TUGAS AKHIR – KI141502

Studi Kinerja 802.11p pada protokol *Dynamic Source Routing* (DSR) di lingkungan *Mobile Ad Hoc Network* (MANET)

PATTOE MARZA
NRP 5110 100 185

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

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

TUGAS AKHIR – KI141502

Studi Kinerja 802.11p pada protokol *Dynamic Source Routing* (DSR) di lingkungan *Mobile Ad Hoc Network* (MANET)

PATTOE MARZA
NRP 5110100 185

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

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

(Halaman ini sengaja dikosongkan)

FINAL PROJECT – KI141502

Study of 802.11p for Dynamic Source Routing (DSR) protocol on Mobile AdHoc Network (MANET)

PATTOE MARZA
NRP 5110100 185

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

INFORMATICS DEPARTMENT
Faculty of Information Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2017

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

STUDI KINERJA 802.11p PADA PROTOKOL DYNAMIC SOURCE ROUTING (DSR) DI LINGKUNGAN MOBILE AD HOC NETWORK (MANET)

TUGAS AKHIR

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

Oleh:
Pattoe Marza
NRP. 5110 100 185

Disetujui oleh Pembimbing Tugas Akhir:

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

SURABAYA
JUNI, 2017

(Halaman ini sengaja dikosongkan)

STUDI KINERJA 802.11p PADA PROTOKOL DYNAMIC SOURCE ROUTING (DSR) DI LINGKUNGAN MOBILE AD HOC NETWORK (MANET)

Nama Mahasiswa : **Pattoe Marza**
NRP : **5110 100 185**
Jurusan : **Teknik Informatika FTIf - ITS**
Dosen Pembimbing : **Dr.Eng. Radityo Anggoro, S.Kom., M.Sc.**

ABSTRAK

Dewasa ini perkembangan dunia internet semakin berkembang dengan pesat, mulai digunakan untuk browsing, chatting dan lain sebagainya. Kegiatan tersebut merupakan termasuk suatu aplikasi yang berhubungan dengan proses pengiriman paket data dalam jaringan internet. Ini tidak lepas dari teknologi nirkabel yang terus berkembang lebih maju tiap saat.

Berdasarkan hasil survei yang dilakukan oleh Asosiasi Penyelenggara Jasa Internet Indonesia (APJII) dengan Pusat Kajian Komunikasi Universitas Indonesia (PUSKAKOM UI). Hasil survei menunjukkan bahwa jumlah pengguna internet di Indonesia untuk saat ini telah mengalami peningkatan sebesar 34,9% dari jumlah seluruh penduduk Indonesia. Jika pada tahun sebelumnya ada sekitar 92 juta pengguna internet di Indonesia, maka pada semester pertama 2017 ini jumlah tersebut mencapai sekitar 132 juta pengguna. Dengan semakin banyak pengguna internet ini semakin bertambah pula kebutuhan yang diperlukan untuk berkomunikasi antar pengguna dan pasti akan semakin kompleks.

Tidak dipungkiri, hampir semua kalangan, dari anak-anak hingga dewasa sudah menggunakan perangkat bergerak, mulai dari smartphone, notebook, tablet hingga PC (Personal Computer) dan perangkat itu semua terkoneksi dengan internet. Itulah mengapa pada saat ini teknologi internet dan nirkabel khususnya

ini sering dikatakan menjadi indikator kemajuan peradaban manusia karena dengan teknologi ini pengguna memungkinkan berkomunikasi secara langsung dengan pengguna lain dalam kondisi yang tidak menetap. Namun saat ini sebagian besar koneksi antara perangkat nirkabel masih mengunakan layanan penyedia infrastruktur yang tetap. Dengan demikian diperlukan sebuah model baru untuk menyediakan layanan jaringan tanpa infrastruktur.

Maka dari itu, pada Tugas Akhir ini yang diteliti adalah skema MANET yang dihasilkan oleh filenode-movement dan traffic-pattern yang telah ada pada distribusi network simulator. Penelitian ini menggunakan NS-2 sebagai network simulator dengan protokol proaktif MANET jenis DSR (Dynamic Source Routing).

Kemudian sistem pada Tugas Akhir ini diuji fungsionalitas dan performanya dengan melalui beberapa skenario yang telah ditentukan. Hasil dari pengujian adalah kinerja 802.11p pada protokol routing DSR di lingkungan MANET dengan menggunakan NS-2. Performa protokol routing tersebut diukur berdasarkan routing overhead, packet delivery ratio, dan delay pengiriman paket dari satu node ke node lainnya.

Kata Kunci: MANET, 802.11p, Network Simulator, NS-2, DSR.

STUDY OF 802.11p FOR Dynamic Source Routing (DSR) ON MOBILE AD HOC NETWORK USING NETWORK SIMULATOR 2 (NS-2)

Name : Pattoe Marza
NRP : 5110 100 185
Major : Informatics Engineering, IT Dept – ITS
Advisor : Dr. Eng. Radityo Anggoro,S.Kom., M.Sc.

ABSTRACT

Internet has been rapidly growing these days, it is used for browsing, chatting, and other activities. Those activities are commonly using application that are related to data package delivering in the internet network. Of course this phenomenon is strongly related with the wireless technology which is also rapidly growing every single day.

Based on a survey by Asosiasi Penyelenggara Jasa Internet Indonesia (APJII) dengan Pusat Kajian Komunikasi Universitas Indonesia (PUSKAKOM UI), the result shows that internet users in Indonesia has increased 34,9% of Indonesia's population. Previously, there are about 92 million internet users in Indonesia, but on the first quarter of 2017, the amount has increased to 132 million people. With the increasing percentage of the users, the needs of communication between users are also more complex.

Almost every kind of people, either kids or adults, are using mobile devices like smartphones, tablets, notebook, or Personal Computer (PC), and all of them are most likely connected to the internet. That's why wireless internet connection can be measured as an indicator of the advancement of human knowledge. By using this technology, people can communicate to each other regardless where they are right now. However, most of wireless connection right now are still using static infrastructure service,

thus it requires a new model of wireless service with no infrastructure.

Therefore, the main focus in this final project is to do a research of MANET scheme which is produced by filenode-movement and traffic-pattern that has been in the network simulator distribution. This research uses NS-2 as the network simulator with proactive MANET protocol of type DSR(Dynamic Source Routing).

Keywords:MANET, 802.11p, Network Simulator, NS-2, DSR.

KATA PENGANTAR

Bismillahirrohmanirohim.

Alhamdulilahirabil'alamin, segala puji bagi Allah SWT, atas segala rahmat dan karunia-Nya yang tak terhingga sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul:

"Studi Kinerja 802.11p pada protokol Dynamic Source Routing (DSR) di lingkungan Mobile Ad Hoc Network (MANET)".

Terselesaikannya Tugas Akhir ini tidak terlepas dari bantuan banyak pihak. Oleh karena itu melalui lembar ini penulis ingin mengucapkan terima kasih dan penghormatan sebesar-besarnya kepada pihak-pihak sebagai berikut.

1. Allah SWT, karena limpahan rahmat dan karunia-Nya lah penulis dapat menyelesaikan Tugas Akhir dan juga perkuliahan di Teknik Informatika ITS.
2. Ayah penulis, Zainul Arif yang selalu memberikan nasihat kehidupan, membimbing penulis menjadi manusia yang lebih baik, dan selalu memberikan semangat agar penulis dapat menyelesaikan Tugas Akhir.
3. Ibu penulis, Masnun Sadariah tiada henti memberikan semangat, doa serta dukungan penuh kepada penulis selama ini sehingga dapat menyelesaikan Tugas Akhir ini dan perkuliahan di Teknik Informatika ITS Surabaya.
4. Adik penulis, Deo Rezeki yang telah memberikan dukungan dan doa kepada penulis untuk terus semangat dalam penyelesaian Tugas Akhir ini.
5. Bapak Dr.Eng. Radityo Anggoro S.Kom., M.Sc.selaku dosen pembimbing dari penulis yang telah memberikan bimbingan, dukungan, masukan, nasihat dan banyak

arahan kepada penulis dalam menyelesaikan Tugas Akhir ini.

6. Bapak Daniel O. Siahaan, S.Kom., M.Sc., PDEng. selaku dosen wali dari penulis yang selalu memberi nasihat kepada penulis selama menjalani perkuliahan di Teknik Informatika ITS.
7. Bapak Dr. Darlis Herumurti, S.Kom., M.Kom. selaku Ketua Jurusan Teknik Informatika ITS.
8. Sahabat dari penulis, Remy Giovanni Mangowal yang selalu mendengarkan keluh kesan penulis, memberikan dukungan, dan menemani keseharian penulis serta mengingatkan penulis agar cepat menyelesaikan Tugas Akhir.
9. Anton, yang selalu setia menemani, membantu, memberikan dukungan, doa serta semangat yang tiada henti kepada penulis dengan penuh sabar, suka, duka serta kepedulian yang sangat besar kepada penulis.
10. Sahabat baik dari penulis, Agil, Zendra, RM, Boz, Afid, Fauzi, Aaz, Adit, dan Raedy yang selalu menguatkan, memberikan dukungan, semangat, nasihat, hiburan dan canda tawa kepada penulis hingga saat ini.
11. Bulik Diyah dan Puput, tante dan sepupu dari penulis yang selalu memberikan motivasi, dukungan dan semangat selama mengerjakan Tugas Akhir serta menemani keseharian penulis.
12. Hasbi, teman seperjuangan dalam mengerjakan Tugas Akhir yang selalu bimbingan bersama.
13. Hidari, yang mau sudah mau direpotkan dengan pertanyaan-pertanyaan yang berhubungan dengan topik Tugas Akhir dari penulis.
14. Teman-teman TC angkatan 2010, kakak angkatan, dan juga adik angkatan yang telah ramah dan berbaik hati membantu penulis selama berada di Teknik Informatika.Terima kasih atas rasa kekeluargaan yang telah kalian berikan kepada penulis.

15. Pihak-pihak yang tidak dapat penulis sebutkan satu per satu yang telah membantu penulis dalam menyelesaikan Tugas Akhir ini.

Penulis mengharapkan adanya saran dan kritik yang membangun dari pembaca, sehingga memperlancar Tugas Akhir ini agar dapat menjadi manfaat bagi masyarakat.

Surabaya, Mei 2016

Pattoe Marza

(Halaman ini sengaja dikosongkan)

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

(Halaman ini sengaja dikosongkan)

BAB I
PENDAHULUAN

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

1.1. Latar Belakang

Dewasa ini perkembangan dunia internet semakin berkembang dengan pesat, mulai digunakan untuk *browsing*, *chatting* dan lain sebagainya. Kegiatan tersebut merupakan termasuk suatu aplikasi yang berhubungan dengan proses pengiriman paket data dalam jaringan internet. Ini tidak lepas dari teknologi nirkabel yang terus berkembang lebih maju tiap saat. Berdasarkan hasil survei yang dilakukan oleh Asosiasi Penyelenggara Jasa Internet Indonesia (APJII) dengan Pusat Kajian Komunikasi Universitas Indonesia (PUSKAKOM UI).

Hasil survei menunjukkan bahwa jumlah pengguna internet di Indonesia untuk saat ini telah mengalami peningkatan sebesar 34,9% dari jumlah seluruh penduduk Indonesia. Jika pada tahun sebelumnya ada sekitar 92 juta pengguna internet di Indonesia, maka pada semester pertama 2017 ini jumlah tersebut mencapai sekitar 132 juta pengguna. Dengan semakin banyak pengguna internet ini semakin bertambah pula kebutuhan yang diperlukan untuk berkomunikasi antar pengguna dan pasti akan semakin kompleks. Tidak dipungkiri, hampir semua kalangan, dari anak-anak hingga dewasa sudah menggunakan perangkat bergerak, mulai dari *smartphone*, *notebook*, *tablet* hingga PC (*Personal Computer*) dan perangkat itu semua terkoneksi dengan internet. Itulah mengapa pada saat ini teknologi internet dan nirkabel khususnya ini sering dikatakan menjadi indikator kemajuan peradaban manusia karena dengan teknlogi ini pengguna memungkinkan berkomunikasi secara langsung dengan pengguna lain dalam kondisi yang tidak menetap. Namun saat ini sebagian besar koneksi antara perangkat nirkabel masih mengunakan

layanan penyedia infrastruktur yang tetap. Dengan demikian diperlukan sebuah model baru untuk menyediakan layanan jaringan tanpa infrastruktur.

Salah satu model baru tersebut adalah *Mobile Ad-hoc Network* (MANET). MANET merupakan konsep dari *Wireless Ad-Hoc Network* (WANET). MANET tergolong ke dalam jaringan komunikasi nirkabel dimana setiap perangkat yang terhubung di dalam MANET terhubung dengan *Ad-Hoc*. MANET merupakan teknologi jaringan dimana jaringan dapat membentuk diri sendiri, memperbaiki diri sendiri, dan dapat berkomunikasi tanpa resources yang terpusat dan infrastruktur yang tetap.

Implementasi pada lingkungan MANET dapat dilakukan dengan menggunakan simulasi sehingga penelitian ini dapat dilakukan untuk mempelajari sistem dengan baik. Simulasi dilakukan dengan menggunakan *Network Simulator 2* (NS-2). Implementasi ini akan dilakukan analisa performa 802.11p (WAVE) pada protokol *routing reactive* yaitu *Dynamic Source Routing* (DSR).

Implementasi pada lingkungan MANET dapat dilakukan dengan menggunakan simulasi sehingga penelitian ini dapat dilakukan untuk mempelajari sistem dengan baik. Simulasi dilakukan dengan menggunakan *Network Simulator 2* (NS-2). Implementasi ini akan dilakukan analisa performa *Dynamic Source Routing* (DSR).

