TESIS - KI142502

PENGEMBANGAN PROTOKOL *ROUTING MULTICAST AD HOC ON-DEMAND DISTANCE VECTOR* DENGAN MEMPERHITUNGKAN JARAK *EUCLIDEAN* BERDASARKAN POSISI, KECEPATAN DAN *DELAY* TRANSMISI PADA VANET

NURDIANSYAH REZKINANDA
NRP. 5114201023

DOSEN PEMBIMBING

Dr. Eng. Radityo Anggoro, S.Kom., M.Sc

PROGRAM MAGISTER
BIDANG KEAHLIAN KOMPUTASI BERBASIS JARINGAN
JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2016

THESIS - KI142502

ROUTING PROTOCOL IMPROVEMENT OF MULTICAST AD HOC ON-DEMAND DISTANCE VECTOR BY CALCULATING EUCLIDEAN DISTANCE BASED ON POSITION, SPEED AND TRANSMISSION DELAY IN VANET

NURDIANSYAH REZKINANDA
NRP. 5114201023

DOSEN PEMBIMBING
Dr. Eng. Radityo Anggoro, S.Kom., M.Sc

MASTER PROGRAM
NET CENTRIC COMPUTATION FIELD
INFORMATICS DEPARTMENT
FACULTY OF INFORMATION TECHNOLOGY
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2016

LEMBAR PENGESAHAN
TESIS

Tesis disusun untuk memenuhi salah satu syarat memperoleh gelar
Magister Komputer (M.Kom.)
di
Institut Teknologi Sepuluh Nopember Surabaya

oleh:
NURDIANSYAH REZKINANDA
NRP. 5114201023

Dengan judul :
Pengembangan Protokol *Routing Multicast Ad Hoc On-Demand Distance Vector* dengan Memperhitungkan Jarak *Euclidean* berdasarkan Posisi, Kecepatan dan *Delay* Transmisi pada VANET

Tanggal Ujian : 4 - 1 - 2017
Periode Wisuda : 2016 Gasal

Disetujui oleh:

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

Prof.Ir.Supeno Djanali, M.Sc, Ph.D (Penguji 1)
NIP. 194806191973011001

Ir. Muchammad Husni, M.Kom (Penguji 2)
NIP. 196002211984031001

Royyana Muslim I, S.Kom, M.Kom, Ph.D (Penguji 3)
NIP. 197708242006041001

a.n Direktur Program Pasca Sarjana,
Asisten Direktur Program Pascasarjana

Prof. Dr. Ir. Tri Widjaja, M.Eng
NIP. 196110211986031001

[Halaman ini sengaja dikosongkan]

PENGEMBANGAN PROTOKOL *ROUTING MULTICAST AD HOC ON-DEMAND DISTANCE VECTOR* DENGAN MEMPERHITUNGKAN JARAK *EUCLIDEAN* BERDASARKAN POSISI, KECEPATAN DAN *DELAY* TRANSMISI PADA VANET

Nama Mahasiswa : Nurdiansyah Rezkinanda
NRP : 5114201023
Pembimbing : Dr. Eng. Radityo Anggoro, S.Kom., M.Eng.Sc

ABSTRAK

Protokol *routing* dalam jaringan *Vehicular Ad hoc Network* (VANET) saat ini masih dikembangkan untuk memperbaiki kinerja transmisi paket data pada jaringan antar kendaraan. Berbagai pendekatan konektivitas antar jaringan kendaraan terus dilakukan karena VANET memiliki karakteristik yang berbeda dengan *Mobile Ad hoc Network* (MANET) yaitu dalam hal kecepatan *node*. Protokol *Multicast Ad hoc On-Demand Distance Vector* (MAODV) yang termasuk protokol *routing* reaktif dengan transmisi *multicast* banyak dilakukan penelitian dan peningkatan dengan berbagai pendekatan. Pada penelitian ini dilakukan pengembangan protokol *routing* MAODV dengan memperhitungkan jarak *Euclidean* antar *node* berdasarkan posisi, kecepatan dan *delay* transmisi. Protokol *routing Multicast Adaptif Structured-tree* berbasis *Reactive Euclidean Node Knowledge* (MAS-BRENK) diusulkan untuk memperbaiki mekanisme *multicast tree maintenance* yaitu proses *join* dan *prune*. Perhitungan bobot *weighted product* digunakan untuk menghitung bobot antar *node* berdasarkan jarak *euclidean*, kecepatan dan *delay* transmisi. Protokol yang diusulkan tersebut diujikan ke dalam skenario jalan perkotaan. Hasil pengujian menunjukkan bahwa protokol MAS-BRENK mengalami peningkatan pengiriman paket MACT sebesar 1.3% dan penerimaan paket MACT dengan *flag prune* sebesar 8.2%. Selain itu penerimaan paket MACT dengan *flag join* menurun sebesar 1.2%. Hasil akhir yang didapatkan yaitu peningkatan PDR sebanyak 0.2% dan *throughput* sebanyak 2%, serta penurunan *delay* 14.3% dari protokol MAODV.

Kata Kunci : VANET, *multicast*, MAODV, *euclidean distance*, *weighted product*

[Halaman ini sengaja dikosongkan]

ROUTING PROTOCOL IMPROVEMENT OF MULTICAST AD HOC ON-DEMAND DISTANCE VECTOR BY CALCULATING EUCLIDEAN DISTANCE BASED ON POSITION, SPEED AND TRANSMISSION DELAY IN VANET

Student Name	: Nurdiansyah Rezkinanda
NRP	: 5114201023
Supervisor	: Dr. Eng. Radityo Anggoro, S.Kom., M.Eng.Sc

ABSTRACT

Routing protocol on Vehicular Ad hoc Network (VANET) was developed to improve the performance of data packet transmission in vehicular network. Various approach of vehicular connectivity continued because VANET has different characteristics with Mobile Ad hoc Network (MANET) especially in node velocity. Multicast Ad hoc On-Demand Distance Vector (MAODV) Protocol which includes a reactive routing protocol with multicast transmissions done much research and an improve in the variety of approaches.

In this study, we improve MAODV routing protocol by calculating euclidean distance between nodes based on position, velocity and transmission delay. We propose Multicast Adaptive Structured-tree Base on Euclidean Node Knowledge (MAS-BRENK) routing protocol to improve multicast tree maintenance mechanism that join and prune process. Weighted calculation of weighted product is used to calculate the weights between nodes based on Euclidean distance of euclidean distance, velocity and transmission delay. The proposed protocol was tested in a scenario of urban roads. The test results showed that the MAS-BRENK protocol MACTs packet delivery increased by 1.3% and the MACTs packet receipt with prune flag by 1.2%. In additions MACT packet reception join flag decreased by 1.2%. The final result is an increase in PDR by 0.2% and throughput by 2%, and the delay decreased 14.3% from MAODV protocol.

Keywords: VANET, multicast, MAODV, euclidean distance, weighted product

[Halaman ini sengaja dikosongkan]

DAFTAR ISI

DAFTAR GAMBAR

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Mobile Ad hoc Network (MANET) merupakan gabungan dari beberapa perangkat bergerak atau *mobile node* yang dapat berkomunikasi secara nirkabel membentuk topologi jaringan tanpa menggunakan infrastruktur jaringan yang ada. Setiap *node* berperan sebagai *router* yang dapat bergerak secara dinamis setiap waktu. *Vehicle Ad hoc Network* (VANET) merupakan perkembangan dari *Mobile Ad hoc Network* (MANET) dimana *node* memiliki karakteristik mobilitas yang sangat tinggi dan terbatas pada pola pergerakannya (Harri et al., 2006). Hal tersebut membuat topologi jaringan VANET lebih dinamis dibandingkan dengan MANET.

Mobilitas *node* yang sangat tinggi pada VANET berpengaruh besar terhadap perubahan topologi jaringan setiap waktu. Selain menyebabkan perubahan topologi, hal tersebut juga dapat menyebabkan rute terputus karena *node* keluar dari jangkauan sinyal transmisi (Su et al., 2000). Dengan adanya mobilitas *node*, kegagalan *node*, dan karakteristik mobilitas *node* yang dinamis, *link* pada sebuah rute dapat menjadi tidak tersedia. Rute yang rusak memaksa *node* sumber mencari ulang rute untuk mentransmisikan data ke *node* tujuan. Hal ini dapat menyebabkan *delay* dan banyaknya paket yang hilang (Balakrishna et al., 2010).

Protokol *routing* AODV merupakan salah satu protokol yang menggunakan transmisi *unicast* untuk melakukan pengiriman data dari *node* sumber menuju *node* tujuan (Perkins, 1999). AODV merupakan penggabungan dari mekanisme *route discovery* dan *route maintenance* pada protokol *routing Dynamic Source Routing* (DSR) serta penggunaan *hop-by-hop routing*, *squence number* dan *periodic beacon* pada protokol *routing Destination-Sequenced Distance Vector* (DSDV). Kelebihan AODV yaitu menyimpan rute yang dibutuhkan sehingga kebutuhan memori sedikit, perbaikan rute yang rusak dapat ditangani dengan baik dan mempunyai skalabilitas yang tinggi pada jumlah *node* (Perkins, 1999). Diajukannya prediksi jarak antar *node* berdasarkan posisi *node* dengan nilai *threshold* kurang dari sama dengan jangkauan transmisi *node*, serta *root node* dan *road static unit* (RSU) sebagai

kontrol antar *node* (Joshi, A. 2015), secara signifikan dapat meningkatkan *throughput* dan *packet delivery ratio* serta menurunkan *overhead* dan *end-to-end delay*. Namun protokol *routing* AODV yang masih menggunakan transmisi *unicast* masih perlu dioptimalkan.

Namun ketika *node* sumber melakukan pengiriman data menuju lebih dari satu *node* tujuan, metode transmisi pada protokol *routing* menjadi masalah penting. Karena pada transmisi *unicast*, *node* sumber mengirimkan paket data berulang kali bergantung pada jumlah *node* tujuan. Masalah protokol *routing* dengan transmisi *unicast* dapat ditangani dengan menggunakan transmisi *multicast* ketika *node* sumber mengirimkan paket data kepada lebih dari satu *node* tujuan. Salah satu pengembangan dari AODV dengan menerapkan penggunaan transmisi *multicast* adalah protokol *routing Multicast Ad hoc On-Demand Distance Vector* (MAODV). Protokol *routing* MAODV (Royer & Perkins, 2000) memungkinkan *node* sumber mendapatkan rute menuju beberapa *node* tujuan secara *multicast* dengan membentuk *multicast tree* yang terbagi dalam kelompok *multicast* (*multicast group*) tertentu.

Pada protokol *routing* MAODV, terdapat proses *multicast tree maintenance* dimana memungkinkan *node* dikeluarkan atau *prune* dari *multicast group* dan melakukan perbaikan rute yang terputus dengan bergabung atau *join* terhadap *multicast group* tertentu untuk melakukan pencarian rute baru. Pada penelitian (Ghazemi, 2012) ditunjukkan bahwa MAODV memiliki tingkat *scalability* dan *overhead* yang sedang dibandingkan dengan protokol *Adaptive Demand-driven Multicast Routing* (ADMR) dan *Ad hoc Multicast Routing* (AMR). Permasalahan utama pada protokol *routing* MAODV yaitu tidak disertakannya informasi jarak antar *node* baik dalam proses *multicast tree maintenance*. Penelitian (Shurdi et al., 2011) menunjukkan bahwa protokol MAODV mengalami penurunan *throughput* ketika kecepatan *node* meningkat dibandingkan dengan protokol ADMR dan *On-Demand Multicast Routing Protocol* (ODMRP).

Sehubungan dengan mobilitas *node* yang tinggi, *node* yang memiliki potensi mobilitas menjauh dari jangkauan *node* lain dapat menyebabkan kerusakan dan perpindahan rute. Perpindahan rute terjadi saat sebuah *node* mengirim pesan *error* ke *node* sumber. Meskipun rute baru dapat dibentuk ketika kerusakan terjadi,

berulang kali membangun kembali rute baru dapat menimbulkan *delay* dan *overhead* yang besar (Xia et al., 2014). Disisi lain topologi jaringan VANET tidak terlepas dari informasi posisi, kecepatan dan *delay* transmisi yang selalu berubah setiap saat secara dinamis. Maka informasi tersebut hingga pergerakan setiap *node* dapat diprediksi atau tidak acak, hal ini dapat membantu mempermudah pengiriman paket data (Menouar et al, 2006). Diajukannya *Motion-MAODV* (Jemaa et al., 2015) dengan perhitungan *route stability* yang mempertimbangkan kecepatan dan jumlah *hop node* menuju *group leader*, membuktikan dapat meningkatkan *packet delivery ratio*, namun *end-to-end delay* juga mengalami peningkatan.

Berdasarkan penelitian dan permasalahan yang telah dijabarkan sebelumnya, diusulkan suatu mekanisme pembentukan dan pelepasan *multicast tree* yang berbeda pada protokol *routing* MAODV dengan memperhitungkan jarak *Euclidean* berdasarkan posisi, kecepatan dan *delay* transmisi. Dalam penelitian ini dinamakan dengan protokol *routing Multicast Adaptif Structured-tree based on Reactive Euclidean Node Knowledge* (MAS-BRENK). Dengan tingginya mobilitas kendaraan pada VANET, perhitungan *Euclidean Node Knowledge* (ENK) dapat dipergunakan untuk melakukan *pruning* atau pelepasan anggota *multicast group* sebelum rute terputus sehingga proses *prune* tidak lagi menunggu rute *timeout*. Selain itu perhitungan tersebut digunakan *node* sebagai pertimbangan dalam melakukan *join* kembali kepada *multicast group* tertentu, sehingga intensitas pembentukan rute secara berulang kali dapat diminimalisasi. Diharapkan mekanisme pemutusan rute dan pembentukan ulang rute baru yang diusulkan dapat mengurangi *end-to-end delay* serta *packet delivery ratio* (PDR) dan *throughput* dapat ditingkatkan.

1.2 Perumusan Masalah

Rumusan masalah dalam penelitian ini adalah sebagai berikut:

1. Bagaimana proses *prune* dilakukan sebelum terjadi kerusakan rute.
2. Bagaimana proses *join* dilakukan untuk mempertimbangkan rute baru.
3. Bagaimana mengukur kinerja dari protokol *routing* MAS-BRENK.

1.3 Tujuan dan Manfaat Penelitian

Tujuan dari penelitian ini adalah memperbaiki mekanisme *multicast tree maintenance* dari protokol *routing* MAODV pada proses *prune* dan *join multicast group*. Manfaat yang diharapkan yaitu dapat mengurangi *end-to-end delay* serta meningkatkan *packet delivery ratio* dan *throughput*.

1.4 Batasan Masalah

Batasan masalah dalam penelitian sebagai berikut:

1. Implementasi menggunakan Network Simulator 2 (NS-2) versi 2.35.
2. Wilayah kerja jaringan berupa peta *Grid*.
3. Model pergerakan *node* menggunakan *Simulator of Urban Mobility* (SUMO).

1.5 Kontribusi Penelitian

Kontribusi dalam penelitian ini adalah mengusulkan mekanisme baru dalam proses *prune* dan *join multicast group* pada protokol *routing* MAODV dengan perhitungan jarak *euclidean* berdasarkan posisi, kecepatan dan *delay* transmisi.

BAB 2
KAJIAN PUSTAKA

2.1 *Vehicular Ad-hoc Network* (VANET)

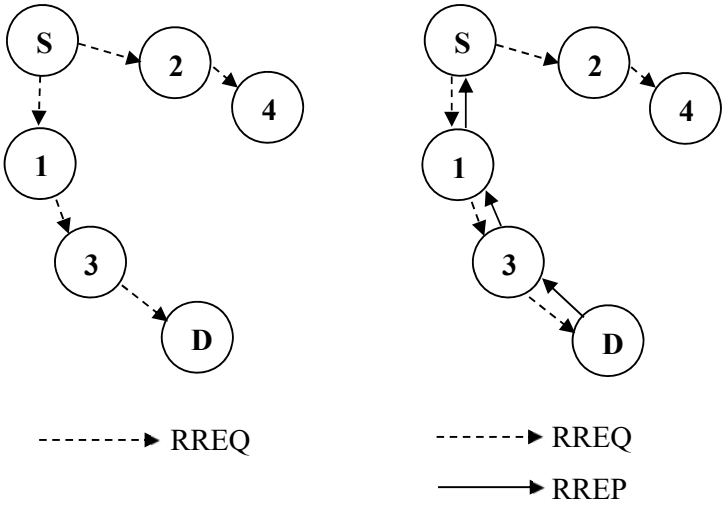
Vehicular Ad-hoc Network (VANET) merupakan pengembangan dari *Mobile Ad-hoc Network* (MANET) dimana pengembangannya difokuskan pada kendaraan atau *vehicle* yang dapat saling berkomunikasi maupun mengirimkan data. VANET adalah sebuah teknologi baru yang memadukan kemampuan komunikasi nirkabel kendaraan menjadi sebuah jaringan yang bebas infrastuktur (Najafabadi, 2011) serta memiliki karakteristik mobilitas yang sangat tinggi dan terbatas pada pola pergerakannya (Harri et al., 2006). *Node* dalam jaringan dianggap sebagai *router* yang bebas bergerak dan bebas menentukan baik menjadi *client* maupun menjadi *router*. Protokol *routing* pada VANET memiliki dua model yaitu protokol *routing* reaktif yang membentuk tabel *routing* hanya saat dibutuhkan dan protokol *routing* proaktif yang melakukan pemeliharaan tabel *routing* secara berkala pada waktu tertentu (Corson & Macker, 1999).

Properti umum yang harus dipenuhi oleh protokol jaringan *ad hoc* adalah *route discovery*, *packet delivery* dan *loop freedom* (Corson dan Macker, 1999). Karena VANET merupakan komunikasi antar kendaraan diasumsikan memiliki perangkat *Global Positioning System* (GPS). Maka informasi kecepatan, jalan, arah hingga pergerakan setiap *node* dapat diprediksi / tidak acak. Hal ini dapat membantu mempermudah pengiriman paket data (Menouar et al, 2006). VANET akan membentuk jaringan *multi-hop* antar *node* untuk mengirimkan data menuju *node* lain ataupun *static intersection node* (Nakamura et al., 2010).

Pergerakan *node* pada VANET bisa berubah setiap saat dan terbatas pada rute lalu lintas yang dapat ditentukan dari koordinat peta. Sehingga setiap *node* akan terus memperbarui informasi dalam tabelnya sesuai informasi dari *node* lain. Perubahaan pergerakan pada VANET menjadi salah satu permasalahan dalam pengiriman paket data sehingga dibutuhkan informasi jarak antar *node*, kecepatan dan *delay* transmisi.

2.2 *Ad hoc On-Demand Distance Vector* (AODV)

Ad hoc On-Demand Distance Vector (AODV) merupakan protokol reaktif pada MANET yang memiliki standar waktu untuk menentukan berapa lama sebuah rute dapat digunakan (*route validity*) sehingga properti *route discovery* dan *packet delivery* harus dapat dipenuhi dalam waktu tertentu yang telah dispesifikasikan (Johnson *et al*., 2007). Pada AODV terdapat tiga tahap utama yang digunakan untuk menentukan dan memelihara rute yaitu *route request* (RREQ), *route reply* (RREP) dan *route error* (RERR).



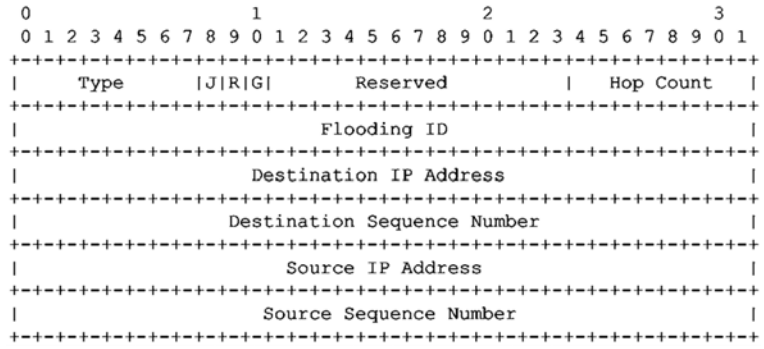
Gambar 2.1 AODV *routing protocol* (Perkins, 1999)

Proses untuk menentukan rute pada protokol AODV yaitu dengan menggunakan dua pesan yaitu dengan mengirimkan pesan RREQ dan RREP (Perkins, 1999). Ketika *node* sumber melakukan pencarian rute menuju *node* tujuan namun belum terbentuk rute yang tepat, maka *node* sumber akan melakukan inisialisasi *route discovery proccess* untuk menentukan rute ke arah *node* tujuan seperti pada Gambar 2.1.

2.2.1. Pembentukan Rute

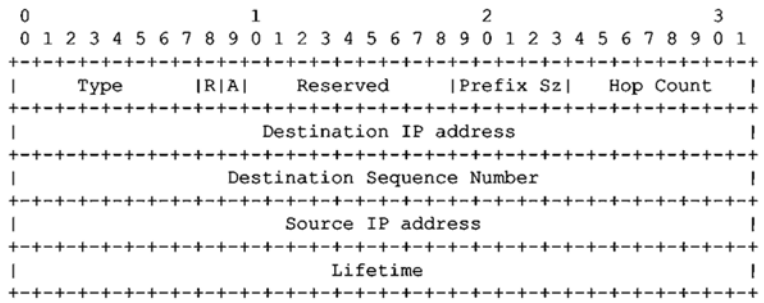
Pembentukan rute pada AODV dilakukan dengan tanda *message* (pesan) RREQ dan RREP. Ketika *node* sumber menginginkan suatu rute menuju *node* tujuan tetapi belum mempunyai rute yang benar maka *node* sumber akan menginisialisasi *route discovery proccess* untuk menemukan rute ke *node* tujuan

dengan cara melakukan *broadcast* paket RREQ menuju *node* tetangganya, paket RREQ berisi informasi sebagai berikut :



Gambar 2.2 Format Pesan RREQ (Perkins, 1999)

Nilai *Flooding ID* (biasa disebut *broadcast ID*) akan bertambah satu setiap *node* mengirimkan RREQ yang baru dan digunakan sebagai identifikasi sebuah paket RREQ. Jika *node* yang menerima RREQ memiliki informasi rute menuju *node* tujuan, maka *node* tersebut akan mengirim paket RREP kembali menuju *node* sumber. Tetapi jika tidak memiliki informasi rute maka *node* tersebut akan melakukan *broadcast* ulang RREQ ke *node* tetangganya setelah menambahkan nilai *hop counter*. *Node* yang menerima RREQ dengan nilai *source address* dan *broadcast ID* yang sama dengan RREQ yang diterima sebelumnya, akan membuang RREQ tersebut. *Source sequence number* digunakan untuk memelihara informasi yang valid mengenai *reverse path* (rute balik) menuju ke *node* sumber. Pada saat RREQ mengalir menuju *node* tujuan yang diinginkan, dia akan menciptakan *reverse path* menuju ke *node*, setiap *node* akan membaca RREQ dan mengidentifikasi alamat dari *node* tetangga yang mengirim RREQ tersebut.

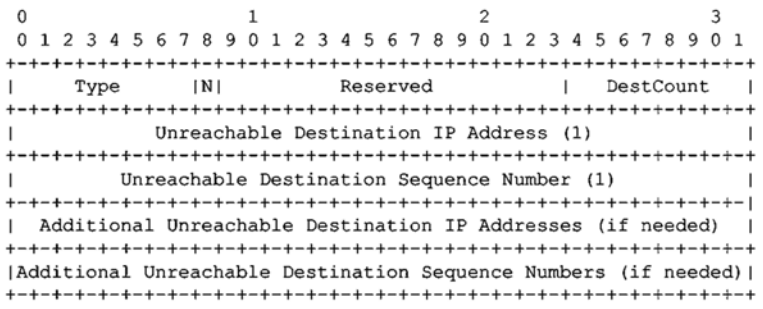


Gambar 2.3 Format Pesan RREP (Perkins, 1999)

Ketika *node* tujuan atau *node* yang memiliki informasi rute menuju *destination* menerima RREQ maka *node* tersebut akan membandingkan nilai *destination sequence number* yang dia miliki dengan nilai *destination sequence number* yang ada di RREQ. RREP akan dikirim menuju *node* sumber apabila nilai *destination squence number* yang ada di *node* lebih besar atau sama dengan nilai yang ada di RREQ, namun jika lebih besar maka akan di *broadcast* kembali ke *node* tetangganya. *Intermediate node* yang menerima RREP akan melakukan *update* informasi *route time out* (masa aktif rute) yang telah diciptakan. Informasi rute *source* ke *destination* akan dihapus apabila waktu *time out* habis.

2.2.2. Pemeliharaan Rute

Setiap *node* bertanggungjawab untuk memelihara informasi rute yang telah disimpan di dalam tabel *routing*. Bila terjadi perubahan topologi yang mengakibatkan suatu *node* tidak dapat dituju dengan menggunakan informasi rute yang ada pada tabel *routing*, maka *node* lain akan mengirim pesan RRER ke *node* tetangganya hingga *node* sumber menerima RRER tersebut. RERR yang dikirimkan berisi informasi sebagai berikut :

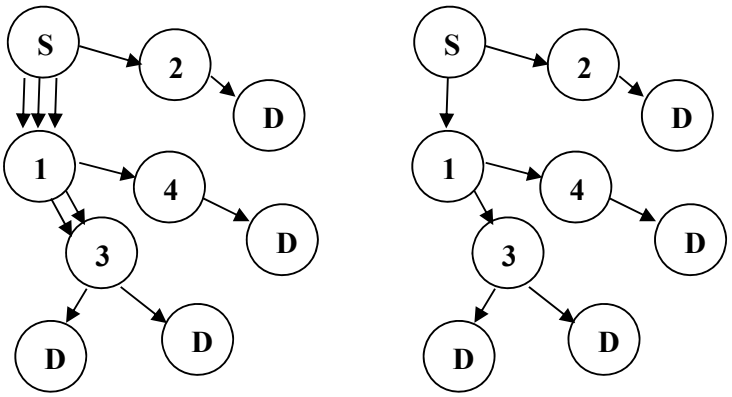


Gambar 2.4 Format Pesan RRER (Perkins, 1999)

Setiap *node* yang menerima pesan RRER akan menghapus informasi rute *routing* yang mengalami *error*. Kemudian *node* sumber akan melakukan *route discovery* kembali apabila rute tersebut masih diperlukan.

2.3 *Multicast Ad-hoc On-Demand Distance Vector* (MAODV)

Berdasarkan jumlah penerima paket, tipe transmisi data terbagi menjadi beberapa macam yaitu *unicast*, *broadcast*, *geocast* dan *multicast*. Beberapa aplikasi seperti *remote desktop* dan pengiriman *file* pada FTP mungkin membutuhkan transmisi *unicast* dengan satu alamat tujuan. Namun aplikasi seperti *video streaming* membutuhkan transmisi *multicast* untuk mengirimkan paket data ke beberapa tujuan dengan sekali pengiriman.

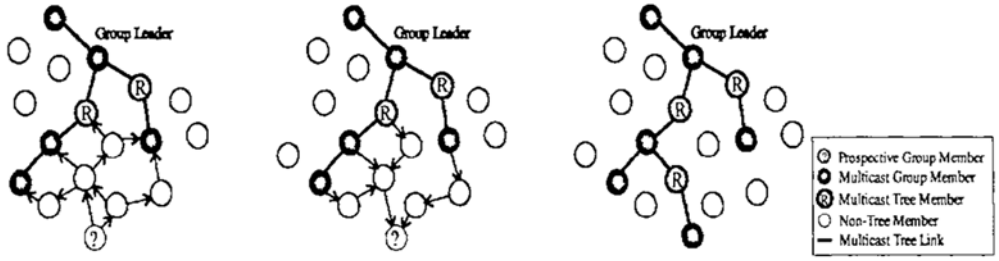


Gambar 2.5 Perbedaan Transmisi *Unicast* dan *Multicast*

Transmisi *unicast* merupakan transmisi yang umum diterapkan dalam jaringan *ad hoc* dimana suatu paket data dikirimkan dari satu *node* sumber ke satu *node* tujuan. Cara kerja transmisi *unicast* secara umum yaitu paket data yang akan dikirimkan terlebih dahulu diberi alamat tujuan. Kemudian paket tersebut dikirimkan melalui jaringan sesuai dengan rute yang sudah tersedia hingga paket tersebut sampai ke tujuan.

Pada metode transmisi *multicast*, paket data yang dikirimkan oleh *node* sumber disertai dengan alamat *multicast* tujuan agar dapat diterima oleh lebih dari satu *node* tujuan. Dengan kata lain, dalam transmisi *multicast* paket data akan dikirimkan ke alamat dalam kelompoknya saja. *Node* yang menjadi tetangga dari *node* sumber akan menggandakan paket data dan menentukan tujuan alamat masing-masing paket data tersebut melalui rute yang telah ditentukan ke dalam kelompok *multicast* sampai diterima oleh *node* tujuan.

Metode transmisi *multicast* diusulkan kedalam protokol *routing* AODV yang dinamakan *Multicast Ad hoc On-Demand Distance Vector* (Royer, 2000). Pada protokol *routing* MAODV, ditambahkan tabel *routing* yaitu tabel *multicast group* yang berisi *multicast group IP address*, *multicast group leader IP address*, *multicast group squence number*, *next hops*, *hop count to multicast group leader*, dan *hop count to multicast group member*. Selain itu juga ditambahkan tabel *group leader* berisi mengenai *multicast group IP address* dan *group leader IP address*. Format *header* RREQ juga dimodifikasi untuk memenuhi kebutuhan *multicast group* menjadi *<J_flag, R_flag, Broadcast_ID, Source_Addr, Source_Seq#, Dest_Addr, Dest_Seq#, Hop_Cnt>* dan *format header* RREP menjadi *<R_flag, U_flag, Dest_Addr, Dest_Seq#, Hop_Cnt, Lifetime>*.



