Implementasi *Routing Protocol* DSR pada Skenario *Mobility Random Waypoint* dengan menggunakan Propagasi Nakagami

Hasbi As Shiddi Qi, Radityo Anggoro, Muchammad Husni
Departemen Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember (ITS)
e-mail: onggo.informatika@gmail.com

***Abstrak*—*Mobile Ad hoc Network* (MANET) merupakan jaringan *wireless* yang berasal dari kumpulan *mobile node* yang topologinya dapat berubah dengan cepat dan kapan saja. Aspek yang penting dalam MANET adalah protokol rute dimana protokol inilah yang mengatur sistem pencarian rute paket data dalam jaringan tersebut. Ada beberapa macam protokol rute pada MANET salah satunya adalah DSR. DSR merupakan pengembangan dari AODV. Perbedaan antara AODV dengan DSR adalah jumlah rute yang ditemukan dalam setiap proses pencarian rute. Dalam Studi ini, dilakukan penelitian terhadap kinerja DSR menggunakan *Network Simulator* 2 (NS-2). Uji coba dilakukan dengan membuat pola *traffic connection* dan pola pergerakan *node* yang kemudian disimulasikan dengan menggunakan *script* DSR.tcl. Proses tersebut akan menghasilkan file otuput berupa trace file. Trace file hasil dari simulasi akan dianalisis untuk menghitung Packet Delivery Ratio (PDR), Routing Overhead (RO), dan End-to-End Delay.**

Kata kunci*—*MANET, AODV, DSR, NS-2.

I. PENDAHULUAN

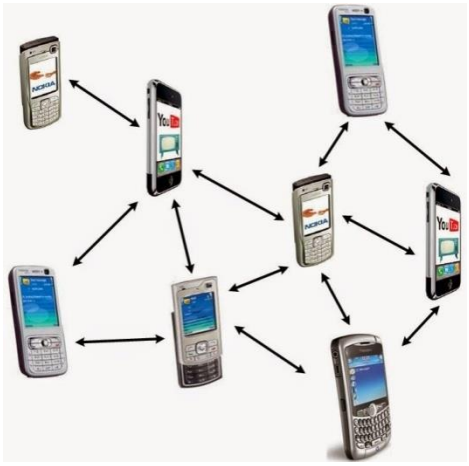
PERKEMBANGAN teknologi informasi dan komunikasi di era globalisasi saat ini berkembang dengan pesat, yaitu ditunjukkan dengan terciptanya berbagai macam teknologi yang membantu meningkatkan produktivitas manusia. Salah satu teknologi yang memudahkan manusia untuk saling berkomunikasi adalah *Mobile Ad hoc Network* (MANET). Pada MANET *mobile host* yang terhubung dengan *wireless* dapat bergerak bebas dan juga berperan sebagai router. Jaringan MANET adalah kumpulan dari beberapa *wireless node* yang dapat di *set-up* secara dinamis.

Dalam Studi ini, akan dilakukan studi kinerja terhadap protokol *routing* DSR pada topologi jaringan MANET berdasarkan *Packet Delivery Ratio* (PDR), *Routing Overhead* (RO), dan *End-to-End Delay*.

II. TINJAUAN PUSTAKA

A. *Mobile Ad Hoc Network (MANET)*

Mobile Ad hoc Network (MANET) memungkinkan terjadinya komunikasi jaringan tanpa bergantung pada ketersedian infrastruktur jaringan yang tetap. Setiap *node* dalam jaringan MANET dapat bertindak sebagai *host* dan *router*. Setiap *node* dapat saling melakukan komunikasi antara yang satu dengan yang lainnya tanpa adanya *access point*.



