ANALISIS KINERJA PROTOKOL ROUTING AD HOC ON-DEMAND DISTANCE VECTOR (AODV) DAN DESTINATION SEQUENCE DISTANCE VECTOR (DSDV) PADA MOBILE AD HOC NETWORK (MANET)

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Tanudjaya Tri Ratna Sari
NIM: 125150200111005

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2016

PENGESAHAN

ANALISIS KINERJA PROTOKOL *ROUTING* AD HOC ON-DEMAND DISTANCE VECTOR (AODV) DAN DESTINATION SEQUENCE DISTANCE VECTOR (DSDV) PADA MOBILE AD HOC NETWORK (MANET)

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Tanudjaya Tri Ratna Sari
NIM: 125150200111005

Skripsi ini telah diuji dan dinyatakan lulus pada
11 Agustus 2016
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Mochammad Hannats Hanafi I., S.ST, M.T	Rakhmadhany Primananda, S.T, M.Kom
NIK: 201405 881229 1 001	NIK: -

Mengetahui
Ketua Jurusan Teknik Informatika

Tri 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 pustaka.

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, 2 Agustus 2016

Tanudjaya Tri Ratna Sari

NIM: 125150200111005

KATA PENGANTAR

Puji syukur pada Tuhan Yang Maha Esa sehingga penulis dapat menyelesaikan skripsi ini dengan baik. Pada kesempatan kali ini penulis juga ingin menyampaikan rasa terima kasih yang sebesar-besarnya kepada berbagai pihak yang telah membantu penyelesaian skripsi ini. Penulis mengucapkan terima kasih terutama kepada:

1. Mama tercinta yang menjadi motivasi utama penulis dalam menyelesaikan studinya atas dukungan materi dan moral serta doa yang terus-menerus mengalir;
2. Kakak-kakak dan adik atas dukungan dan semangat;
3. Bapak Mochammad Hannats Hanafi I., S.ST, M.T dan Bapak Rakhmadhany Primananda, S.T, M.Kom atas pendampingan dan bimbingan selaku dosen pembimbing skripsi;
4. Lyna Dwi Maryati dan Sarah Fitriannisa Ruzis atas bantuan dan dukungan yang diberikan;
5. Ibu Sriredjeki Pratama Wulandari dan Ibu Tutik atas doa dan dukungan;
6. Mbak Elviana atas semangat dan dukungan;
7. Teman-teman SMAN 1 Balikpapan dan SMPN 1 Balikpapan terutama Kiki Puspitasari dan Celindianova Putri atas doa dan dukungan yang diberikan;
8. Teman-teman dari Kelas A Informatika 2012, terutama Anita Kusumawardhani dan Kusnul Aeni atas dukungannya;
9. Teman-teman kos Bapak Juhana Sutibi atas dukungan dan semangat yang diberikan;
10. Teman-teman jurusan Teknik Informatika Universitas Brawijaya angkatan 2012;
11. Serta pihak-pihak lain yang tidak bisa penulis sebutkan satu persatu.

Akhir kata semoga skripsi ini bermanfaat bagi perkembangan ilmu pengetahuan bidang teknologi informasi dan jaringan komputer.

Malang, 2 Agustus 2016

Penulis

125150200111005@mail.ub.ac.id

ABSTRAK

Mobile Ad Hoc Network (MANET) merupakan jaringan nirkabel yang terdiri dari kumpulan mobile *node* yang sanggup melakukan operasi otonom tanpa bergantung pada struktur jaringan permanen untuk melakukan komunikasi antar *node*. Mobile *node* pada MANET berperan ganda sebagai host dan router untuk bertukar informasi didalam jaringan. Karena sifatnya yang mobile, *node* pada MANET juga terus menerus berpindah lokasi sehingga topologi dan *path* antar *node* ikut berubah. Oleh karena itu peran protokol *routing* sangat penting dalam menentukan rute paket serta pemeliharaan informasi mengenai *node-node* yang berada didalam jaringan yang dinamis. Untuk mendukung mekanisme komunikasi antar *node* tersebut beberapa protokol *routing* sengaja dirancang khusus menyesuaikan karakteristik dari ad hoc network, beberapa yang paling sering digunakan adalah AODV (*Ad hoc On-demand Distance Vector*) dan DSDV (*Destination Sequence Distance Vector*). Masing-masing protokol tadi merupakan protokol *table-driven routing* dan *on-demand routing* yang memiliki metode implementasi yang berbeda dalam mengatur *traffic* didalam jaringan. Penelitian dilakukan untuk mencari tahu dan membandingkan performansi dari AODV dan DSDV terhadap skenario simulasi yang diujikan. Adapun simulasi dilakukan dengan mengunakan Network Simulator 3 untuk mendapatkan ukuran performansi seperti *average end to end delay*, *packet delivery ratio*, dan *routing overhead*. Dari simulasi yang dilakukan terhadap jumlah *node*, luas area jaringan, dan mobilitas jaringan dapat disimpulkan bahwa performa DSDV secara umum lebih unggul dari AODV dilihat dari nilai *average end to end delay*, *packet delivery ratio* dan *routing overhead*nya. DSDV juga lebih cocok diaplikasikan pada jaringan *ad hoc* yang memiliki mobilitas rendah karena nilai PDR-nya yang tinggi dan lebih stabil dibandingkan AODV tanpa terpengaruh dengan peningkatan jumlah *node*. Akan tetapi jika diaplikasikan pada luas area jaringan yang lebih luas atau mobilitas yang tinggi, performansi AODV lebih baik dibandingkan DSDV.

Kata kunci: MANET, protokol *routing*, AODV, DSDV, *delay*, *routing overhead*, *packet delivery ratio*

ABSTRACT

Mobile ad hoc network (MANET) is a wireless network that consisted of mobile nodes which able to do autonomous operation without relying to permanent infrastructure for communicating among nodes. Mobile nodes in MANET has two functions as a host and a router for exchanging information within network. Due to its nature for being mobile, nodes in MANET can continuously changing its location and at the same time changing network topology and path. That is why routing protocol is very important for determining packet route and maintaining information about other nodes location in such dynamic network. For that kind of inter-node communication mechanism, there are several routing protocols that specifically designed to suit ad hoc network's characteristic, two of them are AODV (Ad hoc On-demand Distance Vector) and DSDV (Destination Sequence Distance Vector). Each of those protocols are table-driven routing protocol and on-demand routing protocol which has its own implementation method for controlling network traffic. This research is conducted to find and compare performance of AODV and DSDV as routing protocol in simulation scenarios that would be tested. Simulation is done by using Network Simulator 3, Flow Monitor and awk script to get the values of performance metrics such as average end to end delay, packet delivery ratio and routing overhead. From simulations against network load, network size and network mobility. Those simulations provide results that indicates DSDV overall performance to be better than AODV. DSDV is also convenient for ad hoc network with fewer mobility because its PDR is higher and stable than AODV even though there are added nodes. But if it is for a bigger network area or high mobility, AODV performance is better than DSDV.

Keywords: MANET, routing protocol, AODV, DSDV, delay, routing, routing overhead, packet delivery ratio.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB 1 PENDAHULUAN

1.1 Latar belakang

Mobile ad hoc network (MANET) merupakan jaringan nirkabel *multi-hop* yang terdiri dari kumpulan *mobile host* yang sanggup melakukan operasi otonom tanpa bergantung pada struktur jaringan permanen. Kemampuan dari *node* yang berada pada jaringan MANET untuk berpindah-pindah lokasi membuat topologi jaringan dan *path* antar *node* terus menerus berubah. Kebanyakan penelitian mengenai MANET berasumsi bahwa informasi jaringan terkait alamat IP, *netmask* dan sebagainya telah dikonfigurasi secara statikal sebelum *node* tersebut bergabung di jaringan MANET (Nesargi, 2002). Padahal kenyataannya tidak semua *node* memiliki alamat IP permanen sebelum masuk jaringan MANET, *node* tersebut bisa jadi menggunakan *dynamic host configuration protocol* untuk mendapatkan alamat IP. Akan tetapi solusi ini tidak dapat diterapkan pada MANET yang tidak memiliki *server* terpusat, melainkan *node* itu sendiri yang berperan ganda sebagai *host* dan *router*.

Selain itu dengan keterbatasan jangkauan dari *interface* jaringan nirkabel membuat satu *node* harus beberapa kali melakukan *hop*/lompatan untuk dapat berkomunikasi dengan *node* lainnya di jaringan MANET yang sama. Untuk mendukung komunikasi *multi-hop* tadi, diperlukan protokol *routing* yang dapat menentukan rute yang harus ditempuh suatu *node* untuk berkomunikasi satu sama lain yang secara dinamis mengikuti perubahan topologi jaringan itu sendiri.

Adapun beberapa protokol *routing* yang khusus dikembangkan untuk *ad hoc network* diklasifikasikan menjadi beberapa kelompok:

a. *Table-driven (proactive) routing*

   Table-driven protocol memelihara informasi *routing* yang ada pada setiap *node* di jaringan agar tetap konsisten dan *up-to-date.* Protokol *routing* proaktif mengharuskan tiap *node* untuk menyimpan informasi *routing*-nya dan memperbaharui semua informasi *routing* dari tiap *node* dalam jaringan setiap kali terdapat perubahan didalamnya (Jayakumar, 2007). Contoh: *Destination sequenced distance vector routing* (DSDV), *Optimised Link State Routing Protocol* (FSR).

b. *On-demand (reactive) routing*

   Pada *on-demand protocol*, rute dibuat hanya pada saat dibutuhkan. Saat *node* satu ingin mengirimkan pesan ke *node* tujuan, mekanisme

penemuan rute akan dilakukan untuk mencari jalur dari *node* sumber ke *node* tujuan. Rute tersebut akan terus dipertahankan selama *node* tujuan masih dapat dicapai atau hingga rute tersebut tidak lagi dibutuhkan (Jayakumar, 2007). Contoh: *Ad hoc On Demand Distance Vector* (AODV), *Dynamic Source Routing Protocol* (DSR), *Temporarly ordered routing algorithm* (TORA).

c. *Hybrid (proactive-reactive) routing*

Tipe protokol ini mengkombinasikan keuntungan dari penggunaan *routing* proaktif dan reaktif. *Routing* dilakukan dengan menetapkan rute awal yang menjadi dasar informasi *routing* dari *node* yang berada di jaringan *ad hoc*, kemudian seterusnya tiap kali ada penambahan *node* baru protokol *hybrid* akan melakukan *reactive flooding* untuk mendapatkan rute tambahan sesuai permintaan (Samar, 2004). Contoh*: Zone Protocol Routing* (ZRP).

d. *Hierarchical routing*

Dengan *hierarchical routing*, pemilihan penggunaan *proactive* dan *reactive routing* bergantung dari tingkat hierarki dari *node* yang bersangkutan. Rute awal didapatkan dengan pendekatan proaktif, lalu secara reaktif melayani permintaan dari *node* tambahan yang berada pada tingkat hierarki yang lebih rendah (Mauve, 2001). Contoh: Cluster Based *Routing* Protocol (CBRP*), Zone-based Hierarchical Link State Routing Protocol* (ZHLS).

Pada penelitian ini akan dilakukan perbandingan kinerja antar *reactive routing*, yakni AODV dan DSDV sebagai protokol *routing* pada MANET untuk melihat performansi dari masing-masing protokol terhadap topologi MANET yang akan disimulasikan.

1.2 Rumusan masalah

Pertanyaan penelitian:

1. Bagaimana cara kerja dari protokol *routing* AODV dan DSDV pada topologi MANET?
2. Bagaimana performansi dari masing-masing protokol *routing* dalam memfasilitasi komunikasi antar *node* pada skenario MANET yang diujikan?

1.3 Tujuan

Tujuan yang ingin dicapai melalui penelitian ini adalah:

1. Mempelajari cara kerja dari protokol *routing* AODV dan DSDV pada jaringan MANET.
2. Mengetahui kelebihan dan kekurangan dari penggunaan AODV dan DSDV pada topologi MANET.
3. Mengetahui performansi dari protokol *routing* AODV dan DSDV terhadap skenario pengujian.

1.4 Manfaat

Manfaat dari dilakukannya penelitian ini antara lain:

1. Memahami cara kerja dari AODV dan DSDV.
2. Dapat mencari protokol *routing* yang paling sesuai dalam membangun aplikasi yang ditujukan pada jaringan MANET dengan karakteristik tertentu.

1.5 Batasan masalah

Batasan masalah dalam penelitian ini adalah:

1. Analisis kinerja AODV dan DSDV sebagai protokol komunikasi pada jaringan MANET dilihat dari data hasil simulasi dengan memperhatikan *Quality of Service* pada nilai *overhead routing*, *average end-to-end delay*, dan *packet delivery ratio*.
2. Data perbandingan dari protokol AODV dan DSDV didapatkan dari hasil simulasi yang dilakukan dengan Network Simulator 3.25 untuk menguji kinerja protokol AODV dan DSDV dengan perubahan jumlah *node*, mobilitas *node* jaringan dan luas jaringan.

1.6 Sistematika pembahasan

Skripsi nantinya akan disusun berdasarkan sistematika penulisan yang terdiri dari tujuh bab utama, antara lain :

Bab I Pendahuluan

Bab pendahuluan menguraikan latar belakang, rumusan masalah, tujuan, manfaat penelitian, batasan masalah, sistematika pembahasan hingga jadwal penelitian yang akan dilakukan.

