TESIS-KI142502

PERBAIKAN KINERJA PROTOKOL ROUTING THE-DIFFERENT NEIGHBOUR HISTORY SPRAY AND WAIT(DNH-SAW) BERDASARKAN PERFORMA NODE SERTA DESTINATION MESSAGE HISTORY PADA LINGKUNGAN JARINGAN DELAY TOLERANT NETWORKS

TRI RAMADANI
NRP. 5114201011

DOSEN PEMBIMBING
Waskitho Wibisono, S.Kom, M.Eng, Ph.D

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

TESIS-KI142502

IMPROVING THE-DIFFERENT NEIGHBOUR HISTORY SPRAY AND WAIT (DNH-SAW) ROUTING PROTOCOL WITH NODE PERFORMANCES AND DESTINATION MESSAGE HISTORY IN DELAY TOLERANT NETWORKS

TRI RAMADANI
NRP. 5114201011

SUPERVISOR
Waskitho Wibisono, S.Kom, M.Eng, Ph.D

MASTER PROGRAM
THE EXPERTISE FIELD OF NET CENTRIC COMPUTING
DEPARTEMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2016

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

oleh:
TRI RAMADHANI
Nrp. 5114201011

Dengan judul :
PERBAIKAN KINERJA PROTOKOL ROUTING THE-DIFFERENT NEIGHBOUR HISTORY SPRAY AND WAIT(DNH-SaW) BERDASARKAN PERFORMANSI NODE SERTA DESTINATION MESSAGE HISTORY PADA LINGKUNGAN JARINGAN DELAY TOLERANT NETWORKS

Tanggal Ujian : 29-6-2016
Periode Wisuda : 2015 Genap

Disetujui oleh:

Waskitho Wibisono, S.Kom, M.Eng, Ph.D
NIP. 197410222000031001
(Pembimbing 1)

Royyana Muslim I, S.Kom, M.Kom, Ph.D
NIP. 197708242006041001
(Penguji 1)

Dr.Eng. Radityo Anggoro, S.Kom, M.Sc
NIP. 1984101620081210002
(Penguji 2)

Baskoro Adi Pratomo, S.Kom, M.Kom
NIP. 198702182014041001
(Penguji 3)

Direktur Program Pasca Sarjana,

Prof.Ir.Djauhar Manfaat, M.Sc., Ph.D.
NIP. 196012021987011001

PERBAIKAN KINERJA PROTOKOL ROUTING THE-DIFFERENT NEIGHBOUR HISTORY SPRAY AND WAIT(DNH-SaW) BERDASARKAN PERFORMA NODE SERTA DESTINATION MESSAGE HISTORY PADA LINGKUNGAN JARINGAN DELAY TOLERANT NETWORKS

Nama Mahasiswa : Tri Ramadani

NRP : 5114201011

Pembimbing : Waskitho Wibisono, S.Kom, M.Eng, Ph.D

ABSTRAK

Jaringan Delay Tolerant Network(DTN) dikembangkan berdasarkan kondisi jaringan yang tidak menitikberatkan kepada *end-to-end connection* seperti jaringan pada umumnya. Setiap *node* pada lingkungan jaringan DTN berfungsi sebagai *router* dan memiliki *storage* yang digunakan untuk menyimpan pesan. Protokol *routing* pada lingkungan DTN lebih menekankan kepada seberapa besar pesan yang dikirimkan dari *source node* sampai kepada *destination node*. Protokol *routing epidemic* bekerja dengan melakukan *flooding* pesan pada jaringan, dengan tujuan agar pesan cepat sampai pada *destination node* tanpa melihat aspek *overhead ratio* dan *delay* yang dihasilkan. Protokol *routing Spray and Wait(SaW)* memperbaiki kinerja dari protokol *epidemic* dengan membagi dua fase yaitu fase *spray* dan *wait*. Protokol *SaW* mengurangi *overhead ratio* yang dihasilkan dari protokol *routing epidemic*. Protokol *The Different-Neighbour History Spray and Wait(DNH-SaW)* melakukan modifikasi pada protokol *routing* SaW dengan menambahkan performa histori *node* yang ditemui dengan tujuan untuk mengatur jumlah banyaknya salinan pesan yang diberikan kepada node relay untuk diteruskan kepada *destination node*.

Pada protokol *routing DNH-SaW* penentuan *forward strategy message* hanya terfokus pada jumlah node yang ditemui, akan tetapi mengabaikan faktor seberapa banyak frekuensi pertemuan di dalam histori serta kecepatan node. Dengan memperhatikan frekuensi pertemuan dan kecepatan node, memungkinkan fase *spray* berubah dan masuk ke tahap fase *wait*.

Hasil penelitian menunjukkan bahwa pengaruh terhadap performa node histori, frekuensi dan kecepatan dapat meningkatkan *delivery ratio* pengiriman pesan dan penurunan terhadap *latency* pada protokol DNHSaWNew, tetapi memberikan efek yang jelek terhadap peningkatan *overhead ratio* yang dihasilkan.

Kata kunci: Delay Tolerant Networks, Spray and Wait, DNH-SaW, DNHSaWNew, delivery ratio, overhead ratio, latency.

[Halaman ini sengaja dikosongkan]

IMPROVING THE-DIFFERENT NEIGHBOUR HISTORY SPRAY AND WAIT (DNH-SaW) ROUTING PROTOKOL WITH NODE PERFORMANCES AND DESTINATION MESSAGE HISTORY IN DELAY TOLERANT NETWORKS

Student Name	: Tri Ramadani
NRP	: 5114201011
Supervisor	: Waskitho Wibisono, S.Kom, M.Eng, Ph.D

ABSTRACT

Network Delay Tolerant Network (DTN) is developed based on network conditions that do not prioritize the end-to-end connection such as a network in general. Each node on the network environment DTN serves as a router and have storage that is used to store messages. DTN routing protocols on the environment it is more about how large a message sent from the source node to the destination node. Epidemic routing protocol works by flooding a message on the network, with the aim that the instant messages to the destination node without seeing overhead aspect ratio and the resulting delay. The routing protocol Spray and Wait (SAW) improve the performance of the protocol epidemic by dividing the two phases which spray and wait. Saw protocol reduces the overhead ratio resulting from routing protocol epidemic. The protocol-Neighbour History Different Spray and Wait (DNH-SAW) make modifications in the routing protocol performance history SAW by adding nodes encountered with the aim of regulating the amount of the number of copies of the messages given to relay nodes to be forwarded to the destination node.

In the DNH-Saw routing protocols determine message forward strategy focused only on the number of nodes encountered, but ignore the factor of how much the frequency of meetings in history as well as the speed of the node. Having regard to the frequency of meetings and the speed of the node, enabling spray phase change and entered the stage of phase wait.

The results showed that the influence of the node performance history, frequency and speed of delivery can improve message delivery ratio and decreased the latency in protocols DNHSaWNew, but give bad effect to the increased overhead ratio is generated.

Keyword: Delay Tolerant Networks, DNH-SaW, DNHSaWNew, Delivery ratio, Overhead ratio, Latency.

[Halaman ini sengaja dikosongkan]

DAFTAR ISI

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Pengiriman pesan dikatakan berjalan baik apabila kondisi antara node sumber*(source node)* dengan node tujuan*(destination node)* mempunyai konektifitas jaringan yang stabil*(connection established)*. Parameter utama untuk menentukan suatu jaringan dikatakan *connection established* yaitu dilihat dari *end-to-end connections*. End-to-end *connections* digambarkan berupa proses pengiriman pesan dari *source node* kepada *destination node* tanpa mengalami kendala berupa *delay* sehingga tidak menyebabkan kerusakan pada pesan. *Delay Tolerant Network(DTN)* dikembangkan pada suatu jaringan yang mempunyai kondisi tidak adanya koneksifitas yang stabil antara *source node* dan *destination node*. Pengembangan jaringan DTN bertujuan mengirimkan data dari *source node* kepada *destination node* dengan lebih memperhatikan pesan yang diterima disebut *message-oriented*. Salah satu hal penting dalam pengiriman pesan agar pesan yang dikirimkan oleh *source node* sampai kepada *destination node* yaitu *protokol routing*.(Jain S.,dkk, 2004).

Protokol *Routing Epidemic* merupakan salah satu protokol *routing* yang dikembangkan pada lingkungan jaringan DTN. Protokol ini bekerja dengan cara melakukan *flooding* pesan antar node yang saling bertemu. Hal ini dilakukan terus menerus sampai pesan diterima oleh *destination node* atau *time-to-live(TTL)* dari pesan tersebut habis. Hasil yang diperoleh dari protokol *epidemic* yaitu tingkat *delivery ratio* yang tinggi dan *overhead ratio* yang besar. Tingkat *delivery ratio* yang tinggi disebabkan setiap node yang bertemu akan bertukar pesan tanpa mengetahui node yang membawa pesan akan atau pernah bertemu dengan node tujuan pesan, sehingga sangat dimungkinkan salah satu node tersebut bertemu dengan *destination* dan pesan terkirim dengan baik. *Overhead ratio* yang besar menjadi kelemahan dari protokol ini disebabkan masing – masing *node relay* mempunyai salinan pesan dan bisa menyebabkan kemacetan*(network congestion)* di dalam jaringan. (Spyropoulos T.,dkk, 2005; Pan D., 2012).

Protokol *Routing Spray and Wait(SaW)* dikembangkan untuk mengurangi *overhead ratio* yang diperoleh pada protokol *routing epidemic*. Konsep kerja protokol *SaW* dibagi menjadi dua fase yaitu fase *spray* dan fase *wait*. Fase *spray* mengambil kelebihan dari *protokol routing epidemic* yaitu dengan melakukan *flooding* terhadap jaringan. *Source node* akan memberikan salinan pesan*(L-message)* yang dimilikinya kepada node relay. Sedangkan fase *wait* mengambil kelebihan dari protokol *routing direct delivery* yaitu membawa sisa salinan pesan yang dimiliki sampai bertemu dengan *destination node*. Hasil yang diperoleh dari protokol SaW yaitu *delivery ratio relatif* baik, *overhead ratio* lebih kecil, dan *delay* lebih kecil apabila dibandingkan dengan protokol *routing epidemic*. (Spyropoulos T.,dkk, 2005; Vittawus P.,dkk, 2011).

Protokol *Routing Binary Spray and Wait(BSaW)* dikembangkan dari *spray and wait*, yaitu pada fase *spray* pesan dibagi menjadi *L/2* salinan dan dikirimkan ke *node relay*, sehingga diharapkan hasil *delivery ratio* pengiriman yang lebih baik, nilai *overhead ratio* dan *delay* lebih rendah daripada *SaW*. (Spyropoulos T.,dkk, 2005).

Pengembangan lain dari protokol *routing SaW* yaitu protokol *routing The Different-Neighbour History Spray and Wait(DNH-SaW)* yaitu dengan menambahkan nilai performa pada setiap node. Nilai performa diambil dari perhitungan histori pertemuan antar node. Apabila terdapat 2 node berupa *source node* dan *node relay* yang saling terhubung, maka proses yang terjadi yaitu *node relay* akan melakukan *update* terlebih dahulu tabel histori pertemuan dengan *source node*, kemudian dilanjutkan dengan jenis pesan yang akan dikirimkan oleh *source node* di dalam *memory buffer*. Apabila pesan yang akan dikirmkan oleh *source node* tidak ada di dalam *memory buffer node relay*, maka *node relay* akan memberikan tanda kepada *source node* agar mengirimkan pesan. *Source node* akan menghitung nilai performa yang dimiliki oleh *node relay*, lalu dikalikan dengan jumlah salinan pesan yang dimiliki oleh *source node*, sehingga bisa diketahui berapa jumlah salinan pesan yang diberikan kepada *node relay*. Hasil yang diperoleh dari protokol *routing DNH-SaW* apabila dibandingkan dengan protokol *routing epidemic* dan *BSaW* yaitu tingkat *delivery ratio* lebih besar,

overhead ratio lebih kecil tapi *delay* yang dimiliki lebih besar.(Vittawus P.,dkk, 2011).

Pada protokol *routing DNH-SaW* penentuan *forward strategy message* hanya terfokus pada histori *node* yang ditemui, akan tetapi mengabaikan faktor seberapa banyak frekuensi pertemuan *node* serta kecepatan *node*. Dengan memperhatikan frekuensi pertemuan *node* dan kecepatan *node*, memungkinkan fase *spray* pada protokol DNHSaW berubah dan masuk ke dalam tahap fase *wait*. Begitupula dengan keberadaan histori *node,* dengan *node* tujuan pesan tersebut. Jika suatu *node* pernah bertemu dengan *node* tujuan dari pesan, maka bisa dimungkinkan *node* tersebut akan bertemu kembali.

Dalam penelitian ini mengusulkan metode baru yaitu Perbaikan Kinerja Protokol Routng The Different Neighbour History Spray and Wait(DNH-SaW)B Berdasarkan Performa Node serta Destination Message History pada Lingkungan Jaringan Delay Tolerant Networks. Performa *node* yang diperhatikan dilhiat dari histori, frekuensi, dan kecepatan. Penelitian ini diharapkan untuk meningkatkan *delivery ratio*, mengurangi *overhead ratio*, dan *latency*.

1.2 Perumusan Masalah

Berdasarkan latar belakang yang sudah diuraikan sebelumnya, perumusan masalah yang terdapat pada penelitian ini antara lain:

1. Bagaimana meningkatkan performa *node* berdasarkan histori, frekuensi dan kecepatan saat proses fase spray?
2. Bagaimana menentukan pengambilan keputusan pendistribusian pesan berdasarkan *destination message history* pada proses *forward stategy*?
3. Bagaimanakah kinerja modifikasi DNH-SaW berdasarkan performa node?

1.3 Tujuan dan Manfaat Penelitian

Tujuan dari penelitian ini yaitu diharapkan dapat meningkatkan *delivery ratio* pengiriman pesan dengan melakukan modifikasi pada protokol *routing* DNH-SaW berdasarkan performa *node* serta *destination history message*.

Manfaat dari penelitian ini yaitu dapat memberikan kontribusi terhadap perkembangan protokol *routing* pada lingkungan jaringan DTN.

1.4 Batasan Masalah

Dalam penelitan ini, batasan masalah yang dibahas diuraikan sebagai berikut:

1. Dataset yang digunakan merupakan potongan peta kota Helsinki dari negara Finlandia.
2. Evaluasi performa yang diujikan pada metode baru yang diusulkan terdiri dari *delivery ratio*, *overhead ratio*, dan *delay*.
3. Implementasi pengujian metode baru yang diusulkan menggunakan simulator ONE (Opportunistic Network Enviroment).

1.5 Kontribusi

Kontribusi penelitian ini adalah mencari nilai performa *node* berdasarkan histori, frekuensi dan kecepatan serta menentukan pengambilan keputusan pendistribusian pesan berdasarkan *destination history message.*

BAB 2
DASAR TEORI DAN KAJIAN PUSTAKA

2.1 Delay Tolerant Networks

Konsep *Delay Tolerant Networks (DTN)* pertama kali diperkenalkan oleh Kevin Fall yaitu suatu arsitektur jaringan yang menantang *(challenged),* artinya jaringan dengan penuh masalah seperti waktu tunggu *(delay)* yang lama, koneksi yang sering terputus atau bahkan tidak ada sama sekali, dan tingkat kesalahan *(error)* yang tinggi (Kevin Fall, 2003). Contoh lain jaringan yang mempunyai sifat yang menantang yaitu:

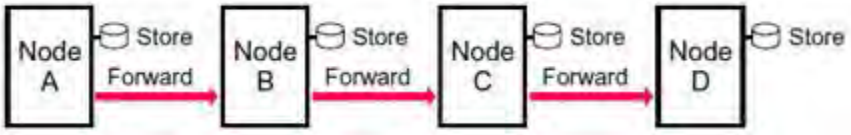
- Jaringan luar angkasa *(interplanetary network),* konsep jaringan yang memungkinkan akses internet di luar angkasa.
- *Military ad-hoc network*. Pasukan militer biasanya seringkali ditempatkan suatu daerah terpencil yang tidak berpenghuni dengan kondisi tidak adanya koneksi yang memadai. Konsep DTN dapat digunakan untuk membangun jaringan komputer dalam keadaan seperti ini.
- Jaringan *sensor/actuator* yang diterapkan oleh jaringan *Wireless Sensor Network (WSN)*.