Hasil yang diharapkan dari Tugas Akhir ini adalah suatu model transmisi 802.11p pada protokol *routing* DSR di lingkungan MANET dengan menggunakan aplikasi NS-2. Performa protokol *routing* tersebut diukur berdasarkan *routing overhead*, *packet delivery ratio*, dan *delay* pengiriman paket dari node ke node lainnya.

1.2. Rumusan Permasalahan

Rumusan masalah yang diangkat dalam Tugas Akhir ini adalah sebagai berikut.

1. Bagaimana kinerja *protokol* routing DSR pada lingkungan MANET?

2. Bagaimana perbandingan kinerja DSR dengan MAC 802.11p pada skenario dinamis??

1.3. Batasan Permasalahan

Permasalahan yang dibahas dalam Tugas Akhir ini memiliki beberapa batasan, di antaranya sebagai berikut.

1. Protokol routing hanya dijalankan dan diujicobakan pada aplikasi *Network Simulator 2* (NS-2)
2. Protokol routing yang diujicobakan adalah DSR.
3. Lingkungan jaringan digunakan untuk uji coba adalah *Mobile Ad Hoc Network* (MANET).
4. Jumlah node yang digunakan dalam masing-masing scenario, yaitu, 50, 75, 100, 125.
5. 3 variasi kecepatan yang akan digunakan adalah 10 m/s, 20 m/s dan 30 m/s

1.4. Tujuan dan Manfaat

Tujuan dari pembuatan Tugas Akhir ini adalah untuk mengetahui kinerja 802.11p pada protokol DSR yang ditinjau dari pengukuran nilai PDR, RO, dan E2D.

Sedangkan manfaatnya adalah dapat digunakan sebagai acuan penelitian terkait performa protokol *routing* di MANET.

1.5. Metodologi

Metodologi yang akan dilakukan didalam pengerjaan Tugas Akhir ini terdiri atas beberapa tahapan sebagai berikut.

1. **Studi Literatur**
   Pada tahap ini dilakukan pencarian informasi dan studi literatur yang digunakan untuk pemrosesan data dan desain perangkat lunak yang akan dibuat. Informasi didapatkan dari buku dan literatur lain yang berhubungan dengan algoritma yang digunakan dalam pengerjaan Tugas Akhir.

2. **Implementasi**
   Tahap Implementasi merupakan tahap pembangunan perangkat lunak yang mengimplementasikan algoritma-

algoritma yang sudah diajukan. Sesuai dengan rancangan yang diajukan pada proposal, pembangunan perangkat lunak diimplementasikan sesuai dengan konsep yang telah didapatkan saat studi literatur.

3. **Pengujian dan Evaluasi**
   Pada tahap ini dilakukan uji coba terhadap aplikasi yang telah dibuat. Pengujian dan evaluasi akan dilakukan dengan melihat kesesuain dengan perencanaan. . Uji coba ini dilakukan untuk membuktikan bahwa perangkat lunak yang dibangun telah bekerja sesuai dengan tujuan dan menjadi solusi dari permasalahan.

4. **Penyusunan Buku Tugas Akhir**
   Pada tahap ini dilakukan penyusunan laporan yang menjelaskan dasar teori dan metode yang digunakan dalam Tugas Akhir ini serta hasil dari implementasi aplikasi yang telah dibuat. Sistematika penulisan buku Tugas Akhir secara garis besar antara lain sebagai berikut.

   1. Pendahuluan
      a. Latar Belakang
      b. Rumusan Permasalahan
      c. Batasan Permasalahan
      d. Tujuan dan Manfaat
      e. Metodologi
      f. Sistematika Penulisan
   2. Tinjauan Pustaka
   3. Perancangan Sistem
   4. Implementasi
   5. Pengujian dan Evaluasi
   6. Penutup

1.6. Sistematika Penulisan

Buku Tugas Akhir ini terdiri atas beberapa bab yang dijelaskan sebagai berikut.

- Bab I. Pendahuluan
  Bab ini berisi latar belakang masalah, permasalahan, batasan masalah, tujuan Tugas Akhir, manfaat Tugas Akhir, metodologi yang digunakan, dan sistematika penyusunan buku Tugas Akhir.
- Bab II. Tinjauan Pustaka
  Bab ini membahas beberapa teori penunjang yang berhubungan dengan pokok pembahasan dan mendasari pembuatan Tugas Akhir ini.
- Bab III. Perancangan Sisten
  Bab ini berisi perancangan metode yang nantinya akan diimplementasikan dan dilakukan pengujian dari aplikasi yang akan dibangun.
- Bab IV. Implementasi
  Bab ini menjelaskan tentang pembangunan aplikasi, penjelasan fungsi-fungsi yang digunakan dan diaplikasikan agar perangkat lunak berjalan sesuai dengan rencana yang diajukan.
- Bab V. Pengujian dan Evaluasi
  Bab ini menjelaskan tahap pengujian sistem dan performa dalam skenario mobilitas yang telah dibuat
- Bab VI. Penutup
  Bab ini berisi kesimpulan dari hasil pengujian yang dilakukan terhadap rumusan masalah yang ada serta saran untuk pengembangan selanjutnya.

(Halaman ini sengaja dikosongkan)

BAB II
TINJAUAN PUSTAKA

Bab ini berisi penjelasan teori-teori yang berkaitan dengan pengimplementasian perangkat lunak. Penjelasan ini bertujuan untuk memberikan gambaran secara umum terhadap *routing protocol* yang dibuat dan berguna sebagai penunjang dalam pengembangan perangkat lunak.

2.1. Mobile Ad Hoc Network (MANET)

Mobile Ad Hoc Network (MANET) adalah sebuah jaringan terorganisir yang dibentuk dengan menghubungkan beberapa *wireless mobile* tanpa harus ada administrasi yang terpusat atau infrastruktur tetap. MANET topologi bersifat dinamis, *nodes* terhubung dan memutuskan hubungan secara terus menerus. Beberapa contoh skenario MANET diantaranya adalah rapat, komunikasi militer, dan juga *sensor networks*.

Beberapa MANET terbatas pada perangkat *wireless* di *local area* seperti contohnya kelompok oleh beberapa *laptop* atau *computer*, dan ada juga yang terhubung ke internet, contohnya subset dari MANET, yaitu *Vehicular Ad Hoc Network* (VANET). Karena sifat dinamis dari MANET, jaringan MANET tidak terlalu aman, oleh karena itu data yang dikirim dalam jaringan MANET harus diperhatikan secara baik-baik. Terdapat berbagai jenis protokol *routing* untuk MANET yang secara keseluruhan dapat dibagi menjadi beberapa kelompok, antara lain:

a. *Proactive Routing*
   Algoritma ini akan mengelola daftar tujuan dan rute terbaru masing-masing dengan cara mendistribusikan routing table ke seluruh jaringan, sehingga jalur lalu lintas (traffic) akan sering dilalui oleh *routing table* tersebut. Hal ini akan memperlambat aliran data jika terjadi restrukturisasi *routing table*. Contoh *proactive routing* adalah *Intrazone Routing Protocol* (IARP).

b. *Reactive Routing*

Tipe ini akan mencari rute (*on demand*) dengan cara membanjiri jaringan dengan paket *route request*. Sehingga dapat menyebabkan jaringan akan penuh (*clogging*). Contoh *reactive routing* adalah *Ad Hoc On Demand Distance Vector* (AODV), *Dynamic MANET On Demand Routing* (DYMO).

c. *Flow Oriented Routing*
Tipe protokol ini mencari rute dengan mengikuti aliran yang disediakan. Salah satu pilihan adalah dengan *unicast* secara terus-menerus ketika meneruskan data saat mempromosikan *link* baru. Beberapa kekurangan tipe protokol ini adalah membutuhkan waktu yang lama untuk mencari rute yang baru. Contoh *flow oriented routing* adalah *Interzone Routing Protocol* (IERP), *Lightweight Underlay Network Ad Hoc Routing* (LUNAR), *Signal Stability Routing* (SSR).

d. *Hybrid Routing*
Tipe protokol ini menggabungkan antara *proactive routing* dengan *reactive routing*. Contoh *hybrid routing* adalah *Hybrid Routing Protocol for Large Scale MANET* (HRPLS), *Hybrid Wireless Mesh Protocol* (HWMP), *Zone Routing Protocol* (ZRP).

2.2. Dynamic Source Routing (DSR)

Dynamic Source Routing adalah *routing protocol* sederhana dan efisien yang didesain khusus untuk mobile nodes di *multi-hop wireless ad hoc networks* . DSR secara resmi didefinisikan dalam internet-draft pada tahun 2003. Pada awalnya, DSR dikembangkan untk meciptakan *routing protocol* yang dapat bereaksi dengan cepat pada perubahan di dalam *network*, dan dapat menyediakan respon yang cepat untuk membantu memastikan pengiriman data yang sukses meskipun terjadi banyak perubahan pada kondisi *network*.

Protocol DSR terdiri dari 2 mekanisme utama yang bekerja bersama untuk mengizinkan *discovery* dan *maintenance* dari *source routes* di *ad hoc network*.

Route Discovery adalah mekanisme untuk mendapatkan *source route* dimana ketika *node* S berharap untuk mengirim paket data menuju *node* D. *Route Discovery* hanya digunakan ketika *node* S mencoba untuk mengirim paket ke *node* D dan *node* S belum tahu rute menuju ke *node* D.

Sedangkan *Route Maintenance* adalah mekanisme dimana *node* S dapat mendeteksi perubahan pada link jika tidak memungkinkan untuk melanjutkan pengiriman ketika sedang menggunakan *source route* menuju ke *node* D. Ketika *Route Maintenance* mengindikasikan bahwa *source route* tidak dapat digunakan, *node* S dapat mencoba untuk menggunakan rute lain yang diketahui untuk mencapai *node* D, atau mencoba kembali *Route Discovery* untuk menemukan jalur baru menuju ke *node* D. *Route Maintenance* hanya digunakan ketika *node* S benar-benar mengirim paket menuju *node* D [1].

2.3. 802.11p – Wireless Access Vehicular Environment (WAVE)

Untuk memanfaatkan potensi – potensi yang ada pada sistem komunikasi , saat ini IEEE sedang mengembangkan suatu perubahan standar IEEE 802.11p atau yang biasa disebut dengan WAVE (*Wireless Access Vehicular Environments*). WAVE merupakan penyempurnaan standar IEEE 802.11 yang diperlukan untuk mendukung pengaplikasian ITS (*Intelligent Transportation Systems*).

WAVE juga merupakan pengembangan sistem IEEE 802.11a dengan memperkenalkan *physical layer* dan *MAC layer* yang dapat meningkatkan sistem operasi dan aplikasi keselamatan dengan memberikan tingkat *latency* rendah. WAVE sendiri beroperasi pada *band* 5.9 GHz dengan menggunakan sistem *multiplexing* OFDM (*Orthogonal Frequency Division Multiplexing*) dan dapat mencapai kecepatan pentransmisian data antara 6 – 27 Mbps.

WAVE terdiri dari tujuh channel pada frekuensi 10 MHz yang dari satu control channel dan enam *service channel* pada *band* 5.9 GHz. *Service channel* digunakan untuk *public safety* dan *private service*, sedangkan *control channel* digunakan sebagai referensi *channel* untuk membangun link komunikasi antara RSU (*Road – Side Unit*) dan OBU (*On – Board Unit*) [2].

2.4. Network Simulator 2 (NS-2)

Network Simulator 2 (NS-2) [3] adalah suatu *interpreter* yang *object-oriented*, dan *discrete event-driven* yang dikembangkan oleh *University of California Berkeley* dan USC ISI sebagai bagian dari projek *Virtual Internet Testbed* (VINT). NS menjadi salah satu *tool* yang sangat berguna untuk menunjukkan simulasi jaringan melibatkan *Local Area Network* (LAN), *Wide Area Network* (WAN), tapi fungsi dari *tools* ini telah berkembang selama beberapa tahun belakangan ini untuk memasukkan didalamnya jaringan nirkabel (*wireless*) dan juga jaringan *ad hoc*.

Network Simulator pertama kali dibangun sebagai varian dari *REAL Network Simulator* pada tahun 1989 di *University of California Berkeley* (UCB). Pada tahun 1995 pembangunan *Network Simulator* didukung oleh *Defense Advanced Research Project Agency* (DARPA) melalui VINT Project, yaitu sebuah tim riset gabungan yang beranggotakan tenaga-tenaga ahli dari beberapa instansi ternama. Ada beberapa keuntungan menggunakan NS sebagai perangkat lunak simulasi pembantu analisi dalam riset, antara lain adalah NS dilengkapi dengan *tool* validasi yang digunakan untuk menguji kebenaran pemodelan yang ada pada NS. Secara default, semua pemodelan NS akan dapat melewati proses validasi ini. Pemodelan media, protokol dan komponen jaringan yang lengkap dengan perilaku trafiknya sudah disediakan pada *library* NS. NS juga bersifat *open source* dibawah *GNU Public License* (GPL), sehingga NS dapat diunduh dan digunakan secara gratis. *Simulator* ini dapat diunduh melalui *website* NS yaitu http://www.isi.edu/nsnam/. Sifat dari *simulator* ini yang *open source* juga mengakibatkan pengembangan NS menjadi lebih dinamis dan lebih mudah.

Pada Tugas Akhir ini digunakan NS-2 versi 2.35 sebagai aplikasi simulasi jaringan skenario MANET yang dihasilkan oleh *program default* dari NS-2 yaitu *generator filenode-movement* dan *traffic-connection pattern* menggunakan protokol DSR.

