**TUGAS AKHIR – KI141502**

**STUDI KINERJA DYNAMIC SOURCE ROUTING (DSR) PADA TOPOLOGI YANG SANGAT DINAMIS DI MOBILE AD-HOC NETWORKS (MANET)**

JORDY ADHITYO PRADONO

NRP 5111 100 101

Dosen Pembimbing

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

Ir. Muchammad Husni, M.Kom.

JURUSAN INFORMATIKA

Fakultas Teknologi Informasi Dan Komunikasi

Institut Teknologi Sepuluh Nopember

Surabaya 2018

**TUGAS AKHIR – KI141502**

**STUDI KINERJA DYNAMIC SOURCE ROUTING (DSR) PADA TOPOLOGI YANG SANGAT DINAMIS DI MOBILE AD-HOC NETWORKS (MANET)**

JORDY ADHITYO PRADONO

NRP 5111 100 101

Dosen Pembimbing

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

Ir. Muchammad Husni, M.Kom.

JURUSAN INFORMATIKA

Fakultas Teknologi Informasi Dan Komunikasi

Institut Teknologi Sepuluh Nopember

Surabaya 2018

*[Halaman ini sengaja dikosongkan]*

**FINAL PROJECT – KI141502**

**STUDY OF DYNAMIC SOURCE ROUTING (DSR) PERFORMANCE ON DYNAMIC TOPOLOGY IN MOBILE AD-HOC NETWORKS (MANET)**

JORDY ADHITYO PRADONO

NRP 5111 100 101

Dosen Pembimbing

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

Ir. Muchammad Husni, M.Kom.

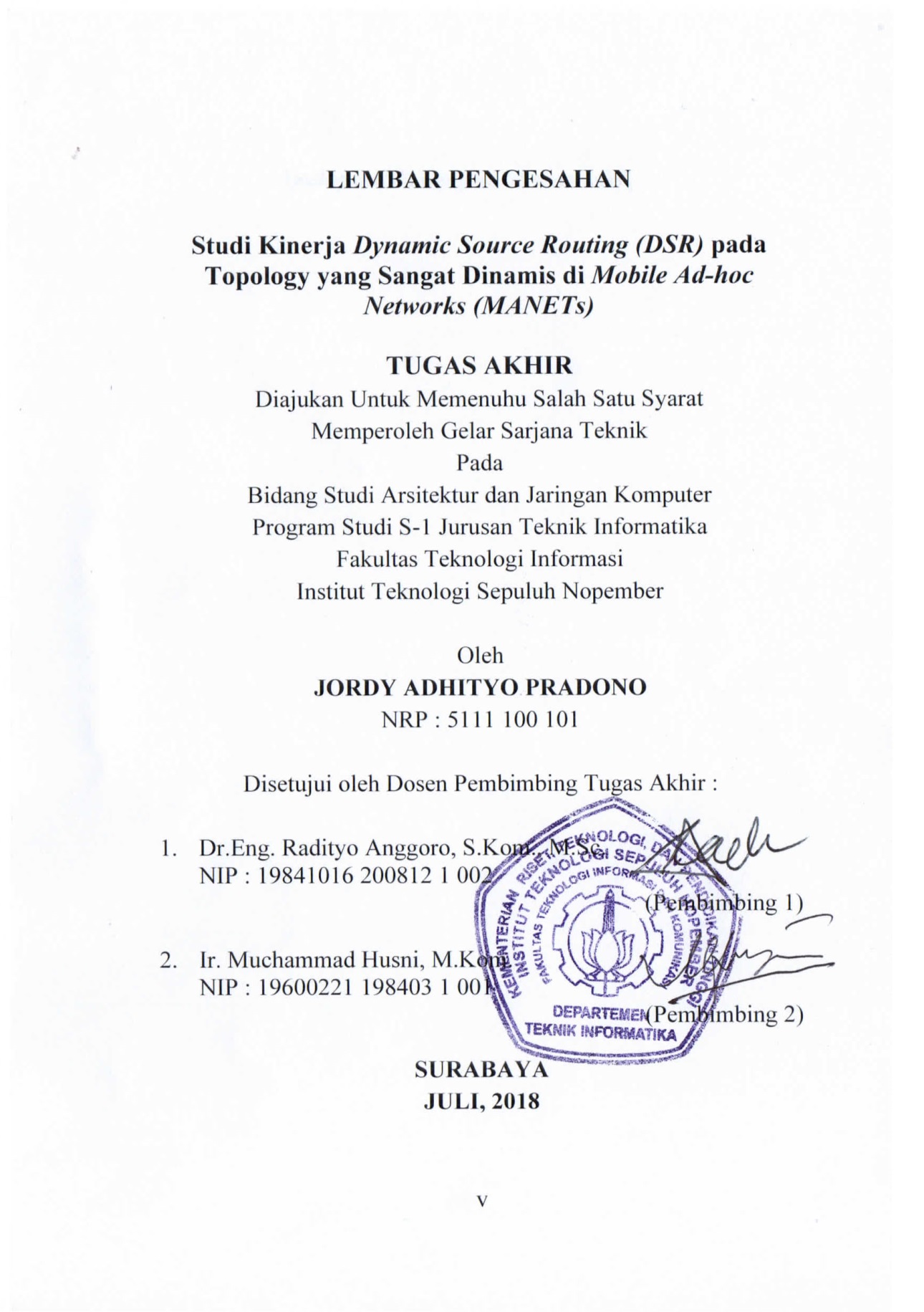
DEPARTMENT OF INFORMATICS

Faculty Of Information Technology And Comunication

Institute Of Technology Sepuluh Nopember

Surabaya 2018

*[Halaman ini sengaja dikosongkan]*

*****[Halaman ini sengaja dikosongkan]*

**Studi Kinerja *Dynamic Source Routing (DSR)* pada Topologi yang Sangat Dinamis di *Mobile Ad-hoc Networks (MANET)***

**Nama Mahasiswa : Jordy Adhityo Pradono  
NRP : 5111100101  
Jurusan : Informatika FTIK – ITS  
Dosen Pembimbing 1 : Dr.Eng. Radityo Anggoro, S.Kom., M.Sc.  
Dosen Pembimbing 2 : Ir. Muchammad Husni, M.Kom.**

# Abstrak

Mobile Ad-Hoc Network (MANET) adalah jenis jaringan ad hoc yang dapat mengubah lokasi dan mengkonfigurasi dirinya sendiri dengan cepat. Karena MANETS bersifat mobile, mereka menggunakan koneksi nirkabel untuk terhubung ke berbagai jaringan. Ini bisa menjadi koneksi Wi-Fi standar, atau media lain, seperti transmisi seluler atau satelit. Beberapa MANET dibatasi untuk area lokal perangkat nirkabel (seperti sekelompok computer/ laptop), sementara yang lain mungkin terhubung ke Internet.[1] Ada beberapa macam protokol routing pada MANET salah satunya adalah Dynamic Source Routing (DSR). DSR merupakan protokol routing reaktif dimana aspek utamanya adalah menyimpan keseluruhan jalur dari sumber ke tujuan di tabel routing alih-alih menyimpan hop berikutnya. Dalam Tugas Akhir ini protokol routing DSR akan dieksploitasi dengan menggunakan NS-2 berdasarkan packet delivery ratio, end-to-end delay, dan routing overhead dari satu node ke node lainnya.

**Kata kunci: MANET, DSR, NS-2.**

*[Halaman ini sengaja dikosongkan]***Study of Dynamic Source Routing (DSR) Performance on Dynamic Topology in Mobile Ad-hoc Networks (MANET)**

**Student’s Name : Jordy Adhityo Pradono  
Student ID : 5111100101  
Department : Informatika FTIK – ITS  
First advisor : Dr.Eng. Radityo Anggoro,S.Kom., M.Sc.  
Second advisor : Ir. Muchammad Husni, M.Kom.**

# Abstract

*Mobile Ad-Hoc Network (MANET) is a type of ad hoc network that can change location and configure itself quickly. Because MANETS are mobile, they use a wireless connection to connect to various networks. This can be a standard Wi-Fi connection, or other media, such as mobile or satellite transmissions. Some MANETs are limited to local area wireless devices (such as groups of computers / laptops), while others may be connected to the Internet There are several kinds of routing protocols in MANET one of them is Dynamic Source Routing (DSR). DSR is a reactive routing protocol where the main aspect is storing the entire path from source to destination in the routing table instead of storing the next hop. In this final project DSR routing protocol will be exploited using NS-2 based on packet delivery ratio, end-to-end delay, and routing overhead from one node to another node.*

***Keywords: MANET, DSR, NS-2.***

*[Halaman ini sengaja dikosongkan]*

# KATA PENGANTAR

Puji syukur kepada Allah SWT berkat Rahmat, Hidayah, dan Karunia-Nya kepada penulis sehingga penulis dapat menyelesaikan Tugas Akhir dengan judul “Studi Kinerja Dynamic Source Routing (DSR) pada Topology yang Sangat Dinamis di Mobile Ad-hoc Networks (MANETs)”.

Penulis menyadari dalam penyusunan proposal skripsi ini tidak akan selesai tanpa bantuan dari berbagai pihak. Karena itu pada kesempatan ini penulis ingin mengucapkan terima kasih kepada :

1. Allah SWT yang telah memberikan kesehatan, karunia, serta kemampuan kepada penulis untuk menyelesaikan Tugas Akhir ini.
2. Kedua orang tua penulis, Bapak R. Bambang Witjaksono dan Ibu Noer Tjahyawati yang senantiasa mendampingi serta memberikan doa kepada penulis.
3. Saudari kandung penulis yaitu Chany yang telah menemani dalam pengerjaan Tugas Akhir ini.
4. Bapak Dr. Eng. Radityo Anggoro, S.Kom., M.Sc. selaku dosen pembimbing penulis pertama yang selalu meluangkan waktu dan memberikan nasihat kepada penulis.
5. Bapak Ir. Muchammad Husni, M.Kom. selaku pembimbing kedua penulis.
6. Teman-teman mahasiswa Teknik Indormatika ITS angkatan 2011 dan adik angkatan juga yang membantu menyemangati penulis di Teknik Informatika ITS.
7. Kepada semua pihak yang belum sempat disebutkan satu per satu yang telah membantu penulis untuk menyelesaikan Tugas Akhir ini.

Penulis menyadari Tugas Akhir ini tidak luput dari berbagai kekurangan. Penulis mengharapkan saran dan kritik demi kesempurnaan dan perbaikannya sehingga akhirnya Tugas Akhir ini dapat memberikan manfaat bagi bidang pendidikan dan penerapan dilapangan serta bisa dikembangkan lagi lebih lanjut.

Surabaya, Juli 2018

Jordy Adhityo Pradono

# DAFTAR ISI

[LEMBAR PENGESAHAN v](#_Toc519534801)

[Abstrak vii](#_Toc519534802)

[Abstract ix](#_Toc519534803)

[KATA PENGANTAR xi](#_Toc519534804)

[DAFTAR ISI xiii](#_Toc519534805)

[DAFTAR GAMBAR xvi](#_Toc519534806)

[DAFTAR TABEL xix](#_Toc519534807)

[BAB I PENDAHULUAN xx](#_Toc519534808)

[1.1 Latar Belakang 1](#_Toc519534809)

[1.2 Rumusan Permasalahan 2](#_Toc519534810)

[1.3 Batasan Permasalahan 2](#_Toc519534811)

[1.4 Tujuan dan Manfaat 2](#_Toc519534812)

[1.5 Metodologi 3](#_Toc519534813)

[1. Penyusunan Proposal Tugas Akhir 3](#_Toc519534814)

[2. Studi Literatur 3](#_Toc519534815)

[3. Implementasi Protokol *Routing* 3](#_Toc519534816)

[4. Pengujian dan Evaluasi 3](#_Toc519534817)

[5. Penyusunan Buku Tugas Akhir 4](#_Toc519534818)

[1.6 Sistematika Penulisan 4](#_Toc519534819)

[BAB II TINJAUAN PUSTAKA 6](#_Toc519534820)

[2.1 *Mobile Ad-hoc Network (MANET)* 7](#_Toc519534821)

[2.2 *Dynamic Source Routing (DSR)* 8](#_Toc519534822)

[2.3 Network Simulator 2 (NS-2) 10](#_Toc519534823)

[2.3.1 Instalasi NS-2 10](#_Toc519534824)

[2.3.2 NS-2 *Trace* *File* 13](#_Toc519534825)

[2.4 *Generator File Node-Movement* dan *Random Traffic Pattern* 14](#_Toc519534826)

[2.4.1 *File Node-Movement (Mobility Generator)* 14](#_Toc519534827)

[2.4.2 *Random Traffic Pattern* 16](#_Toc519534828)

[2.5 AWK 17](#_Toc519534829)

[BAB III PERANCANGAN SISTEM 19](#_Toc519534830)

[3.1 Deskripsi Umum 19](#_Toc519534831)

[3.2 Perancangan Skenario 21](#_Toc519534832)

[3.2.1 Skenario *Mobility Generator* 21](#_Toc519534833)

[3.2.2 *Traffic Connection Pattern Generation* 22](#_Toc519534834)

[3.3 Perancangan Simulasi pada NS-2 22](#_Toc519534835)

