ANALISIS PERBANDINGAN KINERJA PROTOKOL AOMDV, DSR, DAN AODV SEBAGAI PROTOKOL *ROUTING* PADA *MOBILE AD-HOC NETWORK* (MANET)

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Muhammad Arhangga Satriawan
NIM: 165150207111075

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2020

PENGESAHAN

ANALISIS PERBANDINGAN KINERJA PROTOKOL AOMDV, DSR, DAN AODV SEBAGAI PROTOKOL *ROUTING* PADA *MOBILE AD-HOC NETWORK* (MANET)

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Muhammad Arhangga Satriawan
NIM: 165150207111075

Skripsi ini telah diuji dan dinyatakan lulus pada
17 Juli 2020
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I

Ir. Heru Nurwarsito., M.Kom.
NIP: 19650402 199002 1 001

Mengetahui
Ketua Jurusan Teknik Informatika

Fr. Astoto Kurniawan, S.T., M.T., Ph.D.
NIP: 19710518 200312 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar referensi.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 14 Agustus 2020

Muhammad Arhangga Satriawan

NIM: 165150207111075

ABSTRAK

Muhammad Arhangga Satriawan, Analisis Perbandingan Kinerja Protokol AOMDV, DSR, Dan AODV Sebagai Protokol *Routing* Pada *Mobile Ad-Hoc Network* (MANET)

Pembimbing: Ir. Heru Nurwarsito., M.Kom.

Mobile Ad-Hoc Network (MANET) merupakan jaringan *mobile* dimana perangkat yang terhubung secara *wireless* dapat mengonfigurasi jaringan *mobile*nya sendiri secara terus menerus tanpa dibutuhkan suatu infrastruktur. MANET dapat diartikan sebagai kumpulan *node* pada jaringan *wireless* yang bebas bergerak. *Node* mempunyai tanggung jawab dalam hal meneruskan data dari *source* ke *destination*. Permasalahan pada protokol AOMDV dan AODV yakni tidak memiliki mekanisme *route cache* yang dimiliki protokol DSR sehingga dapat menyebabkan proses pengiriman paket data antara *source* ke *destination* menjadi lebih lambat. Mekanisme *route cache* pada protokol DSR dapat menjadi solusi dikarenakan dapat mempercepat proses pengiriman paket data antara *source* ke *destination*. Penelitian dilakukan dengan cara membandingan kinerja protokol *routing* AOMDV, DSR, dan AODV berdasarkan parameter *throughput* dan *end to end delay* dengan variasi jumlah *node*. Hasil pada penelitian ini ialah pada protokol DSR memiliki kinerja terbaik pada *throughput* dengan nilai 215273,36 Kbps. Protokol DSR juga memiliki kinerja terbaik pada *end to end delay* dengan nilai 44,68 ms.

Kata kunci: *AOMDV, AODV, DSR, MANET*

ABSTRACT

Muhammad Arhangga Satriawan, Analisis Perbandingan Kinerja Protokol AOMDV, DSR, Dan AODV Sebagai Protokol *Routing* Pada *Mobile Ad-Hoc Network* (MANET)

Supervisors: Ir. Heru Nurwarsito., M.Kom.

Mobile Ad-Hoc Network (MANET) is a mobile network where wirelessly connected devices can configure their own mobile network continuously without the need for an infrastructure. MANET can be defined as a collection of nodes on a wireless network that are free to move. Nodes have the responsibility of forwarding data from source to destination. The problem with the AOMDV and AODV protocols is that they do not have the route cache mechanism that the DSR protocol has, so it can cause the process of sending data packets between source to destination to be slower. The route cache mechanism in the DSR protocol can be a solution because it can speed up the process of sending data packets between source to destination. The research was conducted by comparing the performance of the AOMDV, DSR, and AODV routing protocols based on the throughput and end to end delay parameters with variations in the number of nodes. The result of this research is that the DSR protocol has the best performance at the throughput with a value of 215273.36 Kbps. The DSR protocol also has the best performance on end to end delay with a value of 44.68 ms.

Keyword: *AOMDV, AODV, DSR, MANET*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Perubahan telah dialami oleh jaringan komputer dalam hal teknologi dari komputer-komputer yang terhubung menggunakan kabel menjadi tanpa kabel atau yang biasa disebut *wireless*. Menurut Jyoti & Saini (2017) Jaringan nirkabel atau yang biasa disebut dengan *wireless network* merupakan suatu jaringan yang menggunakan frekuensi radio ataupun sinyal untuk dijadikan acuan, sehingga mempunyai suatu fungsi agar setiap perangkat atau *device* dapat berbagi informasi terkait sumber daya. Telepon seluler, sensor nirkabel, dan laptop termasuk beberapa contoh perangkat *wireless*. Jaringan *wireless ad hoc* adalah bagian dari *Wireless Local Area Network* (WLAN) dan dapat didefinisikan sebagai komunikasi yang dilakukan oleh sekumpulan *node* antara satu simpul dengan simpul yang lain. *Interface wireless* dimiliki oleh tiap *node* pada jaringan *ad hoc*. Pada jaringan *ad hoc*, setiap *node* memiliki sifat dinamis dimana dapat berubah-ubah. Setiap simpul pada *ad hoc network* memiliki fungsi yaitu sebagai pendukung suatu jaringan seperti *router* dan sebagai penerima maupun pengirim suatu informasi.

Mobile Ad-Hoc Network (MANET) merupakan jaringan *mobile* dimana perangkat yang terhubung secara *wireless* dapat mengonfigurasi jaringan *mobile*nya sendiri secara terus menerus tanpa dibutuhkan suatu infrastruktur. MANET termasuk dalam *ad hoc network*. MANET dapat diartikan sebagai kumpulan *node* pada jaringan *wireless* yang bebas bergerak. *Node* mempunyai tanggung jawab dalam hal meneruskan data dari *source* ke *destination* (Khurana, 2017). Topologi jaringan pada MANET berbeda dengan yang lain. Pada MANET topologi bersifat dinamis serta kumpulan *node* bergerak secara acak dan cepat. Selain itu, MANET memiliki batasan *bandwidth* karena *node* selalu bergerak secara bebas serta berkomunikasi menggunakan jaringan *wireless*. Pada MANET, energi yang digunakan juga terbatas dikarenakan *node* berlaku sebagai *router* sehingga harus dirancang secara efisien dalam hal penggunaan energi.

Pada *Mobile Ad-Hoc Network* (MANET), protokol *routing* yang dapat digunakan dibagi menjadi tiga yaitu protokol *routing* proaktif, reaktif, dan *hybrid*. Protokol *routing hybrid* merupakan gabungan dari protokol *routing* proaktif dan reaktif. *Routing* proaktif adalah protokol yang memperbarui informasi tabel *routing* secara priodik pada saat terjadi perubahan pada topologi jaringan. Selanjutnya, *routing* reaktif merupakan protokol yang memperbarui informasi rute jika hanya dibutuhkan.

Pada penelitian ini nantinya akan membandingkan protokol dari ruang lingkup reaktif, hal ini dikarenakan protokol reaktif merupakan protokol yang sangat efektif apabila digunakan pada saat keadaan simpul sumber dan simpul tujuan memiliki jarak yang jauh atau tidak saling berdekatan. Selain itu, pada saat *node* sumber ingin melakukan pengiriman data namun tidak memiliki informasi terkait rute yang akan dilalui, protokol reaktif dapat memperbarui informasi rute

tersebut. Protokol *Ad Hoc on Demand Multipath Distance Vector* (AOMDV), *Dynamic Source Routing* (DSR), dan *Ad Hoc on Demand Distance Vector* (AODV) merupakan protokol reaktif yang akan dipilih pada *Mobile Ad-Hoc Network* (MANET). Alasan pemilihan protokol AOMDV, DSR, dan AODV untuk dibandingkan kinerjanya, dikarenakan pada MANET sering menerapkan protokol-protokol tersebut dan mempunyai perbedaan dalam segi algoritma walaupun sama-sama berjenis protokol reaktif.

Berdasarkan latar belakang diatas timbul permasalahan yakni permasalahan pada protokol AOMDV dan AODV yakni tidak memiliki mekanisme *route cache* yang dimiliki protokol DSR sehingga dapat menyebabkan proses pengiriman paket data antara *source* ke *destination* menjadi lebih lambat. Mekanisme *route cache* pada protokol DSR dapat menjadi solusi dikarenakan dapat mempercepat proses pengiriman paket data antara *source* ke *destination*.

Pada penelitian ini, nantinya menggunakan perangkat atau *tools Network Simulator* yang berfungsi sebagai simulasi jaringan sehingga didapatkan hasil kinerja antara protokol *routing* AOMDV, DSR, dan AODV untuk dibandingkan nantinya. Beberapa parameter perbandingan yang akan digunakan adalah *end-to-end delay* dan *throughput*. Dengan dilakukannya penelitian ini, diharapkan nantinya dapat diperoleh asumsi bahwa hasil analisis kinerja protokol DSR lebih baik dibandingkan dengan kinerja protokol AOMDV dan AODV.

1.2 Identifikasi Masalah

Identifikasi masalah pada penelitian ini merujuk atau mereferensi pada permasalahan penelitian yaitu :

1. Protokol *routing* yang dibandingkan yakni AOMDV, DSR, dan AODV mempunyai perbedaan dalam segi algoritma walaupun sama-sama berjenis protokol reaktif.
2. Protokol *routing* AOMDV dan AODV tidak memiliki mekanisme *route cache* yang dimiliki protokol DSR sehingga dapat menyebabkan proses pengiriman paket data antara *source* ke *destination* menjadi lebih lambat dan kecepatan transfer data menjadi tidak efektif.

1.3 Rumusan Masalah

Merujuk pada latar belakang yang telah dibahas sebelumnya, maka rumusan masalah pada penelitian ini yaitu:

1. Bagaimana implementasi protokol *routing* AOMDV, DSR, dan AODV pada *Mobile Ad-Hoc Network* (MANET)?
2. Bagaimana kinerja protokol *routing* AOMDV, DSR, dan AODV pada *Mobile Ad-Hoc Network* (MANET)?

1.4 Tujuan

Tujuan dari penelitian ini adalah sebagai berikut:

Tujuan Umum:

Mendapatkan hasil protokol *routing* mana yang memiliki kinerja terbaik pada protokol jenis reaktif yang terdiri dari AOMDV, DSR, dan AODV.

Tujuan Khusus:

1. Implementasi protokol *routing* AOMDV, DSR, dan AODV pada *Mobile Ad-Hoc Network* (MANET).
2. Mengetahui kinerja protokol *routing* AOMDV, DSR, dan AODV pada *Mobile Ad-Hoc Network* (MANET).

1.5 Manfaat

Penelitian ini memiliki manfaat yaitu:

1. Dapat dijadikan sebagai bahan acuan untuk penelitian selanjutnya yang membahas terkait membandingkan protokol *routing* pada lingkungan *Mobile Ad-Hoc Network* (MANET).
2. Dapat dijadikan referensi untuk penelitian selanjutnya dengan merujuk atau mereferensi pada kesimpulan serta saran pada penelitian ini.

1.6 Batasan Masalah

Pada penelitian ini, fokus permasalahan yang akan dibahas yaitu:

1. Pengujian berfokus pada analisis perbandingan kinerja tiga protokol *routing* pada *Mobile Ad-Hoc Network* (MANET).
2. Parameter uji yang digunakan untuk menganalis kinerja *routing protocol* AOMDV, DSR, dan AODV yaitu *throughput* dan *end-to-end delay*.
3. Protokol *routing* yang akan dianalisis perbandingan kinerjanya adalah AOMDV, DSR, dan AODV.
4. *Network Simulator* versi 2 merupakan *tools* yang digunakan untuk simulasi jaringan.
5. Aspek keamanan jaringan diabaikan pada penelitian ini.

1.7 Sistematika Pembahasan

Sistematika pembahasan dan penyusunan laporan pada penelitian ini dijelaskan sebagai berikut :

BAB 1 PENDAHULUAN

Bagian pendahuluan terdiri dari tujuh bagian yakni latar belakang, identifikasi masalah, rumusan masalah, tujuan, manfaat, batasan masalah dan sistematika pembahasan dari "Analisis Perbandingan Kinerja Protokol AOMDV, DSR, Dan AODV Sebagai Protokol *Routing* Pada *Mobile Ad-Hoc Network* (MANET)".

BAB 2 LANDASAN KEPUSTAKAAN

Landasan kepustakan merupakan bab selanjutnya setelah pendahuluan yang membahas terkait dasar teori dari *Mobile Ad Hoc Network* (MANET), protokol *routing Ad Hoc on Demand Multipath Distance Vector* (AOMDV), *Dynamic Source Routing* (DSR), dan *Ad Hoc on Demand Distance Vector* (AODV), yang menjadi rujukan ataupun referensi pada penelitian ini.

BAB 3 METODOLOGI

Bagian metodologi merupakan bab selanjutnya setelah landasan kepustakaan yang menjelaskan terkait metode yang digunakan pada penelitian diantaranya ialah studi literatur, analisis kebutuhan, perancangan sistem, implementasi, pengujian dan analisis serta yang terakhir pengambilan kesimpulan dan saran.

BAB 4 PERANCANGAN DAN IMPLEMENTASI

Bagian perancangan dan implementasi merupakan bab setelah metodologi yang menjelaskan terkait perancangan atau rencana pada penelitian. Selain itu, pada bab ini akan dijelaskan terkait implementasi pada MANET menggunakan protokol *routing* AOMDV, DSR, dan AODV.

BAB 5 HASIL DAN PEMBAHASAN

Bagian hasil dan pembahasan merupakan bab setelah perancangan dan implementasi yang membahas terkait pengujian dan hasil agar diketahui kinerja dari masing-masing protokol untuk dibandingkan nantinya.

BAB 6 PENUTUP

Bagian penutup merupakan bab setelah pengujian dan hasil dimana menjelaskan terkait kesimpulan. Kesimpulan tersebut diperoleh dari pengujian yang telah dilakukan pada penelitian ini, serta saran-saran untuk pengembangan lebih lanjut.

BAB 2 LANDASAN KEPUSTAKAAN

2.1 Kajian Pustaka

Kajian pustaka pada penelitian ini membahas terkait bahasan penelitian sebelumnya yang dapat menunjang dan dijadikan referensi maupun rujukan untuk penelitian yang diusulkan. Persamaan penelitian ini dengan penelitian sebelumnya yakni membahas terkait membandingkan protokol *routing* pada ruang lingkup MANET. Perbedaan penelitian ini dengan penelitian sebelumnya yaitu pada protokol *routing* yang dianalisis, skenario pengujian, serta parameter pengujian yang berbeda.

Tabel 2.1 Perbandingan Penulisan Dengan Penelitian Terkait

No.	Nama Penulis, Tahun, Judul	Persamaan	Perbedaan	
			Penelitian Terdahulu	Rencana Penelitian
1	Jogendra Kumar, Annapurna Singh, M. K. Panda, H. S. Bhadauria, 2016, "Study and Performace Analysis of Routing Protocol Based on CBR"	Protokol *routing* yang digunakan ialah AODV & DSR, parameter uji *throughput* & *end to end delay*	Membandingkan kinerja protokol AODV, DSR, DYMO, dan ZRP berdasarkan *average jitter*, *throughput*, *average end to end delay,* rasio pengiriman paket.	Membandingkan kinerja protokol AOMDV, DSR, dan AODV berdasarkan *throughput, end to end delay*. Diharapkan hasil dari penelitian ialah dapat mengetahui kinerja ketiga protokol *routing* tersebut
2	Rajesh Kochher, Ritu Mehta, 2018, "Performance Analysis of Reactive AODV and DSR with Hybrid GRP Routing Protocols under IEEE 802.11g MANET"	Protokol *routing* yang digunakan ialah AODV & DSR, parameter uji *throughput, end to end delay*	Membandingkan kinerja protokol AODV, DSR, dan GRP berdasarkan *end to end delay, network load, throughput, number of hops/route,* PDR dan NRL	Protokol *routing* AOMDV, DSR, dan AODV akan dibandingkan kinerjanya berdasarkan *throughput, end to end delay*.
3	K. Neeraj, K. Yedupati, A. SAI	Protokol *routing* yang	Membandingkan kinerja protokol	Membandingkan kinerja protokol

	Soumya, S. SAI Krishna, 2018, "PERFORMANCE ANALYSIS OF DIFFERENT ROUTING PROTOCOLS IN MANET USING DIFFERENT PARAMETERS IN DIFFERENT CHANGES"	digunakan ialah AODV & DSR, parameter uji *throughput, end to end delay*	*routing* pada MANET yaitu DSDV, AODV dan DSR berdasarkan *throughput, end to end delay,* dan *bundle conveyance proportion builds*	AOMDV, DSR, dan AODV berdasarkan *throughput, end to end delay*. Diharapkan hasil dari penelitian ialah dapat mengetahui kinerja ketiga protokol *routing* tersebut berdasarkan analisis pengujian
4	M. Irfan Haris, 2019, "Perbandingan Kinerja Protokol DSDV dan FSR Terhadap Model Node Tetap dan Node Bergerak"	parameter uji *end to end delay*	Membandingkan kinerja dua protokol *routing* pada MANET yaitu DSDV, dan FSR berdasarkan nilai waktu konvergensi, *end to end delay,* dan *routing overhead*	Kinerja pada protokol *routing* AOMDV, DSR, dan AODV akan dibandingkan kinerjanya berdasarkan *throughput, end to end delay*
5	Fatkhurrozi, 2018, "Analisis Perbandingan Kinerja Protokol AOMDV, DSDV, Dan ZRP Sebagai Protokol *Routing* Pada *Mobile Ad-Hoc Network* (MANET)	Protokol *routing* yang digunakan salah satunya AOMDV, parameter uji *throughput, end to end delay*	Membandingkan kinerja protokol *routing* pada MANET yaitu DSDV, AOMDV dan ZRP berdasarkan *throughput, end to end delay*, dan *packet delivery ratio*	Protokol *routing* AOMDV, DSR, dan AODV akan dibandingkan kinerjanya berdasarkan *throughput, end to end delay*.

Penelitian sebelumnya membandingkan protokol *routing* dari jenis protokol reaktif dan hybrid. Protokol *routing Ad-hoc On Demand Distance Vector* (AODV), *Dynamic Source Routing* (DSR), *Dynamic MANET On Demand Routing* (DYMO) untuk jenis protokol reaktif dan protokol *Zone Routing Protocol* untuk jenis protokol hybrid. Kinerja perbandingan dibuat dengan jaringan yang kecil pada aplikasi CBR yang berlaku dari sumber simpul ke simpul tujuan. Model mobilitas yang digunakan adalah *Random Way Point Model* untuk penempatan *node* secara acak karena *node* dinamis sering berubah setiap *node* diatur secara acak. Analisis dan evaluasi kinerja merujuk pada metrik kinerja yang dilakukan seperti *average*

jitter, throughput, average end to end delay, rasio pengiriman paket pada beban lalu lintas CBR. Simulasi pada penelitian ini dilakukan dengan menggunakan *tools* simulator Qualnet 5.0.2. Berdasarkan hasil pengujian, Protokol *Routing* DSR adalah yang terbaik dalam kasus *average jitter*, AODV dan DYMO memberikan kinerja yang konstan dan ZRP memberikan kinerja yang buruk. Pada kasus average *end to end delay*, protokol *routing* AODV, DSR, dan DYMO adalah protokol *routing* yang baik dan ZRP merupakan protokol yang buruk. Protokol DYMO dan DSR memberikan *throughput* yang optimal dibandingkan dengan protokol *routing* yang lainnya. (Kumar, et al., 2016).

Penelitian yang dilakukan oleh Kochher & Mehta (2016) telah membandingkan protokol *routing* dari jenis protokol yang berbeda yaitu protokol reaktif dan hybrid. Protokol *Ad-hoc On Demand Distance Vector* (AODV), *Dynamic Source Routing* (DSR) untuk jenis protokol reaktif dan *Geographical Routing Protocol* (GRP) untuk jenis protokol hybrid. *Geographical Routing Protocol* (GRP) terdiri dari dua properti yaitu *routing* reaktif dan proaktif untuk memberikan performa yang tinggi pada parameter *Quality of Service* (QoS). Parameter QoS yang diteliti diantaranya ialah *end to end delay, network load, throughput,* jumlah *hops* atau rute, PDR & NRL. Simulasi pada penelitian ini dilakukan dengan menggunakan *tools* OPNET Modeler versi 14.5. Dari hasil simulasi eksperimental, protokol GRP *hybrid* memiliki performansi lebih baik dibandingkan dengan protokol *routing* reaktif AODV dan DSR dalam hal QoS. Kinerja AODV dan DSR semakin menurun seiring dengan peningkatan kepadatan *node* dibandingkan dengan GRP. Selanjutnya, dari hasil investigasi juga diamati protokol *routing* GRP *hybrid* memiliki beban normalisasi yang sangat tinggi dibandingkan dengan protokol *routing* yang reaktif.

Penelitian selanjutnya membandingkan protokol *routing* dari jenis protokol yang berbeda yaitu protokol proaktif dan protokol reaktif. Protokol reaktif terdiri dari *Ad-Hoc On Demand Distance Vector* (AODV) dan *Dynamic Source Routing* (DSR), untuk protokol proaktif terdiri dari *Destination Sequenced Distance Vector* (DSDV). Simulasi pada penelitian ini dilakukan dengan menggunakan *tools* atau sistem uji *Network Simulator* 2 (NS-2). Berdasarkan hasil pengujian, dapat disimpulkan bahwa jika jumlah *node* lebih rendah maka protokol AODV memiliki *delay* lebih sedikit. Jika jumlah *node* lebih tinggi maka protokol DSDV memiliki sedikit penundaan. Pada parameter *throughput,* AODV adalah protokol terbaik diantara ketiganya. Sedikit jumlah *node*, energi rata-rata yang dikonsumsi oleh DSDV selalu lebih sedikit untuk sejumlah *node*. Untuk pertimbangan rata-rata konsumsi energi DSDV adalah protokol terbaik, sedangkan untuk parameter PDR dengan jumlah yang lebih sedikit DSR adalah protokol terbaik (Neeraj, et al., 2018).

Penelitian berikutnya membandingkan dua protokol *routing* pada *Mobile Ad-Hoc Network* (MANET) diantaranya ialah *Destination Sequenced Distance Vector* (DSDV), dan *Fisheye State Routing* (FSR). Parameter uji yang digunakan pada penelitian ini adalah *routing overhead*, *end to end delay*, dan waktu konvergensi. Skenario pengujian pada penelitian ini dibagi menjadi tiga skenario yaitu variasi jumlah paket yang dikirim, jumlah *node* terhadap *node* bergerak, serta variasi

jumlah *node* terhadap *node* tidak bergerak. Hasil dari dilakukannya pengujian yakni protokol DSDV lebih cocok diterapkan pada jaringan dengan skala yang kecil, dengan nilai *routing overhead*, waktu konvergensi, dan *end to end delay* secara berturut turut adalah 8.875, 0,1 sec, dan 31,607 ms. Sedangkan protokol FSR lebih cocok diterapkan pada jaringan skala besar dengan nilai *routing overhead*, waktu konvergensi, dan *end to end delay* secara berturut turut adalah 22.748, 0,118 sec, dan 20.813 ms (Haris, et al., 2019).

