cara mengurangi retransmisi (pentransmisian ulang) pada daerah yang sama. Setiap *node* pada jaringan akan memilih sejumlah *hop* tetangga 1-*hop* nya yang bersifat simetris yang akan melakukan transmisi ulang pesan-pesannya. Sejumlah *node* tetangga itulah yang disebut MPR. Setiap tetangga yang tidak terpilih sebagai *node* MPR akan tetap menerima pesan tersebut dan memproses namun tidak akan meneruskan atau mengirimkan kembali pesan-pesan tersebut. Pemilihan *node* untuk dijadikan MPR selain harus bersifat simetris juga harus dapat menjangkau sejumlah *node* tetangga 2-*hop*. Makin sedikit jumlah MPR maka penggunaan *control traffic overhead* yang digunakan dalam *routing protocol* makin sedikit. Perbandingan

kinerja pengiriman paket untuk OLSR dan *link state protocol* pada umumnya.



Gambar 2.3 (a) *Flooding* biasa (b) *flooding* menggunakan MPR

Setiap *node* yang dipilihnya sebagai MPR akan menyimpan informasi tentang *nodes* tetangga yang telah dipilihnya sebagai MPR "MPR sef" yang berisi alamat-alamat *node* MPR tersebut. Selain itu setiap *node* juga akan menyimpan informasi tentang siapa-siapa saja *node* yang menjadikannya sebagai MPR. Informasi tersebut disimpan dalam "MPR *selector sef"* MPR *set* harus dihitung oleh *node*, melalui *node* tetangga pada MPR *set* sehingga *node* dapat menjangkau semua *node* simetris tetangga 2-*hop*. MPR *set* akan dihitung ulang jika terjadi perubahan pada hubungan simetris *node* tetangga 1-*hop* atau hubungan simetris *node* tetangga 2-*hop*.MPR akan dihitung pada setiap *interface*.

Setiap *node* melakukan *broadcast* secara berkala ke tetangga 1-*hop* dengan menggunakan pesan *HELLO*. Berdasarkan *broadcast* ini, setiap *node* memilih *subset node* tetangga 1-*hop* yang terkecil yang bisa menjangkau semua *node* tetangga 2-*hop*. *Subset node* yang terpilih ini lalu dijadikan *node* MPR. Hanya

MPR yang dapat meneruskan pesan TC. Pesan TC digunakan untuk perhitungan *routing table*.

Langkah-langkah pemilihan MPR adalah sebagai berikut:

a. Setiap *node* melakukan *broadcast* ke *node* tetangga 1-*hop* secara berkala dengan menggunakan HELLO *messages*.

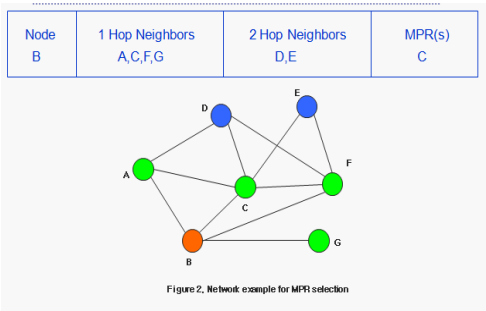
b. Berdasarkan *broadcast*, setiap *node* memilih *subset node* tetangga 1-*hop* yang terkecil yang bisa menjangkau semua *node* tetangga 2-*hop*. Subset *node* yang terpilih ini lalu dijadikan *node* MPR.

c. *Node* MPR melakukan *broadcast topology control* (TC) *messages* setiap interfal TC untuk menginformasikan *link state*.

•TC *Message* berisi tetangga 1-*hop* yang terpilih sebagai MPR.

•Hanya MPR yang dapat meneruskan TC *Message*.

•TC *Message* digunakan untuk perhitungan *routing table*.



Gambar 2.4 *Network* MPR *selection*

2.3. *IEEE WLAN 802.11a*

IEEE WLAN 802.11a adalah perubahan standar 802.11 untuk LAN nirkabel. 802.11 lebih dikenal dengan nama Wi-Fi.
IEEE WLAN 802.11a menggunakan frekuensi radio 5 GHz dan mampu mentranfser data hingga 54 Mbps. Standar ini menggunakan protokol dasar yang sama dengan standar 802.11 yang asli, tetapi menggunakan *ortogonal frequency-division multiplexing* (OFDM).

2.4. *IEEE WLAN 802.11p*

IEEE 802.11p atau biasa disebut dengan WAVE (*Wireless Access Vehicular Environment*) merupakan penyempurnaan standar IEEE 802.11 yang diperlukan untuk mendukung pengaplikasiaan ITS (*Intelligent Transportation Systems*).
IEEE WLAN 802.11p menggunakan frekuensi 5.9 GHz dan mampu mentranfser data hingga 27 Mbps.

2.5. *Network Simulator 2 (NS-2)*

Network Simulator (NS) adalah suatu interpreter yang berorientasi objek, dan *discrete event-driven* yang dikembangkan oleh *University of* California Berkeley dan USC ISI sebagai bagian dari projek *Virtual INternet Testbed* (VINT). NS yang banyak dikenal dengan NS-2 (versi 2) menjadi salah satu *tool* yang sangat berguna untuk menunjukkan simulasi jaringan melibatkan *Local Area Network* (LAN), *Wide Area Network* (WAN), tapi fungsi dari *tool* ini telah berkembang selama beberapa tahun belakangan untuk memasukkan jaringan nirkabel (*wireless*) dan juga jaringan *ad hoc*.

NS2 adalah alat simulasi jaringan *open source* yang banyak digunakan dalam mempelajari struktur dinamik dari jaringan komunikasi. Simulasi dari jaringan nirkabel dan protokol (seperti algoritma *routing*, TCP, dan UDP) dapat diselesaikan dengan baik dengan simulator ini. Karena kefleksibelannya, NS2 menjadi

popular dikalangan komunitas peneliti sejak awal kemunculannya pada tahun 1989.

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

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

2.6. *AWK*

AWK merupakan sebuah pemrograman seperti pada shell atau C yang memiliki karakteristik yaitu sebagai alat yang cocok untuk memanipulasi sebuah text yang dapat digunakan sebagai ekstraksi dari sebuah *dataset*. AWK ditulis menggunakan Bahasa pemrogramannya sendiri yaitu *awk programming language*.

AWK digunakan sebagai alat untuk membuat script dalam perhitungan *Packet Delivery Ratio* (PDR), *End-to-End Delay* (E2E), dan *Routing Overhead* (RO) dari hasil *trace* NS-2.

2.7. *Generator File Node Movement*

Carnegie Mellon University mengembangkan sebuah alat bernama 'setdest' yang digunakan untuk men-*generate random movement* dari *node* di jaringan *wireless*. *Node movement* dihasilkan dengan kecepatan maksimal yang spesifik serta penempatan setiap *node*nya yang acak ataupun yang telah ditentukan. Apabila *node* yang sudah ditentukan sampai pada lokasi tujuan maka *node* tersebut akan berpindah ke lokasi selanjutnya. Pergerakan *node* tersebut dapat dihentikan sementara.

Sebelum menjalankan program simulasi pengguna harus menjalankan program 'setdest'. Format *command line* 'setdest' ditunjukan pada Kode Sumber 2.1 dan keterangannya ditunjukan pada Tabel 2.1.

```
./setdest [-v version] [-n num_of_nodes] [-p pausetime] [-M maxspeed]
      [-t simtime] [-x maxx] [-y maxy] > [outdir/movement-file]
```

Kode Sumber 2.1 Format C*ommand Line* 'setdest'

Tabel 2.1 Penjelasan *Command Line* 'setdest'

Parameter	Keterangan
-v *version*	Menentukan versi dari 'Setdest' yang digunakan
-n *num*	Menentukan jumlah *node* yang dibuat pada skenario
-p *pausetime*	Menentukan durasi berhenti pada sebuah paket apabila sudah sampai di tujuan
-M *maxspeed*	Menentukan kecepatan maksimum dari pengiriman paket
-t *simtime*	Menentukan lama waktu jalannya simulasi

-x *max* x	Menentukan panjang maksimum dari area simulasi
-y *max* y	Menentukan lebar maksimum dari area simulasi

Command line 'setdest' yang sudah di *generate* menghasilkan *file* yang berisi jumlah *node* dan pergerakan dari *node* yang sudah dibuat dalam *file* berbentuk .tcl. Selain pergerakan *node file* tersebut juga berisi tentang perpindahan rute.

Untuk membuat skenario *node-movement* yang terdiri dari 50 *node*, bergerak dengan kecepatan maksimum 5 m/s, simulasi akan berhenti setelah 100 detik dengan batas topologi yang diartikan sebagai 800 x 800 meter, contoh *command line* seperti pada Kode Sumber 2.2.