2.5. Generator FileNode-Movement dan Traffic-Connection Pattern

2.5.1. FileNode-Movement (Mobility Generator)

Tools ‘setdest’ digunakan untuk menghasilkan *random movement* dari *node* dalam jaringan nirkabel. *Node movement* dihasilkan dengan kecepatan gerak yang spesifik bergerak menuju lokasi acak atau lokasi spesifik yang berada dalam kawasan yang telah ditentukan sebelumnya. Ketika *node* sampai, *node* tersebut bisa diatur untuk berhenti untuk sementara waktu. Setelah itu, *node* terus bergerak menuju lokasi berikutnya. Lokasi ‘setdest’ berada pada direktori ‘~ns/indep-utils/cmu-scen-gen/setdest/’

Pengguna harus menjalankan program ‘setdest’ terlebih dahulu sebelum menjalankan program simulasi yang akan dilakukan. Format dari *Command Line* ‘setdest’ ditunjukkan pada **Error! Reference source not found.**1 dan keterangannya itunjukkan pada Tabel 2.1.

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

Gambar 2.1 Command Line setdest

Tabel 2.1 Command Line 'setdest'

Parameter	Keterangan
-v version	Versi ‘setdest’ dari simulator yang digunakan
-n num	Jumlah *node* dalam skenario
-p pausetime	Durasi ketika sebuah *node* tetap diam setelah tiba di lokasi pergerakan. Jika nilai ini diatur ke 0, maka *node* tidak akan berhenti ketika tiba di lokasi pergerakan dan akan terus bergerak sampai akhirnya tiba di lokasi
Parameter	**Keterangan**
-M maxspeed	Kecepatan maksimum sebuah *node*
-t simtime	Waktu simulasi
-x max x	Panjang maksimum area simulasi
-y max y	Lebar maksimum area simulasi

Tools ‘setdest’ akan menghasilkan *file output* yang isinya adalah jumlah *node* dan mobilitas yang akan digunakan dalam file Tcl selama simulasi. File output akan berisi skrip pergerakan dan

juga mengandung beberapa statistik lain tentang perubahan link dan rute.

Untuk membuat skenario *node-movement* yang terdiri dari 50 *node*, bergerak dengan kecepatan maksimum 10 m/s dengan *pause time* sebesar 2 detik, berhenti setelah 200 detik dengan besar topologi yaitu 800 x 800 meter^2, maka contoh *command line*-nya akan terlihat seperti seperti pada **Error! Reference source not ound.**2

```
setdest -v 1 -n 50 -p 2.0 -M 10.0 -t 200 -x 800 -y
800 > sc_n50_m10_1
```

Gambar 2.2 Contoh Command Line setdest

File *output* ditulis ke "stdout" secara *default*. Di sini *output* disimpan ke dalam file " sc_n50_m10_1". File dimulai dengan posisi awal *node* dan berlanjut menetapkan *node-movement* seperti terlihat pada **Error! Reference source not found.**3.

```
$ns_ at 2.000000000000 "$node_(0) setdest
90.441179033457 44.896095544010
1.373556960010
```

Gambar 2.3 file sc_n50_m10_1

Command line pada **Error! Reference source not ound.**3 dari 'sc_n50_m10_1' menunjukan bahwa *node* (0) pada detik ke 2.0 mulai bergerak ke arah tujuan (90.441179033457, 44. 896095544010) dengan kecepatan 1.373556960010 m/s. *Command line* ini dapat digunakan untuk mengubah arah dan kecepatan gerak dari *mobile node*. Objek GOD disini digunakan untuk menyimpan informasi global tentang keadaan dari lingkungan jaringan dan *node* di sekitarnya.

Dalam simulasi ini objek GOD digunakan untuk menyimpan sebuah *array* dari jumlah *hop* terpendek yang diperlukan untuk mencapai satu *node* ke *node* yang lain. Objek GOD tidak menghitung jumlah *hop* yang diperlukan selama simulasi berjalan. Namun GOD menghitung *hop* di akhir simulasi. Informasi yang dimuat ke dalam objek terdapat pada baris perintah di **Error! Reference source not found.**4.

```
$ns_ at 899.642 "$god_ set-dist 23 46 2"
```

Gambar 2.4 Command Line GOD pada sc_n50_m10_1

Gambar diatas menunjukan bahwa jarak terpendek antara *node* 23 dan *node* 46 berubah menjadi 2 *hop* diwaktu 899,642 detik. Program 'setdest' menghasilkan *filenode-movement* menggunakan algoritma *random way point*. Perintah-perintah yang termasuk dalam program utama untuk memuat file-file ini dalam objek GOD [4].

2.5.2. File Traffic-Connection Pattern

Untuk menghasilkan alur *traffic* yang acak, dapat digunakan skrip Tcl di dalam ns-2 yang disebut "cbrgen". Skrip ini digunakan untuk menghasilkan *traffic load*. Skrip ini disimpan di dalam file 'CMU-scen-gen ' yang terletak di dalam direktori~ns/indep-utils/cmu-scen-gen. Program "cbrgen.tcl" digunakan sesuai dengan *command line* pada **Error! Reference ource not found.**5 dan dengan keterangan yang ditunjukkan pada Tabel 2.2.

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

Gambar 2.5 Format cbrgen.tcl

Tabel 2.2 Command Line file cbrgn.tcl

Parameter	Keterangan
-type cbr|tcp	Jenis *traffic* yang digunakan TCP atau CBR
-nn nodes	Jumlah total *node*
-s seed	*Random seed*
-mc connections	Jumlah koneksi
-rate rate	Jumlah paket per detik. Pada CBR, panjang paket adalah tetap yaitu sebesar 512 bytes selama simulasi

Command line pada **Error! Reference source not ound.**6 membuat sebuah file koneksi CBR antara 50 *node*, yang memiliki maksimal 20 koneksi, dengan nilai *seed* 1.0 dan jumlah paket per detik sebanyak 4.0.

```
ns cbrgen.tcl -type cbr -nn 50 -seed 1.0 -mc 20 -
rate 4.0 > cbrcoba
```

Gambar 2.6 Contoh Command Line cbrgen.tcl

File cbrcoba dijalankan sehingga menghasilkan salah suatu koneksi CBR yang terlihat seperti pada **Error! Reference source ot found.**7.

```
#
# 2 connecting to 3 at time 82.557023746220864
#
set udp_(0) [new Agent/UDP]
$ns_ attach-agent $node_(2) $udp_(0)
set null_(0) [new Agent/Null]
$ns_ attach-agent $node_(3) $null_(0)
set cbr_(0) [new Application/Traffic/CBR]
$cbr_(0) set packetSize_ 512
$cbr_(0) set interval_ 0.25
$cbr_(0) set random_ 1
$cbr_(0) set maxpkts_ 10000
$cbr_(0) attach-agent $udp_(0)
$ns_ connect $udp_(0) $null_(0)
$ns_ at 82.557023746220864 "$cbr_(0) start"
```

Gambar 2.7 Koneksi CBR pada cbrcoba

Agent yang akan digunakan adalah UDP. Dengan demikian koneksi UDP adalah setup antara *node* 2 dan 3. Jumlah sumber UDP (dipilih antara node 0-50) dan jumlah setup koneksi diindikasikan sebagai 20 koneksi di akhir file cbrcoba.

2.6. NS-2 TraceFile

NS-2 *TraceFile* merupakan output hasil menjalankan file NS-2 yang berekstensi .tr. *TraceFile* ini berisi *log* tentang semua

jenis pengiriman dan penerimaan paket yang terjadi selama simulasi. Di dalam TraceFile tercatat berbagai jenis paket sesuai jenis protokol *routing* yang digunakan. Tiap baris *log* yang ada di *TraceFile* ini akan dianalisis untuk mendapatkan performa protokol pada akhirnya. Contoh pengiriman data paket pada NS-2 TraceFile dapat dilihat pada **Error! Reference source not found.**8.

```
s 26.000000000 _1_ AGT  --- 618 cbr 64 [0 0 0 0] --
----- [1:0 0:0 32 0] [26] 0 0
```

Gambar 2.8 Trace pengiriman paket data

Huruf "s" di kolom pertama menandakan pengiriman paket , kolom beikutnya berisi waktu pengiriman paket, kolom ketiga merupakan node tempat *event* terjadi, kolom ke empat berisi AGT yang menandakan pengiriman paket data, kolom ke lima merupakan tempat terjadinya *event* spesial, kolom ke enam merupakan id unik paket, kolom ke tujuh berisi tipe paket yang dikirimkan dan kolom ke delapan merupakan ukuran paket dalam byte.

Untuk penerimaan paket kolom pertama ditandai dengan huruf inisial "r" yang menandakan bahwa paket diterima dan format selanjutnya sama persis dengan paket pengiriman. Contoh penerimaan data paket pada NS-2 TraceFile dapat dilihat pada **Error! Reference source not found.**9.

```
r 26.011627928 _0_ AGT  --- 618 cbr 84 [13a 0 16
800] ------- [1:0 0:0 27 0] [26] 5 0
```

Gambar 2.9 Trace Penerimaan Paket Data

Contoh format untuk paket *routing* DSR pada *tracefile* seperti pada **Error! Reference source not found.**0.

```
r 30.334151244 _12_ RTR  --- 741 DSR 64 [0 ffffffff
18 800] ------- [24:255 -1:255 32 0] [1 17 [HELLO 24
0 17]]
```

Gambar 2.10 Trace Pengiriman Paket Routing DSR

2.7. Awk

Awk adalah sebuah pemrograman seperti pada *shell* atau C yang memiliki karakteristik yaitu sebagai *tools* yang cocok untuk filter / manipulasi. Awk adalah penggabungan dari nama lengkap sang author, yaitu : Alfred V. Aho, Peter J. Weinberger dan Brian W. Kernighan. Awk atau juga disebut Gawk (GNU awk), yaitu bahasa pemrograman umum dan *utility standard* POSIX 1003.2. Jika kecepatan merupakan hal yang penting, awk adalah bahasa yang sangat sesuai. Awk sangat baik untuk manipulasi file teks. Secara umum bahasa pemrograman awk dapat digunakan untuk mengelola *database* sederhana, membuat laporan, memvalidasi data, menghasilkan indeks dan menampilkan dokumen, membuat algoritma yang digunakan untuk mengubah bahasa komputer ke bahasa lainnya. Dengan kata lain awk menyediakan fasilitas yang dapat memudahkan untuk memecah bagian data untuk proses selanjutnya, mengurutkan data dan menampilkan komunikasi jaringan yang sederhana.

Fungsi dasar awk adalah untuk mencari *file* per baris (atau unit teks lain) yang berisi pola tertentu. Ketika suatu baris sesuai dengan pola, awk melakukan aksi yang khusus pada baris tersebut.awk tetap memproses baris input sedemikian hingga mencapai akhir baris input. Program pada awk berbeda dari program di kebanyakan bahasa lain, karena program awk bersifat "*data-driven*" yang mana diperlukan pendeskripsian data yang dikehendaki untuk bekerja dan kemudian apa yang akan dilakukan saat data tersebut ditemukan. Kebanyakan bahasa lainnya bersifat "*procedural*" maka dari itu diharuskan mendeskripsikannya secara detail setiap langkah program yang harus dijalankan. Ketika bekerja dengan bahasa prosedural, biasanya sangat sulit untuk mendeskripsikan data yang hendak diproses oleh program. Oleh karena itu, program awk sering kali terasa lebih mudah untuk ditulis dan dibaca [5].

Pada tugas akhir ini AWK digunakan untuk membuat *script* dalam penghitungan *Packet Delivery Ratio* (PDR), *End-to-End Delay* (E2D), dan *Routing Overhead* (RO) dari hasil *trace* NS-2.

(Halaman ini sengaja dikosongkan)

BAB III
PERANCANGAN SISTEM

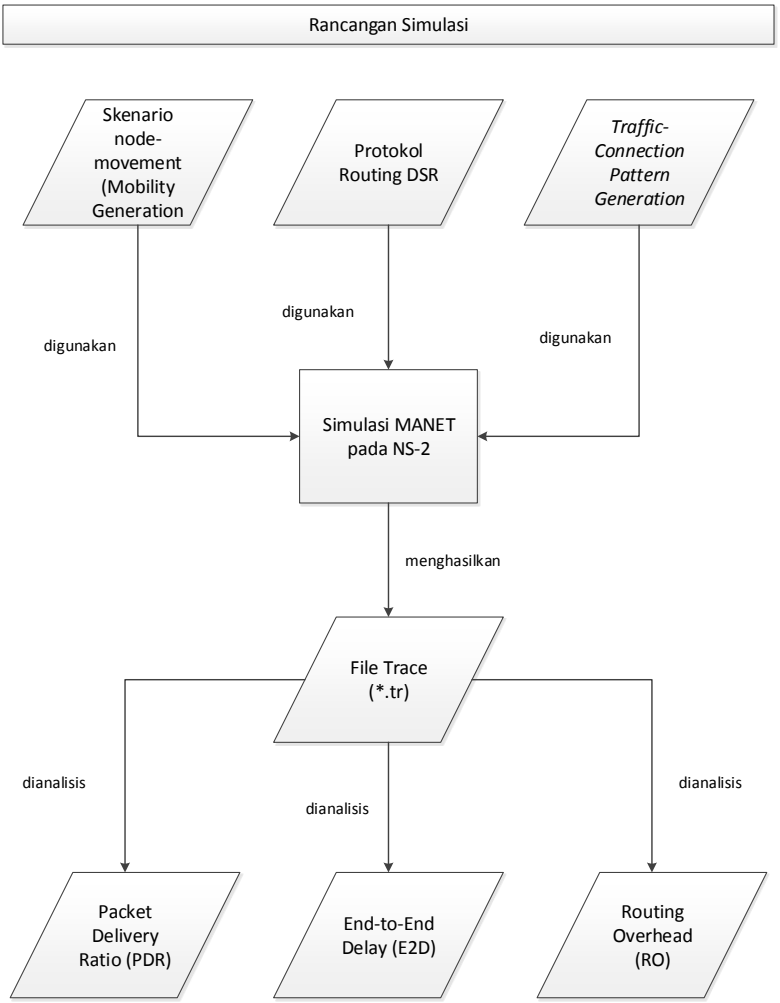
Pada bab ini dijelaskan mengenai dasar perancangan dari perangkat lunak yang akan dibangun di dalam Tugas Akhir ini. Secara khusus akan dibahas mengenai deskripsi umum sistem, perancangan skenario, alur, serta gambaran implementasi sistem yang diterapkan pada *Network Simulator 2* (NS-2).