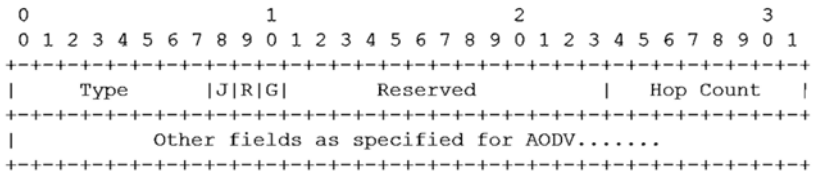
Gambar 2.6 Mekanisme RREQ, RREP dan *Multicast Tree* (Royer, 2000)

Dalam satu kelompok *multicast group*, mekanisme pengiriman data yang dipergunakan adalah protokol *routing* AODV. Antara satu sama lain *multicast group*, protokol *routing* MAODV yang berperan membentuk *tree* dan melakukan pemeliharaan rute. Jika sebuah *node* ingin menjadi *member* dari *multicast group* tertentu, maka *node* tersebut harus melakukan mekanisme *route discovery* terlebih dahulu. Setiap *node* dalam *multicast group* diatur oleh satu *node* yaitu *group leader*. *Group leader* berperan penting memelihara *group squence numbers* dari *multicast group* dari anggotanya. Oleh karena itu *group leader* selalu mengirimkan *group hello message* kepada anggotanya.

2.3.1. Format Pesan

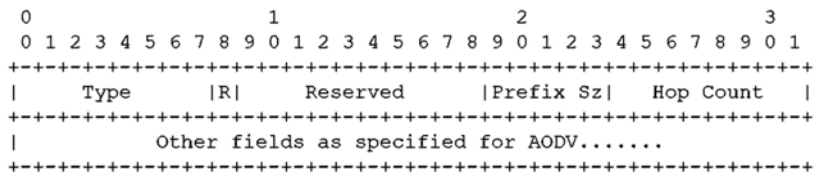
Terdapat beberapa perubahan fungsi pada format *header* dari pesan RREQ pada MAODV. Flag *J* (*join flag*) pada *header* RREQ menandakan bahwa *node*

ingin menjadi satu anggota dari *multicast group* tertentu. Ketika sebuah *node* ingin berkomunikasi secara *unicast* dengan *multicast group* menuju *group leader*, maka paket *header* memberikan *multicast group leader extension* sebagai informasi tambahan. Sedangkan *flag R* (*repair flag*) menandakan bahwa *node* ingin melakukan perbaikan terhadap *multicast tree* jika terjadi *broken link* maupun sejenisnya. Ketika sebuah rute terputus maka *node* melakukan perbaikan *multicast tree* dengan menambahkan *multicast group rebuild extension* pada *header* RREQ.



Gambar 2.7 Format Pesan RREQ pada MAODV (Royer, 2000)

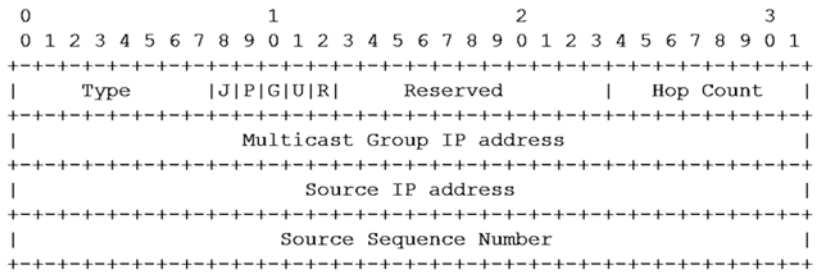
Pesan *RREP* mempunyai *flag R* (*repair flag*) yang menandakan bahwa *node* penerima menanggapi *flag R* dari RREQ node pengirim untuk melakukan perbaikan rute dan melakukan pembentukan ulang *multicast tree*.



Gambar 2.8 Format Pesan RREP pada MAODV (Royer, 2000)

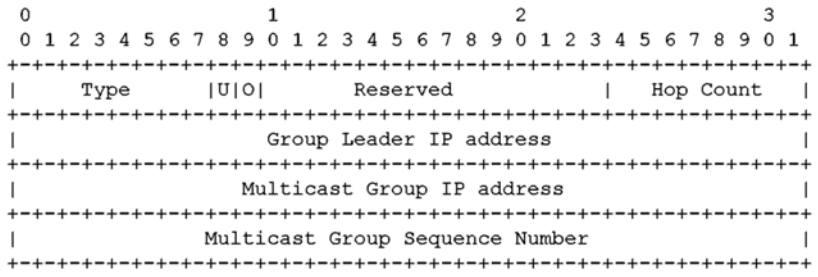
Pesan *multicast route activation* (MACT) ditujukan untuk pembentukan *multicast group* setelah sebuah *node* menerima RREP. Terdapat beberapa *flag* sebagai penanda pada mekanisme protokol *routing* MAODV. *Flag J* (*join flag*) ditujukan ketika *node* akan bergabung dengan *multicast group*. Berkebalikan dengan *flag P* (*prune flag*) dimana ditujukan ketika *node* akan melepaskan keanggotaan (*pruning*) dari *multicast group*. *Flag G* (*group leader flag*) ditujukan kepada anggota *multicast tree* jika terjadi kerusakan rute yang tidak bisa diperbaiki dan mengindikasikan untuk mencari *group leader* baru. *Flag U* (*update flag*) ditujukan kepada anggota *multicast tree* bahwa rute yang rusak telah diperbaiki dan informasi rute menuju *group leader* telah diperbaharui. Sedangkan *flag R* (*reboot*

flag) ditujukan ketika sebuah *node* melakukan proses *reboot*. Sedangkan *hop count* menandakan seberapa banyak jumlah *hop* dari *node* pengirim menuju *multicast group leader*. Format pesan dari MACT adalah sebagai berikut :



Gambar 2.9 Format Pesan MACT (Royer, 2000)

Mekanisme pemeliharaan rute oleh *group leader* dilakukan dengan mengirimkan pesan *group hello* (GRPH) kepada anggota *multicast group*. *Flag U* (*update flag*) dikirimkan ketika ada informasi dari group leader seperti rute, anggota, dan lain-lain. *Flag O* (*off_Mtree flag*) dikirimkan ketika *node* menerima pesan *group hello* dari *node* baru diluar *multicast tree*.



Gambar 2.10 Format Pesan GRPH (Royer, 2000)

2.3.2. Pembentukan *Multicast Tree*

Pada mekanisme pembentukan *multicast tree*, *node* sumber akan melakukan *broadcast* pesan RREQ terlebih dahulu untuk mencari rute menuju *destination node*. Pesan RREQ disertai *flag J* dikirimkan oleh *node* sumber jika ingin membentuk atau menjadi bagian kelompok *multicast group* dan begitu pula sebaliknya. Kemudian membuat daftar rute kosong pada tabel *multicast route* dan menjadikan dirinya sebagai *group member* tanpa *group leader address*. Hal tersebut dilakukan bilamana *node* sumber mendapatkan rute menuju *multicast group* lain

yang sudah terbentuk, maka *node* sumber akan mengisi informasi mengenai tabel *multicast* dan tabel *group leader*. Pesan *RREP* dengan *flag J* dipergunakan *node neighboor* untuk membalas pesan *node* sumber yang telah mengirimkan *route request* sebelumnya. Jika beberapa *node neighboor* mengirimkan pesan RREP kepada *node* sumber, maka *node* sumber memilih rute dari RREP dengan *squence number multicast group* yang lebih besar. Pesan *multicast route activation* (MACT) dipergunakan untuk menggabungkan informasi rute dari *node* maupun *multicast group* ke dalam *multicast group* lainnya. Kemudian membentuk rute sesuai informasi yang diperoleh dan menyimpannya kedalam tabel *multicast route*.

2.3.3. Pemeliharaan *Multicast Tree*

Jika beberapa *node neighboor* yang menerima RREQ dengan mengirimkan pesan RREP kepada *node* sumber, maka *node* sumber memilih rute dari RREP dengan *squence number multicast group* maupun *squence number* anggota *group* dari *multicast group* yang lebih besar. Setiap *node* yang memiliki anggota *multicast group* maupun menjadi penghubung antar *multicast group* bertugas untuk memelihara rute. Dan setiap *node* tersebut disebut dengan *group leader*, dimana setiap waktu tertentu melakukan *broadcast* pesan *group hello* (GPRH). Anggota *multicast group* maupun *node* penghubung antar *multicast group* melakukan *update* terhadap tabel *group leader*.

Untuk memelihara rute antar *node*, digunakan pesan *one-hop neighbor-hello*. Pada mekanisme pemeliharaan *multicast tree*, *node* pengirim pesan RREQ disebut dengan *upstream* serta *node* pengirim pesan RREP disebut dengan *downstream*. *Upstream* akan mengirimkan pesan *one-hop neighbor-hello* kepada *downstream* setiap waktu tertentu. Jika *downstream* merupakan *group leader* dari *multicast group* dan tidak menerima pesan *broadcast* dari *upstream*. Maka *downstream* menganggap bahwa rute telah terputus. Kemudian informasi rute menuju *upstream* pada *tabel multicast* dihapus dengan mengirimkan pesan *multicast activation* dengan *flag P* (MACT_P) kepada anggota *multicast tree*. Kemudian *downstream* akan mengirim RREQ kembali untuk mendapatkan rute yang baru sesuai dengan mekanisme pembentukan *multicast tree*. Setelah sebuah *node* terpilih menjadi *group leader* dari *upstream* tersebut, maka *group leader* akan mengirimkan

broadcast pesan GRPH dengan *flag U* (*update flag*). Jika *upstream* tidak menerima pesan RREP dari *downstream*, maka *upstream* akan melakukan hal yang sama yaitu menghapus informasi rute menuju *downstream* dengan mengirimkan MACT_P ke *multicast tree*.

2.4 *Road Static Unit* (RSU)

Selain komunikasi antar kendaraan atau *Vehicle to Vehicle Communication* (V2V), VANET juga mendukung komunikasi dengan infrastruktur yang dipasang secara statik atau *Vehicle to Infrastructure Communication* (V2I). Salah satunya adalah penelitian tentang *static intersection node* (SIN) dengan transmisi *multicast* yang adaptif (Anggoro, R., 2008). Dimana SIN yang menentukan jarak terpendek antara *node* sumber dan *node* tujuan dengan cara membandingkan jarak antara posisi *node* sumber berada beserta *delay* dengan semua SIN yang berdekatan.

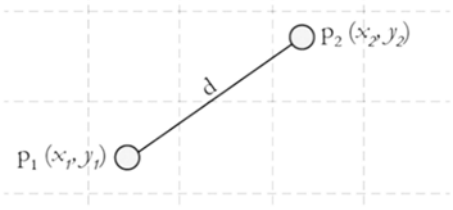
Penelitian tentang *hybrid communication* untuk mencegah terjadinya kecelakaan (Festag, A. et al, 2008) mengusulkan ketika sebuah kendaraan masuk kedalam jangkauan sinyal RSU maka kendaaraan tersebut akan mendapatkan data terbaru untuk wilayah tersebut (cuaca, halangan dll) sehingga setelah dilakukan pemrosesan data dapat diketahui kondisi jalan didepannya. Infrastruktur dapat berupa *Wireless Access Point* yang disebut dengan SIN (Calvacante, et al 2012).

Shortest Path Based Traffic Aware Routing (STAR) adalah sebuah protokol yang dikembangkan (Ramachandran, L. et al 2013) untuk memanfaatkan lampu lalu lintas persimpangan jalan dengan asumsi kendaraan yang sedang berhenti di lampu merah memastikan *end to end connectivity*. Namun *broadcast hello message* dapat menyebabkan *broadcast storm* sehingga diusulkan untuk menggunakan teknik *Red Light First Forwarding* (RLFF). Ketika *node* sampai di persimpangan jalan, maka kendaraan yang sedang berhenti karena lampu merah akan meneruskan data tersebut.

2.5 *Euclidean Distance*

Setiap *node* yang bergerak maupun diam mempunyai informasi posisi *node* yang diasumsikan sebagai titik koordinat (*x,y*) pada bidang kartesius. Perpindahan

posisi *node* direpresentasikan dalam koordinat dua dimensi (*x,y*) yaitu $P_1 = (x_1,y_1)$ dan $P_2 = (x_2,y_2)$ pada gambar dibawah ini :



Gambar 2.11 Jarak *d* Antara *Node* P_1 dengan *Node* P_2

Perhitungan jarak *euclidean* antar kedua titik diatas mempunyai variabel koordinat (*x,y*) dapat dihitung dengan persamaan :

$$d = \sqrt{|P_1|^2 + |P_2|^2}$$
$$= \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \quad (2.1)$$

dengan :

P_1 = titik P_1

P_2 = titik P_2

d = jarak antara P_1 dan P_2

Perhitungan jarak *euclidean* dari titik P_1 ke titik P_2 mendukung perhitungan lebih dari tiga (*n*) variabel antara kedua objek sehingga dapat disimpulkan :

$$\mathrm{d_{op}} = \sqrt{\sum_{i=1}^{n} |\mathrm{o_n} - p_n|^2} \quad (2.2)$$

dengan :

d_{op} = Jarak antara titik O dan P

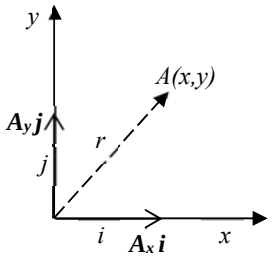
n = Banyaknya variabel

o_n = Variabel O ke *n*

p_n = Variabel P ke *n*

2.6 Vektor Posisi, Kecepatan dan Jarak

Vektor satuan digunakan untuk menunjukkan arah dan posisi dalam suatu bidang atau ruang. Posisi titik materi pada suatu bidang dapat dinyatakan dalam bentuk vektor. Vektor satuan *i* dan *j* dan masing-masing menyatakan arah sumbu *x* dan *y* positif dalam bidang kartesian. Sebuah vektor *A* berada pada bidang *xy*, hasil komponen vektor A_x dan vektor satuan *i* adalah vektor $A_x i$ yang berada pada sumbu *x* dan besarnya $|A_x|$. Demikian dengan vektor $A_y j$ yang berada pada sumbu *y* dan besarnya $|A_y|$. Sehingga vektor satuan untuk vektor *A* adalah :



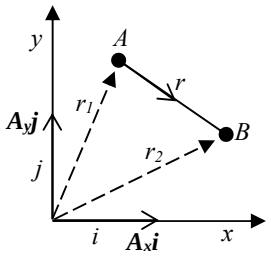
Gambar 2.12 Vektor Posisi

Jika, $r = x_i + y_j$
dan, $A = A_x i + A_y j$ (2.3)

dengan :

r = vektor perpindahan
x_i = vektor satuan *i* yang berada pada sumbu *x*
y_j = vektor satuan *j* yang berada pada sumbu *y*
A = vektor *A*
$A_x i$ = vektor *A* dengan satuan *i* yang berada pada sumbu *x*
$A_y j$ = vektor *A* dengan satuan *j* yang berada pada sumbu *y*

Kecepatan merupakan perpindahan posisi suatu benda terhadap satuan waktu. Kecepatan rata-rata ($\tilde{v}$) dirumuskan sebagai perbandingan antara perpindahan (Δr) yang ditempuh benda terhadap selang waktu (Δt).



Gambar 2.13 Vektor Perpindahan

Gerak sebuah benda pada bidang *x,y* dapat dirumuskan jika posisi benda berpindah terhadap selisih waktu tertentu. Ketika posisi benda berpindah dari titik awal A ke titik akhir B adalah *Δ*r dengan selang waktu *Δ*t maka :

$$\tilde{v} = \frac{\Delta r}{\Delta t}$$
$$= \frac{rB - rA}{tB - tA} \quad (2.4)$$

dengan :

$\tilde{v}$ = kecepatan rata-rata

t = waktu

Kecepatan dapat diperoleh dari diferensial persamaan jarak vektor *x* dan *y* terhadap waktu. Jika posisi benda $r = x_i + y_j$ dan $\Delta r = \Delta x_i + \Delta y_j$ maka dapat diperoleh :

$$v = \frac{dr}{dt}$$
$$\frac{dr}{dt} = \frac{dx}{dt} i + \frac{dy}{dt} j$$
$$v = v_x i + v_y j \quad (2.5)$$

dengan :

dx = diferensial posisi pada sumbu *x*

dy = diferensial posisi pada sumbu *y*

$v_x i$ = kecepatan vektor *i* pada sumbu *x*

$v_y j$ = kecepatan vektor *j* pada sumbu *y*

Posisi dapat diperoleh dari integral persamaan kecepatan vektor *x* dan *y* terhadap waktu. Jika :

$$dr = v * dt$$

$$\int_{t_0}^{t} dr = \int_{t_0}^{t} v * dt$$

$$r - r_0 = \int_{t_0}^{t} v * dt$$

$$r = r_0 + \int_{t_0}^{t} v * dt \quad (2.6)$$

Jika *r* berubah terhadap waktu dengan persamaan vektor posisi $r = x_i + y_j$ dan vektor kecepatan $v = v_x i + v_y j$ maka posisi *x,y* dapat dirumuskan sebagai berikut :

$$x = x_0 + \int_{t_0}^{t} v_x * dt \quad (2.7)$$

$$y = y_0 + \int_{t_0}^{t} v_y * dt \quad (2.8)$$

dengan :

x = Posisi akhir B pada sumbu *x*

y = Posisi akhir B pada sumbu *y*

x_0 = Posisi awal A pada sumbu *x*

y_0 = Posisi awal A pada sumbu *y*

v_x = kecepatan vektor *i* pada sumbu *x*

v_y = kecepatan vektor *j* pada sumbu *y*

2.7 *Weighted Product*

Metode *Weighted Product* merupakan metode penentuan prioritas dalam analisis multi kriteria pada *multi atributte decision making* (MADM). Seperti pada penelitian (Anupama, K.S.S et al, 2015) tentang penentuan algoritma penggunaan jaringan nirkabel (UMTS1, UMTS2, Wifi, WiMax) pada kasus pengiriman data tertentu. Contohnya pada aplikasi percakapan dan *streaming* yang sensitif terhadap

delay dan *jitter* meskipun tidak menghabiskan trafik data yang besar. Dengan membandingkan beberapa metode MADM diantaranya *Preference Ranking Organization Method for Enrichment Evaluation* (PROMETHEE), *Simple Additive Weighting* (SAW) dan *Weighted Product Method* (WPM) untuk menentukan penggunaan jaringan nirkabel terbaik dalam pengiriman data tertentu seperti data percakapan dan *streaming*.

Pada penelitian tersebut disimpulkan bahwa dengan menggunakan metode PROMETHEE menentukan penggunaan jaringan lebih optimal menggunakan jaringan *wireless* UMTS1 dibandingkan UMTS2, Wifi maupun WiMax ketika dibutuhkan untuk pengiriman data percakapan dan *streaming* yang membutuhkan atribut atau kriteria khusus seperti *delay* dan *jitter*. Ketika pengiriman data heterogen, baik percakapan, *streaming*, pengiriman *file*, aplikasi yang membutuhkan sinkronisasi, dan lain-lain berjalan bersama dalam melakukan pengiriman data, maka metode penentuan yang terbaik yaitu SAW dan WPM.

Pada penelitian lain (Savitha, K., 2011) mengenai proses pengalihan penggunaan konektivitas perangkat mobile (proses *Handover*) juga membutuhkan penentuan dengan mempertimbangkan *delay, banwidth, cost* dan *jitter*. Pada penelitian tersebut dipilih metode SAW dan WPM untuk menentukan penggunaan konektivitas perangkat yaitu Wifi dan WiMax. Dari hasil penelitian tersebut, metode WPM menunjukkan hasil penentuan yang lebih baik 35,75% dibandingkan metode SAW 12.64%.

Metode WPM mengevaluasi beberapa alternatif terhadap sekumpulan atribut atau kriteria, dimana setiap atribut saling tidak bergantung satu dengan yang lainnya. Menggunakan teknik perkalian untuk menghubungkan rating atribut, dimana rating tiap atribut harus dipangkatkan terlebih dahulu dengan bobot atribut seperti proses normalisasi. Preferensi untuk *alternative* S_i yaitu sebagai berikut :

$$S_i = \prod_{j=1}^{n} X_{ic}^{w_c} \tag{2.9}$$

dengan :

S = Preferensi alternatif sebagai vektor *S*

i = Alternatif

n = Banyaknya kriteria

X = Nilai kriteria

w = Bobot kriteria

c = Kriteria

Proses selanjutnya yaitu preferensi alternatif vektor V untuk langkah terakhir melakukan tahap perangkingan. Preferensi untuk *alternative* V_i diberikan sebagai berikut :

$$V_i = \frac{S_i}{\sum S_i} \tag{2.10}$$

dengan :

V_i = Preferensi alternatif sebagai vektor V

S_i = Preferensi alternatif sebagai vektor S

BAB 3
METODOLOGI PENELITIAN

3.1 Rancangan Protokol *Routing*

Rancangan sistem protokol *routing Multicast Adaptif Structured-Tree Based on Reactive Euclidean Node Knowledge* (MAS-BRENK) dikembangkan dari protokol *routing* MAODV. Beberapa bagian dijelaskan pada sub bab selanjutnya.

3.1.1. *Euclidean Node Knowledge* (ENK)

Perhitungan jarak *euclidean* antar *node* dengan mempertimbangkan parameter posisi, kecepatan dan *delay* transmisi dalam penelitian ini disebut sebagai *euclidean node knowledge* (ENK). Perhitungan ENK dipergunakan untuk memutuskan proses *join multicast group* (jika belum mempunyai *multicast group*) atau *prune multicast group* (jika sudah menjadi anggota *multicast group*). Perhitungan tersebut juga dipergunakan untuk melakukan pemilihan *group leader* dalam *multicast group*. Posisi *node* menjadi acuan penting dalam perhitungan ENK. Posisi masing-masing *node* dinotasikan sebagai berikut :

$$\mathrm{Pos_i(t)} = \begin{bmatrix} x_i + dx_i.t \\ y_i + dy_i.t \end{bmatrix} \tag{3.1}$$

dengan :

$\mathrm{Pos_i(t)}$ = posisi *node i* pada waktu *t*

$[x_i, y_i]^T$ = posisi awal *node i*

$[dx_i, dy_i]^T$ = kecepatan *node i*

Delay transmisi menjadi pertimbangan penting, dikarenakan *delay* pengiriman paket data akan selalu berubah setiap waktu karena mobilitas kendaraan yang sangat dinamis. Maka perhitungan jarak *euclidean* diusulkan dengan menyertakan waktu pengiriman data ditambah dengan *delay* transmisi, dinotasikan sebagai t_d. Ketika sebuah *node* menerima *request* maupun *reply* yang berisi informasi posisi, kecepatan dari *node* pengirim. *Delay* transmisi menyebabkan

informasi yang diterima menjadi kurang valid. Maka informasi *node* pengirim (*node i*) yang diterima oleh *node* penerima (*node j*) harus divalidasi.

Diasumsikan percepatan *node* tidak berubah atau mengalami percepatan konstan. Maka posisi *node* pengirim harus divalidasi oleh *node* penerima untuk menghitung jarak antar *node* tersebut. Jika kecepatan *node* pengirim adalah v_{0i} dan percepatan a_i maka *node* penerima harus melakukan validasi, dengan menghitung perpindahan posisi *node* pengirim (s_i) dari posisi awal (x,y) menuju posisi akhir terhadap *delay* transmisi. Dengan perhitungan jarak, perpindahan posisi *node i* pada sumbu x dan y dapat dinotasikan sebagai berikut :

$$\mathrm{Pos_i}(t_j) = \begin{bmatrix} x_i + (dx_i.(t_j - t_i)) \\ y_i + (dy_i.(t_j - t_i)) \end{bmatrix} \tag{3.2}$$

dengan :

$\mathrm{Pos_i(t)}$ = posisi *node i* pada waktu *t*

t_i = waktu node *i* mengirimkan pesan

t_j = waktu node *j* menerima pesan

Setelah informasi posisi *node i* valid, maka *node j* dapat menentukan posisi dan jarak *euclidean* terhadap *node i*. Jarak euclidean *node j* terhadap *node i* dapat dirumuskan sebagai berikut :

$$\begin{aligned} \mathrm{d_{ij}(t_j)} &= \sqrt{\mathrm{Pos_i}(t_j)^2 - \mathrm{Pos_j}(t_j)^2} \\ &= \sqrt{|\mathrm{x_i} - x_\mathrm{j}|^2 + |\mathrm{y_i} - \mathrm{y_j}|^2} \end{aligned} \tag{3.3}$$

dengan :

$\mathrm{d_{ij}(t)}$ = jarak *euclidean node j* terhadap *node i*

x_i = posisi akhir *node i* pada sumbu *x*

y_i = posisi akhir *node i* pada sumbu *y*

v_i = kecepatan akhir node *i*

t_j = waktu node *j* menerima pesan dari *node i*

Setelah diketahui jarak *euclidean node j* terhadap *node i* sebagai salah satu parameter perhitungan ENK. Batas maksimal ENK diambil dari perhitungan terbesar perkalian bobot *node j* terhadap *node i* dan *node* lainnya yang telah mengirimkan informasi pada pesan MACT. Perhitungan batas maksimal ENK dihitung dengan rumus *weighted product* mempertimbangkan parameter jarak *euclidean*, kecepatan dan *delay* transmisi. Keempat parameter tersebut menunjukkan banyaknya kriteria, dibagi menjadi empat bagian yaitu kriteria jarak (c_1), kriteria kecepatan (c_2), dan kriteria *delay* transmisi (c_3). Alternatif *i* merupakan *node i* yang mengirim pesan MACT berisi informasi *node* pengirim yang diterima oleh *node j*. Sebagai berikut :

$$S_i = \prod_{j=1}^{n} X_{ic}^{w_c} \tag{3.4}$$

dengan :

S = Preferensi alternatif sebagai vektor *S*

i = Alternatif *node*

n = Banyaknya kriteria

X = Nilai kriteria

w = Bobot masing-masing kriteria

c = Kriteria informasi *node*

Langkah perhitungan bobot ENK adalah sebagai berikut :

1. Alternatif *i* merupakan *node* yang mengirimkan pesan MACT dan pesan tersebut diterima oleh *node j*. Contoh jika ada tujuh *node* yang mengirimkan pesan MACT, maka ketujuh *node* tersebut merupakan alternatif *node*.
2. Kriteria *c* yaitu kriteria jarak, kecepatan dan *delay* transmisi.
3. Bobot awal kriteria dibagi menjadi tiga tingkatan. Bobot terbesar dengan nilai tiga, yaitu pada kriteria *delay* transmisi (w_c3). Jarak (w_c1) mempunyai nilai bobot dua. Kecepatan (w_c2) mempunyai nilai bobot satu.
4. Jumlah bobot keseluruhan Σw_c adalah 1. Dan bobot dari masing-masing kriteria (w_1, w_2, w_3) adalah bobot awal kriteria dibagi jumlah bobot keseluruhan.

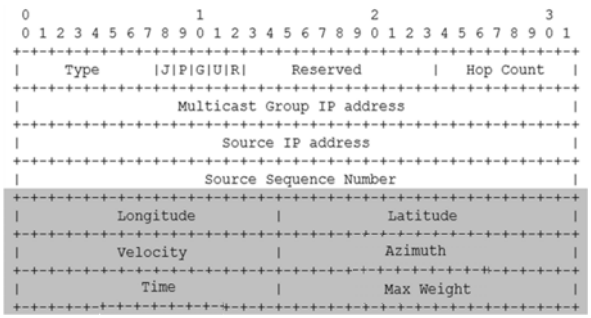
5. Vektor S_i dihitung dari kriteria *c* dari masing-masing alternatif *node* dipangkatkan dengan bobot.
6. Perangkingan alternatif *node* dapat dilakukan dengan perhitungan vektor V_i dimana setiap vektor S_i dibagi jumlah keseluruhan dari vektor S_i.

Masing - masing alternatif *node* mempunyai nilai *Vi* yang dapat dipergunakan untuk menentukan batas maksimal dan minimal. Dari sini dapat ditentukan bahwa alternatif *node* dengan nilai *Vi* terbesar merupakan batas maksimal ENK. Alternatif *node* dengan nilai *Vi* terkecil dapat dijadikan sebagai pertimbangan pemilihan *group leader* pada *multicast group*.

3.1.2. Pembentukan *Structured-tree*

Pembentukan *structured-tree* pada protokol *routing* MAODV dilakukan dengan mengirimkan propagasi pesan *multicast* RREQ hingga mencapai *node* tujuan. Setelah pesan *multicast* RREQ diterima oleh *node* tujuan, maka *node* tujuan mengirimkan *multicast* RREP menuju *node* sumber. Sehingga *node* sumber dapat mengetahui rute mana saja yang dapat dilalui menuju *node* tujuan. Setelah rute terbentuk, *node* sumber dapat melakukan proses pembentukan *multicast group*. Pesan *multicast activation* (MACT) dikirimkan oleh *node* sumber untuk membentuk *multicast group*.

Modifikasi protokol *routing* yang diusulkan, dilakukan dengan menambahkan perhitungan jarak *euclidean* menggunakan parameter posisi, kecepatan dan waktu pengiriman. Ketika sebuah *node* melakukan aktivasi jalur *multicast*, maka *node* tersebut mengirimkan pesan MACT dengan *flag _j*. Pada pengiriman pesan MACT ditambahkan skrip parameter posisi, kecepatan dan *delay* transmisi. Paket *header* pada pesan MACT dimodifikasi dengan menambahkan parameter tersebut untuk dilakukan perhitungan bobot. Setelah rute *multicast* terbentuk, maka *node* akan melakukan aktifasi rute dengan mengirimkan pesan MACT. Hal tersebut bertujuan untuk melakukan validasi bahwa pengiriman data dilakukan dari rute yang sudah terbentuk. Pesan MACT dikirimkan dengan format sebagai berikut:



Gambar 3.1 Perubahan Paket *Header* pada Pesan MACT

Setelah *multicast group* terbentuk, maka pembentukan *multicast group* yang saling terhubung akan membentuk *multicast tree*. Sehingga pengiriman paket dapat dilakukan menuju ke beberapa *node* tujuan.

```
struct hdr_aodv_mact {
  u_int8_t  mact_type;
  u_int8_t  mact_flags;
  u_int8_t  reserved;
  u_int8_t  mact_hop_count;
  nsaddr_t  mact_grp_dst;
  nsaddr_t  mact_src;
  u_int32_t mact_src_seqno;

  //tambahan masbrenk
  double  mact_x;
  double  mact_y;
  double  mact_dx;
  double  mact_dy;
  double  mact_speed;
  double  mact_delay;
  double  mact_enk;
```

Gambar 3.2 Penambahan Parameter pada *Header* MACT

```
mact->mact_type = AODVTYPE_MACT;
mact->mact_flags = flags;
mact->mact_hop_count = hop_count;
mact->mact_grp_dst = dst;
mact->mact_src = index;
mact->mact_src_seqno = seqno;

//tambahan masbrenk
iNode = (MobileNode *) (Node::get_node_by_address(index));
mact->mact_x = iNode->X();
mact->mact_y = iNode->Y();
mact->mact_dx = iNode->dX();
mact->mact_dy = iNode->dY();
mact->mact_speed = iNode->speed();
mact->mact_delay = CURRENT_TIME;
```

Gambar 3.3 Penambahan Parameter pada Pesan MACT

Ketika salah satu node dipilih menjadi *group leader*, dimana *group leader* bertanggungjawab untuk memelihara atau *maintain* rute terhadap anggota *multicast group* dan *multicast group* lainnya. Pemeliharaan rute ditandai dengan

dikirimkannya pesan *group hello* (GPRH) oleh *group leader* kepada anggota *multicast group* dan *group leader* dari *multicast group* lain.