Bab II Landasan Kepustakaan

Bab ini memuat kajian pustaka terkait penelitian-penelitian terdahulu yang berhubungan dengan topik protokol *routing* AODV dan DSDV pada MANET dan menunjukkan persamaan serta perbedaan antara skripsi yang dikerjakan dengan penelitian terdahulu. Pada bab ini juga disertakan landasan teori dari brbagai sumber pustaka yang berkaitan dengan teori dan metode yang

digunakan dalam penelitian, seperti definisi MANET, AODV, DSDV, alat simulasi jaringan, parameter pengukuran kinerja dan seterusnya.

Bab III Metodologi Penelitian

Bab ini menjelaskan metode penelitian yang digunakan untuk menyelesaikan masalah yang sudah dijabarkan sebelumnya pada rumusan masalah.

Bab IV Hasil

Bab ini menyajikan data hasil pelaksanaan penelitian dari simulasi yang telah dilakukan menggunakan Network Simulator – 3 (ns-3).

Bab V Pembahasan

Bab ini lebih lanjut menjelaskan makna dari hasil penelitian yang diperoleh untuk menjawab pertanyaan atau menyelesaikan masalah penelitian. Pembahasan juga menjelaskan pemahaman baru yang didapatkan dari hasil penelitian untuk pengembangan keilmuan.

Bab VI Penutup

Bab ini memuat kesimpulan dan saran terhadap skripsi.

Bab VII Daftar Pustaka

Bab daftar pustaka berisi sitasi-sitasi dan referensi literatur yang digunakan pada laporan skripsi.

BAB 2 LANDASAN KEPUSTAKAAN

Sebelumnya terdapat penelitian terdahulu dengan topik yang serupa, berikut perbandingannya dengan penelitian yang dilakukan oleh penulis pada Tabel 2.1 berikut ini:

Tabel 2.1 Penelitian terdahulu dengan topik yang serupa

Penulis	Judul Penelitian	Protokol *Routing*	Skenario Simulasi	Parameter yang diukur
Nurhayati Jiatmiko & Yudi Prayudi	Simulasi Jaringan MANET dengan NS3 Untuk Membandingkan Performa *Routing* Protokol AODV dan DSDV	AODV dan DSDV	Jumlah *node*: 20, 35 dan 50 *node* Model Propagasi: *TwoRayGround* Kecepatan *node*: 1 m/s dan 5 m/s Model Mobilitas: *Random Way Point* Luas Area Jaringan: 250x250 m Waktu Simulasi: 200 s Waktu Jeda: 0 s Ukuran Paket: 64 *byte*	*Average Throughput* *Delay* *Packet Delivery Ratio* *Packet Loss*
	Analisis Pengaruh Penggunaan Protokol *Routing* AODV, DSDV, dan ZRP pada Performansi Jaringan *Ad Hoc* Hibrid	AODV, DSDV dan ZRP	Skenario koneksi dan skenario mobilitas	*Average End to end Delay* *Packet Delivery Ratio* *Average Throughput* *Routing Overhead*
Faizal	Analisis	DSDV,	Jumlah *node*: 20,	*Packet*

Penulis	Judul Penelitian	Protokol *Routing*	Skenario Simulasi	Parameter yang diukur
Arif, Sofia Naning Hertiana, & Tody Ariefianto Wibowo	Performansi Protokol *Routing* DSDV, AODV dan DSR pada *Mobile Adhoc Network* Terhadap Model Pergerakan *Manhattan Grid*	AODV dan DSR	30, 40, 50 *node* Kecepatan *node*: 1, 2, 5 dan 10 m/s Model Mobilitas: *Manhattan Grid*	*Delivery Ratio* *Average Delay End-to-end* *Packet Loss* *Throughput*

Sementara penelitian yang dilakukan dirancang berdasarkan tabel dibawah ini:

Tabel 2.2 Rencana penelitian yang dilakukan

Penulis	Judul Penelitian	Protokol *Routing*	Skenario Simulasi	Parameter yang diukur
Tanudjaya Tri Ratna Sari	Analisis Kinerja Protokol *Routing Ad hoc On-Demand Distance Vector* (AODV) dan *Destination Sequence Distance Vector* (DSDV) pada *Mobile Ad Hoc Network* (MANET)	AODV dan DSDV	Jumlah *node*: 20, 50, 80 dan 100 *node* Model Propagasi: *TwoRayGround* Kecepatan maksimal tiap *node*: 20 m/s Model Mobilitas: *Random Way Point* Luas Area Jaringan: 200x200m, 500x500m, 800x800m, 1000x1000m, 1200x1200m Waktu Simulasi: 200 s	*Routing Overhead* *Average End-to-end Delay* *Packet Delivery Ratio*

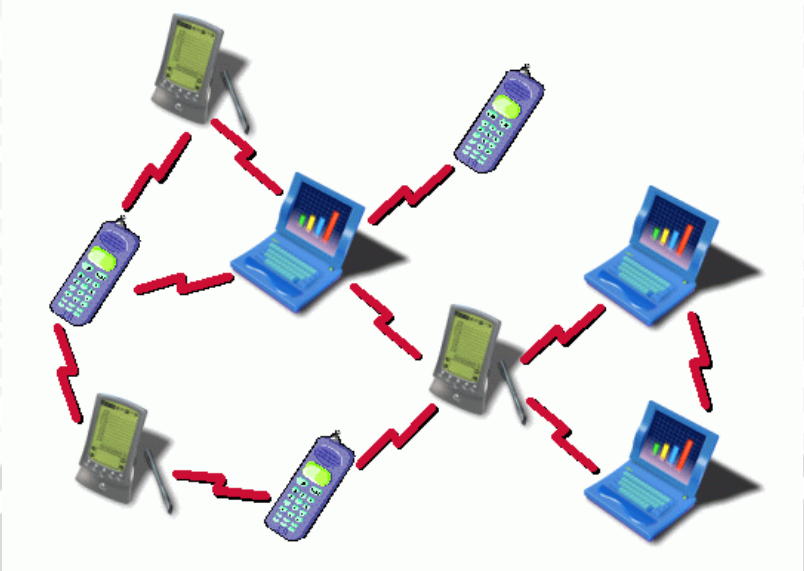
Penulis	Judul Penelitian	Protokol *Routing*	Skenario Simulasi	Parameter yang diukur
			Waktu Jeda: 0 s, 20 s, 50 s, 100 s Ukuran paket data: 512 *byte*	

2.1 *Mobile Ad Hoc Network* (MANET)

Ad hoc network merupakan salah satu kunci evolusi dari jaringan nirkabel. *Ad hoc network* biasanya terdiri dari *node* identik yang berkomunikasi lewat *wireless link* tanpa adanya kontrol pusat. Hingga saat ini penggunaan *ad hoc network* masih paling banyak pada komunikasi militer akan tetapi perkembangan minat komersial terhadap tipe jaringan ini terus bertambah.

Pada *mobile ad hoc network* masih terdapat permasalahan yang biasanya ditemui saat menggunakan komunikasi *wireless* dan *mobile*, yakni optimasi *bandwidth* dan kualitas transmisi data. Selain itu sifat *multi-hop* dan tidak adanya infrastruktur tetap menyebabkan masalah baru pada konfigurasi *advertising, discovery, maintenance, ad-hoc addressing* dan *self-routing.*

Sebuah MANET terdiri dari berbagai *mobile platform* (contoh: *router* dengan beberapa *host* dan alat komunikasi nirkabel lainnya), berikutnya disederhanakan sebagai *node*, yang bebas bergerak semaunya (Gambar 2.1). *Node* pada MANET dilengkapi dengan *transmitter* nirkabel dan *receiver* menggunakan *antenna* yang *omnidirectional (broadcast), highly-directional (point-to-point)* atau kombinasi keduanya (Corson, 1999).



Gambar 2.1 Ilustrasi MANET

Sumber: www.engr.iupui.edu

MANET juga memiliki karakteristik umum, diantaranya (Rana, 2013):

1. Topologi yang dinamik: *Node* bebas berpindah tempat sehingga topologi jaringan yang biasanya *multihop* akan berubah secara acak dan cepat pada waktu yang tidak dapat diprediksi dan bisa jadi terdiri dari *link* bidireksional dan *link* unidireksional secara bersamaan.
2. Multi-*hop routing*: *Node* yang ingin mengirim pesan ke *node* lainnya yang berada diluar jangkauan wilayah komunikasinya, maka pesan tersebut akan diteruskan lewat *intermediate node*.
3. Operasi terdistribusi: Karena tidak adanya *node* yang berperan sebagai kontrol pusat, maka semua *node* harus saling bekerjasama dan melakukan operasi sesuai perannya masing-masing agar dapat berkomunikasi.
4. *Autonomous terminal*: Setiap *node* berperan ganda sebagai *router* dan *host*.
5. Spesifikasi *node* yang ringan: Pada sebagian besar kasus, karena *node* bersifat *mobile* maka *node* memiliki kapabilitas CPU, kapasitas memori dan penyimpanan yang lebih sedikit dibandingkan *node* yang permanen.
6. Media fisik dipakai bersama: Media nirkabel yang digunakan untuk komunikasi dapat dijangkau oleh entitas apapun selama memiliki *resource* yang cocok dan memadai. Oleh karena itu akses ke *channel* komunikasi tidak dapat dibatasi.

2.2 *Ad Hoc On-Demand Distance Vector* (AODV) *Routing Protocol*

Salah satu *reactive routing protocol* yang paling sering digunakan pada jaringan MANET adalah AODV. *Ad hoc On-Demand Distance Vector (AODV) routing protocol* dibuat untuk *mobile protocol node* pada *ad hoc network*. AODV menawarkan adaptasi cepat terhadap kondisi *dynamic link*, minim proses dan *memory overhead*, minim utilisasi jaringan dan menentukan rute *unicast* ke destinasi dalam *ad hoc network*. AODV menggunakan nomor sekuen destinasi untuk memastikan tidak ada *loop* dan menghindari masalah yang biasanya dialami protokol *distance vector* pada umumnya (Zapata, 2002).

Protokol *routing Adhoc On-demand Distance Vector* bersifat reaktif, sehingga rute dicari hanya ketika dibutuhkan. Jika rute ke suatu destinasi tidak ditemukan pada tabel *routing* maka *node* akan mengirimkan paket *route request* (RREQ) ke seluruh *node* yang ada di jaringan melalui *broadcast*. Paket RREQ berisi alamat IP *node* sumber, nomor sekuen terbaru, *broadcast ID* dan nomor sekuen terbaru untuk destinasi yang diketahui oleh *node* sumber (Maan&Mazhar, 2011). *Node* yang dilewati paket RREQ dari *node* sumber tadi akan melakukan pembaharuan informasi mengenai *node* sumber. Jika terdapat *node* lain yang memiliki rute ke destinasi yang hendak dituju dengan nomor sekuen yang lebih tinggi daripada yang ada di paket RREQ, *node* tersebut akan mengirimkan paket route reply (RREP) ke *node* sumber, jika tidak maka paket RREQ akan terus di-broadcast hingga sampai ke destinasi. Saat *node* destinasi menerima paket dari *node* sumber, *node* destinasi akan mengirimkan paket RREP ke *node* sumber menggunakan jalur yang sama dengan jalur yang ditempuh paket RREQ untuk sampai ke destinasi. *Node* lain yang dilewati paket RREP tadi akan mencatat alamat IP dan *broadcast ID* dari sumber paket RREQ sehingga *node* yang dilewati tidak melakukan broadcast duplikasi dari paket RREQ. Jika ada jalur yang rusak pada rute yang aktif, *node* destinasi akan mengirim paket *route error* ke *node* sumber untuk menginformasikan bahwa rute tersebut sudah tidak valid lagi sehingga *node* sumber dapat melakukan pencarian terhadap rute baru ke *node* destinasi (Perkins, Belding-Royer, & Das, 2003).

2.3 *Destination-Sequenced Distance Vector Routing* (DSDV)

DSDV merupakan protokol *routing* proaktif yang menggunakan table *routing* untuk mengetahui informasi jalur-jalur yang ada pada MANET. DSDV dikembangkan oleh Charles E. Perkins dan Pravin Bhagwat pada tahun 1994 dengan memodifikasi algoritma *routing* Bellman-Ford sebagai protokol *routing* pada MANET.

Setiap *node* pada jaringan MANET memelihara table *routing* yang berisikan entri untuk setiap destinasi pada jaringan dan jumlah *hop* yang dibutuhkan untuk mencapai setiap destinasi. Setiap entri memiliki *sequence number* yang membantu untuk mengenali entri yang sudah tidak terpakai. Mekanisme ini membuat protokol dapat menghindari terbentuknya *routing loop*. Setiap *node* secara periodik mengirimkan informasi baru mengenai lokasinya melalui *broadcast* keseluruh jaringan. Rute yang dilabeli dengan *sequence number* terbaru sajalah yang akan dipakai oleh *node* lainnya untuk mencari rute dan mengirimkan data. Ketika *neighbor* dari *node* tersebut menerima *update*, mereka akan mengetahui bahwa lokasinya hanya satu *hop* dari *node* yang mengirimkan update tadi dan memperbaharui informasi ini dengan menambahkannya di *distance vector* yang mereka miliki. Setiap *node* menyimpan dan mengelola *next routing hop* untuk setiap *node* yang dapat mereka capai pada *routing* table mereka.