3.1. Deskripsi Umum

Pada Tugas Akhir ini akan dilakukan analisis tentang performa 802.11p WAVE pada MANET. Dalam pembuatan skenario MANET menggunakan *Mobility Generator* yang bersifat *Random Way Point* dan telah ada dan terintegrasi pada *Network Simulator-2* (NS-2) yaitu dengan cara men-*generate filenode-movement* (*mobility generation*) dan membuat koneksi antar *node* menggunakan file *traffic-connection* pattern. Rancangan Simulasi awal yang telah dibuat dapat dilihat pada Gambar 3.1.

Dalam penelitian ini, Simulasi skenario yang dihasilkan oleh *mobility generator* akan dijalankan pada NS-2 menggunakan protokol *routing* DSR pada sistem operasi Linux.

Pada tiap skenario, kecepatan maksimum pergerakan antar node dibuat bervariasi yaitu 10, 20 dan 30 m/s. Hasil uji coba dari tiap skenario yang dilakukan akan menghasilkan sebuah *tracefile* yang nantinya akan dilakukan analisis perhitungan metrik *Packet Delivery Ratio* (PDR), *End-to-End* Delay (E2D), dan *Routing Overhead* (RO).



Gambar 3.1 Tahapan Rancangan Simulasi

3.2. Perancangan Skenario

Perancangan skenario uji coba MANET diawali dengan melakukan pembuatan skenario pada *mobility generation* yang bersifat *Random Way Point* kemudian membuat koneksi dengan

menggunakan file *traffic-connection* yang sudah terintegrasi pada NS-2. Pada Tugas Akhir ini pembuatan skenario untuk melihat pergerakan antar node dibedakan berdasarkan 3 kecepatan maksimum yaitu 10, 20 dan 30 m/s. Penjelasan untuk perancangan skenario pada *mobility generator* dan pembuatan koneksi antar node-nya dijelaskan sebagai berikut:

3.2.1. Skenario Node-Movement (Mobility Generation)

Skenario *mobility generation* dibuat dengan men-*generate filenode-movement* yang telah ada pada NS-2 atau *tools*-nya biasa disebut 'setdest' dan digunakan dalam file Tcl selama simulasi pada NS-2 sebagai bentuk pergerakan node yang berpindah-pindah.

Tabel 3.1 Parameter Skenario Node-Movement

No.	Parameter	Spesifikasi
1	Jumlah Node	50, 75, 100, 125
2	Waktu Simulasi	100 detik
3	Area	800 m x 800 m
3	Kecepatan Maksimal	- 10 m/s - 20 m/s - 30 m/s
5	Sumber Traffic	CBR
6	Waktu Jeda (dalam detik)	10
7	Ukuran Paket	512 bytes
8	Rate Paket	4 paket per detik
9	Jumlah maksimal koneksi	1
10	Model mobilitas yang digunakan	*Random Way Point*

3.2.2. Traffic-Connection Pattern Generation

Traffic-Connection dibuat dengan menjalankan program cbgren.tcl yang telah terintegrasi pada NS-2 dan digunakan sebagai koneksi untuk menghubungkan antar node yang ada pada skenario selama simulasi pada NS-2.

Tabel 3.2 Parameter Traffic-Connection Pattern

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

3.3. Perancangan Simulasi pada NS-2

Pada perancangan kode NS-2 dengan konfigurasi MANET, dilakukan penggabungan skenario mobilitas dan *traffic-connection* dengan skrip TCL yang diberikan parameter-parameter untuk membangun percobaan simulasi MANET pada NS-2. Berikut adalah beberapa parameter simulasi perancangan sistem MANET yang akan digunakan dapat dilihat padaTabel 3.3.

Tabel 3.3 Parameter Simulasi pada NS-2

No.	Parameter	Spesifikasi
1	*Network simulator*	NS- 2.35
2	Protokol *Routing*	DSR
3	Waktu simulasi	100 detik
4	Waktu Pengiriman Paket Data	-*TwoRayGround* = 0 – 100 detik
5	Area simulasi	800 m x 800 m

No.	Parameter	Spesifikasi
6	Banyak node	50, 75, 100, 125
7	Radius transmisi	100 m
8	Tipe koneksi	UDP
9	Tipe data	*Constant Bit Rate* (CBR)
10	*Source / Destination*	Statik (Node 1 / Node2)
11	Kecepatan generasi paket	1 paket per detik
12	Ukuran paket data	512 bytes
13	Protokol MAC	802.11p
14	Mode Transmisi	*-TwoRayGround*
15	Tipe Antena	*OmniAntenna*
16	Tipe *Interface Queue*	*CMUPriQueue*
17	Tipe Peta	MANET (random way point)
18	Tipe kanal	*Wireless channel*
19	Tipe *trace*	*Old Wireless Format Trace*

3.4. Perancangan Metrik Analisis

Metrik yang akan dianalisis pada Tugas Akhir ini adalah *Packet Delivery Ratio* (PDR), *End-to-End Delay* (E2D), dan *Routing Overhead* (RO). Penjelasannya sebagai berikut:

3.4.1. Packet Delivery Ratio (PDR)

PDR dihitung dari perbandingan antara paket yang dikirim dengan paket yang diterima. PDR dihitung dengan menggunakan persamaan (1) dimana received adalah banyaknya paket data yang diterima dan sent adalah banyaknya paket data yang dikirimkan.

$$\text{Packet Delivery Ratio} = \frac{\text{packet received}}{\text{packet sent}} \times 100\% \quad \textbf{(1)}$$

3.4.2. End-to-End Delay (E2D)

E2D dapat dihitung dari rata-rata *delay* antara waktu paket diterima dan waktu paket dikirim. E2D dihitung dengan menggunakan persamaan (2) dimana $t_{received[i]}$ adalah waktu penerimaan paket dengan urutan / id ke-i, $t_{sent[i]}$ adalah waktu pengiriman paket dengan urutan / id ke-i, dan sent adalah banyaknya paket data yang dikirimkan.

$$\text{End to End} = \frac{\sum_{i \le sent}^{i=0} t_{received[i]} - t_{sent[i]}}{\text{packet sent}} \quad \textbf{(2)}$$

3.4.3. Routing Overhead (RO)

Routing Overhead adalah jumlah paket kontrol *routing* yang ditransimisikan per data paket yang terkirim ke tujuan selama simulasi terjadi. RO dihitung berdasarkan jumlah paket *routing* yang ditransmisikan. Baris yang mengandung *routing overhead* pada *tracefile* ditandai dengan paket yang bertipe *send* (s) / *forward* (f) dan terdapat header paket dari protokol DSR.

BAB IV
IMPLEMENTASI

Bab ini membahas implementasi perancangan perangkat lunak dari aplikasi ini yang merupakan penerapan data, kebutuhan, dan alur sistem yang mengacu pada desain dan perancangan yang telah dibahas sebelumnya.

4.1. Lingkungan Pembangunan Perangkat Lunak

Pembangunan perangkat lunak dilakukan pada lingkungan pengembangan sebagai berikut:

4.1.1. Lingkungan Perangkat Lunak

Adapun perangkat lunak yang digunakan untuk pengembangan sistem adalah sebagai berikut:

- Sistem Operasi *Ubuntu 16.04 LTS 64 bit* untuk lingkungan NS-2;
- *Network Simulator 2* versi 2.35.

4.1.2. Lingkungan Perangkat Keras

Spesifikasi perangkat keras yang digunakan untuk implementasi perangkat lunak Tugas Akhir adalah sebagai berikut:

- *Processor Intel*(R) *Core*(TM) i5-2450M CPU @2.50GHz;
- Media penyimpanan sebesar 500GB;
- RAM sebesar 8 GB DDR3.

4.2. Implementasi Skenario

Implementasi skenario mobilitas MANET dipelajari dalam kondisi yang berbeda-beda seperti jumlah node yang berbeda dan kecepatan yang berbeda. Model yang digunakan untuk studi simulasi pada jaringan MANET adalah *mobility generation*, yang digunakan untuk mempelajari mempelajari pengaruh mobilitas dari *node* pada kinerja keseluruhan jaringan dan *traffic-connection generation*, yang digunakan untuk mempelajari pengaruh beban *traffic* pada jaringan. Implementasi skenarionya adalah sebagai berikut:

4.2.1. Skenario FileNode-Movement (Mobility Generation)

Dalam implementasi skenario pada *mobility generation* menggunakan *tools generate default* yang dimiliki oleh NS-2 yaitu 'setdest'. File skenario *node-movement* (*mobility generation*) digunakan untuk setiap simulasi yang ditandai dengan jeda waktu. Untuk mempelajari efek mobilitas, simulasi dilakukan dengan pola gerakan yang dihasilkan dari kecepatan maksimal yang berbeda. Setting pada file skenario pergerakan *node* sesuai dengan mobilitas yang berbeda, dibuat dengan memvariasikan kecepatan maksimal. Program 'setdest' pada NS-2 digunakan untuk menghasilkan *filenode-movement* dengan menggunakan algoritma *Random Way Point*. Sebagai contoh format *command line* pada Gambar 4.1 yang digunakan untuk menghasilkan gerakan acak pada *node* adalah sebagai berikut:

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

Gambar 4.1 Format Command Line 'setdest'

Ketentuan-ketentuan yang diuji cobakan pada skenarionya berturut-turut adalah versi 'setdest' *simulator* yaitu 1, jumlah *node* dalam skenario yaitu 50, 75, 100, dan 125 waktu jeda (*pause time*) yaitu 10 detik, kecepatan maksimalnya yaitu skenario A sebesar 10 m/s, skenario B sebesar 20 m/s dan skenario C sebesar 30 m/s, waktu simulasi yaitu 100 detik, panjang dan lebar maksimal area simulasi yaitu 800 meter. Kemudian file mobilitas yang dihasilkan disimpan dalam direktori " ~ ns /indep-utils/CMU-scen-gen/setdest / ". Pada Gambar 4.2 dapat dilihat contoh implementasi *command line* pada 'setdest' dengan berbagai kecepatan maksimal yang berbeda sesuai dan *node* sebanyak 50 sampai 125. Dan untuk setiap kecepatan maksimal dibuat 5 buah file.

```
./setdest -v 1 -n 50 -p 10 -M 10 -t 100 -x 800 -y
800 > sc_n50_m10_1
~
~
./setdest -v 1 -n 50 -p 10 -M 30 -t 100 -x 800 -y
800 > sc_n125_m10_5
```

Gambar 4.2 Implementasi pada 'setdest'

Pada Gambar 4.3 terdapat potongan dari file mobilitas sc_n50_m10_1 hasil *generate* 'setdest' yang menunjukkan penempatan awal setiap *node* dalam sebuah area dan dinyatakan dalam sumbu X, Y dan Z adalah sebagai berikut:

```
#
# nodes: 50, pause: 10.00, max speed: 10.00, max x:
800.00, max y: 800.00
#
$node_(0) set X_ 615.278253512894
$node_(0) set Y_ 187.070083923307
$node_(0) set Z_ 0.000000000000
$node_(1) set X_ 425.359792372872
$node_(1) set Y_ 170.175747991669
$node_(1) set Z_ 0.000000000000
```

Gambar 4.3 Posisi Node dalam X, Y dan Z

Sedangkan untuk Gambar 4.4 menunjukkan pergerakan *node* tersebut selama waktu simulasi dijalankan untuk setiap *node* diberikan posisi awal dan berkelanjutan untuk menentukan pergerakan *node* berikutnya. Dari potongan dari sc_n50_m10_1 pada Gambar 4.4, baris pertama mendefinisikan untuk *node*(0) pada detik ke 10 mulai bergerak ke arah tujuan (340.456, 696.77) dengan kecepatan 6.38 m/s. Baris perintah ini dapat digunakan untuk mengubah arah dan kecepatan pada pergerakan mobilitas *node*.

```
$ns_ at 10.000000000000 "$node_(0) setdest
340.456610269606 696.779013641652 6.381346801603"
```

Gambar 4.4 Perpindahan/Pergerakan Node

Kemudian pada Gambar 4.5 terdapat potongan dari sc_n50_m10_1 yang menunjukkan penentuan GOD untuk setiap node. GOD sendiri merupakan kepanjangan dari *General Operations Director*, yang berguna untuk menyimpan informasi global tentang jumlah dan pergerakan *node*. Saat ini, objek GOD hanya digunakan untuk menyimpan *array* terpendek dari *hop* yang diperlukan untuk mencapai dari satu *node* ke *node* yang lain.

```
$god_ set-dist 0 1 1
$god_ set-dist 0 2 2
$god_ set-dist 0 3 4
$god_ set-dist 0 4 3
$god_ set-dist 0 5 1
```

Gambar 4.5 Pembuatan GOD untuk Setiap Node

Informasi yang dimuat ke dalam objek GOD dari *filenode-movement* ditunjukkan pada Gambar 4.6. Pada baris pertama dari potongan file sc_n50_m10_1 tersebut, digunakan untuk memuat objek GOD dengan informasi yaitu jalan terpendek antara *node* 4 dan *node* 46 berubah menjadi 2 *hop* pada detik ke-10.076902213061.

```
$ns_ at 10.076902213061 "$god_ set-dist 4 46 2"
$ns_ at 10.090818302061 "$god_ set-dist 13 47 2"
$ns_ at 10.090818302061 "$god_ set-dist 20 47 1"
$ns_ at 10.653488979610 "$god_ set-dist 6 9 2"
$ns_ at 10.653488979610 "$god_ set-dist 6 19 2"
```