Penelitian yang dilakukan oleh Fatkhurrozi (2018) telah membandingkan protokol *routing* dari jenis protokol yang berbeda yaitu protokol proaktif, reaktif dan hybrid. Protokol *Destination Sequenced Distance Vector* (DSDV) untuk jenis protokol proaktif, *Ad-Hoc On Demand Multipath Distance Vector* (AOMDV) untuk jenis protokol reaktif dan *Zone Routing Protocol* (ZRP) untuk jenis protokol *hybrid*. *Network Simulator* 2 (NS-2) merupakan *tools* yang digunakan untuk melakukan simulasi pada penelitian ini. *Normalized routing load, end-to-end delay, throughput* serta *packet delivery ratio* merupakan empat parameter yang dijadikan rujukan saat melakukan pengujian. Mereferensi pada skenario variasi jumlah *node* dan luas area dapat ditarik kesimpulan yakni protokol DSDV memiliki nilai rata-rata terbaik pada aspek *normalized routing load* dan *end-to-end delay*, sedangkan nilai rata-rata *throughput* dan *packet delivery ratio* tertinggi dimiliki oleh protokol AOMDV.

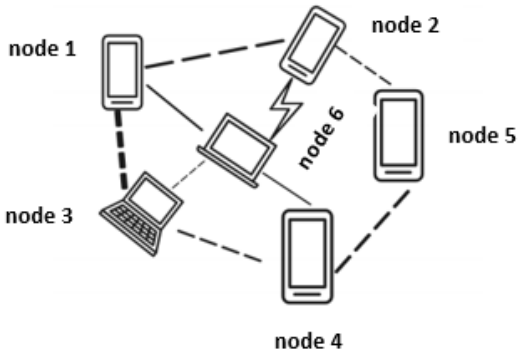
2.2 Dasar Teori

Dasar teori berisi pembahasan dan uraian terkait teori, konsep, analisis, dan model yang berasal dari metode sistem, atau pustaka ilmiah, yang berkaitan dengan penelitian ini. Dasar teori pada penelitian ini menyangkut MANET, protokol *routing* AOMDV, DSR, AODV, dan parameter pengujian.

2.2.1 *Mobile Ad-Hoc Network* (MANET)

Menurut Conti (2003) *Mobile Ad-Hoc Network* (MANET) merepresentasikan sistem sistem *mobile* berbasis nirkabel (tanpa kabel) dimana dapat mengonfigurasi dirinya sendiri secara acak maupun dinamis. MANET terdiri dari berbagai *node* seluler yang merupakan gabungan dari koneksi nirkabel. *Router* berfungsi untuk membuat rute dan setiap ponsel dijadikan sebagai *host*. Setiap *node* seluler bertindak sebagai *router* pemancar atau *transmitter*. Protokol perutean MANET dibagi menjadi tiga jenis protokol yaitu protokol proaktif, reaktif, dan *hybrid*. Cara kerja protokol jenis proaktif yaitu dengan cara mengelola daftar *destination node* dan *source node* yang terbaru pada masing-masing melalui pendistribusian tabel perutean ke seluruh jaringan. Hal tersebut dapat menyebabkan perlambatan pada aliran data jika terjadi restrukturisasi tabel perutean dikarenakan *traffic* atau jalur lalu lintas yang sering dilalui oleh tabel perutean tersebut. Protokol reaktif bekerja dengan cara *on demand* atau mencari rute dengan *flooding* pada jaringan dengan paket RREQ atau yang biasa disebut dengan *route request*. Hal tersebut dapat menyebabkan *clogging* atau penuh pada jaringan. Protokol jenis *hybrid* merupakan jenis protokol yang menggabungkan

keuntungan atau kelebihan yang ada pada protokol proaktif dan reaktif yakni dengan cara awalnya, mekanisme proaktif digunakan untuk membuat rute. Selanjutnya, reaktif *flooding* akan mengirimkan paket RREQ ke *node* terdekat. Beberapa contoh protokol routing pada MANET diantaranya ialah *Optimized Link State Routing* (OLSR), *Intrazone Routing Protocol* (IARP), *Witness Aided Routing* (WAR), Ad Hoc on Demand Distance Vector (AODV).



Gambar 2.1 Ilustrasi Struktur Jaringan MANET

Sumber : (Jaganath & Vikram, 2019)

Mobile Ad-Hoc Network (MANET) berbeda dengan *Vehicular Ad-Hoc Network* (VANET) dalam hal karakteristik. Karakteristik pada MANET diantaranya ialah :

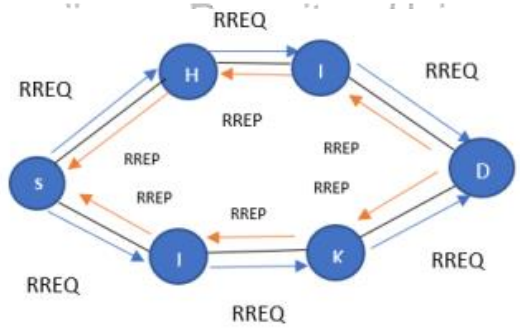
- Pada MANET, *node* menjalin komunikasi dengan *node* yang lainnya berbasis tanpa kabel atau yang biasa disebut dengan nirkabel. Selain itu, *node* juga dapat berbagi melalui media yang sama. Beberapa contoh media antara lain ialah inframerah, radio, dll.)
- MANET merupakan jaringan yang berbasis *ad-hoc*. Jaringan seluler *ad-hoc* merupakan jaringan yang bersifat sementara dimana jaringan tersebut terbentuk secara dinamis oleh sekumpulan *node* pada saat diperlukan.
- MANET tidak bergantung pada administrasi terpusat atau yang biasa disebut dengan *centralized administration*. MANET juga tidak bergantung pada infrastruktur dimana setiap simpul melakukan operasi dengan mode *peer to peer* terdistribusi. Selain itu, setiap simpul juga dapat menghasilkan data yang independen dan dapat bertindak sebagai *router* independen.
- Pada MANET, tidak diperlukan maupun dibutuhkan *router* khusus karena setiap simpul bertindak sebagai *router* yang mempunyai fungsi yaitu memungkinkan untuk berbagi informasi antara *mobile hosts* berdasarkan masing-masing paket yang telah diteruskan sebelumnya.
- Setiap simpul pada MANET dapat bebas bergerak sekaligus berkomunikasi dengan simpul lain. Sifat dinamis dimiliki oleh topologi jaringan *ad-hoc* dikarenakan simpul yang berpartisipasi terus bergerak. Hal tersebut menyebabkan perubahan yang terus menerus pada pola komunikasi antar simpul (Basagni, et al., 2004).

Nodes dalam fungsi MANET baik sebagai host maupun sebagai *router* untuk merutekan data antar *node* dalam jaringan. Ketika *source node* mengirim data dan jika dalam hal ini *node destination* tidak dalam koleksi *source node*, maka teknik *circumstance routing* digunakan untuk memastikan bahwa paket yang diteruskan menyentuh *node* tujuan. Pada MANET, informasi yang ada bertukar detail perutean dan juga menyimpan data kedalam tabel routing dengan cara yang benar (Jaganath & Vikram, 2019).

2.2.2 *Ad-Hoc on Demand Multipath Distance Vector* (AOMDV)

Ad-Hoc on Demand Multipath Distance Vector (AOMDV) merupakan sebuah protokol *routing* yang bekerja pada ruang lingkup reaktif. Menurut Basagni (2004) Protokol *routing* pada jaringan *ad-hoc* memiliki fungsi utama yakni untuk membuat rute dimana rute tersebut efisien dan benar antar *node*, sehingga pesan yang dikirimkan tepat waktu.

AOMDV merupakan protokol *routing* jenis reaktif yang diterapkan pada lingkungan *Mobile Ad-Hoc Network* (MANET). Protokol jenis reaktif memiliki mekanisme atau cara kerja yaitu dengan cara *on demand* atau mencari rute dengan *flooding* pada jaringan dengan paket RREQ atau yang biasa disebut dengan *route request*. Hal tersebut dapat menyebabkan *clogging* atau penuh pada jaringan. AOMDV merupakan protokol *routing* yang mempunyai mekanisme utama yakni *route discovery* dan *route maintenance*. Berbeda dari protokol *routing* AODV, pada AOMDV *destination node* akan menerima semua paket *route request* baik yang terduplikasi ataupun tidak. *Route request* (RREQ) tersebut ditandai dengan *sequence number* yang berbeda, dimana akan digunakan untuk membentuk rute cadangan dari *source node* menuju *destination node*. Sehingga, AOMDV memiliki lebih dari satu *path* atau *multipath* (Bahari, et al., 2019).



Gambar 2.2 Route Discovery AOMDV

Sumber : (Bahari, et al., 2019)

Ad-Hoc on Demand Multipath Distance Vector (AOMDV) memiliki beberapa karakteristik diantaranya ialah :

- Protokol AOMDV menggunakan pendekatan *hop-by-hop*
- AOMDV merupakan protokol *routing* berbasis vektor
- Melakukan mekanisme pencarian rute atau yang biasa disebut dengan *route discovery* hanya ketika dibutuhkan saja

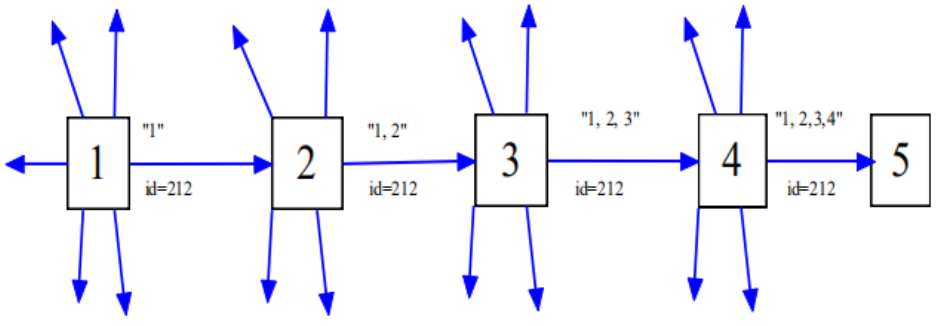
- Pada AOMDV, *source node* akan mempertimbangkan setiap *route reply* sehingga dapat ditemukan beberapa jalur dalam *route discovery* yang dilakukan satu kali
- Beberapa jalur atau *path* yang telah ditemukan dapat dijadikan jalur alternatif pada saat terjadi kegagalan rute
- Pada saat semua rute yang ditemukan mengalami kegagalan, maka akan dilakukan *route discovery* yang baru

Menurut Anisia (2016) Protokol *routing* AOMDV dan AODV mempunyai perbedaan yang utama yakni merujuk pada proses *route discovery* atau yang biasa disebut dengan pencarian rute terhadap jumlah *route* yang ditemukan. Dalam proses *route discovery* atau pencarian rute, AODV hanya memilih satu *route reply* atau yang biasa disebut dengan RREP. Pada AOMDV, beberapa jalur bisa ditemukan dalam satu *route discovery* dikarenakan setiap *route reply* akan dipertimbangkan oleh *source node*. Berdasarkan mekanisme tersebut, jika ada kerusakan atau kegagalan pada rute maka dapat dialihkan menuju *route* alternatif lain dengan merujuk pada penemuan beberapa jalur atau *path* sebelumnya. Selanjutnya, apabila semua *route* yang telah ditemukan kegagalan, maka pencarian rute yang baru akan dilakukan.

2.2.3 *Dynamic Source Routing* (DSR)

Dynamic Source Routing (DSR) merupakan sebuah protokol *routing* yang bekerja pada ruang lingkup reaktif. Protokol *routing* pada jaringan *ad-hoc* memiliki fungsi utama yakni untuk membuat rute dimana rute tersebut efisien dan benar antar *node*, sehingga pesan yang dikirimkan tepat waktu.

DSR merupakan protokol *routing* jenis reaktif yang diterapkan pada lingkungan *Mobile Ad-Hoc Network* (MANET). Protokol jenis reaktif memiliki mekanisme atau cara kerja yaitu dengan cara *on demand* atau mencari rute dengan *flooding* pada jaringan dengan paket RREQ atau yang biasa disebut dengan *route request*. Hal tersebut dapat menyebabkan *clogging* atau penuh pada jaringan. Mekanisme utama pada protokol DSR dibagi menjadi dua, mekanisme pertama ialah *route discovery* yang berfungsi untuk proses penemuan rute pada saat simpul sumber ingin mengirim paket data menuju simpul tujuan. Selanjutnya mekanisme kedua ialah *route maintenance* atau pemeliharaan rute berfungsi untuk memilih jalur alternatif pada saat jalur rute untuk mengirim paket data antara simpul sumber dan simpul tujuan rusak (Kumar, et al., 2016).



Gambar 2.3 Route Discovery DSR

Sumber : (Kumar, et al., 2016)

Dynamic Source Routing (DSR) memiliki beberapa karakteristik diantaranya ialah :

- Memiliki 2 mekanisme utama yaitu *route discovery* dan *route maintenance*
- *Route discovery* berfungsi untuk proses penemuan rute pada saat simpul sumber ingin mengirim paket data menuju simpul tujuan
- *Route maintenance* atau pemeliharaan rute berfungsi untuk memilih jalur alternatif pada saat jalur rute untuk mengirim paket data antara simpul sumber dan simpul tujuan rusak
- Beberapa jalur atau *path* yang telah ditemukan dapat dijadikan jalur alternatif pada saat terjadi kegagalan rute
- DSR memiliki tiga tipe pesan yang terdiri dari RREQ, RREP, dan RRER

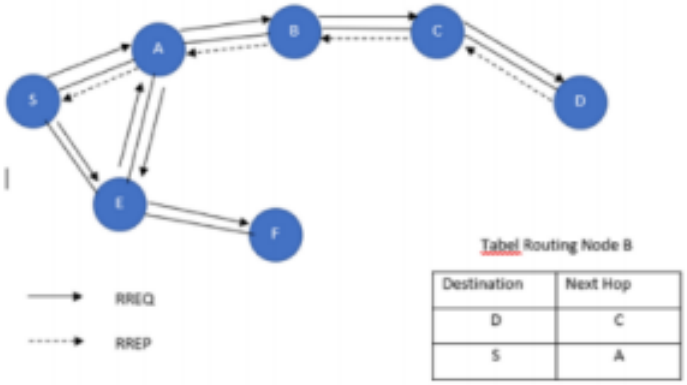
Protokol *routing* DSR memiliki mekanisme yang hampir sama dengan protokol *routing* AODV yakni apabila simpul sumber tidak memiliki informasi mengenai jalur rute pada saat akan melakukan pengiriman paket data, DSR akan memulai proses *route discovery* dengan cara menyebarkan paket RREQ ke *node* terdekatnya. Alamat pengirim dan alamat tujuan merupakan isi dari paket RREQ. Simpul yang menerima paket RREQ akan menyimpan informasi mengenai jalur tersebut kedalam *cache memory*. Selanjutnya, jika rute telah ditemukan maka simpul akan mengirimkan paket RREP untuk membalas rute asalnya. Paket RREP dikirimkan melalui jalur *reverse path* yang terbentuk pada saat pengiriman paket RREQ sebelumnya.

2.2.4 *Ad-Hoc on Demand Distance Vector* (AODV)

Ad-Hoc on Demand Distance Vector (AODV) merupakan sebuah protokol *routing* yang bekerja pada ruang lingkup reaktif. Protokol *routing* pada jaringan *ad-hoc* memiliki fungsi utama yakni untuk membuat rute dimana rute tersebut efisien dan benar antar *node*, sehingga pesan yang dikirimkan tepat waktu.

Ad-Hoc on Demand Distance Vector (AODV) merupakan protokol *routing* jenis reaktif yang dapat diterapkan pada *Mobile Ad-Hoc Network* atau yang biasa disebut dengan MANET. Protokol jenis reaktif memiliki mekanisme atau cara kerja yaitu dengan cara *on demand* atau mencari rute dengan *flooding* pada jaringan dengan paket RREQ atau yang biasa disebut dengan *route request*. Hal tersebut dapat menyebabkan *clogging* atau penuh pada jaringan. Protokol *routing* AODV termasuk kedalam jenis protokol reaktif dikarenakan protokol *routing* ini bekerja ketika ada *request* yang berasal dari simpul asal untuk mencari tahu jalur-jalur atau *path* dimana jalur tersebut digunakan untuk mengirim sebuah pesan menuju simpul tujuan. Pada protokol *routing* ini, pesan yang dikirim akan sampai pada simpul tujuan dengan cara menemukan jalur atau *path* terpendek maupun jalur yang tidak terdapat *loop* atau putaran. Berdasarkan yang telah dijelaskan

sebelumnya, protokol *routing* AODV bekerja saat ada *request* yang berasal dari simpul asal untuk menemukan *route* menuju simpul tujuan.



Tabel Routing Node B

Destination	Next Hop
D	C
S	A

Gambar 2.4 Route Discovery AODV

Sumber: (Bahari, et al., 2019)

Ad-Hoc on Demand Dictance Vector (AODV) memiliki beberapa karakteristik diantaranya ialah :

- AODV memiliki tiga tipe pesan yang terdiri dari RREQ, RREP, dan RRER
- Memiliki dua proses utama yakni *route maintenance* (pemeliharaan rute) dan *route discovery* (pencarian rute)
- Proses *route discovery* ialah *broadcoast* berupa RREQ yang berisikan simpul mana yang akan dijadikan *destination* yang disebar oleh simpul sumber, selanjutnya menunggu RREP
- Berbeda dengan AOMDV, pada AODV duplikasi RREQ pada sebuah simpul akan dibuang
- Paket *Route Error* atau yang biasa disebut dengan RRER bertugas untuk melakukan *route maintenance* (Bahari, et al., 2019)

AODV adalah sebuah protokol *routing* jenis reaktif yang mempunyai fungsi yakni untuk mencari jalur(*path*) yang terbaik dari *source node* ke *destination node*. Terdapat tiga tipe pesan pada protokol *routing* ini yakni *Route Error (RRER), Route Reply (RREP)*, dan *Route Request (RREQ)*. Protokol ini juga mempunyai dua proses utama yakni *Route Maintenance* dan *Route Discovery*. *Route Discovery* bekerja dengan cara *source node* menyebarkan *broadcoast* berupa *RREQ*. *Route request* tersebut berisi *node* mana yang akan menjadi *destination* dan kemudian menunggu *RREP*.

2.2.5 Parameter Pengujian

Parameter pengujian pada protokol *routing* AOMDV, DSR, dan AODV pada MANET meliputi *throughput* serta *end to end delay*. Parameter tersebut dapat dijadikan rujukan dalam melakukan pengujian pada penelitian.

2.2.5.1 Throughput

Throughput merupakan parameter uji yang merujuk pada kecepatan transfer data efektif dimana data tersebut dikalkulasi dalam *bit per second* atau bps sebagai total banyaknya paket yang diterima dibagi dengan waktu pengiriman. Perhitungan pada *throughput* ialah jumlah paket yang diterima dibagi dengan jumlah waktu pengamatan. Berikut formulasi perhitungan dari *throughput*:

$$Throughput = \frac{jumlah\ paket\ data\ diterima}{jumlah\ waktu\ pengamatan}$$

Throughput mereferensi atau merujuk pada tingkat transmisi efektif secara menyeluruh, dimana mempertimbangkan beberapa hal mencakup inefisiensi protokol, lalu lintas yang saling bersaing, dan overhead transmisi. *Throughput* pada umumnya diukur di lapisan jaringan dimana lapisan jaringan tersebut lebih tinggi daripada kecepatan data (Dordal, 2020).

2.2.5.2 End-to-end Delay

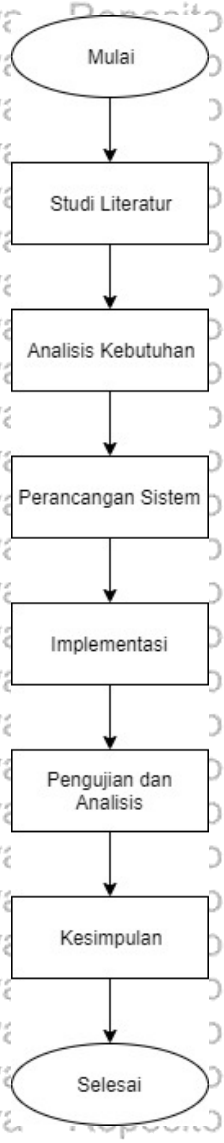
End-to-end delay merupakan parameter uji yang merujuk pada total waktu yang diambil yang berasal dari paket data yang tiba pada *destination*. Paket data yang dihitung ialah paket data yang berhasil dikirim ke *destination*. Perhitungan pada *end-to-end delay* ialah jumlah total waktu paket yang diterima dibagi dengan jumlah paket yang diterima. Berikut formulasi perhitungan dari *end-to-end delay*:

$$End\ to\ end\ Delay = \frac{jumlah\ total\ waktu\ paket\ diterima}{jumlah\ paket\ diterima}$$

Menurut Kurose & Ross (2017) *End-to-end delay* merupakan pemrosesan, akumulasi dari sebuah transimi, dan penundaan antrian pada router, keterlambatan pemrosesan pada sistem akhir, dan keterlambatan propagasi dalam tautan.

BAB 3 METODOLOGI

Pada bab ini dijelaskan langkah – langkah serta metode yang akan digunakan dalam penelitian ini. Tahapan metodologi penelitian yang digunakan pada laporan skripsi ini adalah studi literatur, analisis kebutuhan, perancangan sistem, implementasi, serta pengujian dan analisis. Adapun tahapan metodologi penelitian ditunjukkan pada diagram alir yang ada pada Gambar 3.1.



Gambar 3.1 Diagram Alur Metodologi Penelitian

3.1 Studi Literatur

Dalam tahap studi literatur, dilakukan studi pustaka dan literatur untuk mendapatkan informasi yang berhubungan dengan penelitian untuk mendapatkan referensi terkait analisis perbandingan suatu protokol *routing*. Informasi tersebut dapat dijadikan dasar pemilihan dan rujukan dalam melakukan penelitian. Literatur-literatur yang digunakan diperoleh dari buku, paper

internasional, dokumentasi internet, maupun pekerjaan yang menggunakan topik yang sama yaitu tentang analisis perbandingan suatu protokol *routing*. Selanjutnya, akan dibahas mengenai analisis kebutuhan-kebutuhan apa saja yang harus dipenuhi dalam melakukan penelitian ini.

3.2 Analisis Kebutuhan

Pada analisis kebutuhan membahas terkait kebutuhan-kebutuhan apa saja yang harus dipenuhi dalam melakukan penelitian ini. Pada penelitian ini, menggunakan sebuah perangkat yang dapat melakukan simulasi protokol *routing* AOMDV, DSR, dan AODV pada jaringan *Mobile Ad-Hoc Network* (MANET). Terdapat dua kebutuhan pada penelitian ini yaitu kebutuhan perangkat keras dan kebutuhan perangkat lunak. Kebutuhan perangkat lunak dan perangkat keras masing-masing terdiri atas dua komponen.

3.2.1 Kebutuhan Perangkat Lunak

1. *Network Simulator* 2 (NS-2)
2. Sistem operasi Linux

3.2.2 Kebutuhan Perangkat Keras

1. Laptop
2. RAM 4 GB

3.3 Perancangan Sistem

Pada perancangan sistem ini menjelaskan tentang rancangan kerja dari protokol AOMDV, DSR, dan AODV pada MANET secara menyeluruh agar mencapai tujuan penelitian. Pada tahap perancangan yang dilakukan yaitu perancangan topologi, konfigurasi protokol *routing* dan perancangan pengujian. Perancangan topologi menjelaskan terkait rancangan topologi yang digunakan untuk referensi atau rujukan pengujian nantinya. Perancangan pengujian menjelaskan terkait rancangan skenario berdasarkan nilai – nilai dari parameter yang akan digunakan sebagai karakteristik untuk jaringan MANET. Nilai-nilai parameter pada rancangan skenario diantaranya ialah simulator yang digunakan, protokol yang akan diuji, lama waktu uji, jumlah *node*. Selain itu, perancangan pengujian juga digunakan untuk rujukan atau referensi dalam melakukan pengujian nantinya terhadap protokol AOMDV, DSR, dan AODV. Parameter pengujian yang dilakukan pada penelitian ini yaitu *end-to-end delay* dan *throughput*.

