TUGAS AKHIR – KI141502

PENGEMBANGAN ZONE ROUTING PROTOCOL UNTUK PROSES DATA GATHERING PADA LINGKUNGAN WIRELESS SENSOR NETWORK DENGAN PENGHEMATAN ENERGI

GEDE WAYAN DHARMAWAN
NRP 05111440000133

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

Departemen Informatika
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2018

TUGAS AKHIR – KI141502

PENGEMBANGAN ZONE ROUTING PROTOCOL UNTUK PROSES DATA GATHERING PADA LINGKUNGAN WIRELESS SENSOR NETWORK DENGAN PENGHEMATAN ENERGI

GEDE WAYAN DHARMAWAN
NRP 05111440000133

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

Departemen Informatika
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2018

(Halaman ini sengaja dikosongkan)

UNDERGRADUATE THESES – KI141502

DEVELOPMENT OF ZONE ROUTING PROTOCOL FOR DATA GATHERING PROCESSES IN WIRELESS SENSOR NETWORK ENVIRONMENTS WITH ENERGY SAVING

GEDE WAYAN DHARMAWAN
NRP 05111410000133

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

Department of Informatics
Faculty of Information, Communication and Technology
Sepuluh Nopember Institute of Technology
Surabaya 2018

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

PENGEMBANGAN ZONE ROUTING PROTOCOL UNTUK PROSES DATA GATHERING PADA LINGKUNGAN WIRELESS SENSOR NETWORK DENGAN PENGHEMATAN ENERGI

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Komputasi Berbasis Jaringan
Program Studi S-1 Departemen Informatika
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember

Oleh:
GEDE WAYAN DHARMAWAN
NRP: 05111440000133

Disetujui oleh Pembimbing Tugas Akhir:

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

SURABAYA
JUNI, 2018

(Halaman ini sengaja dikosongkan)

PENGEMBANGAN ZONE ROUTING PROTOCOL UNTUK PROSES DATA GATHERING PADA LINGKUNGAN WIRELESS SENSOR NETWORK DENGAN PENGHEMATAN ENERGI

Nama Mahasiswa : GEDE WAYAN DHARMAWAN
NRP : 05111440000133
Jurusan : Teknik Informatika FTIF-ITS
Dosen Pembimbing 1 : Waskitho Wibisono, S.Kom., M.Eng., Ph.D.

Abstrak

Jaringan sensor nirkabel merupakan jaringan yang terdiri dari node-node sensor yang memiliki fungsi penting tetapi memiliki kekurangan pada bagian daya yang sangat terbatas. Jaringan sensor nirkabel umumnya digunakan untuk mengawasi suatu kejadian dan akan memberikan informasi jika terjadi suatu perubahan pada lingkungan yang diawasinya.

Semakin besar dan banyak node yang digunakan dalam jaringan sensor nirkabel, maka akan semakin besar pula energi yang digunakan dalam transmisi. Perlu dilakukan mekanisme tertentu agar energi yang digunakan dapat dihemat tetapi dengan tanpa mengurangi waktu tiba data hasil pengamatan.

Pada tugas akhir ini, jaringan sensor nirkabel dibagi kedalam 16 zona berdasarkan posisi dari suatu node. Pada masing – masing zonanya akan dipilih satu node yang akan menjadi cluster head. Data dari setiap node akan dikirim dan dikumpulkan pada cluster head untuk mengurangi jumlah transmisi. Selain dengan menggunakan cluster head, pada setiap node juga terdapat mekanisme aggregasi yang membuat data hasil pengamatan tidak akan langsung dikirimkan melainkan ditampung terlebih dahulu sampai pada batas tertentu. Selain dengan cluster head, penampungan ini akan mengurangi jumlah transmisi. Dengan

mengurangi jumlah transmisi data, penggunaan energi dapat ditekan sehingga akan memperpanjang network lifetime.

.

Kata kunci: jaringan sensor nirkabel, simulasi, sidnet-swans, routing protocol, node, network lifetime.

DEVELOPMENT OF ZONE ROUTING PROTOCOL FOR DATA GATHERING PROCESS ON ENVIRONMENTAL WIRELESS SENSOR NETWORK WITH ENERGY SAVING

Student's Name	**: GEDE WAYAN DHARMAWAN**
Student's ID	**: 05111440000133**
Department	**: Teknik Informatika FTIF-ITS**
First Advisor	**: Waskitho Wibisono, S.Kom., M.Eng., Ph.D.**

Abstract

Wireless sensor network is a network that consists of sensor nodes that have important functions but have deficiencies in the power. Wireless sensor network are generally used to monitor an event and will provide information in case of a change in the environment it is watching.

The more number of nodes used in the wireless sensor network, the greater energy used in the transmission. A certain mechanism is necessary to use to save energy without reducing the latency of the observed data.

In this undergraduate thesis, wireless sensor network is divided into 16 zones based on the position of a node. At each zone will be selected one node which will become cluster head. Data from each node will be sent and collected on the cluster head to reduce the number of transmitions. In addition to using the cluster head, on each node there is also an aggregation mechanism that makes the observation data will not be sent directly but accommodated in advanve to a certain extent to reducing the number of transmission By reducing the number of data transmition, the use of energy can be reduce so that it will extend the lifetime network.

Keywords : wireless sensor network, simulation, sidnet-swans, routing protocol, node, network lifetime.

KATA PENGANTAR

"Om Awighnam Astu Namo Sidham
Om Sidhirastu Tad Astu Svaha"

Segala puji dan syukur kehadapan Ida Sang Hyang Widhi Wasa, Tuhan yang Maha Esa, karena berkat ilmu pengetahuan, petunjuk dan restu dari-Nya, penulis dapat menyelesaikan Tugas Akhir yang berjudul: "**Pengembangan Zone Routing Protocol untuk Proses Data Gathering pada Lingkungan Wireless Sensor Network dengan Penghematan Energi"** dengan baik.

Dalam pelaksanaan dan pembuatan Tugas Akhir ini tentunya sangat banyak bantuan-bantuan yang penulis terima dari berbagai pihak, tanpa mengurangi rasa hormat penulis ingin mengucapkan terima kasih sebesar-besarnya kepada:

1. Ida Sang Hyang Widhi Wasa karena tuntunan dan restu-Nya sehingga penulis dapat menyelesaikan Tugas Akhir ini dengan baik.
2. Kedua orang tua penulis, Bapak Made Wiarta Susila dan Ibu Ni Ketut Sri Tanjung dan adik saya I Gede Made Teddy Dharmawan dan Ni Luh Komang Prita Dewi Dharmawan serta keluarga di Tabanan-Bali yang telah memberikan motivasi kepada penulis untuk menyelesaikan Tugas Akhir ini
3. Bapak Waskitho Wibisono, S.Kom., M.Eng., Ph.D. selaku dosen wali dan dosen pembimbing yang telah memberikan kepercayaan, dukungan, bimbingan, nasehat, perhatian dan segala hal yang telah diberikan kepada penulis.
4. Bapak dan Ibu Dosen Teknik Informatika yang telah banyak memberikan wawasan dan ilmu pengetahuan kepada penulis selama menjalani masa perkuliahan.
5. Teman-teman senasib dan seperjuangan di S1 Informatika ITS 2014 yang saling mendukung baik dalam perkuliahan maupun dalam pembuatan Tugas Akhir ini.

6. Teman-teman di MES-E103 Nobby, Widhi, Tanto, Fahmi dan Hanendyo, yang selalu memberikan semangat dan motivasi untuk menyelesaikan Tugas Akhir dengan tepat waktu.
7. Teman-teman senasib dan seperjuangan di TPKH-ITS angkatan 2014.
8. Semua pihak yang tidak bisa penulis sebutkan satu persatu dalam kata pengantar ini yang telah 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 2018

DAFTAR ISI

(Halaman ini sengaja dikosongkan)

DAFTAR GAMBAR

(***Halaman ini sengaja dikosongkan***)

DAFTAR TABEL

(***Halaman ini sengaja dikosongkan***)

DAFTAR KODE SUMBER

(Halaman ini sengaja dikosongkan)

BAB I
PENDAHULUAN

1.1 Latar Belakang

Komunikasi nirkabel merupakan bentuk komunikasi yang tidak menggunakan media kabel. Komunikasi nirkabel dilakukan oleh dua *device* yang disebut dengadn *transmitter* dan *receiver* tanpa menggunakan perantara kabel [1]. *Transmitter* berfungsi sebagai pengirim data, sedangkan *receiver* berfungsi sebagai penerima data. Pada komunikasi nirkabel terdapat dua jenis *device* yaitu *field device* dan *infrastructure device*. *Field device* berfungsi sebagai pengukur nilai dari suatu lingkungan, sedangkan *infrastructure device* berfungsi sebagai tempat berkumpulnya semua informasi yang dikirim oleh *field device*.

Wireless Sensor Network (WSN) merupakan sekumpulan *device* yang berukuran kecil, memiliki berat yang ringan, dan termasuk dalam *low-cost network*. WSN atau jaringan sensor nirkabel beroperasi dengan menggunakan sumber daya terbatas dan memiliki kemampuan *sensing*, *computation*, dan komunikasi nirkabel [2]. *Node* dari sebuah jaringan sensor nirkabel memiliki tujuan yang sama, seperti mendeteksi lingkungan atau suatu kejadian pada sebuah lingkungan. Beberapa contoh penerapannya, antara lain: skenario industri, *emergency response*, *traffic monitoring*, serta pada bidang kesehatan dan medis. Setiap *node* sensor dilengkapi dengan satu atau lebih sensor bertenaga rendah, prosesor, memori, baterai, dan radio [3].

Baterai dari *node* sensor tidak tergantikan atau tidak dapat diisi ulang, sehingga penggunaan energi menjadi batasan utama dalam WSN. Sumber energi harus digunakan dengan bijak untuk memperpanjang masa pakai *node* sensor.

Pada proses pengumpulan data (*data gathering*), jika semua *node* secara langsung mengirimkan datanya ke tujuan, maka jaringan akan penuh dengan pesan berisi data dari setiap *node* dan ini mengakibatkan energi yang digunakan akan sangat banyak.

Selain energi yang digunakan banyak, akan ada banyak pesan yang tidak sampai ke tujuan dikarenakan penuhnya jaringan dengan pesan – pesan berisi data tadi. Oleh karena itu, diperlukan suatu cara untuk mengoptimalkan penggunaan energi sehingga masa hidup dari setiap *node* akan bertambah dan pesan yang dikirimkan juga semakin banyak yang sampai ke tujuan.

Clustering telah menjadi teknik yang dapat digunakan dalam menghadapi masalah yang telah dibahas diatas. Metode *cluster* dalam jaringan sensor nirkabel mengumpulkan semua *node* dalam suatu grup yang disebut sebagai *cluster*. Dalam setiap *cluster* terdapat satu *node* yang memiliki peran tambahan yang membedakannya dengan *node* lainnya, *node* tersebut disebut sebagai *cluster head* dari *cluster* tersebut. Sedangkan *node* lain selain *cluster head* disebut sebagai anggota yang masing – masing memiliki peran yang sama. [4]

Pembagian *node* kedalam suatu *cluster* dilakukan berdasarkan lokasi dari suatu *node*. Metode ini sering disebut dengan *Grid Based Clustering*. *Zone Routing Protocol* yang akan penulis terapkan berdasarkan pada metode *Grid Based Clustering* dengan membagi area kedalam zona – zona berdasarkan pada lokasi *node* tersebut berada. Setiap *node* akan mengirimkan data hasil *sensing* ke *cluster head* dari masing – masing zona. Di dalam *cluster head*, akan dilakukan aggregasi data sebelum kemudia dilakukan pengiriman data menuju lokasi (*sink node*). Pengiriman menuju *sink node* dilakukan oleh *cluster head* dengan mengirimkannya melalui zona disebelahnya jika *sink node* tidak berada di zona *node* tersebut atau secara langsung jika *sink node* berada satu zone dengan *node* tersebut.

Peneliti akan melihat kinerja dari metode yang dikerjakan dengan menerapkan dan melakukan pengujian pada ruang lingkup simulasi agar dapat melibatkan banyak *node* dan area yang luas dengan menekan biaya pengujian. Selain itu, banyaknya penelitian menggunakan simulator juga menjadi acuan. Simulator yang digunakan adalah SIDnet-SWANS yang merupakan simulator

yang berjalan diatas simulator JiST-SWANS. Simulator ini juga pernah digunakan pada penelitian sebelumnya oleh Xu sehingga simulator ini layak digunakan. [5]

1.2 Rumusan Masalah

Tugas akhir ini mengangkat beberapa rumusan masalah sebagai berikut:

1. Bagaimana menentukan *node* yang akan menjadi *cluster head* dengan melakukan kontrol terhadap kondisi energi *node* pada lingkungan WSN?
2. Bagaimana menentukan *routing table* dengan memperhatikan kondisi baterai, jumlah tetangga dan jarak menuju *sink* pada lingkungan WSN?
3. Bagaimana membangun protokol pengiriman data dengan konsep diatas pada lingkungan WSN?

1.3 Batasan Permasalahan

Permasalahan yang dibahas pada tugas akhir ini memiliki batasan sebagai berikut:

1. Simulator yang digunakan adalah SIDnet-SWANS yang memiliki versi 1.5.6 yang berjalan diatas Java.
2. Setiap *node* pada simulator memiliki 1 sensor *Global Positioning System* (GPS) untuk mengetahui lokasi *node*.
3. Setiap *node* mampu mengubah level daya transmisinya tergantung jarak ke *receiver*.
4. Setiap *node* dapat menghitung perkiraan jarak ke *node* lain untuk daya transmisi tertentu.
5. Jumlah *sink node* pada simulator adalah 1 *node*.
6. Pada wilayah yang diawasi di simulator, semua *node* yang berada pada wilayah tersebut menjadi *source node*.
7. Sumber energi dari *node* bersifat terbatas pada simulator.

