TUGAS AKHIR – KI141502

DETEKSI *MALICIOUS NODE* PADA *ZONE ROUTING PROTOCOL* DI JARINGAN *MOBILE ADHOC NETWORK*

SETIYO ADIWICAKSONO
NRP 5113100020

Dosen Pembimbing I
Henning Titi Ciptaningtyas, S.Kom., M.Kom.

Dosen Pembimbing II
Dr. Eng. Radityo Anggoro, S.Kom.,M.Sc

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

TUGAS AKHIR – KI141502

DETEKSI *MALICIOUS NODE* PADA *ZONE ROUTING PROTOCOL* DI JARINGAN *MOBILE ADHOC NETWORK*

SETIYO ADIWICAKSONO
NRP 5113100020

Dosen Pembimbing I
Henning Titi Ciptaningtyas, S.Kom., M.Kom.

Dosen Pembimbing II
Dr. Eng. Radityo Anggoro, S.Kom.,M.Sc

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

[Halaman ini sengaja dikosongkan]

UNDERGRADUATE THESES – KI141502

MALICIOUS NODE DETECTION ON ZONE ROUTING PROTOCOL IN MOBILE ADHOC NETWORK

SETIYO ADWICAKSONO
NRP 5113100020

Supervisor I
Henning Titi Ciptaningtyas, S.Kom., M.Kom.

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

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2017

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

DETEKSI *MALICIOUS NODE* PADA *ZONE ROUTING PROTOCOL* DI JARINGAN *MOBILE ADHOC NETWORK*

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Rumpun Mata Kuliah Komputasi Berbasis Jaringan
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh
SETIYO ADWICAKSONO
NRP : 5113 100 020

Disetujui oleh Dosen Pembimbing Tugas Akhir:

1. Henning Titi C. , S.Kom., M.Kom. .....
   NIP: 19840708 201012 2 004 (Pembimbing 1)

2. Dr. Eng. Radityo Anggoro, S.Kom., M.Sc .....
   NIP: 19841016 200812 1 002 (Pembimbing 2)

SURABAYA
JUNI, 2017

[Halaman ini sengaja dikosongkan]

DETEKSI *MALICIOUS NODE* PADA *ZONE ROUTING PROTOCOL* DI JARINGAN *MOBILE ADHOC NETWORK*

Nama Mahasiswa	**: Setiyo Adiwicaksono**
NRP	**: 5113100020**
Jurusan	**: Teknik Informatika FTIF-ITS**
Dosen Pembimbing 1	**: Henning Titi C. , S.Kom., M.Kom.**
Dosen Pembimbing 2	**: Dr. Eng. Radityo Anggoro, S.Kom.,M.Sc**

Abstrak

Mobile Ad-Hoc Network (MANET) adalah sebuah lingkungan jaringan untuk berkomunikasi yang terdiri dari beberapa node yang bergerak secara bebas dan dinamis sehingga posisi masing-masing node tidak tetap. Hal ini menyebabkan informasi rute komunikasi antar node dapat berubah-ubah karena pergerakan node yang dinamis. MANET tidak memiliki topologi yang tetap dan posisi node yang pasti. Hal ini menyebabkan sulitnya mendeteksi node node lain disekitarnya aman atau tidak untuk dijadikan media untuk berkomunikasi.

Node-node yang membawa sifat dan perilaku yang berbahaya bagi jaringan disebut malicious node. Malicios node ini mampu masuk ke dalam jaringan dengan cara menyamar atau mengaku dirinya sebagai node normal yang aman untuk dijadikan rute pengiriman pesan. Salah satu sifat malicious node adalah blackhole attack. Blackhole sendiri mempunyai sifat yaitu ketika ada paket atau pesan masuk dan sampai kepada dirinya, maka paket dan pesan tersebut akan di drop sehingga paket dan pesan tersebut tidak akan pernah sampai ke tujuan. Pada tugas akhir ini, akan dilakukan pendeteksian dan pencegahan blackhole pada jaringan MANET.

Metode yang digunakan adalah dengan membuat sebuah daftar alamat node-node berbahaya dan memanfaatkan

Intra-Zone Routing Protocol (IARP) dan Inter-Zone Routing Protocol (IERP) yang ada pada protocol Zone Routing Protocol (ZRP). Pada IARP, ketika pengenalan tetangga didalam zonannya, maka sebelum ditambahkan pada daftar tetangga, di cek terlebih dahulu apakah alamat tersebut ada pada daftar alamat node berbahaya. Lalu pada IERP, ketika penyambungan rute ketika node berada diluar zona, sebelum disambungkan, di cek terlebih dahulu apakah alamat node tersebut ada pada daftar alamat node berbahaya.

Tujuan pembuatan tugas akhir ini adalah untuk mencegah serangan malicious node, dalam hal ini blackhole, sehingga paket yang dikirimkan dari node sumber bisa tersampaikan dengan aman ke node tujuan dan terhindar dari serangan malicious node.

Kata kunci: MANET, Network Simulator, NS-2, ZRP, Malicious Node.

MALICIOUS NODE DETECTION ON ZONE ROUTING PROTOCOL IN MOBILE ADHOC NETWORK

Nama Mahasiswa	**: Setiyo Adiwicaksono**
NRP	**: 5113100020**
Jurusan	**: Teknik Informatika FTIF-ITS**
Dosen Pembimbing 1	**: Henning Titi C. , S.Kom., M.Kom.**
Dosen Pembimbing 2	**: Dr. Eng. Radityo Anggoro, S.Kom.,M.Sc**

Abstract

Mobile Ad-Hoc Network (MANET) is a network environment to communicate consisting of several nodes that move freely and dynamically so that the position of each node is not fixed. This causes the communication route information between the nodes to vary due to the dynamic movement of its node. MANET does not have a fixed topology and a definite node position. This causes the difficulty of detecting other nodes around the node is safe or not to be a medium to communicate.

Nodes that carries a harmful properties and behaviors to the network are called malicious nodes. Malicious nodes are able to enter the network by acting like a normal node that safe to be used as a delivery route of messages. One kind of the malicious node is the blackhole attack. When there is a packet or message recieved to the blackhole, the blackhole node will drop the package and the message so that the packet and the message will never get to the destination. In this final project, blackhole detection and prevention of MANET network will be performed.

The method used is to create a list of malicious nodes address and utilize the Intra-Zone Routing Protocol (IARP) and Inter-Zone Routing Protocol (IERP) that exist in the Zone Routing Protocol (ZRP). On IARP, when neighbor introductions are inside the zone, before being added to the neighbor list, check first whether the address is on the list of

blacklist or not. Then on IERP, when connecting the routes when nodes are outside the zone, before connecting, first check whether the node's address is on the list of blacklist or not.

The purpose of this final task is to prevent malicious node attacks, in this case blackhole, so packets sent from the source node can be safely delivered to the destination node and protected from malicious node attacks.

Keywords: MANET, Network Simulator, NS-2, ZRP, Malicious Node

KATA PENGANTAR

بِسۡمِ ٱللَّهِ ٱلرَّحۡمَٰنِ ٱلرَّحِيمِ

Alhamdulillahirabbil'alamin, segala puji bagi Allah SWT, yang telah melimpahkan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul **"DETEKSI *MALICIOUS NODE* PADA *ZONE ROUTING PROTOCOL* DI JARINGAN *MOBILE ADHOC NETWORK"*.** Dengan pengerjaan Tugas Akhir ini, penulis bisa belajar lebih banyak untuk memperdalam dan meningkatkan apa yang telah didapatkan penulis selama menempuh perkuliahan di Teknik Informatika ITS. Dengan Tugas Akhir ini penulis juga dapat menghasilkan suatu implementasi dari apa yang telah penulis pelajari.

Selesainya Tugas Akhir ini tidak lepas dari bantuan dan dukungan beberapa pihak. Sehingga pada kesempatan ini penulis mengucapkan syukur dan terima kasih kepada:

1. Allah SWT dan Nabi Muhammad SAW.
2. Keluarga di Jakarta khususnya Bapak, Ibu dan Kakak yang telah memberikan dukungan moral dan material serta do'a yang tak terhingga untuk penulis. Serta selalu memberikan semangat dan motivasi pada penulis dalam mengerjakan Tugas Akhir ini.
3. Ibu Henning Titi C., S.Kom., M.Kom selaku pembimbing I yang telah membantu, membimbing, dan memotivasi penulis dalam menyelesaikan Tugas Akhir ini.
4. Bapak Dr. Eng. Radityo Anggoro, S.Kom.,M.Sc selaku pembimbing II dan juga koordinator Tugas Akhir yang juga telah membantu, membimbing, dan memotivasi kepada penulis dalam mengerjakan Tugas Akhir ini.
5. Bapak Dr. Darlis Herumurti, S.Kom., M.Kom. selaku Kepala Jurusan Teknik Informatika ITS.

6. Para Bapak/Ibu Dosen Teknik Informatika yang telah memberikan ilmunya.
7. Teman-teman administrator laboratorium Arsitektur dan Jaringan Komputer(AJK), Wicak, Uul, Daniel, Nindy, Zaza, Risma, Syukron, Oing, Fatih, Ambon, Thoni, Bebet, Vivi, Awan, Fuad, Didin, dan Satria.
8. Teman-teman Diary of Silver, Bagus, Kevin, Arvi, Idang, Sum, Bawanta, Nyoman, yang berperan mengobati dikala stress dan pusing melanda saat pengerjaan Tugas Akhir ini
9. Teman-teman angkatan 2013 yang yang telah berbagi ilmu, dan memberi motivasi kepada penulis.
10. Serta semua pihak yang yang telah turut membantu penulis dalam menyelesaikan Tugas Akhir ini.

Penulis menyadari bahwa Tugas Akhir ini masih memiliki banyak kekurangan. Sehingga dengan kerendahan hati, penulis mengharapkan kritik dan saran dari pembaca untuk perbaikan ke depannya.

Surabaya, Juni 2017

DAFTAR ISI

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

DAFTAR KODE SUMBER

BAB I
PENDAHULUAN

1.1 Latar Belakang

Informasi adalah sebuah hal mendasar yang sangat dibutuhkan pada masa kini. Karena sangat dibutuhkannya informasi, maka kecepatan pengiriman informasi juga merupakan hal yang penting dalam proses pengiriman informasi. Oleh Karena itu, kebutuhan koneksi internet pun menjadi hal yang utama dalam pengirmian data. Tetapi, pada daerah-daerah tertentu, atau dalam kondisi-kondisi tertentu, seperti pada kondisi daerah yang terkena bencana, daerah yang sedang pada kondisi peperangan, atau di daerah terpencil yang masih belum terdapat koneksi intenet, koneksi internet terkadang sulit untuk didapatkan. Oleh Karena itu, pada kondisi tersebut, *Mobile Ad Hoc Network* (MANET) bisa dijadikan salah satu opsi yang solutif untuk memecahkan masalah tersebut.

MANET sendiri adalah jaringan *Ad Hoc* yang berada pada perangkat mobile. Jaringan *Ad Hoc* sendiri adalah jaringan komputer sederhana yang berbasis *peer-to-peer* yang dibuat secara cepat, singkat, dan otomatis mengkonfigurasikan dirinya sendiri, untuk menghubungkan minimal 2 buah perangkat agar bisa saling berkomunikasi dan bertukar data secara langsung, tanpa perlu perangkat lain sebagai perantara. Implementasi MANET ini telah banyak digunakan pada kegiatan sehari-hari, tetapi salah satu masalah dalam MANET ini adalah masalah keamanan data dan jaringan dari MANET. Karena perantara yang berubah-ubah, maka sangat mudah sekali perusak atau hacker masuk kedalam jaringan.

Namun, dalam setiap jaringan pasti terdapat serangan. Serangan yang terjadi sangat mengganggu jalnnya jaringan dalam melakukan pengiriman. Ada beberapa jenis serangan yang beberapa tahun terakhir sedang didalami dan dipelajari untuk dicari pencegahannya. Contohnya, *Flooding Attack*, *Blackhole Attack*,

Wormhole Attack, dan masih banyak lagi jenis-jenis serangan pada jaringan *Ad Hoc*.

Pada jaringan *Mobile Ad Hoc Network*, ancaman atau serangan pada jaringan ini di bawa oleh node-node perantara yang ada pada jaringan. Node yang membawa informasi atau serangan pada jaringan disebut *Malicious Node*. Pada Tugas Akhir ini, penulis memilih *Malicious Node* yang bersifat *Blackhole*, sehingga skenario serangan berupa *Blackhole Attack*.

Blackhole Attack sendiri adalah salah satu serangan pada jaringan *Mobile Ad Hoc Network* yang bersifat menghilangkan packet data sebelum sampai tujuannya. *Malicious node* akan menyamarkan dirinya layaknya sebuah node normal. Ketika dideteksi sebagai node normal, maka akan dikirimi paket layaknya node normal juga. Tetapi paket tidak akan pernah sampai tujuan, karena sifat *Blackhole* yang menghilangkan paket, sehingga sebelum sampai tujuan, paket dihilangkan dan tidak akan pernah sampai ke tujuan.

Metode pengiriman paket pun ada berbagai macam, yang biasa kita sebut *Routing Protocol*. Salah satu *routing protocol* adalah ZRP, yang nantinya akan digunakan pada Tugas Akhir ini. *Zone Routing Protocol* yang selanjutnya akan disingkat ZRP adalah sebuah protocol MANET yang mengimplimentasikan protokol *Hybrid*, yaitu protocol yang menggabungkan kedua sifat protokol MANET, yaitu Proactive dan Reactive. Implementasi protokol ini adalah mengambil keuntungan dari reaktif dan proaktif dan digabungkan menjadi satu protokol bersifat *hybrid* yang diberi nama ZRP. Konsep ZRP adalah membagi node-node perantara dalam bentuk zona-zona seluas radius yang sudah ditentukan sebelumnya.

Pada studi sebelumnya, pencegahan *malicious node* lebih banyak diteliti pada protokol seperti DSDV yang bersifat proaktif atau pada protokol AODV yang bersifat reaktif [1] [2]. Pada Tugas Akhir ini, dengan mengunakan protokol MANET *Zone Routing*

Protocol (ZRP) yang bersifat *hybrid* (gabungan antara *proactive* dan *reactive routing protocol)*, akan dicari cara bagaimana sebuah koneksi MANET mampu mendeteksi node-node yang berbahaya (*malicious node*) dan mampu mencegah node tersebut untuk masuk ke jaringan. Jenis *Malicious Node* yang digunakan adalah *Blackhole*. *Malicious node* akan dideteksi berdasarkan kecocokan dengan daftar *Blacklist* yang sudah disediakan. Modifikasi akan dilakukan pada bagian protokol ZRP.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam Tugas Akhir ini adalah sebagai berikut:

1. Bagaimana pengaruh penambahan *malicious node* terhadap *Packet Delivey Ratio* pada jaringan MANET?
2. Bagaimana pengaruh penambahan *malicious node* terhadap *Delay* pada jaringan MANET?
3. Bagaimana pengaruh penambahan *malicious node* terhadap *Routing Overhead* pada jaringan MANET?

1.3 Batasan Masalah

Batasan dalam Tugas Akhir ini, yaitu:

1. *Routing Protocol* yang digunakan adalah ZRP
2. Uji Coba menggunakan *Network Simulator 2* (NS2).
3. Maksimal jumlah *malicious node* yang bisa didaftarkan pada daftar *blacklist* adalah 10

1.4 Tujuan

Tujuan pengerjaan Tugas Akhir ini adalah untuk mendeteksi dan menghindari *malicious node* dalam proses pengiriman data paket pada protokol ZRP di jaringan MANETs.

1.5 Manfaat

Manfaat dari pengerjaan Tugas Akhir ini adalah:

1. Dapat mengindentifikasi apakah sebuah node termasuk ke dalam node berbahaya atau tidak pada jaringan MANET
2. Dapat meningkatkan sekuritas pada koneksi MANET karena mampu menghindari serangan malicious node yang masuk ke dalam jaringan.
3. Dapat menghindari *malicious node* yang masuk ke dalam jaringan
4. Dapat meningkatkan *packet delivery ratio* pada protokol ZRP.

1.6 Metodologi

Tahapan-tahapan yang dilakukan dalam pengerjaan Tugas Akhir ini adalah sebagai berikut:

1. Penyusunan proposal Tugas Akhir.

   Tahap awal untuk memulai pengerjaan Tugas Akhir adalah penyusunan proposal Tugas Akhir. Penyusunan proposal Tugas Akhir dilaksanakan untuk merumuskan masalah serta melakukan penetapan rancangan dasar dari sistem yang akan dikembangkan dalam pelaksanaan Tugas Akhir ini.

2. Studi literatur

   Pada tahap ini dilakukan pemahaman informasi dan literatur yang diperlukan untuk tahap implementasi program. Tahap ini diperlukan untuk membantu memahami bagian mana yang harus dimodifikasi untuk bias mengatasi malicious node yang ada

3. Analisis dan perancangan perangkat lunak

   Tahap ini meliputi perancangan sistem berdasarkan studi literatur dan pembelajaran konsep teknologi dari perangkat lunak yang ada. Tahap ini mendefinisikan alur dari implementasi. Langkah-langkah yang dikerjakan juga didefinisikan pada tahap ini. Pada tahapan ini dibuat *prototype*

sistem, yang merupakan rancangan dasar dari sistem yang akan dibuat. Serta dilakukan desain fungsi yang akan dibuat yang ditunjukkan melalui *pseudocode*.

4. Implementasi perangkat lunak

   Implementasi perangkat lunak merupakan tahap membangun rancangan program yang telah dibuat. Pada tahap ini akan direalisasikan mengenai rancangan apa saja yang telah didefiniskan pada tahap sebelumnya. Fungsi yang ada pada tahap ini merupakan fungsi hasil implementasi dari tahap analisis dan perancangan perangkat lunak.

5. Pengujian dan evaluasi

   Pada tahap ini dilakukan uji coba pada data yang telah dikumpulkan. Tahap ini digunakan untuk mengevaluasi kinerja program serta mencari masalah yang mungkin timbul saat program dievaluasi serta melakukan perbaikan jika terdapat kesalahan pada program.

