ANALISIS KINERJA *ROUTING PROTOCOL LOW POWER AND LOSSY NETWORK* PADA *CONTIKI COOJA* DENGAN MENGGUNAKAN METODE *TIME OF ARRIVAL*

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Teknik

Disusun oleh:
Putri Ayu Delina Sari
NIM:145150301111028

PROGRAM STUDI TEKNIK KOMPUTER
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2019

PENGESAHAN

ANALISIS KINERJA *ROUTING PROTOCOL LOW POWER AND LOSSY NETWORK* PADA *CONTIKI COOJA* DENGAN MENGGUNAKAN METODE *TIME OF ARRIVAL*

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Teknik

Disusun Oleh :
Putri Ayu Delina Sari
NIM : 145150301111028

Skripsi ini telah diuji dan dinyatakan lulus pada
18 Juli 2019
Telah diperiksa dan disetujui oleh :

Dosen Pembimbing I	Dosen Pembimbing II
Mochammad Hannats Hanafi Ichsan, S.ST, M.T.	Rakhmadhany Primananda, S.T, M.Kom.
NIP. 19881229 201903 1010	NIK. 201609 860406 1 001

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T, M.T, Ph.D.
NIP. 19710518 200312 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata di dalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 18 Juli 2019

Putri Ayu Delina Sari

NIM : 145150301111028

KATA PENGANTAR

Puji syukur penulis panjatkan kehadirat Tuhan Yang Maha Esa, karena atas berkat, rahmat dan hidayah-Nya penulis dapat menyelesaikan penulisan skripsi berjudul "**Analisis Kinerja *Routing Protocol Low Power and Lossy Network* Pada *Contiki Cooja* dengan Menggunakan Metode *Time Of Arrival***" ini dapat terselesaikan.

Dalam penulisan skripsi ini, peneliti menyadari bahwa skripsi ini tidak akan berhasil tanpa bantuan dari berbagai pihak, oleh karena itu penulis ingin menyampaikan rasa hormat beserta terima kasih kepada:

1. Ibu Ninik Setyowati dan Alm. Bapak Sutrisno selaku kedua Orang Tua penulis serta Sigit Atmaja Prihantara dan juga Samantha Era saudara kandung penulis yang telah mendidik dan membesarkan penulis serta yang tiada hentinya memberi dukungan, do'a, dan semangat hingga terselesainya skripsi ini.
2. Bapak Mochammad Hannats Hanafi Ichsan, S.ST, M.T beserta Bapak Rakhmadhany Primananda, S.T, M.Kom selaku dosen pembimbing skripsi yang telah meluangkan waktu pikiran dan tenaga sehingga skripsi ini dapat terselesaikan.
3. Bapak Tri Astoto Kurniawan, S.T., M.T., Ph.D selaku Ketua Jurusan Teknik Informatika.
4. Muhammad Naufal S.T yang telah mendampingi hari-hari penulis dalam pengerjaan skripsi serta memberi semangat dan doa kepada penulis.
5. Nahja Falahil Umam yang telah membantu penulis dan memberi arahan dalam pengerjaan skripsi.
6. Shinta Aprilisia, Irma Ardhi, Rini Meidita Audrey, Septya Mega, Ilham Satrio Utomo dan Anita Puspa Carolina selaku sahabat penulis yang selalu memberi dukungan semangat dan menemani penulis yang luar biasa mulai dari proses pertama perkuliahan hingga akhir proses skripsi ini
7. Teman-teman D'Gengs yang selalu memberikan bantuan dan dukungan kepada penulis.
8. Semua pihak yang tidak dapat penulis sebutkan satu-persatu. Terima kasih atas bantuannya.

Penulis menyadari bahwa skripsi ini masih jauh dari sempurna karena masih banyak kekurangan yang disebabkan oleh keterbatasan kemampuan dan ilmu yang dimiliki penulis. Saran dan kritik yang membangun dengan senang hati penulis terima demi perbaikan dan pengembangan skripsi ini. Semoga skripsi ini dapat memberikan manfaat yang besar bagi pembaca dan semua orang.

Malang, 20 Mei 2019

Putri Ayu Delina Sari

putridelina15.@gmail.com

ABSTRAK

Putri Ayu Delina Sari, Analisis Kinerja *Routing Protocol Low Power and Lossy Network* Pada *Contiki Cooja* dengan Menggunakan Metode *Time Of Arrival*

Pembimbing: Mochammad Hannats Hanafi Ichsan, S.ST, M.T dan Rakhmadhany Primananda, S.T, M.Kom

Wireless Sensor Network (WSN) adalah teknologi yang berkembang sangat pesat dan menyebabkan digunakan pada perangkat yang sehari-hari kita jumpai. Seluruh node sensor memiliki pengalamatan yang bermacam-macam dan memiliki kemampuan untuk mengambil, mengolah dan mengirim data. Dikarenakan keterbatasan sumber daya yang dimiliki, node WSN memiliki kemampuan memproses dan penggunanan daya konsumsi energi yang pada setiap node rendah, Hal tersebut membuat penulis merasa diperlukannya penelitian kinerja RPL yaitu *routing protocol* yang dikhususkan untuk *low power radio* pada simulator Contiki Cooja dan memiliki kelebihan yaitu setiap nodenya memiliki daya prosessor terbatas, memori rendah, konsumsi energi yang sedikit dan layanan *low data rates.* Penelitian ini melakukan pengujian dengan lima skenario yaitu 10, 20, 30, 40 dan 50 Node dengan parameter pengujian jarak hasil *localization, routing overhead* dan *packet delivery ratio*. Hasil pengujian menunjukan pengaruh parameter tersebut dengan jumlah node yang berbeda dalam suatu jaringan. Hasil pengujian *localization* menunjukan hasil pengujian dimana apabila komunikasi antara *node client* dan *node server* memakan waktu lama, maka dapat diketahui jarak antar node tersebut semakin jauh. Hasil pengujian *routing overhead* menunjukan trend nilai yang meningkat seiiring dengan bertambah jumlah node. Pada pengujian *routing overhead* dengan menggunakan 10, 20, 30, 40 dan 50 Node berturut-turut adalah 64.58%, 72%, 72.93%, 73.01% dan 75.56%. Hasil tersebut menunjukan bahwa semakin banyak node pada suatu jaringan, semakin besar pula *Packet Delivery Ratio*-nya.

Kata kunci: *Wireless Sensor Network, Routing Protocol Low Power and Lossy Network, IPv6, Time Of Arrival, Contiki Cooja*

ABSTRACT

Putri Ayu Delina Sari, Analisis Kinerja *Routing Protocol Low Power and Lossy Network* Pada *Contiki Cooja* dengan Menggunakan Metode *Time Of Arrival*

Supervisors: **Mochammad Hannats Hanafi Ichsan, S.ST, M.T and Rakhmadhany Primananda, S.T, M.Kom**

Wireless Sensor Network (WSN) is a technology that develops very rapidly and causes it to be used on devices that we encounter everyday. All sensor nodes have various addresses and have the ability to retrieve, process and send data. Due to limited resources, WSN nodes have the ability to process and use energy consumption power which at each node is low, which makes the writer feel the need for RPL performance research, namely routing protocol specifically for low power radio on Contiki Cooja simulator and has the advantage that each node has processor power limited, low memory, low energy consumption and low data rates. This study tested five scenarios, namely 10, 20, 30, 40 and 50 nodes with parameters of distance localization, routing overhead and packet delivery ratio. The test results show the influence of these parameters with the number of different nodes in a network. The localization test results show the results of testing where if the communication between client nodes and server nodes takes a long time, it can be seen that the distance between these nodes is getting farther away. The routing overhead test results show a trend of increasing values as the number of nodes increases. In routing overhead testing using 10, 20, 30, 40 and 50 Nodes, respectively 64.58%, 72%, 72.93%, 73.01% and 75.56%. The results show that the more nodes on a network, the greater the Packet Delivery Ratio.

Keywords: *Wireless Sensor Network, Routing Protocol Low Power and Lossy Network, IPv6, Time Of Arrival, Contiki Cooja*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Wireless Sensor Network (WSN) adalah teknologi yang berkembang pesat hari ini dan menyebabkan banyaknya perangkat yang menggunakan teknologi WSN pada keseharian manusia (Imaduddin, 2015). *Wireless Sensor Network* adalah teknologi pengambilan dan pemantauan data pada sebaran area tertentu yang berisi sensor-sensor ataupun *sensor node* yang berada menyebar dalam lingkup area yang telah didefinisikan serta dikoneksikan dengan perantara sebuah sistem jaringan tanpa kabel. Piranti dari teknologi WSN yang memunculkan banyak pandangan baru tentang teknologi salah satunya adalah *Internet of Things* (IoT). Sebuah perangkat memiliki keterbatasan daya, menyebabkan perangkat pada WSN mengirimkan sinyal dengan energi yang rendah serta kemampuan komputasi yang berdaya rendah juga. *Internet Engineering Task Force* (IETF) membentuk kelompok kerja *IPv6 Over Low Power Wireless Personal Area Network* (6LoWPAN) IP untuk melengkapi konektifitas yang ada di teknologi WSN dan jaringan internet dan mampu menghubungkan perangkat kecil berdaya rendah yang ada didalam lingkup jaringan nirkabel, dan pada akhirnya dikembangkan menjadi standar protokol *routing* yang diberi nama *Routing Protocol Low-Power and Lossy Network* (RPL).

Pada teknologi WSN terdapat sebuah metode yang dipakai untuk mencari letak fisik dari *node-node* disekelilingnya atau biasa disebut juga dengan lokalisasi atau *localization*. Metode *Localization* merupakan estimasi posisi melewati koneksi antara *node* lokal dan *unlocalized node* untuk mengetahui penempatan geometris dari sebuah node (Alrajeh, Bashir, & Shams, 2013). Dalam sebuah sistem WSN, node cenderung memakai lebih sebuah node sensor yang saling berkomunikasi antara node satu dengan node yang lain. Setiap node sensor memiliki pengalamatan yang berbeda dan memiliki kemampuan untuk mendapatkan data, memproses data dan mentransmisikan data.

Pada Algoritma *localization* terdapat dua kategori yaitu metode *localization range based* dan metode *localization range free*. *Localization range based* adalah sebuah protokol yang menggunakan estimasi jarak absolut *point-to-point* (kisaran) atau perkiraan sudut untuk menghitung lokas modei. Metode ini menggunakan pengukuran seperti sudut kedatangan (*Angle of Arrival),* waktu kedatangan (*Time of Arrival*), perbedaan waktu *range free* kedatangan (*Time Difference of Arrival*) dan penerimaan kuat sinyal (*Received Signal Strength*). (Empriantomo, Kristalina, & Pratiarso, 2012).

Menurut Penelitian berjudul ”Analisis RSSI (Receive Signal Strength Indicator) Terhadap Ketinggian Perangkat Wi-Fi Di Lingkungan Indoor” oleh (Puspitasari, 2010) metode RSSI pada *localization* memiliki beberapa kelemahan yaitu saat *transmitter* dan *receiver* terhalang sebuah objek disekelilingnya maka dapat terjadi peredaman sinyal, pembelokan sinyal dan pemantulan sinyal yang menyebabkan turunnya kekuatan sinyal yang ditransmisikan oleh *transmiter*

menuju *receiver*, walau sebenarnya jarak antar *transmiter* dan *receiver* cukup dekat, apabila terhalang oleh adanya objek disekitarnya, maka kekuatan sinyalnya akan turun dan probabilitas kekuatan sinyal nya bakal sama untuk kekuatan sinyal jika dibandingkan dengan jarak antar *transmiter* dan *receiver* lumayan cukup jauh, tapi tidak berhadapan dengan penghalang disekelilingnya. Menurut Jurnal Ilmiah berjudul "*Comparison Of Time-Difference-Of-Arrival And Angle-Of-Arrival Methods Of Signal Geolocation*" (Union, 2018), metode TDOA dan AOA tidak cocok pada lingkungan yang memiliki banyak emitor. Hal ini menyebabkan penurunan transmisi data yang drastis. Dengan mengacu pada permasalahan ini, penulis berniat menguji metode *localization* dengan metode lain yaitu *Time of Arrival*, dengan metode ini hal serupa diharapkan tidak akan terjadi karena metode *Time of Arrival* menguujur jarak dengan waktu paket dikirim dan paket diterima antar node.

Terkait sumber daya dari node WSN yang terbatas dikarenakan kemampuan komputasi yang rendah serta penggunanan daya konsumsi energi yang digunakan tiap node rendah (Danuansa, Yulianto, & Prabowo, 2017), penulis merasa diperlukan penelitian yang mengambil tema tentang kinerja RPL yaitu routing protocol yang dikhususkan untuk *low power radio* yang memiliki kelebihan yaitu setiap nodenya memiliki daya prosessor terbatas, memory rendah, konsumsi energi yang sedikit dan layanan *low data rates.*

Simulator Cooja adalah simulator yang dapat dapat mensimulasikan WSN dengan berbasis ContikiOS dan TinyOS. Karakteristik utama dari Simulator Cooja yaitu dapat mensimulasikan *sensor node* mengacu dari karakter perangkat yang sebenarnya, karena simulator Cooja memiliki kemampuan Java Native Interface (JNI) untuk menjalankan *source code* ContikiOS dan TinyOS. JNI memberikan hubungan antar *source code* mengunakan bahasa pemrograman C yang termasuk dalam Java Virtual Machine. Karena karakterisis simulator tersebut, simulator Cooja dapat mensimulasikan berbagai *platform sensor node* mendekati perangkat *sensor node* yang sebenarnya. (Hendrawan, 2018).

Pada penelitian ini, penulis memilih *localization* dengan metode *Time of Arrival* (ToA) dan menguji kinerja dari protokol RPL dengan 5 skenario pengujian yang berbeda berdasarkan parameter *Packet Routing Overhead* yang dimana *Packet Routing Overhead* adalah parameter yang mengukur keefisienan kinerja suatu protokol *routing* dengan menghitung banyaknya jumlah paket *routing* yang di transmisikan (Putra, Yulianto, & Herutomo, 2015) serta menggunakan parameter *Packet Delivery Ratio* yang dapat mengukur presentase keberhasilan paket data sampai ke tujuan(Rianda, Munadi, & Negara, 2016).

Berdasarkan uraian diatas, penulis melakukan sebuah peneitian dengan metode localization dan routing RPL menggunakan Simulator Cooja yang bertujuan untuk menganalisa kinerja routing RPL dengan metode Time Of Arrival. Maka dibuatlah penelitian dengan judul "Analisis Kinerja *Routing Protocol Low Power And Lossy Network* Pada Contiki Cooja Dengan Menggunakan Metode *Time Of Arrival*".

1.2 Rumusan Masalah

Berdasarkan latar belakang tersebut, akan dirumuskan permasalahan pada penelitian ini sebagai berikut:

1. Bagaimana membangun lingkungan simulasi WSN dengan protokol *routing Routing Protocol for Low Power and Lossy Network (RPL)*.
2. Bagaimana analisis hasil kinerja *routing protocol low power and lossy network* pada *simulator contiki cooja* dengan menggunakan metode localization *time of arrival.*
3. Bagaimana analisis hasil kinerja *routing protocol low power and lossy network* pada *simulator contiki cooja* dengan pengujian *Packet Routing Overhead.*
4. Bagaimana analisis hasil kinerja *routing protocol low power and lossy network* pada *simulator contiki cooja* dengan pengujian *Packet Delivery Ratio.*

1.3 Tujuan

Setelah dirumuskan beberapa masalah, dapat didefinisikan tujuan penelitian ini sebagai berikut:

1. Untuk mengetahui cara membangun simulasi WSN berbasis RPL dengan Simulator *Contiki Cooja* .
2. Untuk mengetahui hasil kinerja *routing protocol low power and lossy network* pada *contiki cooja* dengan menggunakan metode localization *time of arrival.*
3. Untuk mengetahui hasil hasil kinerja *routing protocol low power and lossy network* pada *contiki cooja* dengan pengujian *Packet Routing Overhead.*
4. Untuk mengetahui besaran paket *Routing Overhead* dan *Packet Delivery Ratio* pada protokol RPL.