[3.4 Perancangan Metriks Analisis 23](#_Toc519534836)

[3.4.1 *Packet Delivery Ratio* (PDR) 23](#_Toc519534837)

[3.4.2 *End-to-End Delay* (E2D) 24](#_Toc519534838)

[3.4.3 *Routing Overhead* (RO) 24](#_Toc519534839)

[BAB IV IMPLEMENTASI 25](#_Toc519534840)

[4.1 Lingkungan Pembangunan Perangkat Lunak 25](#_Toc519534841)

[4.1.1 Lingkungan Perangkat Lunak 25](#_Toc519534842)

[4.1.2 Lingkungan Perangkat Keras 25](#_Toc519534843)

[4.2 Implementasi Skenario 25](#_Toc519534844)

[4.2.1 Skenario *File Node-Movement (Mobility Generation)* 26](#_Toc519534845)

[4.2.2 *File Traffic-Connection Pattern Generation* 29](#_Toc519534846)

[4.3 Implementasi Simulasi pada NS-2 31](#_Toc519534847)

[4.4 Implementasi Metrik Analisis 37](#_Toc519534848)

[4.4.1 *Packet Delivery Ratio* (PDR) 37](#_Toc519534849)

[4.4.2 *End-to-End Delay* (E2D) 38](#_Toc519534850)

[4.4.3 Routing Overhead (RO) 40](#_Toc519534851)

[BAB V PENGUJIAN DAN EVALUASI 41](#_Toc519534852)

[5.1 Lingkungan Pengujian 41](#_Toc519534853)

[5.2 Kriteria Pengujian 41](#_Toc519534854)

[5.3 Analisis *Packet Delivery Ratio* (PDR) 42](#_Toc519534855)

[5.4 Analisis *End-to-End Delay* (E2D) 44](#_Toc519534856)

[5.5 Analisis *Routing Overhead* (RO) 46](#_Toc519534857)

[BAB VI PENUTUP 49](#_Toc519534858)

[6.1 Keimpulan 49](#_Toc519534859)

[6.2 Saran 50](#_Toc519534860)

[DAFTAR PUSTAKA 51](#_Toc519534861)

[LAMPIRAN 53](#_Toc519534862)

[BIODATA PENULIS 63](#_Toc519534863)

*[Halaman ini sengaja dikosongkan]*

# DAFTAR GAMBAR

[Gambar 2.1 Skema MANET](#_Gambar_2.1_Skema) 8

[Gambar 2.2 Prosedur penemuan rute pada DSR routing protokol](#_Gambar_2.2_Prosedur) 9

[Gambar 2.3 Export path di dalam .bashrc](#_Gambar_2.3_Export) 12

[Gambar 2.4 Tampilan NAM pada NS-2](#_Gambar_2.4_Tampilan) 12

[Gambar 2.5 Contoh event pada trace file](#_Gambar_2.5_Contoh) 13

[Gambar 3.1 Tahapan Rancangan Simulasi](#_Gambar_3.1_Tahapan) 20

[Gambar 4.1 Implementasi pada ‘setdest’](#_Gambar_4.1_Implementasi) 27

[Gambar 4.2 Posisi node dalam X, Y, dan Z](#_Gambar_4.2_Posisi) 27

[Gambar 4.3 Pergerakan Node](#_Gambar_4.3_Pergerakan) 28

[Gambar 4.4 Pembuatan GOD setiap Node](#_Gambar_4.4_Pembuatan) 28

[Gambar 4.5 Access Point](#_Gambar_4.5_Access) 29

[Gambar 4.6 Implementasi Koneksi cbrgen.txt](#_Gambar_4.6_Implementasi) 30

[Gambar 4.7 Output cbr\_test.txt](#_Gambar_4.7_Output) 30

[Gambar 4.8 Konfigurasi awal parameter NS-2](#_Gambar_4.8_Konfigurasi) 32

[Gambar 4.9 Konfigurasi Transmission Range pada NS-2](#_Gambar_4.9_Konfigurasi) 32

[Gambar 4.10 Konfigurasi Trace File, NAM dan Pergerakan Node pada NS-2](#_Gambar_4.10_Konfigurasi) 33

[Gambar 4.11 Konfigurasi pengiriman paket data NS-2](#_Gambar_4.11_Konfigurasi) 35

[Gambar 4.12 Perintah Eksekusi TwoRayGround](#_Gambar_4.12_Perintah) 36

[Gambar 4.13 Visualisasi hasil simulasi pada NAM](#_Gambar_4.13_Visualisasi) 36

[Gambar 4.14 Pseudecode PDR](#_Gambar_4.14_Pseudecode) 38

[Gambar 4.15 Pseudeucode E2D](#_Gambar_4.15_Pseudeucode) 39

[Gambar 4.16 Pseudeucode RO](#_Gambar_4.16_Pseudeucode) 40

[Gambar 5.1 Grafik PDR Skenario Mobility Generator](#_Gambar_5.1_Grafik) 43

[Gambar 5.2 Grafik E2D Skenario Mobility Generator](#_Gambar_5.2_Grafik) 45

[Gambar 5.3 Grafik RO Skenario Mobility Generator](#_Gambar_5.3_Grafik) 47

[Gambar 7.1 Posisi Node dari potongan skenario](#_Gambar_7.1_Posisi) 53

[Gambar 7.2 Pembuatan GOD setiap node dari potongan skenario](#_Gambar_7.2_Pembuatan) 54

[Gambar 7.3 Pergerakan setiap node dari potongan skenario](#_Gambar_7.3_Pergerakan) 55

[Gambar 7.4 Informasi pada GOD dari potongan skenario](#_Gambar_7.4_Informasi) 56

[Gambar 7.5 Koneksi yang digunakan pada cbr\_test.txt](#_Gambar_7.5_Koneksi) 57

[Gambar 7.6 File .tcl dari ptotokol routing DSR](#_Gambar_7.6_File) 59

[Gambar 7.7 Implementasi Packet Delivery Ratio](#_Gambar_7.7_Implementasi) 60

Gambar 7.8 Implementasi End-to-End Delay 60

[Gambar 7.9 implementasi Routing Overhead](#_Gambar_7.9_implementasi) 61

# DAFTAR TABEL

[Tabel 3.1 Parameter Skenario Node Movement](#_Tabel_3.1_Parameter) 21

[Tabel 3.2 Parameter Traffic Connection Pattern](#_Tabel_3.2_Parameter) 22

[Tabel 3.3 Parameter Simulasi pada NS-2](#_Tabel_3.3_Parameter) 23

[Tabel 5.1 Spesifikasi Laptop yang Digunakan](#_Tabel_5.1_Spesifikasi) 41

[Tabel 5.2 Kriteria Pengujian](#_Tabel_5.2_Kriteria) 42

[Tabel 5.3 PDR Skenario Mobility Generator](#_Tabel_5.3_PDR) 42

[Tabel 5.4 E2D Skenario Mobility Generator](#_Tabel_5.4_E2D) 44

[Tabel 5.5 RO Skenario Mobility Generator](#_Tabel_5.5_RO) 46

*[Halaman ini sengaja dikosongkan]*

# BAB I PENDAHULUAN

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

## 1.1 Latar Belakang

MANET dapat dikategorikan sebagai sebuah jaringan yang didalamnya terdapat kumpulan dari *mobile nodes* yang saling berhubungan menggunakan media komunikasi nirkabel dengan cara berkomunikasi dinamik dan sesuai kebutuhan masing-masing *mobile nodes*. MANET sangat cocok diaplikasian pada daerah yang sangat kekurangan akan infrastruktur telekomunikasi seperti solusi telekomunikasi disaat terjadinya bencana alam, upaya rekonstruksi sehabis bencana, operasi militer hingga pada kondisi dimana kita terdesak untuk membutuhkan komunikasi (seperti proses evakuasi tim sar di hutan-hutan). Salah satu perbedaan yang dimiliki teknologi MANET jika dibandingkan dengan teknologi komunikasi kabel lainnya adalah tidak diperlukannya sebuah router yang mengatur seluruh proses routing pada topoli jaringan MANET ini dikarenakan setiap device berfungsi sebagai router untuk menentukan ke arah mana tujuan yang akan mereka pilih.[2]

Karakteristik dari MANET sendiri adalah MANET memiliki topologi yang sangat dinamis. Dimana *node* pada MANET dapat bergerak secara bebas dan berpindah-pindah kemana saja. Topologi jaringan yang bisanya *multihop* dapat berubah secara acak dan tidak terpola. Dikatakan menjadi sangat dinamis apabila kondisi node tersebut berada diatas kecepatan normal. Karena MANET menggunakan komunikasi nirkabel, kecepatan normal yang biasa digunakan adalah 3 m/s. Maka bila kecepatan yang digunakan diatas kecepatan tersebut dapat dikatakan topologi yang digunakan sangat dinamis. Oleh karena itu, diterapkanlah protokol routing multipath, yang dapat memberikan lebih dari satu rute ke node tujuan. Sehingga node sumber dan node perantara dapat menggunakan rute ini sebagai rute utama maupun sebagai rute cadangan. Salah satunya adalah protokol *routing* DSR. Pada Tugas Akhir ini, penulis menggunakan NS-2 untuk menganalisis kinerja protokol *routing* DSR yang di implementasi pada lingkungan MANET menggunakan Network Simulator 2 (NS-2).

Hasil yang diharapkan dari Tugas Akhir ini adalah suatu hasil analisa studi kinerja pada protokol routing DSR di lingkungan MANET dengan menggunakan aplikasi. 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 MANET?

2. Bagaimana hasil analisa studi kinerja protokol routing DSR di lingkungan MANET?

## 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).

## 1.4 Tujuan dan Manfaat

Tujuan dari pembuatan Tugas Akhir ini adalah untuk memberikan hasil analisa studi kerja protokol routing DSR di lingkungan MANET dengan menggunakan aplikasi Network Simulator 2 (NS-2).

Tugas akhir ini diharapkan dapat menghasilkan hasil analisa kerja protokol routing DSR yang efisien dengan parameter Packet Delivery Ratio, Routing Overhead, dan Delay.

## 1.5 Metodologi

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

### 1. Penyusunan Proposal Tugas Akhir

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

### 2. Studi Literatur

Pada tahap ini dilakukan studi literatur mengenai *tools* dan metode yang digunakan. Literatur yang dipelajari dan digunakan meliputi buku referensi, artikel, jurnal dan dokumentasi dari internet.

### 3. Implementasi Protokol *Routing*

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

### 4. Pengujian dan Evaluasi

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

### 5. 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. Tujuan Pustaka  
3. Perancangan Sistem  
4. Implementasi  
5. Pengujian dan Implementasi  
6. Penutup

## 1.6 Sistematika Penulisan

Buku Tugas Akhir ini terdiri dari 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. Tinjuan Pustaka

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

* Bab III. Perancangan

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

* Bab IV. Implementasi

Bab ini membahas implementasi dari rancangan sistem atau desain yang akan dilakukan pada tahap perancangan. Penjelasan berupa implementasi scenario mobilitas *node-node* pada jaringan *wireless* yang mempunyai *router* tetap yang dibuat menggunakan *file node-movement* dan *traffic-pattern* yang ada pada *network simulator*, konfigurasi sistem dan skrip analisi yang digunakan untuk menguji performa protokol routing.

* Bab V. Pengujian dan Evaluasi

Bab ini menjelaskan tahap pengujian sistem dan performa dalam scenario mobilitas *ad hoc* yang dibuat oleh distribusi *mobility* dalam *network simulator*.

* 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]*

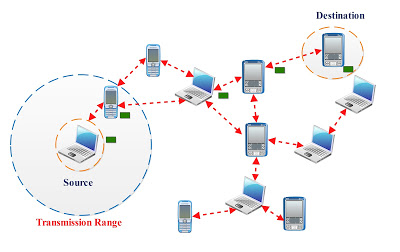
# BAB II TINJAUAN PUSTAKA

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

## 2.1 *Mobile Ad-hoc Network (MANET)*

*Mobile ad-hoc Network* (MANET), juga dikenal sebagai *Wireless ad-hod network*, adalah kumpulan dari beberapa wireless node yang dapat di set-up secara dinamis dimana saja dan kapan saja, tanpa menggunakan infrastruktur jaringan yang ada. MANET juga merupakan jaringan sementara yang dibentuk oleh 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.

