TUGAS AKHIR – IF4802

IMPLEMENTASI *ROUTING PROTOCOL* AODV DENGAN *ADAPTIF HELLO INTERVAL* PADA MANETS

MUCHAMAD BUYUNG ABIYOSO
NRP 05111440000150

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

Departemen Informatika
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2019

(Halaman ini sengaja dikosongkan)

TUGAS AKHIR – IF4802

IMPLEMENTASI *ROUTING PROTOCOL* AODV DENGAN *ADAPTIF HELLO INTERVAL* PADA MANETS

MUCHAMAD BUYUNG ABIYOSO
NRP 05111440000150

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

Departemen Informatika
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2019

(Halaman ini sengaja dikosongkan)

UNDERGRADUATE THESES – IF4802

IMPLEMENTATION OF AODV ROUTING PROTOCOL WITH ADAPTIF HELLO INTERVAL IN MANETS

MUCHAMAD BUYUNGABIYOSO
NRP 05111440000150

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

Department of Informatics
Faculty of Information and Communication Technology
Sepuluh Nopember Institute of Technology
Surabaya 2019

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

IMPLEMENTASI ROUTING PROTOCOL AODV DENGAN ADAPTIF HELLO INTERVAL PADA MANETS

TUGAS AKHIR

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

Oleh:
MUCHAMAD BUYUNG ABIYOSO
NRP: 05111440000150

Disetujui oleh Pembimbing Tugas Akhir:

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

SURABAYA
JANUARI, 2019

(Halaman ini sengaja dikosongkan)

IMPLEMENTASI *ROUTING PROTOCOL* AODV DENGAN *ADAPTIF HELLO INTERVAL* PADA MANETS

Nama Mahasiswa	**: Muchamad Buyung Abiyoso**
NRP	**: 05111440000150**
Departemen	**: Informatika FTIK-ITS**
Dosen Pembimbing 1	**: Dr.Eng. Radityo Anggoro, S.Kom., M.Sc.**

Abstrak

Perkembangan teknologi *wireless* dewasa ini berkembang dengan pesat. Salah satunya adalah teknologi jaringan *Mobile Ad hoc Network* (MANET). Dengan jaringan MANET memungkinkan komunikasi tanpa bergantung pada ketersedian insfratuktur yang tetap. Jaringan MANET adalah kumpulan dari beberapa *wireless node* yang dapat di *set-up* secara dinamis dimana saja dan kapan saja tanpa menggunakan infrastruktur jaringan yang ada.

Dalam teknologi MANET tidak bisa dilepaskan dengan proses pengiriman paket data. Proses pencarian jalur untuk mengirimkan data dari sumber ke tujuan disebut dengan *routing*. Dalam proses *routing*, data yang berasal dari *node* sumber dikirimkan ke *node-node* lain hingga mencapai *node* tujuan. Terdapat beberapa metode *routing* pada jaringan MANET diantaranya *routing protocol* AODV (*Ad Hoc On-Demand Distance Vector*).

Dalam tugas akhir ini, dilakukan implementasi routing protocol AODV pada jaringan MANET dengan menggunakan *Adaptif Hello Interval* pada Network Simulator 2. Keefektifan kinerja *routing protocol* tersebut dapat diketahui dari tiga parameter yaitu *packet delivery ratio*, *routing overhead* dan *delay* pengiriman.

Kata kunci: MANET, AODV, Hello Interval

(Halaman ini sengaja dikosongkan)

IMPLEMENTATION OF AODV ROUTING PROTOCOL WITH ADAPTIF HELLO INTERVAL IN MANETS

Student's Name	**:**	**Muchamad Buyung Abiyoso**
Student's ID	**:**	**05111440000150**
Department	**:**	**Informatics – FTIK ITS**
First Advisor	**:**	**Dr.Eng. Radityo Anggoro, S.Kom., M.Sc.**

Abstract

Lately, development of wireless technology is growing rapidly. One of those technology is Mobile Ad hoc Network (MANET). MANET allows communication without depend on fixed infrastructure availability. MANET is a set of several wireless node that can be set-up anywhere and anytime without using available network infrastructure.

MANET technology cannot be separated with the process of data packet transmission. The process of searching paths for sending data from source to destination is called routing. In the routing process, the data from the source node is sent to another nodes until it reaches the destination node. There are several routing methods on the MANET. One of those is AODV routing protocol (Ad Hoc On-Demand Distance Vector).

In this final project, the AODV routing protocol is implemented on the MANET by using Adaptif Hello Interval on Network Simulator 2. The effectiveness of the routing protocol can be seen from three parameters, namely the packet delivery ratio, overhead routing and delivery delay.

Keywords: MANET, AODV, Hello Interval

(Halaman ini sengaja dikosongkan)

KATA PENGANTAR

Puji syukur kepada Allah Yang Maha Esa atas segala karunia dan rahmat-Nya penulis dapat menyelesaikan Tugas Akhir yang berjudul **"Implementasi *Routing Protocol* AODV dengan *Adaptif Hello Interval* pada MANETS".**

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. Ayah, Ibu , dan adik-adik penulis atas segala dukungan berupa motivasi serta doa sehingga penulis dapat mengerjakan Tugas Akhir ini.
2. Bapak Dr. Eng. Radityo Anggoro, S.Kom., M.Sc. selaku dosen pembimbing atas arahan dan bantuannya dalam pengerjaan Tugas Akhir ini.
3. Rizky Fenaldo Maulana, S.Kom atas segala bantuan, saran, serta dukungannya selama pengerjaan tugas akhir ini.
4. Ilham M. Misbahudin atas segala bantuan selama perkuliahan jarnil dan pengerjaan tugas akhir.
5. Paul Aldy, Faris Salbari, Muchammad Baruno, Brilian WM, Hamka Aminullah, M. Luqmanul Hakim, Hanendyo Indira, Aldi Febriansyah, Anandi Jaya, Bayu Aji dan Naufan Arifie yang telah berjuang bersama dan selalu membantu penulis dalam mengerjakan tugas akhir di semester 9 ini.
6. Teman-teman Warkop x Jojoran atas segala bantuan, dukungan, dan saran, serta pengalaman hidup yang berharga selama 4,5 tahun masa perkuliahan penulis di ITS.
7. Sahabat-sahabat penulis di group chat BFF-NYA PAPAH yang selama ini telah menemani penulis di saat suka dan duka.

8. Teman-teman TC’14 yang sudah menyemangati Penulis selama menjalani masa perkuliahan di ITS.
9. 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, 4 Januari 2019

Muchamad Buyung Abiyoso

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 *wireless* dewasa ini berkembang dengan pesat. Salah satunya adalah teknologi jaringan *Mobile Ad hoc Network* (MANET). Dengan jaringan MANET memungkinkan komunikasi tanpa bergantung pada ketersedian insfratuktur yang tetap. Jaringan MANET adalah kumpulan dari beberapa *wireless node* yang dapat di *set-up* secara dinamis dimana saja dan kapan saja tanpa menggunakan infrastruktur jaringan yang ada.

Dalam teknologi MANET tidak bisa dilepaskan dengan proses pengiriman paket data. Proses pencarian jalur untuk mengirimkan data dari sumber ke tujuan disebut dengan *routing.* Dalam proses *routing,* data yang berasal dari *node* sumber dikirimkan ke *node-node* lain hingga mencapai *node* tujuan. Terdapat beberapa metode *routing* pada jaringan MANET diantaranya *routing protocol* AODV (*Ad Hoc On-Demand Distance Vector*).

Hello Interval merupakan interval waktu yang dibutuhkan *node* untuk mem-*broadcast Hello Message.* Dimana *Hello Message* bertugas untuk mengecek *node* tetangga mana saja yang tersedia sebagai kandidat *forwarding node* untuk menerima data paket ke *destination.* Oleh karena itu, apabila terdapat perubahan pada *Hello Interval* maka akan sangat berpengaruh pada AODV secara keseluruhan.

Pengiriman *Hello Message* secara berkala adalah skema yang paling sering digunakan untuk memperoleh informasi jaringan local. Akan tetapi apabila interval waktu yang di atur terlalu cepat akan mengurangi daya dari device karena mengirim *Hello message* yang tidak diperlukan. Dan apabila *Hello interval* di atur terlalu lama akan menyebabkan node kesulitan untuk mencari rute pengiriman paket sehingga menurunkan tingkat

keberhasilan pengiriman paket. Oleh karena itu dibutuhkan pengaturan *Hello interval* yang dapat mengurangi pengiriman *Hello Message* yang tidak diperlukan tanpa menurunkan tingkat keberhasilan pengiriman paket secara signifikan.

Melihat permasalahan di atas, pada Tugas Akhir ini akan dilakukan analisis kinerja pada *routing protocol* AODV dengan mengubah *Hello Interval*. Hasil akhir yang diharapkan adalah mengetahui peran *Hello Interval* pada kinerja AODV yang diukur berdasarkan performansi *Packet Delivery Ratio* (PDR), *Routing Overhead*, dan *End-to-End Delay*. Sehingga di kemudian hari bisa dijadikan pedoman penelitian.

1.2 Rumusan Masalah

Berikut yang menjadi rumusan masalah pada Tugas Akhir ini adalah:

1. Bagaimana pengaruh *Hello Interval* pada kinerja AODV dalam aspek *Packet Delivery Ratio*, *End-to-End Delay*, dan *Routing Overhead* di lingkungan MANET?