1.4 Tujuan

Tujuan dari tugas akhir ini adalah sebagai berikut:

1. Dapat menentukan *node* yang tepat untuk dijadikan sebagai *cluster head* pada lingkungan WSN.
2. Dapat menentukan jalur dalam pengiriman data dengan memperhatikan kondisi daya baterai, jumlah tetangga dan jarak dengan *sink* pada lingkungan WSN.
3. Dapat membangun protokol pengiriman data pada lingkungan WSN.

1.5 Manfaat

Dengan dibuatnya tugas akhir ini diharapkan dapat memberikan manfaat untuk membagi paket secara merata pada lingkungan WSN sehingga meningkatkan *Packet Delivery Ratio* (PDR) dan menambah *network lifetime*.

Sedangkan bagi penulis, tugas akhir ini bermanfaat sebagai sarana untuk mengimplementasikan ilmu yang telah dipelajari selama kuliah agar berguna bagi masyarakat.

1.6 Metodologi

Pembuatan tugas akhir ini dilakukan dengan menggunakan metodologi sebagai berikut:

1.6.1 Penyusunan Proposal

Proposal tugas akhir ini berisi tentang deskripsi pendahuluan dari tugas akhir yang akan dibuat. Pendahuluan ini terdiri atas hal yang menjadi latar belakang diajukannya usulan tugas akhir, rumusan masalah yang diangkat, batasan masalah untuk tugas akhir, tujuan dari pembuatan tugas akhir, dan manfaat dari hasil pembuatan tugas akhir. Selain itu, dijabarkan tinjauan pustaka yang digunakan sebagai referensi pendukung pembuatan tugas akhir. Sub bab metodologi berisi penjelasan mengenai

tahapan penyusunan tugas akhir mulai dari penyusunan proposal hingga penyusunan buku tugas akhir. Terdapat pula sub bab jadwal kegiatan yang menjelaskan jadwal pengerjaan tugas akhir.

1.6.2 Studi Literatur

Studi literatur yang dilakukan bertujuan untuk memperoleh informasi yang berkaitan dengan penelitian yang dilakukan, seperti:

1. Mekanisme pengiriman data yang sudah pernah diimplementasikan dalam lingkungan WSN.
2. Penelitian-penelitian terdahulu yang terkait dengan penerapan dan modifikasi metode routing di WSN.
3. Cara membuat simulasi simulator SIDnet-SWANS yang digunakan untuk implementasi sistem.

1.6.3 Perancangan Perangkat Lunak

Proses analisis dilakukan dengan menganalisis data-data yang didapat dari simulator SIDnet-SWANS setelah menerapkan zone routing protocol (ZRP).

Perangkat lunak yang digunakan adalah NET Beans dengan menjalankan simulator SIDnet-SWANS versi 1.5.6 yang berjalan diatas Java dengan memodifikasi alur pengiriman data dari source node menuju sink node.

1.6.4 Pengujian dan Evaluasi

Pengujian dilakukan untuk menguji apakah protokol yang diterapkan berjalan sesuai dengan aslinya. Kriteria – kriteria yang akan diujikan berupa average remain energy, time to die dari node, dan delay yang terjadi.

1.6.5 Penyusunan Buku

Pada tahap ini disusun buku sebagai dokumentasi dari pelaksanaan tugas akhir yang mencangkup seluruh konsep, teori, implementasi, serta hasil yang telah dikerjakan.

1.7 Sistematika Penulisan Laporan

Sistematika penulisan laporan tugas akhir adalah sebagai berikut:

1. Bab I. Pendahuluan
   Bab yang berisi mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, dan manfaat dari pembuatan tugas akhir. Selain itu metodologi yang digunakan dan sistematika penulisan laporan akhir juga merupakan bagian dari bab ini.
2. 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.
3. Bab III. Perancangan Perangkat Lunak
   Bab ini berisi tentang analisis permasalahan, deskripsi umum sistem, spesifikasi kebutuhan perangkat lunak, lingkungan perancangan, perancangan arsitektur sitem, diagram kelas, dan struktur data.
4. Bab IV. Implementasi
   Bab ini menjelaskan implementasi yang berbentuk *Pseudocode* yang berupa *Pseudocode* dari kelas – kelas yang dimodifikasi maupun yang baru dibuat untuk mendukung pembuatan *Zone Routing Protokol*.
5. Bab V. Hasil Uji Coba dan Evaluasi
   Bab ini berisikan hasil uji coba dari penerapan *Zone Routing Protocol* pada lingkungan jaringan sensor nirkabel dengan menerapkan beberapa modifikasi. Uji coba dilakukan dengan mengujikan beberapa faktor yang dirasa akan mempengaruhi kinerja metode yang dikerjakan.

6. Bab VI. Kesimpulan dan Saran
   Bab ini merupakan bab yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan, masalah-masalah yang dialami pada proses pengerjaan Tugas Akhir, dan saran untuk pengembangan solusi ke depannya.
7. Daftar Pustaka
   Bab ini berisi daftar pustaka yang dijadikan literatur dalam tugas akhir.
8. Lampiran
   Dalam lampiran terdapat tabel-tabel data hasil uji coba dan *Pseudocode* program secara keseluruhan.

(Halaman ini sengaja dikosongkan)

BAB II
TINJAUAN PUSTAKA

Bab ini berisi pembahasan mengenai teori-teori dasar yang digunakan dalam tugas akhir. Teori-teori tersebut diantaranya adalah komunikasi nirkabel, *Wireless Sensor Network*, *network lifetime*, *Grid Based Clustering*, SIDnet-SWANS dan beberapa teori lain yang mendukung pembuatan tugas akhir.

2.1 Komunikasi nirkabel

Komunikasi nirkabel adalah komunikasi yang menggunakan media tanpa kabel yang dilakukan antara dua perangkat yang disebut dengan *transmitter* dan *receiver*. Saat berkomunikasi informasi diubah menggunakan alat konversi dan selanjutnya ditumpangkan pada gelombang elektromagnetik untuk disampaikan ke penerima. Pada sisi penerima, sinyal – sinyal gelombang elektromagnetik dikonversi kembali menjadi informasi. [1]

Komunikasi nirkabel bisa berupa komunikasi jarak pendek seperti remote control televisi atau juga sampai jarak jauh. Komunikasi nirkabel menawarkan banyak keuntungan, diantaranya fleksibel, yaitu tidak dibatasi oleh jangkauan kabel, mobilitas dan skalabilitas yang tinggi, yaitu mudah untuk di *upgrade* baik dari sisi pengguna, jangkuan dan kapasitas. [1]

Pada komunikasi nirkabel t*ransmitter* adalah pengirim informasi atau pesan sedangkan *receiver* adalah penerima informasi atau pesan yang dikirimkan oleh *transmitter*. Pada perangkat *wireless*, *transmitter* dan *receiver* umumnya sudah terintegrasi menjadi satu perangkat. Jadi perangkat tersebut dapat bertindak sebagai *transmitter* sekaligus juga bertindak sebagai *receiver*.

Komunikasi nirkabel memiliki beberapa kelebihan diantaranya yaitu hemat dari segi perangkat keras karena komunikasi nirkabel tidak menggunakan media kabel untuk

berkomunikasi. Selain itu juga hemat dari segi skalabilitas karena tidak perlu menyiapkan *port* tetapi cukup dengan menghidupkan perangkatnya saja dan melakukan otentikasi agar dapat langsung berkomunikasi.

2.2 Jaringan Sensor Nirkabel (*Wireless Sensor Network*)

Wireless Sensor Network (WSN) merupakan jaringan nirkabel yang terdiri dari perangkat – perangkat yang didistribusikan menggunakan sensor untuk memantau kondisi fisik atau lingkungan dengan kemampuan komunikasi nirkabel. WSN menggabungkan gateway yang menyediakan konektivitas nirkabel dengan *node* – *node* terdistribusi. [3]

Pada umumnya setiap *node* pada dalam jaringan ini bekerja bersama – sama dengan tujuan yang sama seperti untuk deteksi suatu kejadian seperti kebakaran hutan atau untuk pemantauan lingkungan. Umumnya WSN digunakan untuk pengamatan daerah yang sulit seperti *gletser*, laut dalam dan lain – lain.

Energi dalam setiap *node* digunakan pada CPU, sensor dan radio yang merupakan bagian yang mengonsumsi energi terbanyak. Energi tidak selalu diperbaharui karena masalah biaya, lingkungan atau bentuk, jadi perlu dilakukan efisiensi penggunaan energi.

Gambar 2.1 Komponen WSN, *gateway*, dan *node* [3]

2.3 *Network Lifetime*

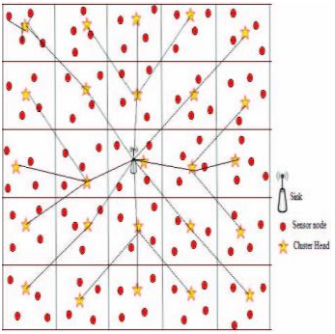
Network lifetime merupakan waktu *node* sensor yang pertama mengalami kehabisan energy untuk beroperasi [6]. Dalam beberapa pengamatan yang dilakukan WSN, diperlukan kemampuan untuk dapat berkomunikasi lebih lama, sehingga perlu dilakukan efisiensi penggunaan energi.

Network lifetime telah menjadi karakteristik kunci untuk mengevaluasi jaringan sensor dengan cara yang spesifik, terutama mengenai ketersediaan node, cakupan sensor, konektivitas, serta masa pakai jaringan. Banyak penelitian yang telah dilakukan serta telah banyak algoritma dan metode yang diusulkan untuk memperpanjang network lifetime. *Network lifetime* mengarah kepada kondisi baterai dari source node ketika mentransmisikan informasi menuju *destination node*. [4]

2.4 *Grid Based Clustering*

Grid Based Clustering merupakan metode yang dilakukan dengan membagi area pengamatan kedalam area – area kecil yang memiliki ukuran area dengan luas yang sama berdasarkan pada

range transmisi dari *node* sensor. Setiap area kecil (zona) akan membentuk satu *cluster*. Pada satu *cluster* akan dipilih satu *node* yang memiliki peran tambahan yaitu menerima data hasil pengamatan dari *node* lain yang masih berada pada areanya. *Node* yang terpilih ini akan disebut sebagai *cluster head*. *Cluster head* dipilih dengan mempertimbangkan beberapa faktor, diantaranya memiliki jarak terdekat dengan *node* lain di *clusternya* dan memiliki sisa energi lebih dari batas yang telah ditentukan. *Cluster head* akan berkomunikasi dengan *sink node* melalui *node* lainnya sebagai perantara. [6]

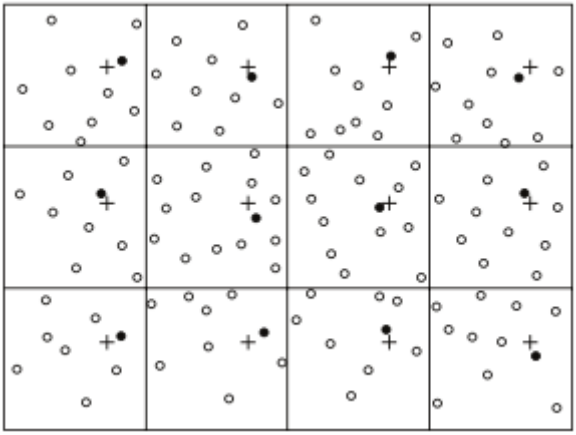


Gambar 2.2 Pengiriman data dengan *Grid Based Clustering* [6]

2.5 *Position-Based Aggregator Node Election* (PANEL)

Dalam metode PANEL, setiap *node* dibagi kedalam kelompok – kelompok berdasarkan letak geografisnya dimana setiap kelompok akan berbentuk kotak yang memiliki ukuran yang sama dengan kelompok lainnya. PANEL juga dapat digunakan untuk area dan bentuk *cluster* dengan bentuk yang lebih kompleks. Pengelompokan ini telah ditentukan sebelum penyebaran jaringan dan setiap *node* telah diberikan informasi *cluster* miliknya. Informasi *cluster* yang dimiliki dari setiap *node* meliputi koordinat sudut kiri bawah *cluster* miliknya dan ukuran dari *cluster* tersebut. [7]

Pada setiap *cluster* akan dipilih satu *node* terdekat dengan *sink node* sebagai aggregator untuk periode tertentu. Prosedur pemilihan ini membtuhkan komunikasi dalam *cluster*. PANEL memanfaatkan komunikasi ini untuk membuat *routing table* untuk *routing* didalam *cluster*. Diakhir pemilihan, aggregator akan mencari loncatan selanjutnya untuk keluar dari *cluster* tempatnya berada. [7]

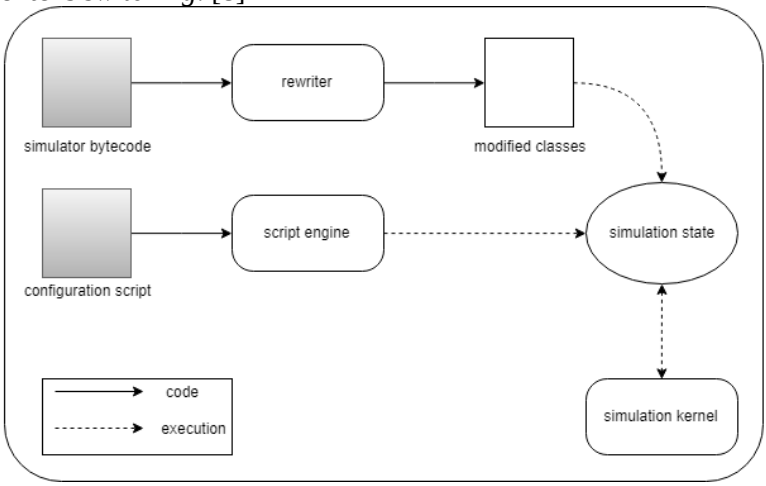


Gambar 2.3 Persebaran *node* kedalam *grid – grid* beserta *node aggregator* dari setiap *grid* [7]

