**PENERAPAN ALGORITMA EA-SHORT PADA PROTOKOL ROUTING AOMDV UNTUK MENEMUKAN RUTE YANG HANDAL BERBASIS ENERGI DI JARINGAN MANET**

Tugas Akhir

Untuk memenuhi sebagai persyaratan

mencapai derajat Sarjana S-1 Program studi Teknik Informatika



**Oleh:**

**B. NURWAHYU HAIRANI**

**F1D 015 017**

**PROGRAM STUDI TEKNIK INFORMATIKA**

**FAKULTAS TEKNIK**

**UNIVERSITAS MATARAM**

**2020**

**TUGAS AKHIR**

**PENERAPAN ALGORITMA EA-SHORT PADA PROTOKOL ROUTING AOMDV UNTUK MENEMUKAN RUTE YANG HANDAL BERBASIS ENERGI DI JARINGAN MANET**

**Oleh**

**B. NURWAHYU HAIRANI**

F1D 015 017

Telah diperiksa oleh tim pembimbing:

1. Pembimbing Utama



**Andy Hidayat Jatmika, S.T., M.Kom.** Tanggal :20/07/2020

**NIP. 19831209 201212 1 001**

1. Pembimbing Pendamping



**Fitri Bimantoro, S.T, M.Kom.** Tanggal : 13/07/2020

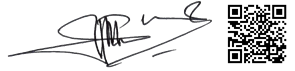
**NIP. 19860622 201504 1 002**

Mengetahui,

Ketua Program Studi Teknik Informatika

Fakultas Teknik

Universitas Mataram



**Prof. Dr. Eng. I Gede Pasek Suta Wijaya, S.T., M.T.**

**NIP: 19731130 200003 1 001**

**TUGAS AKHIR**

**PENERAPAN ALGORITMA EA-SHORT PADA PROTOKOL ROUTING AOMDV UNTUK MENEMUKAN RUTE YANG HANDAL BERBASIS ENERGI DI JARINGAN MANET**

**Oleh**

**B. NURWAHYU HAIRANI**

F1D 015 017

Telah diajukan di depan Penguji

Pada tanggal 7 Juli 2020

Dan dinyatakan telah memenuhi syarat mencapai derajat Sarjana S-1

Program Studi Teknik Informatika

Susunan Tim Penguji:

1. Penguji I



**Dr.Eng. I Gede Putu Wirama Wedashwara W. S.T., M.T.** Tanggal : 09/07/2020

**NIP. 19840919 201803 1 001**

1. Penguji II



**Ahmad Zafrullah Mardiansyah, S.T., M.Eng.** Tanggal : 10/07/2020

**NIP.-**

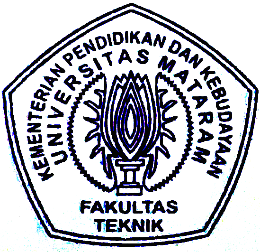
1. Penguji III



**Ramaditia Dwiyansaputra, S.T., M.Eng**. Tanggal: 13/072020

**NIP. -**

Mataram, 21 Juli 2020

Dekan Fakultas Teknik

Universitas Mataram



**Akmaluddin, S.T., M.Sc.(Eng.), Ph.D**

**NIP: 19681231 199412 1 001**

# PERNYATAAN KEASLIAN TUGAS AKHIR

Dengan ini saya menyatakan bahwa dalam Tugas Akhir ini tidak terdapat karya yang pernah diajukan untuk memperoleh gelar kesarjanaan di suatu Perguruan Tinggi, dan sepanjang pengetahuan saya juga tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali secara tertulis diacu dalam naskah ini dan disebutkan dalam daftar pustaka.

Demikian surat pernyataan ini dibuat tanpa tekanan dari pihak mana pun dan dengan kesadaran penuh terhadap tanggung jawab dan konsekuensi dari pernyataan tersebut.

Mataram, Juli 2020

B. Nurwahyu Hairani

# PRAKATA

Puji syukur penulis panjatkan kehadirat Allah Subhanahu Wa Ta’ala serta junjungan besar Nabi Muhammad Sallallahu ‘Alaihi Wasallam, yang telah melimpahkan Rahmat dan karunia-Nya, sehingga penulis dapat menyelesaikan penyusunan Tugas Akhir untuk memenuhi persyaratan akademis guna mencapai gelar kesarjanaan di Fakultas Teknik Universitas Mataram dengan judul “Penerapan Algoritma EA-SHORT Pada Protokol Routing AOMDV untuk Menemukan Rute Handal Berbasis Energi di Jaringan MANET”.

Dalam penulisan Tugas Akhir ini tentunya tidak luput dari kekurangan, baik aspek kualitas maupun aspek kuantitas dari materi penelitian yang disajikan. Semua ini didasarkan dari keterbatasan yang dimiliki penulis. Penulis menyadari bahwa Tugas Akhir ini masih jauh dari kata sempurna sehingga penulis membutuhkan kritik dan saran yang membangun untuk kemajuan teknologi di masa yang akan datang. Akhir kata semoga tidaklah terlampau berlebihan, bila penulis berharap agar karya ini dapat bermanfaat bagi pembaca.

Mataram, Juli 2020

B. Nurwahyu Hairani

# UCAPAN TERIMA KASIH

Tugas Akhir ini dapat diselesaikan berkat bimbingan dan dukungan ilmiah maupun materiil dari berbagai pihak, oleh karena itu pada kesempatan ini penulis menyampaikan ucapan terima kasih yang setulus-tulusnya kepada:

1. Kedua Orang Tua dan keluarga yang selama ini telah memberikan bimbingan, doa dan dukungannya dalam menjalani perkuliahan dan pengerjaan Tugas Akhir ini.
2. Bapak Andy Hidayat Jatmika, S.T., M.Kom., selaku dosen pembimbing utama yang telah memberikan bimbingan dan arahan kepada penulis selama penyusunan Tugas Akhir ini, sehingga dapat terselesaikan dengan baik.
3. Bapak Fitri Bimantoro, S.T., M.Kom., selaku dosen pembimbing pendamping yang telah memberikan bimbingan dan arahan kepada penulis selama penyusunan Tugas Akhir ini, sehingga dapat terselesaikan dengan baik.
4. Bapak Dr. Eng. I Gede Putu Wirarama Wedashwara W., S.T., M.T., Bapak Ahmad Zafrullah Mardiansyah, S.T., M.Eng. dan Bapak Ramaditia Dwiyansaputra, S.T., M.Eng., selaku dosen penguji yang telah memberikan kritik dan saran yang bersifat membangun dalam penyelesaian tugas akhir ini.
5. Sahabat penulis Mila Rosiana, Relanti Saleha, Islam Hidayah, beserta keluarganya, serta teman-teman Teknik Informatika angkatan 2015 dan Mahasiswa Universitas Mataram lainnya atas segala bentuk dukungan dan semangat yang telah diberikan kepada penulis.
6. Semua pihak yang tidak dapat penulis sebutkan satu persatu, yang telah memberikan dukungan dan bimbingan kepada penulis dalam menyelesaikan Tugas Akhir ini.

Semoga Tuhan Yang Maha Pengasih lagi Maha Penyayang memberikan imbalan yang setimpal atas bantuan yang telah diberikan kepada penulis.

# DAFTAR ISI

[DAFTAR ISI](#_Toc44327556) vi

[DAFTAR GAMBAR vii](#_Toc44327557)

[DAFTAR TABEL viii](#_Toc44327558)

[ABSTRAK ix](#_Toc44327559)

[BAB I](#_Toc44327560) [PENDAHULUAN 1](#_Toc44327561)

[1.1 Latar Belakang 1](#_Toc44327562)

[1.2 Rumusan Masalah 3](#_Toc44327563)

[1.3 Batasan Masalah 3](#_Toc44327564)

[1.4 Tujuan Penelitian 3](#_Toc44327565)

[1.5 Manfaat Penelitian 3](#_Toc44327566)

[1.6 Sistematika Penulisan 4](#_Toc44327567)

[BAB II](#_Toc44327568) [TINJAUAN PUSTAKA DAN DASAR TEORI 5](#_Toc44327569)

[2.1 Tinjauan Pustaka 5](#_Toc44327570)

[2.1.1 Penelitian Terkait 5](#_Toc44327571)

[2.1.2 Penelitian yang Diusulkan 7](#_Toc44327572)

[2.2 Dasar Teori 8](#_Toc44327573)

[BAB III](#_Toc44327574) [METODE PENELITIAN 15](#_Toc44327575)

[3.1 Diagram Alir Penelitian 15](#_Toc44327576)

[3.2 Algoritma Penemuan Rute Menggunakan Protokol *Routing* AOMDV 18](#_Toc44327577)

[3.3 Algoritma EA*-*SHORT 20](#_Toc44327578)

[3.4 Lingkungan Simulasi 24](#_Toc44327579)

[BAB IV HASIL DAN PEMBAHASAN 27](#_Toc44327580)

[4.1 Implementasi Metode 27](#_Toc44327581)

[4.2 Menentukan Skenario Simulasi 30](#_Toc44327582)

[4.3 Hasil Uji Coba 37](#_Toc44327583)

[4.4 Analisis Hasil Uji Coba 44](#_Toc44327584)

[BAB V](#_Toc44327585) [KESIMPULAN DAN SARAN 54](#_Toc44327586)

[5.1 Kesimpulan 54](#_Toc44327587)

[5.2 Saran 54](#_Toc44327588)

[DAFTAR PUSTAKA](#_Toc44327589)

# DAFTAR GAMBAR

[Gambar 2. 1 Struktur jaringan nirkabel [7]. 8](#_Toc44609603)

[Gambar 2. 2 Contoh rute multipath. 12](#_Toc44609604)

[Gambar 3.1 Diagram alir penelitian. 15](#_Toc44609537)

[Gambar 3.2 *Flowchart route discovery* pada AOMDV. 19](#_Toc44609538)

[Gambar 3.3 Proses *route discovery* pada AOMDV. 19](#_Toc44609539)

[Gambar 3.4 Contoh operator pengalihan rute lokal berturut-turut (tautan yang ditunjukkan pada gambar adalah tautan nirkabel). 21](#_Toc44609540)

[Gambar 3.5 *Flowchart* algoritma EA-SHORT. 23](#_Toc44609541)

[Gambar 3.6 *Flowchart* algoritma EA-SHORT (Lanjutan Gambar 3.5). 24](#_Toc44609542)

[Gambar 4. 1 Inisialisasi variabel untuk menampung informasi node. 27](#_Toc44609714)

[Gambar 4.2 Inisialisasi nilai variabel. 28](#_Toc44609715)

[Gambar 4.3 Potongan *script* EA- AOMDV pada fungsi *recv* dan *recvRequest*. 28](#_Toc44609716)

[Gambar 4.4 Potongan *script* EA- AOMDV pada fungsi *recvReplay* (part 1). 29](#_Toc44609717)

[Gambar 4.5 Potongan *script* EA- AOMDV pada fungsi *recvReplay* (part 2). 30](#_Toc44609718)

[Gambar 4.6 Mekanisme cara menentukan pola *traffic*. 31](#_Toc44609719)

[Gambar 4.7 Potongan *script* hasil pola *traffic* dengan cbrgen.tcl. 31](#_Toc44609720)

[Gambar 4.8 Mekanisme pola pergerakan node. 32](#_Toc44609721)

[Gambar 4.9 Potongan *script* hasil pola pergerakan *node* pada posisi awal. 32](#_Toc44609722)

[Gambar 4.10 Potongan *script* hasil pola pergerakan *node* pada posisi tujuan. 33](#_Toc44609723)

[Gambar 4.11 *Script* *file* TCL. 34](#_Toc44609724)

[Gambar 4.12 Perintah untuk menjalankan simulasi. 35](#_Toc44609725)

[Gambar 4.13 Potongan *script trace file*. 35](#_Toc44609726)

[Gambar 4.14 Tampilan simulasi pergerakan *node* pada *network animator (nam)*. 35](#_Toc44609727)

[Gambar 4.15 Perintah proses *filtering file.tr* dengan awk *script*. 36](#_Toc44609728)

[Gambar 4.16 Potongan AWK *script* untuk *throughput*. 36](#_Toc44609729)

[Gambar 4.17 Potongan AWK *script* untuk PDR dan *end-to-end delay*. 37](#_Toc44609730)

[Gambar 4.18 Grafik perbandingan nilai *throughput* pada 3 percobaan.. 46](#_Toc44609731)

[Gambar 4.19 Grafik perbandingan nilai *packet delivery ratio* pada 3 percobaan 49](#_Toc44609732)

[Gambar 4. 20 Grafik perbandingan nilai *end-toend delay* pada 3 percobaan 52](#_Toc44609733)

# DAFTAR TABEL

[Tabel 3.1 Spesifikasi *hardware* yang digunakan. 16](#_Toc44610032)

[Tabel 3.2 Parameter skenario. 26](#_Toc44610033)

[Tabel 4.1 Hasil uji coba 1 untuk *throughput* 38](#_Toc44610045)

[Tabel 4. 2 Hasil uji coba 2 untuk *throughput*. 38](#_Toc44610046)

[Tabel 4. 3 Hasil uji coba 3 untuk *throughput*. 39](#_Toc44610047)

[Tabel 4.4 Hasil uji coba 1 untuk *packet delivery ratio*. 40](#_Toc44610048)

[Tabel 4. 5 Hasil uji coba 2 untuk *packet delivery ratio*. 41](#_Toc44610049)

[Tabel 4.6 Hasil uji coba 3 untuk *packet delivery ratio*. 41](#_Toc44610050)

[Tabel 4.7 Hasil uji coba 1 untuk *average end-to-end delay*. 42](#_Toc44610051)

[Tabel 4.8 Hasil uji coba 2 untuk *average end-to-end delay*. 43](#_Toc44610052)

[Tabel 4. 9 Hasil uji coba 3 untuk *average end-to-end delay*. 43](#_Toc44610053)

# ABSTRAK

*Mobile Ad Hoc Network (*MANET*)* adalah jaringan nirkabel yang berdiri sendiri, terdiri dari sejumlah *node* yang mampu bergerak ke segala arah secara bebas. Protokol *routing* yang digunakan sebagai objek penelitian ini adalah *Ad-Hoc On-Demand Multipath Distance Vektor* (AOMDV). Penelitian ini melakukan proses perhitungan energi pada suatu *node* yang dapat digunakan sebagai rute baru berbasis energi dengan memanfaatkan perubahan topologi. Efisiensi energi pada *routing protocol* dapat dilakukan dengan menggunakan metode *Energy Aware* SHORT (*Self Healing and Optimizing Routing Techniques*). Tujuan utama dari algoritma *Energy Aware* SHORT (EA-SHORT) adalah untuk mendistribusikan beban lalu lintas setiap *node* pada MANET secara adil, dengan cara memperpanjang masa pakai *node* dan jaringan dengan merutekan paket melalui *node* yang memiliki sisa daya cukup dan menghindari *node* yang memiliki daya rendah. Sehingga dapat meningkatkan kinerja dan konservasi daya pada MANET. Kinerja AOMDV akan dibandingkan dengan AOMDV yang telah dimodifikasi dengan EA-SHORT diukur dari nilai parameter yang telah ditentukan. Hasil Analisa menunjukkan penerapan algoritma EA- SHORT dalam efisiensi pencarian rute berhasil meningkatkan kinerja. Hasil *throughput* pada EA-AOMDV mengalami peningkatan sebesar 13,904 % untuk luas area 500x500 m2 dan 13,905 % pada1000x1000 m2. *Packet delivery ratio* meningkat sebesar 0,91% dan 2,273 %. *Average end-to-end delay* menurun sebesar 20,482% dan 18,734.

**Kata kunci:** MANET, AOMDV, *Energy Aware* SHORT, EA-AOMDV, Protokol *Routing*

# ABSTRACT

*Mobile Ad Hoc Network (MANET) is a stand-alone wireless network, consisting of several nodes that can move in all directions freely. The routing protocol used as the object of this study is Ad-Hoc On-Demand Multipath Distance Vector (AOMDV). This research carries out an energy calculation process at a node that can be used as a new energy-based route by utilizing topology changes. Energy efficiency in the routing protocol can be done using the Energy-Aware SHORT (Self Healing and Optimizing Routing Techniques) method. The main purpose of the Energy-Aware SHORT (EA-SHORT) algorithm is to distribute the traffic load of each node on MANET fairly, by extending the life of the node and network by routing packets through nodes that have enough remaining power and avoiding nodes that have low power. AOMDV performance will be compared with AOMDV which has been modified with EA-SHORT measured from the specified parameter values. Analysis results show that the application of the EA-SHORT algorithm in the efficiency of route search succeeded in improving performance. The results of throughput on EA-AOMDV increased by 13.904% for an area of ​​500x500 m2 and 13.905% at 1000x1000 m2. The packet delivery ratio increased by 0.91% and 2.273%. Average end-to-end delay decreased by 20.482% and 18.734%.*

***Keywords:*** *MANET, AOMDV, Energy Aware SHORT, EA-AOMDV, Routing protocol.*

# BAB I

# PENDAHULUAN

## Latar Belakang

*Mobile Ad Hoc Network* (MANET) adalah jaringan nirkabel yang berdiri sendiri, terdiri dari sejumlah *node* yang mampu bergerak ke segala arah secara bebas. Akibat pergerakan yang bebas tersebut topologi jaringan MANET cenderung sering berubah [1]. *Node*-*node* pada jaringan MANET terdiri dari *node* sumber sebagai pengirim dan *node*  tujuan sebagai penerima, dan antar *node* pengirim dan penerima disebut *node* tengah yang berfungsi sebagai perantara kedua *node* tersebut. Masing-masing *node* pada jaringan MANET memiliki level yang sama, artinya semua *node*  berperan sebagai *router* [2].