Gambar 4.6 Access Point

4.2.2. File Traffic-Connection Pattern Generation

Dalam implementasi *random traffic-connection generation* untuk TCP dan CBR dapat di setting dengan pergerakan antar *node* menggunakan skrip *traffic-scenario generator*. Skrip *traffic generator* ini terdapat pada direktori ~ns/indep-utils/cmu-scen-gen dan disimpan dalam bentuk *file* cbrgen.tcl. File ini digunakan untuk membuat *traffic-connection* CBR pada jaringan pergerakan antar *node*. Pada saat menjalankan perintah pada *filetraffic-connection* cbrgen.tcl ini, harus mendefinisikan kan tipe *traffic-connection*-nya yaitu CBR, banyaknya *node* dan koneksi maksimal yang ada pada jaringan tersebut, *random seed* dan pada koneksi CBR, *rate* yang nilai kebalikannya digunakan untuk menghitung waktu *interval* antar paket CBR yang lalu disimpan dalam sebuah *file traffic*. Format *command line* pada Gambar 4.7 yang digunakan untuk menghasilkan gerakan acak pada *node* adalah sebagai berikut:

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

Gambar 4.7 Format Command Line cbrgen.tcl

Kemudian pada Gambar 4.8 merupakan bentuk implementasi untuk menjalankan cbrgent.tcl untuk membuat file koneksi CBR diantara 2 *node*, memiliki maksimal 1 koneksi dengan nilai *seed* 1.0 dan jumlah paket per detik sebanyak 0.25 yang disimpan dalam cbrcoba yang nantinya akan digunakan pada saat simulasi NS-2.

```
ns cbrgen.tcl CBR –nn 2 –seed 1.0 –mc 1 –rate 0.25
> cbrcoba
```

Gambar 4.8 Implementasi Koneksi cbrgen.tcl

Kemudian Gambar 4.9 menunjukkan *output*-nya yang disimpan dalam cbrcoba sehingga menghasilkan koneksi CBR dan menggunakan Agent UDP. Koneksi UDP di sini merupakan konfigurasi antara *node* acak ke-1 dan 2. *Interval* pengiriman paket

data dilatkukan setiap satu detik dengan besar paket data 512 *byte* dan maksimal pengiriman paket 10.000.

```
#
# nodes: 2, max conn: 1, send rate: 0.25, seed: 1.0
#
# 1 connecting to 2 at time 2.5568388786897245
#
set udp_(0) [new Agent/UDP]
$ns_ attach-agent $node_(1) $udp_(0)
set null_(0) [new Agent/Null]
$ns_ attach-agent $node_(2) $null_(0)
set cbr_(0) [new Application/Traffic/CBR]
$cbr_(0) set packetSize_ 512
$cbr_(0) set interval_ 1
$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"
```

Gambar 4.9 Output cbrcoba

4.3. Implementasi Simulasi pada NS-2

Untuk mensimulasikan MANET dalam lingkungan NS-2, langsung dapat dilakukan karena telah tersedia protokol *routing* DSR. Parameter-parameter tersebut dibuat menggunakan bahasa *Tcl*/*Otcl*.

Pada Gambar 4.10 menunjukkan skrip konfigurasi awal parameter-parameter yang diberikan untuk menjalankan MANET pada NS-2. Baris pertama merupakan konfigurasi tipe *channel* yang digunakan yaitu *Wireless Channel* karena menggunakan MANET. Baris kedua merupakan tipe transmisi yang digunakan yaitu *TwoRayGround* untuk pengujian simulasi. Tipe *Mac* yang digunakan adalah *Mac* 802.11. Untuk protokol DSR, *interface queue* yang digunakan adalah *CMUPriQueue*, selain dari itu akan menyebabkan *Segmentation Fault* . Baris ke-8 sampai baris ke-16 berturut-turut merupakan koordinat x serta koordinat y yang besarnya 800 meter, jumlah paket dan *node* yaitu 50 *node*, protokol *routing* yang digunakan yaitu DSR, besarnya *seed* yaitu 0.0, waktu

simulasi diakhiri pada detik ke-100, *filetraffic-connection* yang digunakan yaitu cbrcoba dan terakhir adalah file skenario *node-movement* (*mobility generation*) yang digunakan adalah sc_n50_m10_1.

```
set val(chan)   WirelessChannel;
set val(prop)   TwoRayGround;
set val(netif)  Phy/WirelessPhy;
set val(mac)    Mac/802_11;
set val(ifq)    CMUPriQueue;
set val(ll)     LL;
set val(ant)    Antenna/OmniAntenna;
set opt(x)      800;
set opt(y)      800;
set val(ifqlen) 50;
set val(nn)     50;
set val(seed)   0.0;
set val(rp)     DSR;
set val(stop)   100
set val(cp)     "cbrcoba";
set val(sc)     "sc_n50_m10_1";
```

Gambar 4.10 Konfigurasi awal parameter NS-2

Pada Gambar 4.11 merupakan pengaturan dari *transmission range* yang digunakan pada simulasi. Nilai yang diubah ialah *RXThresh_* (*Receiver Sensitivity Threshold*). Nilai `1.42681e-08` pada variabel tersebut memiliki artian bahwa *range* atau jangkauan yang dapat dicapai ialah sejauh 100 meter.

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

Gambar 4.11 Konfigurasi Transmission Range pada NS-2

```
# Initialize Global Variables
# create simulator instance
set ns_       [new Simulator]

# create trace object for ns
set tracefd   [open DSR_sc_n50_m10_1.tr w]

$ns_ trace-all $tracefd

# set up topology object
set topo     [new Topography]
$topo load_flatgrid $opt(x) $opt(y)

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

# Configure node
$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 OFF \
               -movementTrace ON \

for {set i 0} {$i < $val(nn)} {incr i} {
  set node_($i) [$ns_ node]
  $node_($i) random-motion 0 ;# disable random
motion
}
```

Gambar 4.12 Konfigurasi TraceFile dan Pergerakan Node pada NS-2

Skrip yang ditunjukkan pada Gambar 4.12 merupakan skrip untuk pengaturan variabel *global*, membuat *topology*, *GOD*, dan *output* akhir *tracefile*.

Skrip pada Gambar 4.13 merupakan bagian akhir dari keseluruhan skrip yang digunakan untuk menginisialisasi penempatan awal *node-node* yang dibut pada skenario *node-movement* (*mobility generation*), pergerakan *node* selama simulasi dilakukan dan melakukan konfigurasi pengiriman paket data yang dilakukan yang nantinya dihasilkan pada *fileoutput* .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 dikonfigurasi sebelumnya pada Gambar 4.10

```
# 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

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

    # 20 defines the node size, 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).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
```

Gambar 4.13 Konfigurasi pengiriman paket data NS-2

Pada Gambar 4.14 merupakan contoh skrip untuk 802.11p

```
# 802.11p default parameters

Phy/WirelessPhyExt set CSThresh_
3.9810717055349694e-13
Phy/WirelessPhyExt set Pt_                      0.1

Phy/WirelessPhyExt set freq_                    5.9e+9
Phy/WirelessPhyExt set noise_floor_             1.26e-13

Phy/WirelessPhyExt set L_                       1.0

Phy/WirelessPhyExt set PowerMonitorThresh_
3.981071705534985e-18
Phy/WirelessPhyExt set HeaderDuration_          0.000040

Phy/WirelessPhyExt set BasicModulationScheme_   0
Phy/WirelessPhyExt set PreambleCaptureSwitch_   1
Phy/WirelessPhyExt set DataCaptureSwitch_       1
Phy/WirelessPhyExt set SINR_PreambleCapture_    3.1623

Phy/WirelessPhyExt set SINR_DataCapture_        10.0

Phy/WirelessPhyExt set trace_dist_              1e6

Phy/WirelessPhyExt set PHY_DBG_                 0

Mac/802_11Ext set CWMin_                        15
Mac/802_11Ext set CWMax_                        1023
Mac/802_11Ext set SlotTime_                     0.000013
Mac/802_11Ext set SIFS_                         0.000032
Mac/802_11Ext set ShortRetryLimit_              7
Mac/802_11Ext set LongRetryLimit_               4
Mac/802_11Ext set HeaderDuration_               0.000040
Mac/802_11Ext set SymbolDuration_               0.000008
Mac/802_11Ext set BasicModulationScheme_        0
Mac/802_11Ext set use_802_11a_flag_             true
Mac/802_11Ext set RTSThreshold_                 2346
Mac/802_11Ext set MAC_DBG                       0
```

Gambar 4.14 Skrip untuk 802.11p

Hasil yang diperoleh dari eksekusi berkas .tcl berupa *tracefile* berbentuk .tr. File.tr inilah yang akhirnya akan dianalisis parameter-parameternya didalamnya yang berupa PDR, E2D, dan RO.

4.4. Implementasi Metrik Analisis

Hasil menjalankan skenario MANET dalam NS-2 dalam bentuk *TraceFile* berekstensi .tr dianalisis dengan 3 cara yaitu *Packet Delivery Ratio* (PDR), *End-to-End Delay* (E2D), dan *Routing Overhead* (RO). Implementasi dari tiap cara menggunakan bahasa pemrograman AWK dan dijelaskan sebagai berikut.

4.4.1. Packet Delivery Ratio (PDR)

Implementasi nilai dari *Packet Delivery Ratio* didapatkan dari perhitungan setiap baris yang terjadi *event* pengiriman dan penerimaan paket data yang dikirim melalui agen pada *tracefile*. Pada skrip dilakukan pengecekan pada setiap baris yang terdapat *string* AGT karena *event* tersebut berhubungan dengan paket data. Paket data yang dikirim dan diterima dibedakan dari kolom pertama pada baris yang telah dicek. Setelah pembacaan setiap baris dalam *tracefile* selesai dilakukan, selanjutnya dilakukan perhitungan PDR dengan persamaan yang telah dijelaskan sebelumnya.

Psedeucode PDR terlihat pada **Error! Reference source not found.** dan implementasinya pada tugas akhir dapat dilihat pada Lampiran.

```
ALGORITMA PDR(tracefile)
//Input: tracefile simulasi skenario
//Ouput: jumlah packet sent, packet received, dan //
         PDR
BEGIN (
sent ←0
recv ←0
recv_id ←0
pdr←0)

(if ($1 == "s" and $3 == "_1_" and $4 == "AGT" and
       $7 == "cbr" )
       sent + 1;
if ($1 == "r" and $3 == "_2_" and $4 == "AGT" and
       $7 == "cbr")
       recv +1;
)
END (
pdr ← ( recv / sent ) * 100
print sent
print recv
print pdr)


if ($1 == "r" and $3 == "_0_" and $4 == "AGT" and
       $7 == "AGT")
       recv +1;
)
END (
pdr ← ( recv / sent ) * 100
print sent
print recv
print pdr)
```

Gambar 4.15 Pseudeucode PDR

Contoh perintah untuk analisa PDR dari *tracefile* 802.11p dengan protokol DSR dan kecepatan maksimal perpindahan *node* sebesar 10 m/s seperti pada **Error! Reference source not found.**5.

```
awk -f ALG_PDR.awk DSR_sc_n50_m10_1.tr
```

Gambar 4.16 Perintah Menjalankan Skrip ALG_PDR.awk

Contoh *output* dari menjalankan skrip tersebut ditunjukkan pada **Error! Reference source not found.**7.

```
Packet Sent =  318
Packet Received =  237
Packet delivery ratio =  74.528 %
```

Gambar 4.17 Hasil Running skrip ALG_PDR.awk

4.4.2. End-to-End Delay (E2D)

Untuk *End-to-End Delay*, implementasi Dalam pembacaan baris *tracefile* terdapat 5 (lima) kolom yang harus diperhatikan, yaitu kolom pertama yang berisi penanda *event* pengiriman atau penerimaan, kolom kedua yang berisi waktu terjadinya *event*, kolom keempat yang berisi informasi *layer* komunikasi paket, kolom keenam yang berisi ID paket dan kolom ketujuh yang berisi tipe paket.

Dari setiap paket yang ada terdapat *delay* yang dihitung dengan cara mengurangi waktu penerimaan dengan waktu pengiriman berdasarkan ID paket. Hasil pengurangan waktu dari masing-masing paket dijumlahkan dan dibagi dengan jumlah paket CBR yang ID-nya terlibat dalam perhitungan dalam mencari nilai *delay* tersebut seperti pada Persamaan yang telah dijelaskan sebelumnya. *Psedeucode* E2D ditunjukkan pada **Error! Reference source not found.**18 dan implementasinya dapat dilihat pada Lampiran.

```
ALGORITMA E2D(tracefile)
//Input: tracefile simulasi skenario
//Ouput: jumlah packet receieved, total delay, dan
//        E2D
BEGIN (
for i in pkt_id
pkt_id[i] ← 0
for i in pkt_sent
       pkt_sent[i] ← 0
for i in pkt_recv
       pkt_recv[i] ← 0
delay = avg_delay ← 0
recv ← 0
recv_id ← 0)

(if ($1 == "s" and $3 == "_1_" and $4 == "AGT" and
     $7 == "cbr)
     pkt_sent[$6]← $2

if ($1 == "r" and $3 == "_2_" and $4 == "AGT" and
    $7 == "cbr" and  recv_id != $6 )
              recv + 1
recv_id ← $6
              pkt_recv[$6] ← $2;
)
END (
for i in pkt_recv
       delay += pkt_recv[i] - pkt_sent[i]
avg_delay←delay / recv;
print  recv
print delay
print avg_delay
)
```

Gambar 4.18 Pseudeucode E2D

Contoh perintah untuk analisa E2D dari 802.11p dengan protokol DSR dan kecepatan maksimal perpindahan *node* sebesar 10 m/s seperti **Error! Reference source not found.**19 dan hasilnya dapat dilihat pada **Error! Reference source not found.**0.

```
awk -f ALG_E2D.awk DSR_sc_n50_m10_1.tr
```

Gambar 4.19 Perintah Menjalankan skrip ALG_E2D.awk

```
Packet = 236
Delay = 314.461 second
Average Packet Delay =  1.332462 second
```

Gambar 4.20 Hasil Running skrip ALG_E2D.awk

4.4.3. Routing Overhead (RO)

Implementasi perhitungan metrik *Routing Overhead* DSR dihitung apabila kondisi yang ada terpenuhi yaitu pada kolom pertama diawali dengan huruf "s" yang berarti *send packet* atau huruf "f" yang berarti *forward packet*, kolom ke-4 mengandung huruf "RTR" yang berarti paket *routing* dan kolom ke-7 mengandung "DSR" yang berarti paket *routing* DSR. seperti pada **Error! Reference source not found.**1. Implementasi yang telah dilakukan dapat dilihat pada Lampiran.

```
ALGORITMA RO-DSR(tracefile)
//Input: tracefile simulasi skenario
//Ouput: jumlah routing overhead protokol DSR
BEGIN (
rt_pkts ← 0)
(if (($1=="s" || $1=="f") && $4 == "RTR" &&
      $7 == "DSR")
      rt_pkts + 1)
)
END (
print rt_pkts)
```

Gambar 4.21 Pseudeucode RO

Contoh perintah untuk analisis RO dari *tracefile* model 802.11p dengan protokol DSR dan kecepatan maksimal perpindahan *node* sebesar 10 m/s seperti **Error! Reference source not found.**2.

```
awk -f ALG_RO.awk DSR_sc_n50_m10_1.tr
```

Gambar 4.22 Perintah menjalankan skrip ALG_RO.awk

Contoh output dari menjalankan skrip diatas seperti pada **Error! Reference source not found.**3.

```
Total packet : 7670
```

Gambar 4.23 Hasil running skrip ALG_RO.awk

(halaman ini sengaja dikosongkan)

BAB V
PENGUJIAN DAN EVALUASI

Bab ini membahas mengenai pengujian dari skenario NS-2 yang telah dibuat. Pengujian fungsionalitas akan dibagi ke dalam beberapa skenario pengujian.