Pada studi kasus pengiriman data *interplanetary internet* dari bumi ke sebuah kendaraan yang ada di luar angkasa, hampir mustahil koneksi *end-to-end* dengan TCP/IP bisa dibangun dengan baik, sebab *delay* yang tinggi. Agar data bisa dikirimkan, maka mekanisme pengirimannya dilakukan secara bertahap dari satu *node* ke *node* berikutnya, lalu kemudian masing – masing akan menyimpan data untuk sementara waktu. Apabila kondisi dimana terdapat *link* koneksi yang baik antar *node*, maka data yang sudah disimpan yang ada di dalam masing-masing *node* akan dikirimkan ke *node* berikutnya sampai bertemu dengan *node* tujuan.

2.1.1. Metode Store-Carry-Forward

Metode yang digunakan untuk mengirimkan pesan dari *node* sumber *(source node)* kepada node tujuan *(destination node)* pada jaringan DTN

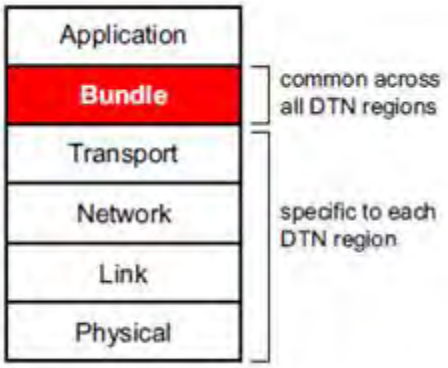
dinamakan dengan *store-carry-forward*. Cara kerja metode *store-carry-forward* pada jaringan DTN dengan menyimpan data yang diterima dari *node* sebelumnya ke dalam sebuah *buffer memory* sebelum diteruskan ke *node relay*. Hal ini dilakukan untuk mengantisipasi *delay* apabila *node relay* tidak dapat dijangkau atau *link* koneksi putus (Bijal Patel, dkk, 2014).



Gambar 2.1 Metode *Store-Carry and Forward*.(Patel B., 2014)

Gambar 2.1 menunjukkan proses pengiriman data dari *node* A ke *node* D. Saat melewati *node* B dan *node* C sebagai perantara, pesan disimpan terlebih dahulu sebelum dikirimkan apabila koneksi dengan *node* berikutnya telah siap. Metode *store-carry-forward* berbeda dengan proses pengiriman data pada TCP/IP. Pada TCP/IP, *node relay* hanya menerima data dan langsung meneruskan. Akibatnya jika koneksi putus disuatu tempat, maka data yang sedang dalam proses pengiriman akan hilang dan harus melakukan pengiriman ulang kembali.

Metode *store-carry-forward* memiliki konsekuensi setiap *node* harus memiliki media penyimpanan*(storage)* yang mencukupi. *Storage* digunakan untuk menyimpan data apabila koneksi dengan *node relay* belum tersedia. Semakin besar media penyimpan yang dimiliki oleh *node*, maka semakin besar *node* tersebut dapat membawa data yang akan dikirimkan kepada *node* berikutnya. Proses *store-carry-forward* pada jaringan DTN dilakukan pada sebuah layer tambahan yaitu layer *bundle*. Fungsi dari *layer bundle* untuk melakukan modifikasi terhadap paket data dengan fasilitas yang disediakan oleh DTN. Posisi *layer bundle* berada tepat dibawah layer aplikasi. Pada layer aplikasi paket data yang akan dikirimkan dipecah menjadi paket *bundle* kecil (Bijal Patel, dkk, 2014). Paket *bundle* ini yang akan dikirim ke *transport layer* yang ada di dibawahnya untuk diproses lebih lanjut. Posisi layer bundle dalam pengiriman data digambarkan pada gambar 2.2.



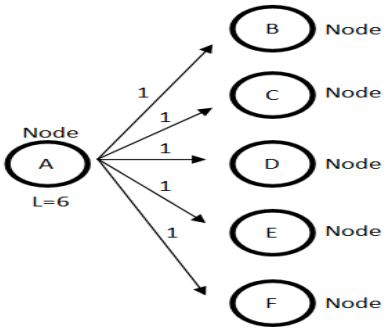
Gambar 2.2 *Layer budle* pada DTN(Warthman F., 2015)

2.1.2. Protokol Routing Spray and Wait

Cara kerja protokol *routing Spray and Wait(SaW)* dengan memanfaatkan keunggulan dari skema protokol *routing epidemic* dan *direct delivery*. Skema protokol *routing direct delivery* mengirimkan pesan ketika *source node* dan *node relay* saling bertemu dan berkomunikasi. Skema ini menjamin penghematan *resources*, baik itu *bandwith* maupun *buffer*, namun skema ini memiliki tingkat *delivery ratio* yang tinggi, *overhead ratio* yang dihasilkan sangat rendah, dan *delay* yang sangat besar. Skema protokol *routing epidemic* mengirimkan salinan pesan dari *source node* kepada *node relay* yang ditemuinya. Setiap *node relay* juga akan melakukan hal yang sama seperti yang dilakukan oleh *source node* sampai salinan pesan tersebut sampai kepada *destination node*. Dari skema protokol *routing epidemic* menghasilkan *overhead ratio* yang tinggi, *latency* yang rendah, dan *delivery ratio* pengiriman yang tinggi. Skema ini akan berjalan dengan baik apabila setiap node memiliki tempat penyimpanan*(buffer)* yang besar. Tapi kenyataanya setiap *node* memiliki tempat penyimpanan yang terbatas. Skema protokol *routing spray and wait* akan membuat sejumlah salinan pesan sejumlah yang sudah ditentukan*(L-copies)* seperti yang diilustrasikan pada gambar 2.3 skema protokol *routing spray and wait*.(Thrasyvoulus S, dkk, 2005).

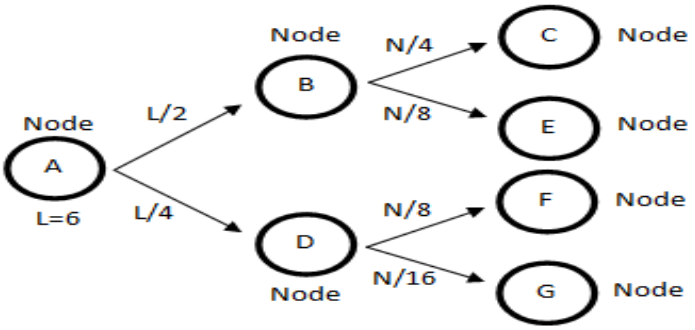
- *Fase spray*: *source node* akan menyimpan sejumlah salinan pesan*(L)* yang disiapkan untuk dikirimkan kepada *node relay*. *Source node* melakukan distribusi salinan pesan kepada *node relay* sebanyak jumlah banyaknya pesan yang dimiliki oleh *source node* dikurangi 1*(L-1)*.

- *Fase wait*: *source node* akan menyimpan dan membawa sisa salinan pesan yang dimiliki setelah fase *spray* hingga bertemu dengan *destination node*. Apabila fase *wait source node* bertemu dengan *node relay* lain maka sisa salinan yang dimiliki tidak ada akan diberikan kepada node tersebut.



Gambar 2.3 Skema Protokol Routing Spray and Wait.(Spyropoulos T., dkk, 2005)

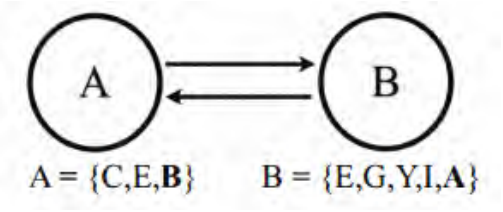
Masalah yang muncul dari skema protokol *routing spray and wait* yaitu apabila salinan pesan yang dimiliki oleh *source node* tidak sebanyak jumlah *node* yang ditemuinya, sehingga menyebabkan sisa salinan yang dimiliki oleh *source node* menjadi banyak. Ada metode lain yang dikembangkan dari *SaW* yang disebut dengan *Binary Spray and Wait(BSaW)*. Apabila *source node* bertemu dengan *node relay*, salinan pesan yang dimiliki oleh *source node* akan diberikan sebanyak *L/2*. Lalu *source node* menyimpan sisa dari salinan pesan yang dikirimkan kepada *node relay(L-(L/2))*. Proses ini dilakukan terus menerus sampai sisa salinan pesan yang dimiliki oleh *source node* sebanyak L=1 sehingga bisa langsung merubah fase menjadi fase *wait*.(Thrasyvoulus S, dkk, 2005). Ilustrasi skema pengiriman pesan pada metode *BSaW* terlihat pada gambar 2.4.



Gambar 2.4 Skema Protokol *Routing BSaW*. .(Spyropoulos T., dkk, 2005)

2.2 Protokol Routing The Different-Neighbour History Spray and Wait(DNH-SaW)

Protokol *Routing DNH-SaW* dikembangkan dengan tujuan untuk meningkatkan kinerja fase *spray* dari protokol *routing BSaW*, dengan cara melihat performa dari masing–masing *node*. Nilai performa dihasilkan dari jumlah histori pertemuan antar *node* dibagi dengan *complete set*. *Complete set* dihasilkan dari penjumlahan total keseluruhan histori yang dimiliki oleh *node* yang saling bertemu. Setelah nilai performa diketahui, maka bisa diketahui jumlah banyaknya salinan pesan *(spray copy)* yang diberikan kepada *node* selanjutnya, dengan mengalikan salinan pesan yang dimiliki dari node sumber.(Vittawus P., dkk, 2011).



Gambar 2.5 Ilustrasi Pertukaran Salinan Pesan Antar Node.(Pueksasri V., dkk, 2011)

Dari gambar 2.5 terlihat ilustrasi skema protokol *routing DNH-SaW*. Node A mempunyai informasi histori dengan *node* sebelumnya yang berada di dalam tabel histori yaitu C, E, dan B, sedangkan *node* B mempunyai informasi histori *node* dengan E, G, Y, I, dan A. Apabila *node* A bertemu dengan B, maka yang harus dilakukan terlebih dahulu yaitu melakukan *update* informasi tabel histori pertemuan dari masing-masing node. Node A mencatat pertemuan dengan node B, dan sebaliknya node B mencatat pertemuan dengan node A. Selanjutnya menghitung nilai performa *node* B dilihat dari node A (Vittawus P., dkk, 2011), dengan menggunakan rumus (2.1):

$$P = \frac{V}{S} \tag{2.1}$$

dimana:

P = performa *node* B dilihat dari *node* A

V = jumlah informansi node yang ada didalam tabel histori *node* B dan yang

belum pernah bertemu dengan *node* A, yaitu *node* G, Y, dan I.

S = *complete set* antara node A dan node B, yaitu node A, B, C, E, G, Y, I.

Setelah diperoleh nilai performa *node* B, selanjutnya menghitung jumlah banyaknya salinan pesan yang akan dikirimkan dari *node* A kepada node B (Vittawus P., dkk, 2011), dengan menggunakan rumus (2.2):

$$N2 = [P\ x\ N1] \tag{2.2}$$

dimana:

N2 = jumlah banyaknya salinan pesan yang dikirimkan dari *node* A ke B

P = performa *node* B dilihat dari *node* A

N1 = jumlah salinan pesan yang saat ini dimiliki oleh *node* A

Setelah salinan pesan dikirimkan dari *node* A ke B (Vittawus P., dkk, 2011), maka untuk mendapatkan sisa salinan pesan yang dimiliki oleh *node* A dengan menggunakan menggunakan rumus (2.3):

$$N1" = N1 - N2 \tag{2.3}$$

dimana:

N1" = sisa salinan pesan yang dimiliki oleh node A setelah melalui proses pengiriman pesan dari *node* A kepada *node* B

N1 = jumlah salinan pesan yang dimiliki oleh *node* A sebelum dikirimkan ke node B

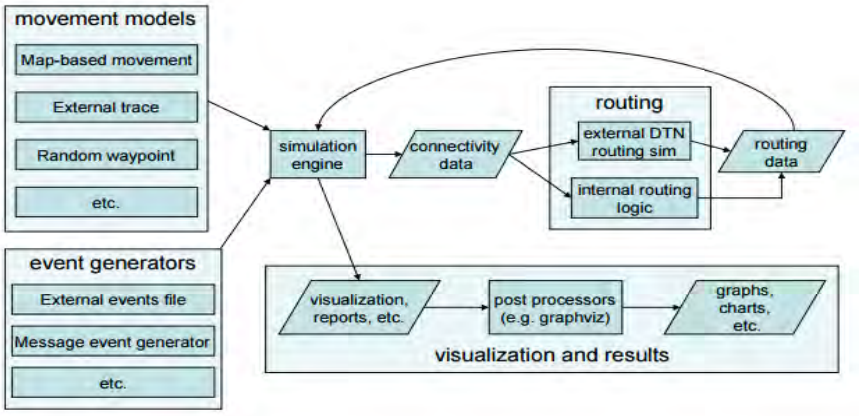
N2 = jumlah pesan yang sudah dikirimkan dari *node* A ke node B

Pada fase *spray source node* akan melakukan proses pengiriman salinan pesan kepada setiap *node* yang ditemui dengan menggunakan mekanisme perhitungan dari rumus yang sudah ditentukan, sampai menyisakan 1 salinan pesan. Kemudian *node* sumber akan berpindah fase ke dalam fase *wait* dan mengirimkan sisa salinan pesan dengan menggunakan protokol *routing direct delivery*.

2.3 Network Simulator ONE

ONE(Opportunistic Network Enviroment) adalah salah satu *tools* simulasi berbasis bahasa pemrograman java yang dibuat untuk melakukan

simulasi jaringan dalam lingkungan jaringan *delay tolerant network*. Di dalam simulator ONE terdapat beberapa paket yang saling terkait penggunaannya. Paket tersebut berisi modul baris program java. Fungsi utama dari modul paket yang ada didalam simulator ONE yaitu *node movement, inter-node contacts, routing* dan *message handling.* Ini terlihat seperti pada gambar 2.6.

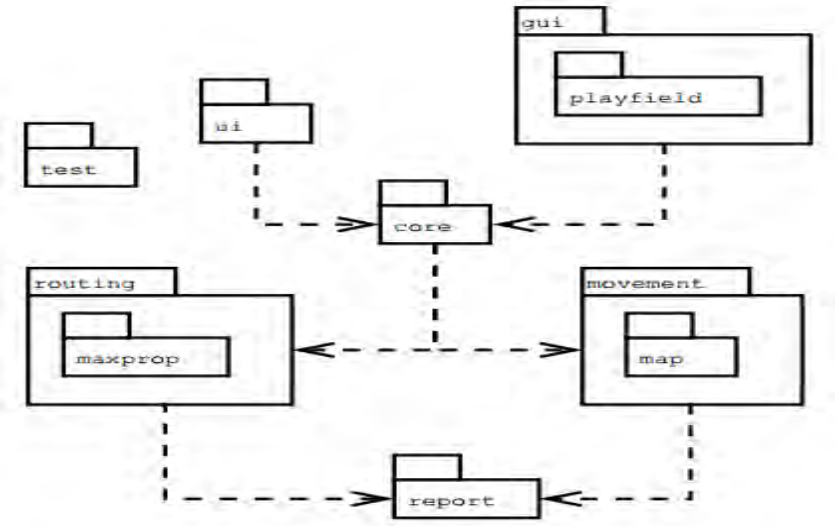


Gambar 2.6 Pengenalan Lingkungan Simulator ONE(Keranen A., dkk, 2009)

Pada gambar 2.6 terlihat blok paket untuk melakukan proses simulasi modul yang digunakan akan saling berelasi antar modul satu dengan modul lainnya. Modul *movement models, event generators, routing, visuzalization and results* akan saling bersinergi dan dieksekusi dalam modul *simulation engine*. Untuk menggunakan modul tersebut, maka harus dilakukan inisialisasi terlebih dahulu. Hal ini dilakukan agar saat simulasi berlangsung beberapa paket yang sudah di-*set* mudah dipanggil dan digunakan. Inisialisasi ini dilakukan di dalam *file* konfigurasi.

2.3.1. Hirarkhi Network Simulator ONE

ONE simulator dijalankan dari dua mode yang berbeda yaitu melalui *GUI(Graphical User Interface)* dan *batch mode*. Mode *GUI* digunakan untuk melakukan *testing, debugging,* dan demonstrasi secara visual, sedangkan *mode batch* digunakan untuk menjalankan simulasi dalam jumlah yang besar.