6. Penyusunan buku Tugas Akhir

   Pada tahap ini disusun buku yang memuat dokumentasi mengenai perancangan, pembuatan serta hasil dari implementasi perangkat lunak yang telah dibuat.

1.7 Sistematika Penulisan Laporan Tugas Akhir

Buku Tugas Akhir ini bertujuan untuk mendapatkan gambaran dari pengerjaan Tugas Akhir ini. Secara garis besar, buku Tugas Akhir terdiri atas beberapa bagian seperti berikut ini:

Bab I Pendahuluan

Bab yang berisi mengenai latar belakang, tujuan, dan manfaat dari pembuatan Tugas Akhir. Selain itu perumusan masalah, batasan masalah, metodologi yang digunakan, dan sistematika penulisan juga merupakan bagian dari bab ini.

Bab II Tinjauan Pustaka

Bab ini berisi penjelasan secara detail mengenai dasar-dasar penunjang dan teori-teori yang digunakan untuk mendukung pembuatan Tugas Akhir ini.

Bab III Analisis dan Perancangan Perangkat Lunak

Bab ini berisi tentang dasar dari algoritma yang akan diimplementasikan pada Tugas Akhir ini.

Bab IV Implementasi

Bab ini membahas mengenai implementasi dari rancangan yang telah dibuat pada bab sebelumnya.

Bab V Uji Coba Dan Evaluasi

Bab ini menjelaskan mengenai kemampuan perangkat lunak dengan melakukan pengujian kebenaran dan pengujian kinerja dari perangkat lunak yang telah dibuat sesuai dengan data yang diujikan.

Bab VI Kesimpulan Dan Saran

Bab ini merupakan bab terakhir yang menyampaikan kesimpulan dari hasil uji coba yang telah dilakukan dan saran untuk pengembangan perangkat lunak ke depannya.

BAB II
TINJAUAN PUSTAKA

Bab ini berisi penjelasan teori-teori yang berkaitan dengan algoritma yang diajukan pada pengimplementasian program. Penjelasan ini bertujuan untuk memberikan gambaran secara umum terhadap program yang dibuat dan berguna sebagai penunjang dalam pengembangan perangkat lunak.

2.1 NS 2 Network Simulator

NS 2 adalah simulator untuk jaringan yang biasa digunakan untuk penelitian dan simulasi jaringan. NS sudah mulai dikembangkan pada 1989 dan pengembangannya masih berlanjut hingga sekarang. NS 2 mendukung simulasi untuk TCP, *routing*, dan protokol *multicast* pada jaringan kabel maupun nirkabel. Pada Tugas Akhir ini, NS-2 digunakan untuk menjalankan skenario simulasi jaringan MANET. NS-2 digunakan karena peneliti jaringan saat ini banyak menggunakan NS-2 sebagai simulator untuk uji coba hasil penelitiannya berkaitan dengan jaringan.

NS 2 berjalan pada Bahasa C++ dan skrip *object-oriented Tcl* (OTcl). NS 2 dapat diinstal pada sistem operasi linux maupun windows [3].

Pada Tugas Akhir ini digunakan NS-2 versi 2.35 sebagai aplikasi simulasi jaringan skenario MANET yang dihasilkan oleh program default dari NS-2 yaitu *node-movement generator file*. NS-2 dijalankan pada sistem operasi Linux.

Tools yang disebut ‘setdest’ dikembangkan oleh CMU (Carnegie Mellon University) untuk menghasilkan pergerakan pada node dalam jaringan nirkabel. Pergerakan node dihasilkan dengan kecepatan gerak yang spesifik menuju lokasi acak atau lokasi spesifik yang berada dalam kawasan yang telah ditentukan. Ketika node tiba ke lokasi pergerakan, node tersebut bisa diatur untuk berhenti.

```
./setdest [-v version] [-n num_of_nodes] [-p
pausetime] [-M maxspeed] [-t simtime] [-x maxx] [-
ymaxy] > [outdir/movement-file]
```

Kode Sumber 2.1 Format Command Line "setdest"

Pengguna harus menambahkan kode program 'setdest' pada skenario tcl. Format kode program 'setdest' ditunjukkan pada Kode Sumber 2.1 dan keterangannnya ditunjukkan pada Tabel 2.1.

Tabel 2.1 Keterangan pada Command Line 'setdest'

Parameter	**Keterangan**
-v version	Versi 'setdest' simulator yang digunakan
-n num	Jumlah node dalam skenario
-p pausetime	Durasi ketika sebuah node tetap diam setelah tiba di lokasi pergerakan. Jika nilai ini diatur ke 0, maka node tidak akan berhenti ketika tiba di lokasi pergerakan dan akan terus bergerak
-M maxspeed	Kecepatan maksimum sebuah node . Node akan bergerak pada kecepatan acak dalam rentang [0, maxspeed]
-t simtime	Waktu simulasi
-x max x	Panjang maksimum area simulasi
-y max y	Lebar maksimum area simulasi

Perintah ‘setdest’ menghasilkan file output yang berisi jumlah node dan mobilitas yang akan digunakan dalam file Tcl selama simulasi. File output, selain mengandung skrip pergerakan, juga mengandung beberapa statistik lain tentang perubahan link dan rute. Untuk contoh dapat dilihat pada Kode Sumber 2.2.

```
./setdest -v 1 -n 50 -p 2.0 -M 20.0 -t 200 -x 500
-y 500 > scen-20-test
```

Kode Sumber 2.2 Hasil Output pada file ‘scen-20-test’

Untuk membuat skenario pergerakan node yang terdiri dari 50 node, bergerak dengan kecepatan maksimum 20.0 m/s dengan jeda rata-rata antar gerakan sebesar 2 detik, simulasi akan berhenti setelah 200 detik dengan batas topologi yang diartikan sebagai 500 x 500 meter², command line-nya terlihat seperti pada Kode Sumber 2.2.

File output ditulis ke "stdout" secara default. Di sini output disimpan ke dalam file "scen-20-test". File dimulai dengan posisi awal node dan berlanjut menetapkan pergerakan node seperti terlihat pada Kode Sumber 2.3.

```
$ns_ at 2.000000000000 "$node_(0) setdest
90.441179033457 44.896095544010” 1.373556960010
```

Kode Sumber 2.3 Contoh Command Line ‘setdest’

Command line pada Kode Sumber 2.3 dari 'scen-20-test' mendefinisikan bahwa node (0) pada detik ke 2.0 mulai bergerak ke arah tujuan (90.44, 44.89) dengan kecepatan 1.37 m/s. Command line ini dapat digunakan untuk mengubah arah dan kecepatan gerak dari *mobile node*. Arahan untuk *General Operations Director* (GOD) yang ada juga di file pergerakan node. Objek "GOD" digunakan untuk menyimpan informasi global tentang keadaan dari lingkungan jaringan dan node di sekitarnya. Namun isi dari file “GOD” tidak boleh diketahui oleh setiap bagian dalam simulasi.

Dalam simulasi di sini objek “GOD” hanya digunakan untuk menyimpan sebuah *array* dari jumlah hop terpendek yang perlukan untuk mencapai satu node ke node yang lain. Objek “GOD" tidak menghitung jumlah hop yang diperlukan selama simulasi berjalan, karena akan cukup memakan waktu. Namun “GOD” menghitung hop di akhir simulasi. Informasi yang dimuat ke dalam objek "GOD" dari pola pergerakan file terdapat pada baris perintah di Kode Sumber 2.4.

```
$ns_ at 899.642 "$god_ set-dist 23 46 2"
```

Kode Sumber 2.4 Command Line "GOD" pada 'scen-20-test'

Ini berarti bahwa jarak terpendek antara node 23 dan node 46 berubah menjadi 2 hop di waktu 899,642 detik. Program ‘setdest’ menghasilkan file pergerakan node menggunakan algoritma *random way point*. Perintah-perintah yang termasuk dalam program utama untuk memuat file-file ini dalam objek “GOD”. [4]

2.2 Mobile Ad Hoc Network

Mobile Ad Hoc Network (MANET) adalah jaringan nirkabel yang memiliki struktur jaringan yang abstrak, dan mampu mengatur konfigurasinya sendiri. MANET terdiri dari beberapa node yang mampu mengatur dirinya sendiri tanpa kontrol terpusat, dan masing-masing node dapat bergerak secara acak. Dengan kondisi tersebut, jaringan MANET menjadi salah satu jaringan yang fleksibel dan bagus digunakan pada kondisi-kondisi yang sulit jaringan seperti kondisi militer, bencana, dan fasilitas medis darurat. [5]

Pada Tugas Akhir ini, jenis jaringan yang diteliti adalah jaringan *ad-hoc* yang bersifat *mobile* yaitu MANET. MANET dipilih karena teknologi ini sedang banyak dikembangkan dan dilakukan riset untuk penggunaannya pada dunia nyata.

MANET bekerja pada lingkungan jaringan *routable* diatas *layer* jaringan *ad hoc*. MANET bekerja menggunakan metode *peer-to-peer* dan bergerak pada frekuensi radio (30MHz – 5GHz).

Pada dasarnya, MANET adalah jaringan nirkabel sementara yang tidak memiliki infrastruktur tetap untuk digunakan. Jadi pada MANET, topologi yang terbentuk sering berubah karena node seluler bergerak secara independen dan mengubah hubungan mereka ke nodus lainnya dengan sangat cepat. Setiap node *mobile* bertindak sebagai *router* dan meneruskan lalu lintas ke node lain dalam jaringan. Secara teori, dua simpul yang bergerak dan berada dalam jangkauan transmisi masing-masing dapat berkomunikasi secara langsung, jika tidak, node yang berada di antara keduanya harus dapat meneruskan paket agar berhasil berkomunikasi. Untuk meneruskan paket data dari sumber ke tujuan, setiap node dalam jaringan harus bersedia berpartisipasi dalam proses penyampaian paket data. Sebuah file tunggal dibagi menjadi sejumlah paket data dan kemudian paket data ini dikirim melalui jalur yang berbeda. Pada node tujuan, semua paket ini digabungkan secara berurutan untuk menghasilkan file asli [6]

2.3 Zone Routing Protocol

Zone Routing Protocol (ZRP) adalah protokol jaringan nirkabel yang *hybrid* yang menggunakan keuntungan dari *reactive* dan *proactive routing protocol* saat mengirimkan pesan pada jaringan. Protokol ini digunakan untuk mengurangi *control overhead* dari protokol routing proaktif, dan mengurangi *latency* dari protokol routing reaktif. ZRP dirancang untuk mempercepat pengiriman dan mengurangi proses dengan memilih cara memilih protokol yang paling efisien untuk digunakan pada seluruh rute. ZRP digunakan sebagai *routing protocol* pada Tugas Akhir ini. ZRP dipilih karena ZRP adalah salah satu jenis protocol yang bersifat *hybrid* yang mengoptimalkan kelemahan-kelemahan yang ada pada protokol reaktif dan protokol proaktif.

Secara garis besar, konsep ZRP adalah membangun zona di jaringan pada setiap node sehingga pada sebuah jaringan memungkinkan banyak sekali zona-zona yang dibangun oleh setiap node. Untuk node yang berada didalam wilayah geografis yang sudah di tentukan sebelumnya yang selanjutnya akan kita sebut *radius*, akan dikatakan bahwa node tersebut berada dalam zona routing node tersebut. Untuk routing yang berada didalam zona, digunakan pendekatan routing proaktif. Untuk routing yang berada diluar zona, digunakan pendekatan routing reaktif. Jadi dapat disimpulkan bahwa ZRP menggabungkan beberapa karakteristik protokol proaktif dan beberapa karakteristik protokol reaktif, dengan mempertahankan informasi zona intra secara proaktif dan informasi antar zona secara reaktif, menjadi satu untuk mendapatkan solusi yang lebih baik untuk jaringan MANET.

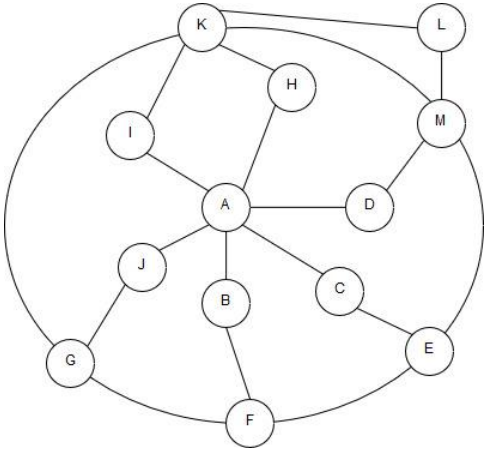
Secara umum, cara kerja ZRP adalah setiap node membuat sebah zone pada masing-masing node nya. Lalu untuk daerah yang masih didalam zona, maka rute menuju node tersebut akan ditemukan, namun ketika node tujuan berada di luar zona routing, maka akan dilakukan prosedur penemuan rute. Pada setiap zona pasti ada 1 node yang saling beririsan atau berada pada 2 zona yang berbeda. Besaran setiap zona tergantung pada *radius* yang didefinsikan pada jumlah hop (lompatan) pada setiap nodenya.

Pada Gambar 2.1, zona routing dari dari node A dengan radius adalah 2 ditunjukkan. Node yang termasuk dalam zona routing Gambar 2.1 adalah semua node kecuali node L, karena posisi node L yang berada diluar zona routing node A. Penentuan zona routing tidak ditentukan dari jarak fisik, tetapi dari hop (lompatan). Pada zona routing terdapat 2 jenis node:

- *Peripheral Node*
- *Interior Node*

Node yang berada pada jarak maksimum radius yang sudah ditentukan disebut *peripheral node*, dan node yang berada pada posisi kurang dari radius (berada didalam zona) disebut *interior*

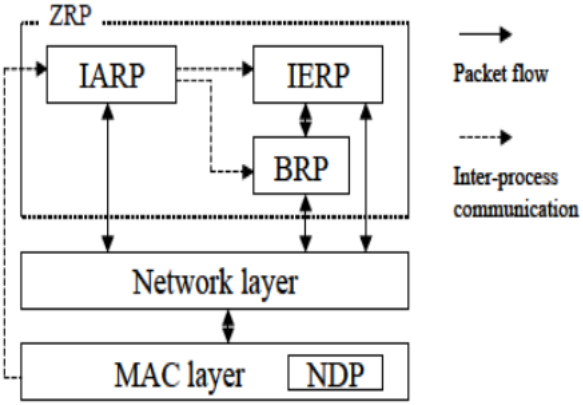
node. Pada Gambar 2.1, *Peripheral Node* adalah node E, F, G, K, M dan *Interior Node* adalah B, C, D, H, I, J. Node L berada diluar zona node A.



Gambar 2.1 Contoh Zona Routing Node A dengan Radius = 2 [6]

Untuk cara kerja pembentukan rute dari ZRP pertama Node sumber mengirimkan permintaan rute ke *Peripheral Node* miliknya. Permintaan rute berisi alamat sumber, alamat tujuan dan *sequence number*, lalu setiap *peripheral node* akan mengecek zona nya apakah node tujuan ada di zonanya. Jika pada zona *peripheral node* tidak terdapat tujuannya, maka peripheral zone akan menambahkan alamatnya kepada paket *route-request* dan meneruskan paket kepada *peripheral node* nya. Jika node tujuan berada didalam zona nya, maka *peripheral node* akan mengirimkan *route reply* kembali menuju node sumber. Node sumber menggunakan jalur yang disimpan dalam paket *route-reply* untuk mengirim paket data ke tujuan. Pada ZRP, protokol *routing* proaktif lokal (dalam zona) disebut *IntrA-zone Routing Protocol* (IARP), dan protokol routing reaktif global (diluar zona) disebut *IntEr-zone Routing Protocol* (IERP). Untuk arsitekturnya ditunjukkan pada Gambar 2.2. IARP memelihara informasi routing dari node-node yang berada dalam zona routing sebuah node.

Route discovery dan *route maintenance* dilakukan oleh IERP. Bila diperlukan penemuan global, jika topologi zona lokal diketahui, maka hal tersebut bisa digunakan untuk mengurangi lalu lintas. Untuk mem-*broadcast* paket, ZRP menggunakan konsep *Bordercasting*, yang layanannya disediakan oleh *Bordercasting Resolution Protocol* (BRP). BRP menggunakan zona routing yang sudah diperluas, yang disediakan oleh IARP lokal, untuk membangun *Bordercast tree* bersamaan dengan paket permintaan yang diarahkan. BRP menggunakan mekanisme kontrol kueri yang sangat khusus untuk mengarahkan permintaan rute dari area jaringan yang telah dikover oleh kueri sebelumnya. [6]



Gambar 2.2 Arsitektur dari ZRP [6]

2.4 Malicious Node

Malicious node adalah node yang membawa ancaman atau gangguan pada suatu jaringan. *Malicious node* bergerak secara *random* menyerang sebuah jaringan yang ada. *Malicious node* bisa masuk ke dalam jaringan karena sifat dari *malicious node* ini yang menyerupai bahkan sama seperti node normal pada umumnya, sehingga node-node lain menganggap bahwa node ini adalah node

yang aman dan normal, padahal didalam *malicious node* ini mengandung unsur-unsur yang berbahaya. Terdapat beberapa contoh jenis *malicious* berdasarkan gangguan yang dibawanya

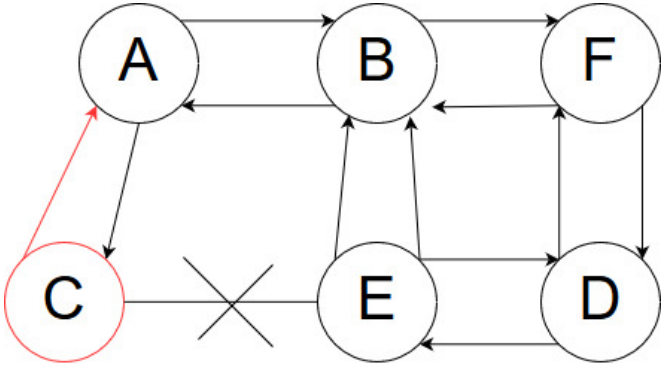
- *Flooding Attack*
- *Blackhole Attack*
- *Link Withholding Attack*
- *Link Spoofing Attack*
- *Replay Attack*
- *Wormhole Attack*

Dan masih banyak contoh lainnya dari jenis-jenis *malicious node* berdasarkan sifat gangguannya. Untuk Tugas Akhir ini, *malicious node* yang digunakan adalah *Blackhole Attack*. [7]

Serangan blackhole adalah salah satu ancaman atau gangguan pada jaringan MANET yang bersifat menghilangkan atau men-drop packet dan data sebelum paket dan data tersebut sampai di tujuan, karena paket di drop di node *blackhole* tersebut. Terdapat beberapa jenis *blackhole*, untuk tugas akhir kali ini, penulis akan menggunakan *Single Blackhole Attack*. *Blackhole* dipilih sebagai jenis serangan pada jaringan MANET. *Blackhole* dipilih sebagai jenis serangan karena pada salah satu masalah yang banyak diteliti pada jaringan MANET ini adalah serangan *Blackhole*.

