

TUGAS AKHIR - KI141502

Analisis Kinerja Destination Sequenced Distance Vector (DSDV) dengan Model Propagasi Nakagami pada Mobile Ad Hoc Network (MANET)

MAHARANI WAHYU SIWI

NRP 5112 100 216

Dosen Pembimbing

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

HENNING TITI CIPTANINGTYAS, S.Kom., M.Kom.

DEPARTEMEN INFORMATIKA

Fakultas Teknologi Informasi dan Komunikasi

Institut Teknologi Sepuluh Nopember

Surabaya 2018

RBTC



TUGAS AKHIR - KI141502

Analisis Kinerja Destination Sequenced Distance Vector (DSDV) dengan Model Propagasi Nakagami pada Mobile Ad Hoc Network (MANET)

MAHARANI WAHYU SIWI
NRP 5112 100 216

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

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

[Halaman ini sengaja dikosongkan]

RBTC



FINAL PROJECT - KI141502

Performance Analysis of Destination Sequenced Distance Vector (DSDV) with Nakagami Propagation Model on Mobile Ad Hoc Network (MANET)

MAHARANI WAHYU SIWI
NRP 5112 100 216

Advisor

Dr. Eng. RADITYO ANGGORO, S.Kom., M.Sc.
HENNING TITI CIPTANINGTYAS, S.Kom., M.Kom.

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

[Halaman ini sengaja dikosongkan]

RBTC

LEMBAR PENGESAHAN

ANALISIS KINERJA DESTINATION SEQUENCED DISTANCE VECTOR (DSDV) DENGAN MODEL PROPAGASI NAKAGAMI PADA 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 Departemen Informatika
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember Surabaya

Oleh:

MAHARANI WAHYU SIWI

NRP. 5112 100 216

Disetujui oleh Pembimbing Tugas Akhir

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



Henning Titi Ciptaningtyas, S.Kom. M.Kom.
NIP: 198407082010122004

**SURABAYA
JANUARI, 2018**

[Halaman ini sengaja dikosongkan]

RBTC

Analisis Kinerja Destination Sequenced Distance Vector (DSDV) dengan Model Propagasi Nakagami pada Mobile Ad Hoc Network (MANET)

Nama Mahasiswa : Maharani Wahyu Siwi
NRP : 5112 100 216
Jurusan : Teknik Informatika FTIf - ITS
Dosen Pembimbing : Dr. Eng. Radityo Anggoro, S.Kom.,
M.Sc.
Henning Titi Ciptaningtyas, S.Kom.,
M.Kom.

ABSTRAK

Zaman sekarang sudah semakin canggih, banyak berkembang teknologi jaringan nirkabel (wireless) pada perangkat mobile. Teknologi nirkabel memungkinkan dua perangkat untuk saling berkomunikasi secara langsung dalam keadaan bergerak tanpa memerlukan jaringan infrastruktur yang tetap. Jaringan yang bersifat sementara ini disebut dengan Mobile Ad Hoc Network (MANET).

Mobile Ad Hoc Network (MANET) adalah kumpulan dari beberapa wireless node yang dapat di set-up secara dinamis dimana saja dan kapan saja, tanpa menggunakan infrastruktur jaringan yang ada. MANET juga merupakan jaringan sementara yang dibentuk oleh beberapa mobile node tanpa adanya pusat administrasi dan infrastruktur kabel. Namun untuk membuat implementasi MANET secara real di dunia nyata masih sulit dilakukan karena berbagai faktor. Oleh karenanya, penelitian yang dilakukan masih banyak yang menggunakan simulator. Simulator yang biasa digunakan untuk melakukan penelitian terhadap MANET adalah Network Simulator.

Dalam Tugas Akhir ini dilakukan penelitian terhadap skema MANET yang dihasilkan oleh file node-movement dan traffic-pattern yang telah ada pada distribusi network simulator.

Penelitian ini menggunakan NS-2 sebagai network simulator dengan protokol proaktif MANET jenis DSDV (Destination Sequenced Distance Network) sebagai protokol routing yang digunakan serta menggunakan model transmisi Nakagami yang ada pada NS-2.

Setelah skema MANET berhasil dibuat akan dilakukan uji fungsionalitas dan performa melalui beberapa skenario yang telah ditentukan. Hasil dari pengujian adalah suatu grafik performa model transmisi Nakagami pada protokol routing DSDV di lingkungan MANET dengan menggunakan NS-2 yang diukur berdasarkan routing overhead, packet delivery ratio, dan delay pengiriman paket dari satu node ke node lainnya.

Kata Kunci: MANET, Nakagami, Network Simulator, NS-2, DSDV.

Performance Analysis of Destination Sequenced Distance Vector (DSDV) with Nakagami Propagation Model on Mobile Ad Hoc Network (MANET) v

Name : Maharani Wahyu Siwi
NRP : 5112 100 216
Major : Informatics Engineering, IT Dept – ITS
Advisor : Dr. Eng. Radityo Anggoro, S.Kom., M.Sc.
Henning Titi Ciptaningtyas, S.Kom., M.Kom.

ABSTRACT

There are many kinds of wireless network technology for mobile devices. Wireless technology allows two devices to communicate directly with each other in an active state without requiring a fixed infrastructure network. This temporary network is called Mobile Ad Hoc Network (MANET).

Mobile Ad Hoc Network (MANET) is a collection of multiple wireless nodes that can be set-up dynamically anywhere and anytime, without using an existing network infrastructure. It is also a temporary network formed by several mobile nodes without a central administration nor cable infrastructure. But implementing MANET in real life is still difficult due to various factors. Therefore, researcher use a Network Simulator to learn more about MANET.

This Final Project studied a MANET scheme generated by the node-movement and traffic-pattern files that already exist in the network simulator distribution. This study used NS-2 as a network simulator with the Destination Sequenced Distance Network (DSDV) proactive protocol as the routing protocol and the Nakagami transmission model.

Once the MANET scheme has been successfully established, functionality and performance tests will be performed through some predefined scenarios. The result of the test is a

performance graph of the Nakagami transmission model on the DSDV routing protocol in the MANET environment using NS-2 as measured by routing overhead, packet delivery ratio, and packet delivery delay from one node to another.

Keywords: MANET, Nakagami, Network Simulator, NS-2, DSDV.

RBTC

KATA PENGANTAR

Bismillahirrohmanirohim.

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

“Analisis Kinerja Destination Sequenced Distance Vector (DSDV) dengan Model Propagasi Nakagami pada Mobile Ad Hoc Network (MANET)”.

Terselesainya 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. Bapak (Alm), Ibu, Orang Saudara dan seluruh keluarga penulis atas dukungan dan doa yang diberikan selama ini.
3. Bapak Dr. Eng. Radityo Anggoro S.Kom., M.Sc. selaku dosen pembimbing 1 dari penulis yang telah memberikan bimbingan, dukungan, masukan, nasihat dan banyak arahan kepada penulis dalam menyelesaikan Tugas Akhir ini.
4. Ibu Henning Titi Ciptaningtyas, S.Kom., M.Kom. selaku dosen pembimbing 2 dari penulis yang telah memberikan bimbingan, dukungan, masukan, nasihat dan banyak arahan kepada penulis dalam menyelesaikan Tugas Akhir ini
5. Bapak Dr. Darlis Herumurti, S.Kom., M.Kom. selaku Ketua Jurusan Teknik Informatika ITS.

6. Ibu Cony, Ustadzah Endarty dan Ustadzah Santi. Terima kasih untuk segala bimbingan, arahan dan motivasi untuk penulis sehingga bisa menjadi pribadi yang lebih baik. Terima kasih karena tidak pernah mengijinkan penulis menyerah. Terima kasih karena telah percaya.
7. Anak-anak Rantau (Atika, Bian, Damas, Farah, Hendy, Kelly, Natasha, Uti dan Yaya), Poci Lover (Linggar dan Freeska), Solo Coret (Leli, Mei, Fina) selaku sahabat, teman main dan teman curhat dari penulis.
8. Teman-teman PH Berkarya HMTc ITS 2014-2015 dan staff maupun staff ahli PSDM Berkarya yang telah memberikan banyak pelajaran, pengalaman dan ilmu.
9. Ima dan Endah yang telah menemani tahun terakhir penulis menjalani masa perkuliahan dan mencoba hal-hal baru yang sebelumnya tidak pernah terpikirkan oleh penulis.
10. Teman-teman TC 2012 yang telah memberikan warna warni kehidupan kampus selama penulis menjalani perkuliahan.
11. Pihak-pihak yang tidak dapat disebutkan satu per satu yang telah membantu penulis dalam menyelesaikan Tugas Akhir ini.

Penulis menyadari bahwa laporan Tugas Akhir ini masih jauh dari kata sempurna. Oleh karena itu dengan segala kerendahan hati mengharapkan kritik dan saran dari pembaca untuk perbaikan penulis ke depannya. Penulis berharap laporan Tugas Akhir ini dapat berguna bagi pembaca secara umum. Semoga Allah SWT. memberkati dan membalas semua kebaikan yang telah dilakukan

Surabaya, Januari 2018

Maharani Wahyu Siwi

DAFTAR ISI

LEMBAR PENGESAHAN	vii
Abstrak.....	ix
Abstract.....	xi
KATA PENGANTAR	xiii
DAFTAR ISI.....	xv
DAFTAR GAMBAR	xvii
1 BAB I PENDAHULUAN.....	1
1.1. Latar Belakang	1
1.2. Rumusan Permasalahan	2
1.3. Batasan Permasalahan	2
1.4. Tujuan dan Manfaat	3
1.5. Metodologi	3
1.6. Sistematika Penulisan	5
2 BAB II TINJAUAN PUSTAKA.....	7
2.1. Destination Sequenced Distance Vector (DSDV).....	7
2.2. Mobile Ad Hoc Network (MANET).....	8
2.3. Model Transmisi Nakagami.....	9
2.4. Network Simulator 2 (NS-2).....	11
2.5. Generator File Node-Movement dan Traffic-Connection Pattern	11
2.5.1. <i>File Node-Movement (Mobility Generator)</i>	11
2.5.2. <i>File Traffic-Connection Pattern</i>	14
2.6. NS-2 Trace File.....	15
2.7. Awk.....	16
3 BAB III PERANCANGAN SISTEM.....	19
3.1. Deskripsi Umum	19
3.2. Perancangan Skenario	19
3.2.1. Skenario <i>Node-Movement (Mobility Generation)</i>	20
3.2.2. <i>Traffic-Connection Pattern Generation</i>	20
3.3. Perancangan Simulasi pada NS-2	21
3.4. Perancangan Metrik Analisis	22

3.4.1.	<i>Packet Delivery Ratio (PDR)</i>	22
3.4.2.	<i>End-to-End Delay (E2D)</i>	23
3.4.3.	<i>Routing Overhead (RO)</i>	23
4 BAB IV	IMPLEMENTASI	25
4.1.	Lingkungan Pembangunan Perangkat Lunak	25
4.1.1.	Lingkungan Perangkat Lunak	25
4.1.2.	Lingkungan Perangkat Keras	25
4.2.	Implementasi Skenario	25
4.2.1.	Skenario <i>File Node-Movement (Mobility Generation)</i>	26
4.2.2.	<i>File Traffic-Connection Pattern Generation</i>	29
4.3.	Implementasi Simulasi pada NS-2	31
4.4.	Implementasi Metrik Analisis	36
4.4.1.	<i>Packet Delivery Ratio (PDR)</i>	36
4.4.2.	<i>End-to-End Delay (E2D)</i>	38
4.4.3.	<i>Routing Overhead (RO)</i>	40
5 BAB V	PENGUJIAN DAN EVALUASI	42
5.1.	Lingkungan Pengujian	42
5.2.	Kriteria Pengujian	42
5.3.	Analisis Packet Delivery Ratio (PDR)	43
5.4.	Analisis End-to-End Delay (E2D)	45
5.5.	Analisis Routing Overhead (RO)	47
6 BAB VI	PENUTUP	50
6.1.	Kesimpulan	50
6.2.	Saran	51
DAFTAR PUSTAKA	52
7 LAMPIRAN	54
BIODATA PENULIS	77

DAFTAR GAMBAR

Gambar 2.1 PDF Kanal Nakagami-m-m untuk Parameter- m yang Berbeda [8].....	10
Gambar 2.2 Koneksi CBR pada 'cbra3'	15
Gambar 4.1 Posisi <i>Node</i> dalam X, Y dan Z	27
Gambar 4.2 Perpindahan/Pergerakan <i>Node</i>	28
Gambar 4.3 Pembuatan GOD untuk Setiap <i>Node</i>	28
Gambar 4.4 <i>Access Point</i>	29
Gambar 4.5 <i>Output</i> cbrtest.txt.....	30
Gambar 4.6 Konfigurasi awal parameter NS-2.....	32
Gambar 4.7 Konfigurasi <i>Trace File</i> dan Pergerakan <i>Node</i> pada NS-2	33
Gambar 4.8 Konfigurasi pengiriman paket data NS-2.....	35
Gambar 4.9 Perintah Eksekusi Model Propagasi Nakagami	36
Gambar 4.10 <i>Pseudeuocode</i> PDR.....	37
Gambar 4.11 <i>Pseudeuocode</i> E2D	39
Gambar 4.12 <i>Pseudeuocode</i> RO	40
Gambar 5.1 Grafik PDR Skenario <i>node-movement</i> Nakagami...	44
Gambar 5.2 Grafik PDR Skenario <i>Node Movement</i> Nakagami & <i>TwoRayGround</i>	45
Gambar 5.3 Grafik E2D Skenario <i>node-movement</i> Nakagami ...	46
Gambar 5.4 Grafik E2D Skenario <i>node-movement</i> Nakagami & <i>TwoRayGround</i>	46
Gambar 5.5 Grafik RO Skenario <i>node-movement</i> Nakagami	47
Gambar 5.6 Grafik RO Skenario <i>node-movement</i> Nakagami & <i>TwoRayGround</i>	48
Gambar 7.1 Posisi <i>node</i> dari potongan Skenario	57
Gambar 7.2 Pembuatan GOD setiap <i>node</i> dari potongan Skenario	61
Gambar 7.3 Pergerakan setiap <i>node</i> dari potongan Skenario	63
Gambar 7.4 iInformasi pada GOD dari potongan Skenario.....	64
Gambar 7.5 Koneksi yang digunakan pada cbrtest.txt.....	65
Gambar 7.6 <i>File</i> .tcl untuk Protokol <i>Routing</i> DSDV	69
Gambar 7.7 Implementasi <i>Packet Delivery Ratio</i>	70