1.3 Batasan Permasalahan

Batasan masalah pada tugas akhir ini adalah sebagai berikut:

1. Jaringan yang digunakan adalah jaringan *Mobile Ad hoc Networks* (MANET).
2. *Routing protocol* yang diujicobakan yaitu AODV.
3. Simulasi pengujian jaringan menggunakan *Network Simulator 2* (NS-2).
4. Menganalisa kinerja Routing Protocol AODV menggunakan *Adaptive Hello Interval*.
5. Analisis Kinerja Didasarkan pada *Packet Delivery Ratio (PDR), Routing Overhead (RO),* dan *End-to-End Delay.*

1.4 Tujuan

Tujuan dari Tugas Akhir ini adalah mengetahui dampak *Hello Interval* pada kinerja AODV di lingkungan MANET.

1.5 Manfaat

Manfaat yang didapatkan dengan dibuatnya Tugas Akhir ini adalah sebagai pedoman penelitian yang berhubungan dengan pengembangan *Hello Interval* pada AODV di lingkungan MANET di masa yang akan datang.

1.6 Metodologi

Pembuatan Tugas Akhir ini dilakukan dengan menggunakan metodologi sebagai berikut:

1.6.1 Penyusunan Proposal Tugas Akhir

Proposal tugas akhir ini berisi tentang deskripsi pendahuluan dari tugas akhir yang akan dibuat. Pendahuluan ini terdiri atas hal yang menjadi latar belakang diajukannya usulan tugas akhir, rumusan masalah yang diangkat, batasan masalah untuk tugas akhir, tujuan dari pembuatan tugas akhir, dan manfaat dari hasil pembuatan tugas akhir. Selain itu dijabarkan pula tinjauan pustaka yang digunakan sebagai referensi pendukung pembuatan tugas akhir. Sub bab metodologi berisi penjelasan mengenai tahapan penyusunan tugas akhir mulai dari penyusunan proposal hingga penyusunan buku tugas akhir. Terdapat pula sub bab jadwal kegiatan yang menjelaskan jadwal pengerjaan tugas akhir. Tinjauan pustaka digunakan sebagai referensi yang mendukung pembuatan tugas akhir.

1.6.2 Studi Literatur

Literatur yang akan dipelajari untuk menunjang pembuatan sistem ini antara lain mengenai *Mobile Ad Hoc*

Networks (MANETs), *reactive routing protocol* AODV, dan *Network Simulator 2* (NS-2).

1.6.3 Analisis dan Desain Sistem

Pada tahap ini dilakukan analisis kinerja dari hasil percobaan pengubahan *Hello Interval* pada *reactive routing protocol* AODV. Data yang dianalisis berasal dari perhitungan *Packet Delivery Ratio* (PDR), *Routing Overhead* (RO), dan *End-to-End Delay*. Hal ini dimaksudkan untuk mengetahui dampak perubahan *Hello Interval* pada kinerja *reactive routing protocol* AODV dalam topologi MANET.

1.6.4 Implementasi Sistem

Pada tahap ini dilakukan perancangan model jaringan, implementasi yang dibuat berupa perubahan *Hello Interval* pada protokol AODV untuk mengetahui kinerja *Hello Interval*.

1.6.5 Pengujian dan Evaluasi

Pengujian dilakukan dengan NS-2 *Network Simulator* dan akan menghasilkan *trace file*. Dari *trace file* tersebut akan dihitung *Packet Delivery Ratio* (PDR), *Routing Overhead* (RO), dan *End-to-End Delay* untuk mengetahui performa *routing protocol* yang *Hello Interval*nya telah dirubah.

1.6.6 Penyusunan Buku

Pada tahap ini dilakukan penyusunan laporan yang menjelaskan dasar teori dan metode yang digunakan dalam tugas akhir ini serta hasil dari implementasi sistem yang telah dibuat.

1.7 Sistematika Penulisan Laporan

Sistematika penulisan laporan Tugas Akhir adalah sebagai berikut:

1. Bab I. Pendahuluan
   Bab ini berisikan penjelasan mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi, dan sistematika penulisan dari pembuatan Tugas Akhir.
2. Bab II. Tinjauan Pustaka
   Bab ini berisi kajian teori atau penjelasan dari metode, algoritma, *library*, dan *tools* yang digunakan dalam penyusunan Tugas Akhir ini. Bab ini berisi tentang penjelasan singkat mengenai MANET, AODV, NS2, AWK, dan *Hello Interval*.
3. Bab III. Perancangan
   Bab ini berisi pembahasan mengenai perancangan skenario mobilitas *grid* dan *real*, perancangan simulasi pada NS2, perancangan perubahan *Hello Interval* pada AODV, serta perancangan metrik analisis (*Packet Delivery Ratio*, *Routing Overhead*, dan *End-to-End Delay*).
4. Bab IV. Implementasi
   Bab ini menjelaskan implementasi yang berbentuk kode sumber dari proses perubahan *Hello Interval* pada protokol AODV, pembuatan simulasi pada NS2, dan perhitungan metrik analisis.
5. Bab V. Uji Coba dan Evaluasi
   Bab ini berisikan hasil uji coba dan evaluasi dari implementasi yang telah dilakukan untuk menyelesaikan masalah yang dibahas pada Tugas Akhir.
6. 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.

7. Daftar Pustaka
   Bab ini berisi daftar pustaka yang dijadikan literatur dalam Tugas Akhir.
8. Lampiran
   Dalam lampiran terdapat tabel-tabel data hasil uji coba dan kode sumber program secara keseluruhan.

BAB II
TINJAUAN PUSTAKA

Bab ini berisi pembahasan mengenai teori-teori dasar atau penjelasan dari metode dan *tools* yang digunakan dalam Tugas Akhir.

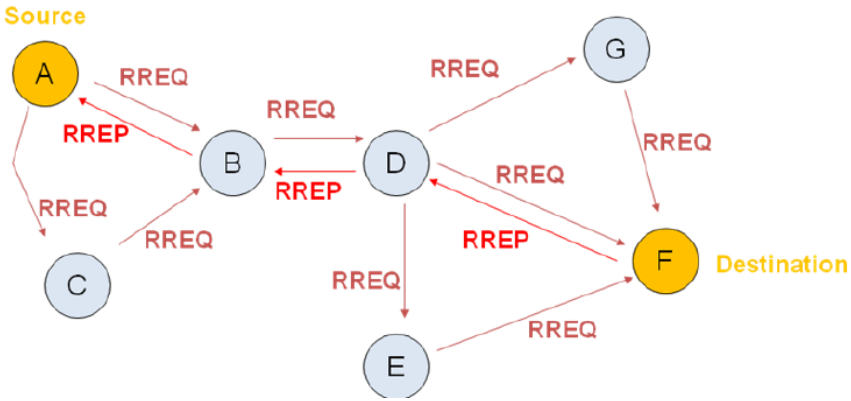
2.1 MANET

Mobile Ad-Hoc Network (MANET) adalah kumpulan *wireless mobile hosts* yang membentuk jaringan sementara tanpa bantuan infrastruktur yang berdiri sendiri atau administrasi terpusat. MANET juga merupakan jaringan sementara yang terbentuk dari beberapa *mobile node* tanpa adanya pusat administrasi dan infrastruktur kabel. Pada MANET, *mobile host* yang terhubung dengan *wireless* dapat bergerak bebas dan juga berperan sebagai *router*. MANET memiliki beberapa karakteristik yaitu di antaranya konfigurasi jaringan yang dinamis, *bandwidth* yang terbatas, keterbatasan daya untuk tiap-tiap operasi, keterbatasan keamanan dan setiap *node* pada MANET berperan sebagai *end-user* sekaligus sebagai router yang menghitung sendiri *route-path* yang selanjutnya akan dipilih [1].

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

AODV adalah protokol *routing on-demand* yang terdiri dari metode *unicast* dan *multicast*. Umumnya di AODV, cara korespondensi antar node sumber dan tujuan dihitung saat ada kebutuhan transfer data, kapanpun sumber ingin mengirim paket data ke tujuan, maka perlu dicari jalur *routing* antara *source* dan *destination.* AODV menemukan jalur terpendek dengan bantuan mekanisme perutean dan jalur perutean ini harus dipelihara sampai *routing link* terputus. Jalur dibangun dan dipertahankan dengan: *RREQs Control Packet (Route REQuests), RREP Control Packet (Route Reply), RERRs Control Packet (Route Errors) dan RREP-Back Control Packet (Route Reply Acknowledgement)*.