Dengan adanya perhitungan ENK, maka *group leader* dapat mengetahui informasi *node* dan rute yang diperlihara. Selain itu *group leader* bertanggungjawab untuk melakukan pencarian rute kembali ketika salah satu *node* yang dipelihara akan keluar dari keanggotaan *multicast group*. Keluarnya *node* dari keanggotaan *multicast group* ditandai dengan dikirimnya pesan MACT dengan *flag _p* oleh *group leader* kepada anggota *multicast group*.

3.1.3. *Adaptif Structured-tree* (*Joining, Pruning*)

Dari setiap pembentukan *multicast group* antar *node* dengan mobilitas kendaraan yang berubah-ubah setiap waktu, maka sangat dimungkinkan terjadi *link breakage* dan *membership revocation*. *Link breakage* merupakan kejadian kerusakan link pada *tree* karena kondisi tertentu. Terjadinya kerusakan link menyebabkan struktur *tree* akan terpecah. Terpecahnya struktur *tree* ditandai dengan pengiriman pesan MACT dengan *flag _p*. Maka *node neighbour* harus melakukan *join* untuk memperbaiki *multicast tree* (*restructured-tree*), dimana *group leader* akan melakukan pencarian rute baru dengan mengirimkan pesan MACT ditandai dengan *flag _j* (*join*). Penerima pesan MACT dengan *flag _j* akan melakukan perhitungan ENK kembali dan memutuskan apakah *node* tersebut dapat bergabung dalam kelompok *multicast group*.

Membership revocation merupakan pencabutan keanggotaan *node* oleh *group leader* dari anggota *multicast group* karena sebab tertentu. Dengan perhitungan ENK, maka keluarnya *node* dari keanggotaan *multicast group* tertentu dapat diprediksi sehingga akibat yang akan terjadi yaitu *link breakage* dapat segera ditangani. Keluarnya *node* dari keanggotaan *multicast group* (*pruning*) menandakan bahwa *node* tersebut diprediksi tidak mampu lagi melanjutkan pengiriman data karena sebab tertentu. Dengan perhitungan ENK, maka *group leader* akan mengirimkan MACT dengan *flag _p* untuk mencabut keanggotaan *node* tersebut dari *multicast group*. Sehingga *node* yang menerima pesan tersebut akan menghapus rute *multicast*

Anggota *group* (*group member*) menjadi tanggung jawab *group leader* pada pemeliharaan rute. Meskipun bobot ENK masih berada pada nilai kurang dari batas maksimal, *group leader* bisa saja terputus dari *multicast tree* karena sebab tertentu. Jika hal tersebut terjadi maka *group member* harus melakukan pencarian rute kembali untuk membentuk *multicast tree* yang terputus. Dalam hal ini perhitungan ENK tidak lagi dipertimbangkan dalam melakukan penghapusan rute karena rute menuju *group leader* sudah dianggap terputus. Sehingga *group member* akan melakukan proses pencarian rute kembali menuju *multicast tree.* Hal tersebut merupakan pertimbangan penting mengingat pergerakan pada VANET sangat dinamis, dan dibutuhkan sebuah *node* untuk melakukan pemeliharaan rute.

3.2 Rancangan Mobilitas

Untuk mengetahui kinerja dari metode yang diusulkan maka perlu dilakukan uji coba. Dalam penelitian ini skenario mobilitas *node* menggunakan simulator SUMO untuk membangun sebuah peta grid (Dimyati, M. et al., 2016). Peta grid yang dirancang mempunyai jumlah garis x (horisontal) 10 dan jumlah garis y (vertikal) 10. Panjang persimpangan pada garis horisontal 200 m dan garis vertikal 200m, sehingga ukuran peta grid keseluruhan adalah 2000m x 2000m. Model pergerakan kendaraan yang digunakan yaitu sesuai karakteristik kendaraan dengan kecepatan antara 0 - 10 m/s (36 km/jam) (Dorle et al., 2011). Kepadatan kendaraan juga disesuaikan dengan kondisi daerah perkotaan yaitu 50 *node* (Dorle et al., 2011; Vidhale, B. & Dorle, S., 2011). Parameter simulasi dapat dilihat pada Tabel 3.1.

Tabel 3.1 Parameter Mobilitas

No	Parameter	Spesifikasi
1.	Waktu Simulasi	900s / 15menit
2.	Area Simulasi	2000m x 2000m
3.	Jumlah Kendaraan	50 *node*
4.	Kecepatan	0 – 10 m/s
5.	Model Mobilitas	Perkotaan (*urban*)

Untuk membuat peta grid dengan area simulasi 2000m x 2000m yang terbagi dalam jumlah garis x (horisontal) dan y (vertikal) serta panjang persimpangan diatas, perintah pada SUMO dapat dilihat pada gambar dibawah:

```
netgenerate --grid --grid.length 200 --grid.number 10 --default.lanenumber 2
--default.speed 10 --keep-edges.min-speed 1 --output-file nyoba1.net.xml
```

Gambar 3.4 Perintah Membuat Peta

Setiap kendaraan mempunyai titik awal dan tujuan yang telah ditentukan. Modul *randomTrips.py* digunakan untuk menentukan titik awal dan tujuan dari setiap kendaraan. Untuk menghasilkan 50 kendaraan, dibutuhkan waktu 51 detik. Perintah untuk menghasilkan kendaraan dan titik awal serta tujuannya menggunakan modul *randomTrips.py* sebagai berikut:

```
"C:\Users\MasPipis\AppData\Local\Programs\Python\Python35\python.exe"
"D:\Program Files (x86)\DLR\Sumo\tools\randomTrips.py" -n nyoba1.net.xml
--intermediate 100 -l -e 51 -o nyoba1.trip.xml
```

Gambar 3.5 Perintah Membuat Mobilitas Kendaraan

Kendaraan yang telah dihasilkan dari perintah sebelumnya, kemudian dilakukan penentuan rute mobilitas menggunakan modul *duarouter.exe*. Rute dibentuk berdasarkan asal dan tujuan setiap kendaraan dengan waktu simulasi sesuai parameter yang telah ditentukan yaitu 900s. Perintah untuk membuat rute kendaraan sebagai berikut:

```
"D:\Program Files (x86)\DLR\Sumo\bin\duarouter.exe" -n nyoba1.net.xml -t
nyoba1.trip.xml --route-steps=900 -b 1 -e 900 -o nyoba1.rou.xml
--ignore-errors true --randomize-flows=true
```

Gambar 3.6 Perintah Membuat Rute Kendaraan

Tahap terakhir dalam pembuatan skenario yaitu menggabungkan semua file yang telah dibuat sebelumnya dengan konfigurasi file *sumo.cfg* dengan konfigurasi sebagai berikut:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<configuration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://sumo.sf.net/xsd/sumoConfiguration.xsd">
    <input>
        <net-file value="nyoba1.net.xml"/>
        <route-files value="nyoba1.rou.xml"/>
    </input>
    <time>
        <begin value="0"/>
        <end value="900"/>
    </time>
    <processing>
        <time-to-teleport value="-1"/>
    </processing>
    <report>
        <xml-validation value="never"/>
        <no-duration-log value="true"/>
        <no-step-log value="true"/>
    </report>
</configuration>
```

Gambar 3.7 Konfigurasi Skenario Mobilitas

Hasil konfigurasi tersebut harus diintegrasikan terlebih dahulu dengan Network Simulator. File *xml* diubah menggunakan modul *traceExporter.py* dengan perintah *–ns2mobility-output <nama-file.tcl>* sebagai berikut:

```
sumo -c nyoba1.sumocfg --fcd-output nyoba1.xml
"D:\Program Files (x86)\DLR\Sumo\tools\traceExporter.py" --fcd-input
nyoba1.xml --ns2mobility-output nyoba1.tcl
```

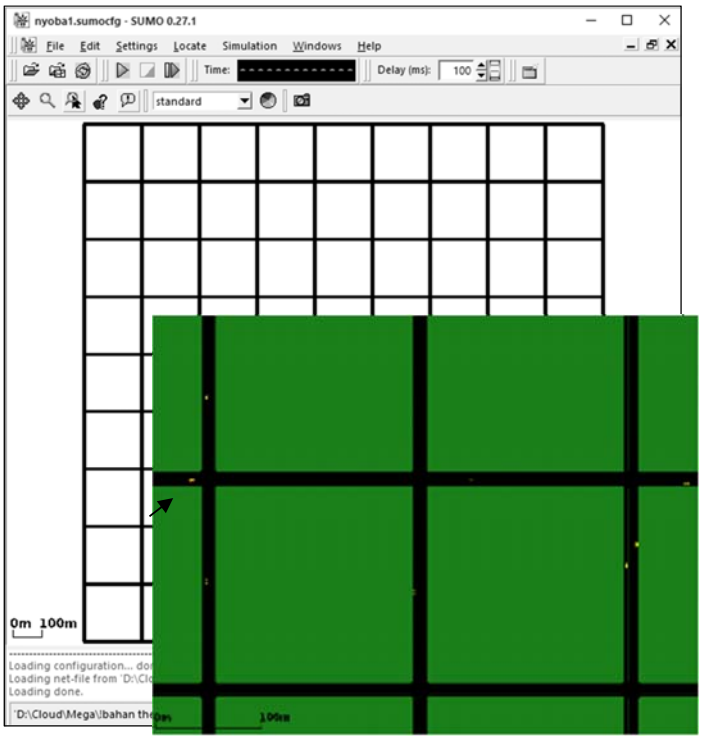
Gambar 3.8 Integrasi Skenario Mobilitas

Dari perintah konfigurasi diatas, dihasilkan skenario mobilitas yang dapat diintegrasikan kedalam aplikasi *Network Simulator* sebagai berikut:

```
$ns_ at 55.0 "$node_(7) setdest 625.39 1395.05 18.76"
$ns_ at 55.0 "$node_(8) setdest 427.12 1595.05 19.63"
$ns_ at 56.0 "$node_(0) setdest 195.05 1086.31 19.27"
$ns_ at 56.0 "$node_(9) setdest 1204.95 823.74 19.66"
$ns_ at 56.0 "$node_(10) setdest 4.95 1578.89 18.95"
$ns_ at 56.0 "$node_(11) setdest 795.05 1384.14 19.96"
$ns_ at 56.0 "$node_(12) setdest 1404.2 1404.95 6.25"
$ns_ at 56.0 "$node_(13) setdest 423.85 1801.65 19.50"
$ns_ at 56.0 "$node_(14) setdest 1395.05 1213.23 19.64"
$ns_ at 56.0 "$node_(15) setdest 604.95 364.27 19.35"
$ns_ at 56.0 "$node_(16) setdest 1204.95 366.0 19.19"
$ns_ at 56.0 "$node_(17) setdest 1053.63 404.95 18.83"
$ns_ at 56.0 "$node_(18) setdest 647.33 604.95 18.97"
$ns_ at 56.0 "$node_(1) setdest 918.69 1004.95 19.26"
$ns_ at 56.0 "$node_(19) setdest 1663.57 1204.95 19.62"
$ns_ at 56.0 "$node_(20) setdest 404.95 1310.72 19.90"
$ns_ at 56.0 "$node_(21) setdest 1480.89 1404.95 18.93"
$ns_ at 56.0 "$node_(22) setdest 404.95 503.23 19.88"
$ns_ at 56.0 "$node_(23) setdest 1521.49 1804.95 19.25"
$ns_ at 56.0 "$node_(24) setdest 395.05 1318.9 19.63"
$ns_ at 56.0 "$node_(25) setdest 604.95 1063.84 19.49"
$ns_ at 56.0 "$node_(26) setdest 941.13 1404.95 18.85"
$ns_ at 56.0 "$node_(27) setdest 595.05 1534.68 19.92"
$ns_ at 56.0 "$node_(28) setdest 1001.65 234.52 19.81"
$ns_ at 56.0 "$node_(2) setdest 795.05 1119.87 18.76"
$ns_ at 56.0 "$node_(29) setdest 1801.65 438.76 19.92"
$ns_ at 56.0 "$node_(30) setdest 398.35 1560.37 19.37"
$ns_ at 56.0 "$node_(31) setdest 395.05 1767.73 18.87"
```

Gambar 3.9 Potongan Skrip Hasil Integrasi Skenario Mobilitas

Skenario mobilitas diatas dapat dijalankan langsung tanpa protokol *routing* seperti pada gambar dibawah ini:



Gambar 3.10 Simulasi Skenario Mobilitas

3.3 Skenario Mobilitas

Berikut perangkat lunak yang digunakan untuk pengembangan metode dari protokol *routing* yang diusulkan sebagai berikut:

a. Sistem operasi CentOS 6.2 32bit untuk lingkungan simulasi protokol *routing* NS-2.26,
b. Sistem operasi Windows 10 64bit untuk lingkungan simulasi mobilitas dan skenario SUMO v.0.27.1,

Untuk perangkat keras yang digunakan sebagai lingkungan implementasi perangkat lunak dalam penelitian ini adalah sebagai berikut:

a. Processor Intel(R) Core(TM) i5 2410m 2.3GHz
b. RAM 8GB DDR3
c. Media penyimpanan sebesar 10GB

Dari potongan skrip hasil integrasi skenario mobilitas pada Gambar 3.9 dapat dilihat bahwa pergerakan *node* telah dibuat secara acak oleh SUMO. Satu contoh pergerakan *node* yang bergerak sesuai karakteristik VANET yaitu *node_(23)* dapat dilihat pada gambar berikut :

```
$node_(23) set X_ 1604.95
$node_(23) set Y_ 1613.15
$node_(23) set Z_ 0.000000000000
$ns_ at 24.0 "$node_(23) setdest 1604.95 1613.15 0.00"
$ns_ at 25.0 "$node_(23) setdest 1604.95 1614.56 1.41"
$ns_ at 26.0 "$node_(23) setdest 1604.95 1618.54 3.98"
$ns_ at 27.0 "$node_(23) setdest 1604.95 1624.06 5.52"
$ns_ at 28.0 "$node_(23) setdest 1601.65 1631.36 7.30
$ns_ at 29.0 "$node_(23) setdest 1601.65 1640.28 8.92"
$ns_ at 30.0 "$node_(23) setdest 1601.65 1649.15 8.87"
$ns_ at 31.0 "$node_(23) setdest 1601.65 1658.71 9.56"
$ns_ at 32.0 "$node_(23) setdest 1601.65 1667.44 8.72"
$ns_ at 33.0 "$node_(23) setdest 1601.65 1677.42 9.98"
$ns_ at 34.0 "$node_(23) setdest 1601.65 1687.19 9.77"
$ns_ at 35.0 "$node_(23) setdest 1601.65 1696.87 9.68"
$ns_ at 36.0 "$node_(23) setdest 1601.65 1705.72 8.85"
$ns_ at 37.0 "$node_(23) setdest 1601.65 1715.51 9.79"
$ns_ at 38.0 "$node_(23) setdest 1601.65 1724.85 9.34"
$ns_ at 39.0 "$node_(23) setdest 1601.65 1733.79 8.93"
$ns_ at 40.0 "$node_(23) setdest 1601.65 1742.87 9.09"
$ns_ at 41.0 "$node_(23) setdest 1601.65 1752.1 9.23"
$ns_ at 42.0 "$node_(23) setdest 1601.65 1761.7 9.60"
$ns_ at 43.0 "$node_(23) setdest 1601.65 1771.18 9.48"
$ns_ at 44.0 "$node_(23) setdest 1601.65 1780.01 8.83"
$ns_ at 45.0 "$node_(23) setdest 1601.65 1787.2 7.19"
$ns_ at 46.0 "$node_(23) setdest 1601.65 1791.39 4.19"
```

Gambar 3.11 Skenario Mobilitas Pergerakan *node_(23)*

Pada Gambar 3.11 diketahui bahwa salah satu dari 50 *node* yaitu *node_(23)* mempunyai titik awal pergerakan yaitu dari koordinat (1604.95 , 1613.15) dan dimulai pada detik ke 24. Kemudian pada detik ke 25, *node_(23)* bergerak menuju koordinat (1604.95 , 1614.56) dengan kecepatan $1.41^{m}/_{s}$. Pada detik ke 26, *node_(23)* bergerak menuju koordinat (1604.95 , 1618.54) dengan kecepatan $3.48^{m}/_{s}$. Selanjutnya pada detik ke 27, *node_(23)* bergerak menuju koordinat (1604.95 , 1624.06) dengan kecepatan $5.52^{m}/_{s}$ dan seterusnya sesuai pada skrip yang terlampir.

Mobilitas tersebut nantinya akan diintegrasikan dengan skrip *tcl* yang telah dibuat. Beberapa parameter pada skrip *tcl* yang akan diuji dalam penelitian ini diantaranya :

```
set opt(stop) 910.0
set nodes   50
set mobility  1
set scenario  [lindex $argv 2]
set pausetime 0
set traffic cbr
set senders [lindex $argv 0]
set receivers   [lindex $argv 1]

set ns_ [new Simulator]
set topo [new Topography]
$topo load_flatgrid 2000 2000
```

Gambar 3.12 Parameter Pengujian dalam Skrip *tcl*

Pada Gambar 3.12 diketahui bahwa parameter pengujian dalam skrip *tcl* tersebut terdiri dari durasi simulasi yaitu 900 detik, jumlah *node* 50, trafik pengiriman data *constant bit rate* (CBR), dengan lingkungan topologi peta *grid* berukuran 2000m x 2000m.

Tabel 3.2 Skenario Mobilitas

No	Nama Skenario	Jml. Pengirim	Jml. Penerima
1.	cbr-50-01-02	1	2
2.	cbr-50-01-05	1	5
3.	cbr-50-01-10	1	10
4.	cbr-50-01-30	1	30
5.	cbr-50-01-50	1	50
6.	cbr-50-02-02	2	2
7.	cbr-50-02-05	2	5
8.	cbr-50-02-10	2	10
9	cbr-50-02-30	2	30
10.	cbr-50-02-50	2	50
11.	cbr-50-05-05	5	5
12.	cbr-50-05-10	5	10
13.	cbr-50-05-30	5	30
14.	cbr-50-05-50	5	50
15.	cbr-50-10-10	10	10
16.	cbr-50-10-30	10	30
17.	cbr-50-10-50	10	50
18.	cbr-50-30-30	30	30
19.	cbr-50-30-50	30	50

Pada Tabel 3.2 rancangan uji coba dilakukan pada beberapa macam pengujian dengan beberapa variasi jumlah *node* pengirim dan *node* penerima. Masing –

masing skrip skenario pengujian tersebut dalam skrip *tcl* terdiri dari beberapa parameter.

```
for {set i 0} {$i < 2} {incr i} {
  set udp_($i) [new Agent/UDP]
  $udp_($i) set dst_addr_ 0xE000000
  $ns_ attach-agent $node_($i) $udp_($i)

  set cbr_($i) [new Application/Traffic/CBR]
  $cbr_($i) set packetSize_ 256
  $cbr_($i) set interval_ 0.50
  $cbr_($i) set random_ 1
  # send enough packets to keep simulation nearly busy: 2 packets
  # a second, starting at 30, stopping at 899: 2*870 = 1740
  $cbr_($i) set maxpkts_ 1740
  $cbr_($i) attach-agent $udp_($i)
  $cbr_($i) set dst_ 0xE000000
  $ns_ at 30.0 "$cbr_($i) start"
}

for {set i 45} {$i < 50} {incr i} {
  $ns_ at 0.0100000000 "$node_($i) aodv-join-group 0xE000000"
}
```

Gambar 3.13 Skenario Pengujian dalam Skrip *tcl*

Parameter dalam skenario mobilitas diantaranya terdiri dari jumlah *node* pengirim, alamat *multicast node* penerima, pengiriman data menggunakan CBR, ukuran paket, maksimal jumlah paket yang dikirimkan dan waktu pengiriman data dimulai. Gambar 3.17 merupakan contoh skenario pengujian dengan jumlah *node* pengirim sebanyak 2 *node* (yaitu *node* 0 dan 1) dan jumlah *node* penerima sebanyak 5 *node* (yaitu *node* 45, 46, 47, 48, 49). Dengan ukuran paket per paket data sebesar 256b, maksimal jumlah paket yang dikirimkan sejumlah 1740 dan pengiriman data dimulai pada detik ke 30. Koneksi antara *node* sumber dengan *node* tujuan *multicast* menggunakan *constant bit rate* (CBR).

3.4 Evaluasi Kinerja

Untuk evaluasi kinerja dari metode yang diusulkan, data hasil penelitian berupa *trace file* diolah dengan dilakukan perbandingan terhadap beberapa parameter berikut:

- *End-to-End Delay*, ditentukan dari waktu yang dibutuhkan untuk mengirimkan paket data dari *node* sumber menuju *node* tujuan. Jika waktu *delay* yang dihasilkan dari pengujian semakin rendah, maka kinerja protokol semakin baik.

- *Packet Delivery Ratio* (PDR), ditentukan dari perbandingan jumlah paket data yang diterima oleh *node* tujuan dengan jumlah paket data yang dikirim oleh *node* sumber.

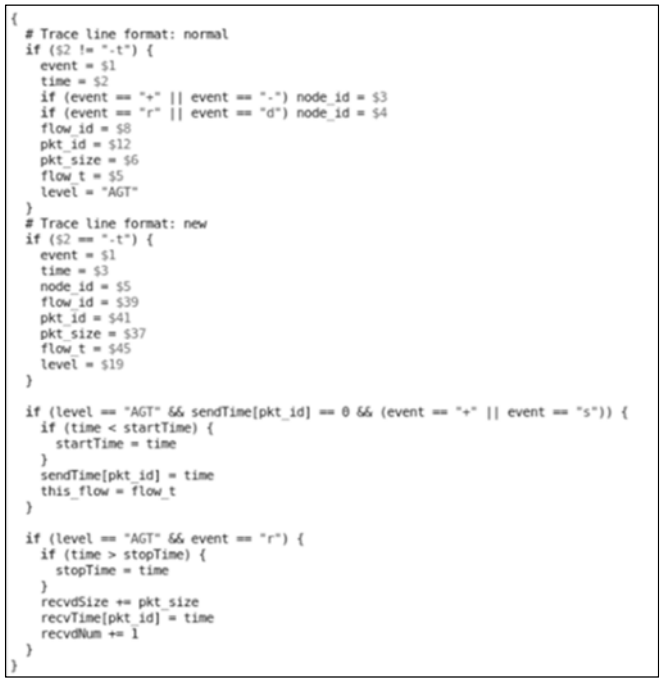
Perhitungan *packet delivery ratio* dan *end to end delay* dari setiap *trace file* yang dihasilkan oleh masing – masing skenario dihitung sesuai dengan rumus diatas dengan skrip sebagai berikut:

```
while (<MYFILE>)
{
  $current_time = findTime($_);
  if (m/cbr/ && m/^s/ && m/AGT/) {
    $id = findPacketId($_);
    $send[$id] = 1;
    $sendTime[$id] = $current_time;
  }
  elsif(m/cbr/ && m/^r/ && m/RTR/){
    $id = findPacketId($_);
    $node = findNodeId($_);
    if ($recv[$id][$node] == 0){
      $recv[$id][$node] = 1;
      $latency[$id][$node] = $current_time - $sendTime[$id];
    }
  }
}
close (MYFILE);

$totalSend = 0;
$totalRecv = 0;
$totalLatency = 0;
$i = 0;
while ($i < $totalPid){
  if ($send[$i] == 1){
    $totalSend++;
    $j = $totalNodes - $totalReceivers;
    if ($j == 0 ) {$j = 1;}
    while ($j < $totalNodes){
      if ($recv[$i][$j] == 1){
        $totalRecv++;
        $totalLatency += $latency[$i][$j];
      }
      $j++;
    }
  }
  $i++;
}
```

Gambar 3.14 Skrip Perhitungan PDR dan *End to End Delay*

- *Throughput*, ditentukan oleh kecepatan pengiriman data dari *node* sumber dalam ukuran tertentu yang dikirimkan menuju *node* tujuan dalam satuan waktu tertentu.

Perhitungan *throughput* dari setiap *trace file* dihitung sesuai dengan rumus diatas dengan skrip sebagai berikut:



```
{
  # Trace line format: normal
  if ($2 != "-t") {
    event = $1
    time = $2
    if (event == "+" || event == "-") node_id = $3
    if (event == "r" || event == "d") node_id = $4
    flow_id = $8
    pkt_id = $12
    pkt_size = $6
    flow_t = $5
    level = "AGT"
  }
  # Trace line format: new
  if ($2 == "-t") {
    event = $1
    time = $3
    node_id = $5
    flow_id = $39
    pkt_id = $41
    pkt_size = $37
    flow_t = $45
    level = $19
  }

  if (level == "AGT" && sendTime[pkt_id] == 0 && (event == "+" || event == "s")) {
    if (time < startTime) {
      startTime = time
    }
    sendTime[pkt_id] = time
    this_flow = flow_t
  }

  if (level == "AGT" && event == "r") {
    if (time > stopTime) {
      stopTime = time
    }
    recvdSize += pkt_size
    recvTime[pkt_id] = time
    recvdNum += 1
  }
}
```

Gambar 3.15 Skrip Perhitungan *Throughput*

Setiap hasil *trace file* yang dihasilkan akan dihitung PDR, *end to end delay* dan *throughput*. Skrip diatas menghasilkan informasi hasil perhitungan dari setiap *trace file* sehingga hasil perhitungan tersebut nantinya dapat dianalisa dan dilakukan perbandingan antara protokol *routing* MAODV dengan protokol *routing* yang diusulkan yaitu MAS-BRENK.

[Halaman ini sengaja dikosongkan]

BAB 4
HASIL DAN PEMBAHASAN

4.1 Tahapan Implementasi Metode

Langkah-langkah implementasi yang akan dibahas sesuai dengan tahapan-tahapan sebagai berikut :

1. Perancangan dan implementasi pengembangan protokol MAODV dengan parameter posisi, kecepatan, dan *delay* transmisi pada NS-2 menggunakan bahasa C. Pada tahap ini dihasilkan modul protokol MAODV yang telah dimodifikasi disebut protokol MAS-BRENK.
2. Perancangan dan implementasi skenario pengujian yang digunakan untuk menguji kinerja protokol yang telah dihasilkan. Skenario pengujian meliputi perancangan peta dan mobilitas node pada simulator SUMO.
3. Hasil implementasi dari kedua tahap diatas dijalankan dalam lingkungan simulator NS-2. Kedua tahap diatas menghasilkan data-data yang dapat diolah untuk kemudian dilakukan analisa lebih lanjut.

4.1.1. Pengembangan Protokol MAODV

Pengembangan protokol dimulai dari perancangan dan modifikasi skrip yang berkaitan dengan protokol MAODV pada NS-2. Modifikasi skrip meliputi bagian yang berkaitan dengan pengiriman pesan MACT, penerimaan pesan MACT dan perhitungan bobot berdasarkan parameter posisi, kecepatan dan *delay* transmisi. Skrip yang telah dimodifikasi kemudian di-*compile* untuk dijalankan pada simulator NS-2.

```
mact->mact_type = AODVTYPE_MACT;
mact->mact_flags = flags;
mact->mact_hop_count = hop_count;
mact->mact_grp_dst = dst;
mact->mact_src = index;
mact->mact_src_seqno = seqno;

//tambahan masbrenk
iNode = (MobileNode *) (Node::get_node_by_address(index));
mact->mact_x = iNode->X();
mact->mact_y = iNode->Y();
mact->mact_dx = iNode->dX();
mact->mact_dy = iNode->dY();
mact->mact_speed = iNode->speed();
mact->mact_delay = CURRENT_TIME;

FILE *fp;
fp = fopen("send_mact.txt", "a");
aodv_mt_entry *mt;

fprintf(fp, "NODE: %i, Time: %f, src: %i, flag: %d, hopCt: %i, grpDst: %i, seqno: %i, \n \
  x: %.2f, y: %.2f, dx: %.2f, dy: %.2f, speed: %.2f, sentTime: %f, enk: %f \n \n",
  index, CURRENT_TIME, mact->mact_src, mact->mact_flags, mact->mact_hop_count, mact->mact_grp_dst, mact->mact_src_seqno,
  mact->mact_x, mact->mact_y, mact->mact_dx, mact->mact_dy, mact->mact_speed, mact->mact_delay, mact->mact_enk);
fclose(fp);
```

Gambar 4.1 Skrip Pengiriman Pesan MACT

Setiap pengiriman pesan MACT, *node* akan mengirimkan posisi, kecepatan, dan waktu pengiriman pesan. Kemudian pengiriman setiap pesan dicatat dalam *file* teks yaitu *send_mact.txt*.