Gambar 7.8 Implementasi <i>Routing Overhead</i>	70
Gambar 7.9 Implementasi <i>End-to-End Delay</i>	72
Gambar 7.10 Perintah instalasi dependensi NS-2	72
Gambar 7.11 Proses pengubahan ls.h.....	73
Gambar 7.12 Screenshot proses pengubahan ls.h.....	73
Gambar 7.13 Proses pengubahan <i>line of code</i> ls.h	73
Gambar 7.14 File .bashrc.....	74
Gambar 7.15 Perintah pengecekan NS-2.....	75

RBTC

DAFTAR TABEL

Tabel 2.1 Keterangan pada <i>Command Line</i> 'setdest'.....	12
Tabel 2.2 Keterangan <i>Command Line file</i> cbrgn.tcl	14
Tabel 3.1 Parameter Skenario <i>Node-Movement</i>	20
Tabel 3.2 Parameter <i>Traffic-Connection Pattern</i>	21
Tabel 3.3 Parameter Simulasi pada NS-2	21
Tabel 5.1 Spesifikasi Komputer yang Digunakan	42
Tabel 5.2 Kriteria Pengujian	43
Tabel 5.3 PDR Skenario <i>Node-Movement</i>	43
Tabel 5.4 E2D Skenario <i>Node-Movement</i>	45
Tabel 5.5 RO Skenario <i>Node-Movement</i>	47

[Halaman ini sengaja dikosongkan]

RBTC

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

Seiring dengan perkembangan zaman, teknologi informasi berkembang dengan pesatnya dan kebutuhan masyarakat akan komunikasi dan mengakses informasi pun semakin mudah. Perangkat *mobile* seperti *notebook*, *handphone*, tablet dan lain lain mulai berkembang adanya teknologi nirkabel (*wireless*) saat ini yang menjadi indikator kemajuan peradaban manusia memungkinkan perangkat komunikasi dapat berkomunikasi secara langsung dengan perangkat lainnya dalam posisi bergerak dan tanpa adanya jaringan infrastruktur yang tetap. Teknologi jaringan nirkabel (*wireless*) dapat membuat beberapa perangkat *mobile* tersebut yang berada di dalam suatu area dimana fasilitas nirkabel tersebut bisa saling terkoneksi dan saling menjangkau akan sangat mungkin membentuk sebuah jaringan yang bersifat sementara dan jaringan semacam ini disebut sebagai *Mobile Ad Hoc Network* (MANET) [1].

Jaringan *Mobile Ad Hoc Network* (MANET) merupakan suatu jaringan yang terorganisir secara mandiri tanpa adanya dukungan infrastruktur. Dalam MANET, setiap *node* bergerak secara bebas, sehingga jaringan dapat mengalami perubahan topologi dengan cepat. Karena *node* dalam MANET memiliki jarak propagasi yang terbatas, beberapa *node* tidak bisa berkomunikasi secara langsung dengan *node* lainnya. Pada MANET, jalur *routing* mengandung beberapa hop dan setiap *node* berfungsi sebagai router untuk menentukan ke arah mana tujuan atau rute yang akan mereka pilih. Dalam menentukan setiap jalur *routing* pada MANET terdapat tiga jenis protokol *routing* yang diklasifikasikan menjadi tiga, diantaranya protokol *routing*

proactive, reactive dan *hybrid*. Pada protokol *routing proactive* akan terus mempelajari perubahan topologi secara *real-time* melalui *node* jaringan tetangganya. Oleh karena itu, setiap ada permintaan rute dari *node* sumber ke *node* tujuan, informasi *routing* tersebut sudah tersedia pada *node* sumber. Seiring dengan perubahan topologi jaringan, akan dapat terjadinya peningkatan keseluruhan biaya pemeliharaan jaringan.

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 protokol *routing proactive* yaitu *Destination Sequenced Distance Vector* (DSDV) yang menggunakan model propagasi Nakagami yang ada pada NS-2.

Hasil yang diharapkan dari Tugas Akhir ini adalah suatu hasil analisa studi kinerja model propagasi Nakagami pada protokol *routing* DSDV di lingkungan MANET dengan menggunakan aplikasi. Performa protokol *routing* tersebut diukur berdasarkan *Routing Overhead*, *Packet Delivery Ratio*, dan *End-to-End 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* DSDV pada MANET?
2. Bagaimana hasil analisa studi kinerja model propagasi Nakagami pada protokol *routing* DSDV 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 *Destination Sequenced Distance Vector* (DSDV).
3. Lingkungan jaringan digunakan untuk uji coba adalah *Mobile Ad Hoc Network* (MANET).
4. Model propagasi yang akan dianalisa dalam Tugas Akhir ini adalah Nakagami.

1.4. Tujuan dan Manfaat

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

Tugas akhir ini diharapkan dapat menghasilkan hasil analisa kerja model propagasi Nakagami pada protokol DSDV yang efisien dengan parameter *Packet Delivery Ratio* (PDR), *Routing Overhead* (RO), dan *End-to-End Delay* (E2D).

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

Proposal Tugas Akhir ini merupakan pendahuluan dalam menyelesaikan Tugas Akhir. Proposal ini selain berisi ringkasan tugas akhir juga meliputi latar belakang dari masalah, rumusan masalah yang akan diselesaikan, batasan masalah, tujuan dan manfaat dalam tugas akhir yang dibuat, tinjauan pustaka dalam mengerjakan tugas akhir serta metodologi yang berisi tahapan-tahapan dalam menyusun tugas akhir.

2. Studi Literatur

Studi literatur yang dilakukan dengan pengumpulan informasi mengenai apa saja yang bisa dijadikan referensi dalam pengerjaan Tugas Akhir. Mengumpulkan informasi dan studi

literatur mengenai protokol DSDV, model propagasi nakagami, dan juga tentang MANET. Informasi didapatkan dari buku, paper, journal dan materi-materi kuliah yang berhubungan dengan topik Tugas Akhir.

3. Implementasi Protokol *Routing*

Implementasi yang akan dilakukan yaitu perancangan sistem berdasar studi literatur dan pengumpulan informasi yang telah dilakukan. Tahap ini merupakan suatu bentuk awal aplikasi yang akan diimplementasi didefinisikan. Yang dibuat adalah merupakan bentuk prototype sistem atau simulasi dari protokol *routing* DSDV yang menggunakan propagasi Nakagami pada MANET.

4. Pengujian dan Evaluasi

Pengujian dilakukan secara bertahap dan dari setiap tahapan akan dilakukan evaluasi. Pengujian akan dilakukan dengan melihat kesesuaian hasil dengan perencanaan. Dan juga dilakukan evaluasi, kemungkinan masalah yang akan timbul dan solusi untuk memperbaiki masalah yang timbul.

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 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
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 dilakukan pada tahap perancangan. Penjelasan berupa implementasi skenario mobilitas *node-node* pada jaringan *wireless* yang tidak mempunyai *router* tetap yang dibuat menggunakan *file node-movement* dan *traffic-pattern* yang ada pada *network simulator*, konfigurasi sistem dan skrip analisis yang digunakan untuk menguji performa protokol *routing*.
- Bab V. Pengujian dan Evaluasi
Bab ini menjelaskan tahap pengujian sistem dan performa dalam skenario mobilitas *ad hoc* yang dibuat oleh distribusi *mobility* dalam *network simulator*. Pengujian nantinya akan melakukan *running* program DSDV dengan model propagasi Nakagami. Hasil yang di dapat merupakan sebuah metrik nilai parameter *Packet Delivery Ratio* (PDR), *End to End Delay*

(E2D) dan *Routing Protocol* (RO) yang digambarkan ke dalam grafik dan bisa dilihat nilai performa masing-masing parameter.

- Bab VI. Penutup

Bab ini berisi kesimpulan dari hasil pengujian yang dilakukan terhadap rumusan masalah yang ada serta saran untuk pengembangan selanjutnya.

RBTC

BAB II TINJAUAN PUSTAKA

Bab ini membahas mengenai dasar-dasar teori yang berkaitan dengan pengimplementasian perangkat lunak. Penjelasan ini bertujuan untuk memberikan gambaran atau definisi secara umum terhadap alat, protokol *routing* serta definisi yang digunakan dalam pembuatan Tugas Akhir.

2.1. *Destination Sequenced Distance Vector (DSDV)*

Destination Sequenced Distance Vector adalah skema *routing* berbasis tabel untuk *mobile ad hoc network* berdasarkan algoritma Bellman-Ford yang dikembangkan oleh C. Perkins dan P. Bhagwat pada tahun 1994. DSDV merupakan salah satu algoritma *routing protocol* proaktif untuk mengatasi *Routing Loop*. Dalam metode DSDV, setiap *node* yang berada dalam jaringan akan mengirimkan paket *routing update* ke seluruh *node* tetangganya secara periodik. Setiap *node* menyimpan tabel *routing* yang mengandung informasi yang dibutuhkan untuk sampai ke *node* tujuan, setiap entri tabel memiliki nomor urut yang akan bertambah setiap kali sebuah *node* mengirim update pesan. Terdapat dua tabel *routing* yang disimpan oleh setiap *node* DSDV, yaitu tabel paket *forwarding* dan tabel informasi paket *routing* tambahan. Informasi *routing* yang dikirim antara lain berisi nomor urut baru, alamat *node* tujuan, jumlah hop ke *node* tujuan, dan nomor urut dari tujuan. Saat tabel *routing* dalam satu *node* telah diupdate, maka akan dipilih rute untuk mencapai *node* tujuan dengan mempertimbangkan nomor urut dari setiap entri tabel dan atau nilai metriknya. [2] [3]

Penelitian ini melakukan simulasi menggunakan aplikasi *Network Simulator 2 (NS-2)* untuk menganalisis kinerja DSDV pada *mobile ad hoc network*. Parameter kinerja yang dipakai untuk menganalisis hasil simulasi ialah *Packet Delivery Ratio*, *Routing Overhead*, dan *End to End Delay*. Simulasi dibagi menjadi beberapa skenario untuk mengetahui pengaruh mobilitas *node* dan trafik terhadap kinerja DSDV. Skenarionya antara lain, selang

kecepatan, *pause time*, jumlah *node*, luas area simulasi, ukuran paket data, jumlah koneksi maksimum, dan *packet rate*. [4]

2.2. *Mobile Ad Hoc Network (MANET)*

Mobile Ad Hoc Network (MANET) adalah kumpulan dari beberapa *wireless node* yang dapat di *set-up* secara dinamis dimana saja dan kapan saja, tanpa menggunakan infrastruktur jaringan yang ada. MANET juga merupakan jaringan sementara yang dibentuk oleh beberapa *mobile node* tanpa adanya pusat administrasi dan infrastruktur kabel. [5] Pada MANET, *mobile host* yang terhubung dengan *wireless* dapat bergerak kesegala arah 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.

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 Babel, *Intrazone Routing Protocol (IARP)*,

Destination Sequenced Distance Vector (DSDV) dan *Optimized Link State Routing (OLSR)*.

b. *Reactive Routing*

Tipe ini akan mencari rute (*on demand*) dengan cara membanjiri jaringan dengan paket *router 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)*. [6]

2.3. Model Transmisi Nakagami

Model Nakagami-m-m bersifat lebih umum dan dapat diterapkan untuk berbagai kondisi fading, tergantung pada parameter-m yang digunakan. Kanal Nakagami-m-m memiliki *probability density function (PDF)* yang dinyatakan sebagai persamaan (1) [7].

$$P_R(R) = \frac{2m^m R^{2m-1}}{\Gamma(m)\Omega^m} \exp\left(-\frac{mR^2}{\Omega}\right), \quad R \geq 0 \quad (1)$$

dimana :

m = parameter fading, $m \geq 0.5$ sampai ∞ (integer positif)

R = amplitudo fading

Γ = fungsi gamma

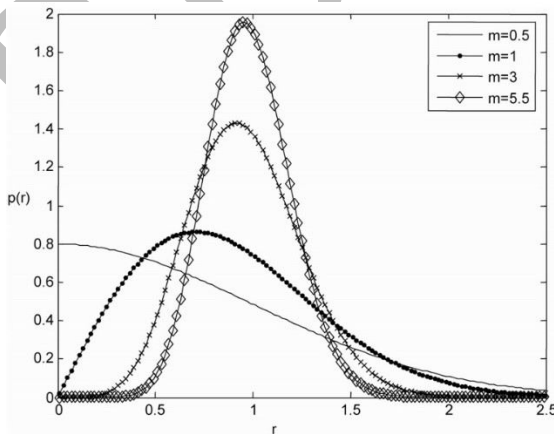
$\Omega = E [R^2]$

Pengaruh parameter- m pada kanal Nakagami- m - m :

1. Apabila $m \leq 1$, maka persamaan (2) menjadi *probabilitas density function* dari kanal fading Rayleigh.
2. Apabila $m > 1$, merujuk ke kanal fading Rician. Dimana kanal fading Rician memiliki faktor K yang memiliki pengaruh disini. Sehingga nilai m adalah sebagai persamaan (2). [8]

$$m = \frac{(1 + k)^2}{1 + 2k}, \quad k \geq 0 \quad (2)$$

Pengaruh dari pada parameter- m dapat dilihat pada grafik *Probability Density Function* (PDF) kanal Nakagami- m - m seperti pada Gambar 2.1.



Gambar 2.1 PDF Kanal Nakagami- m - m untuk Parameter- m yang Berbeda [8]

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

Network Simulator 2 (NS-2) merupakan alat simulasi jaringan yang bersifat *open source* yang banyak digunakan dalam mempelajari struktur dinamik dari jaringan komunikasi. Simulator ini ditargetkan pada penelitian jaringan dan memberikan dukungan yang baik untuk simulasi *routing*, protokol *multicast* dan protokol IP, seperti UDP, TCP, RTP, jaringan nirkabel dan jaringan satelit. Beberapa keuntungan menggunakan *network simulator* sebagai perangkat lunak simulasi yaitu *network simulator* dilengkapi dengan *tools* validasi, pembuatan simulasi dengan menggunakan *network simulator* jauh lebih mudah daripada menggunakan *software developer* seperti Delphi atau C++, *network simulator* bersifat *open source* di bawah GPL (Gnu Public License) dan dapat digunakan pada sistem operasi Windows dan sistem operasi Linux. [9] [10]

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 file node-movement* dan *traffic-connection pattern* menggunakan protokol DSDV. NS-2 dijalankan pada sistem operasi Linux dan proses instalasinya akan disajikan pada bagian Lampiran.

