TUGAS AKHIR – KI141502

Studi Kinerja *Ad-hoc On-Demand Distance Vector* (AODV) pada Lingkungan Dinamis yang Ekstrim di Mobile Ad-hoc Networks (MANETs)

RIZQI HIDAYATULLAH
NRP 5110 100 167

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

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

TUGAS AKHIR – KI141502

Studi Kinerja *Ad-hoc On-Demand Distance Vector* (AODV) pada Lingkungan Dinamis yang Ekstrim di Mobile Ad-hoc Networks (MANETs)

RIZQI HIDAYATULLAH
NRP 5110 100 167

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

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

[Halaman ini sengaja dikosongkan]

FINAL PROJECT – KI141502

Perfomance Studies of *AdHoc On-Demand Distance Vector* (AODV) on Extreme Dynamic Environment of *Mobile Ad Hoc Network* (MANETs)

RIZQI HIDAYATULLAH
NRP 5110 100 017

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

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

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

STUDI KINERJA *ADHOC ON-DEMAND DISTANCE VECTOR* (AODV) PADA LINGKUNGAN DINAMIS YANG EKSTRIM PADA *MOBILE AD HOC NETWORK* (MANETS)

TUGAS AKHIR

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

Oleh:
RIZQI HIDAYATULLAH
NRP. 5110 100 167

Disetujui oleh Pembimbing Tugas Akhir:

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

SURABAYA
JUNI, 2017

[Halaman ini sengaja dikosongkan]

Studi Kinerja *Ad-hoc On-Demand Distance Vector* (AODV) pada Lingkungan Dinamis yang Ekstrim di Mobile Ad-hoc Networks (MANETs)

Nama Mahasiswa	**: Rizqi Hidayatullah**
NRP	**: 5110 100 167**
Jurusan	**: Teknik Informatika FTIf - ITS**
Dosen Pembimbing	**: Dr. Eng. Radityo Anggoro, S.Kom., M.Sc.**

ABSTRAK

Perangkat mobile seperti laptop, ponsel, dan gadget lainnya mulai semakin berkembang dengan adanya teknologi nirkabel (wireless) yang memungkinkan perangkat-perangkat tersebut dapat saling berkomunikasi secara langsung dalam posisi yang dinamis (tidak diam hanya pada satu posisi saja) dan tidak memerlukan jaringan infrastruktur yang lengkap serta bersifat sementara. Jaringan seperti itu disebut sebagai MANET (Mobile AdHoc Network).

Dalam MANET, setiap node yang terlibat bergerak secara bebas sehingga jaringan tersebut dapat mengalami perubahan topologi dengan cepat. Karena masing-masing node dalam MANET memiliki jarak transmisi yang terbatas, beberapa node tidak bisa berkomunikasi secara langsung sehingga mereka saling mengirim pesan - pesan tertentu kepada node-node lain untuk memungkinkan terjadinya komunikasi. Namun implementasi MANET di dunia nyata masih cukup sulit untuk dilakukan sehingga banyak penelitian dilakukan dengan membuat simulasi MANET menggunakan Network Simulator.

Maka dari itu, pada Tugas Akhir ini yang diteliti adalah 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 AODV (AdHoc On

Demand Distance Vector) *sebagai protokol routing yang digunakan pada NS-2.*

Kemudian sistem pada Tugas Akhir ini diuji fungsionalitas dan performanya dengan melalui beberapa skenario yang telah ditentukan. Hasil dari pengujian adalah performa protokol routing yang diukur berdasarkan routing overhead, packet delivery ratio, dan delay pengiriman paket dari satu node ke node lainnya pada NS-2.

Kata Kunci: MANET Dinamis, AODV, Network Simulator.

Perfomance Studies of *AdHoc On-Demand Distance Vector* on Extreme Dynamic Environment of *Mobile Ad Hoc Network* (MANETs)

Name : Rizqi Hidayatullah
NRP : 5110 100 167
Major : Informatics Engineering, IT Dept – ITS
Advisor : Dr. Eng. Radityo Anggoro, S.Kom., M.Sc.

ABSTRACT

Mobile devices such as laptops, mobile phones, and other handheld gadgets began to grow in the presence of wireless technologies which allow those devices can communicated directly in dynamic positions (not only at fixed position) and needs no complete network, and also temporary. That kind of network is called MANET (Mobile AdHoc Network).

On the MANET, each involved nodes moves random so the network topology may change rapidly. Because the nodes in MANET has a limited transmission distance, some nodes can not directly communicate with the other so they must send some messages to each other so communication can be occured. But implementation of MANET in the real world, is still difficult to do, so many research was done by making a MANET simulation using network simulator.

Therefore, in this final project studied the scheme of MANET which generated by the file node-movement and traffic pattern that already available in the distribution of network simulator. This study uses the NS-2 as a network simulator with proactive MANET protocol type AODV (AdHoc On Demand Distance Vector) as the routing protocol that exist in NS-2.

Then the system in this final project tested the functionality and performance through a few scenarios that have been determined. The results of the test is performance of routing

protocol measured based on routing overhead, packet delivery ratio, and delay delivery of packets from one node to another in NS-2.

Keywords: Dynamic MANET, AODV, Network Simulator.

KATA PENGANTAR

Bismillahirrohmanirohim.

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

"STUDI KINERJA *AD-HOC ON-DEMAND DISTANCE VECTOR* (AODV) PADA LINGKUNGAN DINAMIS YANG EKSTRIM DI *MOBILE AD-HOC NETWORKS* (MANETs)"

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

1. Allah SWT, karena limpahan rahmat dan karunia-Nya lah penulis dapat menyelesaikan Tugas Akhir dan juga perkuliahan di Teknik Informatika ITS.
2. Orang tua dari sang penulis, Suryawati Idris dan Fauzi Jailani, yang tiada henti memberikan semangat, doa serta dukungan penuh kepada penulis selama ini sehingga dapat menyelesaikan Tugas Akhir ini dan perkuliahan di Teknik Informatika ITS Surabaya dalam berbagai macam bentuk.
3. Bapak Dr. Eng. Radityo Anggoro S.Kom., M.Sc. selaku dosen pembimbing dari penulis yang telah memberikan bimbingan, dukungan, masukan, nasihat dan banyak arahan kepada penulis dalam menyelesaikan Tugas Akhir ini.
4. Bapak Fajar Baskoro, S.Kom., M.T., selaku dosen wali dari penulis yang selalu memberi nasihat kepada penulis selama menjalani perkuliahan di Teknik Informatika ITS.
5. Bapak Dr. Darlis Herumurti, S.Kom., M.Kom. selaku Ketua Jurusan Teknik Informatika ITS.
6. Sahabat – sahabat dekat dari penulis, Zendra Anugerah Andromedha yang selalu mendengarkan keluh kesan

penulis, memberikan dukungan, hiburan, semangat, nasihat dan menemani keseharian penulis serta mengingatkan penulis agar cepat menyelesaikan Tugas Akhir, Fauzan Arif yang sama-sama saling menyemengati untuk mengerjakan Tugas Akhir, Alfian Maulana Azhari yang menyempatkan diri untuk datang ke kos walaupun hanya sekedar bermalam.

7. Big Boss, David, serta Jack yang memberikan banyak inspirasi kepada penulis agar tetap semangat dalam perjuangan mengerjakan Tugas Akhir hingga selesai.
8. *Lone Wanderer* dan *Sole Survivor* yang mau menemani pengerjaan Tugas Akhir ini, karena "*War Never Changes*."
9. Teman-teman lulusan TC angkatan 2010 yang masih mau menyempatkan diri untuk membantu penulis hingga saat ini, seperti Azi Prastyo, Dhiyan Sabila, Shulhan Khairy.
10. Hasbi As Shidiqi, Adri Ramadhan, dan Pattoe Marza, teman seperjuangan dalam mengerjakan Tugas Akhir yang ikut mengerjakan Tugas Akhir serta bimbingan bersama.
11. Adik tingkat di TC yang telah ramah dan berbaik hati membantu penulis selama berada di Teknik Informatika. Terima kasih atas rasa kekeluargaan yang telah kalian berikan kepada penulis.
12. *Cobalt Saber Fire*, *Revolver Hades*, *Cobalt Blaster*, serta *Garuburn* yang mau menemani di kos selama mengerjakan Tugas Akhir.
13. Pihak-pihak yang tidak dapat penulis sebutkan 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 penulis 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, Juni 2017

Rizqi Hidayatullah

[Halaman ini sengaja dikosongkan]

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

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

1.1. Latar Belakang

Zaman sekarang sudah banyak teknologi-teknologi *portable* yang senantiasa terus berkembang dan semakin canggih, beberapa diantaranya yaitu perangkat-perangkat *mobile* seperti *handphone*, *tablet*, ataupun *notebook* yang juga diimbangi dengan berkembangnya teknologi nirkabel (*wireless*). Teknologi jaringan nirkabel tersebut dapat membuat beberapa perangkat *mobile* 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).

Jaringan *Mobile Ad Hoc Network* (MANET) adalah suatu jaringan yang terorganisasi secara mandiri tanpa adanya dukungan dari suatu infrastruktur. Dalam MANET, masing-masing *node* bergerak secara bebas sehingga jaringan dapat mengalami perubahan topologi dengan cepat. Tetapi karena jarak transmisi antara masing-masing *node* dalam MANET cukup terbatas, maka beberapa *node* tersebut tidak bisa berkomunikasi langsung dengan *node* lainnya. Jalur *routing* pada MANET memiliki 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 jenis-jenis protokol *routing* yang diklasifikasikan menjadi tiga, diantaranya adalah 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. [1]

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). Dalam implementasi ini akan dilakukan analisa kinerjaa protokol *routing Ad-hoc On-Demand Distance Vector* (AODV) pada lingkungan dinamis di MANET.

Hasil yang diharapkan dari tugas akhir ini adalah hasil studi penggunaan *Ad-hoc On-Demand Distance Vector* (AODV) pada lingkungan dinamis yang ekstrim di MANET dengan menggunakan aplikasi. Kinerja protokol tersebut akan diukur berdasarkan *routing overhead, packet delivery ratio,* dan *delay* pengiriman paket antar *node*.

1.2. Rumusan Permasalahan

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

1. Bagaimana kinerja AODV pada lingkungan dinamis yang ekstrim di MANET.
2. Bagaimana menganalisa performa dari protokol *routing* dengan parameter berupa *Packet Delivery Ratio, Routing Overhead* dan *delay* pengiriman.
3. Bagaimana mensimulasikan dan mengintegrasikan protocol *routing* AODV menggunakan *Network Simulator-2* dengan skenario diatas.

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 AODV.
3. Analisis kinerja jaringan didasarkan pada *packet delivery ratio*, *routing overhead*, dan *delay* pengiriman.

1.4. Tujuan dan Manfaat

Tujuan dari pembuatan Tugas Akhir ini adalah untuk memberikan hasil studi kinerja *Ad-hoc On-Demand Distance Vector* (AODV) pada lingkungan dinamis yang ekstrim di MANET dengan menggunakan aplikasi *Network Simulator* 2 (NS-2).

Dengan dibuatnya Tugas Akhir ini akan memberikan sebuah manfaat yang bisa didapatkan berupa hasil kinerja *Ad-hoc On-Demand Distance Vector* (AODV) pada lingkungan dinamis yang ekstrim di MANET.

1.5. Metodologi

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

1. **Penyusunan Proposal Tugas Akhir**
   Pada tahap ini dilakukan penyusunan proposal Tugas Akhir yang berisi tentang deskripsi umum rancangan Tugas Akhir yang akan dibuat. Penyusunan ini terdiri dari menentukan judul Tugas Akhir, latar belakang, rumusan masalah, batasan masalah, tujuan Tugas Akhir, manfaat Tugas Akhir, tinjauan

pustaka, ringkasan Tugas Akhir, metodologi, serta rencana jadwal kegiatan pengerjaan Tugas Akhir.

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

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

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

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

   1. Pendahuluan
      a. Latar Belakang
      b. Rumusan Permasalahan
      c. Batasan Permasalahan

d. Tujuan dan Manfaat
e. Metodologi
f. Sistematika Penulisan

2. Tujuan Pustaka
3. Perancangan Sistem
4. Implementasi
5. Pengujian dan 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*.
- Bab VI. Penutup

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

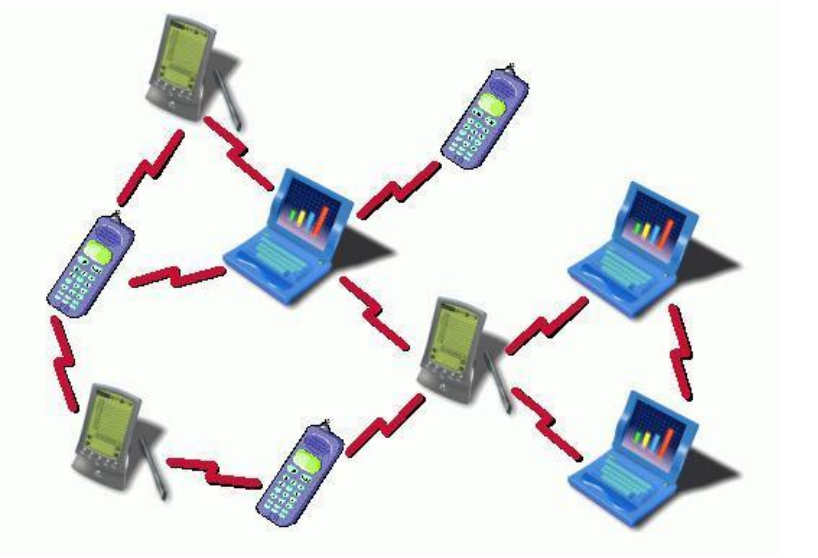
BAB II
TINJAUAN PUSTAKA

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. *Mobile Ad Hoc Network* (MANET)

Mobile Ad Hoc Network (MANET) adalah kumpulan dari *mobile node* bertenaga baterai yang beragam, tak berinfrastruktur, dan mandiri dengan ketersediaan sumber daya dan kemampuan komputasi yang berbeda. Sifat dinamis dan terdistribusi yang dimiliki MANET membuatnya cocok untuk penyebaran di kondisi yang ekstrim dan tidak stabil. MANET sering digunakan dalam berbagai bidang seperti operasi militer, pengawasan lingkungan, operasi penyelamatan, dan lain-lain. Masing-masing *node* pada MANET dilengkapi dengan *transmitter* dan *receiver* nirkabel (*wireless*), yang memungkinkan *node-node* tersebut dapat saling berkomunikasi dalam jarak transmisi nirkabel tersebut. Namun, karena terbatasnya jarak komunikasi nirkabel dan pergerakan *node*, *node-node* yang ada di dalam MANET harus saling bekerjasama untuk menyediakan layanan jaringan di antara mereka sendiri. Karena itu, masing-masing *node* di dalam MANET berfungsi sebagai *host* dan juga sebagai *router*. [2]

Pada Gambar 2.1 ditunjukkan penerapan jaringan *AdHoc* yang dibentuk dari sekumpulan perangkat *mobile* seperti ponsel dan laptop. Posisi perangkat-perangkat tersebut tentu tidak diam di satu tempat saja, terutama pada ponsel di mana penggunanya pasti sambil bepergian ke suatu tujuan sehingga posisi ponsel mereka pasti berpindah-pindah.



Gambar 2.1 Skema Jaringan MANET [3]

MANET digunakan pada tempat-tempat yang sangat membutuhkan rekonfigurasi dinamis serta penyebaran yang cepat serta jaringan kabel tidak tersedia seperti medan pertempuran, pencarian darurat, situs penyelamatan, ruang kelas atau konvensi di mana para partisipan saling berbagi informasi menggunakan perangkat *mobile* mereka. Manajemen topologi dan *routing* pada MANET menjadi masalah yang penting, banyak protokol-protokol *routing* efisien yang memastikan koneksi antar *node* untuk saling mengirim dan menerima dengan *delay* dan kontrol *overhead* yang tidak dibutuhkan seminimum mungkin. [4]

Untuk MANET terdapat beberapa jenis protokol *routing* yang secara keseluruhan terbagi menjadi beberapa kelompok, antara lain :

1. *Proactive Routing*

   Protokol yang termasuk pada bagian ini menjaga informasi-informasi *routing* bahkan sebelum hal tersebut diperlukan. Masing-masing dan setiap *node* di dalam jaringan

mempertahankan informasi *routing* tersebut ke semua *node* lainnya di dalam jaringan. Informasi *routing* biasanya disimpan di dalam *routing tables* dan senantiasa diperbarui secara berkala setiap topologi jaringan berubah. Kebanyakan protokol *routing* di sini berasal dari *link-state routing* seperti *Optimized Link State Routing* (OLSR).