Control packet RREQ pertama kali dimulai oleh *source node* untuk menghitung jalur terpendek dalam multicast sort. *Control packet* RREP dikirimkan oleh *destination node* dan dijawab ke node terdekat saat menerima *control packet* RREQ untuk menemukan tipe jalur *unicast*. Umumnya dalam protokol *routing* AODV *Hello message* pertama kali digunakan untuk menjaga keseragaman jalur rute yang telah dikenal sebelumnya. Setiap node dalam jaringan nirkabel mempertahankan tabel *routing*nya sendiri yang membantu untuk menemukan jalur *routing*, setiap tabel routing berisi informasi seperti: alamat tujuan, informasi lengkap node tetangga terdekat, jumlah *hop* antara *source node* dan *destination node*, periode penghentian setelah bagian paket dibuang, protokol *routing* AODV menggunakan konsep *sequence number* yang ditugaskan ke semua paket, yang membatasi penyiaran *control packet* awal yang tidak perlu; *sequence number* ini memungkinkan penggunaan rute baru



Gambar 2.1 Mekanisme Route Discovery AODV

untuk perpindahan node [3]. Ilustrasi pencarian rute oleh AODV dapat dilihat pada Gambar 2.1

Pada setiap *node* yang menggunakan protokol AODV pasti memiliki sebuah *routing table* dengan *field* sebagai berikut:

- *Destination Address:* berisi alamat dari *node* tujuan.

- *Destination Sequence Number*: *sequence number* dari jalur komunikasi.
- *Next Hop*: alamat *node* yang akan meneruskan paket data.
- *Hop Count*: jumlah *hop* yang harus dilakukan agar paket dapat mencapai *node* tujuan.
- *Lifetime*: waktu dalam milidetik yang diperlukan *node* untuk menerima RREP.
- *Routing Flags*: status jalur. Terdapat tiga tipe status, yaitu *up* (valid), *down* (tidak valid) atau sedang diperbaiki.

Sebagai contoh proses *route discovery* dalam AODV, ilustrasi pada Gambar 2.2 menggambarkan bagaimana *source node*, yaitu *node* A mencari rute untuk menuju *destination node* yaitu *node* F. *Node* A akan membuat paket RREQ dan melakukan *broadcast* kepada semua *node* tetangganya (*neighbor node*). Jika *destination sequence number* yang terdapat pada paket RREQ sama atau lebih kecil dari yang ada pada *routing table* dan rute menuju *node* tujuan belum ditemukan, maka paket tersebut tidak akan dilanjutkan (*drop*). Jika *destination sequence number* pada RREQ lebih besar dibandingkan dengan yang terdapat pada *routing table*, maka *entry* pada *routing table* akan diperbarui dan paket tersebut akan diteruskan oleh *neighbor node* sekaligus membuat *reverse path* menuju *source node*. Paket RREQ akan diteruskan hingga mencapai *node* F. Kemudian, jika rute menuju *node* F sudah terbentuk di dalam *routing table* dan memiliki *routing flags* "*up*", maka *node* F akan mengirimkan paket RREP melalui rute tersebut menuju *node* .

Pada Tugas Akhir ini, digunakan *routing protocol* AODV yang akan diimplementasikan pada lingkungan MANET dengan beberapa skenario.

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

NS-2 adalah sebuah simulator jaringan yang dikembangkan di UC Berkely dan ditulis dalam bahasa C ++ dan OTcl yang

berorientasi objek. NS-2 adalah alat yang sangat umum digunakan untuk mensimulasikan jaringan area lokal dan interlokal. Ini mengimplementasikan protokol jaringan seperti TCP dan UPD; perilaku sumber lalu lintas seperti FTP, Telnet, Web, CBR dan VBR; Mekanisme manajemen antrian *router* seperti Drop Tail, RED dan CBQ; algoritma *routing* seperti Dijkstra, dan banyak lagi. NS-2 juga mengimplementasikan *multicasting* dan beberapa lapisan protokol MAC untuk simulasi LAN. Simulator ini merupakan *open source*, sehingga memungkinkan setiap orang untuk melakukan perubahan pada kode yang ada, selain menambahkan protokol dan fungsionalitas baru untuk itu. Hal ini yang membuat NS-2 lebih populer dibandingkan dengan aplikasi lainnya yang mana memudahkan pengguna untuk mengevaluasi fungsionalitas dan desain pada penelitian sebuah jaringan. Simulator dikembangkan dalam dua bahasa: C++ dan OTcl[1]. C++ digunakan untuk implementasi protokol yang terperinci seperti TCP. *TCL scripting* adalah aplikasi antarmuka untuk NS-2 yang digunakan untuk membuat pengaturan perintah dan antarmuka [4].

Pada Tugas Akhir ini, NS-2 digunakan untuk melakukan simulasi lingkungan MANET menggunakan protokol AODV yang *Hello Interval*nya sudah mengalami perubahan . *Trace file* yang dihasilkan oleh NS-2 juga digunakan untuk mengukur peforma *routing* protokol AODV yang sudah mengalami perubahan *Hello Interval*. NS-2 yang digunakan pada tugas akhir ini adalah NS-2 versi 2.35.

2.3.1 Instalasi

NS-2 membutuhkan beberapa *package* yang harus sudah ter*install* sebelum memulai instalasi NS-2. Untuk meng*install dependency* yang dibutuhkan dapat dilakukan dengan *command* yang ditunjukkan pada Gambar 2.

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

Gambar 2.2 Perintah untuk meng*install dependency* NS-2

Setelah meng*install dependency* yang dibutuhkan, ekstrak *package* NS-2 dan ubah baris kode ke-137 pada *file* ls.h di *folder* linkstate menjadi seperti pada Gambar 2.3.

```
1.  void eraseAll(){this->erase(baseMap::begin(), baseMap::end()); }
```

Gambar 2.3 Baris kode yang diubah pada file ls.h

Install NS-2 dengan menjalankan perintah ./install pada folder NS-2.

2.3.2 *Trace File*

Trace file merupakan *file* hasil simulasi yang dilakukan oleh NS-2 dan berisikan informasi detail pengiriman paket data. *Trace file* digunakan untuk menganalisis performa *routing protocol* yang disimulasikan. Detail penjelasan *trace file* ditunjukkan pada Tabel 2.1.

Tabel 2.1 Detail Penjelasan *Trace File* AODV

Kolom ke-	Penjelasan	Isi
1	*Event*	s : *sent* r : *received* f : *forwarded* D : *dropped*
2	*Time*	Waktu terjadinya *event*
3	ID *Node*	_x_ : identifier dari node yang melakukan *event*
4	*Layer*	Network layer tempat terjadi event AGT : *application* RTR : *routing* LL : *link layer*

		IFQ : *packet queue* MAC : *MAC* PHY : *physical*
5	*Flag*	--- : Tidak ada
6	*Sequence Number*	Nomor urut 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 asal 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*)

2.4 AWK

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

Pada tugas akhir ini penulis menggunakan AWK sebagai alat untuk membuat script dalam perhitungan *Packet Delivery Ratio* (PDR), *End-to-End Delay* (E2D), dan *Routing Overhead* (RO) dari hasil *trace* NS-2.[5]

2.5 *Hello Interval*

Hello Interval adalah jeda waktu yang diperlukan fungsi *sendHello*() pada AODV untuk mengecek node tetangga. Waktu yang dihasilkan dari rumus asli *Hello Interval* adalah antara 0.2 sampai 0.7 detik. Pada Tugas Akhir ini *Hello Interval* diubah menjadi statis dengan angka 5, 10, dan 15 detik.

2.6 Generator File *Node Movement*

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

Sebelum menjalankan program simulasi pengguna harus menjalankan program ‘setdest’. *Command line* ‘setdest’yang sudah di-*generate* menghasilkan *file* yang berisi jumlah *node* dan pergerakan dari *node* yang sudah dibuat dalam *file* berbentuk .tcl.

Selain pergerakan *node file* tersebut juga berisi tentang perpindahan rute.

Untuk membuat skenario *node-movement* yang terdiri dari 50 *node*, bergerak dengan kecepatan maksimum 5 m/s, simulasi akan berhenti setelah 100 detik dengan batas topologi yang diartikan sebagai 800 x 800 meter

2.7 File Traffic Connection Pattern

Random Traffic Connection memiliki dua buah tipe *traffic* yaitu TCP dan CBR. Keduanya dapat dibuat *menggunakan script traffic scenario generator*. *Script* ini dapat membantu kita untuk men-*generate* beban *traffic*. *Script* yang dimaksud di dalam sini adalah 'cbrgen.tcl'. program "cbrgen.tcl" diigunakan sesuai dengan *command line* pada gambar 2.5 dan dengan keterangan yang ditunjukan pada tabel 2.2.[6]

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

Gambar 2.4 Format *Command Line* 'cbrgen.tcl'

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

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

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

Gambar 2.5 Contoh C*ommand Line* ‘cbrgen.tcl’