```
NODE: 38, Time: 63.500000, src: 38, flag: 2, hopCt: 0, grpDst: 234881024, seqno: 76,
 x: 1004.95, y: 882.48, dx: 0.00, dy: 1.00, speed: 9.32, sentTime: 63.500000
NODE: 18, Time: 67.000000, src: 18, flag: 2, hopCt: 0, grpDst: 234881024, seqno: 42,
 x: 721.45, y: 721.55, dx: -0.89, dy: -0.45, speed: 9.00, sentTime: 67.000000
NODE: 2, Time: 92.000000, src: 2, flag: 2, hopCt: 0, grpDst: 234881024, seqno: 56,
 x: 798.98, y: 848.83, dx: -0.06, dy: -1.00, speed: 9.63, sentTime: 92.000000
NODE: 18, Time: 92.011080, src: 18, flag: 2, hopCt: 0, grpDst: 234881024, seqno: 50,
 x: 605.53, y: 856.05, dx: -0.01, dy: 1.00, speed: 8.84, sentTime: 92.011080
NODE: 33, Time: 99.386986, src: 33, flag: 2, hopCt: 0, grpDst: 234881024, seqno: 40,
 x: 300.19, y: 1199.72, dx: -1.00, dy: 0.09, speed: 7.18, sentTime: 99.386986
NODE: 33, Time: 99.392638, src: 33, flag: 2, hopCt: 0, grpDst: 234881024, seqno: 40,
 x: 300.15, y: 1199.73, dx: -1.00, dy: 0.09, speed: 7.18, sentTime: 99.392638
NODE: 33, Time: 99.428413, src: 33, flag: 2, hopCt: 0, grpDst: 234881024, seqno: 40,
 x: 299.89, y: 1199.75, dx: -1.00, dy: 0.09, speed: 7.18, sentTime: 99.428413
NODE: 33, Time: 99.462423, src: 33, flag: 2, hopCt: 0, grpDst: 234881024, seqno: 40,
 x: 299.65, y: 1199.77, dx: -1.00, dy: 0.09, speed: 7.18, sentTime: 99.462423
NODE: 33, Time: 99.486330, src: 33, flag: 2, hopCt: 0, grpDst: 234881024, seqno: 40,
 x: 299.48, y: 1199.79, dx: -1.00, dy: 0.09, speed: 7.18, sentTime: 99.486330
NODE: 33, Time: 99.491318, src: 33, flag: 2, hopCt: 0, grpDst: 234881024, seqno: 40,
 x: 299.44, y: 1199.79, dx: -1.00, dy: 0.09, speed: 7.18, sentTime: 99.491318
NODE: 33, Time: 99.496067, src: 33, flag: 2, hopCt: 0, grpDst: 234881024, seqno: 40,
 x: 299.41, y: 1199.80, dx: -1.00, dy: 0.09, speed: 7.18, sentTime: 99.496067
NODE: 33, Time: 99.506575, src: 33, flag: 2, hopCt: 0, grpDst: 234881024, seqno: 40,
 x: 299.34, y: 1199.80, dx: -1.00, dy: 0.09, speed: 7.18, sentTime: 99.506575
NODE: 33, Time: 99.513544, src: 33, flag: 2, hopCt: 0, grpDst: 234881024, seqno: 40,
 x: 299.29, y: 1199.81, dx: -1.00, dy: 0.09, speed: 7.18, sentTime: 99.513544
NODE: 33, Time: 99.521963, src: 33, flag: 2, hopCt: 0, grpDst: 234881024, seqno: 40,
 x: 299.23, y: 1199.81, dx: -1.00, dy: 0.09, speed: 7.18, sentTime: 99.521963
NODE: 33, Time: 99.532483, src: 33, flag: 2, hopCt: 0, grpDst: 234881024, seqno: 40,
 x: 299.15, y: 1199.82, dx: -1.00, dy: 0.09, speed: 7.18, sentTime: 99.532483
NODE: 33, Time: 99.542243, src: 33, flag: 2, hopCt: 0, grpDst: 234881024, seqno: 40,
 x: 299.08, y: 1199.83, dx: -1.00, dy: 0.09, speed: 7.18, sentTime: 99.542243
NODE: 33, Time: 99.554053, src: 33, flag: 2, hopCt: 0, grpDst: 234881024, seqno: 40,
 x: 299.00, y: 1199.83, dx: -1.00, dy: 0.09, speed: 7.18, sentTime: 99.554053
NODE: 33, Time: 99.562797, src: 33, flag: 2, hopCt: 0, grpDst: 234881024, seqno: 40,
 x: 298.93, y: 1199.84, dx: -1.00, dy: 0.09, speed: 7.18, sentTime: 99.562797
NODE: 42, Time: 100.500000, src: 42, flag: 2, hopCt: 0, grpDst: 234881024, seqno: 28,
 x: 478.45, y: 404.95, dx: -1.00, dy: 0.00, speed: 9.11, sentTime: 100.500000
```

Gambar 4.2 Aktivitas Pengiriman Pesan MACT

Pada pesan MACT yang dikirimkan juga terdapat *flag* dimana masing – masing *flag* merupakan tanda dari isi pesan MACT baik *flag _j* (*join*), *flag _p* (*prune*) dan *_gl* (*groupleader*). Setiap pesan MACT yang diterima, akan dilakukan

perhitungan bobot. Ketika bobot melebihi batas ENK, maka *node* penerima akan mengirimkan pesan MACT dengan *flag _p* kepada *node* pengirim.

```
//tambahan masbrenk
iNode = (MobileNode *) (Node::get_node_by_address(index));
mact->mact_x = iNode->X();
mact->mact_y = iNode->Y();
mact->mact_dx = iNode->dX();
mact->mact_dy = iNode->dY();
mact->mact_speed = iNode->speed();
mact->mact_delay = CURRENT_TIME - delay;

euclidxy = sqrt( (x * mact->mact_x) + (y * mact->mact_y) );
euclidkec = sqrt(speed * mact->mact_speed);

bobotjarak = 2;
bobotkec = 1;
bobotdelay = 3;
wjarak = (bobotjarak / (bobotjarak + bobotkec + bobotperc + bobotdelay));
wkec = (bobotkec / (bobotjarak + bobotkec + bobotperc + bobotdelay));
wdelay = (bobotdelay / (bobotjarak + bobotkec + bobotperc + bobotdelay));
wpjarak = wjarak * euclidxy;
wpkec = wkec * euclidkec;
wpperc = wperc * euclidperc;
wpdelay = wdelay * mact->mact_delay;
enk = wpjarak * wpkec * wpdelay;

if (enk == 0) return;
else
  {
    enk == ((enk + enk1) / 2);
    enk1 == enk;
  }

FILE *fp;
fp = fopen("recvMACT_J.txt", "a");
fprintf(fp, "Node: %i, recvTime: %f, flag: %d, hopCt: %i, grpDst: %i, seqno: %i, \n \t \
x: %.2f, y: %.2f, dx: %.2f, dy: %.2f, speed: %.2f, sentTime: %f, enk: %f \n # %f, %f, %f, %f, %f, %f, %f # \n \
menerima paket dari src: %i ... \n  \
x: %.2f, y: %.2f, dx: %.2f, dy: %.2f, speed: %.2f, accel: %f, delay: %f, enk: %f \n \n",
  index, CURRENT_TIME, mact->mact_flags, mact->mact_hop_count, mact->mact_grp_dst, mact->mact_src_seqno,
  x, y, dx, dy, speed, delay, enk, euclidxy, euclidkec, euclidperc, wpjarak, wpkec, wpperc, wpdelay,
  mact->mact_src, iNode->X(), iNode->Y(), iNode->dX(), iNode->dY(), iNode->speed(), (CURRENT_TIME - delay), enk
  );
fclose(fp);
```

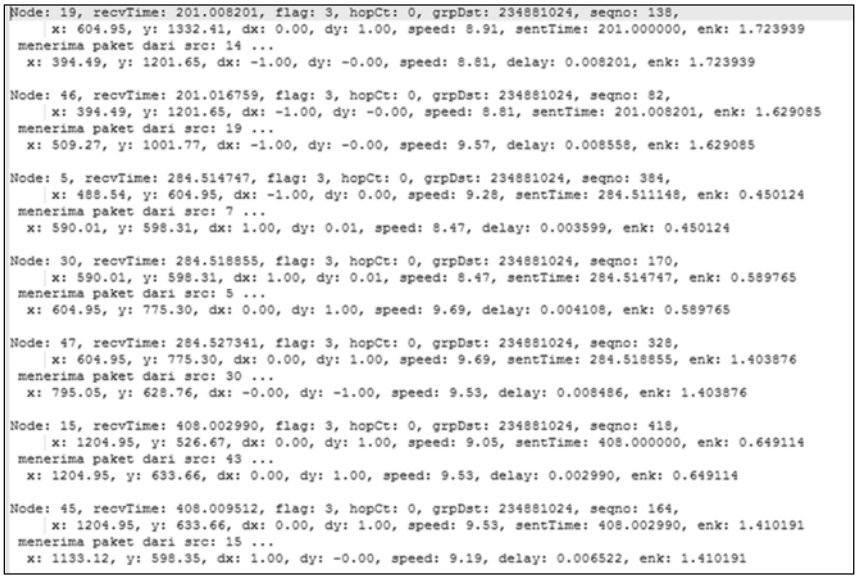
Gambar 4.3 Skrip Penerimaan Pesan MACT dengan *flag _j* (*join*)

Aktifitas penerimaan setiap pesan MACT juga dicatat dalam *file* teks. Penerimaan pesan MACT dengan *flag _j* disimpan dalam *file* teks bernama *recvMACT_J.txt*. Penerimaan pesan MACT tersebut menunjukkan sebagai berikut:

```
Node: 37, recvTime: 101.511409, flag: 1, hopCt: 2, grpDst: 234881024, seqno: 142,
     x: 711.76, y: 595.05, dx: 1.00, dy: -0.00, speed: 9.36, accel: 0.000000, sentTime: 101.500000, enk: 1.534552
 # 832.648915, 9.234154, 0.000000, 237.899690, 1.319165, 0.000000, 0.004890 #
 menerima paket dari src: 45 ...
  x: 604.95, y: 441.52, dx: 0.00, dy: 1.00, speed: 9.11, accel: 0.000000, delay: 0.011409, enk: 1.534552

Node: 49, recvTime: 101.517056, flag: 1, hopCt: 1, grpDst: 234881024, seqno: 26,
     x: 604.95, y: 441.52, dx: 0.00, dy: 1.00, speed: 9.11, accel: 0.000000, sentTime: 101.511409, enk: 0.606001
 # 675.361469, 9.084966, 0.000000, 192.960420, 1.297852, 0.000000, 0.002420 #
 menerima paket dari src: 37 ...
  x: 465.94, y: 394.64, dx: 1.00, dy: 0.03, speed: 9.06, accel: 0.000000, delay: 0.005646, enk: 0.606001

Node: 44, recvTime: 102.002694, flag: 1, hopCt: 3, grpDst: 234881024, seqno: 24,
     x: 1255.26, y: 1204.95, dx: -1.00, dy: 0.00, speed: 8.72, accel: 0.000000, sentTime: 102.000000, enk: 0.747951
 # 1744.005450, 9.101648, 0.000000, 498.287271, 1.300235, 0.000000, 0.001154 #
 menerima paket dari src: 19 ...
  x: 1195.05, y: 1279.27, dx: -0.00, dy: -1.00, speed: 9.50, accel: 0.000000, delay: 0.002694, enk: 0.747951

Node: 12, recvTime: 102.012509, flag: 1, hopCt: 2, grpDst: 234881024, seqno: 38,
     x: 1195.05, y: 1279.27, dx: -0.00, dy: -1.00, speed: 9.50, accel: 0.000000, sentTime: 102.002694, enk: 2.744291
 # 1729.593500, 9.241483, 0.000000, 494.169572, 1.320212, 0.000000, 0.004206 #
 menerima paket dari src: 44 ...
  x: 999.27, y: 1404.95, dx: -1.00, dy: 0.00, speed: 8.99, accel: 0.000000, delay: 0.009815, enk: 2.744291

Node: 48, recvTime: 102.023123, flag: 1, hopCt: 1, grpDst: 234881024, seqno: 62,
     x: 999.27, y: 1404.95, dx: -1.00, dy: 0.00, speed: 8.99, accel: 0.000000, sentTime: 102.012509, enk: 2.768704
 # 1605.955885, 9.285155, 0.000000, 458.844539, 1.326451, 0.000000, 0.004549 #
 menerima paket dari src: 12 ...
  x: 804.95, y: 1263.20, dx: -0.00, dy: 1.00, speed: 9.59, accel: 0.000000, delay: 0.010614, enk: 2.768704

Node: 38, recvTime: 107.506922, flag: 1, hopCt: 2, grpDst: 234881024, seqno: 48,
     x: 1195.05, y: 1228.95, dx: -0.00, dy: -1.00, speed: 9.08, accel: 0.000000, sentTime: 107.500000, enk: 1.798025
 # 1667.969838, 8.903280, 0.000000, 476.562811, 1.271897, 0.000000, 0.002966 #
 menerima paket dari src: 44 ...
  x: 1001.24, y: 1290.20, dx: -0.04, dy: 1.00, speed: 8.73, accel: 0.000000, delay: 0.006922, enk: 1.798025
```

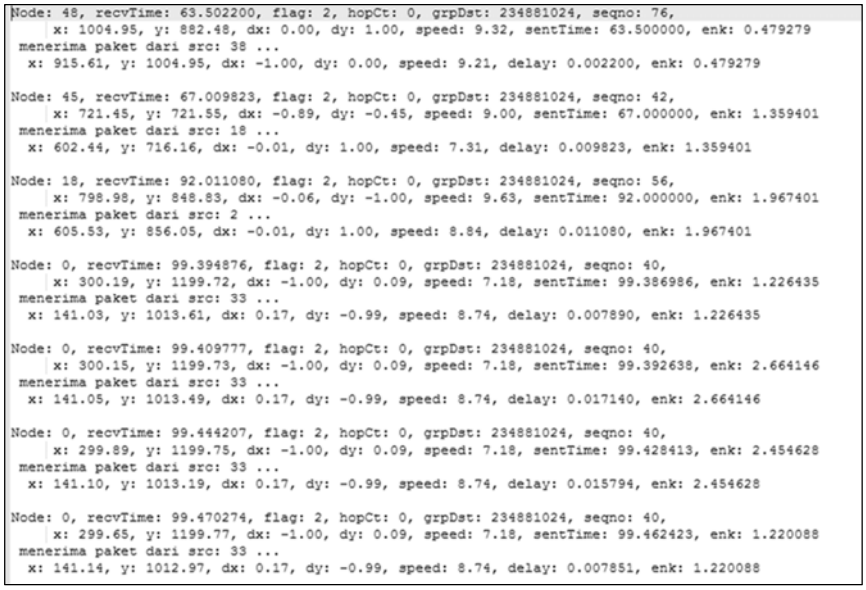
Gambar 4.4 Penerimaan Pesan MACT dengan *flag _j* (*join*)

Aktifitas penerimaan setiap pesan dengan *flag _gl* dicatat dalam *file* teks bernama *recvMACT_GL.txt*. Salah satu pengiriman pesan MACT dengan *flag _gl* yang dicatat menunjukkan sebagai berikut:



Gambar 4.5 Penerimaan Pesan MACT dengan *flag _gl* (*groupleader*)

Aktifitas penerimaan setiap pesan dengan *flag _p* dicatat dalam *file* teks bernama *recvMACT_P.txt*. Salah satu pengiriman pesan MACT dengan *flag _p* yang dicatat menunjukkan sebagai berikut:



Gambar 4.6 Penerimaan Pesan MACT dengan *flag _p* (*prune*)

4.1.2. Pembangunan Mobilitas Kendaraan

Mobilitas kendaraan dan bentuk peta pada penelitian ini menggunakan simulator SUMO seperti yang telah dijelaskan pada sub bab 3.2. Simulasi mobilitas kendaraan hasil dari simulator SUMO diberi nama *grid-50-10* (terlampir). Terdiri dari satu hingga tiga puluh *node* sumber dan dua hingga lima puluh *node* sebagai *node* tujuan *multicast*.

4.1.3. Pengujian Sistem

Pengujian sistem adalah menjalankan semua sistem yang telah dirancang untuk menguji kinerja protokol MAS-BRENK pada VANET. Parameter pengujian yang digunakan untuk pengujian protokol MAS-BRENK dijalankan dengan skrip *tcl* sesuai dengan sub bab 3.3. Skrip *tcl* yang dijalankan menggunakan NS-2 menghasilkan sebuah *trace file*. Hasil dibawah ini merupakan potongan hasil *trace file* bernama *trace-scen-50-10-01-05*.