Dalam hal komunikasi antar *node*, perlu adanya proses *routing*. Proses ini biasanya dikerjakan oleh suatu protokol *routing* agar *node*-*node* mampu saling berkirim data. Protokol *routing* yang digunakan sebagai objek penelitian ini adalah *Ad Hoc On-Demand Multipath Distance Vektor* (AOMDV). AOMDV merupakan protokol *routing* yang bersifat reaktif. AOMDV memiliki mekanisme *route discovery* dan *route maintenance* [2].

Pada AOMDV setiap *route replay* (RREP) akan dipertimbangkan oleh *node* asal sehingga beberapa *path* bisa ditemukan dalam satu pencarian rute. Dengan ditemukannya beberapa *path* atau pilihan rute, apabila terjadi kegagalan rute maka dapat dialihkan ke rute alternatif lain. Dan pencarian rute baru hanya akan dilakukan apabila semua rute yang sudah ditemukan mengalami kegagalan [3]. Pencarian rute pada protokol *routing* AOMDV dilakukan dengan cara memperhitungkan jarak berdasarkan jumlah *hop*. Apabila pada *routing* *table* sudah tidak ada rute cadangan lagi, maka AOMDV akan melakukan pencarian rute kembali dari awal. Pada AOMDV perpindahan rute mengakibatkan banyaknya paket data *drop* saat proses transisi antar rute. Sehingga rute yang stabil sangat diperlukan untuk mengatasi masalah akibat mobilitas *node* tersebut. Perubahan topologi dan mobilitas *node* dapat mengakibatkan rute memiliki kemungkinan terputus yang disebabkan *node* keluar dari jangkauan sinyal transmisi [4]. Namun di sisi lain perubahan topologi ini dapat dimanfaatkan untuk melakukan *redirection* agar mendapatkan rute baru yang handal dengan menghitung aspek energi pada *node*. Sehingga diharapkan mampu meningkatkan *throughput* dan menurunkan *delay*.

*Routing* dalam MANET merupakan suatu tantangan yang menarik karena MANET memiliki fitur yang dinamis, serta dibatasi oleh *bandwidth* dan energi. Setiap *wireless mobile device* pada MANET eksistensinya dibatasi oleh adanya ketersediaan energi. Bila energi sebuah *wireless mobile device* habis, maka *wireless mobile device* tersebut tidak eksis lagi pada MANET [5]. Keterbatasan energi terjadi karena setiap *node* selalu bergantung kepada energi yang terbatas yang dimilikinya dalam rangka menyelesaikan tugas utamanya. Masalah yang dapat dilihat yakni skalabilitas dalam kebijakan *overhead* yang berimbas pada jumlah *node* yang beroperasi di jaringan, juga terkait erat dengan konsumsi energi karena jumlah paket kontrol yang lebih tinggi berarti lebih banyak konsumsi energi yang dihabiskan dalam transmisi, penerimaan (*reception*) dan mendengar (*overhearing*)[6]. Karena adanya keterbatasan energi terdapat kemungkinan terjadi putusnya komunikasi di tengah jalan pada *path* dikarenakan kehabisan energi. Pada penelitian ini diusulkan adanya proses perhitungan energi pada suatu *node*  yang dapat digunakan sebagai rute baru yang berbasis energi dengan memanfaatkan perubahan topologi. Energi disini merupakan kekuatan aliran listrik yang digunakan dalam transfer paket data yang dinyatakan dengan satuan *joule*. Efisiensi energi pada *routing* *protocol* dapat dilakukan dengan menggunakan metode EA-SHORT (*Energy* *Aware – Self Healing and Optimizing Routing Techniques*). Tujuan utama dari algoritma EA-SHORT adalah untuk menghemat energi pada MANET, dengan cara memperpanjang masa pakai *node* dan jaringan dengan merutekan paket melalui *node* yang memiliki sisa daya yang cukup dan menghindari *node* yang memiliki daya rendah. Sehingga dapat meningkatkan kinerja dan konservasi daya pada MANET.

Penelitian ini mengajukan modifikasi pada kerangka protokol *routing* AOMDV untuk mengoptimalkan rute yang mempertimbangkan aspek energi dengan menggunakan algoritma EA-SHORT. Kinerja AOMDV dengan EA-SHORT akan diukur dari nilai parameter yang telah ditentukan yaitu *Packet Delivery Ratio* (PDR), *throughput* dan *end-to-end delay* yang dilakukan pada sejumlah skenario berbeda.

## Rumusan Masalah

Berdasarkan uraian pada latar belakang, maka dapat dirumuskan permasalahan yaitu:

1. Bagaimana menyimulasikan penerapan algoritma EA-SHORT pada kerangka protokol *routing* AOMDV untuk menemukan rute handal berbasis energi?
2. Bagaimana performa protokol *routing* AOMDV setelah dimodifikasi menggunakan algoritma EA*-*SHORT?

## Batasan Masalah

Untuk membatasi ruang lingkup dari permasalahan yang ada, serta agar mencapai tujuan dan sasaran berdasarkan rumusan masalah di atas, maka diberikan beberapa batasan masalah yaitu:

1. Menggunakan algoritma *Energy Aware* SHORT yang diterapkan pada kerangka kerja *routing protocol* AOMDV.
2. Simulasi menggunakan *Network Simulator* 2 versi 2.35 (NS-2.35).
3. Parameter uji yang digunakan adalah *throughput, Packet Delivery Ratio* (PDR) dan *end to end delay*.
4. Tidak membahas mengenai keamanan jaringan.

## Tujuan Penelitian

Tujuan dari penelitian tugas akhir ini sebagai berikut :

1. Menemukan rute handal yang hemat energi dengan menerapkan algoritma *Energy Aware* SHORT pada kerangka protokol *routing* AOMDV.
2. Menganalisaperforma protokol *routing* AOMDV setelah dimodifikasi menggunakan algoritma *Energy Aware* SHORT

## Manfaat Penelitian

Manfaat dari penelitian ini adalah dapat mengetahui kinerja dari protokol *routing* AOMDV setelah dimodifikasi menggunakan algoritma EA-SHORT dalam menentukan rute handal berbasis energi.

## Sistematika Penulisan

Penyusunan sistematika penulisan ini untuk memberikan gambaran materi-materi yang dibahas secara menyeluruh dalam proposal yang terdiri dari lima bab sebagai berikut:

1. BAB I PENDAHULUAN

Pada bab ini menjelaskan tentang latar belakang penulisan tugas akhir, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, dan sistematika penulisan pembuatan Tugas Akhir ini.

1. BAB II DASAR TEORI

Bab ini menjelaskan mengenai tinjauan pustaka yang berkaitan dengan penelitian yang dilakukan sebelumnya dan menjelaskan penelitian yang diusulkan pada tugas akhir, serta berisi uraian mengenai dasar-dasar teori mengenai segala sesuatu yang dibutuhkan dalam penelitian, baik berupa bidang penelitian yang dilakukan, metode atau konsep yang digunakan, dan program yang diperlukan selama penelitian dilakukan.

1. BAB III METODE PENELITIAN

Bab ini berisi perencanaan simulasi jaringan, membahas langkah-langkah yang dilakukan dalam penelitian, konsep penelitian yang digunakan, dan jadwal penelitian yang akan dilakukan.

1. BAB IV ANALISIS HASIL SIMULASI

Dari implementasi yang dilakukan di Bab 3 maka data hasil dari skenario tersebut akan dipaparkan di bab ini. Data yang dihasilkan akan dianalisis dengan metode yang telah dijelaskan di Bab 1, sehingga akan dapat di tarik kesimpulan mengenai data tersebut pada bab selanjutnya.

1. BAB V KESIMPULAN DAN SARAN

Hasil kesimpulan atas analisis data yang dilakukan pada Bab 4 akan dipaparkan pada bab ini, bagaimana hasil dari penelitian Tugas Akhir ini, serta saran yang membangun yang akan berguna sebagai dasar untuk penelitian yang lebih lanjut mengenai Tugas Akhir ini.

# BAB II

# TINJAUAN PUSTAKA DAN DASAR TEORI

## Tinjauan Pustaka

## 2.1.1 Penelitian Terkait

Penelitian terkait sebelumnya adalah mengenai analisis performansi porotokol *routing* AOMDV dan AODV pada MANET [7]. Pada penelitian tersebut dilakukan perbandingan dan evaluasi protokol berdasarkan lima parameter uji yaitu *Packet Delivery Ratio* (PDR), *routing overhead*, *Normalized Routing Load* (NRL), *throughput* dan *average delay*. Hasil simulasi pada skenario perubahan jumlah *node*  menunjukkan bahwa AODV lebih baik dari AOMDV dilihat dari parameter performansi PDR sebesar 4.71%, *throughput* sebesar 9.64%, *routing overhead* sebesar 2936.12% dan NRL sebesar 2993.27%. Namun porotokol *routing* AOMDV memiliki performansi lebih baik dari AODV,dilihat dari parameter performansi *average delay* sebesar 423.198% lebih baik. Selanjutnya hasil simulasi pada skenario perubahan kecepatan *node*  juga menunjukkan bahwa AODV lebih baik dari AOMDV dilihat dari parameter performansi PDR sebesar 13.17%, *throughput* sebesar 16.51%, *routing overhead* sebesar 1480.17% dan NRL sebesar 1669.69%. Dan porotokol *routing* AOMDV memiliki performansi lebih baik dari AODV,dilihat dari parameter performansi *average delay* sebesar 238.256% lebih baik. Sehingga dapat disimpulkan protokol *routing* AOMDV hanya lebih unggul dari segi parameter *average delay* daripada protokol *routing* AODV. Persamaan dari penelitian terkait dengan penelitian yang dilakukan yaitu menggunakan protokol *routing* AOMDV. Sedangkan perbedaan penelitian terkait dengan penelitian yang dilakukan yaitu pada penelitian yang dilakukan menggunakan metode EA-SHORT dan penelitian sebelumnya tidak menggunakan metode.

Penelitian terkait selanjutnya adalah melakukan analisis perbandingan protokol *routing,* AOMDV dan AOMDV yang telah dimodifikasi dengan penambahan algoritma perhitungan *Signal to Noise Ratio* (SNR) ke dalam mekanisme pencarian jalur [8]. Penambahan algoritma tersebut bertujuan agar protokol *routing* dapat mencari jalur dengan kualitas yang lebih baik. Hasil simulasi menunjukkan bahwa kinerja protokol *routing* AOMDV yang dimodifikasi lebih baik dari pada AOMDV biasa di semua parameter dan semua skenario perubahan besar paket dan jumlah *traffic*. Pada skenario perubahan besar paket, *throughput* mengalami peningkatan sebesar 20%, penurunan NRL sebesar 3%, dan penurunan *packet loss* sebesar 0,37%. Pada skenario perubahan jumlah *traffic*, *throughput* mengalami peningkatan sebesar 12%, penurunan NRL sebesar 0,5%, dan penurunan *packet loss* sebesar 0,21%. Penambahan algoritma *Signal to Noise Ratio* (SNR) terbukti dapat meningkatkan performansi protokol *routing* baik itu *throughput,* NRL dan *packet loss* yang lebih baik. Persamaan dari penelitian terkait dengan penelitian yang dilakukan yaitu menggunakan protokol *routing* AOMDV. Sedangkan perbedaan penelitian terkait dengan penelitian yang dilakukan yaitu pada penelitian yang dilakukan menggunakan metode EA-SHORT dan penelitian sebelumnya menggunakan metode SNR.

Penelitian terkait selanjutnya adalah penelitian untuk mengimplementasikan metode perbaikan protokol *routing* [1]. Di mana penelitian Optimalisasi *routing* dengan menggunakan metrik yang berbeda seperti panjang lintasan dan pemakaian energi di sepanjang jalur didefinisikan. Kemudian diusulkan sebuah kerangka kerja *Self-Healing and Optimizing Routing Techniques* (SHORT) untuk MANET. Saat menggunakan SHORT, semua *node*  tetangga memantau rute dan mencoba mengoptimalkannya bila ada jalur yang lebih baik. Pada penelitian ini, algoritma SHORT diterapkan untuk protokol *routing* AODV dan DSR. Algoritma SHORT meningkatkan optimalisasi routing dengan memonitor jalur *routing* secara terus menerus dan secara bertahap mengarahkan rute untuk lebih optimal. Hasil simulasi menunjukkan bahwa tingkat pengiriman yang lebih tinggi dan *network* *lifetime* yang lebih panjang. Tingkat pengiriman naik sekitar 8% pada AODV-SHORT dan 6% pada DSR-SHORT, serta *overhead* mengalami penurunan sekitar 10%. Persamaan dari penelitian terkait dengan penelitian yang dilakukan yaitu algoritma EA-SHORT. Sedangkan perbedaan penelitian terkait dengan penelitian yang dilakukan yaitu pada penelitian yang dilakukan menggunakan protokol *routing* AOMDV namun penelitian sebelumnya menggunakan protokol *routing* DSR dan AODV.

Penelitian terkait selanjutnya tentang peningkatan efisiensi yang dilakukan pada protokol *routing* AOMDV menggunakan metode PA-SHORT di jaringan MANET [4]. Penelitian ini membandingkan kinerja protokol *routing* AOMDV dengan protokol *routing Path Aware*-AOMDV (PA-AOMDV). Protokol *routing* PA- AOMDV diperoleh melalui modifikasi kinerja protokol AOMDV dengan algoritma *Path Aware* SHORT. Algoritma *Path Aware* SHORT adalah metode untuk mengurangi jumlah *hop*. SHORT meningkatkan optimisasi *rute* dengan memantau *rute* dan mengoptimalkan *rute* ini yang memiliki jalur yang lebih baik. Kinerja kedua protokol akan dilihat berdasarkan empat parameter, yaitu *throughput*, *averange end- to-end to delay*, *packet delivery ratio*, dan *routing overhead*. Hasil menunjukkan bahwa *throughput* meningkat untuk 50 *node* adalah 61,84% dan untuk 100 *node* adalah 45,2%, *averange end-to-end to delay* menurun untuk 50 *node* adalah 0,066% dan untuk 100 *node* 0,12%, *packet delivery ratio* meningkat untuk 50 *node* adalah 60,87% dan untuk 100 *node* 82,02%, dan routing overhead menurun untuk 50 *node* adalah 67,07% dan 100 *node*  45,36%. Persamaan dari penelitian terkait dengan penelitian yang dilakukan yaitu menggunakan simulasi area 500m x 500m dan 1000m x 1000m serta menggunakan protokol *routing* AOMDV. Sedangkan perbedaannya yaitu pada penelitian yang dilakukan menggunakan algoritma EA-SHORT namun penelitian sebelumnya menggunakan algoritma PA-SHORT.

Penelitian terkait selanjutnya telah dilakukan analisis perbandingan kinerja protokol *routing* TORA dan TORA dengan menggunakan algoritma SHORT [9]. Hasil simulasi menunjukkan bahwa *throughput* yang lebih tinggi meningkat hingga 3-50%, *end-to-end delay* yang lebih rendah hingga 50%, *delay* mengalami penurunan sebesar 50%, penurunan *packet loss* bervariasi dari 12-57%, perbaikan energi sekitar 43%, dan *life time* jaringan yang lebih tinggi. TORA yang dimodifikasi mengungguli protokol routing TORA dalam hal *throughput*, *packet loss*, *end-to-end delay*, dan energi. Persamaan dari penelitian terkait dengan penelitian yang dilakukan yaitu menggunakan algoritma EA-SHORT. Sedangkan perbedaannya yaitu pada penelitian yang dilakukan menggunakan protokol *routing* AOMDV namun penelitian sebelumnya menggunakan protokol *routing* TORA.

## 2.1.2 Penelitian yang Diusulkan

Pada penelitian terkait sebelumnya telah dijelaskan kelemahan dan kelebihan protokol *routing* AOMDV tanpa metode dan keunggulan algoritma *Energy Aware* pada beberapa jenis protokol *routing* di jaringan MANET. Penelitian ini mengajukan modifikasi pada kerangka protokol AOMDV untuk mengoptimalkan *rute* utama menggunakan algoritma *Energy Aware* SHORT (EA-SHORT). Pengoptimalan *rute* dengan EA-SHORT mencoba menemukan ruteutama terbaikketika terjadi mobilitas *node* yang mengakibatkan perubahan bentuk *rute* untuk kecepatan pengiriman paket data. Sehingga dengan diterapkannya algoritma EA-SHORT pada protokol *routing* AOMDV diharapkan dapat mengurangi *latency* dan meningkatkan *throughput*. Kemudian hasil tersebut akan dibandingkan dengan hasil simulasi protokol *routing* AOMDV tanpa modifikasi. Kedua hasil tersebut dibandingkan berdasarkan tiga parameter, yaitu *throughput, average end to end delay* dan *packet delivery ratio*, pada skenario yang berbeda dengan memvariasikan beberapa parameter simulasi.

## Dasar Teori

Dalam penelitian Tugas Akhir ini digunakan beberapa teori untuk melandasi pemecahan masalah yang berisi penjelasan mengenai *tools* dan segala sesuatu yang berkaitan dan digunakan dalam penelitian.

1. ***Mobile ad-hoc network***

*Ad-Hoc* secara bahasa berarti “*untuk suatu keperluan atau tujuan tertentu saja*“. Dalam pengertian lain, jaringan *ad-hoc* adalah jaringan bersifat sementara tanpa bergantung pada infrastruktur yang ada dan bersifat independen [10]*. Mobile Ad-hoc network* (MANET) merupakan sebuah jaringan yang terdiri dari gabungan perangkat-perangkat bergerak (*mobile*) tanpa infrastruktur, sehingga membentuk jaringan yang bersifat sementara. Tiap perangkat memiliki antarmuka nirkabel dan saling berkomunikasi melalui gelombang radio, kemudian tiap perangkat tersebut dinamakan *node* . Beberapa contoh *ad-hoc node* yaitu laptop dan *personal digital assistants* (PDA) yang saling berkomunikasi secara langsung satu sama lain. (Larsson dan Hedman, 1998).