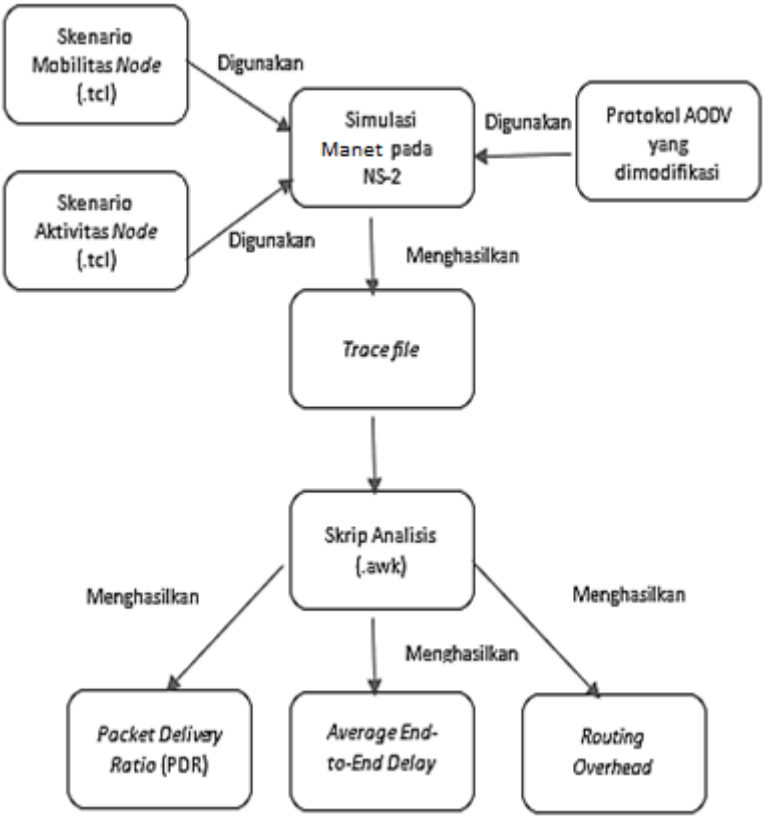
(Halaman ini sengaja dikosongkan)

BAB III
PERANCANGAN

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

3.1 Deskripsi Umum

Dalam tugas akhir ini dilakukan analisa pada routing protocol AODV. Analisa dilakukan dengan menggunakan MANET simulator generator dan traffic generator dengan perubahan *Hello Interval* untuk diujikan. Kemudian simulasi yang dibuat pada MANET simulator generator dan traffic generator dijalankan pada Network Simulator 2 dan akan menghasilkan trace file. Interval yang diujikan pada tugas akhir ini adalah interval 0.5, 1, 1.5, dan 2. Hasil simulasi digunakan untuk mencari kinerja dari *Hello Interval* pada simulasi *routing protocol* AODV yang dijalankan pada NS-2. Kemudian hasil simulasi dari NS-2 dianalisis dengan menggunakan skrip AWK untuk mendapatkan *Packet Delivery Ratio* (PDR), *Routing Overhead* (RO), *End-to-End Delay* (E2E), dan *Forwarded Route Request* (RREQ F). Analisis tersebut bertujuan untuk mengetahui dampak perubahan *Hello Interval* pada kinerja protokol AODV. Diagram rancangan simulasi dapat dilihat pada Gambar 3.1.



Gambar 3.1 Skema Perancangan

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

Tabel 3.1 Daftar Istilah

No.	Istilah	Penjelasan
1	AODV	Singkatan dari *Ad hoc On-demand Distance Vector*. Protokol yang digunakan pada Tugas Akhir ini.

No.	Istilah	Penjelasan
2	PDR	*Packet Delivery Ratio*. Salah satu metrik analisis yang diukur. Berupa rasio jumlah pengiriman paket yang terkirim.
3	E2E	*Average End-to-End Delay*. Jeda waktu yang diukur saat paket terkirim.
4	RO	*Routing Overhead*. Jumlah *control packet* yang terkirim
5	*Hello Interval*	*Hello Interval* adalah jeda waktu yang diperlukan fungsi *sendHello*() pada AODVuntuk mengecek node tetangga.

3.2 Perancangan Skenario

Skenario uji coba dalam Tugas Akhir ini dibuat dengan menggunakan mobility generation setelah itu penguji akan membuat koneksi dari skenario yang sudah ada dengan file traffic connection yang sudah ada pada NS-2. Pada tugas akhir ini skenario dibuat untuk melihat dampak perubahan *Hello Interval* dengan interval 0.5, 1, 1.5, dan 2. Sedangkan untuk koneksinya digunakan dua node untuk menentukan node pengirim dan node penerima paket.

3.2.1 Perancangan Skenario Node Movement (Mobility Generation)

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

Table 3.2 Penjelasan Skenario *node movement*

No.	Parameter	Spesifikasi
1	Jumlah *Node*	20,50,100

2	Waktu Simulasi	100 detik
3	Area	510m x 510m
4	Kecepatan Maksimal	-5m/s -10m/s -15m/s
5	Sumber *Traffic*	CBR
6	Waktu Jeda (Dalam Detik)	10
7	Ukuran Paket	512 bytes
8	*Rate* Paket	0.25 paket per detik
9	Jumlah maksimal koneksi	1
10	Model mobilitas yang digunakan	*Random way point*

3.2.2 *Traffic-Connection Pattern*

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

Table 3.3 Penjelasan *Traffic-connection pattern*

No.	Parameter	Spesifikasi
1	-type cbr|tcp	CBR
2	-nn *nodes*	10
3	-s *seed*	1.0
4	-mc *connection*	8
5	-rate *rate*	0.25

3.3 Perancangan Simulasi pada NS-2

Simulasi MANETs pada NS-2 dilakukan dengan menggabungkan *file* skenario yang telah dibuat dan *file* skrip dengan ekstensi .tcl yang berisikan konfigurasi lingkungan simulasi.

Kode yang diubah diantaranya adalah interval pada *Hello Interval* pada fungsi *sendHello*() AODV. Pada saat simulasi NS-2 dijalankan, maka akan diketahui performa dari *Hello Interval*.

3.4 Perancangan Metrik Analisis

Berikut ini merupakan parameter – parameter yang akan dianalisis pada Tugas Akhir ini untuk dapat mengetahui performa dari *Hello Interval* pada AODV:

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 Rumus untuk menghitung PDR dapat dilihat pada persamaan 3.1.

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

Keterangan:

PDR = *Packet Delivery Ratio*

received = banyak paket data yang diterima

sent = banyak paket data yang dikirimkan

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

Average End-to-End Delay dihitung berdasarkan rata-rata *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 rata – rata E2E, yang dapat dihitung dengan persamaan 3.2.

$$E2E = \frac{\sum_{m=1}^{recvnum} CBRRecvTime - CBRSentTime}{recvnum} \quad (3.2)$$

Keterangan:

E2E = *End-to-End Delay*

CBRRecvTime = Waktu *node* asal mengirimkan paket

CBRSentTime = Waktu *node* tujuan menerima paket

recvnum = 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 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.

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

BAB IV
IMPLEMENTASI

Pada bab ini akan dibahas mengenai implementasi dari perancangan yang sudah dilakukan pada bab sebelumnya. Implementasi berupa kode sumber untuk membangun program.

4.1 Implementasi Skenario

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

4.1.1 Skenario File Node-Movement(Mobility Generation)

Dalam iimplementasi yang dilakukan pada pembuatan skenario dengan *mobility generation* menggunakan *tools generate default* yang sudah disediakan oleh NS-2 ketika kita melakukan installasi sebelumnya yaitu 'setdest'. Skenario yang sudah di-*generate* ini akan digunakan untuk setiap simulasi yang dilakukan beradasarkan kecepatan maksimal yang berbeda beda. Algoritma yang digunakan untuk membuat skenario ini adalah *Random Way Point* sehinnga penempatan *node node* yang ada akan bersifat acak. . *Format command line* pada Gambar 4.1 yang digunakan untuk menghasilkan gerakan acak pada node adalah sebagai berikut:

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

Gambar 4.1 Format kode ‘setdest’

Ketentuan-ketentuan yang diujicobakan pada skenario ini adalah versi ‘setdest’ simulator yaitu , jumlah node dalam scenario A yaitu

20, scenario B 50, dan skenario 100; waktu jeda (*pause time*) yaitu 0 detik,kecepatan maksimalnya sebesar 10m/s, waktu simulasi yaitu 100 detik. Kemudian file mobilitas yang dihasilkan disimpan dalam direktori " ~ns/indep-utils/cmu-scen-gen/setdest/". Dan untuk setiap jumlah node tersebut dibuat 10 buah file untuk satu protokol routing.

4.1.2 File *traffic-connection pattern*

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

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

Gambar 4.2 Format kode 'cbrgen.tcl'

Pada Gambar 4.3 merupakan bentuk implementasi untuk menjalankan cbrgen.tcl untuk membuat file koneksi CBR diantara 19 node memiliki maksimal 10 koneksi dengan nilai seed 1.0 dan jumlah paket per detik sebanyak 0.25 yang disimpan dalam cbr.txt yang nantinya akan digunakan pada saat simulasi NS-2.

```
1. ns cbrgen.tcl -type cbr -nn 19 -seed 1.0 -mc 10 -rate 0.25 > cbrtest.txt
```

Gambar 4.3 Implementasi kode 'cbrgen.tcl'.

4.2 Implementasi Perubahan *Interval* pada Fungsi *sendHello* untuk Menentukan *HelloInterval*

Pada Tugas Akhir ini dilakukan pengaturan pada *routing protocol* AODV dengan mengubah fungsi *sendHello*().Hal tersebut dilakukan dengan cara menentukan *HelloInterval* dari yang semula rumus asli pada AODV menjadi angka statis 0.5, 1, 1.5 dan 2.