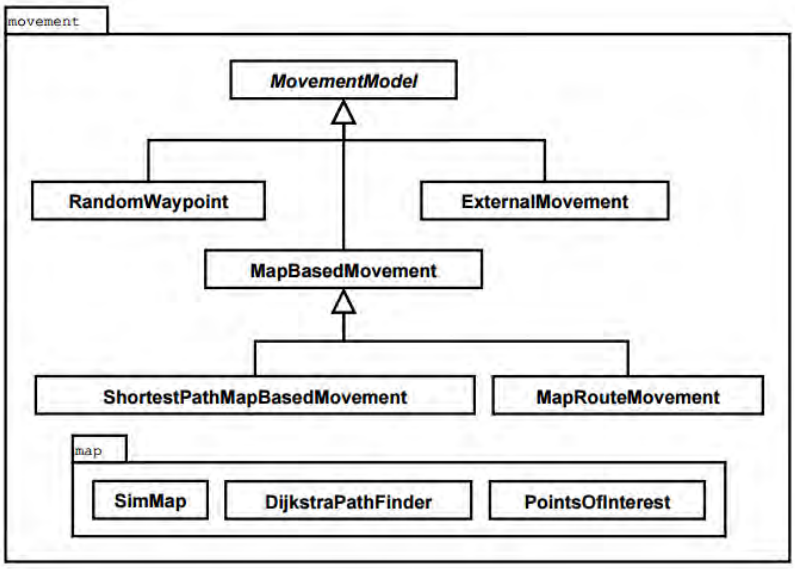
Gambar 2.7. Modul ONE Simulator

Pada gambar 4.1 mode menjalankan simulator ONE berada di dalam paket core yang dihasilkan dari dua parameter yang berbeda yaitu *GUI* dan *batch mode*. Di dalam paket core pada simulator ONE akan dipresentasikan kebutuhan yang digunakan untuk menjalankan sebuah simulasi. Saat simulasi dijalankan, masing-masing paket akan saling berelasi untuk mengambil file kelas yang dibutuhkan seperti protokol *routing spray and wait* berada di dalam paket *routing*. Selain itu, untuk pergerakan masing-masing *node* dari suatu grup diambil dari kelas *movement* tertentu yang berada di dalam paket *movement*. Selama proses simulasi berlangsung, modul *routing* dan *movement* menghasilkan suatu keluaran yang dihasilkan oleh kelas *report* tertentu yang berada didalam paket *report*. Keluaran ini nantinya yang akan dianalisis sehingga mendapatkan kesimpulan yang dapat digunakan untuk menguatkan materi pada penelitian ini.

2.3.2. Movement Models

Pergerakan *node* yang didalam simulator ONE diperoleh dari modul paket *movement models*. Ada dua metode pergerakan yang ada di dalam simulator ONE yaitu dengan menggunakan pergerakan sintetis atau dengan menggunakan pergerakan secara *random*. Ada berbagai macam pergerakan node yang disajikan oleh simulator ONE seperti *RandomWaypoint, ShortestPathMapBasedMovement,*

StationaryMovement, BusMovement, CarMovement, dan sebagainya. Sedangkan interkoneksi antar node dilakukan berdasarkan lokasi *node* tersebut, jarak pancaran komunikasi serta *bit-rate* yang dihasilkan oleh masing-masing node.



Gambar 2.8. Modul Movement Model ONE Simulator

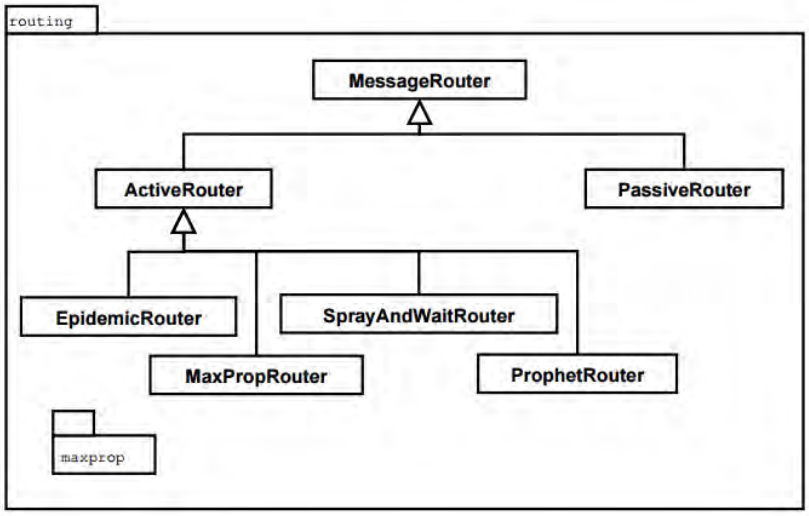
Pada gambar 2.8 terlihat seluruh model *movement* yang digunakan untuk menentukan pergerakan *node* diturunkan dari kelas *MovementModel*. Kelas *MovementModel* merupakan kelas *interface* yang bisa diimplementasikan pada pergerakan yang berbeda, misalnya: *RandomWaypoint* memberikan keluaran pergerakan arah yang zig-zag yang di-*generate* oleh modul kelas *RandomWaypoint*. *ShortestPathMapBasedMovement* mempunyai pergerakan yang memperhitungkan cara tercepat untuk menuju ke titik lokasi tujuan dengan menggunakan algoritma *Djikstra* yang ada didalam kelas *DjikstraPathFinder*.

POI(Point of Interest) digunakan untuk membuat suatu *node* bisa difokuskan untuk melakukan pergerakan pada area yang sudah ditentukan. Konfigurasi *POI* biasanya akan disertakan file peta .wkt yang dijadikan sebagai rujukan suuatu *node* untuk melakukan pergerakan di peta tersebut.

Selain itu, *movement model* yang tidak ada di dalam modul simulator dapat dibuat dengan ekstensi .wkt dan dinisialisasi dengan menggunakan kelas *External Movement*.

2.3.3. Protokol Routing

Protokol Routing pada ONE simulator digunakan untuk mengatur arah pesan yang akan dikirimkan dari suatu *node* ke *node* lainnya. Protokol routing pada lingkungan jaringan DTN berbeda dengan protokol *routing* biasanya. Mekanisme protokol *routing* yang ada didalam lingkungan jaringan DTN akan memperhitungkan terhadap tidak stabilnya interkoneksi antar *node*.



Gambar 2.9. Modul *Package* Protokol *Routing ONE Simulator*

Pada gambar 2.9 terlihat setiap protokol *routing* akan menurunkan kelas *MessageRouter*. Kelas *MessageRouter* akan menyimpan informasi tentang pesan yang datang maupun yang diteruskan ke node lain yang tersimpan didalam *memori buffer* masing-masing node. Pada saat suatu node yang saling bertemu ingin melakukan pengiriman pesan, maka kelas *MessageRouter* akan berelasi dengan memanggil metode *start transfer* dan *receiverMessage*. Selain itu, kelas *MessageRouter* akan mengecek pesan yang dikirimkan sudah terkirim atau ditolak dengan memanggil metode transferDone() dan transferAborted().

Ada beberapa penelitian yang menggunakan *ONE simulator* sebagai salah satu alat untuk memecahkan kasus yang berkaitan dengan jaringan DTN, antara lain:

- Penelitian (Spyropoulos T., dkk, 2005) melakukan simulasi dengan mengembangkan protokol *routing Spray and Wait(SaW)* yang dikembangkan dari protokol *routing direct delivery dan epidemic.*

- Penelitian (Burgess J., dkk, 2006) melakukan simulasi dengan mengembangkan protokol *routing MaxProp* berdasarkan *priority packet*.
- Penelitan (Wang G., dkk, 2010) melakukan simulasi perbaikan kinerja protokol *routing SaW* pada fase *spray* dengan memperhatikan *Quality of Node(QoN)* yang dihasilkan dari masing – masing *node* yang saling bertemu.
- Penelitan (Prueksasri V., dkk, 2011) melakukan simulasi perbaikan kinerja protokol *routing SaW* pada fase *spray* dengan memperhatikan *history* pertemuan *node* dan menghitung nilai performansi *node* yang saling bertemu.

[Halaman ini sengaja dikosongkan]

BAB 3
METODE PENELITIAN

3.1 Rancangan Penelitian

Pada penelitian ini, untuk melakukan implementasi terhadap metode baru yang diusulkan, peta yang dipilih sebagai tempat pengujian yaitu kota Helsinki dengan ukuran 4500 x 3400 meter. Pemilihan kota Helsinki tidak disebabkan oleh karakteristik dari peta tersebut yang berbeda dengan tempat lain, tetapi sebagai perbandingan hasil metode baru yang diusulkan dengan metode sebelumnya yang juga menggunakan peta yang sama. Terdapat 3 jenis kelompok *node* di area tersebut yaitu *pedestrian*, *cars,* dan *tram*. Untuk pergerakan *node pedestrian* dan *cars* menggunakan *Shortest Path Map-Based*, sedangkan *tram* menggunakan *Routed Map-Based*. Ukuran pesan yang beredar di dalam jaringan bervariasi antara 100 Kb sampai 200 Kb. Jumlah salinan pesan sebanyak 6 salinan*(L=6)*. Node sumber *(source node)* dan node tujuan *(destination node)* dilakukan secara acak. Lama waktu simulasi sekitar 2 jam dengan TTL pesan 4 jam. Parameter lain yang belum disebutkan terdapat pada tabel 3.1.

Tabel 3.1 Spesifikasi Skenario Node(Prueksasri V., dkk, 2011)

Parameter	**Tipe Node**		
	Pejalan Kaki	**Mobil**	**Trem**
Model Pergerakan	Shortest Path Map-Based	Shortest Path Map-Based	Routed Map-Based
Waktu tunggu(detik)	10-30	10-30	50-100
Jarak transmisi(meter)	10	10	50
Ukuran buffer(MB)	10	10	50

Selain itu, pada *node pedestrian* akan ditambahkan suatu *Point of Interest(POI)* yang bertujuan untuk membentuk perilaku yang berbeda pada *node pedestrian* dibandingkan dengan node lainnya. Perilaku *POI* yang diberikan pada node

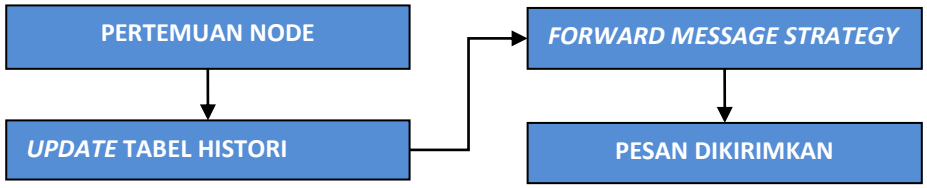
pedestrian diambil dari penelitian yang dilakukan sebelumnya.(Vittawus, P., dkk, 2011).

Tabel 3.2 Point of Interest Node Penjalan Kaki(Prueksasri V., dkk, 2011)

	Point of Interest				
	Park	**Shop**	**Meeting Spot**	**Central Point**	**Elsewhere**
Probabilty	0,1	0,1	0,3	0,2	0,3

3.2 Rancangan Sistem

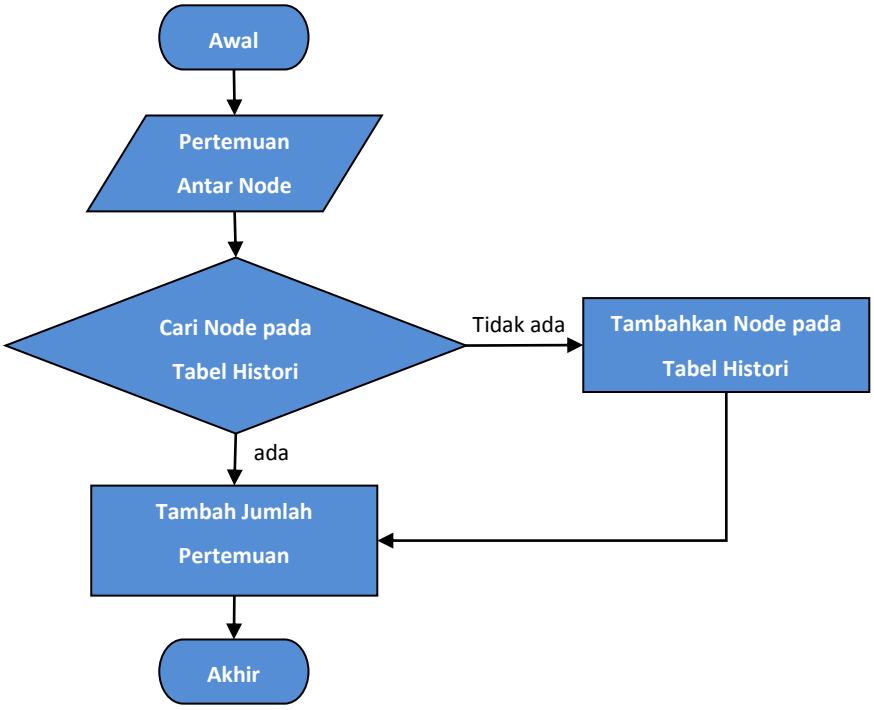
Adapun rancangan sistem dengan metode yang diusulkan digambarkan pada gambar 3.1 berikut ini:



Gambar 3.1 Rancangan Sistem Penelitian

3.2.1. *Update* Tabel Histori

Setiap saat terjadi koneksi antara *source node* dengan *node relay*, pertama kali yang harus dilakukan yaitu melakukan *update* tabel histori pada masing – masing *node* tersebut. *Node* sumber menambahkan histori telah bertemu dengan *node relay* berupa kode *node relay* dan jumlah banyaknya pertemuan dengan *node relay*. Hal ini juga dilakukan oleh *node relay* yaitu melakukan *update* terhadap tabel histori yang dimilikinya dengan menambahkan ID *node* sumber dan jumlah banyaknya pertemuan. Desain alur *update* tabel histori antar *node* ditampilkan pada gambar 3.2.



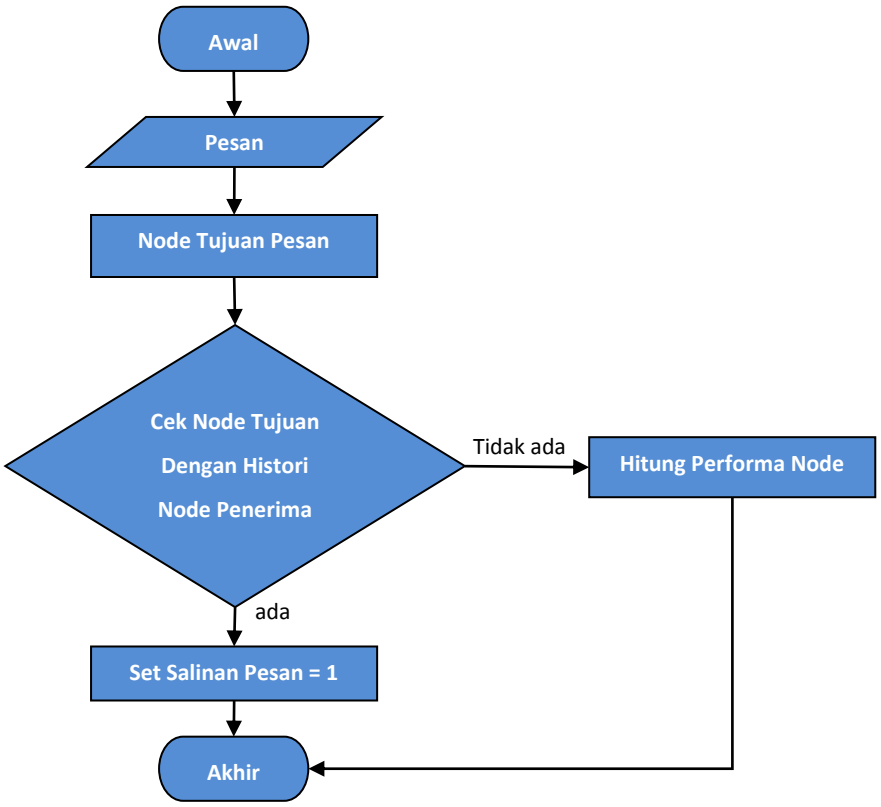
Gambar 3.2 Alur *Update* tabel histori antar *node*

3.2.2. Performa Node

Penentuan jumlah salinan pesan yang akan didistribusikan diperoleh dari performa node. Performa node yang dijadikan sebagai parameter yaitu histori node, frekuensi pertemuan, dan kecepatan node. Seluruh parameter performa node akan dihitung bersama supaya menghasilkan jumlah pesan yang akan didistribusikan.

3.2.2.1. Seleksi Pesan

Seleksi pesan dilakukan untuk mengetahui ID *node* tujuan yang ada di dalam isi pesan yang akan dikirimkan. Apabila *node* tujuan yang ada didalam isi pesan tersebut, terdapat pada histori *node* yang sedang terkoneksi dengan *node* sumber, maka jumlah pesan yang diberikan sebanyak 1 salinan. Hal ini bertujuan untuk mengurangi *overhead ratio* yang terjadi di dalam jaringan, sebab *node* tersebut sudah pernah bertemu dengan *node* tujuan pesan. Alur seleksi pesan ditampilkan pada gambar 3.3.