5.1. Lingkungan Pengujian

Uji coba dilakukan pada laptop yang telah terpasang sistem operasi yaitu *Windows* dan *Linux* pada aplikasi *Virtual Box*. Spesifikasi komputer yang digunakan ditunjukkan pada Tabel 5.1.

Tabel 5.1 Spesifikasi Komputer yang Digunakan

Komponen	**Spesifikasi**
CPU	Intel(R) Core(TM) i5-2450M CPU @2.50GHz
Sistem Operasi	*Windows* 7 *Ultimate* 64, *Linux Ubuntu* 16.04 LTS 64-*bit* (NS-2, DSR, *Mobility Generation*, *Traffic-Connection Generation*, *TwoRayGround*)
Memori	8 GB DDR3
Media Penyimpanan	500 GB

5.2. Kriteria Pengujian

Pengujian pada skenario yang dihasilkan oleh *mobility generator default* dari NS-2 menggunakan beberapa macam kriteria. Pada Tabel 5.2 menunjukkan beberapa kriteria-kriteria yang telah ditentukan di dalam skenario.

Tabel 5.2 Kriteria Pengujian

Kriteria	**Spesifikasi**
Skenario	MANET (*Random Way Point*), *mobility generator*
Kecepatan Maksimal Perpindahan *Node* (m/s)	10, 20, 30

Kriteria	**Spesifikasi**
Jumlah Percobaan	5 kali
Jarak antara 2 *Node* Acak	Acak
Posisi Awal *Node*	Acak
Pergerakan	Acak
Protokol *Routing*	DSR

5.3. Analisis Packet Delivery Ratio (PDR)

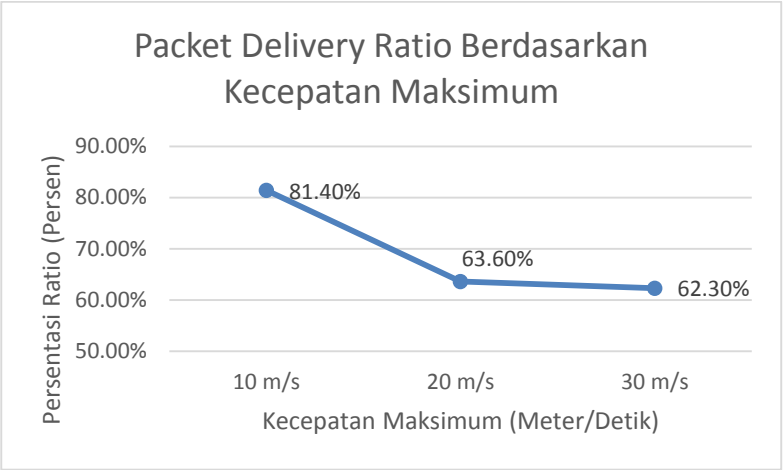
Tracefile dari hasil menjalankan program skenario *node-movement* (*mobility generation*) kemudian dianalisis nilai PDR melalui script ALG_PDR.awk. Hasil tiap perhitungan PDR skenario dihitung menjadi seperti pada Tabel 5.3.

Tabel 5.3 Packet Delivery Ratio Skenario Node-Movement

Kecepatan Maksimal Perpindahan *Node* (m/s)	**PDR (%)**
10	81.423
20	63.616
30	62.259

Pada Tabel 5.3 dan

Gambar 5.1 menunjukkan performa PDR yang dihasilkan oleh protokol DSR dan 802.11p pada jaringan MANET dengan menggunakan skenario *node-movement* (*mobility generation*) yang bersifat *Random Way Point*. Terlihat bahwa PDR yang dihasilkan berdasarkan kecepatan maksimum semakin menurun dengan bertambahnya kecepatan maksimal perpindahan node.

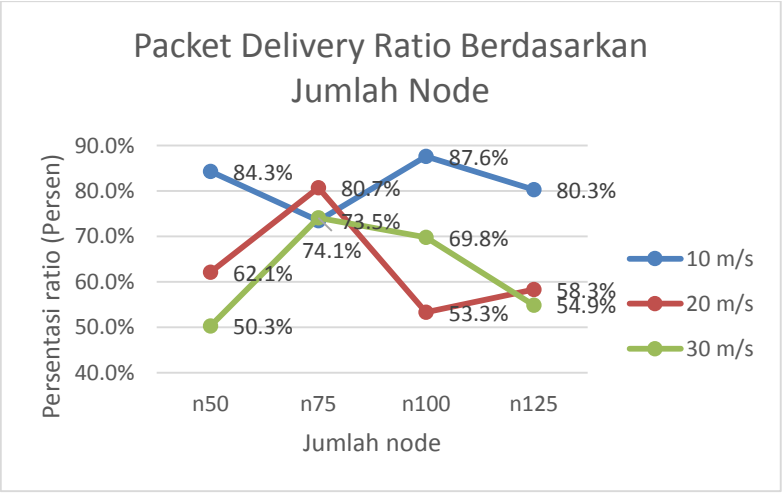


Gambar 5.1 Grafik PDR Berdasarkan kecepatan maksimum

Nilai PDR yang dihasilkan cukup tinggi. Dari grafik bisa dilihat bahwa pada kecepatan 10 m/s, rata-rata yang dihasilkan sampai pada 81,423% *packet delivered*. Sedangkan pada kecepatan 20 m/s, terjadi penurunan 17.807% paket yang diantar, dan pada kecepatan 30 m/s, hanya terjadi penurunan sebesar 1.357% dibandingkan kecepatan 20 m/s.

Terlihat bahwa nilai PDR yang dihasilkan oleh kecepatan rendah sangat tinggi pada skenario *node-movement* (*mobility generator*) MANET. Nilai PDR yang dihasilkan pada kecepatan cukup tinggi mengalami penurunan yang cukup drastis, dan pada kecepatan tinggi berikutnya nilai PDR yang dihasilkan cenderung sangat stabil.

Pada gambar 5.2 akan menunjukan performa *Packet Deliver Ratio* berdasarkan jumlah *node*, yang dapat merefleksikan Grafik PDR berdasarkan kecepatan maksimum



Gambar 5.2 Grafik PDR Berdasarkan Jumlah Node

Dari grafik PDR berdasarkan jumlah *node* bisa dilihat bahwa tren yang ditunjukan konsisten dengan rata-rata akhir dari grafik, hanya beberapa kasus seperti nilai *node* 100 dengan kecepatan 20 m/s yang memiliki nilai PDR 53.332% dan nilai *node* 75 dengan kecepatan 20 m/s yang memiliki nilai PDR 80.673%.

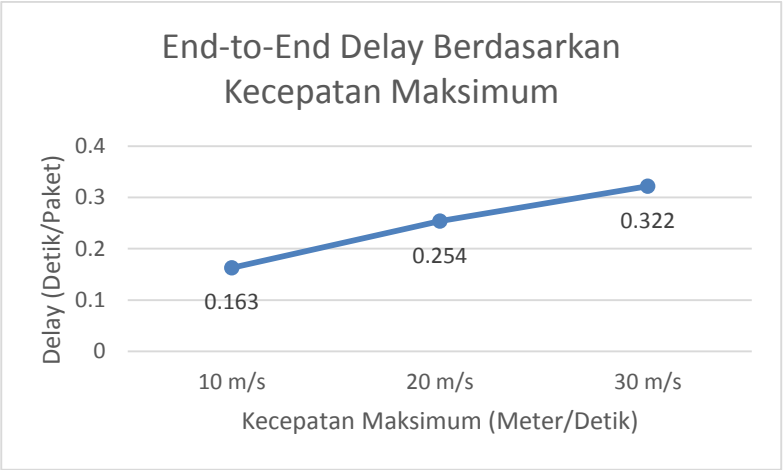
Hal yang dapat mempengaruhi penurunan ataupun peningkatan nilai PDR yang dihasilkan adalah penempatan dan pergerakan secara acak yang di implementasikan pada skenario yang dihasilkan oleh *file node-movement* (*mobility generation*).

5.4. Analisis End-to-End Delay (E2D)

Tracefile hasil menjalankan program skenario *node-movement* (*mobility generation*) menggunakan protokol DSR dan 802.11p kemudian dianalisis nilai *End-to-End Delay* melalui skrip ALG_E2D.awk. Hasil tiap perhitungan E2D scenario berdasarkan kecepatan maksimum ditabulasikan dan dirata-ratakan menjadi seperti pada Tabel 5.5.

Tabel 5.4 End to End Delay Skenario Node-Movement

Kecepatan Maksimal Perpindahan *Node* (m/s)	**E2D (detik/paket) 802.11p**
10	0.163
20	0.254
30	0.322



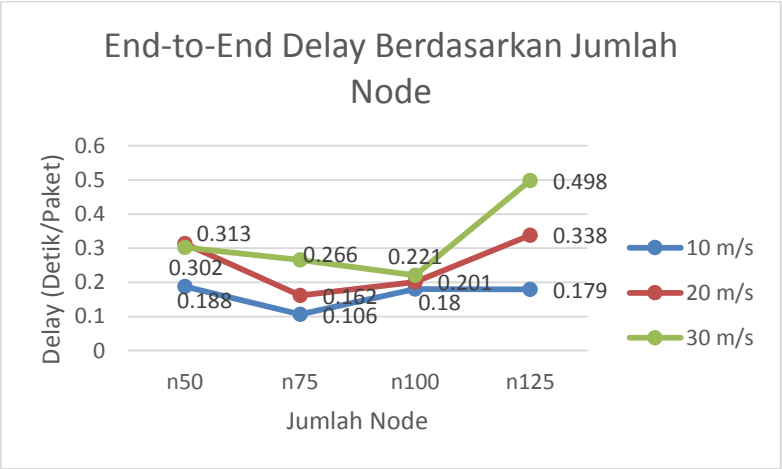
Gambar 5.3 Grafik E2D Berdasarkan kecepatan Maksimum

Performa *End-to-End Delay* pada protokol DSR dan 802.11p menunjukan kenaikan nilai yang stabil. Nilai *End-to-End Delay* pada kecepatan 10 m/s menunjukan nilai 0.163, dan terlihat pada kecepatan 20 m/s, nilai pada grafik menunjukan nilai 0.254, ada peningkatan nilai sebesar 0.091, sedangkan pada kecepatan 30 m/s menunjukan nilai sebesar 0.322, terjadi peningkatan dari kecepatan 20 m/s sebesar 0.068.

Performa *End-to-End Delay* menunjukan peningkatan nilai semakin cepat kecepatan perpindahan *node*, maka semakin besar pula nilai *delay*. Hal ini dapat terjadi akibat mobilitas pengiriman paket antar *node* yang sangat dinamis pada DSR. Mobilitas yang

sangat dinamis ini memungkinkan terjadinya rute putus saat pengiriman paket data sehingga pengiriman paket dimasukkan ke dalam antrian dan menunggu rute baru terbentuk sebelum pengiriman paket data dilanjutkan kembali.

Pada Gambar 5.4 akan ditunjukan grafik End-to-End Delay berdasarkan jumlah node yang dapat merefleksikan grafik rata-rata akhir



Gambar 5.4 Grafik E2D Berdasarkan Jumlah Node

Grafik menunjukan bahwa nilai *delay* sangat stabil kecuali pada jumlah *node* 125 dan kecepatan maksimum 30 m/s yang naik dengan cukup tajam setelah turun dengan stabil. Hal ini dapat dipengaruhi oleh jumlah *node* yang cukup besar yaitu 125 *node* dengan kecepatan acak yang bisa jadi sangat tinggi mendekati 30 m/s.