2.6 *Java in Simulation Time* (JiST)

JiST (*Java in Simulation Time*) adalah salah satu *tools* simulasi yang berbasis Java yang digunakan untuk mengeksekusi simulasi event diskrit secara efisien dan transparan dengan menggunakan simulasi semantic untuk mengeksekusi Java Model. Sistem ini memberikan manfaat dalam hal penggunaan *runtime* Java. JiST adalah *tools* yang berbasis java dan JVM, dimana seluruh komponen yang tersedia murni berbasis java. Komponen yang tersedia pada JiST terdiri dari *compiler*, *bytecode compile rewriter*, *simulation kernel*, dan *virtual machine*. Definisi *kernel* sendiri adalah bagian inti dari simulasi yang akan dirubah ke dalam

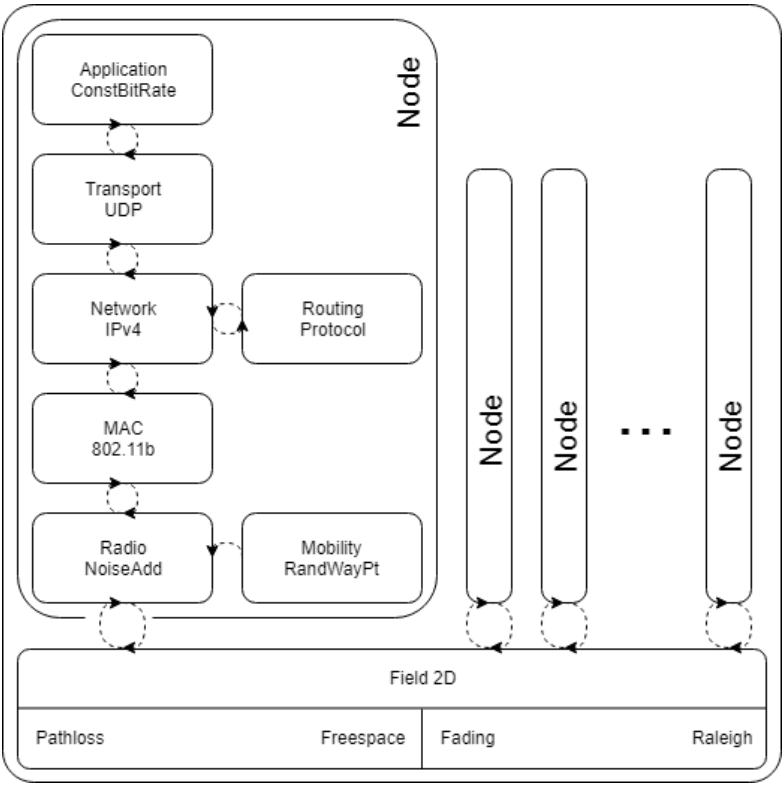
bentuk entity. Metode *invoke* objek ditandai sebagai entitas yang merepresentasikan *simulation events*. Selain itu, karena kernel dan simulasi berjalan dalam ruang proses yang sama, maka hal ini dapat mengurangi terjadinya serialisasi dan *overhead* dalam konteks *switching*. [8]



Gambar 2.4 Ilustrasi Proses Simulasi di JiST [8]

2.7 Scalable Wireless Ad Hoc Network Simulator (SWANS)

SWANS (*Scalable Wireless Ad Hoc Network Simulator*) dibangun diatas platform JiST. SWANS merupakan library untuk simulasi MANET dan bertujuan untuk mengimplementasikan aplikasi JiST. Karakteristik dari SWANS adalah handal dalam menangani jumlah *node* yang banyak (*scalable*, propagasi sinyal yang efisien, dan dapat menangani aplikasi jaringan yang standar dikembangkan dengan menggunakan Bahasa pemrograman java. [8]



Gambar 2.5 Arsitektur Simulator SWANS [8]

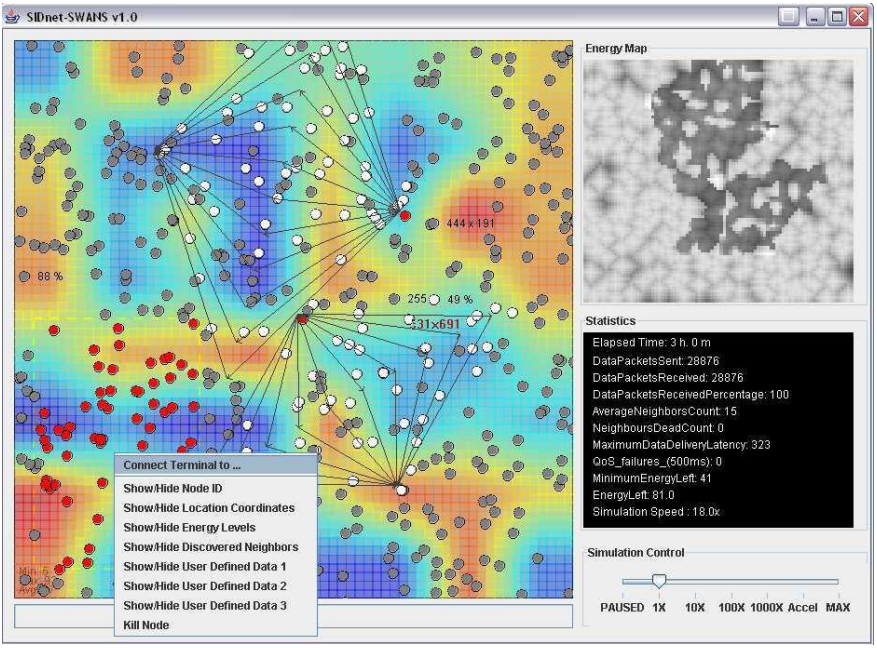
2.8 SIDnet-SWANS

SIDnet-SWANS (*Simulator and Integrated Development Platform for Sensor Networks Applications*) adalah alat visualisasi berbasis java untuk menyimulasikan berbagai aplikasi *Wireless Sensor Network* (WSN). SIDnet adalah simulator yang berbasis Java yang dirancang untuk menunjukan interaksi pada jaringan. Pengguna dapat membuat dan mengamati algoritma baru yang dirancang dan mencoba fluktuasi fenomena seperti apa yang dapat terjadi di kondisi nyata. Selain itu, pada saat *run-time* (melalui

built-in terminal) simulasi bisa dipercepat atau dihentikan sementara, sesuai kebutuhan. SIDnet-SWANS didasarkan pada arsitektur simulator JiST-SWANS dan digunakan untuk mengimplementasikan protokol MAC-layer IEEE 802.15.4. [9]

SIDnet-SWANS memiliki beberapa fitur, fitur – fitur tersebut diantaranya [10]:

1. Antarmuka GUI yang fleksibel bagi pengguna untuk berinteraksi dengan jaringan sensor nirkabel.
2. Data real-time dan analisis visual dari node sensor.
3. Kontrol eksplisit terhadap kecepatan simulasi.
4. Efisiensi terhadap pemodelan dan manajemen menuju area energi dari jaringan sensor nirkabel.
5. Variasi parameter seperti temperatur, kelembaban dan gerakan dinamis bisa dipantau karena dari desain modular.



Gambar 2.6 Simulasi jaringan sensor nirkabel pada SIDnet-SWANS [9]

2.9 Java

Java adalah bahasa pemrograman yang beorientasi objek dan tersusun dari bagian yang disebut *class* (kelas). Setiap kelas terdiri dari fungsi – fungsi yang melakukan pekerjaan masing – masing. Konsep utama dari pemrograman berorientasi objek adalah objek yang merupakan modul yang berisi data dan *subroutines*. Objek dipandang sebagai suatu entitas mandiri yang memiliki keadaan internal dan dapat merespon pesan atau pemanggilan. [11]

(Halaman ini sengaja dikosongkan)

BAB III
PERANCANGAN PERANGKAT LUNAK

Bab ini membahas mengenai perancangan dan pembuatan *Zone Routing Protocol* pada lingkungan *Wireless Sensor Network* dengan mempertimbangkan kondisi baterai, jumlah tetangga dan jarak ke *sink node* dengan menggunakan SIDnet-SWANS sebagai simulatornya.

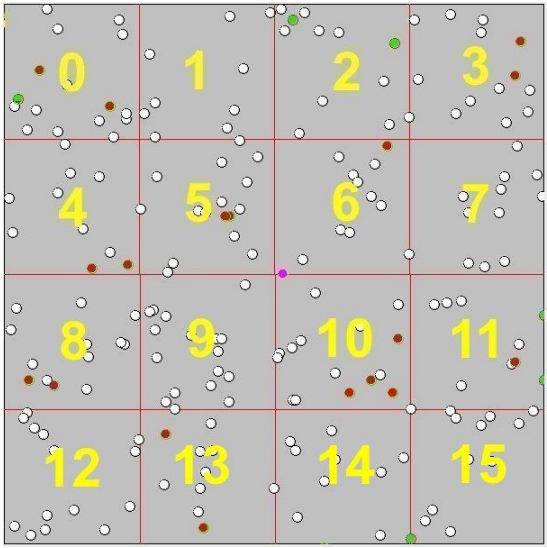
3.1 Data

Data yang digunakan merupakan data suhu yang telah ditentukan sebelumnya. Data ini dibagi menjadi dua tipe yaitu normal dan tinggi. Data suhu normal berkisar pada 18º sampai 38º sedangkan data suhu tinggi berkisar 39º sampai 45º.

Data suhu yang tinggi diberi prioritas 1 yang merupakan data yang dengan prioritas tertinggi untuk dikirimkan. Sedangkan data suhu yang normal diberi prioritas 2 yang merupakan data dengan prioritas terendah untuk dikirimkan. Prioritas ini yang kemudian dijadikan sebagai penentu pada pengiriman data.

3.2 Zone

Semua *node* akan dibagi kedalam 16 zone berdasarkan pada posisi dimana *node* tersebut berada. Setiap zona memiliki ukuran yang sama yaitu 6.25 % dari luas wilayah total dengan ukuran 25% x 25% dari ukuran wilayah total. Setiap *zone* akan memiliki satu *cluster head* yang dipilih secara bergantian dari setiap anggota *zone* tersebut. Pemilihan *cluster head* dilakukan secara bergantian agar dapat membagi beban penggunaan energi pada setiap *node*. Cluster *head* dari suatu zone berperan untuk menampung pesan dari *zone*-nya masing – masing sebelum dikirim menuju *sink node*. Pergantian *cluster head* akan dilakukan ketika *cluster head* sebelumnya bersiap mengirimkan data yang ditampung padanya.

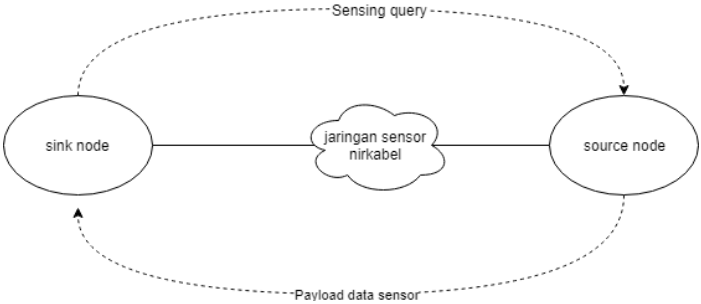


Gambar 3.1 Pembagian zona berdasarkan lokasi *node*

3.3 Desain Umum Sistem

Pada sistem ini, terdapat dua jenis *node*, yaitu *sink node* dan *source node*. *Sink node* memiliki spesifikasi yang berbeda dengan *source node*, yaitu memiliki jumlah baterai dan bandwidth yang lebih besar dari *source node.*

Secara umum, metode ini memiliki dua tahap yaitu menyebarkan *sensing query* ke seluruh *node* dan melakukan pengiriman data sensor dari masing – masing *node* yang masuk ke area pengawasan. Gambar 3.2 merupakan skema secara umum dari metode yang dikerjakan. Desain ini diharapkan mampu membawa paket data sensor yang dikirim dari *source node* menuju *sink node* dengan meningkatkan jumlah data yang berhasil dikirimkan dan mengurangi jumlah transmisi data, sehingga akan mampu mengurangi konsumsi energi yang digunakan untuk mengirimkan data.



Gambar 3.2 Skema umum

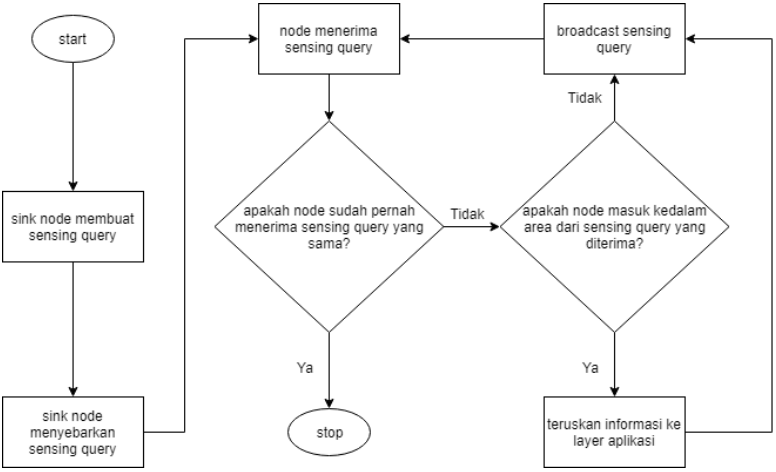
Pada penelitian ini nantinya terdapat beberapa istilah yang digunakan diantaranya :

1. *Payload* : data sensor yang dihasilkan pada *source node* yang merupakan hasil dari pemantauan lingkungan dari *node* tersebut.
2. *Localpool* : variabel yang menampung *payload* sebelum nantinya dikirim menuju *cluster head*.
3. *Localpool size* : ukuran dari *localpool*.
4. *Pool* : variabel yang menampung *payload* dari *source node* di *cluster head*.
5. *Cluster Head* : *node* yang bertugas menampung *payload* yang diterima dari *node* di suatu *zone*.
6. *Region* : wilayah yang diawasi pada jaringan sensor nirkabel dan terdapat pada *sensing query*.
7. *Sensing Query* : informasi mengenai wilayah pengawasan, durasi pengawasan, dan interval pengambilan nilai pemantauan yang dilakukan pada jaringan sensor nirkabel yang dibuat oleh *sink node* dan dikirim ke seluruh *node* dengan cara *broadcasting*.