Di zaman smartphone sekarang ini, MANET bisa jadi hal yang sangat membantu proses pertukaran data antar node. Misal, jika kita ingin melakukan komunikasi data dengan 10 orang yang berada di daerah terbuka, jauh dari jaringan internet, maka MANET menjadi solusi terbaik. Sebab kita tidak harus mengandalkan jaringan internet sebagai pusat penghubungnya. Setiap mobile node akan dapat berkoneksi secara langsung. [3]

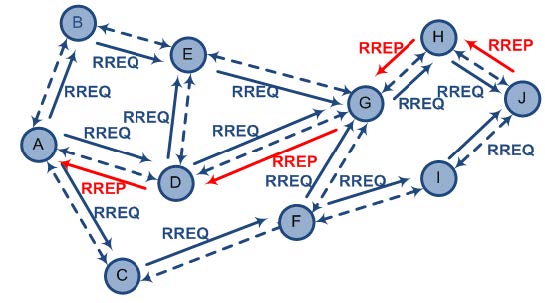


##### Gambar 2.1 Skema MANET

## 2.2 *Dynamic Source Routing (DSR)*

DSR adalah protokol routing reaktif dimana aspek utamanya adalah menyimpan keseluruhan jalur dari sumber ke tujuan di tabel routing alih-alih menyimpan hop berikutnya. Oleh karena itu, header paket harus menyertakan semua node dimana paket harus melakukan perjalanan untuk dikirim ke tempat tujuan. RREQ dan RREP digunakan untuk melakukan penemuan rute dan mengirimkan pesan balasan kembali ke sumbernya. Dalam protokol ini, metode rebardcast pesan RREQ digunakan jika node yang menerima pesan RREQ tidak memiliki informasi tujuan di tabel routing-nya. Namun, dalam protokol routing DSR, mekanisme rute cache digunakan jika terjadi kerusakan tautan. Telah dicatat bahwa rute lain ke simpul tujuan dipertahankan dalam rute cache karena mendengar pesan RREQ oleh node antara melalui berbagai rute. Mekanisme rute cache menghasilkan peningkatan transmisi data. Setelah menerima pesan RERR oleh node sumber, prosedur penemuan rute baru akan dimulai. Pesan RERR akan berasal dan dikirim ke sumber oleh simpul pertama yang lebih dekat ke sumber daripada yang lain. Setelah itu, sumber yang menerapkan strategi piggyback berdasarkan pesan RERR yang diterima dan pesan RREQ yang baru akan disiarkan ke semua node yang digunakan untuk menyebarkan tautan yang gagal. Gambar 1 mengilustrasikan transmisi pasangan <RREQ, RREP> saat melakukan prosedur penemuan rute sampai menerima pesan balasan.

Garis putus-putus mewakili rute yang tersimpan dalam memori rute cache untuk pemanfaatan lebih lanjut saat kerusakan tautan terjadi. Secara kiasan, ukuran paket dalam protokol routing DSR meningkat karena menambahkan spesifikasi simpul yang ada ke dalam header paket. Hal ini dapat dianggap sebagai kelemahan yang mungkin terjadi bila jumlah node meningkat. Masalah lain yang harus diperhatikan adalah tidak mengetahui daftar tetangga atau status link mereka. Karena tidak ada paket pemutakhiran berkala yang dipertukarkan antar node, yang menerapkan mekanisme rute cache dapat menyebabkan kegagalan karena penyebaran tautan tidak valid atau kedaluwarsa.[4]



##### Gambar 2.2 Prosedur penemuan rute pada DSR routing protokol[4]

## 2.3 Network Simulator 2 (NS-2)

Network Simulator 2 atau biasa disingkat NS-2 merupakan sebuah *network simulator* yang dibuat dengan tujuan riset dan pendidikan. Awalnya, NS dibangun sebagai varian dari REAL Network Simulator pada tahun 1989 di UCB (University of California Berkeley).

Dari awal tim ini membangun sebuah perangkat lunak simulasi jaringan internet untuk kepentingan riset interaksi antar protokol dalam konteks pengembangan protokol internet saat ini dan masa yang akan datang. NS-2 Merupakan salah satu perangkat lunak yang dapat menampilkan secara simulasi proses komunikasi dan bagaimana proses komunikasi tersebut berlangsung. NS-2 memiliki beberapa fitur kelebihan yang dapat dimanfaatkan dalam pemodelan dan pengujian MANET. NS-2 memiliki *tools* validasi yang berfungsi untuk menguji validalitas pemodelan yang ada pada NS-2. NS-2 bersifat *open source*  dibawah GPL (*GNU Public Licence).* Sehingga dapat diunduh melalui situs NS-2 https://www.isi.edu/nsnam/ns/ns-build.html.

MANET disimulasikan menggunakan NS-2 melalui pembuatan topologi jaringan dan skenario simulasi penerapan *routing protocol*. Pemodelan media, protokol, dan komponen jaringan lengkap dengan perilaku *traffic*-nya sudah tersedia di *library* NS-2.[5]

Pada Tugas Akhir ini NS-2 yang digunakan adalah versi 2.35 sebagai aplikasi simulasi scenario MANET dengan *routing protovol* DSR.

### 2.3.1 Instalasi NS-2

Tata cara instalasi NS-2 bersumber dari situs https://www.isi.edu/nsnam/ns/ns-build.html. Pada situs ini banyak terdapat pilihan cara instalasi. Pada Tugas Akhir ini digunakan cara instalasi yang dilakukan untuk Ubuntu 16.04. Langkah-langkah instalasi adalah sebagai berikut.

* Tahap pertama ialah instalasi modul build-essential, autoconf, automake, libxmu-dev yang menjadi dependensi untuk instalasi NS-2

$ sudo apt-get update

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

* Unduh berkas NS, dimana dalam Tugas Akhir ini digunakan berkas ‘ns-allinone-2.35.tar.gz’, *unzip atau untar* berkas tersebut.

$ tar –xvzf ns-allinone-2.35.tar.gz

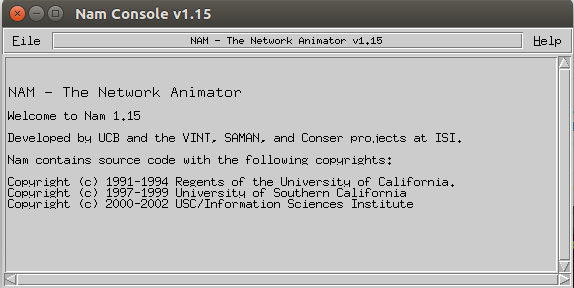
$ cd ns-allinone-2.35/

* Setelah *extract*, dilakukan penyuntingan pada berkas didalam ‘/ns-allinone-2.35/ns-2.35/linkstate/ls.h’. Pindah ke baris 137 dan replace “erase” dengan “this->erase” kemudian simpan.
* Kemudian dilakukan instalasi NS-2 dengan menggunakan ‘./install’ didalam dierktori ‘ns-allinone-2.35’
* Setelah instalasi selesai, *path* diatur terlebih dahulu supaya NS dan NAM (*Network Animator)* dapat bekerja dengan baik pada jendela terminal. Pada Ubuntu 16.04, pengaturan *path* ini dilakukan pada *file* .bashrc.
* *Export path* dilakukan di dalam *file* .bashrc seperti yang terlihat pada Gambar 2.3 berikut.



##### Gambar 2.3 *Export path* di dalam .bashrc

* Setelah penambahan *path,* selanjutnya melakukan perintah seperti skrip dibawah agar ns dan nam dapat dijalankan.
* Tahap akhir, percobaan dilakukan untuk menguji apakah *ns* dan *nam* sudah berjalan atau tidak. Dengan mengetik ‘ns’ pada jendela terminal, apabila berhasil makan akan terliha tanda ‘%’. Sedangkan untuk *nam,* apabila berhasil akan terlihat tampilan window NAM seperti pada gambar 2.4.



##### Gambar 2.4 Tampilan NAM pada NS-2

### 2.3.2 NS-2 *Trace* *File*

*Trace File* NS-2 merupakan keluaran (*output*) hasil uji coba dari NS-2 yang berekstensi .tr. berkas ini mengandung sejumlah *log* mengenai pengiriman dan penerimaan paket yang terjadi saat simulasi dilakukan. Jenis paket yang tercatat pada *Trace file* ini sesuai dengan protokol *routing* yang digunakan. Format *trace file* pada jaringan nirkabel antara lain.

* Kolom 1 berisi *event* : r “*received*”, s “*sent*”, f “*forwarded*”, dan d “*dropped*”.
* Kolom 2 adalah waktu (*time*)
* Kolom 3 adalah nomor node
* Kolom 4 adalah *trace* *level* antara lain MAC yang berhubungan dengan *MAC layer*. Untuk AGT menunjuk ke paket *transport* *layer*. Untuk RTR menunjuk ke paket *route*
* Kolom 5 adalah nomor urut paket
* Kolom selanjutnya adalah tipe paket (UDP, TCP, CBR)
* [13a 0 d 800] menjunjukkan informasi MAC layer
* [1:0 0:0 32 0] menunjukan IP *source* dan alamat tujuan kemudiian TTL (*time to live*) dari paket
* [852] menunjukkan nomor urut dan pemberitahuan nomor
* 0 0 adalah format mekanisme *routing* tipe paket

s 42.365000000 \_1\_ AGT --- 852 cbr 210 [0 0 0 0] ------- [1:0 0:0 32 0] [852] 0 0

r 42.375018532 \_0\_ AGT --- 851 cbr 230 [13a 0 d 800] ------- [1:0 0:0 26 0] [851] 5 0

f 42.353273089 \_13\_ RTR --- 851 cbr 230 [13a d 9 800] ------- [1:0 0:0 26 0] [851] 4 0

d 97.463750000 \_1\_ RTR IFQ 2887 cbr 230 [0 0 0 0] ------- [1:0 0:0 30 0] [2887] 0 0

##### Gambar 2.5 Contoh *event* pada *trace* file

Skrip diatas merupakan salah satu contoh *trace file* yang dihasilkan program NS-2 didalam berkas .tcl.

## 2.4 *Generator File Node-Movement* dan *Random Traffic Pattern*

### 2.4.1 *File Node-Movement (Mobility Generator)*

*Node-movement Generator* tersedia dalam direktori ~ns/indep-utils/cmu-scen-gen/setdest dan *Makefile*. *Tools* ini digunakan untuk menghasilkan gerakan acak dari *node* dalam jaringan nirkabel. Pergerakan *node* dihasilkan dengan kecepatan gerak yang spesifik menuju lokasi acak atau lokasi spesifik yang berada pada kawasan yg telah ditentukan. Untuk menjalankan setdest.cc yang telah terevisi, lakukan hal berikut :

1. Pergi ke direktori ns dan jalankan “configure”. Hal ini akan menciptakan *makefile* untuk setdest
2. Pergi ke indep-utils/cmu-scen-gen/setdest. Jalankan “make”.
3. Jalankan setdest dengan argument sebagai berikut :

./setdest [-n num\_of\_nodes] [-p pausetime] [-M maxspeed] [-t simtime] [-x maxx] [-y maxy] > [outdir/movement-file]

* -n num\_of\_nodes : jumlah *node* pada scenario
* -p pausetime : durasi ketika sebuah *node* tetap diam setelah tiba pada lokasi pergerakan. Jika nilai ini diatur dengan nilai 0, maka *node* tidak akan berhenti ketika tiba pada lokasi pergerakan.
* -M maxspeed : kecepatan maksimum sebuah *node*.
* -t simtime : waktu simulasi
* -x maxx : panjang maksimum area simulasi
* -y maxy : lebar maksimum area simulasi

Katakanlah kita ingin memuat sebuah scenario *node-movement* yang terdiri dari 20 *node* yang bergerak dengan kecepatan maksimum 10.0m/s dengan jeda rata-rata antara gerakan 2s. Simulasi ini berdurasi 200s dan akan menggunakan topologi dengan ukuran 500 x 500. Maka baris perintahnya akan tampak seperti berikut :

./setdest -n 20 -p 2.0 -M 10.0 -t 200 -x 500 -y 500 > scen-20-test

Secara *default* file keluaran akan ditulis pada “stdout”. Disini keluaran disimpan pada *file* scen-20-test. *File* dimulai dengan posisi awal *node* dan berlanjut menetapkan pergerakan *node* seperti pada baris berikut :

$ns\_ at 2.000000000000 "$node\_(0) setdest 90.441179033457 44.896095544010

1.373556960010"

Baris perintah diatas mendefinisikan bahwa pada *node*(0) detik 2.0 mulai bergerak ke arah tujuan (90.44, 44.89) dengan kecepaan 1.37 m/s. Arahan untuk GOD (*Generat Operations Director)* yang ada juga di *file* *node-movement*. Objek “GOD” digunakan untuk menyimpan informasi global tentang keadaan lingkungan, jaringan dan sekitarnya. Isi dari *file* “GOD” sendiri tidak boleh terlihat oleh bagian-bagian dari simulasi.