2. *Reactive Routing*

   Protokol yang termasuk pada bagian ini tidak menjaga informasi-informasi *routing* atau aktivitas *routing* pada *node* di dalam jaringan jika tidak ada komunikasi. Jika ada *node* yang ingin mengirim suatu *packet* ke sebuah *node* lainnya makan protokol ini akan mencari rute sesuai permintaan dan mendirikan koneksi untuk mengirimkan serta menerima *packet*. Pencarian rute tersebut biasanya terjadi dengan menyebarkan permintaan rute (*route request*) di seluruh jaringan. Contoh protokol yang termasuk dalam bagian ini adalah seperti *AdHoc On-Demand Distance Vector*. [5]

3. *Flow Oriented Routing*

   Protokol *routing* ini bersifat *on-demand scheme* yang menggunakan prediksi pergerakan (*mobility prediction*). Hanya rute yang aktif saja yang akan dipertahankan dan tabel rute permanen tidak diperlukan. Di saat suatu sumber memiliki suatu *flow* untuk dikirim, protokol tersebut akan membangun rute mencapai tujuan sesuai permintaan kemudian *flow* tersebut dimasukkan ke dalam rute tersebut. Tempat tujuan kemudian memprediksi perubahan topologi sebelumnya dan menentukan kapan *flow* tersebut perlu dialihkan atau dilepaskan (*rerouted or "handoffed"*) berdasarkan informasi pergerakan yang terdapat di dalam paket data. [6]

4. *Hybrid Routing*

   Protokol ini memiliki keunggulan yang dimiliki oleh protokol *routing Distance Vector* dan protokol *routing Link State* dan menggabungkannya menjadi protokol *routing* baru. Biasanya, *hybrid routing protocol* ini berdasarkan *Distance Vector protocol* namun memiliki banyak fitur dan keunggulan dari *Link State protocol*. Contoh dari protokol ini adalah *Enhanced Interior Gateway Routing Protocol* (EIGRP). [7]

2.2. *AdHoc On-Demand Distance Vector* (AODV)

AdHoc On-Demand Distance Vector (AODV) adalah salah satu protokol *routing* yang sangat sering digunakan pada MANET. AODV adalah algoritma *on-demand* yang mampu *unicast* dan *multicast routing*. Protokol tersebut mencari rute mencapai tujuan di saat rute tersebut diinginkan oleh *node* sumber dan rute tersebut tetap dipertahankan hingga rute tersebut tidak diperlukan lagi. AODV bekerja dalam 2 tahap : mencari dan menemukan rute, serta mempertahankan rute tersebut.

Saat *node* sumber memulai transmisi data, protokol tersebut memulai pencarian untuk mendapatkan rute baru dengan melakukan *broadcast* RREQ kepada *node* sekitarnya. *Node-node* lainnya menerima pesan RREQ tersebut dan memperbarui informasi-informasi mereka pada *routing table* pada *node* sumber. Masing-masing *node* lainnya pada MANET melakukan *unicast* sebuah pesan RREP kepada *node* sumber di mana juga akan terbentuk nomor urut ke jalur kembalinya. Saat *node-node* tersebut mengirim pesan RREP ke *node* sumber, mereka juga membuat *forward pointer* mengarah ke tujuan.

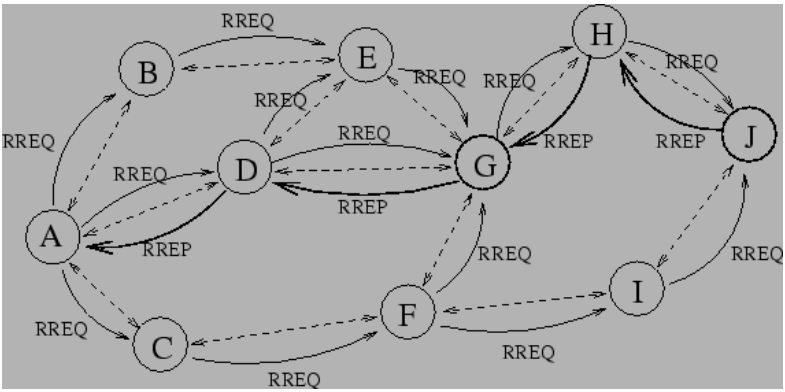
Ada tiga jenis pesan dalam AODV yaitu :

- RREQ, *Route Request*. Pesan ini dikirimkan oleh *node* yang memerlukan suatu rute menuju *node* lainnya.
- RREP, *Route Reply*. Pesan ini dikirimkan menuju *node* yang sebelumnya mengirimkan pesan RREQ jika *receiver* adalah *node* yang menggunakan alamat yang diminta, atau *receiver* tersebut memiliki rute yang valid menuju alamat yang diminta.
- RERR, *node-node* memantau status *link* pada *hops* berikutnya pada rute yang aktif. Jika terdapat *link* yang rusak pada rute yang aktif, maka pesan RERR akan digunakan untuk memberitahu *nodes* yang lain tentang *link* yang hilang tersebut. Agar mekanisme pelaporan ini dapat berjalan, tiap *node* menjaga "*precursor list*", berisi alamat IP untuk masing-masing "tetangga" mereka yang kira-kira akan dipakai pada *hop* berikutnya menuju masing-masing tujuan.

AODV memiliki keunggulan dan batasan, di antaranya adalah sebagai berikut :

- Keunggulan :
  - AODV termasuk protokol reaktif sehingga dapat lebih mudah menangani hal-hal dinamis di dalam jaringan *AdHoc*.
  - Dapat menggunakan *unicast routing* maupun *multicast routing*.
  - Jumlah *overhead* dalam jaringan lebih sedikit.

- Batasan :
  - Untuk mendeteksi *hop node* berikutnya, membutuhkan *broadcast medium* agar masing-masing *node* dapat saling mendeteksi satu sama lain.
  - Karena berdasarkan permintaan, *traffic* menjadi masalah pengukuran karena RREQ dan RREP.
  - Mengukur masa habis suatu rute menjadi sulit jika tidak ada yang ditransfer. [8]



Gambar 2.2 Ilustrasi AODV [9]

Gambar di atas merupakan ilustrasi dari sesi pencarin rute pada protokol AODV. *Node* A ingin membuat *traffic* dengan *node* J yang awalnya tidak berhubungan dengan *node* A. *Node* A melakukan *broadcast* mengirimkan RREQ ke semua *node* yang ada di dalam jaringan. Saat permintaan tersebut telah diterima oleh *node* J dari *node* H, *node* J membuat RREP yang kemudian akan dikirim kembali menuju *node* A melalui jalur yang dibentuk oleh *node* H, G, dan D. [9]

2.3. 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. [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 AODV. NS-2 dijalankan pada sistem operasi Linux dan proses instalasinya akan disajikan pada bagian Lampiran.

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

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

Tools yang disebut 'setdest' adalah *tools* yang digunakan untuk membuat *hops* antara masing-masing *node* dengan objek GOD (*General Operations Director*) dan membuat pergerakan untuk *node-node* dalam satuan detik dalam sebuah skenario untuk simulasi [11]. Lokasi 'setdest' berada pada direktori '~ns/indep-utils/cmu-scen-gen/setdest/'.

Pengguna harus menjalankan program 'setdest' sebelum menjalankan program simulasi. Format *Command Line* untuk menjalankan 'setdest' adalah sebagai berikut :

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

Keterangan untuk format *command line* di atas dapat dilihat pada Tabel 2.1.

Tabel 2.1 Keterangan pada *Command Line* 'setdest'

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

Perintah ‘setdest’ menghasilkan *output file* yang berisi jumlah *node* dan mobilitas yang akan digunakan dalam *file* berekstensi ‘tcl’ selama simulasi. Selain mengandung skrip pergerakan, *output file* juga mengandung beberapa statistik lain tentang perubahan *link* dan rute.

Untuk membuat skenario *node-movement* yang terdiri dari 60 *node*, bergerak dengan kecepatan maksimum 20.0 m/s dengan jeda rata-rata antar gerakan sebesar 2 detik, simulasi akan berhenti setelah 200 detik dengan batas topologi yang diartikan sebagai 1000 x 1000 meter^2, *command line* yang digunakan untuk scenario dengan kondisi tersebut adalah sebagai berikut :

```
“setdest -v 1 -n 60 -p 2.0 -M 20.0 -t 200 -x
1000 -y 1000 > scenario-s20-n60-1”
```

Di sini *output* disimpan ke dalam *file* "scenario-s20-n60-1". *File* dimulai dengan posisi awal *node* dan berlanjut menetapkan *node-movement* seperti terlihat pada Gambar 2.3.

```
$ns_    at    2.000000000000    "$node_(0)    setdest
133.836702871610 885.355102166967 14.857652813753"
```

Gambar 2.3 Hasil *Output* pada *file* 'scenario-s20-n60-1'

Command line pada Gambar 2.3 dari ' scenario-s20-n60-1' mendefinisikan bahwa *node* (0) pada detik ke 2.0 mulai bergerak ke arah tujuan (133.84, 885.36) dengan kecepatan 14,86 m/s. *Command line* ini dapat digunakan untuk mengubah arah dan kecepatan gerak dari *mobile node*. 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 ini, 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 Gambar 2.4.

```
$ns_ at 2.104970638624 "$god_ set-dist 2 46 3"
```

Gambar 2.4 *Command Line* "GOD" pada 'scenario-s20-n60-1'

Ini berarti bahwa jarak terpendek antara *node* 2 dan *node* 46 berubah menjadi 3 *hop* di waktu 2,1 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". [12]

2.4.2. *File Traffic-Connection Pattern Cbrgen.tcl*

Random traffic connection pada TCP dan CBR bisa dikonfigurasikan antara mobilitas *node* menggunakan skrip *traffic-scenario pattern generator*. Untuk menghasilkan alur *traffic* yang acak, dapat digunakan skrip Tcl yang disebut "cbrgen". 'cbrgen.tcl' adalah skrip yang membuat koneksi antar masing-masing *node*, mengatur *rate* pengiriman paket yang akan dikirmkan, mengatur jumlah koneksi yang dapat dibentuk oleh semua *node* dalam jaringan serta membuat jenis *agent* antar *node* yang digunakan, apakah TCP atau CBR [11], pada tugas akhir ini yang digunakan adalah CBR. CBR adalah *Constant Bit Rate*, berarti jumlah *bit rate* atau jumlah *bits* paket yang diproses adalah konstan. Skrip ini disimpan di dalam *file* 'cmu-scen-gen' yang terletak di dalam direktori "~ns/indep-utils/cmu-scen-gen". Program "cbrgen.tcl" digunakan sesuai dengan *command line* berikut :

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

Keterangan *command line* di atas dapat dilihat pada Tabel 2.2 sebagai berikut :

Tabel 2.2 Keterangan Command Line file cbrgn.tcl

Parameter	Keterangan
-type cbr|tcp	Jenis *traffic* yang digunakan, apakah TCP atau CBR
-nn *node*s	Jumlah *node* yang berkomunikasi dalam *traffic*
-s seed	*Random seed*
-mc connections	Jumlah koneksi

-rate rate	Jumlah paket per detik. Pada CBR, panjang paket adalah tetap yaitu sebesar 512 bytes selama simulasi

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

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

Berikut adalah *command line* yang digunakan untuk membuat sebuah *file* koneksi CBR antara 50 *node*, yang memiliki maksimal 20 koneksi, dengan nilai *seed* 1.0 dan jumlah paket per detik sebanyak 1 paket.

"`ns cbrgen.tcl -type cbr -nn 50 -seed 1.0 -mc 20 -rate 1.0 > cbr1`"

Dari *file* cbr1 (ke mana *output* dari generator akan diarahkan) sehingga menghasilkan salah satu koneksi CBR yang terlihat seperti pada Gambar 2.5.

```
#
# nodes: 50, max conn: 1, send rate: 1, seed: 1
#
#
# 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"
#
#Total sources/connections: 1/1
#
```

Gambar 2.5 Koneksi CBR pada 'cbr1'

Agent yang digunakan ialah UDP. Dengan demikian koneksi UDP adalah *setup* antara *node* 1 dan 2. Jumlah sumber UDP (dipilih antara *node* 0-50) dan jumlah *setup* koneksi diindikasikan sebagai 20 koneksi di akhir *file* cbr1.

2.5. 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 pada Gambar 2.6.

```
s 2.556838879 _1_ AGT  --- 0 cbr 512 [0 0 0 0] ----
--- [1:0 2:0 32 0] [0] 0 3
```

Gambar 2.6 *Trace* pengiriman paket data

Huruf "s" di kolom pertama menandakan pengiriman paket (*send*), kolom kedua berisi waktu pengiriman paket pada detik ke 2.56, 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 pada Gambar 2.7.

```
r 2.556838879 _1_ RTR  --- 0 cbr 512 [0 0 0 0] ----
--- [1:0 2:0 32 0] [0] 0 3
```

Gambar 2.7 *Trace* Penerimaan Paket Data

Contoh format untuk paket *routing* AODV pada *trace file* seperti pada Gambar 2.8.

```
r 2.557827174 _13_ RTR  --- 0 AODV 48 [0 ffffffff 1
800] ------- [1:255 -1:255 30 0] [0x2 1 1 [2 0] [1
4]] (REQUEST)
```

Gambar 2.8 *Trace* Pengiriman Paket *Routing* AODV

2.6. *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 dari *Awk* adalah mencari *file* per baris (atau unit teks lain) yang berisi suatu 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.

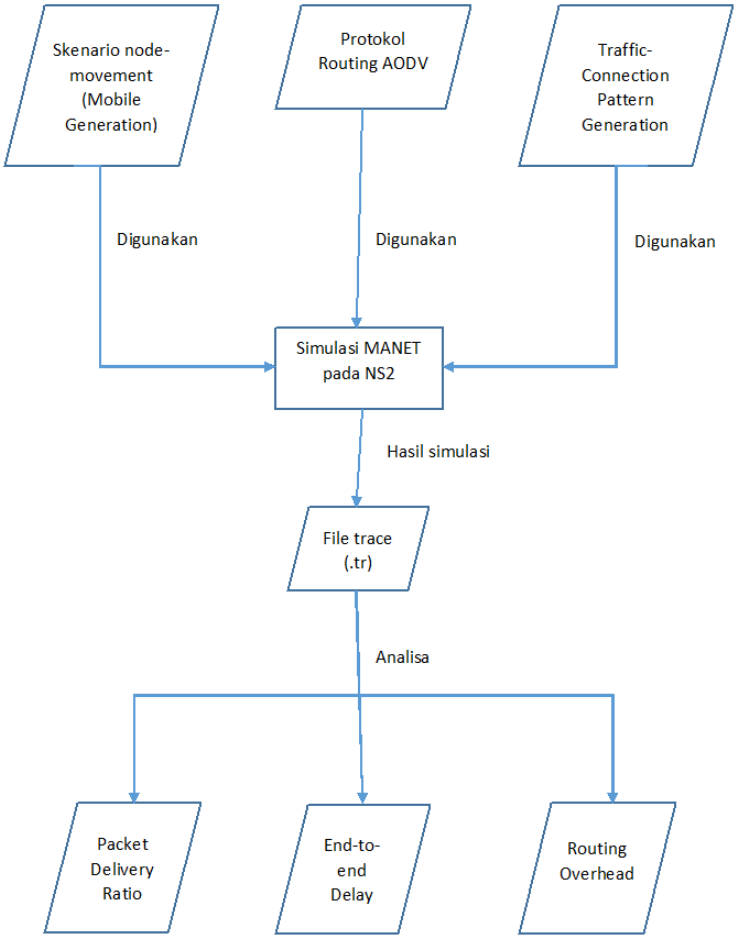
BAB III
PERANCANGAN SISTEM

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

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

Dalam penelitian ini, simulasi skenario yang dihasilkan oleh *mobility generator* akan dijalankan pada NS-2 menggunakan protokol *routing* AODV pada sistem operasi Linux. Pada tiap skenario, kecepatan maksimum pergerakan dari satu *node* ke *node* lainnya dibuat bervariasi yaitu 5, 10, 15 dan 20 m/s. 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 kedua model transmisi yang dibandingkan.



Gambar 3.1 Tahapan Rancangan Simulasi

3.2. Perancangan Skenario

Perancangan skenario uji coba mobilitas MANET diawali 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. Pada Tugas Akhir ini pembuatan skenario untuk melihat pergerakan *node* dibedakan berdasarkan 2 variasi luas daerah yaitu 1000 m x 1000 m dan 1500 m x 1500 m, serta 4 variasi kecepatan maksimum yaitu 5, 10, 15 dan 20 m/s. Sedangkan untuk koneksinya hanya digunakan 2 *node* untuk menentukan *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* yang telah ada pada NS-2 atau *tools*-nya biasa disebut 'setdest' yang nantinya akan menghasilkan *output file* untuk digunakan dalam *file* Tcl selama simulasi pada NS-2 sebagai bentuk pergerakan *node* yang berpindah-pindah.