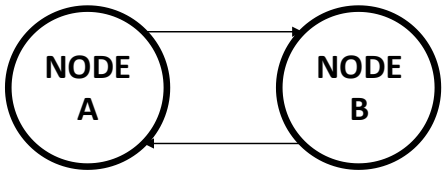
Gambar 3.3 Alur Seleksi Pesan

Pada gambar 3.3 merupakan awal mula proses seleksi pesan dilakukan yaitu setelah proses *update* histori pada masing – masing *node* yang saling bertemu. *Node* tujuan dari pesan yang diambil dari *node* sumber, dibandingkan dengan histori *node* pertemuan yang dimiliki oleh *node* yang saling bertemu. Apabila *node* tujuan dari pesan berada di dalam daftar histori *node*, maka jumlah salinan pesan yang diberikan dari *node* sumber kepada *node* yang bertemu sebanyak 1 salinan. Hal ini diasumsikan bahwa kemungkinan terjadinya pertemuan kembali antara *node* tersebut dengan *node* tujuan bisa terjadi. Sebaliknya bila tujuan *node* pesan tidak berada di dalam histori pertemuan pesan yang dimiliki oleh *node* penerima, maka dihitung nilai performa *node* penerima terhadap *node* pengirim.

3.2.2.2. Perhitungan Performa Node

Perhitungan performa node dilakukan apabila jumlah salinan pesan yang dimiliki oleh node sumber lebih dari *(L>1)* dengan status fase *spray*. Parameter

perhitungan performa node yang diusulkan diperoleh dari perbandingan antara histori, frekuensi pertemuan dan kecepatan *node*.



Gambar 3.4. Pertemuan Node A dengan node B

Gambar 3.4 merupakan ilustrasi pertemuan antara *node* A dengan *node* B dengan informasi yang ditampilkan sebagai berikut:

Node A:

$$H_A = \{c, e, b\}$$
$$F_A = \{(c, 2), (e, 3), (b, 1)\}$$
$$K_A = 1{,}2\ {}^{m}/_{s}$$

dimana:

H_A = informasi pertemuan antara *node* A dengan *node* c, e dan b.

F_A = informasi jumlah pertemuan antara *node* A dengan *node* c, e dan b.

K_A = informasi kecepatan pergerakan *node* A dalam satuan ${}^{m}/_{s}$

Node B:

$$H_B = \{e, g, y, i, a\}$$
$$F_B = \{(e, 4), (g, 2), (y, 2), (i, 1), (a, 1)\}$$
$$K_{B=}7{,}2\ {}^{m}/_{s}$$

dimana:

H_B = informasi pertemuan antara *node* B dengan *node* e, g, y, i dan a.

F_B = informasi jumlah pertemuan antara *node* B dengan *node* e, g, y, i dan a.

K_B = informasi kecepatan pergerakan *node* B dalam satuan ${}^{m}/_{s}$

Parameter histori, frekuensi pertemuan dan kecepatan digabungkan sehingga menghasilkan satu performa node yang lebih kompleks untuk menentukan jumlah banyaknya salinan pesan yang bisa diberikan kepada node berikutnya, dalam hal ini node B bila dilihat dari gambar 3.4. Dari ketiga parameter diatas, maka untuk

mendapatkan hasil performa node B dilihat dari node A dengan menggunakan rumus (3.1):

$$P_{AB} = H_{AB} * F_{AB} * K_{AB} \quad (3.1)$$

dimana:

P_{AB} = performa node B bila dilihat dari node A

H_{AB} = nilai histori node B bila dilihat dari node A

F_{AB} = nilai frekuensi pertemuan node B dilihat dari node A

K_{AB} = nilai kecepatan node B dilihat dari node A

3.2.2.2.1. Perhitungan Histori Node

Histori node dihasilkan dari perbadingan tabel histori yang dimiliki oleh node-node yang saling bertemu. Histori node digunakan untuk mengetahui pengaruh histori pertemuan node terhadap performa node. Semakin banyaknya jumlah pertemuan yang ada di dalam tabel histori, semakin banyak jumlah salinan pesan yang bisa diteruskan. Untuk mendapatkan perhitungan histori node B dilihat dari node A dengan menggunakan rumus(3.1):

$$H_B = \{H_B | H_B \notin H_A\}$$

$$H_{AB} = \frac{H_B}{H_A \cup H_B} \quad (3.2)$$

dimana:

H_{AB} = nilai histori node B dilihat dari node A

$H_B = \{H_B | H_B \notin H_A\}$ = informasi histori node B yang tidak pernah bertemu dengan node A

$H_A \cup H_B$ = informasi keseluruhan histori node A dan node B

3.2.2.2.2. Frekuensi Node

Frekuensi node dihasilkan dari perbandingan jumlah banyaknya pertemuan node pada tabel histori node. Semakin besar frekuensi node yang dihasilkan , maka jumlah distribusi pesan yang diberikan semakin besar. Untuk mendapatkan perhitungan frekuensi pertemuan node B dilihat dari node A menggunakan rumus (3.3):

$$F_{AB} = \frac{\sum_{i=1}^{n} F_{(B,i)}}{\sum_{i=1}^{n} F_{(B,i)} + \sum_{i=1}^{n} F_{(A,i)}} \quad (3.3)$$

dimana:

F_{AB} = frekuensi pertemuan node B dilihat dari node A

$\sum_{i=1}^{n} F_{(B,i)}$ = jumlah frekuensi pertemuan node B dengan histori node B

$\sum_{i=1}^{n} F_{(A,i)}$ = jumlah frekuensi pertemuan node A dengan histori node A

n = jumlah banyaknya informasi node pada tabel histori node

i = informasi urutan node ke - i

3.2.2.2.3. Kecepatan Node

Kecepatan node diperoleh dari kecepatan yang dimiliki oleh masing-masing node yang saling bertemu. Kecepatan node berpengaruh terhadap jumlah distribusi pesan. Semakin besan kecepatan node yang dimiliki oleh node, maka semakin besar jumlah distribusi pesan yang dikirimkan. Untuk mendapatkan kecepatan node B dilhat dari node A dengan menggunakan rumus (3.4):

$$K_{AB} = \frac{K_B}{K_B + K_A} \quad (3.4)$$

dimana:

K_{AB} = kecepatan node B dilihat dari node A

K_A = kecepatan node A

K_B = kecepatan node B

Setelah perhitungan performa *node* didapatkan, selanjutnya yaitu proses penentuan jumlah banyaknya salinan pesan yang bisa diberikan oleh *node* A kepada *node* B.

3.2.3. Mekanisme *Forward Message*

3.2.3.1. Fase Spray

Penentuan jumlah banyaknya salinan pesan yang bisa diberikan oleh *node* A ke *node* B diperoleh dari perkalian performa *node* B dilihat dari *node* A dengan jumlah salinan pesan yang dimiliki oleh *node* A. Untuk menghitung

banyaknya salinan pesan yang bisa dikirimkan dari *node* A ke *node* B dapat menggunakan rumus(3.5):

$$M_{AB} = P_{AB} \, x \, L_A \quad (3.5)$$

dimana:

M_{AB} = banyaknya salinan pesan yang bisa dikirimkan dari *node* A ke *node* B

P_{AB} = performa *node* B dilihat dari *node* A

L_A = salinan pesan yang dimiliki oleh node A

Setelah proses pengiriman salinan pesan dilakukan dengan jumlah sebanyak M_{AB} dari *node* A ke *node* B, maka sisa salinan pesan yang dimilki oleh *node* A dapat menggunakan rumus (3.6):

$$L_A^{"} = L_A - M_{AB} \quad (3.6)$$

dimana:

$L_A^{"}$ = sisa salinan pesan yang dimiliki oleh *node* A

L_A = salinan pesan yang dimiliki oleh *node* A sebelum proses pengiriman

M_{AB} = banyaknya salinan pesan *node* A yang dikirimkan dari *node* A ke *node* B

Bila sisa salinan pesan setelah proses *spray* yang dilakukan oleh node A yaitu $L_A^{"} > 1$, maka proses fase *spray* akan dilakukan terus menerus sampai sisa jumlah salinan pesan yang dimiliki $L_A^{"} = 1$, lalu akan masuk ke dalam fase *wait*.

3.2.3.2. Fase Wait

Fase wait dilakukan pada saat jumlah salinan pesan yang dimiliki oleh node A $L_A^{"} = 1$. Node A akan membawa sisa salinan tersebut sampai bertemu dengan *node* tujuan. Apabila *node* A bertemu dengan *node* lain yang berposisi sebagai node *relay*, maka salinan pesan yang dimiliki tidak ada akan dikirimkan.

3.3 Skenario dan Evaluasi Pengujian

3.3.1. Skenario Pengujian

Pada penelitian ini, pengujian metode baru yang diusulkan akan diujikan dengan menggunakan bantuan simulator *ONE(Opportunistic Network*

Enviroment). Simulator ONE mempunyai fitur yang dapat mengakomodir fitur–fitur yang dibutuhkan untuk mengembangkan protokol *routing* pada jaringan DTN, misalnya struktur pesan sudah didefinisikan dengan baik termasuk *message ID, source node ID, destination node ID, message size, message path, creation time, receive time, initial TTL, dan response size*.

Adapun skenario pengujian dalam penelitian yang akan dilakukan yaitu sebagai berikut:

1. Jumlah *node* yang akan diujikan pada penelitian ini terdiri dari 106 *node* dan dikelompokkan ke dalam *node pedestrian, cars,* dan *node tram.* Jumlah node diambil dari jumlah node pada penelitian Vittawus, P., 2011. Selain itu, jumlah tersebut merupakan perbandingan antara jumlah banyaknya *pedestrian*, *cars* dan *tram*. Terdapat 3 kali pengujian dengan jumlah pada masing-masing kelompok yang berbeda. Ini seperti terlihat pada Tabel 3.3.

   Tabel 3.3. Spesifikasi Data Uji Node

Keterangan	Pedestrian	Cars	Tram
Uji Coba 1	70	30	6
Uji Coba 2	6	70	30
Uji Coba 3	30	6	70

   Pengujian dengan melakukan perubahan terhadap jumlah *node* yang berbeda dalam jaringan dengan tujuan untuk mengetahui seberapa jauh pengaruh jumlah *node* terhadap kinerja protokol dengan metode baru yang diusulkan.

2. Jumlah salinan pesan yang diberikan terdiri dari 10, 20, 30, 40 dalam periode simulasi. Pengujian jumlah salinan pesan bertujuan untuk mengetahui pengaruh jumlah banyaknya salinan pesan yang beredar didalam jaringan terhadap kinerja dari protokol dengan metode baru yang diusulkan.

3.3.2. Evaluasi Pengujian

Berdasarkan dari skenario pengujian yang dilakukan pada penelitian ini akan dibandingkan dengan hasil evaluasi pengujian pada protokol *routing DNH-SaW*. Parameter evaluasi pengujian sebagai berikut:

1. Evaluasi Pengujian Delivery Ratio

   Evaluasi pengujian *delivery ratio* diperoleh dari jumlah banyaknya pesan yang diterima oleh node tujuan dibanding dengan jumlah banyaknya pesan yang dibuat. Untuk menghitung *delivery ratio* menggunakan rumus (3.7):

$$Delivery\ ratio\ = \frac{D}{C} \tag{3.7}$$

   dimana:

   D = jumlah banyaknya pesan yang terkirim dan sampai ke *node* tujuan

   C = jumlah banyaknya pesan yang dibuat.

2. Evaluasi Pengujian Overhead Ratio

   Evaluasi pengujian *overhead ratio* diperoleh dari jumlah banyaknya pesan yang terkirim*(relayed)* dikurangi dengan jumlah banyaknya pesan yang diterima oleh node tujuan dibanding dengan jumlah banyaknya pesan yang diterima oleh node tujuan. Untuk menghitung overhead ratio menggunakan rumus (3.8):

$$Overhead\ ratio\ = \frac{R - D}{D} \tag{3.8}$$

   Dimana:

   R = jumlah banyaknya pesan yang dikirim oleh *node* pengirim*(relay)*

   D = jumlah banyaknya pesan yang diterima oleh *node* tujuan

3. Evaluasi Pengujian Latency

   Evaluasi pengujian *latency* dengan menghitung pesan diterima oleh *node* tujuan dibandingkan dengan jumlah banyaknya pesan yang dibuat oleh *node* sumber. Untuk menghitung nilai *latency* menggunakan rumus (3.9):

$$Latency = \sum_i^n (R_i - C_i) \tag{3.9}$$

   dimana:

   R_i = waktu saat pesan *i* diterima oleh *node* tujuan

   C_i = waktu pesan *i* dibuat

3.3.3. Lingkungan Uji Coba

Dalam proses pengujian, metode baru yang diajukan akan diujikan pada sebuah komputer. Adapun spesifikasi dari komputer yang digunakan sebagai pendukung dalam implementasi penelitian yaitu sebagai berikut:

a. Processor Intel® Core™ i3 CPU M350 @2,27GHz
b. Memory RAM 4 GB
c. Hardisk 250GB
d. Sistem Operasi Windows 7
e. Compiler dan runtime java versi 1.8 update 31
f. Editor Netbeans 8.02

3.3.4. Pengujian dengan Network Simulator ONE

Simulator ONE merupakan *tools* simulasi jaringan yang digunakan untuk melakukan pengujian kinerja dari protokol yang digunakan dalam lingkungan jaringan DTN. Metode baru yang diusulkan dalam penelitian ini akan diuji dengan menggunakan ONE simulator. Terdapat protokol yang nantinya dimodifikasi dengan menggunakan metode baru yaitu protokol DNHSaW. Protokol DNHSaW merupakan pengembangan dari protokol SaW yang sudah ada didalam modul ONE simulator. Protokol DNHSaW dibuat terlebih dahulu dengan mengikuti referensi dari paper "DNH-SaW: The Different Neighbor-History Spray and Wait Routing Scheme for Delay Tolerant Networks" yang sudah diteliti sebelumnya oleh Vittawus, P., dkk tahun 2011.

3.4 Dokumentasi dan Jadwal Pembuatan Sistem

Pada tahap dokumentasi sistem ini akan dilakukan penulisan laporan hasil penelitian dari setiap tahapan proses perancangan, pembuatan, implementasi dan analisis. Kemudian dokumentasi ini juga bisa dijadikan sebagai bahan rujukan atau referensi untuk mengembangkan metode baru lainnya, sehingga dapat menghasilkan nilai penelitian yang baik. Selain itu, untuk melengkapi penelitian dibuat jadwal kegiatan penelitian seperti terlihat pada Tabel 3.3.

Tabel 3.4 Jadwal Kegiatan Penelitian

No.	Kegiatan	Bulan – 1				Bulan - 2				Bulan – 3				Bulan – 4			
		1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1.	Studi literature																
2.	Analisis dan desain																
3.	Implementasi Perangkat Lunak																
4.	Uji coba dan analisis																
5.	Penulisan laporan																

BAB 4
HASIL PENELITIAN DAN PEMBAHASAN

Bab ini akan memaparkan tahapan-tahapan yang dilakukan dalam implementasi perbaikan kinerja protokol *routing* The Different-Neighbour History Spray and Wait(DNH-SaW) berdasarkan *performa node* serta *destination message history* pada lingkungan jaringan *Delay Tolerant Network*s(DTN). Implementasi dilakukan dengan menggunakan simulator yang mendukung dalam lingkungan jaringan DTN yaitu ONE*(Opportunistic Network) Simulator*. Pada bab ini juga akan dijabarkan mengenai penerapan skenario pengujian dan evaluasi metode dengan menganalisa data hasil pengujian yang telah dilakukan. Hasil analisis akan disajikan pada bagian akhir dari bab ini.