Kode implementasi dari routing protocol AODV pada NS-2 versi 2.35 berada pada direktori ns-2.35/aodv. Pada direktori tersebut terdapat beberapa file diantaranya seperti aodv.cc, aodv.h dan sebagainya. Pada Tugas Akhir ini, *file* aodv.cc yang terdapat dalam *folder* ns-2.35/aodv di*setting* untuk mengetahui performa dari *Hello Interval*. Pada bagian ini akan dijelaskan langkah – langkah dalam mengimplementasikan perubahan *Hello Interval* pada *routing protocol* AODV.

```
1.  agent->sendHello();

        double interval = 0.5;

        assert(interval >= 0);

    Scheduler::instance().schedule(this, &intr, interval);
```

Gambar 4.4 Potongan Kode Fungsi *sendHello*()

4.3 Implementasi Simulasi pada NS-2

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

Pada kode sumber 4.5 menunjukan skrip konfigurasi awal parameter paremeter yang diberikan untuk menjalakan simulasi MANET pada NS-2. Isi dari parameternya adalah sebegai berikut.

Table 4.1 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	1500
8	Nilai Y	1500
9	Nilai seed	1.0
10	Routing Protocol	AODV
11	File traffic connection	“cbr020-1.tcl”
12	File node movement	“setdest20node001 ”

```
2.  set val(chan)         Channel/WirelessChannel;
3.  set val(prop)         Propagation/TwoRayGround;
4.  set val(netif)        Phy/WirelessPhy;
5.  set val(mac)          Mac/802_11;
6.  set val(ifq)          Queue/Droptail/PriQueue;
7.  set val(ll)           LL;
8.  set val(ant)          Antenna/OmniAntenna;
9.  set val(ifqlen)       1000;
10. set val(nn)           20;
11. set val(rp)           AODV;
12. set opt(x)            1500;
13. set opt(y)            1500;
14. set val(stop)           200;
15. set val(seed)           1.0;
16. set val(cp)           "cbr020.tcl";
17. set val(sc)           "setdest20node001.tcl";
```

Gambar 4.5 Konfigurasi parameter pada NS-2

Pada Gambar 4.6 merupakan pengaturan dari *transmission range* yang digunakan pada simulasi. Nilai yang diubah ialah *RXThresh_(receiver Sensitivity Threshold).* Nilai 5.57189e-11 pada

variabel tersebut memiliki artian bahwa *range* atau jangkauan yang dapat dicapai ialah sejauh 400 meter.

```
1.  Phy/WirelessPhy set RXThresh_ 5.57189e-11;
```

Gambar 4.6 Pengaturan *Transmission range*

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

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

Gambar 4.7 Pengaturan variabel global pada NS-2

Skrip yang ditunjukan pada Gambar 4.7 merupakan skrip untuk pengaturan variabel global yang diawali dengan set ns merupakan kode untuk pembuatan simulator baru. Set *tracefd* dan *set namtrace* merupakan pengaturan untuk menentukan nama dari *trace file* dan *file network* yang akan dihasilkan dan disimpan setelah simulasi selesai dilaksanakan. *Set topo* merupakan pengaturan untuk objek topografi berdasarkan pada luas koordinat yang telah dikonfigurasi sebelumnya. *Set god* dan *node config – channelType* merupakan konfigurasi yang dilakukan pada *node-node* yang akan dibuat. Terakhir dilakukan perulangan untuk membuat pergerakan dari *node node*. *Node-node* yang dibuat tidak dapat melakukan pergerakan secara acak karena pergerakan *node* merupakan *trace file* yang dihasilkan oleh *mobility generator*.

Skrip pada Gambar 4.8 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 output.tr* pada potongan skrip tersebut, akan dipanggil file skenario *node-movement* (*mobility generation*) dan *traffic-connection pattern* kemudian pengiriman paket data dimulai pada detik ke-0 dan diberhentikan pada detik ke 100 seperti yang telah di konfigurasi sebelumnya.

```
1.  puts "Loading Conneciton Pattern ...."
2.  source $val(cp)
3.
```

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

#define node initial position in nam

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


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

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

puts $tracefd "M 0.0 nn $val(nn) x $opt(x) y $opt(y) rp $val(rp)"
puts $tracefd "M 0.0 sc $val(sc) cp $val(cp) seed $val(seed)"
puts $tracefd "M 0.0 prop $val(prop) ant $val(ant)
puts "Starting Simulation...."
$ns_ run
```

Gambar 4.8 Menginisiasi penempatan awal node

4.4 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, dapat dilakukan analisis performa *routing protocol* dengan mengukur beberapa metrik. Pada Tugas Akhir ini, metrik yang akan dianalisis adalah PDR, E2E, dan RO.

4.4.1 Implementasi *Packet Delivery Ratio* (PDR)

Pada subbab 2.3.2 telah ditunjukkan contoh struktur data *event* yang dicatat dalam *trace file* oleh NS-2. Kemudian, pada

persamaan 3.1 telah dijelaskan bagaimana menghitung PDR. Skrip awk untuk menghitung PDR berdasarkan kedua informasi tersebut dapat dilihat pada lampiran A.4 Kode Skrip AWK *Packet Delivery Ratio*.

PDR didapatkan dengan cara menghitung setiap baris terjadinya *event* pengiriman dan penerimaan paket data yang dikirim melalui agen pada *trace file*. Skrip menyaring setiap baris yang mengandung *string* AGT karena kata kunci tersebut menunjukan *event* yang berhubungan dengan paket komunikasi data. Penghitungan dilakukan dengan menjumlahkan paket yang dikirimkan dan paket yang diterima dengan menggunakan karakter pada kolom pertama sebagai *filter*. Kolom pertama menunjukan event yang terjadi dari sebuah paket. Setelah itu nilai PDR dihitung dengan cara persamaan 3.1. Pseudocode untuk menghitung PDR dapat dilihat pada Gambar 4..

```
sent = 0
received = 0
for i = 1 to the number of rows
     if in a row contains "s" and AGT then
          sent++
     else if in a row contains "r" and AGT then
          received++
     end if
pdr = received / sent
```

Gambar 4.9 Pseudocode untuk Perhitungan Rata-Rata PDR

Contoh perintah pengeksekusian skrip awk untuk menganalisis *trace file* adalah `awk -f pdr.awk result.tr`.

4.4.2 Implementasi *Average End-to-End Delay (E2E)*

Skrip awk untuk menghitung E2E dapat dilihat pada lampiran A.5 Kode Skrip AWK Rata-Rata *End-to-End Delay*.

Dalam perhitungan E2E, langkah yang digunakan untuk mendapatkan E2E hampir sama dengan ketika mencari PDR, hanya saja yang perlu diperhatikan adalah waktu dari sebuah *event* yang tercatat pada kolom ke-2 dengan *filter event* pada kolom ke-4 adalah layer AGT dan *event* pada kolom pertama guna membedakan paket dikirim atau diterima. Setelah seluruh baris yang memenuhi didapatkan, akan dihitung *delay* dari paket dengan mengurangi waktu dari paket diterima dengan waktu dari paket dikirim dengan syarat memiliki *id* paket yang sama.

Setelah mendapatkan *delay* paket, langkah selanjutnya adalah dengan mencari rata-rata dari *delay* tersebut dengan menjumlahkan semua *delay* paket dan membaginya dengan jumlah paket. *Pseudocode* untuk menghitung rata-rata E2E dapat dilihat pada Gambar 4.10.

```
sum_delay = 0
counter = 0
for i = 1 to the number of rows
     counter++
     if layer == AGT and event == s then
      start_time[packet_id] = time
     else if layer == AGT and event == r then
      end_time[packet_id] = time
     end if
     delay[packet_id] = end_time[packet_id] -
start_time[packet_id]
     sum_delay += delay[packet_id]
e2e = sum_delay / counter
```

Gambar 4.10 Pseudocode untuk Perhitungan Rata-Rata E2E

Contoh perintah pengeksekusian skrip awk untuk menganalisis *trace file* adalah `awk -f e2e.awk result.tr`.

4.4.3 Implementasi *Routing Overhead* (RO)

Seperti yang telah dijelaskan sebelumnya, *routing overhead* merupakan jumlah dari paket kontrol *routing* baik itu RREQ, RREP, maupun RERR. Dengan begitu, untuk mendapatkan RO yang perlu dilakukan adalah menjumlahkan tiap paket dengan *filter event sent* pada kolom pertama dan *event layer* RTR pada kolom ke-4.

Perhitungan RO telah dijelaskan pada persamaan 3.3. Skrip AWK untuk menghitung RO dapat dilihat pada lampiran A.6 Kode Skrip AWK *Routing Overhead*. *Pseudocode* untuk menghitung RO dapat dilihat pada Gambar 4..