MANET diharapkan menjadi lebih besar lagi. Diperlukan *router* yang teta (*fixed-router*) maupun lokasi yang tetap (*fixed-location*) pada infrastruktur jaringan, seperti terlihat pada Gambar 2.1(a). sedangkan pada MANET hal ini tidak diperlukan, seperti terlihat pada Gambar 2.1(b). Contoh infrastruktur jaringan adalah jaringan selular, *Local Area Network* (LAN) atau *Wireless Local Area Network* (WLAN) [7].

|  |  |
| --- | --- |
|  |  |
| 1. Infrastruktur Jaringan | 1. *Mobile ad-hoc network*. |

Gambar 2.1 Struktur jaringan nirkabel [7].

Dalam MANET, *node*  juga berfungsi sebagai *router* yang meneruskan paket ke *node*  lainnya. *Node*  dapat bergerak secara bebas, tidak tergantung satu sama lain, topologi pada jaringan ini terus berganti secara dinamis yang membuat *routing* semakin susah. Oleh karena itu, *routing* merupakan salah satu hal yang paling diperhatikan dalam jaringan ini. Protokol *routing* normal yang bekerja dengan baik pada jaringan tetap tidak memperlihatkan performa yang sama pada MANET. Pada jaringan ini protokol *routing* harus selalu dinamis sehingga dapat menanggapi pergantian topologi.

Contoh Penggunaan MANET di lapangan [10]:

1. Operasi militer, seperti yang telah diuji coba pada kawasan pertempuran di Sudan. Dengan jaringan *ad-hoc*, mempermudah untuk akses informasi antar personil militer.
2. Komersial, jaringan *ad-hoc* dapat digunakan pada situasi *emergency* atau upaya penyelamatan (*rescue operation*), seperti banjir atau gempa bumi dan *entertainment* seperti acara *live music*.
3. Jaringan yang cepat tersedia dengan menggunakan *notebook* untuk menyebarkan dan berbagi informasi di antara *user* seperti dalam konferensi atau ruang kuliah.
4. Personal Area *Network*, untuk jarak pendek (*short distance*) lebih kurang 10m, *Ad hoc Network* secara mudah berkomunikasi antar bermacam peralatan (seperti PDA, laptop dan telepon seluler) dengan laju data yang rendah.

Adapun keuntungan MANET yaitu di antaranya [10]:

1. Tidak memerlukan dukungan *backbone* infrastruktur sehingga mudah diimplementasikan dan sangat berguna ketika infrastruktur tidak ada ataupun tidak berfungsi lagi.
2. Mobile *node*  yang selalu bergerak (*mobility*) dapat mengakses informasi secara *real time* ketika berhubungan dengan *mobile* *node*  lain, sehingga pertukaran data dan pengambilan keputusan dapat segera dilaksanakan.
3. Fleksibel terhadap suatu keperluan tertentu karena jaringan ini memang bersifat sementara.
4. Dapat direkonfigurasi dalam beragam topologi baik untuk jumlah *user* kecil hingga banyak sesuai dengan aplikasi dan instalasi (*scalability*).

Kerugian (Permasalahan) MANET di antaranya [10]:

1. *Packet loss* akan terjadi bila transmisi mengalami kesalahan (*error*).
2. Sering kali terjadi *disconnection*, karena tidak selalu berada dalam area cakupan.
3. *Bandwidth* komunikasi yang terbatas
4. *Lifetime* baterai yang singkat.
5. kapasitas kemampuan jangkauan *mobile* *node* yang terbatas dan bervariasi.

**2.2.2 Protokol *routing* jaringan *ad-hoc***

Protokol adalah seperangkat aturan yang mengatur setiap komputer untuk saling bertukar informasi melalui media jaringan, sedangkan *routing* adalah proses memindahkan informasi dari pengirim ke penerima melalui sebuah jaringan. *Protocol routing* diperlukan karena untuk mengirimkan paket data dari *node*  pengirim ke *node*  penerima akan melewati beberapa *node*  penghubung (*intermediate* *node* ), dimana protokol *routing* berfungsi untuk pencarian *route link* yang terbaik dari *link* yang akan dilalui melalui mekanisme pembentukan tabel *routing*. Pemilihan rute terbaik tersebut didasarkan atas beberapa pertimbangan seperti *bandwith link* dan jaraknya [7].

Mekanisme protokol *routing* yang ada di MANET yaitu:

1. Protokol *Routing* Proaktif

Protokol *routing* proaktif merupakan protokol *routing* yang berdasarkan informasi pada *routing table* yang terus di-*update* dalam waktu berkala. Semua informasi mengenai perubahan akan dikirim sesuai dengan periodik pengiriman *update* dari *routing table*. Sehingga, pada protokol *routing* ini akan terjaga *routing table* yang terus *update* tanpa menghilangkan fungsinya untuk memilih jalur terbaik untuk mengirimkan data. Sifat protokol ini memungkinkan *delay* yang lebih rendah dalam mengirimkan data melalui jaringan karena jalur data sudah dikenal. Beberapa contoh protokol *routing* jenis ini adalah *Optimized Link State Routing* (OLSR), *Better Approach To Mobile Ad-hoc Network* (B.A.T.M.A.N), *Destination-Sequenced Distance Vector* (DSDV) dan *Multipath Dynamic Address Routing Protocol* (MDART).

1. Protokol *Routing* Reaktif

Protokol *routing* reaktif merupakan protokol *routing* yang memilih jalur maupun melakukan *update* jalur hanya ketika terdapat *rute* baru atau ketika suatu *rute* terputus. Pada protokol *routing* ini, perhitungan jalur dilakukan sekali saja, kemudian di-*update* hanya ketika ada perubahan. Informasi yang dimiliki setiap *router*  pada protokol *routing* ini sangat terbatas dan akan dihapus ketika tidak lagi dibutuhkan dalam jangka waktu tertentu. Pada beberapa protokol *routing* reaktif, setiap *router*  bahkan hanya memiliki informasi tentang *nexthop* saja. Oleh karena karakteristiknya, protokol *routing* reaktif sering juga disebut dengan *On-Demand* Protokol *routing*. Contoh protokol *routing* ini adalah *Dynamic Source Routing* (DSR), *Ad hoc On-Demand Distance Vector* (AODV) dan *Ad hoc On-Demand Multipath Distance Vector* (AOMDV).

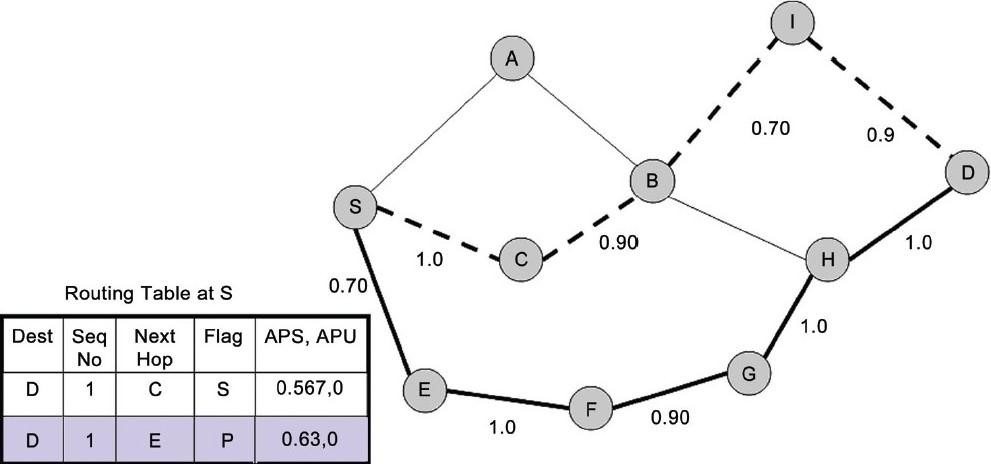
**2.2.3 *Ad-hoc On-demand Multipath Distance Vector* (AOMDV)**

MANET memiliki karakteristik topologi yang dinamis, *bandwidth* kanal yang terbatas dan *power* pada *node*  yang terbatas pula. Akibat dari karakteristik-karakteristik tersebut, *path* yang menghubungkan *node*  asal ke *node*  tujuan menjadi tidak stabil dan bisa *down* kapan saja sehingga dimungkinkan membangun lebih dari satu *path* antara *node*  asal dan *node*  tujuan. Hal ini digunakan untuk menyediakan rute *backup*, ketika rute utama gagal dalam mengirimkan paket ke *node*  tujuan, rute *backup* inilah yang selanjutnya akan dipakai. Sehingga menawarkan toleransi kesalahan yang lebih baik dan juga dalam hal penanganan kegagalan rute yang lebih efisien [7].

*Ad Hoc On-Demand Multipath Distance Vector* atau biasa disingkat AOMDV merupakan protokol *routing* bersifat reaktif hasil pengembangan dari protokol *routing* AODV. Perbedaannya dengan AODV adalah pada jumlah rute yang ditemukan. AODV menemukan satu rute saja, sedangkan AOMDV menemukan lebih dari satu rute. Pengembangan ini memiliki tujuan untuk mengurangi risiko terjadinya kegagalan rute pada jaringan. Mirip seperti protokol AODV, AOMDV juga memiliki mekanisme *route* *discovery* dan *route maintenance*. Paket yang dikirimkan ketika mencari rute juga sama yaitu paket RREQ dan RREP. Untuk proses *route maintenance*, AOMDV juga akan mengirim paket RERR seperti halnya AODV [10].

Keuntungan adanya pada satu kali pencarian rute adalah ditemukannya beberapa rute, jika rute utama mengalami kerusakan maka akan digunakan rute lain yang masih ada pada *routing table* tanpa harus mencari rute lagi dari awal. Pencarian rute dari awal hanya dilakukan ketika semua rute yang ditemukan sudah tidak valid lagi. Pada [11] dijelaskan AOMDV memiliki tiga kelebihan dibanding *multipath routing* lain yaitu. Pertama, AOMDV tidak memiliki *overhead* koordinasi antar *node* yang tinggi karena komunikasi pada AOMDV hanya dilakukan saat dibutuhkan saja. Kedua, AOMDV menjamin rute alternatif saling *disjoint* melalui komputasi yang terdistribusi pada tiap *node* tanpa perlu komputasi dari *node* sumber saja sehingga rute yang ditemukan diharapkan tidak akan terjadi *loop*. Ketiga, AOMDV menghitung atau menemukan alternatif rute dengan tambahan *overhead* yang minim dibandingkan dengan AODV[12]. Hal tersebut dilakukan dengan memanfaatkan semaksimal mungkin informasi *routing* jalur alternatif yang sudah ada. Inti dari protokol AOMDV adalah menjamin *multiple path* yang ditemukan adalah *loop-free*, *disjoint*, dan efisien dalam pencariannya. Aturan *update* rute pada AOMDV dilakukan secara mandiri oleh tiap *node*.

Protokol AOMDV menggunakan tiga jenis *control messages*, yaitu *Route Request* (RREQ), *Route Reply* (RREP) dan *Route Error* (RRER) untuk mencari dan memelihara suatu rute. Pesan kontrol ini dikirim melewati port 654 dan menggunakan *User Datagram Protocol* (UDP) sebagai *transport-*nya. Ketika ada suatu *node* yang membutuhkan rute untuk mengirimkan paket data, AOMDV mengirimkan RREQ secara *broadcast* ke seluruh jaringan. Ketika suatu *node* menerima paket RREQ, *node* tersebut akan memeriksa alamat tujuan dalam RREQ tersebut. Jika *node* tersebut memiliki informasi tentang alamat tujuan itu, atau *node* tersebut merupakan tujuan dari RREQ tersebut, maka *node* tersebut akan mengirimkan paket RREP secara *unicast* ke *node* sumber. Ketika sebuah *node* tidak memiliki rute ke tujuan dan jalur lain tidak tersedia, maka *node* tersebut akan mengirimkan paket RRER ke *node* sumber[13].



Gambar 2. 2 Contoh rute multipath.

**2.2.4 *Energy aware* *routing***

Baterai perangkat seluler yang ringan biasanya memiliki catu daya yang terbatas. *Interface* pada jaringan nirkabel telah diidentifikasi sebagai sumber utama konsumsi daya pada komputer seluler, bersama dengan layar LCD dan *hard* *disk*. Dengan demikian, ketika mempertimbangkan untuk memperpanjang umur atau daya perangkat *mobile*, serta masa pakai seluruh jaringan, efisiensi energi dari protokol *routing* merupakan masalah yang menonjol. Pada penelitian [1] telah dilakukan dengan tujuan menemukan jalur *multi*-*hop* biaya minimum dalam hal konsumsi energi di sepanjang rute metrik *energy aware* yang berbeda untuk menentukan rute telah diusulkan. Penelitian telah dilakukan dengan tujuan untuk menemukan jalur *multi-hop* dengan biaya minimum dalam hal konsumsi energi di sepanjang rute. Metrik *power aware* yang berbeda untuk menentukan rute telah diusulkan. Metrik didasarkan pada tingkat daya baterai dan konsumsi energi di setiap *node* . Tujuan dari *Energy aware* adalah untuk mengembangkan kerangka kerja yang dapat meningkatkan metrik konsumsi daya melalui optimasi rute. Dinamika jaringan dapat mengubah kondisi tautan nirkabel dengan cepat.

Protokol yang sadar energi memperpanjang masa pakai *node*  dan jaringan dengan merutekan paket melalui *node*  yang memiliki daya yang tersisa yang memadai, dan menghindari *node*  yang kekurangan pasokan baterai. Untuk mencapai keadilan perutean, protokol perutean harus dibiarkan untuk sembuh sendiri sehingga dapat menarik dirinya keluar dari situasi di mana *node*  tertentu sedang digunakan berlebihan sementara *node*  lain menganggur. *Energy aware* protokol akan dapat mengalihkan lalu lintas dari area yang dimuat dan menyeimbangkan beban pada semua *node*  dalam jaringan[1].

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

Pada penelitian ini penulis menggunakan NS2 (*Network Simulator* 2) untuk mensimulasikan protokol *routing* pada MANET. NS2 merupakan suatu simulator jaringan yang mendukung banyak aplikasi, protokol, unsur-unsur jaringan, dan model-model *traffic*. NS2 memiliki dasar dari dua bahasa pemrograman yaitu C++ yang digunakan untuk menuliskan simulator berorientasi objeknya dan *interpreter* OTcl ( yang merupakan suatu ekstensi berorientasi objek Tcl) yang digunakan untuk menjalankan *script* perintah dari pengguna [7].

NS-2 adalah suatu *interpreter* yang *object-oriented* dan *discrete event-driven* yang bersifat *open source* untuk menunjukkan simulasi jaringan yang melibatkan *Local Area Network* (LAN), *Wide Area Network* (WAN), jaringan nirkabel (*wireless*), dan jaringan *ad-hoc*. Keuntungan menggunakan *Network Simulator* sebagai alat simulasi pembantu analisis dalam riset, antara lain adalah *Network Simulator* ini dilengkapi dengan *tools* validasi yang digunakan untuk menguji kebenaran pemodelan suatu sistem. Secara *default*, semua pemodelan akan dapat melewati proses validasi ini, seperti media, protokol dan komponen jaringan yang lengkap dengan perilaku *traffic*nya sudah disediakan pada *library* *Network Simulator* [11].

*Network* Simulator (NS) dibangun menggunakan C++ dan Tcl/Otcl. C++ digunakan sebagai *library* yang berisi *event scheduler*, protokol, dan komponen jaringan, Tcl/Otcl digunakan pada *script* simulasi yang ditulis oleh pengguna NS. C++ digunakan karena mampu mendukung *runtime* simulasi yang cepat, meskipun simulasi melibatkan simulasi jumlah paket dan sumber data dalam jumlah besar. *Network* Simulator juga mendukung untuk animasi jaringan yang dibuat. Animasi dijalankan menggunakan *Network* Animator (NAM). *Nam* sendiri merupakan alat untuk animasi berbasis Tcl/TK dan dibuat untuk melihat simulasi jaringan dan paket yang dikirim pada jaringan tersebut[13].

# BAB III

# METODE PENELITIAN

## Diagram Alir Penelitian

Diagram alir penelitian Penerapan algoritma EA-SHORT pada protokol *routing* AOMDV untuk menemukan rute yang handal berbasis energi di jaringan MANET, dapat dilihat pada Gambar 3.1.

|  |
| --- |
|  |

Gambar 3.1 Diagram alir penelitian.

1. **Studi literatur**

Pada tahap studi literatur, dilakukan pengkajian mengenai berbagai penelitian yang telah dilakukan sebelumnya yang terkait dengan penelitian yang akan dilakukan. Pada tahap ini, dilakukan proses pembelajaran materi penelitian melalui pustaka-pustaka baik berupa buku maupun jurnal ilmiah. Adapun pustaka-pustaka yang digunakan di antaranya mengenai performa protokol *routing* AOMDV, teknik untuk memodifikasi *routing protocol*, proses perhitungan energi pada algoritma *Energy aware* untuk meningkatkan pengiriman data pada *mobile ad-hoc network*, dan beberapa penelitian lainnya yang mendukung penelitian yang akan dilakukan.

1. **Menentukan spesifikasi *hardware* dan *software***

Pada penelitian ini uji coba dan simulasi dilakukan dengan menggunakan Laptop HP 14-bs538TU dengan spesifikasi *hardware* (perangkat keras) yang dapat dilihat pada Tabel 3.1.