```
M 103.00000 5 (43.15, 826.75, 0.00), (4.95, 1109.97), 9.85
M 103.00000 6 (1199.73, 166.89, 0.00), (935.91, 204.95), 8.79
M 103.00000 7 (651.40, 1389.47, 0.00), (995.05, 1324.17), 9.55
M 103.00000 8 (608.22, 1561.01, 0.00), (862.39, 1595.05), 9.74
s 103.341017009 _0_ AGT  --- 149 cbr 256 [0 0 0 0] ------- [0:0 234881024:-1 32 0] [149] 0 0
r 103.341017009 _0_ RTR  --- 149 cbr 256 [0 0 0 0] ------- [0:0 234881024:-1 32 0] [149] 0 0
s 103.341017009 _0_ RTR  --- 0 AODV 48 [0 0 0 0] ------- [0:255 -1:255 30 0] [0x2 1 17 [234881024 51] [0 88]] (REQUEST)
r 103.342197483 _39_ RTR  --- 0 AODV 48 [0 ffffffff 0 800] ------- [0:255 -1:255 30 0] [0x2 1 17 [234881024 51] [0 88]] (REQUEST)
r 103.342197612 _5_ RTR  --- 0 AODV 48 [0 ffffffff 0 800] ------- [0:255 -1:255 30 0] [0x2 1 17 [234881024 51] [0 88]] (REQUEST)
s 103.348438847 _39_ RTR  --- 0 AODV 48 [0 ffffffff 0 800] ------- [39:255 -1:255 29 0] [0x2 2 17 [234881024 51] [0 88]] (REQUEST)
r 103.349739321 _0_ RTR  --- 0 AODV 48 [0 ffffffff 27 800] ------- [39:255 -1:255 29 0] [0x2 2 17 [234881024 51] [0 88]] (REQUEST)
r 103.349739359 _5_ RTR  --- 0 AODV 48 [0 ffffffff 27 800] ------- [39:255 -1:255 29 0] [0x2 2 17 [234881024 51] [0 88]] (REQUEST)
s 103.350845136 _5_ RTR  --- 0 AODV 48 [0 ffffffff 0 800] ------- [5:255 -1:255 29 0] [0x2 2 17 [234881024 51] [0 88]] (REQUEST)
r 103.352305648 _39_ RTR  --- 0 AODV 48 [0 ffffffff 5 800] ------- [5:255 -1:255 29 0] [0x2 2 17 [234881024 51] [0 88]] (REQUEST)
r 103.352305738 _0_ RTR  --- 0 AODV 48 [0 ffffffff 5 800] ------- [5:255 -1:255 29 0] [0x2 2 17 [234881024 51] [0 88]] (REQUEST)
s 103.509680892 _47_ RTR  --- 0 AODV 44 [0 0 0 0] ------- [47:255 -1:255 1 0] [0x1 1 [47 36] 4.000000] (HELLO)
r 103.510709292 _23_ RTR  --- 0 AODV 44 [0 ffffffff 2f 800] ------- [47:255 -1:255 1 0] [0x1 1 [47 36] 4.000000] (HELLO)
s 103.611722875 _48_ RTR  --- 0 AODV 44 [0 0 0 0] ------- [48:255 -1:255 1 0] [0x1 1 [48 118] 4.000000] (HELLO)
r 103.612890913 _36_ RTR  --- 0 AODV 44 [0 ffffffff 30 800] ------- [48:255 -1:255 1 0] [0x1 1 [48 118] 4.000000] (HELLO)
r 103.612890970 _32_ RTR  --- 0 AODV 44 [0 ffffffff 30 800] ------- [48:255 -1:255 1 0] [0x1 1 [48 118] 4.000000] (HELLO)
r 103.612891223 _11_ RTR  --- 0 AODV 44 [0 ffffffff 30 800] ------- [48:255 -1:255 1 0] [0x1 1 [48 118] 4.000000] (HELLO)
r 103.612891490 _7_ RTR  --- 0 AODV 44 [0 ffffffff 30 800] ------- [48:255 -1:255 1 0] [0x1 1 [48 118] 4.000000] (HELLO)
r 103.612891536 _26_ RTR  --- 0 AODV 44 [0 ffffffff 30 800] ------- [48:255 -1:255 1 0] [0x1 1 [48 118] 4.000000] (HELLO)
r 103.612891536 _38_ RTR  --- 0 AODV 44 [0 ffffffff 30 800] ------- [48:255 -1:255 1 0] [0x1 1 [48 118] 4.000000] (HELLO)
r 103.612891609 _12_ RTR  --- 0 AODV 44 [0 ffffffff 30 800] ------- [48:255 -1:255 1 0] [0x1 1 [48 118] 4.000000] (HELLO)
r 103.612891624 _27_ RTR  --- 0 AODV 44 [0 ffffffff 30 800] ------- [48:255 -1:255 1 0] [0x1 1 [48 118] 4.000000] (HELLO)
r 103.612891641 _25_ RTR  --- 0 AODV 44 [0 ffffffff 30 800] ------- [48:255 -1:255 1 0] [0x1 1 [48 118] 4.000000] (HELLO)
r 103.612891650 _20_ RTR  --- 0 AODV 44 [0 ffffffff 30 800] ------- [48:255 -1:255 1 0] [0x1 1 [48 118] 4.000000] (HELLO)
r 103.612891663 _1_ RTR  --- 0 AODV 44 [0 ffffffff 30 800] ------- [48:255 -1:255 1 0] [0x1 1 [48 118] 4.000000] (HELLO)
s 103.720127781 _37_ RTR  --- 0 AODV 44 [0 0 0 0] ------- [37:255 -1:255 1 0] [0x1 1 [37 26] 4.000000] (HELLO)
r 103.721336116 _15_ RTR  --- 0 AODV 44 [0 ffffffff 25 800] ------- [37:255 -1:255 1 0] [0x1 1 [37 26] 4.000000] (HELLO)
r 103.721336235 _49_ RTR  --- 0 AODV 44 [0 ffffffff 25 800] ------- [37:255 -1:255 1 0] [0x1 1 [37 26] 4.000000] (HELLO)
r 103.721336342 _42_ RTR  --- 0 AODV 44 [0 ffffffff 25 800] ------- [37:255 -1:255 1 0] [0x1 1 [37 26] 4.000000] (HELLO)
r 103.721336400 _45_ RTR  --- 0 AODV 44 [0 ffffffff 25 800] ------- [37:255 -1:255 1 0] [0x1 1 [37 26] 4.000000] (HELLO)
s 103.755044225 _48_ RTR  --- 0 AODV 36 [0 0 0 0] ------- [48:255 -1:255 30 0] [0x6 0x0 0 48 [234881024 52]] (GROUP HELLO)
r 103.756288267 _36_ RTR  --- 0 AODV 36 [0 ffffffff 30 800] ------- [48:255 -1:255 30 0] [0x6 0x0 0 48 [234881024 52]] (GROUP HELLO)
r 103.756288328 _32_ RTR  --- 0 AODV 36 [0 ffffffff 30 800] ------- [48:255 -1:255 30 0] [0x6 0x0 0 48 [234881024 52]] (GROUP HELLO)
r 103.756288580 _11_ RTR  --- 0 AODV 36 [0 ffffffff 30 800] ------- [48:255 -1:255 30 0] [0x6 0x0 0 48 [234881024 52]] (GROUP HELLO)
```

Gambar 4.7 Potongan Skrip *Trace File*

4.2 Hasil dan Analisa

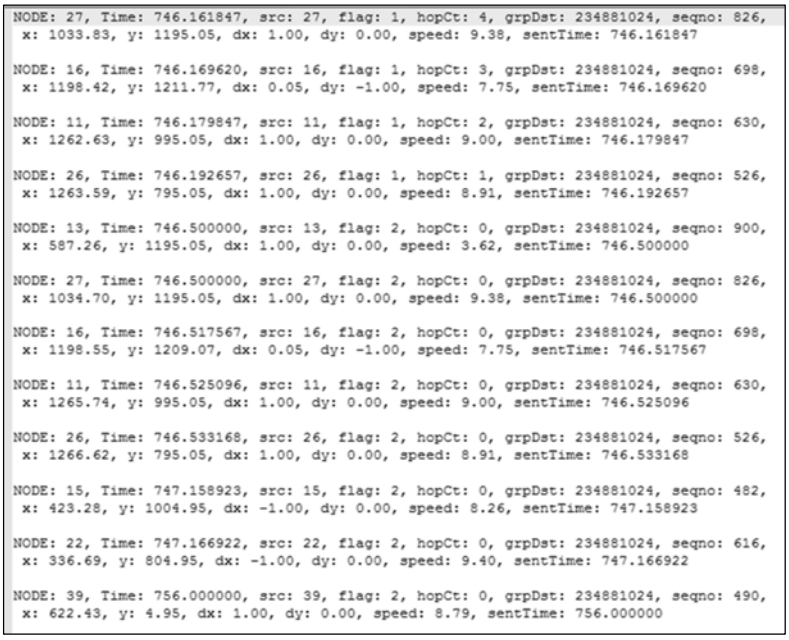
Hasil pengujian sistem dalam penelitian ini berupa *trace file* yang dihasilkan dari simulasi NS-2 dengan menjalankan skrip *tcl*. Dengan menggunakan skrip *awk*, nilai dari hasil *trace file* tersebut dapat dipergunakan sebagai bahan analisa. Data

hasil analisa disajikan dalam bentuk grafik. Analisa dilakukan terhadap parameter pengujian yang dipergunakan dalam penelitian ini yaitu *Packet Delivery Ratio* (PDR), *throughput* dan *average delay*.

Pada simulasi pertama dengan satu *node* pengirim dan dua *node* penerima, analisa file *send_mact.txt* dari protokol MAODV mencatat pesan MACT sebanyak 259 pesan yang dikirimkan. Sedangkan protokol MAS-BRENK mengirimkan pesan MACT sebanyak 255 pesan atau lebih sedikit 1.54% dibandingkan protokol MAODV. Namun pada simulasi kedua dengan satu *node* pengirim dan dua *node* penerima, protokol MAODV mencatat pengiriman pesan MACT sebanyak 804 pesan. Sedangkan protokol MAS-BRENK mengirimkan pesan MACT sebanyak 824 pesan atau lebih banyak 2.49% dari protokol MAODV.

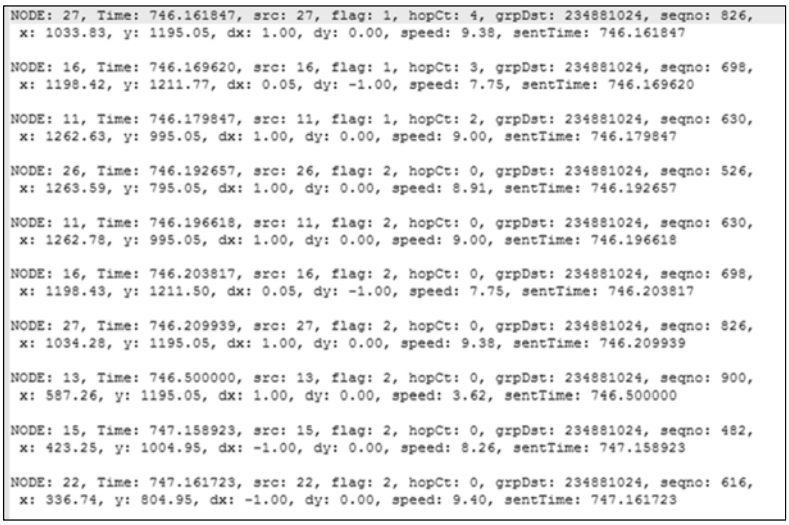
Tabel 4.1 Jumlah Pengiriman Pesan MACT

No	Jml. Node Pengirim	Jml. Node Penerima	Jml. Paket MACT		Perbedaan Jml. Pesan
			MAODV	MAS-BRENK	
1.	1	2	259	255	-4
		5	804	824	20
		10	1011	1033	22
		30	1035	1049	14
		50	617	632	15
2.	2	2	686	390	-296
		5	879	748	-131
		10	1127	1148	21
		30	685	1080	395
		50	652	660	8
3.	5	5	910	979	69
		10	1456	1240	-216
		30	1105	1126	21
		50	648	679	31
4.	10	10	1428	1261	-167
		30	1139	1248	109
		50	645	668	23
5.	30	30	1173	1249	76
		50	650	727	77

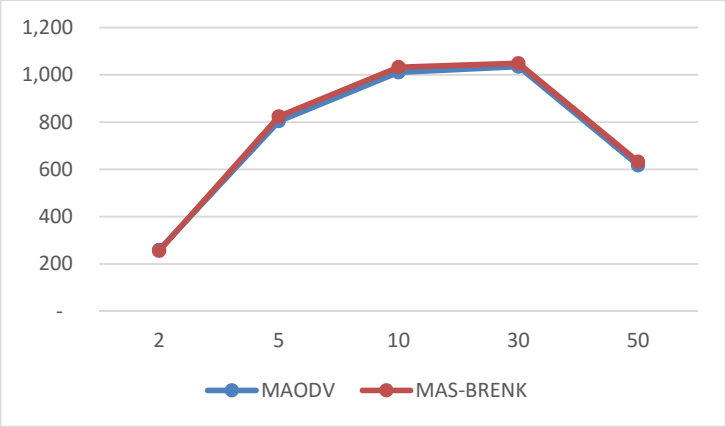


Gambar 4.8 Catatan Pengiriman Pesan MACT pada MAODV

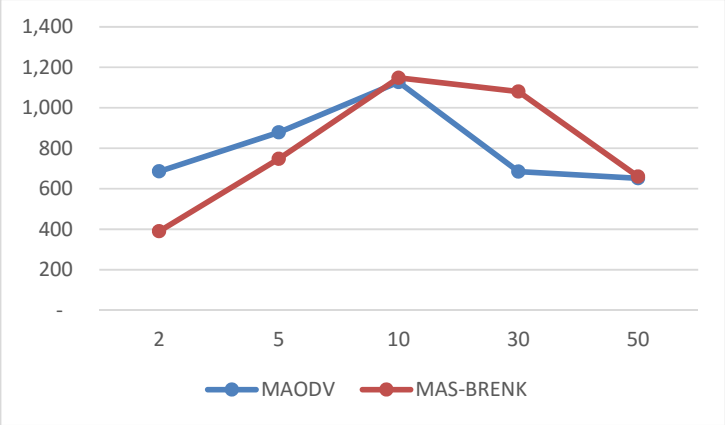
Pada gambar 4.8 merupakan pesan MACT yang dikirim oleh protokol MAODV selama simulasi berlangsung. Sebagai contoh perbedaan yang terjadi pada detik ke 746.192657, *node* 26 mengirimkan pesan MACT dengan *flag _j* pada protokol MAODV. Sebaliknya *node* 26 mengirimkan pesan dengan *flag _p* pada protokol MAS-BRENK.



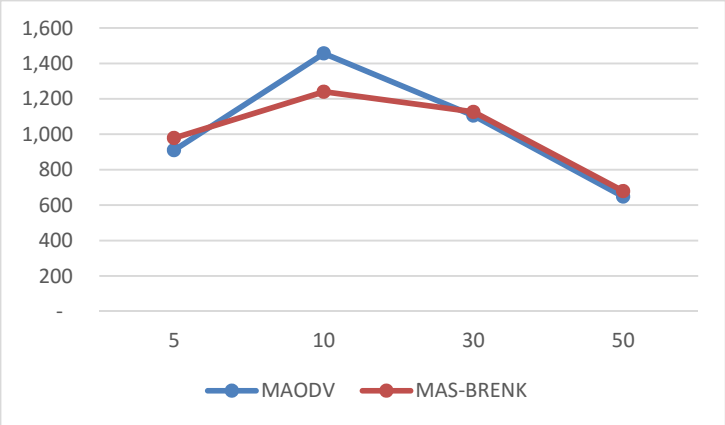
Gambar 4.9 Catatan Pengiriman Pesan MACT pada MAS-BRENK



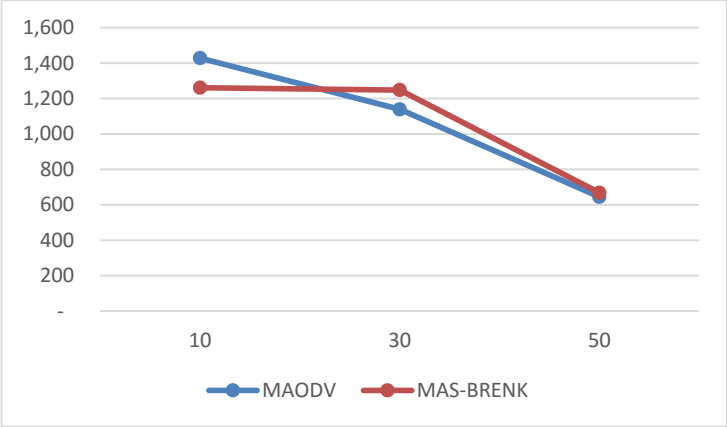
Gambar 4.10 Jumlah Pengiriman Pesan MACT (1 *node* pengirim)



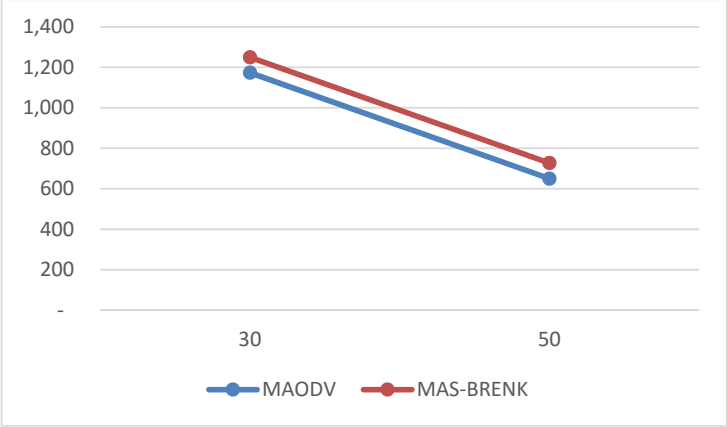
Gambar 4.11 Jumlah Pengiriman Pesan MACT (2 *node* pengirim)



Gambar 4.12 Jumlah Pengiriman Pesan MACT (5 *node* pengirim)



Gambar 4.13 Jumlah Pengiriman Pesan MACT (10 *node* pengirim)



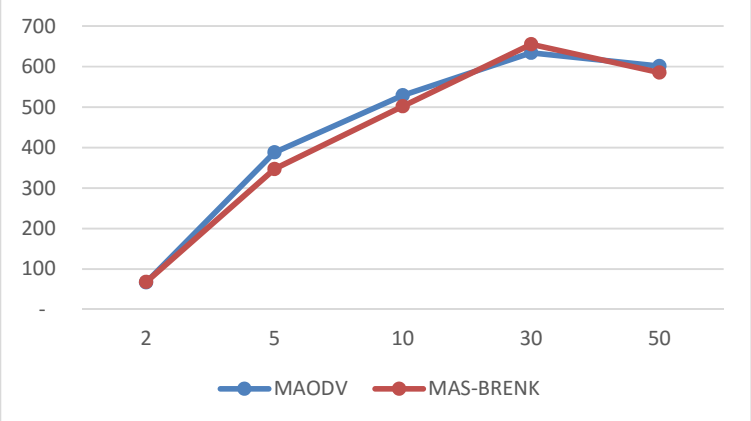
Gambar 4.14 Jumlah Pengiriman Pesan MACT (30 *node* pengirim)

Jumlah pengiriman dan penerimaan pesan MACT bergantung pada banyak faktor terutama faktor yang mempengaruhi nilai bobot ENK, diantaranya posisi *node*, kecepatan dan *delay* transmisi. Karena setiap pengiriman pesan MACT, *node* penerima akan menghitung bobot dan akan mempengaruhi pengiriman pesan MACT selanjutnya. Jika bobot yang dihitung lebih besar atau sama dengan nilai yang telah ditentukan, maka *node* penerima akan melakukan pengiriman pesan MACT dengan *flag _p*. Dan sebaliknya bobot yang dihitung jika lebih kecil, maka *node* penerima akan mencatat *node* pengirim pada tabel *multicast*.

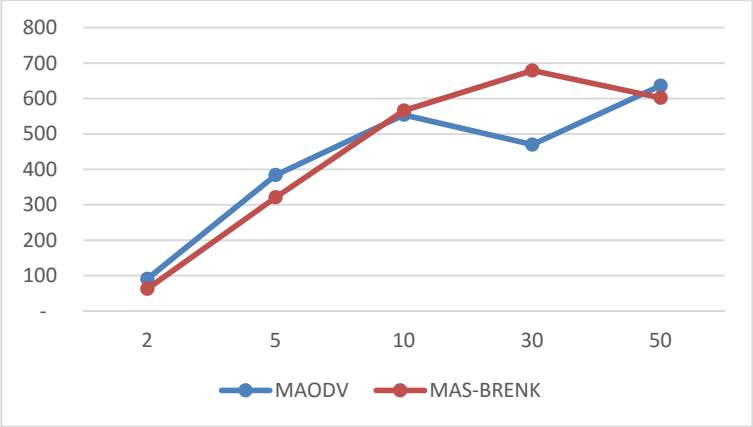
Tabel 4.2 Jumlah Penerimaan Pesan MACT_J

No	Jml. Node Pengirim	Jml. Node Penerima	Jml. Paket MACT_J		Perbedaan Jml. Pesan
			MAODV	MAS-BRENK	
1.	1	2	67	68	1
		5	388	347	-41
		10	529	502	-27
		30	634	655	21
		50	601	585	-16
2.	2	2	91	63	-28
		5	384	321	-63
		10	554	566	12
		30	470	679	209
		50	636	602	-34
3.	5	5	337	321	-16
		10	577	554	-23
		30	708	673	-35
		50	623	612	-11
4.	10	10	555	506	-49
		30	713	676	-37
		50	619	610	-9
5.	30	30	691	692	1
		50	607	623	16

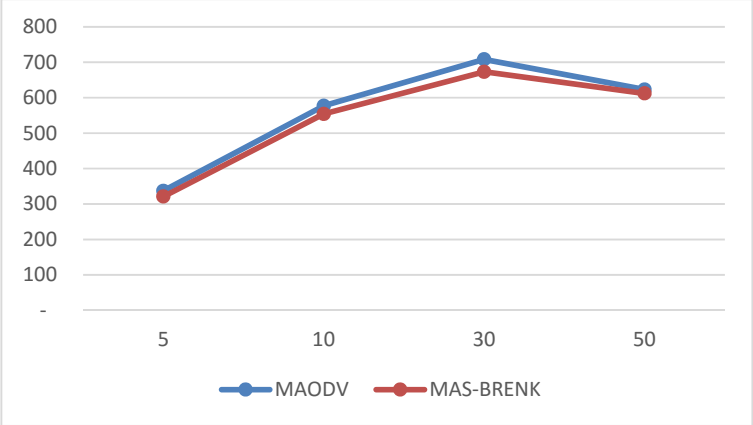
Analisa penerimaan pesan MACT dengan *flag _j* pada file *recvMACT_J.txt* dari protokol MAODV diatas, menerima pesan MACT sebanyak 67 pesan sedangkan protokol MAS-BRENK yang menerima pesan MACT sebanyak 68 pesan, satu paket lebih banyak atau 1.5% lebih banyak jumlah pesan yang diterima protokol MAS-BRENK daripada MAODV.



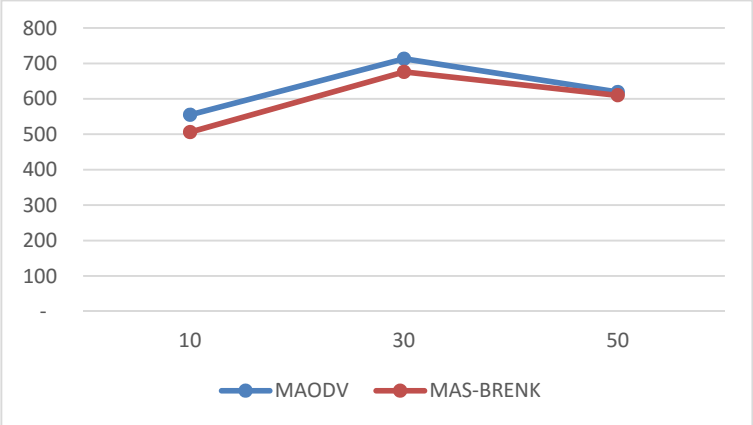
Gambar 4.15 Jumlah Penerimaan Pesan MACT_J (1 *node* pengirim)



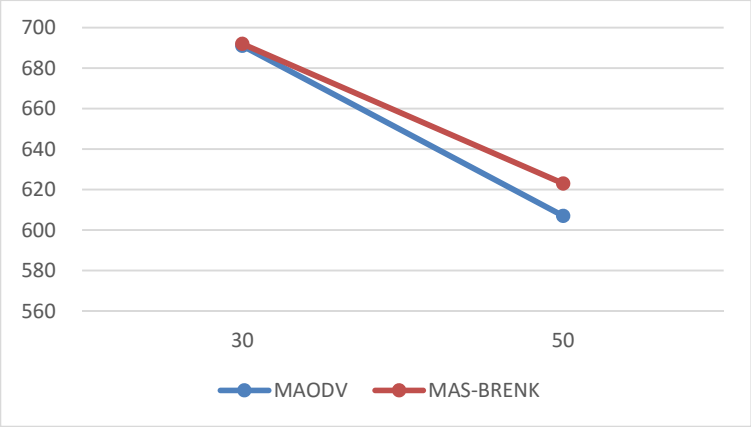
Gambar 4.16 Jumlah Penerimaan Pesan MACT_J (2 *node* pengirim)



Gambar 4.17 Jumlah Penerimaan Pesan MACT_J (5 *node* pengirim)



Gambar 4.18 Jumlah Penerimaan Pesan MACT_J (10 *node* pengirim)



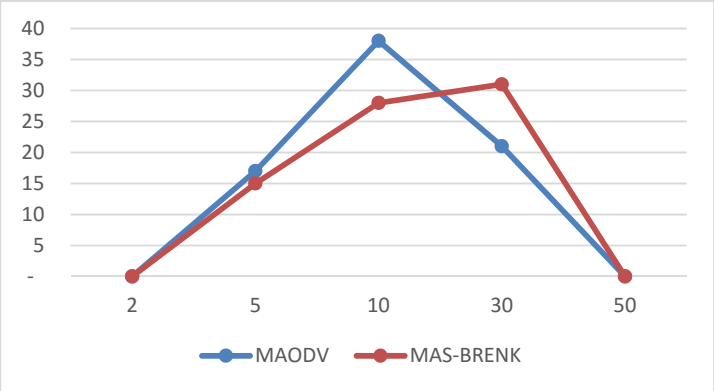
Gambar 4.19 Jumlah Penerimaan Pesan MACT_J (30 *node* pengirim)

Terdapat perbedaan yang bervariasi juga dari penerimaan pesan MACT dengan *flag _gl* pada file *recvMACT_GL.txt* dari protokol MAODV dan protokol MAS-BRENK.

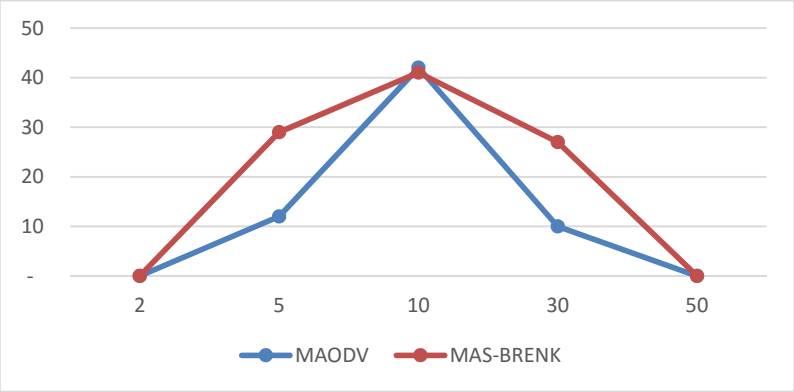
Tabel 4.3 Jumlah Penerimaan Pesan MACT_GL

No	Jml. Node Pengirim	Jml. Node Penerima	Jml. Paket MACT		Perbedaan Jml. Pesan
			MAODV	MAS-BRENK	
1.	1	2	-	-	0
		5	17	15	-2
		10	38	28	-10
		30	21	31	10
		50	-	-	0
2.	2	2	-	-	0
		5	12	29	17
		10	42	41	-1
		30	10	27	17
		50	-	-	0
3.	5	5	23	15	-8
		10	68	39	-29
		30	40	32	-8
		50	-	-	0
4.	10	10	41	33	-8
		30	30	39	9
		50	-	-	0
5.	30	30	26	33	7
		50	-	-	0

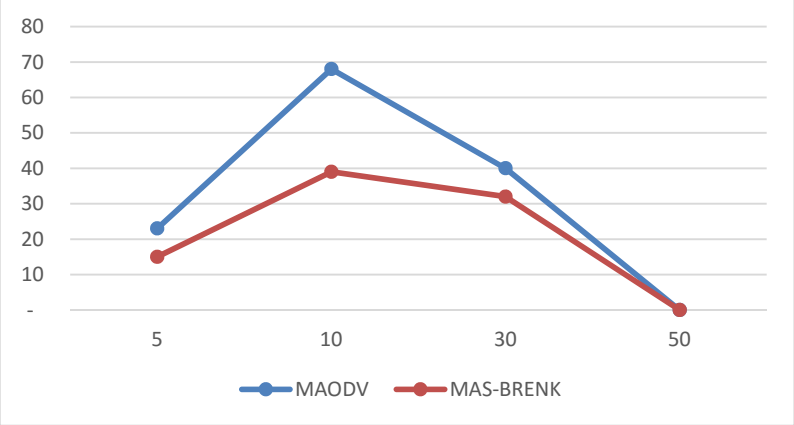
Beberapa skenario dengan jumlah penerima yaitu dua *node* dan lima puluh *node* tidak satu pun mengirimkan pesan MACT dengan *flag _gl*.



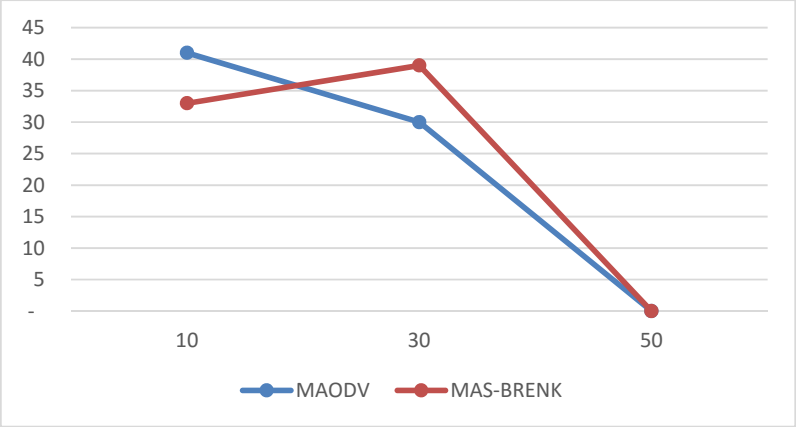
Gambar 4.20 Jumlah Penerimaan Pesan MACT_GL (1 *node* pengirim)



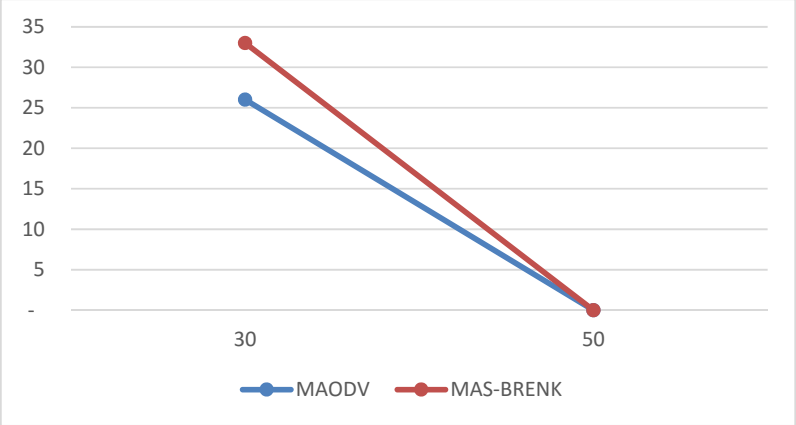
Gambar 4.21 Rata-rata Jumlah Penerimaan Pesan MACT_GL (2 *node* pengirim)



Gambar 4.22 Rata-rata Jumlah Penerimaan Pesan MACT_GL (5 *node* pengirim)



Gambar 4.23 Rata-rata Jumlah Penerimaan Pesan MACT_GL (10 *node* pengirim)



Gambar 4.24 Rata-rata Jumlah Penerimaan Pesan MACT_GL (30 *node* pengirim)

Untuk penerimaan pesan MACT dengan *flag _p* pada file *recvMACT_P.txt* dari protokol MAODV, pesan MACT yang diterima sebanyak 187 pesan sedangkan protokol MAS-BRENK menerima pesan MACT dengan *flag _p* sebanyak 181 pesan atau 6% lebih sedikit dibandingkan protokol MAODV.

Tabel 4.4 Jumlah Penerimaan Pesan MACT_P

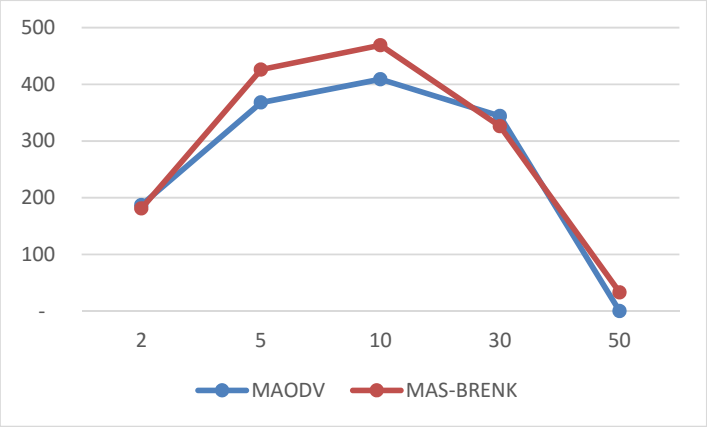
No	Jml. Node Pengirim	Jml. Node Penerima	Jml. Paket MACT		Perbedaan Jml. Pesan
			MAODV	MAS-BRENK	
1.	1	2	187	181	-6
		5	368	426	58
		10	409	469	60

		30	344	326	-18
		50	-	33	33
2.	2	2	584	322	-262
		5	454	375	-79
		10	494	492	-2
		30	174	337	163
		50	-	36	36
3.	5	5	524	611	87
		10	771	614	-157
		30	317	389	72
		50	2	40	38
4.	10	10	794	684	-110
		30	358	488	130
		50	-	30	30
5.	30	30	407	476	69
		50	1	67	66

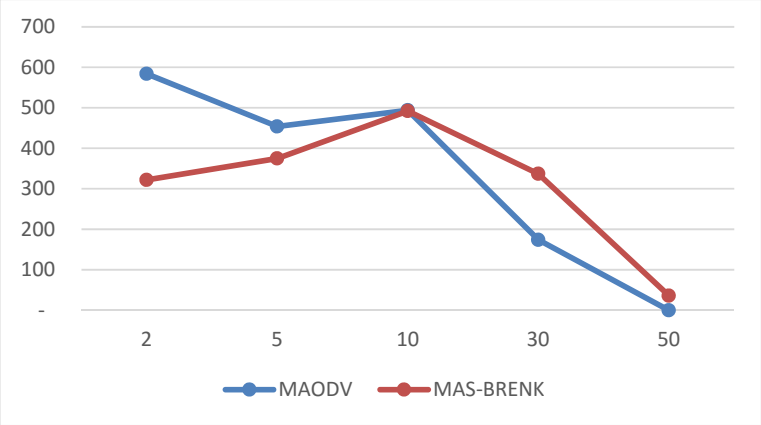
Gambar 4.9 menunjukkan bahwa dengan perhitungan ENK, *node* yang mempunyai bobot lebih dari batas yang ditentukan harus melepaskan diri dari anggota *multicast*. Hal ini ditunjukkan pada gambar 4.11 dibawah ini bahwa pada detik 746.196618, node 11 menerima pesan MACT dengan *flag _p* dari *node* 26 sebagai tanda bahwa *node* 26 melepaskan diri dari keanggotaan *multicast*.

```
Node: 49, recvTime: 746.023881, flag: 2, hopCt: 0, grpDst: 234881024, seqno: 526,
     x: 1262.05, y: 795.05, dx: 1.00, dy: 0.00, speed: 8.91, accel: 0.000000, sentTime: 746.020721, enk: 0.770863
 menerima paket dari src: 26 ...
  x: 1363.01, y: 601.65, dx: -1.00, dy: 0.00, speed: 9.93, accel: 0.000000, delay: 0.003160, enk: 0.770863

Node: 11, recvTime: 746.196618, flag: 2, hopCt: 0, grpDst: 234881024, seqno: 526,
     x: 1263.59, y: 795.05, dx: 1.00, dy: 0.00, speed: 8.91, accel: 0.000000, sentTime: 746.192657, enk: 0.958625
 menerima paket dari src: 26 ...
  x: 1262.78, y: 995.05, dx: 1.00, dy: 0.00, speed: 9.00, accel: 0.000000, delay: 0.003961, enk: 0.958625

Node: 16, recvTime: 746.203817, flag: 2, hopCt: 0, grpDst: 234881024, seqno: 630,
     x: 1262.78, y: 995.05, dx: 1.00, dy: 0.00, speed: 9.00, accel: 0.000000, sentTime: 746.196618, enk: 1.734236
 menerima paket dari src: 11 ...
  x: 1198.43, y: 1211.50, dx: 0.05, dy: -1.00, speed: 7.75, accel: 0.000000, delay: 0.007199, enk: 1.734236

Node: 27, recvTime: 746.209939, flag: 2, hopCt: 0, grpDst: 234881024, seqno: 698,
     x: 1198.43, y: 1211.50, dx: 0.05, dy: -1.00, speed: 7.75, accel: 0.000000, sentTime: 746.203817, enk: 1.496704
 menerima paket dari src: 16 ...
  x: 1034.28, y: 1195.05, dx: 1.00, dy: 0.00, speed: 9.38, accel: 0.000000, delay: 0.006122, enk: 1.496704

Node: 7, recvTime: 746.513771, flag: 2, hopCt: 0, grpDst: 234881024, seqno: 900,
     x: 587.26, y: 1195.05, dx: 1.00, dy: 0.00, speed: 3.62, accel: 0.000000, sentTime: 746.500000, enk: 1.942515
 menerima paket dari src: 13 ...
  x: 781.50, y: 1204.95, dx: -1.00, dy: 0.00, speed: 9.46, accel: 0.000000, delay: 0.013771, enk: 1.942515

Node: 22, recvTime: 747.161723, flag: 2, hopCt: 0, grpDst: 234881024, seqno: 482,
     x: 423.25, y: 1004.95, dx: -1.00, dy: 0.00, speed: 8.26, accel: 0.000000, sentTime: 747.158923, enk: 0.420887
 menerima paket dari src: 15 ...
  x: 336.74, y: 804.95, dx: -1.00, dy: 0.00, speed: 9.40, accel: 0.000000, delay: 0.002799, enk: 0.420887