Jika salah satu *neighbor* A misalnya, mengetahui bahwa *node* B yang ada pada *routing* tablenya tidak lagi dapat dicapai, rute ke *node* B akan diperbaharui dengan metrik tak terhingga dan *sequence number* yang ditambah satu dari sequence number sebelumnya. Hal ini akan menyebabkan semua *node* yang memiliki rute ke *node* B melalui A ikut memperbaharui *routing* table mereka sesuai informasi dari *node* B tadi.

DSDV memperbaharui *routing table* menggunakan dua tipe paket *update*:

1. *Full dump*: Paket *update* ini mengandung seluruh informasi *routing* yang terdapat pada sebuah *node*. Sebagai gantinya beberapa *Network Protocol Data Unit* (NPDU) perlu ditransfer jika ukuran *routing* table tersebut cukup besar. Paket *full dump* hanya akan dikirimkan jika *node* sering mengalami pergerakan.
2. *Incremental*: Paket ini hanya memiliki informasi yang berbeda dari perubahan yang terjadi sejak paket *full dump* terakhir dikirimkan. Sehingga paket *incremental* hanya menggunakan sedikit *network resource* dibandingkan *full dump packet* (Soewito, 2014).

2.4 Network Simulator 3

Network Simulator 3 merupakan *discrete-event network simulator* untuk kebutuhan simulasi aplikasi, protokol, tipe jaringan, elemen-elemen jaringan, pemodelan jaringan dan pemodelan lalu lintas jaringan. Ns-3 adalah perangkat lunak gratis yang berlisensi dibawah lisensi GNU GPLv2 dan tersedia bagi publik untuk kebutuhan penelitian, edukasional, pengembangan dan pemakaian.

Proyek ns-3 dimulai pada tahun 2006 dan dikembangkan sebagai proyek *open-source*.

Ns-3 sendiri dikembangkan untuk menyediakan *platform* simulasi jaringan yang terbuka dan dapat diekstensi untuk keperluan pnelitian maupun pendidikan mengenai jaringan. Secara singkat ns-3 menyediakan model tentang bagaimana paket data jaringan bekerja dan menyediakan mesin simulasi bagi pengguna untuk melakukan eksperimen. Alasan lain untuk menggunakan ns-3 adalah agar dapat melakukan penelitian yang sulit atau bahkan tidak mungkin dilakukan menggunakan perangkat dan system di dunia nyata karena keterbatasan *resource* dan mempelajari perilaku system pada lingkungan yang sepenuhnya dapat dikontrol dan dapat diproduksi berulang kali serta mempelajari bagaimana jaringan-jaringan tadi bekerja.

Sebagai *tools* simulasi jaringan, ns-3 mempunyai model-model untuk semua elemen jaringan yang terdapat pada jaringan nyata. Elemen-elemen jaringan tersebut adalah (Wehler, 2010):

1. *Node*. Dalam istilah internet, perangkat komputer yang terhubung ke jaringan disebut *host*. Sedangkan pada ns-3 abstraksi perangkat komputer disebut *node*. Abstraksi ini diwakili dalam C++ dengan kelas *node*. Kelas *node* menyediakan metode untuk mengelola representasi perangkat komputasi di simulasi.

2. Aplikasi. Dalam ns-3 abstraksi dasar untuk program pengguna yang menghasilkan beberapa kegiatan yang akan disimulasikan adalah aplikasi. Abstraksi ini diwakili dalam C++ oleh kelas *Application*. Kelas *Application* menyediakan metode untuk mengelola representasi versi ns-3 pada aplikasi-aplikasi *level user* dalam simulasi. Pengembang diharapkan untuk mengkhususkan kelas *Application* dalam pengertian pemrograman berorientasi obyek untuk membuat aplikasi baru.

3. *Channel*. Media dimana aliran data dalam jaringan mengalir disebut *channel*. Dalam dunia simulasi ns-3, seseorang menghubungkan sebuah *node* ke objek yang mewakili sebuah saluran komunikasi. Di ns-3 abstraksi komunikasi dasar *subnetwork* disebut *channel* dan diwakili di C++ oleh kelas *channel.*

4. *Net Device*. Untuk terhubung dengan jaringan, komputer harus memiliki perangkat keras yang disebut dengan *peripheral card*. *Peripheral card* tersebut diimplementasikan beberapa fungsi jaringan, sehingga disebut *Network Interface Cards* (NICs). NIC tidak akan berfungsi tanpa sebuah

software driver untuk mengontrol perangkat keras tersebut. Pada Unix (atau Linux), sebuah *peripheral hardware* disebut sebagai *device. Device* dikontrol menggunakan *device driver*, dan NIC dikontrol menggunakan *network device driver* yang disebut dengan *net device*. Di ns-3, *net device* meliputi baik *software driver* dan simulasi *hardware.* Sebuah *net device* 'di-instalasi' pada sebuah *node* agar memungkinkan *node* untuk berkomunikasi dengan *node* lainnya dengan simulasi melalui *channels*. Abstraksi *net device* direpresentasikan dengan C++ oleh kelas *NetDevice*. Kelas *NetDevice* menyediakan metode untuk mengatur koneksi ke objek *node* dan *channel*.

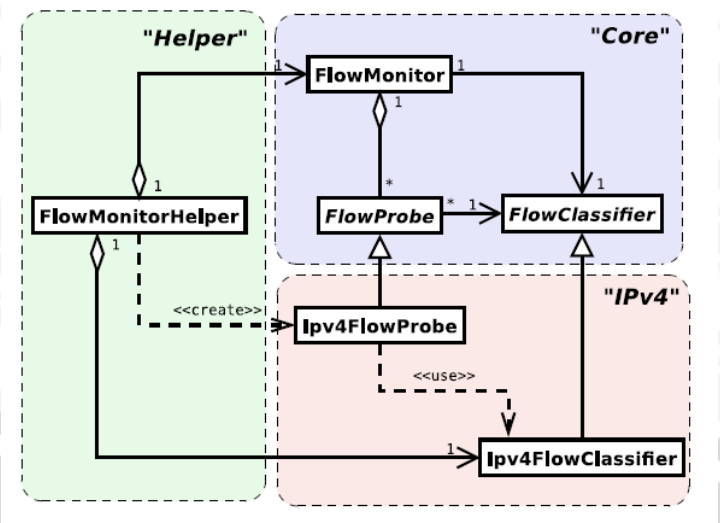
5. *Topology Helpers.* Dalam sebuah jaringan simulasi besar akan diperlukan banyak koneksi untuk mengatur antara *node*, *NetDevice* serta *channel.* ns-3 menyediakan apa yang disebut objek *Topology Helpers* untuk mengatur simulasi–simulasi jaringan semudah mungkin.

2.5 Flow Monitor

Flow Monitor adalah *network monitoring framework* yang digunakan untuk mendeteksi semua *flow* yang melewati jaringan dan menyimpan informasi metrik yang diperlukan untuk analisis seperti *throughput*, durasi, *delay*, ukuran paket, rasio *packet loss* dan lain-lain.

Kelas dasar dari Flow Monitor terdiri dari beberapa kelas seperti yang ditunjukkan pada Gambar 2.2 antara lain:

1. FlowMonitor
2. FlowProbe
3. FlowClassifier
4. FlowMonitorHelper



Gambar 2.2 Arsitektur dari FlowMonitor

Sumber: Carneiro, G., Fortuna, P., Ricardo, M., 2009. FlowMonitor—a network monitoring framework for the Network Simulator ns-3. Proceedings of NSTools.

Kelas FlowMonitor bertugas untuk mengkoordinasi hasil terkait *probe* dan mengumpulkan statistic *end-to-end flow*. Kelas FlowProbe bertugas untuk mendengarkan packet event dalam poin spesifik dari *space* yang disimulasikan kemudian melaporkan *event* tadi ke kelas global FlowMonitor dan mengumpulkan statistik *flow* terkait *probe* miliknya sendiri. Terakhir kelas FlowClassifier yang menyediakan metode untuk menerjemahkan paket data mentah tadi ke abstrak parameter *flow-identifier* dan *packet-identifier*. *Identifier* tadi merupakan bilangan integer *unsigned 32-bit* yang unik bagi setiap *flow* dan paket yang berada dalam *flow* tadi untuk keseluruhan simulasi. *Identifier* ini yang digunakan untuk komunikasi antara kelas FlowProbe dan FlowMonitor untuk menjaga arsitektur utama agar tetap bersifat umum dan tidak terpengaruh metode pengangkapan maupun klasifikasi paket tertentu (Carneiro, Fortuna, & Ricardo, 2013).

2.6 AWK

AWK adalah sebuah bahasa pemrograman yang dikembangkan pada tahun 1977 oleh Drs. A. Aho, P. Weinberger, dan B. Kernighan. AWK merupakan bahasa pemroses teks dan pencocokan pola, biasa disebut sebagai bahasa berbasis data – statemen program AWK mengolah data input dengan mencocokkan lalu proses daripada serangkaian langkah program seperti bahasa pemrograman lainnya. Program dengan bahasa AWK mencari input dengan mencocokkan *record* dengan suatu pola tertentu kemudian melakukan aksi yang sudah ditentukan terhadap *record* tersebut hingga program mencapai akhir dari data input.

Program dengan bahasa AWK sangat baik untuk tugas yang berhubungan dengan database dan data dalam tabel, misalnya analisis data dari beberapa data set (Stutz, 2006).

2.7 Protocol Control Overhead

Protocol control overhead ialah jumlah keseluruhan dari paket *routing* yang ditransmisikan selama simulasi. Untuk paket yang dikirimkan melalui banyak *hop*, setiap transmisi dari sebuah paket (setiap *hop*) dihitung sebagai satu transmisi (Dirganto, 2010).

2.8 Average End-to-end Delay

Average end-to-end delay atau *one-way delay* menunjukkan waktu rata-rata yang dibutuhkan oleh sebuah paket untuk ditransmisikan di jaringan dari *node* sumber ke *node* tujuan.

2.9 Packet Delivery Ratio

Merupakan rasio jumlah data paket yang terkirim ke *node* tujuan dengan jumlah data paket yang dikirim dari *node* sumber.

2.10 Model Mobilitas

Mobilitas dari tiap *node* menciptakan lingkungan yang sangat dinamis yang menjadi salah satu tantangan terbesar bagi jaringan *ad hoc.* Karena *node* bersifat *mobile,* pergerakan relatif antar *node* membuat atau memecah koneksi *wireless* sehingga menyebabkan perubahan topologi jaringan. Protokol *routing* harus cukup cerdas dalam menghadapi dinamika tersebut dan mampu mendukung komunikasi antar *node* tanpa merusak performansi. Karenanya model mobilitas memegang peran penting dalam melakukan evaluasi dari sebuah protokol *routing* pada jaringan *ad hoc*.

Model mobilitas mendefinisikan karakter dinamik dari pergerakan node. Penggunaan model mobilitas yang dapat meniru pergerakan *node* seperti pada aplikasi dunia nyata sangat penting karena performansi dari protokol *routing* sangat bergantung pada pola mobilitas. Terdapat banyak model pergerakan yang tersedia untuk jaringan *ad hoc* seperti *Random Walk, Random Way Point (RWP),* dan *Reference Point Group Mobility (RPGM).* Pada penelitian ini akan dibahas model mobilitas *Random Way Point* secara lebih mendalam karena RWP adalah model mobilitas yang dipakai saat simulasi (Kumar, 2009).

2.11 Random Way Point

Pada model mobilitas RWP, setiap node secara independen memilih destinasi yang hendak dituju secara acak didalam batasan jaringan dan bergerak menuju destinasi dengan kecepatan konstan. Kecepatan dipilih secara acak dari distribusi umum antara 0 hingga v_{max}. Setelah *node* mencapai destinasinya, *node* tersebut akan berhenti pada waktu jeda yang ditentukan dan kemudian akan memilih destinasi dan kecepatan secara acak lagi untuk bergerak ke destinasi baru. Proses ini terus diulang selama waktu simulasi (Kumar, 2009).

2.12 Model Propagasi: *Two Ray Ground*

Model propagasi radio yang diimplementasikan pada Network Simulator 3 antara lain: *free sace model, two-ray ground reflection model,* dan *shadowing model.* Pada penelitian ini model ropagasi radio yang digunakan ialah *Two Ray Ground.* Model propagasi ini sendiri digunakan untuk memprediksi kekuatan sinyal dari tiap paket. Pada *physical layer* dari setiap *wireless node* akan terdapat *threshold* penerimaan. Setiap kali paket diterima dan kekuatan sinyalnya dibawah *threshold* penerimaan, maka paket tersebut akan ditandai sebagai paket *error* dan di-*drop* oleh *MAC layer.*

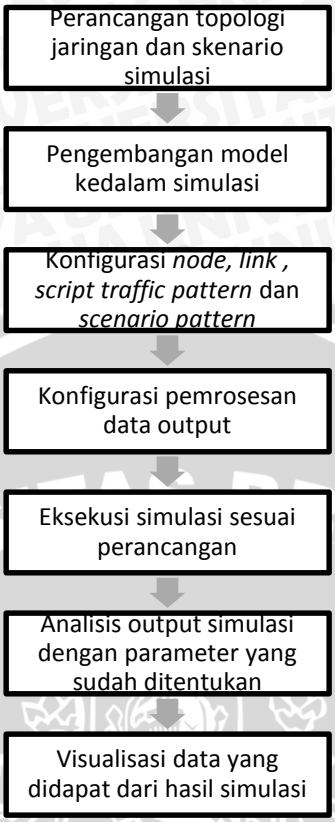
Model *two-ray ground reflection* mempertimbangkan baik *direct path* maupun *ground reflection path*. Model ini memberi prediksi yang lebih u=akurat pada jarak jauh jika dibandingkan dengan model *free space* (Henderson, 2011).

