TUGAS AKHIR - IF184802

PENGEMBANGAN AD-HOC ON DEMAND DISTANCE VECTOR (AODV) DENGAN KONSEP POWER AND DELAY AWARE DI LINGKUNGAN DENGAN ENERGI YANG HETEROGEN PADA MOBILE AD-HOC NETWORK (MANET)

AHMAD HANAN
NRP 05111440007005

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

Departemen Teknik Informatika
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

(Halaman ini sengaja dikosongkan)

TUGAS AKHIR – IF184802

PENGEMBANGAN AD-HOC ON DEMAND DISTANCE VECTOR (AODV) DENGAN KONSEP POWER AND DELAY AWARE DI LINGKUNGAN DENGAN ENERGI YANG HETEROGEN PADA MOBILE AD-HOC NETWORK (MANET)

AHMAD HANAN
NRP 05111440007005

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

Departemen Teknik Informatika
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

(Halaman ini sengaja dikosongkan)

UNDERGRADUATE THESES – IF184802

AD-HOC ON DEMAND DISTANCE VECTOR (AODV) DEVELOPMENT WITH POWER AND DELAY AWARE CONCEPTS IN THE ENVIRONMENT WITH HETEROGEN ENERGY IN MOBILE AD-HOC NETWORK (MANET)

AHMAD HANAN
NRP 05111440007005

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

Department of Informatics
Faculty of Intelligent Electrical and Informatics Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2020

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

PENGEMBANGAN AD-HOC ON DEMAND DISTANCE VECTOR (AODV) DENGAN KONSEP POWER AND DELAY AWARE DI LINGKUNGAN DENGAN ENERGI YANG HETEROGEN PADA MOBILE AD-HOC NETWORK (MANET)

TUGAS AKHIR

Diajukan untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Arsitektur dan Jaringan Komputer
Program Studi S-1 Departemen Teknik Informatika
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember

Oleh:
Ahmad Hanan
NRP: 05111440007005

Disetujui oleh Pembimbing Tugas Akhir:

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

SURABAYA
JANUARI, 2020

(Halaman ini sengaja dikosongkan)

PENGEMBANGAN AD-HOC ON DEMAND DISTANCE VECTOR (AODV) DENGAN KONSEP POWER AND DELAY AWARE DI LINGKUNGAN DENGAN ENERGI YANG HETEROGEN PADA MOBILE AD-HOC NETWORK (MANET)

Nama Mahasiswa : Ahmad Hanan
NRP : 05111440007005
Departemen : Teknik Informatika – FTEIC ITS
Dosen Pembimbing 1 : Dr.Eng. Radityo Anggoro, S.Kom., M.Sc

Abstrak

Mobile Ad-hoc Network (MANET) atau jaringan *Mesh* adalah sebuah jaringan yang terdiri dari banyak *node* bergerak yang saling berkomunikasi satu sama lain. Jaringan MANET memiliki karakteristik berupa topologi yang bersifat dinamis dan terbatasnya energi baterai yang dimiliki oleh *node* bergerak. Ada beberapa masalah yang ditimbulkan karena habisnya energi baterai suatu *node* yang bisa menyebabkan rusaknya rute pada jaringan MANET. Masalah-masalah itu di antaranya adalah hilangnya paket; *node* akan menginisialisasi *route discovery* atau pencarian rute ulang, sehingga menyebabkan banyaknya konsumsi *bandwith*, meningkatkan *delay*, dan mengurangi *throughput*.

AODV merupakan salah satu *routing protocol* yang termasuk dalam klasifikasi *reactive routing protocol*, sebuah protokol yang hanya akan membuat rute ketika *node* sumber membutuhkannya. AODV memiliki dua fase, yaitu *route discovery* dan *route maintenance. Route discovery* digunakan untuk meminta dan meneruskan informasi rute yang terdiri dari proses pengiriman *Route Request* (RREQ) dan *Route Reply* (RREP), sedangkan *route maintenance* digunakan untuk mengetahui informasi adanya

kesalahan pada rute. Pada fase ini terdapat proses pengiriman *Route Error* (RERR).

Dalam tugas akhir ini, telah dilakukan studi kinerja jaringan MANET dengan menggunakan menggunakan energi yang heterogen dengan menggunakan *routing protocol* baru bernama *Power and Delay-aware Multi-path Routing Protocol* (PDMRP) yang memperbaiki kekurangan dari *Stable Path Routing Protocol based on Power Awareness* (SPR), dan *Modified Ad-hoc On-Demand Distance Vector Routing Protocol* (MAODV) dalam segi *throughput*, *end-to-end delay*, dan *loss rate*.

Kata kunci: MANET, AODV, PDMRP, NS2, AWK

AD-HOC ON DEMAND DISTANCE VECTOR (AODV) DEVELOPMENT WITH POWER AND DELAY AWARE CONCEPTS IN THE ENVIRONMENT WITH HETEROGEN ENERGY IN MOBILE AD-HOC NETWORK (MANET)

Student's Name : Ahmad Hanan
Student's ID : 05111440007005
Department : Informatics – FTEIC ITS
First Advisor : Dr.Eng. Radityo Anggoro, S.Kom., M.Sc.

Abstract

Mobile Ad-hoc Networks (MANETs) also called mesh networks, consist of a large number of mobile nodes that communicate with each other. The principle characteristics of MANET are the dynamic topology and the limited battery power of mobile nodes. The discharge of the battery causes many problems such as the loss of the packets and the re-initialization of route discovery which leads to lot of bandwidth consumption, increase in the delay and decrease in the throughput.

AODV is an example of reactive routing protocol classification, a protocol that will only create a route when the source node needs it. AODV have 2 phase which are route discovery and route maintenance. Route discovery is used for requesting and forwarding a route information that consist of Route Request (RREQ) and Route Reply (RREP), meanwhile route maintenance that consist of Route Error (RERR) is used for finding out an error information in route.

In this thesis, a study on MANET networks using heterogeneous energy using a new routing protocol called Power

and Delay-aware Multi-path Routing Protocol (PDMRP) that improve the shortcomings of Stable Path Routing Protocol based on Power Awareness (SPR), and Modified Ad-hoc On-Demand Distance Vector Routing Protocol (MAODV) in terms of throughput, end-to-end delay, and loss rates.

Keyword: MANET, AODV, PDMRP, NS2, AWK

KATA PENGANTAR

Bismillaahi ar-Rahmaani ar-Rahiim

Puji syukur kepada Allah Yang Maha Esa atas segala karunia dan rahmat-Nya penulis dapat menyelesaikan Tugas Akhir yang berjudul **"Pengembangan *Ad-hoc On demand Distance Vector* (AODV) dengan Konsep *Power and Delay Aware* di Lingkungan dengan Energi yang Heterogen pada *Mobile Ad-hoc Network* (MANET)".**

Harapan penulis semoga apa yang tertulis di dalam buku Tugas Akhir ini dapat bermanfaat bagi pengembangan ilmu pengetahuan saat ini.

Dalam pelaksanaan dan pembuatan Tugas Akhir ini tentunya sangat banyak bantuan yang penulis terima dari berbagai pihak, tanpa mengurangi rasa hormat penulis ingin mengucapkan terima kasih sebesar-besarnya kepada:

1. Allah SWT atas semua rahmat yang diberikan sehingga penulis dapat menyelesaikan Tugas Akhir ini.
2. Bapak Mochamad Nawawi dan Ibu Muslikhah selaku kedua orangtua penulis atas segala dukungan berupa motivasi serta doa sehingga penulis dapat mengerjakan Tugas Akhir ini.
3. Zia Ul Haq selaku adik penulis atas segala dukungan yang telah diberikan sehingga penulis tetap semangat dalam mengerjakan Tugas Akhir ini.
4. Bapak Dr.Eng. Radityo Anggoro, S.Kom., M.Sc., selaku dosen pembimbing, atas arahan dan bantuannya dalam pengerjaan Tugas Akhir ini.
5. Bapak/Ibu dosen, staf, dan karyawan Departemen Teknik Informatika ITS yang telah banyak memberikan dukungan, ilmu, dan bimbingan yang tak ternilai kepada penulis.

6. Teman seperjuangan semasa kuliah, khususnya teman-teman angkatan 2014 yang selama ini sudah membantu penulis dalam menyelesaikan Tugas Akhir ini.
7. Teman-teman yang turut membantu penulis dalam menyelesaikan Tugas Akhir ini.
8. Teman seangkatan di CSSMoRA ITS D'14 (Zaka, Ishar, Elva, Stone, Faiz, Satria, Fikri, Ica, Cut, Ocha, Nani, Zuli, Farid, (alm.) Udin, Rifqi, Mahbub, Umdah, Fariz, Atok) yang selama ini selalu menjadi teman Penulis sejak masa matrikulasi dan selama kuliah di ITS.
9. Sahabat-sahabat di IKSAB Cabang Surabaya yang selama ini sudah menyemangati Penulis dalam menjalankan kuliah di ITS.
10. Sahabat-sahabat baik di UKM maupun organisasi ekstra kampus yang senantiasa menyemangati Penulis untuk mengerjakan dan menyelesaikan Tugas Akhir ini.
11. Juga tidak lupa kepada semua pihak yang belum sempat disebutkan satu per satu yang telah membantu penulis dalam menyelesaikan Tugas Akhir ini.

Penulis menyadari bahwa masih terdapat kekurangan, kesalahan, maupun kelalaian yang telah penulis lakukan. Oleh karena itu, saran dan kritik yang membangun sangat dibutuhkan untuk penyempurnaan Tugas Akhir ini.

Surabaya, Januari 2020
Penulis

Ahmad Hanan

DAFTAR ISI

(Halaman ini sengaja dikosongkan)

DAFTAR GAMBAR

(Halaman ini sengaja dikosongkan)

DAFTAR TABEL

(Halaman ini sengaja dikosongkan)

BAB I
PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi bertujuan untuk memudahkan pengguna dari teknologi tersebut serta untuk mengatasi berbagai kekurangan yang ada pada teknologi sebelumnya. Alhasil kekurangan itu bisa diatasi sehingga hasil yang didapatkan menjadi lebih sempurna daripada sebelumnya.

Mobile Ad-hoc Network (MANET) atau jaringan *Mesh* adalah sebuah jaringan yang terdiri dari banyak *node* bergerak yang saling berkomunikasi satu sama lain. Jaringan MANET memiliki karakteristik berupa topologi yang bersifat dinamis dan terbatasnya energi baterai yang dimiliki oleh *node* bergerak. Ada beberapa masalah yang ditimbulkan karena habisnya energi baterai suatu *node* yang bisa menyebabkan rusaknya rute pada jaringan MANET. Masalah-masalah itu di antaranya adalah hilangnya paket; *node* akan menginisialisasi *route discovery* atau pencarian rute ulang, sehingga menyebabkan banyaknya konsumsi *bandwith*, meningkatkan *delay*, dan mengurangi *throughput*.

Dalam tugas akhir ini, telah dilakukan studi kinerja jaringan MANET dengan menggunakan energi yang heterogen dengan menggunakan *routing protocol* baru bernama *Power and Delay-aware Multi-path Routing Protocol* (PDMRP) yang memperbaiki kekurangan dari *Stable Path Routing Protocol based on Power Awareness* (SPR), dan *Modified Ad-hoc On-Demand Distance Vector Routing Protocol* (MAODV) dalam segi *throughput*, *end-to-end delay*, dan *loss rate* [1].

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam Tugas Akhir ini adalah membahas bagaimana kinerja *routing protocol* bernama

Power and Delay-aware Multi-path Routing Protocol (PDMRP) dengan energi yang heterogen pada MANET?

1.3 Batasan Permasalahan

Permasalahan yang dibahas pada Tugas Akhir ini memiliki batasan sebagai berikut:

1. Jaringan yang digunakan adalah jaringan *Mobile Ad-hoc Networks* (MANET).
2. *Routing protocol* yang diujicobakan yaitu *Ad-hoc On-Demand Distance Vector* (AODV).
3. Simulasi pengujian jaringan menggunakan *Network Simulator 2* (NS-2).
4. Analisis kinerja didasarkan pada *Packet Dilivery Ratio* (PDR), *Routing Overhead* (RO), dan *End-to-End Delay* (E2E).

1.4 Tujuan

Tujuan dari pembuatan Tugas Akhir ini adalah untuk melakukan evaluasi kinerja *routing protocol* bernama *Power and Delay-aware Multi-path Routing Protocol* (PDMRP) pada lingkungan dengan energi yang heterogen pada MANET.

1.5 Manfaat

Manfaat dari hasil pengerjaan Tugas Akhir ini adalah mengetahui kinerja *routing protocol* bernama *Power and Delay-aware Multi-path Routing Protocol* (PDMRP) dengan energi yang heterogen pada MANET.

1.6 Metodologi

Berikut ini adalah beberapa metodologi yag digunakan dalam pembuatan Tugas Akhir ini:

1.6.1 Penyusunan Proposal Tugas Akhir

Tahapan awal dari Tugas Akhir ini adalah penyusunan Proposal Tugas Akhir. Proposal Tugas Akhir berisi pendahuluan, deskripsi, dan gagasan metode-metode yang dibuat dalam Tugas Akhir ini. Pendahuluan ini terdiri atas hal yang menjadi latar belakang diajukannya Tugas Akhir, rumusan masalah yang diangkat dalam Tugas Akhir, batasan masalah untuk Tugas Akhir, dan manfaat dari hasil pembuatan Tugas Akhir ini. Selain itu dijabarkan pula tinjauan pustaka yang digunakan sebagai referensi pendukung pembuatan Tugas Akhir. Terdapat pula sub bab jadwal kegiatan yang menjelaskan jadwal pengerjaan Tugas Akhir.

1.6.2 Studi Literatur

Tahapan studi literatur pada Tugas Akhir ini merupakan tahap untuk melakukan pembelajaran, pencarian, dan pengumpulan informasi yang berguna dalam penyusunan Tugas Akhir ini. Beberapa literatur yang akan dibahas pada tahap ini adalah literatur yang berhubungan tentang *Mobile Ad-hoc Network* (MANET), *Ad-hoc On demand Distance Vector* (AODV), simulator jaringan *Network Simulator*-2 (NS-2), AWK, *Node-Movement Generator*, dan *File Traffic Connection Pattern*.

1.6.3 Analisis dan Desain Sistem

Pada tahap ini dilakukan analisis dari hasil percobaan modifikasi *routing protocol* AODV yang dibuat. Data yang dianalis berasal dari perhitungan *Packet Delivery Ratio* (PDR), *Routing Overhead* (RO), dan *End-to-End Delay* (E2E) paket dari *node* ke *node* lainnya. Hal ini bertujuan untuk merumuskan solusi yang tepat untuk konfigurasi *routing protocol* AODV yang dimodifikasi dalam lingkungan topologi jaringan MANET.

1.6.4 Implementasi

Implementasi model jaringan pada tahap ini akan dilakukan dalam *routing protocol* AODV yang telah dimodifikasi pada lingkungan dengan energi heterogen pada jaringan MANET. Modifikasi di sini akan dibuat berdasarkan studi literatur dan pembelajaran konsep teknologi yang berasal dari perangkat lunak yang ada serta mempelajari modifikasi *routing protocol* yang pernah dibuat sebelumnya. Skenario-skenario yang digunakan pada Tugas Akhir ini akan dibuat menggunakan *Node-Movement Generator* dan disimulasikan ke dalam simulator jaringan NS-2. Kinerja dari *routing protocol* hasil modifikasi akan dianalisis menggunakan AWK dengan parameter *Packet Delivery Ratio* (PDR), *Routing Overhead* (RO), dan *End-to-End Delay* (E2E)

1.6.5 Pengujian dan Evaluasi

Pada tahap ini, akan dilakukan pengujian menggunakan simulator jaringan NS-2 yang akan menghasilkan *trace file*. Dari *Trace file* tersebut akan dihitung *Packet Delivery Ratio* (PDR), *Routing Overhead* (RO), dan *End-to-End Delay* (E2E) untuk menguji performa AODV yang telah dimodifikasi.

1.6.6 Penyusunan Buku

Pada tahap ini dilakukan penyusunan buku yang menjelaskan seluruh konsep, teori dasar dari metode yang digunakan, implementasi, serta hasil yang telah dikerjakan sebagai dokumentasi dari pelaksanaan Tugas Akhir.

1.7 Sistematika Penulisan Laporan

Sistematika penulisan laporan Tugas Akhir adalah sebagai berikut:

Bab I. Pendahuluan

Bab ini berisi tentang penjelasan mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi, dan sistematika penulisan dari pembuatan Tugas Akhir.

Bab II. Tinjauan Pustaka

Bab ini berisi pembahasan mengenai teori-teori dasar atau penjelasan dari metode dan *tools* yang digunakan dalam Tugas Akhir. Bab ini berisi tentang penjelasan singkat mengenai jaringan MANET, *routing protocol* AODV, simulator jaringan bernama NS-2, AWK, *Node-Movement Generator*, dan *File Traffic Connection Pattern*.

Bab III. Analisis dan Desain Sistem

Bab ini berisi pembahasan mengenai analisis dan desain sistem menggunakan skenario mobilitas, perancangan simulasi pada simulator jaringan NS-2, perancangan modifikasi *routing protocol* AODV, serta perancangan metrik analisis yang meliputi *Packet Delivery Ratio* (PDR), *Routing Overhead* (RO), dan *End-to-End Delay* (E2E).

Bab IV. Implementasi

Bab ini akan berisi penjelasan mengenai implementasi Tugas Akhir berdasarkan rancangan perangkat lunak yang meliputi lingkungan pembangunan perangkat lunak, implementasi skenario, implementasi simulasi NS-2, dan implementasi metrik analisis yang telah dilakukan pada bab sebelumnya.

Bab V. Uji Coba dan Evaluasi

Bab ini berisikan hasil uji coba dan evaluasi dari implementasi pada *routing protocol* AODV yang telah dilakukan untuk menyelesaikan masalah yang dibahas pada Tugas Akhir.