5.5. Analisis Routing Overhead (RO)

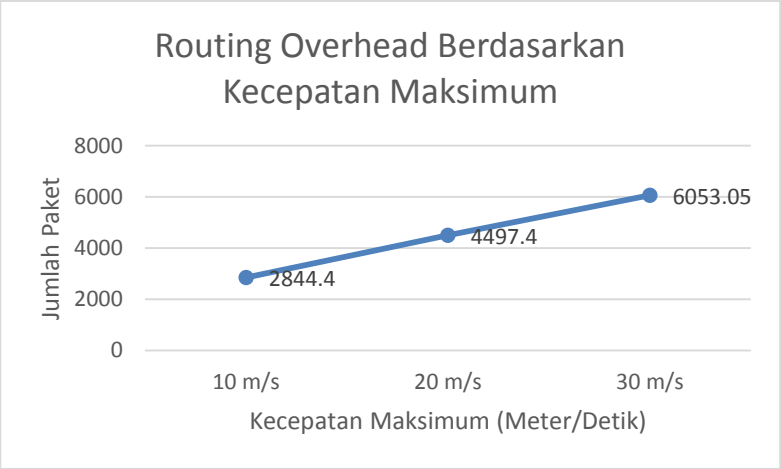
Tracefile hasil menjalankan program skenario *node-movement* (*mobility generation*) menggunakan 802.11p kemudian dianalisis nilai *Routing Overhead* melalui script ALG_RO.awk.

Hasil tiap perhitungan RO berdasarkan kecepatan maksimum ditabulasikan dan dirata-ratakan menjadi Tabel 5.6.

Tabel 5.5 Routing Overhead Skenario Node-Movement

Kecepatan Maksimal Perpindahan *Node* (m/s)	***Routing Overhead* (Paket) 802.11p**
10	2844.4
20	4497.4
30	6053.05

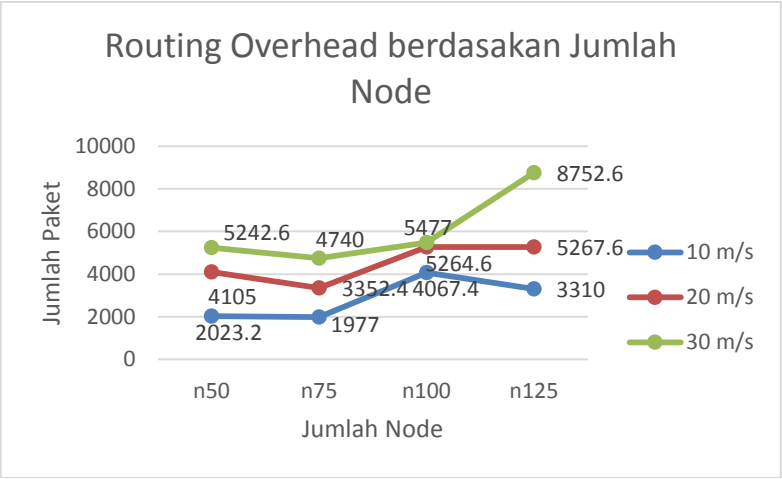
Pada Tabel 5.6 dan **Error! Reference source not found.** menunjukkan pengujian 802.11p untuk nilai *Routing Overhead* yang dihasilkan memiliki nilai yang stabil beranjak naik berdasarkan penambahan kecepatan maksimal perpindahan *node* yang terdapat dalam simulasi. Pada saat kecepatan maksimal perpindahan *node* sebesar 10 m/s nilai *Routing Overhead* yaitu sebanyak 2844.4 paket. Kemudian bertambah pada saat kecepatan maksimal perpindahan *node* sebesar 20 m/s yaitu sebanyak 4497.4 paket. Dan kembali beranjak naik pada kecepatan 30 m/s yaitu senilai 6053.05.



Gambar 5.5 Grafik RO berdasarkan kecepatan maksimum

Dan pada Grafik 5.6 akan ditunjukkan grafik RO berdasakan jumlah *node* yang dapat merefleksikan nilai rata-rata akhir dari *Routing Overhead*. Bisa dilihat bahwa kembali nilai yang didapatkan cukup stabil dan hanya naik secara drastis pada kasus jumlah *node* 125 dengan kecepatan maksimum 30 m/s.

Hal yang mempengaruhi hal ini yaitu jumlah node yang cukup besar dan nilai kecepatan maksimum acak yang mungkin sangat tinggi.



Gambar 5.6 Grafik RO Berdasarkan Jumlah Node

BAB VI
PENUTUP

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

6.1. Kesimpulan

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

1. Skenario MANET yang dihasilkan oleh *node-movement* (*mobility generation*) dan dijalankan menggunakan 802.11p dengan penambahan kecepatan maksimal perpindahan *node* memiliki performa sebagai berikut:
   - Performa *Packet Delivery Ratio* yang dihasilkan semakin menurun secara drastis kemudian stabil pada rentang nilai 17.807% menjadi 1.357% dari kecepatan maksimal perpindahan *node* sebesar 10 m/s hingga 30 m/s.
   - Performa *End to End Delay* yang dihasilkan memiliki nilai *delay* stabil , yaitu sedikit demi sedikit meningkat pada saat kecepatan maksimal perpindahan *node* sebesar 10 m/s menuju kecepatan maksimal 30 m/s.
   - *Routing Overhead* yang dihasilkan memiliki nilai yang juga stabil, yaitu terjadi kenaikan yang cukup banyak pada saat kecepatan maksimal perpindahan *node* sebesar 10 m/s menuju kecepatan maksimal perpindahan *node* sebesar 30 m/s.
2. Hal – hal yang dapat mempengaruhi nilai PDR, E2D dan RO yang dihasilkan adalah:
   - Posisi awal *node* yang dibuat secara acak.
   - Pergerakan *node* yang dibuat secara acak.
   - Lingkungan jaringan yang digunakan.
   - Keadaan sekitar simulasi.

- Ketinggian antena

3. Dari hasil percobaan pada skenario MANET, didapatkan nilai E2D dan RO yang stabil tetapi tidak pada PDR, nilai PDR menurun cukup drastis dari kecepatan 10 m/s menuju kecepatan 20 m/s tetapi menjadi cukup stabil setelahnya.

6.2. Saran

Dalam pengerjaan Tugas Akhir ini terdapat beberapa saran untuk perbaikan serta pengembangan sistem yang telah dikerjakan sebagai berikut:

1. Dapat dilakukan percobaan pada lingkungan VANET untuk penerapan 802.11p
2. Dapat dilakukan pengurangan atau penambahan jumlah *node* dan penambahan jumlah percobaan untuk skenario *node-movement* (*mobility generation*).
3. Dapat dilakukan modifikasi pada parameter-parameter yang digunakan untuk uji coba MANET pada NS-2 seperti modifikasi pada parameter transmission range, modifikasi pada *pause time* selama waktu simulasi berlangsung.

DAFTAR PUSTAKA

[1] D. Maltz, "The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks for IPv4," Microsoft Research, 2007.

[2] V. D. Khairnar and D. K. Kotecha, "Performance of Vehicle-to-Vehicle Communication using IEEE 802.11p in Vehicular Ad-hoc Network Environment," International Journal of Network Security & Its Applications (IJNSA), vol. V, no. 2, pp. 143-170, 2013.

[3] K. Fall and K. Varadhan, The ns Manual (formerly ns Notes and Documentation), Berkeley: The VINT Project, 2011.

[4] D. A. Maltz, "Simulation and Implementation," in On-Demand Routing in Multi-Hop Wireless Mobile Ad Hoc Network, School of Computer Science Carnegie Mellon University, 2001.

[5] http://bengkelubuntu.org/teks/awk/Praktikum%2001%20-%20Berkenalan%20dengan%20AWK.pdf. [Accessed 22 May 2017].

(halaman ini sengaja dikosongkan)

LAMPIRAN

```
$node_(0) set X_ 615.278253512894
$node_(0) set Y_ 187.070083923307
$node_(0) set Z_ 0.000000000000
$node_(1) set X_ 425.359792372872
$node_(1) set Y_ 170.175747991669
$node_(1) set Z_ 0.000000000000
$node_(2) set X_ 239.052616274670
$node_(2) set Y_ 66.925871446864
$node_(2) set Z_ 0.000000000000
$node_(3) set X_ 24.185797067044
$node_(3) set Y_ 669.231329113272
$node_(3) set Z_ 0.000000000000
$node_(4) set X_ 115.645107080737
$node_(4) set Y_ 186.929074414365
$node_(4) set Z_ 0.000000000000
$node_(5) set X_ 670.433723798537
$node_(5) set Y_ 195.518662190969
$node_(5) set Z_ 0.000000000000
$node_(6) set X_ 85.091121145280
$node_(6) set Y_ 706.041135931517
$node_(6) set Z_ 0.000000000000
$node_(7) set X_ 137.092515946190
$node_(7) set Y_ 762.112847743433
$node_(7) set Z_ 0.000000000000
$node_(8) set X_ 400.434699116838
$node_(8) set Y_ 208.160307839826
$node_(8) set Z_ 0.000000000000
$node_(9) set X_ 516.744339531945
$node_(9) set Y_ 676.245688986756
$node_(9) set Z_ 0.000000000000
$node_(10) set X_ 583.740710238476
$node_(10) set Y_ 774.512208881448
$node_(10) set Z_ 0.000000000000
$node_(11) set X_ 212.428890770583
$node_(11) set Y_ 204.976329820947
$node_(11) set Z_ 0.000000000000
```

Gambar 7.1 Posisi node dari potongan Skenario

```
$god_ set-dist 0 1 1
$god_ set-dist 0 2 2
$god_ set-dist 0 3 4
$god_ set-dist 0 4 3
$god_ set-dist 0 5 1
$god_ set-dist 0 6 4
$god_ set-dist 0 7 4
$god_ set-dist 0 8 1
$god_ set-dist 0 9 3
$god_ set-dist 0 10 3
$god_ set-dist 0 11 2
$god_ set-dist 0 12 3
$god_ set-dist 0 13 3
$god_ set-dist 0 14 2
$god_ set-dist 0 15 3
$god_ set-dist 0 16 1
$god_ set-dist 0 17 2
$god_ set-dist 0 18 2
$god_ set-dist 0 19 3
$god_ set-dist 0 20 3
$god_ set-dist 0 21 2
$god_ set-dist 0 22 2
$god_ set-dist 0 23 3
$god_ set-dist 0 24 1
$god_ set-dist 0 25 2
$god_ set-dist 0 26 2
$god_ set-dist 0 27 1
$god_ set-dist 0 28 1
$god_ set-dist 0 29 2
$god_ set-dist 0 30 4
$god_ set-dist 0 31 3
$god_ set-dist 0 32 4
$god_ set-dist 0 33 2
$god_ set-dist 0 34 3
$god_ set-dist 0 35 3
$god_ set-dist 0 36 3
$god_ set-dist 0 37 1
```

Gambar 7.2 Pembuatan GOD setiap node dari potongan Skenario

```
$ns_    at    10.000000000000    "$node_(0)    setdest
340.456610269606 696.779013641652 6.381346801603"
$ns_    at    10.000000000000    "$node_(1)    setdest
745.878420198036 418.222136560409 9.442571900798"
$ns_    at    10.000000000000    "$node_(2)    setdest
78.043218383796 429.357716559061 9.471053970042"
$ns_    at    10.000000000000    "$node_(3)    setdest
682.760130710459 462.277607655558 4.780872912244"
$ns_    at    10.000000000000    "$node_(4)    setdest
95.550018892252 345.862279259449 5.804863485371"
$ns_    at    10.000000000000    "$node_(5)    setdest
308.912375442165 54.249487278027 7.886805075792"
$ns_    at    10.000000000000    "$node_(6)    setdest
525.271471882348 370.418601540632 4.158483374623"
$ns_    at    10.000000000000    "$node_(7)    setdest
586.519167850722 42.871484560256 3.613939097081"
$ns_    at    10.000000000000    "$node_(8)    setdest
106.116915604174 506.392242696506 0.417433291866"
$ns_    at    10.000000000000    "$node_(9)    setdest
313.737794490871 479.637791720373 0.146214410293"
$ns_    at    10.000000000000    "$node_(10)    setdest
362.868759473018 333.636309904128 4.490112740068"
$ns_    at    10.000000000000    "$node_(11)    setdest
628.129641583880 239.401107699478 1.968559992840"
$ns_    at    10.000000000000    "$node_(12)    setdest
240.929332122510 108.824821430157 5.321399613016"
$ns_    at    10.000000000000    "$node_(13)    setdest
707.897218699246 720.942924627133 0.603551619113"
$ns_    at    10.000000000000    "$node_(14)    setdest
739.141433718945 105.921825494557 3.625305284295"
$ns_    at    10.000000000000    "$node_(15)    setdest
566.775283098514 719.572588445409 1.009938819350"
$ns_    at    10.000000000000    "$node_(16)    setdest
769.743140439171 275.348450167216 8.030071142189"
$ns_    at    10.000000000000    "$node_(17)    setdest
574.808132359779 340.269482968387 4.220056428521"
```