BAB 3 METODOLOGI

Tipe penelitian ini merupakan penelitian non-implementatif analitik, dimana akan dilakukan eksperimen penggunaan protokol *routing* AODV dan DSDV lewat simulasi sebagai landasan pengambilan kesimpulan. Penelitian akan dilakukan di laboratorium jaringan komputer FILKOM, Universitas Brawijaya. Teknik pengumpulan data yang mendukung penelitian adalah dengan melakukan studi dokumen berupa jurnal dan skripsi serta menganalisis hasil simulasi.

Adapun metodologi yang digunakan dalam penelitian ini meliputi beberapa tahapan sebagai berikut: (1) Identifikasi masalah, pada tahap ini dilakukan identifikasi terhadap permasalahan yang ada, (2) Studi literatur, dimana literatur-literatur diambil dari penelitian-penelitian sebelumnya maupun dari jurnal-jurnal ilmiah, baik dalam negeri maupun luar negeri dan dari beberapa buku, (3) Analisis protokol komunikasi AODV dann DSDV pada jaringan MANET lewat simulasi, (4) Evaluasi hasil perbandingan. Evaluasi ini dilakukan untuk melihat seberapa efektif suatu protocol dalam memfasilitasi komunikasi antar *node* pada MANET, dan (5) Penarikan kesimpulan, dimana hasil evaluasi pada tahap 4 akan digunakan sebagai dasar untuk penarikan kesimpulan.

Sementara untuk simulasi jaringan akan dilakukan dengan tahapan yang dapat dilihat pada Gambar 3.1 dibawah ini:



Gambar 3.3 Gambaran Tahapan Simulasi

3.1 Studi Literatur

Pada penelitian ini sebagian besar teori terkait protokol *routing* AODV dan DSDV serta MANET diambil dari berbagai jurnal dan skripsi terkait dengan pengembangan protokol komunikasi untuk jaringan *ad hoc*.

3.2 Analisis Kebutuhan

Perangkat keras yang dibutuhkan dalam penelitian ini adalah laptop atau PC dengan spesifikasi setara:

Prosesor	: Intel Core i3 CPU M 370 @ 2.40 GHz
RAM	: 6 GB DDR3
Sistem Operasi	: Windows 7 / Ubuntu 12.04
Simulator	: Network Simulator 3.25.

3.3 Pengumpulan Data

Pengumpulan data dilakukan dengan berbagai metode:

1. Studi Pustaka

Studi pustaka dilakukan dengan membaca berbagai literatur baik berupa buku, jurnal dan hasil konferensi yang berkaitan dengan teknologi terkini dari protokol komunikasi jaringan *wireless*.

2. Eksplorasi Internet

Mengambil hasil penelitian dari pengembangan protokol AODV dan DSDV dari berbagai organisasi.

3.4 Perancangan Simulasi

Simulasi dilakukan dengan Network Simulator 3.25 dan performansi *average end to end delay* serta *packet delivery ratio* diukur menggunakan Flow Monitor. Sementara untuk *routing packet* dihitung menggunakan awk *script* dari *trace file* berupa *.tr *file* yang dihasilkan oleh Network Simulator 3.25.

Topologi MANET yang akan disimulasi dengan ns-3 dibuat menurut daftar parameter pada Tabel 2.2 dibawah ini.

Tabel 3.3 Parameter Topologi Jaringan

Parameter	Nilai
Jumlah *node*	20, 50, 80 dan 100
Model Propagasi	*TwoRayGround*
Kecepatan maksimal tiap *node*	20 m/s
Model Mobilitas	*Random Way Point*
Luas Area Jaringan	100x100m, 200x200 m, 500x500 m, 800x800 m, 1000x1000m, 1200x1200m
Waktu Simulasi	200 s
Waktu Jeda	0 s, 20 s, 50 s, 80 s, 100 s
Ukuran paket data	512 *byte*

Untuk memulai simulasi dengan ns-3 diperlukan program ns-3 yang ditulis dalam bahasa pemrograman C++ sesuai topologi jaringan dan skenario yang hendak disimulasikan. Pada simulasi ini kode program diambil dari contoh yang tersedia pada folder ns-3.25/examples/*routing*/manet-*routing*-compare.cc yang

kemudian topologi, *traffic data*, *trace file* dan outputnya disesuaikan dengan skenario yang hendak diuji.

Konfigurasi topologi jaringan mulai dari *node*, *channel*, *device* dan mode mobilitas.

```
167   int nWifis = 20;
168   double TotalTime = 200.0;
169   std::string rate ("2048bps");
170   std::string phyMode ("DsssRate11Mbps");
171   std::string tr_name ("manet");
172   int nodeSpeed = 20; //in m/s
173   int nodePause = 0; //in s
174   m_protocolName = "protocol";
176   Config::SetDefault
("ns3::OnOffApplication::PacketSize",StringValue ("512"));
177   Config::SetDefault ("ns3::OnOffApplication::DataRate",
StringValue (rate));
178   Config::SetDefault
("ns3::WifiRemoteStationManager::NonUnicastMode",StringValue
(phyMode));
```

Penjelasan :

Baris 167 : Menginisiasi jumlah *node* pada jaringan MANET

Baris 168 : Menginisiasi total waktu simulasi

Baris 169&177 : Menginisiasi *application rate* dalam mengirim data dengan nilai *default*

Baris 170 &178 : Menginisiasi *phy mode* sesuai standar IEEE 802.11b

Baris 171 : Menginisiasi nama *trace file* yang akan dihasilkan dari simulasi

Baris 172 : Menentukan kecepatan maksimal dari tiap *node*

Baris 173 : Menentukan *pause time* atau waktu jeda bagi setiap *node* pada jaringan

Baris 174 : Menginisiasi variable m_protocolName dengan variabel protokol yang akan menunjukkan tipe protokol yang akan dipakai

Baris 176 : Menentukan ukuran paket sebesar 512 *byte* sebagai nilai *default*

```
206   ObjectFactory pos;
207   pos.SetTypeId ("ns3::RandomRectanglePositionAllocator");
208   pos.Set ("X", StringValue
("ns3::UniformRandomVariable[Min=0.0|Max=500.0]"));
209   pos.Set ("Y", StringValue
("ns3::UniformRandomVariable[Min=0.0|Max=500.0]"));
```

Penjelasan :

Baris 206-209 : Menentukan luas area jaringan dengan membatasi posisi perpindahan *node* secara acak

```
226   AodvHelper aodv;
227   DsdvHelper dsdv;
228
229   Ipv4List*Routing*Helper list;
230   InternetStackHelper internet;
245       internet.Set*Routing*Helper (list);
246       internet.Install (adhoc*Node*s);
```

Penjelasan :

Baris 226-227 : Protokol *helper* untuk AODV dan DSDV

Baris 229&245 : Konfigurasi *routing* protocol

Baris 230&246 : Menambahkan *internet stack* ke *node* yang berada di jaringan

```
250   Ipv4AddressHelper addressAdhoc;
251   addressAdhoc.SetBase ("10.1.1.0", "255.255.255.0");
252   Ipv4InterfaceContainer adhocInterfaces;
253   adhocInterfaces = addressAdhoc.Assign (adhocDevices);
```

Penjelasan :

Baris 250-253 : Konfigurasi alamat IPv4 ke tiap *node* yang ada

```
267       Ptr<UniformRandomVariable> var =
CreateObject<UniformRandomVariable> ();
268       ApplicationContainer temp = onoff1.Install
(adhoc*Node*s.Get (i + nSinks));
269       temp.Start (Seconds (var->GetValue (50.0,51.0)));
270       temp.Stop (Seconds (TotalTime));
```

Penjelasan :

Baris 267&269 : Menentukan waktu mulai pengiriman paket data oleh aplikasi secara acak pada detik simulasi ke-50 atau 51

Baris 268 : Menempatkan aplikasi ke *node* yang bertugas untuk mengirim data

Baris 269 : Pengiriman paket data dihentikan pada saat simulasi berakhir

```
305   Simulator::Stop (Seconds (TotalTime));
306   Simulator::Run ();
```

Penjelasan :

Baris 305 : Simulasi akan dihentikan pada detik ke-200

Baris 306 : Memerintahkan simulasi untuk dijalankan

```
296   FlowMonitorHelper flowmon;
297   Ptr<FlowMonitor> monitor = flowmon.InstallAll();
```

Penjelasan :

Baris 296-297 : Mengaktifkan FlowMonitor untuk me*monitoring* aliran paket dari *node* satu ke *node* lainnya didalam jaringan

```
344 totDelaySum += iter->second.delaySum.GetSeconds();
345 totRcvdPackets += iter->second.rxPackets;
346 totSentPackets += iter->second.txPackets;
347 avgThroughput += iter->second.rxBytes * 8.0 /(iter->second.timeLastRxPacket.GetSeconds()-iter->second.timeFirstTxPacket.GetSeconds()) / 1000;
348 lostPackets += iter->second.lostPackets;
349 totRecievedBytes += iter->second.rxBytes;
350 totalTime += iter->second.timeLastRxPacket.GetSeconds()-iter->second.timeFirstTxPacket.GetSeconds();
```

Penjelasan :

Baris 345-350 : Menghitung statistik performansi seperti *delay,* jumlah paket yang diterima, jumlah paket yang dikirim, *throughput*, jumlah paket yang hilang, jumlah *byte* data yang diterima dan waktu yang dibutuhkan paket untuk sampai ke destinasi dengan Flow Monitor sesuai *flow* yang ingin dianalisis.

Untuk menjalankan program yang sudah dibuat, pindahkan *file* manet.cc ke folder *scratch* yang berada di folder ns-3.25, kemudian masukkan perintah berikut:

```
1 ./waf –run scratch/manet
```

Untuk menyunting program tersebut dapat dilakukan dengan *tools* gedit atau *text processing tools* lainnya yang didukung oleh Linux. Pada penelitian ini kami menggunakan *tools* gedit dengan perintah sebagai berikut:

```
1 gedit scratch/manet.cc
```

Setelah dijalankan program manet.cc dengan Flow Monitor akan mencetak setiap *flow* yang terjadi selama simulasi seperti yang dapat dilihat pada Gambar 3.4.1 dibawah ini.

```
Flow ID: 1 Src Addr 10.1.1.18 Dst Addr 10.1.1.8
Tx Packets = 74
Rx Packets = 72
Throughput: 2.08028 Kbps
end-to-end delays = 1.14702
Flow Time = 146.014
```

Gambar 3.4 Output Tiap *Flow Statistic* dari Flow Monitor

Kemudian setelah selesai mencetak tiap *flow statistic* tadi, di akhir program akan mencetak hasil perhitungan PDR dan *average end to end delay* dari simulasi.

```
Recv Packets = 737
Sent Packets = 740
PDR = 0.995946
Avg end-to-end delay = 0.00474166
Avg Goodput = 2.18052
Avg lost packets = 0.15
```

Gambar 3.5 Perhitungan Akhir Flow Monitor

Untuk menghitung jumlah paket *routing*, proses *trace file* yang dihasilkan selama simulasi dengan awk *script* untuk masing-masing protokol yang digunakan dengan perintah berikut.

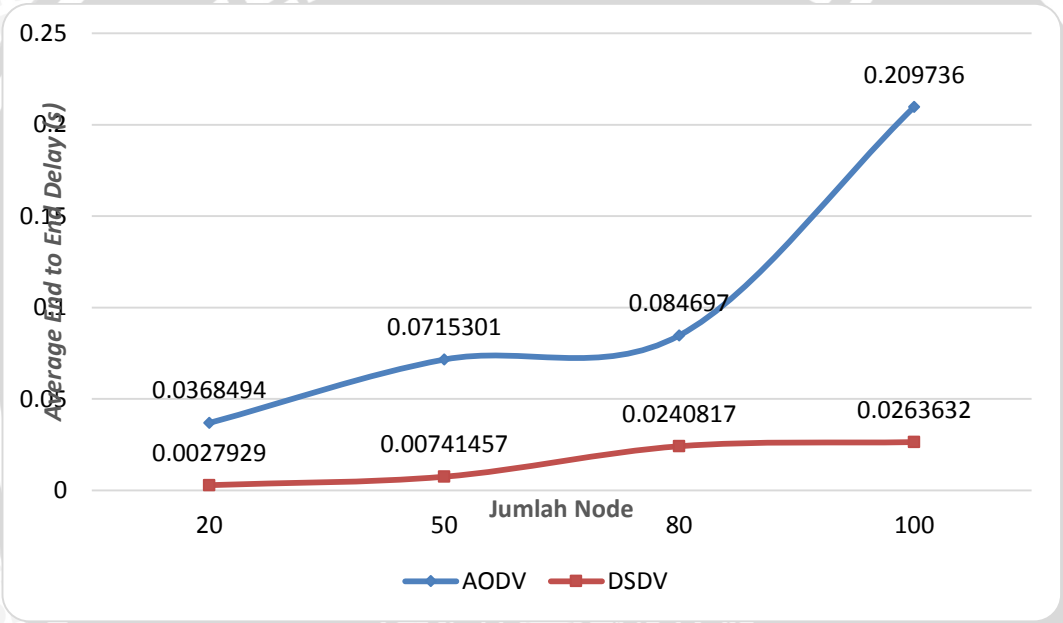
```
awk –f aodv.awk tracefile-name.tr
awk –f dsdv.awk tracefile-name.tr
```

BAB 1 HASIL SIMULASI DAN ANALISA

Pada penelitian ini analisis performansi dilakukan di MANET dengan melakukan perubahan pada tiga parameter, yakni jumlah *node*, waktu jeda dan luas area jaringan, sementara parameter lainnya dibiarkan bernilai konstan. Hasil simulasi dapat dilihat pada uraian berikut.