Berikut contoh *command line* dari skenario.txt:

```
./setdest -v 1 -n 50 -p 10 -M 5 -t 100 -x 800 -y 800 > scenario.txt
```

Kode Sumber 2.2 Contoh C*ommand Line* 'setdest'

2.8. *File Traffic Connection Pattern*

Random Traffic Connection memiliki dua buah tipe *traffic* yaitu TCP dan CBR. Keduanya dapat dibuat *menggunakan script traffic scenario generator*. *Script* ini dapat membantu kita untuk men-*generate* beban *traffic*. *Script* yang dimaksud di dalam sini adalah 'cbrgen.tcl'. Program "cbrgen.tcl" digunakan sesuai dengan *command line* pada Kode Sumber 2.3. Keterangan ditunjukan pada Tabel 2.2.

```
ns cbrgen.tcl [-type cbr|tcp] [-nn nodes] [-seed seed]
      [-mc connections] [-rate rate] > traffic-
```

Kode Sumber 2.3 Format *Command Line* 'cbrgen.tcl'

Tabel 2.2 Penjelasan *Command Line* 'cbrgen.tcl'

Parameter	Keterangan
-type cbr|tcp	Menentukan jenis *traffic* yang digunakan
-nn *nodes*	Menentukan jumlah total *node*
-s *seed*	Menentukan nilai *random seed*
-mc *connection*	Menentukan jumlah koneksi antar *node*
-rate *rate*	Menentukan jumlah paket per detik yang terkirim

Berikut contoh *command line* dari cbrgen.tcl:

```
ns cbrgen.tcl -type cbr -nn 2 -seed 1.0 -mc 1 -rate 0.25 > cbr.txt
```

Kode Sumber 2.4 Contoh C*ommand Line* 'cbrgen.tcl'

BAB III
ANALISIS DAN PERANCANGAN SISTEM

Pada bab ini akan dijelaskan mengenai dasar perancangan dari perangkat lunak yang akan dibangun dalam tugas akhir ini. Perancangan tersebut mencakup deskripsi umum aplikasi, arsitektur sistem, model fungsional dan diagram alur aplikasi.

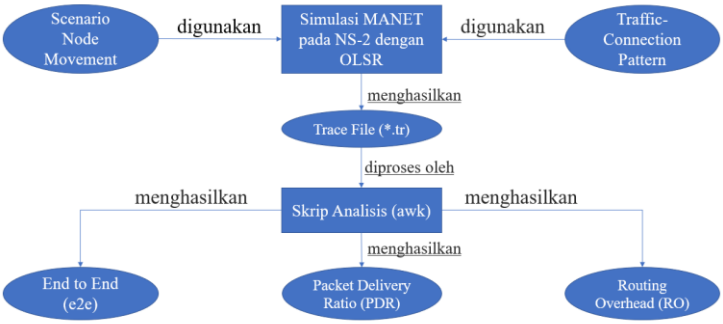
3.1. Deskripsi Umum Sistem

Pada Tugas Akhir ini akan dilakukan analisis tentang performa IEEE 802.11a dan 802.11p pada *routing protocol* OLSR di lingkungan MANET. Pada pembuatan skenario penulis menggunakan *mobility generator* yang bersifat *random way point* dan telah ada pada *Network Simulator-2* (NS-2) yaitu dengan cara men-*generate file node movement* dan membuat koneksi antar *node* menggunakan *file traffic connection pattern*. Rancangan Simulasi yang dibuat dapat dilihat pada Gambar 3.1.

Dalam penelitian ini penulis akan menganalisis performa dari IEEE WLAN 802.11a dan IEEE WLAN 802.11p pada simulasi skenario yang dijalankan pada NS-2 menggunakan *routing protocol* OLSR pada sistem operasi Linux.

Pada tiap skenario, kecepatan maksimum dari satu *node* ke *node* lainnya dibuat bervariasi yaitu 5, 10 dan 15. Dan pada skenario yang lain jumlah *node* dibuat 40, 60, dan 80. Hasil proses uji coba dari tiap skenario akan menghasilkan sebuah *trace file* yang nantinya akan dilakukan analisis perhitungan metrik *End-to-End Delay* (E2E), *Packet Delivery Ratio* (PDR), dan *Routing Overhead* (RO).

Istilah-istilah yang sering digunakan pada buku Tugas Akhir ini dapat dilihat pada Tabel 3.1.



Gambar 3.1 Skema umum sistem

Tabel 3.1 Daftar Istilah

No	Istilah	Keterangan
1	IEEE	*Institute of Electrical and Electronics Engineers*
2	MPR	*Multipoint Relay*
3	UDP	*User Datagram Protocol*
4	TCP	*Traffic Connection Pattern*
5	CBR	*Constant Bit Rate*
6	E2E	*End to End Delay*
7	PDR	*Packet Delivery Ratio*
8	RO	*Routing Overhead*

3.2. Perancangan Skenario

Skenario uji coba dalam Tugas Akhir ini dibuat dengan menggunakan *mobility generation*. Dilakukan sebanyak 60 kali, dengan rincian masing-masing 10 kali percobaan untuk jumlah 40 *node*, 60 *node*, 80 *node* dan kecepatan maksimal 5m/s, 10m/s, 15m/s. Setelah itu penguji akan membuat koneksi dari skenario yang sudah ada dengan *file traffic connection* yang sudah ada pada NS-2. Pada tugas akhir ini skenario dibuat untuk melihat

pergerakan dari *node* yang sudah dibuat berdasarkan pada tiga jumlah *node* yaitu 40, 60 dan 80 dan tiga kecepatan maksimal yaitu 5m/s, 10m/s, 15m/s. masing-masing dari 10 skenario di ujikan ke IEEE WLAN 802.11a dan IEEE WLAN 802.11p.

3.2.1. Perancangan Skenario *Node Movement (Mobility Generation)*

Perancangan skenario dibuat dengan men-*generate file node movement* yang sudah disediakan oleh NS-2 biasa kita kenal dengan *tools* bernama'setdest' yang akan digunakan dalam *file* tcl selama simulasi pada NS-2 sebagai bentuk pergerakan *node* yang berpindah-pindah.

Algoritma yang digunakan untuk pembuatan skenario *node movement* ini adalah *Random Way Point. Random Way Point* akan membuat penempatan *node node* menjadi acak. Lebih spesifiknya, mencakup gerakan, lokasi, kecepatan maupun perubahan percepatan dari waktu ke waktu dari *node node* tersebut. Sementara, sumber *traffic* yang digunakan adalah *Constant Bit Rate (CBR)*.

Tabel 3.2 Penjelasan Skenario *node movement*

No.	Parameter	Spesifikasi
1	Jumlah *Node*	-50 -100
2	Waktu Simulasi	100 detik
3	Area	800m x 800m
4	Kecepatan Maksimal	-5m/s -10m/s -15m/s
5	Jumlah *Node*	40 *node* 60 *node* 80 *node*
5	Sumber *Traffic*	CBR
6	Waktu Jeda (Dalam Detik)	1

7	Ukuran Paket	512 bytes
8	*Rate* Paket	0.25 paket per detik
9	Jumlah maksimal koneksi	1
10	Model mobilitas yang digunakan	*Random way point*

3.2.2. *Traffic-Connection Pattern*

Traffic-Connection dibuat dengan menjalankan program *cbrgren.tcl* yang telah ada pada NS-2 yang digunakan untuk memberikan koneksi dari *node node* yang sudah dibuat dalam skenario diatas selama melakukan simulasi pada NS-2.

Tabel 3.3 Penjelasan *Traffic-connection pattern*

No.	Parameter	Spesifikasi
1	-type cbr|tcp	CBR
2	-nn *nodes*	2
3	-s *seed*	1.0
4	-mc *connection*	1
5	-rate *rate*	0.25

3.3. Perancangan Simulasi NS-2

Pada perancangan kode NS-2 dengan konfigurasi MANET. Dilakukan dengan menggunakan skenario mobilitas dan digabungkan dengan *script* TCL yang berisikan konfigurasi mengenai lingkungan simulasi. Konfigurasi lingkungan simulasi MANET pada NS-2 dapat dilihat di Tabel 3.4.