Dalam serangan black hole, sebuah node jahat menggunakan protokol routing untuk mengakui bahwa *malicious node* ini memiliki jalur terpendek ke node tujuan atau ke paket yang ingin diantarkan. *Malicious node* ini juga mengaku memiliki ketersediaan rute terbaru terlepas dari memeriksa tabel routingnya. Dengan cara ini, *malicious node* akan selalu memiliki ketersediaan dalam membalas *Route-Request* dan dengan demikian, *malicious node* akan mencegat paket data dan menyimpannya. Dalam protokol yang didasarkan pada *flooding*, balasan dari *malicious node* akan diterima oleh node yang meminta sebelum penerimaan *reply* dari node yang sebenarnya, maka rute palsu dibuat. Saat rute

ini terbentuk, sekarang terserah pada *malicious node* apakah akan men-*drop* semua paket atau meneruskannya ke alamat yang tidak diketahui. Cara bagaimana *malicious node* diterima di rute pengiriman data ada bervariasi. Gambar 2.3 adalah contoh blackhole masuk ke dalam jaringan.



Gambar 2.3 Contoh Single Black Hole Attack [8]

Pada contoh diatas, node "A" ingin mengirim paket data ke node "D" dan memulai proses penemuan rute. Jadi jika simpul "C" adalah *malicious node* maka ia akan mengklaim bahwa ia memiliki rute aktif ke tujuan yang ditentukan segera setelah ia menerima paket *Route-Request*. Kemudian akan mengirimkan respon ke node "A" sebelum node lainnya. Dengan cara ini node "A" akan berpikir bahwa ini adalah rute aktif dan penemuan rute aktif selesai. Node "A" akan mengabaikan semua balasan lainnya dan akan memulai pengiriman paket data ke simpul "C". Dengan cara ini semua paket data akan hilang. [8]

2.5 Packet Delivery ratio (PDR)

Teknik penghitungan uji coba untuk skenario simulasi jaringan ada berbagai macam cara. Salah satunya adalah menggunakan *Packet Delivery Ratio* (PDR). PDR merupakan teknik penghitungan perbandingan jumlah paket yang diterima oleh node tujuan dengan jumlah paket yang dikirimkan oleh node

sumber. Penghitungan PDR ini nantinya akan digunakan untuk membuktikan bahwa pencegahan dapat membuat rasio pengiriman data lebih tinggi daripada tanpa pencegahan. Dalam presentasi, maka rumus untuk PDR ini adalah:

$$PDR = \frac{(Packet\ Delivered)}{(Packet\ Sent)} X100\% \quad \textbf{(2-1)}$$

Pada rumus 2-1, didapatkan informasi bahwa.

Packet Delivered: Jumlah total paket yang sampai ke node tujuan

Packet Sent: Jumlah total paket yang dikirimkan oleh node sumber

Tujuan dari teknik penghitungan ini adalah mengetahui rasio paket yang diterima oleh node tujuan dibanding dengan paket yang dikirimkan oleh node sumber. Penghitungan rasio dilakukan menggunakan file .tr yang dihasilkan dari NS-2 setelah menjalankan skenario pada file tcl. Berikut merupakan contoh isi dari file .tr. [9]

```
s 1.083746959 _0_ MAC --- 0 cbr 1078 [13a 4 0 800]
------- [0:0 1:0 30 4] [0] 0 0
```

Kode Sumber 2.5 Contoh salah satu baris pada file .tr yang menyatakan pengiriman paket

Pada Kode Sumber 2.6, didapatkan informasi bahwa.

s	: send, node 0 mengirim paket.
1.083746959	: waktu (timestampe) untuk operasi ini.
_0_	: id node (alamat IP) yang melakukan operasi.
MAC	: trace level (MAC layer)
0	: id paket
cbr	: tipe paket
1078	: ukuran paket yang dikirim (bytes)

2.6 Routing Overhead

Teknik penghitungan uji coba untuk skenario simulasi jaringan ada berbagai macam cara. Salah satunya adalah menggunakan *Routing Overhead* (RO). RO merupakan teknik penghitungan jumlah *control packet routing* (*route request, route reply, route error*) yang ditransmisikan pada proses pengiriman data pada sebuah jaringan. Setiap paket yang di-*forward* juga dihitung sebagai satu transmisi. Penghitungan *routing overhead* digunakan untuk melihat seberapa banyak rute yang dicari dalam jaringan ketika tanpa pencegahan dan ketika dengan pencegahan. Pada perhitungan *Routing Overhead*, yang perlu diperhatikan adalah kita menghitung jumlah total *control packet routing* yang terjadi pada suatu jaringan.

```
s 0.043919391 _2_ RTR --- 1 ZRP 30 [0 0 0 0] -----
-- [2:255 4:255 1 4] ------- [2:255 4:255 1 4]
```

Kode Sumber 2.6 Contoh salah satu baris pada file .tr yang menyatakan control packet routing [7]

Untuk perhitungan RO, kita menghitung jumlah total row yang mempunyai jenis layer **RTR** (layer yang menunjukan proses *routing*) dengan aktifitas **s** (aksi mengirim paket). Penghitungan row dengan ciri-ciri diatas menunjukan ada bahwa pada proses pembentukan rute, terjadi transmisi paket antar node nya, sebelum paket dikirimkan. [9]

2.7 Delay Time

Teknik penghitungan uji coba untuk skenario simulasi jaringan ada berbagai macam cara. Salah satunya adalah penghitungan *delay* pada pengiriman paket. *Delay* merupakan teknik penghitungan waktu yang diperlukan mulai dari paket dikirimkan oleh node sumber sampai paket data tersebut berhasil diterima node tujuan. Pada perhitungan *Delay*, yang perlu diperhatikan adalah kita menghitung waktu paket diterima,

dikurangi dengan waktu paket dikirim, maka akan ditemukan *delay* paket tersebut berhasil terkirim. Penghitungan *delay* dilakukan untuk mengukur seberapa lama keterlambatan paket untuk sampai ke tujuan antara dengan pencegahan dan tanpa pencegahan. Secara rumus, perhitungan delay adalah sebagai berikut:

$$\boldsymbol{Delay = Time\ Packet\ Recieve - Time\ Packet\ Sent} \qquad \textbf{(2-2)}$$

Pada rumus 2-2, waktu *packet sent* dapat dilihat pada file .tr hasil simluasi nya. Baris yang perlu diamati untuk menentukan *packet revieved* adalah sebagai berikut.

```
s 3.565000000 _0_ AGT --- 203 cbr 100 [0 0 0 0] --
----- [0:2 1:0 32 0] [171] 0 0
```

Kode Sumber 2.7 Contoh salah satu baris pada file .tr yang menyatakan pengiriman packet (packet sent)

Pada Kode Sumber 2.9, dapat dilihat salah satu baris yang menyatakan ada terjadi pengiriman paket dari node sumber. **S** pada kode sumber menyatakan bahwa ada aktifitas mengirim paket. **AGT** pada kode sumber menyatakan bahwa aktifitas ini terjadi pada layer agent/aplikasi pada layer ini pengiriman sudah berupa paket asli. Pada bagian yang di tandai warna merah, adalah yang sebut sebagai waktu *packet sent*. Lalu untuk bagian *packet received* adalah sebagai berikut.

```
r 4.385290962 _1_ AGT --- 203 cbr 132 [13a 1 5
800] ------- [0:255 1:0 1 1] [171] 4 0
```

Kode Sumber 2.8 Contoh salah satu baris pada file .tr yang menyatakan penerimaan packet (packet recieved)

Pada Kode Sumber 2.10, dapat dilihat salah satu baris yang menyatakan ada terjadi pengiriman paket dari node sumber. **R** pada kode sumber menyatakan bahwa ada aktifitas penerimaan paket. **AGT** pada kode sumber menyatakan bahwa aktifitas ini terjadi pada layer agent/aplikasi pada layer ini pengiriman sudah

berupa paket asli. Pada bagian yang di tandai warna merah, adalah yang sebut sebagai waktu *packet received*. Sehingga, pada contoh diatas, dapat dilihat bahwa *delay time* nya adalah 4.385290962 - 3.565000000 = 0.820290962. [9]

2.8 Cara Menghindari Blackhole yang Diharapkan

Sekuritas merupakan salah satu penjaminan mutu suatu sistem. Sehingga sekuritas dapat dianggap sebagai suatu hal yang penting. Dalam MANET, metode penyaluran informasi menggunakan *broadcast* ke node tetangga (*intermediate node*). Hal ini membutuhkan kualitas yang baik agar paket dapat sampai ke tujuan. Tujuan untuk mengevaluasi apakah jaringan *mobile ad-hoc* adalah aman atau tidak adalah sebagai berikut:

- Ketersediaan
- Kerahasiaan
- Integritas
- Otentikasi
- Non-penolakan
- Anonimitas

Pada umumnya, mayoritas metode deteksi serangan *blackhole* menggunakan protokol AODV, dan protokol proaktif atau reaktif, sehingga ada protokol *hybrid* ZRP yang digunakan untuk mendeteksi node *blackhole* di MANET. Protokol proaktif dan protokol reaktif memiliki masalah seperti pada *Packet Delivery Ratio (PDR)* nya dan masalah *Routing Overhead (RO)* nya sehingga ada keuntungan dari kombinasi protokol menggunakan protokol hybrid ZRP untuk mendeteksi *malicious node*. Algoritma untuk menghindari serangan blackhole adalah:

Langkah 1 - Menemukan semua node tetangga dari node sumber.
Langkah 2 - Node sumber mendapatkan informasi tabel routing tentang semua node tetangga.
Langkah 3 - Menggunakan node tetangga mengirim (*routing request*) RREQ (*s_addr, d_addr*) ke node perifer.
Langkah 4 - Mendapatkan *route-reply* (*s_addr.n_ip*) dari node.

Periksa apakah ada node di antara *interior node* atau *peripheral node* yang berfungsi sebagai *peripheral node*, adalah *malicious node,* lalu menyiarkan informasi tersebut kepada node lain. [8]

[Halaman ini sengaja dikosongkan]

BAB III
ANALISIS DAN PERANCANGAN PERANGKAT LUNAK

Pada bab ini dijelaskan mengenai dasar perancangan dari perangkat lunak yang dibangun dalam Tugas Akhir ini. Secara khusus akan dibahas mengenai deskripsi umum sistem, perancangan skenario, alur, serta gambaran implementasi sistem yang diterapkan pada Network Simulator 2 (NS-2).

3.1 Deskripsi Umum

Pada Tugas Akhir ini akan dilakukan implementasi metode deteksi dan cara menghindari serangan *blackhole* pada lingkungan jaringan MANET. Protokol yang digunakan adalah ZRP. Dalam pembuatan skenario MANET menggunakan simulator yaitu Network Simulator 2 (NS-2).

Perancangan skenario uji coba mobilitas MANET diawali dengan perancangan skenario yang akan dilakukan pada simulasi *blackhole*. Kemudian melakukan modifikasi pada protokol ZRP untuk menambahkan *blackhole*.

Setelah itu melakukan perancangan untuk deteksi dan pencegahan serangan *blackhole* yaitu dengan merancang metode pengenalan rute milik ZRP, yaitu jika didalam zone menggunakan protokol proaktif, dan jika diluar zona menggunakan protokol reaktif, dan metode *blacklist* yang berisikan daftar node-node berbahaya yang harus dihindari. Dan terakhir melakukan perancangan simulasi pada NS-2 yang dilakukan pada skrip TCL.

3.2 Daftar Istilah

Berikut adalah daftar istilah yang digunakan pada buku tugas akhir ini.

Tabel 3.1 Tabel Daftar Istilah

Mobile Adhoc Network (MANET)	*Mobile Ad Hoc Network* (MANET) adalah jaringan nirkabel yang memiliki struktur jaringan yang abstrak, dan mampu mengatur konfigurasinya sendiri
Zone Routing Protocol (ZRP)	*Zone Routing Protocol* (ZRP) adalah protokol jaringan nirkabel yang *hybrid* yang menggunakan keuntungan dari *reactive* dan *proactive routing protocol* saat mengirimkan pesan pada jaringan
Malicious Node	*Malicious node* adalah node yang membawa ancaman atau gangguan pada suatu jaringan
Packet Delivery Ratio (PDR)	PDR merupakan teknik penghitungan perbandingan jumlah paket yang diterima oleh node tujuan dengan jumlah paket yang dikirimkan oleh node sumber
Delay	*Delay* merupakan teknik penghitungan waktu yang diperlukan mulai dari paket dikirimkan oleh node sumber sampai paket data tersebut berhasil diterima node tujuan.
Routing Overhead	Routing Overhead merupakan teknik penghitungan jumlah *control packet routing* (*route request, route reply, route error*) yang ditransmisikan

	pada proses pengiriman data pada sebuah jaringan.

3.3 Perancangan Skenario

Perancangan skenario uji coba pengiriman paket dari node sumber ke node tujuan dengan adanya *blackhole* pada lingkungan MANET.

Pada simulasi ini, terdapat beberapa node dengan pembagian peran sebagai berikut:

- Node sumber
  Merupakan node awal pengiriman paket.
- Node *intermediate*
  Merupakan node penyalur paket dari node sumber ke node tujuan.
- Node tujuan
  Merupakan node akhir pengiriman paket dari node sumber atau node yang menjadi tujuan pengiriman dari node sumber.
- Node *blackhole*
  Merupakan node yang mengatasnamakan dirinya sebagai node tujuan, sehingga paket yang dikirim dari node sumber tidak pernah sampai ke node tujuan.

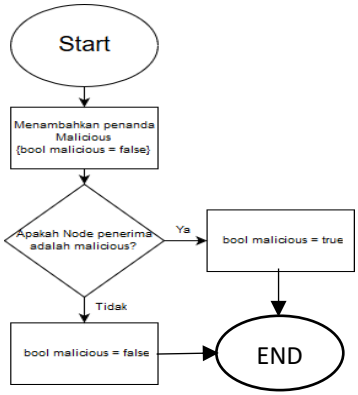
Masing-masing node akan bergerak ke suatu tujuan karena skenario ini akan diimplementasikan pada lingkungan MANET. Pemberian warna untuk masing-masing peran node agar membedakan node sumber, tujuan, *intermediate*, dan *blackhole*.

Dan yang terakhir pencocokan node yang ada apakah ada pada daftar *blacklist* atau tidak oleh protokol ZRP sehingga bisa terhindar dari node yang berupa *blackhole*.

3.4 Perancangan Blackhole pada ZRP

Pada perancangan implementasi *blackhole*, implementasi dilakukan pada file zrp.cc dan zrp.h agar pada skrip TCL dapat dibaca dan berlaku layaknya *blackhole* asli ketika *blackhole* telah ditambahkan. Metode penambahan blackhole ini yaitu yang pertama adalah membaca dari file tcl (file simulasi). Beberapa node akan ditandai sebagai *blackhole*. Kemudian pada protokol ZRP dilakukan pengecekan untuk masing-masing node. Ketika node *blackhole* telah teridentifikasi, maka penanda bahwa node tersebut merupakan *blackhole* diaktifkan.

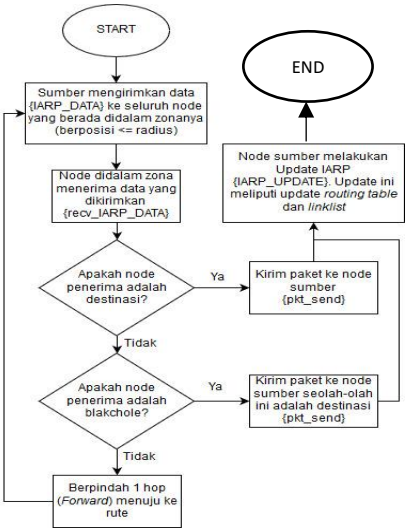
Penggunaan pertama kali (Gambar 3.1) penanda berikut adalah untuk membuat node tersebut bertindak layaknya *blackhole*, yaitu melakukan *drop* paket yang diterima oleh *blackhole* pada saat node sumber mengirim paket. Kemudian penggunaan yang kedua adalah ketika *blackhole* menerima *request* dari node sumber, *blackhole* ini akan mengirimkan balasan dengan mengatas namakan dirinya sebagai node tujuan. Lalu penggunaan yang ketiga adalah untuk membuat id node tersebut didaftarkan ke daftar *Blacklist* agar dicegah masuk ke dalam jaringan.