Tabel 3.1 Spesifikasi hardware yang digunakan.

|  |  |
| --- | --- |
| **Komponen** | **Spesifikasi** |
| CPU | Intel(R) Celeron(R) CPU N3060 @1.60 GHz |
| Memori | 4.00 GB (2 GB untuk virtual simulator) |
| *Hard disk* | 500 GB (10 GB untuk virtual simulator) |

Adapun *software* (perangkat lunak) yang digunakan pada penelitian ini adalah sebagai berikut :

1. *Oracle* VM *VirtualBox* 6.0, digunakan untuk menjalankan simulator. Di mana alokasi pengguanaan memori komputer yang digunakan sebesar 2 GB dan penggunaan *harddisk* untuk menyimpan data sebesar 10GB.
2. *Network Simulator 2* (NS2) versi 2.35 untuk melakukan simulasi MANET.
3. *Linux Ubuntu* 16.04 LTS 64-bit sebagai sistem operasi.
4. *Microsoft Excel* untuk membuat grafik hasil simulasi.
5. *Microsoft Visio* untuk membuat diagram
6. *Microsoft Word* untuk dokumentasi laporan
7. **Pembuatan *script* EA-SHORT AOMDV**

Pembuatan *script* EA-SHORT AOMDV dilakukan dengan cara memodifikasi kerangka protokol *routing* AOMDV dengan menyisipkan Algoritma EA-SHORT pada protokol *routing* AOMDV. Penulisan kode dilakukan melakukan perhitungan berbasis energi pada *node*  untuk menemukan rute hemat energi yang handal. Jika rutetelah ditemukan maka jalur tersebut yang akan digunakan untuk melakukan pengiriman data.

1. **Perancangan skenario simulasi**

Pada bagian ini, peneliti melakukan perancangan skenario simulasi jaringan MANET untuk melakukan analisis performa algoritma EA-SHORT pada protokol *routing* AOMDV dengan kondisi, yaitu protokol AOMDV belum menggunakan EA-SHORT dan kondisi dimana AOMDV sudah dimodifikasi menggunakan EA-SHORT. Adapun topologi jaringan yang digunakan dengan jenis protokol *routing* AOMDV dengan jenis *traffic* menggunakan *Constant Bit Rate* (CBR) serta menggunakan mobilitas *node Random Waypoint*.

1. **Menjalankan skenario simulasi dengan NS2**

Pada tahap ini perancangan skenario simulasi yang telah dibuat akan dijalankan pada simulator NS2 sehingga diperoleh hasil berdasarkan skenario simulasi yang diujikan. Terdapat beberapa parameter uji yang digunakan untuk melakukan analisa hasil pengujian dari skenario simulasi yang dilakukan yaitu *throughput*, *Packet Delivery Ratio* (PDR) dan *end to end delay* (E2D).

1. **Analisis hasil simulasi dari NS2**

Pada tahap ini dilakukan proses analisis terhadap hasil yang diperoleh dari proses pengujian skenario penelitian yang dilakukan, dengan membandingkan hasil simulasi protokol *routing* AOMDV dan protokol *routing* EA*-* SHORT AOMDV dengan percobaan pengiriman data dengan mobilitas *node* berdasarkan jumlah *node* , luas area, dan kecepetan *node* . Menggunakan *script* AWK akan dianalisis parameter berupa *throughput*, *packet delivery ratio* (PDR) dan *end to end delay* (E2D). Hasil Analisa simulasi ditampilkan dalam bentuk grafik.

1. **Membuat kesimpulan**

Pada tahap ini akan dilakukan *review* terhadap hasil modifikasi protokol *routing* AOMDV dan pengujian skenario, untuk mengetahui performa terbaik dari skenario yang telah diuji. Sehingga, dapat ditarik kesimpulan dari penelitian yang dilakukan serta saran sebagai masukan untuk penelitian selanjutnya.

1. **Membuat laporan**

Pada tahap ini akan dilakukan pembuatan laporan dokumentasi terhadap penelitian yang telah dilakukan. Dokumentasi laporan ini diharapkan agar dapat membantu untuk melakukan penelitian berikutnya yang berkaitan dengan penelitian yang dilakukan.

## Algoritma Penemuan Rute Menggunakan Protokol *Routing* AOMDV

AOMDV merupakan konsep *distance vector* yakni pencarian rute berdasarkan jarak terdekat dan menggunakan pendekatan *hop-by-hop routing*[4]. AOMDV menemukan rute berdasarkan permintaan dengan menggunakan prosedur penemuan rute. Keuntungan adanya pada satu kali pencarian rute adalah ditemukannya beberapa rute, jika rute utama mengalami kerusakan maka akan digunakan rute lain yang masih ada pada *routing table* tanpa harus mencari rute lagi dari awal. Pencarian rute dari awal hanya dilakukan ketika semua rute yang ditemukan sudah tidak valid lagi.

Misalkan, ketika *node* sumber *A* akan mengirimkan suatu paket data ke *node* tujuan *B*, pertama-tama *node* akan melihat tabel *routing* apakah sudah ada jalur yang tersedia. Jika jalur dari *A* ke *B* sudah tersedia dan siap digunakan, maka paket data akan dikirim menggunakan jalur yang sudah ada tersebut. Jika jalur dari *A* ke *B* belum tersedia, maka akan dimulai proses pencarian jalur dengan mengirimkan pesan RREQ secara *broadcast*. AOMDV menghitung *multiple path* dan melihat *route advertisement* untuk menentukan jalur alternatif dari sumber ke tujuan selama proses pencarian jalur. Untuk menghindari kemungkinan *loop*, digunakan *advertised hop count* yang ada dalam *routing table*. *Advertised hop count* dari sebuah *node* sumber *A* untuk tujuan *B* adalah kumpulan *hop count* maksimal untuk banyak jalur untuk tujuan *B*. *Advertised hop count* dimulai setiap ada pembaruan *sequence number*. Dengan begitu, AOMDV hanya mengambil rute alternatif dengan jumlah *hop* yang lebih sedikit[13]. Pada AOMDV, propagasi RREQ dari sumber ke tujuan menetapkan beberapa *reverse* *paths* baik di *node*  tengah maupun tujuan. Beberapa RREP melintasi lintasan mundur ini untuk membentuk beberapa lintasan maju ke tujuan di *node*  sumber dan tengah. AOMDV juga menyediakan *node*  tengah dengan jalur alternatif karena mereka bermanfaat untuk mengurangi frekuensi penemuan rute[14].

Ketika *node*  tengah memperoleh jalur baik melalui salinan paket RREQ, *node*  ini akan mengecek apakah ada satu atau lebih *forward path* yang valid ke *node*  tujuan. Jika ada, *node* akan menghasilkan paket RREP *(Route Replay*) dan akan mengirimkannya kembali ke *node* sumber melalui rute cadangan. Jika tidak ada, maka *node*  tengah akan meneruskan paket RREQ hingga ke *node*  tujuan., kemudian *node*  tujuan akan mebalas dengan mengirim RREP. Saat *node*  tujuan menerima salinan RREQ, *node*  tersebut juga membuat rute cadangan dengan cara yang sama dengan yang dilakukan oleh *node*  tengah. Alur proses *route discovery* pada protokol AOMDV dapat dilihat pada Gambar 3.2.

|  |
| --- |
|  |

Gambar 3.2 *Flowchart route discovery* pada AOMDV.

|  |
| --- |
|  |

Gambar 3.3 Proses *route discovery* pada AOMDV.

Proses *route discovery* pada AOMDV dapat dilihat pada Gambar 3.3, dimana node S merupakan *source node* dan node D merupakan *destination node.* Saat *node* S ingin mengirimkan paket ke *node* D. Pertama kali *node* S mengirimkan paket RREQ secara *broadcast* ke *node* tetangganya yaitu *node* U, *node* V dan *node* W. Karena *node* U, *node* V dan *node* W bukan *destination node,* maka ketiga *node* tersebut akan meneruskan paket RREQ secara *broadcast* ke *node* tetangganya kecuali *source node*. Setelah itu *node* U, *node* V dan *node* W akan melakukan pengaturan rute cadangan. Proses tersebut berulang hingga paket RREQ diterima oleh *destination node* (*node* D). Kemudian *node* D akan mengirim paket RREP sebagai balasan dari paket RREQ yang diterimanya. *Node* penerima paket RREP akan melakukan pengaturan untuk *forward path.*

## Algoritma EA*-*SHORT

Tujuan dari penyeimbangan muatan yang sadar energi adalah untuk mendistribusikan secara adil beban lalu lintas di antara semua *node*  yang berpartisipasi di jaringan [1]. Misalnya, pertimbangkan bagian dari MANET ditunjukkan pada Gambar 3.4 (a), jalur S-B-D-F-R adalah jalur optimal untuk koneksi dari S ke R. Metrik untuk optimalitas dapat berupa jumlah *hop* untuk rute jalur terpendek. Dengan demikian, *node*  B, D dan F akan terus digunakan dalam meneruskan lalu lintas, meninggalkan *node*  lain bebas dari beban lalu lintas. Akibatnya, tingkat energi residu dari *node*  menjadi sangat bervariasi. Jika perutean tidak sadar energi, itu akan tetap menggunakan jalur untuk koneksi S-R. *Node*  B, D dan F pada akhirnya akan kehabisan persediaan baterai dan mati lebih awal. Namun, skema perutean yang sadar energi akan mencoba mengalihkan lalu lintas ke *node*  lain. Pada penelitian ini diusulkan skema EA-SHORT berdasarkan *route redirection* (pengalihan rute). Dengan operasi *redirection* lokal yang berurutan, rute secara bertahap akan konvergen ke rute alternatif *node disjoint*.

|  |  |
| --- | --- |
| 1. Jalur awal yang menguras energi pada *node* B, D dan F. | 1. Setelah *redirection, node* B diabaikan. |
| 1. Setelah *redirection, node* D diabaikan. | 1. Setelah *route* *redirection,* rute baru diabaikan. |

Gambar 3.4 Contoh operator pengalihan rute lokal berturut-turut (tautan yang ditunjukkan pada gambar adalah tautan nirkabel).

Perhatikan contoh yang ditunjukkan pada Gambar 3.4 (a), *node* B, D dan F, ketiganya memiliki tingkat energi yang relatif rendah. Setelah adanya *route redirection*, *node* B diabaikan dan jalur S-B-D-F-R menjadi S-A-D-F-R, yang ditunjukkan pada Gambar 3.3 (b). Kemudian, setelah *redirection* pertama dan seterusnya seperti pada Gambar 3.3 (c) memberikan dampak, jalur S-A-C-E-R yang baru akan menggantikan yang asli, seperti yang ditunjukkan pada Gambar 3.3 (d).

Berikut ini langkah-langkah pencarian rute dengan algoritma EA-SHORT. Jika suatu *node* (i) mengetahui kedatangan paket P. Maka;

1. *Node* akan melakukan pengecekan rute untuk paket P pada *tabel* *overhear* . Tabel *overhear* merupakan tabel yang digunakan oleh *node* untuk menyimpan rute suatu paket.
2. Jika tidak ada rute yang cocok. Tambahkan *input* variabel e’, dimana rute (e’) sama dengan rute (p), *sequence number* (e’) sama dengan *sequence number* (p) lalu jadikan rute (e’) sebagai list rute pertama pada *tabel* *overhear* dengan isi (e’) berupa *hope count* (p), level energi (p), dan nomer id (p). proses selesai.
3. Misalkan rute (p) ditemukan pada *input*-an e, dengan kondisi *sequence number* (p) kurang dari *sequence number* (e), maka p akan diabaikan, dan proses selesai.
4. Jika *sequence number* (p) lebih besar dari *sequence number* pada (e), maka perbaharui e dengan ketentuan yakni *sequence number* (e) sama dengan *sequence number* (p), list rute pada tabel *overhear* (e) direset sehingga memiliki satu *input*-an rute dengan isi *hope count* (p), level energi (p). nomer id (p) dan proses selesai.
5. Jika *sequence number* (p) sama dengan *sequence number* (e), maka:
   1. Tambahkan *input*-an <hc (p), lvl (p), nid (p)> ke ov-list (e).
   2. Jika daftar *overhear* pada (e) memiliki tiga *input*-an yakni A,B,C yang memenuhi kondisi berikut, maka subpath ditemukan.
      1. *Hope count* pada (C) sama dengan *hope count* (B)+1 dan sama dengan *hope count* (A)+2
      2. Level energi pada (*node* i) lebih besar atau sama dengan jumlah maksimal energi pada ((A) dan level energi pada (C));
      3. level energi pada (*node* i) – level energi (B) ≥ 2 , maka akan diaktifkan sub-rute baru, dan hapus *input*-an (e) dari tabel *overhear*.
   3. Jika daftar *overhear* pada (e) memiliki dua *input*-an, yakni A dan B, dengan kondisi *hope count* (B) sama dengan *hope count* (A)+1 dan level energi pada *node* (i) ≥ maksimal level energi (A) dan dan level energi (B)+2, tambahkan indikator I pada daftar tunggu indikator (*Waiting Indicator* *List*), dengan kandidat (I) sama dengan B. *sequence number* (I) disamakan dengan (e). rute (I) sama dengan rute (e), maka proses selesai.
   4. Jika daftar *overhear* (e) memiliki dua *input*-an B dan C, dengan kondisi *hope count* (C) sama dengan *hope count* (B)+1 dan level energi pada *node* (i) ≤ dari maksimal level energi pada (B)+2 dan level energi (C), maka *node* I akan menyebarkan pesan SHORT untuk menginformasikan paket Q dengan ketentuan kandidat (Q) sama dengan B, *sequence number* pada (Q) sama dengan *sequence number* pada (e), dan rute (Q) sama dengan rute pada (e).

Ketika *node* i menerima paket informasi pesan SHORT Q, maka:

* + - 1. Bandingkan bidang Q dengan *input*-an yang valid dalam *WaitingIndicator list*.
      2. Jika tidak ada yang cocok, maka abaikan paket Q;

Jika kondisi tidak terpenuhi, maka subrute yang lebih baik ditemukan. Aktifkan subrute ini dengan memperbaharui tabel *routing*.

1. Proses selesai.

Untuk lebih jelasnya, pencarian rute menggunakan algoritma EA-SHORT ditunjukkan pada Gambar 3.5 dan Gambar 3.6.

|  |
| --- |
|  |

Gambar 3.5 *Flowchart* algoritma EA-SHORT.

|  |
| --- |
|  |

Gambar 3.6 *Flowchart* algoritma EA-SHORT (Lanjutan Gambar 3.5).

## Lingkungan Simulasi

Lingkungan simulasi pada penelitian ini akan diterapkan pada protokol *routing* AOMDV dan protokol *routing* AOMDV yang telah dimodifikasi dengan algoritma EA-SHORT. Pada penelitian ini simulasi dilakukan dengan menggunakan *Network Simulator* 2 (NS-2). Pada NS-2 dibutuhkan pola *traffic*, skenario mobilitas *node* dan *script tcl.* Pola *traffic* mendefinisikan koneksi antara *source node* dengan *destination node* dan tipe koneksi yang digunakan (UDP). Skenario mobilitas *node* mendefinisikan posisi awal *node* dalam topologi jaringan, kecepatan mobilitas *node* dan tingkat mobilitas *node*. *Script tcl* merupakan kode program yang berfungsi untuk menciptakan objek simulator, mendefinisikan topologi jaringan, *load* pola *traffic* dan *load* skenario mobilitas *node*. Hasil simulasi dari performa protokol *routing* AOMDV dan protokol *routing* AOMDV yang telah dimodifikasi akan dibandingkan dengan menggunakan parameter pengujian yang telah ditentukan.

1. **Parameter Pengujian**

Adapun beberapa parameter uji yang digunakan untuk menganalisa hasil pengujian dari skenario simulasi yang dilakukan yaitu:

1. *Throughput* : Kecepatan (*rate*) jumlah paket data yang berhasil diterima di sisi penerima pada tiap detik[14]. Semakin tinggi nilai *throughput* maka semakin handal suatu jaringan, hal ini dikarenakan paket data yang berhasil diterima semakin meningkat. *Throughput* dihitung menggunakan persamaan 3.1 sebagai berikut.

Rumus:

*Throughput* (Kbps) = (3.1)

1. *Average end-to-end delay* : waktu jeda yang dibutuhkan dalam pengiriman paket dari pengirim ke penerima paket. Semakin rendah nilai *Average end-to-end delay* pada suatu jaringan maka semakin baik jaringan, karena hal tersebut dapat menghemat penggunaan energi yang dipakai untuk proses *routing* selama waktu jeda. *Average nd-to-end delay* dihitung menggunakan persamaan 3.2 sebagai berikut.

Rumus:

*Average delay* (s) = (3.2)

\*Di mana *delay* sama dengan waktu diterima dikurangi waktu dikirim.

1. *Packet Delivery Ratio* (PDR) : perbandingan antara paket data yang terkirim dengan jumlah paket data yang diterima. Semakin tinggi nilai PDR maka semakin baik, hal ini dikarenakan jumlah paket yang gagal diterima semakin berkurang. PDR dihitung menggunakan persamaan 3.3 sebagai berikut.

Rumus:

*Packet Delivery Ratio* (%) = x100% (3.3)

1. **Parameter skenario simulasi**

Pada bagian ini merupakan skenario uji coba yang akan diimplementasikan menggunakan NS-2 versi 2.35. *Link layer* yang digunakan adalah berdasarkan pada standar IEEE 802.11. Fungsi koordinasi terdistribusi IEEE 802.11 untuk nirkabel LAN digunakan sebagai lapisan MAC. Jumlah *node* yang ada pada area sebanyak 50 dan 100 *node*[4]. Simulasi dilakukan pada area persegi (*network area*) dengan ukuran 500 × 500 m2 dan 1000 ×1000 m2[4]. Tujuan simulasi dilakukan pada dua luas area yang berbeda adalah untuk mengetahui perbandingan kinerja kedua protokol dalam keadaan luas area yang kecil dan besar. Waktu simulasi yang dilakukan selama 600 *second*. Digunakan variasi kecepatan *node* untuk melihat perbandingan kinerja kedua protokol, yakni sebesar 1 m/s, 5 m/s, dan 10 m/s serta menggunakan variasi *pause time* selama 45 s, 90 s, dan 180 s untuk mengetahui perbandingan kinerja kedua protokol. Jenis mobilitas yang digunakan pada simulasi adalah *Random Waypoint*. Setiap *node* secara independen memilih *destination* yang hendak dituju secara acak di dalam batasan jaringan. Parameter skenario simulasi yang lebih lengkap dapat dilihat pada Tabel 3.2.