Bab VI. Kesimpulan dan Saran

Bab ini merupakan bab yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan, masalah-masalah yang

dialami pada proses pengerjaan Tugas Akhir, dan saran untuk pengembangan solusi ke depannya.

Daftar Pustaka

Bab ini berisi daftar pustaka atau daftar literatur dari materi-materi yang dijadikan sebagai referensi dalam pengerjaan Tugas Akhir.

Lampiran

Bab ini merupakan bab tambahan yang berisi lampiran kode-kode yang digunakan pada Tugas Akhir.

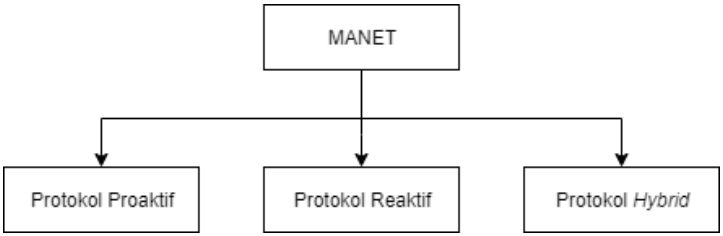
BAB II
TINJAUAN PUSTAKA

Bab ini berisi pembahasan mengenai teori-teori dasar atau penjelasan dari metode dan *tools* yang digunakan dalam Tugas Akhir. Teori-teori yang akan di bahas pada bab ini meliputi penjelasan mengenai MANET, AODV, simulator jaringan NS-2, AWK, *Node-Movement Generator*, dan *File Traffic Connection Pattern*.

2.1 *Mobile Ad-hoc Network* (MANET)

Mobile Ad-hoc Network (MANET) merupakan sebuah jaringan yang terbentuk dari beberapa *node* yang mampu melakukan pergerakan secara bebas dan dinamis. MANET juga memungkinkan terjadinya komunikasi jaringan tanpa adanya ketergantungan pada ketersedian infrastruktur jaringan yang tetap. Setiap *node* yang ada dalam jaringan MANET dapat bertindak sebagai *host* maupun *router* yang mana hal ini memungkinkan *node* untuk meneruskan paket ke *node* berikutnya. Setiap *node* ini dapat saling melakukan komunikasi antara satu dengan yang lainnya tanpa adanya *access point.*

Perangkat yang terdapat pada jaringan MANET harus mampu melakukan pendeteksian terhadap keberadaan perangkat lain dan melakukan pengaturan yang diperlukan untuk melakukan komunikasi dan berbagi data. Selain itu, jaringan MANET juga memungkinkan perangkat untuk mempertahankan koneksi ke jaringan serta dengan mudah menambahkan dan menghapus perangkat yang terdapat pada jaringan. Karena pergerakan *node* yang dinamis, topologi jaringan dapat berubah secara cepat dan tidak dapat diprediksi dari waktu ke waktu. Jaringan MANET ini bersifat desentralisasi, di mana jaringan dan pengiriman pesan harus dijalankan oleh *node* sendiri [2].



Gambar 2.1 Jenis-jenis *routing protocol* pada MANET

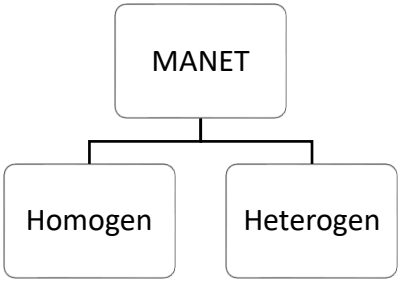
Routing Protocol yang ada pada jaringan MANET dapat dikelompokkan menjadi tiga jenis, yaitu protokol proaktif, reaktif, dan *hybrid*. Berikut ini adalah pengertian dari masing-masing protokol tersebut:

1. **Protokol Proaktif** adalah jenis protokol yang bekerja berdasarkan *routing table* yang terus menerus diperbarui secara reguler. Protokol ini bersifat *table driven* yang bisa diartikan bahwa setiap *node* menyimpan tabel yang berisi informasi rute ke setiap *node* yang diketahuinnya. Artinya sebuah node mengetahui semua rute ke *node* lain yang berada dalam jaringan tersebut. Informasi rute akan diperbaharui secara berkala jika terjadi perubahan link. Berikut ini yang termasuk ke dalam protokol *routing* proaktif adalah *Destination Sequenced Distance Vector* (DSDV), *Cluster Switch Gateway Routing* (CSGR), *Wireless Routing Protocol* (WRP), dan *Optimized Linkstate* (OLSR).
2. **Protokol Reaktif** adalah jenis protokol yang bekerja berdasarkan permintaan untuk membuat rute baru atau perubahan rute. Protokol ini bersifat *on-demand* yang bisa diartikan bahwa protokol ini akan membentuk sebuah rute dari satu *node* sumber ke *node* tujuan hanya berdasarkan pada permintaan *node* sumber tersebut. Sehingga proses pencarian rute hanya dilakukan apabila *node* sumber membutuhkan komunikasi dengan *node* tujuan.Contoh protokol *routing* reaktif adalah *Dynamic*

Source Routing (DSR), *Ad-hoc On-demand Distance Vector* (AODV), *Temporally Ordered Routing Algorithm* (TORA), *Associativy Based Routing* (ABR), dan *Signal Stability Routing* (SSR) [3] [4] [5].

3. **Protokol *Hybrid*** adalah jenis protokol *routing* yang merupakan kombinasi antara kedua tipe protokol *routing*, proaktif dan reaktif. Contoh protokol *routing* hybrid adalah *Zone Routing Protocol* (ZRP) [5].

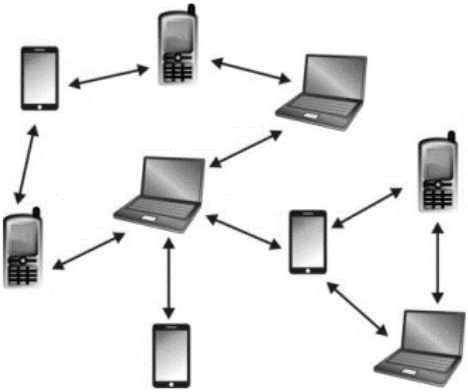
Jaringan MANET bisa dikelompokkan menjadi dua kelompok, yakni Homogen dan Heterogen. Jaringan MANET homogen adalah jaringan MANET yang hanya satu jenis perangkat saja yang ada di dalam jaringan tersebut. Sedangkan jaringan MANET heterogen adalah jaringan MANET yang memiliki ragam dalam jaringannya, dalam hal ini jenis perangkat yang berbeda seperti ponsel dan laptop yang ada pada satu jaringan. **Gambar 2.2** yang ada di bawah ini merupakan ilustrasi dari pengelompokan jaringan MANET ke dalam dua kelompok, yakni Homogen dan Heterogen.



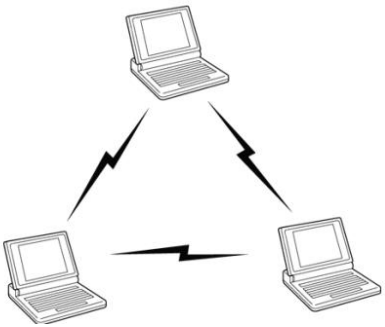
Gambar 2.2 Pengelompokan Jaringan MANET

Pada **Gambar 2.3** dan **Gambar 2.4** yang ada di bawah ini merupakan contoh dari ilustrasi jaringan MANET, yang terbentuk dari sekumpulan perangkat *mobile* seperti ponsel dan laptop. Perangkat yang terhubung ke jaringan tersebut harus mampu melakukan pendeteksian terhadap keberadaan perangkat lain serta

melakukan pengaturan yang diperlukan sehingga dapat proses komunikasi dan pembagian data bisa dilakukan [6] .



Gambar 2.3 Ilustrasi MANET Heterogen

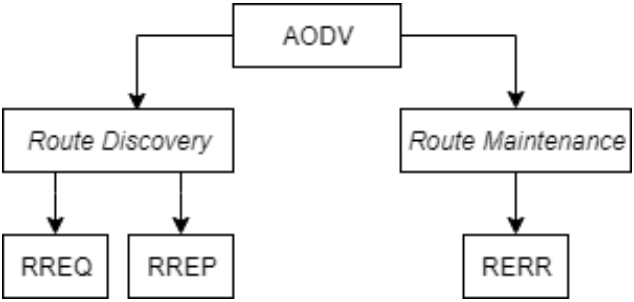


Gambar 2.4 Ilustrasi MANET Homogen

2.2 *Ad-hoc On demand Distance Vector* (AODV)

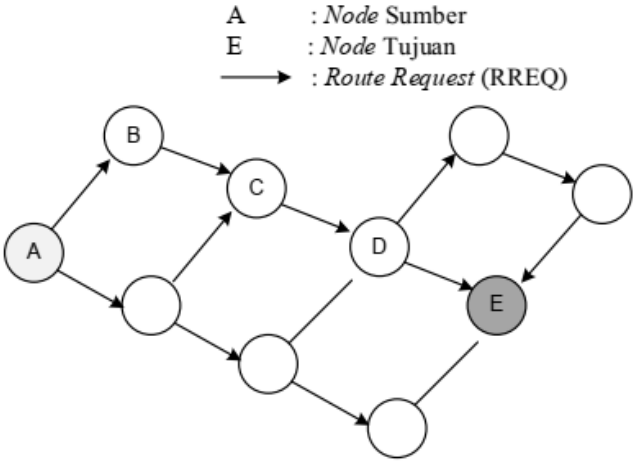
Ad-hoc On demand Distance Vector (AODV) merupakan suatu *routing protocol* yang berjenis protokol reaktif yang mana jenis dari *routing protocol* in hanya akan membuat sebuah rute saat dibutuhkan. AODV sendiri dikembangkan oleh C. E. Perkins, E.M. Belding-Royer, dan S. Das pada RFC 3561.

Menjaga *timer-based state* pada setiap *node* sesuai dengan penggunaan tabel *routing* merupakan ciri utama dari *routing protocol* ini. Tabel *routing* akan kadaluarsa jika jarang digunakan. *Routing protocol* AODV memiliki *route discovery* dan r*oute maintenance. Route discovery* terdiri dari *Route Request* (RREQ) dan *Route Reply* (RREP). Sedangkan *route maintenance* berisi *Route Error* (RERR) [5]. Berikut ini adalah ilustrasi dari ketiga pesan tersebut.

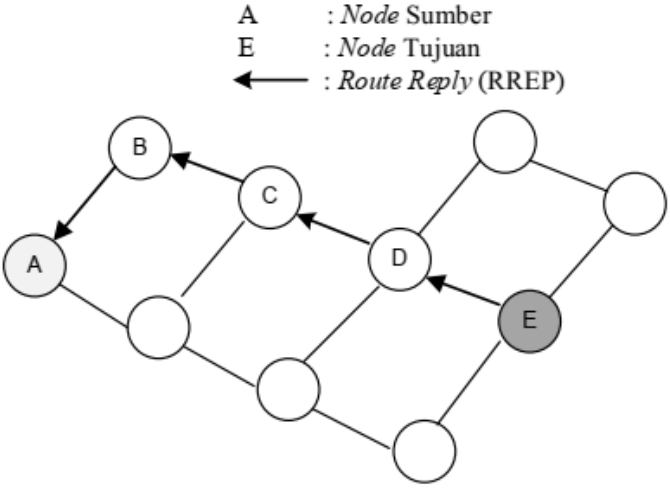


Gambar 2.5 Tiga jenis pesan yang ada pada *routing protocol* AODV

Pada **Gambar 2.6** dan **Gambar 2.7** yang ada di bawah ini, maksud dari masing-masing gambar tersebut adalah mengenai ilustrasi dari *Route Request* (RREQ) dan *Route Reply* (RREP). Di sana menunjukkan bahwa saat *node* sedang melakukan transmisi pesan ke tujuan baru, pada saat itu juga diperlukan *node* sumber yang akan digunakan untuk melakukan penyiaran pesan *route request* atau RREQ. *Node* akan menerima balasan *route request* atau RREQ hanya jika mempunyai rute ke tujuan bersama dengan nomor urut yang lebih besar atau sama dengan yang terdapat pada *route request* atau RREQ. Namun, apabila *node* sumber telah menemukan rute tujuan, maka *node* tujuan akan mengirim pesan respon berupa paket *route reply* atau RREP.



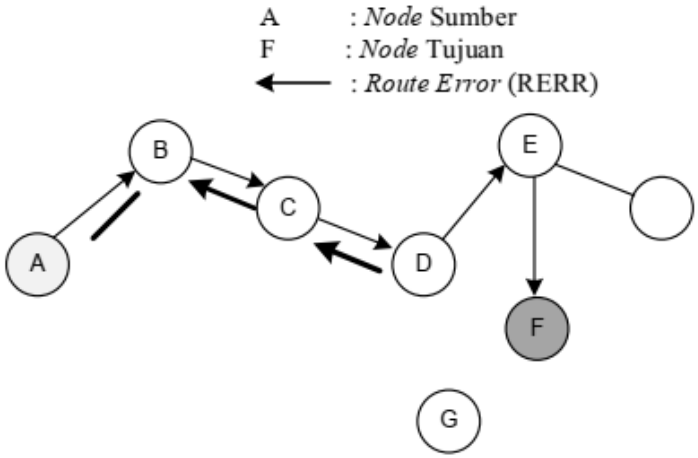
Gambar 2.6 Paket Route Request (RREQ) pada AODV



Gambar 2.7 Paket Route Reply (RREP) pada AODV

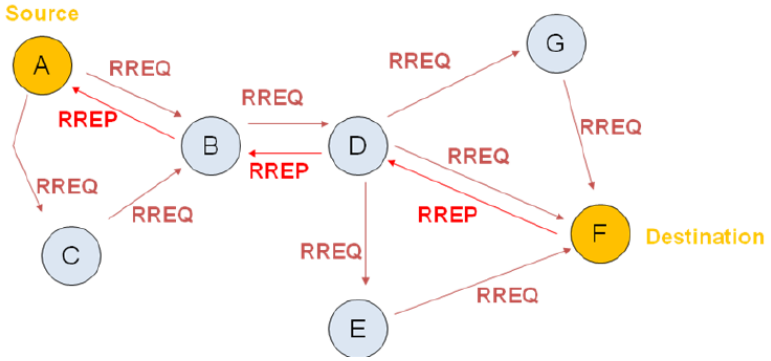
Pada gambar selanjutnya, yakni **Gambar 2.8**, menunjuukan bahwa terdapat pesan kesalahan dalam penggunaan paket *route*

error atau RERR pada tahap pemeliharaan rute. Permintaan rute menggunakan paket *route request* atau RREQ dan *route reply* atau RREP dipertimbangkan. Namun, pada paket kesalahan rute dan tahap pemeliharaan rute dihasilkan karena kegagalan rute. Proses *route reply* atau RREP sama seperti permintaan rute tetapi dilakukan dalam urutan terbalik [7].



Gambar 2.8 Paket Route Error (RERR) pada AODV

Ad-hoc On demand Distance Vector (AODV) merupakan suatu *routing protocol* yang dapat melakukan pengiriman pesan antar *node* yang memungkinkan *node-node* di dalam jaringan untuk melakukan transmisi pesan melalui lingkungan *routing protocol* AODV tersebut menuju *node* yang tidak dapat dihubungi secara langsung. *Routing protocol* AODV melakukan ini dengan cara menemukan rute yang bisa dilalui oleh pesan. Tak hanya itu saja, *routing protocol* AODV juga akan memastikan rute ini tidak mengalami perulangan (*loop*), menangani perubahan rute, dan membuat rute baru apabila terjadi *error*. **Gambar 2.9** yang ada di bawah ini merupakan ilustrasi dari pencarian rute yang terdapat pada *routing protocol* AODV.



Gambar 2.9 Ilustrasi pencarian rute *routing* protocol AODV *[8]*

Ilustrasi yang ada pada **Gambar 2.9** di atas merupakan contoh dari proses *route discovery* yang terjadi pada *routing protocol* AODV. Pada ilustrasi tersebut, terdapat penggambaran mengenai bagaimana *node* asal, yakni *node* A mencari rute untuk menuju *node* tujuan, yakni *node* F. *Node* A akan membuat paket *Route Request* (RREQ) dan melakukan *broadcast* kepada semua *node* tetangganya atau *neighbor node*. Ketika *destination sequence number* yang ada pada paket *route request* atau RREQ mengalami kesamaan atau lebih kecil dari yang ada pada *routing table* dan rute menuju *node* tujuan belum ditemukan, maka paket tersebut tidak akan dilanjutkan (*drop*). Sedangkan ketika *destination sequence number* pada *route request* atau RREQ memiliki nilai lebih besar dari ada pada *routing table*, maka *entry* pada *routing table* akan diperbarui dan paket tersebut akan diteruskan oleh *neighbor node* sekaligus membuat *reverse path* menuju *node* asal. Paket *route request* atau RREQ akan diteruskan hingga mencapai *node* F. Kemudian, jika rute menuju *node* F sudah terbentuk di dalam *routing table* dan memiliki *routing flags* dengan status *up* atau dalam kondisi valid, maka *node* F akan mengirimkan paket *Route Reply* (RREP) melalui rute tersebut menuju *node*.

Berikut ini adalah *field* yang terdapat pada *routing table* di dalam setiap *node* yang menggunakan *routing protocol* AODV:

- *Destination Address:* berisi alamat dari *node* tujuan yang nantinya akan digunakan sebagai penentuan rute.
- *Destination Sequence Number*: *sequence number* dari jalur komunikasi yang nantinya akan saling bekerjasama untuk menentukan rute.
- *Next Hop*: alamat *node* selanjutnya yang akan meneruskan paket data ke *node* tujuan.
- *Hop Count*: jumlah hop dari *node* asal hingga *node* tujuan.
- *Lifetime*: waktu yang diperlukan *node* untuk menerima RREP dengan satuan milidetik.
- *Routing Flags*: menunjukkan status dari sebuah rute. Ada tiga jenis dari status ini, yakni *Up* (valid), *Down* (tidak valid), ataukah sedang diperbaiki [5].