Tabel 3.1 Parameter Skenario *Node-Movement*

No.	Parameter	Spesifikasi
1	Jumlah *Node*	60, 80, 100, 120
2	Waktu Simulasi	200 detik
3	Area	1000 m x 1000 m, 1500 m x 1500 m
4	Kecepatan Maksimal	- 5 m/s - 10 m/s - 15 m/s - 20 m/s
5	Sumber *Traffic*	CBR
6	Waktu Jeda (dalam detik)	10
7	Ukuran Paket	512 bytes

8	*Rate* Paket	0.25 paket per detik
9	Jumlah maksimal koneksi	1
10	Model mobilitas yang digunakan	*Random Way Point*

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 file* 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	-s seed	1.0
3	-mc connections	1
4	-rate rate	0.25
5	*Agent*	UDP

3.3. Perancangan Simulasi pada NS-2

Pada perancangan kode NS-2 dengan konfigurasi MANET, dilakukan penggabungan skenario mobilitas dan *traffic-connection* dengan skrip TCL yang diberikan parameter-parameter untuk membangun percobaan simulasi MANET pada NS-2. Berikut parameter simulasi perancangan 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	*Routing* Protocol	AODV
3	Waktu simulasi	200 detik
4	Waktu Pengiriman Paket Data	0 – 100 detik
5	Area simulasi	1000 m x 1000 m, 1500 m x 1500 m
6	Banyak *node*	60, 80, 100, 120
7	Radius transmisi	100 m
8	Tipe koneksi	UDP
9	Tipe data	Constant Bit Rate (CBR)
10	Source / Destination	Statik (*Node* 1 / *Node* 2)
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 (*random way point*)
18	Tipe kanal	Wireless channel
19	Tipe *trace*	Old Wireless Format *Trace*

3.4. Perancangan Metrik Analisis

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

3.4.1. *Packet Delivery Ratio* (PDR)

PDR dihitung dari perbandingan antara jumlah paket yang dikirim dengan jumlah paket yang diterima. PDR dihitung dengan menggunakan persamaan (1).

$$PDR = \frac{total\ packets\ received}{total\ packets\ sent} \times 100\% \quad \textbf{(1)}$$

3.4.2. *End-to-End Delay*

End-to-end Delay dihitung dari rata-rata *delay* antara waktu paket diterima dan waktu paket dikirim. E2D dihitung dengan menggunakan persamaan (2), di mana $t_{received[i]}$ adalah waktu penerimaan paket dengan urutan / *id* ke-i, $t_{sent[i]}$ adalah waktu pengiriman paket dengan urutan / *id* ke-i.

$$E2D = \frac{\sum_{i \le sent}^{i=0} t_{received[i]} - t_{sent[i]}}{total\ packets\ sent}\ (s) \quad \textbf{(2)}$$

3.4.3. *Routing Overhead* (RO)

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

[Halaman ini sengaja dikosongkan]

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 implemantasi 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 14.04 LTS 64 bit untuk lingkungan NS-2;
- *Network Simulator* 2 versi 2.35.

4.1.2. Lingkungan Perangkat Keras

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

- *Processor* Intel(R) Core(TM) i7-4710HQ CPU @2.50GHz (8 CPUs);
- Media penyimpanan sebesar 500GB;
- RAM sebesar 8 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 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* akan digunakan *tools generate default* yang dimiliki oleh NS-2 yaitu 'setdest'. *File* skenario *node-movement (mobility generation)* digunakan untuk setiap simulasi yang ditandai dengan jeda waktu. Agar dapat mempelajari efek mobilitas, simulasi dilakukan dengan pola gerakan yang dihasilkan dari kecepatan maksimal yang berbeda. *Setting* pada *file* skenario pergerakan *node* sesuai dengan mobilitas yang berbeda, dibuat dengan memvariasikan kecepatan maksimal. Program 'setdest' pada NS-2 digunakan untuk menghasilkan *file node-movement* dengan menggunakan algoritma *Random Way Point*. Format *command line* yang digunakan untuk menghasilkan gerakan acak pada *node* adalah sebagai berikut :

```
"setdest [-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 60, waktu jeda (*pause time*) yaitu 2 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 200 detik, panjang dan lebar maksimal area simulasi yaitu 1000 meter. Kemudian *file* mobilitas yang dihasilkan disimpan dalam direktori " ~ ns /indep-utils/CMU-scen-gen/setdest / ". Berikut terdapat contoh implementasi *command line* pada 'setdest' dengan berbagai kecepatan maksimal yang berbeda sesuai dan *node* sebanyak 60. Dan untuk setiap kecepatan maksimal tersebut dibuat 5 *file* skenario untuk masing-masing variasi jumlah *node* dan kecepatan maksimum.

Berikut adalah *command line* untuk membuat 3 dari 5 skenario dengan kecepatan maksimum sebesar 5 m/s pada 60 *node*.

```
"setdest –v 1 –n 60 –p 2 –M 5 –t 100 –x 1000
–y 1000 > scenario-s5-n60-1
setdest –v 1 –n 60 –p 2 –M 10 –t 100 –x 1000
–y 1000 > scenario-s10-n60-2
setdest –v 1 –n 60 –p 2 –M 15 –t 100 –x 1000
–y 1000 > scenario-s15-n60-3"
```

Pada Gambar 4.1 terdapat potongan dari *file* mobilitas "scenario-s5-n60-1" hasil *generate* 'setdest' yang menunjukkan penempatan awal setiap *node* dalam sebuah area dan dinyatakan dalam sumbu X, Y dan Z adalah sebagai berikut :

```
#
# nodes: 60, pause: 2.00, max speed: 5.00, max x:
1000.00, max y: 1000.00
#
$node_(0) set X_ 285.018740520789
$node_(0) set Y_ 725.297354828066
$node_(0) set Z_ 0.000000000000
$node_(1) set X_ 483.343298671629
$node_(1) set Y_ 443.539824163607
$node_(1) set Z_ 0.000000000000
$node_(2) set X_ 830.708253380684
$node_(2) set Y_ 757.387195373079
$node_(2) set Z_ 0.000000000000
```

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

Sedangkan 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 *script* pada *file* "scenario-s5-n60-1" pada Gambar 4.2, baris pertama mendefinisikan untuk *node* (0) pada detik ke 2 mulai bergerak ke arah tujuan (920.16, 346.04) dengan kecepatan 4,1 m/s. Baris

perintah ini dapat digunakan untuk mengubah arah dan kecepatan pada pergerakan mobilitas *node*.

```
$ns_ at 2.000000000000 "$node_(0) setdest
920.157653138226 346.042720828412 4.098179836856"
$ns_ at 2.000000000000 "$node_(1) setdest
762.362786236093 182.803113717364 1.549699032292"
$ns_ at 2.000000000000 "$node_(2) setdest
743.064234861490 318.801943983604 0.173852116838"
```

Gambar 4.2 Perpindahan/Pergerakan *Node*

Kemudian pada Gambar 4.3 terdapat potongan dari *file* "scenario-s5-n60-1" yang menunjukkan penentuan GOD untuk setiap *node*. GOD sendiri merupakan kepanjangan dari *General Operations Director*, yang berguna untuk menyimpan informasi global tentang jumlah dan pergerakan *node*. Saat ini, objek GOD hanya digunakan untuk menyimpan *array* terpendek dari *hop* yang diperlukan untuk mencapai dari satu *node* ke *node* yang lain. Objek GOD tidak menghitungan *array* ini selama simulasi berjalan karena bisa memakan waktu.

```
$god_ set-dist 0 1 2
$god_ set-dist 0 2 3
$god_ set-dist 0 3 4
$god_ set-dist 0 4 2
$god_ set-dist 0 5 4
$god_ set-dist 0 6 3
$god_ set-dist 0 7 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 dari potongan *file* "scenario-s5-n60-1" tersebut, digunakan untuk memuat objek GOD dengan informasi yaitu jalan terpendek antara *node* 12 dan *node* 47 berubah menjadi 2 *hop* pada detik ke-2.135489837623.

```
$ns_ at 2.135489837623 "$god_ set-dist 12 47 2"
$ns_ at 2.135489837623 "$god_ set-dist 28 47 1"
$ns_ at 2.444482530382 "$god_ set-dist 1 11 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/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 kan 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*. Berikut adalah 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 Gambar 4.5 merupakan bentuk implementasi untuk menjalankan cbrgent.tcl untuk membuat *file* koneksi CBR diantara 50 *node*, memiliki maksimal 1 koneksi dengan nilai *seed* 1.0 dan jumlah paket per detik sebanyak 0.25 yang disimpan dalam cbrtest.txt yang nantinya akan digunakan pada saat simulasi NS-2.

```
ns cbrgen.tcl -type CBR –nn 50 –seed 1.0 –mc 1 –rate
0.25 > cbr1
```

Gambar 4.5 Implementasi Koneksi cbrgen.tcl

Kemudian Gambar 4.6 menunjukkan *output* yang disimpan dalam cbr1 sehingga menghasilkan koneksi CBR dan menggunakan *Agent* UDP. Koneksi UDP di sini merupakan konfigurasi antara *node* ke-1 dan 2. Interval pengiriman paket data dilatkukan setiap satu detik dengan besar paket data 512 byte dan maksimal pengiriman paket 10.000.

```
#
# nodes: 50, max conn: 1, send rate: 1, seed: 1
#
#
# 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"
#
#Total sources/connections: 1/1
#
```

Gambar 4.6 *Output* cbr1

4.3. Implementasi Simulasi pada NS-2

Untuk mensimulasikan MANET dalam lingkungan NS-2, skrip ‘tcl’ yang telah ada pada *pacth* protokol *routing* AODV. Untuk Skenario *node-movement (mobility generation)* dan *traffic-connection generation* diberikan parameter-parameter yang sesuai dengan perancangan agar dapat dijalankan pada NS-2 . Parameter-parameter tersebut dibuat menggunakan bahasa Tcl/Otcl.

Pada Gambar 4.7 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. Tipe *Mac* yang digunakan adalah “Mac 802.11”. Pada baris diatas juga dilakukan konfigurasi tipe *queue* dari *interface*, tipe *link layer*, tipe *antenna* dan jumlah maksimal *packet* pada *interface queue*. Baris ke-9 sampai baris ke-17 berturut-turut merupakan koordinat x serta koordinat y sebesar 1000 meter, jumlah paket 50 dan jumlah *node* yaitu 60 *node*, protokol *routing* yang digunakan yaitu AODV, besarnya *seed* yaitu 0.0, waktu simulasi diakhiri pada detik ke-200, *file traffic-connection* yang digunakan yaitu *cbr1* dan terakhir ialah *file* skenario *node-movement (mobility generation)* yang digunakan adalah *file* "scenario-s5-n120-5".

```
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)              1000;
set opt(y)              1000;
set val(ifqlen)         50;
set val(nn)             120
set val(seed)           0.0
set val(adhocRouting)       AODV
set val(stop)               200;
set val(cp)                 "cbr1";
set val(sc)                 "scenario-s5-n120-5";
```

Gambar 4.7 Konfigurasi awal parameter NS-2

Pada Gambar 4.8 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 adalah sejauh 100 meter.

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

Gambar 4.8 Konfigurasi *Transmission Range* pada NS-2

```
set ns_       [new Simulator]

set topo      [new Topography]

set tracefd   [open AODV_$val(sc).tr w]

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

set topo             [new Topography]
$topo load_flatgrid $opt(x) $opt(y)

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

$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;
}
```

Gambar 4.9 Konfigurasi *Trace File* dan Pergerakan *Node* pada NS-2

Skrip yang ditunjukkan pada Gambar 4.9 merupakan skrip untuk pengaturan variabel global yang diawali dengan *set ns* untuk pembuatan simulator baru. *Set tracefd* dan *set namtrace* merupakan

pengaturan untuk nama *trace file* berekstensi '.tr' akan dihasilkan dan disimpan.

Pada *set tracefd* dapat dilakukan pengaturan untuk menghasilkan *trace file* sesuai dengan keinginan pengguna. *Set topo* adalah pengaturan untuk objek topografi berdasarkan pada luas koordinat yang sebelumnya telah dikonfigurasi. *Create-god* dan *node-config -channelType* adalah 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 4.7 seperti tipe *link layer*, tipe mac dan tipe transmisi. Terakhir dilakukan perulangan untuk membuat pergerakan dari *node-node*. *Node-node* yang dibuat tidak dapat melakukan pergerakan secara acak karena pergerakan *node* merupakan *trace file* yang dihasilkan oleh *mobility generator*.

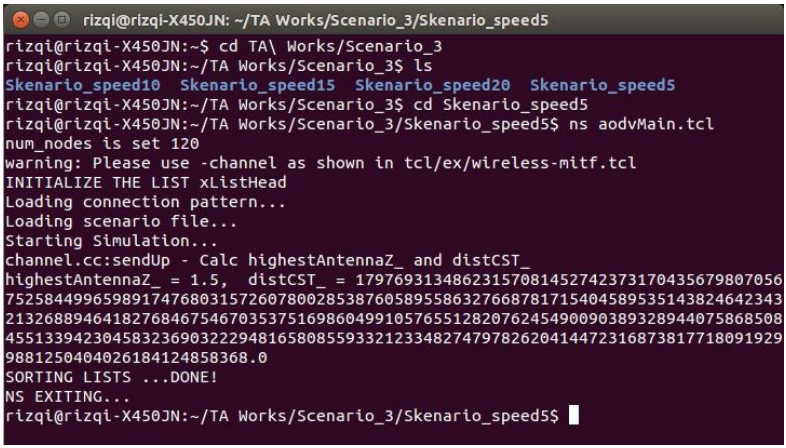
Skrip pada Gambar 4.10 merupakan bagian akhir dari keseluruhan skrip yang digunakan untuk menginisialisasi penempatan awal *node-node* yang dibut pada skenario *node-movement (mobility generation)*, pergerakan *node* tersebut selama waktu simulasi dilakukan dan melakukan konfigurasi pengiriman paket 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-200 seperti yang telah dikonfigurasi sebelumnya pada Gambar 4.7.

```
puts "Loading connection pattern..."
source $val(cp)

puts "Loading scenario file..."
source $val(sc)

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

for {set i 0} {$i < $val(nn) } {incr i} {
    $ns_ at $val(stop).0 "$node_($i) reset";
}

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

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

puts "Starting Simulation..."
$ns  run
```

Gambar 4.10 Konfigurasi pengiriman paket data NS-2

Pada Gambar 4.11 merupakan contoh *running* / eksekusi *file* aodv_Main.tcl.



Gambar 4.11 Perintah Eksekusi aodv_Main.tcl

Hasil yang diperoleh dari *running*/eksekusi berkas .tcl tersebut adalah *trace file* berekstensi '.tr'. *Trace file* '.tr' inilah yang akan digunakan untuk menganalisis parameter-parameter berupa *Packet Delivery Ratio* (PDR), *End-to-end Delay* (E2D), dan *Routing* Overhead (RO).

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

Psedeucode untuk *Packet Delivery Ratio* ditunjukkan pada Gambar 4.12 dan implementasinya dapat dilihat pada Lampiran.

```
BEGIN {
sent=0;
recv=0;
pdr=0;
}
{
#count packet send
if ($1 == "s" && $3 == "_1_" && $4 == "AGT" && $7
== "cbr")
       {
       sent++;
       }
```

```
#count packet receive
if ($1 == "r" && $3 == "_2_" && $4 == "AGT" && $7
== "cbr")
       {
       recv++;
       }
}
END {
pdr = ( recv / sent ) * 100
print "Transmitted packet (s):", sent;
print "Received packet (s):", recv;
print "Packet delivery ratio:", pdr, "%";
}
```

Gambar 4.12 *Pseudeucode* PDR

Contoh perintah untuk analisis PDR dari *trace file* dengan protokol AODV dengan *node* berjumlah 60 dan kecepatan maksimal perpindahan *node* sebesar 5 m/s adalah sebagai berikut :

```
"awk -f PacketDeliveryRatio.awk
AODV_scenario-s5-n60-1.tr"
```

Contoh *output* dari menjalankan skrip tersebut ditunjukkan pada Gambar 4.13.

```
Transmitted packet(s) =  201
Received packet(s) =  175
Packet Delivery Ratio =  87.06468 %
```

Gambar 4.13 Hasil *Running* skrip PacketDeliveryRatio.awk

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

Perhitungan *End-to-end Delay* 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. *Psedeucode* E2D ditunjukkan pada Gambar 4.14 dan implementasinya dapat dilihat pada Lampiran.

```
BEGIN{
for ( i in pkt_id)
     {
               pkt_id[i] = 0;
          }
for ( i in pkt_sent)
     {
               pkt_sent[i] = 0;
          }
for ( i in pkt_recv )
     {
               pkt_recv[i] = 0;
          }
     delay = avg_delay = 0;
     recv = 0;
     recv_id = 0;
}
{
# count packet send
if ( $1 == "s" && $3 == "_1_" && $4 == "AGT" && $7
== "cbr" )
{
  pkt_sent[$6] = $2;
}
# count packet receive
if ( $1 == "r" && $3 == "_2_" && $4 == "AGT" && $7
== "cbr" &&  recv_id != $6 )
{
  recv++;
  recv_id = $6;
  pkt_recv[$6] = $2;
}
}
END{
for (i in pkt_recv)
    {
      delay += pkt_recv[i] - pkt_sent[i];
    }

vg_delay = delay / recv;
```

```
print "Total Packet(s) Receive =", recv;
print "Total Delay =", delay, "second";
print "Average Packet Delivery Delay = ",
avg_delay, "second";
}
```

Gambar 4.14 *Pseudeucode* E2D

Contoh perintah untuk analisis *End-to-end Delay* dari *trace file* dengan protokol AODV dengan *node* berjumlah 60 dan kecepatan maksimal perpindahan *node* sebesar 5 m/s adalah sebagai berikut :

```
"awk -f EndtoEndDelay.awk AODV_scenario-s5-
n60-1.tr"
```

Berikut adalah hasil dari perintah untuk analisis *End-to-end Delay* di atas :

```
Total Packet(s) Receive = 174
Total Delay = 104.872 second
Average Packet Delivery Delay =  0.6024828 second
```

Gambar 4.15 Hasil *Running* skrip EndtoEndDelay.awk

4.4.3. *Routing Overhead* (RO)

Implementasi perhitungan metrik *Routing overhead* AODV dihitung apabila kondisi-kondisi yang ada terpenuhi yatu 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 "AODV" yang berarti paket *routing* AODV. seperti pada Gambar 4.16. Implementasinya dapat dilihat pada Lampiran.