Tabel 3.2 Parameter skenario.

|  |  |
| --- | --- |
| **Parameter Skenario** | |
| MAC *layer* | IEEE 802.11 |
| Jumlah *node* | 50 *node* , 100 *node* |
| *Network* area | 500x500 m2, 1000x1000 m2 |
| Waktu simulasi | 600 s |
| Kecepatan *node* | 1 m/s, 5 m/s, 10 m/s |
| *Pause time* | 45 s, 90 s, 180 s |
| Pergerakan *node* | *Random Waypoint* |
| *Propagation* | *TwoRayGround* |
| Antena | Omni antenna |
| Pola *traffic* | CBR |
| *Channel* | *Wireless* |

# BAB IV

**HASIL DAN PEMBAHASAN**

Bab ini akan membahas implementasi perancangan metode penelitian yang mengacu pada desain dan perancangan simulasi yang telah dibahas pada bab sebelumnya , serta membahas uji coba dan evaluasi hasil simulasi dari skenario *Network Simulator* 2 yang telah dilakukan.

## Implementasi Metode

Pada sub bab ini akan dijelaskan implementasi metode yaitu dengan melakukan penambahan *script* algoritma EA-SHORT pada kerangka protokol AOMDV. Dimana akan dilakukan proses perhitungan energi pada suatu *node* yang dapat digunakan sebagai rute baru yang berbasis energi dengan memanfaatkan perubahan topologi yang diharapkan dapat membantu protokol AOMDV dalam menemukan rute pengiriman paket data yang dapat diandalkan. *Energy aware* diterapkan dengan mempertimbangkan energi yang dimiliki oleh tiap *node* untuk memilih apakah suatu *node* memiliki energi yang cukup untuk menerima dan meneruskan paket data ke *node* selanjutnya. Sehingga, dengan menerapkan metode tersebut diharapkan adanya peningkatan performa kinerja pada protokol AOMDV.

Langkah awal yang dilakukan untuk mengimplementasikan algoritma EA\_SHORT adalah dengan membuat beberapa variabel yang akan digunakan untuk menampung informasi level energi dan *hop count* ketika paket diterima oleh sebuah *node* yang dapat dilihat pada Gambar 4.1. Penambahan dilakukan pada *file* aomdv.h yang merupakan file *script* untuk *header* dari kerangka AOMDV. Berikut adalah *script* deklarasi variabel yang digunakan:

|  |
| --- |
| double hc\_i, hc\_j, lvl, txPower, rxPower, txEnergy, rxEnergy; |

Gambar 4. 1 Inisialisasi variabel untuk menampung informasi node.

Berdasarkan *script* di atas dapat diketahui bahwa pada *file* aomdv.h dilakukan deklarasi variabel dengan keterangan sebagai berikut :

|  |  |  |
| --- | --- | --- |
| * *hc\_i* | : | menyimpan nilai *hop count* *node* ke i; |
| * *hc\_j* | : | menyimpan nilai *hop count* *node* ke j ; |
| * *Lvl* | : | untuk menampung nilai sisa energi yang dimiliki oleh suatu *node;* |
| * *txPower* | : | mendefinisikan besar daya yang dibutuhkan untuk mengirim paket; |
| * *rxPower* | : | mendefinisikan besar daya yang dibutuhkan untuk menerima satu paket; |
| * *txEnergy* | : | mendefinisikan energi yang dibutuhkan *node* untuk mengirim paket; |
| * *rxEnergy* | : | mendefinisikan energi yang dibutuhkan *node* untuk menerima paket. |

Setelah dilakukan pendeklarasian variabel yang digunakan, maka selanjutnya dilakukan inisialisasi nilai dari variabel yang telah dideklarasikan sebelumnya pada *file* aomdv.cc. Pada fungsi AOMDV::AOMDV(nsaddr\_t id): Agent(PT\_AOMDV), btimer (this), htimer(this), ntimer(this), rtimer(this), ltimer(this), rqueue() dilakukan pemberian nilai 0 pada variabel tersebut. Berikut adalah inisialisasi nilai variabel yang digunakan dalam perhitungan energi :

|  |
| --- |
| AOMDV::AOMDV(nsaddr\_t id) : Agent(PT\_AOMDV),  btimer(this), htimer(this), ntimer(this),  rtimer(this), lrtimer(this), rqueue() {  .  .  hc\_i, hc\_j = 0.0;  lvl, rxPower, txPower, rxEnergy, txEnergy = 0.0;  .  .  } |

Gambar 4.2 Inisialisasi nilai variabel.

Langkah selanjutnya adalah membuat *script* pada *file* aomdv.cc pada fungsi AOMDV::recv(Packet \*p, Handler\*) ditambahkan *script* untuk mengecek paket yang datang. Berikut adalah potongan *script* EA- AOMDV yang digunakan untuk proses pengecekan paket yang datang:

|  |
| --- |
| AOMDV::recv(Packet \*p, Handler\*) {  .  .      if ( (u\_int32\_t)ih->daddr() != IP\_BROADCAST)          rt\_resolve(p);        else          forward((aomdv\_rt\_entry\*) 0, p, NO\_AOMDV\_DELAY);  .  .  AOMDV::recvRequest(Packet \*p) {      aomdv\_rt\_entry \*rt;  .  . |

Gambar 4.3 Potongan script EA- AOMDV pada fungsi recv dan recvRequest.

Selanjutnya menambahkan *script* pada fungsi AOMDV::recvReply(Packet \*p) yang berfungsi sebagai penyerima pesan *route Replay* (RREP), untuk menyimpan informasi level energi dan *hop count* paket i pada variabel hc\_i. Kemudian akan dicek apakah level energinya sudah memenuhi kondisi atau tidak, serta *script* untuk mem-*forward* paket. Potongan *script* EA- AOMDV dapat dilihat pada potongan *script* berikut.

|  |
| --- |
| void  AOMDV::recvReply(Packet \*p) {      aomdv\_rt\_entry \*rt0, \*rt;  …  rt = rtable.rt\_lookup(rp->rp\_dst);    *//Kondidi Saat* routing table *dalam keadaan kosong, maka akan ditambahkan rute baru.*    if(rt == 0) {          rt = rtable.rt\_add(rp->rp\_dst);          rt = rp\_first\_hop = rp\_hop\_count ;          rp\_hop\_count = lvl (p);          rp\_hop\_count = hc\_i (p);    *//Kondisi saat sequence number pada (p) < dari sequence number pada (e)atau yang sudah ada di routing table sebelumnya.*  if (rt->rt\_seqno < rp->rp\_dst\_seqno) {          rt->rt\_seqno = rp->rp\_dst\_seqno;          rt->rt\_advertised\_hops = INFINITY;          rt->path\_delete();          rt->rt\_flags = RTF\_UP;          /\* Insert forward path to RREQ destination. \*/  forward\_path = rt->path\_insert(rp->rp\_src, rp->rp\_hop\_count+1, CURRENT\_TIME + rp->rp\_lifetime, rp->rp\_first\_hop);          rt->rt\_last\_hop\_count = rt->path\_get\_max\_hopcount();     }  *//Kondisi saat sequence number pada (p) > dari sequence number pada (e)atau yang sudah ada di routing table sebelumnya.*  if (rt->rt\_seqno > rp->rp\_dst\_seqno) {          rt->rt\_seqno = rp->rp\_dst\_seqno;          rt->rt\_seqno = rp\_hop\_count ;          lvl (rp\_hop\_count) = lvl (p);          hc\_i (rp\_hop\_count) = hc\_i (p);          rt->rt\_advertised\_hops = INFINITY;          rt->path\_delete();          rt->rt\_flags = RTF\_UP;  forward\_path = rt->path\_insert(rp->rp\_src, rp->rp\_hop\_count+1, CURRENT\_TIME + rp->rp\_lifetime, rp->rp\_first\_hop);          rt->rt\_last\_hop\_count = rt->path\_get\_max\_hopcount();  rt = rtable.rt\_add(rp->rp\_dst);  hc\_i (rp\_dst\_seqno) = hc\_i (rp\_hop\_count) = ;  hc\_i (rp\_hop\_count+1) == hc\_i (rp\_hop\_count+2);  lvl (p) >=  lvl (max (rp\_dst, rp\_dst\_seqno));  lvl (p) - lvl (rp\_dst\_hop\_count) >= 2;  assert (rt->rt\_flags == RTF\_UP);  .  . |

Gambar 4.4 Potongan script EA- AOMDV pada fungsi recvReplay (part 1).

|  |
| --- |
| .  .  if ((forward\_path = rt->disjoint\_path\_lookup(rp->rp\_src, rp->rp\_first\_hop))) {         assert (forward\_path->hopcount == (rp->rp\_hop\_count+1));         forward\_path->expire = max(forward\_path->expire, CURRENT\_TIME + rp->rp\_lifetime); }  else if ( rt->new\_disjoint\_path(rp->rp\_src, rp->rp\_first\_hop)&&(rt->rt\_num\_paths\_ < aomdv\_max\_paths\_) && ((rp->rp\_hop\_count+1) - rt->path\_get\_min\_hopcount() <= aomdv\_prim\_alt\_path\_len\_diff\_)) {  */\* Masukkan jalur forward ke tujuan RREQ \*/*  forward\_path = rt->path\_insert(rp->rp\_src, rp->rp\_hop\_count+1, CURRENT\_TIME + rp->rp\_lifetime, rp->rp\_first\_hop);  // *routing table berubah*    rt->rt\_last\_hop\_count = rt->path\_get\_max\_hopcount();                // *routing table berubah*          }  */\* Path tidak ada atau tidak bisa ditambahkan - drop paket.\*/*  else {              Packet::free(p);              return;          }     }     /\* RREP yang diterima tidak mengandung informasi yang lebih baru dari *routing table* - jadi drop paket \*/    else {          Packet::free(p);          return;     }  .  . |

Gambar 4.5 Potongan script EA- AOMDV pada fungsi recvReplay (part 2).

## Menentukan Skenario Simulasi

Pembuatan skenario simulasi untuk melakukan uji coba dilakukan dengan cara mengubah parameter yang terdapat pada NS-2.35. Berikut langkah-langkah yang harus dilakukan dalam pembuatan dan pelaksanaan skenario simulasi, yaitu :

1. **Menentukan pola *traffic***

Pola *traffic* yang digunakan untuk skenario uji coba yaitu pola *traffic* yang dihasilkan secara acak (*traffic source generator*) oleh *file* “*cbrgen.tcl*”. *File* tersebut telah disediakan oleh NS-2.35. Terdapat beberapa parameter yang harus ditentukan untuk membuat skenario simulasi sesuai dengan tabel 3.2. Parameter yang digunakan pada *script* yaitu tipe koneksi yang digunakan, jumlah *node* dalam simulasi, jumlah *seed* yang digunakan, maksimum koneksi setiap *node* dan paket *rate*.

Berikut adalah cara menentukan pola *traffic* yang digunakan untuk menjalankan simulasi, dapat dilihat pada Gambar 4.6.

|  |
| --- |
| //Untuk masuk ke dalam direktori :  $ cd/home/ns-allinone-2.35/ns-2.35/indep-utils/cmu-scen-gen  //Untuk menentukan pola *traffic* yang diinginkan:  $ ns cbrgen.tcl –type [cbr] –nn [node] –seed [seed] –mc [connections]-rate [rate] > [nama file untuk menyimpan data hasil generate] |

Gambar 4.6 Mekanisme cara menentukan pola traffic.

Keterangan :

* *type* : menyatakan *type transport* yang digunakan yaitu CBR;
* *nn* : merupakan jumlah *node* dalam jaringan;
* *seed* : menentukan jumlah *seed* ;
* *mc* : merupakan maksimum koneksi dalam jaringan;
* *rate* : menentukan paket *rate* yang digunakan.

Hasil dari pola *traffic* dengan *cbrgen.tcl* dapat dilihat pada Gambar 4.7 sebagai berikut:

|  |
| --- |
| # nodes: 50, max conn: 20, send rate: 1, seed: 1  //node 4 terkoneksi dengan node 5 pada simulation time=56.333118917575632  # 4 connecting to 5 at time 56.333118917575632  set udp\_(1) [new Agent/UDP]  $ns\_ attach-agent $node\_(4) $udp\_(1)  set null\_(1) [new Agent/Null]  $ns\_ attach-agent $node\_(5) $null\_(1)  //membuat traffic CBR  set cbr\_(1) [new Application/Traffic/CBR]  $cbr\_(1) set packetSize\_ 512  $cbr\_(1) set interval\_ 1  $cbr\_(1) set random\_ 1  $cbr\_(1) set maxpkts\_ 10000  $cbr\_(1) attach-agent $udp\_(1)  $ns\_ connect $udp\_(1) $null\_(1)  //menjalankan *traffic* CBR pada t=56.333118917575632  $ns\_ at 56.333118917575632 "$cbr\_(1) start" |

Gambar 4.7 Potongan script hasil pola traffic dengan cbrgen.tcl.

1. **Menentukan pola pergerakan *node***

Pola pergerakan *node* (*node movement*) dalam simulasi jaringan *ad-hoc* dihasilkan secara acak menggunakan model *random waypoint*. Pada model ini *node-node* akan bergerak secara acak menuju posisi tujuan dengan kecepatan sesuai skenario yang telah ditentukan yaitu antara 1 sampai 10 m/s. Untuk mendapatkan skenario pergerakan tersebut maka digunakan modul yang telah disediakan oleh NS-2.35. Modul tersebut terdiri dari *file setdest.h* dan *setdest.cc*. Adapun beberapa parameter yang digunakan dalam *script* yaitu :

|  |  |  |
| --- | --- | --- |
| -n | : | menyatakan jumlah *node* dalam jaringan; |
| -p | : | menunjukkan *pause time* sebuah *node* saat dalam keadaan tidak bergerak; |
| -M | : | maksimum *speed* suatu *node* dalam keadaan bergerak; |
| -t | : | *Simulation time* yaitu durasi waktu yang dibutuhkan untuk menjalankan simulasi; |
| -x | : | ukuran topologi jaringan (dalam sumbu x) untuk menentukan luas area yang akan digunakan dalam simulasi; |
| -y | : | ukuran topologi jaringan (dalam sumbu y) untuk menentukan luas area yang akan digunakan dalam simulasi. |

Cara menjalankan simulasi untuk perintah pembuatan model pergerakan *node* dapat dilihat pada Gambar 4.8.

|  |
| --- |
| **//Masuk dalam direktori setdest:**  $ cd/home/ns-allinone-2.35/ns-2.35/indep-utils/cmu-scen-gen/setdest  //**Perintah untuk menentukan pola pergerakan node:**  $ ./setdest –n [jumlah node] –p [pause time] –M [max speed] –t [waktu simulasi] –x [max X] –y [max Y] > [nama file untuk menyimpan data hasil generate] |

Gambar 4.8 Mekanisme pola pergerakan node.

Hasil dari pola pergerakan *node* dapat dilihat pada Gambar 4.9 dan Gambar 4.10.

|  |
| --- |
| # nodes: 50, pause: 45.00, max speed: 1.00, max x: 500.00, max y: 500.00  #  $node\_(0) set X\_ 412.388144125402  $node\_(0) set Y\_ 463.368080994245  $node\_(0) set Z\_ 0.000000000000  $node\_(1) set X\_ 452.484598061255  $node\_(1) set Y\_ 68.990314227991  $node\_(1) set Z\_ 0.000000000000  $node\_(2) set X\_ 44.187526845142  $node\_(2) set Y\_ 11.167636565601  $node\_(2) set Z\_ 0.000000000000 |

Gambar 4.9 Potongan script hasil pola pergerakan node pada posisi awal.

Set nilai X dan Y pada *script* Gambar 4.9 berfungsi untuk menentukan lokasi atau posisi *node* dalam bentuk area dua dimensi.

|  |
| --- |
| $ns\_ at 594.566838464624 "$node\_(22) setdest 478.241942234879 192.020413568300 0.000000000000"  $ns\_ at 594.760159721370 "$god\_ set-dist 9 38 2"  $ns\_ at 595.698608789465 "$god\_ set-dist 2 48 2"  $ns\_ at 596.539448647921 "$god\_ set-dist 5 40 1"  $ns\_ at 597.435132103244 "$god\_ set-dist 38 46 2"  $ns\_ at 599.181845863575 "$god\_ set-dist 4 9 2"  $ns\_ at 599.181845863575 "$god\_ set-dist 4 23 3"  #  # Destination Unreachables: 0  #  # Route Changes: 1083  #  # Link Changes: 766 |

Gambar 4.10 Potongan script hasil pola pergerakan node pada posisi tujuan.