1.4 Manfaat

Setelah mendefinisikan tujuan penelitian, beberapa manfaat setelah penelitian sebagai berikut:

1. Mengetahui cara membangun lingkungan simulasi pada RPL pada jaringan sensor nirkabel pada simulator *Contiki Cooja*.
2. Mengetahui kinerja protokol *routing* RPL pada jaringan sensor nirkabel.
3. Dapat membantu para peneliti dalam penelitian menggunakan RPL atau *Localization*.
4. Bagi penulis, penelitian ini dapat membantu untuk menentukan lokasi pada kejadian di dunia nyata contoh : untuk mencari suatu titik pusat lokasi pada kejadian bencana alam, tanah longsor atau gunung meletus.

1.5 Batasan Masalah

Supaya penelitian ini sesuai dengan latar belakang dan berfokus dengan rumusan masalah, maka batasan masalah pada penelitian ini sebagai berikut:

1. Lingkup simulasi jaringan sensor nirkabel dibangung dengan *Cooja Simulator.*
2. Simulasi menggunakan komunikasi routing RPL dengan metode *localization Time of Arrival*
3. Jumlah node adalah 10, 20, 30, 40 dan 50 node yang digunakan untuk pengujian fungsi *localization,* fungsi *routing over head* dan juga fungsi *packet delivery ratio*.
4. Pada fungsi localization jarak dan waktu adalah jarak antara node server dan node client.
5. Menggunakan analisis statistika secara umum.
6. Parameter yang dipakai untuk pengujian kinerja protokol RPL adalah jarak, *routing overhead* dan juga *packet delivery ratio*.

1.6 Sistematika Pembahasan

Sistematika pembahasan dari penyusunan penelitian yang direncanakan adalah sebagai berikut :

BAB I PENDAHULUAN

Bab Pendahuluan berisikan latar belakang penyusunan penelitian, identifikasi, batasan masalah, rumusan masalah, tujuan dan manfaat serta sistematika penulisan dari penelitian berjudul "Analisis Kinerja Routing Protokol Low Power and Lossy Network Pada Contiki Cooja dengan Menggunakan Metode Time Of Arrival".

BAB II LANDASAN KEPUSTAKAAN

Bab Landasan Kepustakaan berisi tentang teori – teori yang digunakan sebagai dasar penelitian, temuan dan bahan penelitian yang diperoleh dari beberapa referensi karya ilmiah yang mendukung penelitian dalam penulisan tugas akhir penulis.

BAB III METODOLOGI

Bab Metodologi berisi langkah-langkah yang dikerjakan pada awal penelitian sampai dengan langkah penyusunan penelitian dibahas dalam bab ini diantaranya studi literatur, perancangan dan implementasi sistem, serta pengujian dan analisis fungsi *localization*, fungsi *routing overhead*, fungsi *packet delivery ratio* dan yang terakhir dilakukan penarikan kesimpulan pada penelitian ini.

BAB IV PERANCANGAN DAN IMPLEMENTASI

Bab Perancangan sistem membahas tentang perancangan sistem yang akan menunjang penelitian ini. Bab ini ini menjelaskan perancangan sistem secara umum, alur kerja sistem, dan perancangan fungsi *localization*, perancangan fungsi *routing overhead* dan perancangan fungsi *packet delivery ratio*. Bab Implementasi menjabarkan bagaimana sistem akan diimplementasi berdasarkan sistematika yang telah disusun sebelumnya secara mendetail beserta langkah-langkah pengerjaan untuk menguji sistem yang akan diimplementasikan.

BAB V HASIL DAN ANALISIS

Bab ini berisi menjelaskan tentang hasil pengujian serta analisis dari kinerja sistem yang telah dibangun. Pengujian dilkakukan dengan menggunakan parameter fungsi *localization*,fungsi *routing overhead* dan juga fungsi packet *delivery ratio*. Pada bab ini dilakukan juga perhitungan *min*, *max,* dan rata-rata dari data yang telah di dapat untuk mengetahui hasil data yang kemudian akan di analisis dengan standar deviasi.

BAB VI PENUTUP

Bab ini berisi kesimpulan yang diambil dari hasil penelitian yang sudah dilakukan, berisi juga sub bab saran yang dapat digunakan untuk penelitian lebih lanjut.

BAB 2 LANDASAN KEPUSTAKAAN

Bab Landasan Kepustakaan beriisikan teori yang mendasari penilitian ini. Bab ini berisi tentang kajian pustaka yang berasal dari penelitian sebelumnya yang topiknya bersinggungan dengan penelitian ini. Sedangkan dasar teori menjelaskan teori-teori pendukung yang menjadi acuan terlaksanakannya penelitian.

2.1 Kajian Pustaka

Tabel 2. 1 Kajian Pustaka

No	Nama Peneliti, Tahun, Judul Karya Ilmiah	Kesamaan	Perbedaan	
			Penelitian Sebelumnya	Rencana penelitian
1.	Kevin Roussel, Ye-Qiong Song, Olivier Zendra, 2016, Using Cooja for WSN Simulations: Some New Uses and Limits	Simulasi WSN menggunakan Cooja	Menganalisis sejauh mana Cooja dapat digunakan untuk simulasi WSN	Berfokus pada kinerja 6LoWPAN pada WSN
2	Wawan Darmawan. Sabriansyah Rizqika Akbar, Mahendra Data, 2018, Analisis Performa Protokol 6LoWPAN pada Jaringan Sensor Nirkabel Menggunakan Cooja Simulator	Simulasi WSN menggunakan Cooja dan protocol 6LoWPAN	Menganilisis kinerja protocol 6LoWPAN dan QoSnya	Menggunakan statisika untuk menganalisis hasil pengujian.

2.2 Dasar Teori

Dasar teori adalah bagian yang menjelaskan tentang teori-teori yang akan mendukung dalam penyusunan penelitian ini.

2.2.1 *Wireless Sensor Network* (WSN)

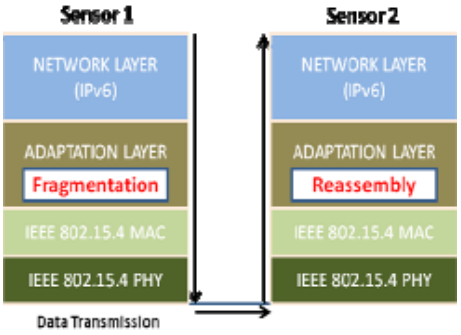
WSN (*Wireless Sensor Network*) adalah suatu teknologi jaringan nirkabel yang beranggotakan beberapa *node sensor* dan menyebar pada area yang sudah ditentukan. Node pada WSN mempunyai kemampuan pengolahan, memiliki beberapa jenis memori, mempunyai *transceiver RF,* memiliki sumber energi (pada WSN sumber daya biasanya baterai), serta dapat di aplikasikan pada berbagai sensor dan aktuator. WSN dapat diaplikasian pada bidang pengawasan keamanan, memonitor lingkungan sekitar, deteksi cahaya, deteksi suhu dan kelembaban, sampai pelacakan node pada jaringan.

Sampai detik ini, ada dua protokol yang banyak digunakan pada komunikasi sensor nirkabel yaitu *Zigbee* dan *6LoWPAN*. Keduanya secara masif dipakai dalam komunikasi WSN berenergi rendah. Pada pengaplikasiannya, protokol *zigbee* memungkinkan sebuah node untuk berjalan dengan status rendah energi sedangkan pengiriman data berbasis *multihop* dijanlakan memakai *tree routing protocol* yang menggunakan *cluster tree hierechy*. Sedangkan pada protokol *6LoWPAN* yang berdasar pada IPv6 bekerja secara asinkronus*,* dengan memakai topologi *mesh* dan protokol routing sehingga tidak mengharuskan agar mengatur node masuk pada posisi "tertidur" dikarenakan operasi ini menggunakan pendekatan *low-power listening* yang mengakibatkan tidak hemat energi (Toscano & Bello, 2012).

Faktor selanjutnya yang membuat berbedanya dua protokol ini adalah interoperabilitas, protokol 6LoWPAN menawarkan interoperabilitas dengan standar *IEEE 802.15.4 physical link devices* serta perangkat yang ada dalam jaringan IP lain contohnya *bridges,* dapat diartikan pengguna bisa melakukan komunikasi menggunakan perangkat apa saja menuju Internet dan tidak mengharuskan pengguna melewati alur konversi *zigbee to IP.* Hal tersebut berlaku juga pada keamanan, ini disebabkan karena jaringan membutuhkan *gateway* yang berfungsi memberi jalur dari perangkat ke jaringan, *zigbee* harus melewati dua proses pengamanan. Proses pertama adalah pada sisi jaringan dan disisi lain adalah Internet, proses tersebut menjadi pengaman dari serangan yang dapat membahayakan integritas *end-to-end link.* Namun karena protokol 6LoWPAN tidak mengharuskan semua itu, tidak dibutuhkan *header* dan menghasilkan ruang yang lebih untuk besar *payload* (Toscano & Bello, 2012).

2.2.2 6LoWPAN (*IPv6 over Low power Wireless Personal Area Networks)*

IPv6 over Low Power Wireless Personal Area Network atau biasa disingkat menjadi *6LoWPAN* adalah sebuah protokol memakai IP untuk berkoneksi dengan perangkat-perangkat yang memiliki daya kecil dan menempati area jaringan nirkabel. WPAN (*Wireless Personal Area Network*) adalah jaringan yang dipakai untuk perangkat pada lingkungan yang lebih spesifik dimana jaringan itu bedasarkan nirkabel. IPv6 dipakai untuk menaikan interoperabilitas dari protokol ini dan pada akhirnya dapat menyesuaikan diri pada perubahan jaringan yang beragam dari IoT. Layer model pada protokol 6LoWPAN dijelaskan pada Gambar 2.1.



Gambar 2. 1 Layer Model Pada Protokol 6LoWPAN

Sumber: Nurul (2012)

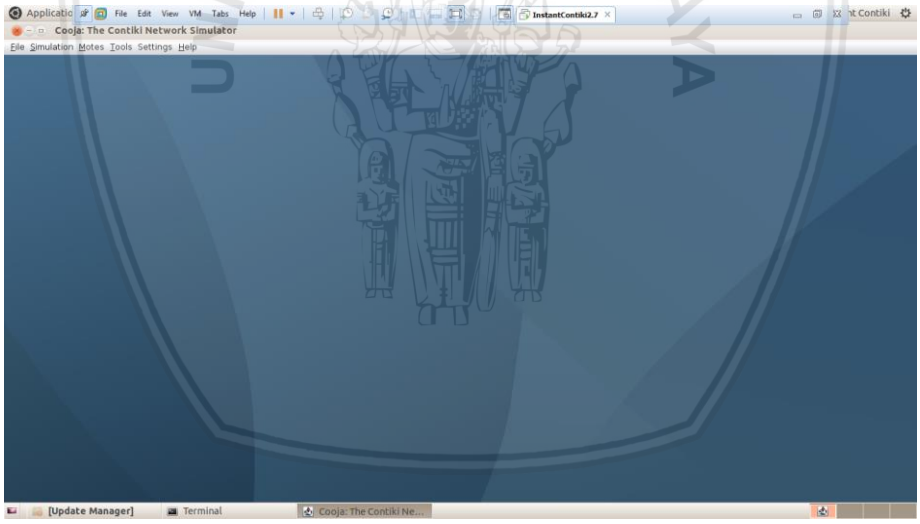
Stack layer dari protokol *6LoWPAN* berisikan *network layer, transport layer, application layer, PHY layer, MAC layer,* dan *adaptation layer*. Protokol 6LoWPAN memiliki dasar yang dapat memungkinkan piranti berstandar 802.15.4 untuk menransmisikan data antar satu sama lain secara nirkabel. Untuk lebih jelasnya, dibawah ini adalah penjelasan masing-masing layer pada protokol 6LoWPAN:

1. *6LoWPAN physical layer*, memberikan dua fungsi yaitu *PHY data sevices* dimana fungsi ini mengkoordinir data dikirim dan data yang diterima, serta *PHY management service* yang memberikan jalan ke *management function* pada setiap layer dan juga menyimpan data informasi *database personal area network* (PAN) yang terkoneksi.
2. *6LoWPAN MAC layer*, pada protocol berstandar IEEE 802.15.4, ada 4 standar pada *MAC layer frame*, yaitu yang pertama adalah *MAC data frame* berguna untuk pengiriman data, *MAC beacon frame* yang dihasilkan di *PAN coordinator* untuk menjalankan sinkronisasi antar node, *MAC command frame* yang dipakai oleh *MAC management entity* dan *MAC Acknowledgement* untuk mengetahui paket yang telah sukses terkirim.
3. *6LoWPAN adaptation layer*, protokol 6LoWPAN mengadopsi berdasarkan IPv6, namun untuk mengaplikasikannya pada IEEE 802.15.4 diperlukam beberapa penyesuaian yaitu menjalankan pemecahan paket data, mengkompresi *header* lalu menjalankan penyusunan ulang paket data yang sebelumnya difragmentasi. 6LoWPAN mengatur bagaimana routing bekerja pada sebuah jaringan pada layer ini
4. *6LoWPAN network layer*, layer ini mengatur interoperabilias antar node sensor dengan node yang lain, karakteristik utama pada layer ini adalah memberikan alamat, *mapping* dan *routing* protocol.
5. *6LoWPAN transport layer*, seperti yang ada pada sistem layer OSI model, layer ini berfungsi untuk mengatur *process-to-process delivery.*
6. *6LoWPAN application layer,* layer ini memakai *interface socket* yang berfungsi untuk penggunaan yang lebih spesifik dan dapat dipakai untuk menerima dan mengirim paket data antar node.

2.2.3 *Cooja Simulator*

Cooja Simulator adalah simulator untuk jaringan WSN yang dibuat dan dikembangkan lebih lanjut oleh *Team Project Contiki OS* . Selain *Cooja Simulator,* ada beberapa *simulator/emulator tools* yang sering kali dipakai untuk meneliti WSN, seperti OpenSim yang dikembangkan oleh OpenWSN atau TOSSIM yang dikembangkan oleh TinyOS, tapi *Cooja Simulator* adalah sebuah alat yang secara masif dipakai pada dunia WSN dan juga telah termasuk di dalamnya terdapat MSPSim dan juga Avrora Software untuk menjalankan fungsi *cycle-exact moth emulation* (Kevin Roussel, 2016).

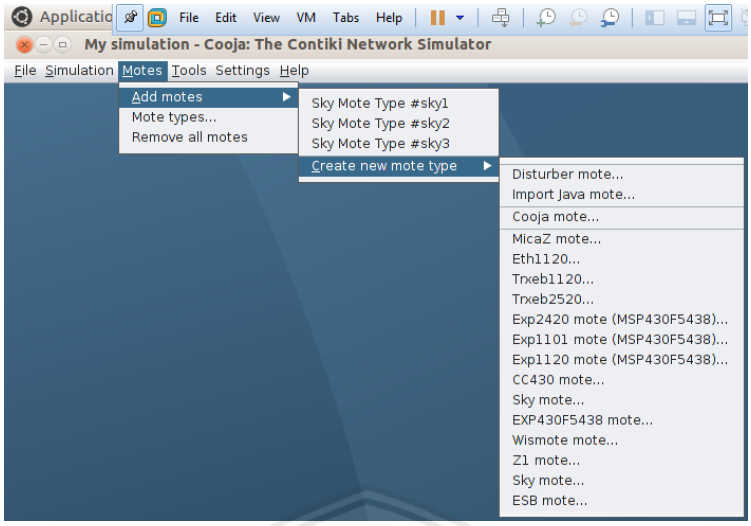
Para peneliti kebanyakan memakai *cooja* secara luas untuk menjalankan simulasi jaringan WSN yang dimulai dari level kecil sampai level yang lebih luas dengan melibatkan perangkat*, sensor*, dan juga *actuator*. Contiki Cooja banyak digunakan juga untuk meneliti efesiensi energi pada protokol keamangan jaringan WSN, analisis perfoma protokol co-ap dan mqtt dan analisis protokol 6LoWPAN. *Simulator* secara khusus akan berguna jika menjalankan *virtual runs* pada lingkup jaringan yang luas sebab secara signifikan bisa menurunkan biaya, jika pemakaian perangkat sudah sangat banyak, dan juga dapat menurunkan waktu untuk mensimulasikan WSN (Garg, 2015). Interface dari *Cooja simulator* dapat dilihat pada Gambar 2.2.