Pada Tugas Akhir ini, penulis menggunakan *routing protocol* AODV yang telah dimodifikasi pada lingkungan dengan energi yang heterogen pada jaringan MANET.

2.3 *Network Simulator-2* (NS-2)

Network Simulator-2 (NS-2) merupakan sebuah *event-driven simulator* yang didesain secara spesifik untuk melakukan penelitian dalam bidang jaringan komunikasi komputer. NS-2 pertama kali dikembangkan oleh University of California Berkeley. Ini merupakan suatu sistem yang bekerja pada sistem Unix/Linux.

NS-2 dibangun dari dua bahasa pemrograman, yaitu pertama bahasa C++ yang berfungsi sebagai *library* yang berisi *event scheduler*, protokol, dan *network component* yang nantinya akan diimplementasikan pada simulasi oleh *user*. Kedua adalah bahasa *object oriented* Tcl/Otcl yang akan digunakan pada *script* simulasi yang ditulis oleh pengguna dari NS-2 ini.

Faktor yang menjadi penyebab dari digunakannya bahasa C++ ini pada *library* adalah karena bahasa C++ mampu mendukung *runtime* simulasi yang cepat, meskipun simulasi ini

melibatkan simulasi jumlah paket dan sumber data dalam jumlah besar. Sedangkan bahasa Tcl memberikan respon *runtime* yang lebih lambat daripada C++, namun jika terdapat kesalahan, respon Tcl terhadap kesalahan *syntax* dan perubahan *script* berlangsung dengan cepat dan interaktif.

NS-2 bersifat *open source* dibawah *Gnu Public License* (GPL), sehingga NS dapat diunduh dan digunakan secara gratis melalui website http://www.isi.edu/nsnam. Sifat *open source* mengakibatkan pengembangan NS menjadi lebih dinamis. NS bisa digunakan di berbagai macam OS seperti Windows, FreeBSD, Linux, Solaris, dan Mac [9].

Pada Tugas Akhir ini, NS-2 digunakan untuk melakukan simulasi lingkungan menggunakan protokol AODV yang sudah dimodifikasi. *Trace file* yang dihasilkan oleh NS-2 juga digunakan untuk mengukur peforma *routing* protokol AODV yang sudah dimodifikasi.

2.3.1 Instalasi

Sebelum memulai instalasi NS-2, ada beberapa *package* yang telah ter-*install*. Untuk melakukan instalasi *dependency* yang dibutuhkan, langkah yang dapat dilakukan adalah dengan menjalankan perintah yang ditunjukkan pada **Gambar 2.10**.

```
sudo apt install build-essential autoconf automake
libxmu-dev wget gdb bz ip2 flex
```

Gambar 2.10 Perintah untuk meng*install dependency* NS-2

Setelah melakukan instalasi *dependency* yang dibutuhkan, langkah selanjutnya adalah melakukan ekstrak *package* NS-2. Perintah untuk mengekstrak keduanya sebagaimana yang ada pada **Gambar 2.11**.

```
tar -xvzf ns-allinone-2.35_gcc5.tar.gz
```

Gambar 2.11 Perintah untuk mengekstrak *package* NS-2

Untuk melakukan instalasi simulator jaringan NS-2, perintah yang harus dijalankan adalah dengan memasukkan perintah `./install` pada folder NS-2.

2.3.2 *Trace File*

Trace file merupakan *file* hasil simulasi yang dilakukan oleh NS-2. Isi dari *trace file* ini adalah mengenai informasi detail pengiriman paket data. Maksud dari digunakannya *trace file* di sini adalah untuk melakukan analisis performa *routing protocol* yang disimulasikan. Detail dari penjelasan mengenai *trace file* sebagaimana ditunjukkan pada **Tabel 2.1**.

Tabel 2.1 Detail Penjelasan *Trace File* AODV

Kolom	Penjelasan	Isi
1	*Event*	s : *sent* r : *received* f : *forwarded* D : *dropped*
2	*Time*	Waktu terjadinya *event*
3	ID *Node*	_x : dari 0 hingga banyak *node* pada topologi
4	*Layer*	AGT : *application* RTR : *routing* LL : *link layer* IFQ : *packet queue* MAC : *MAC* PHY : *physical*
5	*Flag*	--- : Tidak ada
6	*Sequence Number*	Nomor paket
7	*Packet Type*	AODV : paket *routing* AODV cbr : berkas paket CBR (*Constant Bit Rate*) RTS : *Request To Send* yang dihasilkan MAC 802.11

		CTS : *Clear To Send* yang dihasilkan MAC 802.11 ACK : MAC ACK ARP : Paket *link layer address resolution protocol*
8	Ukuran	Ukuran paket pada *layer* saat itu
9	Detail MAC	[a b c d] a : perkiraan waktu paket b : alamat penerima c : alamat penerima d : IP header
10	*Flag*	-------- : Tidak ada
11	*Detail* IP *source, destination,* dan *nexthop*	[a:b c:d e f] a : IP *source node* b : *port source node* c : IP *destination node* (jika -1 berarti *broadcast*) d : *port destination node* e : IP *header* ttl f : IP *nexthop* (jika 0 berarti *node* 0 atau *broadcast*)

Berikut ini adalah contoh potongan isi dari *trace file*:

```
s 2.556838879 _198_ AGT  --- 0 cbr 512 [0 0 0 0]
[energy 80.000000 ei 0.000 es 0.000 et 0.000 er
0.000] ------- [198:0 199:0 32 0] [0] 0 0

r 2.556838879 _198_ RTR  --- 0 cbr 512 [0 0 0 0]
[energy 80.000000 ei 0.000 es 0.000 et 0.000 er
0.000] ------- [198:0 199:0 32 0] [0] 0 0

s 2.556953879 _198_ MAC  --- 0 AODV 106 [0 ffffffff
c6 800] [energy 80.000000 ei 0.000 es 0.000 et 0.000
er 0.000] ------- [198:255 -1:255 30 0] [0x2 1 1
[199 0] [198 4]] (REQUEST)
```

Gambar 2.12 Contoh isi dari *Trace File*

2.3.3 *Network Animator* (NAM) *File*

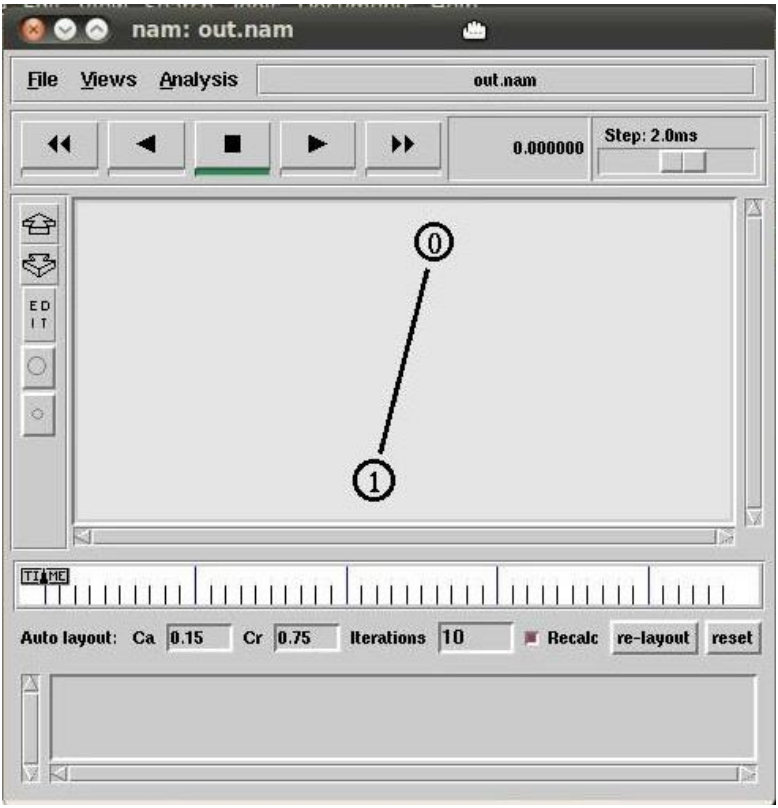
NAM *file* adalah sebuah alat animasi yang berbasis bahasa Tcl/TK yang mana fungsi dari NAM *file* ini adalah untuk membuat visulisasi jejak dari setiap *event* dalam suatu simulasi jaringan dan data jejak paket secara nyata. NAM file mendukung tata letak topologi, animasi tingkat paket, dan berbagai alat inspeksi data. NAM dimulai dari LBL. NAM dibangun dari kolaborasi dengan proyek Vint. Saat ini NAM tengah dikembangkan sebagai sebuah *project* yang bersifat *open source* yang diselenggarakan di Sourceforge.

Pengembangan animator jaringan bernama NAM ini dimulai pada tahun 1990 yang memiliki fungsi sebagai alat sederhana untuk menghidupkan data jejak paket. Data jejak ini biasanya diturunkan sebagai output dari simulator jaringan seperti ns atau dari pengukuran jaringan yang nyata, misalnya menggunakan *tcpdump*. Steven McCanne menulis versi asli sebagai anggota Kelompok Penelitian Jaringan di Laboratorium Nasional Lawrence Berkeley, dan terkadang ia juga memperbaiki desain karena ia membutuhkannya dalam penelitiannya. Marylou Orayani

memperbaikinya penelitian tersebut lebih lanjut dan menggunakannya untuk penelitian pendidikan Master yang sedang ia tempuh selama musim panas 1995 hingga musim semi 1996 [10]. Di bawah ini adalah contoh dari perintah untuk menjalankan NAM *file*.

```
nam out.nam
```

Ketika perintah di atas sudah dijalankan, maka akan ada muncul jendela animasi simulasi jaringan seperti gambar berikut ini.



Gambar 2.13 Contoh jendela munculan yang menampilkan file NAM

2.4 AWK

AWK merupakan bahasa pemrograman yang memungkinkan untuk menangani tugas-tugas (*tasks*) seperti program yang sangat singkat seperti satu atau dua baris, *text processing,* dan ekstraksi data. AWK merupakan sebuah program filter untuk teks, seperti halnya perintah grep pada terminal linux. AWK dapat digunakan untuk mencari bentuk / model dalam sebuah berkas teks ke dalam bentuk teks lain. AWK merupakan urutan pola dan tindakan yang memberi tahu apa yang harus dicari dalam input data dan apa yang harus dilakukan ketika ditemukan. AWK mencari satu set file untuk baris yang cocok dengan salah satu pola; ketika garis yang cocok ditemukan, tindakan yang sesuai dilakukan. AWK dapat juga digunakan untuk melakukan proses aritmatika seperti yang dilakukan oleh perintah expr. AWK sama halnya seperti bahasa *shell* atau C yang memiliki karakteristik yaitu sebagai *tool* yang cocok untuk *jobs* juga sebagai pelengkap untuk *filter* standar [11] [12].

Pada Tugas Akhir ini, AWK digunakan untuk membuat script menghitung *Packet Delivery Ratio* (PDR*), Routing Overhead* (RO), dan *End-to-End Delay* dari *trace file* NS-2.

2.5 *Node-Movement Generator*

Simulasi jaringan bernama *Network Simulator* 2 (NS-2) menyediakan utilitas *shell* "setdest" yang menciptakan pernyataan mobilitas deterministik, yang dapat dimasukkan ke dalam skrip simulasi Tcl. Ditulis dalam C ++, "setdest" yang dapat dieksekusi terletak di direktori `ns/indep-utils/cmu-sen-gen/setdest`.

Simulator jaringan NS-2 ini menyediakan dua versi “setdest”. Untuk versi 1 adalah “setdest” yang dikembangkan oleh Carnegie Mellon University (CMU). Sedangkan versi 2 adalah “setdest” yang dikembangkan oleh University of Michigan [13].

Di bawah ini adalah format *Command line* dari masing-masing versi yang telah disebutkan di atas:

- Versi 1 (Carnegie Mellon University)

```
./setdest -v <1> -n <nodes> -p <pause time> -M <max speed> -t <simulation time> -x <max X> -y <max Y>
```

- Versi 2 (University of Michigan)

```
./setdest -v [2] -n <nodes> -s <speed type> -m <min speed> -M <max speed> -t <simulation time> -P <pause type> -p <pause time> -x <max X> -y <max Y>
```

Pada Tugas Akhir ini, penulis akan menggunakan versi 2 yang dikembangkan oleh University of Michigan sebab memungkinkan untuk digunakan dalam menyelesaikan permasalahan semacam ini.

Sebelum menjalankan program simulasi, pengguna harus menjalankan program "setdest". Caranya adalah dengan menjalankan *Command line* "setdest" seperti yang telah disebutkan di atas atau ditunjukan seperti dibawah ini dan keterangannya ditunjukan pada **Tabel 2.2**.

```
./setdest -v [2] -n <nodes> -s <speed type> -m <min speed> -M <max speed> -t <simulation time> -P <pause type> -p <pause time> -x <max X> -y <max Y>
```

Tabel 2.2 Penjelasan *Command line* 'setdest'

Parameter	Keterangan
-v version	Menentukan versi dari 'Setdest' yang digunakan.
-n num	Menentukan jumlah *node* yang dibuat pada skenario
-s speedtype	Menentukan kecepatan dari *node*.
-m minspeed	Menentukan kecepatan minimum dari sebuah *node.*

-M maxspeed	Menentukan kecepatan maksimum dari sebuah *node*.
-t simtime	Menentukan lama waktu jalannya simulasi.
-P pausetype	Menentukan jenis pemberhentian sebuah paket.
-p pausetime	Menentukan durasi berhenti pada sebuah paket apabila sudah sampai di tujuan
-x max x	Menentukan panjang maksimum dari area simulasi.
-y max y	Menentukan lebar maksimum dari area simulasi

Command line `setdest` yang sudah dilakukan *generate* akan 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. Berikut ini adalah contoh *Command line* yang dilakukan dari perintah tersebut:

```
./setdest -v 2 -n 8-s 1 -m 10 -M 50 -t 30 -
P 1 -p 1 -x 500 -y 500 > setdest_coba.tcl
```

2.6 *File Traffic Connection Pattern*

Ada dua jenis *Random Traffic Connection*, yakni TCP dan CBR. Keduanya dapat dibuat menggunakan skrip *traffic scenario generator*. Skrip ini nantinya akan digunakan untuuk membantu dalam melakukan *generate* beban *traffic*. Skrip yang dimaksud di sini adalah *file* "cbrgen.tcl". Program "cbrgen.tcl" ini akan digunakan sesuai dengan *Command line* yang ada di bawah ini dan penjelasan dari *Command line* tersebut akan ditunjukkan pada **Tabel 2.3**.

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

Berikut ini merupakan salah satu contoh dari pengaplikasian *commnd line* program “cbrgen.tcl”.

```
ns cbrgen.tcl -type cbr -nn 10 -seed 1 -mc 1 -
rate 5.0 > cbr10.tcl
```

Tabel 2.3 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.

BAB III
ANALISIS DAN DESAIN SISTEM

Analisis dan desain sistem merupakan bagian penting dari pengembangan sistem secara teknis sehingga bab ini secara khusus menjelaskan tentang analisis dan sistem yang dibuat dalam Tugas Akhir. Analisis dan desain sistem tersebut meliputi lingkungan implementasi protokol, implementasi pada skenario, implementasi pada simulator jaringan NS-2, implementasi modifikasi, dan terakhir implementasi metrik analisis.

3.1 Deskripsi Umum Sistem

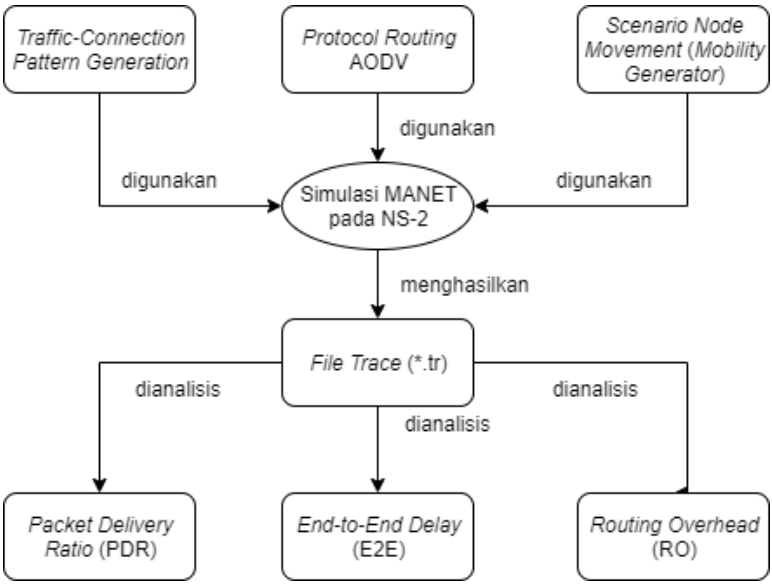
Pada Tugas Akhir ini akan dilakukan analisis mengenai hasil dari pengembangan *routing protocol* bernama *Ad-hoc On demand Distance Vector* (AODV) dengan menggunakan konsep *Power and Delay Aware* pada lingkungan dengan energi heterogen pada *Mobile Ad-hoc Network* (MANET). Untuk melakukan pembuatan skenario, penulis menggunakan *mobile generator* yang bersifat *random way point* dan keberadaannya telah ada pada *Network Simulator*-2 (NS-2), yaitu dengan cara melakukan *generate file traffic connection pattern*. Diagram perancangan simulasi dengan *routing protocol* AODV murni dan *routing protocol* AODV yang telah dimodifikasi dapat dilihat pada **Gambar 3.1** dan **Gambar 3.2**..

Dalam penelitian ini, penulis akan melakukan analisa terhadap hasil dari pengembangan *routing protocol* AODV dengan menggunakan konsep *Power Delay and Aware* di lingkungan dengan energi yang heterogen pada sistem operasi Linux dengan varian Ubuntu 16.04 LTS.