2.5. *Generator File Node-Movement dan Traffic-Connection Pattern*

2.5.1. *File Node-Movement (Mobility Generator)*

Tools yang disebut ‘setdest’ dikembangkan oleh CMU (Carnegie Mellon University) untuk menghasilkan *random movement* dari *node* dalam jaringan nirkabel. *Node movement* dihasilkan dengan kecepatan gerak yang spesifik menuju lokasi acak atau lokasi spesifik yang berada dalam kawasan yang telah ditentukan. Ketika *node* tiba ke lokasi pergerakan, *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’ sebelum menjalankan program simulasi. Format *Command Line* ‘setdest’ ditunjukkan pada *command line* dibawah ini dan keterangannya ditunjukkan pada Tabel 2.1.

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

Tabel 2.1 Keterangan pada *Command Line* ‘setdest’

Parameter	Keterangan
-v version	Versi ‘setdest’ simulator yang digunakan
-n num	Jumlah <i>node</i> dalam skenario
-p pausetime	Durasi ketika sebuah <i>node</i> tetap diam setelah tiba di lokasi pergerakan. Jika nilai ini diatur ke 0, maka <i>node</i> tidak akan berhenti ketika tiba di lokasi pergerakan dan akan terus bergerak
Parameter	Keterangan
-M maxspeed	Kecepatan maksimum sebuah <i>node</i> . <i>Node</i> akan bergerak pada kecepatan acak dalam rentang [0, amxspeed]
-t simtime	Waktu simulasi
-x max x	Panjang maksimum area simulasi
-y max y	Lebar maksimum area simulasi

Command Line ‘setdest’ menghasilkan *file output* yang berisi jumlah *node* dan mobilitas yang akan digunakan dalam *file* Tcl selama simulasi. *File output*, selain mengandung skrip pergerakan, juga mengandung beberapa statistik lain tentang perubahan *link* dan rute.

Untuk membuat skenario *node-movement* yang terdiri dari 50 *node*, bergerak dengan kecepatan maksimum 10.0 m/s dengan jeda rata-rata antar gerakan sebesar 10 detik, simulasi akan berhenti

setelah 200 detik dengan batas topologi yang diartikan sebagai 500 x 500 meter², *command line*-nya terlihat seperti dibawah ini

```
./setdest -v 1 -n 50 -p 10.0 -M 10.0 -t 200 -x 500
-y 500 > scena3
```

File output ditulis ke "stdout" secara *default*. Di sini *output* disimpan ke dalam *file* "scena3". *File* dimulai dengan posisi awal *node* dan berlanjut menetapkan *node-movement* seperti

```
$ns_ at 10.000000000000 "$node_(27) setdest
354.995604774236 171.557602282609 3.699855750792"
```

Command line diatas dari 'scena3' mendefinisikan bahwa *node* (27) pada detik ke 10.0 mulai bergerak ke arah tujuan (354.99, 171.56) dengan kecepatan 3.69 m/s. *Command line* ini dapat digunakan untuk mengubah arah dan kecepatan gerak dari *mobile node*. Arahan untuk *General Operations Director 1* (GOD) yang ada juga di *file node-movement*. Objek "GOD" digunakan untuk menyimpan informasi global tentang keadaan dari lingkungan jaringan dan *node* di sekitarnya. Namun isi dari *file* "GOD" tidak boleh diketahui oleh setiap bagian dalam simulasi.

Dalam simulasi di sini objek "GOD" hanya 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, karena akan cukup memakan waktu. Namun "GOD" menghitung *hop* di akhir simulasi. Informasi yang dimuat ke dalam objek "GOD" dari pola pergerakan *file* terdapat pada baris perintah di bawah ini

```
$ns_ at 11.558983766827 "$god_ set-dist 24 43 2"
```

Ini berarti bahwa jarak terpendek antara *node* 24 dan *node* 43 berubah menjadi 2 *hop* di waktu 11.56 detik. Program 'setdest' menghasilkan *file node-movement* menggunakan algoritma *random way point*. Perintah-perintah yang termasuk dalam

program utama untuk memuat *file-file* ini dalam objek “GOD” [11].

2.5.2. File Traffic-Connection Pattern

Random traffic connection pada TCP dan CBR bisa dikonfigurasi antara mobilitas *node* menggunakan skrip *traffic-scenario pattern generator*. Untuk menghasilkan alur *traffic* yang acak, dapat digunakan skrip Tcl yang disebut "cbrgen". Skrip ini membantu untuk menghasilkan *traffic load* atau beban trafik. Beban dapat berupa TCP atau CBR. Skrip ini disimpan di dalam *file* 'CMU-scen-gen' yang terletak di dalam direktori ~ns/indep-utills/cmu-scen-gen. Program "cbrgen.tcl" digunakan sesuai dengan *command line* dibawah ini 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
```

Tabel 2.2 Keterangan Command Line file cbrgn.tcl

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

Pada CBR, *data rate* dapat dihitung sebagai berikut:

Data Rate (bits/second) = 512 bytes*8 bits/bytes * rate (packets/second defined in "cbrgen")

```
ns cbrgen.tcl -type cbr -nn 50 -seed 1.0 -mc 20 -
rate 4.0 > cbra3
```

Command line diatas 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.

Dari *file* cbra3 (kemana *output* dari generator akan diarahkan) sehingga menghasilkan salah satu koneksi CBR yang terlihat seperti pada Gambar 2.2.

```
#
# 2 connecting to 3 at time 82.557023746220864
#
set udp_(0) [new Agent/UDP]
$nns_ attach-agent $node_(2) $udp_(0)
set null_(0) [new Agent/Null]
$nns_ 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)
$nns_ connect $udp_(0) $null_(0)
$nns_ at 82.557023746220864 "$cbr_(0) start"
```

Gambar 2.2 Koneksi CBR pada 'cbra3'

Agent yang digunakan ialah 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* cbr-20-test [11].

2.6. NS-2 Trace File

NS-2 *Trace File* merupakan *output* hasil *running* NS-2 yang berekstensi .tr. *Trace File* berisi *log* tentang semua jenis pengiriman dan penerimaan paket yang terjadi selama simulasi. Di dalam *Trace File* tercatat berbagai jenis paket sesuai jenis protokol *routing* yang digunakan. Tiap baris *log* ini dianalisis untuk

mendapatkan performa protokol. Contoh pengiriman data paket pada NS-2 *Trace File* dapat dilihat dibawah ini

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

Huruf “s” di kolom pertama menandakan pengiriman paket (*send*), kolom kedua berisi waktu pengiriman paket pada detik ke 26, kolom ketiga merupakan *node* tempat *event* terjadi yaitu pada *node* 1, kolom ke empat berisi AGT yang menandakan pengiriman paket data, kolom ke lima merupakan tempat terjadinya *event* spesial semisal *collision*, kolom ke 6 merupakan id unik paket, kolom ke tujuh berisi tipe paket yang dikirimkan yaitu cbr, kolom ke delapan merupakan ukuran paket dalam *byte* yaitu 64.

Untuk penerimaan data paket seperti berikut tidak jauh beda dengan pengiriman data paket, pembedanya yaitu kolom pertama dengan huruf inisial “r” yang menandakan paket diterima dan format selanjutnya sama persis dengan paket pengiriman. Contoh penerimaan data paket pada NS-2 *Trace File* dapat dilihat dibawah ini.

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

2.7. Awk

Awk adalah sebuah pemrograman seperti pada *shell* atau C yang memiliki karakteristik yaitu sebagai *tools* 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. 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 [12].

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 dikosongka]

RBTC

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 mengenai deskripsi umum sistem, perancangan skenario, alur, serta gambaran implementasi sistem yang diterapkan pada *Network Simulator 2 (NS-2)*.

3.1. Deskripsi Umum

Yang dilakukan pada Tugas Akhir ini adalah analisis performa model tranmisi Nakagami pada MANET. Dalam pembuatan skenario MANET menggunakan *Mobility Generator* yang bersifat *Random Way Point* dengan cara men-generate *file node-movement (mobility generation)* dan membuat koneksi antar *node* menggunakan *file traffic-connection pattern* yang telah ada pada *Network Simulator-2 (NS-2)*. Kemudian untuk simulasi skenario yang dihasilkan oleh *mobility generator* akan dijalankan pada NS-2 menggunakan protokol *routing DSDV* pada sistem operasi Linux.

Kecepatan maksimum pergerakan dari satu *node* ke *node* lainnya dibuat bervariasi yaitu 5, 10 dan 15 m/s pada tiap skenarionya. Hasil proses uji coba dari tiap skenario akan menghasilkan sebuah *trace file* yang nantinya akan dilakukan analisis perhitungan metrik *Packet Delivery Ratio (PDR)*, *End-to-End Delay (E2D)*, dan *Routing Overhead (RO)*. Dari hasil metrik tersebut dianalisis performa model.

3.2. Perancangan Skenario

Perancangan skenario uji coba mobilitas MANET dimulai dengan melakukan pembuatan skenario pada *mobility generation* yang bersifat *Random Way Point* kemudian membuat koneksi dengan menggunakan *file traffic-connection* yang sudah ada pada NS-2. Pembuatan skenario untuk melihat pergerakan *node* dibedakan berdasarkan 3 kecepatan maksimum yaitu 5, 10 dan 15

m/s dan untuk koneksinya hanya digunakan 2 *node* yaitu, *node* pengirim dan *node* penerima paket. Penjelasan untuk perancangan skenario pada *mobility generator* dan pembuatan koneksi antar *node*-nya adalah sebagai berikut:

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

Skenario *mobility generation* dibuat dengan men-generate *file node-movement* atau 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 <i>Node</i>	50
2	Waktu Simulasi	100 detik
3	Area	510 m x 510 m
3	Kecepatan Maksimal	- 5 m/s - 10 m/s - 15 m/s
5	Sumber <i>Traffic</i>	CBR
6	Waktu Jeda (dalam detik)	10
7	Ukuran Paket	512 bytes
8	<i>Rate</i> Paket	1 paket per detik
9	Jumlah maksimal koneksi	1
10	Model mobilitas yang digunakan	<i>Random Way Point</i>

3.2.2. *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 skenario selama simulasi pada NS-2.

Tabel 3.2 Parameter *Traffic-Connection Pattern*

No.	Parameter	Spesifikasi
1	-type cbr tcp	CBR
2	-nn <i>nodes</i>	10
3	-s seed	1.0
4	-mc connections	1
5	-rate rate	1
6	<i>Agent</i>	UDP

3.3. Perancangan Simulasi pada NS-2

Untuk melakukan konfigurasi MANET pada perancangan kode NS-2 dilakukan dengan menggabungkan skenario mobilitas dan *traffic-connection* dengan skrip TCL yang diberikan parameter-parameter untuk membangun percobaan simulasi MANET pada NS-2. Berikut parameter simulasi perancangan sistem MANET yang dapat digunakan dapat dilihat pada Tabel 3.3.

Tabel 3.3 Parameter Simulasi pada NS-2

No.	Parameter	Spesifikasi
1	Network simulator	NS- 2.35
2	<i>Routing Protocol</i>	DSDV
3	Waktu simulasi	100 detik
4	Waktu Pengiriman Paket Data	0 – 100 detik
5	Area simulasi	510 m x 510 m
6	Banyak <i>node</i>	50
7	Radius transmisi	100 m
8	Tipe koneksi	UDP
9	Tipe data	Constant Bit Rate (CBR)

No.	Parameter	Spesifikasi
10	Source / Destination	Statik (<i>Node 1 / Node 2</i>)
11	Kecepatan generasi paket	1 paket per detik
12	Ukuran paket data	512 bytes
13	Protokol MAC	IEEE 802.11
14	Mode Transmisi	Nakagami
15	Tipe Antena	OmniAntenna
16	Tipe Interface Queue	Droptail/PriQueue
17	Tipe Peta	MANET (<i>random way point</i>)
18	Tipe kanal	Wireless channel
19	Tipe <i>trace</i>	Old Wireless Format <i>Trace</i>

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 untuk masing-masing metrik analisis adalah sebagai berikut:

3.4.1. *Packet Delivery Ratio* (PDR)

Dengan membandingkan antara paket yang dikirim dengan paket yang diterima akan diperoleh nilai PDR. (*Packet Delivery Ratio*) PDR dihitung dengan menggunakan persamaan (3), dimana *received* adalah banyaknya paket data yang diterima dan *sent* adalah banyaknya paket data yang dikirimkan. Nilai PDR dinyatakan dalam bentuk persentase.

$$PDR = \frac{received}{sent} \times 100$$

(3)

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

Cara menghitung End-to-End (E2D) adalah dengan menghitung rata-rata *delay* antara waktu paket diterima dan waktu paket dikirim. E2D dihitung dengan menggunakan persamaan (4), dimana $t_{received[i]}$ adalah waktu penerimaan paket, $t_{sent[i]}$ adalah waktu pengiriman paket, dimana i adalah urutan / id ke- i dan *sent* adalah banyaknya paket data yang dikirimkan.

$$E2D = \frac{\sum_{i \leq sent}^{i=0} t_{received[i]} - t_{sent[i]}}{sent} \quad (4)$$

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 jumlah paket *routing* yang ditransmisikan. Baris yang mengandung *routing overhead* pada *trace file* ditandai dengan paket yang bertipe *send* (s) / *forward* (f) dan terdapat *header* paket dari protokol DSDV.

[Halaman ini sengaja dikosongkan]

RBTC

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 analisis.

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 10.04 LTS 64 bit untuk lingkungan NS-2;
- *Network Simulator 2* versi 2.35.
- *Patch DSDV* versi 1.0

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) i3-2330M CPU @2.20GHz*;
- Media penyimpanan sebesar 500GB;
- RAM sebesar 4 GB 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 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'. Program 'setdest' pada NS-2 digunakan untuk menghasilkan *file node-movement* dengan menggunakan algoritma *Random Way Point*. *File skenario node-movement (mobility generation)* digunakan pada 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. Format *command line* 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]
```