Dalam simulasi ini, objek “GOD” hanya digunakan untuk menyimpan *array* dari jumlah *hop* terpendek yang diperlukan untuk mencapai dari satu *node* ke *node* lainnya. Objek “GOD” tidak menghitunh jumlah *hop* yang diperlukan selama simulasi berjalan, karena hal tersebut sangat memakan waktu. Informasi yang dimuat ke dalam “GOD” dari pola pergerakan *file* terdapat pada perintah berikut :

$ns\_ at 899.642 "$god\_ set-dist 23 46 2"

Perintah ini digunakan untuk memuat objek “GOD” dengan informasi bahwa jarak terpendek antar *node* 23 dan *node* 46 berubah menjadi 2 *hop* pada detik 899.642.[6]

### 2.4.2 *Random Traffic Pattern*

*Random traffic connection* dari TCP dan CBR dapat dikonfigurasi antara mobilitas *node*  menggunakan skrip *traffic-scenario generator*. Skrip *traffic generator* tersedia dalam ~ns/indep-utils/cmu-scen-gen dan disebut dengan “cbrgen.tcl”. Untuk membuat sebuah file *traffic connection*, kita haru mendefinisikan tipe *traffic connection* (CBR atau TCP), jumlah *node* dan jumlah koneksi maksimal antar *node*, *seed* acak dan koneksi CBR, nilai *invers* yang digunakan untuk menghitung waktu interval antara paket CBR. Maka perintah yang digukana akan terlihat seperti berikut:

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

Format yang digunakan :

* -type cbr|tcp : jenis *traffic* yang digunakan TCP

atau CBR

* -nn nodes : jumlah total *nodes*
* -seed seed : *random seed*
* -mc connections : jumlah koneksi
* -rate rate : jumlah paket per detik

Waktu awal untuk koneksi TCP/CBR dihasilkan secara acak dengan nilai maksimum 180.0s. Sebagai contoh, kita akan mencoba membuat file koneksi CBR antara 10 *node* dengan maksimum 8 koneksi, dengan nilai *seed* 1.0 dan *rate* 4.0. Maka baris perintahnya akan tampak seperti berikut :

ns cbrgen.tcl -type cbr -nn 10 -seed 1.0 -mc 8 -rate 4.0 > cbr-10-test

Dari *file* cbr-10-test (dimana output generatornya dialihkan) yang telah dibuat, salah satu koneksi CBR-nya akan terlihat seperti ini:

#

# 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"

Agen yang digunakan adalah UDP. Dengan demikian koneksi UDP adalah *setup* antara *node* 2 dan 3. Jumlah sumber UDP dan jumlah *setup* koneksi diindikasikan sebagai 10 koneksi di akhir file cbr-10-test.[6]

## 2.5 AWK

Awk adalah sebuah pemrograman seperti pada shell atau C yang memiliki karakteristik yaitu sebagai tool yang cocok 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. (Portable Operating System Interface for UNIX). Jika kecepatan merupakan hal yang penting, awk adalah bahasa yang sangat sesuai. Awk dan shell, keduanya adalah biasa dipakai untuk aplikasi yang berbeda. Awk sangat baik untuk manipulasi file teks, sedangkan shell sangat baik untuk pelaksana perintah UNIX. Secara umum bahasa pemrograman awk dapat digunakan untuk :

* Mengelola database sederhana.
* Membuat report.
* Memvalidasi data.
* Menghasilkan index and 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.
* Menampilkankomunikasi 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.[7]

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

# BAB III PERANCANGAN SISTEM

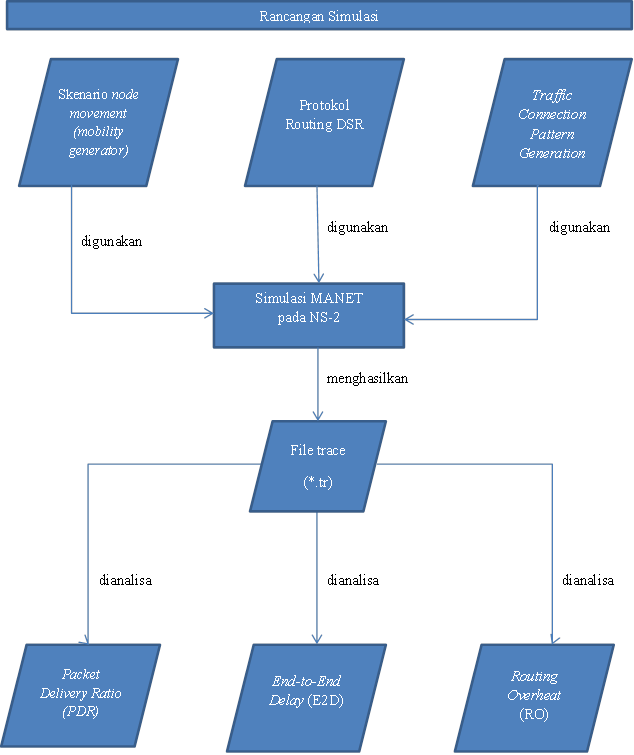
Pada bab ini dijelaskan mengenai dasar perancangan dari perangkat lunak yang akan dibangun dalam Tugas Akhir ini. Secara khusus, akan dibahas dalam deskripsi umum sistem, perancangan scenario, alur, serta gambaran implementasi sistem yang diterapkan pada *Network Simulator 2*(NS-2).

## 3.1 Deskripsi Umum

Pada tugas akhir ini akan dilakukan analisis tentang performa protokol *routing* DSR pada MANET. Dalam pembuatan skenario MANET menggunakan *Mobility Generator* yang bersifat *Random Way Point* dan telah ada pada *Network Simulator2* (NS-2) yaitu dengan cara men­-*generate­ ­mobility generator* dan membuat koneksi antar *node* menggunakan *file traffic-connection pattern*. Rancangan simulasi yang dibuat dapat dilihat pada Gambar 3.1.

Dalam penelitian ini, hanya terdapat 1 jenis model topologi yang digunakan untuk pengukuran performa dari *mobility generator* yaitu topologi *dinamis* (bergerak-gerak). Kemudian untuk simulasi skenario yang dihasilkan oleh *mobility generator* akan dijalankan pada NS-2 menggunakan protokol *routing* DSR dengan model transmisi yang berbeda pada sistem operasi Linux.

Dalam penelitian ini terdapat beberapa skenario yang digunakan sebagai perbandingannya. Yaitu skenario “a” dimana node yang digunakan adalah 50, skenario “b” dimana node yang digunakan adalah 100, dan skenario “c” dimana node yang digunakan adalah 150. Pada tiap skenario, kecepatan maksimum pergerakan dari satu *node* ke *node* lainnya dibuat bervariasi yaitu 5,10,15, dan 20 m/s. Hasil uji coba dari tiap skenario akan menghasilkan sebuah *trace file* yang nantinya akan dilakukan analisi perhitungan metric *Packet Delivery Ratio* (PDR), *End-to-End Delay* (E2D), dan *Routing Overhead* (RO).



##### Gambar 3.1 Tahapan Rancangan Simulasi

dianalisa

digunakan

*Traffic Connection Pattern Generation*

## 3.2 Perancangan Skenario

Perancangan skenario uji coba mobilitas MANET diawali dengan melakukan pembuatan skenario *mobility generation* yang bersifat *random way point* kemudian membuat koneksi dengan menggunakan *file traffic connection* yang sudah ada pada NS-2. Pada Tugas Akhir ini pembuatan scenario untuk melihat pergerakan *node* dibedakan berdasarkan 3 jumlah node yang berbeda yaitu 50, 100, dan 150 dan berdasarkan 4 kecepatan maksimum yaitu 5,10,15, dan 20 m/s. Sedangkan koneksinya hanya menggunakan 2 *node* untuk menentukan *node* pengirim dan *node* penerima paket. Penjelasan untuk perancanagan skenario pada *mobility generator* dan pembuatan koneksi antar *node-nya* sebagai berikut :

### Skenario *Mobility Generator*

Skenario *mobility generator*  dibuat dengan men-*generate file node movement* yang telah ada pada NS-2 atau *tools* yang biasa disebut ‘setdest’ yang nantinya akan menghasilkan *output* dalam bentuk .txt 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, 100, 150 |
| 2 | Waktu Simulasi | 100 detik |
| 3 | Area | 500 m x 500 m |
| 4 | Kecepatan Maksimum | 5 m/s, 10 m/s, 15 m/s, 20 m/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* |
|  |  |  |

### *Traffic Connection Pattern Generation*

*Traffic Connection* dibuat dengan menjalankan program ‘cbgren.tcl’ yang telah ada pada NS-2 yang nantinya akan menghasilkan *output* dalam bentuk .txt dan digunakan sebagai koneksi untuk menghubungkan antar *node* yang ada pada scenario selama simulasi pada NS-2.

##### Tabel 3.2 Parameter Traffic Connection Pattern

|  |  |  |
| --- | --- | --- |
| No | Parameter | Spesifikasi |
| 1 | -type cbr|tcp | CBR |
| 2 | -nn *nodes* | 10 |
| 3 | -s seed | 1.0 |
| 4 | -mc connections | 8 |
| 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 parameter simulasi perancangan system MANET yang dapat digunakan.

##### Tabel 3.3 Parameter Simulasi pada NS-2

|  |  |  |
| --- | --- | --- |
| **No.** | **Parameter** | **Spesifikasi** |
| 1 | Network Simulator | NS-2.35 |
| 2 | *Routing Protocol* | DSR |
| 3 | Waktu simulasi | 100 detik |
| 4 | Waktu Pengiriman Paket Data | 0 – 100 detik |
| 5 | Area simulasi | 500 m x 500 m |
| 6 | Banyak *node* | 50, 100, 150 |
| 7 | Radius transmisi | 100 m |
| 8 | Tipe Koneksi | UDP |
| 9 | Tipe data | CBR |
| 10 | Source / Destination | Statik (*Node 1 / Node 2)* |
| 11 | Kecepatan generasi paket | 1 paket/detik |
| 12 | Ukuran paket data | 512 bytes |
| 13 | Tiper Antena | OmniAntenna |
| 14 | Tipe Peta | MANET |
| 15 | Tipe kanal | Wireless channel |
|  |  |  |

## 3.4 Perancangan Metriks Analisis

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

### 3.4.1 *Packet Delivery Ratio* (PDR)

*Packet delivery ratio* dihitung dari perbandingan antara paket yang dikirim dengan paket yang diterima. *Packet delivery ratio* dihitung dengan persamaan (1), dimana *received* adalah banyaknya paket data yang diterima dan *sent* adalah banyaknya paket data yang dikirimkan.

Persamaan (1). Formula menghitung PDR

### 3.4.2 *End-to-End Delay* (E2D)

*End-to-End Delay*  dihitung dari rata-rata *delay* antara waktu paket diterima dan waktu paket dikirim. *Average delivery delay* dihitung dengan menggunakan persamaan (2), di treceived[i]adalah waktu penerimaan paket dengan urutan / id ke-*i­*, tsend[i] adalah waktu pengiriman paket dengan urutan / id ke-*I*, dan *sent* adalah banyaknya paket data yang dikirimkan.

Persamaan (2). Formula menghitung E2D

### 3.4.3 *Routing Overhead* (RO)

*Routing Overhead* adalah jumlah paket kontrol *routing* yang ditransmisikan per data paket yang terkirim ke tujuan selama simulasi terjadi. RO dihitung berdasarkan paket *routing* yang ditransmisikan baik itu *Router Request* (RREQ), *Router Replay* (RREP) *,* dan *Router Error* (RERR).

# BAB IV IMPLEMENTASI

Bab ini membahas tentang implementasi dari perancangan perangkat lunak. Implementasi yang dijelaskan meliputi lingkungan pembangunan perangkat lunak, implementasi skenario, implementasi simulasi pada NS-2, dan implementasi matrik analis.

## 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 system 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) Celeron(R) CPU N2840 @ 2.16GHz 2.16 GHz;
* Media penyimpanan sebesar 500GB;
* RAM sebesar 4GB DDR3

## 4.2 Implementasi Skenario

Implementasi skenario mobilitas MANET dipelajari dalam kondisi yang berbeda pada beban *traffic-*nya dan mobilitas/pergerakan *node-*nya. Dua 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 *File Node-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 *file node movement* dengan menggunakan algoritma *Random Way Point*. Format *command line* yang digunakan untuk menghasilkan gerakan acak pada *node* adalah sebagai berikut :

./setdest [-n num\_of\_nodes] [-p pausetime] [-M maxspeed] [-t simtime] [-x maxx] [-y maxy] > [outdir/movement-file]

Ketentuan-ketentuan yang diujicobakan pada skenarionya berturut-turut adalah versi ‘setdest’ simulator yaitu 1, jumlah *node* dalam skenario yaitu 50, 2 jumlah *node* dalam skenario yaitu 100, dan 3 jumlah *node* dalam skenario yaitu 150, waktu jeda yaitu 10 detik, kecepatan maksimalnya yaitu skenario A sebesar 2 m/s, skenario B sebesar 4 m/s, skenario C sebesar 6 m/s, skenario D sebesar 8 m/s, skenario E sebesar 10 m/s, skenario F sebesar 12 m/s, skenario G sebesar 14 m/s, skenario H sebesar 16 m/s, skenario I sebesar 18 m/s, dan skenario J sebesar 20 m/s, waktu simulasi yaitu 100 detik, panjang dan lebar maksimal area simulasi yaitu 500 meter. Kemudian *file* mobilitas yang dihasilkan disimpan dalam direktori “ ~ ns /indep-utils/CMU-scen-gen/setdest /”. Pada gambar 4.1 dapat dilihat implementasi *command line* pada ‘setdest’ dengan berbagai kecepatan maksimal yang berbeda sesuai dan *node* sebanyak 50, 100, dan 150. Dan untuk setiap kecepatan maksimal tersebut dari 10 buah *file* untuk satu protocol *routing* pada satu model transmisi.