1.1 Analisis Performansi terhadap Jumlah *Node*

Pada skenario pertama untuk mengetahui performansi protokol terhadap beban jaringan, simulasi dijalankan dengan jumlah *node* sebanyak 20 *node*, 50 *node,* 80 *node* dan 100 *node* secara bergantian menggunakan protokol *routing* AODV dan DSDV dengan luas area jaringan seluas 500x500m dan waktu jeda 0 s.

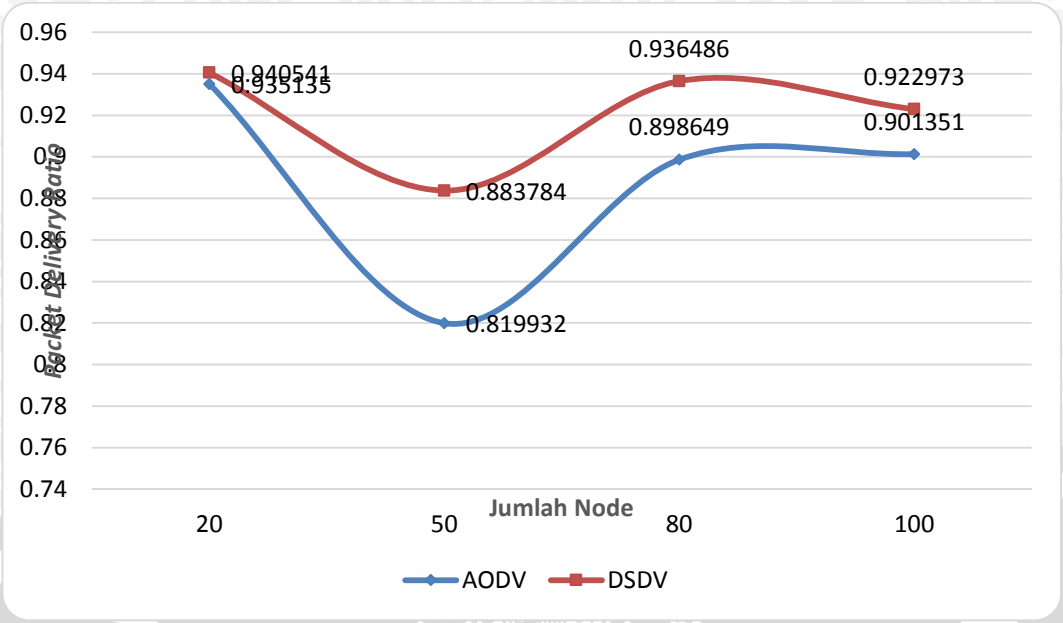


Gambar 4.6 Pengaruh Jumlah *Node* terhadap *Average End to End Delay*

Pada Gambar 4.1 dapat dilihat bahwa secara umum performa DSDV lebih baik dibandingkan AODV pada pertambahan jumlah *node*. Hal ini ditunjukkan dengan nilai *average end to end delay* dari protokol DSDV yakni 0,0027929 s dengan skenario jumlah *node* sebanyak 20 *node*, lebih rendah dari AODV dengan 0,0368494 s pada skenario yang sama. Begitu pula pada skenario dengan 50 *node*, dimana DSDV dengan nilai rata-rata *end to end delay* sebesar 0,00741457s, jauh lebih rendah dari AODV dengan nilai 0,0715301 s. Hal ini juga terjadi pada skenario dengan jumlah *node* sebanyak 80 *node* dan 100 *node*.

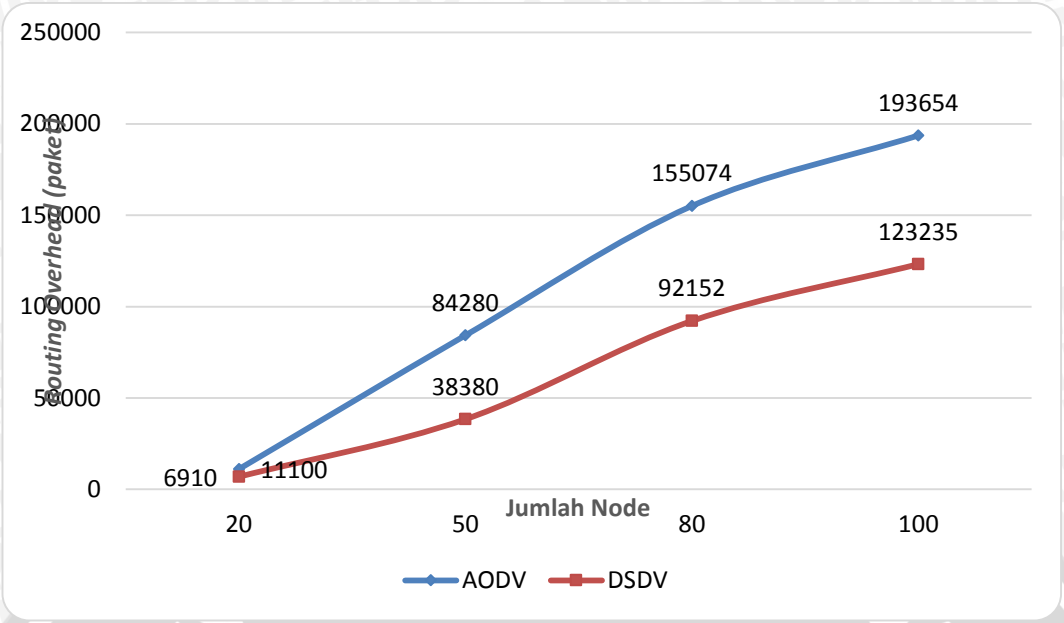
Dari Gambar 4.1 diatas dapat disimpulkan bahwa pada AODV, nilai *average end to end delay* semakin meningkat seiring pertambahan *node*, hal ini

menandakan bahwa beban jaringan berupa banyaknya *node* yang aktif sangat mempengaruhi waktu yang dibutuhkan paket untuk sampai ke tujuan.



Gambar 4.7 Pengaruh Jumlah *Node* terhadap *Packet Delivery Ratio*

Jika dilihat dari Gambar 4.2 performa DSDV juga lebih baik dari AODV dalam rasio jumlah paket yang diterima dengan jumlah paket yang dikirim. Hal ini terlihat dari nilai PDR protokol DSDV sebesar 0,940541 yang lebih tinggi dari nilai PDR protokol AODV sebesar 0,935135 pada skenario 20 *node.* Kemudian pada skenario 50 *node*, PDR dari protokol DSDV berada pada 0,883784, masih lebih tinggi dari protokol AODV dengan nilai PDR 0,819932. Berikutnya pada skenario 80 *node*, nilai PDR dari DSDV kembali mengungguli AODV dengan 0,936486. Kemudian dari scenario 100 *node*, nilai PDR dari AODV masih lebih rendah jika dibandingkan dengan DSDV dengan nilai PDR 0,922973. Dari Gambar 4.2 terlihat kedua protokol tersebut menunjukkan fluktuasi nilai PDR terhadap jumlah *node* yang ada pada jaringan. Hal ini mungkin juga dipengaruhi mode pergerakan dari *node* pada simulasi MANET yang dikonfigurasikan menjadi *Random Waypoint.*



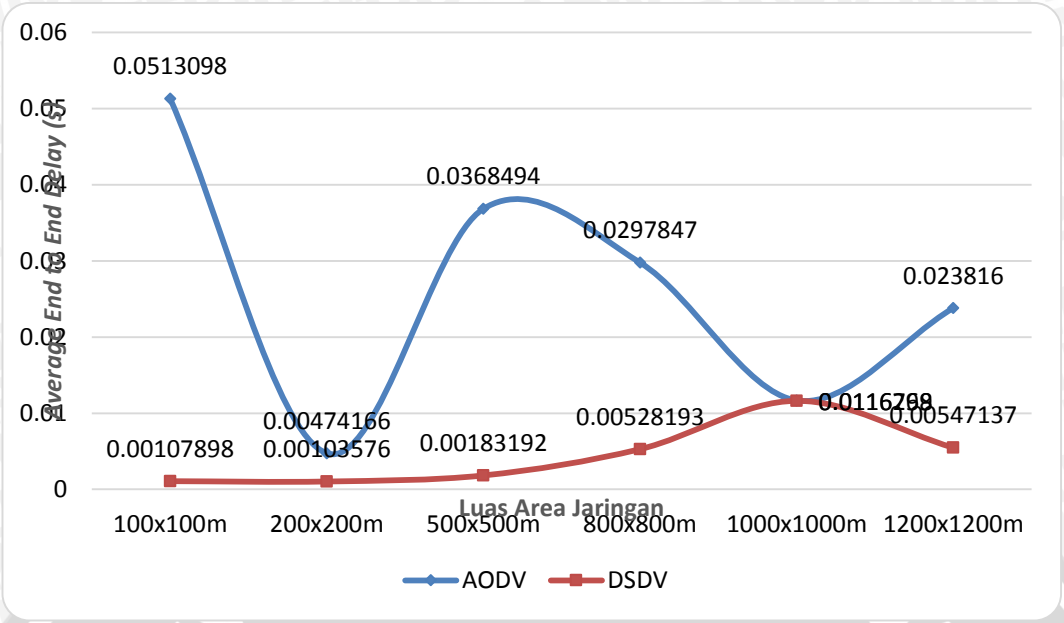
Gambar 4.8 Pengaruh Jumlah *Node* terhadap *Routing Overhead*

Sedangkan pada Gambar 4.3 menunjukkan jumlah *routing* overhead DSDV yang lebih sedikit dibandingkan AODV secara keseluruhan. Misalnya pada skenario dengan jumlah *node* sebanyak 20, jumlah paket *routing* dari AODV adalah 11100 paket, sedangkan DSDV hanya memerlukan 6910 paket. Berikutnya pada skenario dengan jumlah *node* 50, 80 dan 100 *node*, jumlah paket *routing* dari AODV ialah 84280 paket, 155074 paket, 193654 paket secara berurutan. Jumlah ini lebih besar dibandingkan dengan jumlah paket *routing* dari DSDV yang hanya 38380 paket, 92152 paket dan 123235 paket secara berurutan untuk skenario yang sama. Hal ini sejalan dengan karakteristik AODV yang bersifat reaktif sehingga *node* terus menerus mentransmisikan paket *routing* untuk memperbaharui informasi mengenai *neighbor* yang ada didekatnya setiap kali topologi MANET berubah.

Dapat disimpulkan bahwa pada jaringan dengan jumlah *node* yang meningkat, performansi DSDV secara umum lebih baik dibandingkan AODV.

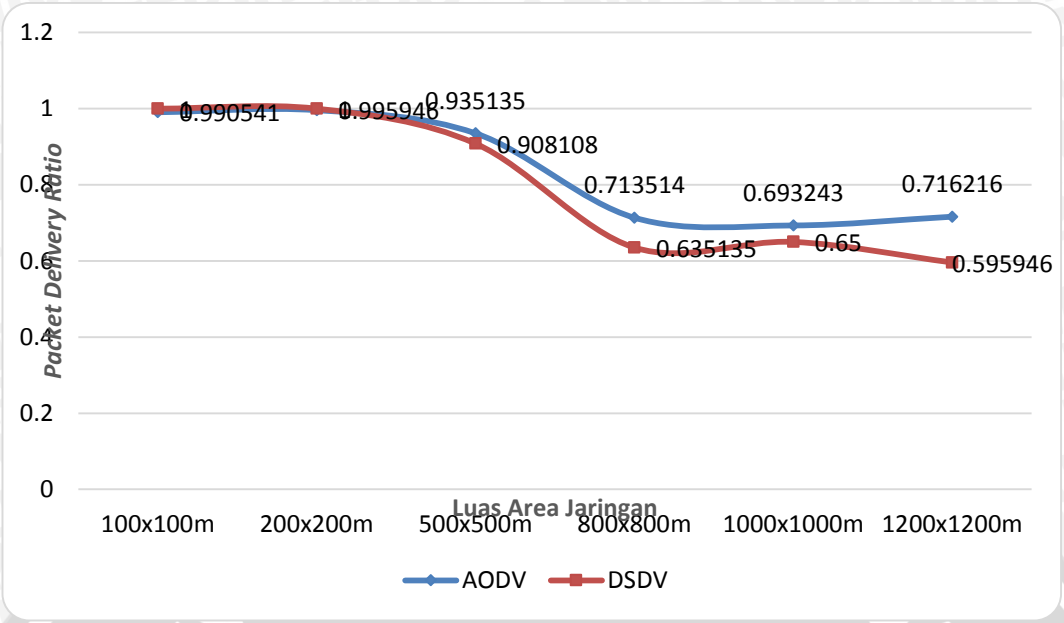
1.2 Analisis Performansi terhadap Luas Area Jaringan

Pada skenario kedua untuk mengetahui performansi protokol terhadap luas area jaringan, simulasi dijalankan dengan ukuran jaringan sebesar 100x100m, 200x200m, 500x500m, 800x800m, 1000x1000m dan 1200x1200m secara bergantian menggunakan protokol *routing* AODV dan DSDV. Sementara untuk parameter lainnya seperti jumlah *node* dan waktu jeda diberi nilai konstan sebesar 20 *node* dan 0 s.