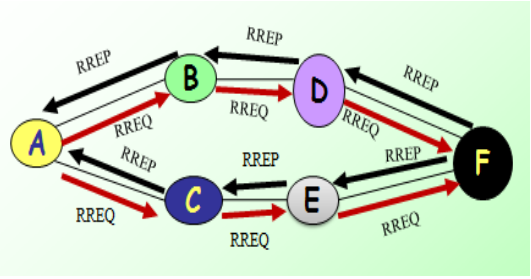
Gambar 1. Jaringan MANET [1]

Pada gambar 1 merupakan contoh penerapan jaringan MANET yang dibentuk dari sekumpulan perangkat *mobile* seperti *ponsel*. Perangkat *mobile* seperti *ponsel* harus mampu mendeteksi keberadaan perangkat lain dan melakukan pengaturan yang diperlukan untuk melakukan komunikasi dan berbagi data. Pada MANET memungkinkan perangkat untuk mempertahankan koneksi ke jaringan serta dengan mudah menambahkan dan menghapus perangkat pada jaringan. Karena pergerakan *node* yang dinamis, topologi jaringan dapat berubah dengan cepat dan tak terduga dari waktu ke waktu. Jaringan MANET bersifat desentralisasi, di mana organisasi jaringan dan pengiriman pesan harus dijalankan oleh *node* sendiri. [1]

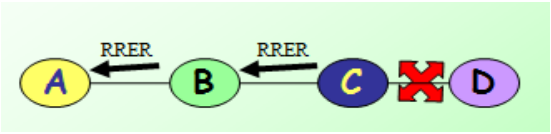
Karakteristik dan kompleksitas dari MANET antara lain memiliki topologi dinamis, menggunakan *multi-hop routing*, jaringan topologi dinamis, energi dan *bandwidth* yang terbatas, keterbatasan keamanan dan melakukan pembangunan serta pengaturan jaringan secara mandiri [2].

B. *Dynamic Source Routing (DSR)*

Protokol *Dynamic Source Routing* (DSR) [3] memiliki banyak persamaan karakteristik dengan Protokol *routing* AODV. DSR memiliki konsep bebasis vektor dan menggunakan pendekatan *multihop*. DSR juga memiliki dua fitur utama yang mirip dengan AODV yaitu *route discovery* dan *route maintenance*. Perbedaan utama antara AODV dan DSR adalah jumlah rute yang ditemukan di setiap pencarian rute atau *route discovery*.



Gambar 2. *Route discovery* pada DSR



Gambar 3. Paket *RRER* pada DSR

Route discovery pada DSR terjadi ketika *source node* memerlukan rute untuk melakukan komunikasi dengan *destination node*, maka *source node* akan mengirimkan paket *route request* (RREQ) secara *broadcast* ke *node-node* tetangga di dalam jaringan dan menunggu *route reply* (RREP). Proses *route discovery* pada DSR ditunjukkan Pada Gambar 2.

Ketika *node* A (*source node*) hendak melakukan komunikasi terhadap *node* F (*destination node*), pertama kali *node* A mengirimkan paket RREQ secara *broadcast* ke *node-node* tetangganya yaitu ke *node* B dan *node* C. Apabila *node-node* tersebut dalam hal ini *node* B dan *node* C bukan merupakan *destination node*, maka *node* tersebut akan meneruskan paket RREQ secara *broadcast* ke *node* tetangganya tetapi tidak ke *source node*. Selain itu, *node* B dan *node* C akan melakukan *set up reverse path*. Proses tersebut berulang hingga paket RREQ tersebut diterima oleh *destination node* yatiu *node* F. Selanjutnya, *node* F akan mengiirmkan paket RREP sebagai balasan dari paket RREQ yang diterimanya. *Node-node* penerima paket RREP akan melakukan *set-up forward path*.

Route maintenance pada DSR adalah pengembangan sederhana yang ada pada *route maintenance* AODV. DSR juga menggunakan paket *route error* (RERR) untuk mengirimkan pesan *error*. Proses pengiriman paket RRER pada DSR ditunjukkan pada Gambar 3.

Pada saat terjadi kerusakan *link* yang menghubungkan antara *node* C dengan *node* D, maka *node* C akan mengirimkan paket RRER ke *node* B. Selanjutnya, node B akan mengirimkan paket RRER ke node A. Setelah *node* A menerima paket RRER, proses komunikasi akan dilakukan kembali dengan menggunakan rute cadangan apabila masih tersedia. Apabila tidak tersedia rute cadangan, maka *node* A akan melakukan proses *route discovery*.

C. VirtualBox

VirtualBox adalah aplikasi *multiplatform* karena digunakan untuk meng-*install* sistem operasi di dalam sistem operasi utama pada sebuah perangkat komputer. Dengan menggunakan VirtualBox, sebuah perangkat komputer dapat menjalankan beberapa sistem operasi dalam waktu yang sama. Misalnya, pengguna dapat menjalankan Mac dan Linux di Windows, menjalankan Ubuntu di komputer dengan sistem operasi Windows, dan sebagainya.