```
BEGIN {
rt_pkts = 0;
}
{
if (($1 == "s" || $1 == "f") && ($4 == "RTR") && ($7
== "AODV"))
   rt_pkts++;
}
END {
printf ("Total number of routing packets\t%d\n",
rt_pkts);
}
```

Gambar 4.16 *Pseudeucode* RO

Contoh perintah untuk analisis RO dari *trace file* dengan protokol AODV dengan *node* berjumlah 60 dan kecepatan maksimal perpindahan *node* sebesar 5 m/s adalah sebagai berikut :

```
"awk -f RoutingOverhead.awk AODV_scenario-s5-
n60-1.tr"
```

Berikut adalah hasil dari perintah untuk analisis *End-to-end Delay* di atas :

```
Total number of routing packets 1559
```

Gambar 4.17 Hasil *running* skrip RoutingOverhead.awk

BAB V
PENGUJIAN DAN EVALUASI

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

5.1 Lingkungan Pengujian

Uji coba dilakukan pada laptop yang telah terpasang dua buah sistem operasi yaitu Windows dan Linux. Spesifikasi komputer yang digunakan untuk lingkungan pengujian ditunjukkan pada Tabel 5.1.

Tabel 5.1 Lingkungan Pengujian

Komponen	**Spesifikasi**
CPU	Intel(R) Core(TM) i7-4710HQ (8 CPUs) @2.5 GHz
Sistem Operasi	Windows 8.1 64-bit, Linux Ubuntu 14.045 64-bit (NS-2, *Mobility Generation, Traffic-Connection Generation*,)
Memori	8 GB DDR3
Media Penyimpanan	500 GB

5.2 Kriteria Pengujian

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

Tabel 5.2 Kriteria Pengujian

Kriteria	**Spesifikasi**
Skenario	MANET (*Random Way Point*), *mobility generator*
Kecepatan Maksimal Perpindahan *Node* (m/s)	5, 10, 15, 20
Jumlah Percobaan	160 kali
Jarak *Node* 1 dan *Node* 2	Acak
Posisi Awal *Node*	Acak
Pergerakan	Acak
Protokol *Routing*	AODV
Pengiriman Paket Data	100 – 200 detik

5.3. *Analisis Packet Delivery Ratio (PDR)*

Packet Delivery Ratio didapatkan dari file berekstensi ".tr" yang sebelumnya didapatkan dari file "aodv_Main.tcl" kemudian menjalankan *script* PacketDeliveryRatio.awk. Hasil tiap perhitungan PDR pada masing-masing skenario ditabulasikan dan dirata-ratakan menjadi seperti pada Tabel 5.3 dan Tabel 5.4

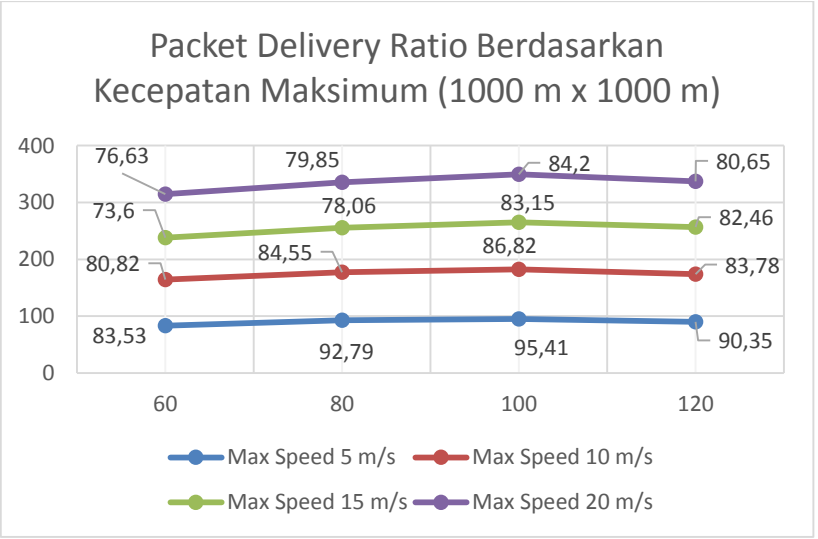
Tabel 5.3 Rata-rata *Packet Delivery Ratio* pada luas area $1000^2 m^2$

Max Speed (m/s)	Percentage by *Nodes* (%)			
	60	80	100	120
5	83,53	92,79	95,41	90,35
10	80,82	84,55	86,82	83,78
15	73,60	78,06	83,15	82,46
20	76,63	79,85	84,20	80,65

Tabel 5.4 Rata-rata *Packet Delivery Ratio* pada luas area $1500^2 m^2$

Max Speed (m/s)	Percentage by *Nodes* (%)			
	60	80	100	120
5	0,47	5,08	1,69	1,10
10	1,60	2,04	1,81	2,55
15	2,47	2,56	2,42	2,66
20	2,63	2,33	4,17	2,42

Pada Tabel 5.3 dan Tabel 5.4 menunjukkan hasil rata-rata PDR yang dihasilkan pada jaringan MANET berdasarkan rata-rata yang diperoleh dari masing-masing 5 skenario pada masing-masing variasi kecepatan maksimum dan jumlah *node*. Secara keseluruhan, pada area dengan luas $1000^2\ m^2$ terlihat bahwa rata-rata PDR yang dihasilkan mengalami peningkatan seiring dengan penambahan jumlah *node* dari 60 *node* hingga 100 *node*, tetapi mengalami penurunan saat jumlah *node* ditambah menjadi 120. Pada area dengan luas $1500^2\ m^2$, rata-rata PDR yang dihasilkan memiliki pola yang fluktuatif kecuali pada variasi kecepatan maksimum 5 m/s, di mana peningkatan hanya terjadi dari jumlah *node* 60 hingga 80 namun selebihnya mengalami penurunan hingga jumlah *node* 120.



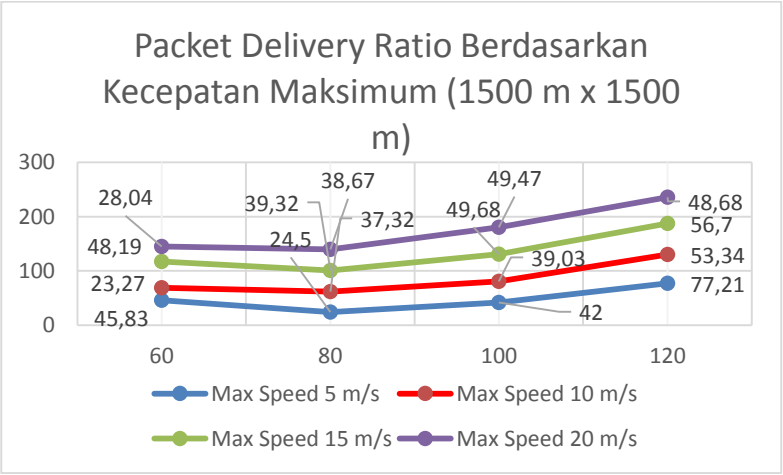
Gambar 5.1 Rata-rata PDR Berdasarkan Kecepatan Maksimum pada Luas Area 1000m x 1000m

Pada area dengan luas 1000 m x 1000 m, PDR yang dihasilkan pada kecepatan maksimum 5 m/s pada jumlah *node* 60 adalah 83,53. Saat jumlah *node* ditambah menjadi 80, PDR yang dihasilkan mengalami peningkatan sebesar 11,1% menjadi 92,79. Saat jumlah *node* kembali ditambah menjadi 100, PDR yang dihasilkan kembali meningkat sebesar 3% menjadi 95,41. Namun saat jumlah *node* ditambah menjadi 120, PDR yang dihasilkan mengalami penurunan sebesar 5,3% menjadi 90,35.

Dengan kecepatan maksimum 10 m/s, PDR yang dihasilkan pada jumlah *node* 60 adalah 80,82. Saat jumlah *node* ditambah menjadi 80, PDR yang dihasilkan mengalami peningkatan sebesar 4,6% menjadi 84,55. Saat jumlah *node* kembali ditambah menjadi 100, PDR yang dihasilkan kembali meningkat sebesar 3% menjadi 86,82. Namun saat jumlah *node* ditambah menjadi 120, PDR yang dihasilkan mengalami penurunan sebesar 4% menjadi 83,78.

Dengan kecepatan maksimum 15 m/s, PDR yang dihasilkan pada jumlah *node* 60 adalah 73,6. Saat jumlah *node* ditambah menjadi 80, PDR yang dihasilkan mengalami peningkatan sebesar 6,1% menjadi 78,06. Saat jumlah *node* kembali ditambah menjadi 100, PDR yang dihasilkan kembali meningkat sebesar 7% menjadi 83,15. Namun saat jumlah *node* ditambah menjadi 120, PDR yang dihasilkan mengalami penurunan sebesar 1% menjadi 82,46.

Dengan kecepatan maksimum 20 m/s, PDR yang dihasilkan pada jumlah *node* 60 adalah 80,82. Saat jumlah *node* ditambah menjadi 80, PDR yang dihasilkan mengalami peningkatan sebesar 4,2% menjadi 84,55. Saat jumlah *node* kembali ditambah menjadi 100, PDR yang dihasilkan kembali meningkat sebesar 5% menjadi 86,82. Namun saat jumlah *node* ditambah menjadi 120, PDR yang dihasilkan mengalami penurunan sebesar 4% menjadi 83,78.



Gambar 5.2 Rata-rata PDR Berdasarkan Kecepatan Maksimum pada Luas Area 1500m x 1500m

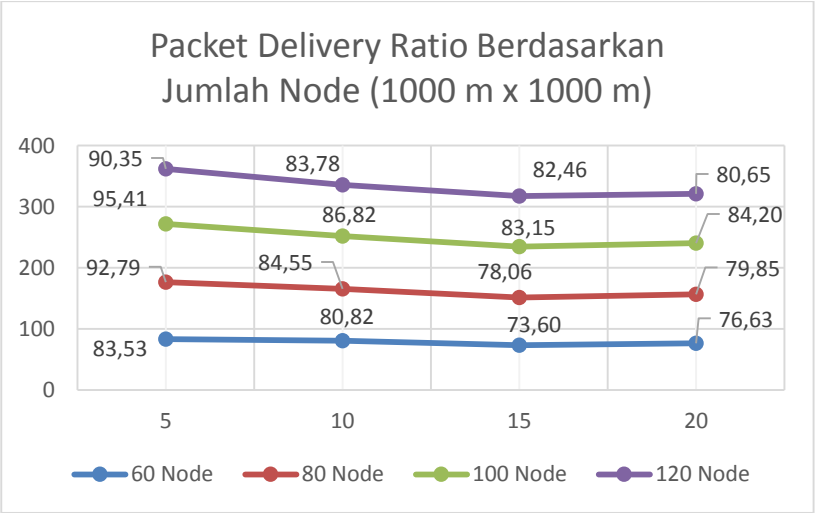
Pada area dengan luas 1500 m x 1500 m, PDR yang dihasilkan pada kecepatan maksimum 5 m/s pada jumlah *node* 60

adalah 45,83. Saat jumlah *node* ditambah menjadi 80, PDR yang dihasilkan mengalami penurunan sebesar 47% menjadi 24,5. Saat jumlah *node* kembali ditambah menjadi 100, PDR yang dihasilkan kembali meningkat sebesar 71% menjadi 42. Kemudia saat jumlah *node* ditambah menjadi 120, PDR yang dihasilkan mengalami peningkatan sebesar 84% menjadi 77,21.

Dengan kecepatan maksimum 10 m/s, PDR yang dihasilkan pada jumlah *node* 60 adalah 23,27. Saat jumlah *node* ditambah menjadi 80, PDR yang dihasilkan mengalami peningkatan sebesar 60% menjadi 37,32. Saat jumlah *node* kembali ditambah menjadi 100, PDR yang dihasilkan kembali meningkat sebesar 5% menjadi 39,03. Kemudian pada saat jumlah *node* ditambah menjadi 120, PDR yang dihasilkan mengalami peningkatan sebesar 37% menjadi 53,34.

Dengan kecepatan maksimum 15 m/s, PDR yang dihasilkan pada jumlah *node* 60 adalah 48,19. Saat jumlah *node* ditambah menjadi 80, PDR yang dihasilkan mengalami penurunan sebesar 18% menjadi 39,32. Saat jumlah *node* kembali ditambah menjadi 100, PDR yang dihasilkan meningkat sebesar 26% menjadi 49,68. Kemudian saat jumlah *node* ditambah menjadi 120, PDR yang dihasilkan mengalami peningkatan sebesar 14% menjadi 56,7.

Dengan kecepatan maksimum 20 m/s, PDR yang dihasilkan pada jumlah *node* 60 adalah 28,04. Saat jumlah *node* ditambah menjadi 80, PDR yang dihasilkan mengalami peningkatan sebesar 38% menjadi 38,67. Saat jumlah *node* kembali ditambah menjadi 100, PDR yang dihasilkan kembali meningkat sebesar 28% menjadi 49,47. Namun saat jumlah *node* ditambah menjadi 120, PDR yang dihasilkan mengalami penurunan sebesar 2% menjadi 48,68.



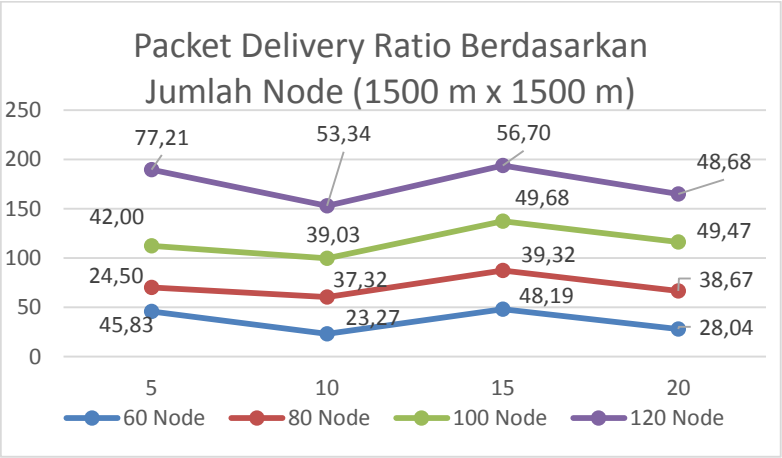
Gambar 5.3 Rata-rata PDR Berdasarkan Jumlah *Node* pada Luas Area 1000m x 1000m

Pada area dengan luas 1000 m x 1000 m, PDR yang dihasilkan pada jumlah *node* 60 dengan kecepatan maksimum 5 m/s adalah 83,53. Saat kecepatan maksimum ditambah menjadi 10 m/s, PDR yang dihasilkan mengalami penurunan sebesar 3,2% menjadi 80,82. Saat kecepatan maksimum kembali ditambah menjadi 15 m/s, PDR yang dihasilkan kembali menurun sebesar 8,9% menjadi 73,6. Namun saat kecepatan maksimum ditambah menjadi 20 m/s, PDR yang dihasilkan mengalami peningkatan sebesar 4,1% menjadi 76,63.

Dengan jumlah *node* 80, PDR yang dihasilkan pada kecepatan maksimum 5 m/s adalah 92,79. Saat kecepatan maksimum ditambah menjadi 10 m/s, PDR yang dihasilkan mengalami penurunan sebesar 8,9% menjadi 84,55. Saat kecepatan maksimum kembali ditambah menjadi 15 m/s, PDR yang dihasilkan kembali menurun sebesar 7,7% menjadi 78,06. Namun saat kecepatan maksimum ditambah menjadi 20 m/s, PDR yang dihasilkan mengalami peningkatan sebesar 2,3% menjadi 79,85.

Dengan jumlah *node* 100, PDR yang dihasilkan pada kecepatan maksimum 5 m/s adalah 95,41. Saat kecepatan maksimum ditambah menjadi 10 m/s, PDR yang dihasilkan mengalami peningkatan sebesar 9% menjadi 86,82. Saat kecepatan maksimum kembali ditambah menjadi 15 m/s, PDR yang dihasilkan kembali meningkat sebesar 4,2% menjadi 83,15. Namun saat kecepatan maksimum ditambah menjadi 20 m/s, PDR yang dihasilkan mengalami penurunan sebesar 1,3% menjadi 84,2.

Dengan jumlah *node* 120, PDR yang dihasilkan pada kecepatan maksimum 5 m/s adalah 90,35. Saat kecepatan maksimum ditambah menjadi 10 m/s, PDR yang dihasilkan mengalami penurunan sebesar 7,3% menjadi 83,78. Saat kecepatan maksimum kembali ditambah menjadi 15 m/s, PDR yang dihasilkan kembali menurun sebesar 1,6% menjadi 82,46. Namun saat kecepatan maksimum ditambah menjadi 20 m/s, PDR yang dihasilkan mengalami peningkatan sebesar 2,2% menjadi 80,65.



Gambar 5.4 Rata-rata PDR Berdasarkan Jumlah *Node* pada Luas Area 1500m x 1500m

Pada area dengan luas 1500 m x 1500 m, PDR yang dihasilkan pada jumlah *node* 60 dengan kecepatan maksimum 5

m/s adalah 45,83. Saat kecepatan maksimum ditambah menjadi 10 m/s, PDR yang dihasilkan mengalami penurunan sebesar 49% menjadi 23,27. Namun saat kecepatan maksimum kembali ditambah menjadi 15 m/s, PDR yang dihasilkan meningkat sebesar 107% menjadi 48,19. Kemudian saat kecepatan maksimum ditambah menjadi 20 m/s, PDR yang dihasilkan mengalami penurunan sebesar 42% menjadi 28,04.

Dengan jumlah *node* 80, PDR yang dihasilkan pada kecepatan maksimum 5 m/s adalah 24,50. Saat kecepatan maksimum ditambah menjadi 10 m/s, PDR yang dihasilkan mengalami peningkatan sebesar 52% menjadi 37,32. Saat kecepatan maksimum kembali ditambah menjadi 15 m/s, PDR yang dihasilkan kembali meningkat sebesar 5% menjadi 39,32. Namun saat kecepatan maksimum ditambah menjadi 20 m/s, PDR yang dihasilkan mengalami penurunan sebesar 2% menjadi 38,67.

Dengan jumlah *node* 100, PDR yang dihasilkan pada kecepatan maksimum 5 m/s adalah 42,00. Saat kecepatan maksimum ditambah menjadi 10 m/s, PDR yang dihasilkan mengalami penurunan sebesar 7% menjadi 39,03. Namun saat kecepatan maksimum kembali ditambah menjadi 15 m/s, PDR yang dihasilkan meningkat sebesar 27% menjadi 49,68. Pada saat kecepatan maksimum ditambah menjadi 20 m/s, PDR yang dihasilkan tidak mengalami perubahan, tetap 49,47.

Dengan jumlah *node* 120, PDR yang dihasilkan pada kecepatan maksimum 5 m/s adalah 77,21. Saat kecepatan maksimum ditambah menjadi 10 m/s, PDR yang dihasilkan mengalami penurunan sebesar 31% menjadi 53,34. Namun saat kecepatan maksimum kembali ditambah menjadi 15 m/s, PDR yang dihasilkan meningkat sebesar 6% menjadi 56,7. Kemudian pada saat kecepatan maksimum ditambah menjadi 20 m/s, PDR yang dihasilkan mengalami penurunan sebesar 14% menjadi 48,68.

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

End-to-end Delay didapatkan dari file berekstensi ".tr" yang sebelumnya didapatkan dari file "aodv_Main.tcl" kemudian menjalankan *script* EndtoEndDelay.awk. Hasil tiap perhitungan E2D pada masing-masing skenario ditabulasikan dan dirata-ratakan menjadi seperti pada Tabel 5.5 dan Tabel 5.6.

Tabel 5.5 Rata-rata *End-to-end Delay* pada luas area $1000^2 m^2$

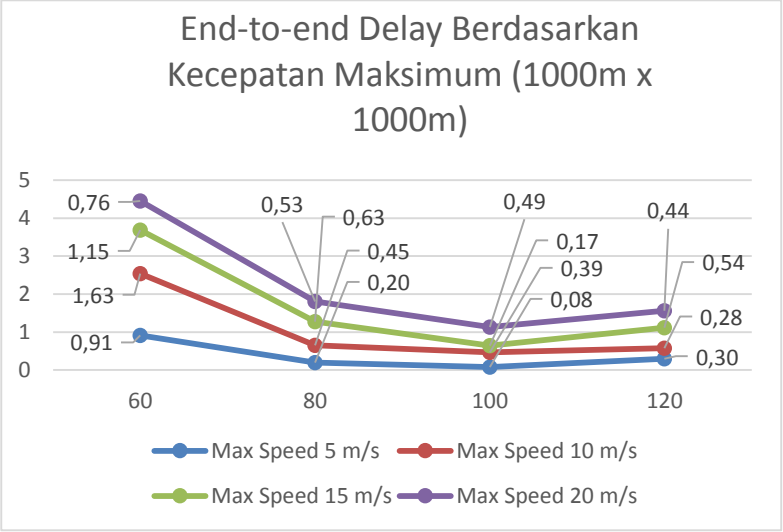
Max Speed (m/s)	Average Delay by Nodes (s)			
	60	80	100	120
5	0,91	0,20	0,08	0,30
10	1,63	0,45	0,39	0,28
15	1,15	0,63	0,17	0,54
20	0,76	0,53	0,49	0,44

Tabel 5.6 Rata-rata *End-to-end Delay* pada luas area $1500^2 m^2$

Max Speed (m/s)	Average Delay by Nodes (s)			
	60	80	100	120
5	0,47	5,08	1,69	1,10
10	1,60	2,04	1,81	2,55
15	2,47	2,56	2,42	2,66
20	2,63	2,33	4,17	2,42

Pada Tabel 5.5 dan Tabel 5.6 menunjukkan hasil rata-rata E2D yang dihasilkan pada jaringan MANET berdasarkan rata-rata yang diperoleh dari masing-masing 5 skenario pada masing-masing variasi kecepatan maksimum dan jumlah *node*. Secara keseluruhan, pada area dengan luas $\mathbf{1000^2\ m^2}$ terlihat bahwa rata-rata E2D yang dihasilkan cenderung semakin berkurang seiring dengan penambahan jumlah *node* namun mengalami peningkatan pada saat mencapa jumlah *node* tertentu. Pada area dengan luas $\mathbf{1500^2\ m^2}$, rata-rata E2D yang dihasilkan bersifat fluktuatif di

mana masing-masing hasil mengalami peningkatan dan penurunan di titik-titik tertentu.



Gambar 5.5 Rata-rata E2D Berdasarkan Kecepatan Maksimum pada Luas Area 1000m x 1000m

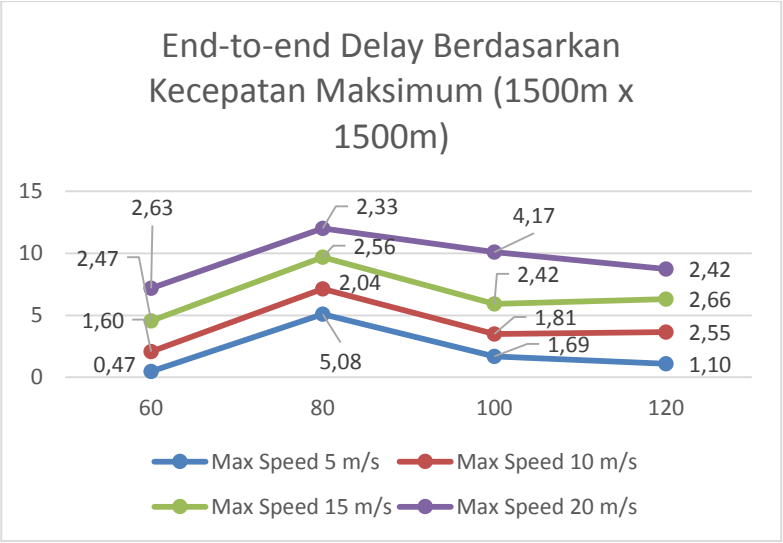
Pada area dengan luas 1000 m x 1000 m, E2D yang dihasilkan pada kecepatan maksimum 5 m/s pada jumlah *node* 60 adalah 0,91. Saat jumlah *node* ditambah menjadi 80, E2D yang dihasilkan mengalami penurunan sebesar 78% menjadi 0,2. Saat jumlah *node* kembali ditambah menjadi 100, E2D yang dihasilkan kembali menurun sebesar 60% menjadi 0,08. Namun saat jumlah *node* ditambah menjadi 120, E2D yang dihasilkan mengalami penurunan sebesar 275% menjadi 0,3.

Dengan kecepatan maksimum 10 m/s, E2D yang dihasilkan pada jumlah *node* 60 adalah 1,63. Saat jumlah *node* ditambah menjadi 80, E2D yang dihasilkan mengalami penurunan sebesar 72,4% menjadi 0,45. Saat jumlah *node* kembali ditambah menjadi 100, E2D yang dihasilkan kembali menurun sebesar 13%

menjadi 0,39. Kemudian saat jumlah *node* ditambah menjadi 120, E2D yang dihasilkan mengalami penurunan sebesar 28% menjadi 0,28.

Dengan kecepatan maksimum 15 m/s, E2D yang dihasilkan pada jumlah *node* 60 adalah 1,15. Saat jumlah *node* ditambah menjadi 80, E2D yang dihasilkan mengalami penurunan sebesar 45,2% menjadi 0,63. Saat jumlah *node* kembali ditambah menjadi 100, E2D yang dihasilkan kembali menurun sebesar 73% menjadi 0,17. Namun saat jumlah *node* ditambah menjadi 120, E2D yang dihasilkan mengalami peningkatan sebesar 217,6% menjadi 0,54.

Dengan kecepatan maksimum 20 m/s, E2D yang dihasilkan pada jumlah *node* 60 adalah 0,76. Saat jumlah *node* ditambah menjadi 80, E2D yang dihasilkan mengalami penurunan sebesar 30,3% menjadi 0,53. Saat jumlah *node* kembali ditambah menjadi 100, E2D yang dihasilkan kembali menurun sebesar 8% menjadi 0,49. Kemudian saat jumlah *node* ditambah menjadi 120, E2D yang dihasilkan mengalami penurunan sebesar 1% menjadi 044.



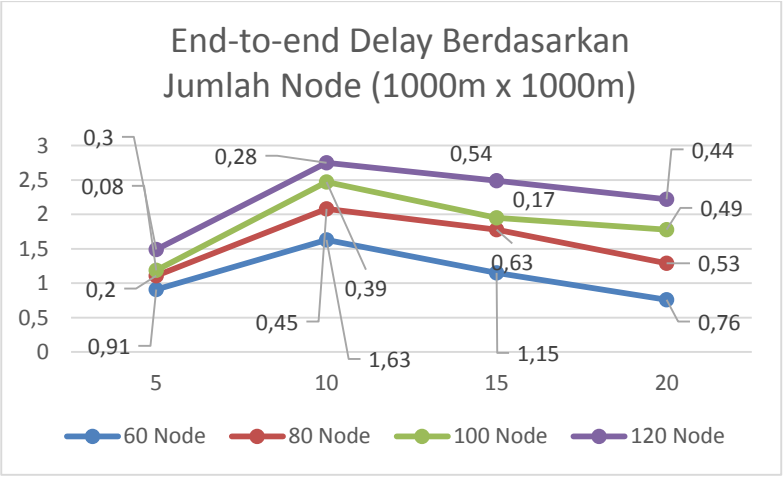
Gambar 5.6 Rata-rata E2D Berdasarkan Kecepatan Maksimum pada Luas Area 1500m x 1500m

Pada area dengan luas 1500 m x 1500 m, E2D yang dihasilkan pada kecepatan maksimum 5 m/s pada jumlah *node* 60 adalah 0,47. Saat jumlah *node* ditambah menjadi 80, E2D yang dihasilkan mengalami peningkatan sebesar 981% menjadi 5,08. Saat jumlah *node* kembali ditambah menjadi 100, E2D yang dihasilkan mengalami penurunan sebesar 67% menjadi 1,69. Kemudia saat jumlah *node* ditambah menjadi 120, E2D yang dihasilkan mengalami penurunan sebesar 35% menjadi 1,1.

Dengan kecepatan maksimum 10 m/s, E2D yang dihasilkan pada jumlah *node* 60 adalah 1,6. Saat jumlah *node* ditambah menjadi 80, E2D yang dihasilkan mengalami peningkatan sebesar 28% menjadi 2,04. Saat jumlah *node* kembali ditambah menjadi 100, E2D yang dihasilkan menurun sebesar 11% menjadi 1,81. Kemudian pada saat jumlah *node* ditambah menjadi 120, E2D yang dihasilkan mengalami peningkatan sebesar 41% menjadi 2,55.

Dengan kecepatan maksimum 15 m/s, E2D yang dihasilkan pada jumlah *node* 60 adalah 2,47. Saat jumlah *node* ditambah menjadi 80, E2D yang dihasilkan mengalami penurunan sebesar 4% menjadi 2,56. Saat jumlah *node* kembali ditambah menjadi 100, E2D yang dihasilkan menurun sebesar 5% menjadi 2,42. Kemudian saat jumlah *node* ditambah menjadi 120, E2D yang dihasilkan mengalami peningkatan sebesar 10% menjadi 2,66.

Dengan kecepatan maksimum 20 m/s, E2D yang dihasilkan pada jumlah *node* 60 adalah 2,63. Saat jumlah *node* ditambah menjadi 80, E2D yang dihasilkan mengalami penurunan sebesar 11% menjadi 2,33. Saat jumlah *node* kembali ditambah menjadi 100, E2D yang dihasilkan mengalami meningkat sebesar 79% menjadi 4,17. Namun saat jumlah *node* ditambah menjadi 120, E2D yang dihasilkan mengalami penurunan sebesar 42% menjadi 2,42.



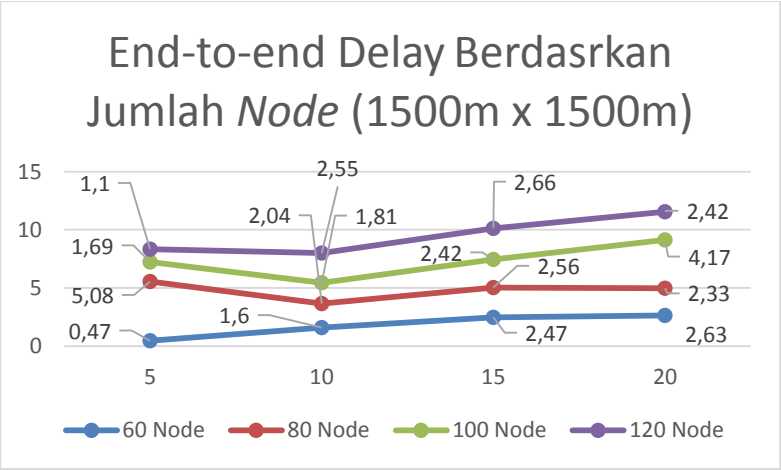
Gambar 5.7 Rata-rata E2D Berdasarkan Jumlah *Node* pada Luas Area 1000m x 1000m

Pada area dengan luas 1000 m x 1000 m, E2D yang dihasilkan pada jumlah *node* 60 dengan kecepatan maksimum 5 m/s adalah 0,91. Saat kecepatan maksimum ditambah menjadi 10 m/s, E2D yang dihasilkan mengalami peningkatan sebesar 79,1% menjadi 1,63. Namun saat kecepatan maksimum kembali ditambah menjadi 15 m/s, E2D yang dihasilkan menurun sebesar 29,4% menjadi 1,15. Kemudian pada saat kecepatan maksimum ditambah menjadi 20 m/s, E2D yang dihasilkan mengalami penurunan sebesar 33,9% menjadi 0,76.

Dengan jumlah *node* 80, E2D yang dihasilkan pada kecepatan maksimum 5 m/s adalah 0,2. Saat kecepatan maksimum ditambah menjadi 10 m/s, E2D yang dihasilkan mengalami peningkatan sebesar 125% menjadi 0,45. Saat kecepatan maksimum kembali ditambah menjadi 15 m/s, E2D yang dihasilkan kembali meningkat sebesar 40% menjadi 0,63. Namun saat kecepatan maksimum ditambah menjadi 20 m/s, E2D yang dihasilkan mengalami penurunan sebesar 15,9% menjadi 0,53.

Dengan jumlah *node* 100, E2D yang dihasilkan pada kecepatan maksimum 5 m/s adalah 0,08. Saat kecepatan maksimum ditambah menjadi 10 m/s, E2D yang dihasilkan mengalami peningkatan sebesar 387,5% menjadi 0,39. Namun saat kecepatan maksimum kembali ditambah menjadi 15 m/s, E2D yang dihasilkan mengalami penurunan sebesar 56,4% menjadi 0,17. Kemudian saat kecepatan maksimum ditambah menjadi 20 m/s, E2D yang dihasilkan mengalami penurunan sebesar 188,2% menjadi 0,49.

Dengan jumlah *node* 120, E2D yang dihasilkan pada kecepatan maksimum 5 m/s adalah 0,3. Saat kecepatan maksimum ditambah menjadi 10 m/s, E2D yang dihasilkan mengalami penurunan sebesar 6,7% menjadi 0,28. Saat kecepatan maksimum kembali ditambah menjadi 15 m/s, E2D yang dihasilkan mengalami penurunan sebesar 92,9% menjadi 0,54. Namun saat kecepatan maksimum ditambah menjadi 20 m/s, E2D yang dihasilkan mengalami peningkatan sebesar 18,5% menjadi 0,44.



Gambar 5.8 Rata-rata E2D Berdasarkan Jumlah *Node* pada Luas Area 1500m x 1500m

Pada area dengan luas 1500 m x 1500 m, E2D yang dihasilkan pada jumlah *node* 60 dengan kecepatan maksimum 5 m/s adalah 0,47. Saat kecepatan maksimum ditambah menjadi 10 m/s, E2D yang dihasilkan mengalami peningkatan sebesar 240% menjadi 1,6. Pada saat kecepatan maksimum kembali ditambah menjadi 15 m/s, E2D yang dihasilkan meningkat sebesar 54% menjadi 2,47. Kemudian saat kecepatan maksimum ditambah menjadi 20 m/s, E2D yang dihasilkan mengalami peningkatan sebesar 6% menjadi 2,63.