Network Simulator 2 atau yang biasa disebut dengan NS-2 akan dijadikan sebuah *tool* yang digunakan untuk melakukan simulasi jaringan pada penelitian ini. *Throughput* dan *end-to-end delay* merupakan dua parameter pengujian yang menjadi fokus pada penelitian ini. Perancangan pengujian merujuk atau mereferensi pada skenario yang digunakan pada protokol yang diterapkan di *Network Simulator* 2. Jumlah *node*, lama waktu uji, protokol yang akan disimulasi

serta *tool* simulator yang digunakan untuk mensimulasikan jaringan merupakan beberapa parameter yang akan dijadikan rujukan pada penelitian ini.

3.4 Implementasi

Implementasi merupakan tahap selanjutnya setelah tahap perancangan pada sistem yang akan diteliti. Implementasi merupakan tahap yang membahas terkait penerapan protokol *Ad-Hoc on Demand Multipath Distance Vector* (AOMDV), *Dynamic Source Routing* (DSR), dan *Ad-Hoc on Demand Distance Vector* (AODV) menggunakan *tool* NS 2 pada *Mobile Ad-Hoc Network* (MANET). Sistem operasi linux merupakan sistem operasi yang digunakan untuk men*download*, memasang serta implementasi *tool* simulasi jaringan yaitu NS 2.

3.5 Pengujian dan Analisis

Setelah selesai pada tahap implementasi, dilakukan tahap pengujian dan analisis. Tahap pengujian dan analisis bertujuan untuk menguji kinerja dari setiap protokol *routing*. Hasil dari pengujian kemudian dianalisis untuk mengetahui perbandingan kinerja setiap protokol *routing* berdasarkan pengaruh dari skenario penelitian serta parameter uji *throughput* serta *end to end delay*. Selanjutnya, dari hasil pengujian dan analisis yang dilakukan akan dibahas mengenai kesimpulan dan saran. Kesimpulan dan saran merujuk atau mereferensi pada pengujian dan analisis yang telah dilakukan sebelumnya yang membahas terkait kesimpulan apa saja yang didapat. Saran berguna untuk penelitian-penelitian selanjutnya di masa mendatang. Adapun skenario pengujian pada penelitian ini yaitu mencakup :

1. *Throughput*

   Throughput merupakan parameter uji yang merujuk pada kecepatan transfer data efektif dimana data tersebut dikalkulasi dalam *bit per second* atau bps sebagai total banyaknya paket yang diterima dibagi dengan waktu pengiriman. Perhitungan pada *throughput* ialah jumlah paket yang diterima dibagi dengan jumlah waktu pengamatan.

2. *End-to-End Delay*

   End-to-end delay merupakan parameter uji yang merujuk pada total waktu yang diambil yang berasal dari paket data yang tiba pada *destination*. Paket data yang dihitung ialah paket data yang berhasil dikirim ke *destination*. Perhitungan pada *end-to-end delay* ialah jumlah total waktu paket yang diterima dibagi dengan jumlah paket yang diterima.

3.6 Kesimpulan dan Saran

Pada sub bab ini, dilakukan pengambilan kesimpulan berdasarkan pada hasil dari perancangan, implementasi, pengujian dan analisis yang telah dirancang. Saran diharapkan dapat memberikan pertimbangan untuk melakukan penelitian lebih lanjut.

BAB 4 PERANCANGAN DAN IMPLEMENTASI

Kebutuhan sistem, perancangan dan implementasi yang dilakukan pada penelitian akan dijelaskan didalam bab ini. Kebutuhan sistem menjelaskan terkait kebutuhan apa saja yang dibutuhkan oleh sistem. Perancangan topologi variasi jumlah *node*, konfigurasi protokol *routing* serta perancangan pengujian merupakan perancangan yang akan dibahas. Penjelasan terkait implementasi protokol *routing* serta implementasi NS-2.35 akan dijelaskan pada bagian implementasi.

4.1 Kebutuhan Sistem

Pada kebutuhan sistem membahas terkait kebutuhan-kebutuhan apa saja yang harus dipenuhi dalam melakukan penelitian ini. Kebutuhan sistem dibagi menjadi dua yaitu kebutuhan fungsional dan kebutuhan non-fungsional. Kebutuhan fungsional menjelaskan tentang *requirements* apa saja yang harus dicapai pada sistem, sedangkan kebutuhan perangkat lunak dan kebutuhan perangkat keras akan dibahas pada kebutuhan non-fungsional.

4.1.1 Kebutuhan Fungsional

Kebutuhan fungsional merupakan bagian yang membahas atau menjelaskan terkait kebutuhan-kebutuhan atau *requirements* apa saja yang harus dipenuhi maupun dicapai oleh sistem dimana akan dijelaskan pada tabel berikut :

Tabel 4.1 Kebutuhan Fungsional

No	Kebutuhan Fungsional
1	Protokol *routing* dapat diimplementasikan pada perangkat lunak *Network Simulator* 2.35
2	Pengujian protokol *routing* dapat dijalankan pada perangkat lunak *Network Simulator* 2.35 berdasarkan skenario yang telah ditetapkan sebelumnya
3	Pengujian *QoS* berupa pengujian *throughput* serta *end-to-end delay* dapat dijalankan pada perangkat lunak *Network Simulator* 2.35

4.1.2 Kebutuhan Non-Fungsional

Kebutuhan non-fungsional merupakan bagian yang membahas atau menjelaskan terkait kebutuhan yang dibagi menjadi dua yaitu kebutuhan perangkat lunak dan kebutuhan perangkat keras dimana akan dijelaskan pada tabel berikut :

Tabel 4.2 Kebutuhan Non-Fungsional Perangkat Keras

No	Perangkat Keras
1	Laptop

2	RAM 4 GB

Tabel 4.3 Kebutuhan Non-Fungsional Perangkat Lunak

No	Perangkat Lunak
1	*Network Simulator* 2 (NS-2)
2	Sistem Operasi Linux

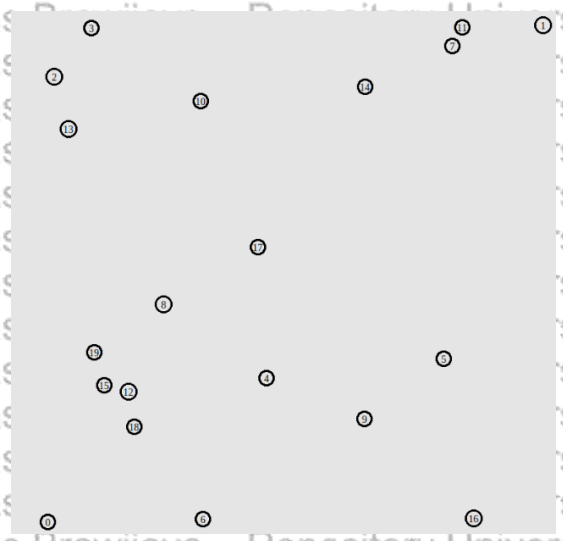
4.2 Perancangan Sistem

Pada perancangan sistem ini menjelaskan tentang rancangan kerja dari protokol AOMDV, DSR, dan AODV pada MANET secara menyeluruh agar mencapai tujuan penelitian. Pada tahap perancangan yang dilakukan yaitu perancangan topologi, konfigurasi protokol *routing* dan perancangan pengujian. Perancangan topologi menjelaskan terkait rancangan topologi yang digunakan untuk referensi atau rujukan pengujian nantinya. Perancangan pengujian menjelaskan terkait rancangan skenario berdasarkan nilai – nilai dari parameter yang akan digunakan sebagai karakteristik untuk jaringan MANET. Nilai-nilai parameter pada rancangan skenario diantaranya ialah simulator yang digunakan, protokol yang akan diuji, lama waktu uji, besar area simulasi saat pengujian, jumlah *node*. Selain itu, perancangan pengujian juga digunakan untuk rujukan atau referensi dalam melakukan pengujian nantinya terhadap protokol AOMDV, DSR, dan AODV. Parameter pengujian yang dilakukan pada penelitian ini yaitu *end-to-end delay, throughput*.

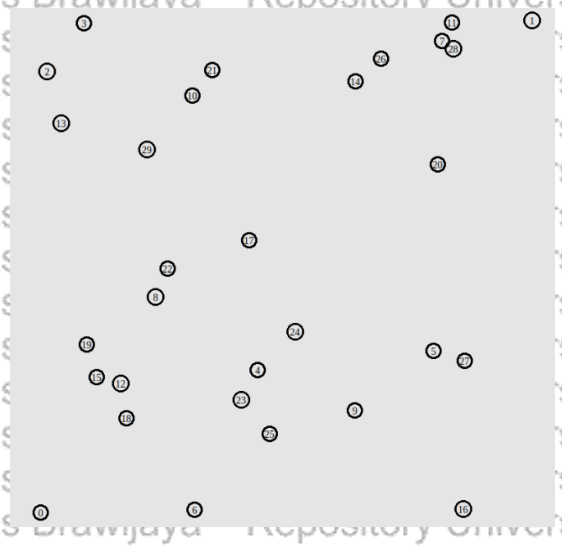
4.2.1 Perancangan Topologi

Beberapa rancangan dari simulasi yang akan dijalankan atau dilakukan pada aplikasi *Network Simulator* 2.35 merujuk pada perancangan topologi. Protokol *routing* yang mempunyai kinerja terbaik akan merujuk atau mereferensi pada perancangan topologi ini. Perancangan topologi variasi jumlah *node* merupakan perancangan yang akan digunakan pada penelitian ini.

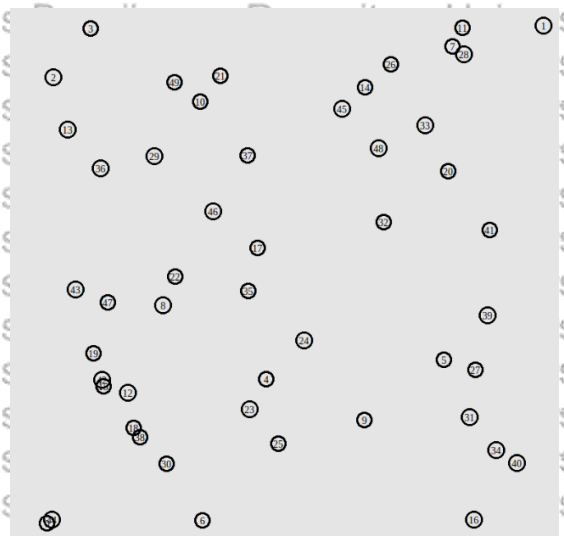
Kinerja dari setiap protokol *routing* yang terdiri dari AOMDV, DSR, dan AODV terhadap kepadatan jumlah *node* merujuk atau mereferensi pada perancangan topologi jumlah *node* ini. Perancangan topologi variasi jumlah *node* berguna agar penelitian yang dilakukan memfokuskan pada batasan-batasan *node* yang sudah ditentukan sebelumnya seperti berapa banyak *node* yang akan digunakan nantinya. Proses eksekusi *file* tcl pada NS-2.35 nantinya akan dipengaruhi oleh banyaknya atau jumlah *node* yang sudah ditentukan sebelumnya. Penentuan variasi jumlah *node* merujuk atau mereferensi pada proses eksekusi *file* .tcl yang telah dikonfigurasi nantinya. Variasi jumlah *node* ditetapkan sebanyak 20, 30, 50, 70, 90, dan 100 *node* dikarenakan menyesuaikan dengan kapasitas eksekusi pada konfigurasi *file* .tcl sehingga proses eksekusi tidak terhenti. Berikut merupakan contoh gambaran perancangan topologi yakni berjumlah 20, 30, 50, 70, 90, dan 100 *node* pada gambar 4.1, 4.2, 4.3, 4.4, 4.5, dan 4.6.



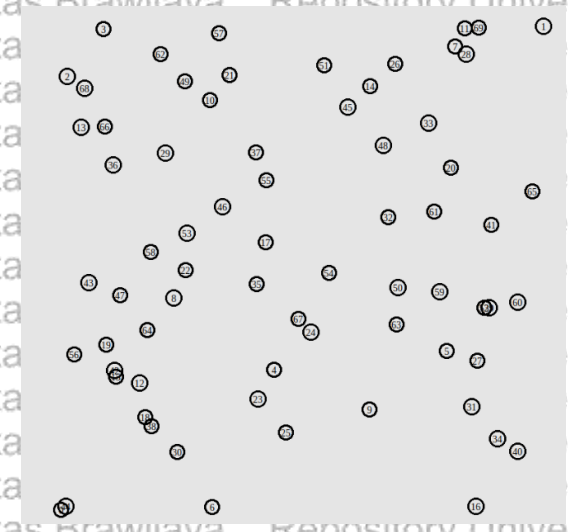
Gambar 4.1 Perancangan Topologi 20 *Node*



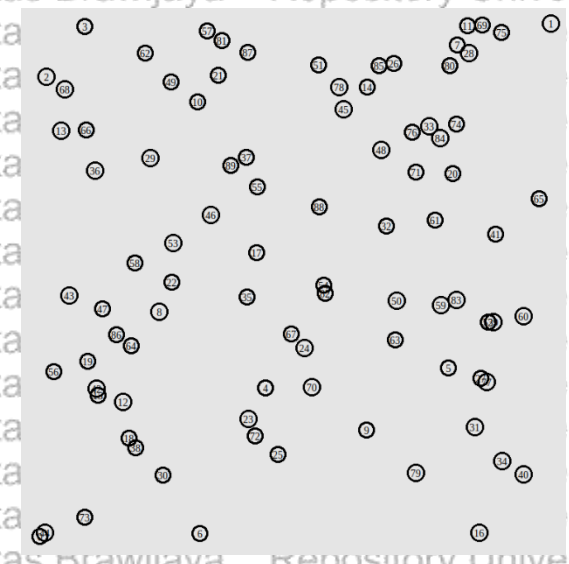
Gambar 4.2 Perancangan Topologi 30 *Node*



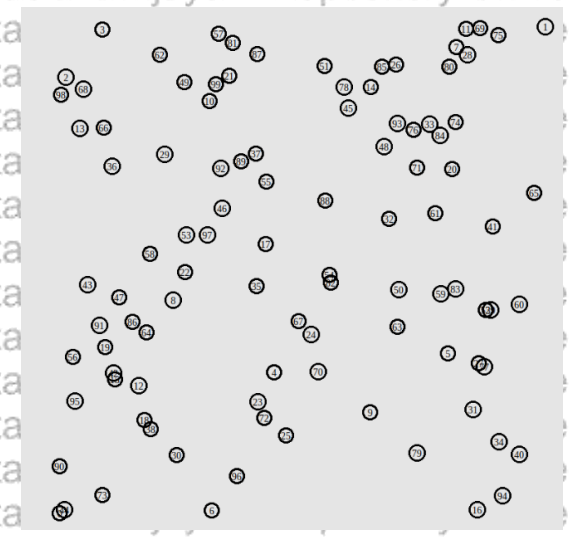
Gambar 4.3 Perancangan Topologi 50 *Node*



Gambar 4.4 Perancangan Topologi 70 *Node*



Gambar 4.5 Perancangan Topologi 90 *Node*



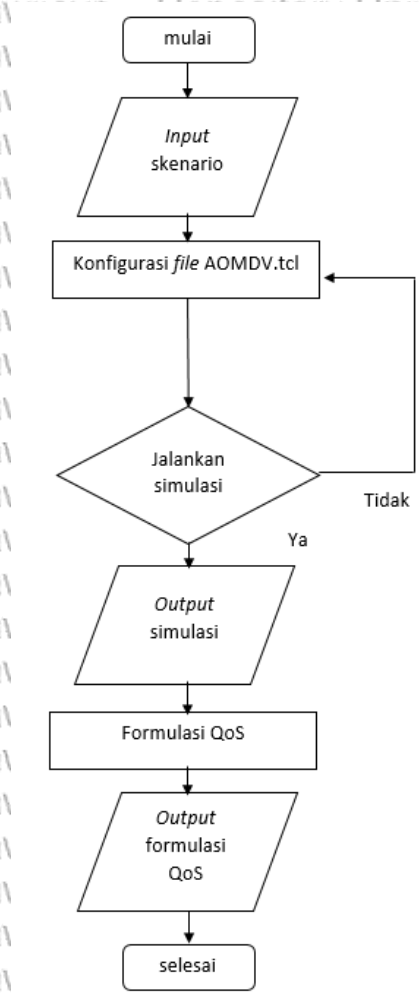
Gambar 4.6 Perancangan Topologi 100 *Node*

Penempatan *node* pada rancangan topologi variasi jumlah *node* ditunjukkan pada gambar 4.1, 4.2, 4.3, 4.4, 4.5, dan 4.6. Proses penempatan setiap *node* dilakukan secara acak dan tetap. Simpul sumber dan simpul tujuan ditempatkan menggunakan penempatan *node* tetap. Penempatan *node* tetap bertujuan atau berfungsi agar setiap *node* yang saling menjalin komunikasi tidak bergerak mendekat maupun menjauh dan berada pada posisi tetap didalam penelitian ini. *Pseudo random* digunakan untuk penempatan node dimana bertujuan agar saat terjadi penambahan jumlah *node,* pola penempatan setiap *node* tidak akan berubah. Simpul sumber berada pada koordinat x, y (100, 100), sedangkan simpul tujuan berada pada koordinat x, y (999, 999). Selanjutnya, penempatan *node* acak digunakan untuk menempatkan *node* yang bukan simpul sumber maupun simpul tujuan. Luas area yang telah ditentukan yakni 1000 m x 1000 m akan diisi oleh *node* acak yang tersebar secara *random*. *Random way point* merupakan pergerakan *node* selain simpul sumber maupun simpul tujuan dimana *node* tersebut bergerak bebas menuju titik koordinat tertentu. Kecepatan pergerakan dari *node* ini diatur secara *random* antara 0,5 m/s sampai 2 m/s.

Simulasi pada penelitian ini berdasarkan skenario variasi jumlah *node* dilakukan selama 1000 detik. *User Datagram Protocol* (UDP) merupakan koneksi yang digunakan pada penelitian ini dengan paket aplikasi *Constant Bit Rate* (CBR). Simulasi dimulai ketika paket CBR sebesar 512 *Bytes* dikirimkan melalui protokol UDP dari *node(0)* yang merupakan *node* sumber menuju *node(1)* yang merupakan *node* tujuan. *Interval* pengiriman paket CBR diatur sebesar 1 sehingga paket akan dikirimkan setiap 1 detik. Pengiriman paket CBR dimulai dari detik ke-0 sampai detik ke-1000.

4.2.2 Konfigurasi Protokol *Routing Ad-Hoc On Demand Multipath Distance Vector* (AOMDV)

Protokol *routing Ad-Hoc On Demand Multipath Distance Vector* (AOMDV) dikonfigurasi menggunakan bahasa pemrograman TCL dengan format file berekstensi file .tcl pada perangkat lunak atau *tools Network Simulator* versi 2.35. *Package* protokol *routing* AOMDV sudah terdapat didalam *package* NS-2.35 yang telah terinstall sebelumnya sehingga tidak perlu mengunduh *patch* terlebih dahulu. Berikut merupakan alur konfigurasi protokol *routing* AOMDV digambarkan dalam bentuk *flowchart*:



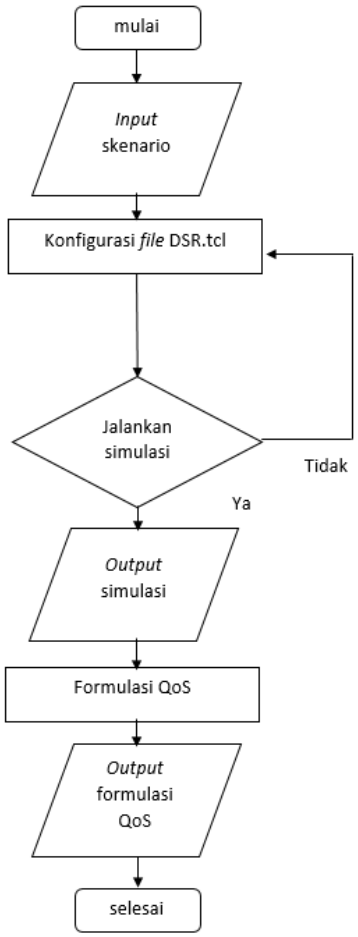
Gambar 4.7 Alur Konfigurasi Protokol *Routing* AOMDV Pada NS-2

Alur konfigurasi protokol *routing* AOMDV merujuk atau mereferensi pada gambar 4.7. Input skenario simulasi yang dipakai merupakan langkah pertama yang harus dilakukan. *Tool* simulator yang digunakan, protokol yang akan disimulasi, lama waktu uji, besar area simulasi serta jumlah simpul merupakan beberapa skenario yang dijadikan rujukan maupun referensi dalam melakukan penelitian. Selanjutnya ialah penerapan skenario yang ada pada *file* tcl berdasarkan konfigurasi pada *file* tcl yang telah dilakukan sebelumnya. Simulasi akan dijalankan dengan merujuk pada konfigurasi sebelumnya. Output simulasi akan diperoleh jika simulasi berhasil dijalankan, konfigurasi *file* tcl akan dilakukan jika simulasi gagal pada saat dijalankan. Selanjutnya, melakukan formulasi Quality of Service. Formulasi QoS dilakukan agar dapat diketahui hasil dari perhitungan *throughput* serta *end-to-end delay*.

4.2.3 Konfigurasi Protokol *Routing Dynamic Source Routing* (DSR)

Protokol *routing Dynamic Source Routing* (DSR) dikonfigurasi menggunakan bahasa pemrograman TCL dengan format file berekstensi file .tcl pada perangkat lunak atau *tools Network Simulator* versi 2.35. *Package* protokol *routing* DSR sudah terdapat didalam *package* NS-2.35 yang telah terinstall sebelumnya

sehingga tidak perlu mengunduh *patch* terlebih dahulu. Berikut merupakan alur konfigurasi protokol *routing* DSR digambarkan dalam bentuk *flowchart*:



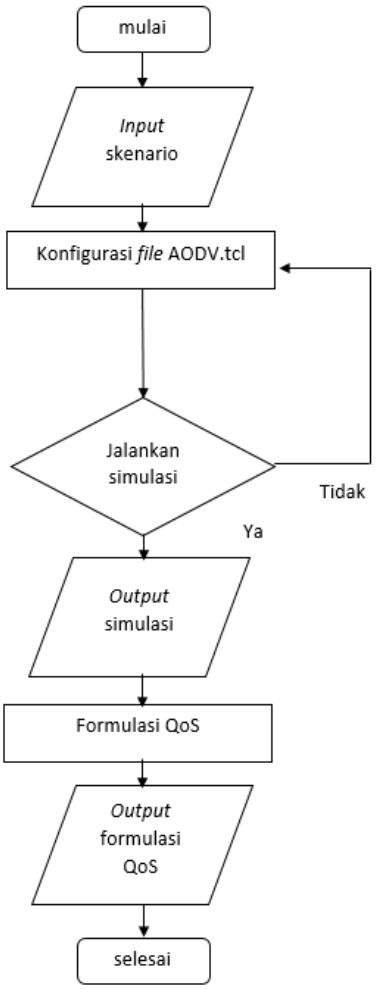
Gambar 4.8 Alur Konfigurasi Protokol *Routing* DSR Pada NS-2

Alur konfigurasi protokol *routing* DSR merujuk atau mereferensi pada gambar 4.8. Input skenario simulasi yang dipakai merupakan langkah pertama yang harus dilakukan. *Tool* simulator yang digunakan, protokol yang akan disimulasi, lama waktu uji, besar area simulasi serta jumlah simpul merupakan beberapa skenario yang dijadikan rujukan maupun referensi dalam melakukan penelitian. Selanjutnya ialah penerapan skenario yang ada pada *file* tcl berdasarkan konfigurasi pada *file* tcl yang telah dilakukan sebelumnya. Simulasi akan dijalankan dengan merujuk pada konfigurasi sebelumnya. Output simulasi akan diperoleh jika simulasi berhasil dijalankan.