Node: 48, recvTime: 747.168665, flag: 2, hopCt: 0, grpDst: 234881024, seqno: 616,
     x: 336.74, y: 804.95, dx: -1.00, dy: 0.00, speed: 9.40, accel: 0.000000, sentTime: 747.161723, enk: 0.856678
 menerima paket dari src: 22 ...
  x: 269.73, y: 595.07, dx: 1.00, dy: -0.00, speed: 9.29, accel: 0.000000, delay: 0.006943, enk: 0.856678
```

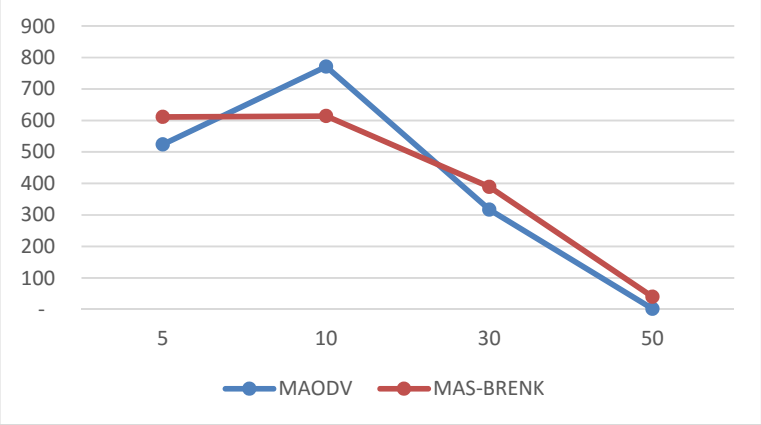
Gambar 4.25 *Node* Menerima Pesan MACT pada MAS-BRENK



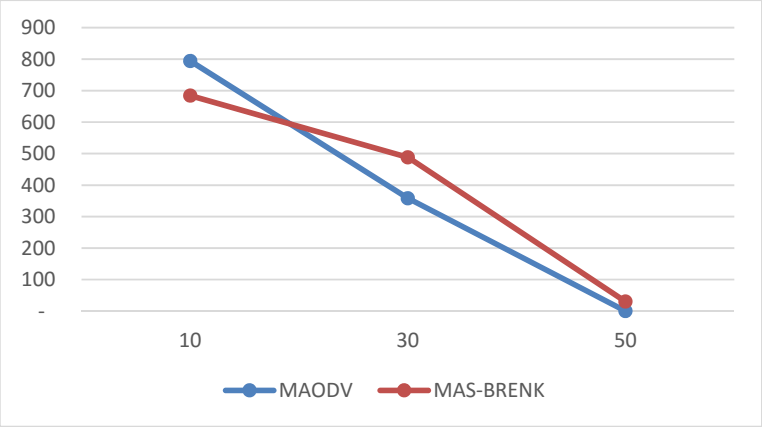
Gambar 4.26 Jumlah Penerimaan Pesan MACT_P (1 *node* pengirim)



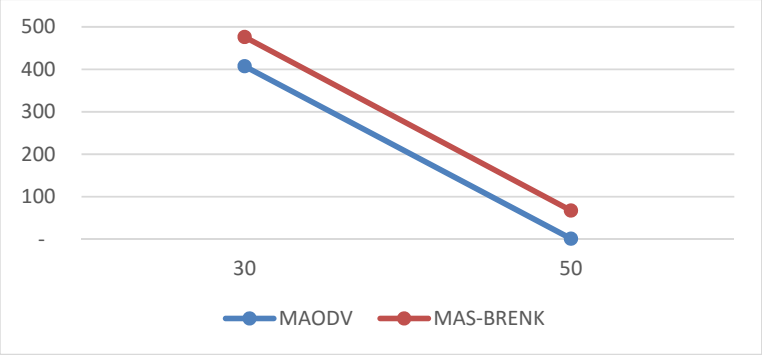
Gambar 4.27 Jumlah Penerimaan Pesan MACT_P (2 *node* pengirim)



Gambar 4.28 Jumlah Penerimaan Pesan MACT_P (5 *node* pengirim)



Gambar 4.29 Jumlah Penerimaan Pesan MACT_P (10 *node* pengirim)



Gambar 4.30 Jumlah Penerimaan Pesan MACT_P (30 *node* pengirim)

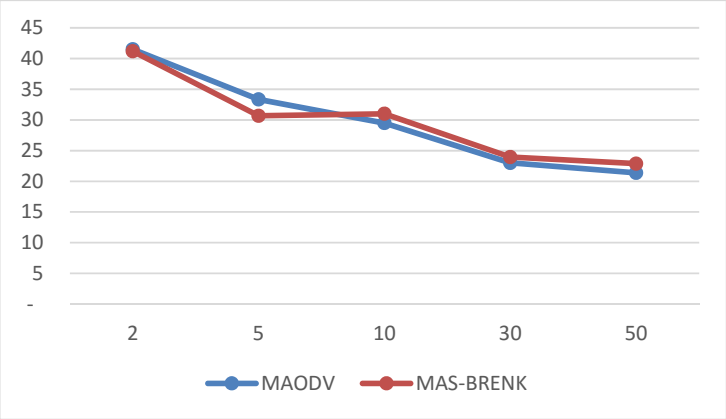
4.2.1. *Packet Delivery Ratio* (PDR)

Packet Delivery Ratio (PDR) merupakan perbandingan antara paket data yang diterima dengan paket data yang dikirimkan. Peningkatan dan penurunan PDR pada protokol MAS-BRENK bervariasi. Berikut data hasil perbandingan PDR yang dihasilkan oleh protokol MAODV dan MAS-BRENK.

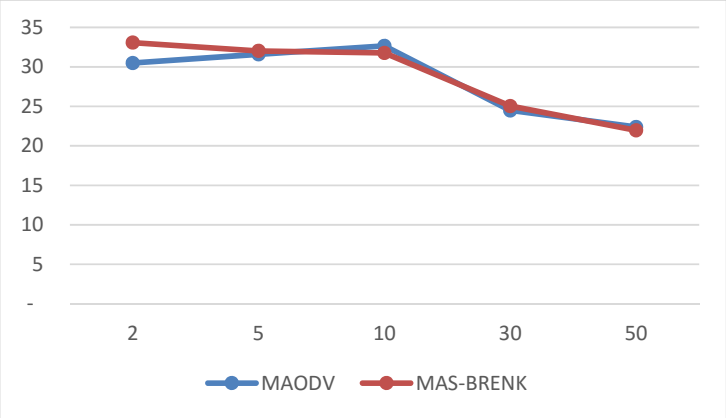
Tabel 4.5 Perbandingan PDR

No	Sender	Receiver	*Packet Delivery Ratio* (%)		Perbedaan (%)
			MAODV	MAS-BRENK	
1.	1	2	41.494	41.178	-0.762
		5	33.333	30.667	-8.000
		10	29.489	31.000	5.127

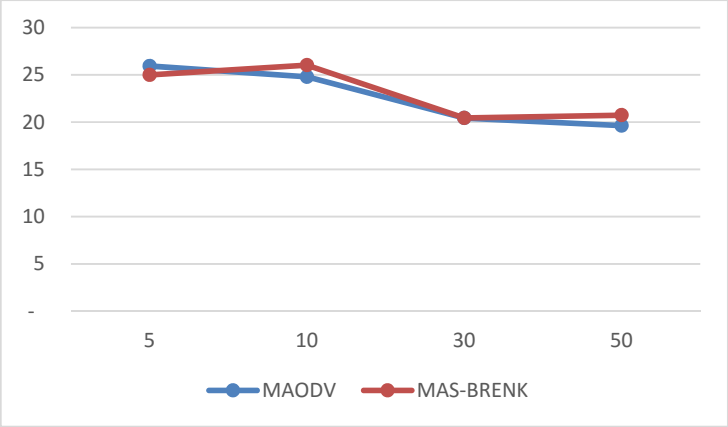
		30	22.992	23.960	4.208
		50	21.370	22.875	7.042
2.	2	2	30.489	33.075	8.483
		5	31.601	32.023	1.335
		10	32.664	31.764	-2.754
		30	24.488	25.033	2.226
		50	22.380	21.966	-1.852
3.	5	5	25.938	25.002	-3.607
		10	24.793	26.031	4.993
		30	20.439	20.441	0.009
		50	19.629	20.738	5.650
4.	10	10	25.890	24.974	-3.536
		30	19.970	20.281	1.561
		50	19.535	18.911	-3.193
5.	30	30	18.825	18.091	-3.899
		50	16.947	17.375	2.523



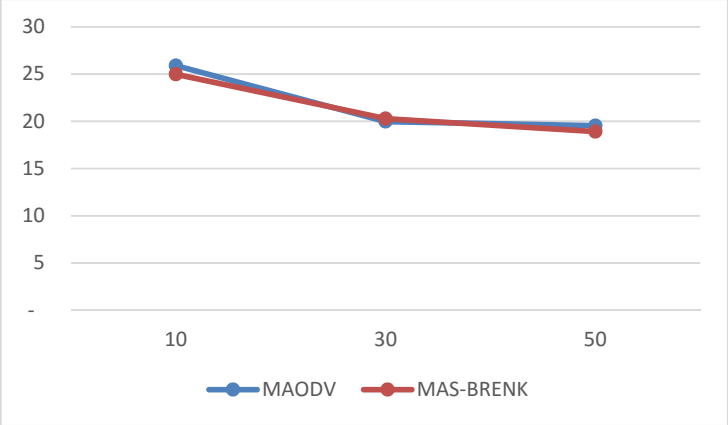
Gambar 4.31 Grafik Perbandingan PDR (1 *node* pengirim)



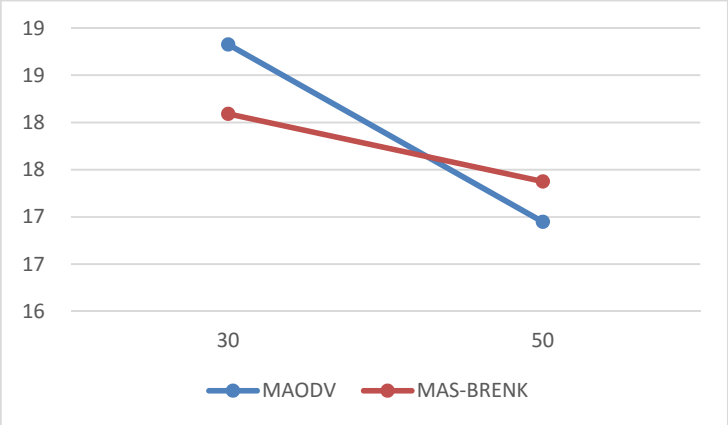
Gambar 4.32 Grafik Perbandingan PDR (2 *node* pengirim)



Gambar 4.33 Grafik Perbandingan PDR (5 *node* pengirim)



Gambar 4.34 Grafik Perbandingan PDR (10 *node* pengirim)



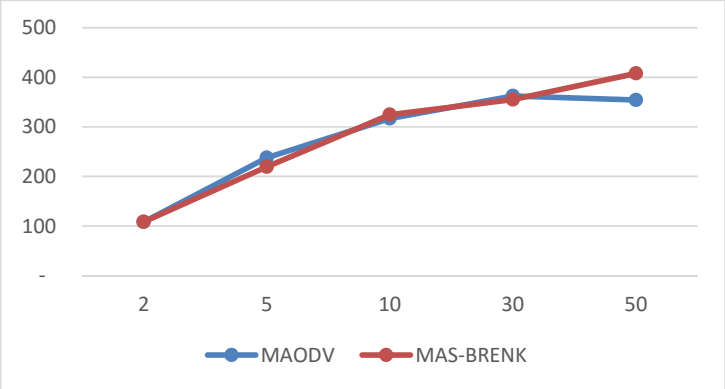
Gambar 4.35 Grafik Perbandingan PDR (30 *node* pengirim)

4.2.2. *Throughput*

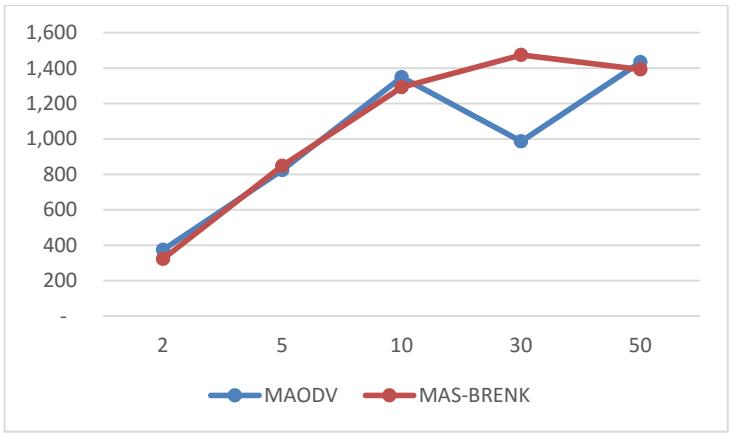
Throughput adalah banyaknya paket data yang dikirimkan berdasarkan satuan waktu tertentu. Beberapa skenario juga menunjukkan adanya peningkatan *throughput*. Berikut tabel perbandingan *throughput* yang dihasilkan oleh protokol MAODV dan MAS-BRENK.

Tabel 4.6 Tabel Perbandingan *Throughput*

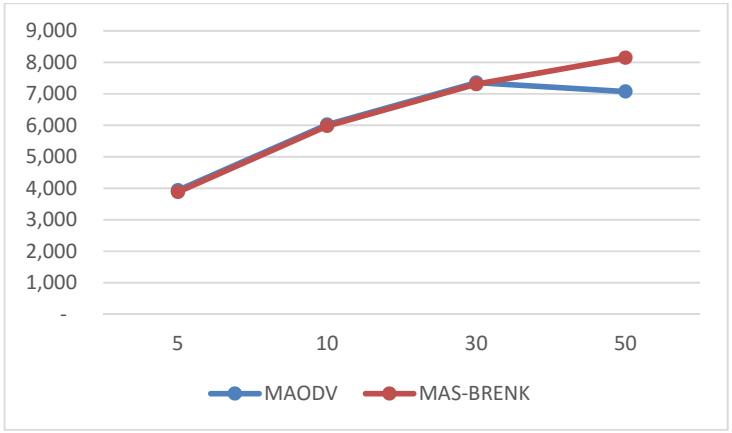
No	Sender	Receiver	*Throughput* (kbps)		Perbedaan (kbps)
			MAODV	MAS-BRENK	
1.	1	2	108.287	108.613	0.326
		5	237.795	219.560	-18.235
		10	316.952	324.749	7.797
		30	362.288	355.249	-7.039
		50	354.322	407.886	53.564
2.	2	2	374.081	322.997	-51.084
		5	824.338	848.079	23.741
		10	1,348.560	1,291.990	-56.570
		30	986.224	1,474.030	487.806
		50	1,433.230	1,392.290	-40.940
3.	5	5	3,935.610	3,881.710	-53.900
		10	6,019.550	5,977.770	-41.780
		30	7,354.020	7,303.880	-50.140
		50	7,072.530	8,146.010	1,073.480
4.	10	10	26,734.000	25,867.700	-866.300
		30	30,053.500	31,696.700	1,643.200
		50	29,973.700	29,301.300	-672.400
5.	30	30	246,982.000	234,258.000	-12,724.000
		50	248,632.000	242,574.000	-6,058.000



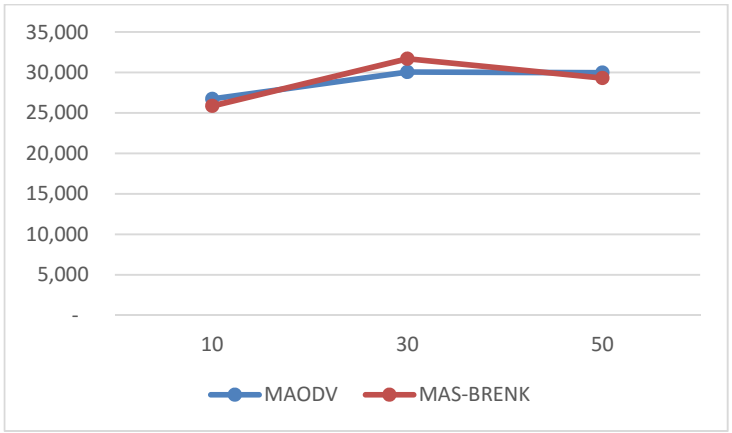
Gambar 4.36 Grafik Perbandingan *Throughput* (1 *node* pengirim)



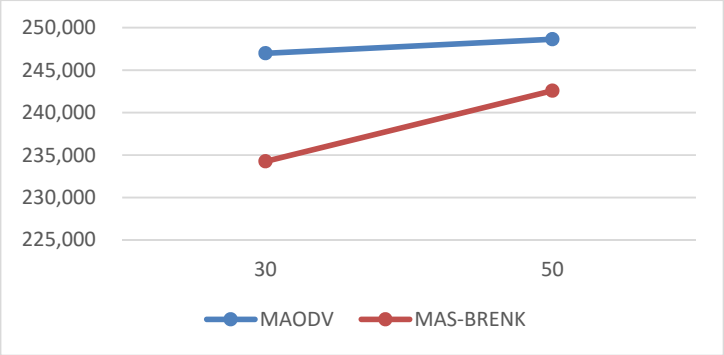
Gambar 4.37 Grafik Perbandingan *Throughput* (2 *node* pengirim)



Gambar 4.38 Grafik Perbandingan *Throughput* (5 *node* pengirim)



Gambar 4.39 Grafik Perbandingan *Throughput* (10 *node* pengirim)



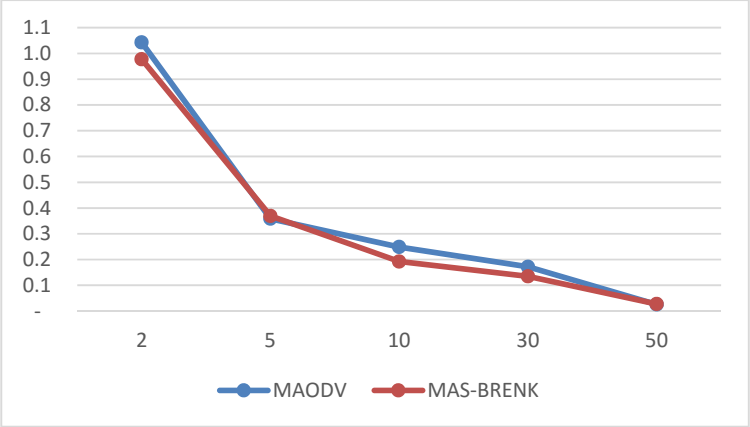
Gambar 4.40 Grafik Perbandingan *Throughput* (30 *node* pengirim)

4.2.3. *End to End Delay*

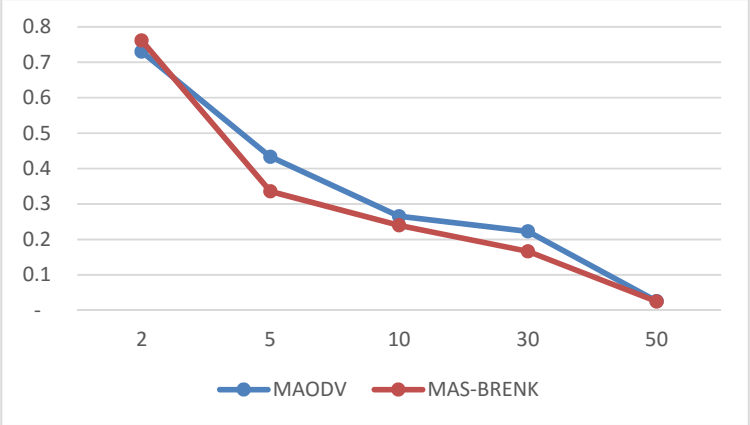
Delay adalah banyaknya paket data yang dikirimkan berdasarkan satuan waktu tertentu. Beberapa skenario menunjukkan adanya peningkatan dan penurunan *delay* pada protokol MAS-BRENK. Berikut tabel perbandingan *delay* yang dihasilkan oleh protokol MAODV dan MAS-BRENK.

Tabel 4.7 Tabel Perbandingan *Delay*

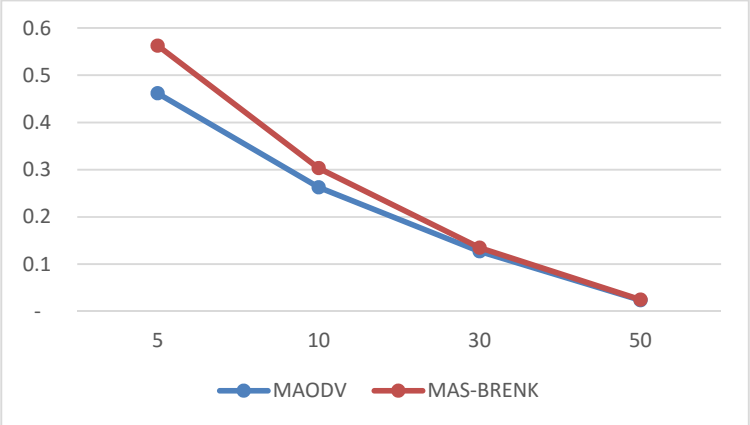
No	Sender	Receiver	*End to End Delay* (s)		Perbedaan (s)
			MAODV	MAS-BRENK	
1.	1	2	108.287	108.613	0.326
		5	237.795	219.560	-18.235
		10	316.952	324.749	7.797
		30	362.288	355.249	-7.039
		50	354.322	407.886	53.564
2.	2	2	374.081	322.997	-51.084
		5	824.338	848.079	23.741
		10	1,348.560	1,291.990	-56.570
		30	986.224	1,474.030	487.806
		50	1,433.230	1,392.290	-40.940
3.	5	5	3,935.610	3,881.710	-53.900
		10	6,019.550	5,977.770	-41.780
		30	7,354.020	7,303.880	-50.140
		50	7,072.530	8,146.010	1,073.480
4.	10	10	26,734.000	25,867.700	-866.300
		30	30,053.500	31,696.700	1,643.200
		50	29,973.700	29,301.300	-672.400
5.	30	30	246,982.000	234,258.000	-12,724.000
		50	248,632.000	242,574.000	-6,058.000



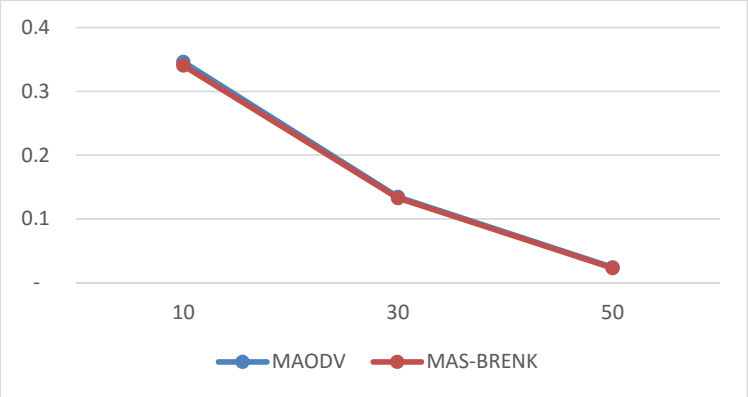
Gambar 4.41 Grafik Perbandingan *Delay* (1 *node* pengirim)



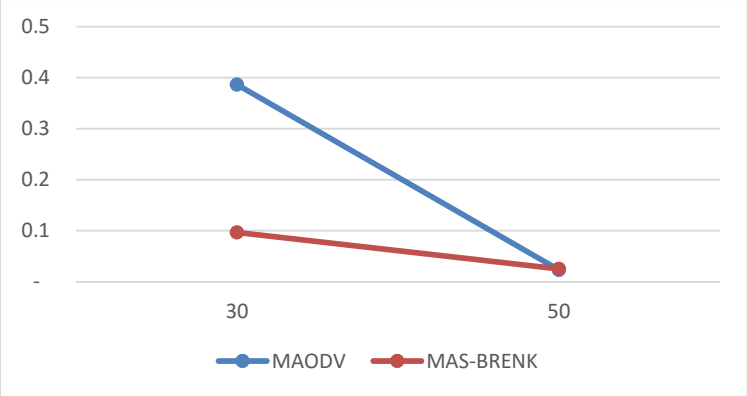
Gambar 4.42 Grafik Perbandingan *Delay* (2 *node* pengirim)



Gambar 4.43 Grafik Perbandingan *Delay* (5 *node* pengirim)



Gambar 4.44 Grafik Perbandingan *Delay* (10 *node* pengirim)



Gambar 4.45 Grafik Perbandingan *Delay* (30 *node* pengirim)

4.2.4. Perbandingan Jumlah Paket

Dari beberapa perbandingan hasil skenario diatas diambil rata – rata jumlah paket dari pembanding yaitu *send_mact, recvMACT_J, recvMACT_GL, recvMACT_P* terhadap jumlah pengirim. Didapatkan secara keseluruhan protokol MAS-BRENK mengalami peningkatan pengiriman jumlah paket data kecuali *recvMACT_J*.

Tabel 4.8 Perbandingan Jumlah Paket Data

No.	Jml. Pengirim	*send_mact*		*recvMACT_J*		*recvMACT_GL*		*recvMACT_P*	
		A	B	A	B	A	B	A	B
1.	1	745	759	444	431	15	15	262	287
2.	2	806	805	427	446	13	19	341	312

3.	5	1030	1,006	561	540	33	22	404	414
4.	10	1071	1,059	629	597	24	24	384	401
5.	30	912	988	649	658	13	17	204	272

*catatan : A = MAODV
B = MAS-BRENK

Dari hasil rata – rata perbandingan jumlah paket data diatas, dengan jumlah *node* pengirim sebanyak 1 *node* protokol MAS-BRENK mengalami peningkatan jumlah pengiriman paket *MACT* rata - rata sebesar 1.8% dari 745 paket menjadi 759 paket dibandingkan dengan protokol MAODV dan 8.4% dengan jumlah *node* pengirim sebanyak 30 *node* dari 992 paket menjadi 988 paket. Namun pengiriman paket *send_mact* mengalami penurunan rata – rata sebesar 0.1% dengan jumlah *node* pengirim sebanyak 2 *node* dari 806 paket menjadi 805 paket dan mengalami penurunan 2.3% dengan jumlah *node* pengirim sebanyak 5 *node*. Dari keseluruhan rata – rata protokol MAS-BRENK mengalami peningkatan jumlah pengiriman paket *send_mact* sekitar 1.3% dibandingkan protokol MAODV.

Pengiriman paket MACT berisi pengiriman paket dengan *flag* tertentu menuju *node* penerima. Jumlah paket MACT dengan *flag _j* yang diterima pada protokol MAS-BRENK mengalami penurunan rata – rata keseluruhan yaitu sebesar 1.2% dibandingkan protokol MAODV. Dengan jumlah *node* penerima paket sebanyak 1 *node*, protokol MAS-BRENK mengalami penurunan jumlah paket rata - rata sebesar 2.8% dari 444 paket menjadi 431 paket dibandingkan dengan protokol MAODV dan 3.8% yaitu dari 561 paket menjadi 540 paket dengan jumlah *node* pengirim sebanyak 5 *node*. Serta mengalami penurunan dengan jumlah *node* pengirim sebanyak 10 *node* sebesar 5% dengan jumlah *node* pengirim sebanyak 10 *node*. Namun penerima paket MACT dengan *flag _j* mengalami peningkatan rata – rata sebesar 4.5% dengan jumlah *node* pengirim sebanyak 2 *node*.

Protokol MAS-BRENK juga mengalami peningkatan penerimaan paket MACT dengan *flag _p* yang cukup besar yaitu rata – rata 8.23%. Dari hasil rata – rata perbandingan tersebut, dengan jumlah *node* pengirim sebanyak 1 *node*, protokol MAS-BRENK mengalami peningkatan penerima paket rata - rata sebesar 9.7% dibandingkan dengan protokol MAODV dari 262 paket menjadi 287 paket

dan 33% dengan jumlah *node* pengirim sebanyak 30 *node* yaitu dari 204 paket menjadi 272 paket. Namun mengalami penurunan rata - rata 8.4% dengan jumlah *node* pengirim sebanyak 2 *node* pengirim dari 341 paket menjadi 312 paket.

4.2.5. Perbandingan PDR, *Throughput* dan *End to End Delay*

Beberapa perbandingan hasil skenario diatas juga diambil rata – rata pembanding yaitu PDR, *throughput* dan *delay* terhadap jumlah pengirim. Didapatkan protokol MAS-BRENK mengalami peningkatan PDR dan *throughput* serta mengalami penurunan *end to end delay*.

Tabel 4.9 Perbandingan PDR, *Throughput* dan *Delay*

No.	Jml. Pengirim	PDR (%)		*Throughput* (kbps)		*Delay* (s)	
		MAODV	MAS-BRENK	MAODV	MAS-BRENK	MAODV	MAS-BRENK
1.	1	29.736	29.936	275.9	283.2	0.369	0.340
2.	2	28.324	28.772	993.3	1,065.9	0.335	0.305
3.	5	22.700	23.053	6,095.4	6,327.3	0.219	0.256
4.	10	21.798	21.389	28,920.4	28,955.2	0.168	0.165
5.	30	17.886	17.733	247,807.0	238,416.0	0.205	0.061

Dari hasil rata – rata perbandingan diatas, dengan jumlah *node* pengirim sebanyak satu *node* protokol MAS-BRENK mengalami peningkatan PDR rata - rata sebesar 0.7% dari 29.736% menjadi 29.936% dibandingkan dengan protokol MAODV dan 1.5% dengan jumlah *node* pengirim sebanyak 2 dan 5 *node*. Namun PDR mengalami penurunan rata – rata sebesar 1.9% dengan jumlah *node* pengirim sebanyak 10 *node* dari 21.798% menjadi 21.389% dan mengalami penurunan 9% dengan jumlah *node* pengirim sebanyak 30 *node*. Dari keseluruhan rata – rata protokol MAS-BRENK mengalami peningkatan PDR sekitar 0.2% dibandingkan protokol MAODV.

Dari sisi *throughput* protokol MAS-BRENK juga mengalami peningkatan rata – rata keseluruhan yaitu sebesar 2% dibandingkan protokol MAODV. Dengan jumlah *node* pengirim sebanyak satu *node* protokol MAS-BRENK mengalami peningkatan *throughput* rata - rata sebesar 2.7% dari 275.9kbps menjadi 283.2kbps dibandingkan dengan protokol MAODV dan 7.3% yaitu dari 993.3kbps menjadi

1065.9kbps dengan jumlah *node* pengirim sebanyak 2 *node*. Serta mengalami peningkatan dengan jumlah node pengirim sebanyak 5 *node* sebesar 3.8% dan 0.12% dengan jumlah *node* pengirim sebanyak 10 *node*. Namun *throughput* mengalami penurunan rata – rata sebesar 3.8% dengan jumlah node pengirim sebanyak 30 *node*.

Protokol MAS-BRENK juga mengalami penurunan delay yang cukup besar yaitu rata – rata 14.3%. Dari hasil rata – rata perbandingan tersebut, dengan jumlah *node* pengirim sebanyak satu *node* protokol MAS-BRENK mengalami penurunan *delay* rata - rata sebesar 7.9% dibandingkan dengan protokol MAODV dari 0.369s menjadi 0.364s, mengalami penurunan 8.9% dengan jumlah *node* pengirim sebanyak 2 *node* pengirim dari 0.335s ke 0.305s dan 1.7% dengan jumlah *node* pengirim sebanyak 10 *node*. Namun *delay* mengalami peningkatan rata – rata sebesar 17.2% dengan jumlah *node* pengirim sebanyak 5 *node* dari 0.219s menjadi 0.256s. Penurunan *delay* cukup besar yaitu 70.3% terjadi ketika jumlah *node* pengirim sebanyak 30 *node* yaitu dari 0.205s menjadi 0.061s.

[Halaman ini sengaja dikosongkan]

BAB 5
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Kesimpulan yang dapat diambil dari penelitian ini adalah sebagai berikut:

1. Protokol MAS-BRENK mempertimbangkan beberapa parameter berdasarkan bobot posisi, kecepatan, dan *delay* transmisi. Tidak semua *node* dapat melakukan *join* menjadi anggota *multicast*. Dan tidak semua *node* menunggu *time-out* untuk melakukan *prune* dari anggota *multicast*.
2. Jumlah *node* pada protokol MAS-BRENK yang melakukan *join* lebih sedikit daripada MAODV. Sehingga didapatkan sedikit peningkatan pada PDR dan *throughput*.
3. Jumlah *node* yang melakukan *pruning* pada protokol MAS-BRENK juga lebih banyak daripada protokol MAODV. Sehingga *end-to-end delay* yang dihasilkan lebih kecil.
4. Rata – rata PDR dan *throughput* protokol MAS-BRENK meningkat, rata – rata *delay* juga menurun daripada protokol MAODV. Namun secara detail dari skenario jumlah pengirim, dari beberapa skenario terjadi penurunan PDR dan *throughput* serta peningkatan *delay*.

5.2 Saran

1. Parameter lain untuk mempertimbangkan perhitungan bobot pada protokol MAS-BRENK perlu ditambahkan.
2. Perlu dilakukan penelitian lebih lanjut mengenai proses *pruning* agar pelepasan keanggotaan dari anggota *multicast* lebih optimal lagi dari sisi *PDR* dan *throughput*.
3. Dibutuhkan lebih banyak skenario mobilitas dan protokol pembanding lain untuk menguji protokol MAS-BRENK.

[Halaman ini sengaja dikosongkan]

DAFTAR PUSTAKA

Anggoro, R., (2008). "*Static Intersection Node-based Multicast Protocol in VANET*", Thesis, Computer Science and Information Engineering Department in National Taiwan University of Science and Technology.

Anupama, K.S.S., Gowri, S.S., PrabhakaraRao, B., Rajesh. P., (2015). "*Application of MADM Algorithms of Network Selection*", International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering (IJIREEICE), Vol.3 Issue 6.

Balakrishna, R., Rao, U. R., & Geethanjali, N. (2010). "*Performance issues on AODV and AOMDV for MANETS*". International Journal of Computer Science and Information Technologies (IJCSIT), (hal. 38-43).

Cavalcante, E. S., Aquino, A. L. L., Gisele L. Pappa, (2012), "*Roadside Unit Deployment for Information Dissemination in a VANET: An Evolutionary Approach*", GECCO'12 Companion, July 7–11.

Corson, S. & Macker, J., (1999), "*Mobile Ad-hoc Network (MANET) : Routing Protocol Performance Issues and Evaluation Considerations*". *Request for Comments* (International) RFC 2503.

Dimyati, M., Anggoro, R. & Wibisono, W., (2016). "Pemilihan *Node Rebroadcast* untuk Meningkatkan Kinerja Protokol *Multicast* AODV (MAODV) pada VANETs". Volume 14, Nomor 2. JUTI.

Dorle, S., Vidhale, B. & Chakole, M., (2011). "*Evaluation of Multipath, Unipath and Hybrid Routing Protocols for Vehicular Ad Hoc Networks*". In Fourth International Conference on Emerging Trends in Engineering & Technology., 2011. IEEE Press.

Festag, A., Hessler, A., Baldessari, R., Le, L., Zhang, W., & Westhoff, D., (2008) "*Vehicle-To-Vehicle And Road-Side Sensor Communication For Enhanced Road Safety*", 9th International Conference On Intelligent Tutoring Systems (ITS 2008) IEEE Press

Ghazemi, M., & Bag-Mohammady, M., (2012), "*Classification of Multicast Routing Protocols for Mobile Ad Hoc Network*", International Journal of IEEE, ICTC12, pp. 789-794.

Harri, J., Filali, F. & Bonnet, C., (2006). "*Mobility Models for Vehicular Ad Hoc Network: A Survey and Taxonomy*". Research Report. Sophia Antipolis: Eurecom.

Jemaa, I. B., Shagdar, O., Martinez, F. J., Garrido, P., Nashashibi, F., (2015), "*Extended Mobility Management and Routing Protocols for Internet-to-VANET Multicasting*", IEEE 12th Consumer Communication and Networking Conferences, pp. 904–909.

Joshi, A., Kaur, R., (2015), "*A Novel Multicast Routing Protocol for VANET*", International Journal of IEEE, IACC15, pp. 41-45.

Li, F. & Wang, T., (2007). "*Routing in Vehicular Ad hoc Network: A survey*". Vehicular Technology Magazine, June. pp.12-22.

Menouar, H., Filali, F. and Lenardi, M., (2006). "*A survey and qualitative analysis of MAC protocols for vehicular ad hoc networks*". Wireless Communications, IEEE, 13 (5), pp. 30--35.

Najafabadi, R.T., (2011). "*A Survey on Routing Technique for Vehicular Ad-hoc Networks*".

Nakamura, M., Kitani, T., Sun, W., Shibata, N., Yasumoto, K. and Ito, M., (2010). "*A method for improving data delivery efficiency in delay tolerant vanet with scheduled routes of cars*". pp. 1--5.

Perkins, C.E., Royer, E.M., 1999, "*Ad hoc On-Demand distance Vector routing*", Proceedings of 2nd IEEE workshop on mobile computing systems and applications, pp 90-100.

Ramachandran, L., Sukumaran, S., Sunny, S. R., (2013). "*An Intersection Based Traffic Aware Routing With Low Overhead in VANET*", International Journal of Digital Information and Wireless Communications (IJDIWC) 3(2): 190-196

Royer, E.M., Perkins, C.E., (2000). "*Multicast Ad hoc On-Demand Distance Vector MAODV*) *Routing*", IETF, Internet Draft: draft-ietf-manet-maodv-00.txt.

Savitha, K., Chandrasekar, C., (2011), "*Vertical Handover Decition Shcemes using SAW and WPM for Network Selection in Heterogeneous Wireless Networks*", Global Journal of Computer Science and Technology, Vol.11 Issue 9.

Shurdi, O., Miho, R., Kamo, B., Kholici, V., Rakipi. A., (2011), "*Performance Analysis of Multicast Routing Protocols MAODV, ODMRP, and ADMR for MANETs*", International Conference of IEEE NBiS'11.

Su, W., Lee, S.-J., & Gerla, M., (2000). "*Mobility Prediction and Routing in Ad Hoc Wireless Networks*". International Journal of Network Management.

Su, W., Lee, S.-J., & Gerla, M., (2000). "*Mobility Prediction in Wireless Networks*". IEEE Military Communications Conference, (pp. 491-495).

Vidhale, B. & Dorle, S., 2011. "*Performance Analysis of Routing Protocols in Realistic Environtment for Vehicular Ad Hoc Network*". In 21st International Conference on System Engineering. IEEE Press.

Xia, H., Yu, J., Xia, S., Jia, Z., Sha, E. H. M., (2014). "*Applying Link Stability Estimation Mechanism to Multicast Routing in MANETs*". Journal of System Architecture 60, pp 467-480.

Yong Ding, Chen Wang, Li Xiao, (2007). "*A Static-Node Assisted Adaptive Routing Protocol in Vehicular Networks*", VANET '07 Proceedings of the fourth ACM international workshop on Vehicular ad hoc networks Pages 59 – 68

[Halaman ini sengaja dikosongkan]

Lampiran 1

File tesis50-10.tcl