##### Gambar 4.1 Implementasi pada ‘setdest’

./setdest -n 50 -p 10 -M 5 -t 100 -x 500 -y 500 > scena1.txt  
./setdest -n 50 -p 10 -M 10 -t 100 -x 500 -y 500 > scenb1.txt  
./setdest -n 50 -p 10 -M 15 -t 100 -x 500 -y 500 > scenc1.txt  
./setdest -n 50 -p 10 -M 20 -t 100 -x 500 -y 500 > scend1.txt

Pada Gambar 4.2 terdapat potongan dari *file* mobilitas scena1.txt 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: 2.00, max x: 500.00, max y: 500.00

#

$node\_(0) set X\_ 184.546814506347

$node\_(0) set Y\_ 245.248521680872

$node\_(0) set Z\_ 0.000000000000

$node\_(1) set X\_ 328.398303549468

$node\_(1) set Y\_ 152.552318580188

$node\_(1) set Z\_ 0.000000000000

##### Gambar 4.2 Posisi *node* dalam X, Y, dan Z

Sedangkan untuk Gambar 4.3 menunjukkan pergeraka *node* tersebut selama waktu simulasi dijalankan untuk setiap *node* diberikan posisi awal dan berkelanjutan untuk menentukan pergerakan *node* berikutnya. Dari potongan dari scena1.txt pada Gambar 4.3, baris pertama mendefinisikan untuk *node* (0) pada detik ke 10 mulai bergerak kea rah tujuan ( 40.51, 219.22) dengan kecepatan 1.23 m/s. baris perintah ini dapat digunakan untuk mengubah arah dan kecepatan pada pergerakan mobilitas *node*.

##### Gambar 4.3 Pergerakan *Node*

$ns\_ at 10.000000000000 "$node\_(0) setdest 40.515052370525 219.222091626887 1.235537673112"  
$ns\_ at 10.000000000000 "$node\_(1) setdest 163.593932992671 118.701264422821 0.920436059928"  
$ns\_ at 10.000000000000 "$node\_(2) setdest 185.263014057350 27.437921428841 0.182305179052"

Kemudian pada gambar 4.4 terdapat potongan dari scena1.txt yang menunjukkan penentuan GOD untuk setiap *node*. GOD (*General Operations Director*) berguna untukmenyimpan informasi global tentang jumlah dan pergerakan *node*. Objek GOD hanya bisa digunakan untuk menyimpan *array* terpendek dari *hop* yang diperlukan untuk mencapai dari satu *node* ke *node* lain. Objek GOD tidak menghitung *array* selama simulasi karena memakan banyak waktu.

$god\_ set-dist 0 1 1  
$god\_ set-dist 0 2 1  
$god\_ set-dist 0 3 1  
$god\_ set-dist 0 4 1  
$god\_ set-dist 0 5 2

##### Gambar 4.4 Pembuatan GOD setiap *Node*

Lalu informasi yang dimuat ke dalam objek GOD dari *file node movement* ditunjukkan pada gambar 4.5. Pada baris pertama dari potongan *file* scena1.txt, informasi yang dimuat pada objek GOD yaitu jalan terpendek antar *node* 5 dan *node* 21 berubah menjadi 2 *hop* pada detik ke 10.73283530487.

##### Gambar 4.5 *Access Point*

$ns\_ at 10.732835304870 "$god\_ set-dist 5 21 2"  
$ns\_ at 10.732835304870 "$god\_ set-dist 21 30 1"  
$ns\_ at 10.765647542719 "$god\_ set-dist 22 23 1"  
$ns\_ at 10.765647542719 "$god\_ set-dist 23 38 2"  
$ns\_ at 10.765647542719 "$god\_ set-dist 23 44 2"

### 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-sce-gen” dan disimpan dalam bentuk *file* cbrgen.tcl. *File* ini dapat digunakan untuk membuat *traffic connection* CBR ataupun TCP pada jaringan pergerakan antar *node*. Pada saat menjalankan perintah pada *file connection* cbrgen.tcl ini, kita harus mendefinisikan tipe *traffic connection*-nya (CBR atau TCP), banyaknya *node* dan koneksi maksimal yang ada pada jaringan tersebut, *random* *seed*, *rate* yang nilai kebalikannya digunakan untuk menghitung waktu interval antar paket CBR yang kemudian disimpan dalam sebuah *file traffic*. Format *command line* yang digunakan untuk menghasilkan gerakan acak pada *node* adalah sebagai berikut:

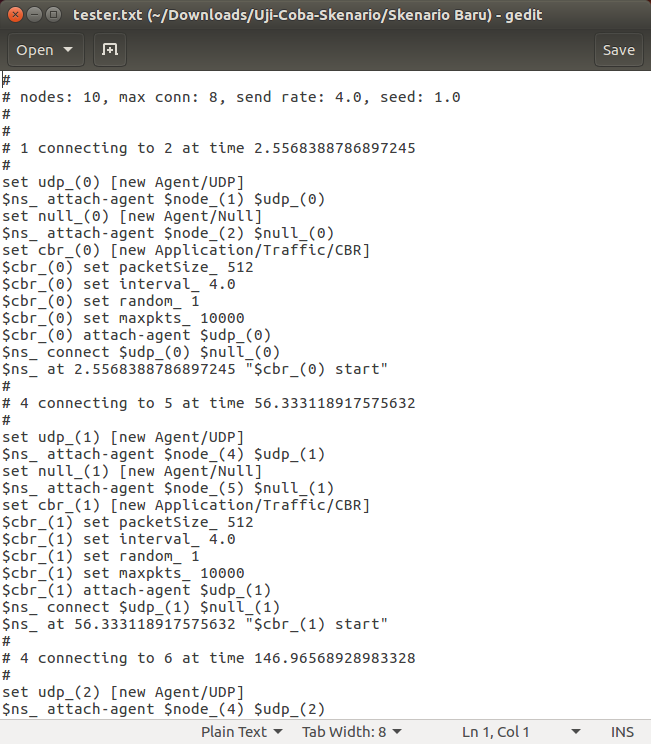
ns cbrgen.tcl [-type cbr|tcp] [-nn nodes] [-seed seed] [-mc connections] [-rate rate] > [outdir/movement file]

Waktu awal untuk koneksi TCP/CBR secara acak yang dihasilkan dengan nilai maksimal ditetapkan pada 180.0 detik. Pada Gambar 4.6 merupakan bentuk implementasi untuk menjalankan cbrgen.tcl untuk membuat *file* *connection* CBR diantara 2 *node*, memiliki maksimal 1 koneksi dengan nilai *seed* 1.0 dan jumlah paket per detik sebanyak 0.25 yang disimpan dalam cbr\_test.txt yang nantinya akan digunakan pada saat simulasi NS-2.

##### Gambar 4.6 Implementasi Koneksi cbrgen.txt

ns cbrgen.tcl -type cbr -nn 10 -seed 1.0 -mc 8 -rate 0.25 > cbr\_test.txt

Kemudian pada Gambar 4.7 menunjukkan *output* yang disimpan dalam cbr\_test.txt sehingga menghasilkan koneksi CBR dan menggunakan *Agent* UDP. Koneksi UDP di sini merupakan konfigurasi antara *node* ke-1 dan 2. Interval pengiriman paket yang dilakukan setiap satu detik dengan besar paket 512 byte dan maksimal pengiriman paket 10000.



#### Gambar 4.7 *Output* cbr\_test.txt

## 4.3 Implementasi Simulasi pada NS-2

Untuk mensimulasikan MANET dalam lingkungan NS-2, skrip tcl yang telah ada pada *patch* protocol *routing* DSR. Untuk skenario *node movement (mobility generation)* dan *traffic connection generation* dan yang disimpan dalam bentuk .txt diberikan parameter-parameter yang sesuai dengan perancangan agar dapat dijalankan pada NS-2. Parameter-parameter tersebut dibuat menggunakan Bahasa Tcl/Otcl.

Pada Gambar 4.8 menunjukkan skrip konfigurasi awal parameter-parameter yang diberikan untuk menjalankan MANET pada NS-2. Baris pertama merupakan konfigurasi tipe *channel* yang digunakan *Wireless Channel*. Baris kedua merupakan tipe transmisi yang digunakan pada Tugas Akhir ini yang digunakan adalah transmisi *TwoRayGround* untuk pengujian simulasi. Tipe Mac yang digunakan adalah Mac 802.11. Pada baris diatas juga dilakukan konfigurasi tipe *queue* dari *interface*, tipe *link layer*, tipe *antenna* dan jumlah maksimal *packet* pada *interface queue*. Baris ke-9 sampai baris ke-17 berturut-turut merupakan koordinat x serta koordinat y sebesar 500 meter, jumlah paket dan *node* yaitu 50 *nodes*, protocol *routing* yang digunakan yaitu DSR, besarnya *seed* yaitu 0.0, waktu simulasi diakhiri pada detik ke-100, *file traffic connection* yang digunakan yaitu cbr\_test.txt dan terakhir ialah *file* skenario *node movement* yang digunakan adalah scena1.txt.

##### Gambar 4.8 Konfigurasi awal parameter NS-2

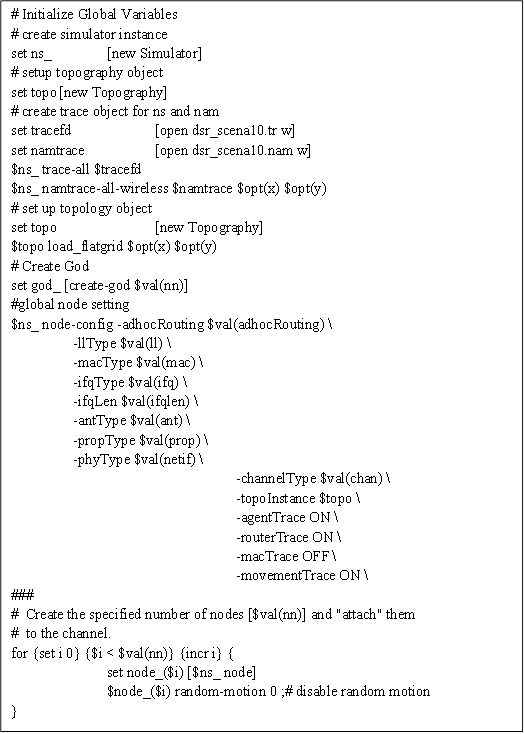
set val(chan) Channel/WirelessChannel  
set val(prop) Propagation/TwoRayGround  
set val(netif) Phy/WirelessPhy  
set val(mac) Mac/802\_11  
set val(ifq) CMUPriQueue  
set val(ll) LL  
set val(ant) Antenna/OmniAntenna  
set opt(x) 500   
set opt(y) 500   
set val(ifqlen) 50   
set val(nn) 50   
set val(seed) 0.0  
set val(adhocRouting) DSR  
set val(stop) 100   
set val(cp) "cbr\_test.txt"   
set val(sc) "scena1.txt"

Pada gambar 4.9 merupakan pengaturan dari *transmission range* yang digunakan pada simulasi. Nilai yang diubah adalah *RXThresh\_ (Receiver Sensitivity Threshold)*. Nilai 1.4268ie-08 pada variable tersebut memiliki artian bahwa jangkauan yang dapat dicapai sejauh 100 meter.

Phy/WirelessPhy set RXThresh\_ 1.42681e-08 ;#100m

##### Gambar 4.9 Konfigurasi *Transmission Range* pada NS-2

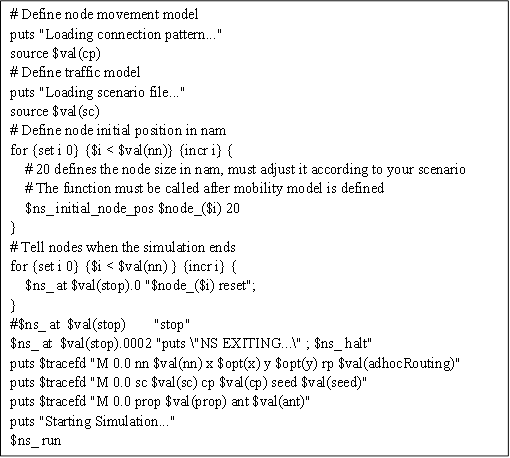
Gambar 4.10 merupakan skrip yang digunakan untuk pengaturan variable global yang diawali dengan *set ns* untuk pembuatan simulator baru. *Set tracefd* dan *set namtrace* merupakan pengaturan nama *trace file* berekstensi .tr dan *file network* animator .nam akan dihasilkan dan disimpan.



##### Gambar 4.10 Konfigurasi *Trace File*, NAM dan Pergerakan *Node* pada NS-2

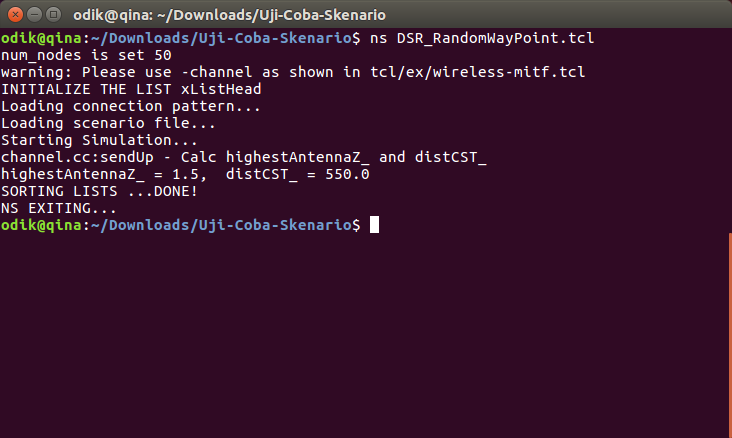
Pada *set tracefd* dapat dilakukan pengaturan untuk menghasilkan *trace file* sesuai dengan keinginan pengguna. Terdapat 2 jenis format *trace file* namun pada Tugas Akhir ini digunakan *old trace format*. *Set topo* merupakan pengaturan untuk objek topografi berdasarkan pada luas koordinat yang telah dikonfigurasi sebelumnya. *Create-god* dan *node-config-channelType* merupakan konfigurasi yang dilakukan pada *node-node* yang akan dibuat. Pada *create-god* dilakukan implementasi *node-node* yang akan dibuat sesuai dengan parameter pada *set-val(nn)* sedangkan pada *node-config-channelType* merupakan konfigurasi *node* sesuai dengan parameter-parameter yang telah ditambahkan sebelumnya pada gambar 4.8 seperti tipe *link layer*, tipe mac dan tipe transmisi. 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 4.11 merupakan bagian akhir dari keseluruhan skrip yang digunakan untuk menginisialisasi 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 movemet (mobility generation)* dan *traffic connection pattern* kemudian pengiriman paket data dimulai pada detik ke-0 dan berhenti pada detik ke-100 seperti yang telah dikonfigurasi sebelumnya pada Gambar 4.8.