VirtualBox didesain untuk para profesional dan pengembang di bidang Teknologi Informasi. VirtualBox dapat berjalan pada sistem operasi Windows, Mac OS X, Linux dan Oracle Solaris yang sangat ideal diaplikasikan untuk pengujian, pengembangan, dan simulasi berbagai sistem operasi pada satu mesin. [4]

Pada Studi ini, VirtualBox digunakan untuk menjalankan sistem operasi Linux yang berisi *software* NS-2 dan juga sebagai penghubung antara sistem operasi Windows dan Linux.

D. Network Simulator 2 (NS-2)

Network Simulator 2 atau biasa disebut NS-2 merupakan *software* simulasi jaringan dengan bahasa *script* yang sederhana, sangat memudahkan peneliti untuk melakukan konfigurasi jaringan dan mengamati hasil simulasi dari NS-2. NS-2 dikembangkan mengguanakan bahasa pemrograman C ++ dan Otcl. Berawal dari pengembangan simulator jaringan nyata oleh University of California Berkeley yang merupakan cikal bakal lahirnya *Network Simulator*. Pada tahun 1995 Defense Advanced Research Projects Agency (DARPA) membantu pengembangan *Network Simulator* melalui proyek *Virtual Internetwork Terstbed* (VINT). Sampai saat ini, para peneliti dan pengembang terus bekerja untuk menjadikan NS-2 lebih baik. [5]

Pada Studi ini digunakan NS-2 versi 2.35 untuk simulasi protokol *routing* DSR.

E. Propagasi Nakagami

Propagasi Nakagami awalnya diusulkan karena cocok dengan hasil empiris untuk propagasi ionosfer gelombang pendek. Dalam komunikasi nirkabel saat ini, peran utama propagasi Nakagami dapat diringkas sebagai berikut :

- Menggambarkan amplitudo sinyal yang diterima setelah dilakukan perhitungan rata-rata *tracefile*.
- Menyalurkan amplitudo sinyal secara terdistribusi.
- Mendistribusikan transmisi sesuai dengan data empiris yang lebih baik daripada model propagasi lainnya.

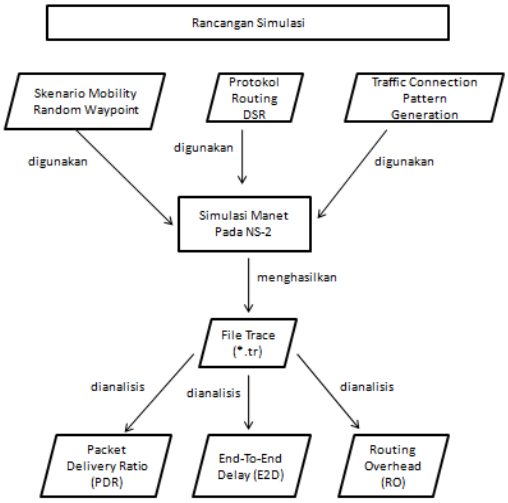
F. AWK

Awk adalah bahasa pemrograman yang digunakan untuk melakukan manipulasi data dan membuat laporan. Nama awk diambil dari nama akhir pembuatnya yaitu Alfred **A**ho, Peter **W**einberger, dan Brian **K**erningham. Pada beberapa sistem seperti linux, awk di-link-kan ke versi terbaru yang digunakan. Misalnya dalam distribusi Linux redhat, gawk merupakan link dari awk [6].

Awk dapat digunakan baik dalam *command line* maupun dimasukkan dalam sebuah skrip dengan melakukan *scan* baris per baris yang diperoleh dari standar *input*, *file* maupun *output* proses. Pada Studi ini AWk digunakan mendapatkan skrip dalam penghitungan *Packet delivery ratio* (PDR), *End-to-end delay* (E2D) dan *Routing Overhead* (RO) dari hasil *trace* pada *Network Simulator* 2.