1. **Membuat dan menjalankan *file* TCL**

Langkah selanjutnya setelah membuat pola *traffic* dan pola pergerakan *node* yaitu membuat *file* TCL untuk simulasi skenario uji coba. *File* TCL berisi parameter yang berisi pola *traffic* dan pergerakan *node* akan dipanggil melalui *file* TCL. Berikut *script* untuk menjalankan *file* TCL yang telah dibuat, dapat dilihat pada Gambar 4.11.

|  |
| --- |
| set val(chan) Channel/WirelessChannel  set val(prop) Propagation/TwoRayGround  set val(netif) Phy/WirelessPhy  set val(mac) Mac/802\_11  set val(ifq) Queue/DropTail/PriQueue  set val(ll) LL  set val(ant) Antenna/OmniAntenna  set val(x) 1000  set val(y) 1000  set val(ifqlen) 20  set val(seed) 1.0  set val(adhocRouting) AOMDV  set val(nn) 100  set val(cp) "cp100"  set val(sc) "sp100n-10M-180pt-1000xy"  set val(stop) 600 ;  # ============================================================  # Main Program  # ============================================================  set ns\_ [new Simulator]  set topo [new Topography]  # membuat trace file untuk ns dan nam  set tracefd [open eaomdv-18.tr w]  set namtrace [open eaomdv-18.nam w]  $ns\_ trace-all $tracefd  $ns\_ namtrace-all-wireless $namtrace $val(x) $val(y)  $topo load\_flatgrid $val(x) $val(y)  set god\_ [create-god $val(nn)]  # mendefinisikan pembuatan node  set chan [new $val(chan)]  $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) \  -channel $chan \  -topoInstance $topo \  -agentTrace ON \  -routerTrace ON \  -macTrace OFF  #menentukan jumlah node sesuai nilai pada [$val(nn)] dan "attach" ke channel.  for {set i 0} {$i < $val(nn) } {incr i} {  set node\_($i) [$ns\_ node]  $node\_($i) random-motion 0 ;# disable random motion  }  #memanggil pola pergerakan node  puts "Loading connection pattern..."  source $val(sc)  #memanggil model *traffic*(cbrgen)  puts "Loading scenario file..."  source $val(cp)  # menentukan posisi awal node di nam  for {set i 0} {$i < $val(nn)} {incr i} {  $ns\_ initial\_node\_pos $node\_($i) 20  }  # memberitahu node kondisi saat simulasi berakhir  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 $val(x) y $val(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 4.11 Script file TCL.

Berikut perintah untuk menjalankan simulasi dengan menggunakan *file* TCL, dapat dilihat pada Gambar 12.

|  |
| --- |
| $ ns file\_name.tcl |

Gambar 4.12 Perintah untuk menjalankan simulasi.

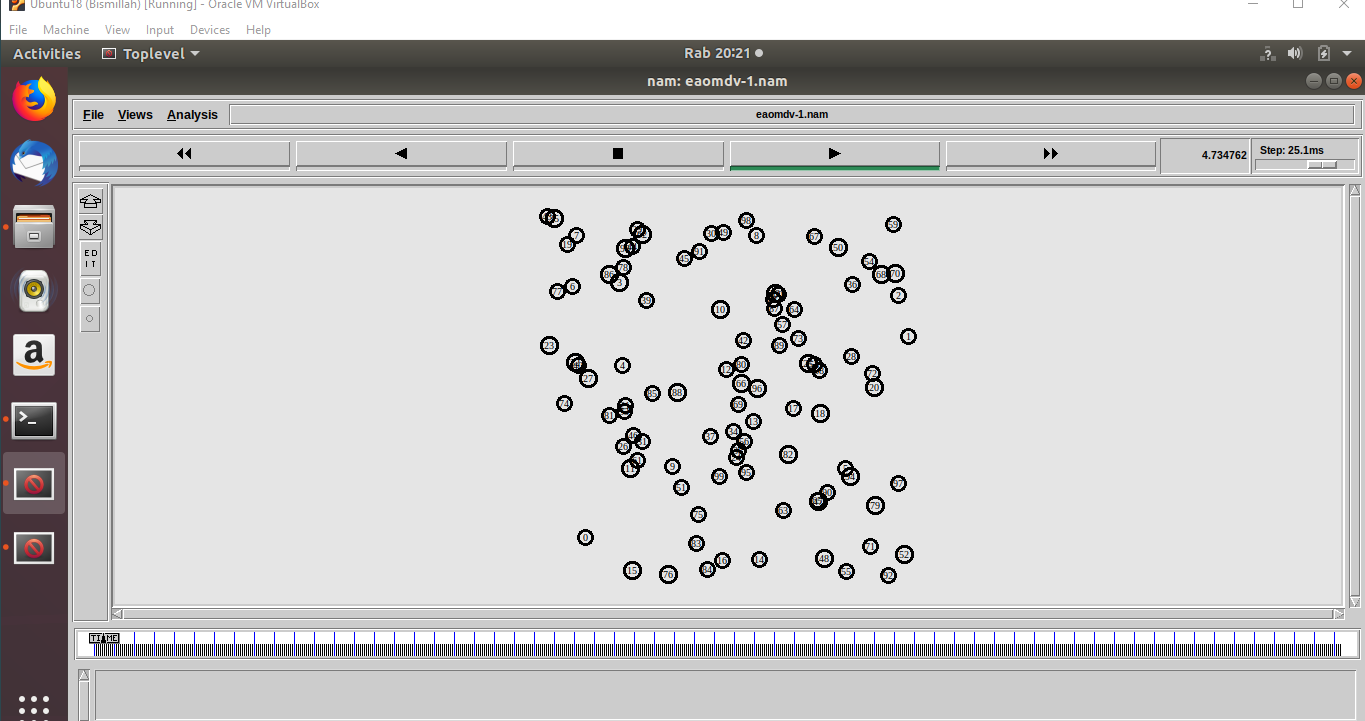
1. **Analisis Hasil Simulasi Menggunakan *File* AWK**

Setelah *file* TCL dijalankan berdasarkan skenario simulasi, akan terbentuk dua buah file yaitu *file*\*.nam (*network animator*) dan *file\*.tr* (*trace file*). *File\*.nam* (*network animator*) digunakan untuk memperlihatkan animasi simulasi jaringan terlihat pada Gambar 4.5. *Trace file* yang diperoleh tersebut akan di- *filtering* yang bertujuan untuk mengetahui data parameter nilai dari *throughput*, *packet delivery ratio*, dan *average end-to-end delay*.

|  |
| --- |
| M 0.0 nn 50 x 500 y 500 rp AOMDV  M 0.0 sc 50n-45pt-1M-600t-500xy cp 50n seed 1.0  M 0.0 prop Propagation/TwoRayGround ant Antenna/OmniAntenna  s 2.556838879 \_1\_ AGT --- 0 cbr 512 [0 0 0 0] ------- [1:0 2:0 32 0] [0] 0 1  r 2.556838879 \_1\_ RTR --- 0 cbr 512 [0 0 0 0] ------- [1:0 2:0 32 0] [0] 0 1  s 2.556838879 \_1\_ RTR --- 0 AOMDV 52 [0 0 0 0] ------- [1:255 -1:255  30 0] [0x2 0 1 [2 0] [1 4]] (REQUEST)  r 2.558319005 \_43 RTR --- 0 AOMDV 52 [0 ffffffff 1 800] -------  [1:255 -1:255 30 0] [0x20 1 [20] [1 4]] (REQUEST)  r 2.558319005 \_40 RTR --- 0 AOMDV 52 [0 ffffffff 1 800] -------  [1:255 -1:255 30 0] [0x20 1 [20] [1 4]] (REQUEST)  r 2.558319017 \_13 RTR --- 0 AOMDV 52 [0 ffffffff 1 800] -------  [1:255 -1:255 30 0] [0x20 1 [20] [1 4]] (REQUEST) |

Gambar 4.13 Potongan script trace file.

Hasil *filtering* AWK *script* yang berupa *file* *nam* dapat dilihat pada Gambar 4.14.



Gambar 4.14 Tampilan simulasi pergerakan node pada network animator (nam).

Berikut adalah perintah pemanggilan *file awk* pada NS2 berdasarkan *file* .tr untuk melihat nilai dari parameter uji, dapat dilihat pada Gambar 4.15.

|  |
| --- |
| $ awk –f nama file.awk nama file \*.tr |

Gambar 4.15 Perintah proses filtering file.tr dengan awk script.

Berikut AWK *Script* yang digunakan untuk melakukan uji coba performansi *routing* yaitu *throughput*, *Packet Delivery Ratio* (PDR) dan *End-to-End Delay.*

|  |
| --- |
| BEGIN {  recvdSize = 0  startTime = 0  stopTime = 0  }  {  event = $1  time = $2  node\_id = $3  pkt\_size = $8  level = $4  # menyimpan waktu mulai  if (level == "AGT" && event == "s" && pkt\_size >= 512) {  if (time < startTime) {  startTime = time  }  }  # memperbaharui total paket diterima dan menyimpan paket waktu kedatangan  if (level == "AGT" && event == "r" && pkt\_size >= 512) {  if (time > stopTime) {  stopTime = time  }  recvdSize += pkt\_size  }  }  END {  printf("Average Throughput[kbps] = %.2f\n",(recvdSize/(stopTime-startTime))\*(8/1000),startTime,stopTime)  } |

Gambar 4.16 Potongan AWK script untuk throughput.

Gambar 4.16 merupakan AWK *Script* yang digunakan untuk melakukan uji coba performansi protokol *routing* AOMDV yaitu nilai *throughput*.

|  |
| --- |
| BEGIN {  seqno = -1;  no\_of\_packets = 0;  droppedPackets = 0;  receivedPackets = 0;  total\_delay = 0;  }  {  if($4 == "AGT"){  if($1 == "s"){  seqno = $6;  no\_of\_packets++;  start\_time[$6] = $2;  }    else if($1 == "r"){  receivedPackets++;  end\_time[$6] = $2; #end-to-end delay  }  }  }    END {  for(i=0; i<=seqno; i++) {  delay[i] = end\_time[i] - start\_time[i];  if(delay[i] > 0)  total\_delay = total\_delay + delay[i];  }  print "Packet Delivery Ratio : " receivedPackets/no\_of\_packets\*100"%"; #packet delivery ratio  print "Average Delay : " total\_delay/receivedPackets;  } |

Gambar 4.17 Potongan AWK script untuk PDR dan end-to-end delay.

Gambar 4.17 merupakan AWK *Script* yang digunakan untuk melakukan uji coba performansi protokol *routing* AOMDV yaitu nilai *Packet Delivery Ratio* (PDR) dan *End-to-End Delay.*

## Hasil Uji Coba

Hasil penelitian yang dilakukan meliputi analisis performansi protokol *routing* terhadap perubahan jumlah kepadatan *node*, perubahan luas area, kecepatan *node* dan *pause time*. Hasil percobaan yang diperoleh merupakan hasil perulangan percobaan yang dilakukan sebanyak 3 kali pada setiap skenario. Hal tersebut dilakukan untuk mendapatkan akurasi data yang baik dalam membandingkan performansi protokol *routing* AOMDV dengan protokol *routing* AOMDV yang telah dimodifikasi. Parameter yang digunakan untuk menganalisis kinerja protokol *routing* adalah *throughput*, *packet delivery ratio* (PDR) dan *average end-to-end delay*. Berikut adalah data yang diperoleh selama simulasi terhadap protokol *routing* AOMDV dan protokol *routing* AOMDV dengan algoritma EA-SHORT.

Pada sub bab 4.3.1, sub bab 4.3.2 dan sub bab 4.3.3 akan dipaparkan hasil percobaan yang dilakukan dengan 50 *node*, 100 *node*, luas area 500 x 500 m2 dan 1000 x 1000 m2 terhadap *pause time* dan kecepatan, dalam bentuk tabel.

1. **Pergerakan *node* terhadap *throughput***

Skenario uji coba pertama adalah mencari rata-rata jumlah paket data yang berhasil diterima di sisi penerima setiap detiknya (*throughput*). *Throughput* adalah rata-rata data yang dikirim dalam suatu jaringan. *Throughput* merujuk pada besar data yang dibawa oleh semua *traffic* jaringan dan selalu dikaitkan dengan *bandwidth*. Tabel 4.1, 4.2 dan 4.3 merupakan nilai kinerja algoritma protokol *routing* berdasarkan rata-rata *throughput* didapat dari hasil uji coba skenario MANET dengan variasi kecepatan maksimal *node* (*maximal speed*) yaitu 1m/s, 5m/s, dan 10m/s, serta dengan variasi *pause time* yaitu, 45s, 90s, dan 180s.

Tabel 4.1 Hasil uji coba 1 untuk *throughput*.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***Network Area* (m2)** | **Kecepatan *Node* (m/s)** | ***Pause Time* (s)** | **AOMDV (Kbps)** | | **EA-AOMDV (Kbps)** | |
| **50 *Node*** | **100 *Node*** | **50 *Node*** | **100 *Node*** |
| **500x500** | 1 | 45 | 64,76 | 65,11 | 74,75 | 74,43 |
| 90 | 65,89 | 65,59 | 74,47 | 74,36 |
| 180 | 65,32 | 65,78 | 74,93 | 74,92 |
| 5 | 45 | 64,66 | 62,40 | 72,74 | 73,54 |
| 90 | 63,88 | 63,12 | 73,75 | 73,19 |
| 180 | 62,11 | 63,86 | 74,20 | 73,67 |
| 10 | 45 | 62,00 | 61,15 | 71,12 | 72,75 |
| 90 | 63,13 | 62,88 | 73,19 | 73,19 |
| 180 | 64,82 | 63,77 | 73,39 | 74,17 |
| **1000x1000** | 1 | 45 | 64,10 | 62,82 | 70,48 | 73,18 |
| 90 | 63,43 | 63,97 | 72,77 | 72,99 |
| 180 | 64,57 | 63,85 | 73,27 | 73,33 |
| 5 | 45 | 58,72 | 58,22 | 66,42 | 66,51 |
| 90 | 58,91 | 57,73 | 66,94 | 68,63 |
| 180 | 61,11 | 62,86 | 70,92 | 70,82 |
| 10 | 45 | 53,05 | 50,74 | 63,59 | 64,19 |
| 90 | 58,40 | 56,28 | 60,25 | 65,21 |
| 180 | 58,32 | 59,75 | 66,74 | 68,66 |

Tabel 4. 2 Hasil uji coba 2 untuk throughput.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***Network Area* (m2)** | **Kecepatan *Node* (m/s)** | ***Pause Time* (s)** | **AOMDV (Kbps)** | | **EA-AOMDV (Kbps)** | |
| **50 *Node*** | **100 *Node*** | **50 *Node*** | **100 *Node*** |
| **500x500** | 1 | 45 | 64,32 | 64,23 | 73,89 | 73,91 |
| 90 | 64,70 | 63,94 | 73,29 | 73,28 |
| 180 | 63,43 | 62,45 | 74,68 | 74,93 |
| 5 | 45 | 62,90 | 61,91 | 71,18 | 72,28 |
| 90 | 62,05 | 61,04 | 72,20 | 74,06 |
| 180 | 61,25 | 60,51 | 72,89 | 72,14 |
| 10 | 45 | 60,63 | 59,99 | 68,75 | 74,08 |
| 90 | 61,34 | 60,00 | 71,53 | 71,55 |
| 180 | 64,26 | 64,13 | 72,29 | 75,02 |
| **1000x1000** | 1 | 45 | 62,75 | 61,72 | 69,47 | 71,97 |
| 90 | 62,01 | 60,71 | 71,25 | 73,96 |
| 180 | 63,26 | 62,38 | 71,39 | 72,07 |
| 5 | 45 | 56,53 | 54,77 | 62,99 | 68,98 |
| 90 | 55,93 | 53,68 | 63,66 | 65,40 |
| 180 | 58,63 | 56,58 | 67,67 | 73,51 |
| 10 | 45 | 49,93 | 46,93 | 62,28 | 62,54 |
| 90 | 54,99 | 52,01 | 58,37 | 66,55 |
| 180 | 54,47 | 51,35 | 65,37 | 67,20 |

Tabel 4. 3 Hasil uji coba 3 untuk throughput.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***Network Area* (m2)** | **Kecepatan *Node* (m/s)** | ***Pause Time* (s)** | **AOMDV (Kbps)** | | **EA-AOMDV (Kbps)** | |
| **50 *Node*** | **100 *Node*** | **50 *Node*** | **100 *Node*** |
| **500x500** | 1 | 45 | 64,41 | 63,64 | 73,98 | 73,82 |
| 90 | 65,46 | 64,49 | 74,05 | 74,04 |
| 180 | 64,41 | 65,17 | 74,81 | 78,80 |
| 5 | 45 | 63,89 | 61,63 | 72,21 | 73,31 |
| 90 | 63,06 | 62,70 | 73,50 | 72,76 |
| 180 | 61,99 | 63,74 | 73,77 | 73,02 |
| 10 | 45 | 61,27 | 60,62 | 70,51 | 73,32 |
| 90 | 62,68 | 62,23 | 72,87 | 72,89 |
| 180 | 64,39 | 63,34 | 73,27 | 74,04 |
| **1000x1000** | 1 | 45 | 63,78 | 62,50 | 70,25 | 72,75 |
| 90 | 63,31 | 63,85 | 72,34 | 72,87 |
| 180 | 64,14 | 63,62 | 72,62 | 73,30 |
| 5 | 45 | 58,29 | 57,79 | 65,99 | 65,98 |
| 90 | 58,18 | 57,60 | 66,64 | 68,38 |
| 180 | 60,68 | 62,43 | 70,79 | 70,39 |
| 10 | 45 | 52,93 | 50,62 | 63,16 | 63,42 |
| 90 | 57,97 | 55,85 | 60,13 | 64,79 |
| 180 | 57,59 | 59,45 | 66,71 | 68,54 |

Pada Tabel 4.1 yaitu hasil uji coba 1 pada *throughput*, nilai tertinggi yang didapatkan dari hasil uji coba perbandingan AOMDV dengan EA-AOMDV pada 50 *node* yaitu 65,89 Kbps dan 78,93 Kbps. Sedangkan nilai tertinggi yang didapatkan pada 100 *node* yaitu AOMDV 65,78 Kbps dan EA- AOMDV 74,93 Kbps. Pada Tabel 4.2 yaitu hasil uji coba 2 pada *throughput*, nilai tertinggi yang didapatkan dari hasil uji coba perbandingan AOMDV dengan EA-AOMDV pada 50 *node* yaitu 64,70 Kbps dan 74,68 Kbps. Sedangkan nilai tertinggi yang didapatkan pada 100 *node* yaitu EA-AOMDV 75,02 Kbps dan AOMDV 64,23 Kbps. Pada Tabel 4.3 yaitu hasil uji coba 3 pada *throughput*, nilai tertinggi yang didapatkan dari hasil uji coba perbandingan pada 50 *node* yaitu EA-AOMDV 74,81 dan AOMDV 65,46 Kbps. Sedangkan nilai tertinggi yang didapatkan pada 100 *node* yaitu EA- AOMDV 78,81 Kbps dan AOMDV 65,17 Kbps.