```
ro = 0
for i = 1 to the number of rows
     if in a row contains "s" and RTR then
        ro++
     end if
```

Gambar 4.11 Pseudocode untuk Perhitungan *Routing Overhead*

Contoh perintah pengeksekusian skrip awk untuk menganalisis *trace file* adalah `awk -f ro.awk result.tr`

BAB V
UJI COBA DAN EVALUASI

Pada bab ini akan dilakukan tahap ujicoba 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.

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**
CPU	AMD FX-7600P R7, 12 Compute Cores 4C +8G 2.70GHz
Sistem Operasi	Ubuntu 16.04 LTS
Linux Kernel	Linux kernel 4.4
Memori	4.0 GB

Adapun versi perangkat lunak yang digunakan dalam Tugas Akhir ini adalah NS-2 versi 2.35 untuk simulasi skenario MANET.

Parameter lingkungan uji coba yang digunakan pada NS-2 dapat diihat pada Tabel 5.2. Pengujian dilakukan dengan menjalankan skenario yang disimulasikan pada NS-2. Dari simulasi tersebut dihasilkan sebuah *trace file* dengan ekstensi .tr yang akan dianalisis dengan bantuan skrip awk untuk mendapatkan PDR, E2E, RO, dan RREQ F menggunakan kode yang terdapat pada lampiran A.4 Kode Skrip AWK *Packet Delivery Ratio*, A.5 Kode Skrip AWK Rata-Rata *End-to-End Delay*, A.6 Kode Skrip AWK *Routing Overhead*, dan A.7 Kode Skrip AWK *Forwarded Route Request.*

Tabel 5.2 Lingkungan Uji Coba

No.	**Parameter**	**Spesifikasi**
1	Network simulator	NS-2.35

2	Routing protocol	AODV
3	Waktu simulasi	200 detik
4	Area simulasi	1500 m x 1500 m
5	Jumlah *Node*	20, 50, 100
6	Radius transmisi	400m
7	Kecepatan maksimum	10 m/s
8	Protokol MAC	IEEE 802.11p
9	Model Propagasi	*Two-ray ground*

5.2 Hasil Uji Coba

Pengujian pada skenario digunakan untuk melihat perbandingan PDR, RO, dan E2E antara *routing protocol* AODV asli dan AODV yang telah mengalami perubahan pada *Hello Interval* dalam pemilihan *node* yang dapat menerima paket *route request*.

Pengambilan data uji PDR, RO, dan E2E pada skenario dilakukan sebanyak 10 kali dengan skenario mobilitas *random* dengan luas area 1500 m x 1500 m dan *node* sebanyak 20 untuk lingkungan yang jarang, 50 *node* untuk lingkungan yang sedang, dan 100 *node* untuk lingkungan yang padat dilakukan pada kecepatan 10 m/s. Untuk uji coba setiap lingkungan menggunakan interval yang berbeda-beda untuk mencari nilai interval yang terbaik dari hasil skenario. Interval waktu yang digunakan adalah 0.5 detik, 1 detik, 15 detik, dan 2 detik.

Pada Tugas Akhir yang terdahulu sudah dilakukan penelitian terhadap pencarian *node* tetangga. Maka pada Tugas Akhir ini berfokus pada waktu yang diperlukan untuk pencarian *node* tetangga. Pada kasus ini maka digunakan waktu yang sudah ditentukan intervalnya. Hasil simulasi dapat dilihat pada Tabel 5.3, Tabel 5.4, Tabel 5.5, dan Tabel 5.6.

Tabel 5.3 Hasil Rata - Rata PDR

Jumlah *Node*	AODV Asli	AODV dengan perubahan *Hello Interval*				Perbedaan			
		0.5s	1s	1.5s	2s	0.5s	1s	1.5s	2s
20	0.87	0.85	0.86	0.86	0.85	0.02	0.01	0.01	0.02
50	0.93	0.91	0.84	0.90	0.90	0.02	0.09	0.04	0.03
100	0.64	0.69	0.63	0.69	0.65	0.05	0.01	0.05	0.01

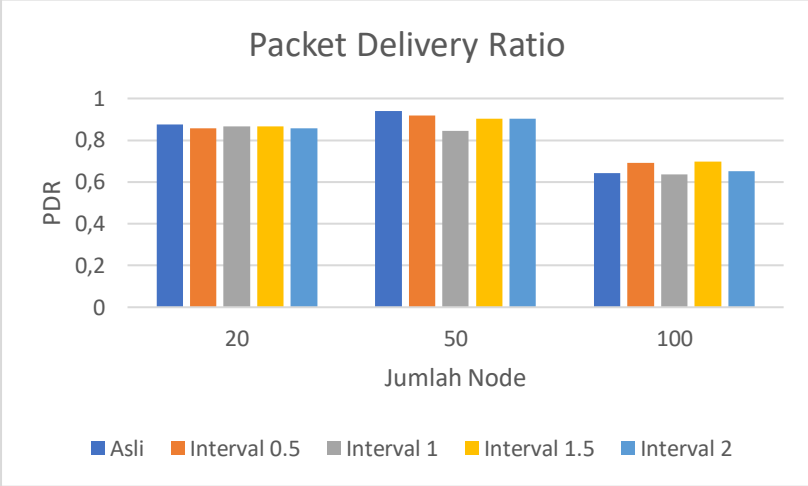
Tabel 5.4 Hasil Rata - Rata E2E

Jumlah *Node*	AODV Asli	AODV dengan perubahan *Hello Interval*				Perbedaan			
		0.5s	1s	1.5s	2s	0.5s	1s	1.5s	2s
20	92.81	100.53	78.24	68.44	75.93	7.72	14.56	24.36	16.87
50	101.12	103.22	122.46	125.43	127.72	2.09	21.33	24.30	26.59
100	249.66	240.67	255.77	245.27	262.58	8.98	6.11	4.38	12.92

Tabel 5.5 Hasil Rata - Rata RO

Jumlah *Node*	AODV Asli	AODV dengan perubahan *Hello Interval*				Perbedaan			
		0.5s	1s	1.5s	2s	0.5s	1s	1.5s	2s
20	4,784	8,717	5,596	4,118	3,724	3,933	812	665	1,059
50	12,713	22,602	15,176	11,371	9,981	9,888	2,462	1,342	2,732
100	29,037	43,580	32,638	25,778	23,177	14,542	3,600	3,258	5,860

5.2.1 Analisa Packet Delivery Ratio (PDR)



Gambar 5.1 Grafik PDR

Berdasarkan grafik pada Gambar 5.1, dapat dilihat bahwa *routing protocol* AODV asli dan AODV dengan interval 0.5 detik, 1 detik, 1.5 detik, dan 2 detik mengalami perubahan yang fluktuaktif. Pada lingkungan yang jarang dengan *node* berjumlah 20 dan dibandingkan dengan hasil PDR AODV asli, AODV dengan interval 0.5 detik menghasilkan perbedaan selisih PDR sebesar 0.019, dimana terjadi penurunan sebesar 2.16 %, pada AODV dengan interval 1 detik menghasilkan perbedaan selisih PDR sebesar 0.01038, dimana terjadi penurunan sebesar 1.18 %, pada AODV dengan interval 1.5 detik menghasilkan perbedaan selisih PDR sebesar 0.01108, dimana terjadi penurunan sebesar 1.26 %, dan pada AODV dengan interval 2 detik menghasilkan perbedaan selisih PDR sebesar 0.01754, dimana terjadi penurunan sebesar 2 %. Berdasarkan hal tersebut, routing protocol AODV asli berhasil mengungguli *routing protocol* AODV pada interval 0.5, 1, 1.5, dan 2.

Pada lingkungan yang sedang dengan *node* berjumlah 50 dan dibandingkan dengan hasil PDR AODV asli, AODV dengan interval

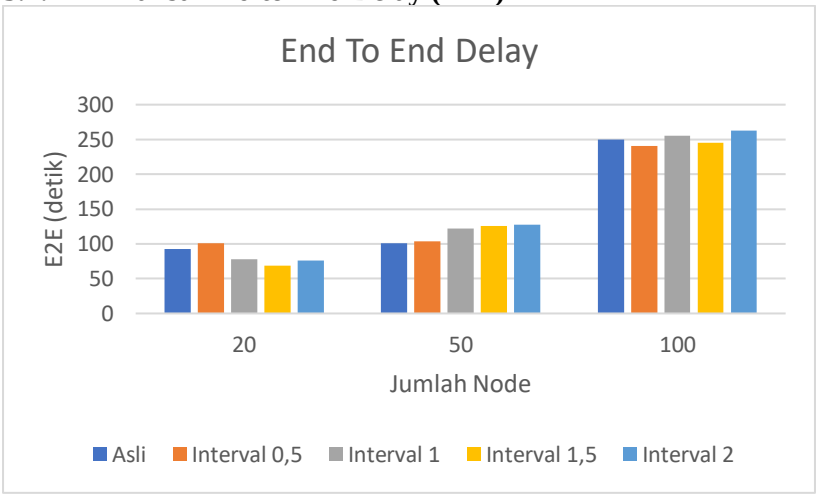
0.5 detik menghasilkan perbedaan selisih PDR sebesar 0.0214, dimana terjadi penurunan sebesar 2.27 %, pada AODV dengan interval 1 detik menghasilkan perbedaan selisih PDR sebesar 0.09347, dimana terjadi penurunan sebesar 9.95 %, pada AODV dengan interval 1.5 detik menghasilkan perbedaan selisih PDR sebesar 0.03722, dimana terjadi penurunan sebesar 3.94 %, dan pada AODV dengan interval 2 detik menghasilkan perbedaan selisih PDR sebesar 0.03542, dimana terjadi penurunan sebesar 3.77 %. Berdasarkan hal tersebut, routing protocol AODV asli berhasil mengungguli *routing protocol* AODV pada interval 0.5, 1, 1.5, dan 2.

Pada lingkungan yang padat dengan *node* berjumlah 100 dan dibandingkan dengan hasil PDR AODV asli, AODV dengan interval 0.5 detik menghasilkan perbedaan selisih PDR sebesar 0.05047, dimana terjadi penurunan sebesar 7.86 %, pada AODV dengan interval 1 detik menghasilkan perbedaan selisih PDR sebesar 0.00542, dimana terjadi kenaikan sebesar 0.84 %, pada AODV dengan interval 1.5 detik menghasilkan perbedaan selisih PDR sebesar 0.05534, dimana terjadi penurunan sebesar 8.62 %, dan pada AODV dengan interval 2 detik menghasilkan perbedaan selisih PDR sebesar 0.01068, dimana terjadi kenaikan sebesar 1.66 %. Berdasarkan hal tersebut, *routing protocol* AODV dengan interval 1,5 detik berhasil mengungguli *routing protocol* AODV dengan interval 0.5 detik, *routing protocol* AODV dengan interval 1 detik, *routing protocol* AODV dengan interval 2 detik, dan *routing protocol* AODV asli.

Pada lingkungan *node* dengan tingkat kepadatan jarang, AODV asli menghasilkan *Packet Delivery Ratio* (PDR) yang paling tinggi dibanding AODV yang lain yaitu sebesar 0.87593. Pada lingkungan *node* dengan tingkat kepadatan sedang, AODV asli menghasilkan *Packet Delivery Ratio* (PDR) yang paling tinggi dibanding AODV yang lain yaitu sebesar 0.93893. Pada lingkungan *node* dengan tingkat kepadatan padat, AODV dengan interval 1.5 detik menghasilkan *Packet Delivery Ratio* (PDR) yang paling tinggi dibanding AODV yang lain yaitu sebesar 0.64132. Berdasarkan hasil simulasi pada ketiga lingkungan tersebut, dapat dilihat bahwa tidak dapat disimpulkan mana yang lebih unggul karena pada setiap

percobaan dihasilkan *Packet Delivery Ratio* (PDR) yang berbeda di lingkungan *node* dengan tingkat kepadatan jarang, sedang, maupun padat.

5.2.2 Analisa End to End Delay (E2E)



Gambar 5.2 Grafik E2E

Berdasarkan grafik pada Gambar 5.2 dapat dilihat bahwa rata-rata E2E antara *routing protocol* AODV asli dan AODV dengan interval 0.5 detik, 1 detik, 1.5 detik, dan 2 detik mengalami perubahan yang fluktuatif. Pada lingkungan yang jarang dengan jumlah 20 *node* dan dibandingkan dengan AODV asli, AODV dengan interval 0.5 detik menghasilkan perbedaan selisih *end-to-end delay* sebesar 7.7252 ms dimana AODV asli unggul dalam hal ini, pada AODV dengan interval 1 detik menghasilkan perbedaan selisih *end-to-end delay* sebesar 14.56724 ms dimana AODV dengan interval 1detik unggul dalam hal ini, pada AODV dengan interval 1.5 detik menghasilkan perbedaan selisih *end-to-end delay* sebesar 24.36828 ms dimana AODV dengan interval 1.5 detik unggul dalam hal ini, dan pada AODV dengan interval 2 detik menghasilkan perbedaan