```
if {$argc != 3} {error "Usages: ns tesis50-10.tcl <no_of_senders>
<no_receivers> <scenario>"
}

set opt(stop) 910.0
set nodes   50
set mobility      1
set scenario      [lindex $argv 2]
set pausetime     0
set traffic cbr
set senders [lindex $argv 0]
set receivers   [lindex $argv 1]

set ns_ [new Simulator]
set topo [new Topography]
$topo load_flatgrid 2000 2000

set tracefd [open ./trace-scen-$nodes-$scenario-$senders-
$receivers w]
$ns_ trace-all $tracefd

set god_ [create-god $nodes]
$ns_ node-config -adhocRouting AODV \
   -llType LL \
   -macType Mac/802_11 \
   -ifqLen 50 \
   -ifqType Queue/DropTail/PriQueue \
   -antType Antenna/OmniAntenna \
   -propType Propagation/TwoRayGround \
   -phyType Phy/WirelessPhy \
   -channel [new Channel/WirelessChannel] \
   -topoInstance $topo \
   -agentTrace ON \
   -routerTrace ON \

for {set i 0} {$i < $nodes} {incr i} {
   set node_($i) [$ns_ node]
   $node_($i) random-motion 0;
}

source "traffic/$traffic-$nodes-$senders-$receivers"
source "scenarios/grid-$nodes"

for {set i 0} {$i < $nodes} {incr i} {
   $ns_ at $opt(stop) "$node_($i) reset";
}

$ns_ at $opt(stop) "$ns_ halt"
$ns_ run
```

[Halaman ini sengaja dikosongkan]

Lampiran 2

File grid-50

```
$node_(0) set X_ 13.15
$node_(0) set Y_ 1395.05
$node_(0) set Z_ 0.000000000000
$node_(1) set X_ 1386.85
$node_(1) set Y_ 1004.95
$node_(1) set Z_ 0.000000000000
$node_(2) set X_ 1004.95
$node_(2) set Y_ 1013.15
$node_(2) set Z_ 0.000000000000
$node_(3) set X_ 1788.35
$node_(3) set Y_ 4.95
$node_(3) set Z_ 0.000000000000
$node_(4) set X_ 1804.95
$node_(4) set Y_ 613.15
$node_(4) set Z_ 0.000000000000
$node_(5) set X_ 186.85
$node_(5) set Y_ 404.95
$node_(5) set Z_ 0.000000000000
$node_(6) set X_ 1604.95
$node_(6) set Y_ 13.15
$node_(6) set Z_ 0.000000000000
$node_(7) set X_ 213.15
$node_(7) set Y_ 1395.05
$node_(7) set Z_ 0.000000000000
$node_(8) set X_ 204.95
$node_(8) set Y_ 1413.15
$node_(8) set Z_ 0.000000000000
$node_(9) set X_ 1013.15
$node_(9) set Y_ 595.05
$node_(9) set Z_ 0.000000000000
$node_(10) set X_ 186.85
$node_(10) set Y_ 1804.95
$node_(10) set Z_ 0.000000000000
$node_(11) set X_ 613.15
$node_(11) set Y_ 1595.05
$node_(11) set Z_ 0.000000000000
$node_(12) set X_ 1413.15
$node_(12) set Y_ 1395.05
$node_(12) set Z_ 0.000000000000
$node_(13) set X_ 604.95
$node_(13) set Y_ 1613.15
$node_(13) set Z_ 0.000000000000
$node_(14) set X_ 1404.95
$node_(14) set Y_ 1213.15
$node_(14) set Z_ 0.000000000000
$node_(15) set X_ 413.15
$node_(15) set Y_ 195.05
$node_(15) set Z_ 0.000000000000
$node_(16) set X_ 1386.85
$node_(16) set Y_ 204.95
```

```
$node_(16) set Z_ 0.000000000000
$node_(17) set X_ 1386.85
$node_(17) set Y_ 404.95
$node_(17) set Z_ 0.000000000000
$node_(18) set X_ 795.05
$node_(18) set Y_ 786.85
$node_(18) set Z_ 0.000000000000
$node_(19) set X_ 1613.15
$node_(19) set Y_ 1195.05
$node_(19) set Z_ 0.000000000000
$node_(20) set X_ 213.15
$node_(20) set Y_ 1195.05
$node_(20) set Z_ 0.000000000000
$node_(21) set X_ 1413.15
$node_(21) set Y_ 1395.05
$node_(21) set Z_ 0.000000000000
$node_(22) set X_ 395.05
$node_(22) set Y_ 586.85
$node_(22) set Z_ 0.000000000000
$node_(23) set X_ 1604.95
$node_(23) set Y_ 1613.15
$node_(23) set Z_ 0.000000000000
$node_(24) set X_ 395.05
$node_(24) set Y_ 1586.85
$node_(24) set Z_ 0.000000000000
$node_(25) set X_ 413.15
$node_(25) set Y_ 995.05
$node_(25) set Z_ 0.000000000000
$node_(26) set X_ 1004.95
$node_(26) set Y_ 1213.15
$node_(26) set Z_ 0.000000000000
$node_(27) set X_ 413.15
$node_(27) set Y_ 1595.05
$node_(27) set Z_ 0.000000000000
$node_(28) set X_ 813.15
$node_(28) set Y_ 195.05
$node_(28) set Z_ 0.000000000000
$node_(29) set X_ 1795.05
$node_(29) set Y_ 586.85
$node_(29) set Z_ 0.000000000000
$node_(30) set X_ 404.95
$node_(30) set Y_ 1413.15
$node_(30) set Z_ 0.000000000000
$node_(31) set X_ 213.15
$node_(31) set Y_ 1795.05
$node_(31) set Z_ 0.000000000000
$node_(32) set X_ 986.85
$node_(32) set Y_ 1604.95
$node_(32) set Z_ 0.000000000000
$node_(33) set X_ 604.95
$node_(33) set Y_ 1013.15
$node_(33) set Z_ 0.000000000000
$node_(34) set X_ 1186.85
$node_(34) set Y_ 404.95
$node_(34) set Z_ 0.000000000000
$node_(35) set X_ 386.85
$node_(35) set Y_ 604.95
$node_(35) set Z_ 0.000000000000
```

```
$node_(36) set X_ 1186.85
$node_(36) set Y_ 1404.95
$node_(36) set Z_ 0.000000000000
$node_(37) set X_ 604.95
$node_(37) set Y_ 13.15
$node_(37) set Z_ 0.000000000000
$node_(38) set X_ 1004.95
$node_(38) set Y_ 813.15
$node_(38) set Z_ 0.000000000000
$node_(39) set X_ 4.95
$node_(39) set Y_ 786.85
$node_(39) set Z_ 0.000000000000
$node_(40) set X_ 1786.85
$node_(40) set Y_ 604.95
$node_(40) set Z_ 0.000000000000
$node_(41) set X_ 1395.05
$node_(41) set Y_ 1186.85
$node_(41) set Z_ 0.000000000000
$node_(42) set X_ 613.15
$node_(42) set Y_ 395.05
$node_(42) set Z_ 0.000000000000
$node_(43) set X_ 613.15
$node_(43) set Y_ 1395.05
$node_(43) set Z_ 0.000000000000
$node_(44) set X_ 1213.15
$node_(44) set Y_ 1395.05
$node_(44) set Z_ 0.000000000000
$node_(45) set X_ 604.95
$node_(45) set Y_ 613.15
$node_(45) set Z_ 0.000000000000
$node_(46) set X_ 1786.85
$node_(46) set Y_ 1004.95
$node_(46) set Z_ 0.000000000000
$node_(47) set X_ 1788.35
$node_(47) set Y_ 1804.95
$node_(47) set Z_ 0.000000000000
$node_(48) set X_ 986.85
$node_(48) set Y_ 1004.95
$node_(48) set Z_ 0.000000000000
$node_(49) set X_ 404.95
$node_(49) set Y_ 13.15
$node_(49) set Z_ 0.000000000000
$ns_ at 55.0 "$node_(5) setdest 4.95 658.87 8.87"
$ns_ at 55.0 "$node_(6) setdest 1374.96 204.95 8.83"
$ns_ at 55.0 "$node_(7) setdest 625.39 1395.05 8.76"
$ns_ at 55.0 "$node_(8) setdest 427.12 1595.05 9.63"
$ns_ at 56.0 "$node_(0) setdest 195.05 1086.31 9.27"
$ns_ at 56.0 "$node_(9) setdest 1204.95 823.74 9.66"
$ns_ at 56.0 "$node_(10) setdest 4.95 1578.89 8.95"
$ns_ at 56.0 "$node_(11) setdest 795.05 1384.14 9.96"
$ns_ at 56.0 "$node_(12) setdest 1404.2 1404.95 6.25"
$ns_ at 56.0 "$node_(13) setdest 423.85 1801.65 9.50"
$ns_ at 56.0 "$node_(14) setdest 1395.05 1213.23 9.64"
$ns_ at 56.0 "$node_(15) setdest 604.95 364.27 9.35"
$ns_ at 56.0 "$node_(16) setdest 1204.95 366.0 9.19"
$ns_ at 56.0 "$node_(17) setdest 1053.63 404.95 8.83"
$ns_ at 56.0 "$node_(18) setdest 647.33 604.95 8.97"
$ns_ at 56.0 "$node_(1) setdest 918.69 1004.95 9.26"
```

```
$ns_ at 56.0 "$node_(19) setdest 1663.57 1204.95 9.62"
$ns_ at 56.0 "$node_(20) setdest 404.95 1310.72 9.90"
$ns_ at 56.0 "$node_(21) setdest 1480.89 1404.95 8.93"
$ns_ at 56.0 "$node_(22) setdest 404.95 503.23 9.88"
$ns_ at 56.0 "$node_(23) setdest 1521.49 1804.95 9.25"
$ns_ at 56.0 "$node_(24) setdest 395.05 1318.9 9.63"
$ns_ at 56.0 "$node_(25) setdest 604.95 1063.84 9.49"
$ns_ at 56.0 "$node_(26) setdest 941.13 1404.95 8.85"
$ns_ at 56.0 "$node_(27) setdest 595.05 1534.68 9.92"
$ns_ at 56.0 "$node_(28) setdest 1001.65 234.52 9.81"
$ns_ at 56.0 "$node_(2) setdest 795.05 1119.87 8.76"
$ns_ at 56.0 "$node_(29) setdest 1801.65 438.76 9.92"
$ns_ at 56.0 "$node_(30) setdest 398.35 1560.37 9.37"
$ns_ at 56.0 "$node_(31) setdest 395.05 1767.73 8.87"
$ns_ at 56.0 "$node_(32) setdest 798.65 1594.04 9.21"
$ns_ at 56.0 "$node_(33) setdest 594.97 1201.22 9.96"
$ns_ at 56.0 "$node_(34) setdest 1009.12 404.95 5.34"
$ns_ at 56.0 "$node_(35) setdest 218.14 601.65 9.81"
$ns_ at 56.0 "$node_(36) setdest 1028.28 1401.65 9.98"
$ns_ at 56.0 "$node_(37) setdest 604.95 164.69 9.77"
$ns_ at 56.0 "$node_(38) setdest 1004.95 954.34 9.82"
$ns_ at 56.0 "$node_(3) setdest 1322.39 1.65 9.86"
$ns_ at 56.0 "$node_(39) setdest 1.65 654.93 9.73"
$ns_ at 56.0 "$node_(40) setdest 1662.43 604.95 9.65"
$ns_ at 56.0 "$node_(41) setdest 1395.05 1074.44 9.24"
$ns_ at 56.0 "$node_(42) setdest 718.83 398.35 9.20"
$ns_ at 56.0 "$node_(43) setdest 704.78 1398.35 9.71"
$ns_ at 56.0 "$node_(44) setdest 1299.07 1398.35 9.88"
$ns_ at 56.0 "$node_(45) setdest 601.65 689.84 9.54"
$ns_ at 56.0 "$node_(46) setdest 1722.72 1004.95 9.16"
$ns_ at 56.0 "$node_(47) setdest 1729.67 1804.95 9.19"
$ns_ at 56.0 "$node_(48) setdest 947.55 1004.95 9.59"
$ns_ at 56.0 "$node_(4) setdest 1536.56 804.95 9.66"
$ns_ at 56.0 "$node_(49) setdest 404.95 48.61 9.29"
$ns_ at 56.0 "$node_(5) setdest 4.95 668.55 9.68"
$ns_ at 56.0 "$node_(6) setdest 1365.01 204.95 9.95"
$ns_ at 56.0 "$node_(7) setdest 634.83 1395.05 9.44"
$ns_ at 56.0 "$node_(8) setdest 436.53 1595.05 9.40"
$ns_ at 57.0 "$node_(0) setdest 195.05 1077.01 9.30"
$ns_ at 57.0 "$node_(9) setdest 1204.95 833.5 9.76"
$ns_ at 57.0 "$node_(10) setdest 4.95 1569.14 9.76"
$ns_ at 57.0 "$node_(11) setdest 795.05 1374.18 9.96"
$ns_ at 57.0 "$node_(12) setdest 1396.02 1404.95 8.19"
$ns_ at 57.0 "$node_(13) setdest 414.12 1801.65 9.73"
$ns_ at 57.0 "$node_(14) setdest 1395.05 1204.33 8.90"
$ns_ at 57.0 "$node_(15) setdest 604.95 373.63 9.36"
$ns_ at 57.0 "$node_(16) setdest 1204.95 375.77 9.77"
$ns_ at 57.0 "$node_(17) setdest 1043.89 404.95 9.74"
$ns_ at 57.0 "$node_(18) setdest 637.58 604.95 9.75"
$ns_ at 57.0 "$node_(1) setdest 909.17 1004.95 9.52"
$ns_ at 57.0 "$node_(19) setdest 1654.62 1204.95 8.95"
$ns_ at 57.0 "$node_(20) setdest 404.95 1319.47 8.75"
$ns_ at 57.0 "$node_(21) setdest 1471.03 1404.95 9.86"
$ns_ at 57.0 "$node_(22) setdest 404.95 513.17 9.94"
$ns_ at 57.0 "$node_(23) setdest 1511.98 1804.95 9.51"
$ns_ at 57.0 "$node_(24) setdest 395.05 1308.95 9.95"
$ns_ at 57.0 "$node_(25) setdest 604.95 1072.61 8.78"
$ns_ at 57.0 "$node_(26) setdest 931.2 1404.95 9.93"
```

```
$ns_ at 57.0 "$node_(27) setdest 595.05 1525.67 9.01"
$ns_ at 57.0 "$node_(28) setdest 1001.65 243.29 8.77"
$ns_ at 57.0 "$node_(2) setdest 795.05 1111.04 8.83"
$ns_ at 57.0 "$node_(29) setdest 1801.65 447.92 9.17"
$ns_ at 57.0 "$node_(30) setdest 398.35 1550.76 9.61"
$ns_ at 57.0 "$node_(31) setdest 395.05 1759.03 8.70"
$ns_ at 57.0 "$node_(32) setdest 798.35 1584.74 9.32"
$ns_ at 57.0 "$node_(33) setdest 585.51 1201.65 9.49"
$ns_ at 57.0 "$node_(34) setdest 1004.95 408.92 6.95"
$ns_ at 57.0 "$node_(35) setdest 212.06 601.65 6.07"
$ns_ at 57.0 "$node_(36) setdest 1018.69 1401.65 9.59"
$ns_ at 57.0 "$node_(37) setdest 604.95 174.05 9.36"
$ns_ at 57.0 "$node_(38) setdest 1004.95 963.56 9.22"
$ns_ at 57.0 "$node_(3) setdest 1313.11 1.65 9.28"
$ns_ at 57.0 "$node_(39) setdest 1.65 646.17 8.76"
$ns_ at 57.0 "$node_(40) setdest 1653.44 604.95 8.99"
$ns_ at 57.0 "$node_(41) setdest 1395.05 1065.04 9.40"
$ns_ at 57.0 "$node_(42) setdest 728.77 398.35 9.94"
$ns_ at 57.0 "$node_(43) setdest 714.13 1398.35 9.36"
$ns_ at 57.0 "$node_(44) setdest 1308.67 1398.35 9.59"
$ns_ at 57.0 "$node_(45) setdest 601.65 699.13 9.30"
$ns_ at 57.0 "$node_(46) setdest 1713.82 1004.95 8.90"
$ns_ at 57.0 "$node_(47) setdest 1720.15 1804.95 9.52"
$ns_ at 57.0 "$node_(48) setdest 938.37 1004.95 9.18"
$ns_ at 57.0 "$node_(4) setdest 1526.85 804.95 9.71"
$ns_ at 57.0 "$node_(49) setdest 404.95 57.4 8.79"
$ns_ at 57.0 "$node_(5) setdest 4.95 677.69 9.14"
$ns_ at 57.0 "$node_(6) setdest 1355.14 204.95 9.87"
$ns_ at 57.0 "$node_(7) setdest 644.33 1395.05 9.50"
$ns_ at 57.0 "$node_(8) setdest 446.5 1595.05 9.98"
$ns_ at 58.0 "$node_(0) setdest 195.05 1067.56 9.45"
$ns_ at 58.0 "$node_(9) setdest 1204.95 842.47 8.97"
$ns_ at 58.0 "$node_(10) setdest 4.95 1560.43 8.71"
$ns_ at 58.0 "$node_(11) setdest 795.05 1365.1 9.07"
$ns_ at 58.0 "$node_(12) setdest 1386.16 1404.95 9.86"
$ns_ at 58.0 "$node_(13) setdest 405.02 1801.65 9.10"
$ns_ at 58.0 "$node_(14) setdest 1395.05 1194.78 9.55"
$ns_ at 58.0 "$node_(15) setdest 604.95 382.57 8.94"
$ns_ at 58.0 "$node_(16) setdest 1204.95 384.91 9.14"
$ns_ at 58.0 "$node_(17) setdest 1035.17 404.95 8.72"
$ns_ at 58.0 "$node_(18) setdest 627.68 604.95 9.89"
$ns_ at 58.0 "$node_(1) setdest 900.22 1004.95 8.95"
$ns_ at 58.0 "$node_(19) setdest 1644.9 1204.95 9.72"
$ns_ at 58.0 "$node_(20) setdest 404.95 1329.31 9.84"
$ns_ at 58.0 "$node_(21) setdest 1461.48 1404.95 9.54"
$ns_ at 58.0 "$node_(22) setdest 404.95 522.89 9.72"
$ns_ at 58.0 "$node_(23) setdest 1502.26 1804.95 9.72"
$ns_ at 58.0 "$node_(24) setdest 395.05 1299.96 8.99"
$ns_ at 58.0 "$node_(25) setdest 604.95 1082.09 9.48"
$ns_ at 58.0 "$node_(26) setdest 922.42 1404.95 8.78"
$ns_ at 58.0 "$node_(27) setdest 595.05 1515.87 9.79"
$ns_ at 58.0 "$node_(28) setdest 1001.65 253.17 9.88"
$ns_ at 58.0 "$node_(2) setdest 795.05 1101.54 9.50"
$ns_ at 58.0 "$node_(29) setdest 1804.95 457.72 9.79"
$ns_ at 58.0 "$node_(30) setdest 395.05 1541.41 9.35"
$ns_ at 58.0 "$node_(31) setdest 395.05 1749.67 9.36"
$ns_ at 58.0 "$node_(32) setdest 798.35 1575.24 9.50"
$ns_ at 58.0 "$node_(33) setdest 576.25 1201.65 9.27"
```

```
$ns_ at 58.0 "$node_(34) setdest 1004.95 417.84 8.92"
$ns_ at 58.0 "$node_(35) setdest 203.66 600.95 8.48"
$ns_ at 58.0 "$node_(36) setdest 1011.27 1401.65 7.42"
$ns_ at 58.0 "$node_(37) setdest 604.95 183.53 9.48"
$ns_ at 58.0 "$node_(38) setdest 1004.95 972.73 9.17"
$ns_ at 58.0 "$node_(3) setdest 1304.32 1.65 8.79"
$ns_ at 58.0 "$node_(39) setdest 1.65 637.33 8.84"
$ns_ at 58.0 "$node_(40) setdest 1644.41 604.95 9.03"
$ns_ at 58.0 "$node_(41) setdest 1395.05 1056.31 8.72"
$ns_ at 58.0 "$node_(42) setdest 737.91 398.35 9.15"
$ns_ at 58.0 "$node_(43) setdest 724.07 1398.35 9.93"
$ns_ at 58.0 "$node_(44) setdest 1317.95 1398.35 9.28"
$ns_ at 58.0 "$node_(45) setdest 601.65 708.71 9.57"
$ns_ at 58.0 "$node_(46) setdest 1704.52 1004.95 9.30"
$ns_ at 58.0 "$node_(47) setdest 1710.99 1804.95 9.16"
$ns_ at 58.0 "$node_(48) setdest 928.68 1004.95 9.69"
$ns_ at 58.0 "$node_(4) setdest 1517.26 804.95 9.59"
$ns_ at 58.0 "$node_(49) setdest 404.95 66.85 9.45"
$ns_ at 58.0 "$node_(5) setdest 4.95 686.66 8.97"
$ns_ at 58.0 "$node_(6) setdest 1345.75 204.95 9.39"
$ns_ at 58.0 "$node_(7) setdest 653.19 1395.05 8.86"
$ns_ at 58.0 "$node_(8) setdest 455.62 1595.05 9.12"
$ns_ at 59.0 "$node_(0) setdest 195.05 1057.81 9.74"
$ns_ at 59.0 "$node_(9) setdest 1204.95 851.53 9.07"
$ns_ at 59.0 "$node_(10) setdest 4.95 1550.52 9.91"
$ns_ at 59.0 "$node_(11) setdest 795.05 1356.4 8.71"
$ns_ at 59.0 "$node_(12) setdest 1377.01 1404.95 9.15"
$ns_ at 59.0 "$node_(13) setdest 396.08 1801.65 8.94"
$ns_ at 59.0 "$node_(14) setdest 1395.05 1185.24 9.54"
$ns_ at 59.0 "$node_(15) setdest 604.95 391.89 9.32"
$ns_ at 59.0 "$node_(16) setdest 1205.86 394.35 9.70"
$ns_ at 59.0 "$node_(17) setdest 1025.17 404.95 10.00"
$ns_ at 59.0 "$node_(18) setdest 618.19 604.95 9.49"
$ns_ at 59.0 "$node_(1) setdest 891.12 1004.95 9.10"
$ns_ at 59.0 "$node_(19) setdest 1635.41 1204.95 9.49"
$ns_ at 59.0 "$node_(20) setdest 404.95 1338.37 9.06"
$ns_ at 59.0 "$node_(21) setdest 1452.18 1404.95 9.30"
$ns_ at 59.0 "$node_(22) setdest 404.95 531.76 8.87"
$ns_ at 59.0 "$node_(23) setdest 1493.1 1804.95 9.16"
$ns_ at 59.0 "$node_(24) setdest 395.05 1290.36 9.60"
$ns_ at 59.0 "$node_(25) setdest 604.95 1091.34 9.24"
$ns_ at 59.0 "$node_(26) setdest 913.51 1404.95 8.91"
$ns_ at 59.0 "$node_(27) setdest 595.05 1506.06 9.81"
$ns_ at 59.0 "$node_(28) setdest 1004.95 262.37 9.20"
$ns_ at 59.0 "$node_(2) setdest 795.05 1092.76 8.79"
$ns_ at 59.0 "$node_(29) setdest 1804.95 467.4 9.68"
$ns_ at 59.0 "$node_(30) setdest 395.05 1532.07 9.33"
$ns_ at 59.0 "$node_(31) setdest 395.05 1740.42 9.25"
$ns_ at 59.0 "$node_(32) setdest 798.35 1566.52 8.72"
$ns_ at 59.0 "$node_(33) setdest 566.79 1201.65 9.45"
$ns_ at 59.0 "$node_(34) setdest 1004.95 427.3 9.45"
$ns_ at 59.0 "$node_(35) setdest 198.56 593.42 9.69"
$ns_ at 59.0 "$node_(36) setdest 1002.07 1401.65 9.20"
$ns_ at 59.0 "$node_(37) setdest 604.95 192.39 8.85"
$ns_ at 59.0 "$node_(38) setdest 1004.95 982.57 9.84"
$ns_ at 59.0 "$node_(3) setdest 1294.78 1.65 9.54"
$ns_ at 59.0 "$node_(39) setdest 1.65 628.49 8.83"
$ns_ at 59.0 "$node_(40) setdest 1635.42 604.95 8.99"
```