Tabel 3.4 Penjelasan simulasi NS-2

No	Parameter	Spesifikasi
1	*Network Simulator*	NS-2, 2.35
2	*Routing Protocol*	OLSR
3	Waktu Simulasi	100 detik
4	*Area* Simulasi	800m x 800m
5	Banyak *node*	50
6	Radius Transmisi	250m
7	Agen Pengirim	*Constant Bit Rate* (CBR)
8	*Source/Destination*	Statis
9	*Packet Rate*	512 bytes
10	Ukuran Paket	32
11	Protokol MAC	IEEE 802.11a/p
12	Propagasi Sinyal	*TwoRayGround*
13	Tipe Kanal	*Wireless Channel*

3.4. Perancangan Metriks Analisis

Metrik yang akan dianalisis pada Tugas Akhir ini adalah *Packet Delivery Ratio* (PDR), *End-to-End Delay* (E2E), dan *Routing Overhead* (RO). Penjelasannya sebagai berikut:

3.4.1. *Packet Delivery Ratio* (PDR)

Packet delivery ratio merupakan perbandingan dari jumlah paket komunikasi yang dikirimkan dengan paket komunikasi yang diterima. *Packet delivery ratio* dihitung menggunakan persamaan (1), dimana *received* adalah jumlah paket komunikasi yang diterima dan *sent* adalah jumlah paket komunikasi yang dikirimkan. Semakin tinggi *packet delivery ratio* semakin berhasil pengiriman paket yang dilakukan.

$$Packet\ Delivery\ Ratio = \frac{Received}{Sent} \quad (1)$$

3.4.2. *End to End Delay* (E2E)

End to End Delay dapat di definisikan sebagai selisih waktu pengiriman sebuah paket saat dikirimkan dengan saat paket tersebut diterima pada *node* tujuan. E2E dihitung dengan menggunakan persamaan (2), dimana t*received* adalah waktu penerimaan paket, t*sent* adalah waktu pengiriman paket, dan sent adalah banyaknya paket data yang dikirimkan.

$$(2) \quad End\ to\ End\ Delay = \frac{t^{received} - t^{sent}}{sent}$$

3.4.3. Routing Overhead (RO)

Routing Overhead adalah jumlah paket *routing* yang ada didalam sebuah jaringan dibagi dengan jumlah keseluruhan paket data yang diterima, perhitungan menggunakan persamaan (3). Jika nilai *overhead ratio* rendah maka dapat dikatakan bahwa *routing protocol* tersebut memiliki kinerja yang cukup baik dalam hal pengiriman paket.

$$Routing\ Overhead = \frac{packet\ routing}{received} \quad (2)$$

BAB IV
IMPLEMENTASI

Pada bab ini dibahas mengenai implementasi.

4.1. Lingkungan Implementasi Protokol

Pada bab ini menjelaskan tentang lingkungan implementasi protokol.

1. Perangkat Keras
   (a) *Processor* Intel(R) *Core* (TM) i5-5200U CPU @ 2.20GHz
   (b) *Memory* 12GB
   (c) HDD 500 GB
2. Perangkat Lunak
   (a) Sistem Operasi Ubuntu 18.04
   (b) *Network Simulator* 2, 2.35

4.2. Implementasi Skenario

Implementasi skenario mobilitas MANET dimulai dengan melakukan pembuatan skenario oleh *Mobility Generation* untuk menempatkan dan menentukan *node node* yang akan di simulasikan. Setelah membuat skenario maka kita membuat jalur jalur yang menghubungkan antar *node* dengan *traffic connection pattern*. Implementasi skenarionya adalah sebagai berikut.

4.2.1. Skenario *File Node-Movement (Mobility Generation)*

Dalam implementasi yang dilakukan pada pembuatan skenario dengan *mobility generation* menggunakan *tools generate*

default yang sudah disediakan oleh NS-2 ketika kita melakukan instalasi sebelumnya yaitu 'setdest'. Skenario yang sudah di-*generate* ini akan digunakan untuk setiap simulasi yang dilakukan beradasarkan kecepatan maksimal yang berbeda beda. Algoritma yang digunakan untuk membuat skenario ini adalah *Random Way Point* sehingga penempatan *node node* yang ada akan bersifat acak. Format *command line* pada Kode Sumber 4.1 yang digunakan untuk menghasilkan gerakan acak pada *node* adalah sebagai berikut:

```
./setdest [-v version] [-n num_of_nodes] [-p pausetime] [-M maxspeed]
      [-t simtime] [-x maxx] [-y maxy] > [outdir/movement-file]
```

Kode Sumber 4.1 Format kode 'setdest'

Ketentuan-ketentuan yang diujicobakan pada skenario ini adalah versi 'setdest' *simulator* yaitu 1, jumlah *node* dalam skenario yaitu 50, waktu jeda (*pause time*) yaitu 10 detik, kecepatan maksimalnya yaitu skenario A sebesar 5m/s, skenario B sebesar 10m/s dan skenario C sebesar 15m/s, waktu simulasi yaitu 100 detik. Kemudian *file* mobilitas yang dihasilkan disimpan dalam direktori "~ns/indep-utils/cmu-scen-gen/setdest/". Pada Kode Sumber 4.2 dapat dilihat implementasi *command line* pada 'setdest' dengan berbagai kecepatan maksimal yang berbeda sesuai dan *node* sebanyak 50. Dan untuk setiap kecepatan maksimal tersebut dibuat 10 buah *file* untuk satu *routing protocol* dan satu model propagasi.

```
./setdest -v 1 -n 50 -p 10 -M 5  -t 100 -x 800 -y 800 >        scenario.txt
./setdest -v 1 -n 50 -p 10 -M 10 -t 100 -x 800 -y 800 >        scenario.txt
./setdest -v 1 -n 50 -p 10 -M 15 -t 100 -x 800 -y 800 >        scenario.txt
```

Kode Sumber 4.2 Implementasi 'setdest' untuk *speed*

Dan untuk percobaan ke 2 digunakan jumlah *node* sebagai pembandingnya.

```
./setdest -v 1 -n 40 -p 10 -M 10 -t 100 -x 800 -y 800 >        scenario.txt
./setdest -v 1 -n 60 -p 10 -M 10 -t 100 -x 800 -y 800 >        scenario.txt
./setdest -v 1 -n 80 -p 10 -M 10 -t 100 -x 800 -y 800 >        scenario.txt
```

Kode Sumber 4.3 Implementasi 'setdest' untuk jumlah *node*

4.2.2. *File traffic-connection pattern*

Dalam implementasi *random traffic connection* yang menggunakan tipe CBR ini di-*setting* dengan menggunakan skrip *traffic scenario generator*. skrip *traffic scenario generator* akan menghasilkan file bernama 'cbrgen.tcl' yang mana akan digunakan unruk membuat jaringan atau hubungan antar *node node* yang sudah dibuat pada skenario sebelumnya. Ketika akan menggunakan *file* 'cbrgen.tcl' ini, harus menentukan tipe koneksi yang digunakan (apakah CBR atau TCP), banyaknya *node* yang ada, koneksi maksimal yang digunakan, dan nilai *random seed*. *Format command line* pada Kode Sumber 4.4 yang digunakan untuk menghasilkan gerakan acak pada *node* adalah sebagai berikut.

```
ns cbrgen.tcl [-type cbr|tcp] [-nn nodes] [-seed seed]
     [-mc connections] [-rate rate] > traffic-
```

Kode Sumber 4.4 Format kode 'cbrgen.tcl'

Pada Kode Sumber 4.5 merupakan bentuk implementasi untuk menjalankan cbrgen.tcl untuk membuat *file* koneksi CBR diantara 2 *node* memiliki maksimal 1 koneksi dengan nilai *seed* 1.0 dan jumlah paket per detik sebanyak 0.25 yang disimpan dalam cbr.txt yang nantinya akan digunakan pada saat simulasi NS-2.

```
ns cbrgen.tcl -type cbr -nn 2 -seed 1.0 -mc 1
          -rate 0.25 > cbr.txt
```

Kode Sumber 4.5 Implementasi kode 'cbrgen.tcl'

4.3. Implementasi Simulasi pada NS-2

Implementasi simulasi NS-2 dilakukan dengan cara pendeskripsian lingkungan simulasi pada sebuah *file* dengan ekstensi .tcl dan .txt *file-file* ini berisi konfigurasi setiap *node* dan proses yang dilakukan selama simulasi berjalan.
Pada Kode Sumber 4.6 menunjukan skrip konfigurasi awal parameter-paremeter yang diberikan untuk menjalakan simulasi MANET pada NS-2. Isi dari parameternya adalah sebegai berikut.