3.3.1 Pengiriman *Sensing Query*

Pada fase ini, *sink node* akan membuat *sensing query* yang kemudian disebarkan pada jaringan sensor nirkabel dengan

melakukan *broadcasting* ke semua *node* yang terdeteksi oleh *sink node*. Gambar 3.3 merupakan diagram yang menjelaskan proses dari *sensing query* menuju *source node*.



Gambar 3.3 Skema *Sensing Query*

Node yang terdeteksi oleh *sink node* akan menerima *sensing query* dan selanjutnya informasi yang terdapat pada *sensing query* akan diperiksa oleh *node* tersebut. *Node* yang menerima *sensing query* akan mencocokkan posisinya dan jika *node* berada pada *region* yang diawasi, maka *node* tersebut akan melakukan proses sensing dengan memberi informasi tersebut ke layer aplikasi. Untuk semua *node*, baik yang berada pada *region* yang diawasi atau tidak berada pada *region* yang diawasi, akan menyebarkan informasi *sensing query* ini ke *node* lain disekitarnya. Pada sensing query terdapat *id* yang kemudian akan digunakan sebagai pembeda agar tidak terjadi duplikasi saat data diterima. Jadi ketika suatu *node* telah menerima *sensing query* menerima *sensing query* lagi dengan *id* yang sama, maka akan diabaikan agar tidak terjadi *broadcasting* yang berulang terus menerus.

3.3.2 Pengiriman *Payload* Data Sensor

Fase ini dimulai setelah layer aplikasi dari suatu *node* menerima informasi *sensing query*. *Node* tersebut akan memulai pengamatan selama waktu yang diberikan pada *sensing query*. Fase ini terdapat 2 tahap yaitu pemrosesan data pada *source node*, dan pemrosesan data pada *cluster head* yang kemudian diteruskan menuju *sink node*.

3.3.2.1 Pemrosesan Data pada *Source Node*

Pada tahap ini, *source node* akan menghasilkan data sensor. Data sensor ini tidak langsung dikirimkan menuju *cluster head*, melainkan ditampung terlebih dahulu di *localpool* milik *source node* tersebut berdasarkan prioritasnya. Setelah penuh, barulah data tersebut dikirimkan ke *cluster head* dari masing – masing *zone*. Data sensor yang dihasilkan dari setiap *node* adalah berjumlah satu data yang mewakili *range* waktu dari *node* tersebut.

Tahap ini diawali pada layer aplikasi dengan melakukan pengecekan terhadap kondisi baterai saat ini dari suatu *node*. Jika kondisi baterai berada dibawah 5%, maka hentikan proses. Selanjutnya dilakukan pengecekan terhadap durasi sensing, jika sudah melewati masa durasi sensing, maka hentikan proses. Jika sudah melewati pengecekan tersebut, maka *node* akan melakukan pengambilan nilai dari sensor yang kemudian akan dimasukkan kedalam *localpool* berdasarkan prioritasnya. Ketika *localpool* sudah terisi penuh, maka selanjutnya akan dilakukan pengiriman data tersebut ke layer *network*.

Pada layer *network*, data yang diterima dari layer aplikasi selanjutnya akan dikirim menuju *cluster head*. Jika *node* itu sendiri adalah *cluster head*, maka pesan tadi akan dimasukkan ke dalam *pool* dari *node* tersebut, yang artinya *node* tersebut menganggap dirinya sebagai *cluster head*. Jika *cluster head* merupakan *node* lain, maka data tersebut akan dikirim ke *node* tersebut melalui layer MAC dan kemudian ditampung kembali kedalam *pool* ketika data tersebut tiba di *cluster head*.

Pada layer MAC, data yang dikirim tidak selalu berhasil terkirim. Ketika terjadi kegagalan pengiriman, maka layer MAC akan memberikan informasi kegagalan kepada layer aplikasi yang kemudian akan memperbesar *localpool size* untuk memperlambat pengiriman data. Ketika data berhasil terkirim, maka *localpool size* pada layer aplikasi akan diperkecil sehingga pengiriman data akan lebih cepat. Ketika gagal, selain memperbesar *localpool size*, juga dilakukan percobaan pengiriman sebanyak 3 kali, lalu jika tetap gagal, maka *node* yang dituju akan dihapus dari daftar *node* tetangga.

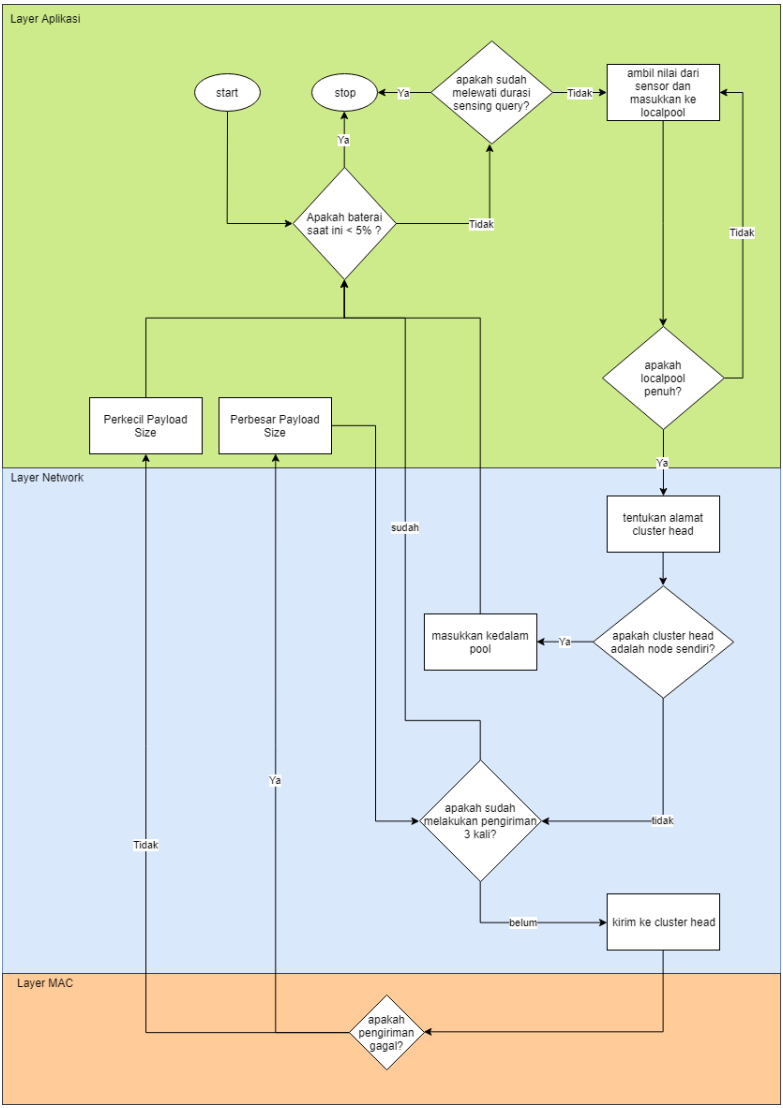
```
1   VAR {global variable}
2      buffer
3      file
4      pool
5      interval_sleep

6   FUNCTION sensing (param)
7      VAR {local variable}
8         samplingInterval <- param[0]
9         endTime <- param[1]
10        queryId <- param[2]
11        sequenceNumber <- param[3]
12        sinkAddress <- param[4]
13        sinkLocation <- param[5]
14        zoneId <- param[6]

15     BEGIN
16        sleepBlock(samplingInterval)
17        if myNode.battery >= 1 then
18           if getTime() < endTime then
19              sensedValue <- file.getData()
20              buffer.put(sensedValue,queryId)
21              if pool[1] is full then
22                 send_message(1)
23                 if pool[2] is full then
24                    sleepBlock(interval_sleep)
25                 end if
26              end if
```

```
27              if pool[2] is full then
28                 send_message(2)
29              end if

30              param[0] <- samplingInterval
31              param[1] <- endTime
32              param[2] <- queryId
33              param[3] <- sequenceNumber+=1
34              param[4] <- sinkAddress
35              param[5] <- sinkLocation
36              param[6] <- zoneId

37              sensing(param)
38           else
39              stop
40           end if
41        end if
42     END
```

Pseudocode 3-1 Fase *Sensing* di *Source Node*

```
1   VAR {global variable}
2      pool
3      myNode
       CH_pool
4
5   FUNCTION handleDataValue(pool_id)
6      VAR {local variable}
7         myCH <- getMyClusterHead()
          msg <- pool[pool_id].getMessage()
8
9      BEGIN
10        if myCH.ip == myNode.ip then
11           CH_pool[CH_pool.size()+1] <- msg
12        else
13           nextHop <- findNextHop(myCH)
14           to_link_layer(msg,nextHop)
15        end if
       END
```

Pseudocode 3-2 Fase Pengiriman Pesan dari *Source Node* menuju *Sink Node*



Gambar 3.4 Pemrosesan Data pada *Source Node*

3.3.2.2 Pengumpulan Data pada *Cluster Head*

Pada *cluster head* akan dilakukan pengumpulan data. Data yang terkumpul tersebut akan ditampung kedalam pada *pool* yang juga memperhatikan prioritas data tersebut. Setelah ditampung kedalam *pool*, lalu akan dilakukan pengiriman data keluar dari *zone*. Pengiriman ini dilakukan pada rentang interval tertentu.

Jika kondisi pengiriman sudah terpenuhi, maka akan dilakukan pengiriman ke *node* selanjutnya untuk kemudian diteruskan ke *sink node*. Jika kondisi pengiriman belum terpenuhi, maka data akan tetap ditampung kedalam *pool*. Proses pengiriman dimulai dengan mencari node tujuan. Jika node tujuan atau *sink node* adalah tetangga langsung dari *node* ini, maka *node* ini akan mengirimkan langsung ke *sink node*, akan tetapi jika *sink node* bukan tetangga langsung dari *node* ini, maka akan dicarikan *node* lain yang akan berperan sebagai perantara. Pemilihan *node* perantara dilakukan dengan memilih *node* tetangga dari *cluster head* yang memiliki jarak ke *sink node* kurang dari jarak *cluster head* ke *sink node* (Pseudocode 3-4). Dari setiap *node* yang terpilih akan dipilih *node* secara berurutan untuk rute pengiriman data dari *cluster head* menuju *sink node* (Pseudocode 3-5). Node yang terpilih akan mengulang proses pada tahap ini, mulai dari menampung kedalam *pool* sampai pada proses pengirimannya.

Pada layer MAC, data yang dikirim tidak selalu berhasil terkirim. Ketika terjadi kegagalan pengiriman, maka layer MAC akan memberikan informasi kegagalan kepada layer aplikasi yang kemudian akan melalukan percobaan pengiriman sebanyak 3 kali, lalu jika tetap gagal, maka pesan akan dibuang (Pseudocode 3-6) dan *node* yang dituju akan dihapus dari daftar *node* tetangga dengan tujuan untuk mencegah terjadinya kegagalan pengiriman selanjutnya.

```
1   VAR {global variable}
2      CH_pool

3   FUNCTION timingSend(interval)
4      BEGIN
5         sleepBlock(interval)

6         if CH_pool is not empty then
7            changeCH()
8            for i=1 to CH_pool.size()
9               msg <- CH_pool[i]
10              send_message(msg)
11           end for
12        end if
13        CH_pool.clear()
14        timingSend(interval)
15     END
```

Pseudocode 3-3 Fase Pengiriman Pesan dari *Pool* di *Cluster Head* menuju *Sink*

```
1   VAR {global variable}
2      sink
3      neighbourList
4      nextHop<-array()

5   FUNCTION daftarOut()
6      VAR {local variable}
7         threshold<-jarak node ini ke sink

8      BEGIN
9         for i=1 to neighbourList.size
10           distance<-jarak neighbourList[i] ke sink
11              if distance < threshold then
12                 nextHop.add(neighbourList[i])
13              end if
14        end for
15     END
```