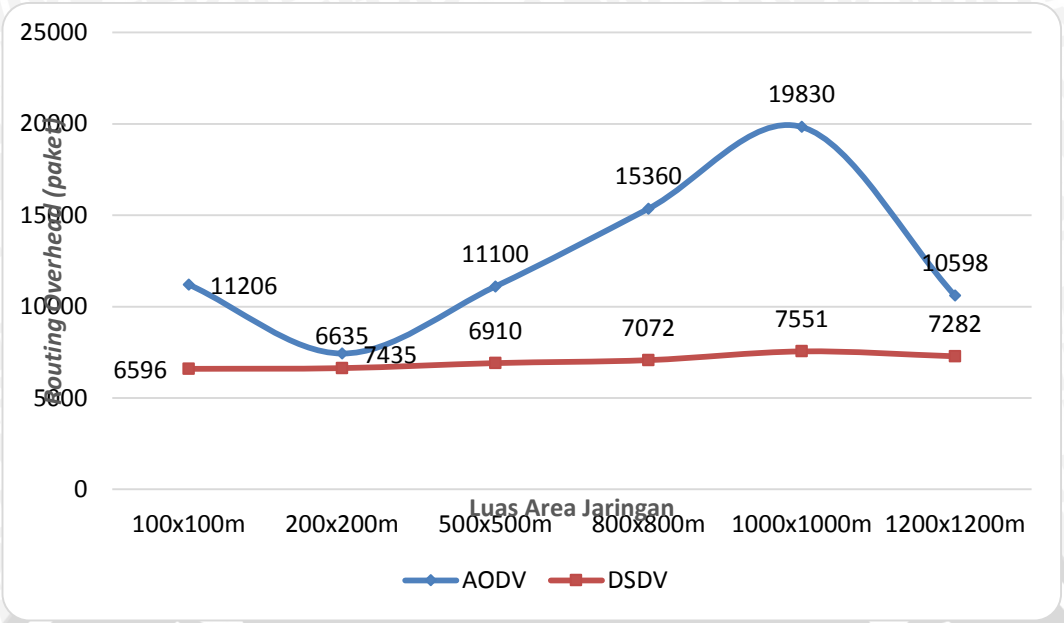
Gambar 4.9 Pengaruh Luas Area Jaringan terhadap *Average End to End Delay*

Sesuai Gambar 4.4 nilai *Average end to end delay* pada DSDV lebih rendah daripada AODV jika area jaringan diperluas. Fakta ini dapat ditelusuri dari nilai *average end to end delay* dari protokol AODV yang berkisar di angka 0,0513098 s, 0,00474166 s, 0,0368494 s, 0,0297847 s, 0,0116769 s dan 0,023816 s untuk masing-masing skenario dengan area jaringan seluas 100x100m, 200x200m, 500x500m, 800x800m, 1000x1000m dan 1200x1200m secara berurutan. Nilai-nilai tersebut lebih tinggi dari nilai *average end to end delay* protokol DSDV pada skenario yang sama, yakni 0,00107898 s, 0,00103576 s, 0,00183192 s, 0,00528193 s, 0,0116298 s dan 0,00547137 s. Hal ini menunjukkan bahwa DSDV lebih unggul dibandingkan AODV dilihat dari nilai *Average end to end delay* pada peningkatan luas area jaringan.



Gambar 4.10 Pengaruh Luas Area Jaringan terhadap *Packet Delivery Ratio*

Gambar 4.5 menunjukkan nilai PDR yang lebih baik oleh DSDV daripada AODV pada luas jaringan 100x100m dan 200x200m dengan nilai PDR yang bernilai 1 di kedua skenario tersebut. Akan tetapi keadaan berbalik pada luas jaringan 500x500m, 800x800m, 1000x1000m dan 1200x1200m justru AODV yang lebih unggul dengan masing-masing nilai PDR sebesar 0,935135, 0,713514, 0,693243 dan 0,716216 dibandingkan DSDV dengan nilai PDR sebesar 0,908108, 0,635135, 0,65, 0,595946 untuk skenario yang sama.



Gambar 4.11 Pengaruh Luas Area Jaringan terhadap *Routing Overhead*

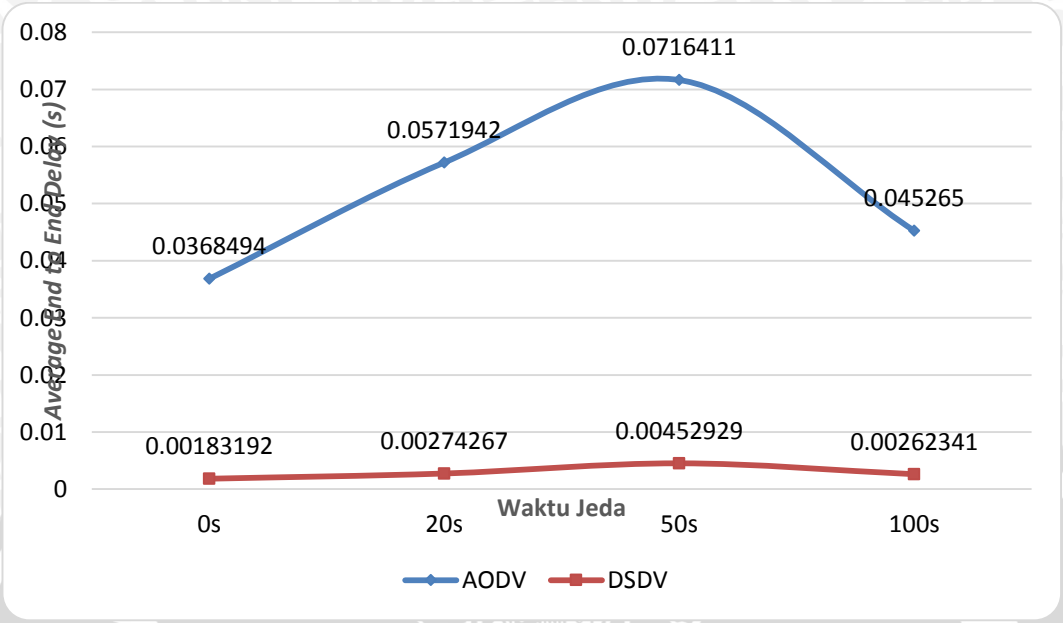
Nilai *routing* overhead pada protokol AODV secara keseluruhan lebih tinggi dari DSDV seperti yang ditunjukkan oleh Gambar 4.2.3 diatas. Saat luas area jaringan seluas 100x100m, jumlah paket *routing* dari AODV sebesar 11206 paket, angka ini lebih besar jika dibandingkan dengan DSDV yang hanya 6596 paket. Pada luas area jaringan dengan luas 200x200m, nilai PDR antara AODV dan DSDV dengan 7435 paket dan 6635 paket. Sedangkan pada luas area jaringan 500x500m, nilai PDR dari AODV berada di angka 11100 paket dan DSDV di angka 6910 paket.

Untuk luas area jaringan seluas 800x800m, nilai PDR dari AODV adalah sebesar 15360 paket dan dari DSDV sebesar 7072 paket. Pada luas area jaringan seluas 1000x1000m, nilai PDR dari AODV adalah 19380 paket lebih besar dari protokol DSDV dengan 7551 paket. Keadaan ini masih sama pada skenario dengan luas area jaringan seluas 1200x1200m, walaupun jumlah paket *routing* AODV mengalami penurunan yang tajam menjadi 105898 paket, akan tetapi DSDV masih lebih unggul dengan jumlah paket *routing* sebanyak 7282 paket. Dalam performansi terhadap luas area jaringan dari nilai *routing overhead* ini, DSDV dianggap lebih unggul daripada AODV.

1.3 Analisis Performansi terhadap Mobilitas Jaringan

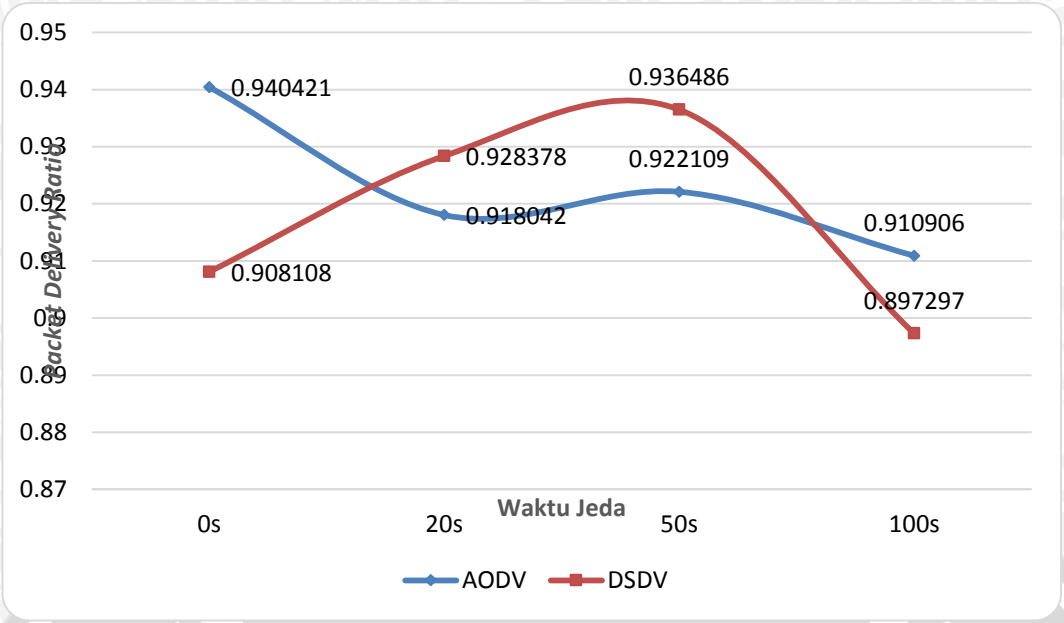
Pada skenario terakhir untuk mengetahui performansi protokol terhadap mobilitas jaringan, simulasi dijalankan dengan mengubah waktu jeda menjadi 0s, 20s, 50 s, 100s pada masing-masing protokol *routing* AODV dan DSDV. Setiap

node memiliki kecepatan maksimal 20m/s untuk bergerak dengan leluasa di area jaringan seluas 500x500m degan jumlah *node* sebanyak 20 *node*.



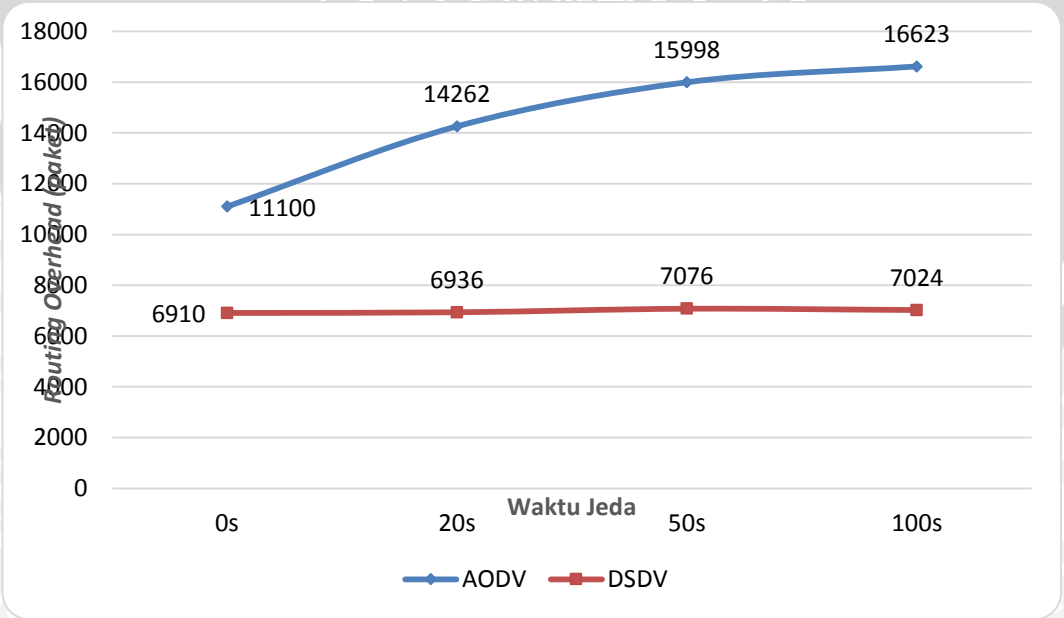
Gambar 4.12 Pengaruh Mobilitas Jaringan terhadap *Average End to End Delay*

Nilai *Average end to end delay* pada AODV lebih tinggi dibandingkan dengan DSDV pada seluruh skenario perubahan waktu jeda. Nilai *average end to end delay* dari AODV dengan waktu jeda 0 s misalnya, berada di angka 0,0368494 s, nilai ini lebih tinggi dari DSDV yang hanya 0,00183192s. Hal ini juga terjadi pada skenario-skenario berikutnya dimana DSDV memimpin dengan nilai *average end to end delay* yang lebih rendah pada setiap simulasi yang dilakukan. Sesuai dengan Gambar 4.7 performa DSDV lebih baik daripada AODV dan konsisten dengan nilai *Average end to end delay* yang rendah tanpa terpengaruh mobilitas jaringan yang berubah-ubah.