4.2.4 Konfigurasi Protokol *Routing Ad-Hoc On Demand Distance Vector* (AODV)

Protokol *routing Ad-Hoc On Demand Distance Vector* (AODV) dirancang menggunakan bahasa pemrograman TCL dengan format file berekstensi file .tcl pada perangkat lunak atau *tools Network Simulator* versi 2.35. *Package* protokol *routing* AODV sudah terdapat didalam *package* NS-2.35 yang telah terinstall

sebelumnya sehingga tidak perlu mengunduh *package* terlebih dahulu. Berikut merupakan alur konfigurasi protokol *routing* AODV digambarkan dalam bentuk *flowchart*:



Gambar 4.9 Alur Konfigurasi Protokol *Routing* AODV Pada NS-2

Alur konfigurasi protokol *routing* AODV merujuk atau mereferensi pada gambar 4.9. Input skenario simulasi yang dipakai merupakan langkah pertama yang harus dilakukan. *Tool* simulator yang digunakan, protokol yang akan disimulasi, lama waktu uji, besar area simulasi serta jumlah simpul merupakan beberapa skenario yang dijadikan rujukan maupun referensi dalam melakukan penelitian. Selanjutnya ialah penerapan skenario yang ada pada *file* tcl berdasarkan konfigurasi pada *file* tcl yang telah dilakukan sebelumnya. Simulasi akan dijalankan dengan merujuk pada konfigurasi sebelumnya. Output simulasi akan diperoleh jika simulasi berhasil dijalankan, konfigurasi *file* tcl akan dilakukan jika simulasi gagal pada saat dijalankan. Selanjutnya, melakukan formulasi *Quality of Service*. Formulasi QoS dilakukan agar dapat diketahui hasil dari perhitungan *throughput* serta *end-to-end delay*.

4.3 Perancangan Pengujian

Rancangan pengujian merujuk atau mereferensi berdasarkan pengaruh dari skenario penelitian serta parameter uji *throughput*, serta *end to end delay*. Selanjutnya, dari hasil pengujian dan analisis yang dilakukan akan dibahas mengenai kesimpulan dan saran. Kesimpulan dan saran merujuk atau mereferensi pada pengujian dan analisis yang telah dilakukan sebelumnya yang membahas terkait kesimpulan apa saja yang didapat. Saran berguna untuk penelitian-penelitian selanjutnya di masa mendatang.

4.3.1 Perancangan Parameter Pengujian

Perancangan pengujian membahas terkait parameter-parameter uji yang akan dijadikan rujukan maupun referensi nantinya yang terdiri dari dua parameter yakni *throughput* serta *end to end delay*. Alasan pemilihan parameter *end to end delay* dan *throughput* dikarenakan terdapat korelasi antara mekanisme dari protokol reaktif yakni reaktif *flooding* dimana mengirim paket *route request* ke seluruh jaringan dengan kecepatan transfer serta lama waktu dalam pengiriman paket data antara *source node* sampai *destination node*. Selanjutnya, akan dilakukan analisis *Quality of Service* yang merujuk pada pengujian yang telah dilakukan sebelumnya berdasarkan parameter-parameter yang sudah ditetapkan. Tabel 4.4 merupakan informasi terkait parameter, penjelasan serta formulasi dari setiap parameter:

Tabel 4.4 Informasi Parameter Pengujian

No	Parameter	Formulasi	Keterangan
1	*Throughput*	$Throughput = \frac{jumlah\ paket\ data\ diterima}{jumlah\ waktu\ pengamatan}$	*Throughput* merupakan parameter uji yang merujuk pada kecepatan transfer data efektif dimana data tersebut dikalkulasi dalam *bit per second* atau bps sebagai total banyaknya paket yang diterima dibagi dengan waktu pengiriman. Perhitungan pada *throughput* ialah jumlah paket yang diterima dibagi

			dengan jumlah waktu pengamatan.
2	*End-to-End Delay*	$End\ to\ end\ Delay = \frac{jumlah\ total\ waktu\ paket\ diterima}{jumlah\ paket\ diterima}$	*End-to-end delay* merupakan parameter uji yang merujuk pada total waktu yang diambil yang berasal dari paket data yang tiba pada *destination*. Paket data yang dihitung ialah paket data yang berhasil dikirim ke *destination*. Perhitungan pada *end-to-end delay* ialah jumlah total waktu paket yang diterima dibagi dengan jumlah paket yang diterima.

4.3.2 Perancangan Skenario Pengujian Variasi Jumlah *Node*

Perancangan skenario pengujian membahas terkait rancangan skenario variasi jumlah *node*. Proses penempatan setiap *node* dilakukan secara acak dan tetap. Simpul sumber dan simpul tujuan ditempatkan menggunakan penempatan *node* tetap. Penempatan *node* tetap bertujuan atau berfungsi agar setiap *node* yang saling menjalin komunikasi tidak bergerak mendekat maupun menjauh dan berada pada posisi tetap didalam penelitian ini. *Pseudo random* digunakan untuk penempatan node dimana bertujuan agar saat terjadi penambahan jumlah *node,* pola penempatan setiap *node* tidak akan berubah. Simpul sumber berada pada koordinat x, y (100, 100), sedangkan simpul tujuan berada pada koordinat x, y (999, 999). Selanjutnya, penempatan *node* acak digunakan untuk menempatkan *node* yang bukan simpul sumber maupun simpul tujuan. Luas area yang telah ditentukan yakni 1000 m x 1000 m akan diisi oleh *node* acak yang tersebar secara *random*. *Random way point* merupakan pergerakan *node* selain simpul sumber maupun simpul tujuan dimana *node* tersebut bergerak bebas menuju titik koordinat tertentu. Kecepatan pergerakan dari *node* ini diatur secara *random* antara 0,5 m/s sampai 2 m/s.

Simulasi pada penelitian ini berdasarkan skenario variasi jumlah *node* dilakukan selama 1000 detik. *User Datagram Protocol* (UDP) merupakan koneksi yang digunakan pada penelitian ini dengan paket aplikasi *Constant Bit Rate* (CBR). Simulasi dimulai ketika paket CBR sebesar 512 *Bytes* dikirimkan melalui protokol UDP dari *node(0)* yang merupakan *node* sumber menuju *node(1)* yang merupakan *node* tujuan. *Interval* pengiriman paket CBR diatur sebesar 1 sehingga paket akan dikirimkan setiap 1 detik. Pengiriman paket CBR dimulai dari detik ke-0 sampai detik ke-1000. Konfigurasi yang digunakan untuk melakukan simulasi berdasarkan variasi jumlah *node* akan ditampilkan pada tabel 4.5.

Tabel 4.5 Konfigurasi Skenario Pengujian Variasi Jumlah *Node*

Parameter	Keterangan
Perangkat Lunak *Simulator*	*Network Simulator* versi 2.35 (NS-2)
Protokol *Routing*	• *Ad-Hoc On Demand Multipath Distance Vector* (AOMDV) • *Dynamic Source Routing* (DSR) • *Ad-Hoc On Demand Distance Vector* (AODV)
Jumlah *Node*	20, 30, 50, 70, 90, 100
Jenis Koneksi	*User Datagram Protocol* (UDP)
Jenis *Wireless*	Standar *Wireless* 802.11
Jenis Paket	*Constant Bit Rate* (CBR)
Ukuran Paket	512 *Bytes*
Interval Pengiriman	1 detik
Luas Area	1000 meter x 1000 meter
Waktu Simulasi	1000 detik
Tipe Mobilitas	*Random way point*
Kecepatan *Node*	0,5 sampai 2 m/s

4.4 Implementasi

Penerapan sistem sebagai lingkungan yang akan digunakan pada proses pengujian protokol akan dijelaskan pada bagian implementasi. Aplikasi *Network Simulator* 2.35 yang dijadikan sebagai simulator sangat diperlukan untuk penerapan nantinya serta untuk menunjang penelitian ini. Penjelasan terkait implementasi protokol *routing* serta implementasi NS-2.35 akan dijelaskan pada bagian implementasi.

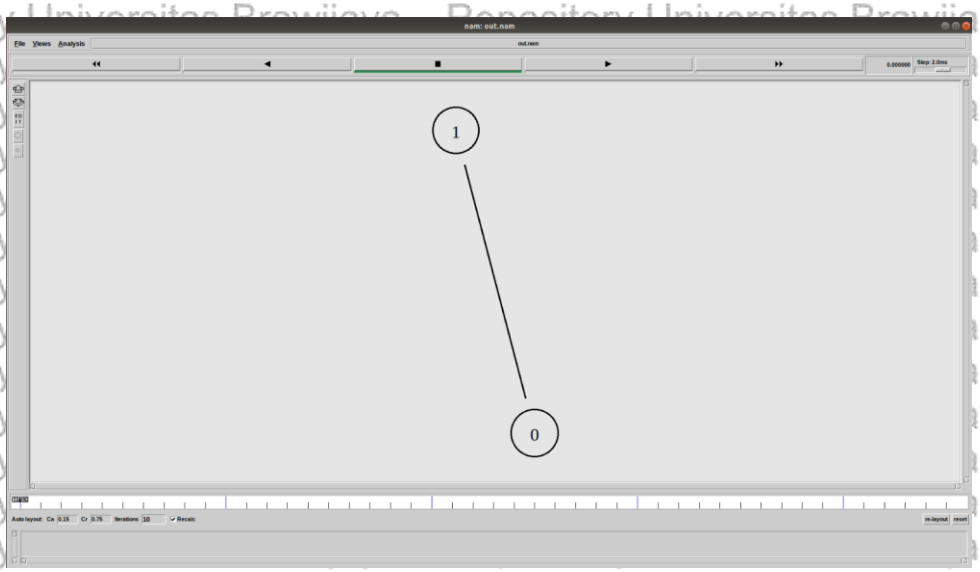
4.4.1 *Network Simulator* 2.35

Simulasi yang dijalankan dengan merujuk atau merefrensi pada sub-bab sebelumnya memerlukan instalasi *Network Simulator* 2.35. Diperlukan perangkat lunak tambahan yang mendukung kinerja dari aplikasi *Network Simulator* 2.35 dalam melakukan instalasi NS-2.35. Selanjutnya, proses instalasi akan dilakukan berdasarkan perintah yang telah dijalankan sebelumnya. Setelah proses instalasi NS-2.35 telah selesai dilakukan, maka selanjutnya akan menjalankan proses instalasi *dependency Network Simulator* 2.35. Selanjutnya, proses instalasi *dependency Network Simulator* 2.35 akan dilakukan berdasarkan perintah yang telah dijalankan sebelumnya. Setelah proses instalasi serta *dependency tools Network Simulator* telah selesai dilakukan, selanjutnya simulator dapat digunakan untuk menjalankan maupun mengimplementasikan simulasi jaringan. Tabel 4.6 merupakan perintah untuk menjalankan atau mengimplementasikan *file* dengan format .tcl.

Tabel 4.6 Perintah Untuk Implementasi Tampilan Awal NS-2

Script 1: Perintah first.tcl	
1	/home/arhangga# cd Desktop
2	/arhangga/Desktop# ns first.tcl

Selanjutnya, file berekstensi .tcl akan dijalankan dengan merujuk atau mereferensi pada perintah yang telah dituliskan sebelumnya. Setelah *file* tcl dijalankan maka akan dihasilkan file berekstensi ".tr" dan ".nam". Didalam *file* tr berisi terkait baris-baris rekam proses yang terjadi selama simulasi. Dalam melakukan analisis kinerja *routing protocol*, digunakan file tr tersebut. Tampilan animasi dari simulasi yang dijalankan dapat didapatkan dari *file* nam. Perintah nam ditambah dengan *file* ".nam" berfungsi untuk menjalankan file nam. Tampilan implementasi simulasi jaringan terdapat pada gambar 4.10.



Gambar 4.10 Tampilan Awal NS-2

4.4.2 Implementasi Topologi

Pada bagian implementasi topologi berisi terkait penerapan topologi. Implementasi topologi dilakukan menggunakan perangkat lunak *Network Simulator* versi 2.35 (NS-2.35). Implementasi topologi dimulai melalui konfigurasi pada *file* .tcl sesuai dengan jumlah variasi *node* yang telah ditentukan sebelumnya. *File* AODV.tcl digunakan untuk rujukan dalam implementasi topologi berjumlah 20, 30, 50, 70, 90, dan 100 *node*.

Tabel 4.7 Potongan Kode Sumber AODV.tcl

Script 2: Kode Sumber Topologi

```
1  # ======================================================================
2  # Define options
3  # ======================================================================
4  set opt(chan)   Channel/WirelessChannel  ;# channel type
5  set opt(prop)   Propagation/TwoRayGround ;# radio-propagation
6  model
7  set opt(netif)    Phy/WirelessPhy            ;# network interface
8  type
9  set opt(mac)      Mac/802_11                 ;# MAC type
10 set opt(ifq)      Queue/DropTail/PriQueue    ;# interface queue
11 type
12 set opt(ll)       LL                         ;# link layer type
13 set opt(ant)      Antenna/OmniAntenna        ;# antenna model
14 set opt(ifqlen)   50                         ;# max packet in
15 ifq
16 set opt(nn)       20                         ;# number of
17 mobilenodes
18 set opt(minSpeed) 0.5                        ;# movement minimum
19 speed [m/s]
20 set opt(maxSpeed) 2                          ;#          movement
21 maximum speed [m/s]
22 set opt(rp)       AODV                       ;# routing protocol
23 set opt(seed)     1                          ;# general pseudo-
24 random sequence generator
25 set opt(x)        1000                       ;# x coordinate of
26 topology
27 set opt(y)        1000                       ;# y coordinate of
28 topology
29 set opt(stop)     1000                       ;# time to stop
30 simulation
31 # ======================================================================
32 #Initialization
33 # ======================================================================
34 # create simulator instance
35 set ns_ [new Simulator]
36
37 # open traces
38 set tracefd  [open aodv0mal.tr w]
39 set namtrace [open aodv0mal.nam w]
40
41 $ns_ trace-all $tracefd
42 $ns_ namtrace-all-wireless $namtrace $opt(x) $opt(y)
43
44 # create topography object
45 set topo [new Topography]
46
```

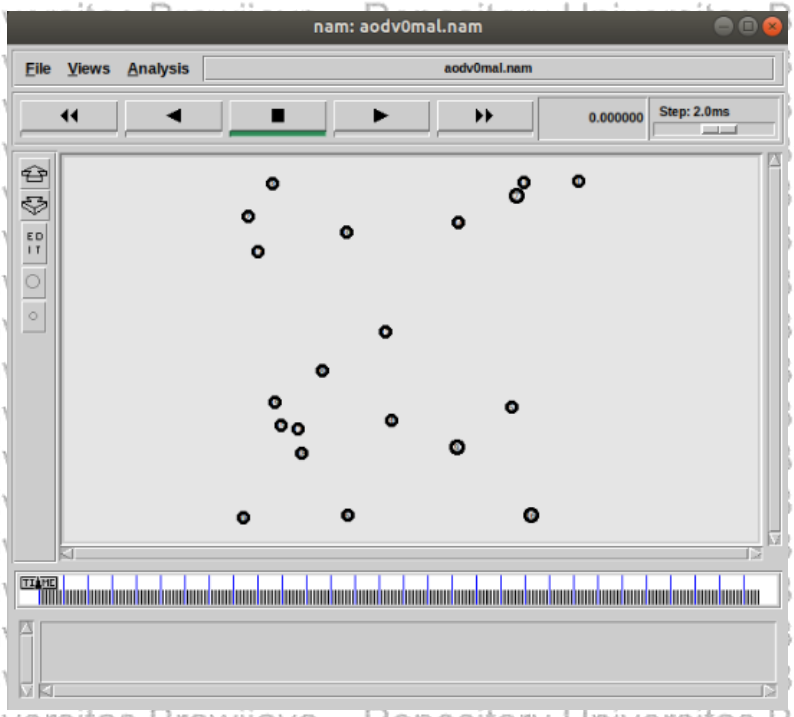
```
47
48  # define topology
49  $topo load_flatgrid $opt(x) $opt(y)
50
51  # create God
52  set god_ [create-god $opt(nn)]
53
54  # configure mobile nodes
55  $ns_ node-config -adhocRouting $opt(rp) \
56                   -llType $opt(ll) \
57                   -macType $opt(mac) \
58                   -ifqType $opt(ifq) \
59                   -ifqLen $opt(ifqlen) \
60                   -antType $opt(ant) \
61                   -propType $opt(prop) \
62                   -phyType $opt(netif) \
63                   -channelType $opt(chan) \
64                   -topoInstance $topo \
65                   -wiredRouting OFF \
66                   -agentTrace ON \
67                   -routerTrace ON \
68                   -macTrace OFF
```

Pada tabel 4.7, baris 4-30 mendeklarasikan terkait opsi setiap variabel yang berguna untuk rujukan konfigurasi nantinya. Pada baris 16, terdapat sebuah opsi untuk menentukan jumlah *mobile nodes* yang akan digunakan pada penelitian. Variasi jumlah *node* berjumlah 20, 30, 50, 70, 90, dan 100 akan digunakan opsi untuk rujukan implementasi topologi pada penelitian ini. Berikut merupakan perintah untuk menjalankan *file* AODV.tcl:

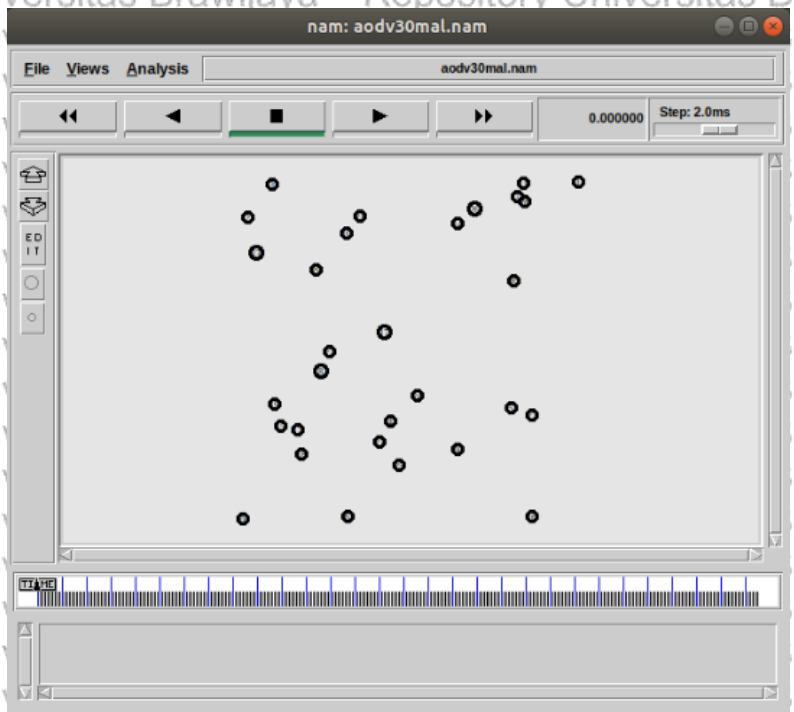
Tabel 4.8 Perintah Untuk Menjalankan *File* AODV.tcl

```
Script 3: Perintah Topologi
1  /arhangga/Desktop# cd protokol
2  /protokol/mobile# cd AODV
3  /mobile/AODV# ns AODV.tcl
```

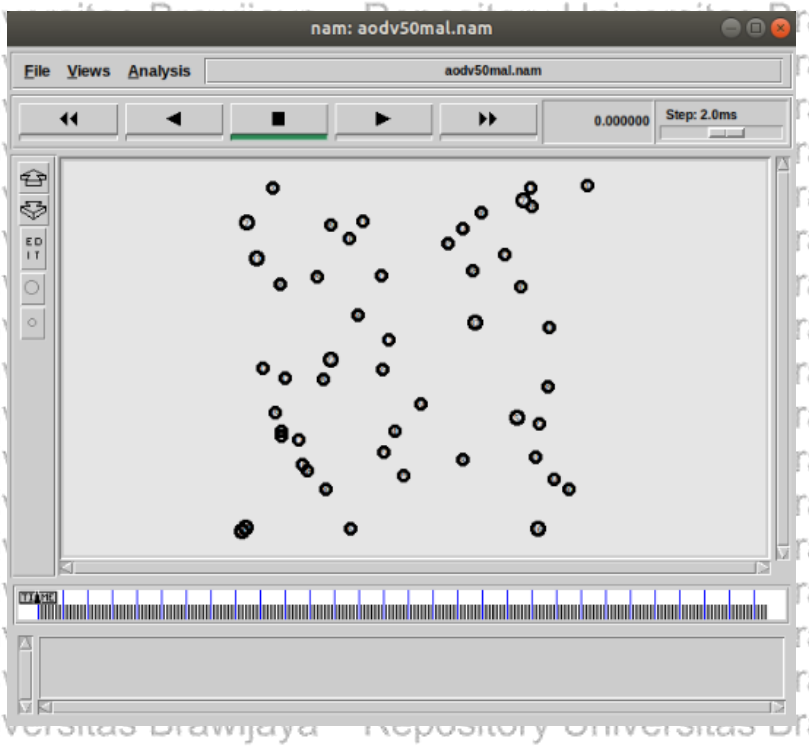
Tampilan *file* dengan ekstensi .nam yang berasal dari *output file* berkekstensi AODV.tcl berjumlah 3. *Output* pertama yakni menampilkan terkait konfigurasi variasi jumlah *node* berjumlah 20. Tampilan yang kedua berisi *output* variasi jumlah *node* berjumlah 30 dan tampilan yang ketiga berjumlah 50 node. *Output* keempat yakni menampilkan terkait konfigurasi variasi jumlah *node* berjumlah 70. Tampilan kelima dan keenam menampilkan terkait konfigurasi vasiasi jumlah *node* 90 dan 100. Berikut merupakan tampilan *file* .nam:



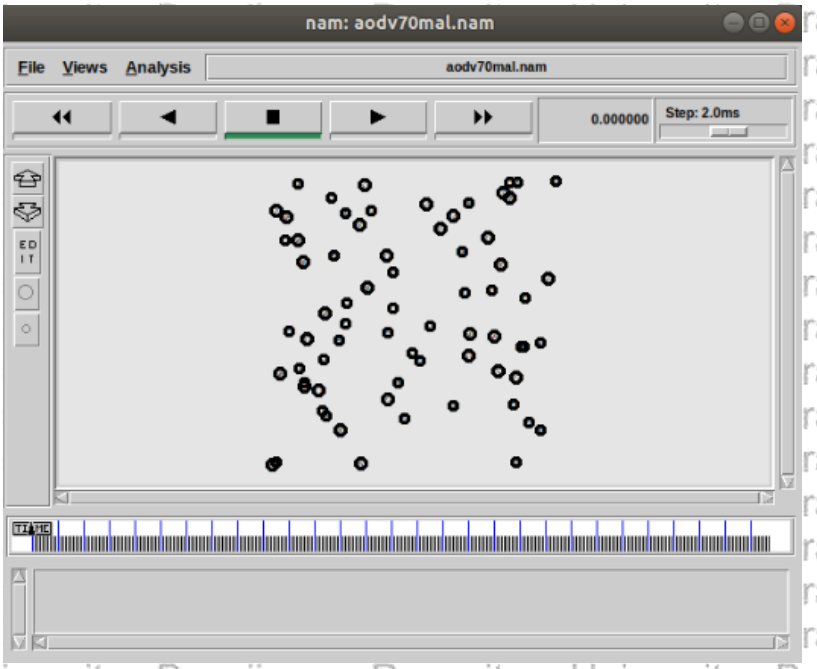
Gambar 4.11 Implementasi Topologi 20 *Node*