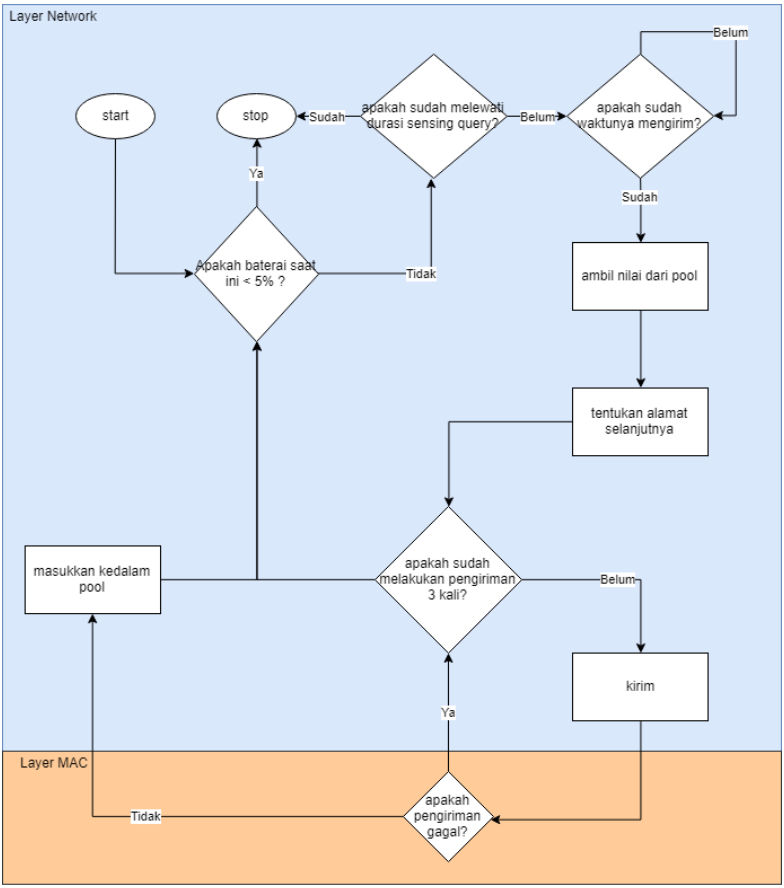
Pseudocode 3-4 Menentukan Daftar Jalur Keluar dari *Cluster Head*

```
1   VAR {global variable}
2      index<-0
3      nextHop<-daftarOut()

4   FUNCTION findNextHop_out()
5      BEGIN
6         index <- index +1
7         if index == NextHop.size then
8            index <- 0
9         end if
10        return nextHop[index]
11     END
```

Pseudocode 3-5 Pemilihan jalur keluar dari *Cluster Head*

```
1   FUNCTION dropNotify(reason,msg,dest)
2      BEGIN
3         ...
4         if reason == UNDELIVERABLE then
5            for i=1 to 3
6               to_link_layer(msg,dest)
7               if reason == PACKET_DELIVERED then
8                  return
9               end if
10           end for
11           if reason == UNDELIVERABLE then
12              drop_message(msg)
13           end if
14        end if
15     END
```

Pseudocode 3-6 Fase *Retry* Jika Pengiriman Pesan Gagal



Gambar 3.5 Pengumpulan Data pada *Cluster Head*

3.4 Tipe Pesan yang dikirim

Pada sub bab ini akan dijelaskan mengenai tipe – tipe pesan yang digunakan sebagai media penyampaian infomasi dari *source node* untuk selanjutnya diolah dan dikirim menuju *sink node*. Setiap tipe pesan memiliki fungsinya masing – masing demi

memperoleh penghematan daya baterai dan peningkatan ratio pengiriman data pada lingkungan *Wireless Sensor Network*.

3.4.1 Tipe Pesan Heartbeat

Tipe pesan *heartbeat* merupakan tipe pesan yang digunakan untuk mencari tetangga dari suatu *node*. Tipe pesan ini digunakan oleh protokol *heartbeat*. Pesan ini disebarkan ketika aplikasi dijalankan pertama kali sebelum memulai memilih *sink node* dan melakukan pengamatan.

3.4.2 Tipe Pesan *Query*

Tipe pesan *query* merupakan tipe pesan yang digunakan untuk mencari *node – node* yang akan menjadi source *node* pada sistem ini. Tipe pesan *query* ditransmisikan secara *broadcast* dari *sink node* ke semua *node* pada sistem, lalu kemudian *node* yang dipilih berdasarkan pesan *query* yang dikirimkan akan menandai dirinya sebagai *source node*.

3.4.3 Tipe Pesan *Data Value*

Tipe pesan *data value* merupakan tipe pesan yang dikirimkan *node* yang merupakan informasi utama dari *source node* yang kemudian akan dikirimkan ke *sink node*. Tipe pesan ini berisikan data pengamatan yang berupa nilai rata – rata, nilai maksimal, nilai minimal, dan jumlah pengamatan, dan beberapa informasi lain seperti id *zone*, prioritas, id kueri, dan informasi mengenai lokasi *cluster head* dan alamat ip *cluster head*..

3.4.4 Tipe Pesan *Pool Data Value*

Tipe pesan *pool data value* merupakan tipe pesan yang digunakan untuk meneruskan pesan dari *cluster head* menuju *node* lain yang kemudian akan diteruskan menuju *sink node*. Tipe pesan ini berisikan data pengamatan yang berupa nilai rata – rata, nilai

maksimal, nilai minimal, jumlah pengamatan, dan beberapa informasi lain seperti id *zone*, prioritas, id kueri, dan informasi mengenai lokasi *sink* dan alamat ip dari *sink*.

3.4.5 Tipe Pesan *Drop Notify*

Tipe pesan *drop notify* merupakan tipe pesan yang dikirimkan *link layer* dari *node* ke *app layer* yang merupakan informasi untuk memperbesar atau memperkecil *local pool size* yang nantinya akan mempercepat pengiriman atau memperlambat pengiriman data..

3.5 Faktor Penentu *Cluster Head*

Pada metode ini, terdapat istilah *cluster head* yang bertugas menampung sementara data sensor dari *source node*. *Cluster head* merupakan beberapa *node* yang memiliki jarak kurang dari 30 dari pusat dari masing – masing *zone*. Pada suatu waktu masa pengiriman, hanya terdapat 1 *node* yang berperan sebagai *cluster head* yang dipilih secara bergantian dari daftar node dengan jarak kurang dari 30 dari pusat *zone*. Pergantian dari *cluster head* dilakukan ketika suatu *cluster head* melakukan pengiriman data *CH_pool* nya ke node lain.

Pengiriman data dari *source node* menuju *sink node* melalui *cluster head* dilakukan dengan memilih *node* tetangga dari *cluster head* yang memiliki jarak ke *sink node* kurang dari jarak *cluster head* ke *sink node*. Dari setiap *node* yang terpilih akan dipilih *node* secara berurutan untuk rute pengiriman data dari *cluster head* menuju *sink node*.

```
1   VAR {global variable}
2      threshold <- 30
3      cluster_data

4   FUNCTION daftarCH(i)
5      VAR {local variable}
6         node_list <- daftar node di zone ke i
7         CH_list <- array()

8      BEGIN
9         for i=1 to node_list.size
10           distance<-jarak   node_list   [i]   ke
    cluster_data.middle
11              if distance < threshold then
12                 CH_list.add(node_list[i])
13              end if
14        end for
15        return CH_list
16     END
```

Pseudocode 3-7 Daftar *Cluster Head* dari Suatu *Zone*

3.6 Metode Pendukung

Pada penerapan metode Zone Routing Protocol ini, terdapat beberapa metode pendukung yang digunakan dalam proses pembangunannya. Terdapat beberapa metode pendukung yang digunakan, yaitu *Heartbeat protocol* dan metode yang berasal dari konsep *adaptive payload.*

3.6.1 Heartbeat Protocol

Heartbeat protocol digunakan untuk mendapatkan ketetanggaan dari suatu node. Protokol ini akan berjalan paling pertama sehingga, *node – node* dapat mengenali tetangganya masing – masing yang selanjutnya akan digunakan untuk penentuan *node* selanjutnya yang akan dituju dalam proses pengiriman data.

3.6.2 Adaptive Payload

Konsep dari *adaptive payload* ini digunakan dalam proses *pool* di *source node*. Ukuran dari *localpool* yang akan bersifat dinamis. Ukuran dari *localpool* ini akan berubah tergantung dari proses pengiriman data. Jika proses pengiriman selalu berhasil, maka ukuran *localpool* akan kecil, tetapi jika pengiriman selalu gagal, maka ukuran *localpool* akan besar.

Selain pada ukuran *localpool*, konsep ini juga digunakan pada waktu tunggu dari pengiriman data di *cluster head*. Jika proses pengiriman data di *cluster head* mengalami kegagalan, maka waktu tunggu untuk pengiriman selanjutnya akan diperbesar, tetapi jika pengiriman berhasil, maka waktu tunggu dari pengiriman selanjutnya akan dipercepat.

Penerapan dari konsep ini akan menekan jumlah transmisi, memberikan variasi ukuran *localpool* dan variasi waktu tunggu pada pengiriman data dari *cluster head*.

BAB IV
IMPLEMENTASI

Bab ini berisi penjelasan mengenai implementasi dari perancangan yang sudah dilakukan pada bab sebelumnya. Implementasi berupa *Pseudocode* untuk membangun program.

4.1 Lingkungan Implementasi

Lingkungan implementasi merupakan suatu lingkungan dimana sistem akan dibangun. Pada bagian lingkungan implementasi ini, untuk mempermudah akan dibagi menjadi dua bagian, yaitu perangkat keras dan perangkat lunak.

4.1.1 Perangkat Keras

Pada bagian ini akan dibahas mengenai perangkat keras apa saja yang digunakan dalam perancanan sistem. Spesifikasi komputer yang digunakan dalam perancangan sistem ini, antara lain :

1. Model : Aces E1-422
2. Processor : AMD E1-2500, 1.4 GHz
3. RAM : 6 GB
4. Hardisk : 1.5 TB
5. Sistem Operasi : Elementary OS

4.1.2 Perangkat Lunak

Pada bagian ini akan dibahas mengenai perangkat lunak apa saja yang digunakan dalam perancangan sistem. Detail perangkat lunak yang digunakan diantaranya :

1. Software IDE : Netbeans 8.2
2. Versi Java : Development Kit 1.6

4.2 Implementasi

Pada sub bab implementasi ini menjelaskan mengenai pembangunan perangkat lunak secara detail dan menampilkan *Pseudocode* yang digunakan pada SIDnet-SWANS. Implementasi ini dilakukan dengan memodifikasi modul – modul yang terdapat pada simulator jaringan sensor nirkabel SIDnet-SWANS. Modifikasi yang dilakukan meliputi beberapa modul, diantaranya modul *application layer*, modul *driver*, modul *routing protocol*, modul.MAC 802.15 *layer*, pembuatan modul *pool item*, dan pembuatan tipe pesan yang dikirim. Beberapa modifikasi dilakukan dengan menggandakan modul yang sudah ada ke dalam *package sidnet.stack.users.ZRP_route* dan memodifikasi yang digandakan tersebut.

4.2.1 Modifikasi *App Layer*

Modifikasi ini dilakukan dengan menggandakan modul *application layer* yang sudah disediakan secara *default* dari *package users.java.sidnet.stack.users.sample_p2p.app*. Pada package tersebut terdapat 3 file diantaranya AppSampleP2P.java, MessageDataValue.java, dan MessageQuery.java. Untuk file AppSampleP2P.java akan diganti namanya menjadi AppLayer.java. Setelah itu dibuat beberapa file pendukung baru yaitu DropperNotify.java, dan LocalPool.java.

Pada file AppLayer.java dilakukan beberapa modifikasi fungsi diantaranya fungi Sensing() dan Receive(). Fungsi Sensing() digunakan untuk melakukan pemantauan terhadap lingkungan, sedangkan Receive() digunakan untuk menerima pesan masuk dari *network layer*.

4.2.1.1 Modifikasi pada fungsi Sensing()

Pada fungsi ini dilakukan perubahan untuk mengurangi jumlah pesan yang dihasilkan dengan menggunakan *pool*. Modifikasi ini berfokus pada pembagian data berdasarkan

prioritasnya dan juga untuk setiap prioritas tersebut memiliki ukuran *pool* yang berbeda. Modifikasi yang dilakukan ditunjukkan oleh Pseudocode 4-1.

```
1   VAR {global variable}
2      buffer
3      file
4      pool
5      interval_sleep

6   FUNCTION sensing (param)
7      VAR {local variable}
8         samplingInterval <- param[0]
9         endTime <- param[1]
10        queryId <- param[2]
11        sequenceNumber <- param[3]
12        sinkAddress <- param[4]
13        sinkLocation <- param[5]
14        zoneId <- param[6]

15     BEGIN
16        sleepBlock(samplingInterval)
17        if myNode.battery >= 1 then
18           if getTime() < endTime then
19              sensedValue <- file.getData()
20              buffer.put(sensedValue,queryId)
21              if pool[1] is full then
22                 send_message(1)
23                 if pool[2] is full then
24                    sleepBlock(interval_sleep)
25                 end if
26              end if
27              if pool[2] is full then
28                 send_message(2)
29              end if
```

```
30              param[0] <- samplingInterval
31              param[1] <- endTime
32              param[2] <- queryId
33              param[3] <- sequenceNumber+=1
34              param[4] <- sinkAddress
35              param[5] <- sinkLocation
36              param[6] <- zoneId

37              sensing(param)
38           else
39              stop
40           end if
41        end if
42     END
```

Pseudocode 4-1 Modifikasi fungsi Sensing di AppLayer

4.2.1.2 Modifikasi pada fungsi Receive()

Pada fungsi ini dilakukan perubahan untuk mengolah data yang diterima dari *network layer* berupa pesan yang berisi data hasil pengamatan dan berupa pesan *notify* yang berguna pada bagian pembesaran atau pengecilan ukuran *pool*. Pseudocode 4-2 menunjukkan modifikasi yang dilakukan.

```
1   VAR {global variable}
2      myNode

3   FUNCTION receive (msg)
4      BEGIN
5         if myNode.battery >= 5 then
6            if msg instance of DropperNotify then
7               handle_drop_notify(msg)
8            else if msg instance of MessageQuery then
9               handle_query(msg)
10           else if msg instance of PoolDataValue
11  then
12              handle_pool(msg)
13        end if
14     END
```

Pseudocode 4-2 Modifikasi fungsi Receive di AppLayer

4.2.2 Modifikasi *Routing Protocol*

Modifikasi ini dilakukan dengan menggandakan modul *routing* milik *Shortest Geo Path* dari *package sidnet.stack.std.routing.shortestgeopath*. Pada package tersebut terdapat 2 file diantaranya SGPWrapperMessage.java dan ShortestGeoPathRoutingProtocol.java. Untuk file ShortestGeoPathRoutingProtocol.java akan diganti namanya menjadi RoutingProtocol.java dan file SGPWrapperMessage.java diganti menjadi ProtocolMessageWrapper.java. Setelah itu dibuat beberapa file pendukung baru yaitu MessagePoolDataValue.java, dan LocalPool.java.