Ketentuan-ketentuan yang diujicobakan pada skenarionya berturut-turut adalah versi 'setdest' simulator yaitu 1, jumlah *node* dalam skenario yaitu 50, waktu jeda (*pause time*) yaitu 10 detik, kecepatan maksimalnya yaitu skenario A sebesar 5 m/s, skenario B sebesar 10 m/s dan skenario C sebesar 15 m/s, waktu simulasi yaitu 100 detik, panjang dan lebar maksimal area simulasi yaitu 510 meter. Kemudian *file* mobilitas yang dihasilkan disimpan dalam direktori " ~ ns /indep-utils/CMU-scen-gen/setdest / ". Pada *script* dibawah dapat dilihat implementasi *command line* pada 'setdest' dengan berbagai kecepatan maksimal yang berbeda sesuai dan *node* sebanyak 50. Dan untuk setiap kecepatan maksimal tersebut dibuat 10 buah *file* untuk satu protokol *routing* dan satu model transmisi.

```

./setdest -v 1 -n 50 -p 10 -M 5 -t 100 -x 510 -y 510
> scena3.txt
./setdest -v 1 -n 50 -p 10 -M 10 -t 100 -x 510 -y
510 > scenb3.txt
./setdest -v 1 -n 50 -p 10 -M 15 -t 100 -x 510 -y
510 > scenc3.txt

```

Penempatan awal setiap *node* dalam sebuah area dan dinyatakan dalam sumbu X, Y dan Z ditunjukkan pada potongan *file* mobilitas scena3.txt hasil generate ‘setdest’ ada pada Gambar 4.1 berikut:

```

#
# nodes: 50, pause: 10.00, max speed: 5.00, max x:
510.00, max y: 510.00
#
$node_(0) set X_ 451.785953254783
$node_(0) set Y_ 126.212616074417
$node_(0) set Z_ 0.000000000000
$node_(1) set X_ 86.444329049070
$node_(1) set Y_ 211.568272257270
$node_(1) set Z_ 0.000000000000
$node_(2) set X_ 129.607413806567
$node_(2) set Y_ 161.706642991626
$node_(2) set Z_ 0.000000000000
$node_(3) set X_ 149.137357289570
$node_(3) set Y_ 304.073658859665

```

Gambar 4.1 Posisi *Node* dalam X, Y dan Z

Pada Gambar 4.2 menunjukkan pergerakan *node* tersebut selama waktu simulasi dijalankan untuk setiap *node* diberikan posisi awal dan berkelanjutan untuk menentukan pergerakan *node* berikutnya. Dari potongan dari scena3.txt pada Gambar 4.2, baris pertama mendefinisikan untuk *node* (0) pada detik ke 10 mulai bergerak ke arah tujuan (33.39, 112. 55) dengan kecepatan 2.11 m/s. Baris perintah ini dapat digunakan untuk mengubah arah dan kecepatan pada pergerakan mobilitas *node*.

```

$ns_ at 10.000000000000 "$node_(0) setdest
33.392695634552 112.547007091292 2.113849896630"
$ns_ at 10.000000000000 "$node_(1) setdest
489.712329995857 6.865637462598 3.922220837749"
$ns_ at 10.000000000000 "$node_(2) setdest
409.459129683091 55.089756748329 3.585334639239"
$ns_ at 10.000000000000 "$node_(3) setdest
180.629308850247 482.003026138206 4.422707437753"
$ns_ at 10.000000000000 "$node_(4) setdest
285.822893523416 486.211624697786 2.297733046098"
$ns_ at 10.000000000000 "$node_(5) setdest
468.696253529941 438.087795810369 1.352863243407"
$ns_ at 10.000000000000 "$node_(6) setdest
28.204259280229 334.754005518936 4.618745659408"
$ns_ at 10.000000000000 "$node_(7) setdest
456.078617557500 321.652292242198 2.116869301607"
$ns_ at 10.000000000000 "$node_(8) setdest
444.144080107093 178.023176174336 2.726062831241"
$ns_ at 10.000000000000 "$node_(9) setdest
74.259304752558 291.392042154806 3.323235037092"
$ns_ at 10.000000000000 "$node_(10) setdest
169.822967728408 304.162859196280 3.998979754231"

```

Gambar 4.2 Perpindahan/Pergerakan Node

Kemudian pada Gambar 4.3 terdapat potongan dari *scena3.txt* 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. Objek GOD tidak menghitung *array* ini selama simulasi berjalan karena bisa memakan waktu.

```

$god_ set-dist 0 1 2
$god_ set-dist 0 2 2
$god_ set-dist 0 3 2
$god_ set-dist 0 4 2
$god_ set-dist 0 5 2
$god_ set-dist 0 6 3

```

Gambar 4.3 Pembuatan GOD untuk Setiap Node

Lalu informasi yang dimuat ke dalam objek GOD dari *file node-movement* ditunjukkan pada Gambar 4.4. Pada baris pertama tertulis set-dist “15 19 1” yang artinya jalan terpendek *node* 15 ke *node* 19 adalah 1 hop. Pada baris pertama digunakan untuk memuat objek GOD dengan informasi yaitu jalan terpendek antara *node* 15 dan *node* 19 berubah menjadi 1 hop pada detik ke-10.042788185115.

```
$ns_ at 10.042788185115 "$god_ set-dist 15 19 1"
$ns_ at 10.134183422957 "$god_ set-dist 22 28 1"
$ns_ at 10.249586178225 "$god_ set-dist 4 26 1"
$ns_ at 10.431163522947 "$god_ set-dist 3 33 1"
$ns_ at 10.773426893360 "$god_ set-dist 10 34 1"
$ns_ at 10.783429117575 "$god_ set-dist 13 34 1"
$ns_ at 10.814885521465 "$god_ set-dist 8 16 1"
$ns_ at 10.826047682487 "$god_ set-dist 5 26 1"
$ns_ at 10.826047682487 "$god_ set-dist 23 26 2"
$ns_ at 10.826047682487 "$god_ set-dist 26 31 2"
$ns_ at 11.205153460654 "$god_ set-dist 45 46 1"
$ns_ at 11.208904416845 "$god_ set-dist 25 30 1"
$ns_ at 11.224078341276 "$god_ set-dist 18 32 1"
$ns_ at 11.233267390250 "$god_ set-dist 18 22 2"
$ns_ at 11.233267390250 "$god_ set-dist 22 32 1"
$ns_ at 11.233267390250 "$god_ set-dist 22 48 2"
$ns_ at 11.410358037743 "$god_ set-dist 34 46 1"
$ns_ at 11.422167101582 "$god_ set-dist 15 20 1"
$ns_ at 11.447460643615 "$god_ set-dist 10 33 2"
$ns_ at 11.447460643615 "$god_ set-dist 13 33 2"
```

Gambar 4.4 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-allinone-2.35/ns-2.35/indep-utils/cmu-scen-gen` dan disimpan dalam bentuk *file* `cbrgen.tcl`. *File* ini dapat digunakan untuk membuat *traffic-conneciton* CBR ataupun TCP pada jaringan pergerakan antar *node*. Pada saat menjalankan perintah pada *file traffic-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* dan misalkan pada koneksi CBR, *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] > traffic-file
```

Waktu awal untuk koneksi TCP/CBR secara acak yang dihasilkan dengan nilai maksimal ditetapkan pada 180,0 detik. Pada *command line* dibawah ini merupakan bentuk implementasi untuk menjalankan cbrgen.tcl untuk membuat *file* koneksi CBR diantara 10 *node*, memiliki maksimal 1 koneksi dengan nilai *seed* 1.0 dan jumlah paket per detik sebanyak 1 yang disimpan dalam cbrtest.txt yang nantinya akan digunakan pada saat simulasi NS-2.

```
ns cbrgen.tcl CBR -nn 10 -seed 1.0 -mc 1 -rate 1 >
cbrtest.txt
```

```
# nodes: 10, max conn: 1, send rate: 1, 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 4.5 Output cbrtest.txt

Kemudian Gambar 4.5 diatas menunjukkan *output*-nya yang disimpan dalam *cbrtest.txt* sehingga menghasilkan koneksi CBR dan menggunakan *Agent* UDP. Koneksi UDP di sini merupakan konfigurasi antara *node* ke-1 dan 2. Interval pengiriman paket data dilakukan setiap satu detik dengan besar paket data 512 byte dan maksimal pengiriman paket 10.000.

4.3. Implementasi Simulasi pada NS-2

Untuk mensimulasikan MANET dalam lingkungan NS-2, skrip *tcl* yang telah ada pada *pacth* protokol *routing* DSDV. 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 4.6 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*. Baris kedua merupakan tipe transmisi yang digunakan yaitu Nakagami. Tipe Mac yang digunakan adalah Mac 802.11. Pada baris tersebut 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 510 meter, jumlah maksimal paket dan *node* yaitu 50 *node*, besarnya *seed* yaitu 0.0, protokol *routing* yang digunakan yaitu DSDV, waktu simulasi diakhiri pada detik ke-100, *file traffic-connection* yang digunakan yaitu *cbrtest.txt* dan terakhir ialah *file* skenario *node-movement* (*mobility generation*) yang digunakan adalah *scena3.txt*.

```

set val(chan)    Channel/WirelessChannel;
set val(prop)    Propagation/Nakagami;
set val(netif)   Phy/WirelessPhy;
set val(mac)     Mac/802_11;
set val(ifq)     Queue/DropTail/PriQueue;
set val(ll)      LL;
set val(ant)     Antenna/OmniAntenna;
set opt(x)       510;
set opt(y)       510;
set val(ifqlen)  50;
set val(nn)      50;
set val(seed)    0.0;
set val(adhocRouting) DSDV;
set val(stop)    100
set val(cp)      "cbrtest.txt";
set val(sc)      "scena3.txt";

```

Gambar 4.6 Konfigurasi awal parameter NS-2

Pada *command line* dibawah ini 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
```

Skrip yang ditunjukkan pada Gambar 4.7 dibawah ini merupakan skrip untuk pengaturan variabel global. Pengaturan variable global diawali dengan *set ns* untuk pembuatan simulator baru. *Set tracefd* dan *set namtrace* merupakan pengaturan untuk nama *trace file* berekstensi .tr dan *file network* animator .nam akan dihasilkan dan disimpan.

```

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

# setup topography object
set topo [new Topography]

# create trace object for ns and nam
set tracefd [open dsdv_outnakaa3.tr w]
set namtrace [open dsdv_outnakaa3.nam w]

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

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

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

#global node setting
$ns_ node-config -adhocRouting $val(adhocRouting) \
                -llType $val(ll) \
                -macType $val(mac) \
                -ifqType $val(ifq) \
                -ifqLen $val(ifqlen) \
                -antType $val(ant) \
                -propType $val(prop) \
                -phyType $val(netif) \
                -channelType $val(chan) \
                -topoInstance $topo \
                -agentTrace ON \
                -routerTrace ON \
                -macTrace 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 4.7 Konfigurasi *Trace File* dan Pergerakan *Node* pada NS-

Pada *set tracefd* dapat dilakukan pengaturan untuk menghasilkan *trace file* sesuai dengan keinginan pengguna. Terdapat dua jenis format *trace file* yang disediakan yaitu *old trace format* dan *new trace format* 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 4.6 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*.

Gambar 4.8 dibawah ini 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 yang nantinya dihasilkan pada *file output .tr*. Pada potongan skrip tersebut, akan dipanggil *file skenario node-movement (mobility generation)* dan *traffic-connection pattern* kemudian pengiriman paket data dimulai pada detik ke-0 dan diberhentikan pada detik ke-100 seperti yang telah dikonfigurasi sebelumnya pada Gambar 4.6.

```

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

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

# Define node initial position in nam
for {set i 0} {$i < $val(nn)} {incr i} {

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

    $ns_ initial_node_pos $node_($i) 20
}

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

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

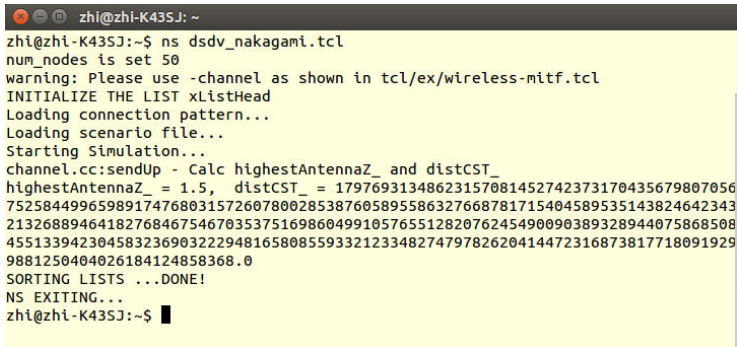
puts $tracefd "M 0.0 nn $val(nn) x $opt(x) y
$opt(y) rp $val(adhocRouting)"
puts $tracefd "M 0.0 sc $val(sc) cp $val(cp) seed
$val(seed)"
puts $tracefd "M 0.0 prop $val(prop) ant
$val(ant)"

puts "Starting Simulation..."
$ns_ run

```

Gambar 4.8 Konfigurasi pengiriman paket data NS-2

Contoh hasil *running*/eksekusi file .tcl pada propagasi Nakagami ditunjukkan pada Gambar 4.9 dibawah ini :



```

zhi@zhi-K43SJ:~$ ns dsdv_nakagami.tcl
num_nodes is set 50
warning: Please use -channel as shown in tcl/ex/wireless-mitf.tcl
INITIALIZE THE LIST xListHead
Loading connection pattern...
Loading scenario file...
Starting Simulation...
channel.cc:sendUp - Calc highestAntennaZ_ and distCST_
highestAntennaZ_ = 1.5, distCST_ = 17976931348623157081452742373170435679807056
75258449965989174768031572607800285387605895586327668781715404589535143824642343
21326889464182768467546703537516986049910576551282076245490090389328944075868508
4551339423045832369032294816580855933212334827479782620414472316873817718091929
9881250404026184124858368.0
SORTING LISTS ...DONE!
NS EXITING...
zhi@zhi-K43SJ:~$

```

Gambar 4.9 Perintah Eksekusi Model Propagasi Nakagami

4.4. Implementasi Metrik Analisis

Hasil menjalankan skenario MANET dalam NS-2 dalam bentuk *Trace File* berekstensi .tr dianalisis dengan 3 (tiga) metrik yaitu *Packet Delivery Ratio* (PDR), *End-to-End Delay* (E2D), dan *Routing Overhead* (RO). Implementasi dari tiap metrik menggunakan bahasa pemrograman 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.