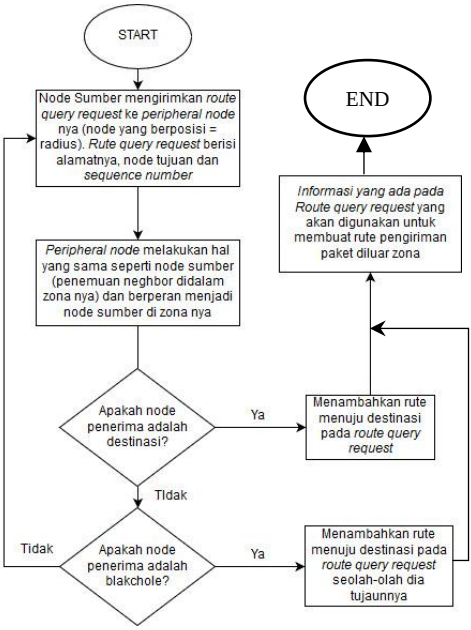
Gambar 3.1 Instansiasi Penanda pada Blackhole

3.5 Perancangan Pengenalan Rute dan Node

Data informasi tentang *neighbor* (tetangga) terdapat pada pesan balasan (*reply*) yang dikirimkan dari node tujuan maupun *blackhole*. Metode pengecekan menggunakan metode proaktif ketika berada di zona nya. Untuk mengenali node yang berada diluar zona, maka node sumber akan mengirimkan *route query packet* ke node-node terluarnya, atau node yang terhitung hop nya sama dengan radius, yang biasanya disebut *peripheral node*. Lalu *peripheral* node akan melakukan hal yang sama seperti node sumbernya lakukan. Ketika di suatu zona sudah menemukan tujuan utamanya, maka tujuan utamanya tersebut akan membalas dengan *route-reply* dan menghubungkan jalurnya. Node-node yang didalam zona baik intermediate node maupun blackhole akan berkomunikasi melalui *Intra-Zone Routing Protocol* (IARP) dengan menggunakan metode protokol proaktif. Sedangkan yang berada diluar zona baik intermediate node maupun blackhole akan berkomunikasi melalui *Inter-Zone Routing Protocol* (IERP) dengan menggunakan metode protokol reaktif.



Gambar 3.2 Implementasi Blackhole pada protocol IARP

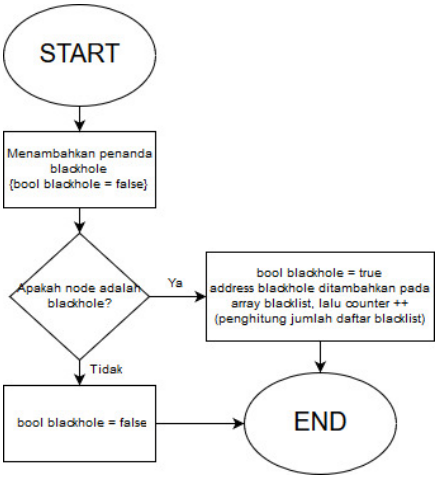


Gambar 3.3 Implementasi Blackhole pada protocol IERP

3.6 Perancangan Penambahan Node yang Terindikasi Blackhole pada Daftar Blacklist

Pada perancangan ini, setiap node *blackhole* yang sudah diberi penanda akan didaftarkan pada daftar *blacklist*. Daftar *blacklist* ini berisi daftar-daftar alamat node-node yang berbahaya pada jaringan tersebut. Perancangan ini diimplementasikan pada file zrp.cc, dimana *blacklist* berupa *array* yang berisi *integer* alamat node. Pada implementasinya, ketika sebuah node sudah di instansiasi sebagai node *blackhole*, maka secara otomatis, alamat node tersebut juga akan didaftarkan pada daftar *blacklist*, dan variabel penghitung jumlah node yang ada pada *blacklist* akan bertambah. Dengan demikian, nantinya *blacklist* ini akan digunakan untuk mencegah agar node-node yang terdaftar pada

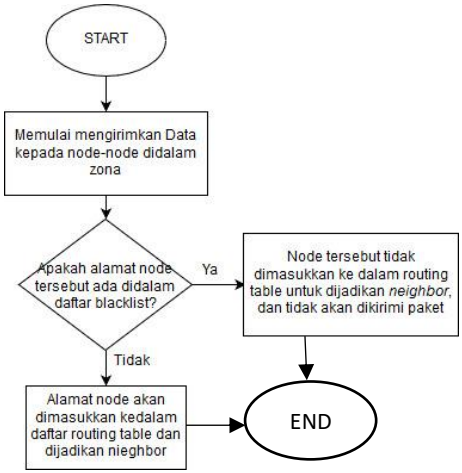
blacklist tidak akan masuk di *routing table* pada IARP, dan tidak akan disambungkan ke rute yang ada pada IERP.



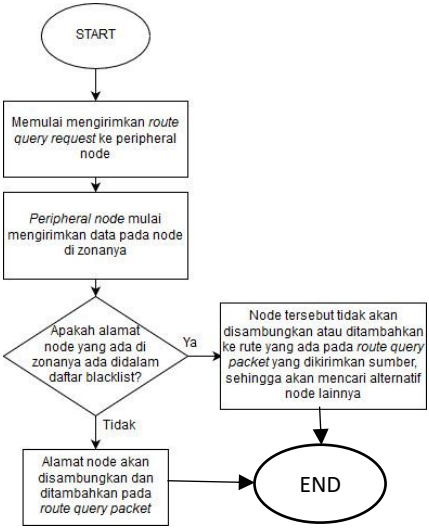
Gambar 3.4 Pendaftaran malicious node pada daftar blacklist

3.7 Perancangan Pengenalan Neighbor apakah Blackhole atau Tidak serta Pencegahannya

Pada perancangan ini, setiap sebelum node melakukan pengiriman pesan, akan dilakukan pengenalan rute dan pengenalan node seperti pada bagian 3.4. Sebelum pengenalan rute dan node, setiap node akan mengecek ke daftar *blacklist*, apakah node yang nanti akan dikenalinya ini terdaftar sebagai node yang berbahaya. Jika terdaftar, maka node tersebut tidak akan dimasukkan ke dalam *routing table* pada jika node nya berada di dalam zona, dan tidak akan disambungkan/ ditambahkan *link* nya jika node nya berada diluar zona. Jika node yang dikenalinya tidak ada dalam daftar blacklist, maka pengenalan bisa dilakukan. Perancangan ini diimplementasikan pada file zrp.cc.



Gambar 3.6 Implementasi Pencegahan pada Kasus Malicious Node berada didalam Zona (IARP)



Gambar 3.5 Implementasi Pencegahan pada Kasus Malicious Node berada diluar Zona (IERP)

3.8 Perancangan Simulasi pada NS-2

Pada perancangan kode NS-2 dengan konfigurasi MANET, dengan skrip TCL yang diberikan parameter-parameter untuk membangun percobaan simulasi MANET pada NS-2. Berikut parameter simulasi perancangan sistem MANET yang dapat digunakan dapat dilihat pada Tabel 3.1

Tabel 3.2 Tabel Parameter Simulasi NS-2

No	Parameter	Spesifikasi
1	Network Simulator	NS-2.35
2	*Routing Protocol*	ZRP
3	Radius	3
4	Waktu Simulasi	100 detik
5	Waktu Pengiriman Paket Data	15-100 detik
6	Area Simulasi	500 x 500
7	Banyak Node	20
8	Kecepatan pergerakan node	2 m/s
9	Banyak Blackhole	2 (0,1)
10	Radius Transmisi	250 m
11	Tipe Koneksi	UDP
12	Tipe Data	Constant Bit Rate (CBR)
13	Source / Destination	Statik (*Node* 18 / *Node* 19)
14	Kecepatan Generasi Paket	1 paket per detik
15	Ukuran paket data	100 bytes

16	Protokol MAC	IEEE 802.11
17	Tipe Antena	OmniAntenna
18	Tipe Interface Queue	Droptail/PreQueue
19	Tipe Kanal	Wireless Channel
20	Tipe *trace*	Old Format *Trace*

BAB IV
IMPLEMENTASI

Pada bab ini akan dibahas mengenai implementasi yang dilakukan berdasarkan rancangan yang telah dijabarkan pada bab sebelumnya. Implementasi yang dijelaskan meliputi lingkungan pembangunan perangkat lunak, implementasi skenario implementasi *blackhole* pada protokol ZRP, implementasi pencegahannya, implementasi simulasi pada NS-2.

4.1 Lingkungan Pembangunan Sistem

Pembangunan perangkat lunak dilakukan pada lingkungan pengembangan sebagai berikut:

4.1.1 Lingkungan Perangkat Lunak

Adapun perangkat lunak yang digunakan untuk pengembangan sistem adalah sebagai berikut:

- Sistem Operasi Linux Ubuntu 14.04 LTS 64 bit untuk lingkungan NS-2
- *Network Simulator 2* (NS-2) versi 2.35

4.1.2 Lingkungan Perangkat Keras

Spesifikasi perangkat keras yang digunakan untuk implementasi perangkat lunak Tugas Akhir adalah sebagai berikut:

- Processor Intel® Core TM i3-2120 CPU @ 3.30 GHz x 4 core
- Media Penyimpanan 250 GB
- RAM sebesar 8 GB DDR3

4.2 Instalasi Protokol ZRP pada NS-2

Untuk protokol ZRP, normalnya tidak ada pada bawaan instalasi NS 2.35 standar, perlu ditambahkan dan dilakukan *patch* tersendiri. Untuk langkah instalasi protokol ZRP ini adalah:

1. Download *ns-allinone-2.35* pada tautan http://bit.ly/InstallNS235
   atau
   https://sourceforge.net/projects/nsnam/files/allinone/ns-allinone-2.35/ns-allinone-2.35.tar.gz/download
2. Download *patch* protokol ZRP nya pada tautan http://bit.ly/PatchZRP-NS235
   atau
   https://drive.google.com/file/d/0B7S255p3kFXNNUVuOE9MQWhRNTQ/view?usp=sharing
3. Ekstrak file *ns-allinone-2.35* yang tadi sudah di unduh
4. Masuk ke folder hasil ekstrak tadi
5. Salin (*copy*) file *patch* dari zrp nya (file bernama *zrp-ns235.patch*) ke dalam folder *ns-allinone-2.35* yang tadi sudah di ekstrak
6. Setelah di *copy*, jalankan sintaks berikut untuk melakukan patch

```
patch -p0 < zrp-ns235.patch
```

Kode Sumber 4.1 Command untuk melakukan patch protokol ZRP pada ns 2.35

7. Setelah *patch* dilakukan, lakukan instalasi dengan cara pada folder *ns-allinone-2.35* ketikkan sintaks berikut

```
./install
```

Kode Sumber 4.2 Sintaks instalasi NS 2.35

8. Setting *environment variables* pada .bashrc agar NS-2 bisa dijalankan pada semua lingkungan linux

4.3 Implementasi Skenario

Implementasi skenario mobilitas MANET dengan node-node sesuai peran masing-masing.

- Node n-2: sebagai node sumber (contoh: jumlah node = 10, node sumber = 8)
- Node n-1: sebagai node tujuan (contoh: jumlah node = 10, node sumber = 9)
- Node 0 dan seterusnya sejumlah banyak *blackhole* (contoh: jumlah blackhole = 2, node blackhole = 0, 1)
- Node sisa dan seterusnya merupakan *intermediate node*. (contoh: jumlah node = 10, node intermediate 3 - 7)

Posisi node sumber dan tujuan statis dan tidak bergerak. Sedangkan node *blackhole* dan *intermediate node* posisi dan pergerakannya *random*.

```
set val(nn) 10

set val(source) [expr "$val(nn) - 2"]
set val(destination) [expr "$val(nn) - 1"]

set node_($val(source)) [$ns_ node]
$node_($val(source)) set X_ 0
$node_($val(source)) set Y_ 250
$node_($val(source)) set Z_ 0.0
$ns_ initial_node_pos $node_($val(source)) 20

set node_($val(destination)) [$ns_ node]
$node_($val(destination)) set X_ 500
$node_($val(destination)) set Y_ 250
$node_($val(destination)) set Z_ 0.0
$ns_ initial_node_pos $node_($val(destination)) 20
```

Kode Sumber 4.3 Posisi Statis Node Sumber dan Node Tujuan

Pada Kode Sumber 4.3, jumlah node adalah 10. Jadi dari fungsi diatas, kita bisa menentukan bahwa node ssumber adalah jumlah node dikurangi dengan 2, maka node sumber adalah 8, dan untuk node tujuan adalah jumlah node dikurangi 1, jadi node tujuan adalah 9.

Lalu untuk posisi node sumber dan node tujuan ditentukan juga. Pada Kode Sumber 4.3 dijelaskan bahwa untuk node sumber mempunyai koordinat posisi yaitu 0, 250 (sumbu x, y) dan kedalaman 0 (sumbu z). Sedangkan untuk node tujuan mempunyai koordinat posisi yaitu 500, 250 (sumbu x, y) dan kedalaman 0 (sumbu z).

Pada Kode Sumber 4.4, semua posisi node kecuali node sumber dan node tujuan diatur secara random dengan aturan untuk sumbu x random mulai dari 0 hingga 500, sumbu y random dari 0 hingga 500, dan sumbu z tetap pada titik 0.

```
set rng [new RNG]
$rng seed next-substream

for {set i 0} {$i < $val(source)} {incr i} {
    set node_($i) [$ns_ node]
    $node_($i) set X_ [$rng uniform 0.0 500.0]
    $node_($i) set Y_ [$rng uniform 0.0 500.0]
    $node_($i) set Z_ 0.0 #flat ground
    $ns_ initial_node_pos $node_($i) 20
}
```

Kode Sumber 4.4 Posisi random node blackhole dan node intermediate

Jadi, untuk generasi node secara *random*, tergantung pada jumlah node nya. Mengacu pada jumlah node pada Kode Sumber 4.3, maka untuk perulangan generasi random node adalah dari node 0 hingga node sebelum node sumber, yaitu node ke 7.

Lalu untuk pemberian warna sebagai penanda diperlukan untuk mengecek ketika melakukan pengecekan pada *network animator*. Pada Kode Sumber 4.5, pemberian warna node sesuai dengan peran dari masing-masing node. Untuk warna hijau menyatakan bahwa node tersebut adalah node sumber, lalu warna biru menyatakan node tujuan, node merah menyatakan *blackhole*, dan untuk node lainnya (*intermediate node*) diberi warna standar, yaitu warna hitam.

```
$node_($val(source)) color green
$ns_ at 0.0 "$node_($val(source)) color green"
$ns_ at 0.0 "$node_($val(source)) label Source"

# Destination Node
$node_($val(destination)) color blue
$ns_ at 0.0 "$node_($val(destination)) color blue"
$ns_ at 0.0 "$node_($val(destination)) label
Destination"

# Blackhole Attacker
$node_($val(blackhole)) color red
$ns_ at 0.0 "$node_($val(blackhole1)) color red"
$ns_ at 0.0 "$node_($val(blackhole1)) label
Blackhole"
```

Kode Sumber 4.6 Pemberian warna penanda pada node

Pemberian identitas untuk node *blackhole* dilakukan dengan menambahkan penanda yaitu “blackhole” pada file skenario tcl sebagai berikut.

```
$ns_ at 0.0 "[$node_($val(blackhole)) set ragent_]
blackhole"
```

Kode Sumber 4.5 Pemberian Identitas Node Blackhole

Setelah itu menjadikan semua node kecuali node sumber dan node tujuan bergerak ke suatu lokasi *random*. Generasi pergerakan node *random* menggunakan fitur *setdest* pada NS-2.

Pada Kode Sumber 4.7, -v 1 merupakan versi dari fitur setdest ini, -n 8 merupakan jumlah node yang ingin diimplementasi memiliki pergerakan (mobile), -p 1 merupakan pause time 1 detik sehingga pergerakan dimulai ketika detik ke 1, -M 2 merupakan kecepatan maksimal pergerakan node (m/s), -t 100 merupakan waktu simulasi 100 detik sehingga selama 100 detik maka node

akan aktif terus bergerak, -x 500 merupakan lebar lingkungan simulasi (koordinat x) sehingga posisi node akan di random dari koordinat x 0 sampai 500 saja, dan -y 500 merupakan ketinggian lingkungan simulasi (koordinat y) sehingga posisi node akan di random dari koordinat x 0 sampai 500 saja. Dari hasil running setdest ini, dimasukkan ke dalam file dengan nama "10-random-node" yang nantinya dapat digunakan untuk skenario di file tcl.

```
./setdest -v 1 -n 8 -p 1 -M 2 -t 100 -x 500 -y 500
> 10-random-node
```

Kode Sumber 4.7 Pergerakan random node blackhole dan node intermediate

4.4 Implementasi Blackhole pada ZRP

Untuk implementasi *malicious node* berupa *blackhole* pada ZRP, maka perlu adanya modifikasi pada protokol ZRP yang ada pada *Network Simulator 2* (NS-2).