4.1. Tahapan Implementasi Metode

Penelitian ini disimulasikan dengan menggunakan *network simulator ONE*. Pada sub-bab ini akan disajikan langkah-langkah implementasi secara umum dan akan dibahas lebih terperinci pada sub-bab berikutnya. Tahapan implementasi metode penelitian yang digunakan sebagai berikut:

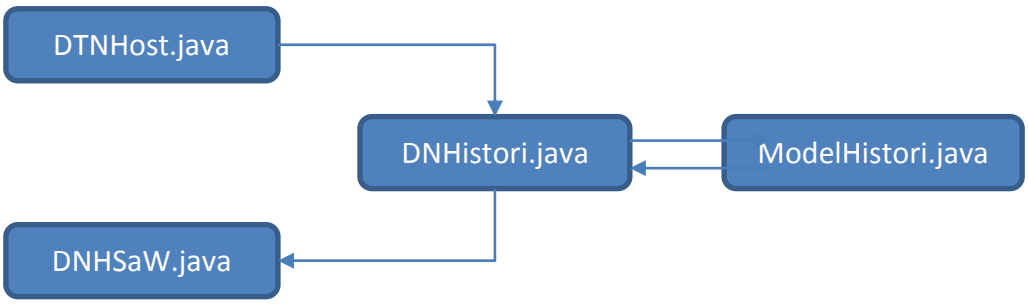
1. Tahapan pertama adalah tahapan perancangan dan implementasi metode perhitungan performa node serta *destination message history* pada protokol DNH-SaW. Tahapan ini diimplementasikan dengan menggunakan simulator ONE berbasis bahasa pemrograman java.
2. Tahapan kedua adalah perancangan skenario pengujian yang akan digunakan dalam menguji kinerja dari metode baru yang diusulkan. Skenario pengujian dirancang meliputi variasi perubahan jumlah node *pedestrian, cars* dan *tram* dengan kombinasi jumlah node 70, 30, dan 6, sehingga total jumlah node sebanyak 106. Selain itu, pengujian juga dilakukan dengan merubah variasi jumlah paket data yang dikirimkan dari node sumber kepada node tujuan yang terdiri dari 10, 20, 30 dan 40 paket data.

3. Tahap ketiga adalah tahapan rangkaian pengujian dari keseluruhan perancangan maupun skenario yang sudah dipersiapkan.

Pembuatan program untuk implementasi metode baru yang diajukan dilakukan dengan melakukan modifikasi beberapa modul yang ada didalam simulator ONE, seperti metode transferDone() dan messageTransferred() yang berada di dalam modul file DNHSaW.java. Pada sub-bab dibawah ini akan dijelaskan lebih terperinci terkait modifikasi apa saja yang dilakukan.

4.1.1. Modifikasi Protokol DNHSaW

Protokol *routing* DNHSaW dikembangkan dari protokol *routing Spray and Wait(SaW)* yang ada didalam modul *routing* pada simulator ONE. Sebelumnya protokol routing DNHSaW sudah dikembangkan oleh peneliti lain yaitu Vittawus P., dkk tahun 2009. Pada penelitian ini protokol DNSaW dibuat kembali dengan menggunakan *paper* yang dihasilkan oleh peneliti Vittawus P., sebagai basis pengembangannya. Protokol DNHSaW dibuat dengan melakukan modifikasi terhadap protokol SaW. Tetapi untuk melakukan implementasi protokol DNHSaW dibuat dengan melakukan *generate* kelas baru yang dinamakan dengan kelas DNHSaW.java. Kelas DNHSaW.java akan melakukan interaksi dengan kelas lainnya yaitu kelas DNHistori.java dan kelas ModelHistori.java. Kedua kelas ini DNHistori.java dan ModelHistori.java digunakan untuk melakukan interaksi seperti menyimpan data histori pertemuan antar node, mencetak histori petermuan antar node pada masing-masing node.



Gambar 4.1. Alur Proses Kinerja Protokol DNHSaW

Proses pergerakan alur kerja protokol DNHSaW dimulai dari saat dua node tertentu saling bertemu, yang diproses pada kelas DTNHost.java. Pada saat proses node saling bertemu, metode connectionUp() pada kelas DTNHost.java akan memanggil kelas DNHistori.java untuk melakukan perubahan*/update* data tabel histori pada masing-masing node. Kelas DNHistori.java akan melakukan validasi terlebih dahulu informasi yang ada di dalam tabel histori. Apabila dua node yang saling bertemu belum pernah melakukan pertemuan, maka masing-masing node akan menambahkan informasi node yang ditemuinya ke dalam tabel histori. Apabila dua node yang saling bertemu pernah melakukan pertemuan dengan node yang diteminya, maka proses penambahan informasi node yang ditemui pada tabel histori tidak perlu dilakukan. Untuk melakukan perubahan data informasi node yang saling bertemu, dilakukan pada salah satu metode yang ada didalam kelas DNHistori.java, yaitu metode addList(). Didalam metode addList() ini, dilakukan terlebih dahulu proses pencarian informasi node yang ada sebelumnya didalam histori node. Apabila node yang ditemui sudah pernah bertemu sebelumnya atau histori node tersebut pernah tersimpan, maka tidak ada aksi apapun. Dan sebaliknya, apabila node yang sedang bertemu belum pernah bertemu sebelumnya, maka node tersebut akan disimpan pada tabel histori.

Setelah proses perubahan informasi node pada tabel histori dilakukan, proses distribusi salinan pesan dilakukan pada kelas DNHSaW.java. Proses salinan pesan dihitung berdasarkan nilai performansi yang dimiliki oleh node tertentu. Perhitungan salinan pesan yang diberikan oleh suatu node didapatkan dari perhitungan performa histori node tertentu yang dihitung dengan menggunakan metode getHistPerform(), dikalikan dengan sisa salinan pesan yang dimiilki oleh node. Kemudian sisa salinan pesan terakhir yang dimilki oleh node pengirim dihasilkan dari sisa salinan awal dikurangi dengan salinan pesan yang didistribusikan ke node setelahnya.

Untuk melakukan implementasi metode baru yang diusulkan, protokol DNHSaW dimodifikasi dengan menambahkan metode baru untuk menghitung performa frekuensi node dan kecepatan. Untuk memudahkan proses implementasi metode baru yang diusulkan, maka dibuat kelas baru yaitu bernama kelas DNHSaWNew.java. Di dalam kelas DNHSaWNew.java ditambahkan variabel

perhitungan performa node histori, frekuensi node dan kecepatan node. Selain itu, untuk distribusi salinan pesan ditambahkan dengan metode *destination message history*.

4.1.2. Destination Message History

Penambahan metode *destination message history* pada protokol DNHSaWNew di dalam file DNHSaWNew.java dilakukan dengan membuat seleksi node dari node tujuan pesan terhadap histori node yang dimiliki oleh node relay. Bila node tujuan dari pesan berada pernah bertemu dengan node relay, maka jumlah salinan pesan yang diberikan sebanyak satu salinan. Proses ini dilakukan untuk mengurangi jumlah peredaran pesan yang ada didalam jaringan.

```
……
import paper.DNHistori;
import paper.ModelHistori;
……

public class DNHSaWNew extends ActiveRouter {
        ……
        @Override
        public Message messageTransferred(String id, DTNHost from) {
                ……
                DNHistori recvHistory  = this.getHost().getHistori();
                ……
                if(recvHistory.inList(msg.getTo().toString()))
                {
                    nrofCopies -= 1;
                }else { …… }

                msg.updateProperty(MSG_COUNT_PROPERTY,nrofCopies);
                return msg;
        }
……
        @Override
        protected void transferDone(Connection con) {
                ……
                DNHistori recvHistory  = hostReceiver.getHistori();
                ……
                if(recvHistory.inList(msg.getTo().toString()))
                {
                    nrofCopies -= 1;
                }else { …… }

               msg.updateProperty(MSG_COUNT_PROPERTY, nrofCopies);
        }
         ……
}
```

Gambar 4.2. Kelas DNHSaWNew.java

Pada gambar 4.2, proses salinan pesan dilakukan pada dua metode yaitu messageTransferred() dan metode transferDone(). Kedua metode mempunyai

pemahaman dari dua sisi yang berbeda. Metode messageTransferred() dijalankan apabila proses pengiriman pesan sudah dilakukan pada *node sender*. Sedangkan metode transferDone() dilakukan sebelum proses pengiriman pesan kepada *node receiver*.

4.1.3. Menambahkan Performa Node

Performa node dihasilkan dari tiga variabel yang berbeda yaitu history node, frekuensi node, dan kecepatan node. Ketiga variabel tersebut dijadikan sebagai pemicu banyaknya jumlah salinan pesan yang bisa dikirimkan. Perubahan data node yang saling bertemu akan diletakkan ke dalam kelas file ModelHistori.java dan DNHistori.java.

```
……
public class ModelHistori {

    private String Node;
    private int jPertemuan;

    public ModelHistori(String Node, int jPertemuan) {
        this.Node = Node;
        this.jPertemuan = jPertemuan;
    }

    public String getNode() {
        return Node;
    }

    public int getjPertemuan() {
        return jPertemuan;
    }

    public void setjPertemuan(int jPertemuan) {
        this.jPertemuan = jPertemuan;
    }
}
```

Gambar 4.3 Kelas ModelHistori.java

Kelas ModelHistori.java pada gambar 4.3 akan dibuat sebagai tempat untuk menampung setiap perubahan berupa penambahan atau cek informasi pada tabel histori node. Kelas ModelHistori.java diinisialisasi pada waktu ONE simulator melakukan inisialisasi node-node yang nantinya digunakan pada waktu simulasi. Inisialisasi ini dilakukan pada oleh kelas DTNHistori.java pada kelas DTNHost.java. Ini terlihat seperti pada gambar 4.4.

```
……
```

```
import paper.DNHistori;
……
public class DTNHost implements Comparable<DTNHost> {
        ……
        public DTNHost(List<MessageListener> msgLs,
                       List<MovementListener> movLs,
                       String groupId, List<NetworkInterface> interf,
                       ModuleCommunicationBus comBus,
                       MovementModel mmProto, MessageRouter mRouterProto) {
                ……
                 this.histori = new DNHistori(this.name);
        }
……
}
```

Gambar 4.4. Inisialiasi TabelHistori pada Kelas DTNHost.java

Pada *constructor* kelas DTNHost.java, inisialisasi dipanggil dengan melewatkan parameter berupa nama node yang saat itu akan dibuat, sehingga masing-masing node yang dibuat akan mempunyai model data berupa tabel histori. Tabel histori inilah yang akan dijadikan sebagai tempat untuk menampung histori pertemuan node yang terjadi.

Untuk menambahkan atau melakukan perubahan terhadap tabel histori yang dimiliki oleh masing-masing node pada saat pertemuan berlangsung, metode insertNewNode() yang ada di dalam kelas DNHistori.java akan dipanggil. Ini terlihat seperti pada gambar 4.5.

```
……
public class DNHistori {
    ……
    private void insertNewNode(String inNode, int inPertemuan)
    {
        ModelHistori modelHistori_Temp = new ModelHistori(inNode, inPertemuan);
        modelHistori.add(modelHistori_Temp);
    }
    ……
}
```

Gambar 4.5. Gambar Kelas DNHistori.java

Pada gambar 4.5 kelas DNHistori.java didalam metode insertNewNode() digambarkan inisialisasi berupa kelas ModelHistori sebelum dilakukan proses penambahan node di dalam tabel histori.

Perhitungan performa node dilakukan apabila *node destination* dari suatu pesan yang dibawa oleh *node sender*, tidak pernah bertemu dengan salah satu *node* yang pernah bertemu dengan *node* penerima pada tabel histori.

4.1.3.1.Menghitung Performa Histori Node

Performa histori *node* dihasilkan dari pertemuan antar node yang saling bertemu dan tersimpan didalam tabel histori. Perubahan histori *node* dilakukan terus menerus saat node saling bertemu. Setelah proses perubahan data yang ada di dalam tabel histori masing-masing *node* yaitu pengirim(*sender)* dan penerima(*receiver)*, maka selanjutnya proses perhitungan performa yang dimiliki oleh *node sender* dan *receiver*. Perhitungan performa dilakukan pada *node sender* dengan melihat performa *node receiver* terhadap *node sender*. Metode perhitungan performa histori node dipanggil dengan menggunakan getHistPerform(). Ini terlihat dari gambar 4.6.

```
……
public class DNHSaWNew extends ActiveRouter {
        ……
        private Double getHistPerform(DTNHost senderHost, DTNHost receiverHost)
        {
            DNHistori sendHistori = senderHost.getHistori();
            DNHistori receivHistori = receiverHost.getHistori();
            double s = 0.0;
            double v = 0.0;
            double p = 0.0;
            for(ModelHistori mhTempA : sendHistori.getAll())
            {
                for(ModelHistori mhTempB : receivHistori.getAll())
                {
                    if(mhTempA.getNode().equals(mhTempB.getNode()))
                    {
                        s++;
                    }
                }
            }

            v = receivHistori.getAll().size() - (s+1);
            s = (sendHistori.getAll().size()+receivHistori.getAll().size())-s;
            p = v/s;
            return p;
        }
……
}
```

Gambar 4.6. Potongan Kode Menghitung Performa Histori Node

Pada gambar 4.6., tabel histori *node sender* dan *node receiver* akan dipanggil dan di cek informasi yang dimiliki keduanya. Apabila pada tabel histori *node receiver*, salah satu node yang ada di dalam tabel tersebut pernah bertemu dengan salah node yang ada di dalam tabel histori *sender*, maka tidak dihitung pada saat proses perhitungan performa node. *Complete set* akan dijadikan sebagai pembanding pada waktu proses perhitungan performa histori yaitu jumlah histori

node antara tabel histori *node sender* ditambah dengan *node receiver* secara ekuivalen, sehingga didapatkan nilai performa histori node receiver.

4.1.3.2.Menghitung Performa Frekuensi Node

Perhitungan performa *frekuensi node* dihasilkan dari jumlah frekuensi pertemuan antara *node receiver* dengan tabel histori *node receiver* dibanding dengan jumlah frekuensi *node receiver* dan *node sender*. Jumlah pertemuan tersebut nantinya akan dijadikan pendukung untuk menentukan jumlah salinan pesan yang akan diberikan. Ini terlihat pada gambar 4.7.

```
……
public class DNHSaWNew extends ActiveRouter {
        ……
        private Double getFreqPerform(DTNHost senderHost, DTNHost receiverHost)
        {
            DNHistori sendHistori = senderHost.getHistori();
            DNHistori receivHistori = receiverHost.getHistori();
            double freqReceiver = getFreqHistoriCount(receivHistori);
            double freqSender = getFreqHistoriCount(sendHistori);
            double sumRS = freqReceiver + freqSender;
            double p = (double) freqReceiver / (freqReceiver + freqSender);

            return p;
        }
        ……
}
```

Gambar 4.7. Potongan Kode Menghitung Performa Frekuensi Node

Pada gambar 4.7 didalam metode getFreqPerform() terdapat variabel freqReceiver yang akan memanggil dan menghitung jumlah frekuensi node receiver dan node sender pada metode getFreqHistoriCount(). Selanjutnya menghitung nilai performa *node receiver* terhadap *node sender* dengan membandingkan antara frekuensi *node receiver* dengan ke dua tersebut yaitu *node sender* dan *receiver*.

4.1.3.3.Menghitung Performa Kecepatan Node

Performa kecepatan diambil kecepatan masing-masing *node* yang saling bertemu dan dilakukan perbandingan antar node tersebut. Kecepatan node yang dijadikan sebagai perhitungan performa kecepatan yaitu diambil dari kecepatan node secara dinamis yang di-*set* pada file konfigurasi. Ini terlihat pada gambar 4.8.

```
……
public class DNHSaWNew extends ActiveRouter {
        ……
        private Double getVeloPerform(DTNHost senderHost, DTNHost receiverHost)
        {
            double velSender = senderHost.getSpeed();
            double velReceiver = receiverHost.getSpeed();
            double p = velReceiver / (velReceiver + velSender);
            return p;
        }
        ……
}
```

Gambar 4.8. Potongan Kode Menghitung Performa Kecepatan Node

Pada gambar 4.8 didalam kelas DNHNew.java terdapat metode getVeloPerform() yang digunakan untuk menghitung kecepatan *node receiver* dan *sender*. Kecepatan masing-masing node diambil dari perubahan kecepatan secara dinamis pada saat simulasi berlangsung. Performa kecepatan *node* pada saat pertemuan berlangsung dihitung dari perbandingan kecepatan *node receiver* terhadap *sender* dibanding dengan jumlah kecepatan *node receiver* dan node *sender.*

4.2. Langkah – langkah Uji Coba

Tahapan pengujian dalam penelitian ini bertujuan untuk membandingkan kinerja protokol modifikasi DNHSaW dengan protokol DNHSaWNew untuk setiap skenario yang telah dijelaskan pada bab 3. Kinerja dari kedua protokol ini dinilai dari parameter pengujiannya.

Langkah–langkah pengujian pada penelitian ini adalah membuat skenario pengujian, menentukan parameter pengujian, dan melakukan analisa terhadap hasil pengujian yang telah dilakukan.