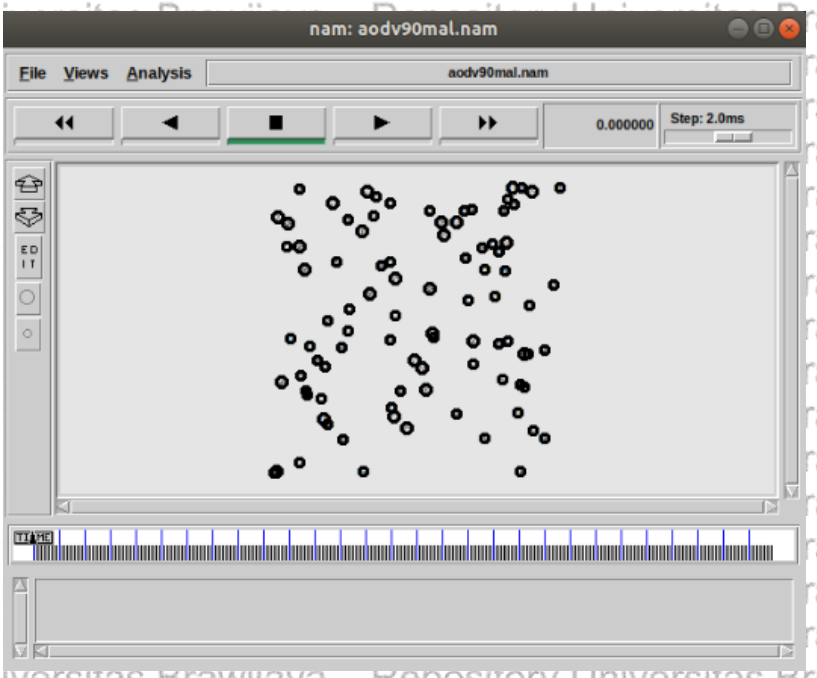
Gambar 4.12 Implementasi Topologi 30 *Node*



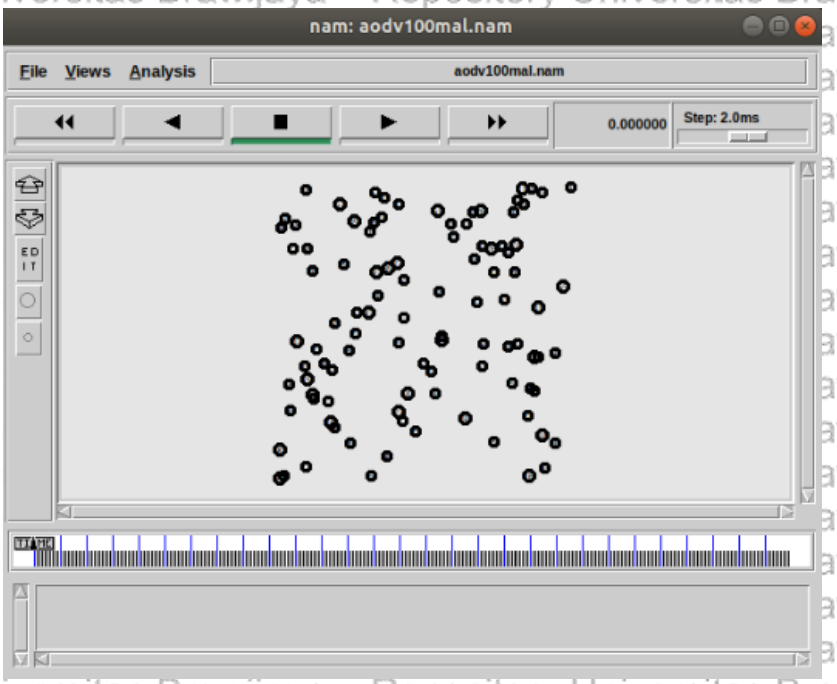
Gambar 4.13 Implementasi Topologi 50 *Node*



Gambar 4.14 Implementasi Topologi 70 *Node*



Gambar 4.15 Implementasi Topologi 90 *Node*



Gambar 4.16 Implementasi Topologi 100 *Node*

4.4.3 Implementasi Protokol *Ad-Hoc On Demand Multipath Distance Vector* (AOMDV)

Pada tahap implementasi sebuah *routing protocol* yakni protokol *Ad-Hoc On Demand Multipath Distance Vector* (AOMDV) menjelaskan terkait konfigurasi yang dilakukan pada file AOMDV.tcl. Konfigurasi yang dilakukan merujuk atau mereferensi pada jumlah *node*, metode pengiriman paket data, jenis paket serta berapa lama waktu simulasi. Tabel 4.9 merupakan potongan kode sumber protokol AOMDV.

Tabel 4.9 Potongan Kode Sumber Protokol AOMDV

Script 4: Kode Sumber AOMDV.tcl

```
# Define options
# ======================================================
set opt(chan)   Channel/WirelessChannel  ;# channel type
set opt(prop)   Propagation/TwoRayGround ;# radio-propagation model
set opt(netif)  Phy/WirelessPhy          ;# network interface type
set opt(mac)    Mac/802_11               ;# MAC type
set opt(ifq)    Queue/DropTail/PriQueue  ;# interface queue type
set opt(ll)     LL                       ;# link layer type
set opt(ant)    Antenna/OmniAntenna      ;# antenna model
set opt(ifqlen) 50                       ;# max packet in ifq
set opt(nn)     20                       ;# number of mobilenodes
set opt(minSpeed) 0.5                    ;# movement minimum speed [m/s]
set opt(maxSpeed) 2                      ;#       movement maximum speed [m/s]
set opt(rp)   AOMDV                      ;# routing protocol
set opt(seed) 1                          ;# general pseudo-random sequence generator
set opt(x)    1000                       ;# x coordinate of topology
set opt(y)    1000                       ;# y coordinate of topology
set opt(stop) 1000                       ;# time to stop simulation
# ======================================================
#Initialization
# ======================================================
    # create simulator instance
    set ns_  [new Simulator]

    # open traces
    set tracefd  [open aomdv0mal.tr w]
    set namtrace [open aomdv0mal.nam w]

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

    # create topography object
```

```
45  set topo [new Topography]
46
47  # define topology
48  $topo load_flatgrid $opt(x) $opt(y)
49
50  # create God
51  set god_ [create-god $opt(nn)]
52
53  # configure mobile nodes
54  $ns_ node-config -adhocRouting $opt(rp) \
55                   -llType $opt(ll) \
56                   -macType $opt(mac) \
57                   -ifqType $opt(ifq) \
58                   -ifqLen $opt(ifqlen) \
59                   -antType $opt(ant) \
60                   -propType $opt(prop) \
61                   -phyType $opt(netif) \
62                   -channelType $opt(chan) \
63                   -topoInstance $topo \
64                   -wiredRouting OFF \
65                   -agentTrace ON \
66                   -routerTrace ON \
67                   -macTrace OFF
68
69  ================================================================
70  # Nodes definition
71  ==========[repository.ub.ac.id]==========[universitas brawijaya]
72
73  for {set i 0} {$i < $opt(nn)} {incr i} {
74      set node_($i) [$ns_ node]
75  }
76
77  for {set i 0} {$i < $opt(nn)} {incr i} {
78      set X [expr [$randomNodeX value] ]
79      $node_($i) set X_ $X
80      set Y [expr [$randomNodeY value] ]
81      $node_($i) set Y_ $Y
82      $node_($i) set Z_ 0.0
83      $node_(0) set X_ 1
84      $node_(0) set Y_ 1
85      $node_(0) set Z_ 0.0
86      $node_(1) set X_ 999
87      $node_(1) set Y_ 999
88      $node_(1) set Z_ 0.0
89  }
90
```

Potongan *source code* diatas merujuk atau mereferensi pada file AOMDV.tcl yang berisi konfigurasi terkait skenario simulasi. Pada baris 4 terdapat opsi untuk menentukan tipe *wireless* apa yang akan digunakan dimana pada penelitian ini menggunakan jenis 802.11. Pada baris 5 terdapat sebuah opsi untuk menentukan jenis propagasi apa yang akan digunakan. Dalam penelitian ini, tipe propagasi yang digunakan ialah *TwoRayGround*. Pada baris 13 terdapat sebuah opsi untuk menentukan jenis antenna apa yang akan digunakan. *Omniantenna* merupakan tipe antenna yang digunakan pada penelitian ini. Pada baris 22 mendefinisikan protokol *routing* apa yang akan digunakan nantinya yaitu AOMDV. Pada baris 29

terdapat sebuah opsi untuk menentukan waktu simulasi nantinya yakni 1000 detik.

Tabel 4.10 Potongan Kode Sumber File CBR Pada Protokol AOMDV

Script 5: Kode Sumber CBR AOMDV.tcl

```
1  # ----------------------------------------------------------
2  # Data load
3  # ----------------------------------------------------------
4  set udp_(0) [new Agent/UDP]
5  $ns_ attach-agent $node_(0) $udp_(0)
6  set null_(0) [new Agent/Null]
7  $ns_ attach-agent $node_(1) $null_(0)
8  set cbr_(0) [new Application/Traffic/CBR]
9  $cbr_(0) set packetSize_ 512
10 #$cbr_(0) set rate_ 0.1mb
11 $cbr_(0) set interval_ 1
12 $cbr_(0) set random_ false
13 $cbr_(0) attach-agent $udp_(0)
14 $ns_ connect $udp_(0) $null_(0)
15 $ns_ at 0.0 "$cbr_(0) start"
```

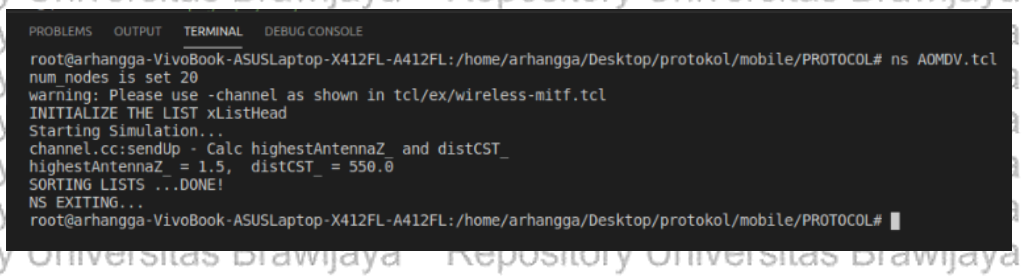
Tabel 4.10 merupakan potongan *source code* yang berisi terkait metode pengiriman data yang akan digunakan nantinya. Baris 4-15 berisi terkait deklarasi metode serta jenis paket data pada saat dikirimkan. Pada baris 4 terdapat sebuah opsi untuk menentukan metode pengiriman data, *User Datagram Protocol* (UDP) merupakan koneksi yang digunakan pada implementasi protokol *routing* AOMDV. Paket data yang akan berjalan pada koneksi *User Datagram Protocol* (UDP) ialah *Constant Bit Rate* (CBR). Pada baris 9 terdapat sebuah opsi untuk menentukan ukuran paket yakni menggunakan ukuran paket 512 *Bytes*. Perintah untuk menjalankan file AOMDV.tcl yang merupakan implementasi dari protokol *routing* AOMDV terdapat pada tabel 4.11.

Tabel 4.11 Perintah Untuk Menjalankan AOMDV.tcl

Script 6: Perintah AOMDV.tcl

```
1  mobile/PROTOCOL# ns AOMDV.tcl
```

File AOMDV.tcl akan dijalankan dengan merujuk atau mereferensi pada perintah yang telah dituliskan sebelumnya. Setelah *file* tcl dijalankan maka akan dihasilkan file berekstensi ".nam". Tampilan animasi dari simulasi yang dijalankan dapat didapatkan dari *file* nam. Perintah nam ditambah dengan *file* ".nam" berfungsi untuk menjalankan file nam. Selain *file* ".nam", terdapat *file* ".tr" yang berisi rekam proses apa saja yang terjadi saat simulasi dijalankan. Berikut merupakan tampilan *output* konfigurasi dan simulasi *file* dengan ekstensi ".nam" yang terdapat pada gambar 4.17 & 4.18:



Gambar 4.17 Tampilan Output Konfigurasi AOMDV

Gambar 4.18 Tampilan Output Simulasi AOMDV

4.4.4 Implementasi Protokol *Ad-Hoc On Demand Distance Vector* (AODV)

Pada tahap implementasi sebuah protokol *routing* yakni protokol *Ad-Hoc On Demand Distance Vector* (AODV) menjelaskan terkait konfigurasi yang dilakukan pada file AODV.tcl. Konfigurasi yang dilakukan merujuk atau mereferensi pada jumlah *node*, metode pengiriman paket data, jenis paket serta berapa lama waktu simulasi. Tabel 4.12 merupakan potongan kode sumber protokol AODV.

Tabel 4.12 Potongan Kode Sumber Protokol AODV

Script 7: Kode Sumber AODV.tcl

```
1  #======================================================
2  # Define options
3  #======================================================
4  set opt(chan)     Channel/WirelessChannel  ;# channel type
5  set opt(prop)     Propagation/TwoRayGround ;# radio-propagation
6  model
7  set opt(netif)    Phy/WirelessPhy          ;# network interface
8  type
9  set opt(mac)      Mac/802_11               ;# MAC type
10 set opt(ifq)      Queue/DropTail/PriQueue  ;# interface queue
11 type
12 set opt(ll)       LL                       ;# link layer type
13 set opt(ant)      Antenna/OmniAntenna      ;# antenna model
14 set opt(ifqlen) 50                         ;# max packet in ifq
15 set opt(nn)     20                      ;# number of mobilenodes
16 set opt(minSpeed)  0.5                    ;# movement minimum
17 speed [m/s]
18 set opt(maxSpeed)  2                      ;#          movement
19 maximum speed [m/s]
20 set opt(rp)          AODV                 ;# routing protocol
21 set opt(seed)          1                  ;# general
22 pseudo-random sequence generator
```

```
set opt(x)              1000                ;# x coordinate of
topology
set opt(y)              1000                ;# y coordinate of
topology
set opt(stop)           1000                ;# time to stop
simulation

# Initialization
#
# create simulator instance
set ns_ [new Simulator]

# open traces
set tracefd  [open aodv0mal.tr w]
set namtrace [open aodv0mal.nam w]

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

# create topography object
set topo [new Topography]

# define topology
$topo load_flatgrid $opt(x) $opt(y)

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

# configure mobile nodes
$ns_ node-config -adhocRouting $opt(rp) \
                 -llType $opt(ll) \
                 -macType $opt(mac) \
                 -ifqType $opt(ifq) \
                 -ifqLen $opt(ifqlen) \
                 -antType $opt(ant) \
                 -propType $opt(prop) \
                 -phyType $opt(netif) \
                 -channelType $opt(chan) \
                 -topoInstance $topo \
                 -wiredRouting OFF \
                 -agentTrace ON \
                 -routerTrace ON \
                 -macTrace OFF

#Nodesdefinition
#

for {set i 0} {$i < $opt(nn)} {incr i} {
    set node_($i) [$ns_ node]
}

for {set i 0} {$i < $opt(nn)} {incr i} {
set X [expr [$randomNodeX value] ]
$node_($i) set X_ $X
set Y [expr [$randomNodeY value] ]
$node_($i) set Y_ $Y
$node_($i) set Z_ 0.0
```

```
82  $node_(0) set X_ 1
83  $node_(0) set Y_ 1
84  $node_(0) set Z_ 0.0
85  $node_(1) set X_ 999
86  $node_(1) set Y_ 999
87  $node_(1) set Z_ 0.0
88  }
```

Potongan *source code* diatas merujuk atau mereferensi pada file AODV.tcl yang berisi konfigurasi terkait skenario simulasi. Pada baris 4 terdapat opsi untuk menentukan tipe *wireless* apa yang akan digunakan dimana pada penelitian ini menggunakan jenis 802.11. Pada baris 5 terdapat sebuah opsi untuk menentukan jenis propagasi apa yang akan digunakan. Dalam penelitian ini, tipe propagasi yang digunakan ialah *TwoRayGround*. Pada baris 13 terdapat sebuah opsi untuk menentukan jenis antenna apa yang akan digunakan. *Omniantenna* merupakan tipe antenna yang digunakan pada penelitian ini. Pada baris 20 mendefinisikan protokol *routing* apa yang akan digunakan nantinya yaitu AODV. Pada baris 27 terdapat sebuah opsi untuk menentukan waktu simulasi nantinya yakni 1000 detik.

Tabel 4.13 Potongan Kode Sumber File CBR Pada Protokol AODV

```
Script 8: Kode Sumber CBR AODV.tcl
1   #
2   # Data load
3   # ------------------------------------------------------------
4   set udp_(0) [new Agent/UDP]
5   $ns_ attach-agent $node_(0) $udp_(0)
6   set null_(0) [new Agent/Null]
7   $ns_ attach-agent $node_(1) $null_(0)
8   set cbr_(0) [new Application/Traffic/CBR]
9   $cbr_(0) set packetSize_ 512
10  #$cbr_(0) set rate_ 0.1mb
11  $cbr_(0) set interval_ 1
12  $cbr_(0) set random_ false
13  $cbr_(0) attach-agent $udp_(0)
14  $ns_ connect $udp_(0) $null_(0)
15  $ns_ at 0.0 "$cbr_(0) start"
```

Tabel 4.13 merupakan potongan *source code* yang berisi terkait metode pengiriman data yang akan digunakan nantinya. Baris 4-15 berisi terkait deklarasi metode serta jenis paket data pada saat dikirimkan. Pada baris 4 terdapat sebuah opsi untuk menentukan metode pengiriman data, *User Datagram Protocol* (UDP) merupakan koneksi yang digunakan pada implementasi protokol *routing* AODV. Paket data yang akan berjalan pada koneksi *User Datagram Protocol* (UDP) ialah *Constant Bit Rate* (CBR). Pada baris 9 terdapat sebuah opsi untuk menentukan ukuran paket yakni menggunakan ukuran paket 512 *Bytes*. Perintah untuk menjalankan file AODV.tcl yang merupakan implementasi dari protokol *routing* AODV terdapat pada tabel 4.14.

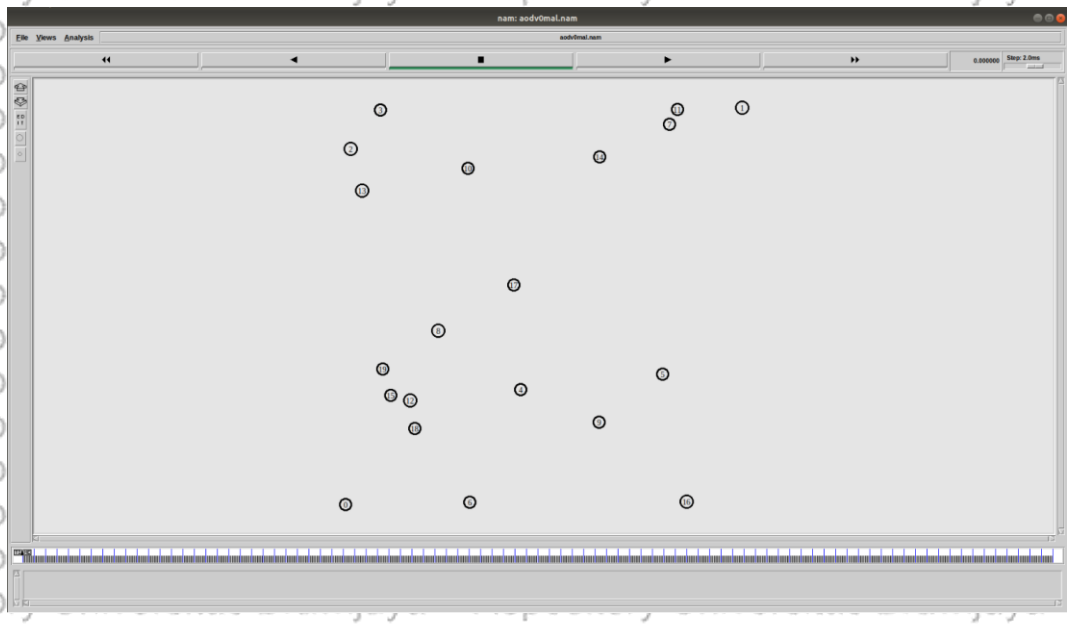
Tabel 4.14 Perintah Untuk Menjalankan AODV.tcl

```
Script 9: Perintah AODV.tcl
1   mobile/AODV# ns AODV.tcl
```

File AODV.tcl akan dijalankan dengan merujuk atau mereferensi pada perintah yang telah dituliskan sebelumnya. Setelah *file* tcl dijalankan maka akan dihasilkan file berekstensi ".nam". Tampilan animasi dari simulasi yang dijalankan dapat didapatkan dari *file* nam. Perintah nam ditambah dengan *file* ".nam" berfungsi untuk menjalankan file nam. Selain *file* ".nam", terdapat *file* ".tr" yang berisi rekam proses apa saja yang terjadi saat simulasi dijalankan. Berikut merupakan tampilan *output* konfigurasi & simulasi *file* dengan ekstensi ".nam" yang terdapat pada gambar 4.19 & 4.20:

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
root@arhangga-VivoBook-ASUSLaptop-X412FL-A412FL:/home/arhangga/Desktop/protokol/mobile/AODV# ns AODV.tcl
num_nodes is set 20
warning: Please use -channel as shown in tcl/ex/wireless-mitf.tcl
INITIALIZE THE LIST xListHead
Starting Simulation...
channel.cc:sendUp - Calc highestAntennaZ_ and distCST_
highestAntennaZ_ = 1.5,  distCST_ = 550.0
SORTING LISTS ...DONE!
NS EXITING...
root@arhangga-VivoBook-ASUSLaptop-X412FL-A412FL:/home/arhangga/Desktop/protokol/mobile/AODV#
```

Gambar 4.19 Tampilan Output Konfigurasi AODV



Gambar 4.20 Tampilan Output Simulasi AODV

4.4.5 Implementasi Protokol *Dynamic Source Routing* (DSR)

Pada tahap implementasi sebuah protokol *routing* yakni protokol *Dynamic Source Routing* (DSR) menjelaskan terkait konfigurasi yang dilakukan pada *file* DSR.tcl. Konfigurasi yang dilakukan merujuk atau mereferensi pada jumlah *node*, metode pengiriman paket data, jenis paket serta berapa lama waktu simulasi. Tabel 4.15 merupakan potongan kode sumber protokol DSR.

Tabel 4.15 Potongan Kode Sumber Protokol DSR

Script 10: Kode Sumber DSR.tcl

```
1  #===================================================================
2  # Define options
3  #===================================================================
4  set opt(chan)   Channel/WirelessChannel  ;# channel type
5  set opt(prop)   Propagation/TwoRayGround ;# radio-propagation
6  model
7  set opt(netif)  Phy/WirelessPhy          ;# network interface
8  type
9  set opt(mac)    Mac/802_11               ;# MAC type
10 set opt(ifq)    Queue/DropTail/PriQueue  ;# interface queue
11 type
12 set opt(ll)     LL                       ;# link layer type
13 set opt(ant)    Antenna/OmniAntenna      ;# antenna model
14 set opt(ifqlen) 50                       ;# max packet in ifq
15 set opt(nn)     20                   ;# number of mobilenodes
16 set opt(minSpeed) 0.5                  ;# movement minimum
17 speed [m/s]
18 set opt(maxSpeed) 2                    ;#          movement
19 maximum speed [m/s]
20 set opt(rp)       DSR                  ;# routing protocol
21 set opt(seed)     1                    ;# general pseudo-
22 random sequence generator
23 set opt(x)        1000                  ;# x coordinate of
24 topology
25 set opt(y)     1000                    ;# y coordinate of
26 topology
27 set opt(stop)  1000                    ;# time to stop
28 simulation
29
30 #===================================================================
31 # Initialization
32 #===================================================================
33 # create simulator instance
34 set ns_ [new Simulator]
35
36 # open traces
37 set tracefd  [open dsr0mal.tr w]
38 set namtrace [open dsr0mal.nam w]
39
40 $ns_ trace-all $tracefd
41 $ns_ namtrace-all-wireless $namtrace $opt(x) $opt(y)
42
43 # create topography object
44 set topo [new Topography]
45
46 # define topology
47 $topo load_flatgrid $opt(x) $opt(y)
48
49 # create God
50 set god_ [create-god $opt(nn)]
51
52 # configure mobile nodes
53 $ns_ node-config -adhocRouting $opt(rp) \
54                  -llType $opt(ll) \
55                  -macType $opt(mac) \
56                  -ifqType $opt(ifq) \
```