Modifikasi yang pertama dilakukan adalah dengan menambahkan variabel *malicious* yang bertipe *public boolean* sebagai penanda bahwa node adalah *blackhole*. Penambahan variabel *malicious* dilakukan didalam kelas *ZRPAgent : public Agent* pada file *zrp.h*. Implementasinya dapat dilihat pada Kode Sumber 4.8

```
class ZRPAgent : public Agent {
	...
	public:
		...
		bool malicious;
	...
}
```

Kode Sumber 4.8 Penambahan variabel malicious pada zrp.h

Setelah penambahan pada *header* di file *zrp.h*, selanjutnya perlu penambahan pada konstruktor dari protokol ini. Pada file *zrp.cc* perlu ditambahkan beberapa sintaks pada fungsi konstruktor ZRP untuk melakukan instansiasi variabel *malicious* agar penanda *blackhole* dapat digunakan dengan benar hanya ketika node *blackhole* berhasil diindentifikasi oleh ZRP. Penambahan ini dapat dilihat pada Kode Sumber 4.9.

```
ZRPAgent::ZRPAgent():Agent(PT_ZRP), myid_('\0'),
myaddr_(0), radius_(DEFAULT_ZONE_RADIUS),
                    sendBuf_(this),
pktUtil_(this), ndpAgt_(this), iarpAgt_(this),
ierpAgt_(this) {
       ...
       malicious = false;
}
```

Kode Sumber 4.9 Instansiasi variabel pada konstruktor ZRP pada file zrp.cc

Pada dasarnya, sebuah node adalah node normal, sehingga penanda *malicious* milik node tersebut bernilai *false*. Ketika sebuah node diinstansiasikan sebagai sebuah node *blackhole* (dapat diinstasiasikan pada file tcl seperti pada Kode Sumber 4.6), maka penanda *malicious* pada node tersebut berubah menjadi *true*. Penambahan ini dapat dilihat pada Kode Sumber 4.10, dengan pemberian indetitas dengan nama “blackhole”.

```
int ZRPAgent::command (int argc, const char*const*
argv) {
       ...
       if(strcmp(argv[1], "blackhole") == 0) {
              malicious = true;
              ...
              return TCL_OK;
      }
       ...
}
```

Kode Sumber 4.10 Pemberian tanda malicious pada node yang diinstansiasikan sebagai node blackhole

Alur kerja ZRP sendiri terbagi menjadi dua bagian karena menggabungkan 2 jenis protokol, yaitu proaktif dan reaktif. Pada bagian routing didalam zona nya, digunakan lah *IntraZone Routing Protokol*, yaitu node sumber mengirimkan IARP data ke semua node yang berada di dalam zona nya. Kemudian node-node didalam zona nya akan menerima data tersebut (*recv_IARP_DATA()*) dan akan melakukan pengecekan apakah node tersebut tujuan, *blackhole*, atau *intermediate node*. Sehingga modifikasi dilakukan pada fungsi *recv_IARP_DATA()* yang ada pada file *zrp.cc*. Modifikasi kode ini dapat dilihat pada Kode Sumber 4.11 untuk sifat *blackhole* yang melakukan *drop* paket, dan sifat *blackhole* juga yang melakukan pengiriman pesan *reply* seolah-olah diri nya node tujuan jika node *blackhole* berada didalam zona.

```
void IARPAgent::recv_IARP_DATA(Packet* p) {

       if (agent_->malicious == true ) {
         (agent_->pktUtil_).pkt_drop(p);
         return;
       }
       ...
       else if(agent_->malicious == TRUE) {
              ...
              hdrc->ptype() = hdrz->enc_ptype_;
              hdrc->direction() = hdr_cmn::UP;

              hdrc->addr_type_ = NS_AF_NONE;
              hdrc->size() -= IP_HDR_LEN;

              hdrip->dport() = hdrz->enc_dport_;
              hdrip->saddr() = hdrz->src_;
              hdrip->daddr() = hdrz->dest_;
              hdrip->ttl() = 1;
              (agent_->pktUtil_).pkt_send(p,
              agent_->myaddr_, 0.00);
       }
       ...
}
```

Kode Sumber 4.11 Melakukan drop packet dan pengiriman reply oleh Blackhole yang berada didalam zona

Lalu untuk kasus node yang berada diluar zona, node sumber akan mengirimkan IERP data ke node-node rute yang berada diluar zona nya. Lalu node tersebut menerima data tersebut (*recv_IERP_DATA()*) dan akan melakukan pengecekan apakah node tersebut tujuan, *blackhole*, atau *intermediate node*. Sehingga modifikasi dilakukan pada fungsi recv_IERP_DATA() yang ada pada file zrp.cc. Modifikasi kode ini dapat dilihat pada Kode Sumber 4.12 untuk sifat *blackhole* yang melakukan *drop* paket, dan sifat *blackhole* juga yang melakukan pengiriman pesan *reply*

seolah-olah diri nya node tujuan jika node *blackhole* berada diluar zona.

```
void IERPAgent::recv_IERP_DATA(Packet* p) {

       if (agent_->malicious == true ) {
         (agent_->pktUtil_).pkt_drop(p);
         return;
       }
       ...
       else if(agent_->malicious == true) {
              ...
              (agent_>pktUtil_).pkt_free_ROUTE_
              space(p);

              hdrc->ptype() = hdrz->enc_ptype_;
              hdrc->direction() = hdr_cmn::UP;
              hdrc->addr_type_ = NS_AF_NONE;
              hdrc->size() -= IP_HDR_LEN;
              hdrip->dport() = hdrz->enc_dport_;
              hdrip->saddr() = hdrz->src_;
              hdrip->daddr() = hdrz->dest_;
              hdrip->ttl() = 1;
              (agent_->pktUtil_).pkt_send(p,
              agent_->myaddr_, 0.00)
       }
       ...
}
```

Kode Sumber 4.12 Melakukan drop packet dan pengiriman reply oleh Blackhole yang berada diluar zona

Dengan modifikasi dari kode-kode program diatas, maka implementasi adanya *blackhole* pada protokol ZRP di NS-2 sudah dapat digunakan.

4.5 Implementasi Pencegahan Blackhole

Untuk implementasi pencegahan blackhole ada 4 tahap, yaitu tahap instansiasi array blacklist dan counter, pendaftaran node berbahaya ke dalam daftar blacklist, pencegahan jika node berada didalam zona, dan yang terakhir pencegahan ketika node berada diluar zona.

4.5.1 Instansiasi Array Blacklist dan Counter

Yang pertama perlu dibuat terlebih dahulu sebuah *array* untuk daftar *blacklist*. Daftar inilah yang nanti digunakan untuk mengecek apakah alamat node tersebut masuk ke daftar *blacklist* atau tidak. Akan tetapi, salah satu batasannya adalah besaran *array* yang terbatas sehingga *blackhole* tidak bisa lebih dari kapasitas *array* nya. Untuk Tugas Akhir ini, batasan besaran *array* nya adalah 50, sehingga maksimum blackhole yang bisa di tampung adalah 50, serta dengan instansiasi awal berisi node dengan alamat -1. Perlu adanya juga penghitung jumlah node yang ada didalam *blacklist*. Pada kasus ini diberi nama *counter*. Implementasi akan dijelaskan pada Kode Sumber 4.13

```
//instansiasi array blacklist dan counter
int blacklist[50]={ -1 };
int counter = 0;
```

Kode Sumber 4.13 Instansiasi array blacklist dan counter untuk daftar node-node yang berbahaya

4.5.2 Pendaftaran Node Berbahaya ke dalam Blacklist

Lalu tahap kedua adalah menambahkan alamat node-node yang berbahaya ke dalam daftar *blacklist*. Pada bagian ini, ketika sebuah node sudah diinstansiasikan sebagai sebuah node berbahaya (*blackhole*), maka secara otomatis alamat node tersebut akan langsung didaftarkan ke daftar *blacklist*. Sehingga deteksi apakah sebuah node adalah *blackhole* atau tidak adalah dari pencocokkan alamat node tersebut apakah ada pada daftar alamat berbahaya di daftar *blacklist*. Jika ada pada daftar *blacklist*, maka node tersebut akan diindentifikasi sebagai *blackhole* dan nantinya akan dicegah agar tidak masuk ke dalam jaringan. Implementasi dijelaskan pada Kode Sumber 4.14.

```
int ZRPAgent::command (int argc, const char*const*
argv) {
	...
	if(strcmp(argv[1], "blackhole") == 0) {
		...
		//alamat ditambahkan ke array
		//blacklist
		blacklist[counter] = myaddr_;
		//counter ditambahkan
		counter ++;
		...
	}
	...
}
```

Kode Sumber 4.14 Penambahan alamat node berbahaya ke dalam daftar blacklist

4.5.3 Pencegahan Blackhole yang Berada didalam Zona

Untuk pencegahan jika node berada didalam zona perlu dilakukan karena sifat routing proaktif yang mengirimkan paket ke seluruh node didalam zonanya untuk mengetahui siapa tetangganya. Ketika node *blackhole* berada di dalam zonanya, maka perlu dicegah alamat node tersebut agar tidak dijadikan tetangga oleh node sumber dan node-node lainnya. Dengan demikian, maka node *blackhole* tidak akan masuk daftrar *neighbor* pada zona tersebut. Cara mencegahnya adalah mengecek, apakah alamat node tetangga di dalam zona nya ada didalam daftar blacklist. Jika ada, maka node tersebut tidak akan ditambahkan pada daftar neighbor. Implementasi akan ditunjukkan pada Kode Sumber 4.15.

4.5.4 Pencegahan Blackhole yang Berada diluar Zona

Untuk pencegahan ketika node *blackhole* berada diluar zona perlu dilakukan karena jika diluar zona, awalnya node sumber tidak mengetahui alamat rute jika destinasi berada diluar zona. Maka perlu ada pencegahan jika terdapat *blackhole* diluar zona. Cara pencegahannya adalah ketika node sumber memberikan *IERP route request* ke semua *peripheral node* nya, maka *peripheral node* akan melakukan persis seperti yang node sumber lakukan, lalu akan menambahkan alamat rute miliknya ke paket *IERP route request* nya. Pencegahan dilakukan jika alamat node yang akan disambungkan maupun node yang akan menyambungkan terdaftar pada blacklist, maka tidak akan ada penambahan *link* pada daftar *link* (*linklist*) dari maupun menuju node tersebut. Implementasi akan ditunjukkan pada Kode Sumber 4.16.

```
NeighborList::addNeighbor(nsaddr_t addr, Time
lastack, int linkStatus) {
	//Pengecekan apakah counter tidak kosong
	if (counter != 0) {
		//rekursi sebanyak jumlah alamat pada
		//daftar blacklist
		for(int i=0; i<counter; i++) {
			//jika alamat terdaftar pada
			//daftar blacklist
			if(addr == blacklist[i]) {
				//tidak ditambahkan
				//menjadi tetangga baru
				return;
			}
			else{
				//ditambahkan menjadi
				//tetangga baru
				Neighbor *newNb = new
				Neighbor(addr, lastack,
				linkStatus);
				...
				newNb->next_ = head_;
				head_ = newNb;
				numNeighbors_++;
			}
		}
	}
	else{
		//ditambahkan menjadi tetangga baru
		Neighbor *newNb = new Neighbor(addr,
		lastack, linkStatus);
		...
		newNb->next_ = head_;
		head_ = newNb;
		numNeighbors_++;
	}
}
```

Kode Sumber 4.15 Implementasi pencegahan jika blackhole berada didalam zona

```
void LinkStateList::addLink(nsaddr_t src, nsaddr_t
dest, int seq, int isup, Time expiry) {
//Pengecekan apakah counter tidak kosong
if (counter != 0){
       //rekursi sebanyak jumlah alamat pada daftar
       //blacklist
       for(int i=0; i<counter; i++) {
              //jika alamat yang akan di sambungkan
              //atau yang akan menyambungkan
              //termasuk dalam daftar blacklist
              if(dest == blacklist[i] || src
              == blacklist[i]) {
                     //tidak ditambah menjadi link baru
                     return;
              }
              else{
                     //ditambahkan menjadi link baru
                     LinkState *newLS = new LinkState
                     (src, dest,seq, isup, expiry);
                     ...
                     newLS->next_ = head_;
                     head_ = newLS;
                     numLinks_++;
              }
       }
}
else{
       //ditambahkan menjadi link baru
       LinkState *newLS = new LinkState(src,
       dest, seq, isup, expiry);
       ...
       newLS->next_ = head_;
       head_ = newLS;
       numLinks_++;
}
}
```

Kode Sumber 4.16 Implementasi pencegahan jika blackhole berada diluar zona

4.6 Implementasi Simulasi pada NS-2

Dilanjutkan dengan implementasi simulasi pada simulator yaitu NS-2. Simulasi ini akan dijalankan menggunakan kode program tcl. Hal pertama yang perlu dilakukan adalah menambahkan pengaturan untuk simulasi. Penambahan pengaturan dapat dilihat pada Kode Sumber 4.17. Setelah melakukan pengaturan parameter simulasi, lakukan inisialisasi program untuk skenario MANET.

```
# channel type
set val(chan)         Channel/WirelessChannel;
# radio-propagation model
set val(prop)         Propagation/TwoRayGround;
# Antenna type
set val(ant)          Antenna/OmniAntenna;
# Link layer type
set val(ll)           LL;
# Interface queue type
set val(ifq)          Queue/DropTail/PriQueue;
# max packet in ifq
set val(ifqlen) 50;
# network interface type
set val(netif)        Phy/WirelessPhy;
# MAC type
set val(mac)          Mac/802_11;

set val(nn)           10;#number of mobilenodes
set val(rp)           ZRP;#routing protocol
set val(x)            500;#X dimensi topotgrafi
set val(y)            500;#Y dimensi topotgrafi
set val(stop)         100;#Waktu simulasi berakhir
Agent/ZRP set radius_ 3 ;#Besaran radius per zona
```

Kode Sumber 4.17 Pengaturan parameter simulasi

```
#Create a ns simulator
set ns_ [new Simulator]

#Setup topography object
set topo       [new Topography]
$topo load_flatgrid $val(x) $val(y)
create-god $val(nn)

#Open the NS trace file
set tracefile [open $file_name.tr w]
$ns_ trace-all $tracefile

#Open the NAM trace file
set namfile [open $file_name.nam w]
$ns_ namtrace-all $namfile
$ns_ namtrace-all-wireless $namfile $val(x)
$val(y)
set chan [new $val(chan)];#Create wireless channel
```

Kode Sumber 4.18 Pengaturan inisialisasi NS-2

Pada Kode Sumber 4.18, ns_ merupakan variabel baru untuk simulator. Pada NS-2 juga perlu adanya inisialisasi untuk pengaturan topografi. Kemudian jika ingin mencatat hasil dari simulasi, maka digunakan perintah *trace-all* yang hasilnya akan dimasukkan ke dalam file yang berekstensi .tr. Pada NS2 juga disediakan *Network Animator* yang dapat menampilkan visualisasi dari simulasi NS. *Network Animator* atau yang biasa disebut dengan NAM, memiliki sistem kerja yang hampir sama dengan file .tr. Keduanya sama-sama melakukan *trace* terhadap skenario pada NS. Perbedaannya terdapat pada bentuk visual untuk file berekstensi .nam, dan bentuk *text* untuk file berekstensi .tr.

Setelah itu, masukkan variabel yang sudah diatur pada Kode Sumber 4.19 ke dalam pengaturan sebagai berikut.

```
$ns_ node-config
          -adhocRouting $val(rp) \
           -llType $val(ll) \
           -macType $val(mac) \
           -ifqType $val(ifq) \
           -ifqLen $val(ifqlen) \
           -antType $val(ant) \
           -propType $val(prop) \
           -phyType $val(netif) \
           -topoInstance $topo \
           -agentTrace ON \
           -routerTrace ON \
           -macTrace ON \
           -movementTrace ON \
           -channel $chan
```

Kode Sumber 4.19 Pengaturan parameter mobile node

Hal selanjutnya adalah dengan melakukan definisi terhadap node-node yang akan digunakan, penentuan posisi awal, pemberian warna untuk simulasi pada NAM, pemberian label untuk masing-masing node, dan pembacaan file hasil generasi pergerakan node seperti yang telah dijelaskan pada bab 4.3.

Kemudian melakukan definisi untuk koneksi. Koneksi yang digunakan pada kasus ini adalah UDP. Kode program untuk konfigurasi koneksi UDP adalah sebagai berikut.

```
#Setup a UDP connection
set udp0 [new Agent/UDP]
$ns_ attach-agent $node_($val(source)) $udp0
set null1 [new Agent/Null]
$ns_ attach-agent $node_($val(destination)) $null1
$ns_ connect $udp0 $null1
```

Kode Sumber 4.20 Pengaturan koneksi UDP

Dan aplikasi yang digunakan pada koneksi UDP adalah CBR. Didalam pengaturan CBR, terdapat definisi ukuran paket yang dikirimkan oleh sumber, dan interval waktu pengiriman paket. Ukuran paket untuk simulasi ini diatur 100 bytes, interval pengiriman 1 detik. Fungsi pemasangan CBR dapat dilihat pada Kode Sumber 4.21.

```
//Fungsi untuk pemasangan CBR
proc attach-CBR-traffic {node sink size interval}
{
   #Get an instance of the simulator
   set ns_ [Simulator instance]
   #Create a CBR agent and attach it to the node
   set cbr [new Agent/CBR]
   $ns_ attach-agent $node $cbr
   $cbr set packetSize_ $size
   $cbr set interval_ $interval

   #Attach CBR source to sink;
   $ns_ connect $cbr $sink
   return $cbr
  }

//Sintaks pemasangan CBR
set cbr0 [attach-CBR-traffic $node_($val(source))
$sink1 100 1]
```

Kode Sumber 4.21 Fungsi pemasangan CBR dan sintaks pemanggilannya

Setelah CBR dipasang, harus ditentukan juga kapan paket mulai dikirmkan dan kapan paket berhenti untuk dikirimkan. Sintak untuk memulai dan mengakhiri ditunjukkan pada Kode Sumber 4.22.

```
$ns_ at 15.0 "$cbr0 start"
$ns_ at $val(stop) "$cbr0 stop"
```

Kode Sumber 4.22 Sintaks memulai & mengakhiri pengiriman paket

Kemudian untuk mengakhiri simulasi ini, perlu adanya penambahan kode program seperti Kode Sumber 4.23.

Pada Kode Sumber 4.24, ns memanggil fungsi finish yaitu yang berisi penutupan penggunaan file tempat menyimpan data trace .tr dan nam.

Pada akhirnya, implementasi pada NS-2 telah selesai, dan untuk menjalankan skenario tersebut perlu menjalankan kode program pada Kode Sumber 4.24 di terminal.

Ketika skenario selesai dijalankan, maka akan dihasilkan 2 file, 1 file berekstensi .tr sebagai file *traceroute*, 1 file berekstensi .nam untuk *network animator* nya. Untuk melihat animasi jaringannya, silahkan jalankan perintah pada Kode Sumber 4.24 di terminal anda.

```
proc finish {} {
    global ns_ f0 f1 tracefile namfile
    $ns_ flush-trace
    close $tracefile
    close $namfile
    close $f0
    close $f1

    exit 0
}
for {set i 0} {$i < $val(nn)} {incr i} {
    $ns_ at $val(stop) "\$node_($i) reset"
}
$ns_ at $val(stop) "$ns_ nam-end-wireless
$val(stop)"
$ns_ at $val(stop) "finish"
$ns_ at $val(stop) "puts \"done\"; $ns_ halt"
$ns_ run
```

Kode Sumber 4.23 Proses finish dari skenario NS-2

```
$ ns namafile.tcl
$ nam namafile.nam
```

Kode Sumber 4.24 Sintaks untuk menjalankan simulasi NS-2

BAB V
UJI COBA DAN EVALUASI

Pada bab ini akan dijelaskan uji coba yang dilakukan pada aplikasi *Network Simulator 2* (NS-2). Pengujian dilakukan dengan menjalankan skenario simulasi pada aplikasi NS-2. Pengujian fungsionalitas akan dibagi ke dalam beberapa skenario pengujian.