III. DESAIN DAN PERANCANGAN

A. Deskripsi Umum



Gambar 4. Rancangan simulasi pada NS-2

Pada Studi ini akan dilakukan analisis tentang performa propagasi Nakagami pada MANET. Dalam pembuatan skenario MANET menggunakan *mobility random waypoint* dan telah ada pada NS-2 yaitu dengan cara men-*generate file node-movement (mobility generation)* dan membuat koneksi antar *node* menggunakan *traffic connection pattern*. Rancangan simulasi yang dibuat dapat dilihat pada Gambar 4.

B. Perancangan Metrik Analisis

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

1) Packet Delivery Ratio (PDR)

PDR dihitung dari perbandingan antara paket yang dikirim dengan paket yang diterima. PDR dihitung dengan menggunakan persamaan (1), dimana *received* adalah banyaknya paket data yang diterima dan *sent* adalah banyaknya paket data yang dikirimkan.

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

2) Routing Overhead (RO)

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

3) End-to-End Delay

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

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

IV. IMPLEMENTASI

A. Implementasi Skenario File Node-Movement

Perancangan skenario uji coba diawali dengan membuat pola *traffic* koneksi antara *node* secara acak menggunakan *mobility random waypoint.* Kemudian membuat koneksi dengan menggunakan *traffic connection pattern*. Pada Studi ini, pergerakan *node-node* menggunakan 4 variasi kecepatan maksimal yaitu 10 m/s, 15 m/s, 20 m/s dan 25 m/s.

Skenario *mobility random waypoint* dibuat dengan men-*generate file node-movement* yang telah ada pada NS-2 atau *tools*-nya biasa disebut 'setdest' yang nantinya akan menghasilkan *output* dalam bentuk .txt dan digunakan dalam *file* Tcl selama simulasi pada NS-2 sebagai bentuk pergerakan *node* yang berpindah-pindah.

B. Implementasi Traffic-Connection Pattern Generation

Traffic-connection dibuat dengan menjalankan *file* Tcl yang bernama cbrgen.tcl. *Output* dari program tersebut akan digunakan pada simulasi sebagai penghubung antara *node.* spesifikasi yang digunakan pada *traffic-connection pattern* ditunjukkan pada Tabel 1.

C. Implementasi Network Skenario 2

Pada implementasi ini dilakukan penggabungan antara hasil skenario *grid* dan skenario riil berupa *file* (mobility.tcl) dengan skrip TCL dari kode NS-2 yang diberikan parameter-parameter untuk selanjutnya dilakukan percobaan simulasi VANET pada NS-2. Parameter simulasi perancangan sistem VANET ditunjukkan pada Tabel 2.

Tabel 1.
Parameter *Traffic-Connection Pattern*

No.	Parameter	Spesifikasi
1	Jenis *traffic*	CBR
2	Jumlah paket per detik	1
3	Jumlah *seed*	1
4	Jumlah koneksi	2
5	*Agent*	UDP

Tabel 2.
Parameter *Simulasi*

No.	Parameter	Spesifikasi
1	*Network simulator*	NS – 2.34
2	*Routing Protocol*	DSR
3	Waktu Simulasi	200 detik
4	Waktu Pengiriman Paket Data	Nakagami = 0 – 200 detik
5	Area	500m x 500 m 900m x 900 m
6	Banyak *node*	60, 70, 80, 90
7	Radius transmisi	100 m
9	Tipe data	*Constant Bit Rate* (CBR)
10	Ukuran paket data	512 *bytes*
11	Protokol MAC	IEEE 802.11
12	Mode Transmisi	Nakagami
13	Tipe Antenna	OmniAntenna
14	Tipe *Interface Queue*	CMUPriQueue
15	Tipe Peta	MANET (*random way point)*
16	Tipe kanal	*Wireless channel*
17	Tipe *trace*	*Old Wireless* Format *Trace*

V. UJI COBA DAN ANALISIS HASIL

Uji coba ini dilakukan dengan menggunakan sebuah komputer dan dalam lingkungan yang dibatasi. Berikut ini adalah spesifikasi perangkat dan lingkungan yang digunakan dalam simulasi ini.