Gambar 4.13 Pengaruh Mobilitas Jaringan terhadap *Packet Delivery Ratio*

Pada Gambar 4.8 secara umum nilai PDR dari AODV lebih baik dibandingkan DSDV dengan rata-rata PDR 0,9228695 dari nilai rata-rata dari DSDV yang hanya 0,91756725. Akan tetapi pada skenario dengan waktu jeda 20 s dan 50 s, nilai PDR dari DSDV sebesar 0,928378 dan 0,936486 sedikit lebih baik jika dibandingkan dengan AODV yang nilai PDRnya sebesar 0.918042 dan 0,922109.



Gambar 4.14 Pengaruh Mobilitas Jaringan terhadap *Routing Overhead*

Pada Gambar 4.9 nilai RO dari protokol *routing* DSDV lebih rendah dari AODV pada waktu jeda 0s, 20s, 50s dan 100s dengan masing-masing jumlah paket sebanyak 6910 paket, 6936 paket, 7076 paket dan 7024 paket untuk DSDV dan

11100 paket, 14262 paket, 15998 paket dan 16623 paket untuk AODV. Hal ini menunjukkan performa DSDV yang lebih baik dari AODV jika dilihat dari jumlah paket *routing* yang ditransmisikan untuk mencari rute ke destinasi.

BAB 2 PENUTUP

2.1 Kesimpulan

Pada penelitian ini ada tiga ukuran performansi yang digunakan, antara lain *routing overhead*, *packet delivery ratio* dan *average end to end delay* dengan skenario yang terdiri dari perubahan jumlah *node*, waktu jeda dan luas area jaringan yang berbeda untuk mensimulasi beban jaringan, mobilitas jaringan dan luas jaringan.

Pada simulasi MANET terhadap jumlah *node* dengan mengubah jumlah *node* yang ada pada jaringan didapatkan hasil bahwa performansi protokol DSDV lebih baik dari protokol AODV. Pada setiap hasil performansi dari nilai *average end to end delay* hingga *routing overhead*, protokol DSDV lebih unggul dibanding AODV. Selanjutnya pada simulasi MANET terhadap luas area jaringan, secara umum performansi DSDV lebih baik dari AODV. Akan tetapi pada luas area jaringan yang lebih luas dari 500x500m, nilai PDR dari DSDV lebih rendah dari AODV. Terakhir pada simulasi MANET terhadap mobilitas jaringan, performansi protokol DSDV kembali mengungguli protokol AODV pada nilai *average end to end delay* dan *routing overhead*. Nilai PDR dari AODV pada waktu jeda 0 s atau pada mobilitas yang tinggi lebih baik dari DSDV dengan nilai PDR sebesar 0,940421. Akan tetapi nilai tersebut terus menurun pada waktu jeda 20 s dan 50 s dengan nilai PDR dari DSDV yang mengungguli lewat selisih yang sangat kecil dengan nilai PDR sebesar 0,928378 dan 0,936486 dibandingkan AODV dengan nilai PDR 0,918042 dan 0,922109.

Dari hasil simulasi diatas dapat disimpulkan bahwa DSDV secara umum lebih unggul dari AODV dilihat dari nilai *average end to end delay*, *packet delivery ratio* dan *routing overhead*nya. DSDV juga lebih cocok diaplikasikan pada jaringan *ad hoc* yang memiliki mobilitas rendah karena nilai PDR-nya yang tinggi dan lebih stabil dibandingkan AODV tanpa terpengaruh dengan peningkatan jumlah *node*. Akan tetapi jika diaplikasikan pada luas area jaringan yang lebih luas atau mobilitas yang tinggi, performansi AODV lebih baik dibandingkan DSDV.

2.2 Saran

Untuk mengembangkan penelitian ini, analisis performansi dapat dilakukan terhadap tipe *ad hoc network* lainnya yang dapat mengaplikasikan AODV dan DSDV seperti VANET dan WANET. Selain itu metrik pengukuran performansi dapat ditambah dengan *throughput*, *jitter*, *power consumption* sesuai kebutuhan penelitian. Simulasi juga dapat dilakukan dengan menambah parameter dalam

skenario misalnya terhadap variasi model mobilitas seperti *random walk* dan *steadystate random waypoint*.

DAFTAR PUSTAKA

Bansal, M., Rajput, R. and Gupta, G., 1999. *Mobile ad hoc networking (MANET): Routing protocol performance issues and evaluation considerations*. The internet society.

Carneiro, G., Fortuna, P., Ricardo, M., 2009. FlowMonitor—a network monitoring framework for the Network Simulator ns-3. Proceedings of NSTools.

Dirganto, S. C., & Husni, M., 2010. *Analisis Kinerja Protokol Routing AODV dan OLSR pada Jaringan Mobile Ad hoc.* Tersedia melalui: Perpustakaan Digital Institut Teknologi Surabaya <https://digilib.its.ac.id> [Diakses 19 Maret 2016]

Giordano, S. 2002. Mobile ad hoc networks. *Handbook of wireless networks and mobile computing*, 325-346.

Henderson, T., 2011. 18. Radio Propagation Models. Tersedia melalui: < http://www.isi.edu/nsnam/ns/doc/node216.html> [Diakses 20 Juli 2016]

Jayakumar, G., & Gopinath, G., 2007. Ad hoc mobile *wireless* networks *routing* protocols–a review. *J. Comput. Sci*, *3*(8), 574-582.

Kumar, T., 2009. Performance Evaluation of AODV and OLSR Under Mobility. Thesis for the degree of Master of Science.

Maan, F., & Mazhar, N., 2011. MANET *Routing* Protocols vs Mobility Models: A Performance Evaluation. *2011 Third International Conference on Ubiquitous and Future Networks (ICUFN)*, IEEE, 179-184.

Mauve, M., Widmer, J., & Hartenstein, H., 2001. A survey on position-based *routing* in mobile ad hoc networks. *Network, IEEE*, *15*(6), 30-39.

Nesargi, S., & Prakash, R., 2002. MANETconf: Configuration of hosts in a mobile ad hoc network. In *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE* (Vol. 2, pp. 1059-1068). IEEE.

Perkins, C., Belding-Royer, E. and Das, S., 2003. *Ad hoc on-demand distance vector (AODV) routing* (No. RFC 3561).

Rana, A., & Gupta, S., 2013. *Review on MANETs Characteristics, Challenges, Application and Security Attacks.* International Journal of Science and Research(IJSR).

Samar, P., Pearlman, M. R., & Haas, Z. J., 2004. Independent zone *routing*: an adaptive hybrid *routing* framework for ad hoc *wireless* networks. *IEEE/ACM Transactions on Networking (TON)*, *12*(4), 595-608.

Soewito, B.. *Destination-Sequenced Distance Vector Routing (DSDV)*. Tersedia melalui: Journal CommIT <http://mti.binus.ac.id/> [Diakses 16 Juli 2016]

Stutz, M., 2006. *Get started with GAWK: AWK language fundamentals*. IBM.

Wehrle, K., Günes, M., & Gross, J. (Eds.)., 2010. *Modeling and tools for network simulation*. Springer Science & Business Media.

Zapata, M.G., 2002. Secure ad hoc on-demand distance vector *routing*. *ACM SIGMOBILE Mobile Computing and Communications Review*, *6*(3), pp.106-107.

LAMPIRAN A HASIL SIMULASI

LAMPIRAN B Average End-to-end Delay

Jumlah Node	AODV	DSDV
20	0.036849	0.002793
50	0.07153	0.007415
80	0.084697	0.024082
100	0.209736	0.026363
Luas Area Jaringan		
100x100m	0.05131	0.001079
200x200m	0.004742	0.001036
500x500m	0.036849	0.001832
800x800m	0.029785	0.005282
1000x1000m	0.011677	0.01163
1200x1200m	0.023816	0.005471
Waktu Jeda		
0s	0.036849	0.001832
20s	0.057194	0.002743
50s	0.071641	0.004529
100s	0.045265	0.002623

LAMPIRAN C Packet Delivery Ratio

Jumlah Node	AODV	DSDV
20	0.935135	0.940541
50	0.819932	0.883784
80	0.898649	0.936486
100	0.901351	0.922973
Area Jaringan		
100x100m	0.990541	1
200x200m	0.995946	1
500x500m	0.935135	0.908108
800x800m	0.713514	0.635135
1000x1000m	0.693243	0.65
1200x1200m	0.716216	0.595946
Waktu jeda		
0s	0.940421	0.908108
20s	0.918042	0.928378
50s	0.922109	0.936486
100s	0.910906	0.897297

LAMPIRAN D Overhead Routing

Jumlah Node	AODV	DSDV
20	11100	6910
50	84280	38380
80	155074	92152

100	193654	123235
Area Jaringan		
100x100m	11206	6596
200x200m	7435	6635
500x500m	11100	6910
800x800m	15360	7072
1000x1000m	19830	7551
1200x1200m	10598	7282
Waktu Jeda		
0s	11100	6910
20s	14262	6936
50s	15998	7076
100s	16623	7024