Tabel 4.1 Penjelasan simulasi NS-2

No.	Parameter	Spesifikasi
1	*Channel*/	Wireless Channel
2	*Propagation*/	TwoRayGround
3	Phy/ WirelessPhy	WirelessPhyExt
4	Mac/ 802.11	802.11Ext
5	Queue Drotail/PriQueue	PriQueue
6	Antenna/OmniAntena	OmniAntena
7	Nilai X	800
8	Nilai Y	800
9	Nilai *seed*	0.0
10	*Routing Protocol*	OLSR
11	*File traffic connection*	"cbr.txt"
12	*File node movement*	"*scenario*.txt"

```
1.  set val(chan)          Channel/WirelessChannel;
2.  set val(prop)          Propagation/TwoRayGround;
3.  set val(netif)          Phy/WirelessPhyExt;
4.  set val(mac)            Mac/802_11Ext;
5.  set val(ifq)            CMUPriQueue;
6.  set val(ll)             LL;
7.  set val(ant)            Antenna/OmniAntenna;
8.  set val(ifqlen)        50;
9.  set val(nn)            50;
10. set val(rp)            OLSR;
11. set opt(x              800;
12. set opt(y)             800;
13. set val(stop)          100;
14. set val(seed)           0;
15. set val(cp)             "cbr.txt";
16. set val(sc)             "scenario.txt";
```

Kode Sumber 4.6 Konfigurasi parameter pada NS-2

Pada Kode Sumber 4.7 merupakan pengaturan dari *transmission range* yang digunakan pada simulasi. Nilai yang diubah ialah RXThresh (*receiver Sensitivity Threshold*). Nilai 1.42681e-08 pada variabel tersebut memiliki artian bahwa *range* atau jangkauan yang dapat dicapai ialah sejauh 100 meter.

```
Phy/WirelessPhy set RXThresh_ 1.42681e-08;
```

Kode Sumber 4.7 Pengaturan *Transmission range*

```
1.      set ns_      [new Simulator]
2.      set tracefd      [open trace.tr w]
3.      #set windowVsTime2 [open win.tr w]
4.      set namtrace      [open simwrls.nam w]
5.
6.
7.      $ns_ trace-all $tracefd
8.      #$ns use-newtrace
9.      $ns_ namtrace-all-wireless $namtrace $opt(x) $opt(y)
10.
11.     # set up topography object
12.     set topo       [new Topography]
13.
14.     $topo load_flatgrid $opt(x) $opt(y)
15.
16.     set god_ [create-god $val(nn)]
17.
18.     #
19.     #  Create nn mobilenodes [$val(nn)] and attach
         them to the channel.
20.     #
21.
22.     # configure the nodes
23.           $ns_ node-config -adhocRouting $val(rp) \
24.                -llType $val(ll) \
25.                -macType $val(mac) \
26.                -ifqType $val(ifq) \
27.                -ifqLen $val(ifqlen) \
28.                -antType $val(ant) \
29.                -propType $val(prop) \
30.                -phyType $val(netif) \
31.                -channelType $val(chan) \
32.                -topoInstance $topo \
33.                -agentTrace ON \
34.                -routerTrace ON \
```

```
35.             -macTrace OFF \
36.             -movementTrace ON
37.         for {set i 0} {$i < $val(nn) } { incr i } {
38.             set node_($i) [$ns_ node]
39.             $node_($i) random-motion 0;
40.         }
```

Kode Sumber 4.8 Pengaturan variabel *global* pada NS-2

Skrip yang ditunjukan pada Kode Sumber 4.8 merupakan skrip untuk pengaturan variabel *global* yang diawali dengan set ns merupakan kode untuk pembuatan *simulator* baru. Set *tracefd* dan set namtrace merupakan pengaturan untuk menentukan nama dari *trace file* dan *file network* yang akan dihasilkan dan disimpan setelah simulasi selesai dilaksanakan. Set topo merupakan pengaturan untuk objek topografi berdasarkan pada luas koordinat yang telah dikonfigurasi sebelumnya. *Set god* dan *node config – channelType* merupakan konfigurasi yang dilakukan pada *nodes* yang akan dibuat. Terakhir dilakukan perulangan untuk membuat pergerakan dari *node*s. *Nodes* yang dibuat tidak dapat melakukan pergerakan secara acak karena pergerakan *node* merupakan *trace file* yang dihasilkan oleh *mobility generator*.

Skrip pada Kode Sumber 4.9 merupakan bagian akhir dari keseluruhan skrip yang digunakan untuk menginisiasi penempatan awal *node*s yang dibuat pada skenario *node-movement* (*mobility generation*), pergerakan *node* tersebut selama waktu simulasi dilakukan dan melakukan konfigurasi pengiriman paket data yang dilakukan nantinya dihasilkan pada *file output*.tr pada potongan skrip tersebut, akan dipanggil *file* skenario *node-movement (mobility generation)* dan *traffic-connection pattern* kemudian pengiriman paket data dimulai pada detik ke-0 dan diberhentikan pada detik ke 100 seperti yang telah di konfigurasi sebelumnya.

```
1.    puts "Loading Conneciton Pattern ...."
2.    source $val(cp)
3.
4.    #Define traffice mode
5.    puts "Loading scenarion file...."
6.    source $val(sc)
7.
8.    #define node initial position in nam
9.
10.       for {set i 0} {$i < $val(nn) } { incr i } {
11.          $ns_ initial_node_pos $node_($i) 20
12.       }
13.
14.
15.   #tell nodes when the simulation ends
16.       for {set i 0} {$i < $val(nn) } { incr i } {
17.          $ns_ at $val(stop).0 "$node_($i) reset";
18.       }
19.
20.   $ns_ at $val(stop).0002 "puts \"NS EXITING....\";
21.   $ns_ halt"
22.
23.   puts $tracefd "M 0.0 nn $val(nn) x $opt(x) y $opt(y) rp
      $val(rp)"
24.   puts $tracefd "M 0.0 sc $val(sc) cp $val(cp) seed
      $val(seed)"
25.   puts $tracefd "M 0.0 prop $val(prop) ant $val(ant)"
26.
27.   puts "Starting Simulation...."
28.   $ns_ run
```

Kode Sumber 4.9 Menginisiasi penempatan awal *node*

4.4. *Implementasi Metriks Analisis*

Setelah selesai melakukan simulasi dengan NS-2 maka akan tercipta sebuah *trace file* yang berisi nilai-nilai yang dapat kita gunakan untuk melakukan analisis dari simulasi yang sudah dilaksanakan. Dalam Tugas Akhir ini penulis akan melakukan analisis dari nilai rata rata *Packet Delivery Ratio* (PDR), rata rata *Routing Overhead* (RO), dan rata-rata dari *End to End Delay* (E2E).

4.4.1. Implementasi *Packet Delivery Ratio*

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

```
awk -f PDR.awk trace.tr
```

Kode Sumber 4.10 Menjalankan *file* PDR.awk

Hasil dari perintah yang dijalankan *adalah packet delivery ratio* dari simulasi yang telah dijalankan dapat dilihat pada Gambar 4.1.

```
irsyad@irsyad-pc:~/Documents/trace/11a/M5$ awk -f PDR.awk trace1.tr
Ratio:57.6923
```

Gambar 4.1 Contoh hasil nilai PDR

4.4.2. Implementasi *End-to-End Delay*

Perhitungan *end to end delay* didasarkan pada Persamaan dibawah dan sudah dijelaskan pada bagian 3.4.2 Skrip awk untuk menghitung *end to end delay* berdasarkan kedua informasi tersebut dapat dilihat pada lampiran. Contoh perintah untuk memanggil skrip tcl untuk menganilisis *trace file* dan *output*nya dapat dilihat pada perintah awk di bawah ini.

```
awk –f e2e.awk trace.tr
```

Kode Sumber 4.11 Menjalankan *file* e2e.awk

Hasil dari perintah yang dijalankan adalah *end-to-end delay* dari simulasi yang telah dijalankan dapat dilihat pada Gambar 4.2.

```
irsyad@irsyad-pc:~/Documents/trace/11a/M5$ awk -f e2e.awk trace1.tr
end to end delay : 3.26505
```

Gambar 4.2 Contoh hasil nilai E2E

4.4.3. Implementasi *Routing Overhead*

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

```
awk –f ro.awk trace.tr
```

Kode Sumber 4.12 Menjalankan *file* ro.awk

Hasil dari perintah yang dijalankan adalah *routing overhead* dari simulasi yang telah dijalankan dapat dilihat pada Gambar 4.3.

```
irsyad@irsyad-pc:~/Documents/trace/11a/M5$ awk -f ro.awk trace1.tr
Routing Overhead : 103.072
```

Gambar 4.3 Contoh hasil nilai RO

[Halaman ini sengaja dikosongkan]

BAB V
PENGUJIAN DAN EVALUASI

Pada bab ini membahas pengujian dan evaluasi pada dari skenario NS-2 yang telah dibuat. Pengujian fungsionalitas akan dibagi ke dalam beberapa skenario pengujian.