Gambar 7.3 Pergerakan setiap node dari potongan Skenario

```
$ns_ at 10.076902213061 "$god_ set-dist 4 46 2"
$ns_ at 10.090818302061 "$god_ set-dist 13 47 2"
$ns_ at 10.090818302061 "$god_ set-dist 20 47 1"
$ns_ at 10.653488979610 "$god_ set-dist 6 9 2"
$ns_ at 10.653488979610 "$god_ set-dist 6 19 2"
$ns_ at 10.653488979610 "$god_ set-dist 6 23 3"
$ns_ at 10.653488979610 "$god_ set-dist 6 31 1"
$ns_ at 10.653488979610 "$god_ set-dist 6 34 3"
$ns_ at 10.653488979610 "$god_ set-dist 6 36 2"
$ns_ at 11.137238265869 "$god_ set-dist 9 34 1"
$ns_ at 11.159235147259 "$god_ set-dist 5 13 4"
$ns_ at 11.159235147259 "$god_ set-dist 5 29 3"
$ns_ at 11.159235147259 "$god_ set-dist 13 27 3"
$ns_ at 11.159235147259 "$god_ set-dist 27 29 2"
$ns_ at 11.308605872626 "$god_ set-dist 15 21 2"
$ns_ at 11.348400718870 "$god_ set-dist 8 28 1"
$ns_ at 11.348400718870 "$god_ set-dist 22 28 2"
$ns_ at 11.348400718870 "$god_ set-dist 25 28 2"
$ns_ at 11.644495080851 "$god_ set-dist 4 16 3"
$ns_ at 11.644495080851 "$god_ set-dist 4 42 2"
$ns_ at 11.644495080851 "$god_ set-dist 16 21 2"
$ns_ at 11.644495080851 "$god_ set-dist 21 42 1"
$ns_ at 11.964110825647 "$god_ set-dist 15 46 2"
$ns_ at 12.042479996750 "$god_ set-dist 5 48 1"
$ns_ at 12.119925426201 "$god_ set-dist 12 25 3"
$ns_ at 12.119925426201 "$god_ set-dist 12 41 4"
$ns_ at 12.119925426201 "$god_ set-dist 25 27 2"
$ns_ at 12.119925426201 "$god_ set-dist 25 31 2"
$ns_ at 12.119925426201 "$god_ set-dist 25 35 2"
$ns_ at 12.119925426201 "$god_ set-dist 25 39 3"
$ns_ at 12.119925426201 "$god_ set-dist 25 40 1"
$ns_ at 12.119925426201 "$god_ set-dist 27 41 3"
$ns_ at 12.119925426201 "$god_ set-dist 31 41 3"
```

Gambar 7.4 Informasi pada GOD dari potongan Skenario

```
1   #
2   # nodes: 10, max conn: 1, send rate: 0.25, seed:
3   1.0
4   #
5   #
6   # 1 connecting to 2 at time 2.5568388786897245
7   #
8   set udp_(0) [new Agent/UDP]
9   $ns_ attach-agent $node_(1) $udp_(0)
10  set null_(0) [new Agent/Null]
11  $ns_ attach-agent $node_(2) $null_(0)
12  set cbr_(0) [new Application/Traffic/CBR]
13  $cbr_(0) set packetSize_ 512
14  $cbr_(0) set interval_ 0.25
15  $cbr_(0) set random_ 1
16  $cbr_(0) set maxpkts_ 10000
17  $cbr_(0) attach-agent $udp_(0)
18  $ns_ connect $udp_(0) $null_(0)
19  $ns_ at 2.5568388786897245 "$cbr_(0) start"
20  #
```

Gambar 7.5 Koneksi yang digunakan pada cbrcoba

```
1   ========================
2   # Define options
3   # ======================
4   set val(chan)  WirelessChannel
5   set val(prop)  TwoRayGround
6   set val(netif) Phy/WirelessPhy
7   set val(mac)   Mac/802_11
8   set val(ifq)   CMUPriQueue
9   set val(ll)    LL
10  set val(ant)   Antenna/OmniAntenna
11  set opt(x)     800  ;# X dimension of the
    topography
12  set opt(y)     800  ;# Y dimension of the
    topography
13  set val(ifqlen)50 ;# max packet in ifq
13  set val(nn) 50;# how many nodes are simulated
    set val(seed) 0.0
14  set val(adhocRouting) DSR
15  set val(stop)        100 ;# simulation time
16  set val(cp)"cbrcoba "      ;#<-- traffic file
17  set val(sc)"sc_n50_m10_1"  ;#<-- mobility file
```

```
18
19  Phy/WirelessPhy     set    RXThresh_ 1.42681e-
20  08 ;#100m
    #
21  ================================================
22  ========================
    # Main Program
23  #
24  ================================================
25  ========================
    # Initialize Global Variables
26  # create simulator instance
27
28  set ns_       [new Simulator]
29
30  # create trace object for ns
31
32  set tracefd   [open DSR_$val(sc).tr w]
33
34  $ns_ trace-all $tracefd
35
36
37  # set up topology object
38  set topo            [new Topography]
39  $topo load_flatgrid $opt(x) $opt(y)
40
41  # Create God
42  set god_ [create-god $val(nn)]
43
44  #global node setting
45  $ns_ node-config -adhocRouting
46  $val(adhocRouting) \
47                   -llType $val(ll) \
48                   -macType $val(mac) \
49                   -ifqType $val(ifq) \
50                   -ifqLen $val(ifqlen) \
51                   -antType $val(ant) \
52                   -propType $val(prop) \
53                   -phyType $val(netif) \
54  -channelType $val(chan) \
55  -topoInstance $topo \
56  -agentTrace ON \
57  -routerTrace ON \
58  -macTrace OFF \
```

```
59  -movementTrace ON \
60
61  # 802.11p default parameters
62
63  Phy/WirelessPhyExt set CSThresh_
64  3.9810717055349694e-13
65  Phy/WirelessPhyExt set Pt_
66  0.1
67  Phy/WirelessPhyExt set freq_
68  5.9e+9
69  Phy/WirelessPhyExt set noise_floor_
    1.26e-13
70  Phy/WirelessPhyExt set L_
71  1.0
72  Phy/WirelessPhyExt set PowerMonitorThresh_
73  3.981071705534985e-18
74  Phy/WirelessPhyExt set HeaderDuration_
75  0.000040
76  Phy/WirelessPhyExt set BasicModulationScheme_
77  0
78  Phy/WirelessPhyExt set PreambleCaptureSwitch_
79  1
80  Phy/WirelessPhyExt set DataCaptureSwitch_
81  1
82  Phy/WirelessPhyExt set SINR_PreambleCapture_
83  3.1623
84  Phy/WirelessPhyExt set SINR_DataCapture_
    10.0
85  Phy/WirelessPhyExt set trace_dist_
    1e6
86  Phy/WirelessPhyExt set PHY_DBG_
87  0
88
89  Mac/802_11Ext set CWMin_
90  15
91  Mac/802_11Ext set CWMax_
92  1023
93  Mac/802_11Ext set SlotTime_
93  0.000013
94  Mac/802_11Ext set SIFS_
95  0.000032
    Mac/802_11Ext set ShortRetryLimit_
96  7
97
```

```
    Mac/802_11Ext set LongRetryLimit_
98  4
    Mac/802_11Ext set HeaderDuration_
99  0.000040
    Mac/802_11Ext set SymbolDuration_
100 0.000008
101 Mac/802_11Ext set BasicModulationScheme_
102 0
    Mac/802_11Ext set use_802_11a_flag_
103 true
    Mac/802_11Ext set RTSThreshold_
104 2346
    Mac/802_11Ext set MAC_DBG
105 0

106 #  Create the specified number of nodes
107 [$val(nn)] and "attach" them
    #  to the channel.
108 for {set i 0} {$i < $val(nn)} {incr i} {
109             set node_($i) [$ns_ node]
110             $node_($i) random-motion 0 ;#
111 disable random motion
    }
112
113 # Define node movement model
114 puts "Loading connection pattern..."
115 source $val(cp)
116
117 # Define traffic model
118 puts "Loading scenario file..."
119 source $val(sc)
120
121 # Define node initial position
122
123 for {set i 0} {$i < $val(nn)} {incr i} {
124
125     # 20 defines the node, must adjust it
126 according to your scenario
127     # The function must be called after
128 mobility model is defined
129
130     $ns_ initial_node_pos $node_($i) 20
131 }
132
```

```
# 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
```

Gambar 7.6 File .tcl untuk Protokol Routing DSR menggunakan 802.11p

```
BEGIN {
sent=0;
recv=0;
pdr=0;
}
{
if ($1 == "s" && $3 == "_1_" && $4 == "AGT" &&
$7 == "cbr")
       {
       sent++;
       }
if ($1 == "r" && $3 == "_2_" && $4 == "AGT" &&
$7 == "cbr")
       {
       recv++;
       }
}
END {
pdr = ( recv / sent ) * 100;
print "Packet Sent :", sent;
print "Packet Received :", recv;
```

```
20  print "Packet delivery ratio :", pdr, "%";
21  }
22
23
```

Gambar 7.7 Implementasi Packet Delivery Ratio

```
1   BEGIN {
2   rt_pkts = 0;
3   }
4   {
5   if (($1 == "s" || $1 == "f") && ($4 == "RTR") &&
    ($7 == "DSR"))
6       rt_pkts++;
7   }
8   END {
9   printf ("Total packet :\t%d\n", rt_pkts);
10  }
```

Gambar 7.8 Implementasi Routing Overhead

```
1   BEGIN{
2   for ( i in pkt_id)
3        {
4                pkt_id[i] = 0;
5           }
6   for ( i in pkt_sent)
7        {
8                pkt_sent[i] = 0;
9           }
10  for ( i in pkt_recv )
11       {
12               pkt_recv[i] = 0;
13          }
14  delay = avg_delay = 0;
15       recv = 0;
16       recv_id = 0;
17  }
18  {
19
20  if ( $1 == "s" && $3 == "_1_" && $4 == "AGT" &&
    $7 == "cbr" )
21  {
```

```
22    pkt_sent[$6] = $2;
23  }
24  # count packet receive
25  if ( $1 == "r" && $3 == "_2_" && $4 == "AGT" &&
    $7 == "cbr" &&  recv_id != $6 )
26  {
27    recv++;
28    recv_id = $6;
29    pkt_recv[$6] = $2;
30  }
31  }
32  END{
33  for (i in pkt_recv)
34      {
35  delay += pkt_recv[i] - pkt_sent[i];
36      }
37
38  avg_delay = delay / recv;
39
40  print "Packet =", recv;
41  print " Delay =", delay, "second";
42  print "Average Packet Delay = ", avg_delay,
43  "second";
44  }
```

Gambar 7.9 Implementasi End-to-End Delay

Instalasi NS-2

Dokumentasi tentang cara instalasi NS-2 bersumber dari nsnam.com. Pada Tugas Akhir ini digunakan cara mengunduh file NS-2 yang telah disediakan oleh nsnam.com. Langkah-langkah lebih mendetail adalah sebagai berikut.

- Pertama kali update komponen terbaru dari Ubuntu dan pastikan Ubuntu yang diinstal ter-updateyang paling baru.

```
$ sudo apt-get update
$ sudo apt-get dist-upgrade
$ sudo apt-get update
```

Gambar 7.10 Perintah update seluruh komponen Ubuntu

- Kemudian lakukan instalasi modul-modul penting untuk pengerjaan tugas akhir.

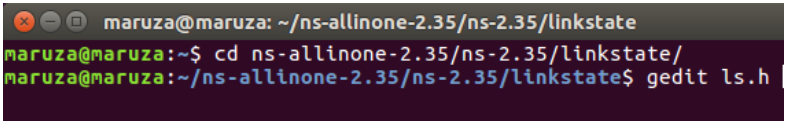
```
$sudo     apt-getinstallbuild-essential     autoconf
automake
$sudo apt-get install tcl8.5-dev tk8.5-dev
$ sudo apt-get install perl xgraph libxt-dev libx11-
dev libxmu-dev
$ sudo apt-get install gcc-4.4
```

Gambar 7.11 Perintah instalasi dependensi NS-2

- Setelah semua lengkap, file ns-2 yang telah diunduh dipindahkan menuju folder /home dan dilakukan proses ektraksi. Kemudian dilakukan proses pengubahan script pada ls.h.

```
$ tar -xvzf ns-allinone-2.35.tar.gz
$ cd /ns-allinone-2.35/ns-2.35/linkstate/
$ gedit ls.h
```

Gambar 7.12 Proses ekstrak dan pengubahan ls.h



Gambar 7.13 Screenshot proses ekstrak dan pengubahan ls.h

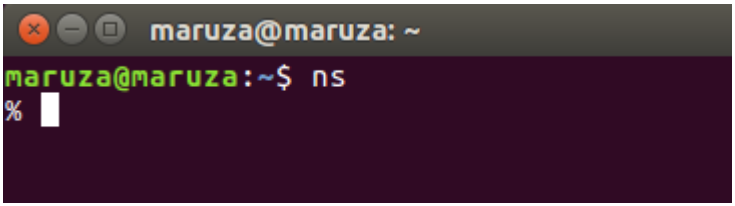
- Setelah semua file lengkap, ns-2 yang telah diunduh dipindahkan menuju folder /home dan dilakukan proses ektraksi. Kemudian dilakukan proses pengubahan Pada line ke-137, erase diubah menjadi this->erase karena jika tidak bisa jadi akan terjadi kegagalan pada saat proses instalasi NS-2.

- Terakhir dilakukan proses instalasi dari NS-2.

```
$ sudo ./install
```

Gambar 7.14 Perintah instalasi NS-2

- Untuk melakukan tes apakah NS-2 telah terinstall dengan baik maka dapat dilakukan dengan mengetikkan command ‘ns’ pada terminal dan apabila muncul tanda ‘%’ pada terminal berarti NS-2 telah terinstall dengan baik.



Gambar 7.15 Perintah pengecekan NS-2

(Halaman ini sengaja dikosongkan)

BIODATA PENULIS

Pattoe Marza, biasa dipanggil O'i, lahir di Tanah Grogot pada 30 Januari 1993. Penulis adalah anak pertama dari tiga bersaudara dan dibesarkan di Tanah Grogot, Kalimantan Timur. Penulis menempuh pendidikan formal di SDN 014 Tanah Grogot (1998-2004), SMPN 1 Tanah Grogot (2004-2007), SMAN 1 Tanah Grogot (2007-2010). Pada tahun 2010, penulis memulai pendidikan S1 Jurusan Teknik Informatika Fakultas Teknologi Informasi di Institut Teknologi Sepuluh Nopember Surabaya angkatan 2010 yang terdaftar dengan NRP 5110100185.

Penulis memiliki hobi membaca dan makan. Selama menempuh kuliah, penulis juga aktif dalam organisasi kemahasiswaan seperti Himpunan Mahasiswa Teknik Computer (HMTC). Penulis dapat dihubungi melalui alamat email marza.tcits@gmail.com.