5.1 Lingkungan Uji Coba

Lingkungan uji coba menjelaskan lingkungan yang digunakan untuk menjalankan skenario simulasi pada aplikasi NS-2. Adapun pengujian dilakukan pada sebuah PC yang sudah diinstal OS Ubuntu 14.04 LTS 64 bit didalamnya. Untuk spesifikasi fisik dari PC tersebut tertera pada Tabel 5.1.

Tabel 5.1 Spesifikasi Komputer yang Digunakan

Komponen	**Spesifikasi**
CPU	Processor Intel® Core TM i3-2120 CPU @ 3.30 GHz x 4 core
Sistem Operasi	Ubuntu 14.04 LTS 64-bit
Memori	8 GB DDR3
Media Penyimpanan	HDD 250 GB

5.2 Kriteria Pengujian

Pengujian pada skenario yang dihasilkan oleh NS-2 menggunakan beberapa kriteria. Pada Tabel 5.2 menunjukkan kriteria-kriteria yang ditentukan di dalam skenario.

Tabel 5.2 Krieria Pengujian

Kriteria	**Spesifikasi**
Skenario	MANET
Jumlah Node	10, 20, 30, 40

Kriteria	**Spesifikasi**
Jumlah Node Blackhole	2, 3, 4, 5
Waktu Simulasi	100 Detik
Protokol *Routing*	ZRP

Untuk mendapatkan hasil yang akurat, maka dilakukan pengujian sebanyak sepuluh kali pada kombinasi jumlah node dan jumlah black hole. Kemudian dipilih lima data terbaik (data yang stabil) dan dihitung rata-rata PDR (*Packet Delivery Ratio*), *Delay Time*, dan *Routing Overhead*. Pengujian ini menggunakan penghitungan PDR, *Delay Time*, dan *Routing Overhead*.

5.3 Pengujian

5.3.1 Nilai PDR dengan 2 Blackhole

Nilai PDR dari pengujian pada lingkungan MANET untuk jumlah *blackhole* 2 dengan jumlah node 10, 20, 30, dan 40 dapat dilihat pada Tabel 5.3, Tabel 5.4, Tabel 5.5, dan Tabel 5.6.

Tabel 5.3 PDR dengan 2 Blackhole dan 10 Node

Percobaan Ke	**Dengan Pencegahan (%)**	**Tanpa Pencegahan (%)**
1	31.76	22.35
2	98.82	0.00
3	63.53	20.00
4	60.00	0.00
5	94.12	88.24
Rata-Rata	69.65	26.12

Tabel 5.4 PDR dengan 2 Blackhole dan 20 Node

Percobaan Ke	Dengan Pencegahan (%)	Tanpa Pencegahan (%)
1	71.76	40.00
2	60.00	10.59
3	92.94	43.53
4	60.00	41.18
5	57.65	74.12
Rata-Rata	68.47	41.88

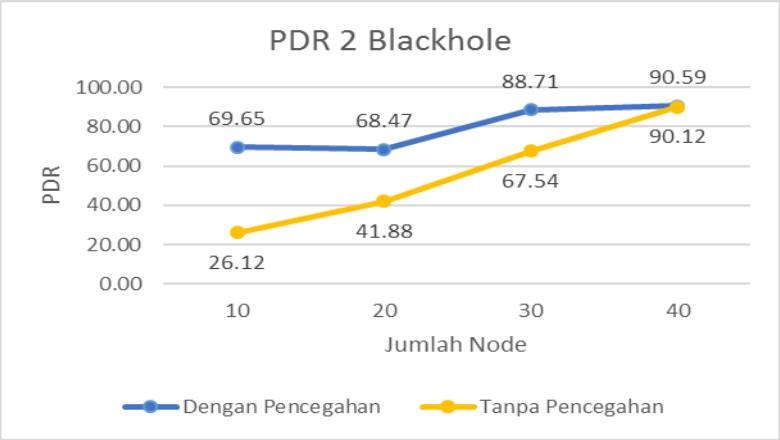
Tabel 5.5 PDR dengan 2 Blackhole dan 30 Node

Percobaan Ke	Dengan Pencegahan (%)	Tanpa Pencegahan (%)
1	85.88	41.18
2	77.65	54.12
3	100.00	100.00
4	91.76	54.18
5	88.24	88.24
Rata-Rata	88.71	67.54

Tabel 5.6 PDR dengan 2 Blackhole dan 40 Node

Percobaan Ke	Dengan Pencegahan (%)	Tanpa Pencegahan (%)
1	85.88	83.53
2	77.65	87.06
3	97.65	92.94
4	100.00	98.82
5	91.76	88.24
Rata-Rata	90.59	90.12

Dari 4 tabel diatas, didapatkan rata-rata untuk setiap skenario dan jika digambarkan dalam grafik garis, akan seperti berikut.



Gambar 5.1 Grafik rata-rata untuk PDR dengan jumlah blackhole 2

5.3.2 Nilai PDR dengan 3 Blackhole

Nilai PDR dari pengujian pada lingkungan MANET untuk jumlah *blackhole* 3 dengan jumlah node 10, 20, 30, dan 40 dapat dilihat pada Tabel 5.7, Tabel 5.8, Tabel 5.9, dan Tabel 5.10.

Tabel 5.7 PDR dengan 3 Blackhole dan 10 Node

Percobaan Ke	**Dengan Pencegahan (%)**	**Tanpa Pencegahan (%)**
1	0.00	0.00
2	98.82	0.00
3	44.71	20.00
4	0.00	0.00
5	0.00	0.00
Rata-Rata	28.71	4.00

Tabel 5.8 PDR dengan 3 Blackhole dan 20 Node

Percobaan Ke	**Dengan Pencegahan (%)**	**Tanpa Pencegahan (%)**
1	75.29	41.18
2	14.12	9.41
3	84.71	43.53
4	42.35	41.18
5	32.94	72.94

Percobaan Ke	Dengan Pencegahan (%)	Tanpa Pencegahan (%)
Rata-Rata	49.88	41.65

Tabel 5.9 PDR dengan 3 Blackhole dan 30 Node

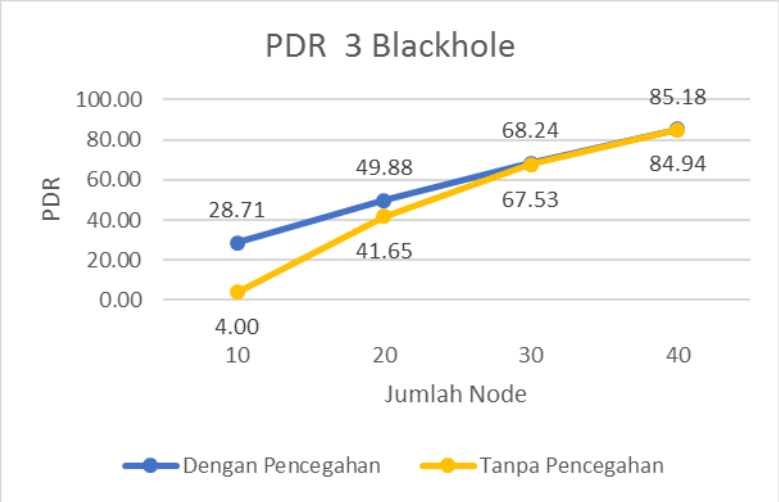
Percobaan Ke	Dengan Pencegahan (%)	Tanpa Pencegahan (%)
1	34.12	41.18
2	61.18	54.12
3	100.00	100.00
4	64.71	54.12
5	81.18	88.24
Rata-Rata	68.24	67.53

Tabel 5.10 PDR dengan 3 Blackhole dan 40 Node

Percobaan Ke	Dengan Pencegahan (%)	Tanpa Pencegahan (%)
1	84.71	83.53
2	69.41	82.35
3	84.71	84.71
4	100.00	98.82
5	87.06	75.29

Percobaan Ke	Dengan Pencegahan (%)	Tanpa Pencegahan (%)
Rata-Rata	85.18	84.94

Dari 4 tabel diatas, didapatkan rata-rata untuk setiap skenario dan jika digambarkan dalam grafik garis, akan seperti berikut.



Gambar 5.2 Grafik rata-rata untuk PDR dengan jumlah blackhole 3

5.3.3 Nilai PDR dengan 4 Blackhole

Nilai PDR dari pengujian pada lingkungan MANET untuk jumlah *blackhole* 4 dengan jumlah node 10, 20, 30, dan 40 dapat dilihat pada Tabel 5.11, Tabel 5.12, Tabel 5.13, dan Tabel 5.14.

Tabel 5.11 PDR dengan 4 Blackhole dan 10 Node

Percobaan Ke	Dengan Pencegahan (%)	Tanpa Pencegahan (%)
1	0.00	0.00
2	98.82	0.00
3	44.71	20.00
4	0.00	0.00
5	8.24	0.00
Rata-Rata	30.35	4.00

Tabel 5.12 PDR dengan 4 Blackhole dan 20 Node

Percobaan Ke	Dengan Pencegahan (%)	Tanpa Pencegahan (%)
1	91.76	41.18
2	1.18	1.18
3	36.47	44.71
4	40.00	41.18
5	37.65	18.82
Rata-Rata	41.41	29.41

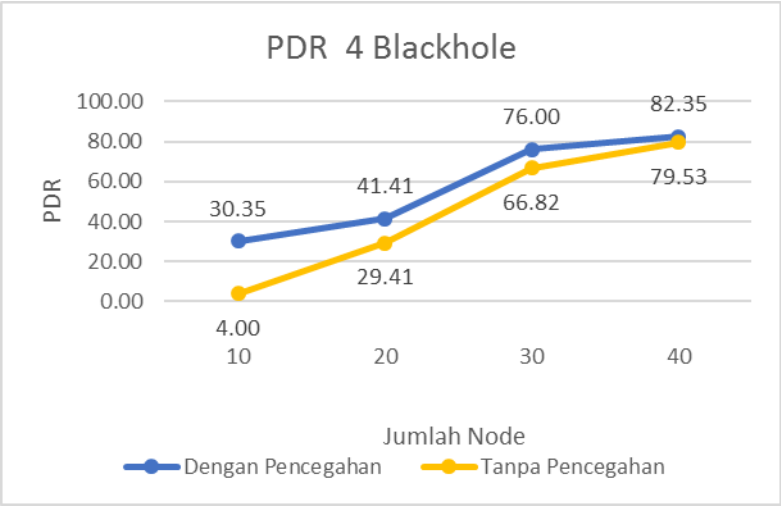
Tabel 5.13 PDR dengan 4 Blackhole dan 30 Node

Percobaan Ke	Dengan Pencegahan (%)	Tanpa Pencegahan (%)
1	47.06	41.18
2	63.53	54.12
3	100.00	100.00
4	77.65	50.58
5	91.76	88.24
Rata-Rata	76.00	66.82

Tabel 5.14 PDR dengan 4 Blackhole dan 40 Node

Percobaan ke	Dengan Pencegahan (%)	Tanpa Pencegahan (%)
1	84.71	83.53
2	85.88	58.82
3	89.41	84.71
4	100.00	98.82
5	51.76	71.76
Rata-Rata	82.35	79.53

Dari 4 tabel diatas, didapatkan rata-rata untuk setiap skenario dan jika digambarkan dalam grafik garis, akan seperti berikut.



Gambar 5.3 Grafik rata-rata untuk PDR dengan jumlah blackhole 4

5.3.4 Nilai PDR dengan 5 Blackhole

Nilai PDR dari pengujian pada lingkungan MANET untuk jumlah *blackhole* 5 dengan jumlah node 10, 20, 30, dan 40 dapat dilihat pada Tabel 5.15, Tabel 5.16, Tabel 5.17, dan Tabel 5.18.

Tabel 5.15 PDR dengan 5 Blackhole dan 10 Node

Percobaan Ke	Dengan Pencegahan (%)	Tanpa Pencegahan (%)
1	0.00	0.00
2	0.00	0.00
3	34.12	0.00
4	0.00	7.06
5	8.24	0.00

Percobaan Ke	Dengan Pencegahan (%)	Tanpa Pencegahan (%)
Rata-Rata	8.47	1.41

Tabel 5.16 PDR dengan 5 Blackhole dan 20 Node

Percobaan Ke	Dengan Pencegahan (%)	Tanpa Pencegahan (%)
1	67.06	38.82
2	11.76	1.18
3	3.53	5.88
4	35.29	28.24
5	37.65	18.82
Rata-Rata	31.06	18.59

Tabel 5.17 PDR dengan 5 Blackhole dan 30 Node

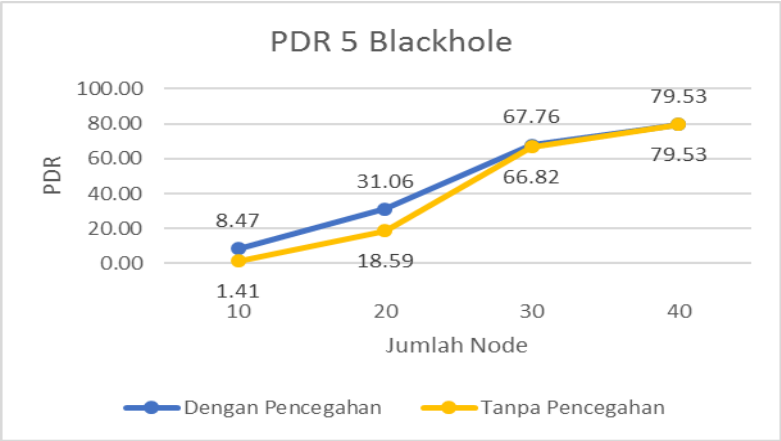
Percobaan Ke	Dengan Pencegahan (%)	Tanpa Pencegahan (%)
1	60.00	41.18
2	45.88	54.12
3	85.88	100.00
4	96.47	50.58
5	50.59	88.24

Percobaan Ke	Dengan Pencegahan (%)	Tanpa Pencegahan (%)
Rata-Rata	67.76	66.82

Tabel 5.18 PDR dengan 5 Blackhole dan 40 Node

Percobaan Ke	Dengan Pencegahan (%)	Tanpa Pencegahan (%)
1	83.53	83.53
2	78.82	58.82
3	78.82	84.71
4	90.59	98.82
5	65.88	71.76
Rata-Rata	79.53	79.53

Dari 4 tabel diatas, didapatkan rata-rata untuk setiap skenario dan jika digambarkan dalam grafik garis, akan seperti berikut.



Gambar 5.4 Grafik rata-rata untuk PDR dengan jumlah blackhole 5

5.3.5 Delay Time dengan 2 Blackhole

Rata-rata delay time untuk paket sampai ke tujuan dari pengujian pada lingkungan MANET untuk jumlah *blackhole* 2 dengan jumlah node 10, 20, 30, dan 40 dapat dilihat pada Tabel 5.19, Tabel 5.20, Tabel 5.21, dan Tabel 5.22.

Tabel 5.19 Delay Time dengan 2 Blackhole dan 10 Node

Percobaan Ke	Dengan Pencegahan (s)	Tanpa Pencegahan (s)
1	0.01	0.06
2	0.01	0.00
3	0.01	0.01
4	0.01	0.00
5	0.01	0.03

Percobaan Ke	Dengan Pencegahan (s)	Tanpa Pencegahan (s)
Rata-Rata	0.01	0.02

Tabel 5.20 Delay Time dengan 2 Blackhole dan 20 Node

Percobaan Ke	Dengan Pencegahan (s)	Tanpa Pencegahan (s)
1	0.02	0.01
2	0.01	0.03
3	0.02	0.02
4	0.02	0.01
5	0.01	0.01
Rata-Rata	0.02	0.02

Tabel 5.21 Delay Time dengan 2 Blackhole dan 30 Node

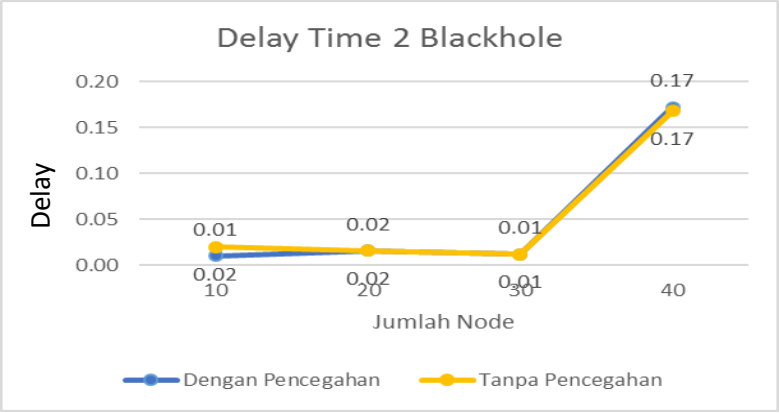
Percobaan Ke	Dengan Pencegahan (s)	Tanpa Pencegahan (s)
1	0.02	0.02
2	0.03	0.03
3	0.02	0.02
4	0.03	0.02
5	0.02	0.02

Percobaan Ke	Dengan Pencegahan (s)	Tanpa Pencegahan (s)
Rata-Rata	0.02	0.02

Tabel 5.22 Delay Time dengan 2 Blackhole dan 40 Node

Percobaan Ke	Dengan Pencegahan (s)	Tanpa Pencegahan (s)
1	0.19	0.11
2	0.15	0.08
3	0.15	0.5
4	0.21	0.08
5	0.16	0.07
Rata-Rata	0.17	0.17

Dari 4 tabel diatas, didapatkan rata-rata untuk setiap skenario dan jika digambarkan dalam grafik garis, akan seperti berikut.