5.1. Lingkungan Uji Coba

Lingkungan uji coba menggunakan sebuah komputer dengan spesifikasi perangkat lunak dan perangkat keras sebagai berikut:

1. Perangkat Keras
   (a) *Processor* Intel (R) *Core* (TM) i5-5200U CPU @ 2.20GHz
   (b) *Memory* 12GB
   (c) HDD 500 GB
2. Perangkat Lunak
   (a) Sistem Operasi Ubuntu 18.04
   (b) *Network Simulator* 2, OLSR, IEEE WLAN 802.11a, IEEE WLAN 802.11p, *Mobility Generation, Traffic Connection Genneration.*

5.2. Kriteria Pengujian

Pengujian pada skenario yang dihasilkan oleh *mobility generator default* dari NS-2 menggunakan beberapa kriteria. Pada Tabel 5.1 menunjukan kriteria-kriteria yang ditentukan didalam skenario.

Tabel 5.1 Penjelasan skenario pengujian

Kriteria	Spesifikasi
Skenario	MANET
Jumlah *node*	40,60,80
Kecepatan Maksimal perpindahan *node* (m/s)	5, 10, 15
Jumlah percobaan	10
Jarak *node* 1 dan *node* 2	Acak
Posisi awal *node*	Acak
Pergerakan	Acak
Protokol *Routing*	OLSR
Pengiriman Paket Data	0 – 100 Detik

5.3. Analisis *Packet Delivery Ratio (PDR)*

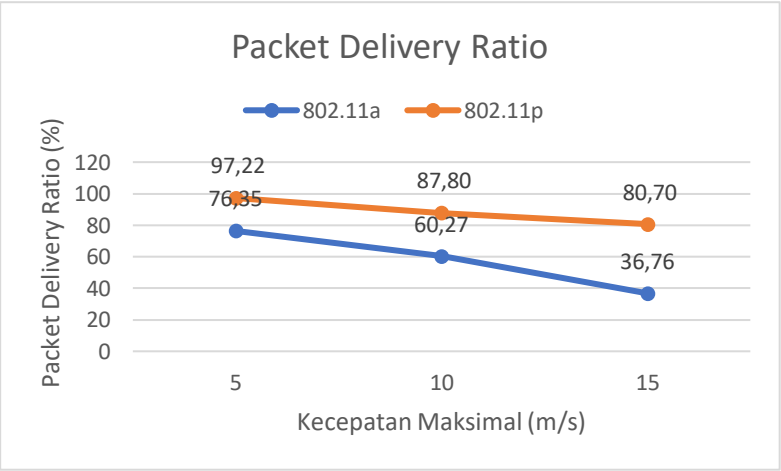
Dari hasil analisis yang dilakukan setelah melakukan uji coba beberapa kali, maka didapat nilai rata-rata PDR dengan menjadikan kecepatan maksimal perpindahan *node* sebagai acuan pada *routing protocol* OLSR. Yang nilainya adalah sebagai berikut:

Tabel 5.2 Kecepatan maksimum PDR 802.11a

Kecepatan Maksimal Perpindahan *Node* (m/s)	PDR (%)
5	76,35
10	60,27
15	36,76

Tabel 5.3 Kecepatan maksimum PDR 802.11p

Kecepatan Maksimal Perpindahan *Node* (m/s)	PDR (%)
5	97,22
10	87,80
15	80,70



Gambar 5.1 Grafik nilai PDR berdasarkan kecepatan

Pada Tabel 5.2 dan Tabel 5.3 menunjukan performa PDR yang dihasilkan oleh IEEE WLAN 802.11a dan IEEE WLAN 802.11p pada *routing protocol* OLSR dengan menggunakan kecepatan maksimum sebagai acuannya. Dapat dilihat bahwa nilai rata-rata dari *Packet Delivery Ratio* mengalami penurunan, nilai rata-rata dari IEEE WLAN 802.11a pada kecepatan maksimal 5m/s

adalah 76,35%, sementara pada kecepatan maksimal 10m/s memiliki nilai 60,27%, dan pada kecepatan maksimal 15m/s bernilai 36,76%.

Nilai dari IEEE WLAN 802.11p pada kecepatan maksimal 5m/s adalah 97,22%, sementara pada kecepatan maksimal 10m/s memiliki nilai 87,80%, dan pada kecepatan maksimal 15m/s bernilai 80,70%. Hal ini menunjukan bahwa semakin tinggi kecepatan maksimal perpindahan *node* dari skenario yang di buat, maka semakin rendah *packet delivery ratio* nya.

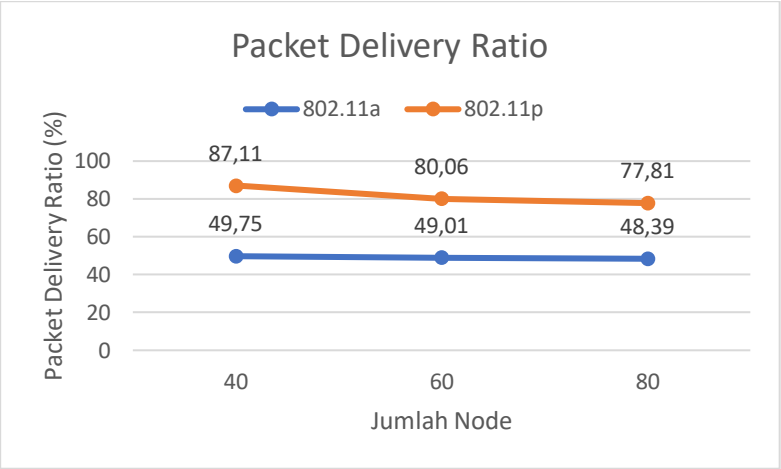
Hasil percobaan yang kedua, memakai jumlah *node* sebagai acuannya. Nilai rata-ratanya bisa dilihat di Tabel berikut:

Tabel 5.4 Jumlah *node* PDR 802.11a

Jumlah *Node*	PDR (%)
40	49,75
60	49,01
80	48,39

Tabel 5.5 Jumlah *node* PDR 802.11p

Jumlah *Node*	PDR (%)
40	87,11
60	80,06
80	77,81



Gambar 5.2 Grafik nilai PDR berdasarkan jumlah *node*

Pada Tabel 5.4 dan Tabel 5.5 menunjukan performa PDR yang dihasilkan oleh IEEE WLAN 802.11a dan IEEE WLAN 802.11p pada *routing protocol* OLSR dengan menggunakan jumlah *node* sebagai acuannya. Dapat dilihat bawah nilai rata-rata dari *Packet Delivery Ratio* mengalami penurunan, nilai rata-rata dari IEEE WLAN 802.11a dengan jumlah *node* 40 adalah 49,75%, sementara dengan jumlah *node* 60 memiliki nilai 49,01%, dan dengan jumlah *node* 80 bernilai 48,39%.

Nilai dari IEEE WLAN 802.11p dengan jumlah *node* 40 adalah 87,11%, sementara dengan jumlah *node* 60 memiliki nilai 80,06% dan dengan jumlah *node* 80 bernilai 77,81%. Hal ini menunjukan bahwa semakin besar jumlah *node* dari skenario yang di buat, maka semakin rendah *packet delivery ratio* nya.

Dari hasil skenario yang didapat, bisa dilihat bahwa nilai rata-rata dari *Packet Delivery Ratio* mengalami penurunan di setiap penambahan jumlah *node*, dan juga penambahan kecepatan maksimal perpindahan *node*. Jumlah pengiriman paket data per

paket data yang diterima berbanding terbalik dengan penambahan jumlah *node* dan juga berbanding terbalik dengan penambahan kecepatan maksimal perpindahan *node*.

5.4. Analisis *Routing Overhead (RO)*

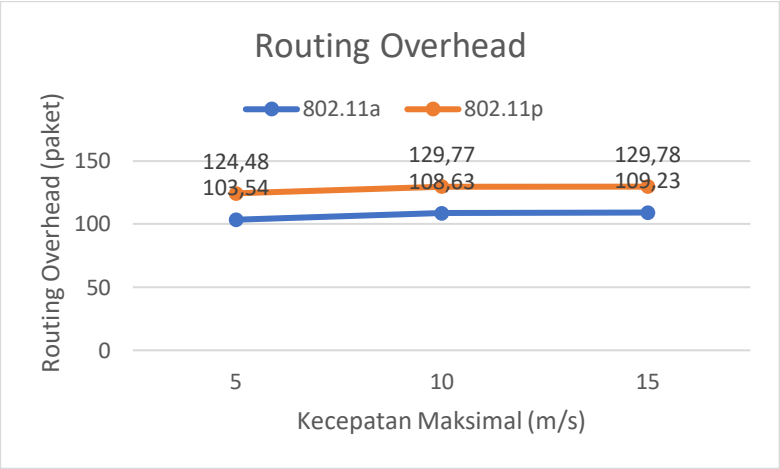
Dari hasil analisis yang dilakukan setelah melakukan uji coba beberapa kali, maka didapat nilai rata-rata RO dengan menjadikan kecepatan maksimal perpindahan *node* sebagai acuan pada *routing protocol* OLSR. Yang nilainya adalah sebagai berikut:

Tabel 5.6 Kecepatan maksimum RO 802.11a

Kecepatan Maksimal Perpindahan *Node* (m/s)	RO (paket)
5	103,54
10	108,63
15	109,23

Tabel 5.7 Kecepatan maksimum RO 802.11p

Kecepatan Maksimal Perpindahan *Node* (m/s)	RO (paket)
5	124,48
10	129,77
15	129,78



Gambar 5.3 Grafik nilai RO berdasarkan kecepatan

Pada Tabel 5.6 dan Tabel 5.7 menunjukan performa RO yang dihasilkan oleh IEEE WLAN 802.11a dan IEEE WLAN 802.11p pada *routing protocol* OLSR dengan menggunakan kecepatan maksimum sebagai acuannya. Dapat dilihat bahwa nilai rata-rata dari *Routing Overhead* mengalami kenaikan, nilai rata-rata dari IEEE WLAN 802.11a pada kecepatan maksimal 5m/s adalah 103,54 sementara pada kecepatan maksimal 10m/s memiliki nilai 108,63 dan pada kecepatan maksimal 15m/s bernilai 109,23.