selisih *end-to-end delay* sebesar 16.87633 ms dimana AODV dengan interval 2 detik unggul dalam hal ini.

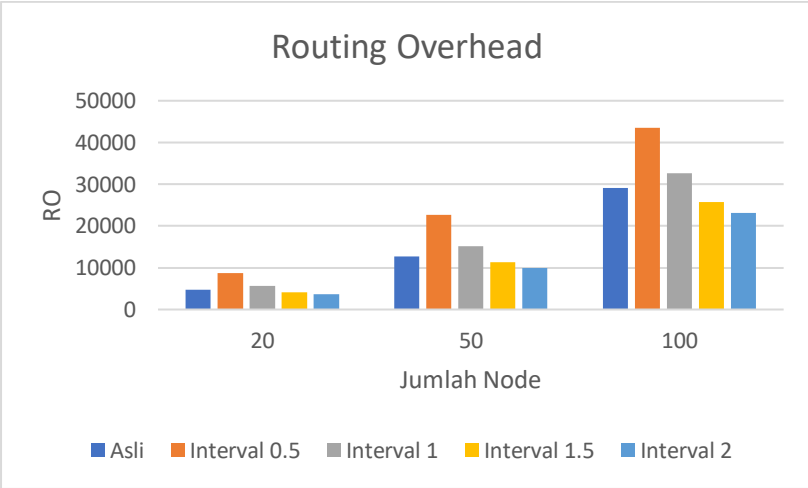
Pada lingkungan yang sedang dengan jumlah 50 *node* dan dibandingkan dengan AODV asli, AODV dengan interval 0.5 detik menghasilkan perbedaan selisih *end-to-end delay* sebesar 2.09957 ms dimana AODV asli unggul dalam hal ini, pada AODV dengan interval 1 detik menghasilkan perbedaan selisih *end-to-end delay* sebesar 21.33615 ms dimana AODV asli unggul dalam hal ini, pada AODV dengan interval 1.5 detik menghasilkan perbedaan selisih *end-to-end delay* sebesar 24.30789 ms dimana AODV asli unggul dalam hal ini, dan pada AODV dengan interval 2 detik menghasilkan perbedaan selisih *end-to-end delay* sebesar 26.59988 ms dimana AODV asli unggul dalam hal ini.

Pada lingkungan yang padat dengan jumlah 100 *node* dan dibandingkan dengan AODV asli, AODV dengan interval 0.5 detik menghasilkan perbedaan selisih *end-to-end delay* sebesar 8.9878 ms dimana AODV dengan interval 0.5 detik unggul dalam hal ini, pada AODV dengan interval 1 detik menghasilkan perbedaan selisih *end-to-end delay* sebesar 6.1146 ms dimana AODV asli unggul dalam hal ini, pada AODV dengan interval 1.5 detik menghasilkan perbedaan selisih *end-to-end delay* sebesar 4.3844 ms dimana AODV dengan interval 1.5 unggul dalam hal ini, dan pada AODV dengan interval 2 detik menghasilkan perbedaan selisih *end-to-end delay* sebesar 12.9266 ms dimana AODV asli unggul dalam hal ini

Berdasarkan hasil simulasi pada ketiga lingkungan tersebut, dapat dilihat bahwa jumlah *node* yang sedikit atau lingkungan jarang menghasilkan rata rata *End to End Delay* yang lebih bagus atau lebih sedikit daripada di lingkungan dengan jumlah *node* yang sedang maupun yang padat baik untuk *routing protocol* AODV asli maupun *routing protocol* AODV yang telah mengalami perubahan *Hello Interval*. Dapat dilihat pula bahwa AODV asli menghasilkan *End to End Delay* yang lebih bagus atau dalam hal ini lebih rendah daripada rata-rata AODV dengan perubahan *Hello Interval* dengan jumlah selisih *End to End Delay* yang cukup signifikan. Hasil rata – rata E2E pada AODV asli yaitu 147,86 ms. Hal ini dikarenakan waktu *delay*

tergantung dari rata – rata waktu paket yang terkirim. Semakin banyak paket yang terkirim, maka semakin beragam *delay*nya.

5.2.3 Analisa Routing Overhead (RO)



Gambar 5.3 Grafik Routing Overhead

Berdasarkan grafik pada Gambar 5.3, dapat dilihat bahwa *routing protocol* AODV dengan interval 0.5 detik, 1 detik, 1.5 detik, 2 detik dan juga *routing protocol* AODV asli mengalami perubahan yang signifikan. Pada lingkungan yang jarang dengan jumlah 20 *node* dan dibandingkan dengan AODV asli, AODV dengan interval 0.5 detik menghasilkan perbedaan selisih RO sebesar 3933.1, dimana terjadi kenaikan RO sebesar 82.21 %, sedangkan pada AODV dengan interval 1 detik menghasilkan perbedaan selisih RO sebesar 812.2, dimana terjadi kenaikan RO sebesar 16.97 %, pada AODV dengan interval 1.5 detik menghasilkan perbedaan selisih RO sebesar 665.3 dimana terjadi penurunan RO sebesar 13.90 %, dan pada AODV dengan interval 2 detik menghasilkan perbedaan selisih RO sebesar 1059.6 dimana terjadi penurunan RO sebesar 22.14 %. Berdasarkan hasil tersebut, maka *routing protocol* AODV dengan interval 2 detik

unggul dalam hal RO tersebut karena menghasilkan RO yang lebih rendah dari AODV asli.

Pada lingkungan yang sedang dengan jumlah 50 *node* dan dibandingkan dengan AODV asli, AODV dengan interval 0.5 detik menghasilkan perbedaan selisih RO sebesar 9888.9, dimana terjadi kenaikan RO sebesar 77.78 %, sedangkan pada AODV dengan interval 1 detik menghasilkan perbedaan selisih RO sebesar 2462.4, dimana terjadi kenaikan RO sebesar 19.36 %, pada AODV dengan interval 1.5 detik menghasilkan perbedaan selisih RO sebesar 1342.4 dimana terjadi penurunan RO sebesar 10.55 %, dan pada AODV dengan interval 2 detik menghasilkan perbedaan selisih RO sebesar 2732.1 dimana terjadi penurunan RO sebesar 21.48 %. Berdasarkan hasil tersebut, maka *routing protocol* AODV dengan interval 2 detik unggul dalam hal RO tersebut karena menghasilkan RO yang lebih rendah dari AODV asli.

Pada lingkungan yang padat dengan jumlah 100 *node* dan dibandingkan dengan AODV asli, AODV dengan interval 0.5 detik menghasilkan perbedaan selisih RO sebesar 14542.6, dimana terjadi kenaikan RO sebesar 50.08 %, sedangkan pada AODV dengan interval 1 detik menghasilkan perbedaan selisih RO sebesar 3600.7, dimana terjadi kenaikan RO sebesar 12.40 %, pada AODV dengan interval 1.5 detik menghasilkan perbedaan selisih RO sebesar 3258.7 dimana terjadi penurunan RO sebesar 11.22 %, dan pada AODV dengan interval 2 detik menghasilkan perbedaan selisih RO sebesar 5860.4 dimana terjadi penurunan RO sebesar 20.18 %. Berdasarkan hasil tersebut, maka *routing protocol* AODV dengan interval 2 detik unggul dalam hal RO tersebut karena menghasilkan RO yang lebih rendah dari AODV asli.

Jika ketiga lingkungan tersebut dibandingkan dapat dilihat bahwa dengan jumlah *node* yang lebih sedikit atau pada lingkungan dengan *node* yang jarang, menghasilkan RO yang lebih sedikit daripada di lingkungan dengan jumlah *node* yang sedang maupun yang padat baik untuk *routing protocol* AODV asli maupun *routing protocol* AODV yang telah mengalami perubahan *Hello Interval*. AODV dengan interval 2 detik berhasil mengungguli AODV asli,

AODV dengan interval 0.5 detik, AODV dengan interval 1 detik, dan AODV dengan interval 1.5 detik dengan nilai rata – rata kenaikan RO pada AODV yang mengalami perubahan *Hello Interval* adalah sebesar 13.28 %. Dapat dilihat pula bahwa AODV dengan interval 2 detik menghasilkan RO yang lebih bagus atau dalam hal ini lebih rendah daripada AODV asli

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 diperoleh pada uji coba dan evaluasi Tugas Akhir ini adalah sebagai berikut:

- Dampak perubahan *Hello Interval* terhadap performa protokol AODV adalah rata – rata kenaikan *Routing Overhead* (RO) sebesar 13.28 %, rata – rata kenaikan *End to End Delay* sebesar 1.99 %, dan rata - rata penurunan *Packet Delivery Ratio* (PDR) sebesar 0.77 %.
- AODV dengan hasil *Packet Delivery Ratio* (PDR) terbaik adalah AODV dengan interval 0.5 detik dengan rata-rata hasil kenaikan *Packet Delivery Ratio* (PDR) sebesar 0.01 %.
- AODV dengan hasil *End-to-End Delay* terbaik adalah AODV Asli.
- AODV dengan hasil *Routing Overhead* (RO) terbaik adalah AODV dengan interval 2 detik dengan rata-rata hasil penurunan *Routing Overhead* (RO) sebesar 21.26 %.
- Aspek yang menjadi lebih baik ketika seiring dengan bertambahnya lama interval adalah *Routing Overhead.*

6.2 Saran

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

1. Lebih banyak uji coba yang dilakukan untuk mendapatkan hasil yang lebih akurat.

2. Menambah parameter dan skenario uji coba sehingga membuat hasil semakin variatif untuk dianalisis.
3. Menambahkan interval waktu yang berbeda untuk memberikan hasil yang semakin bervariasi.

DAFTAR PUSTAKA

[1] M. Affandes, "3. Mobile Ad Hoc Network (MANET)," 30 October 2011. [Online].

[2] M. N. Alslaim, H. A. Alaqel and S. S. Zaghloul, "A Comparative Study of MANET Routing Protocols," *IEEE,* p. 5, 2014.

[3] S. Biradar and P. Kulkarni, "An Improved Quality of Service Using R-AODV Protocol in MANETs," *Third International Conference 3rd on Artificial Intelligence Modelling and Simulation,* p. 6, 2015.

[4] M. Ali, M. Welzl, A. Adnan and F. Nadeem, "Using the NS-2 Network Simulator for Evaluating Network on Chips (NoC)," *2nd International Conference on Emerging Technologies,* p. 7, 2006.

[5] Magista Bella P, Radityo Anggoro, dan Muchammad Husni"Analisis Kinerja Perubahan Hello Interval Pada AODV di Lingkungan VANET".Surabaya, Institut Teknologi Sepuluh Nopember, 2018.

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

(Halaman ini sengaja dikosongkan)

LAMPIRAN

A.1 Kode Fungsi sendHello()