```
57                  -ifqLen $opt(ifqlen) \
58                  -antType $opt(ant) \
59                  -propType $opt(prop) \
60                  -phyType $opt(netif) \
61                  -channelType $opt(chan) \
62                  -topoInstance $topo \
63                  -wiredRouting OFF \
64                  -agentTrace ON \
65                  -routerTrace ON \
66                  -macTrace OFF
67
68  ================================================================
69  #Nodesdefinition
70  ================================================================
71
72  for {set i 0} {$i < $opt(nn)} {incr i} {
73      set node_($i) [$ns_ node]
74  }
75
76  for {set i 0} {$i < $opt(nn)} {incr i} {
77  set X [expr [$randomNodeX value] ]
78  $node_($i) set X_ $X
79  set Y [expr [$randomNodeY value] ]
80  $node_($i) set Y_ $Y
81  $node_($i) set Z_ 0.0
82  $node_(0) set X_ 1
83  $node_(0) set Y_ 1
84  $node_(0) set Z_ 0.0
85  $node_(1) set X_ 999
86  $node_(1) set Y_ 999
87  $node_(1) set Z_ 0.0
88  }
89
```

Potongan *source code* diatas merujuk atau mereferensi pada file DSR.tcl yang berisi konfigurasi terkait skenario simulasi. Pada baris 4 terdapat opsi untuk menentukan tipe *wireless* apa yang akan digunakan dimana pada penelitian ini menggunakan jenis 802.11. Pada baris 5 terdapat sebuah opsi untuk menentukan jenis propagasi apa yang akan digunakan. Dalam penelitian ini, tipe propagasi yang digunakan ialah *TwoRayGround*. Pada baris 13 terdapat sebuah opsi untuk menentukan jenis antenna apa yang akan digunakan. *Omniantenna* merupakan tipe antenna yang digunakan pada penelitian ini. Pada baris 20 mendefinisikan protokol *routing* apa yang akan digunakan nantinya yaitu DSR. Pada baris 28 terdapat sebuah opsi untuk menentukan waktu simulasi nantinya yakni 1000 detik.

Tabel 4.16 Potongan Kode Sumber File CBR Protokol DSR

Script 11: Kode Sumber CBR DSR.tcl

```
1  # ----------------------------------------------------------
2  # Data load
3  # ----------------------------------------------------------
4  set udp_(0) [new Agent/UDP]
5  $ns_ attach-agent $node_(0) $udp_(0)
6  set null_(0) [new Agent/Null]
7  $ns_ attach-agent $node_(1) $null_(0)
8  set cbr_(0) [new Application/Traffic/CBR]
9  $cbr_(0) set packetSize_ 512
10 #$cbr_(0) set rate_ 0.1mb
11 $cbr_(0) set interval_ 1
12 $cbr_(0) set random_ false
13 $cbr_(0) attach-agent $udp_(0)
14 $ns_ connect $udp_(0) $null_(0)
15 $ns_ at 0.0 "$cbr_(0) start"
```

Tabel 4.16 merupakan potongan *source code* yang berisi terkait metode pengiriman data yang akan digunakan nantinya. Baris 4-15 berisi terkait deklarasi metode serta jenis paket data pada saat dikirimkan. Pada baris 4 terdapat sebuah opsi untuk menentukan metode pengiriman data, *User Datagram Protocol* (UDP) merupakan koneksi yang digunakan pada implementasi protokol *routing* DSR. Paket data yang akan berjalan pada koneksi *User Datagram Protocol* (UDP) ialah *Constant Bit Rate* (CBR). Pada baris 9 terdapat sebuah opsi untuk menentukan ukuran paket yakni menggunakan ukuran paket 512 *Bytes*. Perintah untuk menjalankan file DSR.tcl yang merupakan implementasi dari protokol *routing* DSR terdapat pada tabel 4.17.

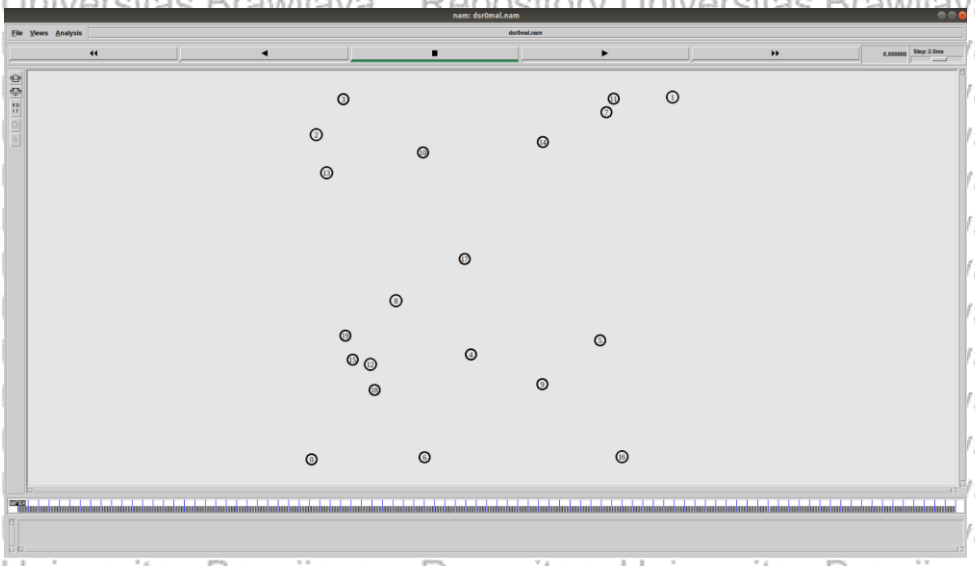
Tabel 4.17 Perintah Untuk Menjalankan DSR.tcl

Script 11: Perintah DSR.tcl

```
1  mobile/PROTOCOL# ns DSR.tcl
```

File DSR.tcl akan dijalankan dengan merujuk atau mereferensi pada perintah yang telah dituliskan sebelumnya. Setelah *file* tcl dijalankan maka akan dihasilkan file berekstensi ".nam". Tampilan animasi dari simulasi yang dijalankan dapat didapatkan dari *file* nam. Perintah nam ditambah dengan *file* ".nam" berfungsi untuk menjalankan file nam. Selain *file* ".nam", terdapat *file* ".tr" yang berisi rekam proses apa saja yang terjadi saat simulasi dijalankan. Berikut merupakan tampilan *output* konfigurasi & simulasi *file* dengan ekstensi ".nam" yang terdapat pada gambar 4.21 & 4.22:

Gambar 4.21 Tampilan Output Konfigurasi DSR



Gambar 4.22 Tampilan Output Simulasi DSR

BAB 5 HASIL DAN PEMBAHASAN

Pada bab hasil dan pembahasan akan membahas terkait hasil pengujian yang terdiri dari hasil pengujian terhadap perubahan jumlah *node*. Hasil pengujian variasi jumlah *node* mencakup parameter-parameter yang telah ditetapkan sebelumnya pada perancangan pengujian dimana terdiri dari *throughput* serta *end to end delay*. Selanjutnya, akan dibahas tentang analisis yang merujuk atau mereferensi pada hasil pengujian yang telah dilakukan sebelumnya. Perbandingan hasil analisis gabungan pengujian tiga *routing protocol* akan dibahas pada subbab terakhir.

5.1 Pengujian & Analisis *Throughput* Skenario Variasi Jumlah *Node*

Pengujian & analisis *throughput* menjelaskan terkait hasil pengujian yang telah dilakukan serta analisis yang merujuk pada hasil pengujian berdasarkan variasi jumlah *node*. Penerapan perubahan jumlah *node* merupakan skenario yang akan dijalankan pada simulasi. *Quality of Service* (QoS) yakni *throughput* dapat dijadikan rujukan maupun referensi parameter untuk tampilan hasil pengujian nantinya.

5.1.1 Tujuan

Skenario variasi jumlah *node* merupakan jumlah yang dapat dijadikan rujukan maupun referensi pada saat simulasi dijalankan berdasarkan konfigurasi yang dilakukan pada *file* .tcl sebelumnya. Protokol *routing* AOMDV, DSR, dan AODV masing-masing nantinya akan disimulasikan mulai dari jumlah *node* terkecil sampai dengan jumlah simpul terbesar. Skenario terhadap perubahan jumlah *node* bertujuan agar dapat diketahui kinerja dari setiap protokol *routing* terhadap kepadatan jumlah simpul.

5.1.2 Mekanisme

Mekanisme pengujian *throughput* yang dilakukan pada simulasi pertama dengan merujuk atau mereferensi berdasarkan variasi jumlah *node* adalah sebagai berikut:

1. Input skenario simulasi
2. Konfigurasi sesuai dengan simulasi yang akan dijalankan
3. Penerapan skenario pada *file* ".tcl" berdasarkan konfigurasi yang telah dilakukan sebelumnya
4. Ouput simulasi akan dihasilkan jika simulasi berhasil dijalankan
5. Analisis *Quality of Service* untuk rujukan kesimpulan

5.1.3 Hasil Pengujian

Throughput merupakan hasil dari pengujian yang merujuk atau mereferensi pada skenario variasi jumlah *node*. Penggambaran *throughput* ialah kecepatan *transfer* paket data antara simpul sumber dan simpul tujuan. Parameter

throughput merupakan parameter *Quality of Service* yang berfungsi untuk perhitungan kecepatan transfer data efektif. Perhitungan pada *throughput* ialah jumlah paket yang diterima dibagi dengan jumlah waktu pengamatan. Dari hasil pengujian yang dilakukan, telah didapatkan beberapa file *trace* yang dapat digunakan untuk rujukan dalam menghitung formulasi *throughput* nantinya. Gambar 5.1 merupakan potongan gambaran file ".tr" berdasarkan skenario variasi 20 *node* protokol AODV.

```
s 0.000000000 _0_ AGT  --- 0 cbr 512 [0 0 0 0] ------- [0:0 1:0 32 0] [0] 0 0
r 0.000000000 _0_ RTR  --- 0 cbr 512 [0 0 0 0] ------- [0:0 1:0 32 0] [0] 0 0
s 0.000000000 _0_ RTR  --- 0 AODV 48 [0 0 0 0] ------- [0:255 -1:255 30 0] [0x2 1 1 [1 0] [0 4]] (REQUEST)
s 1.000000000 _0_ AGT  --- 1 cbr 512 [0 0 0 0] ------- [0:0 1:0 32 0] [1] 0 0
r 1.000000000 _0_ RTR  --- 1 cbr 512 [0 0 0 0] ------- [0:0 1:0 32 0] [1] 0 0
s 2.000000000 _0_ AGT  --- 2 cbr 512 [0 0 0 0] ------- [0:0 1:0 32 0] [2] 0 0
r 2.000000000 _0_ RTR  --- 2 cbr 512 [0 0 0 0] ------- [0:0 1:0 32 0] [2] 0 0
s 2.000000000 _0_ RTR  --- 0 AODV 48 [0 0 0 0] ------- [0:255 -1:255 30 0] [0x2 1 2 [1 0] [0 6]] (REQUEST)
s 3.000000000 _0_ AGT  --- 3 cbr 512 [0 0 0 0] ------- [0:0 1:0 32 0] [3] 0 0
r 3.000000000 _0_ RTR  --- 3 cbr 512 [0 0 0 0] ------- [0:0 1:0 32 0] [3] 0 0
s 4.000000000 _0_ AGT  --- 4 cbr 512 [0 0 0 0] ------- [0:0 1:0 32 0] [4] 0 0
r 4.000000000 _0_ RTR  --- 4 cbr 512 [0 0 0 0] ------- [0:0 1:0 32 0] [4] 0 0
s 5.000000000 _0_ AGT  --- 5 cbr 512 [0 0 0 0] ------- [0:0 1:0 32 0] [5] 0 0
r 5.000000000 _0_ RTR  --- 5 cbr 512 [0 0 0 0] ------- [0:0 1:0 32 0] [5] 0 0
s 6.000000000 _0_ AGT  --- 6 cbr 512 [0 0 0 0] ------- [0:0 1:0 32 0] [6] 0 0
r 6.000000000 _0_ RTR  --- 6 cbr 512 [0 0 0 0] ------- [0:0 1:0 32 0] [6] 0 0
s 6.000000000 _0_ RTR  --- 0 AODV 48 [0 0 0 0] ------- [0:255 -1:255 30 0] [0x2 1 3 [1 0] [0 8]] (REQUEST)
```

Gambar 5.1 Contoh Potongan *File Trace* Untuk Formulasi *Throughput*

Gambar 5.1 merupakan contoh potongan *file* ".tr" yang diambil dari skenario variasi jumlah *node* yang berjumlah 20. Pada *file trace* terdapat beberapa kolom yang mendefinisikan sebuah nilai seperti waktu pengiriman paket, ukuran paket yang dikirimkan serta jenis paket yang dikirimkan. Pengujian yang dilakukan akan mengambil nilai dari kolom waktu dan ukuran paket untuk perhitungan *throughput*. Perintah *awk* akan digunakan untuk formulasi perhitungan *throughput* terdapat pada tabel 5.1.

Tabel 5.1 Perintah AWK Untuk Menghitung Formulasi *Throughput*

Script 1 : Perintah AWK Throughput	
1	`$ awk '{if(($1=="r"))print $2,$8}' aodv0mal.tr >> outa.txt`
2	`$ awk -f throughput2.awk outa.txt >> File1`
3	`$ awk '{print $2}' File1 >> throughput20nodeaodv`

Tabel 5.1 merupakan gambaran perintah *awk* yang diambil dari skenario variasi jumlah *node* berjumlah 20. Perintah *awk* sendiri merupakan *command* yang sudah secara otomatis terdapat pada sistem operasi *linux* yang berfungsi untuk mengonversi *file*. *File* yang dikonversi dapat berasal dari format format tertentu seperti yang terlihat pada gambar. Perintah pada baris pertama berfungsi untuk mengonversi file *trace* menjadi format *text*. Perintah pada baris kedua digunakan untuk mengonversi file dengan format ".txt" menjadi format hasil formulasi *throughput*. Perintah pada baris terakhir berfungsi untuk mengonversi hasil formulasi *throughput* menjadi *file* yang hanya berisi kolom 2 yang diambil dari "File1". Berikut merupakan tampilan hasil formulasi *throughput* terdapat pada gambar 5.2.

```
arhangga@arhangga-VivoBook-ASUSLaptop-X412FL-A412FL:~/Desktop/protokol$ cat throughput
20      1504.61      4325.77      766.09
30      3824.23      10345.33     1222.67
50      9803.71      27311.95     149800.94
70      13524.37     49303.96     284355.46
90      17273.85     77196.19     484009.15
100     28517.71     91508.05     371485.88
```

Gambar 5.2 Hasil Formulasi *Throughput* Variasi Jumlah *Node*

Selanjutnya, hasil formulasi *throughput* dari setiap skenario perubahan *node* merujuk atau mereferensi pada protokol *routing* AOMDV, DSR, dan AODV telah didapatkan. Hasil formulasi *throughput* akan dirata-rata sehingga akan didapatkan hasil berupa protokol *routing* yang memiliki *throughput* terbaik. Hasil akhir tersebut didasarkan pada skenario perubahan atau variasi jumlah *node* yaitu 20, 30, 50, 70, 90, dan 100. Berikut merupakan informasi terkait nilai *throughput* berdasarkan pengujian variasi jumlah *node* yang terdapat pada tabel 5.2.

Tabel 5.2 Hasil Pengujian *Throughput* Variasi Jumlah *Node*

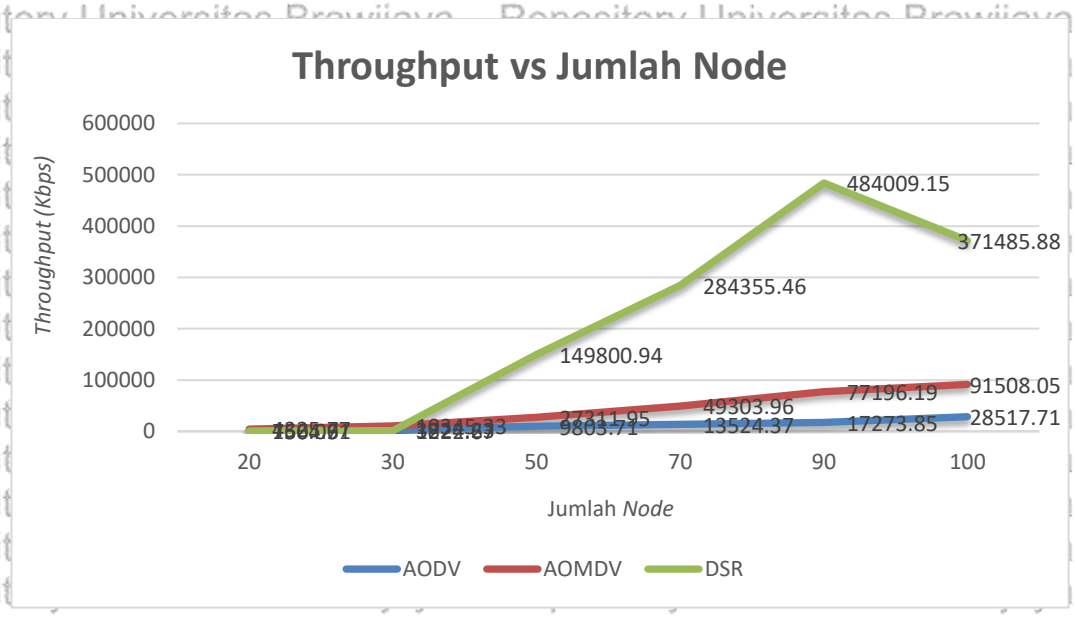
Jumlah *Node*	*Throughput* (Kbps)		
	AODV	AOMDV	DSR
20	1504,61	4325,77	766,09
30	3824,23	10345,33	1222,67
50	9803,71	27311,95	149800,94
70	13524,37	49303,96	284355,46
90	17273,85	77196,19	484009,15
100	28517,71	91508,05	371485,88
Rata-rata	12408,08	43331.87	215273,36

Dari tabel 5.2, nilai *throughput* terhadap penambahan jumlah simpul dapat diketahui. Protokol DSR memiliki nilai tertinggi dibandingkan dengan protokol yang lainnya, sedangkan protokol AODV memiliki nilai terendah diantara protokol *routing* yang lainnya. Diketahui hasil dari *throughput* AODV adalah 1504,61 Kbps, AOMDV memiliki nilai 4325,77 Kbps sedangkan DSR mempunyai nilai 766,09 Kbps pada pengujian dengan skenario 20 simpul. Diketahui hasil dari *throughput* AODV adalah 3824,23 Kbps, AOMDV memiliki nilai 10345,33 Kbps sedangkan DSR mempunyai nilai 1222,67 Kbps pada pengujian dengan skenario 30 simpul. Diketahui hasil dari *throughput* AODV adalah 9803,71 Kbps, AOMDV memiliki nilai 27311,95 Kbps sedangkan DSR mempunyai nilai 149800,94 Kbps pada pengujian dengan skenario 50 *node*. Diketahui hasil dari *throughput* AODV adalah 13524,37 Kbps, AOMDV memiliki nilai 49303,96 Kbps sedangkan DSR mempunyai nilai 284355,46 Kbps pada pengujian dengan skenario 70 *node*. Diketahui hasil dari *throughput* AODV adalah 17273,85 Kbps, AOMDV memiliki nilai 77196,19 Kbps sedangkan DSR mempunyai nilai 484009,15 Kbps pada pengujian dengan skenario 90 *node*. Diketahui hasil dari *throughput* AODV adalah 28517,71 Kbps, AOMDV

memiliki nilai 91508,05 Kbps sedangkan DSR mempunyai nilai 371485,88 Kbps pada pengujian dengan skenario 100 *node*.

5.1.4 Analisis

Parameter QoS *throughput* merupakan parameter pengujian yang dijadikan rujukan maupun referensi dalam membandingan tiga protokol *routing* AOMDV, DSR, dan AODV pada skenario perubahan jumlah *node*. Perubahan jumlah *node* yang berjumlah 20, 30, 50, 70, 90, dan 100 merupakan rincian *node* dalam melakukan analisis hasil pengujian. Tampilan grafik akan digunakan untuk mempermudah dalam analisis hasil pengujian parameter *throughput* pada skenario perubahan jumlah *node*. Gambar 5.3 merupakan tampilan grafik nilai *throughput* terhadap skenario perubahan jumlah *node*.



Gambar 5.3 Grafik Nilai *Throughput* Terhadap Variasi Jumlah *Node*

Hasil perbandingan *throughput* antara protokol *routing* AOMDV, DSR, dan AODV berdasarkan skenario jumlah simpul yang bervariasi ditampilkan pada gambar 5.3 terkait grafik hasil pengujian. Perhitungan pada *throughput* ialah jumlah paket yang diterima dibagi dengan jumlah waktu pengamatan dimana nantinya dari perhitungan tersebut akan menghasilkan nilai *throughput*.

Pada protokol AODV, merujuk atau mereferensi pada grafik nilai *throughput* cenderung mengalami kenaikan. Pengujian skenario variasi *node* berjumlah 20 dengan nilai *throughput* 1504,61 Kbps terlihat mengalami kenaikan pertama pada grafik yaitu nilai *throughput* menjadi 3824,23 Kpbs pada skenario variasi node dengan jumlah 30. Pada pengujian skenario variasi *node* dengan jumlah 50 mengalami kenaikan kedua pada grafik dengan nilai *throughput* menjadi 9803,71 Kbps. Kenaikan ketiga terdapat pada variasi jumlah *node* berjumlah 70 yakni menjadi 13524,37 Kbps. Pada pengujian skenario variasi jumlah 90 dan 100 mengalam kenaikan keempat dan kelima secara berturut turut yakni menjadi

17273,85 Kbps dan 28517,71 Kbps. Total dari keseluruhan yang terlihat pada grafik, protokol AODV memiliki nilai *throughput* yang cenderung naik.

Pada protokol AOMDV, merujuk atau mereferensi pada grafik nilai *throughput* cenderung mengalami kenaikan. Pengujian skenario variasi *node* berjumlah 20 dengan nilai *throughput* 4325,77 Kbps terlihat mengalami kenaikan pertama pada grafik yaitu nilai *throughput* menjadi 10345,33 Kpbs pada skenario variasi node dengan jumlah 30. Pada pengujian skenario variasi *node* dengan jumlah 50 mengalami kenaikan kedua pada grafik dengan nilai *throughput* menjadi 27311,95 Kbps. Kenaikan ketiga terdapat pada variasi jumlah *node* berjumlah 70 yakni menjadi 49303,96 Kbps. Pada pengujian skenario variasi jumlah 90 dan 100 mengalami kenaikan keempat dan kelima secara berturut turut yakni menjadi 77196,19 Kbps dan 91508,05 Kbps. Total dari keseluruhan yang terlihat pada grafik, protokol AOMDV memiliki nilai *throughput* yang cenderung naik.

Pada protokol DSR, merujuk atau mereferensi pada grafik nilai *throughput* cenderung mengalami kenaikan yang cukup signifikan. Pengujian skenario variasi *node* berjumlah 20 dengan nilai *throughput* 766,09 Kbps terlihat mengalami kenaikan pada grafik yaitu nilai *throughput* menjadi 1222,67 Kpbs pada skenario variasi node dengan jumlah 30. Pada pengujian skenario variasi *node* dengan jumlah 50 mengalami kenaikan yang cukup signifikan pada grafik dengan nilai *throughput* menjadi 149800,94 Kbps. Kenaikan ketiga terdapat pada variasi jumlah *node* berjumlah 70 yakni menjadi 284355,46 Kbps. Pada pengujian skenario variasi jumlah 90 mengalami kenaikan keempat menjadi 484009,15 Kbps. Penurunan grafik terjadi pada skenario variasi *node* 100 yakni menjadi 371485,88 Kbps. Total dari keseluruhan yang terlihat pada grafik, protokol DSR memiliki nilai *throughput* yang cenderung naik.

Berdasarkan hasil analisis yang telah ditetapkan sebelumnya, protokol *routing* AODV, DSR, dan AOMDV cenderung mengalami kenaikan pada grafik. Kenaikan grafik ketiga protokol *routing* disebabkan oleh mekanisme *route maintenance* pada ketiga protokol *routing* dapat bekerja saat terjadi kegagalan rute sehingga rute alternatif dapat ditemukan. Rute alternatif dapat menyebabkan kecepatan transfer data semakin tinggi saat proses pengiriman paket data antara *source node* dan *destination node* seiring dengan bertambahnya jumlah *node*.

Protokol *routing* AOMDV memiliki nilai *throughput* yang lebih baik dibandingkan dengan protokol *routing* AODV seiring dengan bertambahnya jumlah kepadatan *node*. Protokol *routing* AOMDV memiliki nilai *throughput* yang lebih baik dibandingkan dengan protokol *routing* AODV dikarenakan mekanisme kerja dari protokol AOMDV yakni *multipath* atau menggunakan lebih dari dari satu jalur dalam pemilihan jalur routing mana yang terbaik dalam proses pengiriman paket data antara *source node* menuju *destination node*.

Protokol *routing* DSR memiliki nilai *throughput* yang cenderung naik seiring dengan bertambahnya jumlah kepadatan *node*. Protokol *routing* DSR memiliki kenaikan nilai *throughput* yang cukup signifikan pada saat kepadatan jumlah *node* berjumlah 90. Berdasarkan *throughput*, protokol *routing* DSR memiliki nilai

throughput terbaik dibandingkan dengan protokol *routing* yang lainnya. Protokol *routing* DSR memiliki nilai *throughput* terbaik dikarenakan *route cache* pada protokol DSR memiliki fungsi yakni dapat menemukan banyak jalur dalam satu pencarian sehingga dapat digunakan untuk *alternative route*. *Alternative route* berfungsi untuk rute alternatif yang dapat digunakan untuk pengiriman paket data jika terjadi kegagalan rute.

5.2 Pengujian & Analisis *End-to-end Delay* Skenario Variasi Jumlah *Node*

Pengujian & analisis *end to end delay* menjelaskan terkait hasil pengujian yang telah dilakukan serta analisis yang merujuk pada hasil pengujian berdasarkan variasi jumlah *node*. Penerapan variasi jumlah simpul merupakan skenario yang akan dijalankan pada simulasi. *Quality of Service* (QoS) yakni *end to end delay* dapat dijadikan rujukan maupun referensi parameter untuk tampilan hasil pengujian nantinya.

5.2.1 Tujuan

Skenario variasi jumlah *node* merupakan jumlah yang dapat dijadikan rujukan maupun referensi pada saat simulasi dijalankan berdasarkan konfigurasi yang dilakukan pada *file* .tcl sebelumnya. Protokol *routing* AOMDV, DSR, dan AODV masing-masing nantinya akan disimulasikan mulai dari jumlah *node* terkecil sampai dengan jumlah simpul terbesar. Skenario terhadap perubahan jumlah *node* bertujuan agar dapat diketahui kinerja dari setiap protokol *routing* terhadap kepadatan jumlah simpul.

5.2.2 Mekanisme

Mekanisme pengujian *end to end delay* yang dilakukan pada simulasi pertama dengan merujuk atau mereferensi berdasarkan variasi jumlah *node* adalah sebagai berikut:

1. Input skenario simulasi
2. Konfigurasi sesuai dengan simulasi yang akan dijalankan
3. Penerapan skenario pada *file* ".tcl" berdasarkan konfigurasi yang telah dilakukan sebelumnya
4. Ouput simulasi akan dihasilkan jika simulasi berhasil dijalankan
5. Analisis *Quality of Service* untuk rujukan kesimpulan

5.2.3 Hasil Pengujian

End to end delay merupakan hasil dari pengujian yang merujuk atau mereferensi pada skenario perubahan jumlah *node*. Penggambaran *end to end delay* ialah sebuah simpul sumber untuk mengirim paket data kepada simpul tujuan terkait berapa lama waktu yang dibutuhkan. Perhitungan pada *end to end delay* ialah jumlah total waktu paket yang diterima dibagi dengan jumlah paket

yang diterima. Dari hasil pengujian yang dilakukan, telah didapatkan beberapa file *trace* yang dapat digunakan untuk rujukan dalam menghitung formulasi *end to end delay* nantinya. Gambar 5.4 merupakan potongan gambaran file ".tr" berdasarkan skenario variasi 20 *node* protokol AODV.