Nilai rata-rata dari IEEE WLAN 802.11p pada kecepatan maksimal 5m/s adalah 124,48 sementara pada kecepatan maksimal 10m/s memiliki nilai 129,77 dan pada kecepatan maksimal 15m/s bernilai 129,78. Hal ini menunjukan bahwa semakin tinggi kecepatan maksimal perpindahan *node* dari skenario yang di buat, maka semakin tinggi pula *routing overhead* nya.

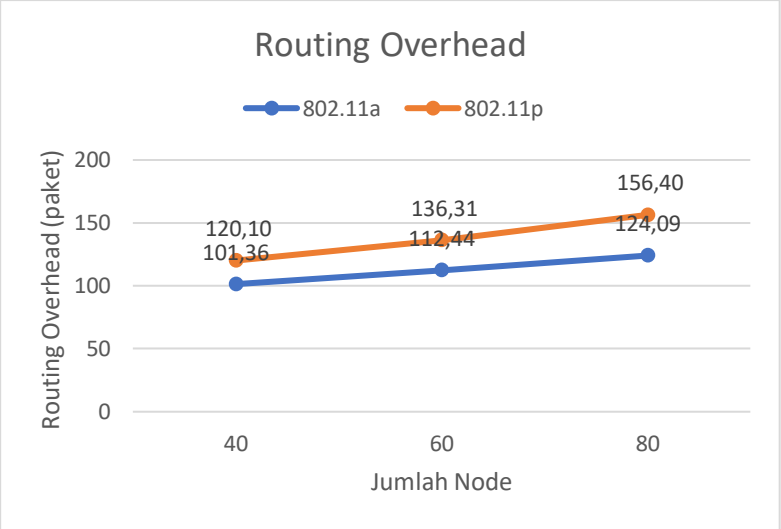
Hasil percobaan yang kedua, memakai jumlah *node* sebagai acuannya. Nilai rata-ratanya bisa dilihat di tabel berikut:

Tabel 5.8 Jumlah *node* RO 802.11a

Jumlah *node*	RO (paket)
40	101,36
60	112,44
80	124,09

Tabel 5.9 Jumlah *node* RO 802.11p

Jumlah *node*	RO (paket)
40	120,10
60	136,31
80	156,40



Gambar 5.4 Grafik nilai RO berdasarkan jumlah *node*

Pada Tabel 5.8 dan Tabel 5.9 menunjukan performa RO yang dihasilkan oleh IEEE WLAN 802.11a dan IEEE WLAN 802.11p pada *routing protocol* OLSR dengan menggunakan jumlah *node* sebagai acuannya. Dapat dilihat bahwa nilai rata-rata dari *Routing Overhead* mengalami kenaikan, nilai rata-rata dari IEEE WLAN 802.11a dengan jumah *node* 40 adalah 101,36, sementara dengan jumlah *node* 60 memiliki nilai 112,44, dan dengan jumlah *node* 80 bernilai 124,09.

Nilai rata-rata dari IEEE WLAN 802.11p dengan jumlah *node* 40 adalah 120,10, sementara dengan jumlah *node* 60 memiliki nilai 136,31, dan dengan jumlah *node* 80 bernilai 156,40. Hal ini menunjukan bahwa semakin banyak jumlah *node* dari skenario yang di buat, maka semakin tinggi pula *routing overhead* nya.

Dari hasil skenario yang didapat, bisa dilihat bahwa nilai rata-rata dari *Routing Overhead* mengalami kenaikan di setiap penambahan jumlah *node*, dan juga penambahan kecepatan maksimal perpindahan *node*. Jumlah paket routing yang ada didalam sebuah jaringan dibagi dengan jumlah keseluruhan paket data yang diterima berbanding lurus dengan penambahan jumlah *node* dan juga berbanding lurus dengan penambahan kecepatan maksimal perpindahan *node*.

5.5. Analisis *End-to-End Delay (E2E)*

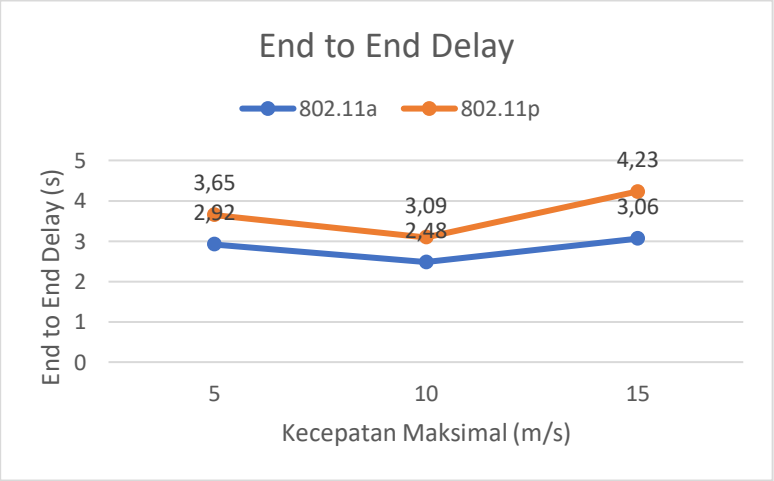
Dari hasil analisis yang dilakukan setelah melakukan uji coba beberapa kali, maka didapat nilai rata-rata E2E dengan menjadikan kecepatan maksimal perpindahan *node* sebagai acuan pada *routing protocol* OLSR. Yang nilainya adalah sebagai berikut:

Tabel 5.10 Kecepatan maksimum E2E 802.11a

Kecepatan Maksimal Perpindahan *Node* (m/s)	*End to End Delay* (Detik)
5	2,92
10	2,48
15	3,06

Tabel 5.11 Kecepatan maksimum E2E 802.11p

Kecepatan Maksimal Perpindahan *Node* (m/s)	*End to End Delay* (Detik)
5	3,65
10	3,09
15	4,23



Gambar 5.5 Grafik nilai E2E berdasarkan kecepatan

Pada Tabel 5.10 dan Tabel 5.11 menunjukan performa E2E yang dihasilkan oleh IEEE WLAN 802.11a dan IEEE WLAN 802.11p pada *routing protocol* OLSR dengan menggunakan kecepatan maksimal sebagai acuannya. Dapat dilihat bahwa nilai rata-rata dari *End to End Delay* mengalami penurunan lalu mengalami kenaikan, nilai rata-rata dari IEEE WLAN 802.11a pada kecepatan maksimal 5m/s adalah 2,92 sementara pada kecepatan maksimal 10m/s memiliki nilai 2,48 dan pada kecepatan maksimal 15m/s bernilai 3,06.

Nilai dari IEEE WLAN 802.11p pada kecepatan maksimal 5m/s adalah 3,65 sementara pada kecepatan maksimal 10m/s memiliki nilai 3,09 dan pada kecepatan maksimal 15m/s bernilai 4,23. Nilai rata-rata dari E2E dengan menggunakan kecepatan maksimal perpindahan *node* sebagai acuan bersifat fluktuatif pada jumlah waktu yang dibutuhkan untuk mengirimkan paket data yang dikirimkan.