Dengan jumlah *node* 80, E2D yang dihasilkan pada kecepatan maksimum 5 m/s adalah 5,08. Saat kecepatan maksimum ditambah menjadi 10 m/s, E2D yang dihasilkan mengalami penurunan sebesar 60% menjadi 2,04. Saat kecepatan maksimum kembali ditambah menjadi 15 m/s, E2D yang dihasilkan meningkat sebesar 25% menjadi 2,56. Namun saat kecepatan maksimum ditambah menjadi 20 m/s, E2D yang dihasilkan mengalami penurunan sebesar 9% menjadi 2,33.

Dengan jumlah *node* 100, E2D yang dihasilkan pada kecepatan maksimum 5 m/s adalah 1,69. Saat kecepatan maksimum ditambah menjadi 10 m/s, E2D yang dihasilkan mengalami peningkatan sebesar 7% menjadi 1,81. Pada saat kecepatan maksimum kembali ditambah menjadi 15 m/s, E2D yang dihasilkan kembali meningkat sebesar 34% menjadi 2,42. Pada saat kecepatan maksimum ditambah menjadi 20 m/s, E2D yang dihasilkan mengalami peningkatan sebesar 72% menjadi 4,17.

Dengan jumlah *node* 120, E2D yang dihasilkan pada kecepatan maksimum 5 m/s adalah 1,1. Saat kecepatan maksimum ditambah menjadi 10 m/s, E2D yang dihasilkan mengalami peningkatan sebesar 132% menjadi 2,55. Pada saat kecepatan maksimum kembali ditambah menjadi 15 m/s, E2D yang dihasilkan meningkat sebesar 4% menjadi 2,66. Namun pada saat kecepatan maksimum ditambah menjadi 20 m/s, E2D yang dihasilkan mengalami penurunan sebesar 9% menjadi 2,42.

5.5. Analisis *Routing Overhead (RO)*

Routing Overhead didapatkan dari file berekstensi ".tr" yang sebelumnya didapatkan dari file "aodv_Main.tcl" kemudian menjalankan *script* RoutingOverhead.awk. Hasil tiap perhitungan RO pada masing-masing skenario ditabulasikan dan dirata-ratakan menjadi seperti pada Tabel 5.7 dan Tabel 5.8.

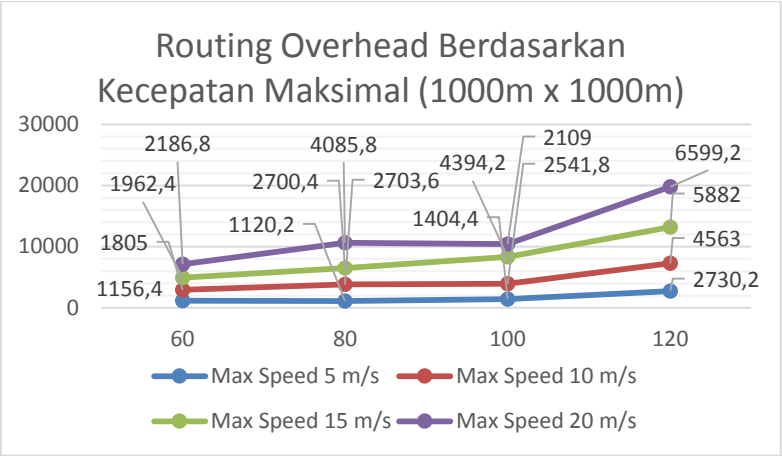
Tabel 5.7 Rata-rata *Routing Overhead* pada luas area $1000^2 m^2$

Max Speed (m/s)	Amount of Packets by Nodes (packets)			
	60	80	100	120
5	1156,4	1120,2	1404,4	2730,2
10	1805	2703,6	2541,8	4563
15	1962,4	2700,4	4394,2	5882
20	2186,8	4085,8	2109	6599,2

Tabel 5.8 Rata-rata *Routing Overhead* pada luas area $1500^2 m^2$

Max Speed (m/s)	Amount of Packets by Nodes (packets)			
	60	80	100	120
5	301	618,4	1535	2143,6
10	339	942	1390,6	4508,2
15	300,2	1509,8	2427,6	4534,2
20	507,4	1282,6	2415,4	3486,6

Pada Tabel 5.7 dan Tabel 5.8 menunjukkan hasil rata-rata RO yang dihasilkan pada jaringan MANET berdasarkan rata-rata yang diperoleh dari masing-masing 5 skenario pada masing-masing variasi kecepatan maksimum dan jumlah *node*. Secara keseluruhan, pada area dengan luas $1000^2\ m^2$ terlihat bahwa rata-rata RO yang dihasilkan bersifat fluktuatif di mana masing-masing hasil mengalami peningkatan dan penurunan di titik-titik tertentu. Pada area dengan luas $1500^2\ m^2$ terlihat bahwa rata-rata RO yang dihasilkan mengalami peningkatan seiring dengan pertambahan jumlah *node*.



Gambar 5.9 Rata-rata RO Berdasarkan Kecepatan Maksimum pada Luas Area 1000m x 1000m

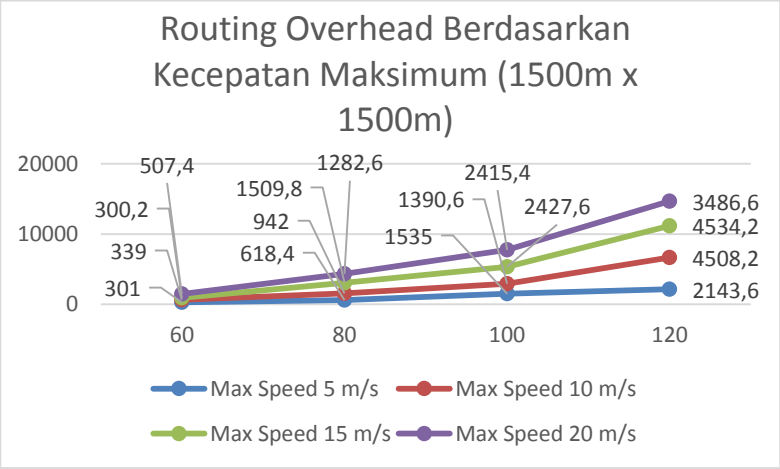
Pada area dengan luas 1000 m x 1000 m, RO yang dihasilkan pada kecepatan maksimum 5 m/s pada jumlah *node* 60 adalah 1156,4. Saat jumlah *node* ditambah menjadi 80, RO yang dihasilkan mengalami penurunan sebesar 3,1% menjadi 1120,2. Saat jumlah *node* kembali ditambah menjadi 100, RO yang dihasilkan kembali meningkat sebesar 25% menjadi 1404,2. Kemudian saat jumlah *node* ditambah menjadi 120, RO yang dihasilkan mengalami peningkatan sebesar 94,4% menjadi 2730,2.

Dengan kecepatan maksimum 10 m/s, RO yang dihasilkan pada jumlah *node* 60 adalah 1805. Saat jumlah *node* ditambah menjadi 80, RO yang dihasilkan mengalami peningkatan sebesar 49,8% menjadi 2703,6. Saat jumlah *node* kembali ditambah menjadi 100, RO yang dihasilkan mengalami penurunan sebesar 6% menjadi 2541,8. Namun saat jumlah *node* ditambah menjadi 120, RO yang dihasilkan mengalami peningkatan sebesar 79,5% menjadi 4563.

Dengan kecepatan maksimum 15 m/s, RO yang dihasilkan pada jumlah *node* 60 adalah 1968,2. Saat jumlah *node* ditambah menjadi 80, RO yang dihasilkan mengalami peningkatan sebesar

37,6% menjadi 2700,4. Saat jumlah *node* kembali ditambah menjadi 100, RO yang dihasilkan kembali meningkat sebesar 63% menjadi 4394,2. Pada saat jumlah *node* ditambah menjadi 120, RO yang dihasilkan meningkat sebesar 33,9% menjadi 5882.

Dengan kecepatan maksimum 20 m/s, RO yang dihasilkan pada jumlah *node* 60 adalah 2186,8. Saat jumlah *node* ditambah menjadi 80, RO yang dihasilkan mengalami peningkatan sebesar 86,8% menjadi 4085,8. Namun saat jumlah *node* kembali ditambah menjadi 100, RO yang dihasilkan menurun sebesar 48% menjadi 2109. Kemudian pada saat jumlah *node* ditambah menjadi 120, RO yang dihasilkan mengalami penurunan sebesar 212,9% menjadi 6599,2.



Gambar 5.10 Rata-rata RO Berdasarkan Kecepatan Maksimum pada Luas Area 1500m x 1500m

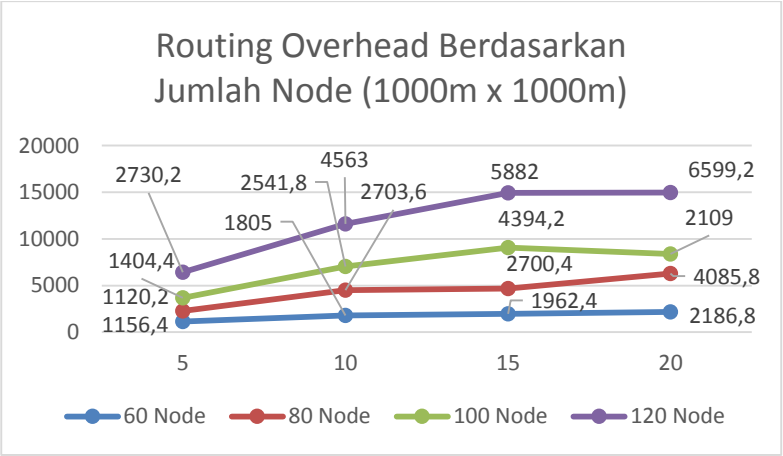
Pada area dengan luas 1500 m x 1500 m, RO yang dihasilkan pada kecepatan maksimum 5 m/s pada jumlah *node* 60 adalah 301. Saat jumlah *node* ditambah menjadi 80, RO yang dihasilkan mengalami peningkatan sebesar 105% menjadi 618,4.

Saat jumlah *node* kembali ditambah menjadi 100, RO yang dihasilkan kembali meningkat sebesar 148% menjadi 1535. Kemudia saat jumlah *node* ditambah menjadi 120, RO yang dihasilkan mengalami peningkatan sebesar 40% menjadi 2143,6.

Dengan kecepatan maksimum 10 m/s, RO yang dihasilkan pada jumlah *node* 60 adalah 339. Saat jumlah *node* ditambah menjadi 80, RO yang dihasilkan mengalami peningkatan sebesar 178% menjadi 942. Saat jumlah *node* kembali ditambah menjadi 100, RO yang dihasilkan kembali meningkat sebesar 48% menjadi 1390,6. Kemudian pada saat jumlah *node* ditambah menjadi 120, RO yang dihasilkan mengalami peningkatan sebesar 224% menjadi 4508,2.

Dengan kecepatan maksimum 15 m/s, RO yang dihasilkan pada jumlah *node* 60 adalah 300,2. Saat jumlah *node* ditambah menjadi 80, RO yang dihasilkan mengalami peningkatan sebesar 403% menjadi 1509,8. Saat jumlah *node* kembali ditambah menjadi 100, RO yang dihasilkan meningkat sebesar 61% menjadi 2427,6. Kemudian saat jumlah *node* ditambah menjadi 120, RO yang dihasilkan mengalami peningkatan sebesar 87% menjadi 4534,2.

Dengan kecepatan maksimum 20 m/s, RO yang dihasilkan pada jumlah *node* 60 adalah 507,4. Saat jumlah *node* ditambah menjadi 80, RO yang dihasilkan mengalami peningkatan sebesar 153% menjadi 1282,6. Saat jumlah *node* kembali ditambah menjadi 100, RO yang dihasilkan kembali meningkat sebesar 88% menjadi 2415,4. Kemudian pada saat jumlah *node* ditambah menjadi 120, RO yang dihasilkan mengalami peningkatan sebesar 44% menjadi 3486,6.



Gambar 5.11 Rata-rata RO Berdasarkan Jumlah *Node* pada Luas Area 1000m x 1000m

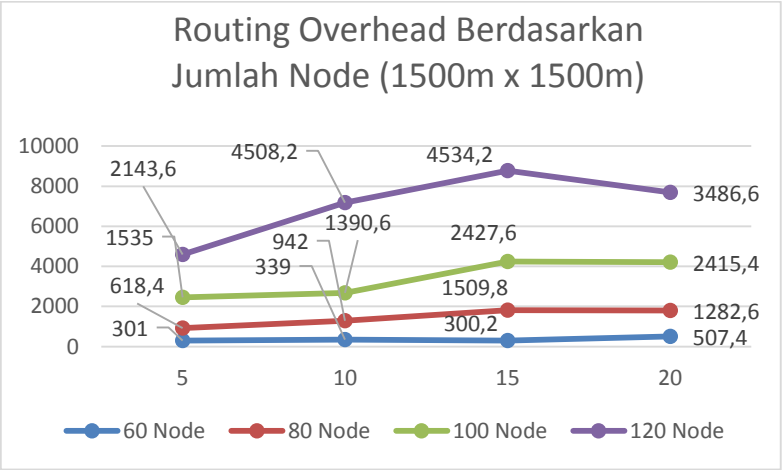
Pada area dengan luas 1000 m x 1000 m, RO yang dihasilkan pada jumlah *node* 60 dengan kecepatan maksimum 5 m/s adalah 1156,4. Saat kecepatan maksimum ditambah menjadi 10 m/s, RO yang dihasilkan mengalami peningkatan sebesar 56,1% menjadi 1805. Kemudian saat kecepatan maksimum kembali ditambah menjadi 15 m/s, RO yang dihasilkan meningkat sebesar 8,7% menjadi 1962,4. Pada saat kecepatan maksimum ditambah menjadi 20 m/s, RO yang dihasilkan mengalami peningkatan sebesar 11,4% menjadi 2186,8.

Dengan jumlah *node* 80, RO yang dihasilkan pada kecepatan maksimum 5 m/s adalah 1120,2. Saat kecepatan maksimum ditambah menjadi 10 m/s, RO yang dihasilkan mengalami peningkatan sebesar 141,3% menjadi 2703,6. Saat kecepatan maksimum kembali ditambah menjadi 15 m/s, RO yang dihasilkan menurun sebesar 1% menjadi 2700,4. Namun saat kecepatan maksimum ditambah menjadi 20 m/s, RO yang dihasilkan mengalami peningkatan sebesar 51,3% menjadi 4085,8.

Dengan jumlah *node* 100, RO yang dihasilkan pada kecepatan maksimum 5 m/s adalah 1404,4. Saat kecepatan

maksimum ditambah menjadi 10 m/s, RO yang dihasilkan mengalami peningkatan sebesar 81% menjadi 2541,8. Saat kecepatan maksimum kembali ditambah menjadi 15 m/s, RO yang dihasilkan kembali meningkat sebesar 72,9% menjadi 4394,2. Namun saat kecepatan maksimum ditambah menjadi 20 m/s, RO yang dihasilkan mengalami penurunan sebesar 52% menjadi 2109.

Dengan jumlah *node* 120, RO yang dihasilkan pada kecepatan maksimum 5 m/s adalah 2730,2. Saat kecepatan maksimum ditambah menjadi 10 m/s, RO yang dihasilkan mengalami peningkatan sebesar 7,3% menjadi 4563. Saat kecepatan maksimum kembali ditambah menjadi 15 m/s, RO yang dihasilkan kembali meningkat sebesar 28,9% menjadi 5882. Kemudian pada saat kecepatan maksimum ditambah menjadi 20 m/s, RO yang dihasilkan mengalami peningkatan sebesar 12,2% menjadi 6599,2.



Gambar 5.12 Rata-rata RO Berdasarkan Jumlah *Node* pada Luas Area 1500m x 1500m

Pada area dengan luas 1500 m x 1500 m, RO yang dihasilkan pada jumlah *node* 60 dengan kecepatan maksimum 5

m/s adalah 301. Saat kecepatan maksimum ditambah menjadi 10 m/s, RO yang dihasilkan mengalami peningkatan sebesar 13% menjadi 339. Namun saat kecepatan maksimum kembali ditambah menjadi 15 m/s, RO yang dihasilkan menurun sebesar 11% menjadi 300,2. Kemudian saat kecepatan maksimum ditambah menjadi 20 m/s, RO yang dihasilkan mengalami peningkatan sebesar 69% menjadi 507,4.

Dengan jumlah *node* 80, RO yang dihasilkan pada kecepatan maksimum 5 m/s adalah 618,4. Saat kecepatan maksimum ditambah menjadi 10 m/s, RO yang dihasilkan mengalami peningkatan sebesar 52% menjadi 942. Saat kecepatan maksimum kembali ditambah menjadi 15 m/s, RO yang dihasilkan kembali meningkat sebesar 6% menjadi 1509,8. Namun saat kecepatan maksimum ditambah menjadi 20 m/s, RO yang dihasilkan mengalami penurunan sebesar 15% menjadi 1282,6.

Dengan jumlah *node* 100, RO yang dihasilkan pada kecepatan maksimum 5 m/s adalah 1535. Saat kecepatan maksimum ditambah menjadi 10 m/s, RO yang dihasilkan mengalami penurunan sebesar 9% menjadi 1390,6. Namun saat kecepatan maksimum kembali ditambah menjadi 15 m/s, RO yang dihasilkan meningkat sebesar 75% menjadi 2427,6. Pada saat kecepatan maksimum ditambah menjadi 20 m/s, RO yang dihasilkan mengalami penurunan sebesar 1% menjadi 2415,4.

Dengan jumlah *node* 120, RO yang dihasilkan pada kecepatan maksimum 5 m/s adalah 2143. Saat kecepatan maksimum ditambah menjadi 10 m/s, RO yang dihasilkan mengalami peningkatan sebesar 110% menjadi 4508,2. Namun saat kecepatan maksimum kembali ditambah menjadi 15 m/s, RO yang dihasilkan meningkat sebesar 1% menjadi 4534,2. Namun pada saat kecepatan maksimum ditambah menjadi 20 m/s, RO yang dihasilkan mengalami penurunan sebesar 23% menjadi 3486,6.

[Halaman ini sengaja dikosongkan]

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 protokol *routing* AODV dengan penambahan kecepatan maksimal perpindahan *node* memiliki performa *Packet Delivery Ratio* sebagai berikut :
   - Dari kecepatan maksimal perpindahan *node* sebesar 5 m/s hingga 20 m/s, performa *Packet Delivery Ratio* yang dihasilkan pada area dengan luas 1000 m x 1000 m mengalami peningkatan saat jumlah *node* diperbanyak. Namun pada saat jumlah *node* melebihi 100, nilai performa tersebut mengalami penurunan. Untuk area dengan luas 1500 m x 1500 m, performa *Packet Delivery Ratio* yang dihasilkan cenderung beragam. Pada kecepatan maksimum 5 m/s dan 15 m/s, nilai *Packet Delivery Ratio* menurun pada saat jumlah *node* 80 kemudian meningkat hingga jumlah *node* 120. Pada kecepatan maksimum 10 m/s, nilai *Packet Delivery Ratio* terus meningkat hingga jumlah *node* 120. Pada kecepatan maksimum 20 m/s, nilai *Packet Delivery Ratio* terus meningkat hingga jumlah *node* 100 kemudian menurun pada jumlah *node* 120.
   - Dari jumlah *node* 60 hingga 120, performa *Packet Delivery Ratio* yang dihasilkan pada area dengan luas 1000 m x 1000 m mengalami penurunan seiring dengan peningkatan kecepatan maksimum. namun mengalami peningkatan

setelah kecepatan maksimum mencapai 20 m/s. Kecuali pada jumlah *node* 120, performa tersebut terus mengalami penurunan. Untuk area dengan luas 1500 m x 1500 m, performa *Packet Delivery Ratio* yang dihasilkan menurun pada kecepatan maksimum 10 m/s dan 20 m/s.

- Pada area dengan luas 1500 m x 1500 m, banyak hasil *Packet Delivery Ratio* yang di bawah 50% dibandingkan dengan hasil *Packet Delivery Ratio* pada area dengan luas 1000 m x 1000m.

2. Skenario MANET yang dihasilkan oleh *node-movement (mobility generation)* dan dijalankan menggunakan protocol *routing* AODV dengan penambahan kecepatan maksimal perpindahan *node* memiliki performa *End-to-end Delay* sebagai berikut :
   - Dari kecepatan maksimal perpindahan *node* sebesar 5 m/s hingga 20 m/s, performa *End-to-end Delay* yang dihasilkan pada area dengan luas 1000 m x 1000 m memiliki nilai *delay* yang fluktuatif pada jumlah *node* tertentu, yaitu mengalami penurunan kemudian mengalami peningkatan untuk jumlah *node* 60 dan 100, serta selalu mengalami penurunan untuk jumlah *node* 80 dan 120. Untuk area dengan luas 1500 m x 1500 m, performa *End-to-end Delay* yang dihasilkan juga memiliki nilai *delay* yang fluktuatif pada kecepatan maksimum 10 m/s, 15 m/s, dan 20 m/s. Pada kecepatan maksimum 5 m/s, nilai *delay* yang dihasilkan meningkat pada jumlah *node* 80 kemudian menurun hingga jumlah *node* 120.
   - Dari jumlah *node* 60 hingga 120, performa *End-to-end Delay* yang dihasilkan pada area dengan luas 1000 m x 1000 m terbaagi menjadi 2 variasi. Pada jumlah *node* 60 hingga 80, nilai performa mengalami peningkatan dari kecepatan maksimum 5 m/s hingga 10 m/s, namun menjadi turun saat kecepatan maksimum menjadi 15 m/s hingga 20 m/s. Pada jumlah *node* 100 dan 120, nilai performa meningkat dari kecepatan 5 m/s hingga 10 m/s, lalu

menurun saat kecepatan maksimal 15 m/s, kemudian meningkat kembali saat kecepatan maksimal 20 m/s. Untuk area dengan luas 1500 m x 1500 m, hasil performa senantiasa meningkat pada jumlah *node* 60 dan 100. Pada jumlah *node* 80, hasil performa menurun pada kecepatan maksimum 10 m/s kemudian terus meningkat sampai kecepatan maksimum 20 m/s. Pada jumlah *node* 120, hasil performa meningkat sampai pada kecepatan maksimum 15 m/s kemudian menurun pada kecepatan maksimum 20 m/s.

- Pada area dengan luas 1500 m x 1500 m, banyak hasil *End-to-end Delay* yang di atas 1,5 s (detik) dibandingkan dengan hasil *End-to-end Delay* pada area dengan luas 1000 m x 1000m.

3. Skenario MANET yang dihasilkan oleh *node-movement (mobility generation)* dan dijalankan menggunakan protokol *routing* AODV dengan penambahan kecepatan maksimal perpindahan *node* memiliki performa *Routing Overhead* sebagai berikut :
   - Dari kecepatan maksimal perpindahan *node* sebesar 5 m/s hingga 20 m/s, *Routing Overhead* yang dihasilkan pada area dengan luas 1000 m x 1000 m memiliki nilai yang senantiasa meningkat walaupun mengalami penurunan di jumlah node tertentu, kecuali pada kecepatan maksimal 15 m/s di mana nilai yang dihasilkan terus meningkat. Untuk area dengan luas 1500 m x 1500 m, hasil performa *Routing Overhead* yang dihasilkan mengalami peningkatan dari jumlah *node* 60 hingga 120.
   - Dari jumlah *node* 60 hingga 120, *Routing Overhead* yang dihasilkan pada area dengan luas 1000 m x 1000 m senantiasa mengalami peningkatan kecuali pada jumlah *node* 80 dan 100. Pada jumlah *node* 80, nilai *Routing Overhead* mengalami penurunan pada kecepatan maksimal 15 m/s sebelum kembali meningkat. Pada jumlah *node* 100, nilai *Routing Overhead* menurun pada kecepatan 20 m/s. Untuk area dengan luas 1500 m x 1500

m, hasil performa *Routing Overhead* yang dihasilkan mengalami perubahan yang fluktuatif. Pada jumlah *node* 60, peningkatan hasil performa terdapat pada kecepatan maksimum 10 m/s dan 20 m/s. Pada jumlah *node* 100, peningkatan hasil performa terdapat pada kecepatan maksimum 15 m/s serta penurunannya terdapat pada kecepatan maksimum 10 m/s dan 20 m/s. Pada jumlah *node* 120, hasil performa senantiasa meningkat hingga kecepatan maksimum 15 m/s kemudian menurun pada kecepatan maksimum 20 m/s.

- Pada area dengan luas 1000 m x 1000 m, semua hasil performa *Routing Overhead* bernilai di atas 1000 paket di mana hasil performa pada area dengan luas 1500 m x 1500 m terdapat beberapa nilai hasil performa yang di bawah 1000 paket.

6.2. Saran

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

1. Dapat dilakukan studi kinerja AODV kepada lingkungan VANET yang memiliki karakteristik mobilitas tinggi.
2. Dapat dilakukan penambahan jumlah koneksi antar *node.*
3. Dapat dilakukan penambahan jumlah skenario pada masing-masing variasi jumlah *node* dan kecepatan maksimal, minimal 10 skenario.

DAFTAR PUSTAKA

[1] I. N. B. Hartawan, "Optimasi Pemilihan Multi-Point Relay dengan Congestion Detection dalam Optimized Link State Routing pada Mobile Ad-Hoc Network," *Digilib ITS,* p. 1, 2014.

[2] B. Subba, S. Biswas and S. Karmakar, "Intrusion detection in Mobile Ad-hoc Networks: Bayesian game," *Engineering Science and Technology, an International Journal,* vol. 19, pp. 782-799, 2016.

[3] "eexploria.com," [Online]. Available: www.eexploria.com/manet-mobile-ad-hoc-network-characteristics-and-features/. [Accessed 17 05 2017].

[4] A. AbdelFattah, "An Efficient Scheme for MANET Domain Formation," *IJACSA) International Journal of Advanced Computer Science and Applications,* vol. 2, pp. 15-20, 2011.

[5] K. Gorantala, "Routing Protocols in Mobile Ad-hoc Networks," *Master's Thesis in Computing Science,* 2006.

[6] W. Su, S. J. Lee and M. Gerla, "MOBILITY PREDICTION IN WIRELESS NETWORKS," *Computer Science Department University of California.*

[7] "http://www.omnisecu.com," [Online]. Available: http://www.omnisecu.com/cisco-certified-network-associate-ccna/introduction-to-hybrid-routing-protocols.php. [Accessed 12 05 2017].

[8] A. Sarma and R. Kumar, "Perfomance Comparison and Detailed Study of AODV, DSDV, DSR. TORA and OLSR Routing Protocols in Ad Hoc Network," in *Fourth International Conference on Parallel, Distributed and Grid Computing (PDGC)*, 2016.

[9] "www.olsr.org," [Online]. Available: http://www.olsr.org/docs/report_html/node16.html. [Accessed 13 05 2017].

[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] T. S. Pradeepkumar, "www.nsnam.com," [Online]. Available: http://www.nsnam.com/2014/04/cbrgen-and-setdest-in-ns2-network.html. [Accessed 18 06 2017].

[12] 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.

[13] 8 November 2014. [Online]. Available: http://bengkelubuntu.org/teks/awk/Praktikum%2001%20-%20Berkenalan%20dengan%20AWK.pdf.. [Accessed 1 April 2017].

LAMPIRAN