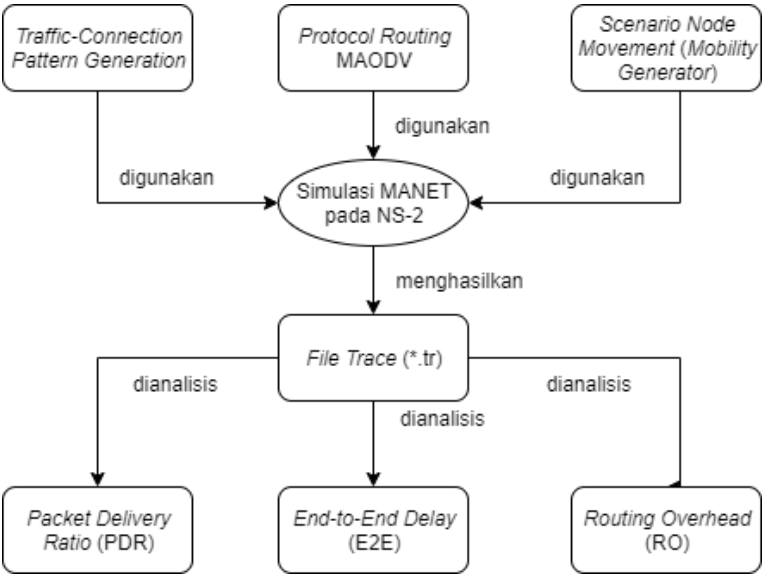
Pada Tugas Akhir ini, jumlah *node* dibuat bervariasi pada masing-masing skenario yang ada, yakni mulai dari 50 *node*, 75 *node*, dan 100 *node*. *Output* dari dari masing-masing skenario akan menghasilkan sebuah *trace file* yang nantinya akan dilakukan analisis mengenai perhitungan metrik *Packet Delivery Ratio* (PDR), *End-to-End Delay* (E2E), dan *Routing Overhead* (RO).

Dari hasil tersebut akan dianalisis performa yang dihasilkan oleh lingkungan dengan energi yang homogen maupun heterogen pada jaringan MANET.

Berikut ini adalah diagram rancangan simulasi *routing protocol Ad-hoc On demand Distance Vector* (AODV) asli dan yang sudah mengalami modifikasi.



Gambar 3.1 Diagram perancangan simulasi AODV asli atau belum mengalami modifikasi



Gambar 3.2 Diagram perancangan simulasi AODV yang telah mengalami modifikasi

Pada **Tabel 3.1** yang ada di bawah ini merupakan daftar dari beberapa istilah yang sering digunakan oleh penulis pada penulisan buku Tugas Akhir ini.

Tabel 3.1 Daftar Istilah

No.	Istilah	Penjelasan
1	MANET	Singkatan dari *Mobile Ad-hoc Network*. Merupakan jenis dari jaringan yang digunakan pada Tugas Akhir ini.
2	AODV	Singkatan dari *Ad-hoc On-demand Distance Vector*. Protokol yang digunakan pada Tugas Akhir ini.
3	PDR	*Packet Delivery Ratio*. Salah satu metrik analisis yang diukur. Berupa

No.	Istilah	Penjelasan
		rasio jumlah pengiriman paket yang terkirim.
4	E2E	*End-to-End Delay*. Jeda waktu yang diukur saat paket terkirim.
5	RO	*Routing Overhead*. Jumlah *control packet* yang terkirim
6	RREQ	*Route Request*. Paket *request* pada AODV yang dikirim untuk mendapatkan rute
7	RREP	*Route Reply*. Paket *reply* pada AODV yang dikirim ke *node* sumber melalui rute yang sudah terbuat.

3.2 Perancangan Skenario

Dalam melakukan perancangan skenario uji coba yang digunakan dalam Tugas Akhir ini, penulis menggunakan menggunakan *Mobility Generation.* Setelah skenario selesai dibuat, koneksi dari skenario yang sudah ada dengan menggunakan *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 jumlah *node*. Skenario pertama sejumlah 50 *node*, skenario kedua sejumlah 75 *node* dan skenario terakhir berjumlah 100 *node*. Sedangkan untuk koneksinya menggunakan dua *node* untuk menentukan *node* pengirim dan *node* penerima paket.

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

Pada tahap ini akan dibuat rancangan skenario dengan melakukan *generate file node movement* yang sudah disediakan oleh NS-2 atau biasa kita kenal dengan *tools* bernama ‘setdest’. Fungsi dari *tools* tersebut adalah berguna untuk digunakan dalam

file .tcl selama simulasi pada NS-2 berlangsung sebagai bentuk pergerakan *node* yang berpindah-pindah. Pada **Error! Reference source not found.** yang ada di bawah ini akan memberikan penjelasan mengenai skenario *node movement*.

Tabel 3.2 Penjelasan Skenario *node movement*

No.	Parameter	Spesifikasi
1	Jumlah *Node*	Skenario 1 = 50 *node* Skenario 2 = 75 *node* Skenario 3 = 100 *node*
2	Waktu Simulasi	200 detik
3	Area	500m x 500m
4	Kecepatan Maksimal	50m/s
5	Sumber *Traffic*	*Constant Bit Rate* (CBR)
6	Waktu Jeda (Dalam Detik)	1
7	Ukuran Paket	512 bytes
8	*Rate* Paket	1 paket per detik
9	Jumlah maksimal koneksi	1 (statis)
10	Model mobilitas yang digunakan	*random way point*
11	Model propagasi yang digunakan	*Two ray ground*

Dari **Tabel 3.2** di atas, akan didapatkan bahwa jika pada skenario *node movement* yang dijalankan terdapat beberapa hal seperti di bawah ini:

1. Terdapat tiga buah skenario, yakni Skenario 1 dengan jumlah *node* yang disimulasikan sebanyak 50 *node*, Skenario 2 dengan jumlah *node* yang disimulasikan sebanyak 75 *node*, dan terakhir pada Skenario 3 terdapat 100 *node* yang disimulasikan.
2. Waktu simulasi pada masing-masing skenario tersebut adalah selama 200 detik.

3. Area dari masing-masing simulasi di atas adalah seluas 500m x 500m.
4. Kecepatan maksimal di masing-masing skenario tersebut adalah 50 m/s.
5. Ukuran paket sebesar 512 bytes.
6. Model mobilitas yang dijalankan pada skenario tersebut adalah berjenis *random way point* dengan model propagasi yang digunakan berjenis *two ray ground.*

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 dala m skenario diatas selama melakukan simulasi pada NS-2. Pada **Tabel 3.3** di bawah ini akan memberikan penjelasan mengenai skenario *node movement.*

Tabel 3.3 Penjelasan *Traffic-connection pattern*

No.	Parameter	Spesifikasi
1	-type cbr|tcp	*Constant Bit Rate* (CBR)
2	-nn *nodes*	10
3	-s *seed*	1.0
4	-mc *connection*	1
5	-rate *rate*	5.0

3.3 Perancangan Simulasi pada NS-2

Simulasi pada NS-2 dilakukan dengan menggabungkan *file* skenario yang telah dibuat menggunakan skenario mobilitas dan *file* skrip dengan ekstensi .tcl yang berisikan konfigurasi dari lingkungan simulasi. Konfigurasi lingkungan simulasi bisa dilihat pada **Tabel 3.4** yang ada di bawah ini.

Tabel 3.4 Penjelasan simulasi pada NS-2

No	Parameter	Spesifikasi
1	*Network Simulator*	NS-2 versi 2.35
2	*Routing Protocol*	AODV
3	Waktu Simulasi	200 detik
4	Area Simulasi	500m x 500m
5	Banyak *node*	50, 75, 100
6	Agen Pengirim	*Constant Bit Rate* (CBR)
8	*Source/Destination*	Statis
9	*Packet Rate*	512 bytes
10	Ukuran Paket	32
11	Protokol MAC	IEEE 802.11
12	Tipe Kanal	*Wireless Channel*

Pada tabel di atas, tertera konfigurasi dari lingkungan simulasi yang digunakan pada Tugas Akhir ini. Beberapa hal yang digunakan meliputi:

1. Jenis dari simulator jaringan atau *network simulator* yang digunakan adalah *Network Simulator*-2 versi 2.35.
2. Jenis dari *routing protocol* yang digunakan adalah *routing protocol* dengan nama *Ad-hoc On demand Distance Vector* (AODV).
3. Waktu simulasi yang digunakan untuk menjalankan simulasi adalah selama 200 detik.
4. Area simulasi yang digunakan pada lingkungan simulasi ini adalah seluas 500m x 500m.
5. Banyak *node* di sini ada tiga jenis, yakni 50 *node*, 75 *node*, dan terakhir 100 *node*.
6. Jenis protokol yang digunakan pada lingkungan simulasi tersebut adalah MAC yang menggunakan teknologi bernama IEEE 802.11.
7. Untuk kanal yang digunakan pada lingkungan simulasi tersebut berjenis *wireless*, sebab jaringan MANET sendiri termasuk ke dalam jaringan nirkabel atau *wireless* dalam menghubungkan antar *node* yang ada.

3.4 Perancangan Metrik Analisis

Berikut ini merupakan parameter-parameter yang akan dianalisis pada Tugas Akhir ini:

3.4.1 *Packet Delivery Ratio* (PDR)

Packet delivery ratio merupakan perbandingan dari jumlah paket data yang dikirim dengan paket data yang diterima. PDR dapat menunjukkan keberhasilan paket yang dikirimkan. Semakin tinggi PDR artinya semakin berhasil pengiriman paket yang dilakukan, semakin baik pula performanya. PDR dapat dihitung dengan persamaan 3.1.

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

Keterangan:

PDR = *Packet Delivery Ratio*

received = Jumlah paket data yang diterima

sent = Jumlah paket data yang dikirimkan

3.4.2 *End-to-End Delay* (E2E)

End-to-End Delay merupakan *delay* antara waktu paket data diterima dan waktu paket dikirimkan dalam satuan detik. *Delay* tiap paket didapat dari rentang waktu antara *node* asal saat mengirimkan paket dan *node* tujuan menerima paket. *Delay* tiap paket tersebut semua dijumlahkan dan dibagi dengan jumlah paket yang berhasil diterima, maka akan didapatkan E2E, yang dapat dihitung dengan persamaan 3.2.

$$E2E = \sum_{i<sent}^{i=0} \frac{t^{received(i)} - t^{sent(i)}}{sent} \qquad (3.2)$$

Keterangan:

E2E = *End-to-End Delay*

$t^{received(i)}$ = waktu penerimaan paket dengan urutan / id ke-i

$t^{sent(i)}$ = waktu pengiriman paket dengan urutan / id ke-i

sent = Jumlah paket yang berhasil diterima

3.4.3 *Routing Overhead* (RO)

Routing Overhead adalah jumlah paket kontrol *routing* yang ditransmisikan per data paket ke *node* tujuan selama proses simulasi terjadi. *Routing Overhead* didapatkan dengan menjumlahkan semua paket kontrol *routing* yang ditransmisikan, baik itu paket *route request* (RREQ), *route reply* (RREP), maupun *route error* (RERR). Perhitungan *Routing Overhead* dapat dilihat dengan persamaan 3.3.

$$RO = \sum_{m=1}^{sentnum} packet\ sent \qquad (3.3)$$

(Halaman ini sengaja dikosongkan)

BAB IV
IMPLEMENTASI

Pada bab ini akan membahas mengenai implementasi Tugas Akhir berdasarkan rancangan perangkat lunak yang meliputi lingkungan pembangunan perangkat lunak, implementasi skenario, implementasi simulasi NS-2, dan implementasi metrik analisis yang telah dilakukan pada bab sebelumnya.

4.1 Lingkungan Implementasi Protokol

Pembahasan yang terdapat pada sub bab ini adalah mengenai lingkungan implementasi protokol yang digunakan pada Tugas Akhir ini. Pada **Tabel 4.1** yang ada di bawah ini merupakan tabel yang berisi mengenai spesifikasi lingkungan implementasi.

Tabel 4.1 Spesifikasi lingkungan implementasi

Perangkat Keras	• Laptop Lenovo Ideapad 320s 14IKB • Intel(R) Core (TM) i5-8250U CPU @ 1.60GHz (8 CPUs), ~ 1.8GHz • HDD 1TB (digunakan untuk *Install* Ubuntu sebesar 30GB) • RAM 4GB
Perangkat Lunak	• Sistem Operasi Ubuntu 16.04 LTS • Network Simulator 2 versi 2.35

Lingkungan implementasi yang digunakan pada Tugas Akhir ini yakni dengan menggunakan laptop bermerk Lenovo Ideapad 320s 14IKB dengan spesifikasi prosesor menggunakan Intel(R) Core (TM) i5-8250U CPU @ 1.60GHz (8 CPUs), ~ 1.8GHz. Kemudian untuk alokasi *harddisk* yang digunakan untuk melakukan instalasi sistem opersi Ubuntu adalah sebesar 30GB dari total 1TB yang tersedia. Sedangkan untuk RAM dari laptop tersebut adalah sebesar 4GB. Untuk perangkat lunak yang

digunakan adalah sistem operasi Ubuntu versi 16.04 dan simulator jaringan bernama *Network Simulator*-2 versi 2.35.

4.2 Implementasi Skenario

Pada tahap ini, penulis akan melakukan proses implementasi skenario mobilitas jaringan MANET yang dimulai dengan melakukan pembuatan skenario dengan bantuan *Mobility Generation* untuk menempatkan dan menentukan *node-node* yang akan disimulasikan. Setelah skenario dibuat, langkah selanjutnya adalah membuat jalur-jalur yang akan membuat antar *node* dengan *node* lainnya saling berhubungan dengan bantuan *traffic connection pattern*. Berikut ini adalah bentuk pengimplementasian tersebut.

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

Dalam pelaksanan implementasi yang dilakukan pada saat pembuatan skenario dengan *mobility generation* menggunakan *tools generate default* yang sudah disediakan oleh simulator jaringan *Network Simulator*-2 (NS-2), ketika proses instalasi sebelumnya dilakukan, 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* sehinnga penempatan *node-node* yang ada akan bersifat acak. Format *Command line* yang digunakan untuk menghasilkan gerakan acak pada *node* adalah sebagai berikut:

```
./setdest -v [2] -n <nodes> -s <speed type> -m
<min speed> -M <max speed> -t <simulation time> -P
<pause type> -p <pause time> -x <max X> -y <max Y>
```

Beberapa ketentuan yang diujicobakan pada skenario ini adalah versi dari ‘setdest’ *simulator* yaitu versi 2, yakni versi University of Michigan. Selanjutnya untuk jumlah *node* masing-

masing skenario terdapat perbedaan, untuk skenario 1 sejumlah 50 *node*, skenario 2 sejumlah 75 *node*, dan skenario 3 sejumlah 100 *node* dengan waktu simulasi sebesar 30 detik. Kemudian file mobilitas yang dihasilkan disimpan dalam direktori "~ns/indep-utils/cmu-scen-gen/setdest/".

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 *file* tersebut akan digunakan sebagai sarana untuk membuat jaringan atau hubungan antar *node-node* yang telah dibuat pada skenario sebelumnya. Ketika kita akan menggunakan *file* "cbrgen.tcl" ini kita harus menentukan tipe koneksi yang digunakan, antara CBR ataukah TCP. Jumlah dari *node* yang ada, koneksi maksimal yang diinginkan, dan nilai dari *random seed*. Format dari *Command line* yang digunakan untuk menghasilkan gerakan acak pada *node* tersebut adalah sebagai berikut ini:

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

Di bawah ini merupakan bentuk implementasi cbrgen.tcl untuk membuat file koneksi CBR diantara 10 *node* memiliki maksimal 1 koneksi dengan nilai seed 1.0 dan jumlah paket per detik sebanyak 1.0 yang disimpan dalam cbr10.tcl yang nantinya akan digunakan pada saat simulasi NS-2 dijalankan.

```
ns cbrgen.tcl -type cbr -nn 10 -seed 1 -mc 1 -
rate 1.0 > cbr.txt
```

4.3 Implementasi Modifikasi

Pada tahap ini akan dilakukan implementasi dari modifikasi yang dilakukan pada AODV dengan melakukan implementasi dari *routing protocol* baru, yakni *routing protocol* bernama *Power and Delay-aware Multi-path Routing Protocol* (PDMRP). Tujuan dari penggunaan protokol tersebut adalah untuk memilih multi-jalur dengan masa hidup terpanjang di jaringan tanpa penurunan kinerja dalam hal waktu tunda. Untuk mencapai tujuan ini, pertama-tama kita perlu menghitung Biaya (disebut C) dari setiap rute dengan menggunakan persamaan berikut:

$$C = \frac{ML}{NH}$$

Berikut ini adalah beberapa modifikasi yang dilakukan di sini antara lain pada:

a) Saat *node* perantara menerima lebih dari satu paket *Routing Request* (RREQ) dari sumber yang sama, nantinya *node* ini akan melakukan perhitungan terhadap setiap paket RREQ nilai C menggunakan persamaan di atas. Nantinya akan melakukan penyiaran ulang paket (*re-broadcast*) nilai C terbesar dan menjatuhkan paket lainnya.

Ubah file pada aodv.cc pada folder NS-2 yg akan dimodifikasi

```
1. // modifikasi
2.     if (id_lookup(rq->rq_src, rq->rq_bcast_id)) {
3.           double prev_cost= rq->rq_cost > 0 ? rq->rq_cost
   : 0;
4.         if (cost < prev_cost){
5.         Packet::free(p);
6.         return;
7.         }
```

```
8.
9.   }
10. else {
11.      rq->rq_cost= cost;
12.     rq->rq_min_life= iEnergy;
13. }
```

Gambar 4.1 Potongan skrip implementasi di atas

b) Setelah paket RREQ pertama diterima, *node* tujuan akan menunggu penerimaan paket RREQ lainnya dari jalur yang berbeda untuk waktu yang telah ditentukan. Kemudian, ia akan mengirimkan kembali RREP setiap paket RREQ sesuai dengan *node* sumber. Ini akan membantu *node* asal dalam menemukan jalur terpendek dan yang paling stabil serta menjaga jalur alternatif yang dapat digunakan ketika jalur primer tidak menjadi fungsional (misalnya ketika kegagalan tautan atau simpul terjadi).