- CPU Intel(R) Core(TM) i3 CPU @ 2.2GHz.
- Sistem Operasi Linux Ubuntu 14.04 64-bit.
- Memori 4 GB.
- Harddisk 640 GB.
- *Network Simulator* ns2.35

A. *Analisis Packet Delivery Ratio (PDR)*

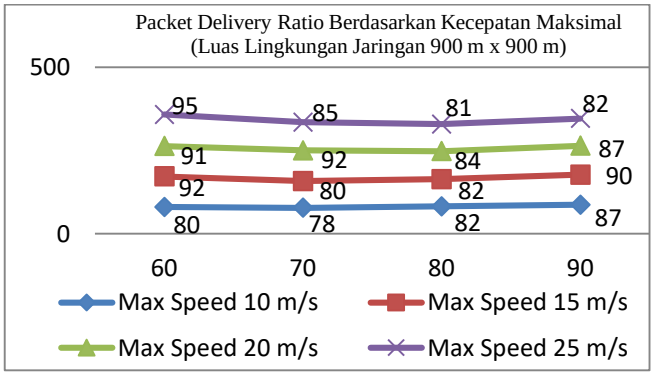
Trace file hasil dari menjalankan program skenario *node-movement* menggunakan model transmisi DSR dianalisis nilai PDR menggunakan *script* pdr.awk. Pada Tabel 3 dan Tabel 4 menunjukkan performa PDR pada skenario *mobility random waypoint* dengan menggunakan propagasi Nakagami menghasilkan nilai yang fluktuatif ketika kecepatan maksimal perpindahan *node* bertambah.

Tabel 3.
Packet Delivery Ratio (PDR) DSR Luas Lingkungan Jaringan 900 m x 900 m

Max Speed (m/s)	PDR (%)			
	Node 60	Node 70	Node 80	Node 90
10	80	78	82	87
15	92	80	82	90
20	91	92	84	87
25	95	85	81	82

Tabel 4.
Packet Delivery Ratio (PDR) DSR Luas Lingkungan Jaringan 500 m x 500 m

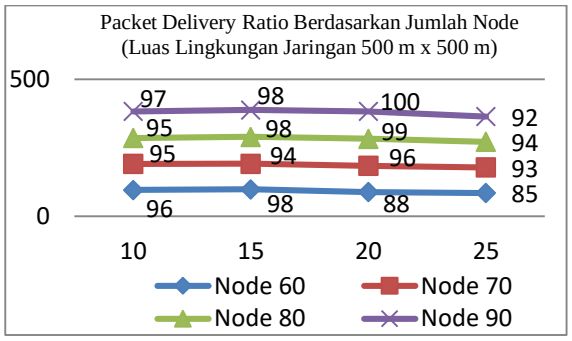
Max Speed (m/s)	PDR (%)			
	Node 60	Node 70	Node 80	Node 90
10	96	95	95	97
15	98	94	98	98
20	88	96	99	100
25	95	93	94	92



Gambar 5. Grafik PDR Berdasarkan Kecepatan Maksimal pada Luas Lingkungan Jaringan 900 m x 900 m

Pada Gambar 5. menunjukkan grafik performa PDR dengan luas lingkungan jaringan 900 m x 900 m. Sebagai contoh, akan digunakan kecepatan maksimal perpindahan *node* sebesar 10 m/s untuk memberikan bukti bahwa nilai PDR yang dihasilkan fluktuatif. Pada saat kecepatan maksimal perpindahan *node* 10 m/s dengan jumlah *node* 60, nilai PDR yang dihasilkan adalah 80%. Pada saat jumlah *node* ditambah menjadi 70, nilai PDR yang dihasilkan mengalami penurunan sebanyak 2,5% menjadi 78%. Kemudian pada saat jumlah *node* ditambah menjadi 80, nilai PDR yang dihasilkan mengalami peningkatan sebanyak 5,13% menjadi 82%. Kemudian pada saat jumlah *node* ditambah menjadi 90, nilai PDR yang dihasilkan kembali mengalami peningkatan sebanyak 6,1% menjadi 87%.

Pada Gambar 6 menunjukkan grafik performa PDR dengan luas lingkungan jaringan 500 m x 500 m. Sebagai contoh, akan digunakan jumlah *node* 70 untuk memberikan bukti bahwa nilai PDR yang dihasilkan fluktuatif. Pada saat jumlah *node* 70 dengan kecepatan maksimal perpindahan *node* 10 m/s, nilai PDR yang dihasilkan adalah 95%. Pada saat jumlah kecepatan maksimal perpindahan *node* ditambah menjadi 15 m/s, nilai PDR yang dihasilkan mengalami penurunan sebanyak 1,05% menjadi 94%. Kemudian pada saat jumlah kecepatan maksimal perpindahan *node* ditambah menjadi 20 m/s, nilai PDR yang dihasilkan kembali mengalami peningkatan sebanyak 2,13% menjadi 96%. Namun pada saat kecepatan maksimal perpindahan *node* ditambah menjadi 25, nilai PDR yang dihasilkan mengalami penurunan sebanyak 3,13% menjadi 93%.