Pada file RoutingProtocol.java dilakukan modifikasi total agar sesuai dengan yang akan dibuat. Pseudocode 4-3, Pseudocode 4-4, Pseudocode 4-5, dan Pseudocode 4-6 menunjukkan beberapa modifikasi yang dilakukan.

```
1   VAR {global variable}
2      CH_pool

3   FUNCTION timingSend(interval)
4      BEGIN
5         sleepBlock(interval)

6         if CH_pool is not empty then
7            changeCH()
8            for i=1 to CH_pool.size()
9               msg <- CH_pool[i]
10              send_message(msg)
11           end for
12        end if
13        CH_pool.clear()
14        timingSend(interval)
15     END
```

Pseudocode 4-3 Modifikasi Routing Protocol Bagian timingSend

```
1   VAR {global variable}
2      myNode

3   FUNCTION send(message)
4      BEGIN
5         if myNode.battery >= 2 then
6            if message instance of MessageDataValue
    then
7               handleDataValue()
8            else if message instance of MessageQuery
    then
9               handleQuery()
10           else if message instance of PoolDataValue
    then
11              handlePoolDataValue()
12           end if
13        end if
14     END
```

Pseudocode 4-4 Modifikasi Routing Protocol Bagian send

```
1   VAR {global variable}
2      myNode

3   FUNCTION receive(message)
4      BEGIN
5         if myNode.battery >= 2 then
6            if message instance of MessageDataValue
    then
7               sendToAppLayer(message)
8            else if message instance of MessageQuery
    then
9               handleQuery()
10           else if message instance of PoolDataValue
    then
11              sendToAppLayer(message)
12           end if
13        end if
14     END
```

Pseudocode 4-5 Modifikasi Routing Protocol Bagian receive

```
1   VAR {global variable}
2      buffer

3   FUNCTION dropNotify(reason,msg,dest)
4      BEGIN
5         if reason == PACKET_SIZE_TOO_LARGE then
6            print "WARNING: Packet size too large"
7         else if reason == NET_QUEUE_FULL then
8            print "ERROR: Net Queue full"
9         else if reason == UNDELIVERABLE then
10           for i=1 to 3
11              to_link_layer(msg,dest)
12              if reason == PACKET_DELIVERED then
13                 return
14              end if
15           end for
16           if reason == UNDELIVERABLE then
17              drop_message(msg)
18           end if
19        else if reason == PACKET_DELIVERED then
20           buffer.size <- buffer.size-1
21        end if
22     END
```

Pseudocode 4-6 Modifikasi Routing Protocol Bagian dropNotify

4.2.3 Modifikasi Driver

Modifikasi ini dilakukan dengan menggandakan modul *driver* yang sudah disediakan secara *default* dari *package users.java.sidnet.stack.users.sample_p2p.driver*. Pada package tersebut terdapat 1 file yaitu Driver_SampleP2P.java yang kemudian akan diganti menjadi ZRP_Driver.java. Setelah itu dibuat beberapa file pendukung baru yaitu SequenceGenerator.java, StatEntry_AliveNodesCount.java, StatEntry_DeadNodesCount.java dan Zone.java.

Pada file ZRP_Driver.java dilakukan beberapa modifikasi diantaranya menambah pembuatan node dengan dua tipe node (*node* biasa dan sink *node*). *Node* biasa memiliki beberapa

perbedaan jika dibandingkan dengan *sink node*, yaitu pada bandwidth dan baterai.

```
1   CLASS ZRP_Driver
2      VAR {global variable}
3         bWidth
4         bWidth_s
5         sink_id

6      ...

7      FUNCTION main()
8         BEGIN
9            ...
10           ris_n <- createShare(..., bWidth,...)
11           ris_s <- createShare(..., bWidth_s,...)
12           ...
13           myNode <- Node[n]
14           for int i to n
15              if i == sink_id then
16                 myNode[i] <- createSink(...,ris_s)
17              else
18                 myNode[i] <- createNode(...,ris_n)
19              end if
20              myNode[i].zoneId<-getId(myNode.loc)
21           end for
22        END

23     FUNCTION createSink(...)
24        BEGIN
25           ...
26        END

27     FUNCTION createNode(...)
28        BEGIN
29           ...
30        END

31     FUNCTION getId(location)
32        BEGIN
33           ...
34        END
```

Pseudocode 4-7 Modifikasi Driver

4.2.4 Modifikasi MAC 802.15

Modifikasi ini dilakukan langsung ke modul asli dengan tanpa melakukan penggandaan pada file tersebut. Modifikasi ini dilakukan dengan menambahkan baris kodingan yang merupakan informasi ketika suatu pesan berhasil terkirim ke tujuannya. Modifikasi ini akan berguna untuk memperlambat atau mempercepat pengiriman pesan pada *source node*. Modifikasi dilakukan dengan mengubah fungsi taskSuccess() di *sidnet.stack.std.mac.ieee802_15_4.Mac802_15_4Impl.java* seperti pada Pseudocode 4-8.

```
1   FUNCTION taskSuccess(...)
2      BEGIN
3         ...
4         dropNotify(
                payload,
                nextHop,
                PACKET_DELIVERED);
5         ...
6      END
```

Pseudocode 4-8 Modifikasi MAC 802.15

4.2.5 Pembuatan Kelas PoolReceivedItem

Kelas ini bertugas sebagai objek penampungan dari suatu data, baik itu data hasil *sensing* yang di tampung pada *localpool* atau data dari *node* yang ditampung di *pool* milik *cluster head.* Kelas ini berisikan informasi seperti nilai maksimal, nilai minimal, rata – rata, banyaknya data, serta prioritas data. Kelas ini dibuat di *package sidnet.stack.users.ZRP_route.ignoredpackage.* Pseudocode 4-9 merupakan implementasi dari kelas ini.

```
1   CLASS PoolReceivedItem
2      VAR {global variable}
3         max
4         min
5         avg
6         count

7      FUNCTION PoolReceivedItem(data,num)
8         BEGIN
9            max <- data
10           min <- data
11           avg <- data
12           count <- num
13        END

14     FUNCTION putAvg(data,num)
15        BEGIN
16           avg <- (avg*count+data*num)/(count+num)
17        END

18     FUNCTION putMax(data)
19        BEGIN
20           if max < data then
21              max <- data
22           end if
23        END

24     FUNCTION putMin(data)
25        BEGIN
26           if min > data then
27              min <- data
28           end if
29        END
```

Pseudocode 4-9 Kelas PoolReceivedItem

4.2.6 Pembuatan Kelas LocalPool

Kelas ini bertugas sebagai penampungan dari suatu data hasil *sensing* yang di tampung pada *source node*. Kelas ini berisikan informasi berupa data hasil sensing, objek PoolReceivedItem dan ukuran dari *pool* tersebut. Pseudocode 4-10 merupakan implementasi dari kelas ini.

```
1   CLASS LocalPool
2      VAR {global variable}
3         pool[]
4         MAX_SIZE
5         PRI_1_SIZE
6         PRI_1_MIN
7         PRI_1_MAX
8         PRI_2_SIZE
9         PRI_2_MIN
10        PRI_2_MAX

11     FUNCTION LocalPool()
12        BEGIN
13           pool[1] <- array(PRI_1_SIZE)
14           pool[2] <- array(PRI_2_SIZE)
15        END

16     FUNCTION getPriority(value)
17        BEGIN
18           if value >= PRI_1_MIN and
19              value <= PRI_1_MAX then
20              return 1;
21           else if value >= PRI_2_MIN and
22                   value <= PRI_2_MAX then
23              return 2;
24           else
25              return -1;
26           end if
27        END

28     FUNCTION putValue(value)
29        BEGIN
30           pri <- getPriority(value)
31           pool[pri].add(value)
32        END

33     FUNCTION isFull(pri)
34        BEGIN
35           if pool[pri].size == MAX_SIZE then
36              return true
37           else
38              return false
39           end if
40        END
```

```
41      FUNCTION isEmpty(value)
42         BEGIN
43            if pool[pool_id].size == 0 then
44               return true
45            else
46               return false
47            end if
48         END

49      FUNCTION getData(pri)
50         BEGIN
51            return pool[pri].data
52         END

53      FUNCTION reduceSize(pri)
54         BEGIN
55            return pool[pri].size-=1
56         END

57      FUNCTION increaseSize(pri)
58         BEGIN
59            return pool[pri].size+=1
60         END
```

Pseudocode 4-10 Kelas LocalPool

4.2.7 Pembuatan Kelas Tipe Paket Pesan

Terdapat beberapa tipe paket data yang digunakan yang semuanya memiliki fungsi yang berbeda – beda tergantung pada tipenya masing – masing. Berikut tipe – tipe pesan tersebut :

1. DropperNotify

   Tipe pesan ini digunakan untuk mengirim informasi ke *application layer* dari *network layer* mengenai hasil pengiriman data. Tipe pesan ini dikirim melalui fungsi DropNotify() di RoutingProtocol. Tipe pesan ini memiliki kelas yang dibuat di *package sidnet.stack.users.ZRP_route.app.*

2. MessageDataValue

   Tipe pesan ini digunakan oleh *application layer* untuk mengirimkan data hasil *sensing* menuju *cluster*

head. Tipe pesan ini memiliki kelas yang dibuat di *package sidnet.stack.users.ZRP_route.app.*

3. MessagePoolDataValue

   Tipe pesan ini digunakan oleh *network layer* untuk mengirimkan data dari *pool* menuju *sink node*. Tipe pesan ini memiliki kelas yang dibuat di *package sidnet.stack.users.ZRP_route.route.*

4. MessageQuery

   Tipe pesan ini digunakan oleh *application layer* pada *sink node* untuk mengirimkan *sensing query* ke seluruh *node* dengan metode *broadcast*. Tipe pesan ini memiliki kelas yang dibuat di *package sidnet.stack.users.ZRP_route.app.*

5. ProtocolMessageWrapper

   Tipe pesan ini merupakan tipe utama yang digunakan karena semua tipe pesan nantinya akan di *wrap* ke bentuk tipe pesan ini. Hal ini digunakan agar protocol mengetahui tipe pesan yang ditanganinya. Tipe pesan ini memiliki kelas yang dibuat di *package sidnet.stack.users.ZRP_route.route.*

4.2.8 Pembuatan Stats Collector

Stats collector merupakan kelas yang digunakan untuk melakukan perhitungan statistic dari simulator. *Stats collector* yang dibuat digunakan untuk menghitung jumlah *node* yang mati dan jumlah *node* yang masih hidup. Kedua *stats collector* ini diberi nama StatEntry_AliveNodeCount dan StatEntry_DeadNodeCount. Selain membuat *stats collector* baru, digunakan juga *stats collector* yang telah disediakan langsung oleh SIDnet-SWANS seperti StatEntry_PacketSentContor, StatEntry_PacketReceivedContor, StatEntry_PacketReceivedPercentage, dan StatEntry_PacketDeliveryLatency.

(Halaman ini sengaja dikosongkan)

BAB V
HASIL UJI COBA DAN EVALUASI

Bab ini berisi penjelasan mengenai skenario uji coba dan evaluasi protocol yang dikerjakan pada simulator SIDnet-SWANS. Hasil uji coba didapatkan dari implementasi pada Bab 4 dengan skenario yang berbeda. Bab ini berisikan pembahasan mengenai lingkungan pengujian, data pengujian, dan uji kinerja.

5.1 Lingkungan Pengujian

Lingkungan implementasi merupakan suatu lingkungan dimana sistem akan dibangun. Pada bagian lingkungan implementasi ini, untuk mempermudah akan dibagi menjadi dua bagian, yaitu perangkat keras dan perangkat lunak.

5.1.1 Perangkat Keras

Pada bagian ini akan dibahas mengenai perangkat keras apa saja yang digunakan dalam perancanan sistem. Spesifikasi komputer yang digunakan dalam perancangan sistem ini, antara lain :

1. Model : Aces E1-422
2. Processor : AMD E1-2500, 1.4 GHz
3. RAM : 6 GB
4. Hardisk : 1.5 TB
5. Sistem Operasi : Elementary OS

5.1.2 Perangkat Lunak

Pada bagian ini akan dibahas mengenai perangkat lunak apa saja yang digunakan dalam perancangan sistem. Detail perangkat lunak yang digunakan diantaranya :

1. Software IDE : Netbeans 8.2
2. Versi Java : Development Kit 1.6

5.1.3 Simulator

Lingkungan simulator berupa ketentuan – ketentuan yang berupa ukuran – ukuran spesifikasi dari masing – masing *node*. Lingkungan simulator dijelaskan lebih lanjut pada Tabel 5-1.