Gambar 2. 2 Tampilan Awal Cooja Simulator

Sumber : Cooja Simulator

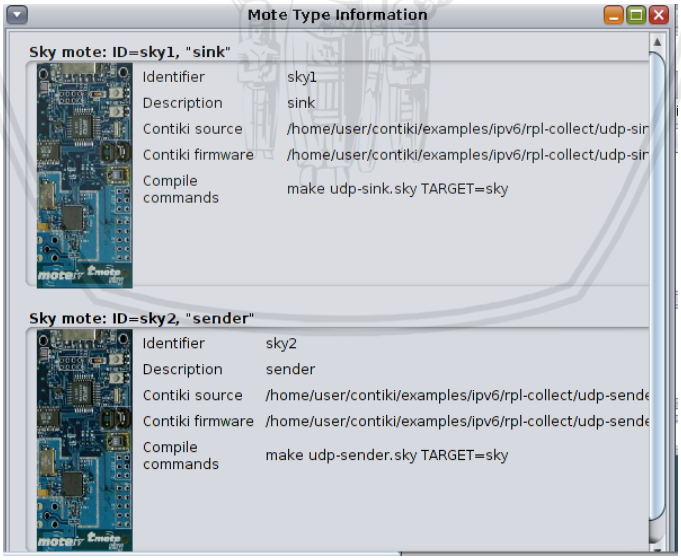
Cooja Simulator memiliki fitur untuk mendesain simulasi jaringan dengan beberapa tipe node/mote yang telah disediakan oleh pengembang aplikasi. Inisialisasi awal untuk membuat sebuah simulasi jaringan adalah sebagai berikut, memilih pada taskbar *file > new simulation,* selanjutnya pengguna bisa memasukan nama simulasi yang dikehendaki, setalah itu pengguna akan membuat node/motes baru dengan memilih *motes > add motes > create new mote type* pada taskbar. Pada langkah ini pengguna akan memiliki beberapa menu motes seperti pada Gambar 2.3 berikut:



Gambar 2. 3 Memilih mote/node yang tersedia pada cooja simulator

Sumber : Cooja Simulator

Motes yang dipakai pada tugas akhir ini adalah tipe skymotes *low power wireless sensor* yang dirancang dan dikerjakan oleh University of California, Berkeley, Amerika Serikat. Skymote memberikan banyak keuntungan diantara lain kemampuan untuk memfasilitasi sensor jaringan data rate tinggi dimana membutuhkan daya yang relatif kecil. Sensor ini biasanya dipergunakan untuk mendapatkan data kelembaban relatif, suhu dan keadaan cahaya. Gambar 2.4 ada interface dari *Mote type information*.

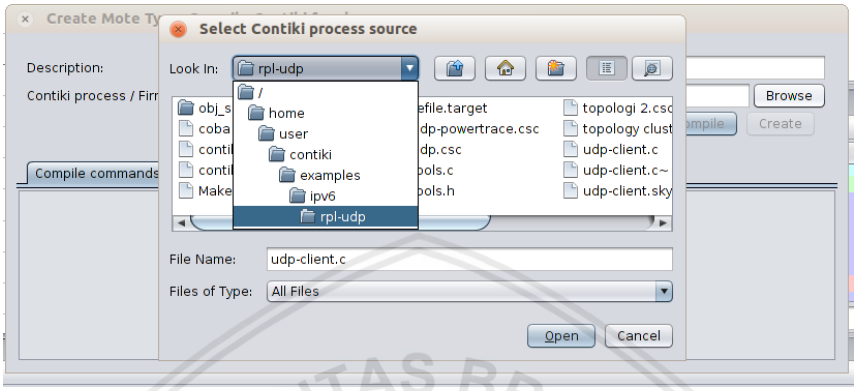


Gambar 2. 4 Tampilan Interface Motes Type Information

Sumber : Cooja Simulator

Setelah pengguna memilih motes/node yang akan digunakan, pengguna akan disughkan dengan menu contiki *process/firmware* adalah sebuah aplikasi berisi serangkaian *code process* dimana bertanggung jawab mengatur fungsi dan peran serta alur kerja mote/node yang telah dipilih. Pada penelitian ini, protokol

yang akan dipakai untuk pengujian adalah 6LoWPAN dan RPl adalah routing protokol yang akan dipakai, hal ini membuat *process/firmware* yang dipilih berada pada folder *home > user > contiki > examples > ipv6 > rpl-udp*, *firmware rpl-udp* digunakan karena pada folder ini node *client* dan *server* dapat menetukan secara mendetail pada saat menentukan *process/firmware* nya. Gambar 2.5 adalah interface saat menentukan *process/firmware* yang akan digunakan.



Gambar 2. 5 Interface Penentuan *process/firmware*

Sumber : *Cooja Simulator*

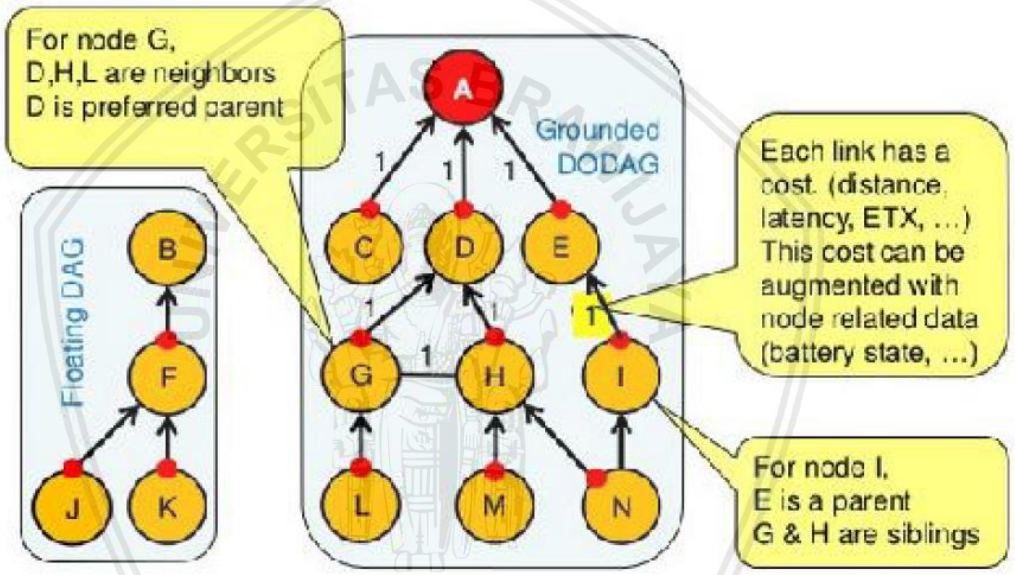
Saat *process/firmware* selesai ditentukan, proses selanjutnya adalah menyusun dan membuat motes yang nanti akan muncul merepresentasikan motes yang sesungguhnya di *Cooja Simulator*, tipe motes lain yang tidak mendukung *process/firmware* yang spesifik digunakan pada protokol 6LoWPAN dan menyebabkan saat proses *compile* muncul notifikasi error, hal ini dikarenakan lapisan jaringan pada protokol *6LoWPAN* berbeda dengan protokol internet lainnya. Karena hal tersebut skymotes dipilih karena tipe motes ini mendukung protokol 6LoWPAN.

Tahap selanjutnya adalah pengguna akan menentukan *firmware* apa yang akan digunakan pada motes tersebut dan motes tersebut akan bertugas sebagai apa. Pada folder *rpl-udp* dua *process/firmware* yang akan digunakan yaitu *udp-server.c* yang berperan sebagai *sink* dan *udp-client.c* yang bertugas menjadi node-node lain mewakili node sensor dan bakal mengirim data kepada node *sink.*

Cooja Simulator memiliki fitur yang unik, yaitu memberikan *tools* yang akan digunakan oleh *user* untuk mempermudah dan mendesain lingkungan simulasi dan juga memberikan hasil simulasi yang sedang berjalan. Agar pengguna dapat menggunakan *tools* yang disedikan pada fitur *cooja simulator, user* cukup dengan memilih menu *Tools* pada pilihan *task bar*. Beberapa contoh *tools* yang akan mempermudah penilitian di *cooja simulator* adalah *radio messages tools*, *tools* ini membuat pengguna dapat menjalankan pelacakan komunikasi data antara motes pada lingkup simulasi, total paket data yang dikirm, asal dan tujuan paket data tersebut dan juga mengetahui apakah motes yang digunakan sudah didukung oleh protokol 6LoWPAN atau belum.

2.2.4 *Routing Protocol for Low Power and Lossy Network*

RPL adalah protokol *Distance Vector* berdasar pada IPv6, dirancang khusus untuk WSN sesuai dengan RFC 6550. Setiap node pada RPL memiliki sebuah *parent* yang bertugas sebagai *gateway* untuk node tersebut. Apabila sebuah node tidakm memiliki informasi *table routing* pada sebuah paket data maka node itu akan mentransmisikannya ke node *parent*. Node *parent* mempunyai rank yang lebih rendah dari rank node tersebut. Semua node pada RPL diharapkan mamou merutekan semua node dalam topologi *tree*, sehingga node yang terdekat dengan root node akan memiliki *table routing* terbesar. Penyeleksian jalur pada RPL berbasiskan beberapa faktor untuk menghitung *routing metric* supaya mendapatkan jalur routing terbaik. Contoh cara kerja pada routing protocol RPL dapat dilihat pada Gambar 2.6.



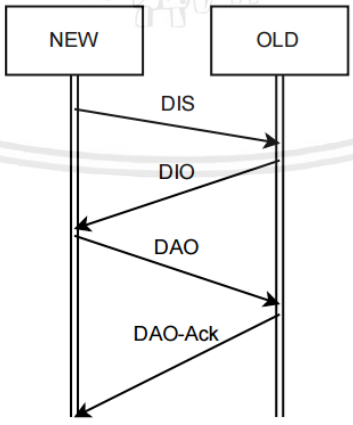
Gambar 2. 6 Cara Kerja Routing RPL

RPL menggunakan *Destination Oriented Directed Acyclic Graph* (DODAG) yakni Graph asiklik berarah untuk tujuan menuju *gateway (sink node)* yang disebut sebagai *Grounded DODAG* ataupun juga *LLN Border Router (LBR*) atau *sink node*. RPL akan membuat topologi jaringan berbentuk pohon *(tree)* yang biasa disebut *Directed Acyclic Graph (DAG*). *Sink node* atau *gateway* bertugas sebagai *root* dari beberapa DAG. Pada suatu jaringan, dapat berisikan beberapa DODAG yang terbuat dengan identifier unik bernama *RPLInstanceID.* DODAG merupakan bagian dari *Directed Acylic Graph* (DAG) yang khusus karena memiliki satu node tujuan, berikut adalah dasar-dasar yang digunakan pada routing protokol RPL:

1. *Directed Acylic Graph (*DAG*)* : Adalah suatu graph yang tidak mempunyai *cycle* pada prosesnya.
2. Root : Adalah sebuah Node yang bertugas menjadi destinasi dari node lainnya pada lingkup DAG.

3. *Destination Oriented Acylic Graph* (*DODAG*) :Adalah sebuah tipe DAG yang cara kerjanya adalah seluruh node akan mengarah ke sebuah node yang dijadikan destinasi node lainnya.
4. GOAL : Adalah destinasi akhir dari DODAG, apabila sebuauh DODAG telah sampai ke GOAL nya maka statusnya node tersebut akan berubah menjadi GROUNDED, apabila belum sampai ke GOAL nya maka status node menjadi FLOATING.
5. Parent dan CHILD : Diasumsikan contoh sebuah jaringan memiliki empat buah node yaitu A, B, C dan D dimana destinasi node B, C dan D mengarah pada node A, maka node A berperan sebagai node *parent*, setelah itu node B, C dan D berperan sebagai *child*. *Parent* mampu mempunyai lebih dari sebuah node yang berperan sebagai *child*, dan juga *child* mampu mempunyai lebih dari node yang berperan sebagai *parent*.
6. Storing : Adalah sebuah node yang mempunyai *routing table*, dan node tersebut mengerti alur kerja sebuah paket data terkirim dari sebuah node ke node yang lainnya lainnya pada jaringan.
7. Non-Storing : Berlawan dengan definisi dari *storing node*, yang berarti *Non-Storing* hanya mengetahui parent-nya saja.

Semua node yang menjankan mekanisme DODAG harus menjaga kesetaraan pada seluruh node yang berstatus *storing* dan seluruh node dengan status *non-storing* yang ada pada jaringan, sedangkan node *root* harus memilki kendisi *storing*. Gambar 2.7 menjelaskan tentang proses terciptanya suatu DODAG pada lingkup *routing protokol* RPL dengan memanfaatkan *Control Messages.*



Gambar 2. 7 Proses terbentuknya DODAG

1. DIO (*DODAG Information Object*) : Adalah sebuah pesan kontrol yang berasa dari node *root* ke seluruh node pada lingkup DODAG, seluruh node yang ada pada jaringan akan menerima dan mengirim *broadcast* pesan DIO. Pesan kontrol ini memberikan informasi tentang kondisi node

DODAG, apakah kondisi node tersebut *grounded* ataupun *floating*, *storing* ataupun *non-storing*. Kira-kira pesan kontrol ini akan berisi pesan bahwa "Apabila ada sebuah node pada jaringan yang akan bergabung, tolong respon dengan mengirimkan pesan balasan"

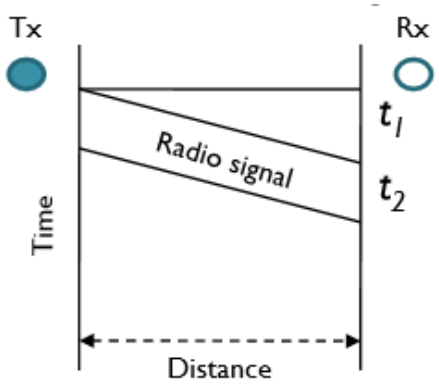
2. DIS (*DODAG Information Solicitation*) : Adalah sebuah pesan kontrol yang dikirimkan oleh suatu node yang belum bergabung dalam skema DODAG, mencari tahu apakah ada yang mempunyai DODAG pada lingkup jaringan itu.
3. DAO *(DODAG Advertisement Object)* : Adalah sebuah pesan kontrol yang ditransmisikan oleh node sebagai respon dari pesan kontrol DIO, pesan kontrol ini berisikan *request* untuk bergabung ke node *root* atau ke node *parent*
4. DAO-ACK : Adalah pesan kontrol balasan terhadap pesan kontrol DAO yang berisi pesan kontrol sebagai tanda izin untuk *join* dalam DODAG.

2.2.5 *Localization*

Localization adalah sebuah metode untuk menentukan posisi (contoh: koordinat) dari sebuah sensor atau hubungan spasial berbagai objek. Fungsi dari *Localization* dapat juga berfungsi untuk menentukan informasi jarak dan posisi sebagian atau seluruh node-node pada sebuah jaringan WSN yang melakukan estimasi posisinya. (Kristalina P. , 2013).