LAMPIRAN E SCRIPT SIMULASI

LAMPIRAN F Script Simulasi manet.cc

```
1  /* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil;
2  -*- */
3  // GPLv2 Licence
4
5  //Memasukkan modul dan header file yang diperlukan selama
   simulasi
6  #include <fstream>
7  #include <iostream>
8  #include "ns3/core-module.h"
9  #include "ns3/network-module.h"
10 #include "ns3/internet-module.h"
11 #include "ns3/mobility-module.h"
12 #include "ns3/wifi-module.h"
13 #include "ns3/aodv-module.h"
14 #include "ns3/dsdv-module.h"
15 #include "ns3/applications-module.h"
16 #include "ns3/flow-monitor-module.h"
17 #include "ns3/trace-helper.h"
18 #include "ns3/stats-module.h"
19
20 //Gunakan namespace ns3 project
21 using namespace ns3;
22
23 //Enable dan disable console message logging dengan
   referensi ke nama program
24 NS_LOG_COMPONENT_DEFINE ("manet");
25
26 class RoutingExperiment
27 {
28 public:
29   RoutingExperiment ();
30   void Run (int nSinks, double txp, std::string
   CSVfileName);
31   std::string CommandSetup (int argc, char **argv);
32
33 private:
34   Ptr<Socket> SetupPacketReceive (Ipv4Address addr,
   Ptr<Node> node);
35   void ReceivePacket (Ptr<Socket> socket);
36   void CheckThroughput ();
37
38   uint32_t port;
39   uint32_t bytesTotal;
40   uint32_t packetsReceived;
41
```

```
42    std::string m_CSVfileName;
43    int m_nSinks;
44    std::string m_protocolName;
45    double m_txp;
46    bool m_traceMobility;
47    uint32_t m_protocol;
48  };
49
50  RoutingExperiment::RoutingExperiment ()
51    : port (9),
52      bytesTotal (0),
53      packetsReceived (0),
54      m_CSVfileName ("manet.csv"),
55      m_traceMobility (false),
56      m_protocol (1) //jenis protokol yang digunakan
57  {
58  }
59
60  static inline std::string
61  PrintReceivedPacket (Ptr<Socket> socket, Ptr<Packet> packet)
62  {
63    SocketAddressTag tag;
64    bool found;
65    found = packet->PeekPacketTag (tag);
66    std::ostringstream oss;
67
68    oss << Simulator::Now ().GetSeconds () << " " << socket-
  >GetNode ()->GetId ();
69
70    if (found)
71      {
72        InetSocketAddress addr =
  InetSocketAddress::ConvertFrom (tag.GetAddress ());
73        oss << " received one packet from " << addr.GetIpv4
  ();
74      }
75    else
76      {
77        oss << " received one packet!";
78      }
79    return oss.str ();
80  }
81
82  void
83  RoutingExperiment::ReceivePacket (Ptr<Socket> socket)
84  {
85    Ptr<Packet> packet;
86    while ((packet = socket->Recv ()))
87      {
```

```
88        bytesTotal += packet->GetSize ();
89        packetsReceived += 1;
90        NS_LOG_UNCOND (PrintReceivedPacket (socket, packet));
91      }
92  }
93
94  void
95  RoutingExperiment::CheckThroughput ()
96  {
97    double kbs = (bytesTotal * 8.0) / 1000;
98    bytesTotal = 0;
99
100   std::ofstream out (m_CSVfileName.c_str (), std::ios::app);
101
102   out << (Simulator::Now ()).GetSeconds () << ","
103       << kbs << ","
104       << packetsReceived << ","
105       << m_nSinks << ","
106       << m_protocolName << ","
107       << m_txp << ""
108       << std::endl;
109
110   out.close ();
111   packetsReceived = 0;
112   Simulator::Schedule (Seconds (1.0),
  &RoutingExperiment::CheckThroughput, this);
113 }
114
115 Ptr<Socket>
116 RoutingExperiment::SetupPacketReceive (Ipv4Address addr,
  Ptr<Node> node)
117 {
118   TypeId tid = TypeId::LookupByName
  ("ns3::UdpSocketFactory");
119   Ptr<Socket> sink = Socket::CreateSocket (node, tid);
120   InetSocketAddress local = InetSocketAddress (addr, port);
121   sink->Bind (local);
122   sink->SetRecvCallback (MakeCallback
  (&RoutingExperiment::ReceivePacket, this));
123
124   return sink;
125 }
126
127 std::string
128 RoutingExperiment::CommandSetup (int argc, char **argv)
129 {
130   CommandLine cmd;
131   cmd.AddValue ("CSVfileName", "The name of the CSV output
  file name", m_CSVfileName);
```

```
132   cmd.AddValue ("traceMobility", "Enable mobility tracing", m_traceMobility);
133   cmd.AddValue ("protocol", "1=AODV;2=DSDV", m_protocol);
134   cmd.Parse (argc, argv);
135   return m_CSVfileName;
136 }
137
138 int
139 main (int argc, char *argv[])
140 {
141   RoutingExperiment experiment;
142   std::string CSVfileName = experiment.CommandSetup (argc,argv);
143   std::ofstream out (CSVfileName.c_str ());
144   out << "SimulationSecond," <<
145   "ReceiveRate," <<
146   "PacketsReceived," <<
147   "NumberOfSinks," <<
148   "RoutingProtocol," <<
149   "TransmissionPower" <<
150   std::endl;
151   out.close ();
152
153   int nSinks = 10;
154   double txp = 7.5;
155
156   experiment.Run (nSinks, txp, CSVfileName);
157 }
158
159 void
160 RoutingExperiment::Run (int nSinks, double txp, std::string CSVfileName)
161 {
162   Packet::EnablePrinting ();
163   m_nSinks = nSinks;
164   m_txp = txp;
165   m_CSVfileName = CSVfileName;
166
167   int nWifis = 20;
168   double TotalTime = 200.0;
169   std::string rate ("2048bps");
170   std::string phyMode ("DsssRate11Mbps");
171   std::string tr_name ("manet");
172   int nodeSpeed = 20; //in m/s
173   int nodePause = 0; //in s
174   m_protocolName = "protocol";
175
176   Config::SetDefault ("ns3::OnOffApplication::PacketSize",StringValue ("512"));
```

```
177   Config::SetDefault ("ns3::OnOffApplication::DataRate",
      StringValue (rate));
178   Config::SetDefault
      ("ns3::WifiRemoteStationManager::NonUnicastMode",StringValue
      (phyMode));
179
180   NodeContainer adhocNodes;
181   adhocNodes.Create (nWifis);
182
183   WifiHelper wifi;
184   wifi.SetStandard (WIFI_PHY_STANDARD_80211b);
185
186   YansWifiPhyHelper wifiPhy =  YansWifiPhyHelper::Default
      ();
187   YansWifiChannelHelper wifiChannel;
188   wifiChannel.SetPropagationDelay
      ("ns3::ConstantSpeedPropagationDelayModel");
189   wifiChannel.AddPropagationLoss
      ("ns3::FriisPropagationLossModel");
190   wifiPhy.SetChannel (wifiChannel.Create ());
191
192   WifiMacHelper wifiMac;
193   wifi.SetRemoteStationManager
      ("ns3::ConstantRateWifiManager",
194                                    "DataMode",StringValue
      (phyMode),
195                                    "ControlMode",StringValue
      (phyMode));
196
197   wifiPhy.Set ("TxPowerStart",DoubleValue (txp));
198   wifiPhy.Set ("TxPowerEnd", DoubleValue (txp));
199
200   wifiMac.SetType ("ns3::AdhocWifiMac");
201   NetDeviceContainer adhocDevices = wifi.Install (wifiPhy,
      wifiMac, adhocNodes);
202
203   MobilityHelper mobilityAdhoc;
204   int64_t streamIndex = 0;
205
206   ObjectFactory pos;
207   pos.SetTypeId ("ns3::RandomRectanglePositionAllocator");
208   pos.Set ("X", StringValue
      ("ns3::UniformRandomVariable[Min=0.0|Max=500.0]"));
209   pos.Set ("Y", StringValue
      ("ns3::UniformRandomVariable[Min=0.0|Max=500.0]"));
210
211   Ptr<PositionAllocator> taPositionAlloc = pos.Create ()-
      >GetObject<PositionAllocator> ();
212   streamIndex += taPositionAlloc->AssignStreams
      (streamIndex);
213
214   std::stringstream ssSpeed;
```

```
215   ssSpeed << "ns3::UniformRandomVariable[Min=0.0|Max=" << nodeSpeed << "]";
216   std::stringstream ssPause;
217   ssPause << "ns3::ConstantRandomVariable[Constant=" << nodePause << "]";
218   mobilityAdhoc.SetMobilityModel ("ns3::RandomWaypointMobilityModel",
219                                   "Speed", StringValue (ssSpeed.str ()),
220                                   "Pause", StringValue (ssPause.str ()),
221                                   "PositionAllocator", PointerValue (taPositionAlloc));
222   mobilityAdhoc.SetPositionAllocator (taPositionAlloc);
223   mobilityAdhoc.Install (adhocNodes);
224   streamIndex += mobilityAdhoc.AssignStreams (adhocNodes, streamIndex);
225
226   AodvHelper aodv;
227   DsdvHelper dsdv;
228
229   Ipv4ListRoutingHelper list;
230   InternetStackHelper internet;
231
232   switch (m_protocol)
233     {
234     case 1:
235       list.Add (aodv, 100);
236       m_protocolName = "AODV";
237       break;
238     case 2:
239       list.Add (dsdv, 100);
240       m_protocolName = "DSDV";
241       break;
242     default:
243       NS_FATAL_ERROR ("No such protocol:" << m_protocol);
244     }
245       internet.SetRoutingHelper (list);
246       internet.Install (adhocNodes);
247
248   NS_LOG_INFO ("assigning ip address");
249
250   Ipv4AddressHelper addressAdhoc;
251   addressAdhoc.SetBase ("10.1.1.0", "255.255.255.0");
252   Ipv4InterfaceContainer adhocInterfaces;
253   adhocInterfaces = addressAdhoc.Assign (adhocDevices);
254
255   OnOffHelper onoff1 ("ns3::UdpSocketFactory",Address ());
256   onoff1.SetAttribute ("OnTime", StringValue ("ns3::ConstantRandomVariable[Constant=1.0]"));
```

```
257   onoff1.SetAttribute ("OffTime", StringValue
      ("ns3::ConstantRandomVariable[Constant=0.0]"));
258
259
260   for (int i = 0; i < nSinks; i++)
261     {
262       Ptr<Socket> sink = SetupPacketReceive
      (adhocInterfaces.GetAddress (i), adhocNodes.Get (i));
263
264       AddressValue remoteAddress (InetSocketAddress
      (adhocInterfaces.GetAddress (i), port));
265       onoff1.SetAttribute ("Remote", remoteAddress);
266
267       Ptr<UniformRandomVariable> var =
      CreateObject<UniformRandomVariable> ();
268       ApplicationContainer temp = onoff1.Install
      (adhocNodes.Get (i + nSinks));
269       temp.Start (Seconds (var->GetValue (50.0,51.0)));
270       temp.Stop (Seconds (TotalTime));
271     }
272
273   std::stringstream ss;
274   ss << nWifis;
275   std::string nodes = ss.str ();
276
277   std::stringstream ss2;
278   ss2 << nodeSpeed;
279   std::string sNodeSpeed = ss2.str ();
280
281   std::stringstream ss3;
282   ss3 << nodePause;
283   std::string sNodePause = ss3.str ();
284
285   std::stringstream ss4;
286   ss4 << rate;
287   std::string sRate = ss4.str ();
288
289   NS_LOG_INFO ("Configure Tracing.");
290   tr_name = tr_name + "_" + m_protocolName +"_" + nodes +
      "nodes_" + sNodeSpeed + "speed_" + sNodePause + "pause_" +
      sRate + "rate";
291
292   AsciiTraceHelper ascii;
293   Ptr<OutputStreamWrapper> osw = ascii.CreateFileStream (
      (tr_name + ".tr").c_str());
294   wifiPhy.EnableAsciiAll (osw);
295
296   FlowMonitorHelper flowmon;
297   Ptr<FlowMonitor> monitor = flowmon.InstallAll();
298
```

```
299
300
301   NS_LOG_INFO ("Run Simulation.");
302
303   CheckThroughput ();
304
305   Simulator::Stop (Seconds (TotalTime));
306   Simulator::Run ();
307
308   monitor->CheckForLostPackets ();
309   Ptr<Ipv4FlowClassifier> classifier = DynamicCast<Ipv4FlowClassifier> (flowmon.GetClassifier ());
310   std::map<FlowId, FlowMonitor::FlowStats> stats = monitor->GetFlowStats ();
311
312 double avgThroughput = 0;
313 double lostPackets = 0;
314 double totRecievedBytes = 0;
315 double totalTime = 0;
316 double totRcvdPackets = 0;
317 double totSentPackets = 0;
318 double totDelaySum = 0;
319
320 for (std::map<FlowId, FlowMonitor::FlowStats>::const_iterator iter = stats.begin (); iter != stats.end (); ++iter)
321     {
322
323
324 NS_LOG_UNCOND("Flow ID: " << iter->first << " Src Addr " << t.sourceAddress << " Dst Addr " << t.destinationAddress);
325 NS_LOG_UNCOND("Tx Packets = " << iter->second.txPackets);
326 NS_LOG_UNCOND("Rx Packets = " << iter->second.rxPackets);
327 NS_LOG_UNCOND("Throughput: " << iter->second.rxBytes * 8.0 / (iter->second.timeLastRxPacket.GetSeconds()-iter->second.timeFirstTxPacket.GetSeconds()) / 1024  << " Kbps");
328 NS_LOG_UNCOND("end-to-end delays = " << iter->second.delaySum.GetSeconds());
329 avgDelay += (iter->second.delaySum.GetSeconds()/iter->second.txPackets);
330 NS_LOG_UNCOND("Flow Time = " << iter->second.timeLastRxPacket.GetSeconds()-iter->second.timeFirstTxPacket.GetSeconds());
331
332 if ((t.sourceAddress == "10.1.1.11" && t.destinationAddress == "10.1.1.1" && t.destinationPort == 9) ||
333 (t.sourceAddress == "10.1.1.12" && t.destinationAddress == "10.1.1.2" && t.destinationPort == 9) ||
334 (t.sourceAddress == "10.1.1.13" && t.destinationAddress == "10.1.1.3" && t.destinationPort == 9) ||
335 (t.sourceAddress == "10.1.1.14" && t.destinationAddress == "10.1.1.4" && t.destinationPort == 9) ||
```

```
336 (t.sourceAddress == "10.1.1.15" && t.destinationAddress == "10.1.1.5" && t.destinationPort == 9) ||
337 (t.sourceAddress == "10.1.1.16" && t.destinationAddress == "10.1.1.6" && t.destinationPort == 9) ||
338 (t.sourceAddress == "10.1.1.17" && t.destinationAddress == "10.1.1.7" && t.destinationPort == 9) ||
339 (t.sourceAddress == "10.1.1.18" && t.destinationAddress == "10.1.1.8" && t.destinationPort == 9) ||
340 (t.sourceAddress == "10.1.1.19" && t.destinationAddress == "10.1.1.9" && t.destinationPort == 9) ||
341 (t.sourceAddress == "10.1.1.20" && t.destinationAddress == "10.1.1.10" && t.destinationPort == 9))
342
343 {
344 totDelaySum += iter->second.delaySum.GetSeconds();
345 totRcvdPackets += iter->second.rxPackets;
346 totSentPackets += iter->second.txPackets;
347 avgThroughput += iter->second.rxBytes * 8.0 /(iter->second.timeLastRxPacket.GetSeconds()-iter->second.timeFirstTxPacket.GetSeconds()) / 1000;
348 lostPackets += iter->second.lostPackets;
349 totRecievedBytes += iter->second.rxBytes;
350 totalTime += iter->second.timeLastRxPacket.GetSeconds()-iter->second.timeFirstTxPacket.GetSeconds();
351 }
352
353 }
354 std::cout<<"Recv Packets = " << totRcvdPackets<< std::endl;
355 std::cout<<"Sent Packets = " << totSentPackets<< std::endl;
356 std::cout<<"PDR = " << totRcvdPackets/totSentPackets<< std::endl;
357 avgDelay /= nWifis;
358 avgThroughput /= nWifis;
359 lostPackets /= nWifis;
360 double avgGoodput = totRecievedBytes*8/totalTime/1000;
361 double avgEndtoEndDelay = totDelaySum/totRcvdPackets;
362 std::cout<<"Avg end-to-end delay = " << avgEndtoEndDelay<<std::endl;
363 std::cout<<"Avg Goodput = " << avgGoodput<<std::endl;
364 std::cout<<"Avg lost packets = " << lostPackets<<std::endl;
365
366   Simulator::Destroy ();
367 }
```

LAMPIRAN G Script aodv.awk

```
1 BEGIN {
2 sent=0
3 }
4 {
```

```
5  action=$1
6  time=$2
7  idipv4=$29
8  src=$40
9  aodvPacket=$50
10 #menghitung jumlah paket routing yang dikirim
11 if ( action == "t" && ( aodvPacket == "(RREP)" || aodvPacket == "(RREQ)" || aodvPacket == "(RRER)"))
12 {sent++;}
13 }
14 END {
15 printf("Routing overhead AODV = %.0f\n", sent);
16 }
```

LAMPIRAN H Script dsdv.awk

```
1  BEGIN {
2  sent=0
3  }
4  {
5  action=$1
6  time=$2
7  idipv4=$29
8  dsdvPacket=$49
9  #menghitung jumlah paket routing yang dikirim/di-forward
10 if ( action == "t" && ( dsdvPacket == "ns3::dsdv::DsdvHeader"))
11 {sent++;}
12 }
13 END {
14 printf("Routing overhead DSDV = %.0f\n", sent);
15 }
```