```
#
# nodes: 60, pause: 2.00, max speed: 5.00, max
x: 1000.00, max y: 1000.00
#
$node_(0) set X_ 285.018740520789
$node_(0) set Y_ 725.297354828066
$node_(0) set Z_ 0.000000000000
$node_(1) set X_ 483.343298671629
$node_(1) set Y_ 443.539824163607
$node_(1) set Z_ 0.000000000000
$node_(2) set X_ 830.708253380684
$node_(2) set Y_ 757.387195373079
$node_(2) set Z_ 0.000000000000
$node_(3) set X_ 33.436305344745
$node_(3) set Y_ 150.581148993429
$node_(3) set Z_ 0.000000000000
$node_(4) set X_ 354.483556638607
$node_(4) set Y_ 401.870319989749
$node_(4) set Z_ 0.000000000000
$node_(5) set X_ 495.982265836632
$node_(5) set Y_ 100.674564936270
$node_(5) set Z_ 0.000000000000
$node_(6) set X_ 718.224031708808
$node_(6) set Y_ 377.629196398629
$node_(6) set Z_ 0.000000000000
$node_(7) set X_ 301.370492830189
$node_(7) set Y_ 163.959861757153
$node_(7) set Z_ 0.000000000000
$node_(8) set X_ 434.306024651894
$node_(8) set Y_ 675.629286219108
$node_(8) set Z_ 0.000000000000
$node_(9) set X_ 247.354239097257
$node_(9) set Y_ 296.750847651664
$node_(9) set Z_ 0.000000000000
$node_(10) set X_ 461.030975425253
$node_(10) set Y_ 546.670375323724
$node_(10) set Z_ 0.000000000000
$node_(11) set X_ 904.806804191278
$node_(11) set Y_ 587.561092156150
$node_(11) set Z_ 0.000000000000
$node_(12) set X_ 10.449282166886
$node_(12) set Y_ 582.478202217134
```