2.2.6 *Time Of Arrival*

Time of Arrival adalah sebuah metode *lozalization* dipakai apabila komunikasi terpusat berjalan. Pada *localization* dengan metode *Time of Arrival,* digunakanlah metode *ranging* yang berfungsi untuk menghitung waktu datangnya sebuah sinyal yang dikirimkan *trasmitter* menuju *receiver*. Saat ini, terdapat dua pendekatan yang biasa dilakukan para peniliti untuk mengimplementasikan metode *ranging* pada skema *localization*. Pendekatan metode *ranging* pertama adalah dengan memanfaatkan *transmitter* untuk mengirimkan sinyal menuju node *receiver* pada lingkup jaringan. Node *receiver* akan mentransmisikan waktu sampai sebuah sinyal menuju sebuah node terpusat untuk dibandingkan. Pada pendekatan kedua, node memakai sejumlah *transmitter* untuk mentransmisikan sinyal menuju node *receiver*. Pada tahap ini, node *receiver* melakukan mengkalkulasikan awal waktu kedatangan dari sinyal-sinyal yang datang lalu dibandingkan. Untuk mengetauhi jarak antar pengirim dan penerima sinyal, metode *Time of Arrival* menentukan jarak berdasarkan waktu propagasi sinyal yang ditangkap dan kecepatan sinyal yang diketahui (Prasetya, Renyati, Tatang, Arseno, & Budianto, 2008). Cara kerja metode *Time of Arrival* dapat dilihat pada Gambar 2.8 dibawah ini:



Gambar 2. 8 Diagram Cara Kerja Metode Time Of Arrival

2.2.7 *Routing Overhead*

Routing overhead adalah besarnya jumlah paket kontrol informasi yang dikirimkan sebuah node untuk mencari rute yang dipakai untuk menemukan node tujuan paket data dalam suatu jaringan. Semakin besar jumlah paket kontrol informasi maka semakin besar juga *routing overhead* dan kinerja protokol routing menjadi semakin buruk yang disebabkan oleh pemborosan *bandwidth* dalam transmisi paket tersebut. (Wibowo, Zuliansyah, & Rakhmatsyah, 2008)

2.2.8 *Packet Delivery Ratio*

Pengertian dari *Packet Delivery Ratio* adalah perbandingan antara banyaknya paket yang diterima oleh *destination* dengan banyaknya paket yang dikirim oleh *source* (Hendrantoro, 2013). Perhitungan untuk mendapatkan hasil persentase *packet delivery ratio* dapat dilihat pada rumus dibawah ini :

$$PDR = \frac{Prcv}{Ptr} \qquad (1)$$

Keterangan :

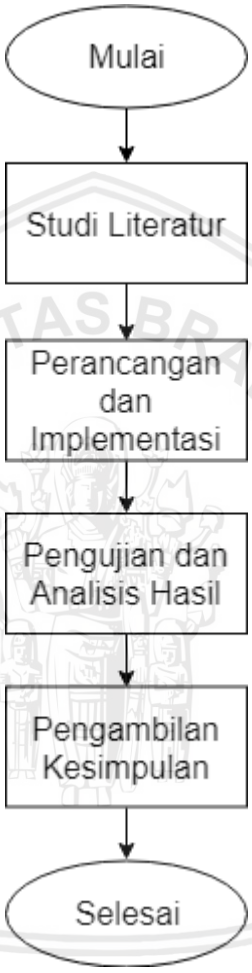
Prcv : Jumlah paket data yang diterima (paket)

Ptr : Jumlah paket data yang dikirim (paket)

PDR : *Packet Delivery Ratio* (%)

BAB 3 METODOLOGI

Bab Metodologi menjelaskan langkah penelitian yang dilakukan oleh penulis yang terdiri dari studi literatur, perancangan dan implementasi, melakukan pengujian untuk mendapatkan hasil, melakukan analisis hasil pengujian dan yang terakhir adalah penrikan kesimpulan. Gambar 3.1 merupakan tahap penelitian yang ditunjukan dalam bentuk *flowchart*.



Gambar 3. 1 Alur Penelitian

Pada Gambar 3.1 menjelaskan alur penelitian yang akan dilakukan dimulai dari studi literatur untuk memerdalam pemahaman sistem yang akan dibuat. Pada bab ini, pertama penulis melakukan studi literatur dari teori dan penelitian sebelumnya, melakukan perancangan sistem dan melakukan implementasi agar sistem berjalan sesuai dengan yang diharapkan. Setelah sistem berjalan, penulis melakukan pengujian sesuai scenario pengujian untuk mendapatkan hasil *output*. Setelah semua scenario pengujian dijalankan, makan penulis melakukan analisis dengan mengacu pada teori-teori yang sudah ada sebleumnya dan yang terakhir penulis bisa menyimpulkan hasil penelitian ini dan memberikan saran untuk referensi pengembangan sistem selanjutnya.

3.1 Studi Literatur

Sebagai bahan penunjang pengerjaan pada penelitian ini dilakukan studi literatur untuk memberikan pengetahuan yang cukup dalam mengembangkan sistem yang dibuat oleh penulis. Studi literatur dibutuhkan untuk mencari informasi dan referensi yang dilakukan dari penelitian-penelitian sebelumnya. Studi literatur dilakukan dengan memperoleh dasar-dasar teori yang berkaitan dengan penelitian ini yang meliputi *Wireless Sensor Network* (WSN)*, 6LoWPAN, Cooja Simulator, Routing Protocol for Low Power and Lossy Network* (RPL)*, Time of Arrival* (TOA), *Routing Overhead* (RO) dan *Packet Delivery Ratio* (PDR).

3.2 Perancangan dan Implementasi

Pada penelitian ini dibutuhkan desain topologi jaringan yangbersifat random. Hal ini bertujuan untuk melihat bagaimana node berkomunikasi satu sama lain dan bagaimana jika ada node yang tidak dapat mencapai node server secara langsung. Perancangan pada penelitian ini meliputi beberapa kebutuhan yaitu fungsi *localization*, fungsi besar paket *routing overhead* dan fungsi besar *packet delivery ratio.* Pada penelitian ini, proses pertama yang akan dilakukan menguji fungsi *localization* yang bertujuan untuk mengetahui jarak suatu node yang belum diketahui. Langkah selanjutnya adalah menguji fungsi besar paket *routing overhead* yang bertujuan untuk rasio antara jumlah paket *routing* dengan paket data yang berhasil diterima. Setelah itu perancangan yang terakhir menguji fungsi *packet delivery ratio* yang bertujuan untuk mengetahui rasio perbandingan total paket yang dikirim dan total paket yang diterima.

Implementasi pada penelitian ini meliputi 3 fungsi yaitu fungsi *Localization*, fungsi besar *routing overhead*, dan implementasi fungsi *packet delivery ratio*. Implementasi yang pertama menjalankan fungsi *localization* dengan cara meletakan satu node sebagai node server dan beberapa jumlah node lainnya sebagai node client yang akan dijalankan pada simulator untuk mendapatkan ouput jarak node yang belum diketahui sebelumnya. Implementasi selanjutnya menjalankan fungsi dari *routing overhead* dengan cara menghitung besar jumlah paket routing protokol RPL sehingga output yang di dapat adalah jumlah paket bedasarkan protokol routing RPL. Implementasi yang terakhir menjalankan fungsi *packet delivery ratio*. *Packet delivery ratio* ini diambil untuk mengetahui rasio perbandingan total paket yang dikirim dan total paket yang diterima. Saat node server mengirimkan data pada node client yang lebih dari satu, dari pengiriman data tersebut bisa dilihat perbandingan data yang bisa diterima oleh node server. Node server bisa menerima semua data atau hanya beberapa data dari node client yang bisa diterima. Bila hasilnya perbandingan yang di dapat adalah 99,9% atau mendekati 100% berarti hasil tersebut tidak eror.

3.3 Pengujian dan Analisis Hasil

Setelah melakukan perancangan dan implementasi sistem tahap selanjutnya yaitu melakukan pengujian kinerja untuk pengujian dan analisis. Pada penelitian ini, pengujian pada fungsi *localization* adalah mencari jarak node yang belum

diketahui disektiarnya. Pengujian selanjutnya menghitung jumlah paket *routing overhead* yang di dapat dari protokol RPL. Pengujian yang terakhir adalah menguji *packet delivery ratio* yaitu membandingkan rasio antara data terkirim dan data diterima pada suatu lingkup jaringan.

Metode analisis yang dilakukan adalah dengan mencari jarak antara *server* dan *client*, lalu mengkalkulasikan total routing yang dikirim dan perbandingan antara total paket yang dikirim dengan banyaknya paket yang terima pada suatu lingkup jaringan. Setelah mendapat hasil dari pengujian tersebut, lalu dianalisis dengan menggunakan menggunakan statistik umum yaitu Min, Max, Rata-rata dan Standar Deviasi.

3.4 Pengambilan Kesimpulan

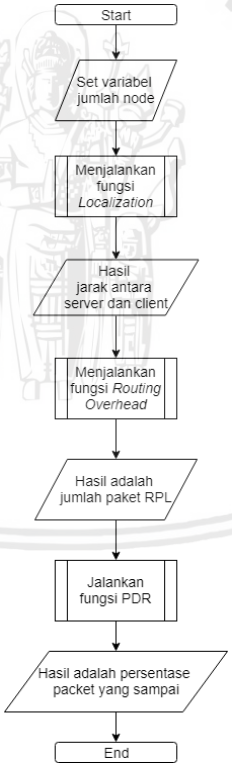
Pengambilan kesimpulan dilaksanakan setelah berhasil mendapatkan data pada pengujian serta dilakukan analisis pada data tersebut. Pengambilan keputusan didukung dengan literatur dan karye ilmiah yang dijadikan acuna pada penelitian ini. Hasil analisis tersebut lalu diolah menjadi sebuah kesimpulan yang dan menjadi jawaban atas rumusan masalah yang sudah didefinisikan sebelumnya. Setelah berhasil menyimpulkan hasil peneitian pada tugas akhir ini, penulis juga memberikan saran untuk peneliti lain agar dapat mengembakan penelitian ini lebih lanjut.

BAB 4 PERANCANGAN DAN IMPLEMENTASI

Bab ini akan membahas tentang implementasi perancangan dari penelitian yang akan dilakukan. Implementasi penelitian dilakukan pada *software* Simulator *Contiki Cooja.* Perancangan dan Implementasi pada penelitian ino meliputi Fungsi *Localization* Fungsi *Packet Routing Overhead*, Fungsi *Packet Delivery Ratio* dan Desain Topologi Jaringan.

4.1 Perancangan Sistem

Perancangan sistem merupakan tahap yang dilakukan setelah semua kebutuhan sistem dianalisis dan terpenuhi. Pada penelitian ini, dibutuhkan perancangan perangkat lunak untuk melakukan analasis kinerja *Routing Protocol for Low Power and Lossy Network* (RPL) pada Contiki Cooja menggunakan metode *Time of Arrival* (TOA). Untuk melakukan penelitian ini, diperlukan tiga buah fungsi yaitu Fungsi Localization, Fungsi *Packet Routing Overhead* dan Fungsi *Packet Delivery Ratio* (PDR). Gambar 4.1 menjelaskan cara kerja system secara keseluruhan :

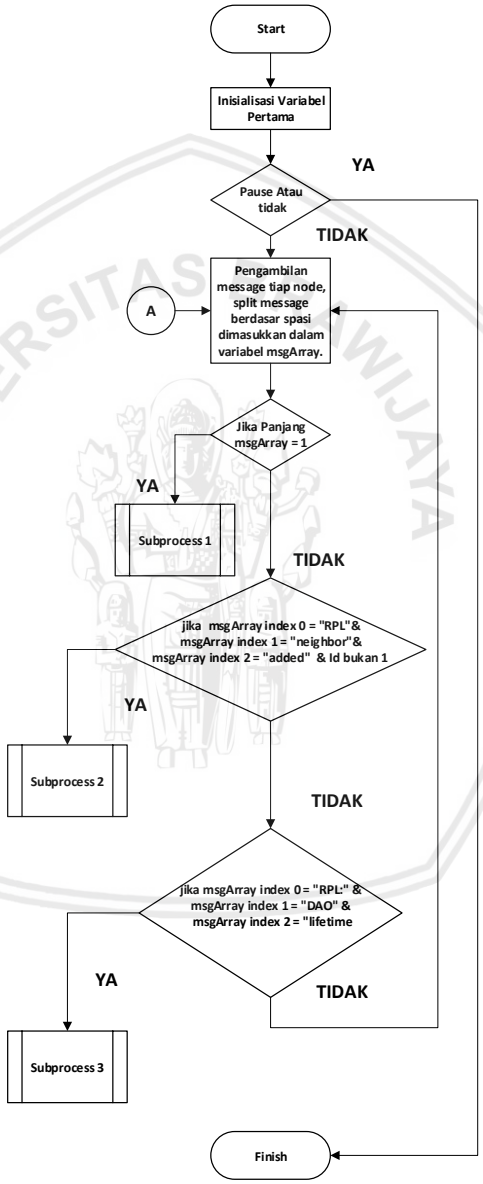


Gambar 4. 1 Flowchart Cara Kerja Sistem

Dimulai dari menjalankan fungsi localization agar node *server* dan *client* dapat mengestimasi jaraknya satu sama lain. Proses selanjutnya node-node tersebut akan dihitung jumlah packet *RPL* dengan fungsi *routing overhead*. Proses selanjutnya adalah menjalankan fungsi *packet delivery ratio* yang berfungsi untuk menghitung ratio packet yang sampai dari *server* ke *client*. Semua fungsi yang dijalankan akan menampilkan outputnya di kotak *message.*

4.1.1 Fungsi *Localization*

Bagian ini berisi penjelasan tentang implementasi *localizatio* yang bertujuan untuk mengetahui jarak antara node satu dengan node lainnya. Implementasi akan dilakukan dengan cara pertama menentukan berapa jumlah node yang akan kita masukan dalam simulator dan memilih suatu jenis node tersebut. Implementasi akan dilakukan di dalam simulator *Contiki Cooja* dan menggunakan IPv6. Flowchart cara kerja Fungsi Localization bisa dilihat pada Gambar 4.2 berikut

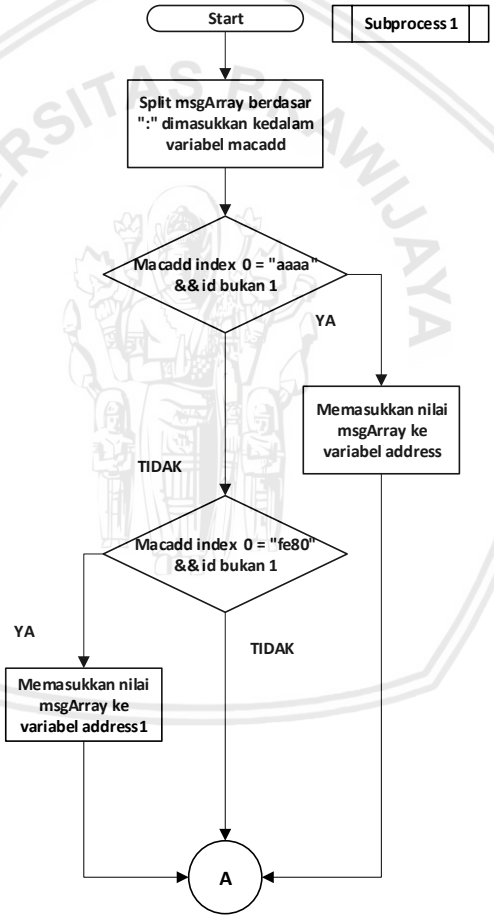


Gambar 4. 2 Flowchart *Localization*

Dimulai dari melakukan inisialisasi variable berupa *serverID*, *start time*, *receive time*, *jarak*, *countID*, address dan juga *time client* yang berfungsi sebagai acuan awal untuk memulai *localization.* Proses selanjutnya mengambil pesan setiap node, berdasarkan spasi yang akan dimasukan dalam variable. Jika Panjang

variable sama dengan 1, jika “YA” akan dilanjutkan untuk proses subproses satu. Namun jika “TIDAK”, proses selanjutnya diteruskan dengan kondisi jika variable indeks ke 0 yang berisi “RPL”, variable indeks 1 berisi “neighbor” dan isi variable indeks ke 2 berisi ”added” & bukan id satu terpenuhi maka akan dilanjutkan ke subproses dua, jika ”TIDAK” proses diteruskan ke proses selanjutnya.