Tabel 5-1 Lingkungan Simulator

Keterangan		**Detail**
***Simulator running time* (sec)**		1000000000
Radio *node*	Bandwidth (bps)	4000
	Transmit (dBm)	-12
Konsumsi energy	ProcessorCurrentDrawn_ActiveMode [mA]	8
	ProcessorCurrentDrawn_SleepMode [mA]	0.015
	RadioCurrentDrawn_TransmitMode [mA]	27
	RadioCurrentDrawn_ReceiveMode [mA]	10
	RadioCurrentDrawn_ListenMode [mA]	3
	RadioCurrentDrawn_SleepMode [mA]	0.5
	SensorCurrentDrawn_ActiveMode [mA]	10
	SensorCurrentDrawn_PassiveMode [mA]	0.001
Node	Placement	Random
	Sensor	GPS, Temperature
Baterai *Sink*	Battery Capacity (mAh)	unlimited
	Battery Voltage (volt)	3
Baterai *Node*	Battery Capacity (mAh)	75
	Battery Voltage (volt)	3
Sensing Query	Duration (hour)	20
	Sampling Interval (sec)	10
	Priority Level (count)	2

Range Priority	Priority Level 1 temperature range (Celcius)	Max	45
		Min	39
	Priority Level 2 temperature range (Celcius)	Max	38
		Min	18
Range Pool Size	Priority Level 1 (high)	Start	30
		Min	10
	Priority Level 2 (normal)	Start	60
		Min	30
Route	Interval timing send (sec)		5
	Jumlah retry maksimal		3

Parameter – parameter pada simulator ini memiliki fungsinya masing – masing, diantaranya :

1. Parameter *simulator running time* merupakan variabel yang menunjukkan seberapa lama simulator dijalankan.
2. Parameter radio *node* merupakan sekumpulan variabel yang menunjukkan mengenai bandwidth dan transmit.
3. Parameter konsumsi energi merupakan sekumpulan variabel yang menjelaskan penggunaan energi dari setiap *node*.
4. Parameter n*ode* menjelaskan bagaimana *node* diletakkan dan sensor yang dipakai pada setiap *node*.
5. Baterai *sink* merupakan kapasitas dan voltase dari *sink node*, dengan asumsi *sink node* terhubung dengan sumber energi.
6. Parameter baterai *node* merupakan kapasitas dan voltase dari *source node*.
7. Parameter *sensing query* merupakan lamanya waktu pengamatan dilakukan dan berapa lama waktu *sampling* (pengumpulan data) dilakukan oleh *source node*.
8. Parameter *range priority* merupakan pembagian data dari *source node* berdasarkan nilai keluaran dari pengamatan sensor. Data keluaran dari *source node*

akan dikelompokan ke dalam 2 prioritas (prioritas 1 dan prioritas 2).

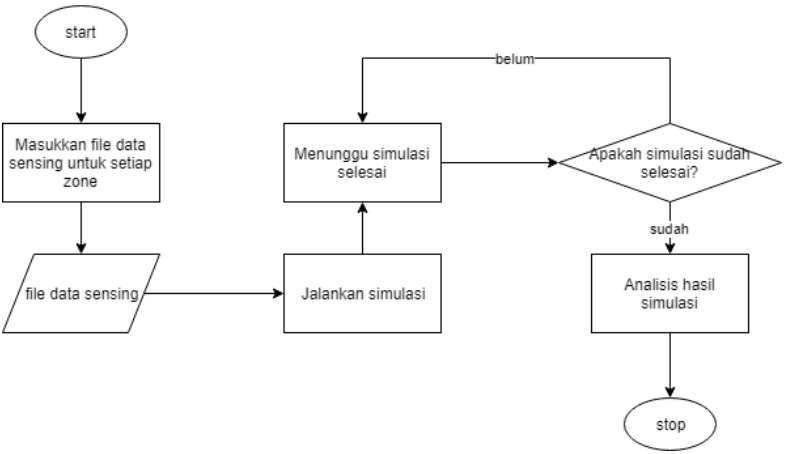
9. *Range Pool Size* merupakan batasan jumlah pesan dari masing – masing prioritas sebelum dikirimkan menuju *cluster head* dari *source node*.

5.2 Skenario Uji Coba

Sebelum melakukan uji coba, perlu ditentukan skenario yang akan digunakan dalam uji coba. Melalui skenario ini, protocol akan diuji apakah sudah berjalan dengan benar dan bagaimana performa pada masing-masing skenario. Dan membandingkan skenario manakah yang memiliki hasil lebih baik.

Skenario pengujian dibagi menjadi 4, yaitu scenario variasi metode, skenario variasi jumlah *node*, skenario variasi luas area, dan skenario metode yang digunakan. Untuk pengujian terhadap metode yang digunakan, terdapat beberapa scenario yaitu dengan menggunakan *pool*, dengan menggunakan atau dengan menggunakan keduanya.

Untuk data dari *sensing* pada *application* layer akan menggunakan file *comma separated values* (**.csv*) untuk masing – masing *zone*. Masing – masing *node* akan membaca file tersebut untuk menentukan nilai pengamatan yang dihasilkan secara random berdasarkan *range* nilai yang telah ditentukan. Untuk alur uji kerja skenario secara umum dijelaskan pada gambar dibawah.



Gambar 5.1 Skema Uji Coba

Terdapat 4 macam skenario uji coba, yaitu:

1. Skenario metode yang digunakan

   Pada skenario ini, kinerja dari protokol yang dikerjakan akan dibandingkan dengan dengan beberapa metode. Parameter jumlah *node* yang digunakan dalam skenario ini adalah 200 *node*, parameter luas area adalah 300x300, dan interval waktu *sampling* 10 detik. Metode yang dibandingkan yaitu :

   - Pengiriman secara langsung dari *source node* ke *sink node* (*direct send* / DS)
   - Dengan hanya menggunakan *pool* pada *source node* (*source pool* / SP)
   - Dengan hanya menggunakan *pool* pada *cluster head* (*cluster head pool* / CHP)
   - Dengan menggunakan *priority* dan *pool* pada *source node* dan *cluster head* (*priority pool* / PP)

2. Skenario variasi jumlah *node*

   Pada skenario ini jumlah *node* dibuat dinamis sedangkan wilayah area simulasi dibuat statis yaitu 300x300 dan interval waktu *sampling* juga statis yaitu 10 detik. Parameter jumlah *node* yang disimulasikan divariasikan yaitu 100, 200, dan 300.
3. Skenario variasi luas area

   Pada skenario ini jumlah *node* dibuat statis 200 *node* dan interval waktu *sampling* juga statis yaitu 10 detiks sedangkan wilayah area simulasi dibuat dinamis yaitu 200x200, 300x300, dan 400x400
4. Skenario variasi waktu sampling

   Pada skenario ini jumlah *node* dibuat statis 200 *node* dan wilayah area simulasi juga statis yaitu 300x300 sedangkan interval waktu *sampling* dibuat dinamis yaitu 5, 10, dan 15 detik

5.2.1 Skenario Uji Coba 1

Pada skenario uji coba 1 ini terdapat 4 metode yang diujikan. Metode tersebut diujikan sebanyak masing – masing 1 kali. Selain parameter pengujian, terdapat beberapa aspek lain yang perlu diperhatikan pada scenario uji coba 1 ini. Aspek tersebut adalah waktu node pertama mati, rasio paket yang diterima dengan paket yang dikirim menuju *sink node*, dan latensi maksimal.

Pengujian dilakukan 1 kali untuk masing – masing metode yang dikerjakan lalu data hasil uji coba disajikan pada Tabel 5-2 dan Tabel 5-3 yang merupakan rangkuman hasil dari 4 kali uji coba.

5.2.2 Skenario Uji Coba 2

Pada skenario uji coba 2 ini terdapat 3 variasi jumlah *node* yang diujikan. Masing – masing variasi tersebut dijalankan sebanyak 1 kali. Selain parameter pengujian, terdapat beberapa aspek lain yang perlu diperhatikan pada scenario uji coba 2 ini.

Aspek tersebut adalah waktu node pertama mati, rasio paket yang diterima dengan paket yang dikirim menuju *sink node*, dan latensi maksimal.

Pengujian dilakukan 1 kali untuk masing – masing variasi jumlah *node* lalu data hasil uji coba disajikan pada Tabel 5-4 dan Tabel 5-5 yang merupakan rangkuman hasil dari 3 kali uji coba.

5.2.3 Skenario Uji Coba 3

Pada skenario uji coba 3 ini terdapat 3 variasi ukuran area yang diujikan. Masing – masing variasi tersebut diujikan sebanyak masing – masing 1 kali. Selain parameter pengujian, terdapat beberapa aspek lain yang perlu diperhatikan pada scenario uji coba 1 ini. Aspek tersebut adalah waktu node pertama mati, rasio paket yang diterima dengan paket yang dikirim menuju *sink node*, dan latensi maksimal.

Pengujian dilakukan 1 kali untuk masing – masing variasi ukuran area lalu data hasil uji coba disajikan pada Tabel 5-6 dan Tabel 5-7 yang merupakan rangkuman hasil dari 3 kali uji coba.

5.2.4 Skenario Uji Coba 4

Pada skenario uji coba 4 ini terdapat 3 variasi interval *sampling* yang diujikan. Masing – masing variasi tersebut diujikan sebanyak masing – masing 1 kali. Selain parameter pengujian, terdapat beberapa aspek lain yang perlu diperhatikan pada scenario uji coba 1 ini. Aspek tersebut adalah waktu node pertama mati, rasio paket yang diterima dengan paket yang dikirim menuju *sink node*, dan latensi maksimal.

Pengujian dilakukan 1 kali untuk masing – masing interval *sampling* lalu data hasil uji coba disajikan pada Tabel 5-8 dan Tabel 5-9 yang merupakan rangkuman hasil dari 3 kali uji coba.

5.3 Hasil Skenario Uji Coba

Berikut adalah hasil uji coba dari masing – masing skenario uji coba yang telah dilakukan :

5.3.1 Hasil Skenario Uji Coba 1

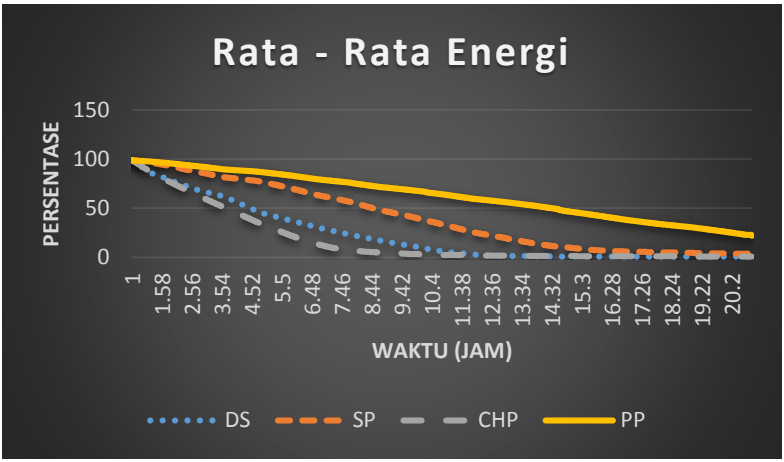
Berikut hasil skenario uji coba yang dilakukan, yaitu *direct send* / DS, *source pool* / SP, *cluster head pool* / CHP, dan *priority pool* / PP :

Tabel 5-2 Hasil Skenario Uji Coba 1

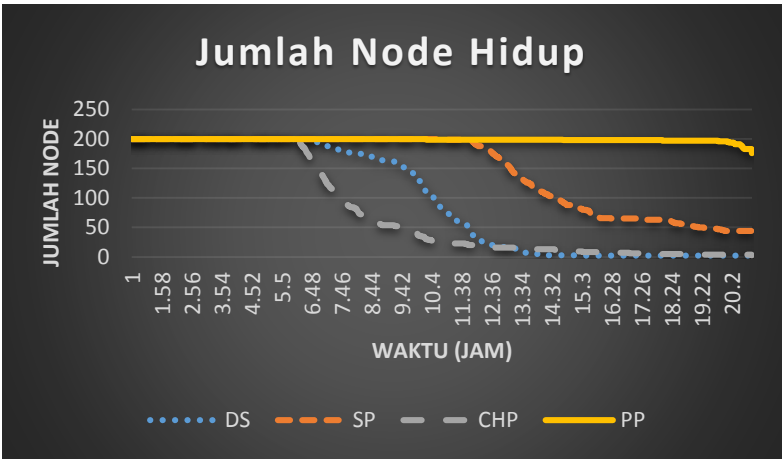
Metode	DS	SP	CHP	PP
Rata – rata sisa energi (%)	0.61	3.04	0.65	21.82
Rata - rata Latensi (ms)	62253	1277	7706	737
Waktu node pertama mati (sec)	22892	22993	26492	41717
Total node mati	198	164	197	24
Total PDR (%)	5	83	54	91
PDR High Priority (%)	5	69	51	89
PDR Normal Priority (%)	4	96	57	97

Tabel 5-3 Pesan terkirim dan pesan diterima pada uji coba 1

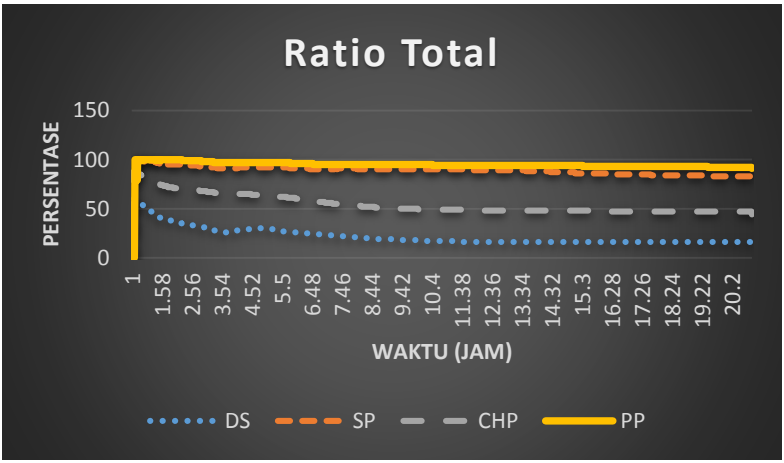
Metode		DS	SP	CHP	PP
Total	Terkirim	927932	207025	624627	155296
	Diterima	38360	171757	336917	140920
High Priority	Terkirim	480763	95366	299885	108362
	Diterima	20554	65217	152185	95412
Normal Priority	Terkirim	44765	111659	324742	46935
	Diterima	17806	106540	184732	45519



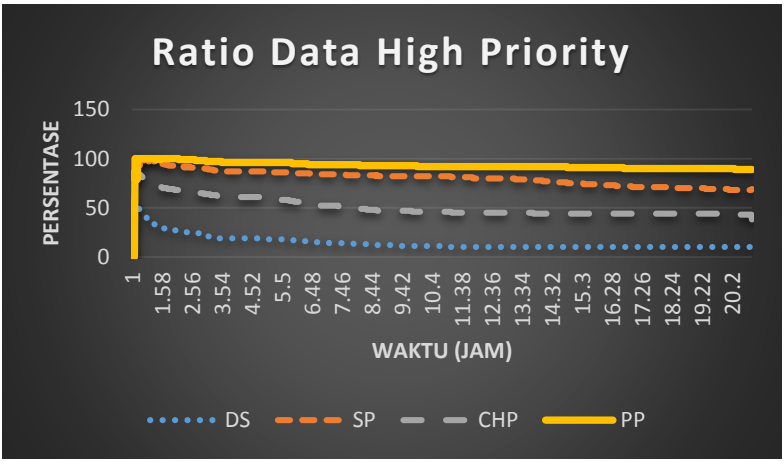
Gambar 5.2 Kondisi rata – rata sisa energi per menit



Gambar 5.3 Jumlah node yang masih hidup per menit



Gambar 5.4 *Packet Delivery Ratio* per menit



Gambar 5.5 *Packet Delivery Ratio* untuk data *high priority* per menit

Berdasarkan hasil yang ditunjukkan pada Tabel 5-2 di atas, terlihat dengan menerapkan metode ini dengan menggunakan *pool* di *source node* dan *cluster head* data menjadi hasil yang terbaik. Dengan rata – rata sisa energy 21.82 %, rata – rata latensi 737 mili detik, *node* mati pertama pada jam ke 11.6 (41.717 detik), total *node* yang mati diakhir pengujian 24 buah (12 % dari total) dan rasio paket yang diterima dan dikirim 91 %.