1. **Pergerakan *node* terhadap *Packet Delivery Ratio* (PDR)**

Skenario uji coba kedua adalah mencari nilai perbandingan antara paket data yang terkirim (paket data yang berhasil diterima oleh *node* tujuan) dengan jumlah paket data yang dikirimkan oleh *node* sumber. *Packet Delivery Ratio* (PDR) menunjukkan keberhasilan protokol dalam mengirim data, tinggi nilai PDR salah satunya disebabkan oleh berhasilnya sebuah protokol dalam melakukan pencarian dan pemeliharaan rutenya. Tabel 4.4, 4.5 dan 4.6 merupakan nilai kinerja algoritma protokol *routing* berdasarkan nilai PDRyang diperoleh dari hasil uji coba skenario MANET dengan variasi kecepatan maksimal *node* (*maximal speed*) yaitu 1m/s, 5m/s, dan 10m/s, serta dengan variasi *pause time* yaitu, 45s, 90s, dan 180s.

Tabel 4.4 Hasil uji coba 1 untuk packet delivery ratio.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***Network Area* (m2)** | **Kecepatan *Node* (m/s)** | ***Pause Time*(s)** | **AOMDV (%)** | | **EA-AOMDV (%)** | |
| **50 *Node*** | **100 *Node*** | **50 *Node*** | **100 *Node*** |
| **500x500** | 1 | 45 | 99,045 | 99,321 | 99,328 | 99,860 |
| 90 | 99,642 | 99,244 | 99,655 | 99,915 |
| 180 | 99,503 | 99,137 | 99,592 | 99,860 |
| 5 | 45 | 98,008 | 96,873 | 98,609 | 97,397 |
| 90 | 96,671 | 97,826 | 97,055 | 98,010 |
| 180 | 95,672 | 98,042 | 96,900 | 98,119 |
| 10 | 45 | 94,039 | 94,336 | 95,452 | 96,500 |
| 90 | 95,555 | 97,101 | 96,034 | 97,358 |
| 180 | 95,640 | 97,365 | 96,802 | 98,865 |
| **1000x1000** | 1 | 45 | 98,026 | 93,436 | 98,576 | 96,592 |
| 90 | 97,090 | 96,433 | 97,491 | 97,021 |
| 180 | 98,103 | 96,473 | 98,382 | 97,307 |
| 5 | 45 | 89,820 | 88,074 | 90,870 | 88,553 |
| 90 | 90,183 | 88,384 | 91,845 | 91,110 |
| 180 | 93,064 | 90,400 | 95,310 | 93,736 |
| 10 | 45 | 80,030 | 79,726 | 87,136 | 85,426 |
| 90 | 88,894 | 84,558 | 89,314 | 86,019 |
| 180 | 88,273 | 88,706 | 91,743 | 90,925 |

Tabel 4. 5 Hasil uji coba 2 untuk packet delivery ratio.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***Network Area* (m2)** | **Kecepatan *Node* (m/s)** | ***Pause Time*(s)** | **AOMDV (%)** | | **EA-AOMDV (%)** | |
| **50 *Node*** | **100 *Node*** | **50 *Node*** | **100 *Node*** |
| **500x500** | 1 | 45 | 97,907 | 95,362 | 99,193 | 96,997 |
| 90 | 98,191 | 97,455 | 99,542 | 97,457 |
| 180 | 98,857 | 98,238 | 99,502 | 98,883 |
| 5 | 45 | 96,343 | 93,973 | 98,652 | 94,841 |
| 90 | 96,669 | 95,282 | 97,671 | 97,735 |
| 180 | 97,806 | 97,055 | 97,808 | 98,057 |
| 10 | 45 | 92,946 | 92,695 | 93,955 | 96,825 |
| 90 | 93,909 | 93,390 | 94,039 | 95,381 |
| 180 | 93,638 | 98,251 | 94,732 | 99,345 |
| **1000x1000** | 1 | 45 | 92,025 | 94,574 | 94,026 | 96,573 |
| 90 | 93,214 | 97,447 | 95,089 | 98,435 |
| 180 | 94,090 | 95,686 | 95,113 | 98,663 |
| 5 | 45 | 87,821 | 82,729 | 88,820 | 84,727 |
| 90 | 87,384 | 88,885 | 90,183 | 91,846 |
| 180 | 90,064 | 95,310 | 92,400 | 95,977 |
| 10 | 45 | 79,726 | 73,137 | 80,030 | 80,138 |
| 90 | 86,654 | 82,740 | 88,894 | 85,750 |
| 180 | 85,235 | 86,314 | 86,706 | 88,425 |

Tabel 4.6 Hasil uji coba 3 untuk packet delivery ratio.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***Network Area* (m2)** | **Kecepatan *Node* (m/s)** | ***Pause Time*(s)** | **AOMDV (%)** | | **EA-AOMDV (%)** | |
| **50 *Node*** | **100 *Node*** | **50 *Node*** | **100 *Node*** |
| **500x500** | 1 | 45 | 98,191 | 97,455 | 98,193 | 97,457 |
| 90 | 97,907 | 96,997 | 98,542 | 97,362 |
| 180 | 97,857 | 99,238 | 98,502 | 99,883 |
| 5 | 45 | 97,806 | 94,842 | 97,808 | 98,841 |
| 90 | 95,669 | 97,055 | 96,671 | 98,057 |
| 180 | 94,343 | 92,973 | 95,652 | 94,282 |
| 10 | 45 | 93,909 | 92,695 | 94,039 | 92,825 |
| 90 | 92,946 | 92,390 | 93,955 | 93,381 |
| 180 | 93,638 | 98,251 | 94,732 | 99,345 |
| **1000x1000** | 1 | 45 | 93,025 | 94,574 | 94,026 | 98,573 |
| 90 | 93,214 | 97,447 | 95,089 | 98,435 |
| 180 | 90,090 | 95,686 | 95,113 | 97,663 |
| 5 | 45 | 87,821 | 84,729 | 88,820 | 88,727 |
| 90 | 87,384 | 88,885 | 90,183 | 92,884 |
| 180 | 90,064 | 95,310 | 92,400 | 95,977 |
| 10 | 45 | 79,726 | 76,137 | 80,030 | 78,138 |
| 90 | 86,654 | 86,314 | 88,894 | 88,425 |
| 180 | 85,235 | 85,750 | 86,706 | 89,740 |

Pada Tabel 4.4 yaitu hasil uji coba 1 pada PDR, nilai tertinggi yang didapatkan dari hasil uji coba perbandingan AOMDV dengan EA-AOMDV pada 50 *node* yaitu 99,642% dan 99,655%. Sedangkan nilai tertinggi yang didapatkan pada 100 *node* yaitu EA- AOMDV 99,915% dan AOMDV 99,321%. Pada Tabel 4.5 yaitu hasil uji coba 2 pada PDR, nilai tertinggi yang didapatkan dari hasil uji coba perbandingan EA-AOMDV dan AOMDV pada 50 *node* yaitu 99,542% dan 98,857. Sedangkan nilai tertinggi yang didapatkan pada 100 *node* yaitu EA-AOMDV 99,3455 dan AOMDV 99,238. Pada Tabel 4.6 yaitu hasil uji coba 3 pada PDR, nilai tertinggi yang didapatkan dari hasil uji coba perbandingan pada 50 *node* yaitu EA-AOMDV 98,542% dan AOMDV 98,191%. Sedangkan nilai tertinggi yang didapatkan pada 100 *node* yaitu EA- AOMDV 99,883% dan AOMDV 99,238%.

1. **Pergerakan *node* terhadap *average end-to-end delay***

Skenario uji coba ketiga adalah mencari nilai rata-rata selang waktu mulai dari paket dikirimkan oleh *node* sumber sampai paket data tersebut berhasil diterima oleh *node* tujuan. Singkatnya *end-to-end delay* adalah waktu yang dibutuhkan dalam pengiriman data dari *node* sumber ke *node* tujuan. Tabel 4.7, 4.8 dan 4.9 merupakan nilai kinerja algoritma protokol *routing* berdasarkan nilai PDRyang diperoleh dari hasil uji coba skenario MANET dengan variasi kecepatan maksimal *node* (*maximal speed*) yaitu 1m/s, 5m/s, dan 10m/s, serta dengan variasi *pause time* yaitu, 45s, 90s, dan 180s.

Tabel 4.7 Hasil uji coba 1 untuk average end-to-end delay.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***Network Area* (m2)** | **Kecepatan *Node* (m/s)** | ***Pause Time* (s)** | **AOMDV (s)** | | **EA-AOMDV (s)** | |
| **50 *Node*** | **100 *Node*** | **50 *Node*** | **100 *Node*** |
| **500x500** | 1 | 45 | 0,01214 | 0,01069 | 0,00988 | 0,01038 |
| 90 | 0,00984 | 0,01376 | 0,00944 | 0,00968 |
| 180 | 0,01371 | 0,01156 | 0,01261 | 0,01117 |
| 5 | 45 | 0,01252 | 0,02102 | 0,01115 | 0,01493 |
| 90 | 0,01158 | 0,01520 | 0,01010 | 0,01201 |
| 180 | 0,01159 | 0,01312 | 0,01005 | 0,01090 |
| 10 | 45 | 0,01190 | 0,02251 | 0,01110 | 0,01792 |
| 90 | 0,01192 | 0,01374 | 0,01107 | 0,01005 |
| 180 | 0,01084 | 0,01323 | 0,01047 | 0,00972 |
| **1000x1000** | 1 | 45 | 0,02598 | 0,03129 | 0,02300 | 0,02419 |
| 90 | 0,01917 | 0,03058 | 0,01671 | 0,02985 |
| 180 | 0,02063 | 0,03002 | 0,01901 | 0,02690 |
| 5 | 45 | 0,02769 | 0,03516 | 0,02071 | 0,02677 |
| 90 | 0,02723 | 0,03335 | 0,02475 | 0,02390 |
| 180 | 0,02034 | 0,02959 | 0,01666 | 0,02434 |
| 10 | 45 | 0,03782 | 0,04092 | 0,03085 | 0,02743 |
| 90 | 0,03192 | 0,02922 | 0,03045 | 0,02440 |
| 180 | 0,02582 | 0,02918 | 0,02110 | 0,02712 |

Tabel 4.8 Hasil uji coba 2 untuk average end-to-end delay.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***Network Area* (m2)** | **Kecepatan *Node* (m/s)** | ***Pause Time* (s)** | **AOMDV (s)** | | **EA-AOMDV (s)** | |
| **50 *Node*** | **100 *Node*** | **50 *Node*** | **100 *Node*** |
| **500x500** | 1 | 45 | 0,01089 | 0,01768 | 0,00979 | 0,00913 |
| 90 | 0,00842 | 0,01497 | 0,00932 | 0,01053 |
| 180 | 0,01348 | 0,01156 | 0,01338 | 0,01108 |
| 5 | 45 | 0,01493 | 0,01602 | 0,01106 | 0,01079 |
| 90 | 0,01146 | 0,01132 | 0,01010 | 0,01076 |
| 180 | 0,01136 | 0,01322 | 0,00880 | 0,00948 |
| 10 | 45 | 0,01581 | 0,02055 | 0,00968 | 0,01756 |
| 90 | 0,01192 | 0,01374 | 0,01084 | 0,01096 |
| 180 | 0,00959 | 0,01723 | 0,01038 | 0,01160 |
| **1000x1000** | 1 | 45 | 0,02305 | 0,02829 | 0,01929 | 0,02396 |
| 90 | 0,01894 | 0,02458 | 0,01448 | 0,02143 |
| 180 | 0,01705 | 0,03402 | 0,01599 | 0,02667 |
| 5 | 45 | 0,02757 | 0,03316 | 0,02195 | 0,02468 |
| 90 | 0,02210 | 0,03135 | 0,02122 | 0,02378 |
| 180 | 0,01825 | 0,02959 | 0,01661 | 0,02425 |
| 10 | 45 | 0,03761 | 0,04092 | 0,02745 | 0,02935 |
| 90 | 0,02629 | 0,02922 | 0,02182 | 0,02435 |
| 180 | 0,02187 | 0,02918 | 0,01920 | 0,02097 |

Tabel 4. 9 Hasil uji coba 3 untuk average end-to-end delay.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***Network Area* (m2)** | **Kecepatan *Node* (m/s)** | ***Pause Time* (s)** | **AOMDV (s)** | | **EA-AOMDV (s)** | |
| **50 *Node*** | **100 *Node*** | **50 *Node*** | **100 *Node*** |
| **500x500** | 1 | 45 | 0,01654 | 0,01092 | 0,00946 | 0,01095 |
| 90 | 0,01241 | 0,01869 | 0,01033 | 0,01454 |
| 180 | 0,01446 | 0,01168 | 0,01050 | 0,01120 |
| 5 | 45 | 0,01825 | 0,02173 | 0,01096 | 0,01222 |
| 90 | 0,01403 | 0,01175 | 0,01067 | 0,01142 |
| 180 | 0,01235 | 0,01423 | 0,00917 | 0,01276 |
| 10 | 45 | 0,01959 | 0,02978 | 0,01091 | 0,02179 |
| 90 | 0,01617 | 0,01386 | 0,01072 | 0,01008 |
| 180 | 0,01092 | 0,01090 | 0,01095 | 0,00937 |
| **1000x1000** | 1 | 45 | 0,02355 | 0,02533 | 0,01939 | 0,01707 |
| 90 | 0,02079 | 0,02581 | 0,01907 | 0,01957 |
| 180 | 0,02793 | 0,02179 | 0,02089 | 0,01997 |
| 5 | 45 | 0,02634 | 0,03503 | 0,02081 | 0,02811 |
| 90 | 0,02209 | 0,03047 | 0,02207 | 0,02721 |
| 180 | 0,02214 | 0,02002 | 0,02075 | 0,01235 |
| 10 | 45 | 0,03286 | 0,04193 | 0,02634 | 0,02791 |
| 90 | 0,03021 | 0,02934 | 0,02170 | 0,02447 |
| 180 | 0,02677 | 0,03019 | 0,01939 | 0,02198 |

Pada Tabel 4.7 yaitu hasil uji coba 1 pada *Average end-to-end delay*, nilai terendah yang didapatkan dari hasil uji coba perbandingan EA-AOMDV dengan AOMDV pada 50 *node* yaitu 0.00944 s dan 0.00948 s. Sedangkan nilai terendah yang didapatkan pada 100 *node* yaitu EA- AOMDV 0.00968 s dan AOMDV 0.01069 s. Pada Tabel 4.8 yaitu hasil uji coba 2 pada *Average end-to-end delay*, nilai terendah yang didapatkan dari hasil uji coba perbandingan EA-AOMDV dan AOMDV pada 50 *node* yaitu 0.00880 s dan 0,00842 s. Sedangkan nilai terendah yang didapatkan pada 100 *node* yaitu EA-AOMDV 0.00913 s dan AOMDV 0.01132 s. Pada Tabel 4.9 yaitu hasil uji coba 3 pada *Average end-to-end delay*, nilai terendah yang didapatkan dari hasil uji coba perbandingan pada 50 *node* yaitu EA-AOMDV 0.00917 s dan AOMDV 0.01092 s. Sedangkan nilai terendah yang didapatkan pada 100 *node* yaitu EA- AOMDV 0.00937 s dan AOMDV 0.01090 s.

## Analisis Hasil Uji Coba

Analisis terhadap hasil uji coba dilakukan untuk mengetahui apakah hasil uji coba yang diperoleh sesuai dengan harapan atau tidak. Analisis yang dilakukan adalah membandingkan kinerja algoritma protokol *routing* AOMDV dengan protokol *routing* EA-AOMDV.

1. **Analisis *throughput***

Hasil rata-rata nilai *throughput* yang sesuai dengan hasil uji coba skenario pada Tabel 4.1, Tabel 4.2 dan Tabel 4.3 dapat dilihat pada Gambar 4.15 berikut ini:

|  |  |
| --- | --- |
|  | |
| (a) *Node 50,* luas area 500x500 m2 percobaan 1 | (b) *Node* 50*,* luas area 1000x1000 m2 percobaan 1 |

|  |  |
| --- | --- |
|  | |
| (c) *Node 100,* luas area 500x500 m2 percobaan 1 | (d) *Node* 100, luas area 1000x1000 m2 percobaan 1 |
|  | |
| (e) *Node 50,* luas area 500x500 m2 percobaan 2 | (f) *Node* 50*,* luas area 1000x1000 m2 percobaan 2 |
|  | |
| (g) *Node 100,* luas area 500x500 m2 percobaan 2 | (h) *Node* 100, luas area 1000x1000 m2 percobaan 2 |

|  |  |
| --- | --- |
|  | |
| (i) *Node* 50*,* luas area 500x500 m2 percobaan 3 | (j) *Node* 50, luas area 1000x1000 m2  percobaan 3 |
|  | |
| (k) *Node 100,* luas area 500x500 m2 percobaan 3 | (l) *Node* 100, luas area 1000x1000 m2  percobaan 3 |

Gambar 4.18 Grafik perbandingan nilai throughput pada 3 percobaan.

Berdasarkan ketiga percobaan yang dilakukan, rata-rata hasil uji coba *throughput* pada luas area 500x500 m2 untuk 50 *node* pada EA-AOMDV adalah 73,04 Kbps dan 63,44 Kbps pada AOMDV. Sedangkan rata-rata hasil uji coba untuk 100 *node* pada EA-AOMDV adalah 73,76 Kbps dan 62,94 Kbps pada AOMDV. Rata-rata hasil uji coba pada luas area 1000x1000 m2 untuk 50 *node* pada EA-AOMDV adalah 67,13 Kbps dan 59,11 Kbps pada AOMDV. Sedangkan rata-rata hasil uji coba untuk 100 *node* pada EA-AOMDV adalah 69,12 Kbps dan 58,15 Kbps pada AOMDV.