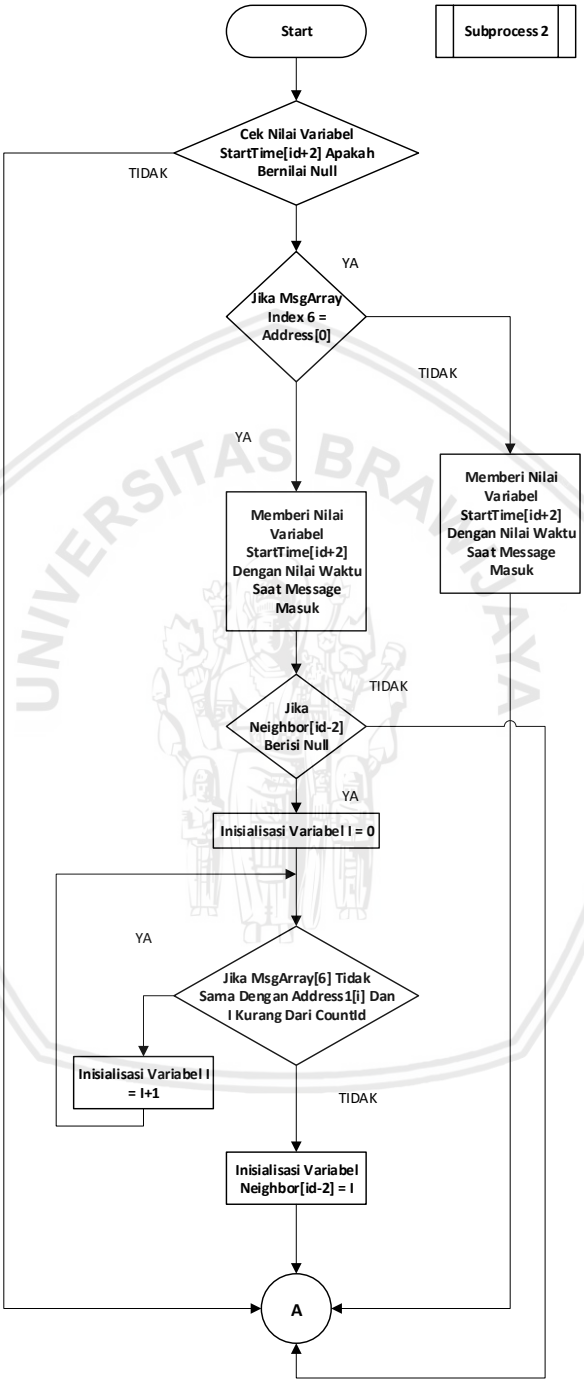
Selanjutnya apabila isi varible indeks ke 0 yang berisi pemecahan pesan “RPL” dan inisalisasi variable msgArray indeks ke satu berisi pesan dari RPL yaitu “DAO” dan isi variable msgArray indeks ke dua adalah “lifetime”. Jika kondisi proses tersebut terpenuhi maka dilanjutkan dengan subproses ke tiga, namun jika “TIDAK” akan kembali pada pengambilan pesan setiap node, dan pesan akan dipecah berdasarkan spasi kemudian dimasukkan dalam inisialisasi variable. Untuk proses berikutnya akan dijelaskan sesuai proses dari subproses satu sesuai pada Gambar 4.3 berikut:



Gambar 4. 3 Subproses satu Flowchart *Localization*

Langkah pertama setelah dimulainya proses yaitu membagi isi variable msgArray berdasarkan spasi yang dimasukan dalam variable *macaddress*. Kemudian macaddress index ke 0 yang berisi “aaaa” dan bukan id satu. Jika “YA” akan diteruskan dengan memasukkan nilai variable msgArray ke dalam variable address. Namun jika ”TIDAK” akan mengalami pemilihan kondisi kembali. Apabila macadd index 0 yang berisi ”fe80” && bukan id satu. Jika ”YA” memasukkan nilai

variable msgArray ke variable address satu. Tetapi jika "TIDAK" kembali pada pengambilan pesan tiap node, dan pesan dipecah berdasar spasi dimasukkan dalam inisialisasi variable. Kemudian untuk proses berikutnya akan dijelaskan pada flowchart subproses dua pada Gambar 4.4:



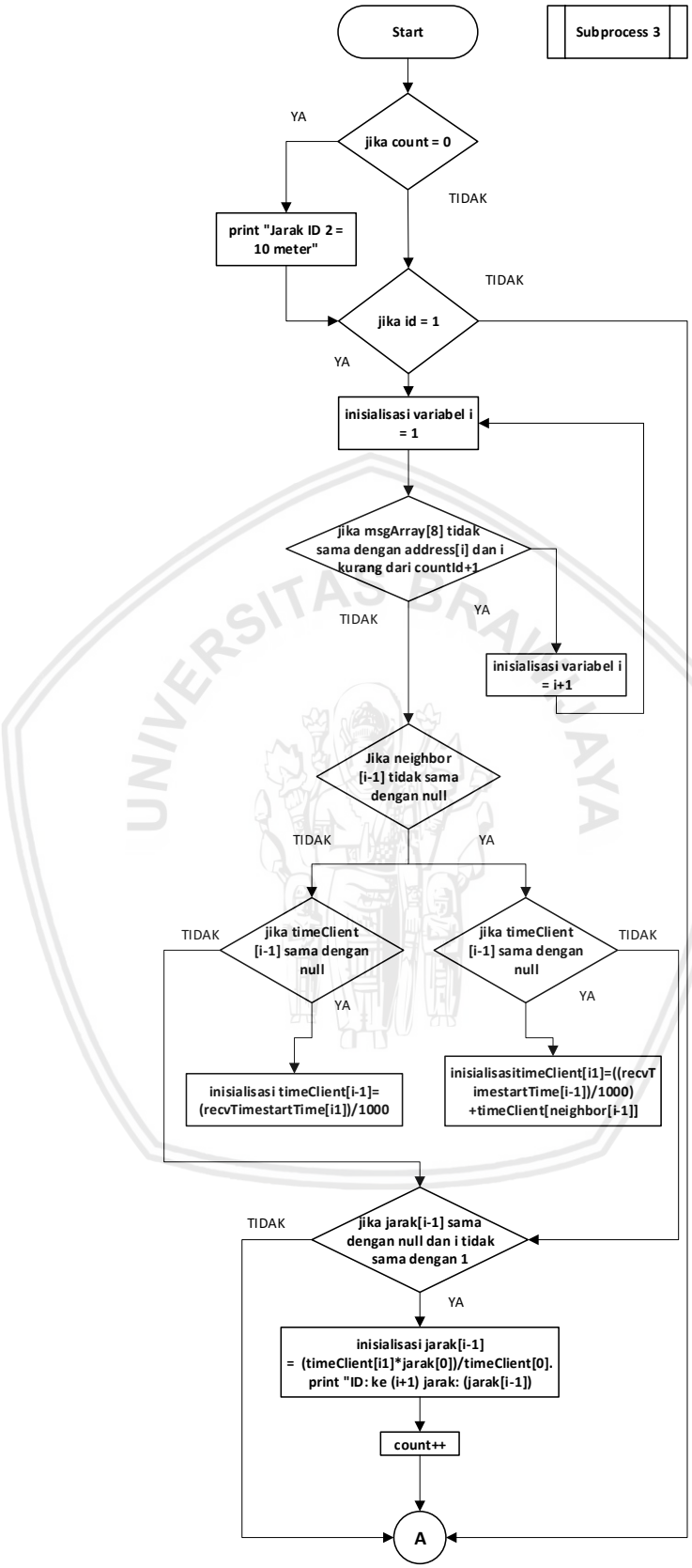
Gambar 4. 4 Subproses dua Flowchart *Localization*

Pada subproses dua flowchart localization menjelaskan proses mengecek nilai variable startTime apabila variable startTime index id +2, bernilai kosong, maka akan diteruskan ke proses selanjutnya. Namun jika "Tidak" kembali pada pengambilan pesan tiap node. Proses selanjutnya adalah apabila isi variable

msgArray index ke 6 yang berisi alamat index ke 0, maka akan memberi nilai variable startTime [id+2] dengan nilai waktu saat message masuk. Lalu jika ”TIDAK” proses berikutnya juga sama seperti proses yang sudah dijelaskan sebelumnya namun kembali melakukan pengambilan pesan tiap node.

Kemudian proses selanjutnya adalah melakukan pengecekan isi variable *neighbor*[id-2]. Apabila berisi null, maka dilanjutkan proses inisialisasi variable i berisi 0, jika tidak maka kembali pada pengambilan pesan tiap node, kemudian pesan dipecah berdasar spasi yang dimasukkan dalam inisialisasi variable. Selanjutnya untuk proses berikutnya adalah mengecek variable msgArray index ke 6, sama dengan address[i] dan i kurang dari jumlah id. Pada proses ini jika ”Ya”diteruskan dengan variable I = i+1, lalu kembali pada proses sebelumnya. Jika “Tidak” inisialisasi variable *neighbor* [id-2] = i dan kembali pada pengambilan pesan tiap node, kemudian pesan dipecah berdasar spasi yang dimasukkan dalam inisialisasi variable.

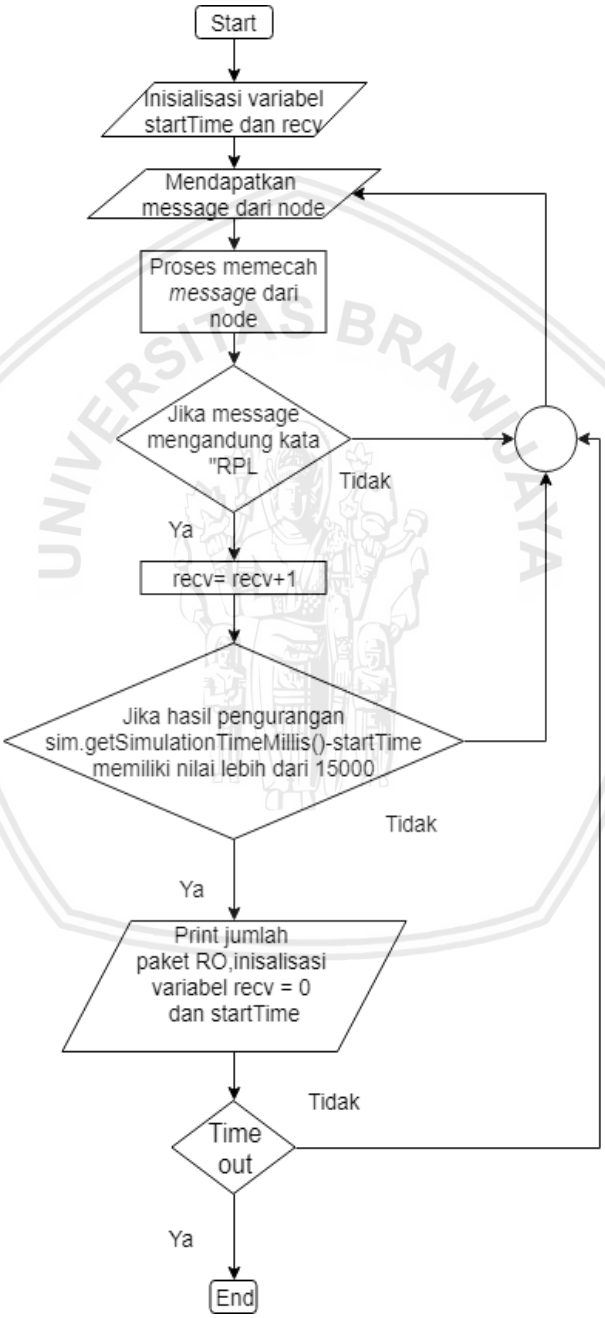
Proses selanjutnya adalah Subpocess 3. Pada subrprocess ini jika variable count bernilai nol, maka akan mencetak nilai jarak, apabila tidak maka melakukan pengecekan jika variable count berisi satu, maka inisialiasi variable i = 1. Setelah itu, jika variable msgArray index ke delapan tidak sama dengan variable address index ke i dan I kurang dari countid +1, maka akan dilakukan inisialisasi variable i = i+1. Proses selanjutnya adalah melakukan pengecekan kondisi variable neighbor index i-1, apabila variable tersebut berisikan null dan variable timeClient index ke i-1 sama dengan null, maka variable recieveTime dan startTime dikurangi lalu dibagi dengan 1000 dan ditambah dengan variable timeClient. Selanjutnya apabila variable tersebut tidak berisikan null dan variable timeClient index ke i-1 sama dengan null, maka variable recieveTime dan startTime dikurangi lalu dibagi dengan 1000. Proses selanjutnya adalah melakukan pengecekan jika variable jarak index i – 1 sama dengan null dan nilai i tidak sama dengan satu, akan dilakukan perhitungan dengan variable timeClient index ke 1 dikali variable jarak index ke 0 lalu mencetak hasil perhitungan tersebut dan melakukan increment pada variable count. Jika kondisi diatas tidak terpenuhi, maka akan kembali ke proses pengambilan message dari node. Untuk subproses tiga bisa dilihat pada Gambar 4.5 :



Gambar 4. 5 Subproses Tiga Flowchart Fungsi Localization

4.1.2 Fungsi *Packet Routing Overhead*

Packet Routing Overhead mempunyai fungsi untuk menerima atau mengetahui jumlah packet yang berhasil diterima dari RPL. Implementasi yang pertama dilakukan yaitu mengawali dengan inisalisasi variable berupa *timeout, startime, dan juga receive. Timeout* berguna untuk mengetahui dimana waktu akan berhenti saat proses pesan atau paket yang berhasil diterima telah selesai. Untuk proses fungsi *packet routing overhead* bisa dilihat pada Gambar 4.6 :



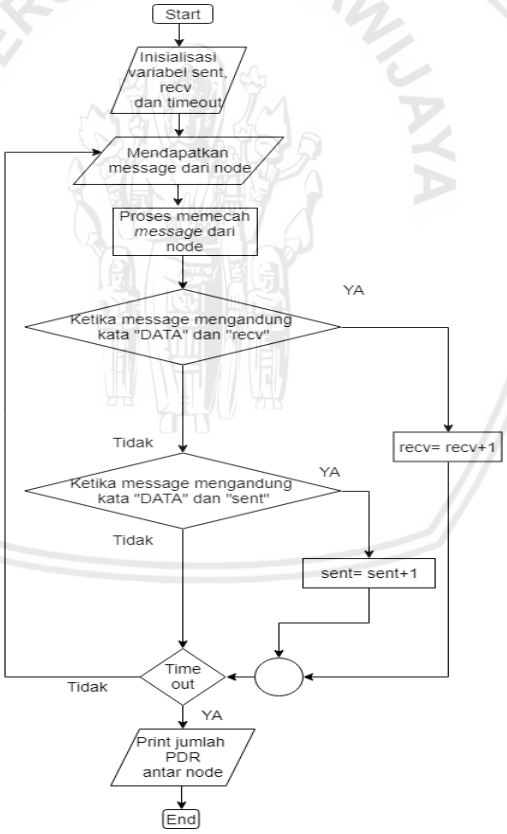
Gambar 4. 6 Fungsi *Packet Routing Overhead*

Proses pertama pada fungsi *packet routing overhead* adalah proses memecah pesan dan node, apabila *message* mengandung kata "RPL", maka pesan

diterima sama dengan pesan yang telah diterima dan bertambah satu, namun jika tidak maka proses kembali pada mendapatkan *message* dari setiap node. Kemudian jika hasil pengurangan untuk mendapatkan waktu timeout dikurangi dengan *startTime* (waktu awal saat pengambilan data) yang memiliki nilai lebih dari 15000. Jika kondisi terpenuhi maka dilaukan print jumlah *packet routing overhead,* inisialisasi variable receive sama dengan 0 dan kembali pada *startTime* waktu awal saat akan mulai pengambilan data. Akan tetapi jika kondisi tidak terpenuhi makan proses akan kembali pada pengambilan pesan dari tiap node tersebut. Untuk proses berikutnya yaitu *timeout*, jika mengalami timeout berarti proses end atau sudah selesai, namun jika tidak maka proses kembali pada pengambilan pesan kembali dari setiap node.