Berdasarkan Gambar 5.2 mengenai perbandingan sisa rata – rata energi per menit dan Gambar 5.3 mengenai jumlah sisa *node* yang masih hidup, didapatkan hasil yang paling baik untuk metode yang diajukan. Hasil pada dua gambar tersebut menghasilkan hasil yang cukup jauh mengungguli dua metode lainnya.

Berdasarkan Gambar 5.4 mengenai perbandingan *packet delivery ratio* untuk semua pesan dan Gambar 5.5 mengenai perbandingan *packet delivery ratio* untuk pesan yang berprioritas tinggi, didapatkan hasil terbaik untuk metode yang diajukan. Untuk metode lain, yaitu *source pool* mendapatkan hasil yang hamper mendekati metode yang diajukan, tetapi untuk kedua metode lainnya menghasilkan hasil yang lebih buruk.

5.3.2 Hasil Skenario Uji Coba 2

Berikut hasil skenario uji coba 2 yang dilakukan untuk jumlah *node* 100, 200, dan 300 :

Tabel 5-4 Hasil Skenario Uji Coba 2

Jumlah Node	100	200	300
Rata – rata sisa energi (%)	44.57	21.82	8.27
Rata - rata Latensi (ms)	568	737	934
Node pertama mati (sec)	-	41717	43653
Total node mati	0	24	172
Rasio paket diterima dikirim (%)	95	91	85
PDR High Priority (%)	93	89	79
PDR Normal Priority (%)	100	97	95

Tabel 5-5 Pesan terkirim dan pesan diterima pada uji coba 2

Jumlah Node		100	200	300
Total	Terkirim	86646	155296	216347
	Diterima	82475	140920	181984
High Priority	Terkirim	51196	108362	141908
	Diterima	53121	95412	111524
Normal Priority	Terkirim	29451	46935	74450
	Diterima	29359	45519	70460

Berdasarkan hasil yang ditunjukkan pada Tabel 5-4 di atas, terlihat bahwa jumlah *node* sangat mempengaruhi hasil yang diperoleh. Dikarenakan luasan area yang statis, maka semakin banyak *node* yang diamati akan membuat semakin banyak pesan data yang harus diterima oleh setiap *node*nya. Ini mengakibatkan penggunaan energi semakin besar, latensi semakin besar, dan PDR akan semakin kecil.

5.3.3 Hasil Skenario Uji Coba 3

Berikut hasil skenario uji coba yang dilakukan untuk luas area 200x200, 300x300 dan 400x400 :

Tabel 5-6 Hasil Skenario Uji Coba 3

Luas Area	200x200	300x300	400x400
Rata – rata sisa energi (%)	19.41	21.82	27.84
Rata - rata Latensi (ms)	649	737	668
Node pertama mati (sec)	47775	41717	-
Total node mati	55	24	0
Rasio paket diterima dikirim (%)	58	91	95
PDR High Priority (%)	54	89	90
PDR Normal Priority (%)	74	97	100

Tabel 5-7 Pesan terkirim dan pesan diterima pada uji coba 3

Luas Area		200x200	300x300	400x400
Total	Terkirim	121742	155296	201861
	Diterima	70137	140920	190045
High Priority	Terkirim	95006	108362	109003
	Diterima	50505	95412	97370
Normal Priority	Terkirim	26646	46935	92864
	Diterima	19632	45519	92687

Berdasarkan hasil yang ditunjukkan pada Tabel 5-6 di atas, terlihat bahwa luas area sangat mempengaruhi hasil yang diperoleh. Dikarenakan jumlah *node* yang statis, maka semakin luas area yang diamati akan membuat semakin jauh jarak masing – masing *node* sehingga suatu *node* tidak menerima pesan terlalu banyak pada suatu waktu. Ini mengakibatkan penggunaan energy semakin berkurang, dan PDR semakin baik

5.3.4 Hasil Skenario Uji Coba 4

Berikut hasil skenario uji coba yang dilakukan untuk waktu *sampling* 5, 10, dan 15 detik :

Tabel 5-8 Hasil Skenario Uji Coba 4

Interval Sampling (sec)	5	10	15
Rata – rata sisa energi (%)	3.56	21.82	42.55
Rata - rata Latensi (ms)	750	737	762
Node pertama mati (sec)	31234	41717	61656
Total node mati	157	24	2
Rasio paket diterima dikirim (%)	81	91	90
PDR High Priority (%)	74	89	86
PDR Normal Priority (%)	94	97	97

Tabel 5-9 Pesan terkirim dan pesan diterima pada uji coba 4

Interval Sampling		5	10	15
Total	Terkirim	255870	155296	96565
	Diterima	207004	140920	86159
High Priority	Terkirim	163706	108362	68165
	Diterima	120680	95412	58617
Normal Priority	Terkirim	92166	46935	28401
	Diterima	86324	45519	27544

Berdasarkan hasil yang ditunjukkan pada Tabel 5-8 di atas, terlihat bahwa lama interval sampling sangat mempengaruhi hasil yang diperoleh. Jika semakin lama waktu sampling, maka jeda waktu pengiriman antar pesan yang dikirimkan akan semakin jauh juga. Sedangkan jika interval sampling diperkecil, maka jeda waktu pengiriman antar pesan yang dikirimkan akan semakin dekat dan membuat jaringan sangat penuh dan berakibat pada semakin banyaknya energi yang digunakan, dan PDR semakin kecil, tetapi latensi akan semakin kecil juga.

BAB VI
KESIMPULAN DAN SARAN

Bab ini berisikan kesimpulan yang dapat dapat diambil dari hasil uji coba yang telah dilakukan. Selain kesimpulan, terdapat juga saran yang ditujukan untuk pengembangan perangkat lunak selanjutnya.

6.1 Kesimpulan

Kesimpulan yang didapatkan berdasarkan penerapan metode yang diajukan pada lingkungan *Wireless Sensor Network* adalah sebagai berikut:

1. *Cluster head* dipilih secara bergantian dari masing – masing *node* pada suatu zona dengan tujuan agar penggunaan energi merata pada setiap *node*.
2. Pengiriman data dilakukan dengan mekanisme *local pool* di *source node* dan *pool* di c*luster head*, pengiriman data dari *source node* akan mencari rute menuju *cluster head* dengan mencari jarak terdekat, sedangkan pengiriman dari c*luster head* menuju *sink node* dilakukan dengan memilih node tetangga *cluster head* secara bergantian.
3. Tujuan dari protokol ini adalah mengurangi penggunaan energy untuk transmisi data sehingga dapat memperpanjang *network lifetime* dari jaringan sensor nirkabel dengan tanpa menambah delay pengiriman di jaringan dan meningkatkan nilai *delivery ratio*. Hal ini ditunjukkan pada hasil uji coba 1 dimana diperoleh rata – rata sisa energi, sisa *node* hidup, PDR lebih baik dari metode lain. Rata – rata sisa energi dari metode yang diajukan lebih baik 21.21% dibandingkan dengan metode *Direct Send*, lebih baik 18.78% dibandingkan dengan metode *Local Pool*, dan lebih baik 21.17 % dibandingkan dengan metode *Cluster Head Pool*. Sisa *node* hidup dari metode yang diajukan lebih baik 87% dibandingkan dengan metode *Direct Send*, lebih baik 70%

dibandingkan dengan metode *Local Pool*, dan lebih baik 86.5% dibandingkan dengan metode *Cluster Head Pool*. PDR dari metode yang diajukan lebih baik 86% dibandingkan dengan metode *Direct Send*, lebih baik 8% dibandingkan dengan metode *Local Pool*, dan lebih baik 37% dibandingkan dengan metode *Cluster Head Pool*.

6.2 Saran

Saran yang diberikan terkait pengembangan pada Tugas Akhir ini adalah:

1. Menambah faktor penentu *zone*.
2. Menambah factor penentu *cluster head*.
3. Jika terjadi kegagalan pengiriman, carikan rute baru dari pesan yang dikirimkan.

DAFTAR PUSTAKA

[1] B. &. R. Y. J. Mehta, "Wireless communication. In: Industrial Process," Oxford: Elsevier's Science & Technology, pp. 417-457, 2015.

[2] A. &. M.-M. K. Udenze, "Renewal theory sleep time optimisation for scheduling events in Wireless Sensor Networks," Adaptive Hardware and Systems, pp. 35-42, 2007.

[3] Anonymous, "What Is a Wireless Sensor Network?," National Instruments, 24 August 2016. [Online]. Available: http://www.ni.com/white-paper/7142/en/. [Diakses 19 March 2018].

[4] F. D. Isabel D, "On the Lifetime of Wireless Sensor Networks," University of Erlangen, 2006.

[5] X. A. R. &. K. A. Xu, "Power-efficient hierarchical data aggregation using compressive sensing in WSNs," IEEE, pp. 1769-1773, 2013.

[6] P. K. J. Srikanth Jannu, "Energy Efficient Grid Based Clustering and Routing Algorithms for Wireless Sensor Networks," 2014 Fourth International Conference on Communication Systems and Network Technologies, 2014.

[7] L. B. &. P. Schaffer, "Position-Based Aggregator Node Election in Wireless Sensor Networks," International Journal of Distributed Sensor Networks, 2010.

[8] R. Barr, "SWANS - Scalable Wireless Ad hoc Network Simulator User Guide," [Online]. Available: http://jist.ece.cornell.edu/docs/040319-swans-user.pdf. [Diakses 19 March 2018].

[9] R. S. Anand Nayyar, "A Comprehensive Review of Simulation Tools for Wireless Sensor Networks (WSNs)," Journal of Wireless Networking and Communications, 2015.

[10] C. O. e. a. Ghica, "SIDnet-SWANS Manual," Evanston: Northwestern University, Inc, 2008.

[11] D. J. Eck, Introduction to Programming Using Java, Geneva: Hobart and William Smith Colleges, 2006, pp. 1-14.

[12] I. G. N. A. Kusuma, "Pengembangan Protokol Pengiriman dan Aggregasi Data untuk Jaringan Sensor Nirkabel dengan Topologi Tree (Tree-Based Network) dengan Kontrol Payload Space yang Adaptif," Institut Teknologi Sepuluh Nopember, Surabaya, 2016.

[13] X. &. C. C. G. Ning, "Optimal Dynamic Sleep Time Control in Wireless Sensor Network," IEEE Conference on Decision and Control, 2008.

[14] T. Agarwal, "Retrieved from Wireless sensor network and their applications," El Pro Cus, [Online]. Available: https://www.elprocus.com/introduction-to-wireless-sensor-networks-types-and-applications/. [Diakses 18 March 2018].

LAMPIRAN

(Halaman ini sengaja dikosongkan)

BIODATA PENULIS

Gede Wayan Dharmawan merupakan anak dari pasangan Bapak Made Wiarta Susila dan Ibu Ni Ketut Sri Tanjung. Lahir di Tabanan pada tanggal 5 Mei 1995. Penulis menempuh pendidikan formal dimulai dari TK Harapan Ibu Tabanan (1999-2001), SD Saraswati Tabanan (2001-2007), SMPN 2 Tabanan (2007-20010), SMAN 1 Kediri (2010-2013) dan S1 Teknik Informatika ITS (2014-2018). Bidang studi yang diambil oleh penulis pada saat berkuliah di Teknik Informatika ITS adalah Komputasi Berbasis Jaringan (KBJ). Penulis aktif dalam organisasi seperti Himpunan Mahasiswa Teknik Computer-Informatika (2015-2016) dan TPKH-ITS (2015-2016). Penulis juga aktif dalam berbagai kegiatan kepanitiaan yaitu SCHEMATICS 2015 dan SCHEMATICS 2016 divisi Hubungan Masyarakat (Humas). Penulis dapat dihubungi melalui email: gd.wyn.dharmawan@gmail.com.