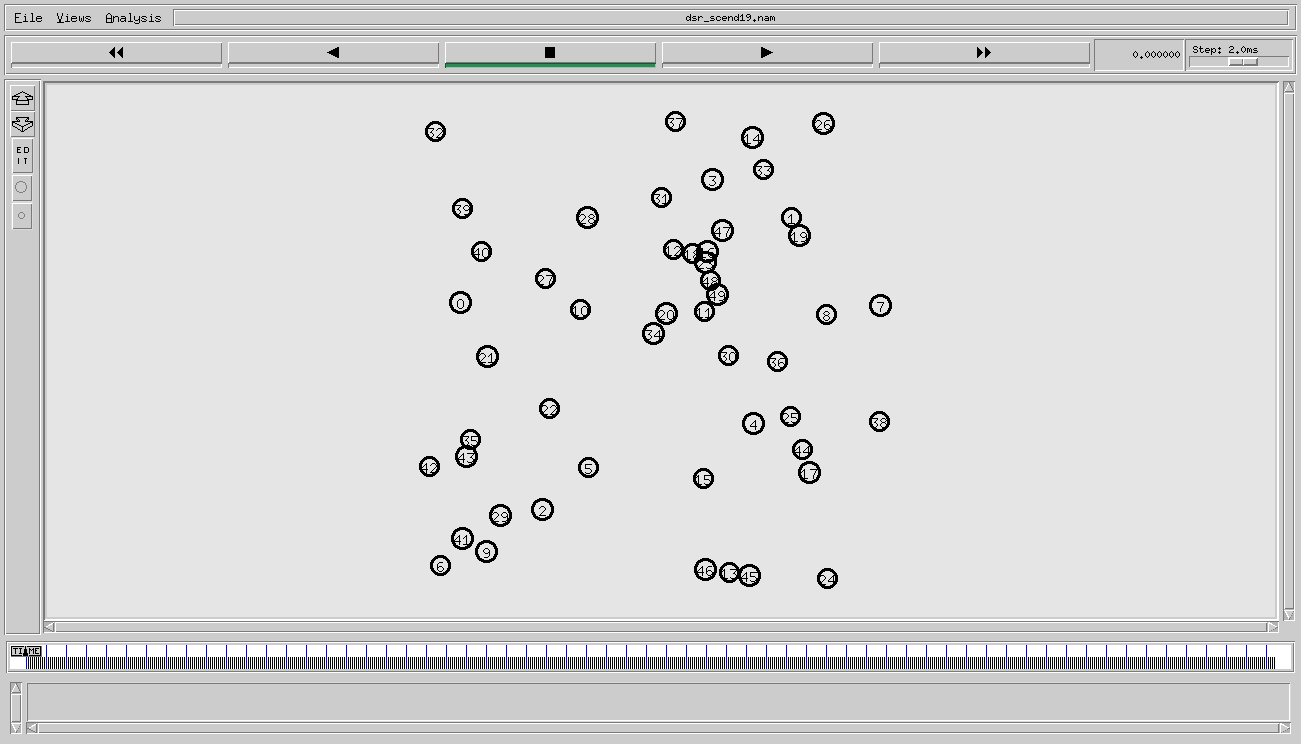
##### Gambar 4.11 Konfigurasi pengiriman paket data NS-2

Pada gambar 4.12 merupakan contoh eksekusi *file* .tcl menggunakan model transmisi *TwoRayGround* dengan jumlah *node* 50 yang bersifat acak atau *Random Way Point* sesuai dengan skenario *node movement* yang telah di-*generate* sebelumnya.

****

##### Gambar 4.12 Perintah Eksekusi *TwoRayGround*

Hasil yang diperoleh dari eksekusi berkas .tcl berupa *trace file* berbentuk .tc dan *file* animasi pengiriman paket data berbentuk .nam. *File* .tr inilah yang akan dianalisis parameter-parameternya berupa PDR, *delay* dan RO. Pada gambar 4.13 menunjukkan tampilan hasil dari *file* .nam yang dihasilkan.



##### Gambar 4.13 Visualisasi hasil simulasi pada NAM

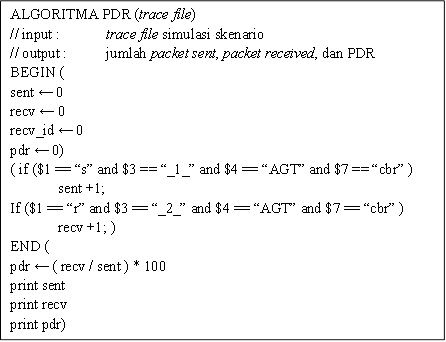
## 4.4 Implementasi Metrik Analisis

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

### 4.4.1 *Packet Delivery Ratio* (PDR)

Proses perhitungan PDR dilakukan dengan menghitung jumlah paket data terkirim yang dilakukan oleh *node 1*  dan jumlah paket data yang diterima oleh *node* 2 pada satu *trace file*. Penambahan jumlah paket terkirim dilakukan apabila pada baris *trace* yang bersangkutan mengandung semua kondisi dimana kolom pertama mengandung huruf “s” yang menandakan *send packet*, kolom ke-3 menunjukkan bahwa *node* yang melakukan pengiriman adalah *node* 1, kolom ke-4 dan kolom ke-7 mengandung huruf “AGT” dan “cbr” yang menandakan pengiriman paket yang dilakukan adalah pengiriman paket data. Pencatatan jumlah paket yang diterima dilakukan apabila pada baris *trace* yang bersangkutan mengandung semua kondisi dimana kolom pertama mengandung huruf “r” yang menandakan *received packet*, kolom ke-3 menunjukkan bahwa *node* yang menerima packet data adalah *node* 2, kolom ke-4 dan kolom ke-7 mengandung huruf “AGT” dan “cbr” yang menandakan penerimaan paket yang diterima adalah paket data. Perhitungan dilakukan sampai baris terakhir *trace file*, dan hasilnya adalah hasil hitung nilai PDR simulasi skenario.

*Psedeucode* PDR ditunjukkan pada Gambar 4.14 dan implementasinya dapat dilihat di lampiran.

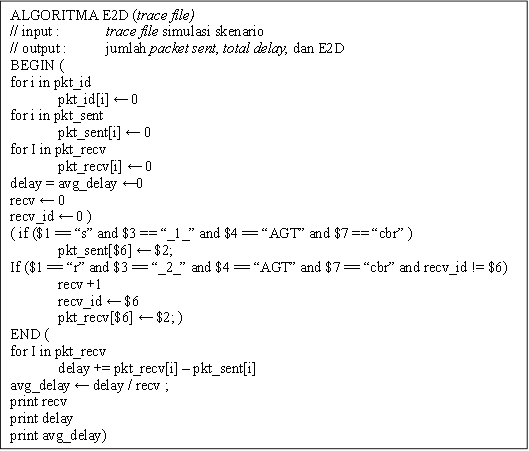


##### Gambar 4.14 *Pseudecode* PDR

### 4.4.2 *End-to-End Delay* (E2D)

Proses perhitungan E2D dilakukan dengan menghitung selisih waktu paket data terkirim yang dilakukan oleh *node* 1 dan waktu paket data yang diterima oleh *node* 2 pada satu *trace file*. Pencatatan waktu paket terkirim pada kolom ke-2 dilakukan apabila pada baris *trace* yang bersangkutan mengandung semua kondisi dimana kolom pertama mengandung huruf “s” yang menandakan *send packet*, kolom ke-3 menunjukkan bahwa *node* yang melakukan pengiriman adalah *node* 1, kolom ke-4 dan kolom ke-7 mengandung huruf “AGT” dan “cbr” yang menandakan pengiriman paket yang dilakukan adalah pengiriman paket data. Perhitungan waktu dan pencatatan ID serta jumlah paket yang diterima dilakukan apabila pada baris *trace* yang bersangkutan mengandung semua kondisi dimana kolom pertama mengandung huruf “r” yang menandakan *received packet*, kolom ke-3 menunjukkan bahwa *node* yang menerima packet data adalah *node* 2, kolom ke-4 dan kolom ke-7 mengandung huruf “AGT” dan “cbr” yang menandakan penerimaan paket yang diterima adalah paket data. Perhitungan dilakukan sampai baris terakhir *trace file*, dan dilakukan perhitungan nilai E2D dengan menghitung selisih *delay* paket mulai dari pengiriman sampai paket diterima pada simulasi skenario.

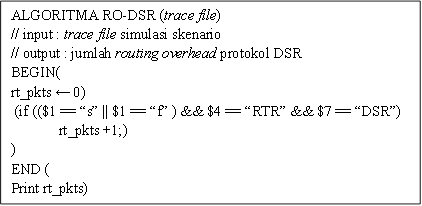
*Psedeucode* E2D ditunjukkan pada Gambar 4.15 dan implementasinya dapat dilihat di lampiran.



##### Gambar 4.15 *Pseudeucode* E2D

### 4.4.3 Routing Overhead (RO)

Implementasi perhitungan metrik *Routing overhead* DSR dihitung apabila kondisi-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 gambar 4.16 implementasinya dapat dilihat pada lampiran.



##### Gambar 4.16 *Pseudeucode* RO

# BAB V PENGUJIAN DAN EVALUASI

Bab ini menjelaskan mengenai pengujian dari skenario NS-2 yang telah dilakukan. Pengujian fungsionalitas dibagi ke dalam beberapa skenario pengujian.

## 5.1 Lingkungan Pengujian

Uji coba dilakukan pada laptop yang telah terapasang dua sistem operasi yaitu Windows dan Linux. Spesifikasi laptop yang digunakan ditunjukkan pada tabel 5.1.

##### Tabel 5.1 Spesifikasi Laptop yang Digunakan

|  |  |
| --- | --- |
| **Komponen** | **Spesifikasi** |
| Prosessor | Intel(R) Celeron(R) CPU N2840 @ 2.16GHz 2.16 GHz |
| Sistem Operasi | Windows 7 Home Edition, Linux Ubuntu 16.04 LTS 64-bit |
| Memori | 4GB DDR3 |
| Media Penyimpanan | 500GB |
|  |  |

## 5.2 Kriteria Pengujian

Seperti yang telah ditunjukkan pada Tabel 3.1 sebagai lingkungan topologi yang sangat dinamis, pengujian pada skenario yang dihasilkan oleh *mobility generator* dari NS-2 menggunakan beberapa kriteria. Pada tabel 5.2 menunjukkan kriteria yang ditentukan didalam scenario.

##### Tabel 5.2 Kriteria Pengujian

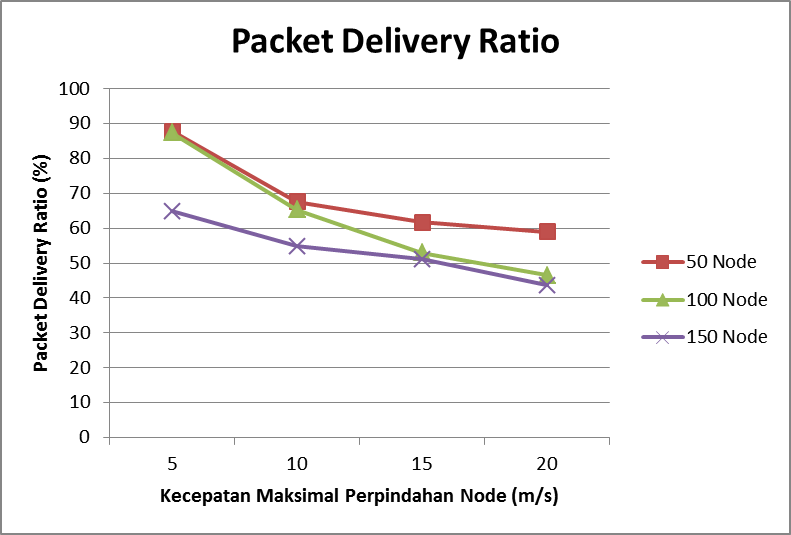
|  |  |
| --- | --- |
| **Kriteria** | **Spesifikasi** |
| Skenario | MANET (*Randow Way Point*), *mobility generator* |
| Kecepatan Maksimal Perpindahan Node (m/s) | 5, 10, 15, 20 |
| Jumlah Percobaan | 10 kali |
| Jarak *Node* 1 dan *Node* 2 | Acak |
| Posisi Awal *Node* | Acak |
| Pergerakan | Acak |
| Protokol *Routing* | DSR |
| Pengiriman Paket Data | *TwoRayGround* |
|  |  |

## 5.3 Analisis *Packet Delivery Ratio* (PDR)

*Trace file* hasil mejalankan skenario *mobility generator* kemudian dianalisis nilai PDR melalui skrip pdr.awk. Hasil tiap perhitungan PDR skenario ditabulasikan seperti pada Tabel 5.3 dan ditampilkan dalam bentuk grafik pada Gambar 5.1.