Gambar 6. Grafik PDR Berdasarkan Kecepatan Maksimal pada Luas Lingkungan Jaringan 500 m x 500 m

B. *Analisis Routing Overhead (RO)*

Tabel 5.
Packet Delivery Ratio (RO) DSR Luas Lingkungan Jaringan 900 m x 900 m

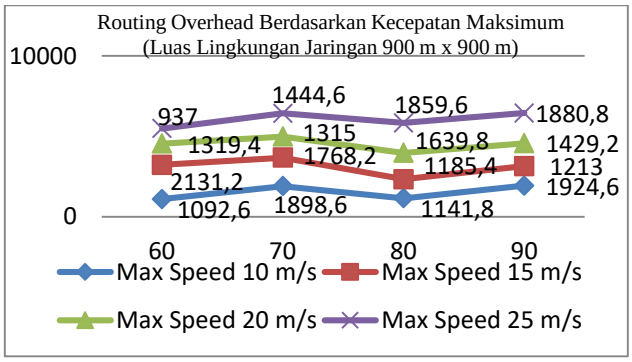
Max Speed (m/s)	RO (Paket)			
	Node 60	*Node* 70	*Node* 80	*Node* 90
10	1092,6	1898,6	1141,8	1924,6
15	2131,2	1768,2	1185,4	1213
20	1319,4	1315	1639,8	1429,2
25	937	1444,6	1859,6	1880,8

Tabel 6.
Packet Delivery Ratio (RO) DSR Luas Lingkungan Jaringan 500 m x 500 m

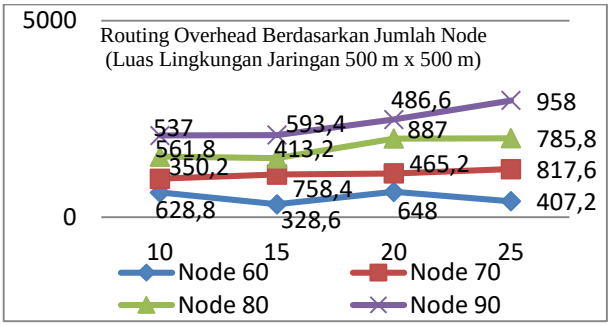
Max Speed (m/s)	RO (Paket)			
	Node 60	*Node* 70	*Node* 80	*Node* 90
10	628.8	350.2	561.8	537
15	328.6	758.4	413.2	593.4
20	648	465.2	887	486.6
25	407.2	817.6	785.8	958

Trace file hasil dari menjalankan program skenario *node-movement* menggunakan model transmisi DSR dianalisis nilai RO menggunakan *script* ro.awk. Pada Tabel 5 dan Tabel 6 menunjukkan performa RO pada skenario *mobility random waypoint* dengan menggunakan propagasi Nakagami menghasilkan nilai yang fluktuatif ketika kecepatan maksimal perpindahan *node* bertambah.

Pada Gambar 5 menunjukkan grafik performa RO dengan luas lingkungan jaringan 900 m x 900 m. Sebagai contoh, akan digunakan kecepatan maksimal perpindahan *node* sebesar 20 m/s untuk memberikan bukti bahwa nilai RO yang dihasilkan fluktuatif. Pada saat kecepatan maksimal *node* 20 m/s dengan menggunakan jumlah *node* 60, nilai RO yang dihasilkan sebanyak 1319,4 paket. Pada saat jumlah *node* ditambah menjadi 70, nilai RO yang dihasilkan mengalami penurunan sebanyak 0,33% menjadi 1315 paket. Namun pada saat jumlah *node* ditambah menjadi 80, nilai PDR yang dihasilkan mengalami peningkatan sebanyak 24,7% menjadi 1639,8 paket. Kemudian pada saat jumlah *node* ditambah menjadi 90, nilai RO yang dihasilkan mengalami penurunan sebanyak 12,84% menjadi 1429,2 paket.