```
s 0.000000000 _0_ AGT  --- 0 cbr 512 [0 0 0 0] ------- [0:0 1:0 32 0] [0] 0 0
r 0.000000000 _0_ RTR  --- 0 cbr 512 [0 0 0 0] ------- [0:0 1:0 32 0] [0] 0 0
s 0.000000000 _0_ RTR  --- 0 AODV 48 [0 0 0 0] ------- [0:255 -1:255 30 0] [0x2 1 1 [1 0] [0 4]] (REQUEST)
s 1.000000000 _0_ AGT  --- 1 cbr 512 [0 0 0 0] ------- [0:0 1:0 32 0] [1] 0 0
r 1.000000000 _0_ RTR  --- 1 cbr 512 [0 0 0 0] ------- [0:0 1:0 32 0] [1] 0 0
s 2.000000000 _0_ AGT  --- 2 cbr 512 [0 0 0 0] ------- [0:0 1:0 32 0] [2] 0 0
r 2.000000000 _0_ RTR  --- 2 cbr 512 [0 0 0 0] ------- [0:0 1:0 32 0] [2] 0 0
s 2.000000000 _0_ RTR  --- 0 AODV 48 [0 0 0 0] ------- [0:255 -1:255 30 0] [0x2 1 2 [1 0] [0 6]] (REQUEST)
s 3.000000000 _0_ AGT  --- 3 cbr 512 [0 0 0 0] ------- [0:0 1:0 32 0] [3] 0 0
r 3.000000000 _0_ RTR  --- 3 cbr 512 [0 0 0 0] ------- [0:0 1:0 32 0] [3] 0 0
s 4.000000000 _0_ AGT  --- 4 cbr 512 [0 0 0 0] ------- [0:0 1:0 32 0] [4] 0 0
r 4.000000000 _0_ RTR  --- 4 cbr 512 [0 0 0 0] ------- [0:0 1:0 32 0] [4] 0 0
s 5.000000000 _0_ AGT  --- 5 cbr 512 [0 0 0 0] ------- [0:0 1:0 32 0] [5] 0 0
r 5.000000000 _0_ RTR  --- 5 cbr 512 [0 0 0 0] ------- [0:0 1:0 32 0] [5] 0 0
s 6.000000000 _0_ AGT  --- 6 cbr 512 [0 0 0 0] ------- [0:0 1:0 32 0] [6] 0 0
r 6.000000000 _0_ RTR  --- 6 cbr 512 [0 0 0 0] ------- [0:0 1:0 32 0] [6] 0 0
s 6.000000000 _0_ RTR  --- 0 AODV 48 [0 0 0 0] ------- [0:255 -1:255 30 0] [0x2 1 3 [1 0] [0 8]] (REQUEST)
```

Gambar 5.4 Contoh Potongan *File Trace* Untuk Formulasi *End-to-end Delay*

Gambar 5.4 merupakan contoh potongan *file* ".tr" yang diambil dari skenario variasi jumlah *node* yang berjumlah 20. Pada *file trace* terdapat beberapa kolom yang mendefinisikan sebuah nilai seperti waktu pengiriman paket, ukuran paket yang dikirimkan serta ukuran bandwith yang dikirimkan. Pengujian yang dilakukan akan mengambil nilai dari kolom waktu dan ukuran paket untuk perhitungan *end to end delay*. Perintah *awk* akan digunakan untuk formulasi perhitungan *end to end delay* terdapat pada tabel 5.3.

Tabel 5.3 Perintah AWK Untuk Menghitung Formulasi *End-to-end Delay*

Script 2 : Perintah AWK End To End Delay	
1	$ awk '{if(($1=="r"))print $2,$8}' aodv0mal.tr >> outa.txt
2	$ awk -f delay2.awk outa.txt >> File1
3	$ awk '{print $2}' File1 >> delay20nodeaodv

Tabel 5.3 merupakan gambaran perintah *awk* yang diambil dari skenario perubahan jumlah *node* berjumlah 20. Perintah pada baris pertama berfungsi untuk mengonversi file *trace* menjadi format *text*. Perintah pada baris kedua digunakan untuk mengonversi file dengan format ".txt" menjadi format hasil formulasi *delay*. Perintah pada baris terakhir berfungsi untuk mengonversi hasil formulasi *delay* menjadi *file* yang hanya berisi kolom 2 yang diambil dari "File1". Berikut merupakan tampilan hasil formulasi *end to end delay* terdapat pada gambar 5.5.

```
arhangga@arhangga-VivoBook-ASUSLaptop-X412FL-A412FL:~/Desktop/protokol$ cat endtoenddelay
20      4963.02         360551.24       208.86
30      7302.55         758072.51       37.34
50      34480.80        2036594.49      0.33
70      83392.98        3930796.52      1.22
90      191811.81       2512161.65      0.74
100     102046.46       2089977.87      19.60
```

Gambar 5.5 Hasil Formulasi *End-to-end Delay* Variasi Jumlah *Node*

Selanjutnya, hasil formulasi *end to end delay* dari setiap skenario variasi *node* yang merujuk atau mereferensi pada protokol *routing* AODV, AOMDV, dan DSR

telah didapatkan. Hasil formulasi *end to end delay* akan dirata-rata sehingga akan didapatkan hasil berupa protokol *routing* yang memiliki *end to end delay* terbaik. Hasil akhir tersebut didasarkan pada skenario perubahan atau perubahan jumlah *node*. Berikut merupakan informasi terkait nilai *end to end delay* berdasarkan pengujian perubahan jumlah *node* yang terdapat pada tabel 5.4.

Tabel 5.4 Hasil Pengujian *End-to-end Delay* Variasi Jumlah *Node*

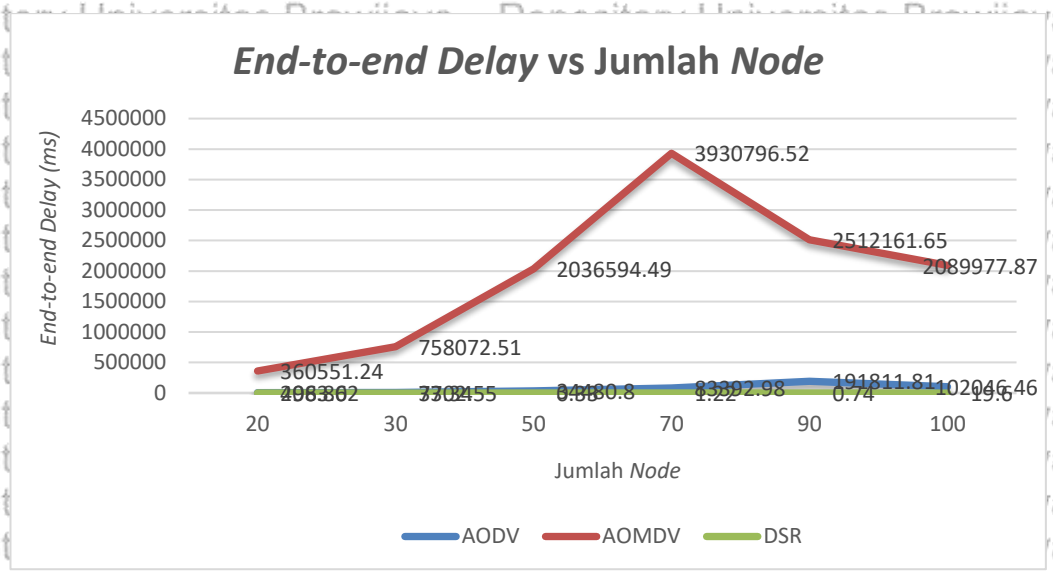
Jumlah *Node*	*End-to-end Delay* (ms)		
	AODV	AOMDV	DSR
20	4963,02	360551,24	208,86
30	7302,55	758072,51	37,34
50	34480,80	2036594,49	0,33
70	83392,98	3930796,52	1,22
90	191811,81	2512161,65	0,74
100	102046,46	2089977,87	19,60
Rata-rata	70666,27	1948025,71	44,68

Dari tabel 5.4, nilai *end to end delay* terhadap penambahan jumlah simpul dapat diketahui. Protokol DSR memiliki nilai terbaik dibandingkan dengan protokol yang lainnya, sedangkan protokol AOMDV memiliki nilai terburuk diantara protokol *routing* yang lainnya. Diketahui hasil dari *end to end delay* AODV adalah 4963,02 ms, AOMDV memiliki nilai 360551,24 ms sedangkan DSR mempunyai nilai 208,86 ms pada pengujian dengan skenario 20 *node*. Diketahui hasil dari *end to end delay* AODV adalah 7302,55 ms, AOMDV memiliki nilai 758072,51 ms sedangkan DSR mempunyai nilai 37,34 ms pada pengujian dengan skenario 30 *node*. Diketahui hasil dari *end to end delay* AODV adalah 34480,80 ms, AOMDV memiliki nilai 2036594,49 ms sedangkan DSR mempunyai nilai 0,33 ms pada pengujian dengan skenario 50 *node*. Diketahui hasil dari *end to end delay* AODV adalah 83392,98 ms, AOMDV memiliki nilai 3930796,52 ms sedangkan DSR mempunyai nilai 1,22 ms pada pengujian dengan skenario 70 *node*. Diketahui hasil dari *end to end delay* AODV adalah 191811,81 ms, AOMDV memiliki nilai 2512161,65 ms sedangkan DSR mempunyai nilai 0,74 ms pada pengujian dengan skenario 90 *node*. Diketahui hasil dari *end to end delay* AODV adalah 102046,46 ms, AOMDV memiliki nilai 2089977,87 ms sedangkan DSR mempunyai nilai 19,60 ms pada pengujian dengan skenario 100 *node*.

5.2.4 Analisis

Parameter QoS *end to end delay* merupakan parameter pengujian yang dijadikan rujukan maupun referensi dalam membandingkan tiga protokol *routing* AOMDV, DSR, dan AODV pada skenario perubahan jumlah *node*. Perubahan jumlah *node* yang berjumlah 20, 30, 50, 70, 90, dan 100 merupakan rincian *node* dalam melakukan hasil pengujian. Tampilan grafik akan digunakan untuk

mempermudah dalam analisis hasil pengujian parameter *end to end delay* pada skenario variasi jumlah *node*. Gambar 5.6 merupakan tampilan grafik nilai *end to end delay* terhadap skenario perubahan jumlah *node*.



Gambar 5.6 Grafik *End-to-end Delay* Terhadap Variasi Jumlah *Node*

Hasil perbandingan *end to end delay* antara protokol *routing* AOMDV, DSR, dan AODV berdasarkan skenario perubahan jumlah *node* yang bervariasi ditampilkan pada gambar 5.6 terkait grafik hasil pengujian. Perhitungan pada *end to end delay* ialah jumlah total waktu paket yang diterima dibagi dengan jumlah paket yang diterima.

Pada protokol AODV, merujuk atau mereferensi pada grafik nilai *end to end delay* cenderung mengalami kenaikan. Pengujian skenario perubahan *node* berjumlah 20 dengan nilai *end to end delay* 4963,02 ms terlihat mengalami kenaikan pertama pada grafik yaitu nilai *end to end delay* menjadi 7302,55 ms pada skenario perubahan *node* dengan jumlah 30. Pada pengujian skenario perubahan *node* dengan jumlah 50 mengalami kenaikan kedua pada grafik dengan nilai *end to end delay* menjadi 34480,80 ms. Kenaikan ketiga terdapat pada variasi jumlah *node* berjumlah 70 yakni menjadi 83392,98 ms. Pada pengujian skenario variasi jumlah 90 mengalami kenaikan keempat menjadi 191811,81 ms. Penurunan grafik terjadi pada skenario variasi *node* 100 yakni menjadi 102046,46 ms. Total dari keseluruhan yang terlihat pada grafik, protokol AODV memiliki nilai *end to end delay* yang cenderung naik.

Pada protokol AOMDV, merujuk atau mereferensi pada grafik nilai *end to end delay* cenderung mengalami kenaikan yang cukup signifikan. Pengujian skenario perubahan *node* berjumlah 20 dengan nilai *end to end delay* 360551,24 ms terlihat mengalami kenaikan pertama pada grafik yaitu nilai *end to end delay* menjadi 758072,51 ms pada skenario perubahan *node* dengan jumlah 30. Pada pengujian skenario perubahan *node* dengan jumlah 50 mengalami kenaikan kedua pada grafik dengan nilai *end to end delay* menjadi 2036594,49 ms. Kenaikan ketiga terdapat pada variasi jumlah *node* berjumlah 70 yakni menjadi 3930796,52 ms.

Pada pengujian skenario variasi jumlah 90 mengalami penurunan menjadi 2512161,65 ms. Penurunan kedua pada grafik terjadi pada skenario variasi *node* 100 yakni menjadi 2089977,87 ms. Total dari keseluruhan yang terlihat pada grafik, protokol AOMDV memiliki nilai *end to end delay* yang cenderung naik.

Pada protokol DSR, merujuk atau mereferensi pada grafik nilai *end to end delay* cenderung mengalami penurunan. Pengujian skenario perubahan *node* berjumlah 20 dengan nilai *end to end delay* 208,86 ms terlihat mengalami penurunan pertama pada grafik yaitu nilai *end to end delay* menjadi 37,34 ms pada skenario perubahan node dengan jumlah 30. Pada pengujian skenario perubahan *node* dengan jumlah 50 mengalami penurunan kedua pada grafik dengan nilai *end to end delay* menjadi 0,33 ms. Kenaikan pertama terdapat pada variasi jumlah *node* berjumlah 70 yakni menjadi 1,22 ms. Pada variasi jumlah *node* 90 terjadi penurunan ketiga yakni nilai *end to end delay* menjadi 0,74 ms. Kenaikan kedua terdapat pada variasi jumlah *node* dengan jumlah 100 yakni menjadi 19,60 ms. Total dari keseluruhan yang terlihat pada grafik, protokol DSR memiliki nilai *end to end delay* yang cenderung turun.

Berdasarkan hasil analisis yang telah ditetapkan sebelumnya, protokol *routing* AOMDV dan AODV cenderung mengalami kenaikan pada grafik sedangkan protokol *routing* DSR cenderung mengalami penurunan pada grafik. Kenaikan pada protokol AOMDV dan AODV dikarenakan pada kedua protokol tersebut tidak memiliki mekanisme *route cache* yang dimiliki protokol DSR sehingga proses pengiriman paket data menjadi lebih lama. Penurunan grafik pada protokol DSR disebabkan oleh mekanisme *route cache* yang dapat menyimpan informasi jalur rute pada saat dibutuhkan sehingga jalur alternatif dapat digunakan saat terjadi kegagalan rute.

Protokol *routing* AODV memiliki nilai *end to end delay* yang lebih baik dibandingkan dengan protokol *routing* AOMDV seiring dengan bertambahnya jumlah kepadatan *node* dikarenakan pada AODV duplikasi RREQ pada sebuah simpul akan dibuang. Protokol *routing* AOMDV memiliki kenaikan nilai *end to end delay* yang cukup signifikan seiring dengan bertambahnya jumlah kepadatan *node* dikarenakan pada protokol AOMDV *destination node* akan menerima semua paket *route request* baik yang terduplikasi ataupun tidak sehingga menyebabkan *delay* lebih lama dibandingkan dengan protokol lainnya.

Protokol *routing* DSR memiliki nilai *end to end delay* yang cenderung turun seiring dengan bertambahnya jumlah kepadatan *node*. Penurunan nilai *end to end delay* seiring dengan bertambahnya jumlah kepadatan *node* pada protokol *routing* DSR menandakan bahwa protokol *routing* DSR memiliki kinerja terbaik pada parameter *end to end delay* dikarenakan durasi pada saat pengiriman paket data antara *source node* menuju *destination node* menjadi semakin cepat seiring dengan bertambahnya jumlah *node* dibandingkan dengan protokol *routing* lainnya dimana semakin bertambahnya jumlah kepadatan *node* maka semakin tinggi nilai *end to end delay*.

Berdasarkan *end to end delay*, protokol *routing* DSR memiliki nilai *end to end delay* terbaik dibandingkan dengan protokol *routing* yang lainnya. Protokol *routing* DSR memiliki *end to end delay* terbaik dikarenakan DSR memiliki *route cache* yang tersedia ketika terjadi kegagalan rute sehingga saat melakukan pemeliharaan rute menjadi lebih cepat. Protokol DSR tidak perlu melakukan pemeliharaan rute ketika terjadi kegagalan rute melainkan dapat melihat *route cache* terlebih dahulu, jika terdapat *route alternative* maka data akan dikirimkan menggunakan rute tersebut sehingga *delay* DSR lebih rendah.

5.3 Analisis Gabungan Perbandingan Hasil Pengujian Protokol AOMDV, DSR dan AODV

Kinerja protokol *routing* terbaik dapat dilihat dengan melakukan perbandingan hasil pengujian antara protokol *routing* AOMDV, DSR, dan AODV dengan merujuk atau mereferensi pada analisis pengujian yang telah dilakukan sebelumnya. Tabel 5.5 menjelaskan terkait hasil analisis pengujian protokol *routing* yang telah dibandingkan dimana mencakup skenario, nama protokol *routing* serta keterangan kinerja.

Tabel 5.5 Perbandingan Hasil Analisis Pengujian Protokol

Skenario		AODV	AOMDV	DSR	Keterangan
Variasi Jumlah *Node*	Rata-rata *Throughput* (Kbps)	12408,08	43331.87	215273,36	DSR lebih baik
	Rata-rata *End-to-end Delay* (ms)	70666,27	1948025,71	44,68	DSR lebih baik

Hasil analisis perbandingan antara *routing protocol* AODV, AOMDV, dan DSR berdasarkan skenario perubahan jumlah *node* ditunjukkan pada tabel 5.5. Penerapan tiap skenario berpengaruh terhadap parameter pengujian yang digunakan yakni *end to end delay* dan *throughput*. Pada skenario perubahan jumlah *node*, protokol *routing* DSR memiliki rata-rata *throughput* dengan nilai terbaik yakni 215273,36 Kbps. Protokol AODV memiliki nilai rata-rata *throughput* terburuk dengan nilai 12408,08 Kbps. Pada skenario perubahan jumlah *node* dengan merujuk atau mereferensi pada parameter pengujian *end to end delay*, protokol *routing* DSR mempunyai nilai rata-rata *end to end delay* terbaik dengan nilai 44,68 ms. Protokol *routing* AOMDV mempunyai nilai rata-rata *end to end delay* terburuk dengan nilai 1948025,71 ms.

BAB 6 PENUTUP

6.1 Kesimpulan

Bagian kesimpulan membahas terkait hasil yang didapatkan selama penelitian yang telah dilakukan. Kesimpulan yang didapatkan dengan merujuk atau mereferensi pada analisis yang telah dilakukan pada bab sebelumnya adalah sebagai berikut:

1. Protokol *routing* AOMDV, DSR, AODV berhasil diimplementasikan pada ruang lingkup MANET dengan pergerakan dan topologi dinamis menggunakan model *Random Way Point* untuk pergerakan setiap *node*.
2. Kinerja protokol *routing* setelah dilakukan analisis yaitu pada parameter *throughput*, DSR memiliki kinerja terbaik dengan nilai 215273,36 Kbps sedangkan AODV memiliki kinerja terburuk dengan nilai 12408,08 Kbps. Pada parameter *end-to-end delay*, DSR memiliki kinerja terbaik dengan nilai 44,68 ms sedangkan AOMDV memiliki kinerja terburuk dengan nilai 1948025,71 ms.

6.2 Saran

Saran untuk penelitian selanjutnya yang berkaitan dengan penggunaan protokol *routing Ad Hoc On Demand Multipath Distance Vector* (AOMDV), *Dynamic Source Routing* (DSR), *Ad Hoc On Demand Distance Vector* (AODV) adalah sebagai berikut:

1. Perlu dilakukan analisis kinerja pada protokol AOMDV, DSR, dan AODV menggunakan simulator yang berbeda.
2. Perlu dilakukan analisis kinerja pada protokol AOMDV, DSR, dan AODV menggunakan tambahan skenario serangan.

DAFTAR REFERENSI

Anisia, R., Munadi, R. & Negara, M. R., 2016. ANALISIS PERFORMANSI ROUTING PROTOCOL OLSR DAN AOMDV PADA VEHICULAR AD HOC NETWORK (VANET). *Jurnal Nasional Teknik Elektro*, 5(1).

Bahari, A. M., Trisnawan, H. P. & Siregar, A. R., 2019. Analisis Kinerja Protokol AODV (*Ad Hoc On-Demand Distance Vector*) dan AOMDV (*Ad Hoc On-Demand Multipath Distance Vector*) Terhadap Serangan Aktif Pada Jaringan Manet (*Mobile Ad Hoc Network*). *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, 3(4).

Barolli, Oda, dkk., 2017. A GA-Based Simulation System for WMNs: Performance Analysis for Different WMN Architectures Considering Exponential Distribution, HWMP and TCP Protocols. *IEEE 31st International Conference on Advanced Information Networking and Applications.*

Basagni, Conti, dkk. 2004. *Mobile Ad Hoc Networking*. New Jersey: Piscataway.

Bonaventure, O., 2011. *Computer Networking: Principles, Protocols and Practice*. Universite catholique de Louvain: The Saylor Foundation.

Dordal, R. P., 2020. *An Introduction to Computer Networks*. Loyola University Chicago: Department of Computer Science.

Fatkhurrozi., Widasari, R. E. & Bhawiyuga, A., 2018. Analisis Perbandingan Kinerja Protokol AOMDV, DSDV, Dan ZRP Sebagai Protokol *Routing* Pada *Mobile Ad-Hoc Network* (MANET). *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, 2(10).

Haris, I. M., Trisnawan, H. P. & Primananda, R., 2019. Perbandingan Kinerja Protokol DSDV dan FSR Terhadap Model Node Tetap dan Node Bergerak. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, 3(9).

Ilyas, M., 2003. *Ad Hoc Wireless Networks*. Florida: Boca Raton.

Kochher, R. & Mehta, R., 2016. Performance Analysis of Reactive AODV and DSR with Hybrid GRP Routing Protocols under IEEE 802.11g MANET. *2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET).*

Khurana, S. & Kumar, S., 2017. A Study of Congestion Control in MANET: Review. *International Journal of Engineering Technology Science and Research (IJETSR).*

Kumar, Singh, dkk., 2016. Study and Performance Analysis of Routing Protocol Based on CBR. *International Conference on Computational Modeling and Security.*

Kurose, J. F. & Ross, K. W., 2017. *Computer Networks A Top-Down Approach Seventh Edition*. New Jersey: Pearson.

Neeraj, Yedupati, dkk. 2018. PERFORMANCE ANALYSIS OF DIFFERENT ROUTING PROTOCOLS IN MANET USING DIFFERENT PARAMETERS IN DIFFERENT RANGES. *2018 2nd International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)*.

Jaganath, M. & Vikram, R., 2019. Performance Evaluation of MANET Routing Protocol under Black Hole Attack Using OPNET Simulator. *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*.

Jyoti, & Saini, H., 2017. A Study on Networks and Comparison of Wired, Wireless and Optical Networks. *International Journal of Innovative Research in Computer and Communication Engineering*.

Tamimi, A. A. & Khalifeh, M. J., 2010. *Computer Networks and Communications Third Edition*. Al-Zaytoonah University: Department of Computer Science.

Thakor, B. J. & Joshi, C. N., 2017. Computer Networks. *International Journal of Science and Research (IJSR)*.

LAMPIRAN A *SCRIPT* TCL

A.1 Kode Sumber AODV.tcl