```
43  $node_(12) set Z_ 0.000000000000
44  $node_(13) set X_ 401.553099165448
45  $node_(13) set Y_ 410.925715806091
46  $node_(13) set Z_ 0.000000000000
47  $node_(14) set X_ 646.195017594975
48  $node_(14) set Y_ 105.029356862909
49  $node_(14) set Z_ 0.000000000000
50  $node_(15) set X_ 858.153549138442
51  $node_(15) set Y_ 260.115429317581
52  $node_(15) set Z_ 0.000000000000
53  $node_(16) set X_ 937.423319086444
54  $node_(16) set Y_ 859.293768818747
55  $node_(16) set Z_ 0.000000000000
56  $node_(17) set X_ 912.637222052678
57  $node_(17) set Y_ 247.107312641647
58  $node_(17) set Z_ 0.000000000000
59  $node_(18) set X_ 249.069181225819
60  $node_(18) set Y_ 95.159494735574
61  $node_(18) set Z_ 0.000000000000
62  $node_(19) set X_ 291.313585730555
63  $node_(19) set Y_ 36.612109890049
64  $node_(19) set Z_ 0.000000000000
65  $node_(20) set X_ 281.793800791053
66  $node_(20) set Y_ 507.724858263221
67  $node_(20) set Z_ 0.000000000000
68  $node_(21) set X_ 588.873846103847
69  $node_(21) set Y_ 996.937953020235
70  $node_(21) set Z_ 0.000000000000
71  $node_(22) set X_ 596.091159616355
72  $node_(22) set Y_ 698.121649494698
73  $node_(22) set Z_ 0.000000000000
74  $node_(23) set X_ 636.088184618005
75  $node_(23) set Y_ 749.798971684227
76  $node_(23) set Z_ 0.000000000000
77  $node_(24) set X_ 186.393804794622
78  $node_(24) set Y_ 256.980331254171
79  $node_(24) set Z_ 0.000000000000
81  $node_(25) set X_ 456.212115216097
82  $node_(25) set Y_ 299.259694816083
83  $node_(25) set Z_ 0.000000000000
84  $node_(26) set X_ 663.224395353038
85  $node_(26) set Y_ 412.731074692696
86  $node_(26) set Z_ 0.000000000000
87  $node_(27) set X_ 797.400540406656
```

```
88  $node_(27) set Y_ 132.465172454891
89  $node_(27) set Z_ 0.000000000000
90  $node_(28) set X_ 249.381902364883
91  $node_(28) set Y_ 630.767456069503
92  $node_(28) set Z_ 0.000000000000
93  $node_(29) set X_ 705.372028219835
94  $node_(29) set Y_ 814.932698734571
95  $node_(29) set Z_ 0.000000000000
96  $node_(30) set X_ 248.839351525201
97  $node_(30) set Y_ 212.684613940865
99  $node_(30) set Z_ 0.000000000000
100 $node_(31) set X_ 262.229500930695
101 $node_(31) set Y_ 232.947133121314
102 $node_(31) set Z_ 0.000000000000
103 $node_(32) set X_ 887.083848126568
104 $node_(32) set Y_ 24.619673406913
105 $node_(32) set Z_ 0.000000000000
106 $node_(33) set X_ 894.322565528353
107 $node_(33) set Y_ 730.923997245774
108 $node_(33) set Z_ 0.000000000000
109 $node_(34) set X_ 389.172609184846
110 $node_(34) set Y_ 603.928863680270
111 $node_(34) set Z_ 0.000000000000
112 $node_(35) set X_ 1.684847835090
113 $node_(35) set Y_ 15.505378186963
114 $node_(35) set Z_ 0.000000000000
115 $node_(36) set X_ 967.414529347378
116 $node_(36) set Y_ 141.337442071686
117 $node_(36) set Z_ 0.000000000000
118 $node_(37) set X_ 963.859493723785
119 $node_(37) set Y_ 868.214397828233
120 $node_(37) set Z_ 0.000000000000
121 $node_(38) set X_ 958.500707840581
122 $node_(38) set Y_ 42.785367905012
123 $node_(38) set Z_ 0.000000000000
124 $node_(39) set X_ 812.591420490997
125 $node_(39) set Y_ 530.680367811936
126 $node_(39) set Z_ 0.000000000000
127 $node_(40) set X_ 569.294659051413
128 $node_(40) set Y_ 684.719493710821
129 $node_(40) set Z_ 0.000000000000
130 $node_(41) set X_ 245.085969329318
131 $node_(41) set Y_ 836.974340512106
132 $node_(41) set Z_ 0.000000000000
```

```
133 $node_(42) set X_ 509.455148122895
134 $node_(42) set Y_ 503.425228808179
135 $node_(42) set Z_ 0.000000000000
136 $node_(43) set X_ 635.551507397255
137 $node_(43) set Y_ 832.944230002444
138 $node_(43) set Z_ 0.000000000000
139 $node_(44) set X_ 99.088123675978
140 $node_(44) set Y_ 161.258404734952
141 $node_(44) set Z_ 0.000000000000
142 $node_(45) set X_ 864.616318335802
143 $node_(45) set Y_ 533.071222686864
144 $node_(45) set Z_ 0.000000000000
145 $node_(46) set X_ 939.524456491016
146 $node_(46) set Y_ 748.985068590589
147 $node_(46) set Z_ 0.000000000000
148 $node_(47) set X_ 450.760989626470
149 $node_(47) set Y_ 779.998830575439
150 $node_(47) set Z_ 0.000000000000
151 $node_(48) set X_ 141.565665939278
152 $node_(48) set Y_ 885.055753889400
153 $node_(48) set Z_ 0.000000000000
154 $node_(49) set X_ 703.560985005620
155 $node_(49) set Y_ 776.052920943826
156 $node_(49) set Z_ 0.000000000000
157 $node_(50) set X_ 392.722282718456
158 $node_(50) set Y_ 108.416211221035
159 $node_(50) set Z_ 0.000000000000
160 $node_(51) set X_ 859.586344751054
161 $node_(51) set Y_ 929.795800567960
162 $node_(51) set Z_ 0.000000000000
163 $node_(52) set X_ 733.777195360897
164 $node_(52) set Y_ 466.272280315085
165 $node_(52) set Z_ 0.000000000000
166 $node_(53) set X_ 207.308308249369
167 $node_(53) set Y_ 978.206814841139
168 $node_(53) set Z_ 0.000000000000
168 $node_(54) set X_ 197.138725082589
170 $node_(54) set Y_ 533.289210154711
171 $node_(54) set Z_ 0.000000000000
172 $node_(55) set X_ 206.022581516927
173 $node_(55) set Y_ 214.652425061843
174 $node_(55) set Z_ 0.000000000000
175 $node_(56) set X_ 575.413671249068
176 $node_(56) set Y_ 835.367738445403
```