```
void
HelloTimer::handle(Event*) {

  #ifdef DEBUG
  FILE *fp = fopen("trace.log","a+");
  fprintf(fp, "HelloTimer::handle\n");
  fclose(fp);
  #endif

   agent->sendHello();
   double interval = 0.5;
   assert(interval >= 0);
   Scheduler::instance().schedule(this,
&intr, interval);
}
```

```
void
HelloTimer::handle(Event*) {

#ifdef DEBUG
FILE *fp = fopen("trace.log","a+");
fprintf(fp, "HelloTimer::handle\n");
fclose(fp);
#endif

agent->sendHello();
double interval = 1;
assert(interval >= 0);
Scheduler::instance().schedule(this,
&intr, interval);
}
```

```
void
HelloTimer::handle(Event*) {

  #ifdef DEBUG
  FILE *fp = fopen("trace.log","a+");
  fprintf(fp, "HelloTimer::handle\n");
  fclose(fp);
  #endif

   agent->sendHello();
   double interval = 1.5;
   assert(interval >= 0);
   Scheduler::instance().schedule(this,
&intr, interval);
}
```

A.2 Kode Skenario NS-2

```
set val(chan)   Channel/WirelessChannel;
set val(prop)   Propagation/TwoRayGround;
set val(netif)  Phy/WirelessPhy;
set val(mac)    Mac/802_11;
set val(ifq)    Queue/DropTail/PriQueue;
set val(ll)     LL;
set val(ant)    Antenna/OmniAntenna;
set opt(x)      1500;
set opt(y)      1500;
set val(ifqlen) 1000;
set val(nn)     20;
set val(seed)   1.0;
set val(adhocRouting)   AODV;
set val(stop)   200;
set val(cp)     "cbr020.txt";
set val(sc)     "setdest20node001.tcl";

set ns_         [new Simulator]

# setup topography object

set topo   [new Topography]

# create trace object for ns and nam

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

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

```
# Create God
set god_ [create-god $val(nn)]

#global node setting
$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)
\
                  -topoInstance $topo \
                  -agentTrace ON \
                  -routerTrace ON \
                  -macTrace ON \
                  -movementTrace ON \

# 802.11p default parameters
Phy/WirelessPhy set  RXThresh_ 5.57189e-
11 ; #400m
Phy/WirelessPhy set  CSThresh_ 5.57189e-
11 ; #400m

#  Create the specified number of nodes
[$val(nn)] and "attach" them
#  to the channel.
for {set i 0} {$i < $val(nn)} {incr i} {
           set node_($i) [$ns_ node]
           $node_($i) random-motion 0 ;#
disable random motion
}
```

```
# 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) 20
}

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

#$ns_ at  $val(stop)  "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.4 Kode Skrip AWK *Packet Delivery Ratio*

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

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

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

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

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

A.5 Kode Skrip AWK Rata-Rata *End-to-End Delay*

```
BEGIN{
      sum_delay = 0;
      count = 0;
}
{
      if ($2 >= 101) {

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

            if($4 == "AGT" && $1 == "s") {
                  start_time[$6] = $2;
            }

            else if(($7 == "cbr") && ($1 ==
"r")) {
                  end_time[$6] = $2;
            }

            else if($1 == "D" && $7 == "cbr")
{
                  end_time[$6] = -1;
            }
      }
}
END {

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

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

A.6 Kode Skrip AWK *Routing Overhead*

```
BEGIN {
     rt_pkts = 0;
}
{
           if (($1 == "s" || $1 == "f")
&& ($4 == "RTR") && ($7 == "AODV")) {

                    rt_pkts++;
           }
     }
}
END {
     printf "Routing Packets \t= %d \n",
rt_pkts;
}
```

BIODATA PENULIS

Muchamad Buyung Abiyoso, lahir di Kediri, 14 Agustus 1995. Penulis adalah anak pertama dari 8 bersaudara. Penulis menempuh pendidikan sekolah dasar di SD Negeri Banjaran 4 Kota Kediri lalu melanjutkan pendidikan sekolah menengah pertama di SMP Negeri 1 Kota Kediri dan penulis menempuh pendidikan menengah atas di SMA Negeri 2 Kediri. Selanjutnya penulis melanjutkan pendidikan sarjana di Departemen Informatika, Fakultas Teknologi Informasi dan Komunikasi, Institut Teknologi Sepuluh Nopember Surabaya.

Dalam menyelesaikan pendidikan S1, penulis mengambil bidang minat Arsitektur dan Jaringan Komputer (AJK). Sebagai mahasiswa, penulis berperan aktif dalam organisasi kampus sebagai anggota non-staf Himpunan Mahasiswa Teknik-Computer (HMTC). Selain itu, penulis juga menjadi staf NLC SCHEMATICS 2015 dan staf NLC SCHEMATICS 2016. Penulis pernah melakukan kerja praktik di Erporate, Yogyakarta, periode Juli – Agustus 2017. Penulis dapat dihubungi melalui nomor *handphone*: 081230217020 atau *email*: buyung14@gmail.com