Pseudeuocode PDR ditunjukkan pada **Gambar 4.10** dan implementasinya dapat dilihat pada Lampiran.

```

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

#count packet send
(if ($1 == "s" and $3 == "_1_" and $4 == "AGT" and
    $7 == "cbr" )
    sent +1;
)

#count packet receive
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 4.10 Pseudeuocode PDR

Contoh perintah untuk analisis PDR dari *trace file* model transmisi *Nakagami* dengan protokol DSDV dan kecepatan maksimal perpindahan *node* sebesar 5 m/s seperti dibawah ini

```
awk -f pdr.awk dsdv_outnakaa3.tr
```

Contoh *output* dari menjalankan skrip tersebut ditunjukkan dengan seperti ini

```
Transmitted packet(s) = 100
Received packet(s) = 100
Packet Delivery Ratio = 100 %
```

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

Perhitungan E2D dilakukan dengan menghitung selisih waktu paket data terkirim yang dilakukan oleh *node* 1 dan waktu paket data diterima oleh *node* 2 di dalam satu *trace file*. Pencatatan waktu paket terkirim pada kolom ke-2 dilakukan apabila pada baris *trace* yang bersangkutan mengandung semua kondisi yaitu kolom pertama mengandung huruf “s” yang menandakan *send packet*, kolom ke-3 menunjukkan *node* yang melakukan pengiriman adalah *node* 1, kolom ke-4 dan kolom ke-7 mengandung huruf “AGT” dan “cbr” yang menunjukkan pengiriman paket yang dilakukan adalah pengiriman paket data. Perhitungan waktu dan pencatatan ID serta jumlah paket diterima dilakukan apabila baris *trace* yang bersangkutan mengandung kondisi dimana yaitu kolom pertama mengandung huruf “r” yang menandakan *received packet*, kolom ke-3 menunjukkan *node* yang menerima paket adalah *node* 2, kolom ke-4 dan kolom ke-7 mengandung huruf “AGT” dan “cbr” yang menunjukkan penerimaan paket yang adalah penerimaan 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. *Pseudocode* E2D ditunjukkan pada Gambar 4.11 dan implementasinya dapat dilihat pada Lampiran.

```

ALGORITMA E2D(trace file)
//Input: trace file 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_rcv
    pkt_rcv[i] ← 0
  delay = avg_delay ← 0
  rcv ← 0
  rcv_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 rcv_id != $6 )
    rcv + 1
  rcv_id ← $6
    pkt_rcv[$6] ← $2;
  )

END (
  for i in pkt_rcv
    delay += pkt_rcv[i] - pkt_sent[i]
  avg_delay ← delay / rcv;
  print rcv
  print delay
  print avg_delay

```

Gambar 4.11 Pseudeucode E2D

Contoh perintah untuk analisis E2D dari *trace file* model transmisi *Nakagami* dengan protokol DSDV dan kecepatan maksimal perpindahan *node* sebesar 5 m/s seperti dibawah ini.

```
awk -f endtoend.awk dsdv_outnaka3.tr
```

Hasil output dari perintah diatas adalah ditunjukkan dibawah ini

```
Total Packet(s) Receive = 100
Total Delay = 0.58314 second
Average Packet Delivery Delay = 0.0058314 second
```

4.4.3. Routing Overhead (RO)

Implementasi perhitungan metrik *Routing overhead* DSDV 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 “DSDV” yang berarti paket *routing* DSDV. seperti pada **Gambar 4.12**. Implementasinya dapat dilihat pada Lampiran.

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

Gambar 4.12 Pseudeucode RO

Contoh perintah untuk analisis RO dari *trace file* model transmisi *Nakagami* dengan protokol DSDV dan kecepatan maksimal perpindahan *node* sebesar 5 m/s seperti ini

```
awk -f ro.awk dsdv_outnakaa3.tr
```

Contoh *output* dari menjalankan skrip di atas seperti pada script dibawah ini