4.1.3 Fungsi *Packet Delivery Ratio*

Fungsi dari *packet delivery ratio* ini adalah untuk membandingkan data keseluruhan yang berhasil diterima dengan data yang tidak diterima *(packet loss*). Untuk penjelasan flowchart dari fungsi *packet delivery ratio* bisa dilihat pada Gambar 4.7:



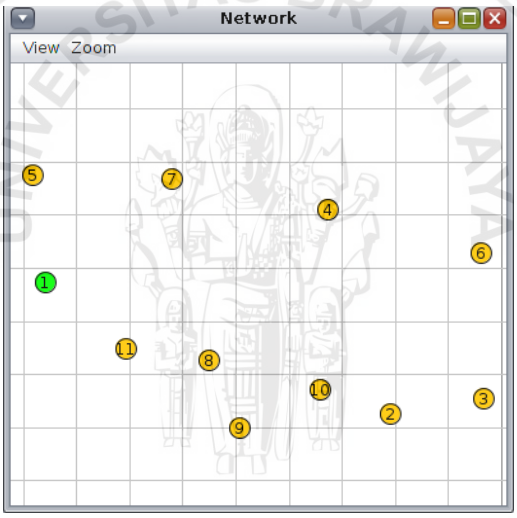
Gambar 4. 7 Fungsi *Packet Delivery Ratio*

Proses pertama fungsi PDR adalah inisialisasi variable antara lain, *sent*, *receive* dan juga *timeout*. Setelah itu dilakukan proses untuk mendapatkan pesan dari setiap node. Lalu proses berikutnya yaitu memecah *message* dari node. Ketika *message* mengandung kata ”data” dan “*receive*”, maka proses diteruskan dengan *receive*(pesan diterima) sama dengan *receive* nilai lebih dari satu. Namun tidak

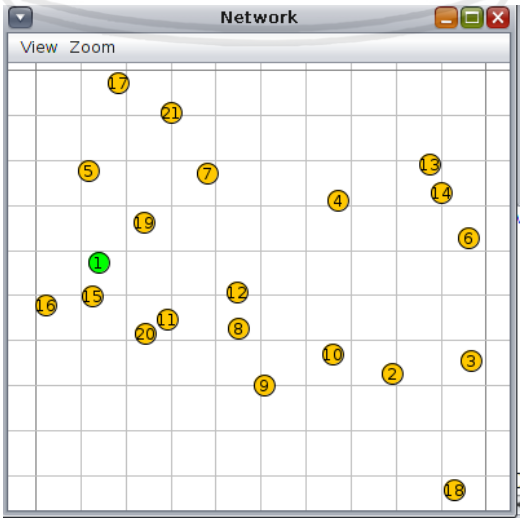
sesuai kondisi tersebut maka akan kembali pada inisialisasi variable seperti proses yang awal. Proses selanjutnya ketika *message* berisi "Data" dan "sent" maka variable sent di increment yang menandakan ada data yang berhasil terkirim. Apabila tidak maka akan dicek apakah sudah *timeout* (waktu berhenti) atau belum, jika kondisi terpenuhi maka menandakan *timeout* maka hasil presentase PDR akan dicetak, jika kondisi tidak terpenuhi maka akan menunggu mendapatkan message lagi dari node.

4.1.4 Desain Topologi Jaringan

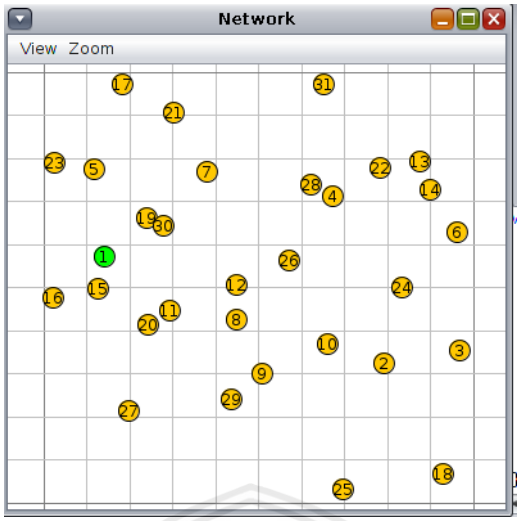
Desain topologi jaringan yang akan digunakan pada penelitian ini adalah toplogi keseluruhan sistem. Hal ini bertujuan untuk melihat skema yang dilakukan saat pengujian pada penelitian ini. Pada penelitian ini, penulis menggunakan peletakan node secara random, node *server* yang digunakan adalah 1 untuk semua pengujian dan node *client* berjumlah 10, 20, 30, 40 dan 50 bergantung dengan pengujian yang sedang dijalankan. Untuk setiap peletakan node yang digunakan pada penelitian ini bisa dilihat pada Gambar 4.8 berikut:



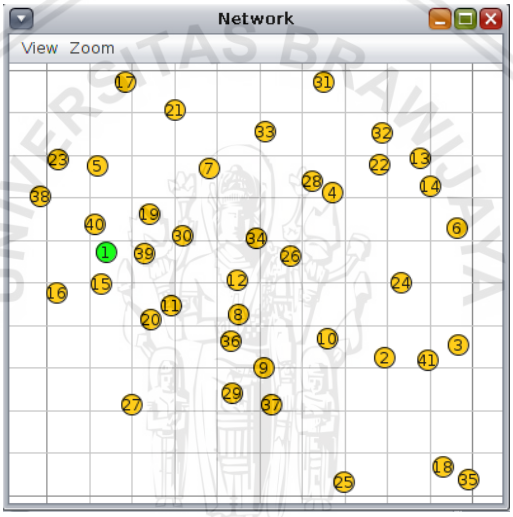
(a) Simulasi 10 Node



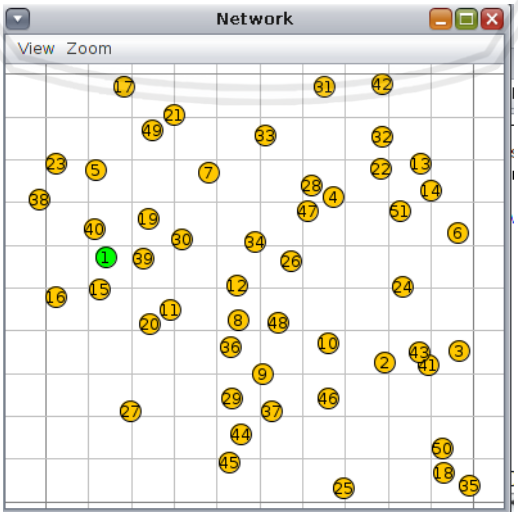
(b) Simulasi 20 Node



(c) Simulasi 30 Node



(d) Simulasi 40 Node



(e) Simulasi 50 Node

Gambar 4. 8 Sebaran Node Pada Simulator Cooja

4.2 Implementasi

Pada sub bab implementasi yaitu membahas mengenai tahapan yang berdasarkan dari perancangan yang telah dirancang sebelumnya dan dibagi menjadi tiga bagian yaitu implementasi fungsi localization, fungsi packet routing overhead dan fungsi packet delivery ratio.

4.2.1 Implementasi Fungsi *Localization*

Pada proses pertama Fungsi localization akan dilakukan dua hal yaitu Inisialisasi dan Proses Memecah Pesan. Pertama adalah melakukan insialisasi variable serverID, startTime, recvTime, jarak, countId, count, address, address1, neighbor dan timeClient. Lalu program utama akan melakukan perulangan dengan while agar program berjalan secara terus menerus. Jika Panjang variable msgArray adalah satu, maka akan dijalnkan proses memecah pesan dengan delimeter dan memasukan hasil proses tersebut ke variable msgArray. Untuk source code fungsi tersebut dapat dilihat pada tabel 4.1 berikut :

Tabel 4. 1 Fungsi Localization Inisialisasi dan Subprocess Satu

Algoritma 1 : Fungsi Localization Inisialisasi dan Subprocess Satu	
1	serverID = 1;
2	startTime = new Array();
3	recvTime = 0;
4	jarak = new Array();
5	jarak[0] = 10;
6	countId = 45;
7	count=0;
8	address = new Array();
9	address[0]="fe80::212:7401:1:101";
10	address1 = new Array();
11	neighbor = new Array();
12	timeClient = new Array();
13	while(1){
14	YIELD();
15	
16	msgArray = msg.split(' ');
17	if(msgArray.length == 1){
18	macadd = msg.split(':');
19	if (macadd[0].equals("aaaa") && id>1){
20	address[(id-1)]=msg.toString();
21	log.log(address[id-1] + "\n");
22	}if (macadd[0].equals("fe80") && id>1){
23	log.log(address1[id-2] + "\n");
24	}
25	}

Setelah proses memecah pesan, proses selanjutnya adalah mengecek apakah pada variable msgArray index ke nol berisi data “RPL”, index pertama berisi data “Neighbor”, index kedua berisi data “added” dan variable id bukan sama dengan satu. Jika kondisi tersebut terpenuhi maka selanjutnya akan dilakukan pengencekan variable StartTime [id+2] bernilai null maka akan dilanjutkan ke proses berikutnya. Sedangkan bila tidak, proses akan kembali ke pengambilan message tiap node. Selanjutnya adalah mengecek apabila data pada variable msgArray index ke enam sama dengan data pada variable address index ke nol, apabila sama maka akan dilakukan input data ke variable StartTime [id-2] dengan

waktu saat message masuk, jika tidak maka akan dilakukan proses yang sama yaitu input data ke variable [id-2] dengan waktu saat message masuk.

Selanjutnya proses akan mengecek variable Neighbor [id-2] apakah null atau tidak. Jika iya, maka akan menginisialisasi variable i = 0, namun bila tidak makan proses akan kembali ke pengambilan message tiap node. Setelah itu program akan melakukan pengecekan kembali pada variable msgArray, address1 dan countid. Jika variable msgArray index ke enam tidak sama dengan variable address1 dan index i di variable address1 kurang dari variable countid maka proses akan melakukan increment pada variable i, jika tidak maka proses akan melakukan inisialiasi variable Neighbor [id-2] ke variable i. Untuk source code fungsi tersebut dapat dilihat pada tabel 4.2 berikut:

Tabel 4. 2 Fungsi Subprocess Dua

Algoritma 2 : Fungsi Subprocess 2	
1	else if(msgArray[0].equals("RPL:") &&
2	
3	msgArray[1].equals("Neighbor") && msgArray[2].equals("added") &&
4	id!=1){
5	
6	if(!startTime[id-2]){
7	
8	if(msgArray[6].equals(address[0]+",")){
9	startTime[id-2]=sim.getSimulationTimeMillis();
10	
11	}else{
12	startTime[id-2]=sim.getSimulationTimeMillis();
13	
14	if(!neighbor[id-2]){
15	
16	i=0;
17	while(!msgArray[6].equals(address1[i]+",") &&
18	i<(countId)){
19	
20	i=i+1;
21	}
22	
23	neighbor[id-2]= i;
24	}
25	
26	}
27	}
28	
29	}
30	

Apabila memasuki kondisi variable msgArray index ke nol sama dengan “RPL”, msgArray index ke satu sama dengan “DAO” dan msgArray index ke dua sama dengan “lifetime :” maka akan dilakukan pengecekan pada variable count. Jika variable count sama dengan nol, maka program akan mencetak jarak antara node client dan node server, setala itu increment pada variable count. Proses selanjutnya apabila id adalah satu, maka akan melakukan inisalisasi variable i sama dengan satu, namun ketika variable msgArray tidak sama dengan variable address1 dan i kurang dari countid+1, maka variable I akan di increment.

Proses selanjutnya adalah melakukan pengencekan pada variable neighbor, jika variable neighbor [i-1] tidak sama dengan null dan variable timeClient index ke i-1 sama dengan null, maka variable recieveTime dan startTime dikurangi lalu dibagi dengan 1000 dan ditambah dengan variable timeClient. Selanjutnya apabila variable tersebut tidak berisikan null dan variable timeClient index ke i-1 sama dengan null, maka variable recieveTime dan startTime dikurangi lalu dibagi dengan 1000. Setelah itu, proses akan melakukan pengecekan jika variable jarak index i – 1 sama dengan null dan nilai i tidak sama dengan satu, akan dilakukan perhitungan dengan variable timeClient index ke 1 dikali variable jarak index ke 0 lalu mencetak hasil perhitungan tersebut dan melakukan increment pada variable count. Jika kondisi diatas tidak terpenuhi, maka akan kembali ke proses pengambilan message dari node. Untuk source code fungsi tersebut dapat dilihat pada tabel 4.3 berikut:

Tabel 4. 3 Fungsi Subprocess Tiga

Algoritma 3 : Fungsi Subprocess Tiga

```
elseif(msgArray[0].equals("RPL:")&&    msgArray[1].equals("DAO")&&
msgArray[2].equals("lifetime:")){

        if(count==0){

            log.log("ID: 2"+" jarak: "+jarak[0]+" meter\n");

            count++;

        }

        if(id==1){

            recvTime=sim.getSimulationTimeMillis();

            i=1;

            while(!msgArray[8].equals(address[i])                &&
i<(countId+1)){

                i=i+1;

            }

            if(neighbor[i-1]){

                if(!timeClient[i-1]){

timeClient[i-1]=((recvTime-startTime[i-
1])/1000)+timeClient[neighbor[i-1]];

                    log.log("neighbor ID "+(i+1)+": "+(neighbor[i-
1]+2)+" \n");

                    log.log("ID: "+(i+1)+" waktu: "+timeClient[i-
1]+" detik\n");

                }

            }else{

                if(!timeClient[i-1]){

                    timeClient[i-1]=(recvTime-startTime[i-
1])/1000;

                    log.log("ID: "+(i+1)+" waktu: "+timeClient[i-
1]+" detik\n");

                }


```

```
44              }
45
46              if(!jarak[i-1] && i!=1){
47  jarak[i-1]=parseInt((timeClient[i-1]*jarak[0])/timeClient[0]);
48
49                  log.log("ID:   "+(i+1)+"   jarak:   "+jarak[i-1]+"
50  meter\n");
51              }
52
53          }
54
55      }
56  else if(msgArray[0].equals("DATA") && timeClient[id-2]){
57
58          if(!jarak[id-2]){
59              jarak[id-2]=parseInt((timeClient[id-
60  2]*jarak[0])/timeClient[0]);
61
62              log.log("ID: "+(id)+" jarak: "+jarak[id-2]+" meter\n");
63
64          }
65      }
66
67  }
68
```

4.2.2 Implementasi *Fungsi Packet Routing Overhead*

Implementasi fungsi ini diawali dengan inisialisiasi timeout pengujian, inisialiasi waktu simulasi ke variable startTime dan inisialisasi variable recv sama dengan nol. Program utama diawali dengan perulangan dan memecah variable msgArray dengan delimeter. Jika variable msgArray index ke nol sama dengan "RPL" maka akan dilakukan increment pada variable recv. Jika waktu awal simulasi dikurangi variable startTime lebih besar dari 15000, maka akan dicetak jumlah paket RO dan inisialiasi variable recv sama dengan nol. Untuk source code fungsi tersebut dapat dilihat pada tabel 4.4 berikut:

Tabel 4. 4 Fungsi Packet Routing Overhead

```
Algortima 4 : Fungsi Packet Routing Overhead
1   TIMEOUT(200000);
2   startTime = sim.getSimulationTimeMillis();
3   recv = 0;
4   while(1){
5       YIELD();
6       msgArray = msg.split(' ');
7       if(msgArray[0].equals("RPL:")){
8           recv = recv+1;
9       }
10      if((sim.getSimulationTimeMillis()-startTime)>15000){
11          log.log("RO : "+recv+" paket\n");
12          recv=0;
13          startTime = sim.getSimulationTimeMillis();
14      }
15  }
16  log.testOK();
17
```

4.2.3 Implementasi *Fungsi Packet Delivery Ratio*

Implementasi fungsi ini diawali dengan inisialisiasi timeout pengujian, inisialiasi perhitungan dan cetak persentase PDR dan inisialisasi variable recv dan sent sama dengan nol. Program utama diawali dengan perulangan dan memecah variable msgArray dengan delimeter. Jika variable msgArray index ke nol sama dengan “DATA” variable msgArray index ke satu sama dengan “recv” maka akan dilakukan increment pada variable recv. Apabila msgArray index ke nol sama dengan “DATA” variable msgArray index ke satu sama dengan “sent” maka akan dilakukan increment pada variable sent. Persentase jumlah PDR dihitung dari variable recv dibagi variable sent lalu dibagi 100. Untuk source code fungsi tersebut dapat dilihat pada tabel 4.5:

Tabel 4. 5 Fungsi Packet Delivery Ratio

Algoritma 5 : Fungsi Packet Delivery Ratio	
1	`TIMEOUT(200000, log.log("packet receive : "+recv+"\npacket sent`
2	`:"+sent+"\nPDR: "+((recv/sent)*100)+" persen"));`
3	`sent=0;`
4	`recv=0;`
5	`while(1){`
6	`    YIELD();`
7	`    msgArr=msg.split(' ');`
8	`    if(msgArr[0].equals("DATA") && msgArr[1].equals("recv")){`
9	`        recv = recv+1;`
10	`    }else if(msgArr[0].equals("DATA") &&`
11	`msgArr[1].equals("send")){`
12	`        sent = sent+1;`
13	`    }`
14	`}`
15	`log.testOK();`
16	
17	

BAB 5 HASIL DAN ANALISIS

Bab ini berisi pembahasan mengenai hasil dan analisis data dari pengujian yang telah dilakukan. Hasil dan analisis memiliki tujuan untuk mengetahui kinerja *Routing Protocol for Low Power and Lossy Network* (RPL) pada simulator Contiki Cooja dengan menggunakan metode *Time of Arrival* (TOA). Hasil dan analisis pada bab ini akan berdasarkan pada tiga pengujian yaitu pengujian *Localization*, pengujian *Routing Overhead* dan pengujian *Packet Delivery Ratio*. Pengujian dilakukan dengan topologi random dan berdasarkan jumlah node yaitu 10, 20, 30 40 dan 50 Node, pada pengujian *localization*, hasil jarak dan waktu dikumpulkan lalu di rata-rata nilainya, sedangkan pada pengujian *Routing Overhead* dan *Packet Delivery Ratio* dihimpun datanya lalu disajikan dalam bentuk grafik.