4.2.1. Implementasi Skenario Pengujian

Pada bab sebelumnya sudah dijelaskan spesifikasi skenario pengujian yang akan digunakan untuk menguji metode baru yang diusulkan. Implementasi skenario pengujian dilakukan dengan membuat skrip pengujian yang terdiri dari spesifikasi seperti perubahan jumlah node, dan perubahan jumlah paket data.

4.2.2. Parameter Pengujian

Berdasarkan tujuan dari penelitian ini, untuk meningkatkan *delivery probability* yaitu dengan menambahkan variabel *histori node*, frekuensi *node*, dan kecepatan node. Selain menambahkan variabel performa *node*, juga ditambahkan variabel penentuan *destination message history* yang diletakkan pada masing-masing *node*. Penelitian akan membandingkan metode baru yang diusulkan dengan metode sebelumnya yaitu DNHSaWNew dan DNHSaW.

Parameter pertama adalah *delivery ratio* yang merupakan perbandingan antara jumlah pesan yang diterima oleh *destination node* terhadap jumlah pesan yang dibuat pada *source node*. Semakin besar jumlah pesan yang dikirim*(relayed)* pada masing-masing *node*, maka seharusnya semakin besar juga nilai *delivery ratio* yang dihasilkan.

Parameter kedua adalah *overhead ratio* dihasilkan dari perbandingan jumlah pesan yang dikirimkan oleh *node* dikurangi dengan jumlah pesan yang diterima oleh *destination node* dibagi dengan jumlah pesan yang diterima oleh *destination node*. *Overhead ratio* juga berhubungan dengan jumlah pesan yang beredar di jaringan.

Parameter ketiga adalah *latency* yang diperoleh dari perbandingan waktu pesan diterima dikurangai dengan waktu pesan dibuat kemudian dibagi dengan jumlah banyaknya pesan yang diterima oleh node destination.

Proses simulasi yang dilakukan akan menghasilkan suatu *report* yang menyatakan informasi tentang parameter-parameter yang dijadikan sebagai rujukan berhasil atau tidaknya suatu metode baru yang diusulkan bekerja. Report ini akan diletakkan pada suatu file tertentu yang ditunjukkan seperti pada gambar 4.9.

```
Message stats for scenario DNHSaWNew_uji_1
sim_time: 36000.1000
created: 10
started: 40
relayed: 40
aborted: 0
dropped: 40
removed: 0
delivered: 10
delivery_prob: 1.0000
```

```
.....
overhead_ratio: 3.0000
latency_avg: 1925.1600
.....
```

Gambar 4.9. Tampilan Report Hasil Simulasi

4.2.3. Pembuatan Skrip Konfigurasi Untuk Pengujian

Sebelum pengujian dilakukan, penentuan parameter simulasi perlu dilakukan terlebih dahulu. Parameter simulasi ini tetap untuk setiap skenario pengujian, sehingga setiap skrip konfigurasi pengujian akan dijalankan pada simulator ONE. Parameter simulasi akan diguankan pada pengujian ini dapat dilihat pada tabel 4.1.

Tabel 4.1. Parameter Simulasi Untuk Pengujian

No.	Keterangan	Spesifikasi
1	Jenis Simulator	ONE(Opportunistic Network)
2	Jumlah Node	106 node
3	Are Simulaasi	4500 x 3400 meter
4	Waktu Simulasi	3000 (detik)
5	*Source/Destination Node*	Random dalam node

Parameter simulasi pada tabel 4.1 akan digunakan untuk membuat skrip konfigurasi pengujian pada masing-masing skenario yang sudah dijelaskan sebelumnya. Skrip konfigurasi pengujian yang digunakan akan dijelaskan pada sub bab berikutnya.

4.2.3.1. Konfigurasi Pengujian Perubahan Node

Skrip konfigurasi pengujian perubahan node dibuat untuk mengetahui perubahan yang terjadi pada protokol dengan metode baru yang diusulkan. Jumlah node yang digunakan sebanyak 106 node yang dibagi menjadi 3 yaitu node pedestrian, cars, dan tram. Masing-masing node dari group pedestrian, cars dan tram akan memiliki jumlah node yang berbeda antara lain: pedestrian 70 node,

cars 30 node, dan tram 6 node. Kemudian untuk pengujian selanjutnya, jumlah node pedestrian dirubah menjadi 6 node, cars sebanyak 70 node, dan tram sebanyak 30 node. Pengujian terakhir perubahan node, jumlah node pedestrian akan dirubah menjadi sebanyak 30 node, cars sebanyak 6 node, dan tram sebanyak 70 node. Masing-masing perubahan jumlah node ini untuk melihat pengaruh jumlah node terhadap protokol dengan metode baru yang diusulkan. Selain itu, untuk jumlah paket data yang dikirimkan pada pengujian ini sebanyak 10 paket data. Hal ini diilustrasikan seperti pada gambar 4.10, 4.11, 4.12.

```
……
#       Group "Pedestrian"
Group1.groupID = P
……
Group1.nrofHosts = 70

#               Group "Car"
Group2.groupID = C
……
Group2.nrofHosts = 30

#               Group "Tram"
Group3.groupID = T
……
Group3.nrofHosts = 6
……
```

Gambar 4.10. Potongan Skrip Implementasi Pengujian Perubahan Node dengan jumlah node pedestrian sebanyak 70, cars sebanyak 30, dan tram sebanyak 6.

```
……
#       Group "Pedestrian"
Group1.groupID = P
……
Group1.nrofHosts = 6

#               Group "Car"
Group2.groupID = C
……
Group2.nrofHosts = 70

#               Group "Tram"
Group3.groupID = T
……
Group3.nrofHosts = 30
……
```

Gambar 4.11. Potongan Skrip Implementasi Pengujian Perubahan Node dengan jumlah node pedestrian sebanyak 6, cars sebanyak 70, dan tram sebanyak 30.

```
……
#       Group "Pedestrian"
Group1.groupID = P
……
Group1.nrofHosts = 30

#               Group "Car"
Group2.groupID = C
……
Group2.nrofHosts = 6

#               Group "Tram"
Group3.groupID = T
……
Group3.nrofHosts = 70
……
```

Gambar 4.12. Potongan Skrip Implementasi Pengujian Perubahan Node dengan jumlah node pedestrian sebanyak 30, cars sebanyak 6, dan tram sebanyak 70.

4.2.3.2. Konfigurasi Pengujian Perubahan Pesan

Parameter skenario pengujian pada perubahan paket data pada jaringan dilakukan untuk melihat efektifitas jumlah paket data yang terkirim dengan paket data yang tidak terkirim.

Tabel 4.2. Parameter Skenario Perubahan Paket Data

No	Parameter	Spesifikasi
1	Jumlah Pesan Pengiriman	10, 20, 30, dan 40
2	Ukuran Paket Data	antara 100 Kb sampai 200 Kb
3	Waktu Pengiriman	antara 0 sampai 500 detik

Implementasi pembuatan variasi pesan dilakukan dengan memanfaatkan toolkit yang ada didalam modul simulator ONE yaitu *file createCreates.pl*. File ini bertugas untuk membuat paket data dengan ukuran yang dapat dirubah sesuai dengan jumlah paket yang diinginkan.

```
createCreates.pl -time 0:500 -nrof 10 -hosts 0:106 -size 100000:200000 >
paketdata_10.txt
```

Gambar 4.13. Potongan Skrip Pembuatan Paket Data

Dari gambar 4.13 terlihat waktu pengiriman paket data antara node sumber dengan node tujuan dibagi 0 sampai 500 detik dengan node tujuan sebanyak 20 node yang tersebar di dalam jaringan, dan ukuran paket data antara 100 Kb sampai 200Kb. Setelah proses pembuatan spesifikasi paket data yang diletakkan ke dalam suatu file, maka file tersebut akan dipanggil pada file konfigurasi. File konfigurasi ini yang akan dibaca oleh simulator untuk menguji protokol dengan metode baru yang diusulkan terhadap perubahan paket data. Hal ini seperti terlihat pada gambar 4.14.

```
……
# Konfigurasi:
#       Events
Events.nrof = 1
Events1.class = StandardEventReader
Events1.filePath = ee/paketdata_10.txt
……
```

Gambar 4.14. Potongan Skrip Implementasi Pengujian Perubahan Jumlah Paket Data 10, 20, 30, dan 40.

4.3. Hasil dan Analisis

Hasil pengujian pada sub-bab ini diperoleh dengan melakukan analisis terhadap file teks *report* yang dihasilkan dari pengujian protokol baru yang diusulkan(DNHSaWNew) dibandingkan dengan protokol DNHSaW. Dari hasil pengujian ini, parameter yang akan dianalisis adalah nilai *delivery probability, overhead ratio, dan latency*. Tampilan hasil pengujian berupa report teks terlihat seperti pada gambar 4.9.

Untuk memudahkan dalam melakukan analisa terhadap hasil yang sudah diperoleh pada saat pengujian, maka penyajiannya akan digambarkan dalam bentuk tabel dan grafik.

Alur proses untuk mendapatkan hasil dari tahap pengujian pada simulator ONE diilustrasikan seperti pada gambar 4.13.

Gambar 4.15. Ilustrasi Alur Pengujian dan Analisa

Pada gambar 4.15, tersebut terlihat alur proses pengujian mulai dari pembuatan skenario konfigurasi sampai dengan hasil dalam bentuk analisis tabel dan grafik. Blok hasil pengujian berupa *report*, *file* teks yang digunakan yaitu berekstensi berupa *.txt*, sehingga memudahkan dalam melakukan analisa.

4.3.1. Analisis Pengaruh Delivery Ratio

Parameter pertama yang diamati pada hasil implementasi dan pengujian yaitu pengaruh *delivery ratio* terhadap kinerja protokol dengan metode baru yang diusulkan. Delivery *ratio* dihasilkan dari jumlah banyaknya paket data yang diterima oleh node tujuan dibanding dengan jumlah banyaknya paket data yang dibuat oleh *node* sumber seperti yang diuraikan pada bab 3. Pada pengujian ini menggunakan pada kategori pengujian dengan perubahan *node* yang berbeda digunakan jumlah paket data sebanyak 10. Hal ini dilakukan untuk mengetahui seberapa efektif paket data yang dikirimkan bisa sampai tujuan. Hasil *delivery ratio* secara keseluruhan dituangkan ke dalam Tabel 4.3.

Tabel 4.3. Hasil Pengujian Delivery Ratio

Skenario		Delivery Ratio	
		Protokol DNHSaW	Protokol DNHSaWNew
Perubahan Node	NODE		
	Pedestrian = 70; Cars = 30; Tram = 6	0.8	0.6
	Pedestrian = 6; cars = 70; Tram = 30	1	1
	Pedestrian = 30; cars = 6; Tram = 70	0.7	0.9
Jumlah Pesan	PESAN		
	10	0.8	0.6
	20	0.65	0.95
	30	0.6667	0.8
	40	0.75	0.95

Dari tabel 4.3 secara keseluruhan dari hasil pengujian dengan menggunakan variabel perubahan jumlah node pada masing-masing *node pedestrian, cars* dan *tram* serta perubahan jumlah paket data, terlihat tren yang baik dihasilkan oleh protokol DNHSaWNew bila dibandingkan dengan protokol

DNHSaW. Pada pengujian dengan spesifikasi node PCT-1, protokol DNHSaWNew menghasilkan tingkat *delivery ratio* yang lebih kecil bila dibandingkan dengan protokol DNHSaW. Hal ini disebabkan nilai performa node yang diperoleh lebih kecil bila dibandingkan dengan performa node pada protokol DNHSaW. Nilai rata-rata *delivery ratio* yang dihasilkan oleh protokol DNHSaWNew pada kategori jumlah *node* dengan jumlah 10 paket data yang dikirimkan yaitu sebesar 83,3% sama seperti yang dihasilkan oleh protokol DNHSaW.

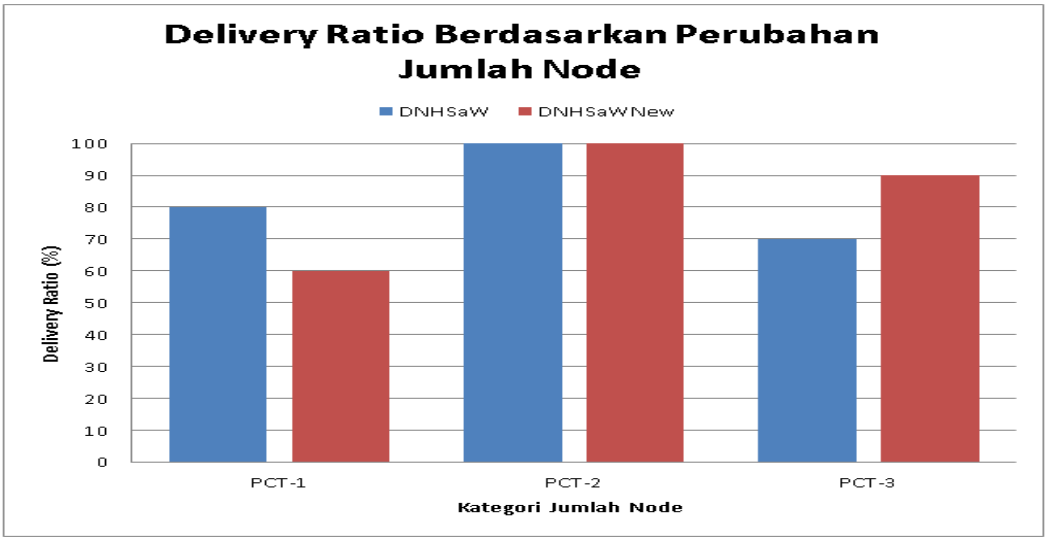
Untuk melihat hasil yang lebih mendekati bentuk asli dari kinerja protokol DNHSaWNew dengan DNHSaW dapat menggunakan perbandingan jumlah paket data yang dikirim dengan jumlah paket data yang diterima. Hasil pengujian berupa perbandingan jumlah paket data yang terkirim disajikan ke dalam Tabel 4.4.

Tabel 4.4. Hasil Pengujian Berdasarkan Jumlah Pesan

Skenario		Protokol DNHSaW				Protokol DNHSaWNew			
		Created	Relayed	Dropped	Delivered	Created	Relayed	Dropped	Delivered
Perubahan Node	NODE								
	Pedestrian = 70; Cars = 30; Tram = 6	10	64	0	8	10	48	0	6
	Pedestrian = 6; cars = 70; Tram = 30	10	88	0	10	10	275	0	10
	Pedestrian = 30; cars = 6; Tram = 70	10	168	0	7	10	386	0	9
Jumlah Pesan	PESAN								
	10	10	64	0	8	10	48	0	6
	20	20	89	0	13	20	373	1	19
	30	30	134	0	20	30	438	2	24
	40	40	228	0	30	40	614	12	38

Pada tabel 4.4 apabila dibandingkan antara protokol DNHSaWNew dengan DNHSaW berdasarkan rata-rata perbandingan jumlah paket yang terkirim, maka akan terlihat perbedaan yang lebih detail. Pada kategori perubahan node, rata-rata pengiriman paket data yang terkirim dan sampai tujuan sebesar 83,3% pada protokol DNHSaWNew dan DNHSaW. Sedangkan pada kategori perubahan jumlah paket data yang dikirimkan, protokol DNHSaWNew menghasilkan sebesar 87%, dan protokol DNHSaW sebesar 71%. Bila dilakukan selisih antara protokol DNHSaWNew dengan protokol DNHSaW, maka hasil selisih yang diperoleh untuk delivery ratio terhadap pengaruh jumlah paket data sebesar 16%.

Gambar 4.16 dan Gambar 4.17 menampilkan pengaruh jumlah node dan jumlah paket data terhadap tingkat *delivery probability* yang dihasilkan dari pengujian. Terlihat tren yang baik dihasilkan oleh protokol DNHSaWNew dibandingkan dengan protokol DNHSaW.