Gambar 7. Grafik RO Berdasarkan Kecepatan Maksimal pada Luas Lingkungan Jaringan 900 m x 900 m



Gambar 8. Grafik RO Berdasarkan Jumlah Node pada Luas Lingkungan Jaringan 500 m x 500 m

Pada Gambar 7 menunjukkan grafik performa RO dengan luas lingkungan jaringan 500 m x 500 m. Sebagai contoh, akan digunakan jumlah *node* 60 untuk memberikan bukti bahwa nilai RO yang dihasilkan fluktuatif. Pada saat jumlah *node* 60 dengan kecepatan maksimal perpindahan *node* 10 m/s, nilai RO yang dihasilkan adalah 628,8 paket. Pada saat jumlah kecepatan maksimal perpindahan *node* ditambah menjadi 15 m/s, nilai RO yang dihasilkan mengalami penurunan sebanyak 47,74% menjadi 328,6 paket. Namun pada saat jumlah kecepatan maksimal perpindahan *node* ditambah menjadi 20 m/s, nilai RO yang dihasilkan mengalami peningkatan sebanyak 97,2% menjadi 648 paket. Kemudian pada saat kecepatan maksimal perpindahan *node* ditambah menjadi 25 m/s, nilai RO yang dihasilkan mengalami penurunan sebanyak 37,16% menjadi 407,2 paket.

C. Analisis End-to-End Delay (E2D)

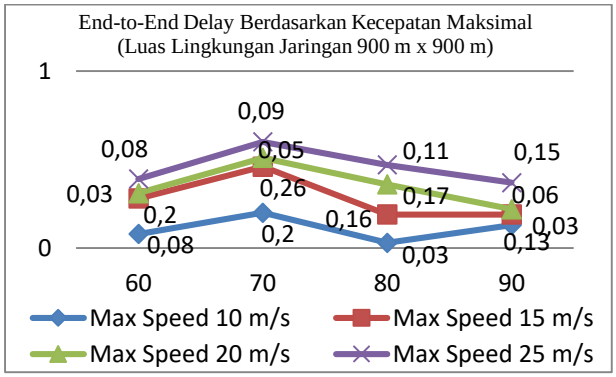
Trace file hasil dari menjalankan program skenario *node-movement* menggunakan model transmisi DSR dianalisis nilai E2D menggunakan *script* delay.awk. Pada Tabel 7 dan Tabel 8 menunjukkan performa *End-to-End Delay* pada skenario *mobility random waypoint* dengan menggunakan propagasi Nakagami menghasilkan nilai yang fluktuatif ketika kecepatan maksimal perpindahan *node* bertambah.

Tabel 7.
End-to-End Delay DSR Luas Lingkungan Jaringan 900 m x 900 m

Max Speed (m/s)	E2D (*second*)			
	Node 60	*Node* 70	*Node* 80	*Node* 90
10	0,08	0,2	0,03	0,13
15	0,2	0,26	0,16	0,06
20	0,03	0,05	0,17	0,03
25	0,08	0,09	0,11	0,15

Tabel 8.
End-to-End Delay DSR Luas Lingkungan Jaringan 500 m x 500 m

Max Speed (m/s)	E2D (*second*)			
	Node 60	*Node* 70	*Node* 80	*Node* 90
10	0.03	0.23	0.02	0.06
15	0.01	0.18	0.01	0.03
20	0.31	0.01	0.26	0.01
25	0.18	0.57	0.16	0.19

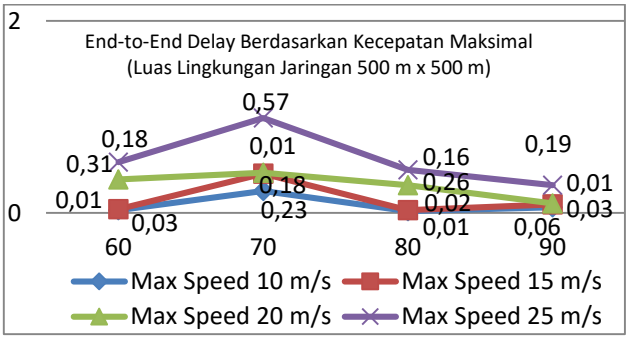


Gambar 9. Grafik E2D Berdasarkan Kecepatan Maksimal pada Luas Lingkungan Jaringan 900 m x 900 m