```
total no of routing packets      1220
```

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 sebuah laptop yang telah terpasang sistem operasi yaitu Windows dan Linux. Spesifikasi komputer yang digunakan ditunjukkan pada Tabel 5.1.

Tabel 5.1 Spesifikasi Komputer yang Digunakan

Komponen	Spesifikasi
CPU	Intel(R) Core(TM) i3-2330M CPU @2.20GHz
Sistem Operasi	Linux Ubuntu 10.04 LTS 64-bit (NS-2, DSDV, <i>Mobility Generation</i> , <i>Traffic-Connection Generation</i> , Nakagami) & Windows 10 Home
Memori	GB DDR3
Media Penyimpanan	500 GB

5.2. Kriteria Pengujian

Pengujian pada skenario yang dihasilkan oleh *mobility generator default* dari NS-2 menggunakan beberapa kriteria. Kriteria tersebut diantaranya adalah skenario yang digunakan, kecepatan maksimal perpindahan node, banyaknya percobaan yang akan dilakukan, jarak antar node, protokol *routing* yang digunakan dalam simulasi, waktu pengiriman paket data dan lain sebagainya. Untuk lebih jelasnya pada Tabel 5.2 dibawah ini menunjukkan kriteria-kriteria yang ditentukan didalam skenario pengujian DSDV di NS-2 menggunakan MANET.

Tabel 5.2 Kriteria Pengujian

Kriteria	Spesifikasi
Skenario	MANET (<i>Random Way Point</i>), <i>mobility generator</i>
Kecepatan Maksimal Perpindahan <i>Node</i> (m/s)	5, 10, 15
Jumlah Percobaan	10 kali
Jarak antar <i>Node</i>	Acak
Posisi Awal <i>Node</i>	Acak
Pergerakan	Acak
Protokol <i>Routing</i>	DSDV
Pengiriman Paket Data	Nakagami 100 – 200 detik

5.3. Analisis *Packet Delivery Ratio (PDR)*

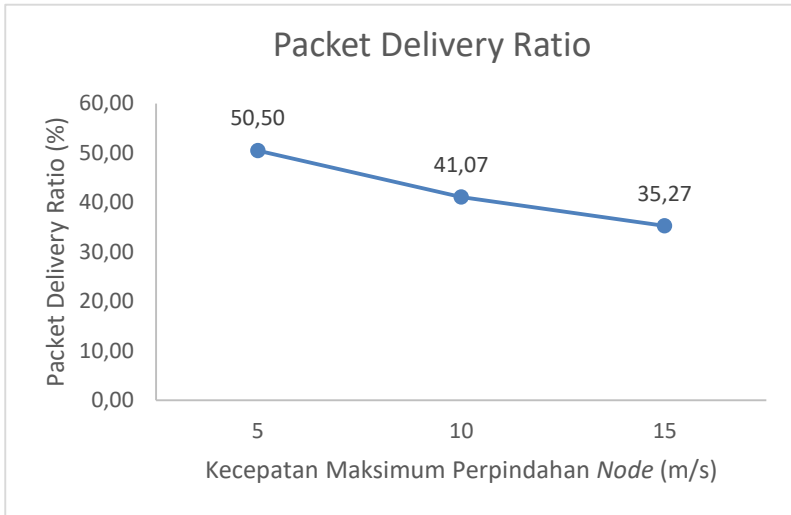
Trace file hasil menjalankan program skenario *node-movement (mobility generation)* menggunakan model transmisi Nakagami kemudian dianalisis nilai PDR melalui *script pdr.awk*. Hasil tiap perhitungan PDR skenario ditabulasikan dan dirata-ratakan menjadi seperti pada Tabel 5.3.

Tabel 5.3 PDR Skenario *Node-Movement*

Kecepatan maksimal perpindahan <i>node</i> (m/s)	PDR Nakagami (%)
5	50,50
10	41,07
15	35,27

Pada Tabel 5.3 dan Gambar 5.1 menunjukkan performa PDR yang dihasilkan oleh model transmisi Nakagami pada jaringan MANET dengan menggunakan skenario *node-movement (mobility generaion)* yang bersifat *Random Way Point*. Terlihat

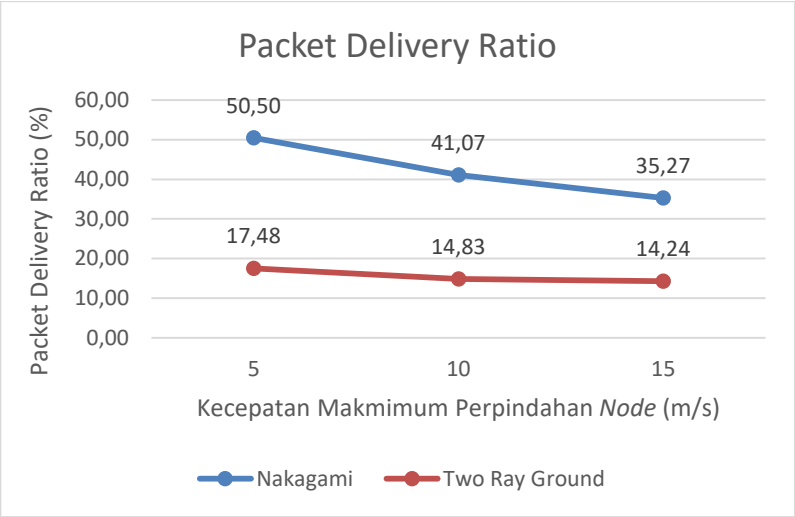
bahwa PDR yang dihasilkan berada pada titik terendah saat kecepatan perpindahan *node* 15 m/s.



Gambar 5.1 Grafik PDR Skenario *node-movement* Nakagami

Pada grafik hasil analisa PDR terlihat bahwa nilai PDR mengalami penurunan secara stabil. Pada hasil analisa PDR seiring dengan meningkatnya kecepatan maksimal perpindahan *node* maka meningkatnya sifat dinamis dari MANET yang menyebabkan topologi berubah dan nilai PDR mengalami penurunan.

Untuk lebih jelasnya tentang analisa PDR pada DSDV dibawah ini pada Gambar 5.2 menunjukkan hasil analisa DSDV dengan propagasi Nakagami dan *TwoRayGround*. Hasil analisa pada grafik menunjukkan kedua model propagasi baik Nakagami maupun *TwoRayGround* sama-sama mengalami penurunan secara stabil, namun nilai PDR Nakagami lebih tinggi dari pada *TwoRayGround*.



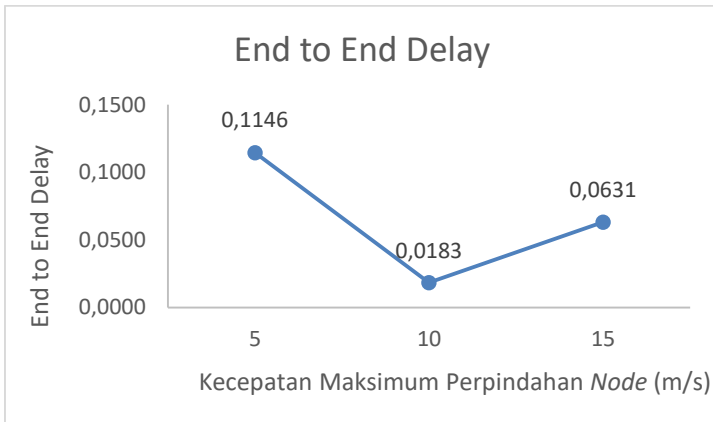
Gambar 5.2 Grafik PDR Skenario *Node Movement* Nakagami & *TwoRayGround*

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

Trace file hasil menjalankan program skenario *node-movement (mobility generation)* menggunakan model transmisi Nakagami kemudian dianalisis nilai *End-to-End Delay* melalui *script* *endtoend.awk*. Hasil tiap perhitungan E2D skenario ditabulasikan dan dirata-ratakan menjadi seperti pada Tabel 5.4.

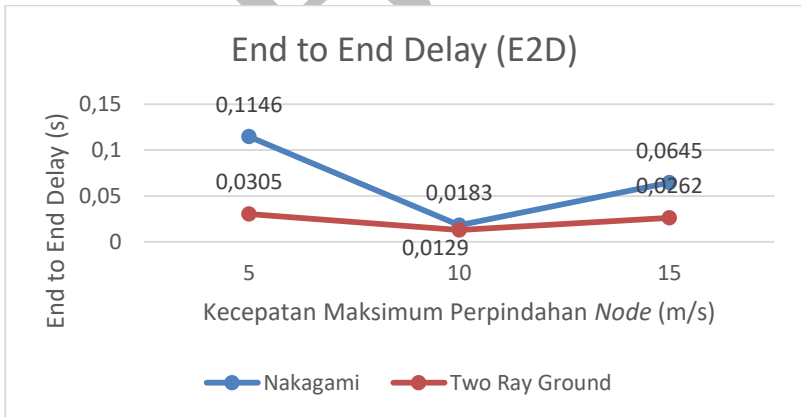
Tabel 5.4 E2D Skenario *Node-Movement*

Kecepatan Maksimum Perpindahan Node (m/s)	End to End Delay
5	0,1146
10	0,0183
15	0,0631



Gambar 5.3 Grafik E2D Skenario *node-movement* Nakagami

Dari grafik diatas bisa dilihat bahwa delay menjadi sangat kecil pada saat kecepatan maksimal 10m/s kemudian naik lagi dari 0.0183 menjadi 0.0631. Nilai E2D cenderung fluktuatif karena pengiriman dihitung berdasarkan jumlah paket sehingga tidak bisa konstan.



Gambar 5.4 Grafik E2D Skenario *node-movement* Nakagami & *TwoRayGround*

Gambar 5.4 diatas menunjukkan hasil E2D untuk Nakagami dan *TwoRayGround*. Nilai E2D kedua model propagasi mengalami fluktuatif kenaikan dan penurunan. Nilai E2D *TwoRayGround* lebih kecil jika dibandingkan dengan Nakagami.

5.5. Analisis *Routing Overhead* (RO)

Trace file hasil menjalankan program Skenario *node-movement* (*mobility generation*) menggunakan model transmisi Nakagami kemudian dianalisis nilai *Routing Overhead* melalui *script* *ro.awk*. Hasil tiap perhitungan RO skenario ditabulasikan dan dirata-ratakan menjadi Tabel 5.5.

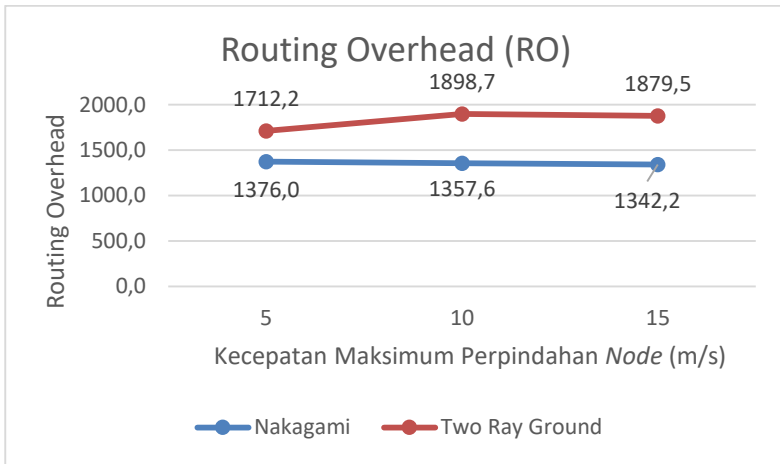
Tabel 5.5 RO Skenario *Node-Movement*

Kecepatan Maksimum Perpindahan <i>Node</i> (m/s)	RO
5	1376,0
10	1357,6
15	1342,2

Pada Tabel 5.5 dan Gambar 5.5 menunjukkan pengujian model transmisi *Nakagami* untuk nilai *Routing Overhead* yang dihasilkan memiliki nilai yang menurun stabil berdasarkan penambahan kecepatan maksimal perpindahan *node* yang terdapat dalam simulasi.



Gambar 5.5 Grafik RO Skenario *node-movement* Nakagami



Gambar 5.6 Grafik RO Skenario *node-movement* Nakagami & *TwoRayGround*

Gambar 5.6 diatas menunjukkan grafik RO DSDV Nakagami dan TwoRayGround untuk mengetahui lebih jelas performa DSDV pada Nakagami. Hasilnya bahwa nilai RO pada Nakagami lebih kecil dan mengalami penurunan secara stabil.

[Halaman ini sengaja dikosongkan]

RBTC

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 model transmisi Nakagami dengan penambahan kecepatan maksimal perpindahan *node* memiliki performa sebagai berikut:
 - Performa *Packet Delivery Ratio* yang dihasilkan mengalami penurunan yang stabil. Menurun secara stabil pada rentang nilai 50.50% menjadi 41.07% saat kecepatan maksimal perpindahan *node* sebesar 10 m/s kemudian turun lagi menjadi 35.27% saat kecepatan maksimal perpindahan *node* sebesar 15 m/s. Hal ini terjadi karena seiring dengan meningkatnya kecepatan maksimal perpindahan *node* maka meningkatnya sifat dinamis dari MANET yang menyebabkan topologi berubah dan nilai PDR mengalami penurunan. Jadi semakin banyak paket yang diterima maka nilai PDR akan semakin kecil.
 - Performa *End to End Delay* yang dihasilkan memiliki nilai *delay* fluktuatif, yaitu menurun tajam dari 0.1146 menjadi 0.0183 pada saat kecepatan maksimal perpindahan *node* sebesar 10 m/s kemudian naik secara tajam menjadi 0.0631 pada saat kecepatan maksimal perpindahan *node* sebesar 15 m/s. Hal yang menyebabkan nilai E2D fluktuatif adalah karena nilai E2D ditentukan berdasarkan hasil rata-rata waktu delay yang dibagi dengan banyaknya paket yang terkirim. Jadi dalam hal ini tidak bisa disamakan dengan

PDR dimana jumlah paket yang terkirim berpengaruh pada nilai PDR.

- *Routing Overhead* yang dihasilkan semakin menurun secara stabil, yaitu pada rentang nilai 1376 paket menjadi 1342.2 paket dari kecepatan maksimal perpindahan *node* sebesar 5 m/s hingga 15 m/s. Nilai *routing overhead* cenderung stabil dengan perubahan yang tidak terlalu signifikan. Hal ini menunjukkan penambahan kecepatan tidak terlalu berpengaruh padah hasil *routing overhead*

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 model transmisi Nakagami.
2. Dapat dilakukan pengurangan atau penambahan jumlah *node* dan penambahan jumlah percobaan untuk skenario *node-movement (mobility generation)*.
3. Dapat dilakukan penambahan kecepatan maksimal perpindahan *node* untuk uji coba agar grafik perubahan yang dihasilkan lebih bervariasi sehingga lebih jelas.
4. Dapat dilakukan modifikasi pada parameter-parameter yang digunakan untuk membangkitkan uji coba MANET pada NS-2 seperti modifikasi pada parameter *transmission range*, modifikasi pada *pause time* selama waktu simulasi berlangsung.

DAFTAR PUSTAKA

- [1] IPB, "Scientific Repository," [Online]. Available: <http://repository.ipb.ac.id/handle/123456789/14523>. [Accessed September 2017].
- [2] C. E. Perkins and P. Bhagwat, "Highly Dynamic Destination-Sequence Distance-Vector Routing (DSDV) for Mobile Computers," in *SIGCOMM*, London, UK, August, 1994.
- [3] "Binus University Master of Information Technology," 15 August 2014. [Online]. Available: <https://mti.binus.ac.id/2014/08/15/destination-sequenced-distance-vector-routing-dsdv/>. [Accessed September 2017].
- [4] A. A. Chavan, P. D. S. Kurule and P. P. U. Dere, "Performance Analysis of AODV and DSDV Routing Protocol in MANET and Modification in AODV against Black Hole Attack," in *7th International Conference on Communication, Computing and Virtualization 2016*, Mumbai, 2016.
- [5] E. R. a. C.-K. Toh, "A Review of Current Routing Protocols for Ad-Hoc Mobile Wireless Networks," *IEEE Personal Communications Magazine*, pp. 46-55, April 1999.
- [6] D. Johnson, "Routing in Ad Hoc Networks of Mobile Hosts," *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications*, December 1994.
- [7] Pajala, Elina, Isotalo, Tero and dkk, "An improved simulation model for Nakagami-m-m fading channels for satellite positioning applications," Finlandia, Institute of Communications Engineering Tampere University of Technology, p. 82.
- [8] P. D. V. L. A. Anjana Jain, "Estimation of Fading Statistics of Nakagami Channel with Weibull Distributed Tolerable Outage Time," *Scientific Research*, vol. 3, no. 2, p. 4 pages, 2012.

- [9] V. P. S. R. S. R. PriyaDharshini, "Research on Implementation and Comparison of Routing Protocol in MANET Using NS2," *International Journal of Science and Research (IJSR)*, vol. 3, no. 4, pp. 675-681, 2014.
- [10] R. Baumann, Engineering and simulation of mobile ad hoc routing protocols for VANET on Highways and in cities, Institute of Technology Zurich, 2014.
- [11] 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.
- [12] O. S. GNU, "<https://www.gnu.org/software/gawk/manual/gawk.html>," [Online]. [Accessed November 2017].

LAMPIRAN

1	#
2	# nodes: 50, pause: 10.00, max speed: 5.00, max
3	x: 510.00, max y: 510.00
4	#
5	\$node_(0) set X_ 451.785953254783
6	\$node_(0) set Y_ 126.212616074417
7	\$node_(0) set Z_ 0.000000000000
8	\$node_(1) set X_ 86.444329049070
9	\$node_(1) set Y_ 211.568272257270
10	\$node_(1) set Z_ 0.000000000000
11	\$node_(2) set X_ 129.607413806567
12	\$node_(2) set Y_ 161.706642991626
13	\$node_(2) set Z_ 0.000000000000
14	\$node_(3) set X_ 149.137357289570
15	\$node_(3) set Y_ 304.073658859665
16	\$node_(3) set Z_ 0.000000000000
17	\$node_(4) set X_ 66.301325245452
18	\$node_(4) set Y_ 288.528262866633
19	\$node_(4) set Z_ 0.000000000000
20	\$node_(5) set X_ 202.690170535249
21	\$node_(5) set Y_ 247.047380343046
22	\$node_(5) set Z_ 0.000000000000
23	\$node_(6) set X_ 2.135761236827
24	\$node_(6) set Y_ 359.869785507423
25	\$node_(6) set Z_ 0.000000000000
26	\$node_(7) set X_ 86.472897461234
27	\$node_(7) set Y_ 108.797999711238
28	\$node_(7) set Z_ 0.000000000000
29	\$node_(8) set X_ 244.735303333714
30	\$node_(8) set Y_ 301.778347145276
31	\$node_(8) set Z_ 0.000000000000
32	\$node_(9) set X_ 290.030386342276
33	\$node_(9) set Y_ 179.678212952118
34	\$node_(9) set Z_ 0.000000000000
35	\$node_(10) set X_ 204.437773437951
36	\$node_(10) set Y_ 7.740988102771
37	\$node_(10) set Z_ 0.000000000000
38	\$node_(11) set X_ 486.284090612803
39	\$node_(11) set Y_ 311.726355252108
40	\$node_(11) set Z_ 0.000000000000
41	\$node_(12) set X_ 190.957410461069
42	\$node_(12) set Y_ 129.452242000510

43	\$node_(12)	set Z_	0.000000000000
44	\$node_(13)	set X_	177.490270300297
45	\$node_(13)	set Y_	8.865096206763
46	\$node_(13)	set Z_	0.000000000000
47	\$node_(14)	set X_	346.402894470795
48	\$node_(14)	set Y_	238.495180499041
49	\$node_(14)	set Z_	0.000000000000
50	\$node_(15)	set X_	46.482936257601
51	\$node_(15)	set Y_	164.929662501758
52	\$node_(15)	set Z_	0.000000000000
53	\$node_(16)	set X_	295.910610046565
54	\$node_(16)	set Y_	53.750345418246
55	\$node_(16)	set Z_	0.000000000000
56	\$node_(17)	set X_	360.214328117338
57	\$node_(17)	set Y_	41.092315355130
58	\$node_(17)	set Z_	0.000000000000
59	\$node_(18)	set X_	127.819217917602
60	\$node_(18)	set Y_	422.848018501510
61	\$node_(18)	set Z_	0.000000000000
62	\$node_(19)	set X_	283.940046727548
63	\$node_(19)	set Y_	86.657700791210
64	\$node_(19)	set Z_	0.000000000000
65	\$node_(20)	set X_	143.617536069929
66	\$node_(20)	set Y_	397.266469102219
67	\$node_(20)	set Z_	0.000000000000
68	\$node_(21)	set X_	475.929540876147
69	\$node_(21)	set Y_	173.497038494652