```
$ns_ at 59.0 "$node_(41) setdest 1395.05 1047.3 9.01"
$ns_ at 59.0 "$node_(42) setdest 747.22 398.35 9.30"
$ns_ at 59.0 "$node_(43) setdest 733.87 1398.35 9.80"
$ns_ at 59.0 "$node_(44) setdest 1327.01 1398.35 9.06"
$ns_ at 59.0 "$node_(45) setdest 601.65 717.76 9.05"
$ns_ at 59.0 "$node_(46) setdest 1695.22 1004.95 9.30"
$ns_ at 59.0 "$node_(47) setdest 1702.23 1804.95 8.76"
$ns_ at 59.0 "$node_(48) setdest 919.37 1004.95 9.31"
$ns_ at 59.0 "$node_(4) setdest 1507.69 804.95 9.57"
$ns_ at 59.0 "$node_(49) setdest 404.95 76.1 9.25"
$ns_ at 59.0 "$node_(5) setdest 4.95 696.17 9.51"
$ns_ at 59.0 "$node_(6) setdest 1335.91 204.95 9.85"
$ns_ at 59.0 "$node_(7) setdest 662.01 1395.05 8.82"
$ns_ at 59.0 "$node_(8) setdest 464.67 1595.05 9.05"
$ns_ at 60.0 "$node_(0) setdest 195.05 1048.75 9.07"
$ns_ at 60.0 "$node_(9) setdest 1204.95 860.57 9.04"
$ns_ at 60.0 "$node_(10) setdest 4.95 1541.68 8.83"
$ns_ at 60.0 "$node_(11) setdest 795.05 1346.96 9.44"
$ns_ at 60.0 "$node_(12) setdest 1367.09 1404.95 9.92"
$ns_ at 60.0 "$node_(13) setdest 386.25 1801.65 9.83"
$ns_ at 60.0 "$node_(14) setdest 1395.05 1175.77 9.46"
$ns_ at 60.0 "$node_(15) setdest 604.95 401.81 9.92"
$ns_ at 60.0 "$node_(16) setdest 1215.31 398.35 9.61"
$ns_ at 60.0 "$node_(17) setdest 1015.37 404.95 9.80"
$ns_ at 60.0 "$node_(18) setdest 611.43 604.95 6.77"
$ns_ at 60.0 "$node_(1) setdest 882.39 1004.95 8.73"
$ns_ at 60.0 "$node_(19) setdest 1625.62 1204.95 9.79"
$ns_ at 60.0 "$node_(20) setdest 404.95 1348.02 9.65"
$ns_ at 60.0 "$node_(21) setdest 1443.46 1404.95 8.72"
$ns_ at 60.0 "$node_(22) setdest 404.95 540.69 8.92"
$ns_ at 60.0 "$node_(23) setdest 1484.22 1804.95 8.88"
$ns_ at 60.0 "$node_(24) setdest 395.05 1280.91 9.45"
$ns_ at 60.0 "$node_(25) setdest 604.95 1100.44 9.11"
$ns_ at 60.0 "$node_(26) setdest 903.87 1404.95 9.63"
$ns_ at 60.0 "$node_(27) setdest 595.05 1496.53 9.53"
$ns_ at 60.0 "$node_(28) setdest 1004.95 271.75 9.38"
$ns_ at 60.0 "$node_(2) setdest 795.05 1083.01 9.74"
$ns_ at 60.0 "$node_(29) setdest 1804.95 477.16 9.76"
$ns_ at 60.0 "$node_(30) setdest 395.05 1522.55 9.52"
$ns_ at 60.0 "$node_(31) setdest 395.05 1731.34 9.08"
$ns_ at 60.0 "$node_(32) setdest 798.35 1557.75 8.77"
$ns_ at 60.0 "$node_(33) setdest 557.52 1201.65 9.28"
$ns_ at 60.0 "$node_(34) setdest 1004.95 436.78 9.49"
$ns_ at 60.0 "$node_(35) setdest 198.35 584.29 9.15"
$ns_ at 60.0 "$node_(36) setdest 993.29 1401.65 8.78"
$ns_ at 60.0 "$node_(37) setdest 604.95 201.38 8.99"
$ns_ at 60.0 "$node_(38) setdest 1004.95 991.94 9.37"
$ns_ at 60.0 "$node_(3) setdest 1284.97 1.65 9.81"
$ns_ at 60.0 "$node_(39) setdest 1.65 618.55 9.94"
$ns_ at 60.0 "$node_(40) setdest 1626.23 604.95 9.19"
$ns_ at 60.0 "$node_(41) setdest 1395.05 1037.43 9.87"
$ns_ at 60.0 "$node_(42) setdest 757.16 398.35 9.95"
$ns_ at 60.0 "$node_(43) setdest 743.68 1398.35 9.82"
$ns_ at 60.0 "$node_(44) setdest 1336.74 1398.35 9.73"
$ns_ at 60.0 "$node_(45) setdest 601.65 726.54 8.78"
$ns_ at 60.0 "$node_(46) setdest 1686.08 1004.95 9.14"
$ns_ at 60.0 "$node_(47) setdest 1693.29 1804.95 8.94"
$ns_ at 60.0 "$node_(48) setdest 909.85 1004.95 9.52"
```

```
$ns_ at 60.0 "$node_(4) setdest 1498.75 804.95 8.94"
$ns_ at 60.0 "$node_(49) setdest 404.95 85.95 9.85"
$ns_ at 60.0 "$node_(5) setdest 4.95 706.02 9.85"
$ns_ at 60.0 "$node_(6) setdest 1326.24 204.95 9.67"
$ns_ at 60.0 "$node_(7) setdest 671.27 1395.05 9.26"
$ns_ at 60.0 "$node_(8) setdest 474.66 1595.05 9.99"
```

Lampiran 3

File cbr-50-01-05

```
set udp_(0) [new Agent/UDP]
$udp_(0) set dst_addr_ 0xE000000
$ns_ attach-agent $node_(0) $udp_(0)
#
set cbr_(0) [new Application/Traffic/CBR]
$cbr_(0) set packetSize_ 256
$cbr_(0) set interval_ 0.50
$cbr_(0) set random_ 1

$cbr_(0) set maxpkts_ 1740
$cbr_(0) attach-agent $udp_(0)
$cbr_(0) set dst_ 0xE000000
$ns_ at 30.0 "$cbr_(0) start"

for {set i 45} {$i < 50} {incr i} {
   $ns_ at 0.0100000000 "$node_($i) aodv-join-group 0xE000000"
}
```

[Halaman ini sengaja dikosongkan]

Lampiran 4

File cbr-50-01-10

```
set udp_(0) [new Agent/UDP]
$udp_(0) set dst_addr_ 0xE000000
$ns_ attach-agent $node_(0) $udp_(0)
#
set cbr_(0) [new Application/Traffic/CBR]
$cbr_(0) set packetSize_ 256
$cbr_(0) set interval_ 0.50
$cbr_(0) set random_ 1

$cbr_(0) set maxpkts_ 1740
$cbr_(0) attach-agent $udp_(0)
$cbr_(0) set dst_ 0xE000000
$ns_ at 30.0 "$cbr_(0) start"

for {set i 40} {$i < 50} {incr i} {
   $ns_ at 0.0100000000 "$node_($i) aodv-join-group 0xE000000"
}
```

[Halaman ini sengaja dikosongkan]

Lampiran 5

File cbr-50-01-30

```
set udp_(0) [new Agent/UDP]
$udp_(0) set dst_addr_ 0xE000000
$ns_ attach-agent $node_(0) $udp_(0)
#
set cbr_(0) [new Application/Traffic/CBR]
$cbr_(0) set packetSize_ 256
$cbr_(0) set interval_ 0.50
$cbr_(0) set random_ 1

$cbr_(0) set maxpkts_ 1740
$cbr_(0) attach-agent $udp_(0)
$cbr_(0) set dst_ 0xE000000
$ns_ at 30.0 "$cbr_(0) start"

for {set i 20} {$i < 50} {incr i} {
   $ns_ at 0.0100000000 "$node_($i) aodv-join-group 0xE000000"
}
```

[Halaman ini sengaja dikosongkan]

Lampiran 6

File cbr-50-01-50

```
set udp_(0) [new Agent/UDP]
$udp_(0) set dst_addr_ 0xE000000
$ns_ attach-agent $node_(0) $udp_(0)
#
set cbr_(0) [new Application/Traffic/CBR]
$cbr_(0) set packetSize_ 256
$cbr_(0) set interval_ 0.50
$cbr_(0) set random_ 1

$cbr_(0) set maxpkts_ 1740
$cbr_(0) attach-agent $udp_(0)
$cbr_(0) set dst_ 0xE000000
$ns_ at 30.0 "$cbr_(0) start"

for {set i 0} {$i < 50} {incr i} {
   $ns_ at 0.0100000000 "$node_($i) aodv-join-group 0xE000000"
}
```

[Halaman ini sengaja dikosongkan]

Lampiran 7

File cbr-50-02-02

```
for {set i 0} {$i < 2} {incr i} {
   set udp_($i) [new Agent/UDP]
   $udp_($i) set dst_addr_ 0xE000000
   $ns_ attach-agent $node_($i) $udp_($i)
#
   set cbr_($i) [new Application/Traffic/CBR]
   $cbr_($i) set packetSize_ 256
   $cbr_($i) set interval_ 0.50
   $cbr_($i) set random_ 1
   # send enough packets to keep simulation nearly busy: 2 packets
   # a second, starting at 30, stopping at 899: 2*870 = 1740
   $cbr_($i) set maxpkts_ 1740
   $cbr_($i) attach-agent $udp_($i)
   $cbr_($i) set dst_ 0xE000000
   $ns_ at 30.0 "$cbr_($i) start"
}

for {set i 48} {$i < 50} {incr i} {
   $ns_ at 0.0100000000 "$node_($i) aodv-join-group 0xE000000"
}
```

[Halaman ini sengaja dikosongkan]

Lampiran 8

File cbr-50-02-05

```
for {set i 0} {$i < 2} {incr i} {
   set udp_($i) [new Agent/UDP]
   $udp_($i) set dst_addr_ 0xE000000
   $ns_ attach-agent $node_($i) $udp_($i)
#
   set cbr_($i) [new Application/Traffic/CBR]
   $cbr_($i) set packetSize_ 256
   $cbr_($i) set interval_ 0.50
   $cbr_($i) set random_ 1
   # send enough packets to keep simulation nearly busy: 2 packets
   # a second, starting at 30, stopping at 899: 2*870 = 1740
   $cbr_($i) set maxpkts_ 1740
   $cbr_($i) attach-agent $udp_($i)
   $cbr_($i) set dst_ 0xE000000
   $ns_ at 30.0 "$cbr_($i) start"
}

for {set i 45} {$i < 50} {incr i} {
   $ns_ at 0.0100000000 "$node_($i) aodv-join-group 0xE000000"
}
```

[Halaman ini sengaja dikosongkan]

Lampiran 9

File cbr-50-02-10

```
for {set i 0} {$i < 2} {incr i} {
   set udp_($i) [new Agent/UDP]
   $udp_($i) set dst_addr_ 0xE000000
   $ns_ attach-agent $node_($i) $udp_($i)
#
   set cbr_($i) [new Application/Traffic/CBR]
   $cbr_($i) set packetSize_ 256
   $cbr_($i) set interval_ 0.50
   $cbr_($i) set random_ 1
   # send enough packets to keep simulation nearly busy: 2 packets
   # a second, starting at 30, stopping at 899: 2*870 = 1740
   $cbr_($i) set maxpkts_ 1740
   $cbr_($i) attach-agent $udp_($i)
   $cbr_($i) set dst_ 0xE000000
   $ns_ at 30.0 "$cbr_($i) start"
}

for {set i 40} {$i < 50} {incr i} {
   $ns_ at 0.0100000000 "$node_($i) aodv-join-group 0xE000000"
}
```

[Halaman ini sengaja dikosongkan]

Lampiran 10

File cbr-50-02-30

```
for {set i 0} {$i < 2} {incr i} {
   set udp_($i) [new Agent/UDP]
   $udp_($i) set dst_addr_ 0xE000000
   $ns_ attach-agent $node_($i) $udp_($i)
#
   set cbr_($i) [new Application/Traffic/CBR]
   $cbr_($i) set packetSize_ 256
   $cbr_($i) set interval_ 0.50
   $cbr_($i) set random_ 1
   # send enough packets to keep simulation nearly busy: 2 packets
   # a second, starting at 30, stopping at 899: 2*870 = 1740
   $cbr_($i) set maxpkts_ 1740
   $cbr_($i) attach-agent $udp_($i)
   $cbr_($i) set dst_ 0xE000000
   $ns_ at 30.0 "$cbr_($i) start"
}

for {set i 20} {$i < 50} {incr i} {
   $ns_ at 0.0100000000 "$node_($i) aodv-join-group 0xE000000"
}
```

[Halaman ini sengaja dikosongkan]

Lampiran 11

File cbr-50-02-50

```
for {set i 0} {$i < 2} {incr i} {
   set udp_($i) [new Agent/UDP]
   $udp_($i) set dst_addr_ 0xE000000
   $ns_ attach-agent $node_($i) $udp_($i)
#
   set cbr_($i) [new Application/Traffic/CBR]
   $cbr_($i) set packetSize_ 256
   $cbr_($i) set interval_ 0.50
   $cbr_($i) set random_ 1
   # send enough packets to keep simulation nearly busy: 2 packets
   # a second, starting at 30, stopping at 899: 2*870 = 1740
   $cbr_($i) set maxpkts_ 1740
   $cbr_($i) attach-agent $udp_($i)
   $cbr_($i) set dst_ 0xE000000
   $ns_ at 30.0 "$cbr_($i) start"
}

for {set i 0} {$i < 50} {incr i} {
   $ns_ at 0.0100000000 "$node_($i) aodv-join-group 0xE000000"
}
```

[Halaman ini sengaja dikosongkan]

Lampiran 12

File // ~/ns2.26/aodv/aodv_mcast.cc

```
void AODV::sendMACT(nsaddr_t dst, u_int8_t flags, u_int8_t
hop_count, nsaddr_t next_hop){
#ifdef DEBUG
    fprintf(stdout,"%s, node %i at %.9f\n", __FUNCTION__, index,
CURRENT_TIME);
#endif

    // as there is no MACT_U, next_hop must be a node address
    // and MACT must be sent out unicastly

    Packet *p = Packet::alloc();
    struct hdr_cmn *ch = HDR_CMN(p);
    struct hdr_ip *ih = HDR_IP(p);
    struct hdr_aodv_mact *mact = HDR_AODV_MACT(p);

    mact->mact_type = AODVTYPE_MACT;
    mact->mact_flags = flags;
    mact->mact_hop_count = hop_count;
    mact->mact_grp_dst = dst;
    mact->mact_src = index;
    mact->mact_src_seqno = seqno;

  //tambahan masbrenk
  iNode = (MobileNode *) (Node::get_node_by_address(index));
  mact->mact_x = iNode->X();
  mact->mact_y = iNode->Y();
  mact->mact_dx = iNode->dX();
  mact->mact_dy = iNode->dY();
  mact->mact_speed = iNode->speed();
  mact->mact_delay = CURRENT_TIME;

  FILE *fp;
  fp = fopen("send_mact.txt", "a");
  aodv_mt_entry *mt;

  fprintf(fp, "NODE: %i, Time: %f, src: %i, flag: %d, hopCt: %i,
grpDst: %i, seqno: %i, \n x: %.2f, y: %.2f, dx: %.2f, dy: %.2f,
speed: %.2f, sentTime: %f, enk: %f \n \n",
     index, CURRENT_TIME, mact->mact_src, mact->mact_flags, mact-
>mact_hop_count, mact->mact_grp_dst, mact->mact_src_seqno,
     mact->mact_x, mact->mact_y, mact->mact_dx, mact->mact_dy,
     mact->mact_speed, mact->mact_mact->mact_delay, mact->mact_enk
     );

  fclose(fp);

    ch->ptype() = PT_AODV;
    ch->size() = IP_HDR_LEN + mact->size();
    ch->iface() = -2;
    ch->error() = 0;
```

```
    ch->prev_hop_ = index;
    ch->direction() = hdr_cmn::DOWN;
    ch->next_hop_ = next_hop;
    ch->addr_type() = NS_AF_INET;

    ih->saddr() = index;
    ih->daddr() = next_hop;
    ih->sport() = RT_PORT;
    ih->dport() = RT_PORT;
    ih->ttl_ = 1;

    Scheduler::instance().schedule(target_, p, 0.01 *
Random::uniform());
}

void AODV::recvMACT(Packet *p){
#ifdef DEBUG
    fprintf(stdout,"%s, node %i at %.9f\n", __FUNCTION__, index,
CURRENT_TIME);
#endif

    struct hdr_aodv_mact *mact = HDR_AODV_MACT(p);

    switch (mact->mact_flags){
        case MACT_J: recvMACT_J(p); return;
        case MACT_P: recvMACT_P(p); return;
        case MACT_GL: recvMACT_GL(p); return;
        default: Packet::free(p); return;
    }
}

void AODV::recvMACT_P(Packet *p){
#ifdef DEBUG
    fprintf(stdout,"%s, node %i at %.9f\n", __FUNCTION__, index,
CURRENT_TIME);
#endif

struct hdr_aodv_mact *mact = HDR_AODV_MACT(p);
    nsaddr_t grp_dst = mact->mact_grp_dst;
    nsaddr_t prev_hop = mact->mact_src;

    //tambahan masbrenk
    x = mact->mact_x;
    y = mact->mact_y;
  dx = mact->mact_dx;
  dy = mact->mact_dy;
  speed = mact->mact_speed;
  delay = mact->mact_delay;

  //tambahan masbrenk
  iNode = (MobileNode *) (Node::get_node_by_address(index));
  mact->mact_x = iNode->X();
  mact->mact_y = iNode->Y();
  mact->mact_dx = iNode->dX();
  mact->mact_dy = iNode->dY();
  mact->mact_speed = iNode->speed();
  mact->mact_delay = CURRENT_TIME - delay;
```

```
  //tambahan masbrenk
  euclidxy = sqrt( (x * mact->mact_x) + (y * mact->mact_y) );
  euclidkec = sqrt(speed * mact->mact_speed);
  bobotjarak = 2;
  bobotkec = 1;
  bobotdelay = 3;
  wjarak = (bobotjarak / (bobotjarak + bobotkec + bobotdelay));
  wkec = (bobotkec / (bobotjarak + bobotkec + bobotdelay));
  wdelay = (bobotdelay / (bobotjarak + bobotkec + bobotdelay));
  wpjarak = wjarak * euclidxy;
  wpkec = wkec * euclidkec;
  wpdelay = wdelay * mact->mact_delay;
  enk = wpjarak * wpkec * wpdelay;

  if (enk == 0) return;
  else
    {
      enk == ((enk + enk1) / 2);
      enk1 == enk;
    }

  FILE *fp;
  fp = fopen("recvMACT_P.txt", "a");
  fprintf(fp, "Node: %i, recvTime: %f, flag: %d, hopCt: %i,
grpDst: %i, seqno: %i, \n \t x: %.2f, y: %.2f, dx: %.2f, dy: %.2f,
speed: %.2f, sentTime: %f, enk: %f \n menerima paket dari src: %i
... \n  x: %.2f, y: %.2f, dx: %.2f, dy: %.2f, speed: %.2f, delay:
%f, enk: %f \n \n",
     index, CURRENT_TIME, mact->mact_flags, mact->mact_hop_count,
mact->mact_grp_dst, mact->mact_src_seqno,
     x, y, dx, dy, speed, delay, enk,
     mact->mact_src, iNode->X(), iNode->Y(), iNode->dX(), iNode-
>dY(),
     iNode->speed(), (CURRENT_TIME - delay), enk
     );

  fclose(fp);

    Packet::free(p);

    // must be on tree and has valid link
    aodv_mt_entry *mt = mtable.mt_lookup(grp_dst);
    if (mt==NULL || mt->mt_node_status == NOT_ON_TREE) return;
    aodv_nh_entry *nh = mt->mt_nexthops.lookup(prev_hop);
    if (nh == NULL || nh->enabled_flag != NH_ENABLE)
        { if (nh) mt->mt_nexthops.remove(nh); return; }

    u_int8_t direction = nh->link_direction;
    mt->mt_nexthops.remove(nh);

    if (direction == NH_UPSTREAM) selectLeader(mt, INFINITY8);
    else mt_prune(mt->mt_dst);
}

void AODV::recvMACT_GL(Packet *p){
#ifdef DEBUG
    fprintf(stdout,"%s, node %i at %.9f\n", __FUNCTION__, index,
CURRENT_TIME);
```

```
#endif

    struct hdr_aodv_mact *mact = HDR_AODV_MACT(p);
    nsaddr_t grp_dst = mact->mact_grp_dst;
    nsaddr_t prev_hop = mact->mact_src;

    //tambahan masbrenk
    x = mact->mact_x;
    y = mact->mact_y;
  dx = mact->mact_dx;
  dy = mact->mact_dy;
  speed = mact->mact_speed;
  delay = mact->mact_delay;

  //tambahan masbrenk
  iNode = (MobileNode *) (Node::get_node_by_address(index));
  mact->mact_x = iNode->X();
  mact->mact_y = iNode->Y();
  mact->mact_dx = iNode->dX();
  mact->mact_dy = iNode->dY();
  mact->mact_speed = iNode->speed();
  mact->mact_delay = CURRENT_TIME - delay;

  //tambahan masbrenk
  euclidxy = sqrt( (x * mact->mact_x) + (y * mact->mact_y) );
  euclidkec = sqrt(speed * mact->mact_speed);
  bobotjarak = 2;
  bobotkec = 1;
  bobotdelay = 3;
  wjarak = (bobotjarak / (bobotjarak + bobotkec + bobotdelay));
  wkec = (bobotkec / (bobotjarak + bobotkec + bobotdelay));
  wdelay = (bobotdelay / (bobotjarak + bobotkec + bobotdelay));
  wpjarak = wjarak * euclidxy;
  wpkec = wkec * euclidkec;
  wpdelay = wdelay * mact->mact_delay;
  enk = wpjarak * wpkec * wpdelay;

  if (enk == 0) return;
  else
    {
      enk == ((enk + enk1) / 2);
      enk1 == enk;
    }

  FILE *fp;
  fp = fopen("recvMACT_GL.txt", "a");
  fprintf(fp, "Node: %i, recvTime: %f, flag: %d, hopCt: %i,
grpDst: %i, seqno: %i, \n \t x: %.2f, y: %.2f, dx: %.2f, dy: %.2f,
speed: %.2f, sentTime: %f, enk: %f \n menerima paket dari src: %i
... \n  x: %.2f, y: %.2f, dx: %.2f, dy: %.2f, speed: %.2f, delay:
%f, enk: %f \n \n",
     index, CURRENT_TIME, mact->mact_flags, mact->mact_hop_count,
mact->mact_grp_dst, mact->mact_src_seqno,
     x, y, dx, dy, speed, delay, enk,
     mact->mact_src, iNode->X(), iNode->Y(), iNode->dX(), iNode-
>dY(),
     iNode->speed(), (CURRENT_TIME - delay), enk
     );
```

```
  fclose(fp);

    Packet::free(p);

    // must be on tree and has valid link
    aodv_mt_entry *mt = mtable.mt_lookup(grp_dst);
    if (mt==NULL || mt->mt_node_status == NOT_ON_TREE || enk >=
BAD_LINK_LIFETIME){
       sendMACT(grp_dst, MACT_P, 0, prev_hop);
        return;
    }
    aodv_nh_entry *nh = mt->mt_nexthops.lookup(prev_hop);
    if (nh == NULL || nh->enabled_flag != NH_ENABLE || enk >=
BAD_LINK_LIFETIME){
        sendMACT(grp_dst, MACT_P, 0, prev_hop);
        if (nh) mt->mt_nexthops.remove(nh);
        return;
    }

    if (nh->link_direction == NH_DOWNSTREAM){
        //printf("WARNING: at %.9f in %s, node %d receives MACT_GL
from donwstream %d\n",
        //            CURRENT_TIME, __FUNCTION__, index, nh-
>next_hop);
        sendMGRPH_U(mt, INFINITY8);
    }
    else {
        // change the upstream direction to downstream and
        // select leader
        nh->link_direction = NH_DOWNSTREAM;
        selectLeader(mt, nh->next_hop);
    }
}

void AODV::recvMACT_J(Packet *p){
#ifdef DEBUG
    fprintf(stdout,"%s at node %i at %.9f\n", __FUNCTION__, index,
CURRENT_TIME);
#endif

    struct hdr_aodv_mact *mact = HDR_AODV_MACT(p);
    aodv_mt_entry *mt = mtable.mt_lookup(mact->mact_grp_dst);
    if (mt == NULL) mt = mtable.mt_add(mact->mact_grp_dst);
    nsaddr_t prev_hop = mact->mact_src;
    u_int8_t hop_count = mact->mact_hop_count - 1;

    //tambahan masbrenk
    x = mact->mact_x;
    y = mact->mact_y;
  dx = mact->mact_dx;
  dy = mact->mact_dy;
  speed = mact->mact_speed;
  delay = mact->mact_delay;

  //tambahan masbrenk
  iNode = (MobileNode *) (Node::get_node_by_address(index));
  mact->mact_x = iNode->X();
```

```
  mact->mact_y = iNode->Y();
  mact->mact_dx = iNode->dX();
  mact->mact_dy = iNode->dY();
  mact->mact_speed = iNode->speed();
  mact->mact_delay = CURRENT_TIME - delay;

  //tambahan masbrenk
  euclidxy = sqrt( (x * mact->mact_x) + (y * mact->mact_y) );
  euclidkec = sqrt(speed * mact->mact_speed);
  bobotjarak = 2;
  bobotkec = 1;
  bobotdelay = 3;
  wjarak = (bobotjarak / (bobotjarak + bobotkec + bobotdelay));
  wkec = (bobotkec / (bobotjarak + bobotkec + bobotdelay));
  wdelay = (bobotdelay / (bobotjarak + bobotkec + bobotdelay));
  wpjarak = wjarak * euclidxy;
  wpkec = wkec * euclidkec;
  wpdelay = wdelay * mact->mact_delay;
  enk = wpjarak * wpkec * wpdelay;

  if (enk == 0) return;
  else
    {
      enk == ((enk + enk1) / 2);
      enk1 == enk;
    }

  FILE *fp;
  fp = fopen("recvMACT_J.txt", "a");
  fprintf(fp, "Node: %i, recvTime: %f, flag: %d, hopCt: %i,
grpDst: %i, seqno: %i, \n \t x: %.2f, y: %.2f, dx: %.2f, dy: %.2f,
speed: %.2f, sentTime: %f, enk: %f \n # %f, %f, %f, %f, %f, %f, %f
# \n menerima paket dari src: %i ... \n  x: %.2f, y: %.2f, dx:
%.2f, dy: %.2f, speed: %.2f, delay: %f, enk: %f \n \n",
    index, CURRENT_TIME, mact->mact_flags, mact->mact_hop_count,
mact->mact_grp_dst, mact->mact_src_seqno,
    x, y, dx, dy, speed, delay, enk,
    euclidxy, euclidkec, euclidperc, wpjarak, wpkec, wpperc,
wpdelay,
    mact->mact_src, iNode->X(), iNode->Y(), iNode->dX(), iNode-
>dY(),
    iNode->speed(), (CURRENT_TIME - delay), enk
    );
  fclose(fp);

  Packet::free(p);

   // a tree member is reached
   if (mt->mt_node_status != NOT_ON_TREE){
       if (mt->mt_grp_leader_addr == INFINITY8 || enk >=
BAD_LINK_LIFETIME){
           //printf("******WARNING: %.9f in %s, node %d send
MACT_P to previous hop %d because its leader info is INIFINITY\n",
           //    CURRENT_TIME, __FUNCTION__, index, prev_hop);
           sendMACT(mt->mt_dst, MACT_P, 0, prev_hop);
           return;
       }
```

```
        if (hop_count != mt->mt_hops_grp_leader) sendMGRPH_U(mt,
prev_hop);

        aodv_nh_entry *nh = mt->mt_nexthops.lookup(prev_hop);
        if (nh && nh->enabled_flag == NH_ENABLE && nh-
>link_direction == NH_UPSTREAM && enk >= BAD_LINK_LIFETIME){
            //printf("******WARNING: %.9f in %s, node %d send
RREQ_J because its previous upstream node %d becomes
downstream\n",
            //    CURRENT_TIME, __FUNCTION__, index, prev_hop);
            nh->link_direction = NH_DOWNSTREAM;
            mt->mt_flags = MTF_IN_REPAIR;
            sendMRQ(mt, RREQ_J);
            return;
        }

        if (nh) mt->mt_nexthops.remove(nh);
        nh = new aodv_nh_entry(prev_hop);
        mt->mt_nexthops.add(nh);
        nh->link_direction = NH_DOWNSTREAM;
        nh->enabled_flag = NH_ENABLE;

        return;
    }

    // a not-on-tree node is reached
    // a not-on-tree node has no any valid link
    if (mt->mt_rep_timeout <= CURRENT_TIME || enk >=
BAD_LINK_LIFETIME){
        //printf("******WARNING: %.9f in %s, node %d send MACT_P
to previous hop %d, because it is a NOT_ON_TREE node and has no
up-to-date reply cache\n",
        //       CURRENT_TIME, __FUNCTION__, index, prev_hop);
        sendMACT(mt->mt_dst, MACT_P, 0, prev_hop);
        return;
    }
    if (prev_hop == mt->mt_rep_selected_upstream || enk >=
BAD_LINK_LIFETIME){
        //printf("******WARNING: %.9f in %s, node %d send MACT_P
to previous hop %d, because its recorded upstream is the same as
previous hop\n",
        //       CURRENT_TIME, __FUNCTION__, index, prev_hop);
        sendMACT(mt->mt_dst, MACT_P, 0, prev_hop);
        clearMRpyState(mt);
        return;
    }
    // recorded grp leader must not be the node itself,
    // because we do the check when recording the reply

    // has valid reply, the not-on-tree node become a tree member
    mt->mt_flags = MTF_UP;
    mt->mt_node_status = ON_TREE;
    mt->mt_grp_leader_addr = mt->mt_rep_grp_leader_addr;
    mt->mt_seqno = mt->mt_rep_seqno;
    mt->mt_hops_grp_leader = mt->mt_rep_hops_grp_leader;

    aodv_nh_entry *nh_d = new aodv_nh_entry(prev_hop);
    mt->mt_nexthops.add(nh_d);
```

```
    nh_d->link_direction = NH_DOWNSTREAM;
    nh_d->enabled_flag = NH_ENABLE;

    aodv_nh_entry *nh_u = new aodv_nh_entry(mt-
>mt_rep_selected_upstream);
    mt->mt_nexthops.add(nh_u);
    nh_u->link_direction = NH_UPSTREAM;
    nh_u->enabled_flag = NH_ENABLE;

    clearMRpyState(mt);
    clearMReqState(mt);

    if (hop_count != mt->mt_hops_grp_leader) sendMGRPH_U(mt,
prev_hop);
     //tambahan masbrenk
     //if  (enk <= BAD_LINK_LIFETIME)
       sendMACT(mt->mt_dst, MACT_J, mt->mt_hops_grp_leader, nh_u-
>next_hop);
     //else
     //  sendMACT(mt->mt_dst, MACT_P, 0, prev_hop);;
}
```

BIODATA PENULIS

Nurdiansyah Rezkinanda, biasa disebut *masbrenk* atau *maspipis* lahir pada 23 Pebruari 1992 di Malang, Jawa Timur. Penulis lulus Strata-1 pada tahun 2013 di STIKI – Malang. Mulai tahun 2010, aktif sebagai tenaga panggilan teknisi jaringan komputer. Kemudian mendirikan usaha dibidang jaringan komputer dan *software house* tahun 2011 yang berjalan sampai sekarang, sekaligus menjadi guru bantu di SMAN 1 Turen, Kab.Malang. Awal tahun 2013 penulis juga diperbantukan sebagai guru honorer di SMK Al-Munawwariyah Bululawang, Kab.Malang. Kemudian melanjutkan pendidikan jenjang Strata-2 di Program Pascasarjana Teknik Informatika ITS pada tahun 2014. Penulis dapat dihubungi melalui email *maspipis@lintaslangit.link*.

[Halaman ini sengaja dikosongkan]