5.1 Hasil dan Analisis Pengujian *Localization*

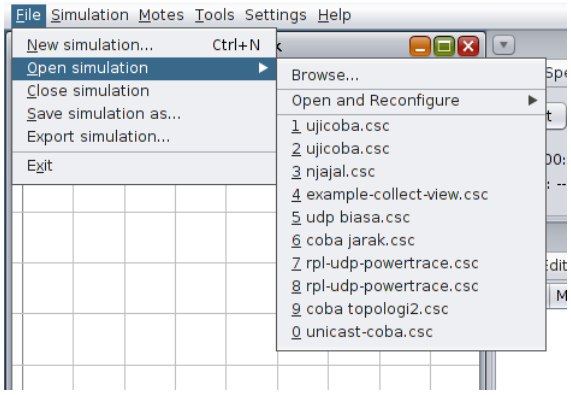
Pada bab ini, pengujian dilakukan dengan mengambil keseluruhan data pengujian *Localization* pada 10, 20, 30, 40 dan 50 Node. Sebelum memulai menjalankan proses pengujian pada localization, ada beberapa tahapan yang harus diketahui antara lain tujuan dari pengujian *Localization*, prosedur pengujian dan yang terakhir hasil pengujian *Localization*.

5.1.1 Tujuan Pengujian *Localization*

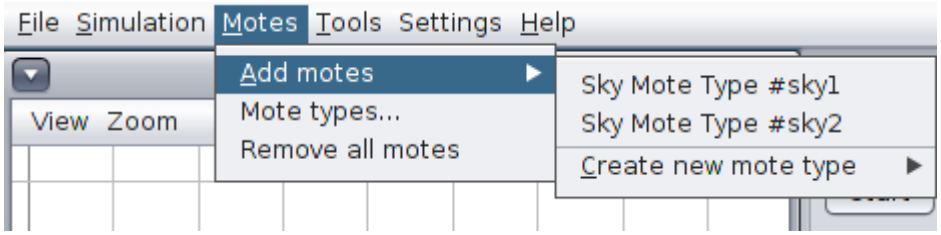
Tujuan dilakukannya pengujian localization yaitu untuk mengetahui suatu estimasi jarak suatu node antara node *client* dan node *server* yang belum diketahui yang menggunakan parameter waktu dan jarak. Semakin jauh jarak yang di dapatkan maka semakin lama juga waktu yang dibutuhkan.

5.1.2 Prosedur Pengujian *Localization*

Untuk memulai langkah awal saat menjalankan pengujian pada fungsi *localization* yaitu pertama membuka *file* pada *taskbar* kemudian pilih *open simulation* dan akan muncul beberapa pilihan *file* untuk menjalankan simulasi, seperti pada Gambar 5.1 dan 5.2 :

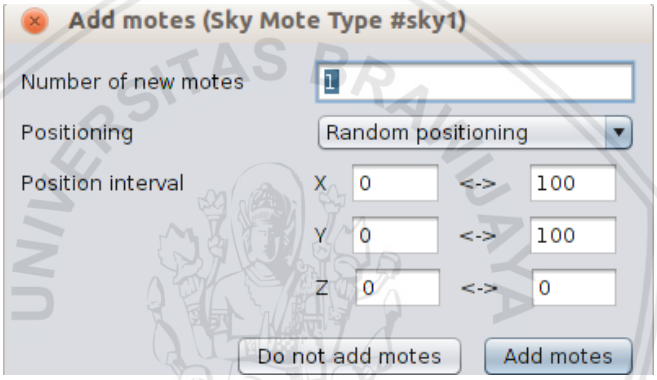


Gambar 5. 1 File Open Simulation

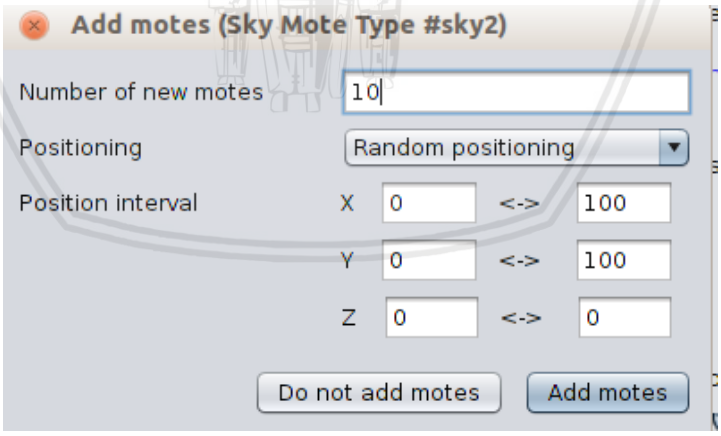


Gambar 5. 2 Pilih Motes

Sky mote type sky 1 berfungsi sebagai server dan *sky mote type sky* 2 berfungsi sebagai node *client*. Untuk langkah awal saat menentukan *mote*, *mote* yang pertama dipilih yaitu *mote* jenis *sky* 1 yang berfungsi sebagai *node server* dan *mote* jenis *sky* 2 untuk memilih *node client*. Untuk gambar pemilihan *node server* dan *client* bisa dilihat pada Gambar 5.3 dan 5.4:

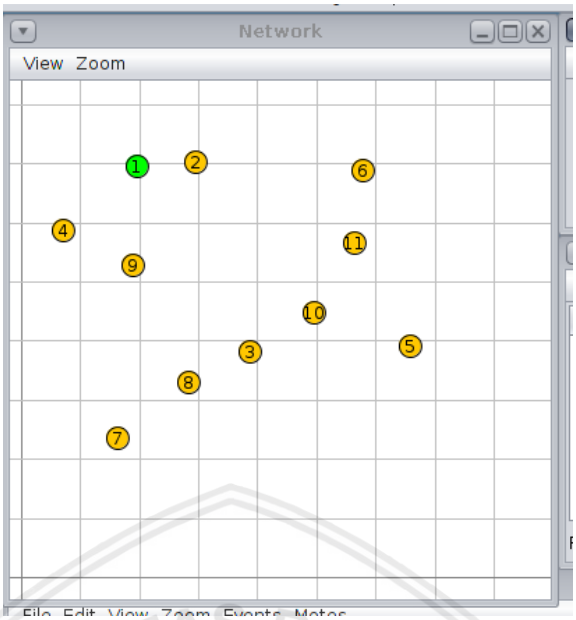


Gambar 5. 3 Memilih Mote Sky 1 Sebagai Node Server



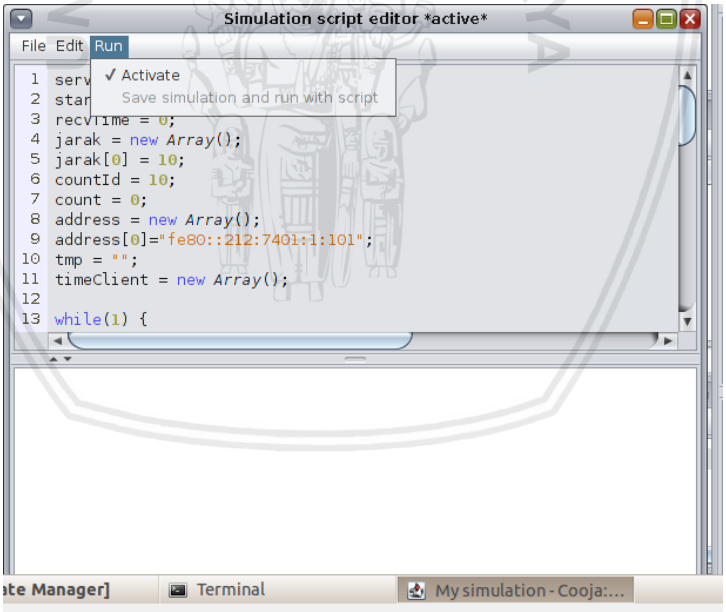
Gambar 5. 4 Memilih Mote Sky 2 Sebagai Node Client

Pada pengujian ini dibuat 1 node sebagi node server, namun pada *node client* sesuai scenario pengujian yaitu 10, 20, 30, 40 dan 50 node menyesuaikan dengan kebutuhan implementasi yang akan digunakan. Untuk membedakan antara node server dan node client bisa dilihat dari warna. Untuk node server ditandai dengan warna hijau, dan node clien ditandai dengan warna kuning. Contoh peletakan node dapat dilihat pada Gambar 5.5 :

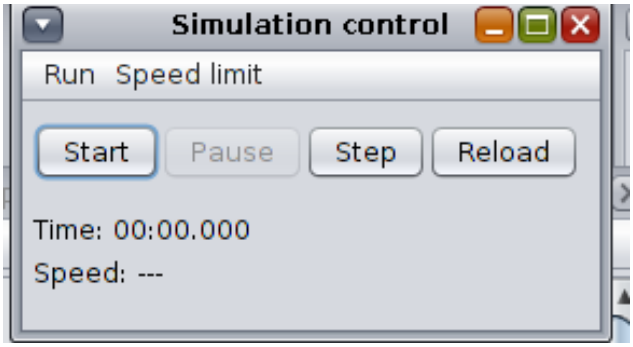


Gambar 5. 5 Keseluruhan Node Server dan Node Client

Setelah memasukan *node server* dan *node client*, langkah berikutnya untuk menguji fungsi *localization* yaitu memasukan script, menjalankan script *localization* seperti 5.6 dan menjalankan simulasi seperti Gambar 5.7:

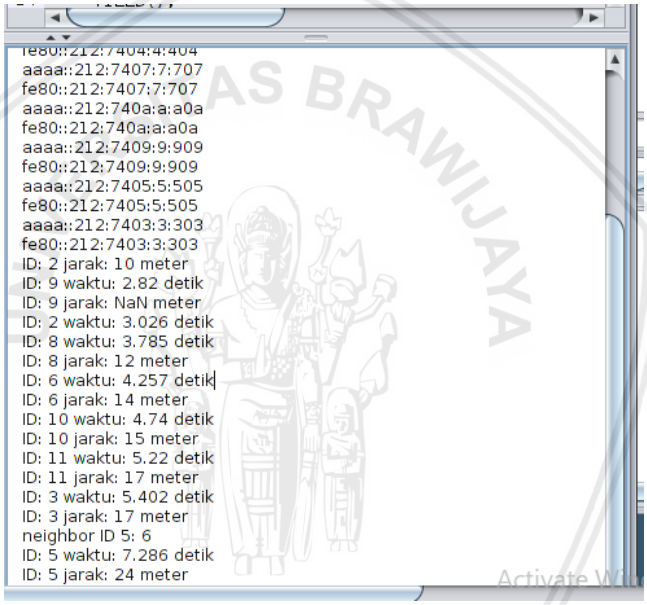


Gambar 5. 6 Script untuk memulai Localization



Gambar 5. 7 Simulation Control

Setelah mengaktifkan script fungsi localization maka pilih menu start yang ada pada simulation control sesuai dengan Gambar 5.7 untuk menjalankan script fungsi localization.

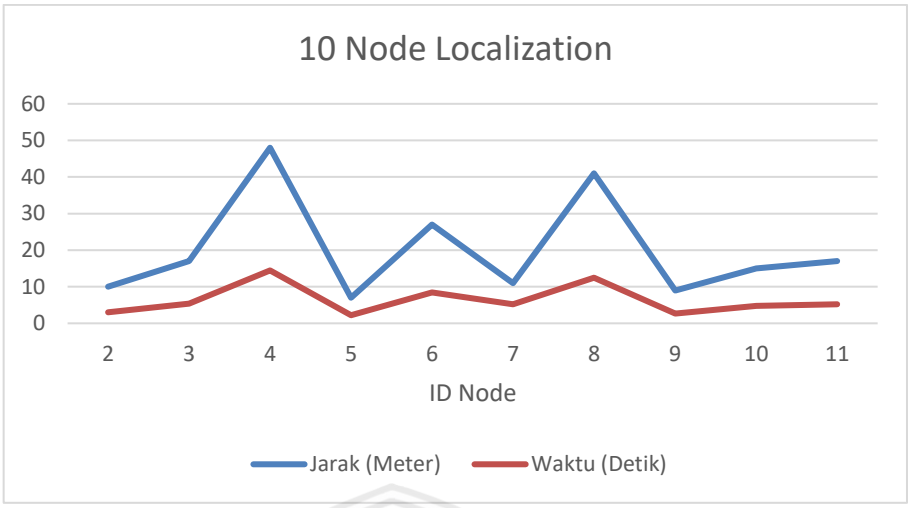


Gambar 5. 8 Output dari Script Fungsi *Localization*

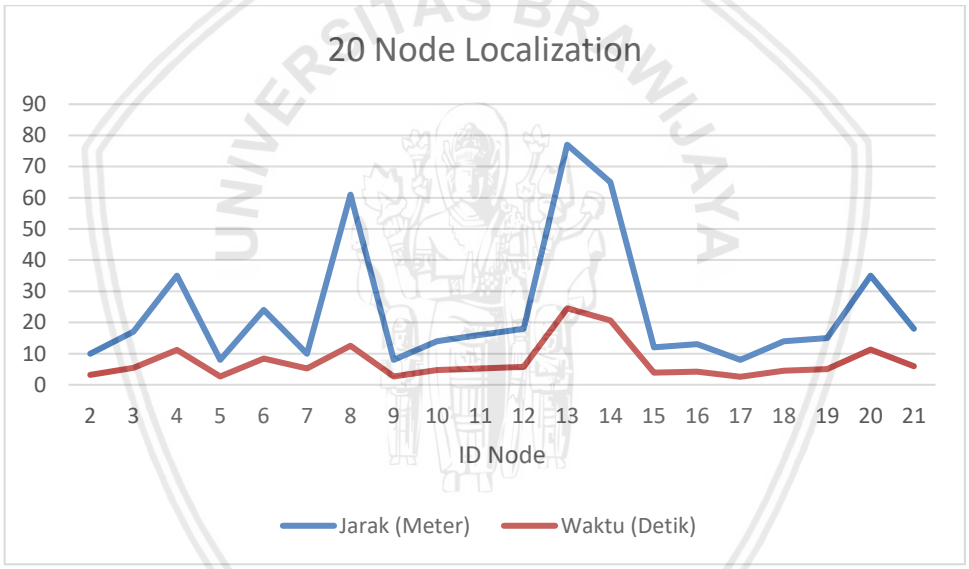
JIka sudah menjalankan beberapa langkah yang sudah dijelaskan diatas, maka Implementasi pada fungsi localization ini berhasil dijalankan dan menghasilkan output data jarak dan waktu.