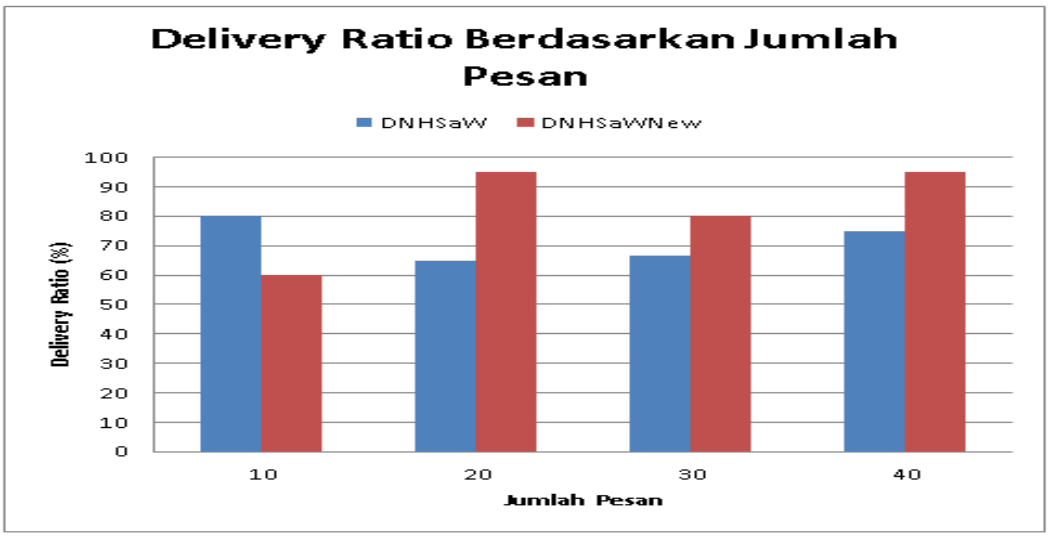
Gambar 4.16. Grafik Delivery Probability Berdasarkan Perubahan Jumlah Node

Keterangan gambar 4.16:

PCT-1 : pedestrian = 70; cars = 30; tram = 6

PCT-2 : Pedestrian = 6; cars = 70; Tram = 30

PCT-3 : Pedestrian = 30; cars = 6; Tram = 70



Gambar 4.17. Grafik Delivery Probability Berdasarkan Jumlah Paket Data

Pada gambar 4.17 terlihat protokol DNHSaWNew mendominasi hasil yang baik bila dibandingkan dengan protokol DNHSaW. Perbedaan yang terjadi antara protokol DNHSaWNew dan DNHSaW yaitu pada kategori perubahan node dengan jumlah node *pedestrian* sebesar 70 *node*, *cars* sebesar 30 *nodes* dan *tram* sebesar 6 *node*(PCT-1). Pada PCT-1 hanya terpaut sedikit lebih kecil yang dihasilkan oleh protokol DNHSaW dibandingkan dengan DNHSaWNew yaitu sekitar 20%. Sedangkan pada gambar 4.17 bila dilihat dari perubahan jumlah paket data yang dikirimkan dari pengirim kepada penerima terdapat selisih perbedaan yaitu pada nilai *variabel* paket data 20, 30 dan 40.

Gambar 4.18. Perbandingan Delivery Ratio Dilihat Dari Parameter Performa Node

Pada gambar 4.18 terlihat pengaruh parameter histori node dan kecepatan terhadap distribusi pesan yang beredar di dalam jaringan delay tolerant network. Bilai dilhat perbandingan masing-masing parameter performa node mempunyai nilai yang berbeda. Dan yang paling berpengaruh terhadap perubahan distribusi pesan yaitu parameter histori node. Perbandingan lain yang terlihat dari gambar 4.18 yaitu pengaruh performa keseluruhan berupa parameter histori, frekuensi dan kecepatan bila dibandingkan dengan performa lainnya, mengalami peningkatan

yang signifikan. Hal ini terlihat performa node keseluruhan mendominasi terhadap performa node lainnya.

4.3.2. Analisis Pengaruh Overhead Ratio

Overhead ratio dihasilkan dari releksi *transmission cost* yang diperoleh dari paket data yang beredar didalam jaringan. Perhitungan *overhead ratio* diperoleh dari jumlah peket data yang di-*relay* dikurangi dengan jumlah paket data yang diterima oleh node tujuan, lalu dibagi dengan jumlah data yang diterima oleh node penerima.

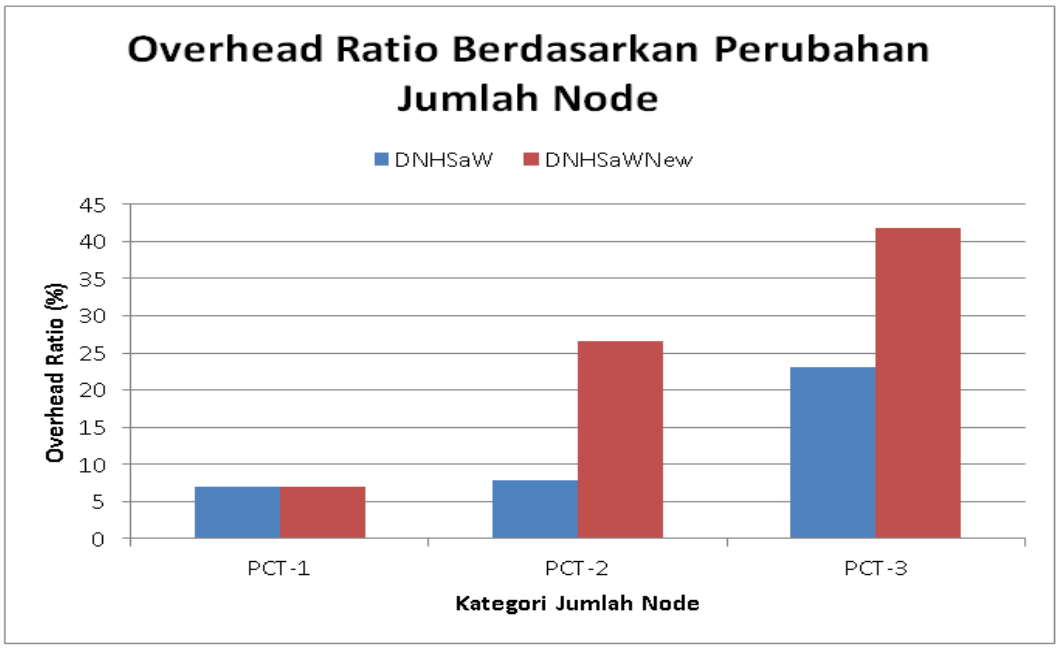
Tabel 4.5 menunjukkan data hasil pengujian skenario dari perubahan variasi jumlah node pedestrian, cars dan tram serta perubahan jumlah paket data yang dikirimkan dari node sumber kepada node penerima terhadap overhead yang dihasilkan di dalam jaringan.

Tabel 4.5. Hasil Pengujian Overhead Ratio

Skenario		Overhead Ratio	
		Protokol DNHSaW	Protokol DNHSaWNew
Perubahan Node	NODE		
	Pedestrian = 70; Cars = 30; Tram = 6	7	7
	Pedestrian = 6; cars = 70; Tram = 30	7.8	26.5
	Pedestrian = 30; cars = 6; Tram = 70	23	41.8889
Jumlah Pesan	PESAN		
	10	7	7
	20	5.8462	18.6316
	30	5.7	17.25
	40	6.6	15.1579

Pada tabel 4.5 terlihat hasil nilai *overhead ratio* pada protokol DNHSaWNew dan DNHSaW dengan perbandingan yang signifikan. Pada protokol DNHSaWNew terlihat hasil pengujian *overhead ratio* tidak seperti yang dihasilkan pada *delivery ratio* yaitu cenderung lebih besar daripada protokol DNHSaW. Hal ini terlihat dari rata-rata *overhead ratio* yang dihasilkan oleh protokol DNHSaWNew pada kategori perubahan jumlah node sebesar 25,12%,

dan protokol DNHSaW sebesar 12,6%. Sedangkan pada kategori perubahan jumlah paket data yang terkirim, pada protokol DNHSaW rata-rata *overhead ratio* yang dihasilkan sebesar 6,28% dan protokol DNHSaWNew sebesar 14,5%.



Gambar 4.19. Grafik Overhead Ratio Berdasarkan Jumlah Node

Keterangan gambar 4.18:

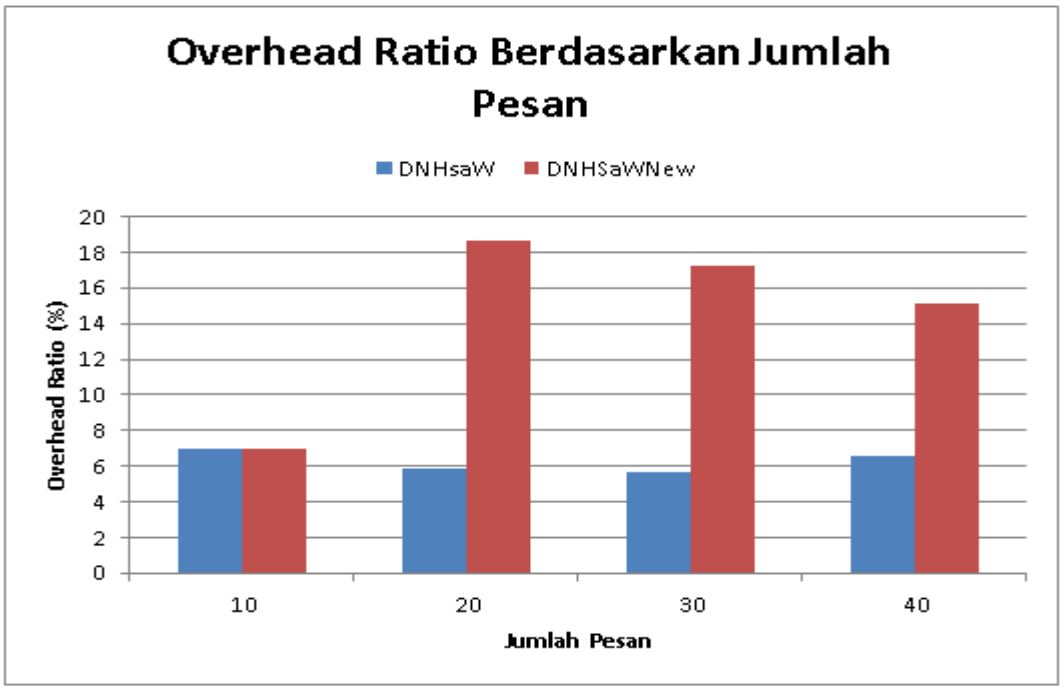
PCT-1 : pedestrian = 70; cars = 30; tram = 6

PCT-2 : Pedestrian = 6; cars = 70; Tram = 30

PCT-3 : Pedestrian = 30; cars = 6; Tram = 70

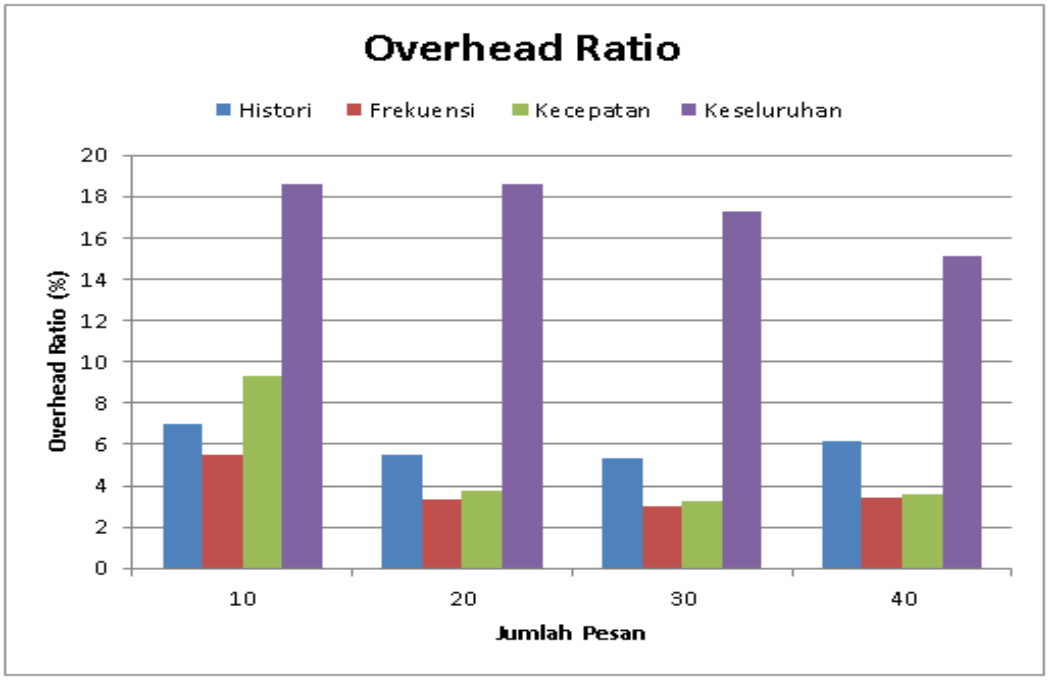
Pada gambar 4.19 terlihat hubungan pengaruh jumlah node terhadap hasil pengujian *overhead ratio* pada protokol DNHSaW dan protokol DNHSaWNew. Protokol DNHSaWNew memiliki *overhead ratio* cenderung lebih tinggi bila dibandingan dengan protokol DNHSaW. Hal ini disebabkan pengaruh performa node dari histori node, frekuensi dan kecepatan node. Sedangkan pada protokol DNHSaW hanya dipengaruhi oleh performa histori node. Performa node yang dihasilkan pada protokol DNHSaWNew tidak bisa berdiri sendiri, sebab harus dilihat dari ketiga variabel performa tersebut. Apabila suatu node memiliki

performa node yang tinggi, maka selain jumlah distribusi pesan yang yang dikirimkan besar, maka nilai overhead ratio yang dihasilkan juga lebih besar.



Gambar 4.20. Grafik Overhead Ratio Berdasarkan Jumlah Pesan

Pada gambar 4.20 menunjukkan pengaruh hubungan antara jumlah paket data yang bervariasi yaitu 10, 20, 30, dan 40 yang dikirimkan dari node sumber ke node tujuan. Protokol DNHSaW dan DNHSaWNew mempunyai pola grafik menanjak atau naik dimulai dari dengan pengujian pada jumlah paket 10. Untuk protokol DNHSaW tren kenaikan nilai *overhead ratio* yang dihasilkan tidak terlalu tinggi peningkatannya kurang lebih sekitar 8%, sedangkan pada protokol DNHSaWNew tren kenaikan nilai *overhead ratio* yang dihasilkan mencapai kurang lebih sekitar 18%. Pola meningkatnya *overhead ratio* yang dihasilkan oleh protokol DNHSaWNew diperoleh dari jumlah paket data yang diberikan sebesar 20, 30 dan 40. Pengaruh pola meningkatnya *overhead ratio* yang terjadi pada protokol DNHSaWNew bisa disebabkan oleh performa *node* dari sisi histori node, frekuensi *node*, kecepatan *node* atau bahkan dari pengaruh perbandingan *destination message history* apabila nilai performa node tidak baik.



Gambar 4.21. Perbandingan Overhead Ratio Dilihat Dari Parameter Performa Node

Pada gambar 4.21 perbandingan performa keseluruhan yang terdiri dari parameter histori, frekuensi dan kecepatan node mempengaruhi terhadap nilai overhead ratio yang dihasilkan. Tren yang terjadi untuk pengiriman pesan antara 10, 20, 30, 40 terlihat mengalami penurunan meskipun nilai yang dihasilkan masih belum baik dengan nilai performa yang lainnya seperti performa histori, frekuensi dan kecepatan.

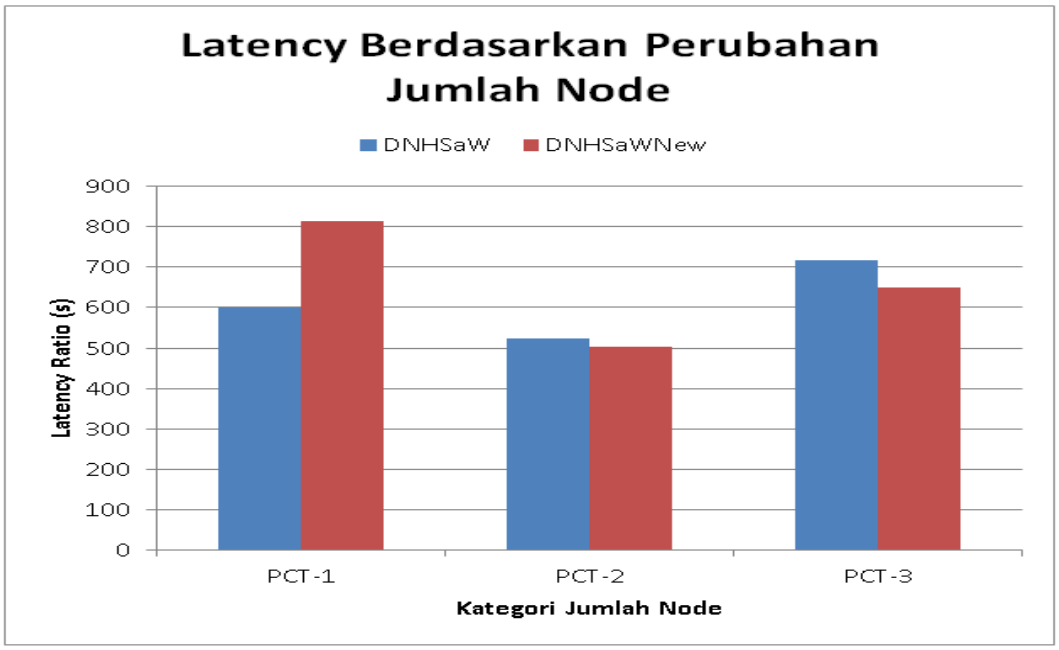
4.3.3. Analisis Pengaruh Latency

Parameter terakhir yang di analisa dalam pengujian dari penelitian ini yaitu pengaruh jumlah *node* dan paket data terhadap *latency* yang dihasilkan dalam jaringan. Latency yang dihasilkan diperoleh dari jumlah keseluruhan waktu paket diterima dikurangi dengan waktu paket dibuat. Hasil data pengujian *latency* yang diperoleh dari pengujian disajikan ke dalam Tabel 4.5.