Ubah file pada aodv.cc pada folder NS-2 yg akan dimodifikasi

```
1.  // modification
2.  last_rreq_on_dest=  CURRENT_TIME;
3.  bool is_it_first= true;;
4.
5.  // wait t second
6.  // sleep(t);e
7.  if (CURRENT_TIME != last_rreq_on_dest){
8.          is_it_first=  false;
9.  }
10.
11.  // select greatest cost
12.  if (is_it_first == false) {
13.         // packet greather than previous
```

```
14.         if (rq->rq_cost < prev_cost_on_dest) {
15.         Packet::free(p);
16.         return;
17.         }
18. }
19. prev_cost_on_dest= rq->rq_cost;
```

4.4 Implementasi Simulasi pada NS-2

Implementasi simulasi yang terjadi pada simulator jaringan NS-2 dilakukan dengan cara melakukan pendeskripsian lingkungan simulasi pada sebuah *file* yang memiliki ekstensi .tcl, dan .txt. *File-file* ini berisi konfigurasi dari setiap *node* dan proses yang dilakukan selama simulasi berjalan.

Pada kode sumber 4.1 menunjukan skrip konfigurasi awal dari parameter-paremeter yang diberikan untuk menjalakan simulasi jaringan MANET pada simulator jaringan NS-2. Isi serta penjelasan dari parameter-parameter tersebut ada pada **Tabel 4.2** sebagaimana yang ada di bawah ini.

Tabel 4.2 Penjelasan simulasi NS-2

No.	**Parameter**	**Spesifikasi**
1	Channel/	Wireless Channel
2	Propagation/	TwoRayGround
3	Phy/ WirelessPhy	WirelessPhy
4	Mac/ 802.11	802.11
5	Queue/Droptail/PriQueue	PriQueue
6	Antenna/OmniAntena	OmniAntena
7	Nilai X	500
8	Nilai Y	500
9	Nilai seed	1.0
10	Routing Protocol	AODV
11	File traffic connection	“cbr.txt”
12	File *node* movement	“scenario.txt”
13	Simulation Time	200 detik

Pada tabel di atas, bisa kita dapatkan bahwa:

1. Jenis *channel* yang dipakai adalah *wireless* sebab MANET termasuk ke dalam kategori jaringan nirkabel.
2. Propagasi yang digunakan adalah berjenis *TwoRayGround*.
3. Nilai sumbu X adalah sebesar 500
4. Nilai sumbu Y adalah sebesar 500.
5. Jenis *routing protocol* yang dipakai adalah *routing protocol* bernama AODV.
6. Jenis antena yang digunakan adalah OmniAntena.
7. Waktu simulasi adalah selama 200 detik.

Berikut ini merupakan skrip megenai konfigurasi parameter yang digunakan pada simulator jaringan *Network Simulator* (NS-2).

```
1. set  val(chan)       Channel/WirelessChannel;  # channel type
2. set val(prop)       Propagation/TwoRayGround; # radio-propagation model
3. set  val(netif)      Phy/WirelessPhy;  # network interface type
4. set val(mac)         Mac/802_11; # MAC type
5. set  val(ifq)        Queue/DropTail/PriQueue;  # interface queue type
6. set val(ll)            LL; # link layer type
7. set val(ant)           Antenna/OmniAntenna;    # antenna model
8. set opt(x)             500; # X dimension of the topography
9. set opt(y)             500; # Y dimension of the topography
10. set val(ifqlen)       50; # max packet in ifq
11. set val(nn)           75; # how many nodes are simulated
12. set val(seed)          1.0;
13. set val(adhocRouting)   AODV; # routing protocol
```

```
14. set val(stop)           200; # simulation time
15. set val(cp)  "cbr.txt"; #<-- traffic file
16. set val(sc)  "scenario.txt"; #<-- mobility file
```

Kode Sumber 4.1 Konfigurasi parameter pada NS-2

Di bawah ini adalah skrip mengenai inisiasi awal untuk energi yang nantinya akan digunakan pada *routing protocol* AODV.

```
1. set val(energy_mod)   EnergyModel;   #   energy
   model
2. set val(energy_init)  80;# init val for energy
3. set val(tx_power)     1.65;# energy consume for
   transmiting packet
4. set val(rx_power)     1.4;# energy consume for
   receiving packet
5. set val(idle_power)   0.5;# energy consume for
   idle
6. set val(sleep_power)  0.3;# energy consume for
   sleep mode
```

Kode Sumber 4.2 Inisiasi awal energi yang digunakan

Di bawah ini merupakan skrip yang digunakan untuk mengatur energi di masing-masing *node*. Pada *node source* dan *destination* energinya diatur dengan angka yang besar, sedangkan energi pada *node intermediate* diatur secara random.

```
1. #  Create the specified number of nodes
   [$val(nn)] and "attach" them
2. #  to the channel.
3. #  plus random energy
4. #  src : http://slogix.in/how-to-find-
   residual-energy-of-the-nodes-in-ns2
5. for {set i 0} {$i < $val(nn)} {incr i} {
6.     if {$i < [expr $val(nn) - 2]} {
```

```
7.          set energy($i) [expr int(40 +
    rand()*60)];
8.      $ns_ node-config -adhocRouting
    $val(adhocRouting) \
9.                  -llType $val(ll) \
10.                 -macType $val(mac) \
11.                 -ifqType $val(ifq) \
12.                 -ifqLen $val(ifqlen) \
13.                 -antType $val(ant) \
14.                 -propType $val(prop) \
15.                 -phyType $val(netif) \
16.                 -channelType $val(chan) \
17.                 -energyModel $val(energy_mod)
18.                 -initialEnergy $energy($i) \
19.                 -txPower $val(tx_power) \
20.                 -rxPower $val(rx_power) \
21.                 -idlePower $val(idle_power) \
22.                 -sleepPower $val(sleep_power)
    \
23.                 -topoInstance $topo \
24.                 -agentTrace ON \
25.                 -routerTrace ON \
26.                 -macTrace ON \
27.                 -movementTrace ON
28.     set node_($i) [$ns_ node]
29.         $node_($i) random-motion 0;# disable
    random motion
30.         $node_($i) color blue; #whatever you
    fill as long as color
31.     set E($i) $energy($i)
32.     } else {
33.         $ns_ node-config -adhocRouting
    $val(adhocRouting)\
34.                 -llType $val(ll) \
35.                 -macType $val(mac) \
36.                 -ifqType $val(ifq) \
37.                 -ifqLen $val(ifqlen) \
38.                 -antType $val(ant) \
```

```
39.                     -propType $val(prop) \
40.                     -phyType $val(netif) \
41.                     -channelType $val(chan) \
42.                     -energyModel $val(energy_mod)
   \
43.                     -initialEnergy
   $val(energy_init)\
44.                     -txPower $val(tx_power) \
45.                     -rxPower $val(rx_power) \
46.                     -idlePower $val(idle_power) \
47.                     -sleepPower $val(sleep_power)
   \
48.                     -topoInstance $topo \
49.                     -agentTrace ON \
50.                     -routerTrace ON \
51.                     -macTrace ON \
52.                     -movementTrace ON
53.     set node_($i) [$ns_ node]
54.     $node_($i) random-motion 0 ;# disable random
   motion
55.     $node_($i) color black; #whatever you fill
   as long as color
56.      }
57. }
```

Kode Sumber 4.3 Pengaturan jumlah energi yang akan digunakan

Skrip pada **Kode Sumber 4.4** merupakan bagian akhir dari keseluruhan skrip yang digunakan untuk menginisiasi penempatan awal *node-node* 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 `result_mod.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 200 seperti yang telah di konfigurasi sebelumnya.

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

# Define traffic model
puts "Loading scenario file..."
source $val(sc)

# Define node initial position in nam

for {set i 0} {$i < $val(nn)} {incr i} {

    # 20 defines the node size in nam, must adjust it according to your scenario
    # The function must be called after mobility model is defined

    $ns_ initial_node_pos $node_($i) 100
}

# 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)
$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(adhocRouting)"
#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..."
```

```
32.$ns_ run
```

Kode Sumber 4.4 Menginisiasi penempatan awal *node*

4.5 Implementasi Metrik Analisis

Simulasi yang telah dijalankan oleh NS-2 menghasilkan sebuah *trace file* yang berisikan data mengenai apa saja yang terjadi selama simulasi dalam bentuk *file* berekstensi .tr. Dari data *trace file* tersebut, penulis akan melakukan analisis performa *routing protocol* dengan mengukur beberapa metrik, yakni *Packet Delivery Ratio* (PDR), *End-to-End Delay* (E2E), dan *Routing Overhead* (RO).

4.5.1 Implementasi *Packet Delivery Ratio* (PDR)

Cara untuk mendapatkan *Packet Delivery Ratio* (PDR) adalah dengan melakukan perbandingan antara pengiriman dan penerimaan data yang dikirimkan oleh agen. Pada Tugas Akhir ini, penulis menggunakan agen dengan pengiriman data CBR. Kemudian, pada persamaan 2 telah dijelaskan mengenai rumus untuk menghitung *packet delivery ratio*. Skrip yang ada awk akan digunakan untuk melakukan perhitungan *Packet Delivery Ratio* (PDR). Berdasarkan kedua informasi tersebut dapat dilihat pada lampiran Tugas Akhir ini. Di bawah ini merupakan contoh dari perintah yang digunakan untuk melakukan proses pengeksekusian skrip awk dalam rangka mencari hasil dari PDR.

```
awk –f pdr.awk result_mod.tr
```

4.5.2 Implementasi *End-to-End Delay (E2E)*

Dalam menghitung hasil dari *End-to-End Delay* (E2E), penulis menggunakan dasar yang telah ada pada persamaan 3 dan telah dijelaskan pada sub bagian 3.4.2. Skrip awk yang digunakan untuk menghitung hasil dari *End-to-End Delay* (E2E) ini

berdasarkan kedua informasi tersebut bisa dilihat pada bagian lampiran Tugas Akhir ini. Di bawah ini merupakan contoh perintah yang digunakan untuk melakukan proses pengeksekusian skrip awk dalam rangka menganalisis *trace file*.

```
awk –f e2e.awk tracefile.tr.
```

4.5.3 Implementasi *Routing Overhead* (RO)

Sebagaimana telah diketahui, *Routing Overhead* (RO) adalah jumlah paket kontrol *routing* yang ditransmisikan per data paket ke *node* tujuan selama proses simulasi terjadi. Untuk melakukan penghitungan *Routing Overhead* (RO), penulis menggunakan dasar persamaan 4 dan hal ini telah tertera pada bagian 3.4.3. Skrip awk yang digunakan untuk melakukan perhitungan RO ini berdasarkan kedua informasi tersebut bisa dilihat di bagian lampiran dari Tugas Akhir ini. Berikut ini adalah contoh dari cara untuk mengeksekusi perintah yang ada pada skrip awk yang digunakan untuk melakukan analisis pada *trace file.*