70	\$node_(21)	set Z_	0.000000000000
71	\$node_(22)	set X_	385.230176194542
72	\$node_(22)	set Y_	25.952710057181
73	\$node_(22)	set Z_	0.000000000000
74	\$node_(23)	set X_	325.668306795184
75	\$node_(23)	set Y_	423.067347849256
76	\$node_(23)	set Z_	0.000000000000
77	\$node_(24)	set X_	359.375354589446
78	\$node_(24)	set Y_	464.209820277906
79	\$node_(24)	set Z_	0.000000000000
80	\$node_(25)	set X_	302.659678664807
81	\$node_(25)	set Y_	46.748754450525
82	\$node_(25)	set Z_	0.000000000000
83	\$node_(26)	set X_	61.293375415034
84	\$node_(26)	set Y_	38.176789491152
85	\$node_(26)	set Z_	0.000000000000
86	\$node_(27)	set X_	97.440628874500

87	\$node_(27)	set Y_	1.014484574788
88	\$node_(27)	set Z_	0.000000000000
89	\$node_(28)	set X_	174.069450256984
90	\$node_(28)	set Y_	160.946497187687
91	\$node_(28)	set Z_	0.000000000000
92	\$node_(29)	set X_	299.970763864442
93	\$node_(29)	set Y_	272.354480430887
94	\$node_(29)	set Z_	0.000000000000
95	\$node_(30)	set X_	188.023010771439
96	\$node_(30)	set Y_	276.001652201718
97	\$node_(30)	set Z_	0.000000000000
98	\$node_(31)	set X_	322.772995847460
99	\$node_(31)	set Y_	442.691395662215
100	\$node_(31)	set Z_	0.000000000000
101	\$node_(32)	set X_	336.099238255676
102	\$node_(32)	set Y_	275.718173268107
103	\$node_(32)	set Z_	0.000000000000
104	\$node_(33)	set X_	334.683265305152
105	\$node_(33)	set Y_	475.112540026523
106	\$node_(33)	set Z_	0.000000000000
107	\$node_(34)	set X_	198.975770156123
108	\$node_(34)	set Y_	261.736869134770
109	\$node_(34)	set Z_	0.000000000000
110	\$node_(35)	set X_	65.722441955066
111	\$node_(35)	set Y_	201.733173672226
112	\$node_(35)	set Z_	0.000000000000
113	\$node_(36)	set X_	126.456874579431
114	\$node_(36)	set Y_	74.457003997419
115	\$node_(36)	set Z_	0.000000000000
116	\$node_(37)	set X_	92.236731149079
117	\$node_(37)	set Y_	14.082360784321
118	\$node_(37)	set Z_	0.000000000000
119	\$node_(38)	set X_	30.188591629087
120	\$node_(38)	set Y_	62.752048932581
121	\$node_(38)	set Z_	0.000000000000
122	\$node_(39)	set X_	383.036545960140
123	\$node_(39)	set Y_	42.453716700052
124	\$node_(39)	set Z_	0.000000000000
125	\$node_(40)	set X_	54.101553702057
126	\$node_(40)	set Y_	28.125901306092
127	\$node_(40)	set Z_	0.000000000000
128	\$node_(41)	set X_	394.540994561866
129	\$node_(41)	set Y_	89.929095151660
130	\$node_(41)	set Z_	0.000000000000

131	\$node_(42)	set X_	468.445035816768
132	\$node_(42)	set Y_	254.071165802144
133	\$node_(42)	set Z_	0.000000000000
134	\$node_(43)	set X_	194.974542866625
135	\$node_(43)	set Y_	29.820961575666
136	\$node_(43)	set Z_	0.000000000000
137	\$node_(44)	set X_	302.031519972383
138	\$node_(44)	set Y_	216.976929367800
139	\$node_(44)	set Z_	0.000000000000
140	\$node_(45)	set X_	225.307422022788
141	\$node_(45)	set Y_	328.052384956949
142	\$node_(45)	set Z_	0.000000000000
143	\$node_(46)	set X_	449.302877694135
144	\$node_(46)	set Y_	208.201574731136
145	\$node_(46)	set Z_	0.000000000000
146	\$node_(47)	set X_	27.674792459318
147	\$node_(47)	set Y_	220.403373109620
148	\$node_(47)	set Z_	0.000000000000
149	\$node_(48)	set X_	343.499946882480
150	\$node_(48)	set Y_	507.470381209589
151	\$node_(48)	set Z_	0.000000000000
152	\$node_(49)	set X_	202.628988946516
153	\$node_(49)	set Y_	40.893545873896
154	\$node_(49)	set Z_	0.000000000000

Gambar 7.1 Posisi *node* dari potongan Skenario

1	\$god_	set-dist	0	1	2
2	\$god_	set-dist	0	2	2
3	\$god_	set-dist	0	3	2
4	\$god_	set-dist	0	4	2
5	\$god_	set-dist	0	5	2
6	\$god_	set-dist	0	6	3
7	\$god_	set-dist	0	7	2
8	\$god_	set-dist	0	8	2
9	\$god_	set-dist	0	9	1
11	\$god_	set-dist	0	10	2
12	\$god_	set-dist	0	11	1
13	\$god_	set-dist	0	12	2
14	\$god_	set-dist	0	13	2
15	\$god_	set-dist	0	14	1
16	\$god_	set-dist	0	15	2
17	\$god_	set-dist	0	16	1

18	\$god_	set-dist	0	17	1
19	\$god_	set-dist	0	18	2
20	\$god_	set-dist	0	19	1
21	\$god_	set-dist	0	20	2
22	\$god_	set-dist	0	21	1
23	\$god_	set-dist	0	22	1
24	\$god_	set-dist	0	23	2
25	\$god_	set-dist	0	24	2
26	\$god_	set-dist	0	25	1
27	\$god_	set-dist	0	26	2
28	\$god_	set-dist	0	27	2
29	\$god_	set-dist	0	28	2
30	\$god_	set-dist	0	29	1
31	\$god_	set-dist	0	30	2
32	\$god_	set-dist	0	31	2
33	\$god_	set-dist	0	32	1
34	\$god_	set-dist	0	33	2
35	\$god_	set-dist	0	34	2
36	\$god_	set-dist	0	35	2
37	\$god_	set-dist	0	36	2
38	\$god_	set-dist	0	37	2
39	\$god_	set-dist	0	38	3
40	\$god_	set-dist	0	39	1
41	\$god_	set-dist	0	40	2
42	\$god_	set-dist	0	41	1
43	\$god_	set-dist	0	42	1
44	\$god_	set-dist	0	43	2
45	\$god_	set-dist	0	44	1
46	\$god_	set-dist	0	45	2
47	\$god_	set-dist	0	46	1
48	\$god_	set-dist	0	47	3
49	\$god_	set-dist	0	48	2
50	\$god_	set-dist	0	49	2
51	\$god_	set-dist	1	2	1
52	\$god_	set-dist	1	3	1
53	\$god_	set-dist	1	4	1
54	\$god_	set-dist	1	5	1
55	\$god_	set-dist	1	6	1
56	\$god_	set-dist	1	7	1
57	\$god_	set-dist	1	8	1
58	\$god_	set-dist	1	9	1
59	\$god_	set-dist	1	10	1
60	\$god_	set-dist	1	11	2
61	\$god_	set-dist	1	12	1

62	\$god_	set-dist	1	13	1
63	\$god_	set-dist	1	14	2
64	\$god_	set-dist	1	15	1
65	\$god_	set-dist	1	16	2
66	\$god_	set-dist	1	17	2
67	\$god_	set-dist	1	18	1
68	\$god_	set-dist	1	19	1
69	\$god_	set-dist	1	20	1
70	\$god_	set-dist	1	21	2
71	\$god_	set-dist	1	22	2
72	\$god_	set-dist	1	23	2
73	\$god_	set-dist	1	24	2
74	\$god_	set-dist	1	25	2
75	\$god_	set-dist	1	26	1
76	\$god_	set-dist	1	27	1
77	\$god_	set-dist	1	28	1
78	\$god_	set-dist	1	29	1
79	\$god_	set-dist	1	30	1
80	\$god_	set-dist	1	31	2
81	\$god_	set-dist	1	32	2
82	\$god_	set-dist	1	33	2
83	\$god_	set-dist	1	34	1
84	\$god_	set-dist	1	35	1
85	\$god_	set-dist	1	36	1
86	\$god_	set-dist	1	37	1
87	\$god_	set-dist	1	38	1
88	\$god_	set-dist	1	39	2
89	\$god_	set-dist	1	40	1
90	\$god_	set-dist	1	41	2
91	\$god_	set-dist	1	42	2
92	\$god_	set-dist	1	43	1
93	\$god_	set-dist	1	44	1
94	\$god_	set-dist	1	45	1
95	\$god_	set-dist	1	46	2
96	\$god_	set-dist	1	47	1
97	\$god_	set-dist	1	48	2
98	\$god_	set-dist	1	49	1
99	\$god_	set-dist	2	3	1
100	\$god_	set-dist	2	4	1
101	\$god_	set-dist	2	5	1
102	\$god_	set-dist	2	6	1
103	\$god_	set-dist	2	7	1
104	\$god_	set-dist	2	8	1
105	\$god_	set-dist	2	9	1

106	\$god_	set-dist	2	10	1
107	\$god_	set-dist	2	11	2
108	\$god_	set-dist	2	12	1
109	\$god_	set-dist	2	13	1
110	\$god_	set-dist	2	14	1
111	\$god_	set-dist	2	15	1
112	\$god_	set-dist	2	16	1
113	\$god_	set-dist	2	17	2
114	\$god_	set-dist	2	18	2
115	\$god_	set-dist	2	19	1
116	\$god_	set-dist	2	20	1
117	\$god_	set-dist	2	21	2
118	\$god_	set-dist	2	22	2
119	\$god_	set-dist	2	23	2
120	\$god_	set-dist	2	24	2
121	\$god_	set-dist	2	25	1
122	\$god_	set-dist	2	26	1
123	\$god_	set-dist	2	27	1
124	\$god_	set-dist	2	28	1
125	\$god_	set-dist	2	29	1
126	\$god_	set-dist	2	30	1
127	\$god_	set-dist	2	31	2
128	\$god_	set-dist	2	32	1
129	\$god_	set-dist	2	33	2
130	\$god_	set-dist	2	34	1
131	\$god_	set-dist	2	35	1
132	\$god_	set-dist	2	36	1
133	\$god_	set-dist	2	37	1
134	\$god_	set-dist	2	38	1
135	\$god_	set-dist	2	39	2
136	\$god_	set-dist	2	40	1
137	\$god_	set-dist	2	41	2
138	\$god_	set-dist	2	42	2
139	\$god_	set-dist	2	43	1
140	\$god_	set-dist	2	44	1
141	\$god_	set-dist	2	45	1
142	\$god_	set-dist	2	46	2
143	\$god_	set-dist	2	47	1
144	\$god_	set-dist	2	48	2
145	\$god_	set-dist	2	49	1
146	\$god_	set-dist	3	4	1
147	\$god_	set-dist	3	5	1
148	\$god_	set-dist	3	6	1
149	\$god_	set-dist	3	7	1

150	\$god_	set-dist	3	8	1
151	\$god_	set-dist	3	9	1
152	\$god_	set-dist	3	10	2
153	\$god_	set-dist	3	11	2
154	\$god_	set-dist	3	12	1
155	\$god_	set-dist	3	13	2
156	\$god_	set-dist	3	14	1
157	\$god_	set-dist	3	15	1
158	\$god_	set-dist	3	16	2
159	\$god_	set-dist	3	17	2
160	\$god_	set-dist	3	18	1
161	\$god_	set-dist	3	19	2
162	\$god_	set-dist	3	20	1
163	\$god_	set-dist	3	21	2
164	\$god_	set-dist	3	22	2
165	\$god_	set-dist	3	23	1
166	\$god_	set-dist	3	24	2
167	\$god_	set-dist	3	25	2
168	\$god_	set-dist	3	26	2
169	\$god_	set-dist	3	27	2
170	\$god_	set-dist	3	28	1
171	\$god_	set-dist	3	29	1
172	\$god_	set-dist	3	30	1
173	\$god_	set-dist	3	31	1
174	\$god_	set-dist	3	32	1
175	\$god_	set-dist	3	33	2
176	\$god_	set-dist	3	34	1
177	\$god_	set-dist	3	35	1
178	\$god_	set-dist	3	36	1
179	\$god_	set-dist	3	37	2
180	\$god_	set-dist	3	38	2
181	\$god_	set-dist	3	39	2
182	\$god_	set-dist	3	40	2
183	\$god_	set-dist	3	41	2
184	\$god_	set-dist	3	42	2
185	\$god_	set-dist	3	43	2
186	\$god_	set-dist	3	44	1
187	\$god_	set-dist	3	45	1
188	\$god_	set-dist	3	46	2
189	\$god_	set-dist	3	47	1
190	\$god_	set-dist	3	48	2

Gambar 7.2 Pembuatan GOD setiap *node* dari potongan Skenario

1	\$ns_ at 10.000000000000 "\$node_(0) setdest 33.392695634552 112.547007091292 2.113849896630"
2	\$ns_ at 10.000000000000 "\$node_(1) setdest 489.712329995857 6.865637462598 3.922220837749"
3	\$ns_ at 10.000000000000 "\$node_(2) setdest 409.459129683091 55.089756748329 3.585334639239"
4	\$ns_ at 10.000000000000 "\$node_(3) setdest 180.629308850247 482.003026138206 4.422707437753"
5	\$ns_ at 10.000000000000 "\$node_(4) setdest 285.822893523416 486.211624697786 2.297733046098"
6	\$ns_ at 10.000000000000 "\$node_(5) setdest 468.696253529941 438.087795810369 1.352863243407"
7	\$ns_ at 10.000000000000 "\$node_(6) setdest 28.204259280229 334.754005518936 4.618745659408"
8	\$ns_ at 10.000000000000 "\$node_(7) setdest 456.078617557500 321.652292242198 2.116869301607"
9	\$ns_ at 10.000000000000 "\$node_(8) setdest 444.144080107093 178.023176174336 2.726062831241"
10	\$ns_ at 10.000000000000 "\$node_(9) setdest 74.259304752558 291.392042154806 3.323235037092"
11	\$ns_ at 10.000000000000 "\$node_(10) setdest 169.822967728408 304.162859196280 3.998979754231"
12	\$ns_ at 10.000000000000 "\$node_(11) setdest 245.362080800187 345.776706210234 2.546702790939"
13	\$ns_ at 10.000000000000 "\$node_(12) setdest 472.147442851371 459.241485698668 3.209519488174"
14	\$ns_ at 10.000000000000 "\$node_(13) setdest 225.464613895547 104.228172923778 3.915994296439"

15	\$ns_ at 10.000000000000 "\$node_(14) setdest 503.674940912609 309.324392755393 2.507903209572"
16	\$ns_ at 10.000000000000 "\$node_(15) setdest 400.568056169002 318.262837261118 0.779322692449"
17	\$ns_ at 10.000000000000 "\$node_(16) setdest 322.045443627390 477.386001724817 2.195293998025"
18	\$ns_ at 10.000000000000 "\$node_(17) setdest 1.753169604265 27.597190882130 1.282000776067"
19	\$ns_ at 10.000000000000 "\$node_(18) setdest 95.540559501954 65.931066207975 4.463717299153"
20	\$ns_ at 10.000000000000 "\$node_(19) setdest 378.969495155768 350.408023201131 0.132626100114"
21	\$ns_ at 10.000000000000 "\$node_(20) setdest 319.478226921398 55.811787626904 1.121082654732"
22	\$ns_ at 10.000000000000 "\$node_(21) setdest 129.404475592573 236.042598659839 4.281032853866"
23	\$ns_ at 10.000000000000 "\$node_(22) setdest 156.415407186003 325.271772878833 4.686859525942"
24	\$ns_ at 10.000000000000 "\$node_(23) setdest 397.211894171795 53.732933445473 2.045564276743"
25	\$ns_ at 10.000000000000 "\$node_(24) setdest 126.732295688781 129.678414781371 2.637930491168"
26	\$ns_ at 10.000000000000 "\$node_(25) setdest 238.312861879607 216.857063725188 4.759932324538"
27	\$ns_ at 10.000000000000 "\$node_(26) setdest 330.304334108555 381.578304075703 3.978760317569"
28	\$ns_ at 10.000000000000 "\$node_(27) setdest 354.995604774236 171.557602282609 3.699855750792"

Gambar 7.3 Pergerakan setiap *node* dari potongan Skenario

1	\$ns_ at 10.042788185115 "\$god_ set-dist 15 19 1"
2	\$ns_ at 10.134183422957 "\$god_ set-dist 22 28 1"
3	\$ns_ at 10.249586178225 "\$god_ set-dist 4 26 1"
4	\$ns_ at 10.431163522947 "\$god_ set-dist 3 33 1"
5	\$ns_ at 10.773426893360 "\$god_ set-dist 10 34 1"
6	\$ns_ at 10.783429117575 "\$god_ set-dist 13 34 1"
7	\$ns_ at 10.814885521465 "\$god_ set-dist 8 16 1"
8	\$ns_ at 10.826047682487 "\$god_ set-dist 5 26 1"
9	\$ns_ at 10.826047682487 "\$god_ set-dist 23 26 2"
10	\$ns_ at 10.826047682487 "\$god_ set-dist 26 31 2"
11	\$ns_ at 11.205153460654 "\$god_ set-dist 45 46 1"
12	\$ns_ at 11.208904416845 "\$god_ set-dist 25 30 1"
13	\$ns_ at 11.224078341276 "\$god_ set-dist 18 32 1"
14	\$ns_ at 11.233267390250 "\$god_ set-dist 18 22 2"
15	\$ns_ at 11.233267390250 "\$god_ set-dist 22 32 1"
16	\$ns_ at 11.233267390250 "\$god_ set-dist 22 48 2"
17	\$ns_ at 11.410358037743 "\$god_ set-dist 34 46 1"
18	\$ns_ at 11.422167101582 "\$god_ set-dist 15 20 1"
19	\$ns_ at 11.447460643615 "\$god_ set-dist 10 33 2"
20	\$ns_ at 11.447460643615 "\$god_ set-dist 13 33 2"
21	\$ns_ at 11.447460643615 "\$god_ set-dist 33 34 1"

Gambar 7.4 iInformasi pada GOD dari potongan Skenario