```
177 $node_(56) set Z_ 0.000000000000
178 $node_(57) set X_ 434.324295339047
179 $node_(57) set Y_ 766.763322634336
180 $node_(57) set Z_ 0.000000000000
181 $node_(58) set X_ 990.838049467261
182 $node_(58) set Y_ 36.798450549617
183 $node_(58) set Z_ 0.000000000000
184 $node_(59) set X_ 558.585641483267
185 $node_(59) set Y_ 965.059491510590
186 $node_(59) 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 3
3   $god_ set-dist 0 3 4
4   $god_ set-dist 0 4 2
5   $god_ set-dist 0 5 4
6   $god_ set-dist 0 6 3
7   $god_ set-dist 0 7 3
8   $god_ set-dist 0 8 1
9   $god_ set-dist 0 9 2
11  $god_ set-dist 0 10 2
12  $god_ set-dist 0 11 4
13  $god_ set-dist 0 12 2
14  $god_ set-dist 0 13 2
15  $god_ set-dist 0 14 5
16  $god_ set-dist 0 15 4
17  $god_ set-dist 0 16 4
18  $god_ set-dist 0 17 4
19  $god_ set-dist 0 18 3
20  $god_ set-dist 0 19 4
21  $god_ set-dist 0 20 1
22  $god_ set-dist 0 21 3
23  $god_ set-dist 0 22 2
24  $god_ set-dist 0 23 2
25  $god_ set-dist 0 24 3
26  $god_ set-dist 0 25 3
27  $god_ set-dist 0 26 3
28  $god_ set-dist 0 27 5
29  $god_ set-dist 0 28 1
30  $god_ set-dist 0 29 3
31  $god_ set-dist 0 30 3
```

```
32  $god_ set-dist 0 31 3
33  $god_ set-dist 0 32 5
34  $god_ set-dist 0 33 4
35  $god_ set-dist 0 34 1
36  $god_ set-dist 0 35 4
37  $god_ set-dist 0 36 5
38  $god_ set-dist 0 37 4
39  $god_ set-dist 0 38 5
40  $god_ set-dist 0 39 4
41  $god_ set-dist 0 40 2
42  $god_ set-dist 0 41 1
43  $god_ set-dist 0 42 2
44  $god_ set-dist 0 43 2
45  $god_ set-dist 0 44 3
46  $god_ set-dist 0 45 4
47  $god_ set-dist 0 46 4
48  $god_ set-dist 0 47 1
49  $god_ set-dist 0 48 1
50  $god_ set-dist 0 49 3
51  $god_ set-dist 0 50 3
52  $god_ set-dist 0 51 3
53  $god_ set-dist 0 52 3
54  $god_ set-dist 0 53 2
55  $god_ set-dist 0 54 1
56  $god_ set-dist 0 55 3
57  $god_ set-dist 0 56 2
58  $god_ set-dist 0 57 1
59  $god_ set-dist 0 58 5
60  $god_ set-dist 0 59 2
61  $god_ set-dist 1 2 3
62  $god_ set-dist 1 3 3
63  $god_ set-dist 1 4 1
64  $god_ set-dist 1 5 2
65  $god_ set-dist 1 6 1
66  $god_ set-dist 1 7 2
67  $god_ set-dist 1 8 1
68  $god_ set-dist 1 9 2
69  $god_ set-dist 1 10 1
70  $god_ set-dist 1 11 3
71  $god_ set-dist 1 12 3
72  $god_ set-dist 1 13 1
73  $god_ set-dist 1 14 3
74  $god_ set-dist 1 15 2
75  $god_ set-dist 1 16 4
```

```
76  $god_ set-dist 1 17 2
77  $god_ set-dist 1 18 3
78  $god_ set-dist 1 19 3
79  $god_ set-dist 1 20 1
80  $god_ set-dist 1 21 3
81  $god_ set-dist 1 22 2
82  $god_ set-dist 1 23 2
83  $god_ set-dist 1 24 2
84  $god_ set-dist 1 25 1
85  $god_ set-dist 1 26 1
86  $god_ set-dist 1 27 3
87  $god_ set-dist 1 28 2
88  $god_ set-dist 1 29 3
89  $god_ set-dist 1 30 2
90  $god_ set-dist 1 31 2
91  $god_ set-dist 1 32 3
92  $god_ set-dist 1 33 3
93  $god_ set-dist 1 34 1
94  $god_ set-dist 1 35 4
95  $god_ set-dist 1 36 3
96  $god_ set-dist 1 37 4
97  $god_ set-dist 1 38 3
98  $god_ set-dist 1 39 2
99  $god_ set-dist 1 40 2
```

Gambar 7.2 Pembuatan GOD setiap *node* dari potongan Skenario

```
1   $ns_ at 2.000000000000 "$node_(0) setdest
    920.157653138226 346.042720828412
    4.098179836856"
2   $ns_ at 2.000000000000 "$node_(1) setdest
    762.362786236093 182.803113717364
    1.549699032292"
3   $ns_ at 2.000000000000 "$node_(2) setdest
    743.064234861490 318.801943983604
    0.173852116838"
4   $ns_ at 2.000000000000 "$node_(3) setdest
    35.593195213793 703.200649764793
    4.475475076097"
5   $ns_ at 2.000000000000 "$node_(4) setdest
    390.637450211819 959.148310241953
    1.653329848520"
6   $ns_ at 2.000000000000 "$node_(5) setdest
    89.406853214983 246.821480416429
    1.675705364567"
7   $ns_ at 2.000000000000 "$node_(6) setdest
    460.416138350628 460.060254599092
    3.170936120338"
8   $ns_ at 2.000000000000 "$node_(7) setdest
    593.852656083496 982.955552277797
    1.454264346391"
9   $ns_ at 2.000000000000 "$node_(8) setdest
    856.220348294320 29.121740501682
    0.246552691395"
10  $ns_ at 2.000000000000 "$node_(9) setdest
    999.649168673676 650.508764503948
    0.871802033236"
11  $ns_ at 2.000000000000 "$node_(10) setdest
    955.039440572309 232.661010323439
    2.719888227465"
12  $ns_ at 2.000000000000 "$node_(11) setdest
    535.739412865089 876.009314137031
    2.191129268788"
13  $ns_ at 2.000000000000 "$node_(12) setdest
    839.479184851905 837.772310305536
    4.995594484285"
14  $ns_ at 2.000000000000 "$node_(13) setdest
    216.121983470715 865.757075389259
    4.522487385123"
```

```
15  $ns_ at 2.000000000000 "$node_(14) setdest
    442.900401335974 317.692299392074
    1.634195392233"
16  $ns_ at 2.000000000000 "$node_(15) setdest
    952.522201725426 279.622106850473
    0.855064785074"
```

Gambar 7.3 Pergerakan setiap *node* dari potongan Skenario

```
1   $ns_ at 2.135489837623 "$god_ set-dist 12 47
    2"
2   $ns_ at 2.135489837623 "$god_ set-dist 28 47
    1"
3   $ns_ at 2.444482530382 "$god_ set-dist 1 11 2"
4   $ns_ at 2.444482530382 "$god_ set-dist 1 52 1"
5   $ns_ at 2.588781206060 "$god_ set-dist 0 5 3"
6   $ns_ at 2.588781206060 "$god_ set-dist 0 10 1"
7   $ns_ at 2.588781206060 "$god_ set-dist 0 14 4"
8   $ns_ at 2.588781206060 "$god_ set-dist 0 15 3"
9   $ns_ at 2.588781206060 "$god_ set-dist 0 25 2"
10  $ns_ at 2.588781206060 "$god_ set-dist 0 26 2"
11  $ns_ at 2.588781206060 "$god_ set-dist 0 27 4"
12  $ns_ at 2.588781206060 "$god_ set-dist 0 32 4"
13  $ns_ at 2.588781206060 "$god_ set-dist 0 36 4"
14  $ns_ at 2.588781206060 "$god_ set-dist 0 38 4"
15  $ns_ at 2.588781206060 "$god_ set-dist 0 39 3"
16  $ns_ at 2.588781206060 "$god_ set-dist 0 45 3"
17  $ns_ at 2.588781206060 "$god_ set-dist 5 48 4"
18  $ns_ at 2.588781206060 "$god_ set-dist 10 48
    2"
19  $ns_ at 2.588781206060 "$god_ set-dist 14 48
    5"
20  $ns_ at 2.588781206060 "$god_ set-dist 15 48
    4"
```

Gambar 7.4 Informasi pada GOD dari potongan Skenario

```
#
# nodes: 50, max conn: 1, send rate: 1, seed: 1
#
#
# 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"
#
#Total sources/connections: 1/1
#
```

Gambar 7.5 Koneksi yang digunakan pada cbr1

```
# Define options

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)           1000;
set opt(y)           1000;
set val(ifqlen)      50;
set val(nn)          120;
set val(seed)        0.0
set val(adhocRouting)   AODV
set val(stop)        200;
set val(cp)          "cbr1"
set val(sc)          "scenario-s5-n120-5";
Phy/WirelessPhy set RXThresh_ 1.42681e-08;
```

```
20  set ns_       [new Simulator]
21
22  set topo      [new Topography]
23
24  set tracefd   [open AODV_$val(sc).tr w]
25
26  $ns_ trace-all $tracefd
27  $ns_ namtrace-all-wireless $namtrace $opt(x)
28  $opt(y)
29
30  set topo      [new Topography]
31  $topo load_flatgrid $opt(x) $opt(y)
32
33  set god_ [create-god $val(nn)]
34
35  $ns_ node-config -adhocRouting
36  $val(adhocRouting) \
37                   -llType $val(ll) \
38                   -macType $val(mac) \
39                   -ifqType $val(ifq) \
40                   -ifqLen $val(ifqlen) \
41                   -antType $val(ant) \
42                   -propType $val(prop) \
43                   -phyType $val(netif) \
44               -channelType $val(chan) \
45               -topoInstance $topo \
46               -agentTrace ON \
47               -routerTrace ON \
48               -macTrace OFF \
49               -movementTrace ON \
50
51  for {set i 0} {$i < $val(nn)} {incr i} {
52        set node_($i) [$ns_ node]
53        $node_($i) random-motion 0 ;
54  }
55
56  puts "Loading connection pattern..."
57  source $val(cp)
58
59  puts "Loading scenario file..."
60  source $val(sc)
61
62  for {set i 0} {$i < $val(nn)} {incr i} {
63      $ns_ initial_node_pos $node_($i) 20
```

```
}

for {set i 0} {$i < $val(nn) } {incr i} {
    $ns_ at $val(stop).0 "$node_($i) reset";
}

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

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

puts "Starting Simulation..."
$ns_ run
```

Gambar 7.6 *File* AODV_Main.tcl untuk Protokol *Routing* AODV

```
BEGIN {
sent=0;
recv=0;
pdr=0;
}
{
#count packet send
if ($1 == "s" && $3 == "_1_" && $4 == "AGT" &&
$7 == "cbr")
 {
 sent++;
 }
#count packet receive
if ($1 == "r" && $3 == "_2_" && $4 == "AGT" &&
$7 == "cbr")
 {
 recv++;
 }
}
END {
pdr = ( recv / sent ) * 100
print "Transmitted packet (s):", sent;
```

```
print "Received packet (s):", recv;
print "Packet delivery ratio:", pdr, "%";
}
```

Gambar 7.7 Implementasi *Packet Delivery Ratio.awk*

```
BEGIN {
rt_pkts = 0;
}
{
if (($1 == "s" || $1 == "f") && ($4 == "RTR") &&
($7 == "OLSR"))
    rt_pkts++;
}
END {
printf ("Total number of routing packets\t%d\n",
rt_pkts);
}
```

Gambar 7.8 Implementasi *Routing Overhead.awk*

```
BEGIN{
for ( i in pkt_id)
     {
              pkt_id[i] = 0;
     }
for ( i in pkt_sent)
     {
              pkt_sent[i] = 0;
     }
for ( i in pkt_recv )
     {
              pkt_recv[i] = 0;
     }
     delay = avg_delay = 0;
     recv = 0;
     recv_id = 0;
}
{
# count packet send
if ( $1 == "s" && $3 == "_1_" && $4 == "AGT" &&
$7 == "cbr" )
```

```
21  {
22    pkt_sent[$6] = $2;
23  }
24  # count packet receive
25  if ( $1 == "r" && $3 == "_2_" && $4 == "AGT" &&
    $7 == "cbr" &&  recv_id != $6 )
26  {
27    recv++;
28    recv_id = $6;
29    pkt_recv[$6] = $2;
30  }
31  }
32  END{
33  for (i in pkt_recv)
34      {
35        delay += pkt_recv[i] - pkt_sent[i];
36      }
37
38  avg_delay = delay / recv;
39
40  print "Total Packet(s) Receive =", recv;
41  print "Total Delay =", delay, "second";
42  print  "Average  Packet  Delivery  Delay  =  ",
43  avg_delay, "second";
44  }
```

Gambar 7.9 Implementasi *End-to-End Delay.awk*

Instalasi NS-2

Pada Tugas Akhir ini digunakan cara mengunduh *file* NS-2. Langkah-langkah detail adalah sebagai berikut.

- Pertama kali lakukan *update* komponen terbaru dari Ubuntu dan pastikan Ubuntu yang diinstal ter-*update* seluruhnya.

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

Gambar 7.10 Perintah *update* seluruh komponen Ubuntu

- Kemudian lakukan instalasi modul-modul 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.

```
$ 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 7.11 Perintah instalasi dependensi NS-2

- Setelah semua dependensi lengkap, *file* ns-2 yang telah diunduh dipindahkan menuju *folder* /home dan dilakukan proses ektraksi. Kemudian dilakukan proses pengubahan script pada ls.h yang terdapat pada *folder* /ns-allinone-2.35/ns-2.35/linkstate/ls.h.

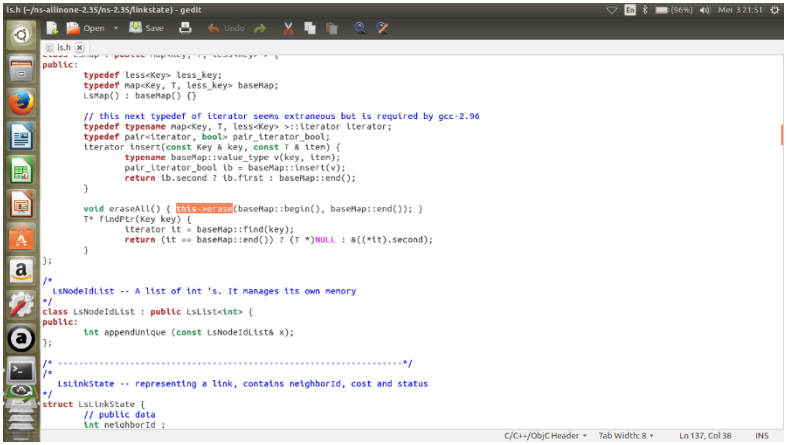
```
$ tar -xvzf ns-allinone-2.35.tar.gz
$ cd /ns-allinone-2.35/ns-2.35/linkstate/
$ gedit ls.h
```

Gambar 7.12 Proses ekstrak dan pengubahan ls.h

```
rizqi@rizqi-X450JN:~$ cd ns-allinone-2.35/ns-2.35/link
link/       linkstate/
rizqi@rizqi-X450JN:~$ cd ns-allinone-2.35/ns-2.35/linkstate/
rizqi@rizqi-X450JN:~/ns-allinone-2.35/ns-2.35/linkstate$ gedit ls.h
```

Gambar 7.13 Screenshot proses ekstrak dan pengubahan ls.h

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



Gambar 7.14 Proses pengubahan *line of code* ls.h

- 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.

```
rizqi@rizqi-X450JN:~/ns-allinone-2.35/ns-2.35/linkstate$ cd
rizqi@rizqi-X450JN:~$ cd TA\ Works/
rizqi@rizqi-X450JN:~/TA Works$ ns
%
```

Gambar 7.15 Perintah pengecekan NS-2

BIODATA PENULIS

Rizqi Hidayatullah, biasa dipanggil Rizqi, lahir di Surakarta pada 12 Juli 1993. Penulis adalah anak pertama dari dua bersaudara dan dibesarkan di Samarinda, Kalimantan Timur. Penulis menempuh pendidikan formal di SD Muhammadiyah 1 Samarinda (1998-2004), SMPN 1 Samarinda (2004-2007), SMAN 1 Samarinda (2007-2010). Pada tahun 2010, penulis memulai pendidikan S1 Jurusan Teknik Informatika Fakultas Teknologi Informasi di Institut Teknologi Sepuluh Nopember Surabaya angkatan 2010 yang terdaftar dengan NRP 5110100167.

Di jurusan Teknik Informatika, penulis mengambil bidang minat Arsitektur Jaringan Komputer (AJK). Selama menempuh kuliah, penulis juga cukup aktif dalam organisasi kemahasiswaan seperti Himpunan Mahasiswa Teknik Computer (HMTC). Penulis dapat dihubungi melalui alamat email hidarizqihidayat@gmail.com.