```
awk –f ro.awk result.tr
```

(Halaman ini sengaja dikosongkan)

BAB V
UJI COBA DAN EVALUASI

Pada bab ini akan dilakukan tahap uji coba dan evaluasi sesuai dengan rancangan dan implementasi. Dari hasil yang didapatkan setelah melakukan uji coba, akan dilakukan evaluasi sehingga dapat ditarik kesimpulan pada bab selanjutnya. Hal-hal yang akan dibahas pada bab ini meliputi lingkungan uji coba, hasil uji coba dan analisis pada *Packet Delivery Ratio* (PDR), *End-to-End Delay* (E2E), dan *Routing Overhead* (RO)

5.1 Lingkungan Uji Coba

Uji coba dilakukan pada perangkat dengan spesifikasi seperti yang tertera pada Tabel **5.1**.

Tabel 5.1 Spesifikasi Perangkat yang Digunakan

Komponen	**Spesifikasi**
Prosesor	Intel(R) Core (TM) i5-8250U CPU @ 1.60GHz (8 CPUs), ~ 1.8GHz
Sistem Operasi	Ubuntu 16.04 LTS
Linux Kernel	Linux kernel 4.4
Memori	4.0 GB
Penyimpanan	30 GB (dari total 1TB)
Versi NS-2	NS-2, 2.35

Pada **Tabel 5.1** di atas, spesifikasi dari perangkat yang digunakan dalam pengerjaan Tugas Akhir ini dibedakan ke dalam dua bagian. Pertama adalah perangkat keras atau *hardware*. Ini meliputi jenis Prosesor yang dipakai, yakni Intel(R) Core (TM) i5-8250U CPU @ 1.60GHz (8 CPUs), ~ 1.8GHz. Selanjutnya ada memori atau RAM dan penyimpanan dengan masing-masing berkapasitas 4GB untuk RAM dan sebesar 30GB untuk penyimpanan yang digunakan untuk melakukan instalasi beberapa perangkat lunak yang diperlukan. Kedua adalah perangkat lunak

atau *software*. Dalam pengerjaan Tugas Akhir ii ada beberapa *software* yang sering digunakan, yaitu sistem operas Ubuntu 16.04 LTS dan simulator jaringan bernama *Network Simulator*-2 (NS-2) versi 2.35.

Sedangkan untuk parameter lingkungan uji coba yang digunakan pada NS-2 dapat dilihat pada **Tabel 5.2**. Agar pengujian bisa dilakukan, langkah yang harus dilakukan adalah dengan menjalankan skenario yang disimulasikan pada NS-2. Setelah simulasi dilakukan, dari simulasi tersebut akan dihasilkan sebuah *trace file* dengan ekstensi .tr yang akan nantinya akan dianalisis dengan bantuan skrip awk untuk mendapatkan *Packet Delivery Ratio* (PDR), *End-to-End Delay* (E2E), dan *Routing Overhead* (RO) menggunakan kode yang terdapat pada lampiran.

Tabel 5.2 Lingkungan Uji Coba

No.	**Parameter**	**Spesifikasi**
1	Network simulator	NS-2 versi 2.35
2	Routing protocol	AODV
3	Waktu simulasi	200 detik
4	Area simulasi	500 m x 500 m
5	Jumlah *Node*	50, 75, 100
6	Radius transmisi	400m
7	Kecepatan maksimum	20 m/s
8	Protokol MAC	IEEE 802.11p
9	Model Propagasi	*Two-ray ground*

Pada tabel di atas, tertera konfigurasi dari lingkungan simulasi yang digunakan pada Tugas Akhir ini. Beberapa hal yang digunakan meliputi:

1. Jenis dari simulator jaringan atau *network simulator* yang digunakan adalah *Network Simulator*-2 versi 2.35.
2. Jenis dari *routing protocol* yang digunakan adalah *routing protocol* berjenis *Ad-hoc On demand Distance Vector* atau lebih dikenal dengan nama AODV.

3. Waktu simulasi yang digunakan untuk menjalankan simulasi adalah selama 200 detik.
4. Area simulasi yang digunakan pada lingkungan simulasi ini adalah seluas 500m x 500m.
5. Banyak *node* di sini ada tiga jenis, yakni 50 *node*, 75 *node*, dan terakhir 100 *node*.
6. Radius transmisi yang ada di sini adalah sebesar 400m.
7. Kecepatan maksimum adalah sebesar 20m/s.
8. Jenis protokol yang digunakan pada lingkungan simulasi tersebut adalah MAC yang menggunakan teknologi bernama IEEE 802.11p.
9. Untuk kanal yang digunakan pada lingkungan simulasi tersebut berjenis *wireless*, sebab jaringan MANET sendiri termasuk ke dalam jaringan nirkabel atau *wireless* dalam menghubungkan antar *node* yang ada.

5.2 Hasil Uji Coba dan Analisis

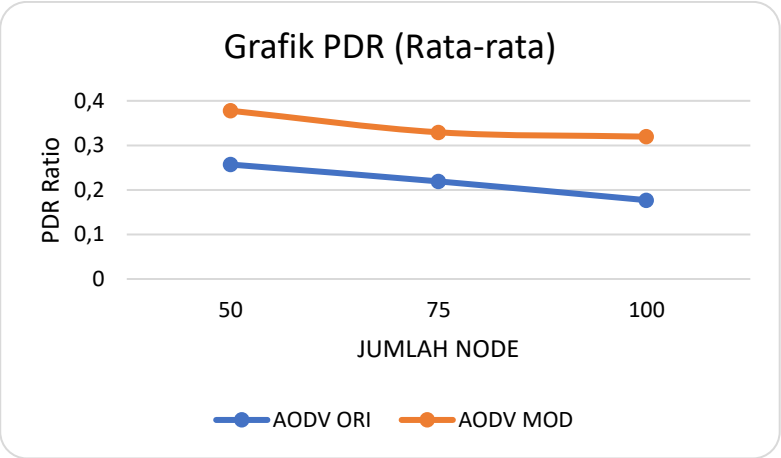
Pada tahap ini, penulis akan membahas mengenai hasil uji coba yang dibutuhkan dalam pembuatan Tugas Akhir ini dengan melalui analisis pada *Packet Delivery Ratio* (PDR), *Routing Overhead* (RO), dan terakhir pada *End-to-End Delay* (E2E).

5.2.1 Analisis *Packet Delivery Ratio* (PDR)

Setelah melakukan uji coba berulang kali dan melakukan analisis dari uji coba tersebut, pada akhirnya penulis mendapatkan nilai rataan pada PDR yang dihasilkan oleh lingkungan dengan energi yang homogen dan heterogen pada jaringan MANET seperti pada tabel 5.3 serta diilustrasikan dengan grafik pada gambar 5.2 untuk *node* yang berjumlah 50 *node*, gambar 5.3 untuk *node* yang berjumlah 75 *node*, dan gambar 5.4 untuk *node* yang berjumlah 100 *node*.

Tabel 5.3 Hasil PDR (Jumlah *node*)

Jumlah *Node*	**PDR (%)**	
	ORI	**MOD**
50	0,25701	0,3778
75	0,21913	0,32938
100	0,17698	0,31967



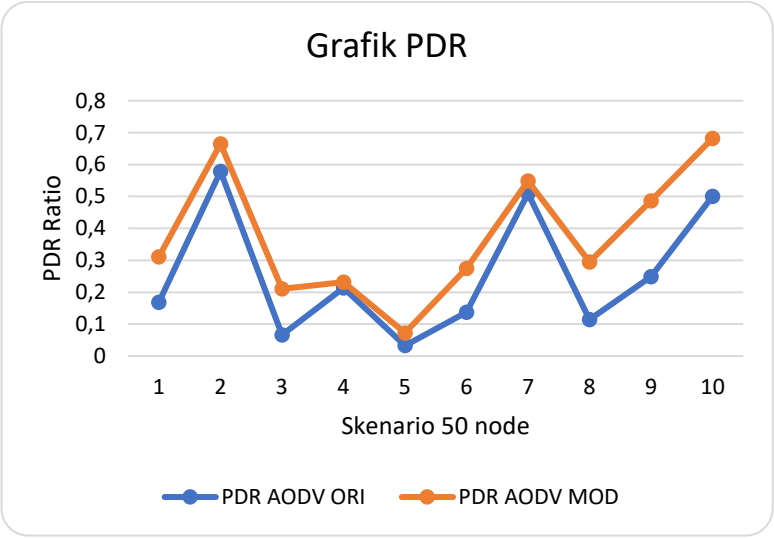
Gambar 5.1 Grafik nilai PDR (rata-rata)

Pada **Tabel 5.3** dan **Gambar 5.1** merupakan performa *Packet Delivery Ratio* (PDR) yang dihasilkan oleh lingkungan dengan energi yang heterogen pada jaringan MANET dengan menggunakan skenario *two Ray ground* (*mobility generation*) yang bersifat *Random Way Point*.

Dari sana dapat dilihat bahwa nilai rata-rata dari PDR yang dihasilkan oleh lingkungan dengan energi yang homogen, dalam hal ini adalah yang masih ori atau belum mengalami modifikasi menunjukan adanya penurunan nilai. Dapat dilihat bahwa nilai rata-rata pada Skenario 50 *node* adalah 0,2570% kemudian pada Skenario 75 *node* bernilai 0,2191% dan pada kecepatan Skenario 100 *node* memiliki nilai 0,1770%. Nilai rata-rata dari PDR yang dihasilkan oleh lingkungan dengan energi yang heterogen atau

yang telah mengalami modifikasi menunjukan nilai adanya penurunan nilai juga, sama seperti halnya pada energi homogen. Nilai rata-rata pada Skenario 50 *node* adalah 0,3778% sementara pada Skenario 75 *node* bernilai 0,3294% dan pada Skenario 100 *node* memiliki nilai 0,3197%. Hal ini menunjukan bahwa penggunaan energi yang heterogen pada *protocol routing* AODV tidak mengalami perubahan yang signifikan.

Berdasarkan hasil uji coba tersebut, bisa didapatkan bahwa nilai PDR yang dihasilkan pada lingkungan dengan energi yang heterogen lebih rendah dibandingkan jika dibandingkan dengan lingkungan dengan energi yang homogen. Pada lingkungan dengan energi yang heterogen, setiap *node* memiliki energi yang acak. Hal ini memungkinkan terjadinya banyak rute putus saat pengiriman paket data ataupun kegagalan pembentukan tabel *routing* yang dibuat oleh AODV sehingga menyebabkan paket data yang dikirimkan tidak sampai ke tujuan.

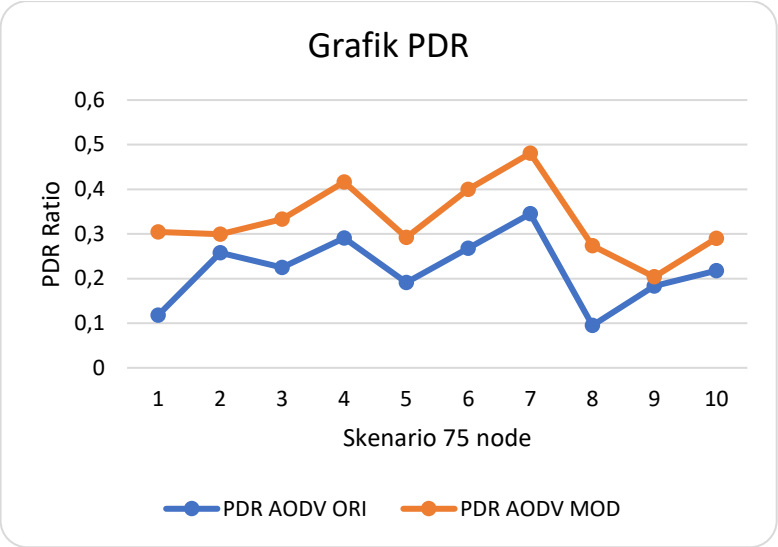


Gambar 5.2 Grafik nilai PDR (50 *node*)

Skenario	PDR	
	ORI	MOD
1	0,1689	0,3108
2	0,5786	0,6646
3	0,0667	0,2105
4	0,2138	0,2318
5	0,0327	0,0728
6	0,1373	0,2752
7	0,5096	0,5484
8	0,1141	0,2953
9	0,2484	0,4865
10	0,5	0,6821

Tabel 5.4 Tabel perbandingan nilai PDR ORI dan MOD (50 *node*)

Gambar 5.2 dan **Tabel 5.4** di atas menunjukkan bahwa *Packet Delivery Ratio* (PDR) yang ada pada lingkungan dengan *node* berjumlah 50 *node* mengalami kenaikan jika dibandingkan nilianya antara lingkungan yang homogen atau lingkungan yang tidak mengalami modifikasi (ori) dengan lingkungan yang heterogen atau lingkungan yang mengalami modifikasi (mod). Dari sana bisa dilihat bahwa semua skenario yang ada, yakni mulai Skenario 1 hingga Skenario ke-10 mengalami kenaikan. Kenaikan pada masing-masing skenario juga beragam. Ada yang mengalami kenaikan yang sangat sedikit seperti yang terjadi pada Skenario ke-4, yakni yang semula bernilai 0,2138, setelah dilakukan modifikasi mengalami kenaikan ke 0,2318. Kenaikan tertinggi di sini terjadi pada Skenario ke-3 yang semula bernilai 0,0667 menjadi 0,2105 setelah mengalami modifikasi.



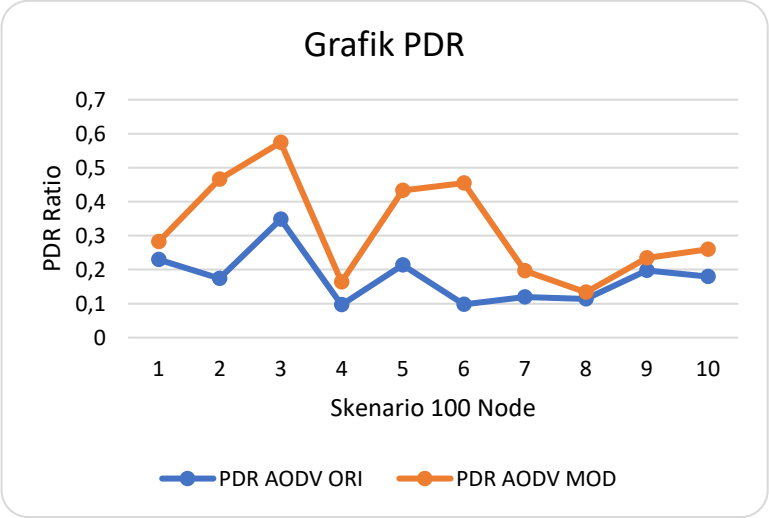
Gambar 5.3 Grafik nilai PDR (75 *node*)

Skenario	PDR	
	ORI	MOD
1	0,118	0,3043
2	0,2577	0,2994
3	0,225	0,3333
4	0,2909	0,4161
5	0,1911	0,2919
6	0,2675	0,4
7	0,3452	0,4808
8	0,0949	0,2739
9	0,1834	0,2038
10	0,2176	0,2903

Tabel 5.5 Tabel perbandingan nilai PDR ORI dan MOD (75 *node*)

Pada **Gambar 5.3** dan **Tabel 5.5** adalah *Packet Delivery Ratio* (PDR) yang terjadi pada lingkungan dengan *node* berjumlah

75 *node*. Dari kedua elemen tersebut bisa dilihat bahwa seluruh *Packet Delivery Ratio* (PDR) yang ada pada semua Skenario mengalami peningkatan nilai. Peningkatan nilai terbesar yang terjadi pada lingkungan ini adalah yang ada pada Skenario ke-8, yakni yang semula bernilai 0,0949 saat ada pada lingkungan yang homogen atau saat belum mengalami modifikasi (ori) kemudian berubah menjadi 0,2739 saat berada pada lingkungan yang heterogen atau saat mengalami modifikasi.



Gambar 5.4 Grafik nilai PDR (100 *node*)

Skenario	PDR	
	ORI	MOD
1	0,2293	0,2821
2	0,1739	0,4658
3	0,3481	0,5743
4	0,0964	0,1635
5	0,2134	0,4331
6	0,0976	0,4545

7	0,1195	0,1962
8	0,1138	0,1333
9	0,1977	0,2342
10	0,1801	0,2597

Tabel 5.6 Tabel perbandingan nilai PDR ORI dan MOD (100 *node*)

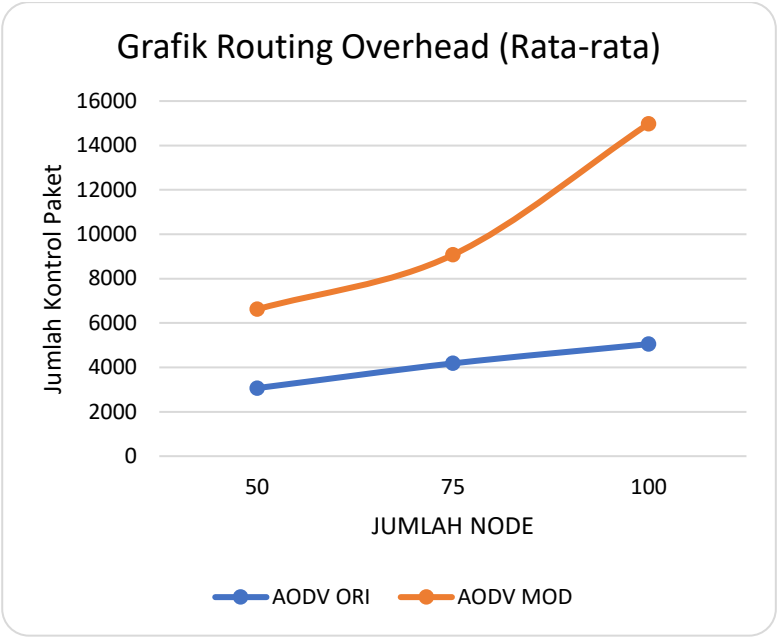
Gambar 5.4 dan **Tabel 5.6** adalah perbandingan antara *Packet Delivery Ratio* (PDR) yang terjadi pada lingkungan dengan *node* berjumlah 100 *node*. Dari kedua elemen tersebut bisa dilihat bahwa semua skenario mengalami kenaikan setelah mengalami modifikasi atau saat berada pada lingkungan yang heterogen jika dibandingan dengan sebelum dilakukan modifikasi atau saat berada pada lingkungan yang homogen. Kenaikan terbesar terjadi pada Skenario ke-6, yakni yang semula bernilai 0,0976 pada saat berada pada lingkungan yang belum mengalami modifikasi (ori) atau pada saat berada pada lingkungan yang homogen, kemudian berubah menjadi 0,4545 setelah mengalami modifikasi atau saat berada pada lingkungan yang heterogen.

5.2.2 Analisis *Routing Overhead* (RO)

Dari hasil analisis yang dilakukan setelah melakukan uji coba berulang kali, penulis mendapatkan nilai rataan pada *Routing Overhead* (RO) yang dihasilkan oleh lingkungan dengan energi yang homogen dan heterogen pada jaringan MANET, hasil dari uji coba tersebut bisa dilihat pada **Tabel 5.7.**

Tabel 5.7 Analisis *Routing Overhead*

Jumlah *Node*	**RO (paket)**	
	ORI	**MOD**
50	3067,3	6619,1
75	4180,7	9069,8
100	5049,1	14977,2

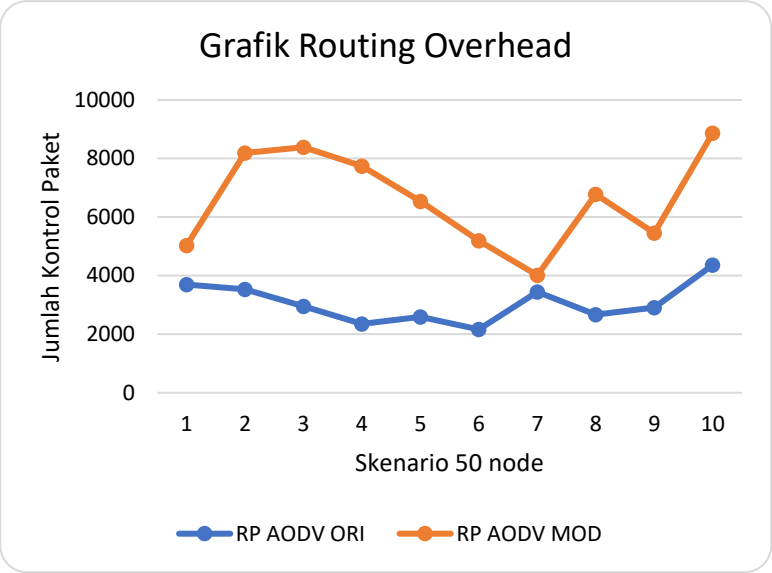


Gambar 5.5 Grafik RO Rata-rata

Pada **Tabel 5.7** serta **Gambar 5.5**, menunjukkan bahwa penggunaan skenario *node-movement* (mobility generation) yang bersifat *Random Way Point* berdasarkan jumlah *node* dengan energi yang heterogen atau yang telah mengalami modifikasi pada lingkungan MANET mengalami performa *Routing Overhead* yang lebih tinggi dibandingkan dengan *node* dengan energi yang homogen atau yang masih asli atau ori. Dapat dilihat bahwa hasil uji coba pada lingkungan dengan energi yang heterogen menunjukan nilai *Routing Overhead* yang kurang signifikan. Pada jumlah *node* 50 nilai RO yang dihasilkan adalah 6619,1 paket, pada jumlah *node* 75 nilai RO yang dihasilkan adalah 9069,8 paket, dan pada jumlah *node* 100 nilai RO yang dihasilkan adalah 14977,2 paket.

Faktor yang menyebabkan meningkatnya nilai rata-rata *Routing Overhead* pada lingkungan dengan energi yang heterogen