```

1  #
2  # nodes: 10, max conn: 8, 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_ 1
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 cbrtest.txt

```

1  # =====
2  # Define options
3  # =====
4
5  set val(chan)          Channel/WirelessChannel
6  set val(prop)          Propagation/Nakagami
7  set val(netif)         Phy/WirelessPhy
8  set val(mac)           Mac/802_11
9  set val(ifq)           Queue/DropTail/PriQueue
10 set val(ll)            LL
11 set val(ant)           Antenna/OmniAntenna
12 set opt(x)             510      ;# X dimension of the
    topography
13 set opt(y)             510      ;# Y dimension of the
    topography
13 set val(ifqlen)        50        ;# max packet in ifq
14 set val(nn)            50        ;# how many nodes are
    simulated
15 set val(seed)          0.0
16 set val(adhocRouting)  DSDV

```

```

17 set val(stop)      100      ;# simulation time
18 set val(cp)    "cbrtest.txt";#<-- traffic file
19 set val(sc)    "scena3.txt" ;#<-- mobility file
20
21 Phy/WirelessPhy      set    RXThresh_ 1.42681e-08
22 ;#100m
23 #=====
24 # Main Program
25 #=====
26 # Initialize Global Variables
27 # create simulator instance
28
29 set ns_                [new Simulator]
30
31 # setup topography object
32
33 set topo                [new Topography]
34
35 # create trace object for ns and nam
36
37 set tracefd [open dsdv_outnakaa4.tr w]
38 set namtrace [open dsdv_outnakaa4.nam w]
39
40 $ns_ trace-all $tracefd
41 $ns_ namtrace-all-wireless $namtrace $opt(x)
42 $opt(y)
43
44 # set up topology object
45 set topo [new Topography]
46 $topo load_flatgrid $opt(x) $opt(y)
47
48 # Create God
49 set god_ [create-god $val(nn)]
50
51 #global node setting
52 $ns_ node-config -adhocRouting
53 $val(adhocRouting) \
54     -llType $val(ll) \
55     -macType $val(mac) \
56     -ifqType $val(ifq) \
57     -ifqLen $val(ifqlen) \
58     -antType $val(ant) \
59     -propType $val(prop) \
60     -phyType $val(netif) \

```

61	-channelType \$val(chan) \
62	-topoInstance \$topo \
63	-agentTrace ON \
64	-routerTrace ON \
65	-macTrace OFF \
66	-movementTrace ON \
67	
68	
69	###
70	
71	# 802.11p default parameters
72	
73	Phy/WirelessPhyExt set CStresh_
	3.9810717055349694e-13 ;# -94 dBm wireless
	interface sensitivity
74	Phy/WirelessPhyExt set Pt_
	0.1 ;# equals 20dBm when
	considering antenna gains of 1.0
75	Phy/WirelessPhyExt set freq_
	5.9e+9
76	Phy/WirelessPhyExt set noise_floor_
	1.26e-13 ;# -99 dBm for 10MHz
	bandwidth
77	Phy/WirelessPhyExt set L_
	1.0 ;# default radio circuit
	gain/loss
78	Phy/WirelessPhyExt set PowerMonitorThresh_
	3.981071705534985e-18 ;# -174 dBm power
	monitor sensitivity (=level of gaussian noise)
79	Phy/WirelessPhyExt set HeaderDuration_
	0.000040 ;# 40 us
80	Phy/WirelessPhyExt set BasicModulationScheme_
	0
81	Phy/WirelessPhyExt set PreambleCaptureSwitch_
	1
82	Phy/WirelessPhyExt set DataCaptureSwitch_
	1
83	Phy/WirelessPhyExt set SINR_PreambleCapture_
	3.1623; ;# 5 dB
84	Phy/WirelessPhyExt set SINR_DataCapture_
	10.0; ;# 10 dB
85	Phy/WirelessPhyExt set trace_dist_
	1e6 ;# PHY trace until distance
	of 1 Mio. km ("infinity")

```

86  Phy/WirelessPhyExt set PHY_DBG_          0
87
88  Mac/802_11Ext set CWMin_
    15
89  Mac/802_11Ext set CWMax_
    1023
90  Mac/802_11Ext set SlotTime_
    0.000013
91  Mac/802_11Ext set SIFS_
    0.000032
92  Mac/802_11Ext set ShortRetryLimit_
    7
93  Mac/802_11Ext set LongRetryLimit_
    4
94  Mac/802_11Ext set HeaderDuration_
    0.000040
95  Mac/802_11Ext set SymbolDuration_
    0.000008
96  Mac/802_11Ext set BasicModulationScheme_
    0
97  Mac/802_11Ext set use_802_11a_flag_
    true
98  Mac/802_11Ext set RTSThreshold_
    2346
99  Mac/802_11Ext set MAC_DBG                0
100
101  ###
102  # Create the specified number of nodes
    [$val(nn)] and "attach" them
103  # to the channel.
104  for {set i 0} {$i < $val(nn)} {incr i} {
105      set node_($i) [$ns_ node]
106      $node_($i) random-motion 0 ;#
107      disable random motion
108  }
109
110  # Define node movement model
111  puts "Loading connection pattern..."
112  source $val(cp)
113
114  # Define traffic model
115  puts "Loading scenario file..."
116  source $val(sc)
117

```

```

118 # Define node initial position in nam
119
120 for {set i 0} {$i < $val(nn)} {incr i} {
121
122     # 20 defines the node size in nam, must
    adjust it according to your scenario
123     # The function must be called after mobility
    model is defined
124
125     $ns_ initial_node_pos $node_($i) 20
126 }
127
128 # Tell nodes when the simulation ends
129 for {set i 0} {$i < $val(nn)} {incr i} {
130     $ns_ at $val(stop).0 "$node_($i) reset";
131 }
132
133 # $ns_ at $val(stop) "stop"
134 $ns_ at $val(stop).0002 "puts \"NS EXITING...\"
    ; $ns_ halt"
135
136 puts $tracefd "M 0.0 nn $val(nn) x $opt(x) y
    $opt(y) rp $val(adhocRouting)"
137 puts $tracefd "M 0.0 sc $val(sc) cp $val(cp)
    seed $val(seed)"
138 puts $tracefd "M 0.0 prop $val(prop) ant
    $val(ant)"
139
140 puts "Starting Simulation..."
141 $ns_ run

```

Gambar 7.6 File .tcl untuk Protokol Routing DSDV


```

1 BEGIN {
2   sent=0;
3   recv=0;
4   pdr=0;
5 }
6 {
7   #count packet send
8   if ($1 == "s" && $3 == "_1_" && $4 == "AGT" &&
9   $7 == "cbr")
10     {
11       sent++;
12     }
13   #count packet receive
14   if ($1 == "r" && $3 == "_2_" && $4 == "AGT" &&
15   $7 == "cbr")
16     {
17       recv++;
18     }
19 }
20 END {
21   pdr = ( recv / sent ) * 100
22   print "Transmitted packet (s):", sent;
23   print "Received packet (s):", recv;
24   print "Packet delivery ratio:", pdr, "%";
25 }

```

Gambar 7.7 Implementasi Packet Delivery Ratio

```

1 BEGIN {
2   rt_pkts = 0;
3 }
4 {
5   if (($1 == "s" || $1 == "f") && ($4 == "RTR"))
6     rt_pkts++;
7 }
8 END {
9   printf ("Total number of routing packets\t%d\n",
10   rt_pkts);
11 }

```

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_rcv )
11    {
12        pkt_rcv[i] = 0;
13    }
14    delay = avg_delay = 0;
15    rcv = 0;
16    rcv_id = 0;
17  }
18  {
19    # count packet send
20    if ( $1 == "s" && $3 == "_1_" && $4 == "AGT" &&
21        $7 == "cbr" )
22    {
23        pkt_sent[$6] = $2;
24    }
25    # count packet receive
26    if ( $1 == "r" && $3 == "_2_" && $4 == "AGT" &&
27        $7 == "cbr" && rcv_id != $6 )
28    {
29        rcv++;
30        rcv_id = $6;
31        pkt_rcv[$6] = $2;
32    }
33  }
34  END{
35    for (i in pkt_rcv)
36      {
37          delay += pkt_rcv[i] - pkt_sent[i];
38      }
39
40    avg_delay = delay / rcv;
41
42    print "Total Packet(s) Receive =", rcv;
43    print "Total Delay =", delay, "second";
44  }

```

45	<code>print "Average Packet Delivery Delay = ",</code>
46	<code>avg_delay, "second";</code>
	<code>}</code>

Gambar 7.9 Implementasi *End-to-End Delay*

Instalasi NS-2

Dokumentasi tentang cara instalasi NS-2 dan proses *patching routing* protokol DSDV bersumber dari nsnam.com. Pada Tugas Akhir ini digunakan cara mengunduh *file* NS-2 dan *patch* DSDV yang disediakan oleh nsnam.com. Langkah-langkah detail adalah sebagai berikut.

- Pertama kali unduh package ns2 yang berekstensi .tar.gz pindahkan ke dalam direktori /home kemudian lakukan ekstraksi file .tar.gz tersebut dengan perintah

```
$ tar -xzf ns-allinone-2.35.tar.gz
```

- Kemudian lakukan instalasi modul-modul dependensi dari NS-2 yaitu *build-essential*, *autoconf*, *automake*, *tcl8.5-dev* *tk8.5-dev*, *perl*, *xgraph*, *libxt-dev*, *libx11-dev*, *libxmu-dev* dan *gcc-4.4*. Sebelumnya lakukan dulu *update* komponen ubuntu

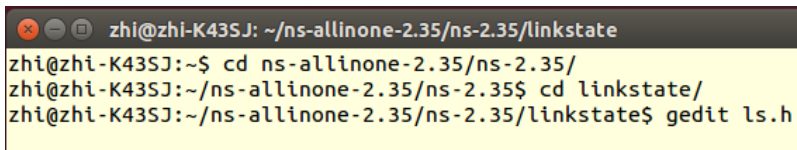
```
$ sudo apt-get update
$ sudo apt-get install build-essential autoconf
automake
$ sudo apt-get install tcl8.5-dev tk8.5-dev
$ sudo apt-get install perl xgraph libxt-dev libx11-
dev libxmu-dev
$ sudo apt-get install gcc-4.4
```

Gambar 7.10 Perintah instalasi dependensi NS-2

- Setelah semua dependensi lengkap, dilakukan proses pengubahan script pada *ls.h* yang terdapat pada *folder* /ns-allinone-2.35/ns-2.35/linkstate/*ls.h*.

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

Gambar 7.11 Proses pengubahan *ls.h*



```
zhi@zhi-K43SJ: ~/ns-allinone-2.35/ns-2.35/linkstate
zhi@zhi-K43SJ:~$ cd ns-allinone-2.35/ns-2.35/
zhi@zhi-K43SJ:~/ns-allinone-2.35/ns-2.35$ cd linkstate/
zhi@zhi-K43SJ:~/ns-allinone-2.35/ns-2.35/linkstate$ gedit ls.h
```

Gambar 7.12 Screenshot proses perubahan ls.h

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

```
template<class Key, class T>
class LsMap : public map<Key, T, less<Key> > {
public:
    typedef less<Key> less_key;
    typedef map<Key, T, less_key> baseMap;
    LsMap() : baseMap() {}

    // this next typedef of iterator seems extraneous but is required by gcc-2.96
    typedef typename map<Key, T, less<Key> >::iterator iterator;
    typedef pair<iterator, bool> pair_iterator_bool;
    iterator insert(const Key & key, const T & item) {
        typename baseMap::value_type v(key, item);
        pair_iterator_bool ib = baseMap::insert(v);
        return ib.second ? ib.first : baseMap::end();
    }

    void eraseAll() { this->erase(baseMap::begin(), baseMap::end()); }
    T* findPtr(Key key) {
        iterator it = baseMap::find(key);
        return (it == baseMap::end()) ? (T *)NULL : &((*it).second);
    }
};
```

Gambar 7.13 Proses pengubahan line of code ls.h

- Kemudian dilakukan intall NS-2.
- ```
$ cd ns-allinone-2.35/
$ sudo ./install
```
- Lakukan perubahan pada *environment variables* dengan mengedit *file* .bashrc

```
$ sudo gedit .bashrc
```

```
LD_LIBRARY_PATH
OTCL_LIB=/home/ns-allinone-2.35/otcl-1.14/
NS2_LIB=/home/ns-allinone-2.35/lib/
USR_Local_LIB=/usr/local/lib/
export
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$OTCL_LIB:$NS2_LIB:
$USR_Local_LIB

TCL_LIBRARY
TCL_LIB=/home/ns-allinone-2.35/tcl8.5.10/library/
USR_LIB=/usr/lib/
export TCL_LIBRARY=$TCL_LIBRARY:$TCL_LIB:$USR_LIB

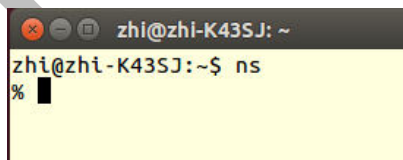
PATH
XGRAPH=/home/ns-allinone-2.35/xgraph-12.2/:/home/ns-
allinone-2.35/bin/:/home/ns-allinone-
2.35/tcl8.5.10/unix/:/home/ns-allinone-
2.35/tk8.5.10/unix/
NS=/home/ns-allinone-2.35/ns-2.35/
NAM=/home/ns-allinone-2.35/nam-1.15/
export PATH=$PATH:$XGRAPH:$NS:$NAM
```

**Gambar 7.14 File .bashrc**

Kemudian lakukan perintah validate

```
$./validate
```

- 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]*

RBTC

## BIODATA PENULIS



Penulis lahir di Karanganyar, 30 Mei 1994 sebagai adak bungsu dari tiga bersaudara. Penulis bernama lengkap Maharani Wahyu Siwi biasa dipanggil Zhi atau Siwi yang menyukai masakan jepang. Penulis menempuh pendidikan formal di TK Aisyah Bustanul Athfal (1998-1999) Sidodadi, SD N 3 Waru (2000-2006), SMP N 1 Kebakkramat (2006-2009) dan SMA N 1 Karanganyar (2009-2012). Tahun 2012 penulis

diterima sebagai mahasiswa S1 Departemen Informatika Fakultas Teknologi Informasi dan Komunikasi Institut Teknologi Sepuluh Nopember Surabaya (ITS).

Selain belajar di bangku perkuliahan penulis juga aktif mengikuti beberapa organisasi kemahasiswaan diantaranya di HMTC ITS, BEM ITS, KMI, dan JMMI ITS pada tahun kedua dan ketiga perkuliahan. Di Departemen Informatika ITS penulis mengambil bidang minat Arsitektur dan Jaringan Komputer dan mengambil topik Tugas Akhir tentang MANET. Penulis dapat dihubungi di telegram @mwsiiwi atau email maharanisiwii@gmail.com.