Tabel 4.6. Hasil Pengujian Latency

Skenario		Latency	
		Protokol DNHSaW	Protokol DNHSaWNew
Perubahan Node	NODE		
	Pedestrian = 70; Cars = 30; Tram = 6	600.175	814.4333
	Pedestrian = 6; cars = 70; Tram = 30	522.51	503.75
	Pedestrian = 30; cars = 6; Tram = 70	716.3	648.5222
Jumlah Pesan	PESAN		
	10	600.175	814.4333
	20	1379.315	1025.3474
	30	1216.875	751.3
	40	1536.283	1075.4605

Pada tabel 4.5 hasil pengujian *latency* pada protokol DNHSaW dan DNHSaWNew yang dipengaruhi oleh jumlah perubahan node dan jumlah paket data terdapat perbedaan signifikan. Rata-rata jumlah *latency* yang dihasilkan oleh protokol DNHSaW pada sebesar 612,995 detik dan DNHSaWNew sebesar 655,569 detik. Latency yang dihasilkan oleh protokol DNHSaWNew lebih besar bila dibandingkan dengan protokol DNHSaW. Sedangkan pada perubahan jumlah paket data dengan jumlah paket data 10, 20, 30 dan 40, rata-rata *latency* protokol DNHSaW yang dihasilkan sebesar 1065,45 detik dan protokol DNHSaWNew sebesar 916,63 detik.



Gambar 4.22. Grafik Latency Berdasarkan Perubahan Jumlah Node

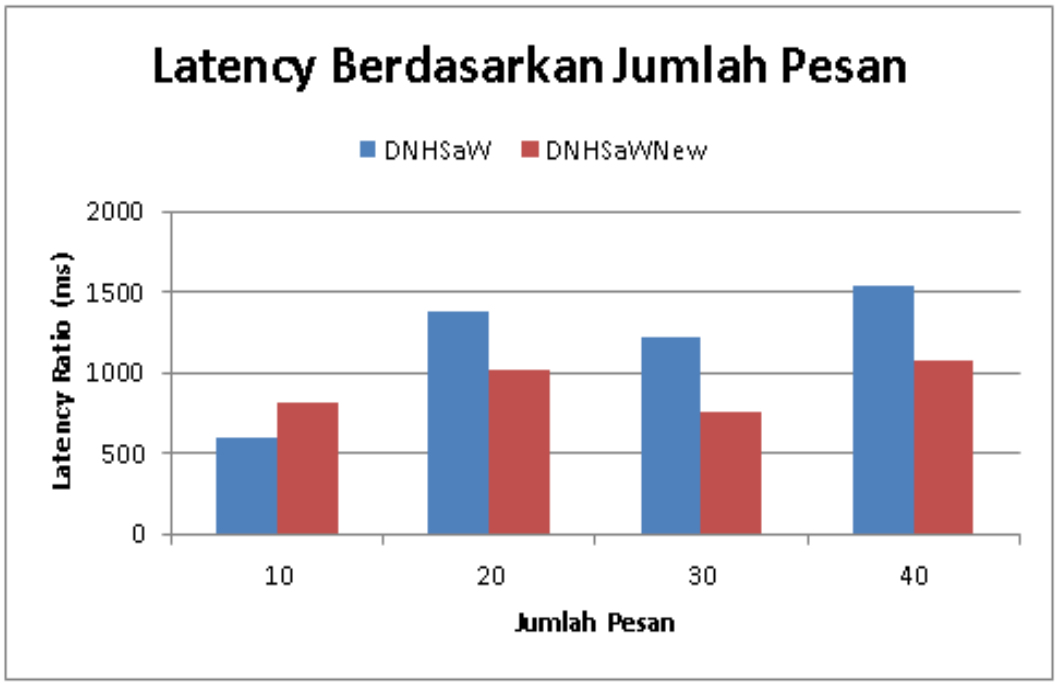
Keterangan gambar 4.18:

PCT-1 : pedestrian = 70; cars = 30; tram = 6

PCT-2 : Pedestrian = 6; cars = 70; Tram = 30

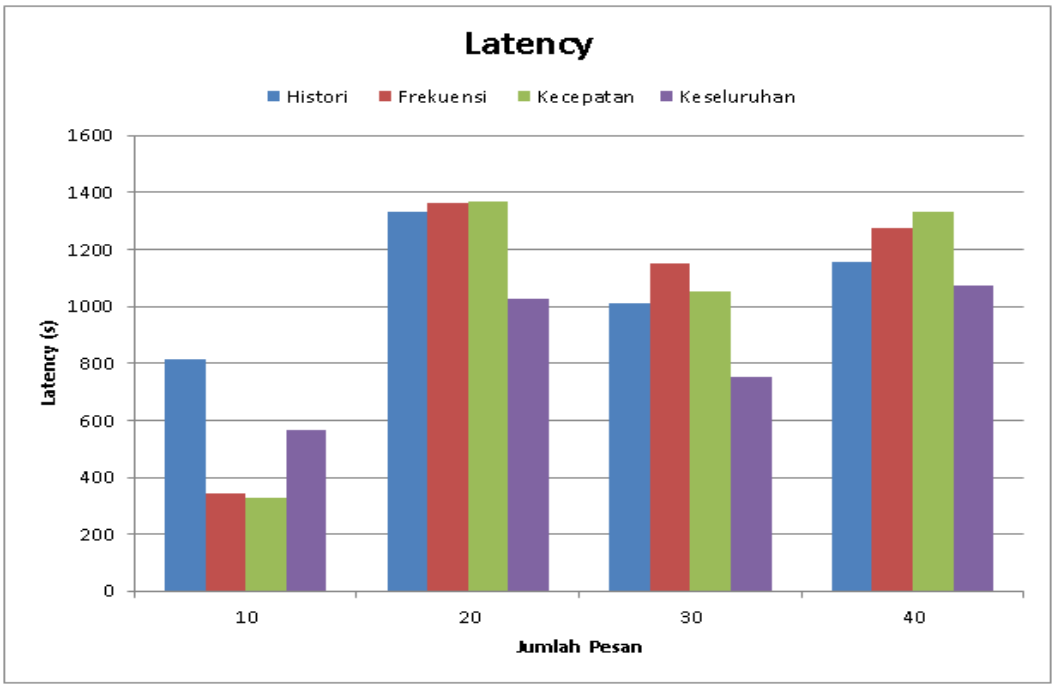
PCT-3 : Pedestrian = 30; cars = 6; Tram = 70

Pada gambar 4.22 terlihat pengaruh perubahan jumlah node terhadap *latency* yang dihasilkan oleh protokol DNHSaW dan DNHSaWNew. Pada kategori jumlah node PCT-1 protokol DNHSaW menghasilkan *latency* yang lebih kecil dibandingkan dengan DNHSaWNew. Hal ini bisa disebabkan waktu paket yang diterima oleh node penerima lebih lama disebabkan oleh nilai performa node yang kecil, sehingga paket lebih lama terkirim. Pada kategori PCT-2 dan PCT-3 *latency* yang dihasilkan oleh protokol DNHSaWNew lebih kecil dibandingan dengan protokol DNHSaW. Penurunan tingkat *latency* yang dihasilkan oleh protokol DNHSaWNew pada kategori PCT-2 dan PCT-3 disebabkan tingkat performa *node* dan *destination message history* yang baik, sehingga pesan yang dikirimkan oleh *node* pengirim sampai kepada node penerima dengan waktu yang tidak relatif lama.



Gambar 4.23. Grafik Latency Berdasarkan Jumlah Paket Data

Pada gambar 4.23 pengaruh jumlah paket data yang berbeda antara 10, 20, 30 dan 40 terhadap nilai latency pada protokol DNHSaWNew lebih kecil dibandingkan dengan protokol DNHSaW. Tren menurunnya nilai *latency* yang dihasilkan oleh protokol DNHSaW dibandingkan dengan protokol DNHSaW terlihat dari perubahan jumlah paket dari jumlah paket data 20 menuju paket data 40.



Gambar 4.24. Perbandingan Latency Dilihat Dari Parameter Performa Node

Pada gamrbar 4.24 terlihat perbandingan latency yang dihasilkan dari performa keseluruhan terdiri dari parameter histori, frekuensi dan kecepatan terhadap performa node yang hanya berupa histori, frekuensi dan kecepatan, memiliki tingkat latency yang lebih rendah. Pada pengiriman pesan sebanyak 10, 20, 30, 40 pada performa keseluruhan lebih stabil turun dibandingan dengan performa kecepatan. Salah satu terhadap menurunnya latency salah satunya yaitu peningkatan *delivery ratio* yang dihasilkan, sehingga pesan yang dikirimkan lebih dapat sampai daripada performa node lainnya.

[Halaman ini sengaja dikosongkan]

BAB 5
KESIMPULAN DAN SARAN

Bab ini memaparkan kesimpulan yang diambil berdasarkan pada penelitian yang telah dilakukan pada bab sebelumnya. Saran juga diuraikan tentang hal-hal yang perlu dipertimbangkan untuk pengembangan penelitian lebih lanjut. Penjelasan yang lebih terperinci tentang hal-hal tersebut diuraikan pada sub-bab berikut.

5.1. Kesimpulan

Adapun kesimpulan yang diambil berdasarkan dari hasil penelitian yang telah dilakukan dan analisa metode yang diusulkan adalah sebagai berikut:

1. Kontribusi yang diusulkan pada penelitian ini berupa performa histori node, frekuensi node, dan kecepatan serta *destination message history* mempunyai dampak yang dapat meningkatkan *delivery ratio*, sehingga terbukti sesuai dengan tujuan dari penelitian. Hal ini terlihat dari hasil pengujian berupa skenario pengujian perubahan jumlah node dan jumlah paket data.
2. Peningkatan *delivery ratio* yang dihasilkan oleh protokol DNHSaWNew tidak terjadi atau hasil yang diperoleh sama seperti protokol DNHSaW pada kategori skenario pengujian perubahan jumlah node. Sedangkan pada kategori skenario pengujian jumlah paket data, protokol DNHSaWNew terjadi peningkatan sebesar 10,83% dari protokol DNHSaW. Peningkatan lain yang dihasilkan oleh protokol DNHSaWNew yaitu pada *overhead ratio* sebesar 12,59% untuk kategori perubahan jumlah node dan 8,22% untuk kategori variasi jumlah paket data terhadap protokol DNHSaW. Pada percobaan masing-masing skenario terlihat hasil *delivery ratio* yang dihasilkan oleh metode baru yang diusulkan berupa protokol DNHSaWNew lebih baik daripada protokol DNHSaW.
3. Peningkatan *delivery ratio* penurunan *latency* pada protokol DNHSaWNew dari metode baru yang diusulkan tidak sejalan dengan

overhead ratio dihasilkan pada jaringan delay tolerant network saat pengujian berlangsung.

5.2. Saran

Dari penelitian yang telah dilakukan diperoleh bahwa metode baru berupa penambahan variabel pada perfoma node histori, frekuensi node, dan kecepatan serta *destination message history* dapat meningkatkan kinerja *delivery ratio* yang dihasilkan pada saat proses pengujian berlangsung. Selain itu ada beberapa saran yang dapat digunakan untuk penelitan selanjutnya dengan tujuan untuk mengurangi kekurangan yang dihasilkan oleh penelitian ini yaitu berupa:

1. Perlu diteliti lebih lanjut mengenai meningkatnya *overhead ratio* dan *latency* yang diperoleh pada saat pengujian, sehingga *delivery ratio* yang dihasilkan sejalan dengan berkurangnya overhead ratio.
2. Perlu diteliti lebih lanjut untuk menerapkan metode *broadcast message information* bagi pesan yang sudah terkirim dan sampai kepada node destination, sehingga nantinya bisa mengurangi overhead ratio yang terjadi di dalam jaringan.

DAFTAR PUSTAKA

Fall, K. (2003a), "A Delay Tolerant Networks Architecture for Challenged Intenets", *ACM SIGCOM*.

Zhang, X., Neglia, G., Kurose, J., Towsley, D., (2006), "Performance Modelling of Epidemic Routing", *Science Direct*.

Amin, V. dan Becker, D. (2000), *Epidemic Routing for Patially Connected Ad-hoc Networks*, Technical Report CS-2000006, Duke University.

Burgess, J., Gallagher, B., Jensen, D., Levene, B. (2000), "MaxProp: Routing for Vehicle-Based Disruption-Tolerant Networks", *Procedding of IEEE INFOCOM*.

Thrasyvoulos S., Konstantinos P., Caugligi S. R., (2005), "Spray and Wait: An Efficient Routing Scheme for Intermitently Connected Mobile Networks", *ACM*.

Patel B., Dave K., Pandya V., (2014), "Spray and Wait Routing Protocol in Delay Tolerant Networks", *International Journal of Emerging Technology and Advanced Engineering*.

Vittawus P., Chalermek I., Kutilda R., (2011), "DNH-SaW: The Different Neighbor-History Spray and Wait Routing Scheme for Delay Tolerant Networks", *International Symposium on Intellegent Signal Processing and Communication Systems*.

Evan, P.C.J. (2006), *Practical Routing in Delay-Tolerant Networks*, Thesis Master, University of Waterloo, Canada.

Johnson, D.B., Maltz,. D.A., Broch, J. (2001), "DSR: The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad-hoc Networks", *Addison-Wesley Longman Publisher.*

Keranen, A., Ott, J., Karkkainen, T., (2009), "The ONE Simulator for DTN Protocol Evaluation", *Proceddings of the 2nd International Conference on Simulation Tools and Techniques, Belgium.*

Wang, G., Wang, B., Gao, Y., (2010), "Dynamic Spray and Wait Routing Algorithm with Quality of Node in Delay Tolerant Networks", *International Conference of Communication and Mobile Computing*, University Anhui, China.

Jain, S., Fall, K., Patra, R. (2004), "Routing in a Delay Tolerant Networks", *SIGCOM*, Portland, Oregon, USA.

Pan, D. (2012), "An Improved Spray and Wait with Probability Choice Routing for Opportunistic Networks", *Journal of Networks, Vol.7, No.9.*

Song, L., Kotz F.D (2007), "Evaluating Opportunistic Routing Protocols with Large Realistic Contact Traces", *CHANS*.

Spryopoulos, T., Psounis, K., Raghavendra, C.S (2008), “Efficient Routing in Intermittenly Connected Mobile Networks: The Single-Copy Case”, *IEEE Transaction on Networing, Vol. 16, No.1*

Keranen, A. (2008), *Opportunistic Network Enviroment Simulator,* Departement of Communication and Networking, Helsinki University of Technology.

BIOGRAFI PENULIS

Tri Ramadhani, dilahirkan di Situbondo, merupakan anak ketiga dari tiga bersaudara. Penulis telah menempuh pendidikan formal yaitu SD Negeri 1 Panarukan, SLTP Negeri 5 Situbondo, SMU Negeri 2 Situbondo, selanjutnya pada tahun 2004 penulis menempuh pendidikan D4 pada Jurusan Teknik Telekomunikasi di Politeknik Negeri Surabaya Institut Teknologi Sepuluh Nopember Surabaya (PENS-ITS), kemudian melanjutkan studi S2 di jurusan Teknik Informatika Fakultas Teknologi Informatika ITS Surabaya. Penulis dapat dihubungi melalui email: tri.ramadhani2014@gmail.com.