adalah terjadinya rute putus saat pengiriman paket, sehingga paket RERR bertambah untuk melakukan maintenance link komunikasi. Hal ini disebabkan karena energi yang acak pada setiap *node*.

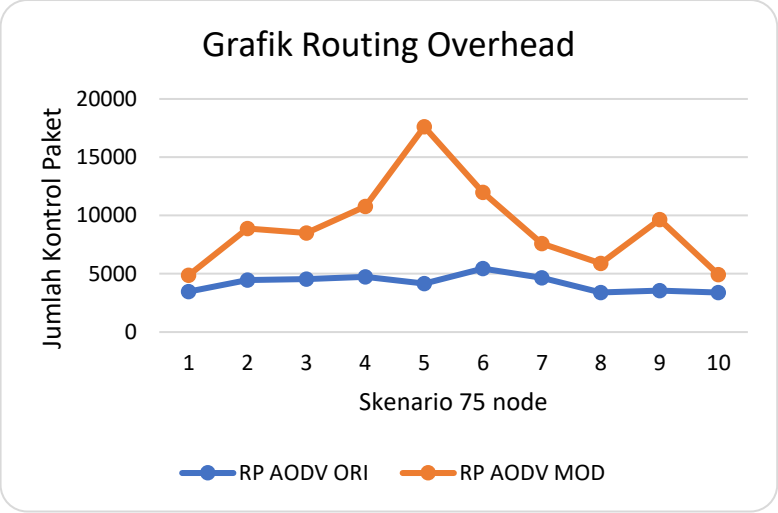


Gambar 5.6 Grafik nilai RO (50 *node*)

Skenario	RO	
	ORI	MOD
1	3700	5034
2	3533	8187
3	2948	8384
4	2349	7744
5	2595	6539
6	2162	5193
7	3441	4012
8	2670	6779
9	2914	5452
10	4361	8867

Tabel 5.8 Tabel perbandingan *Routing Overhead* ORI dan MOD (50 node)

Tabel 5.8 serta **Gambar 5.6** di atas adalah skenario yang tengah berlangsung pada *node* dengan jumlah 50 *node.* Dari sana bisa diketahui bahwa pada *Routing Overhead* (RO) yang terjadi pada lingkungan yang telah dimodifikasi atau pada lingkungan heterogen mengalami keningkatan daripada saat masih ada pada lingkungan yang masih asli atau homogen. Hal itu terjadi di keseluruhan scenario, yakni skenario 1 hingga skenario 10. Dari sepuluh skenario tersebut, bisa dibuktikan bahwa kenaikan *Routing Overhead* (RO) tidak hanya terjadi pada satu skenario saja, namun terjadi di kesepuluh skenario yang ada pada *Routing Overhead* di lingkungan 50 *node.*



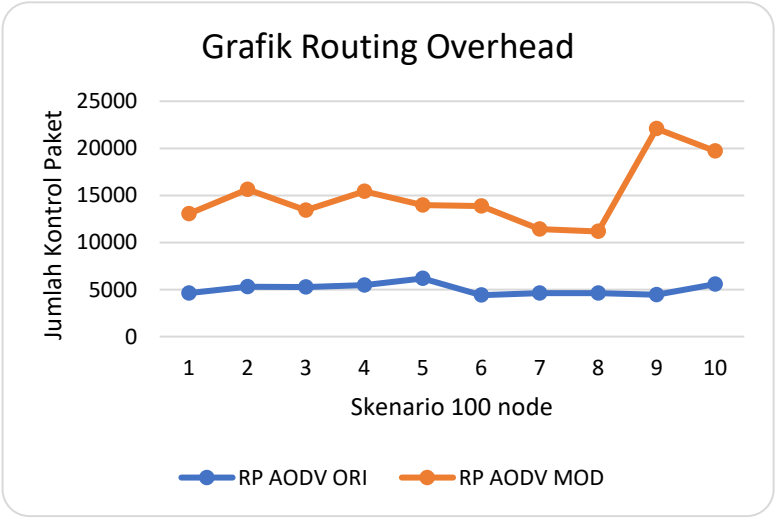
Gambar 5.7 Grafik nilai RO (75 *node*)

Skenario	RO	
	ORI	MOD
1	3463	4882
2	4469	8884

3	4532	8502
4	4734	10784
5	4152	17612
6	5445	11980
7	4646	7586
8	3399	5877
9	3563	9649
10	3404	4942

Tabel 5.9 Tabel perbandingan *Routing Overhead* ORI dan MOD (75 *node*)

Sama seperti pada *node* sebelumya, pada *node* dengan jumlah *node* sebanyak 75 *node Routing Overhead* (RO) juga mengalami kenaikan. Hal ini bisa dilihat pada **Gambar 5.7** dan **Tabel 5.9**. Kenaikan terbesar di sini terletak pada skenario 5 dengan kenaikan hampir empat kali lipatnya, yakni pada lingkungan homogen atau original (asli) memiliki *Routing Overhead* sebesar 4152 dan pada lingkungan heterogen mengalami kenaikan hingga mencapai 17612. Artinya



Gambar 5.8 Grafik nilai RO (100 *node*)

Skenario	RO	
	ORI	MOD
1	4626	13041
2	5302	15648
3	5250	13419
4	5477	15429
5	6178	13986
6	4402	13862
7	4606	11429
8	4630	11166
9	4448	22093
10	5572	19699

Tabel 5.10 Tabel perbandingan *Routing Overhead* ORI dan MOD (100 *node*)

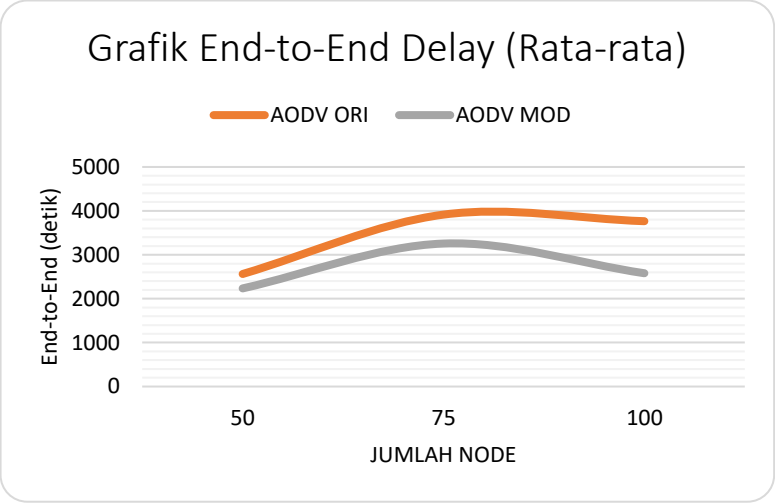
Pada **Gambar 5.8** dan **Tabel 5.10** adalah *Routing Overhead* (RO) yang terjadi pada *node* berjumlah 100 *node*. Di sana bisa dilihat bahwa terjadi kenaikan yang cukup signifikan pada *Routing Overhead* (RO) lingkungan yang heterogen atau yang mengalami modifikasi jika dibandingkan dengan lingkungan yang homogen atau yang tidak mengalami modifikasi (ori).kenaikan yang terjadi di sini rata-rata hampir mencapai dua kali lipat dan kenaikan terbesar pada *Routing Overhead* (RO) terjadi pada Skenario ke-9, yakni yang semula bernilai 4448 mengalami kenaikan hingga 22093 atau mengalami kenaikan hampir lima kali lipatnya.

5.2.3 Analisis *End-to-End Delay* (E2E)

Setelah melakukan uji coba selama berulang kali, pada akhirnya penulis mendapatkan nilai rerata untuk *End-to-End Delay* (E2E) yang mana dihasilkan oleh lingkungan dengan energi yang homogen maupun heterogen pada jaringan MANET dan diolah ke dalam grafik dan tabel di bawah ini.

Tabel 5.11 Hasil nilai E2E (jumlah *node*)

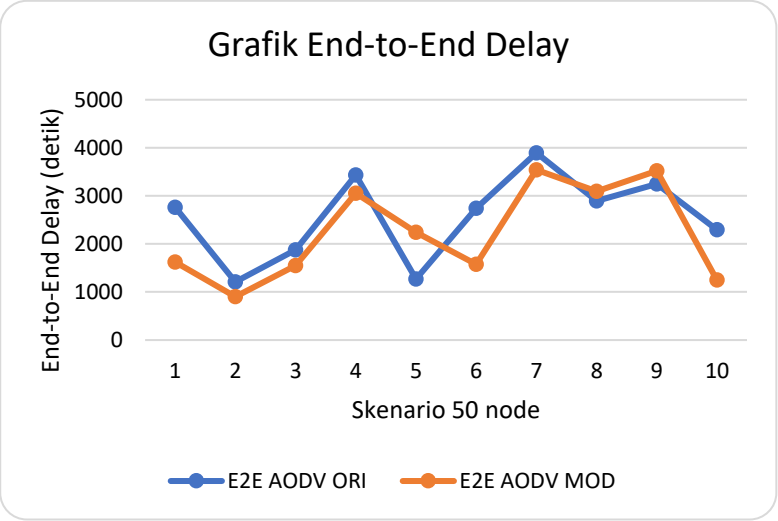
Jumlah Node	*End-to-End Delay* (detik)	
	ORI	MOD
50	2562,864	2235,0733
75	3915,014	3256,153
100	3764,94	2579,845



Gambar 5.9 Grafik E2E (rata-rata)

Berdasarkan **Tabel 5.11** dan **Gambar 5.9** di atas, pengujian yang dilakukan pada lingkungan dengan energi yang heterogen dengan parameter jumlah node yang berbeda pada setiap skenario, yaitu 50 *node* pada Skenario 1, 75 *node* pada Skenario 2, dan 100 *node* pada skenario terakhir, menunjukan nilai rata-rata dari *End-to-End Delay* yang sempat mengalami kenaikan namun kembali turun saat berada di *node* terakhir. Dengan demikian performa yang dihasilkan bisa dikategorikan semakin mengalami penurunan. Pada skenario pertama dengan node sejumlah 50 *node* rata-rata nilainya adalah 2235,0733 ms, kemudian pada *node* sejumlah 75 *node* sebesar 3256,153 ms dan pada node yang berjumlah 100 node nilainya 2579,845 ms. Sedangkan pada

lingkungan dengan energi yang homogen nilai rata-rata dari *End to End Delay* juga bersifat fluktuatif dimana pada skenario dengan 50 node nilai yang dihasilkan 2562,864 ms, lalu pada skenario dengan 75 node berjumlah 3915,014 ms dan meningkat lagi pada saat skenario 100 node menjadi 3764,94 ms.

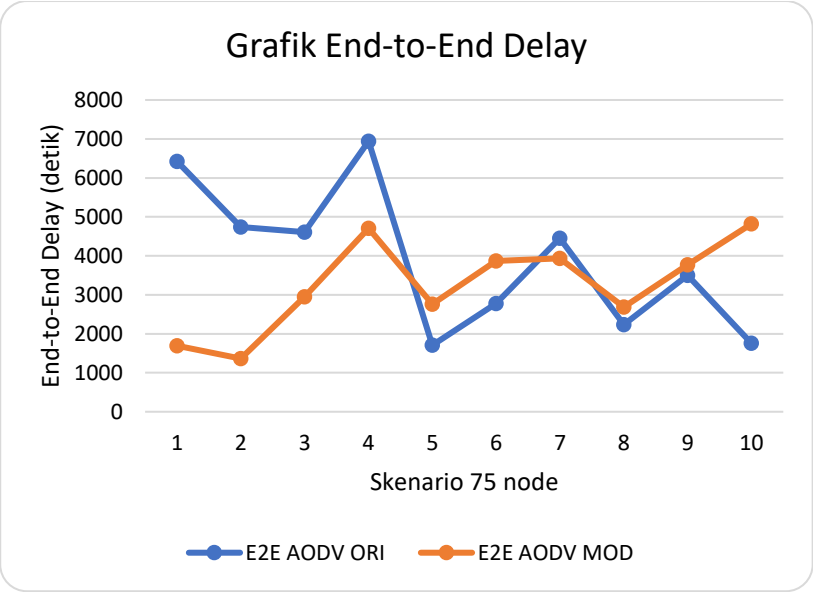


Gambar 5.10 Grafik nilai E2E (50 *node*)

Skenario	*End-to-End Delay*	
	ORI	**MOD**
1	2760,61	1620,41
2	1210,5	901,913
3	1873,11	1548,56
4	3435,51	3051,71
5	1269,03	2245,15
6	2742,83	1577,71
7	3894,93	3542,07
8	2897,96	3092,2
9	3246,49	3522,02
10	2297,67	1248,99

Tabel 5.12 Tabel perbandingan antara E2E ORI dan MOD pada 50 *node*

Sebagaimana yang ada pada **Gambar 5.10** dan **Tabel 5.12**, didapatkan kesimpulan bahwa dampak dari modifikasi yang dilakukan adalah terjadinya penurunan nilai pada skenario yang ada, meskipun dampak penurunan nilai yang ada tidak terjadi pada semua hasil modifikasi. Hal ini bisa dibuktikan dengan adanya beberapa skenario yang mengalami kenaikan dari nilai asli, seperti yang ada pada Skenario ke-5, Skenario ke-8, dan Skenario ke-9.



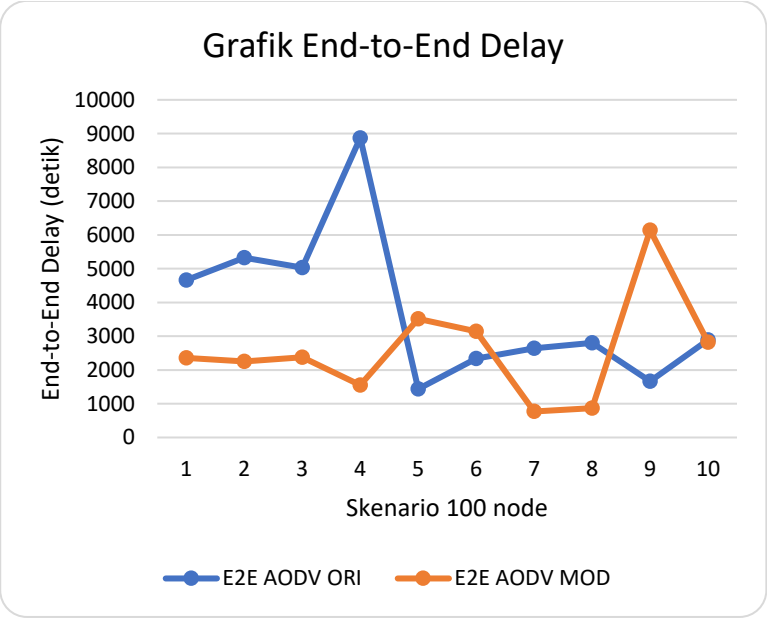
Gambar 5.11 Grafik nilai E2E (75 *node*)

Skenario	*End-to-End Delay*	
	ORI	**MOD**
1	6422,11	1690,65
2	4742,69	1364,16
3	4610,87	2946,75
4	6941,86	4709,78
5	1706,94	2762,09
6	2780,12	3868,44

7	4449,24	3939,88
8	2237,06	2687,45
9	3496,65	3773,04
10	1762,6	4819,29

Tabel 5.13 Tabel perbandingan antara E2E ORI dan MOD pada 75 *node*

Pada **Gambar 5.11** dan **Tabel 5.13**, didapatkan kesimpulan bahwa dampak dari modifikasi yang dilakukan adalah terjadinya penurunan nilai pada skenario yang ada, meskipun dampak penurunan nilai yang ada tidak terjadi pada semua hasil modifikasi. Hal ini bisa dibuktikan dengan adanya beberapa skenario yang mengalami kenaikan dari nilai asli, seperti yang ada pada Skenario ke-2, Skenario ke-5, Skenario ke-6, Skenario ke-8, Skenario ke-9, dan Skenario ke-10.



Gambar 5.12 Grafik nilai E2E (100 *node*)

Skenario	*End-to-End Delay*	
	ORI	**MOD**
1	4661,01	2354,43
2	5319,77	2251,63
3	5025,97	2379,67
4	8873,43	1553,73
5	1436,55	3511,22
6	2335,86	3143,35
7	2641,39	770,617
8	2800,6	869,501
9	1665,9	6141,18
10	2888,92	2823,12

Tabel 5.14 Tabel perbandingan antara E2E ORI dan MOD pada 100 *node*

Gambar 5.12 dan **Tabel 5.14** di atas, dapat diambil kesimpulan bahwa dampak dari modifikasi yang dilakukan adalah terjadinya penurunan nilai pada skenario yang ada, meskipun dampak penurunan nilai yang ada tidak terjadi pada semua hasil modifikasi. Hal ini bisa dibuktikan dengan adanya beberapa skenario yang mengalami kenaikan dari nilai asli, seperti yang ada pada Skenario ke-5, Skenario ke-6, dan Skenario ke-9.

(Halaman ini sengaja dikosongkan)

BAB VI
KESIMPULAN DAN SARAN

Pada bab ini akan diberikan kesimpulan yang diperoleh dari Tugas Akhir yang telah dikerjakan dan saran tentang pengembangan dari Tugas Akhir ini yang dapat dilakukan di masa yang akan datang.

6.1 Kesimpulan

Kesimpulan yang dapat diambil dari Tugas Akhir ini adalah sebagai berikut ini:

Dampak dari modifikasi ini bisa dilihat dari peningkatan PDR dan RO, yakni dengan rincian: Pada *node* 50, PDR mengalami peningkatan sebesar 47% dan RO mengalami peningkatan hingga 116%. Pada *node* 75, PDR mengalami peningkatan sebesar 50% dan RO juga mengalami peningkatan sebesar 117%. Pada *node* 100, PDR mengalami peningkatan sebesar 80,6% sedangkan untuk RO juga mengalami peningkatan hampir dua kali lipatnya, yakni sebesar 197%.

Namun, modifikasi ini tidak berdampak pada kenaikan pada *End-to-End* (E2E). Di sini, E2E mengalami penurunan nilai. Untuk *node* 50, E2E mengalami penurunan sebesar 12,7%. Pada *node* 75 dan *node* 100 juga mengalami penurunan, masing-masing sebesar 16,8% dan 31,4%.

6.2 Saran

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

1. Perhitungan jumlah *node* tetangga yang lebih dinamis pada *routing protocol* AODV. Contohnya saat *node* tetangga menjauh dari jangkauan, maka ada mekanisme untuk memperbarui *list* simpanan *node* tetangga sehingga perhitungan jumlah *node* tetangga lebih akurat.

2. Menambahkan aspek lain untuk melakukan pengembangan penelitian ini sehingga hasilnya lebih akurat.

DAFTAR PUSTAKA

[1] S. Othmen, A. Belghith, F. Zarai, M. S. Obaidat and L. Kamoun, ""Power and Delay-aware Multi-Path Routing Protocol for Ad Hoc Networks"," *2014 International Conference on Computer, Information and Telecommunication Systems (CITS),* pp. 1-6, 2014.

[2] B. B. Putra dan R. Anggoro, "Studi Kinerja Multipath AODV dengan Menggunakan Network simulator 2 (NS-2)," *JURNAL TEKNIK ITS Vol. 5, No. 2, (2016),* pp. A652-A656, 2016.

[3] N. Jiatmiko dan Y. Prayudi, "Simulasi Jaringan MANET dengan NS3 untuk Membandingkan Performa Routing Protokol AODV dan DSDV," dalam *Seminar Nasional Teknologi Informasi, Komunikasi dan Industri (SNTIKI) 7*, Pekanbaru, 2015.

[4] Alamsyah, E. Setijadi, I. K. E. Purnama dan M. H. Purnomo, "Analisis Kinerja Protokol Routing Reaktif dan Proaktif pada MANET Menggunakan NS2," *Jurnal Nasional Teknik Elektro dan Teknologi Informasi (JNTETI), Vol 7, No 2 (2018),* pp. 138-143, 2018.

[5] R. F. Sari, A. Syarif dan B. Budiardjo, "Analisis Kinerja Protokol Routing Ad Hoc On-Demand Distance Vector (AODV) Pada Jaringan Ad Hoc Hybrid: Perbandingan Hasil Simulasi Dengan Ns-2 dan Implementasi Pada Testbed dengan PDA," *MAKARA, Teknologi, Volume 12, No. 1, April 2008,* pp. 7-18, 2008.

[6] D. Irawan dan R. Roestam, "Simulasi Model Jaringan Mobile Ad-Hoc (MANET) dengan NS-3," dalam

Konferensi Nasional Sistem dan Informatika 2011, Bali, 2011.

[7] Alamsyah, P. I. K. Eddy, E. Setijadi dan M. Hery Purnomo, "Analisis Kinerja Protokol Routing AODV, DSR, dan OLSR pada Mobile Ad hoc Network Berdasarkan Parameter Quality of Service," *Jurnal Rekayasa Elektrika Vol. 14, No. 3,* vol. 14, pp. 145-151, 2018.

[8] M. Iqbal, M. Shafiq, H. Attaullah, J.-G. Choi, K. Akram dan X. Wang, "Design and Analysis of a Novel Hybrid Wireless Mesh Network Routing Protocol," p. 22, January 2014.

[9] P. D. Pradana, R. M. Negara dan F. Dewanta, "Evaluasi Performansi Protokol Routing DSR dan AODV pada Simulasi Jaringan Vehicular Ad-hoc Network (VANET) untuk Keselamatan Transportasi dengan Studi Kasus Mobil Perkotaan," *e-Proceeding of Engineering : Vol.4, No.2 Agustus 2017,* pp. 1996-2004, 2017.

[10] "Nam: Network Animator," [Online]. Available: https://www.isi.edu/nsnam/nam/. [Diakses 24 Januari 2020].

[11] The Linux Documentation Project (TLDP), "C.2. AWK," [Online]. Available: http://tldp.org/LDP/abs/html/awk.html. [Diakses 24 Desember 2019].

[12] A. V. Aho, B. W. Kernighan dan P. J. Weinberger, dalam *The AWK programming language*, Addison-Wesley Publishing Company, 1988, p. iii.

[13] T. Issariyakul dan E. Hossain, "12.6.4.1 Mobility Generation Utility "Setdest"," dalam *Introduction to Network Simulator NS2*, Springer, 2010, p. 341.

LAMPIRAN

A.1 Kode Skenario NS-2 Asli