Pada Gambar 8 menunjukkan grafik performa E2D dengan luas lingkungan jaringan 900 m x 900 m. Sebagai contoh, akan digunakan kecepatan maksimal perpindahan *node* sebesar 10 m/s untuk memberikan bukti bahwa nilai E2D yang dihasilkan fluktuatif. Pada saat kecepatan maksimal perpindahan node 10 m/s dengan jumlah *node* 60, nilai E2D yang dihasilkan adalah 0,08 *second*. Pada saat jumlah *node* ditambah menjadi 70, nilai E2D yang dihasilkan mengalami peningkatan sebanyak 150% menjadi 0,2 *second*. Namun pada saat jumlah *node* ditambah menjadi 80, nilai RO yang dihasilkan mengalami penurunan sebanyak 85% menjadi 0,03 *second*. Kemudian pada saat jumlah *node* ditambah menjadi 90, nilai E2D yang dihasilkan

mengalami peningkatan sebanyak 333,33% menjadi 0,13 *second*.

Pada Gambar 9 menunjukkan grafik performa E2D dengan luas lingkungan jaringan 900 m x 900 m. Sebagai contoh, akan digunakan kecepatan maksimal perpindahan *node* sebesar 15 m/s untuk memberikan bukti bahwa nilai E2D yang dihasilkan fluktuatif.Pada saat kecepatan maksimal *node* 15 m/s dengan jumlah *node* 60, nilai E2D yang dihasilkan adalah 0,01 *second*. Pada saat jumlah *node* ditambah menjadi 70, nilai E2D yang dihasilkan mengalami peningkatan sebanyak 1800% menjadi 0,18 *second*. Namun pada saat jumlah *node* ditambah menjadi 80, nilai E2D yang dihasilkan mengalami penurunan sebanyak 94,44% menjadi 0,01 *second*. Kemudian pada saat jumlah *node* ditambah menjadi 90, nilai E2D yang dihasilkan mengalami peningkatan sebanyak 200% menjadi 0,03 *second*.



Gambar 10. Grafik E2D Berdasarkan Kecepatan Maksimal pada Luas Lingkungan Jaringan 500 m x 500 m.

VI. KESIMPULAN DAN SARAN

Dari hasil uji coba yang telah dilakukan, Hal – hal yang dapat mempengaruhi nilai PDR, E2D dan RO yang dihasilkan dari model propagasi Nakagami adalah:

- Jumlah *node* yang digunakan dalam simulasi.
- Kecepatan maksimal perpindahan *node.*
- Luas lingkungan jaringan.

Saran yang dapat diberikan dari hasil uji coba dan evaluasi dari Studi ini untuk pengembangan simulasi kedepan, antara lain:

1. Dapat dilakukan percobaan pada lingkungan VANET untuk penerapan model transmisi Nakagami.
2. Perlu dikembangkan penelitian menggunakan skenario yang lebih riil seperti menggunakan *Simulation of Urban Mobility* (SUMO) atau simulator lainnya.

DAFTAR PUSTAKA

[1] P. R. dan R. Dahiya, "Study and Analysis of Throughput, Delay and Packet Delivery Ratio in Manet Based DSR Routing Protocols," *Int. J. Adv. Res. Eng. Technol.*, vol. 1, no. 2, 2013.

[2] S. dan Anil, "A Protocol for Reducing Routing Overhead in Mobile Ad Hoc Networks," *J. Comput. Sci. Eng. Technol.*, 2014.

[3] M. K. M. dan S. R. Das, "Ad hoc on-demand multipath distance vector routing," *Wirel. Commun. Mob. Comput.*, 2006.

[4] Oracle, "VirtualBox," 2017. [Online]. Available: https://www.virtualbox.org/manual/ch01.html.

[5] T. I. dan E. Hossain, "Introduction to Network Simulator NS2," *Springer*, 2012.

[6] "Bengkel Ubuntu," 2016. [Online]. Available: http://bengkelubuntu.org/teks/awk/Praktikum 01 - Berkenalan dengan AWK.pdf.