```
# ======================================================================
# Define options
# ======================================================================
set opt(chan)           Channel/WirelessChannel    ;# channel type
set opt(prop)           Propagation/TwoRayGround   ;# radio-propagation model
set opt(netif)          Phy/WirelessPhy            ;# network interface type
set opt(mac)            Mac/802_11                 ;# MAC type
set opt(ifq)            Queue/DropTail/PriQueue    ;# interface queue type
set opt(ll)             LL                         ;# link layer type
set opt(ant)            Antenna/OmniAntenna        ;# antenna model
set opt(ifqlen)         50                         ;# max packet in ifq
set opt(nn)             20                         ;# number of mobilenodes
set opt(minSpeed)       0.5                        ;# movement minimum speed [m/s]
set opt(maxSpeed)       2                          ;# movement maximum speed [m/s]
set opt(rp)             AODV                       ;# routing protocol
set opt(seed)           1                          ;# general pseudo-random sequence generator
set opt(x)              1000                       ;# x coordinate of topology
set opt(y)              1000                       ;# y coordinate of topology
set opt(stop)           1000                       ;# time to stop simulation
#
# ======================================================================
# Initialization
#
# ======================================================================
# create simulator instance
set ns_ [new Simulator]

# open traces
set tracefd  [open aodv0mal.tr w]
set namtrace [open aodv0mal.nam w]

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

```
# create topography object
set topo [new Topography]

# define topology
$topo load_flatgrid $opt(x) $opt(y)

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

# configure mobile nodes
$ns_ node-config -adhocRouting $opt(rp) \
                 -llType $opt(ll) \
                 -macType $opt(mac) \
                 -ifqType $opt(ifq) \
                 -ifqLen $opt(ifqlen) \
                 -antType $opt(ant) \
                 -propType $opt(prop) \
                 -phyType $opt(netif) \
                 -channelType $opt(chan) \
                 -topoInstance $topo \
                 -wiredRouting OFF \
                 -agentTrace ON \
                 -routerTrace ON \
                 -macTrace OFF
#
# ============================================================
# Pseudo-random sequence generator
#
# ============================================================
# General pseudo-random sequence generator
set genSeed [new RNG]
$genSeed seed $opt(seed)
set randomSeed [new RandomVariable/Uniform]
$randomSeed use-rng $genSeed
$randomSeed set min_ 1.0
$randomSeed set max_ 1000.0

# Mobility model: x node position [m]
set genNodeX [new RNG]
$genNodeX seed [expr [$randomSeed value]]
set randomNodeX [new RandomVariable/Uniform]
$randomNodeX use-rng $genNodeX
$randomNodeX set min_ 1.0
$randomNodeX set max_ [expr $opt(x) - 1.0]

# Mobility model: y node position [m]
set posNodeY [new RNG]
$posNodeY seed [expr [$randomSeed value]]
set randomNodeY [new RandomVariable/Uniform]
$randomNodeY use-rng $posNodeY
$randomNodeY set min_ 1.0
$randomNodeY set max_ [expr $opt(y) - 1.0]

# Mobility model: node speed [m/s]
set genNodeSpeed [new RNG]
```

```
$genNodeSpeed seed [expr [$randomSeed value]]
set randomNodeSpeed [new RandomVariable/Uniform]
$randomNodeSpeed use-rng $genNodeSpeed
$randomNodeSpeed set min_ $opt(minSpeed)
$randomNodeSpeed set max_ $opt(maxSpeed)

#
# Repository Universitas Brawijaya Repository Universitas Brawijaya
# Nodes definition
#
# Repository Universitas Brawijaya Repository Universitas Brawijaya
for {set i 0} {$i < $opt(nn)} {incr i} {
    set node_($i) [$ns_ node]
}
for {set i 0} {$i < $opt(nn)} {incr i} {
set X [expr [$randomNodeX value] ]
$node_($i) set X_ $X
set Y [expr [$randomNodeY value] ]
$node_($i) set Y_ $Y
$node_($i) set Z_ 0.0
$node_(0) set X_ 1
$node_(0) set Y_ 1
$node_(0) set Z_ 0.0
$node_(1) set X_ 999
$node_(1) set Y_ 999
$node_(1) set Z_ 0.0
}

for {set i 0} {$i < $opt(nn)} {incr i} {
set xr_ [$randomNodeX value]
set yr_ [$randomNodeY value]
set spd_ [$randomNodeSpeed value]
$ns_ at 0 "$node_($i) setdest $xr_ $yr_ $spd_";
set xr1_ [$randomNodeX value]
set yr1_ [$randomNodeY value]
set spd1_ [$randomNodeSpeed value]
$ns_ at 400 "$node_($i) setdest $xr1_ $yr1_ $spd1_";
}
$ns_ at 0.0 "$node_(0) color blue"
$node_(0) color "blue"
$ns_ at 0.0 "$node_(0) label \"Source\""
$ns_ at 0.0 "$node_(1) color green"
$node_(1) color "green"
$ns_ at 0.0 "$node_(1) label \"Destination\""

set udp_(0) [new Agent/UDP]
$ns_ attach-agent $node_(0) $udp_(0)
set null_(0) [new Agent/Null]
$ns_ attach-agent $node_(1) $null_(0)
set cbr_(0) [new Application/Traffic/CBR]
$cbr_(0) set packetSize_ 512
#$cbr_(0) set rate_ 0.1mb
$cbr_(0) set interval_ 1
$cbr_(0) set random_ false
```

```
$cbr_(0) attach-agent $udp_(0)
$ns_ connect $udp_(0) $null_(0)
$ns_ at 0.0 "$cbr_(0) start"

#
# define initial node position in nam
#
for {set i 0} {$i < $opt(nn)} {incr i} {
    $ns_ initial_node_pos $node_($i) 30
}
#
# tell all nodes when the simulation ends
#
for {set i 0} {$i < $opt(nn) } {incr i} {
    $ns_ at $opt(stop).0 "$node_($i) reset";
}
$ns_ at $opt(stop) "$ns_ nam-end-wireless $opt(stop)"
$ns_ at $opt(stop).0002 "puts \"NS EXITING...\" ; $ns_ halt"
$ns_ at $opt(stop).0001 "stop"

proc stop {} {
    global ns_ tracefd namtrace
    $ns_ flush-trace
    close $tracefd
    close $namtrace
}
#
# begin simulation
#
puts "Starting Simulation..."
$ns_ run
```

A.2 Kode Sumber AOMDV.tcl

```
# ======================================================================
# Define options
# ======================================================================
set opt(chan)           Channel/WirelessChannel  ;# channel type
set opt(prop)           Propagation/TwoRayGround ;# radio-
propagation model
set opt(netif)          Phy/WirelessPhy          ;# network
interface type
set opt(mac)            Mac/802_11               ;# MAC type
set opt(ifq)            Queue/DropTail/PriQueue  ;# interface
queue type
set opt(ll)             LL                       ;# link layer
type
set opt(ant)            Antenna/OmniAntenna      ;# antenna model
set opt(ifqlen)         50                       ;# max packet in
ifq
set opt(nn)             20                       ;# number of
mobilenodes
```

```
set opt(minSpeed)       0.5                 ;# movement minimum
speed [m/s]
set opt(maxSpeed)       2                   ;#     movement
maximum speed [m/s]
set opt(rp)             AOMDV               ;# routing
protocol
set opt(seed)           1                   ;# general
pseudo-random sequence generator
set opt(x)              1000                ;# x coordinate
of topology
set opt(y)              1000                ;# y coordinate
of topology
set opt(stop)           1000                ;# time to stop
simulation
#
# ======================================================================
# Initialization
#
# ======================================================================
# create simulator instance
set ns_ [new Simulator]

# open traces
set tracefd  [open aomdv0mal.tr w]
set namtrace  [open aomdv0mal.nam w]

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

# create topography object
set topo [new Topography]

# define topology
$topo load_flatgrid $opt(x) $opt(y)

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

# configure mobile nodes
$ns_ node-config -adhocRouting $opt(rp) \
                 -llType $opt(ll) \
                 -macType $opt(mac) \
                 -ifqType $opt(ifq) \
                 -ifqLen $opt(ifqlen) \
                 -antType $opt(ant) \
                 -propType $opt(prop) \
                 -phyType $opt(netif) \
                 -channelType $opt(chan) \
                 -topoInstance $topo \
                 -wiredRouting OFF \
                 -agentTrace ON \
                 -routerTrace ON \
                 -macTrace OFF
```

```
#
# ======================================================================
# Pseudo-random sequence generator
#
# ======================================================================
# General pseudo-random sequence generator
set genSeed [new RNG]
$genSeed seed $opt(seed)
set randomSeed [new RandomVariable/Uniform]
$randomSeed use-rng $genSeed
$randomSeed set min_ 1.0
$randomSeed set max_ 1000.0

# Mobility model: x node position [m]
set genNodeX [new RNG]
$genNodeX seed [expr [$randomSeed value]]
set randomNodeX [new RandomVariable/Uniform]
$randomNodeX use-rng $genNodeX
$randomNodeX set min_ 1.0
$randomNodeX set max_ [expr $opt(x) - 1.0]

# Mobility model: y node position [m]
set posNodeY [new RNG]
$posNodeY seed [expr [$randomSeed value]]
set randomNodeY [new RandomVariable/Uniform]
$randomNodeY use-rng $posNodeY
$randomNodeY set min_ 1.0
$randomNodeY set max_ [expr $opt(y) - 1.0]

# Mobility model: node speed [m/s]
set genNodeSpeed [new RNG]
$genNodeSpeed seed [expr [$randomSeed value]]
set randomNodeSpeed [new RandomVariable/Uniform]
$randomNodeSpeed use-rng $genNodeSpeed
$randomNodeSpeed set min_ $opt(minSpeed)
$randomNodeSpeed set max_ $opt(maxSpeed)
#
# ======================================================================
# Nodes definition
#
# ======================================================================
for {set i 0} {$i < $opt(nn)} {incr i} {
    set node_($i) [$ns_ node]
}
for {set i 0} {$i < $opt(nn)} {incr i} {
set X [expr [$randomNodeX value] ]
$node_($i) set X_ $X
set Y [expr [$randomNodeY value] ]
$node_($i) set Y_ $Y
$node_($i) set Z_ 0.0
$node_(0) set X_ 1
```

```
$node_(0) set Y_ 1
$node_(0) set Z_ 0.0
$node_(1) set X_ 999
$node_(1) set Y_ 999
$node_(1) set Z_ 0.0
}

for {set i 0} {$i < $opt(nn)} {incr i} {
set xr_ [$randomNodeX value]
set yr_ [$randomNodeY value]
set spd_ [$randomNodeSpeed value]
$ns_ at 0 "$node_($i) setdest $xr_ $yr_ $spd_";
set xr1_ [$randomNodeX value]
set yr1_ [$randomNodeY value]
set spd1_ [$randomNodeSpeed value]
$ns_ at 400 "$node_($i) setdest $xr1_ $yr1_ $spd1_";
}

$ns_ at 0.0 "$node_(0) color blue"
$node_(0) color "blue"
$ns_ at 0.0 "$node_(0) label \"Source\""
$ns_ at 0.0 "$node_(1) color green"
$node_(1) color "green"
$ns_ at 0.0 "$node_(1) label \"Destination\""

set udp_(0) [new Agent/UDP]
$ns_ attach-agent $node_(0) $udp_(0)
set null_(0) [new Agent/Null]
$ns_ attach-agent $node_(1) $null_(0)
set cbr_(0) [new Application/Traffic/CBR]
$cbr_(0) set packetSize_ 512
#$cbr_(0) set rate_ 0.1mb
$cbr_(0) set interval_ 1
$cbr_(0) set random_ false
$cbr_(0) attach-agent $udp_(0)
$ns_ connect $udp_(0) $null_(0)
$ns_ at 0.0 "$cbr_(0) start"

#
# define initial node position in nam
#
for {set i 0} {$i < $opt(nn)} {incr i} {
$ns_ initial_node_pos $node_($i) 30
}

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

$ns_ at $opt(stop) "$ns_ nam-end-wireless $opt(stop)"
$ns_ at $opt(stop).0002 "puts \"NS EXITING...\" ; $ns_ halt"
$ns_ at $opt(stop).0001 "stop"

proc stop {} {
    global ns_ tracefd namtrace
    $ns_ flush-trace
```

```
    close $tracefd
    close $namtrace
}
#
# begin simulation
#
puts "Starting Simulation..."
$ns_ run
```

A.3 Kode Sumber DSR.tcl

```
# ======================================================================
# Define options
# ======================================================================
set opt(chan)           Channel/WirelessChannel  ;# channel type
set opt(prop)         Propagation/TwoRayGround ;# radio-propagation model
set opt(netif)          Phy/WirelessPhy             ;# network interface type
set opt(mac)            Mac/802_11                  ;# MAC type
set opt(ifq)            Queue/DropTail/PriQueue    ;# interface queue type
set opt(ll)             LL                          ;# link layer type
set opt(ant)            Antenna/OmniAntenna         ;# antenna model
set opt(ifqlen)         50                          ;# max packet in ifq
set opt(nn)             20                          ;# number of mobilenodes
set opt(minSpeed)       0.5                         ;# movement minimum speed [m/s]
set opt(maxSpeed)       2                           ;# movement maximum speed [m/s]
set opt(rp)             DSR                         ;# routing protocol
set opt(seed)           1                           ;# general pseudo-random sequence generator
set opt(x)              1000                        ;# x coordinate of topology
set opt(y)              1000                        ;# y coordinate of topology
set opt(stop)           1000                        ;# time to stop simulation
#
# ======================================================================
# Initialization
#
# ======================================================================
# create simulator instance
set ns_ [new Simulator]
```

```
# open traces
set tracefd  [open dsr0mal.tr w]
set namtrace [open dsr0mal.nam w]

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

# create topography object
set topo [new Topography]

# define topology
$topo load_flatgrid $opt(x) $opt(y)

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

# configure mobile nodes
$ns_ node-config -adhocRouting $opt(rp) \
                 -llType $opt(ll) \
                 -macType $opt(mac) \
                 -ifqType $opt(ifq) \
                 -ifqLen $opt(ifqlen) \
                 -antType $opt(ant) \
                 -propType $opt(prop) \
                 -phyType $opt(netif) \
                 -channelType $opt(chan) \
                 -topoInstance $topo \
                 -wiredRouting OFF \
                 -agentTrace ON \
                 -routerTrace ON \
                 -macTrace OFF

#
# ============================================================================
# Pseudo-random sequence generator
#
# ============================================================================

# General pseudo-random sequence generator
set genSeed [new RNG]
$genSeed seed $opt(seed)
set randomSeed [new RandomVariable/Uniform]
$randomSeed use-rng $genSeed
$randomSeed set min_ 1.0
$randomSeed set max_ 1000.0

# Mobility model: x node position [m]
set genNodeX [new RNG]
$genNodeX seed [expr [$randomSeed value]]
set randomNodeX [new RandomVariable/Uniform]
$randomNodeX use-rng $genNodeX
$randomNodeX set min_ 1.0
$randomNodeX set max_ [expr $opt(x) - 1.0]
# Mobility model: y node position [m]
set posNodeY [new RNG]
$posNodeY seed [expr [$randomSeed value]]
```

```
set randomNodeY [new RandomVariable/Uniform]
$randomNodeY use-rng $posNodeY
$randomNodeY set min_ 1.0
$randomNodeY set max_ [expr $opt(y) - 1.0]
# Mobility model: node speed [m/s]
set genNodeSpeed [new RNG]
$genNodeSpeed seed [expr [$randomSeed value]]
set randomNodeSpeed [new RandomVariable/Uniform]
$randomNodeSpeed use-rng $genNodeSpeed
$randomNodeSpeed set min_ $opt(minSpeed)
$randomNodeSpeed set max_ $opt(maxSpeed)
#
# ============================================================================
# Nodes definition
#
# ============================================================================
for {set i 0} {$i < $opt(nn)} {incr i} {
    set node_($i) [$ns_ node]
}

for {set i 0} {$i < $opt(nn)} {incr i} {
set X [expr [$randomNodeX value] ]
$node_($i) set X_ $X
set Y [expr [$randomNodeY value] ]
$node_($i) set Y_ $Y
$node_($i) set Z_ 0.0
$node_(0) set X_ 1
$node_(0) set Y_ 1
$node_(0) set Z_ 0.0
$node_(1) set X_ 999
$node_(1) set Y_ 999
$node_(1) set Z_ 0.0
}

for {set i 0} {$i < $opt(nn)} {incr i} {
set xr_ [$randomNodeX value]
set yr_ [$randomNodeY value]
set spd_ [$randomNodeSpeed value]
$ns_ at 0 "$node_($i) setdest $xr_ $yr_ $spd_";
set xr1_ [$randomNodeX value]
set yr1_ [$randomNodeY value]
set spd1_ [$randomNodeSpeed value]
$ns_ at 400 "$node_($i) setdest $xr1_ $yr1_ $spd1_";
}

$ns_ at 0.0 "$node_(0) color blue"
$node_(0) color "blue"
$ns_ at 0.0 "$node_(0) label \"Source\""
$ns_ at 0.0 "$node_(1) color green"
$node_(1) color "green"
$ns_ at 0.0 "$node_(1) label \"Destination\""

set udp_(0) [new Agent/UDP]
$ns_ attach-agent $node_(0) $udp_(0)
```

```
set null_(0) [new Agent/Null]
$ns_ attach-agent $node_(1) $null_(0)
set cbr_(0) [new Application/Traffic/CBR]
$cbr_(0) set packetSize_ 512
#$cbr_(0) set rate_ 0.1mb
$cbr_(0) set interval_ 1
$cbr_(0) set random_ false
$cbr_(0) attach-agent $udp_(0)
$ns_ connect $udp_(0) $null_(0)
$ns_ at 0.0 "$cbr_(0) start"
#
# define initial node position in nam
#
for {set i 0} {$i < $opt(nn)} {incr i} {
    $ns_ initial_node_pos $node_($i) 30
}
#
# tell all nodes when the simulation ends
#
for {set i 0} {$i < $opt(nn) } {incr i} {
    $ns_ at $opt(stop).0 "$node_($i) reset";
}

$ns_ at $opt(stop) "$ns_ nam-end-wireless $opt(stop)"
$ns_ at $opt(stop).0002 "puts \"NS EXITING...\" ; $ns_ halt"
$ns_ at $opt(stop).0001 "stop"

proc stop {} {
    global ns_ tracefd namtrace
    $ns_ flush-trace
    close $tracefd
    close $namtrace
}
#
# begin simulation
#
puts "Starting Simulation..."

$ns_ run
```

LAMPIRAN B *SCRIPT* AWK

B.1 Kode Sumber throughput.awk

```
BEGIN{
    sum = 0;
}
{
    if($1>0)
    printf("%d\t%f\n",$1,sum/$1);
    sum=sum+$2
}
END{
    puts"Done"
}
```

B.2 Kode Sumber delay.awk

```
BEGIN{
    sum = 0;
}
{
    if($2>0)
    printf("%d\t%f\n",$2,sum/$2);
    sum=sum+$1
}
END{
    puts"Done"
}
```