##### Tabel 5.3 PDR Skenario *Mobility Generator*



****

#### Gambar 5.1 Grafik PDR Skenario Mobility Generator

Pada gambar 5.1 terlihat bahwa performa *Packet elivery Ratio* yang dihasilkan semakin menurun seiring bertambahnya kecepatan maksimal perpindahan *node*. Pada jumlah node 50, ketika kecepatan maksimal perpindahan *node* 5 m/s, PDR yang dihasilkan yaitu 88%. Perubahan nilai PDR terjadi ketika kecepatan maksimal perpindaan *node* menjadi 10 m/s. Nilai PDR menurun menjadi 68 %. Begitu juga pada saat kecepatan maksimal perpindahan *node* dinaikkan lagi menjadi 15 m/s, nilai PDR semakin menurun menjadi 62 %. Dan nilai PDR kembali menurun ketika kecepatan maksimal perpindahan *node* dinaikkan menjadi 20 m/s, nilai PDR menjadi 59 %. Hal ini disebabkan karena dengan bertambahnya kecepatan gerak node maka akan sering terjadi perubahan posisi node sehingga rute rusak, menyebabkan hilangnya paket data yang lewat (drop), akibat akhirnya PDR juga semakin turun.

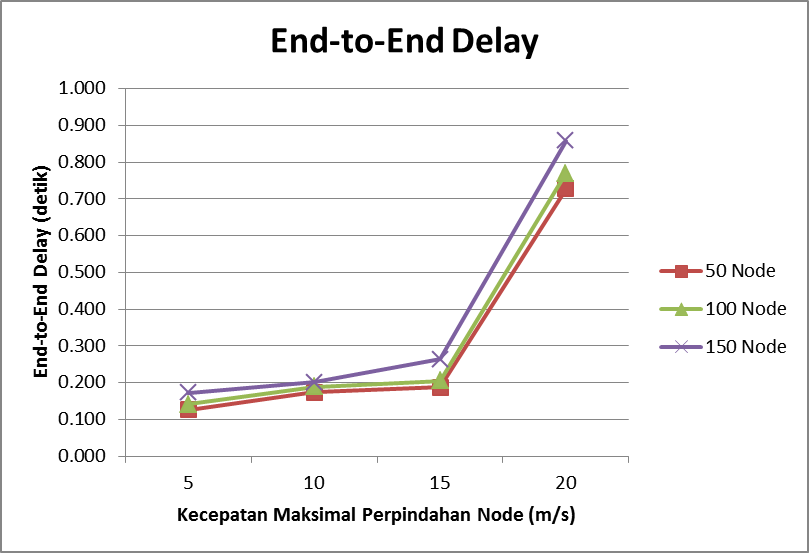
Ketika jumlah *node* semakin bertambah, nilai PDR yang dihasilkan juga semakin menurun. Seperti yang telah ditunjukkan pada Gambar 5.1, terlihat bahwa pada kecepatan maksimal perpindahan *node* 5 m/s nilai PDR pada jumlah *node* 50 adalah 88 %. Sedangkan pada jumlah *node* 100 nilai PDR menurun menjadi 87 % dan pada jumlah *node* 150 nilai PDR kembali menurun menjadi 65 %. Hal ini disebabkan karena semakin bertambahnya jumlah node maka trafik routing akan semakin padat. Routing DSR menggunakan mekanisme flooding pada proses route discovery yang membanjiri jaringan dengan route request. Hal ini menurunkan PDR karena kanal banyak digunakan untuk mengirimkan paket routing dan ketersediaan kanal semakin sedikit untuk mengirimkan paket data.

## 5.4 Analisis *End-to-End Delay* (E2D)

*Trace file* hasil mejalankan skenario *mobility generator* kemudian dianalisis nilai E2D melalui skrip e2d.awk. Hasil tiap perhitungan E2D skenario ditabulasikan seperti pada Tabel 5.4 dan ditampilkan dalam bentuk grafik pada Gambar 5.2.

Tabel 5.4 E2D Skenario *Mobility Generator*





#### Gambar 5.2 Grafik E2D Skenario *Mobility Generator*

Pada gambar 5.2 terlihat bahwa performa *End-to-End Delay* yang dihasilkan semakin meningkat seiring bertambahnya kecepatan maksimal perpindahan *node.* Pada jumlah node 50, ketika kecepatan maksimal perpindahan *node* 5 m/s, E2D yang dihasilkan yaitu 0.126 detik. Perubahan nilai E2D terjadi ketika kecepatan maksimal perpindaan *node* menjadi 10 m/s. Nilai E2D meningkat menjadi 0.174 detik. Begitu juga pada saat kecepatan maksimal perpindahan *node* dinaikkan lagi menjadi 15 m/s, nilai E2D semakin meningkat menjadi 0.188 detik. Dan nilai E2D mengalami peningkatan drastis ketika kecepatan maksimal perpindahan *node* dinaikkan menjadi 20 m/s, nilai E2D menjadi 0.728 detik. Hal ini disebabkan karena pada saat node bergerak dengan kecepatan tinggi, rute menjadi sering tidak tersedia. Sehingga paket data akan tertahan di buffer sementara sampai rute baru ditemukan hal ini akan mengakibatkan rata-rata delay pengiriman paket data akan semakin besar.

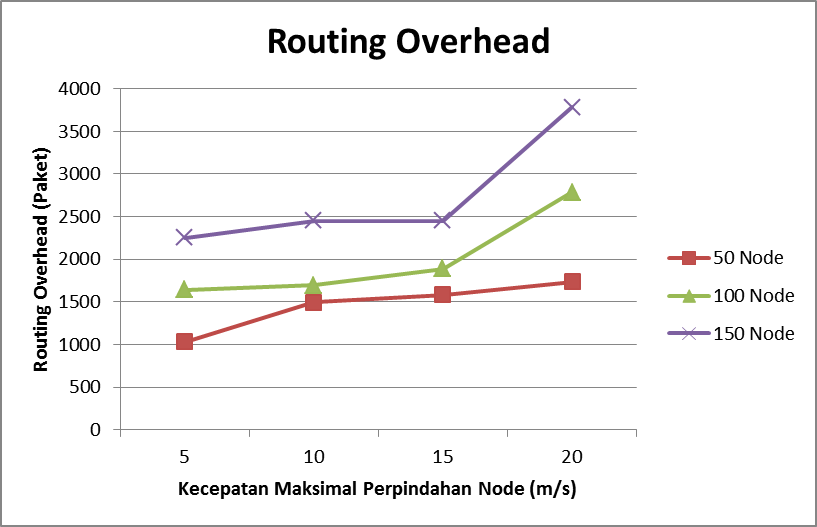
Ketika jumlah *node* semakin bertambah, nilai E2D yang dihasilkan juga semakin meingkat. Seperti yang telah ditunjukkan pada Gambar 5.2, terlihat bahwa pada kecepatan maksimal perpindahan *node* 5 m/s nilai E2D pada jumlah *node* 50 adalah 0.126 detik. Sedangkan pada jumlah *node* 100 nilai E2D meningkat menjadi 0.141 detik dan pada jumlah *node* 150 nilai E2D kembali meningkat menjadi 0.172 detik. Hal ini disebabkan karena pada saat jumlah node bertambah besar, maka jaringan bertambah padat dan peluang tabrakan antar paket bertambah besar.

## 5.5 Analisis *Routing Overhead* (RO)

*Trace file* hasil mejalankan skenario *mobility generator* kemudian dianalisis nilai RO melalui skrip ro.awk. Hasil tiap perhitungan RO skenario ditabulasikan seperti pada Tabel 5.5 dan ditampilkan dalam bentuk grafik pada Gambar 5.3.

##### Tabel 5.5 RO Skenario *Mobility Generator*



****

##### Gambar 5.3 Grafik RO Skenario *Mobility Generator*

Pada gambar 5.3 terlihat bahwa performa *Routing Overhead*  semakin meningkat seiring bertambahnya kecepatan maksimal perpindahan *node*. Pada jumlah node 50, ketika kecepatan maksimal perpindahan *node* 5 m/s, RO yang dihasilkan yaitu 1031 paket. Perubahan nilai RO terjadi ketika kecepatan maksimal perpindaan *node* menjadi 10 m/s. Nilai RO meningkat menjadi 1493 paket. Begitu juga pada saat kecepatan maksimal perpindahan *node* dinaikkan lagi menjadi 15 m/s, nilai RO semakin meningkat menjadi 1584 paket. Dan nilai RO kembali meningkat ketika kecepatan maksimal perpindahan *node* dinaikkan menjadi 20 m/s, nilai RO menjadi 1738 paket. Hal ini disebabkan karena ketika kecepatan gerak *node* bertambah, maka posisi node juga semakin mudah berubah dan konfigurasi jaringan sering berubah-ubah. Sehingga node pengirim akan lebih banyak melakukan broadcast paket routing pada proses route discovery (RREQ dan RREP) untuk mendapatkan rute yang baru.

Ketika jumlah *node* semakin bertambah, nilai RO yang dihasilkan juga semakin meningkat seperti yang telah ditunjukkan pada Gambar 5.3. Terlihat bahwa pada kecepatan maksimal perpindahan *node* 5 m/s nilai RO pada jumlah *node* 50 adalah 1031 paket. Sedangkan pada jumlah *node* 100 nilai RO meningkat menjadi 1642 paket dan pada jumlah *node* 150 nilai RO kembali meningkat menjadi 2252 paket. Hal ini disebabkan karena adanya mekanisme flooding pada algoritma routing DSR. Routing DSR akan melakukan mekanisme flooding dengan mengirimkan paket RREQ dan RREP pada saat pencarian rute dan ketika jumlah node bertambah maka jumlah paket RREQ dan RREP yang dikirimkan juga semakin bertambah besar.

# 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 dating.

## 6.1 Keimpulan

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

1. Performa kinerja protokol *routing* DSR pada jaringan MANET adalah sebagai berikut :
   1. Performa *Packet Delivery Ratio* yang dihasilkan menurun dengan nilai rentang pada kecepatan awal 80 % menjadi 50 % dari kecepatan maksimal perpindahan *node* 5 m/s hingga 20 m/s.
   2. Performa *End-to-End Delay* yang dihasilkan meningkat dengan nilai rentang pada kecepatan awal 0.147 detik menjadi 0.785 detik dari kecepatan maksimal perpindahan *node* 5 m/s hingga 20 m/s.
   3. Performa *Routing Overhead* yang dihasilkan meningkat dengan nilai rentang pada kecepatan awal 1642 paket menjadi 2768 paket dari kecepatan maksimal perpindahan *node* 5 m/s hingga 20 m/s.
2. Kecepatan node sangat mempengaruhi performa kinerja protokol routing DSR. Node yang bergerak dengan cepat menyebabkan perubahan topologi jaringan dengan cepat dan tak terduga dari waktu ke waktu sehingga performa kinerja DSR menjadi buruk.
3. Jumlah node juga sangat mempengaruhi performa kinerja protokol routing DSR. Semakin banyak jumlah *node* yang ada performa DSR akan semakin menurun karena banyaknya *node* yang terhubung.

## 6.2 Saran

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

1. Dapat dilakukan penambahan jumlah percobaan skenario untuk mencapai hasil yang optimal.
2. Dapat dilakukan percobaan pada lingkungan VANET untuk penerapan model transmisi *TwoRayGround.*
3. Memperbesar waktu pengamatan (simulasi), sehingga diperoleh performansi yang paling baik untuk protokol routing DSR.
4. Melakukan pengujian terhadap Protokol routing lainnya, sehingga bisa diketahui protokol routing yang paling tepat untuk jaringan ad hoc.