Gambar 5.5 Grafik rata-rata untuk Delay dengan jumlah blackhole 2

5.3.6 Delay Time dengan 3 Blackhole

Rata-rata delay time untuk paket sampai ke tujuan dari pengujian pada lingkungan MANET untuk jumlah *blackhole* 3 dengan jumlah node 10, 20, 30, dan 40 dapat dilihat pada Tabel 5.23, Tabel 5.24, Tabel 5.25, dan Tabel 5.26

Tabel 5.23 Delay Time dengan 3 Blackhole dan 10 Node

Percobaan Ke	Dengan Pencegahan (s)	Tanpa Pencegahan (s)
1	0.00	0.00
2	0.01	0.00
3	0.01	0.01
4	0.00	0.00
5	0.00	0.00

Percobaan Ke	Dengan Pencegahan (s)	Tanpa Pencegahan (s)
Rata-Rata	0.00	0.00

Tabel 5.24 Delay Time dengan 3 Blackhole dan 20

Percobaan Ke	Dengan Pencegahan (s)	Tanpa Pencegahan (s)
1	0.01	0.01
2	0.01	0.01
3	0.02	0.02
4	0.01	0.01
5	0.01	0.01
Rata-Rata	0.01	0.01

Tabel 5.25 Delay Time dengan 3 Blackhole dan 30

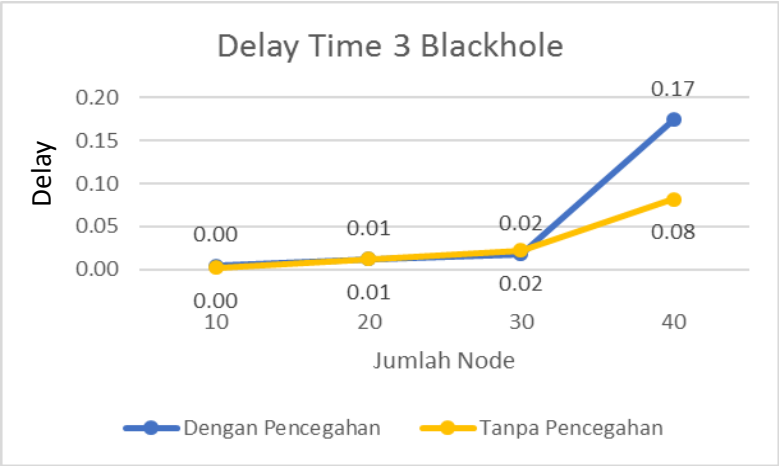
Percobaan Ke	Dengan Pencegahan (s)	Tanpa Pencegahan (s)
1	0.02	0.02
2	0.02	0.03
3	0.01	0.02
4	0.02	0.02
5	0.02	0.02

Rata-Rata	0.02	0.02

Tabel 5.26 Delay Time dengan 3 Blackhole dan 40

Percobaan Ke	**Dengan Pencegahan (s)**	**Tanpa Pencegahan (s)**
1	0.17	0.11
2	0.23	0.09
3	0.14	0.06
4	0.17	0.08
5	0.16	0.07
Rata-Rata	0.17	0.08

Dari 4 tabel diatas, didapatkan rata-rata untuk setiap skenario dan jika digambarkan dalam grafik garis, akan seperti berikut.



Gambar 5.6 Grafik rata-rata untuk Delay dengan jumlah blackhole 3

5.3.7 Delay Time dengan 4 Blackhole

Rata-rata delay time untuk paket sampai ke tujuan dari pengujian pada lingkungan MANET untuk jumlah *blackhole* 4 dengan jumlah node 10, 20, 30, dan 40 dapat dilihat pada Tabel 5.27, Tabel 5.28, Tabel 5.29, dan Tabel 5.30.

Tabel 5.27 Delay Time dengan 4 Blackhole dan 10 Node

Percobaan Ke	Dengan Pencegahan (s)	Tanpa Pencegahan (s)
1	0.00	0.00
2	0.01	0.00
3	0.01	0.01
4	0.00	0.00
5	0.01	0.00

Percobaan Ke	Dengan Pencegahan (s)	Tanpa Pencegahan (s)
Rata-Rata	0.01	0.00

Tabel 5.28 Delay Time dengan 4 Blackhole dan 20 Node

Percobaan Ke	Dengan Pencegahan (s)	Tanpa Pencegahan (s)
1	0.01	0.01
2	0.01	0.02
3	0.01	0.01
4	0.01	0.01
5	0.01	0.01
Rata-Rata	0.01	0.01

Tabel 5.29 Delay Time dengan 4 Blackhole dan 30 Node

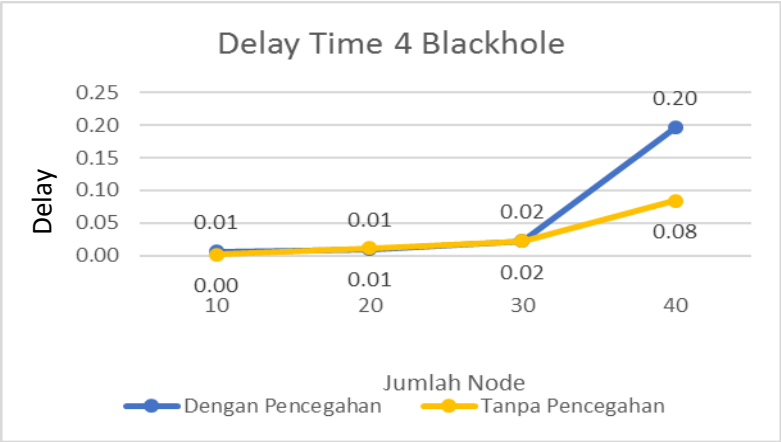
Percobaan Ke	Dengan Pencegahan (s)	Tanpa Pencegahan (s)
1	0.02	0.02
2	0.03	0.03
3	0.02	0.02
4	0.02	0.02
5	0.02	0.02

Percobaan Ke	Dengan Pencegahan (s)	Tanpa Pencegahan (s)
Rata-Rata	0.02	0.02

Tabel 5.30 Delay Time dengan 4 Blackhole dan 40 Node

Percobaan Ke	Dengan Pencegahan (s)	Tanpa Pencegahan (s)
1	0.2	0.11
2	0.23	0.1
3	0.18	0.06
4	0.18	0.08
5	0.19	0.07
Rata-Rata	0.20	0.08

Dari 4 tabel diatas, didapatkan rata-rata untuk setiap skenario dan jika digambarkan dalam grafik garis, akan seperti berikut.



Gambar 5.7 Grafik rata-rata untuk Delay dengan jumlah blackhole 4

5.3.8 Delay Time dengan 5 Blackhole

Rata-rata delay time untuk paket sampai ke tujuan dari pengujian pada lingkungan MANET untuk jumlah *blackhole* 5 dengan jumlah node 10, 20, 30, dan 40 dapat dilihat pada Tabel 5.31, Tabel 5.32, Tabel 5.33, dan Tabel 5.34.

Tabel 5.31 Delay Time dengan 5 Blackhole dan 10 Node

Percobaan Ke	Dengan Pencegahan (s)	Tanpa Pencegahan (s)
1	0.00	0.00
2	0.00	0.00
3	0.01	0.01
4	0.00	0.00
5	0.01	0.00

Percobaan Ke	Dengan Pencegahan (s)	Tanpa Pencegahan (s)
Rata-Rata	0.00	0.00

Tabel 5.32 Delay Time dengan 5 Blackhole dan 20 Node

Percobaan Ke	Dengan Pencegahan (s)	Tanpa Pencegahan (s)
1	0.01	0.01
2	0.01	0.01
3	0.02	0.01
4	0.01	0.01
5	0.01	0.01
Rata-Rata	0.01	0.01

Tabel 5.33 Delay Time dengan 5 Blackhole dan 30 Node

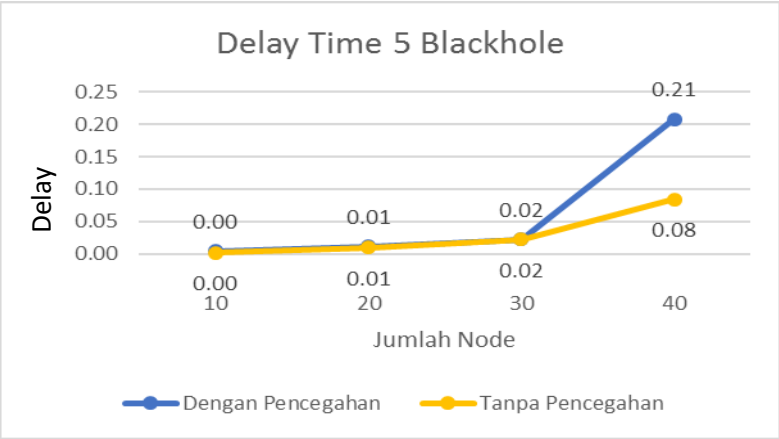
Percobaan Ke	Dengan Pencegahan (s)	Tanpa Pencegahan (s)
1	0.02	0.02
2	0.03	0.03
3	0.02	0.02
4	0.02	0.02
5	0.02	0.02

Rata-Rata	0.02	0.02

Tabel 5.34 Delay Time dengan 5 Blackhole dan 40 Node

Percobaan Ke	**Dengan Pencegahan (s)**	**Tanpa Pencegahan (s)**
1	0.24	0.11
2	0.29	0.1
3	0.16	0.06
4	0.15	0.08
5	0.2	0.07
Rata-Rata	0.21	0.08

Dari 4 tabel diatas, didapatkan rata-rata untuk setiap skenario dan jika digambarkan dalam grafik garis, akan seperti berikut.



Gambar 5.8 Grafik rata-rata untuk Delay dengan jumlah blackhole 5

5.3.9 Besar Routing Overhead dengan 2 Blackhole

Rata-rata besar routing overhead untuk pembentukan rute dari pengujian pada lingkungan MANET untuk jumlah *blackhole* 2 dengan jumlah node 10, 20, 30, dan 40 dapat dilihat pada Tabel 5.35, Tabel 5.36, Tabel 5.37, dan Tabel 5.38.

Tabel 5.35 Routing Overhead dengan 2 Blackhole dan 10 Node

Percobaan Ke	Dengan Pencegahan	Tanpa Pencegahan
1	3677	2613
2	4220	3035
3	3903	2829
4	2894	2163
5	3196	2494

Percobaan Ke	Dengan Pencegahan	Tanpa Pencegahan
Rata-Rata	3578	2627

Tabel 5.36 Routing Overhead dengan 2 Blackhole dan 20 Node

Percobaan Ke	Dengan Pencegahan	Tanpa Pencegahan
1	13041	9701
2	12577	9186
3	12445	9169
4	11310	8873
5	12157	8927
Rata-Rata	12306	9171

Tabel 5.37 Routing Overhead dengan 2 Blackhole dan 30 Node

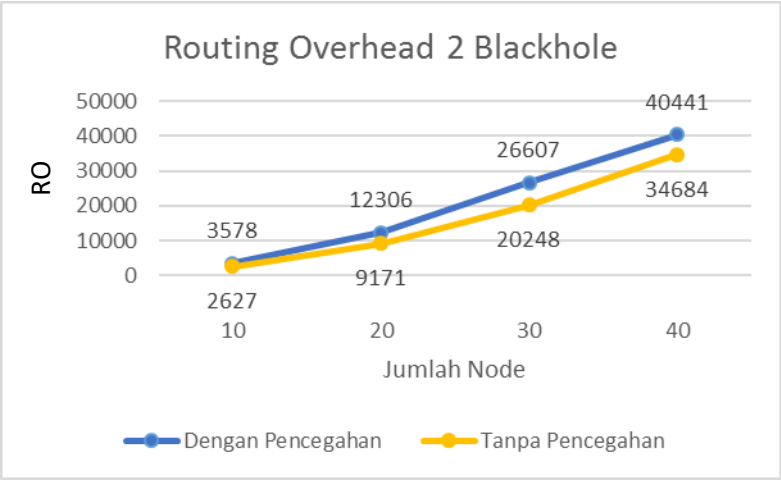
Percobaan Ke	Dengan Pencegahan	Tanpa Pencegahan
1	24029	19422
2	28615	22225
3	27932	19482
4	27932	21601

Percobaan Ke	Dengan Pencegahan	Tanpa Pencegahan
5	24528	18510
Rata-Rata	26607	20248

Tabel 5.38 Routing Overhead dengan 2 Blackhole dan 40 Node

Percobaan Ke	Dengan Pencegahan	Tanpa Pencegahan
1	42311	35913
2	40150	33959
3	39474	34326
4	40718	34610
5	39552	34610
Rata-Rata	40441	34684

Dari 4 tabel diatas, didapatkan rata-rata untuk setiap skenario dan jika digambarkan dalam grafik garis, akan seperti berikut.



Gambar 5.9 Grafik rata-rata Routing Overhead dengan blackhole 2

5.3.10 Besar Routing Overhead dengan 3 Blackhole

Rata-rata besar routing overhead untuk pembentukan rute dari pengujian pada lingkungan MANET untuk jumlah *blackhole* 3 dengan jumlah node 10, 20, 30, dan 40 dapat dilihat pada Tabel 5.39, Tabel 5.40, Tabel 5.41, dan Tabel 5.42.

Tabel 5.39 Routing Overhead dengan 3 Blackhole dan 10 Node

Percobaan Ke	Dengan Pencegahan	Tanpa Pencegahan
1	3564	2631
2	4270	3035
3	3857	2829
4	2809	2163

Percobaan Ke	Dengan Pencegahan	Tanpa Pencegahan
5	3215	2357
Rata-Rata	3543	2603

Tabel 5.40 Routing Overhead dengan 3 Blackhole dan 20 Node

Percobaan Ke	Dengan Pencegahan	Tanpa Pencegahan
1	12555	9764
2	11528	9051
3	12576	9169
4	11573	8889
5	11826	8927
Rata-Rata	12012	9160

Tabel 5.41 Routing Overhead dengan 3 Blackhole dan 30 Node

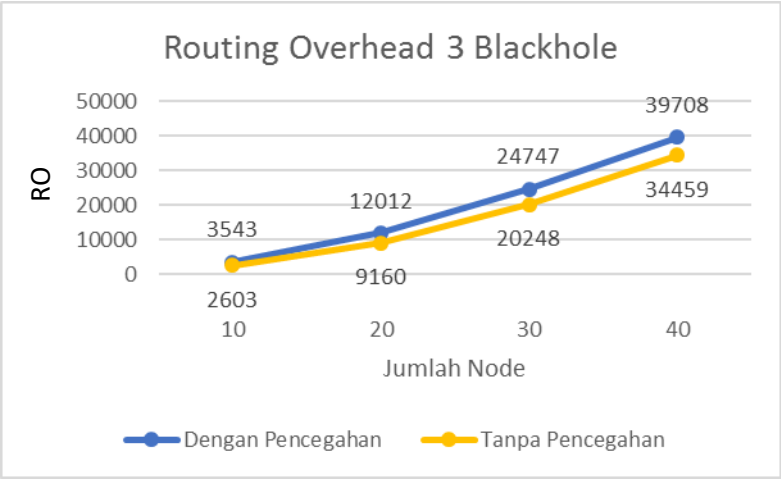
Percobaan Ke	Dengan Pencegahan	Tanpa Pencegahan
1	24447	19422
2	28078	22225
3	24154	19482
4	25755	21601

Percobaan Ke	Dengan Pencegahan	Tanpa Pencegahan
5	21299	18510
Rata-Rata	24747	20248

Tabel 5.42 Routing Overhead dengan 3 Blackhole dan 40 Node

Percobaan Ke	Dengan Pencegahan	Tanpa Pencegahan
1	40204	35913
2	40376	33958
3	39180	34003
4	39079	34610
5	39700	33809
Rata-Rata	39708	34459

Dari 4 tabel diatas, didapatkan rata-rata untuk setiap skenario dan jika digambarkan dalam grafik garis, akan seperti berikut.



Gambar 5.10 Grafik rata-rata Routing Overhead dengan blackhole 3

5.3.11 Besar Routing Overhead dengan 4 Blackhole

Rata-rata besar routing overhead untuk pembentukan rute dari pengujian pada lingkungan MANET untuk jumlah *blackhole* 4 dengan jumlah node 10, 20, 30, dan 40 dapat dilihat pada Tabel 5.43, Tabel 5.44, Tabel 5.45, dan Tabel 5.46.

Tabel 5.43 Routing Overhead dengan 4 Blackhole dan 10 Node

Percobaan Ke	Dengan Pencegahan	Tanpa Pencegahan
1	3192	2631
2	3843	3035
3	3843	2829
4	2822	2163

Percobaan Ke	Dengan Pencegahan	Tanpa Pencegahan
5	3301	2357
Rata-Rata	3400	2603

Tabel 5.44 Routing Overhead dengan 4 Blackhole dan 20 Node

Percobaan Ke	Dengan Pencegahan	Tanpa Pencegahan
1	12545	9764
2	11483	9005
3	12567	9192
4	11539	8882
5	11896	8927
Rata-Rata	12006	9154

Tabel 5.45 Routing Overhead dengan 4 Blackhole dan 30 Node

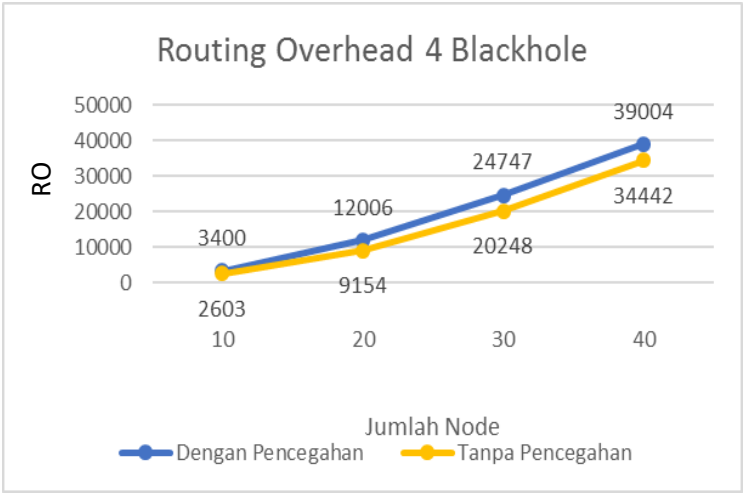
Percobaan Ke	Dengan Pencegahan	Tanpa Pencegahan
1	24826	19422
2	28078	22225
3	23710	19482
4	25436	21651

Percobaan Ke	Dengan Pencegahan	Tanpa Pencegahan
5	21191	18510
Rata-Rata	24648	20258

Tabel 5.46 Routing Overhead dengan 4 Blackhole dan 40 Node

Percobaan Ke	Dengan Pencegahan	Tanpa Pencegahan
1	39668	35913
2	39224	33858
3	38357	34003
4	38153	34610
5	39620	33824
Rata-Rata	39004	34442

Dari 4 tabel diatas, didapatkan rata-rata untuk setiap skenario dan jika digambarkan dalam grafik garis, akan seperti berikut.