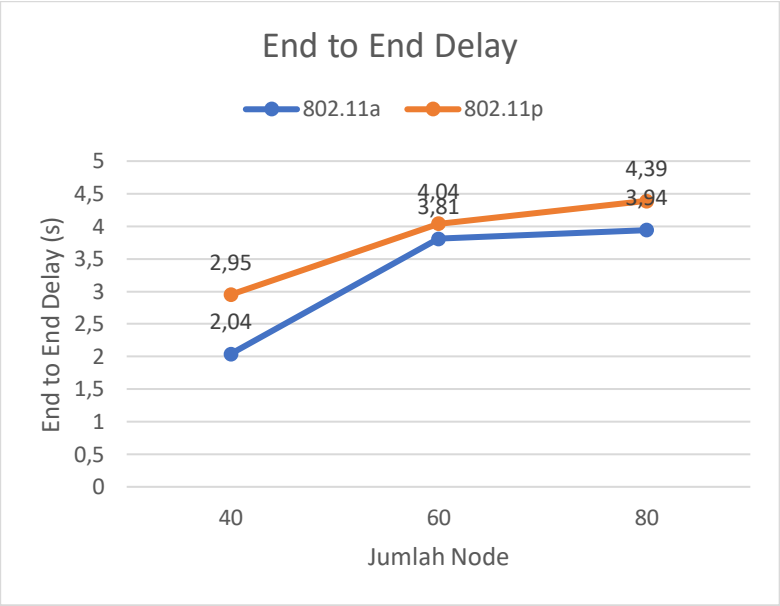
Hasil percobaan yang kedua, memakai jumlah *node* sebagai acuannya. Nilai rata-ratanya bisa dilihat di tabel berikut:

Tabel 5.12 Jumlah *node* E2E 802.11a

Jumlah *node*	*End to End Delay* (Detik)
40	2,04
60	3,81
80	3,94

Tabel 5.13 Jumlah *node* E2E 802.11p

Jumlah *node*	*End to End Delay (*Detik*)*
40	2,95
60	4,04
80	4,39



Gambar 5.6 Grafik nilai E2E berdasarkan jumlah *node*

Pada Tabel 5.12 dan Tabel 5.13 menunjukan performa E2E yang dihasilkan oleh IEEE WLAN 802.11a dan IEEE WLAN 802.11p pada *routing protocol* OLSR dengan menggunakan jumlah *node* sebagai acuannya. Dapat dilihat bahwa nilai rata-rata dari *End to End Delay* mengalami kenaikan, nilai rata-rata dari IEEE WLAN 802.11a dengan jumlah *node* 40 adalah 2,04 sementara dengan jumlah *node* 60 memiliki nilai 3,81 dan dengan jumlah *node* 80 bernilai 3,94.

Nilai dari IEEE WLAN 802.11p dengan jumlah *node* 40 adalah 2,95 sementara dengan jumlah *node* 60 memiliki nilai 4,04 dan dengan jumlah *node* 80 bernilai 4,39. Hal ini menunjukan bahwa semakin banyak jumlah *node* dari skenario yang di buat, maka semakin tinggi pula *end to end delay* nya.

Dari hasil skenario yang didapat, bisa dilihat bahwa nilai rata-rata dari *End to End* mengalami kenaikan di setiap penambahan jumlah *node*. Maka dapat disumpulkan bahwa selisih waktu pengiriman sebuah paket saat dikirimkan dengan saat paket tersebut diterima berbanding lurus dengan penambahan jumlah *node*.

[Halaman ini sengaja dikosongkan]

BAB VI
KESIMPULAN DAN SARAN

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

6.1. Kesimpulan

Kesimpulan yang dapat diambil dalam tugas akhir ini adalah IEEE WLAN 802.11p pada parameter PDR memiliki kinerja lebih baik daripada IEEE WLAN 802.11a karena 802.11p memiliki tingkat keberhasilan dalam pengiriman paket yang lebih tinggi daripada 802.11a. Hal ini dapat dibuktikan salah satunya dengan membandingkan hasil rata-rata PDR yang dimiliki 802.11p dan 802.11a berdasarkan kecepatan maksimal sebagai acuannya. Rata-rata 802.11p pada kecepatan maksimal 5m/s adalah 97,22%, pada kecepatan maksimal 10m/s adalah 87,80%, dan pada kecepatan maksimal 15m/s adalah 80,70. Sedangkan, 802.11a memiliki nilai lebih kecil yaitu pada kecepatan maksimal 5m/s adalah 76,35%, pada kecepatan maksimal 10m/s adalah 60,27%, dan pada kecepatan maksimal 15m/s adalah 36,76%.

IEEE WLAN 802.11a pada parameter RO memiliki kinerja lebih baik daripada IEEE WLAN 802.11p karena 802.11a memiliki jumlah paket *overhead* yang lebih sedikit dari jumlah paket *overhead* yang dimiliki oleh 802.11p. Hal ini dapat dibuktikan salah satunya dengan membandingkan hasil rata-rata RO yang dimiliki 802.11a dan 802.11p berdasarkan kecepatan maksimal sebagai acuannya. Rata-rata 802.11a pada kecepatan maksimal 5m/s adalah 103,54. Pada kecepatan maksimal 10m/s adalah 108,63. Dan pada kecepatan maksimal 15m/s adalah 109,23. Sedangkan, 802.11p memiliki nilai lebih besar yaitu pada kecepatan maksimal 5m/s adalah 124,48. pada kecepatan maksimal 10m/s memiliki nilai 129,77. Dan pada kecepatan maksimal 15m/s adalah 129,78.

IEEE WLAN 802.11a pada parameter E2E memiliki kinerja lebih baik daripada IEEE 802.11p karena 802.11a memiliki selisih waktu pengiriman saat paket dikirimkan dan saat diterima yang lebih kecil daripada 802.11p. Hal ini dapat dibuktikan salah satunya dengan membandingkan hasil rata-rata E2E yang dimiliki 802.11a dan 802.11p berdasarkan kecepatan maksimal sebagai acuannya. Rata-rata 802.11a pada kecepatan maksimal 5m/s adalah 2,92. Pada kecepatan maksimal 10m/s adalah 2,48. Dan pada kecepatan maksimal 15m/s adalah 3,06. Sedangkan, 802.11p memiliki nilai lebih besar yaitu pada kecepatan maksimal 5m/s adalah 3,65. Pada kecepatan maksimal 10m/s adalah 3,09. Dan pada kecepatan maksimal 15m/s adalah 4,23.

6.2. Saran

Berdasarkan pada hasil penelitian yang telah disimpulkan di atas dan dalam rangka pengembangan penelitian, maka dikemukakan beberapa saran, yaitu:

1. Dilakukan simulasi dengan menggunakan *routing protocol* MANET yang berbeda seperti DSR, ZRP, atau AOMDV.
2. Dilakukan simulasi dengan lingkungan topologi VANET.
3. Dilakukan simulasi dengan acuan yang berbeda seperti *pause time*, lama waktu jalannya simulasi, dan lain lain.
4. Dilakukan simulasi dengan IEEE WLAN 802.11 selain IEEE WLAN 802.11a dan IEEE WLAN 802.11p.

DAFTAR PUSTAKA

[1] P. C. Sekhar, M. R. P. Kumar, B. P. Kumar, & C. Koteswararao, "Impact of Routing Overhead in A Real-Time MANET Environment," vol. 4, p. 7, 2013.

[2] "Introduction of Mobile ad hoc network (MANET)," BINUS Malang. [Online]. Available: http://binus.ac.id/malang/2017/09/introduction-of-mobile-ad-hoc-network-manet/. [Accessed: 02-Jan-2019].

[3] N. H. Saeed, M. F. Abbod, & H. S. Al-Raweshidy, "MANET routing protocols taxonomy," 2012, pp. 123–128.

[4] I. N. B. Hartawan, "Optimasi Pemilihan Multi-Point Relay dengan Congestion Detection dalam Optimized Link State Routing pada Mobile Ad-Hoc Network," Digilib ITS, p. 1, 2014.

[5] "Aprillando, A. 2007. Cara Kerja dan Kinerja Protokol Optimized Link State Routing (OLSR) pada Mobile Ad hoc network (MANET), Tugas Akhir. Jakarta: Fakultas Teknik Unika AtmaJaya.

[6] Malatras, A., Pavlou, G., Gouveris, S., Sivavakeesar, S., Self-Configuring and Optimizing Mobile Ad Hoc Networks, Centre for Communications Systems Research, Department of Electronic Engineering, University of Surrey, UK.

[7] A. Goldsmith, *Wireless communications*, 1. publ. Cambridge, U.K.: Cambridge University Press, 2004.

[8] "18.2 Two-ray ground reflection model." [Online]. Available: https://www.isi.edu/nsnam/ns/doc/node218.html. [Accessed: 05-Jan-2019].

[9] R. Kumar, P. Verma, & Y. Singh, "Mobile Ad Hoc Networks and It's Routing Protocols," vol. 7, no. 8, p. 10, 2013.

[10] Close, D. B., Robbins, A. D., Rubin, P. H., Stallman, R., & Oostrum, P. V., 1995. The AWK Manual (2 ed.). Free Software Foundation, Inc.