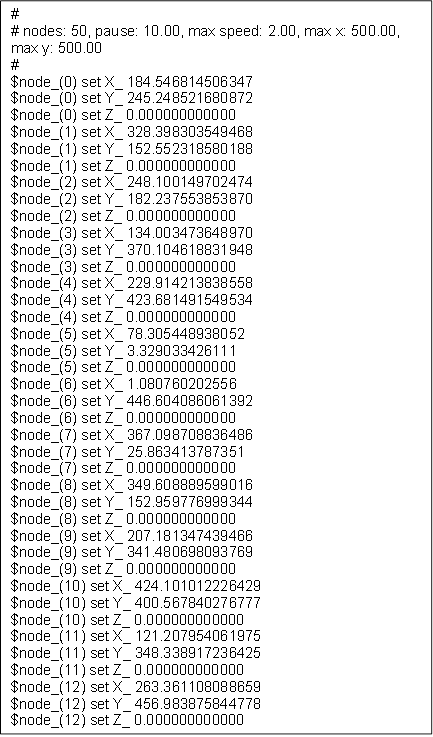
# DAFTAR PUSTAKA

|  |  |
| --- | --- |
| [1] | “MANET” [Online].  Available : <https://techterms.com/definition/manet>. [Accessed 9 Mei 2018] |
| [2] | “Mobile Ad-Hoc Networks (MANET)” [Online]. Available : <http://ezine.echo.or.id/issue26/008.txt>. [Acceessed 9 Mei 2018] |
| [3] | Dhiyaan Sabila Ramadani, Juni. 2016. “Studi Perbandingan Kinerja Model Transmisi TwoRayGround dan Nakagami pada Optimized Link State Routing (OLSR) di Lingkungan Mobile Ad Hoc Network (MANET) menggunakan Network Simulator 2 (NS-2). **Mobile Ad Hoc Network (MANET)** 7 |
| [4] | Amirhossein Moravejosharieh, Hero Modares, Rosli Salleh and Ehsan Mostajeran. Feb. 2013. “Performance Analysis of AODV, AOMDV, DSR, DSDV Routing Protocols in Vehicular Ad Hoc Network” **Dynamic Source Routing (DSR)** Vol2(7), 69 |
| [5] | Firdaus Nutrihadi, Jan. 2016. “Studi Kinerja VANET *Scenario Generators:* SUMO dan VanetMobism untuk Implementasi *Routing Protocol* AODV menggunakan Network Simulator 2 (NS-2). **Network Simulator 2 (NS-2)** 18 |
| [6] | “Generating Traffic-Connection and Movement Files” [Online]. Available : <http://www.engr.iupui.edu/~dskim/tutorials/ns2/?section=10>. [Accessed 14 Februari 2018] |
| [7] | “Praktikum 1Berkenalan dengan “awk”” [Online]. Available : <ftp://ftp.itb.ac.id/pub/ISO-IMAGES/linux/otodidak.freeserver.me/teks/kuliahawk/Praktikum%2001%20-%20Berkenalan%20dengan%20AWK.pdf>  [Accessed 6 Mei 2018] |

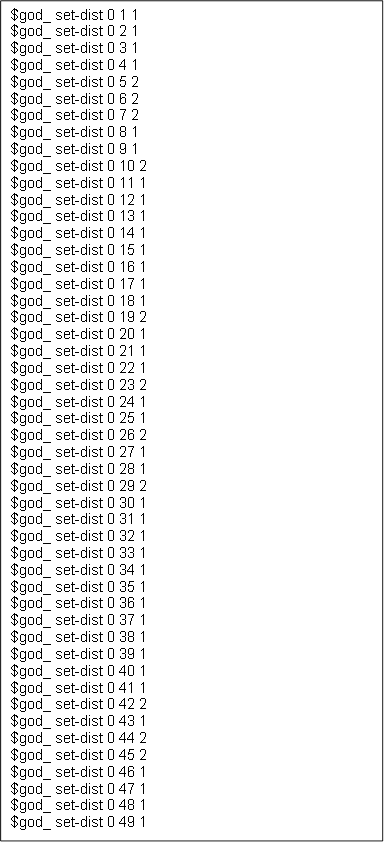
*[Halaman ini sengaja dikosongkan]*

# LAMPIRAN

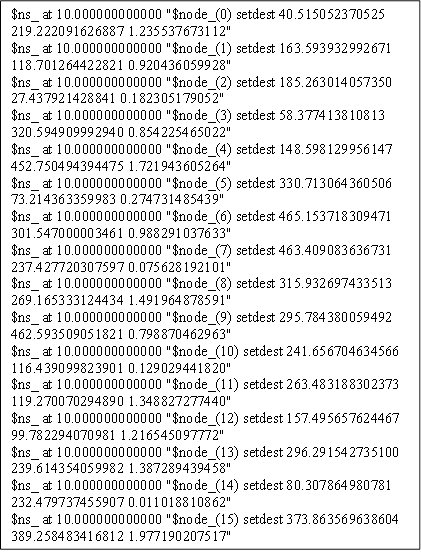
Pada bagian ini adalah lampiran sebagai dokumen lengkap dari buku Tugas Akhir. Pada bagian ini akan diberikan konfigurasi-konfigurasi terhadap posisi antar *node* dan implementasi metrik PDR, E2D, RO serta dilampirkan juga cuplikan hasil trace file.



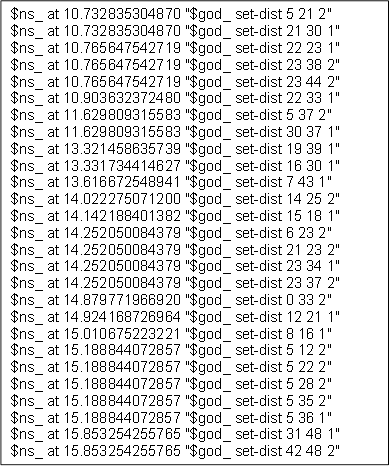
##### Gambar 7.1 Posisi *Node* dari potongan skenario



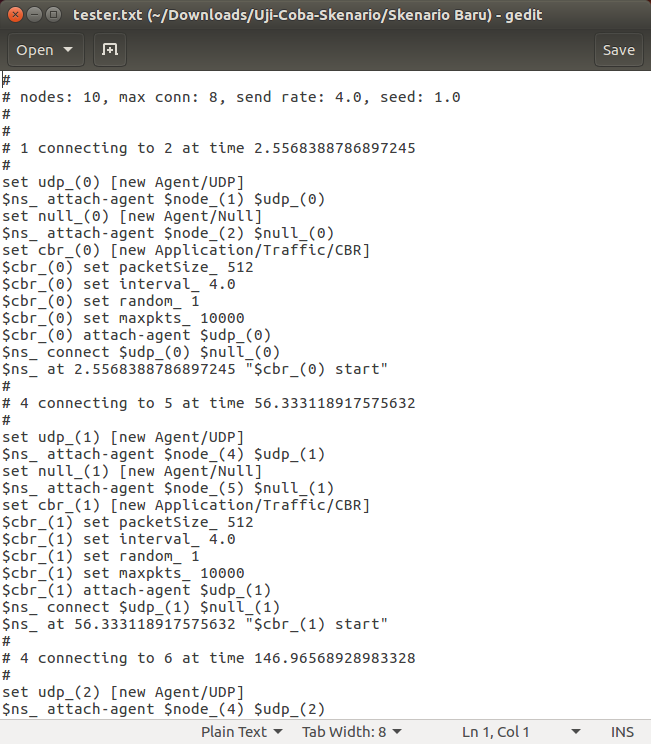
##### Gambar 7.2 Pembuatan GOD setiap *node* dari potongan skenario



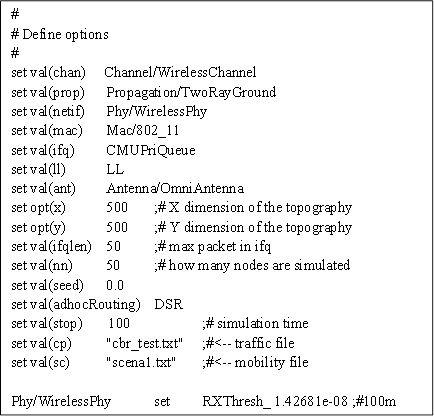
##### Gambar 7.3 Pergerakan setiap *node* dari potongan skenario

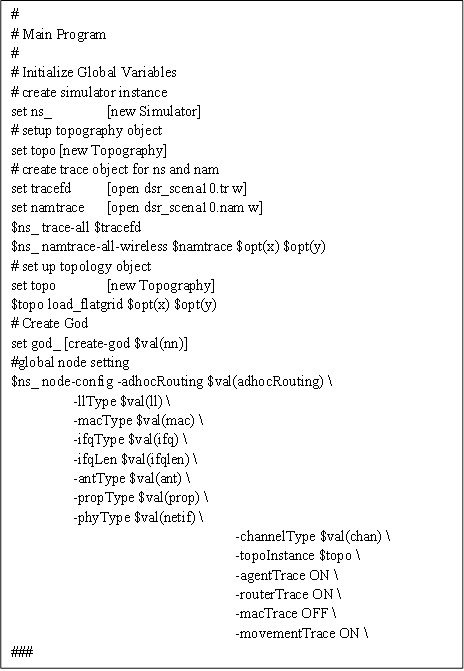


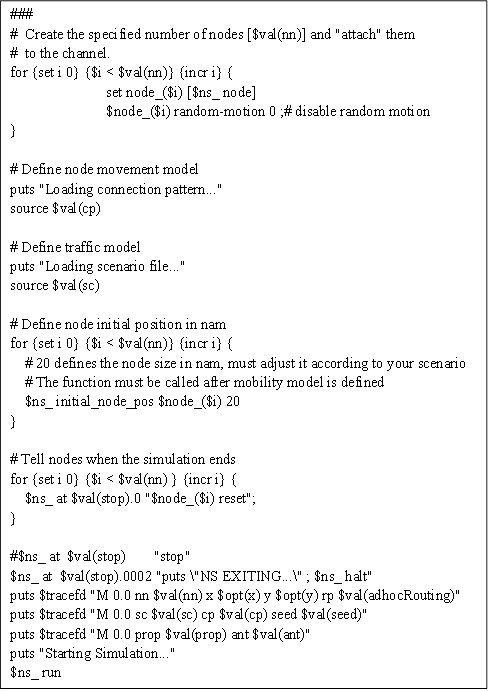
##### Gambar 7.4 Informasi pada GOD dari potongan skenario



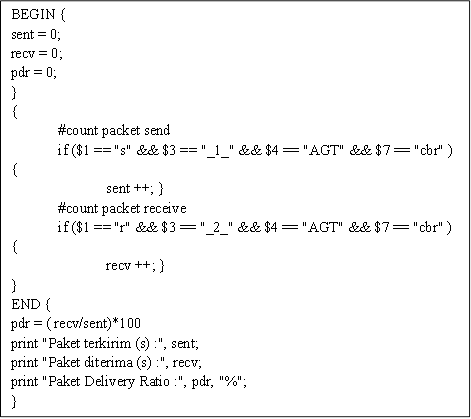
##### Gambar 7.5 Koneksi yang digunakan pada cbr\_test.txt



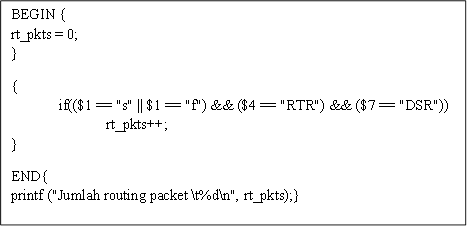




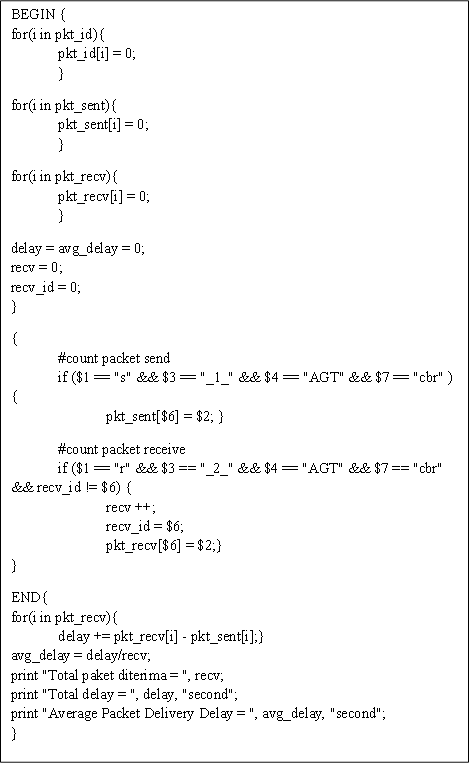
##### Gambar 7.6 *File* .tcl dari ptotokol *routing* DSR



##### Gambar 7.7 Implementasi *Packet Delivery Ratio*



##### Gambar 7.8 implementasi Routing Overhead

Gambar 7.9 Implementasi End-to-End Delay

*[Halaman ini sengaja dikosongkan]*

# BIODATA PENULIS



**Jordy Adhityo Pradono**, biasa dipanggil Oddy, dilahirkan di kota Surabaya pada tanggal 11 Juni 1993. Penulis adalah anak pertama dari dua bersaudara. Penulis menempuh pendidikan TK Al-Manar(1998-1999), TK Hang Tuah 11(1999-2000), SDN Babatan I / 456 Surabaya(2000-2006), SMPN 12 Surabaya (2006-2009), dan SMAN 10 Surabaya (2009-2011). Pada tahun 2011, penulis mengikuti SNMPTN dan diterima di strata satu Jurusan Teknik Informatika Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember Surabaya angkatan 2011 yang terdaftar dengan NRP 5111100101. Di Jurusan Teknik Informatika ini, penulis mengambil bidang minat Arsitektur dan Jaringan Komputer (AJK). Selama menempuh kuliah, penulis juga aktif dalam kegiatan di luar ITS sebagai salah satu anggota panitia event “CHOCODAYS”. Penulis dapat dihubungi melalui alamat *e-mail* edgevanhite11@gmail.com.