Pengiriman paket data oleh *throughput* dapat dipengaruhi oleh paket RREQ yang dikirimkan seperti jumlah *node* yang menentukan kepadatan lalu lintas untuk sampai ke *node* tujuan atau kecepatan transfer yang digunakan. Seperti terlihat pada grafik Gambar 4.18 nilai *throughput* pada *pause time* 45s relatif selalu turun dan naik pada saat berada di kondisi *pause time* 180s, perubahan signifikan dapat dilihat di *network area* 100x1000 m2. Hal ini disebabkan oleh adanya kemungkinan jarak antara *node* sumber dengan *node* tujuan yang sangat jauh di luas area yang besar. Ketika *node* sumber dan *node* tujuan saling bergerak berjauhan maka akan mengakibatkan semakin banyak *rute* yang terputus dan mengakibatkan proses pergantian *rute* semakin sering dilakukan. Pada percobaan ini semakin tinggi *pause time*, maka semakin lama waktu yang dibutuhkan tiap *node* untuk tetap stabil sebelum pergerakan *node* berikutnya.

Selain itu pada grafik terlihat nilai *throughput* pada *node* 100 relatif lebih tinggi dibandingkan *node* 50 terutama nilai EA-AOMDV. Adanya penambahan jumlah *node* menyebabkan nilai *throughput* pada protokol AOMDV mengalami peningkatan, hal ini disebabkan oleh banyak *node* yang terdapat pada lingkungan simulasi akan menetukan kepadatan lalu lintas jaringan sehingga semakin kecil kemungkinan terjadinya *link* terputus pada jalur komunikasi dan menyebabkan *lifetime* jaringan bertahan lebih lama.

Berdasarkan grafik hasil uji coba yang telah dilakukan, pada ketiga percobaan dapat diketahui bahwa nilai *throughput* yang diperoleh pada EA-AOMDV selalu lebih tinggi dibandingkan AOMDV standar. Hal ini disebabkan oleh algoritma EA- SHORT mampu meminimalisir terjadinya *link failure* dengan membandingkan *node-node* tetangga dari suatu *node* asal dan memilih *node* dengan energi paling tinggi untuk menghindari *node* kehabisan energi di tengah sebelum sampai pada *node* tujuan. Jadi berdasarkan grafik pada Gambar 4.18 nilai *throughput* pada protokol *routing* AOMDV dengan diterapkan algoritma EA-SHORT memberikan peningkatan nilai sebesar 10, 2245 Kbps, Ini membuktikan protokol *routing* EA-AOMDV lebih baik dari segi *throughput* jika dibandingkan dengan AOMDV, sehingga dapat dikatakan uji coba ini berhasil.

1. **Analisis *Packet Delivery Ratio* (PDR)**

Hasil rata-rata nilai *packet delivery ratio* yang sesuai dengan hasil uji coba skenario pada Tabel 4.4, Tabel 4.5 dan Tabel 4.6 dapat dilihat pada Gambar 4.19 berikut ini:

|  |  |
| --- | --- |
|  | |
| 1. *Node* 50*,* luas area 500x500 m2   percobaan 1 | (b) *Node* 50, luas area 1000x1000 m2  percobaan 1 |
|  | |
| 1. *Node* 100*,* luas area 500x500 m2   percobaan 1 | (d) *Node* 100, luas area 1000x1000 m2  percobaan 1 |
|  | |
| 1. *Node* 50*,* luas area 500x500 m2   percobaan 2 | (f) *Node* 50, luas area 1000x1000 m2  percobaan 2 |

|  |  |
| --- | --- |
|  | |
| (g) *Node* 100*,* luas area 500x500 m2 percobaan 2 | | (h) *Node* 100, luas area 1000x1000 m2  percobaan 2 | |

|  |  |
| --- | --- |
|  | |
| (i) *Node* 50*,* luas area 500x500 m2  percobaan 3 | | (j) *Node* 50, luas area 1000x1000 m2  percobaan 3 | |

|  |  |
| --- | --- |
|  | |
| (k) *Node* 100*,* luas area 500x500 m2 percobaan 3 | | (l) *Node* 100, luas area 1000x1000 m2  percobaan 3 | |

Gambar 4.19 Grafik perbandingan nilai *packet delivery ratio* pada 3 percobaan

Berdasarkan ketiga percobaan yang dilakukan, rata-rata hasil uji coba *packet delivery ratio* pada luas area 500x500 m2 untuk 50 *node* pada EA-AOMDV adalah 97,258 % dan 96,368 % pada AOMDV. Sedangkan rata-rata hasil uji coba untuk 100 *node* pada EA-AOMDV adalah 97,513% dan 96,631 % pada AOMDV. Rata- rata hasil uji coba pada luas area 1000x1000 m2 untuk 50 *node* pada EA-AOMDV adalah 91,870 % dan 89,797 % pada AOMDV. Sedangkan rata-rata hasil uji coba untuk 100 *node* pada EA-AOMDV adalah 91,512 % dan 89,417 % pada AOMDV.

Pada protokol *routing* AOMDV dan EA-AOMDV *pause time* mempengaruhi pegerakan *node*, semakin sering *node* bergerak, maka topologi yang terbentuk akan semakin sering berubah hal tersebut menyebabkan proses penemuan *rute* semakin sering dilakukan. Seperti terlihat pada grafik Gambar 4.16 nilai *packet delivery ratio* pada *pause time* 45s selalu mengalami penurunan. Selain itu semakin tinggi kecepatan *node*, maka *packet delivery ratio* yang dihasilkan semakin rendah. Dapat dilihat pada grafik saat kecepatan *node* menunjukkan angka tertinggi yaitu 10 m/s nilai *packet delivery ratio* semakin menurun terlebih pada *pause time* 45s. Hal ini disebabkan bertambahnya kecepatan gerak *node* maka akan sering terjadi perubahan posisi *node,* yang menyebabkan rute yang akan dilalui untuk mengirimkan paket juga akan semakin cepat berubah karena *rute* mengalami putus *link*, sehingga terjadi hilangnya paket data yang lewat (*packet drop*). Perubahan signifikan dapat dilihat di *network area* 100x1000 m2. Hal ini disebabkan oleh adanya kemungkinan jarak antara *node* sumbe dengan *node* tujuan sangat jauh di luas area yang besar, sehingga membutuhkan banyak *node* untuk mengirim paket data.

Berdasarkan grafik hasil uji coba yang telah dilakukan, pada ketiga percobaan dapat diketahui bahwa nilai *packet delivery ratio* yang diperoleh pada EA-AOMDV relatif lebih tinggi dibandingkan AOMDV standar. Hal ini disebabkan oleh algoritma EA-SHORT mampu meminimalisir terjadinya *link failure* dan *packet drop*. Jadi berdasarkan grafik pada Gambar 4.16 nilai *throughput* pada protokol *routing* AOMDV dengan diterapkan algoritma EA-SHORT memberikan peningkatan nilai sebesar 1, 48464%. Ini membuktikan protokol *routing* EA-AOMDV lebih baik dari segi *packet delivery ratio* jika dibandingkan dengan AOMDV, sehingga dapat dikatakan uji coba ini berhasil.

1. **Analisis *average end-to-end delay***

Hasil rata-rata nilai *end-to-end delay* sesuai dengan hasil uji coba skenario pada Tabel 4.7, Tabel 4.8 dan Tabel 4.9 dapat dilihat pada Gambar 4.20 berikut ini:

|  |  |
| --- | --- |
|  | |
| (a) *Node* 50*,* luas area 500x500 m2  percobaan 1 | (b) *Node* 50, luas area 1000x1000 m2  percobaan 1 | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | | | | |
| (c) *Node* 100*,* luas area 500x500 m2  percobaan 1 | | (d) *Node* 100, luas area 1000x1000 m2  percobaan 1 | | | |
|  | | | | | |
| (e) *Node* 50*,* luas area 500x500 m2  percobaan 2 | | (f) *Node* 50, luas area 1000x1000 m2  percobaan 2 | | | |
|  | | | | | |
| (g) *Node* 100*,* luas area 500x500 m2  percobaan 2 | | (h) *Node* 100, luas area 1000x1000 m2  percobaan 2 | | | |
|  | | | |
| (i) *Node* 50*,* luas area 500x500 m2  percobaan 3 | | (j) *Node* 50, luas area 1000x1000 m2  percobaan 3 | | |
|  | |
| (k) *Node* 100*,* luas area 500x500 m2  percobaan 2 | | (l) *Node* 100, luas area 1000x1000 m2  percobaan 2 | | | |

Gambar 4. 20 Grafik perbandingan nilai *end-toend delay* pada 3 percobaan.

Berdasarkan ketiga percobaan yang dilakukan, rata-rata hasil uji coba *average end-to-end delay* pada luas area 500 x 500 m2 untuk 50 *node* pada EA- AOMDV adalah 0,01048 s dan 0,01291 s pada AOMDV. Sedangkan rata-rata hasil uji coba untuk 100 *node* pada EA-AOMDV adalah 0,01196 s dan 0,01536 s pada AOMDV. Rata-rata hasil uji coba pada luas area 1000x1000 m2untuk 50 *node* pada EA- AOMDV adalah 0,02117 s dan 0,02526 s pada AOMDV. Sedangkan rata-rata hasil uji coba untuk 100 *node* pada EA-AOMDV adalah 0,02418 s dan 0,03072 s pada AOMDV. Nilai *average end-to-end delay* yang dihasilkan protokol *routing* EA-AOMDV lebih rendah dibandingkan dengan protokol *routing* AOMDV. Ini membuktikan protokol *routing*  EA- AOMDV lebih baik dari segi *average end-to-end delay* jika dibandingkan dengan AOMDV.

Dalam proses pengiriman paket data, terdapat banyak faktor yang mengakibatkan *delay*. Salah satunya terjadi ketika jumlah komunikasi bertambah akibat kepadatan *node,* kepadatan *node* dapat menyebabakn semakin banyak *node* perantara yang terpakai untuk mengirimkan layanan dari sumber ke tujuan sehingga waktu pengiriman semakin lama. Hal ini dapat dilihat dari nilai rata-rata keseluruhan *average end-to-end delay* pada jumlah 100 *node* selalu lebih tinggi dibanding nilai *average end-to-end delay* pada jumlah 50 *node.* Hasil nilai *average end-to-end delay* yang dihasilkan pada percobaan 1, percobaan 2, dan percobaan 3 terdapat beberapa perubahan yang signifikan pada luas area 1000x1000 m2 untuk protokol *routing* AOMDV. Hal ini disebabkan oleh adanya kemungkinan jarak antara *node* sumber dengan *node* tujuan sangat jauh di luas area yang besar, sehingga membutuhkan banyak *node* untuk mengirim paket data.

Adanya penerapan algoritma EA-SHORT pada protokol *routing* AOMDV terbukti dapat menurunkan nilai *delay*. Hal ini dikarenakan algoritma EA-SHORT meningkatkan optimasi *routing* dengan melakukan perhitungan energi sehingga memperkecil kemungkinan rute terputus sebelum *node* sumber sampai pada *node* tujuan. Pada Gambar 4.20 hasil menunjukan nilai *end-to-end delay* pada protokol *routing* AOMDV tanpa menggunakan algoritma dibandingkan dengan nilai *end-to-end delay* pada protokol *routing* dengan menggunakan algoritma EA-SHORT terbukti dapat menurunkan hasil *delay* sebesar 0.04115 s, sehingga dapat dikatakan uji coba pada penelitian ini berhasil.

# BAB V

# KESIMPULAN DAN SARAN

## Kesimpulan

Setelah melakukan simulasi dan analisa terhadap kedua protokol *routing* yaitu EA-AOMDV dan AOMDV maka dapat diambil kesimpulan sebagai berikut :

1. Berdasarkan hasil uji coba penelitian, penerapan algoritma EA-SHORT mampu meningkatkan performa protokol AOMDV. Di mana setelah penerapan algoritma EA-SHORT, rata-rata *throughput* mengalami peningkatan sebesar 13,92 %, *Packet Delivery Ratio* (PDR) sebesar 1,57 % dan *average end-to-end delay* menurun sebesar 19,53 %.
2. Pada protokol *routing* AOMDV dan EA-AOMDV *pause time* dan kecepatan *node* mempengaruhi pergerakan *node*. Semakin tinggi *pause time* dan rendah kecepatan *node*, maka semakin lama waktu yang dibutuhkan tiap *node* untuk tetap stabil sebelum pergerakan *node* berikutnya. Semakin rendah *pause time* dan tinggi kecepatan *node*, semakin sering *node* bergerak, maka rute yang terbentuk akan semakin sering berubah hal tersebut menyebabkan proses penemuan *rute* semakin sering dilakukan sehingga menyebabkan penurunan pada nilai *throughput* dan *packet delivery ratio dan* kenaikan pada nilai *end-to-end delay*.
3. Perubahan signifikan nilai parameter uji selalu terjadi pada *network area* 1000x1000 m2. Hal ini disebabkan oleh adanya jarak antara *node* sumber dengan *node* tujuan yang sangat jauh di luas area yang besar. Ketika *node* sumber dan *node* tujuan saling bergerak berjauhan akan mengakibatkan semakin banyak *rute* yang terputus dan mengakibatkan proses pergantian *rute* semakin sering dilakukan.

## Saran

Berdasarkan penelitian yang telah dilakukam, maka peneliti dapat memberikan saran-saran sebagai berikut :

1. Pada penelitian ini peningkatan nilai PDR tidak terlalu signifikan terlihat, sehingga penulis menyarankan untuk penelitian selanjutnya dapat menggunakan protokol *routing* dan algoritma *energy aware* yang berbeda pada jaringan MANET.
2. Melakukan penelitian selanjutnya dengan menggabungkan algoritma EA-SHORT dengan algoritma lainnya untuk mencari hasil yang lebih optimal.
3. Melakukan penelitian dengan menggunakan luas area simulasi yang bervariasi dan jumlah kepadatan *node* dan parameter uji lainnya yang bervariasi pula, untuk mendapatkan hasil yang valid terkait dengan pengaruh luas area, *pause time,* kecepatan *node* dan jumlah koneksi terhadap simulasi penelitian di jaringan MANET.
4. Melakukan penelitian selanjutnya dengan menggunakan protokol *routing* AOMDV dan algoritma EA-SHORT pada jenis jaringan yang berbeda.

# DAFTAR PUSTAKA

[1] C. Gui and P. Mohapatra, “A framework for self-healing and optimizing routing techniques for mobile ad hoc networks,” *Wirel. Networks*, vol. 14, no. 1, pp. 29–46, 2008.

[2] A. Z. I Made Windra Yudistiana, Andy Hidayat Jatmika, “Analisis Optimasi Kinerja Protokol Routing AODV Dan AOMDV Dengan Menggunakan Metode RFAP Untuk Mencegah RREQ Flooding Attacks Pada Jaringan MANET,” *Jtika*, vol. 1, no. 1, pp. 9–18, 2019.

[3] I. Aulia, F. A. Yulianto, and A. Herutomo, “Analisis Performansi Routing Protocol AOMDV dan DYMO pada Vehicular Ad Hoc Network,” pp. 1–6, 2013.

[4] Y. P. Wulandari, A. H. Jatmika, and F. Bimantoro, “Meningkatkan Efisiensi Rute Pada Protokol Routing AOMDV Menggunakan Metode PA-SHORT,” *Jtika*, vol. 1, no. 1, pp. 77–85, 2019.

[5] A. H. Jatmika, S. Djanali, and M. Husni, “Optimasi Routing pada Jaringan MANET,” pp. 1–7, 2011.

[6] W. A. Rachman, P. H. Trisnawan, and M. A. Fauzi, “Analisis Konsumsi Energi Protokol Routing Fisheye State Routing ( FSR ) Pada Mobile Ad Hoc Network ( MANET ),” vol. 3, no. 5, pp. 4455–4466, 2019.

[7] A. Fauzio, N. M. Adriansyah, and L. V. Yovita, “Analisis Performansi Ad-Hoc On-Demand Multipath Distance Vector Routing (AOMDV) dan Ad-Hoc On-Demand Distance Vector Routing (AODV) Pada MANET,” pp. 1–8, 2007.

[8] L. F. Z. Fauzi, “Analisis Performansi Multipath Routing Pada Jaringan Mobile Adhoc,” Telkom University, 2016.

[9] R. Asokan, A. M. Natarajan, and C. Venkatesh, “Quality-of-Service Routing Using Path and Power Aware Techniques in Mobile Ad Hoc Networks,” *J. Comput. Syst. Networks, Commun.*, vol. 2008, no. 160574, pp. 1–7, 2008.

[10] L. Oktami, “Optimasi Protokol Routing ZRP Menggunakan PA-SHORT Untuk Mengurangi Jumlah Hop Pada Suatu Rute di Jaringan MANET,” Universitas Mataram, 2019.

[11] F. Fitrianto, I. D. Irawati, R. M. Negara, H. Network, I. Vehicle, and C. Ivc, “Simulasi Dan Analisis Kinerja Protokol Routing Aomdv Dan ZRP Pada Jaringan Vehicular Ad Hoc Network ( Vanet ),” pp. 1–10, 2014.

[12] R. Anisia, R. Munadi, and R. M. Negara, “Analisis Performansi Routing Protocol OLSR Dan AOMDV Pada Vehicular Ad Hoc Network (VANET),” *J. Nas. Tek. Elektro*, vol. 5, no. 1, p. 87, 2016.

[13] A. B. Prasetio, “Analisis Performansi Multipath Routing Pada Jaringan Mobile Adhoc,” Universitas Telkom, 2006.

[14] M. K. Marina and S. R. Das, “Ad hoc on-demand multipath distance vector routing,” *Wirel. Commun. Mob. Comput.*, vol. 6, no. 7, pp. 969–988, 2006.