```
# ===================================================
# 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 opt(x)          500;# X dimension of the topography
set opt(y)                      500;# Y dimension of the topography
set val(ifqlen)                 50;# max packet in ifq
set val(nn)                     75;# how many nodes are simulated
set val(seed)                           1.0;
set val(adhocRouting)           AODV;# routing protocol
set val(stop)                   200;# simulation time
set val(cp)                     "cbr.txt";#<-- traffic file
set val(sc)                     "scenario.txt";#<-- mobility file

#energy
```

```
23. #src: https://www.nsnam.com/2012/11/energy-model-in-
    network-simulator-2-ns2.html
24. set val(energy_mod)         EnergyModel;# energy
    model
25. set val(energy_init)        80;# init val for energy
26. set val(tx_power)           1.65;# energy consume
    for transmiting packet
27. set val(rx_power)           1.4;# energy consume for
    receiving packet
28. set val(idle_power)         0.5;# energy consume for
    idle
29. set val(sleep_power)        0.3;# energy consume for
    sleep mode
30.
31. #
    ===================================================
32. # Main Program
33. #
    ===================================================
34. # Initialize Global Variables
35. # create simulator instance
36.
37. set ns_         [new Simulator]
38.
39. # setup topography object
40.
41. set topo        [new Topography]
42.
43. #if { $val(adhocRouting) == "DSR" } {
44. #set val(ifq)            CMUPriQueue
45. #} else {
46. #set val(ifq)            Queue/DropTail/PriQueue
47. #}
48. # create trace object for ns and nam
49.
50. set tracefd      [open result_ori.tr w]
51. set namtrace    [open result_ori.nam w]
52.
53. $ns_ trace-all $tracefd
54. $ns_ namtrace-all-wireless $namtrace $opt(x) $opt(y)
55.
56. # set up topology object
57. set topo
```

```
58. $topo load_flatgrid $opt(x) $opt(y)
59.
60. # Create God
61. set god_ [create-god $val(nn)]
62.
63. #with energy
64. $ns_ node-config    -adhocRouting $val(adhocRouting)
    \
65.                     -llType $val(ll) \
66.                     -macType $val(mac) \
67.                     -ifqType $val(ifq) \
68.                     -ifqLen $val(ifqlen) \
69.                     -antType $val(ant) \
70.                     -propType $val(prop) \
71.                     -phyType $val(netif) \
72.                     -channelType $val(chan) \
73.                     -energyModel $val(energy_mod) \
74.                     -initialEnergy $val(energy_init)
    \
75.                     -txPower $val(tx_power) \
76.                     -rxPower $val(rx_power) \
77.                     -idlePower $val(idle_power) \
78.                     -sleepPower $val(sleep_power) \
79.                     -topoInstance $topo \
80.                     -agentTrace ON \
81.                     -routerTrace ON \
82.                     -macTrace ON \
83.                     -movementTrace ON
84.
85. # 802.11p default parameters
86.
87. Phy/WirelessPhy
88. Phy/WirelessPhy set
89.
90. # 1m : 0.000192278
91. # 5m : 7.69113e-06
92. # 10m : 1.92278e-06
93. # 25m : 3.07645e-07
94. # 50m : 7.69113e-08
95. # 75m : 3.41828e-08
96. # 100m : 1.42681e-08
97. # 125m : 5.8442e-09
98. # 150m : 2.81838e-09
```

```
# 175m : 1.52129e-09
  # 200m : 8.91754e-10
  # 225m : 5.56717e-10
  # 250m : 3.65262e-10
  # 400m : 5.57189e-11
  # 500m : 2.28289e-11
  # 1000m : 1.42681e-12


  #  Create the specified number of nodes [$val(nn)] and "attach" them
  #  to the channel.
  #  plus random energy
  #  src : http://slogix.in/how-to-find-residual-energy-of-the-nodes-in-ns2
  for {set i 0} {$i < $val(nn)} {incr i} {
       $ns_ node-config -adhocRouting $val(adhocRouting) \
                        -llType $val(ll) \
                        -macType $val(mac) \
                        -ifqType $val(ifq) \
                        -ifqLen $val(ifqlen) \
                        -antType $val(ant) \
                        -propType $val(prop) \
                        -phyType $val(netif) \
                        -channelType $val(chan) \
                        -energyModel $val(energy_mod) \
                        -initialEnergy $energy($i) \
                        -txPower $val(tx_power) \
                        -rxPower $val(rx_power) \
                        -idlePower $val(idle_power) \
                        -sleepPower $val(sleep_power) \
                        -topoInstance $topo \
                        -agentTrace ON \
                        -routerTrace ON \
                        -macTrace ON \
                        -movementTrace ON
  set node_($i) [$ns_ node]
  $node_($i) random-motion 0 ;# disable random motion
```

```
135.  $node_($i) color black; #whatever you fill as
      long as color
136.  }
137.
138.
139.  # Define node movement model
140.  puts "Loading connection pattern..."
141.  source $val(cp)
142.
143.  # Define traffic model
144.  puts "Loading scenario file..."
145.  source $val(sc)
146.
147.  # Define node initial position in nam
148.
149.  for {set i 0} {$i < $val(nn)} {incr i} {
150.
151.      # 20 defines the node size in nam, must adjust
      it according to your scenario
152.      # The function must be called after mobility
      model is defined
153.
154.      $ns_ initial_node_pos $node_($i) 100
155.  }
156.
157.  # Tell nodes when the simulation ends
158.  for {set i 0} {$i < $val(nn)} {incr i} {
159.      $ns_ at $val(stop).0 "$node_($i) reset";
160.  }
161.
162.  #$ns_ at  $val(stop)
163.  $ns_ at  $val(stop).0002 "puts \"NS EXITING...\"
      ; $ns_ halt"
164.
165.  #puts $tracefd "M 0.0 nn $val(nn) x $opt(x) y
      $opt(y) rp $val(adhocRouting)"
166.  #puts $tracefd "M 0.0 sc $val(sc) cp $val(cp)
      seed $val(seed)"
167.  #puts $tracefd "M 0.0 prop $val(prop) ant
      $val(ant)"
168.
169.  puts "Starting Simulation..."
170.  $ns_ run
```

A.2 Kode Skenario NS-2 Modifikasi

```
# ======================================================
# 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 opt(x)                      500;# X dimension of the topography
set opt(y)                      500;# Y dimension of the topography
set val(ifqlen)                 50;# max packet in ifq
set val(nn)                     75;# how many nodes are simulated
set val(seed)                           1.0;
set val(adhocRouting)           AODV;# routing protocol
set val(stop)                   200;# simulation time
set val(cp)                     "cbr.txt";#<-- traffic file
set val(sc)                     "scenario.txt";#<-- mobility file

```

```
#energy
#src: ttps://www.nsnam.com/2012/11/energy-model-in-network-simulator-2-ns2.html
set val(energy_mod)         EnergyModel;# energy model
set val(energy_init)        80;# init val for energy
set val(tx_power)           1.65;# energy consume for transmiting packet
set val(rx_power)           1.4;# energy consume for receiving packet
set val(idle_power)         0.5;# energy consume for idle
set val(sleep_power)        0.3;# energy consume for sleep mode

# ==================================================
# Main Program
# ==================================================
# Initialize Global Variables
# create simulator instance

set ns_         [new Simulator]

# setup topography object

set topo        [new Topography]

#if { $val(adhocRouting) == "DSR" } {
#set val(ifq)            CMUPriQueue
#} else {
#set val(ifq)          Queue/DropTail/PriQueue
#}
# create trace bject for ns and nam

set tracefd       [open result_mod.tr w]
set namtrace    [open result_mod.nam w]

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

# set up topology bject
set topo
$topo load_flatgrid $opt(x) $opt(y)
```

```
59.
60. # Create God
61. set god_ [create-god $val(nn)]
62.
63. #with energy
64. $ns_ node-config    -adhocRouting $val(adhocRouting)
    \
65.                     -llType $val(ll) \
66.                     -macType $val(mac) \
67.                     -ifqType $val(ifq) \
68.                     -ifqLen $val(ifqlen) \
69.                     -antType $val(ant) \
70.                     -propType $val(prop) \
71.                     -phyType $val(netif) \
72.                     -channelType $val(chan) \
73.                     -energyModel $val(energy_mod) \
74.                     -initialEnergy $val(energy_init)\
75.                     -txPower $val(tx_power) \
76.                     -rxPower $val(rx_power) \
77.                     -idlePower $val(idle_power) \
78.                     -sleepPower $val(sleep_power) \
79.                     -topoInstance $topo \
80.                     -agentTrace ON \
81.                     -routerTrace ON \
82.                     -macTrace ON \
83.                     -movementTrace ON
84.
85. # 802.11p default arameters
86.
87. Phy/WirelessPhy
88. Phy/WirelessPhy set
89.
90. # 1m : 0.000192278
91. # 5m : 7.69113e-06
92. # 10m : 1.92278e-06
93. # 25m : 3.07645e-07
94. # 50m : 7.69113e-08
95. # 75m : 3.41828e-08
96. # 100m : 1.42681e-08
97. # 125m : 5.8442e-09
98. # 150m : 2.81838e-09
99. # 175m : 1.52129e-09
100.   # 200m : 8.91754e-10
```

```
101.   # 225m : 5.56717e-10
102.   # 250m : 3.65262e-10
103.   # 400m : 5.57189e-11
104.   # 500m : 2.28289e-11
105.   # 1000m : 1.42681e-12
106.
107.
108.   #  Create the specified number of nodes
       [$val(nn)] and "attach" them
109.   #  to the channel.
110.   #  plus random energy
111.   #  src : http://slogix.in/how-to-find-residual-energy-of-the-
       nodes-in-ns2
112.   for {set i 0} {$i < $val(nn)} {incr i} {
113.     if {$i < [expr $val(nn) - 2]} {
114.         set energy($i) [expr int(40 + rand()*60)];
115.        $ns_ node-config -adhocRouting
       $val(adhocRouting) \
116.                         -llType $val(ll) \
117.                         -macType $val(mac) \
118.                         -ifqType $val(ifq) \
119.                         -ifqLen $val(ifqlen) \
120.                         -antType $val(ant) \
121.                         -propType $val(prop) \
122.                        -phyType $val(netif) \
123.                         -channelType $val(chan) \
124.                         -energyModel $val(energy_mod)\
125.                         -initialEnergy $energy($i) \
126.                         -txPower $val(tx_power) \
127.                         -rxPower $val(rx_power) \
128.                         -idlePower $val(idle_power) \
129.                         -sleepPower val(sleep_power) \
130.                         -topoInstance $topo \
131.                         -agentTrace ON \
132.                         -routerTrace ON \
133.                         -macTrace ON \
134.                         -movementTrace ON
135.           set node_($i) [$ns_ node]
136.           $node_($i) random-motion 0;# disable
       random motion
137.           $node_($i) color blue; #whatever you fill
       as long as color
```

```
138.          set E($i) $energy($i)
139.          } else {
140.          $ns_ node-config -adhocRouting
    $val(adhocRouting)\
141.                        -llType $val(ll) \
142.                        -macType $val(mac) \
143.                        -ifqType $val(ifq) \
144.                        -ifqLen $val(ifqlen) \
145.                        -antType $val(ant) \
146.                        -propType $val(prop) \
147.                        -phyType $val(netif) \
148.                        -channelType $val(chan) \
149.                        -energyModel $val(energy_mod)
    \
150.                        -initialEnergy
    $val(energy_init)\
151.                        -txPower $val(tx_power) \
152.                        -rxPower $val(rx_power) \
153.                        -idlePower $val(idle_power) \
154.                        -sleepPower $val(sleep_power)
    \
155.                        -topoInstance $topo \
156.                        -agentTrace ON \
157.                        -routerTrace ON \
158.                        -macTrace ON \
159.                        -movementTrace ON
160.          set node_($i) [$ns_ node]
161.          $node_($i) random-motion 0 ;# disable
    random motion
162.          $node_($i) color black; #whatever you fill
    as long as color
163.        }
164.   }
165.
166.   for {set i 0} {$i < $val(nn)} {incr i} {
167.       set node_($i) [$ns_ node]
168.       $node_($i) random-motion 0 ;# disable random
    motion
169.   }
170.
171.
172.   # Define node movement model
173.   puts "Loading connection pattern..."
```

```
source $val(cp)

# Define traffic model
puts "Loading scenario file..."
source $val(sc)

# Define node initial position in nam

for {set i 0} {$i < $val(nn)} {incr i} {

    # 20 defines the node size in nam, must adjust it according to your scenario
     # The function must be called after mobility model is defined

     $ns_ initial_node_pos $node_($i) 100
}

# 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)
$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(adhocRouting)"
#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
```

A.3 Kode Koneksi yang Digunakan pada ‘cbr.txt’

```
#
# nodes: 75, max conn: 1, send rate: 1.0, seed: 1
#
#
```

```
# 1 connecting to 2 at time 2.5568388786897245
#
set udp_(0) [new Agent/UDP]
$ns_ attach-agent $node (73) $udp_(0)
set null_(0) [new Agent/Null]
$ns_ attach-agent $node_(74) $null_(0)
set cbr_(0) [new Application/Traffic/CBR]
$cbr_(0) set packetSize_ 512
$cbr_(0) set interval_ 1.0
$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
#
```

A.4 Kode Skrip AWK *Packet Delivery Ratio* (PDR)

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

}

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

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

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

END {
   printf "Packet PDR Ratio \t= %.4f \n",
   (recvLine/sendLine);
}
```

A.5 Kode Skrip AWK *End-to-End Delay* (E2E)

```
BEGIN {
      seqno = -1;
      count = 0;
}

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

      #end-to-end delay
      if($4 == "AGT" && $1 == "s") {
            start_time[$6] = $2;
      }
      else if(($7 == "cbr") && ($1 == "r")) {
            end_time[$6] = $2;
      }
      else if($1 == "D" && $7 == "cbr") {
            end_time[$6] = -1;
      }
}
END {
      for(i=0; i<=seqno; i++) {
            if(end_time[i] > 0) {
                delay[i] = end_time[i] -
   start_time[i];
                count++;
            }
            else {
                delay[i] = -1;
            }
      }
      for(i=0; i<=seqno; i++) {
            if(delay[i] > 0) {
                n_to_n_delay = n_to_n_delay +
   delay[i];
            }
      }
      n_to_n_delay = n_to_n_delay/count;
```

```
38.      printf "End-to-End Delay \t= " n_to_n_delay *
    1000 " ms \n";
39. }
```

A.6 Kode Skrip AWK *Routing Overhead* (RO)

```
1.  BEGIN {
2.       rt_pkts = 0;
3.  }
4.       {
5.            if (($1 == "s" || $1 == "f") && ($4 ==
    "RTR") && ($7 == "AODV")) {
6.            rt_pkts++;
7.       }
8.  }
9.  }
10. END {
11.     printf "Routing Packets \t= %d \n", rt_pkts;
12. }
```

BIODATA PENULIS

Ahmad Hanan, dilahirkan di Kabupaten Rembang pada 13 Juni 1996. Penulis adalah anak pertama dari dua bersaudara. Penulis menempuh pendidikan sekolah dasar di SDN Kutoharjo 3 Rembang lalu melanjutkan pendidikan sekolah menengah pertama di MTsN Lasem (sekarang MTsN 1 Rembang) dan penulis menempuh pendidikan menengah atas di MA TBS Kudus. Selanjutnya penulis melanjutkan pendidikan sarjana di Departemen Informatika, Fakultas Teknologi Informasi dan Komunikasi, Institut Teknologi Sepuluh Nopember Surabaya melalui jalur PBSB, suatu program afirmasi antara Kementerian Agama (Kemenag) Republik Indonesia dengan ITS. Dalam menyelesaikan pendidikan S1, penulis mengambil bidang minat Arsitektur dan Jaringan Komputer (AJK). Sebagai mahasiswa, penulis aktif dalam beberapa organisasi kampus seperti staf, Staf Unit Kegiatan Mahasiswa Lembaga Pers Mahasiswa 1.0 ITS, Unit Kegiatan Mahasiswa Cinta Rebana ITS (sekarang UKM Rebana ITS), dan sempat menjadi salah satu reporter dari pers kehumasan kampus, ITS Online, pada 2015. Selain itu, penulis juga menjadi staf SCHEMATICS sub acara National Seminar of Technology (NST) pada tahun 2015 dan 2016. Penulis pernah melakukan kerja praktik di Kemenag RI pada Juli - Agustus 2017 dan membuat aplikasi berbasis web berupa sistem informasi pengelolaan peminjaman ijazah bagi alumni PBSB. Penulis dapat dihubungi melalui nomor *handphone*: 0822-111-78978 atau *email*: ahmadhanan.aan@gmail.com

(Halaman ini sengaja dikosongkan)