Gambar 5.11 Grafik rata-rata Routing Overhead dengan blackhole 4

5.3.12 Besar Routing Overhead dengan 5 Blackhole

Rata-rata besar routing overhead untuk pembentukan rute dari pengujian pada lingkungan MANET untuk jumlah *blackhole* 5 dengan jumlah node 10, 20, 30, dan 40 dapat dilihat pada Tabel 5.47, Tabel 5.48, Tabel 5.49, dan Tabel 5.50.

Tabel 5.47 Routing Overhead dengan 5 Blackhole dan 10 Node

Percobaan Ke	Dengan Pencegahan	Tanpa Pencegahan
1	3192	2631
2	4187	3035
3	3813	2789
4	2822	2163

Percobaan Ke	Dengan Pencegahan	Tanpa Pencegahan
5	3288	2357
Rata-Rata	3460	2595

Tabel 5.48 Routing Overhead dengan 5 Blackhole dan 20 Node

Percobaan Ke	Dengan Pencegahan	Tanpa Pencegahan
1	12642	9762
2	11491	8931
3	12504	9164
4	11623	8882
5	12032	8881
Rata-Rata	12058	9124

Tabel 5.49 Routing Overhead dengan 5 Blackhole dan 30 Node

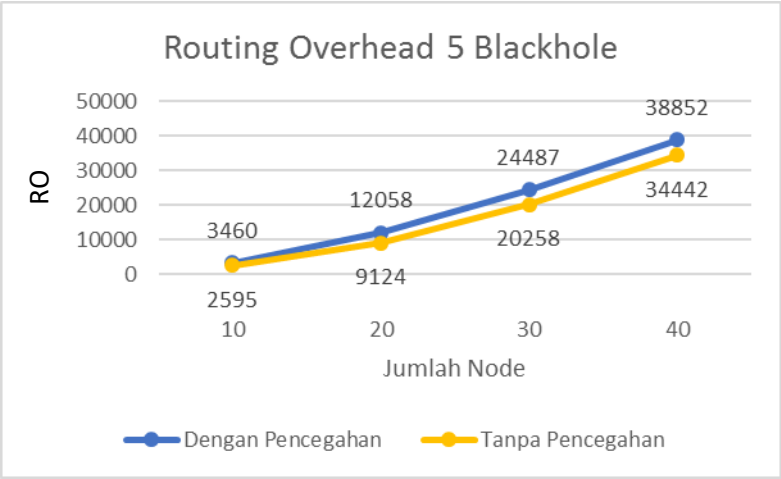
Percobaan Ke	Dengan Pencegahan	Tanpa Pencegahan
1	24275	19422
2	27670	22225
3	24107	19482
4	25557	21651

Percobaan Ke	Dengan Pencegahan	Tanpa Pencegahan
5	20824	18510
Rata-Rata	24487	20258

Tabel 5.50 Routing Overhead dengan 5 Blackhole dan 40 Node

Percobaan Ke	Dengan Pencegahan	Tanpa Pencegahan
1	39131	35913
2	39087	33858
3	38205	34003
4	38356	34610
5	39481	33824
Rata-Rata	38852	34442

Dari 4 tabel diatas, didapatkan rata-rata untuk setiap skenario dan jika digambarkan dalam grafik garis, akan seperti berikut.



Gambar 5.12 Grafik rata-rata Routing Overhead dengan blackhole 5

5.3.13 Perubahan Kenaikan PDR

Dari hasil simulasi perhitungan PDR pada tabel diatas, dapat disimpulkan perbandingan perubahan kenaikan PDR tanpa pencegahan dan dengan pencegahan dapat dilihat pada Tabel 5.51, Tabel 5.52, Tabel 5.53, dan Tabel 5.54.

Tabel 5.51 Perubahan Kenaikan PDR pada 2 Blackhole

No.	Jumlah Node	Kenaikan PDR (%)
1.	10	69.65
2.	20	26.59
3.	30	21.16
4.	40	0.47

Tabel 5.52 Perubahan Kenaikan PDR pada 3 Blackhole

No.	Jumlah Node	Kenaikan PDR (%)
1.	10	24.71
2.	20	8.23
3.	30	0.71
4.	40	0.24

Tabel 5.53 Perubahan Kenaikan PDR pada 4 Blackhole

No.	Jumlah Node	Kenaikan PDR (%)
1.	10	26.35
2.	20	12.00
3.	30	9.18
4.	40	2.82

Tabel 5.54 Perubahan Kenaikan PDR pada 5 Blackhole

No.	Jumlah Node	Kenaikan PDR (%)
1.	10	7.06
2.	20	12.47
3.	30	0.94
4.	40	0.00

5.3.14 Perubahan Kenaikan Delay Time

Dari hasil simulasi perhitungan *Delay Time* pada tabel diatas, dapat disimpulkan perbandingan perubahan kenaikan *Delay Time* tanpa pencegahan dan dengan pencegahan dapat dilihat pada Tabel 5.55, Tabel 5.56, Tabel 5.57, dan Tabel 5.58.

Tabel 5.55 Perubahan Kenaikan Delay Time pada 2 Blackhole

No.	Jumlah Node	Kenaikan Delay Time (s)
1.	10	-0.01
2.	20	0.00
3.	30	0.00
4.	40	0.00

Tabel 5.56 Perubahan Kenaikan Delay Time pada 3 Blackhole

No.	Jumlah Node	Kenaikan Delay Time (s)
1.	10	0.00
2.	20	0.00
3.	30	0.00
4.	40	0.09

Tabel 5.57 Perubahan Kenaikan Delay Time pada 4 Blackhole

No.	Jumlah Node	Kenaikan Delay Time (s)
1.	10	0.00
2.	20	0.00

No.	Jumlah Node	Kenaikan Delay Time (s)
3.	30	0.00
4.	40	0.11

Tabel 5.58 Perubahan Kenaikan Delay Time pada 5 Blackhole

No.	Jumlah Node	Kenaikan Delay Time (s)
1.	10	0.00
2.	20	0.00
3.	30	0.00
4.	40	0.12

5.3.15 Perubakan Kenaikan Routing Overhead

Dari hasil simulasi perhitungan *Routing Overhead* pada tabel diatas, dapat disimpulkan perbandingan perubahan kenaikan *Routing Overhead* tanpa pencegahan dan dengan pencegahan dapat dilihat pada Tabel 5.59, Tabel 5.60, Tabel 5.61, dan Tabel 5.62.

Tabel 5.59 Perubahan Kenaikan Routing Overhead pada 2 Blackhole

No.	Jumlah Node	Kenaikan Routing Overhead
1.	10	3578
2.	20	3135
3.	30	6359
4.	40	5757

Tabel 5.60 Perubahan Kenaikan Routing Overhead pada 3 Blackhole

No.	Jumlah Node	Kenaikan Routing Overhead
1.	10	940
2.	20	2852
3.	30	4499
4.	40	5249

Tabel 5.61 Perubahan Kenaikan Routing Overhead pada 4 Blackhole

No.	Jumlah Node	Kenaikan Routing Overhead
1.	10	797
2.	20	2852
3.	30	4390
4.	40	4563

Tabel 5.62 Perubahan Kenaikan Routing Overhead pada 5 Blackhole

No.	Jumlah Node	Kenaikan Routing Overhead
1.	10	865
2.	20	2934
3.	30	4229
4.	40	4410

5.3.16 Evaluasi Kelemahan Modifikasi ZRP

Pada beberapa kasus tertentu, terjadi sebuah hasil yang perlu diperhatikan. Pada Tabel 5.63, adalah salah satu contoh adanya hasil yang berbeda dengan yang diharapkan.

Tabel 5.63 Contoh kasus simulasi yang hasilnya tidak sesuai harapan

Kasus	Dengan Pencegahan			Tanpa Pencegahan		
	PDR	Delay	RO	PDR	Delay	RO
Percobaan Ke-3 Untuk kasus 20 Node 4 Blackhole	36.47	0.01	12567	44.71	0.01	9192

Hal ini mungkin saja terjadi, dikarenakan node yang di *generate* secara *random*, sehingga pergerakan node menjadi acak dan ini menjadi salah satu faktor utama penyebab adanya hasil yang tidak sesuai dengan harapan.

Pertama, hal ini bisa terjadi karena faktor simulasi yang mungkin saja tidak melewati *blackhole* sama sekali karena pergerakannya yang *random*, sehingga simulasi ketika ada pencegahan akan jauh lebih lambat karena ada pengecekan terlebih

dahulu apakah node tersebut *malicious* atau tidak, dibandingkan tanpa pencegahan yang tidak ada pengecekan.

Lalu, hal ini juga bisa disebabkan oleh karena ketika memang saat simulasi, jarak terpendek menuju node tujuan adalah lewat *blackhole*. Secara langsung, ketika ada pencegahan, maka akan dicari rute selain itu karena node tersebut tidak disambungkan. Mungkin saja, alternatif rute pengalihan agar terhindar dari *malicious node* menjadi sangat banyak sekali lompatannya, sehingga selama proses pengiriman paket berlangsung, banyak sekali paket yang akhirnya di *drop* mungkin karena paket sudah *expire* atau kapasitas node yang tidak bisa menampung banyak paket.

Hal ini juga mungkin terjadi ketika *blackhole* berada pada posisi *peripheral node*. Karena *blackhole* berada pada posisi node perifer, maka node tersebut tidak akan dimasukkan kedalam daftar node perifer, maka secara otomatis, sambungan rute yang berada diluar zona harus dilakukan pada node perifer lain yang mungkin saja menyebabkan paket harus dikirim lewat zona yang jauh dari zona node destinasi.

[Halaman ini sengaja dikosongkan]

BAB VI
KESIMPULAN DAN SARAN

Bab ini membahas mengenai kesimpulan yang dapat diambil dari hasil uji coba yang telah dilakukan pada bab sebelumnya, yaitu Bab Uji Coba dan Evaluasi. Bab ini juga digunakan sebagai jawaban dari rumusan masalah yang dikemukakan pada Bab Pendahuluan. Selain kesimpulan, juga terdapat saran yang ditujukan untuk pengembangan penelitian lebih lanjut.

6.1 Kesimpulan

Dari hasil uji coba yang telah dilakukan dapat diambil beberapa kesimpulan sebagai berikut:

1. Dari hasil uji coba dapat disimpulkan bahwa setelah adanya metode pencegahan, nilai PDR dapat lebih tinggi dari nilai PDR tanpa pencegahan yaitu sebesar 69.65% (diambil nilai kenaikan tertinggi), karena pesan yang di transmisi sampai ke tujuan lebih banyak daripada tanpa pencegahan yang ada beberapa pesan yang masuk ke blackhole sehingga tidak terkirim dengan sempurna
2. Dari hasil uji coba, didapatkan hasil *delay* yang lebih tinggi ketika menggunakan pencegahan daripada tanpa pencegahan yaitu sebesar 0.12 s (diambil nilai kenaikan *delay* tertinggi). Hal ini terjadi karena ketika ada pencegahan, dibutuhkan waktu proses lebih lama untuk mengecek apakah node tersebut *blackhole* atau bukan. Dan ketika ada pencegahan, maka akan dialihkan ke rute yang lebih aman, yang menyebabkan penggunaan rute bukan rute yang tercepat, tetapi rute yang paling aman.
3. Dari hasil uji coba, didapatkan *routing overhead* yang lebih tinggi ketika menggunakan pencegahan daripada tidak dengan nilai 6359 (diambil nilai kenaikan *routing overhead* tertinggi). Hal ini terjadi karena ketika ada *blackhole*, maka akan dicari rute lain, sehingga

banyaknya fase pencarian rute menjadi lebih banyak dibandingkan tanpa pencegahan.

6.2 Saran

Saran yang diberikan untuk pengembangan sistem ini adalah:

1. Perlu adanya optimasi pada bagian pengecekan apakah sebuah node adalah *blackhole* atau tidak, karena pendeteksian masih menggunakan fungsi perulangan sehingga menyebabkan butuh lebih banyak waktu untuk pengecekan dan bisa menyebabkan pesan memasuki masa *expire* sebelum sampai tujuannya.
2. Perlu adanya pengembangan pada bagian kapasitas *blacklist*, yang untuk Tugas Akhir ini dibatasi hanya mampu menampung blackhole sebanyak maksimal 10 node. Serta jenis *blacklist* yang masih *array* dengan kapasitas terbatas, karena pada studi kasus yang nyata, jumlah *blackhole* bisa saja banyak dan beragam.
3. Perlu adanya pengenalan sifat-sifat *malicious node* yang ada pada dunia nyata untuk pendeteksian apakah alamat node tersebut bisa dimasukkan ke dalam daftar *blacklist*, sehingga pendaftaran *blackhole* pada daftar *blacklist* adalah sesuai dengan sifat-sifat asli *malicious node*.
4. Perlu dikembangkan lagi dan dipelajari kembali untuk jenis-jenis *malicious node* selain *blackhole*, Karena pada dunia nyata masih banyak sekali jenis-jenis *malicious node* dengan beragam karakteristiknya.

DAFTAR PUSTAKA

[1] N. Lal, S. Kumar, A. Saxena dan V. K. Chaurasiya, "Detection of Malicious Node Behaviour Via I-Watchdog Protocol in Mobile Ad-Hoc Network with DSDV Routing Scheme," dalam *4th International Conference on Advances in Computing, Communication and Control (ICAC3 '15)*, Allahabad, India, 2015.

[2] J. K. Mandal dan K. L. Hassan, "Analysis of Black Hole and Gray Hole Attack in MANET based on Simulation through NS2," *AMSE JOURNALS* , vol. 20, no. 1, pp. 35-46, 2015.

[3] Information Scientist Institute, "The Network Simulator - ns-2," Information Scientist Institute, 19 December 2014. [Online]. Available: http://www.isi.edu/nsnam/ns/. [Diakses 23 May 2017].

[4] Information Scientist Institute, "XI. Generating node-movement and traffic-connection files for large wireless scenarios," Information Scientist Institute, 19 December 2014. [Online]. Available: http://www.isi.edu/nsnam/ns/tutorial/nsscript7.html. [Diakses 23 May 2017].

[5] J. S. Patil dan K. V. N. Sunitha, "A combined technique for attack monitoring and risk assessment in MANET routing," dalam *2016 Thirteenth International Conference on Wireless and Optical Communications Networks (WOCN)*, Telangana State, 2016.

[6] S. Kaur dan S. Kaur, "ANALYSIS OF ZONE ROUTING PROTOCOL IN MANET," *International Journal of Research in Engineering and Technology*, vol. 02, no. 09, p. 520, 2013.

[7] B. Kannhavong, H. Nakayama, Y. Nemoto dan N. Kato, "A Survey of Routing Attacks in Mobile Ad Hoc Networks," *IEEE Wireless Communications*, p. 85, 2007.

[8] R. D. Vaghela dan N. J. Goswami, "A MODIFIED HYBRID PROTOCOL(ZRP) USED FOR DETECTION AND REMOVAL OF BLACK HOLE," *JOURNAL OF*

INFORMATION, KNOWLEDGE AND RESEARCH IN COMPUTER ENGINEERING , vol. 02, no. 02, p. 326, 2013.

[9] O. Kembuan, Widyawan dan S. S. Kusumawardani, "Analisis Kinerja Reactive Routing Protocol dalam Mobile Ad-Hoc Network (MANET) Menggunakan NS-2 (Network Simulator)," *Jurnal Nasional Teknik Elektro dan Teknik Informatika,* vol. 1, p. 1, 2012.

BIODATA PENULIS

Setiyo Adiwicaksono lahir di Jakarta pada tanggal 11 Februari 1996. Penulis menempuh pendidikan mulai dari SDIT An-Nisaa (2002-2008), SMPI Al-Azhar 1 Kebayoran Baru (2008-2010), SMA Taruna Nusantara (2010-2013), dan S1 Teknik Informatika ITS (2013-2017).

Selama kuliah, penulis aktif dalam organisasi Himpunan Mahasiswa Teknik Computer-Informatika ITS (HMTC). Diantaranya adalah sebagai Staf Departemen Dalam Negri HMTC Berkarya (2014-2015) dan menjadi Kepala Departemen Dalam Negri HMTC Optimasi (2015-2016). Penulis juga aktif dalam kegiatan kepanitiaan Schematics dengan menjadi BPH Reeva Schematics 204 dan Staff Ahli Reeva Schematics 2015. Penulis juga merupakan seorang administrator di Laboratorium Arsitektur dan Jaringan Komputer dan pernah menjadi asisten mata kuliah Sistem Digital, Sistem Operasi dan Jaringan Komputer.

Kritik dan saran sangat diharapkan guna peningkatan kualitas dan penulisan selanjutnya. Untuk itu, silahkan kirim kritik dan saran ke : setiyoadiwicaksono@gmail.com