[Halaman ini sengaja dikosongkan]

LAMPIRAN

```
Phy/WirelessPhyExt set CSTresh_ 6.31e-12;
Phy/WirelessPhyExt set Pt_ 0.001;
Phy/WirelessPhyExt set freq_ 5.18e+9;
Phy/WirelessPhyExt set noise_floor_ 2.512e-13;
Phy/WirelessPhyExt set L_ 1.0;
Phy/WirelessPhyExt set PowerMonitorThresh_ 1.259e-13;
Phy/WirelessPhyExt set HeaderDuration_ 0.000020;
Phy/WirelessPhyExt set BasicModulationScheme_ 0;
Phy/WirelessPhyExt set PreambleCaptureSwitch_ 1;
Phy/WirelessPhyExt set DataCaptureSwitch_ 0;
Phy/WirelessPhyExt set SINR_PreambleCapture 2.5118;
Phy/WirelessPhyExt set SINR_DataCapture_ 100.0;
Phy/WirelessPhyExt set trace_dist_ 1e6;
Phy/WirelessPhyExt set PHY_DBG_ 0;
Mac/802_11Ext set CWMin_ 15;
Mac/802_11Ext set CWMax_ 1023;
Mac/802_11Ext set slotTime_ 0.000009;
Mac/802_11Ext set SIFS_ 0.000016;
Mac/802_11Ext set ShortRetryLimit_ 7;
Mac/802_11Ext set LongRetryLimit_ 4;
Mac/802_11Ext set HeaderDuration_ 0.000020;
Mac/802_11Ext set SymbolDuration_ 0.000004;
Mac/802_11Ext set BasicModulationScheme_ 0;
Mac/802_11Ext set use_802_11a_flag_ true;
Mac/802_11Ext set RTSThreshold_ 2346;
Mac/802_11Ext set MAC_DBG 0;
Mac/802_11Ext set RTS_ 20.0;
Mac/802_11Ext set CTS_ 14.0;
```

Kode Sumber 7.1 Konfigurasi IEEE WLAN 802.11a

```
Phy/WirelessPhyExt set CSTresh_ 3.9810717055349694e-13s;
Phy/WirelessPhyExt set Pt_ 0.002;
Phy/WirelessPhyExt set freq_ 5.9e+9;
Phy/WirelessPhyExt set noise_floor_ 1.26e-13;
Phy/WirelessPhyExt set L_ 1.0;
Phy/WirelessPhyExt set PowerMonitorThresh_ 3.981071705534985e-18;
Phy/WirelessPhyExt set HeaderDuration_ 0.000040;
Phy/WirelessPhyExt set BasicModulationScheme_ 0;
Phy/WirelessPhyExt set PreambleCaptureSwitch_ 1;
Phy/WirelessPhyExt set DataCaptureSwitch_ 1;
Phy/WirelessPhyExt set SINR_PreambleCapture 3.1623;
Phy/WirelessPhyExt set SINR_DataCapture_ 10.0;
Phy/WirelessPhyExt set trace_dist_ 1e6;
Phy/WirelessPhyExt set PHY_DBG_ 0;
Mac/802_11Ext set CWMin_ 15;
Mac/802_11Ext set CWMax_ 1023;
Mac/802_11Ext set slotTime_ 0.000013;
Mac/802_11Ext set SIFS_ 0.000032;
Mac/802_11Ext set ShortRetryLimit_ 7;
Mac/802_11Ext set LongRetryLimit_ 4;
Mac/802_11Ext set HeaderDuration_ 0.000040;
Mac/802_11Ext set SymbolDuration_ 0.000008;
Mac/802_11Ext set BasicModulationScheme_ 0;
Mac/802_11Ext set use_802_11p_flag_ true;
Mac/802_11Ext set RTSThreshold_ 2346;
Mac/802_11Ext set MAC_DBG 0;
```

Kode Sumber 7.2 Konfigurasi IEEE WLAN 802.11p

```
# A 10-node example for ad-hoc simulation with DSR

# Define options
set val(chan)           Channel/WirelessChannel    ;# channel type
set val(prop)           Propagation/TwoRayGround   ;# radio-propagation model
set val(netif)          Phy/WirelessPhyExt         ;# network interface type
set val(mac)            Mac/802_11Ext              ;# 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)             50                         ;# number of mobilenodes
set val(rp)             OLSR                       ;# routing protocol
set opt(x)              800                        ;# X dimension of topography
set opt(y)              800                        ;# Y dimension of topography
set val(stop)           100                        ;# time of simulation end
set val(seed)           0                               ;
set val(cp)                                "cbr.txt";
set val(sc)                                "scenario1.txt";

Phy/WirelessPhy set RXThresh_ 1.42681e-08;

set ns_           [new Simulator]
set tracefd       [open trace1.tr w]
#set windowVsTime2 [open win.tr w]
set namtrace      [open simwrls.nam w]


$ns_ trace-all $tracefd
#$ns use-newtrace
$ns_ namtrace-all-wireless $namtrace $opt(x) $opt(y)

# set up topography object
set topo       [new Topography]

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

```
# set up topography object
set topo       [new Topography]

$topo load_flatgrid $opt(x) $opt(y)

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



#
#  Create nn mobilenodes [$val(nn)] and attach them to the channel.
#

# 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 \
                         -movementTrace ON

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

```
# Define node movement model
puts "Loading Conneciton Pattern ...."
source $val(cp)

#Define traffice mode
puts "Loading scenarion file...."
source $val(sc)

#define node initial position in nam

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


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

$ns_ at $val(stop).0002 "puts \"NS EXITING....\";
$ns_ halt"

puts $tracefd "M 0.0 nn $val(nn) x $opt(x) y $opt(y) rp $val(rp)"
puts $tracefd "M 0.0 sc $val(sc) cp $val(cp) seed $val(seed)"
puts $tracefd "M 0.0 prop $val(prop) ant $val(ant)"

puts "Starting Simulation...."
$ns_ run
```

Kode Sumber 7.3 Konfigurasi OLSR.tcl

```
#
# nodes: 2, max conn: 1, send rate: 4.0, seed: 1.0
#
#
# 1 connecting to 2 at time 2.5568388786897245
#
set udp_(0) [new Agent/UDP]
$ns_ attach-agent $node_(1) $udp_(0)
set null_(0) [new Agent/Null]
$ns_ attach-agent $node_(2) $null_(0)
set cbr_(0) [new Application/Traffic/CBR]
$cbr_(0) set packetSize_ 512
$cbr_(0) set interval_ 4.0
$cbr_(0) set random_ 1
$cbr_(0) set maxpkts_ 10000
$cbr_(0) attach-agent $udp_(0)
$ns_ connect $udp_(0) $null_(0)
$ns_ at 2.5568388786897245 "$cbr_(0) start"
#
#Total sources/connections: 1/1
#
```

Kode Sumber 7.4 Contoh cbr.txt

```
BEGIN {
seqno = -1;
# droppedPackets = 0;
# receivedPackets = 0;
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] = -1;
}
}
for(i=0; i<=seqno; i++) {
if(delay[i] > 0) {
n_to_n_delay = n_to_n_delay + delay[i];
}
}
n_to_n_delay = n_to_n_delay/count;
printf("end to end delay : ")
print n_to_n_delay * 1000 ;
}
```

Kode Sumber 7.5 Implementasi e2e.awk

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

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

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

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

END{
        printf"Ratio:%.4f\n",(recvLine/sendLine)*100;
}
```

Kode Sumber 7.6 Implementasi PDR.awk

```
BEGIN{
        recvs = 0;
        besaran_paket = 0;
}
{
        if (($1 == "s" || $1 == "r") && $4 == "RTR"){
                recvs++;
        }
        if (($1 == "s" || $1 == "r") && $4 == "RTR")
        {
                besaran_paket = besaran_paket + $8;
        }
}
END{
        printf("Routing Overhead : %.3f\n", besaran_paket/recvs);
}
```

Kode Sumber 7.7 Implementasi ro.awk

[Halaman ini sengaja dikosongkan]

BIODATA PENULIS

Irsyad Iswanda Putra, akrab di panggil Irsyad lahir di Yogyakarta pada tanggal 15 Januari 1995. Penulis menempuh pendidikan formal di SD M CondongCatur Sleman, SMPN 8 Yogyakarta, SMAN 9 Yogyakarta, dan saat ini sedang menempuh perguruan tinggi di Institut Teknologi Sepuluh Nopember Departemen Informatika Fakultas Teknologi Informasi dan Komunikasi pada tahun 2013. Terlibat aktif pada organisasi mahasiswa tingkat jurusan antara lain Departemen Dalam Negeri Himpunan Mahasiswa Teknik Computer-Informatika (HMTC) ITS sebagai staff tahun 2014 – 2015.

Penulis dapat dihubungi melalui surat elektronik iswandairsyad@gmail.com.