5.1.3 Hasil Pengujian *Localization*

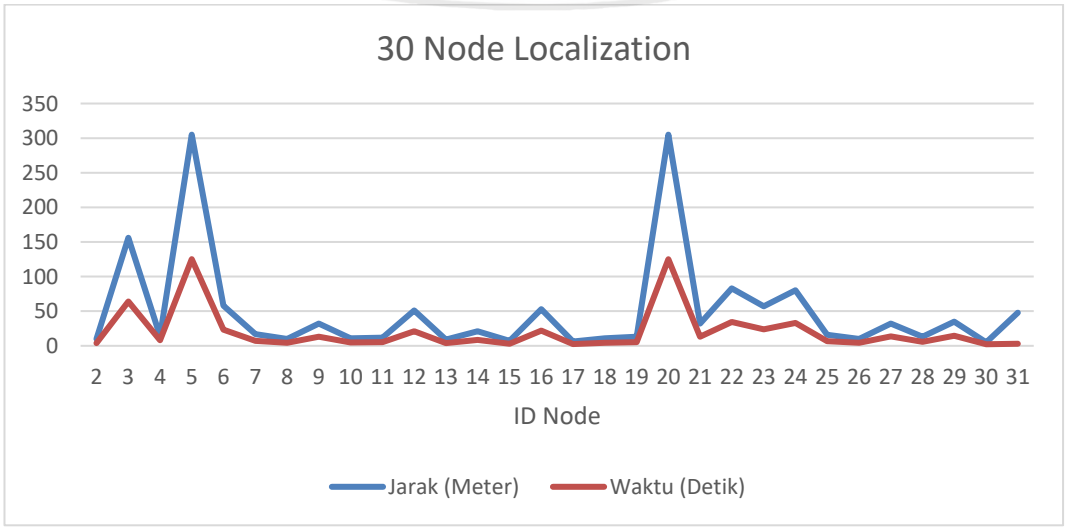
Setelah melakukan langkah – langkah pengujian yang sudah dipaparkan seperti penjelasan diatas, maka proses berikutnya adalah menjadikan hasil jarak dan waktu yang telah didapatkan menjadi kedalam bentuk grafik. Grafik nilai jarak dan waktu pada pengujian localization dapat dilihat pada Gambar 5.9 berikut :



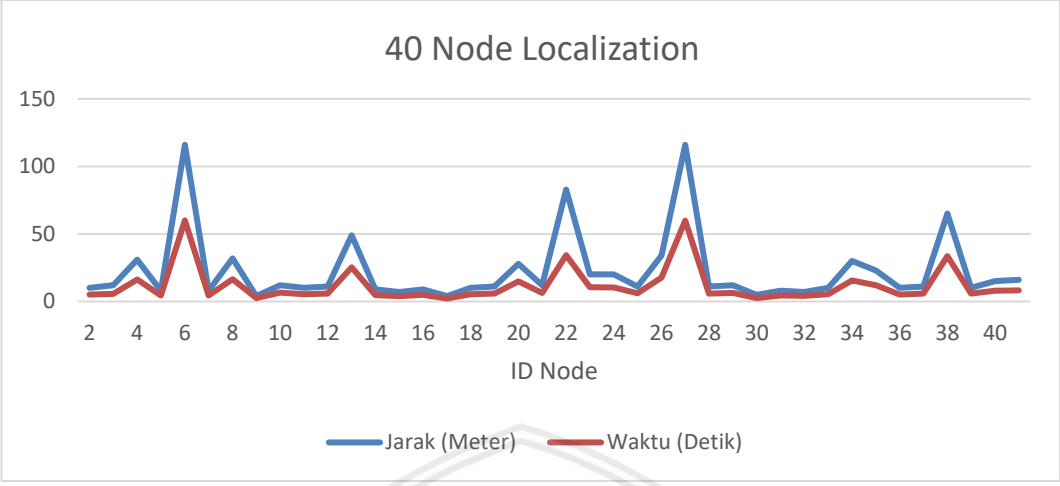
(a) Pengujian dengan 10 Node



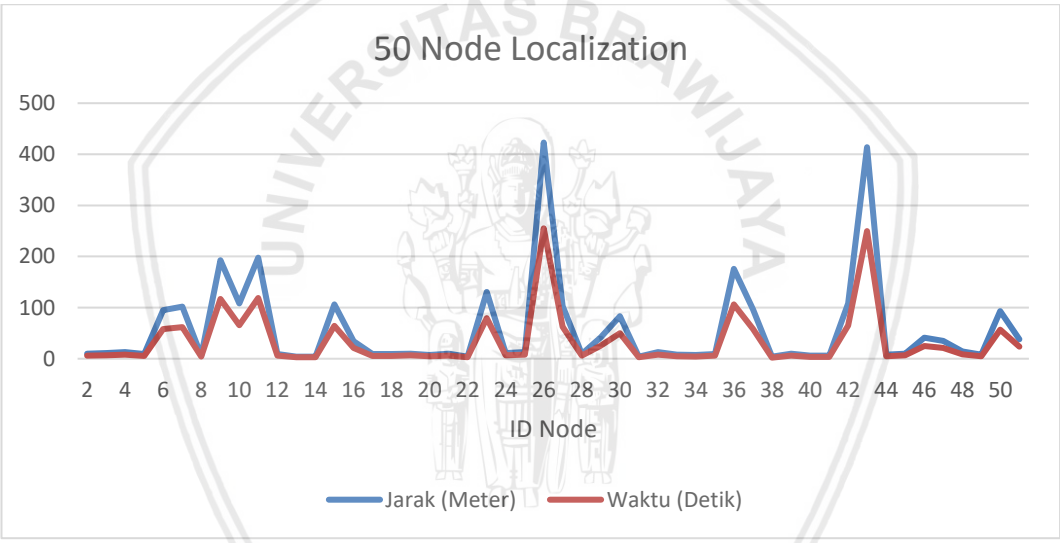
(b) Pengujian dengan 20 Node



(c) Pengujian dengan 30 Node



(d) Pengujian dengan 40 Node



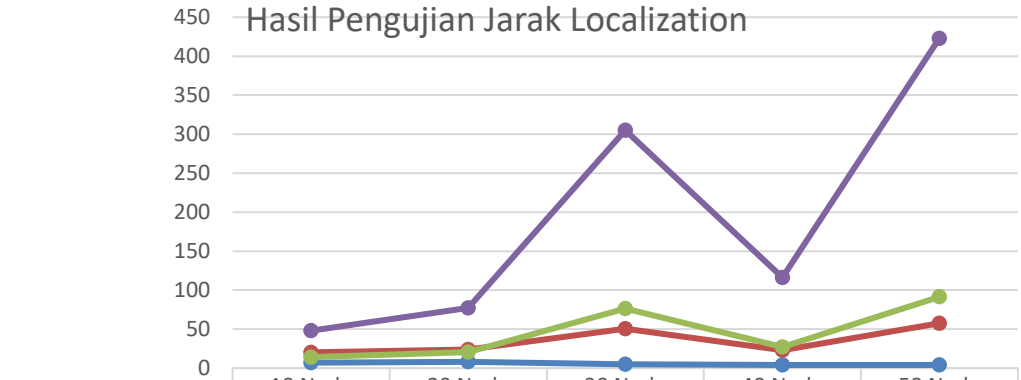
(e) Pengujian dengan 50 Node

Gambar 5. 9 Grafik Jarak dan Waktu *Localization*

Berdasarkan grafik diatas, pengujian 10, 20, 30, 40 dan 50 node memiliki trend yang sama, dimana semakin jauh jarak antara node client dan node server, maka semakin besar pula waktu localization. Hal ini sejalan dengan prinsip metode *localization Time of Arrival (TOA*) dimana jarak dihitung dari awal paket data dikirim sampai dengan paket sampai ke node tujuan.

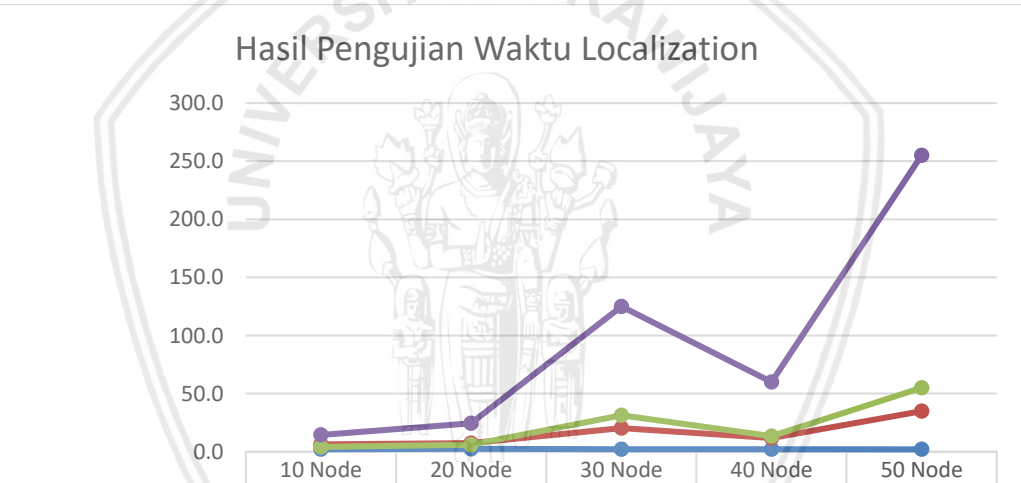
5.2 Analisis dari Pengujian *Localization*

Pada bab ini, pengujian dilakukan dengan mengambil keseluruhan data pengujian Localization pada 10, 20, 30, 40 dan 50 Node, lalu data tersebut diambil min, max, rata-rata dan standar deviasi pada waktu dan jarak locatizationnya. Hasil pengujian dapat dilihat pada grafik di Gambar 5.10 dan 5.11 berikut ini :



	10 Node	20 Node	30 Node	40 Node	50 Node
Min	7	8	5	4	4
Rata-rata	20.2	23.9	50.4	22.8	57.5
Standard deviasi	14.1	20.5	76.2	27.1	91.3
Max	48	77	305	116	423

Gambar 5. 10 Hasil Pengujian Jarak Localization



	10 Node	20 Node	30 Node	40 Node	50 Node
Min	2.2	2.6	2.3	2.2	2.1
Rata-rata	6.4	7.5	20.3	11.7	34.9
Standard deviasi	4.2	5.9	31.4	13.5	55.0
Max	14.5	24.5	125.2	60.0	255.1

Gambar 5. 11 Hasil Pengujian Jarak Localization

Dari Gambar 5.10 dan 5.11 dapat dilihat bahwa semakin trend nilai rata-rata jarak yang berhasil diukur dengan metode localization, maka semakin lama pula waktu yang dibutuhkan untuk mengetahui jarak antar node client dan node server. Pada pengujian 50 Node, nilai standard deviasinya lebih besar 62.9% untuk jarak dan 63.4% untuk waktu localizationnya. Hal ini dikarenakan nilai jarak yang berbanding lurus dengan waktu yang menandakan sebaran nilai datanya memliki varian yang tidak bervariatif dan jarak antar nilainya tidak jauh. Sedangkan pada pengujian 10, 20, 30 dan 40 Node jarak antar standar deviasi dan rata-ratanya tidak terlalu signifikan.

Pengujian *localization* ini menggunakan metode estimasi jarak *Time Of Arrival (TOA)* yang dimana pada metode ini, sebuah sensor mengukur jarak dengan sensor yang lain dengan mengestimasi delay propagasi sinyal dalam udara bebas, dimana sinyal-sinyal radio ditransmisikan dalam kecepatan cahaya yang konstan. Setelah node berhasil mengestimasi delay propagasi antara node server dan node clientnya, jarak antar node client dan server akan di estimasi berdasarkan waktu paket terkirim dikurangi waktu paket diterima.

5.3 Hasil dan Analisis Pengujian *Packet Routing Overhead*

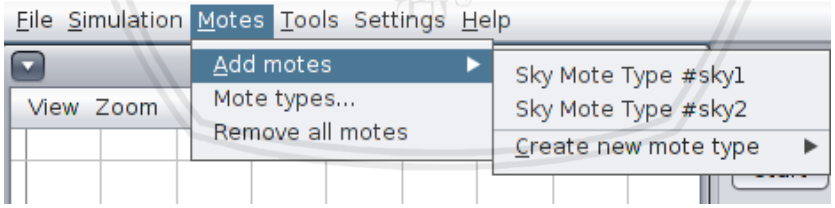
Pada bab ini akan dijelaskan hasil pengujian dari pengujian *Packet Routing Overhead* pada 10, 20, 30, 40 dan 50 Node. Sebelum memulai menjalankan proses pengujian pada *Packet Routing Overhead*, ada beberapa tahapan yang harus diketahui antara lain tujuan dari pengujian *Packet Routing Overhead*, prosedur pengujian dan yang terakhir hasil pengujian *Packet Routing Overhead*.

5.3.1 Tujuan Pengujian *Packet Routing Overhead*

Tujuan dilakukan dari pengujian packet routing overhead yaitu untuk mengetahui jumlah paket routing RPL. Semakin banyak node maka jumlah paket routing yang di dapat juga semakin banyak.

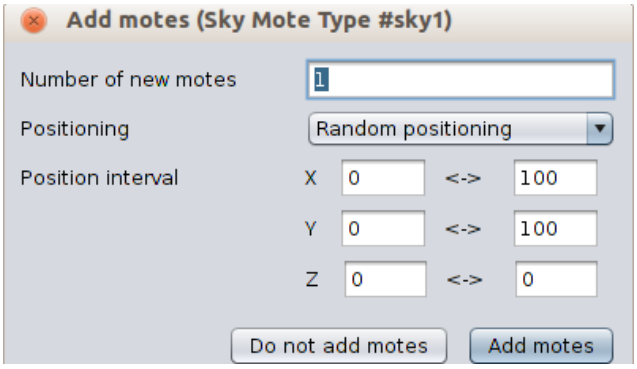
5.3.2 Prosedur Pengujian *Packet Routing Overhead*

Untuk memulai langkah awal saat menjalankan pengujian pada fungsi *packet routing overhead.* Pertama dengan membuka file kemudian pilih open simulation dan akan muncul beberapa pilihan file mana lagi yang akan dibuka, dan setelah itu memilih node apa saja yang akan digunakan. Sama dengan fungsi localization pada implementasi juga dilakukan pemilihan node server dan node client seperti Gambar 5.12:

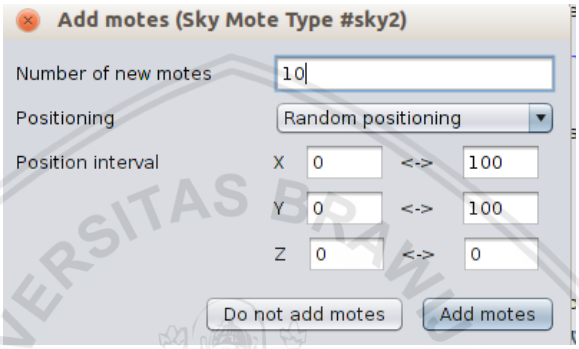


Gambar 5. 12 Pilih Mote

Sky mote type sky 1 berfungsi sebagai server dan *sky mote type sky* 2 berfungsi sebagai node *client*. Untuk langkah awal saat menentukan *mote*, *mote* yang pertama dipilih yaitu *mote* jenis *sky* 1 yang berfungsi sebagai *node server* dan *mote* jenis *sky* 2 untuk memilih *node client*. Untuk gambar pemilihan *node server* dan *client* bisa dilihat pada Gambar 5.13 dan 5.14:

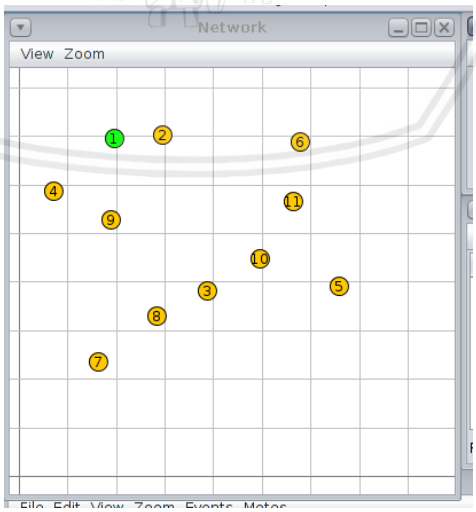


Gambar 5. 13 Pilih Mote Sky 1 untuk Server



Gambar 5. 14 Sky Mote Type 2 sebagai Node Client

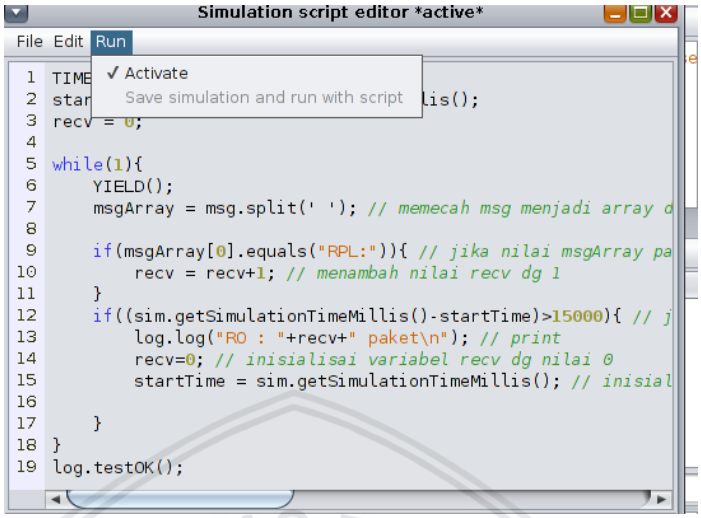
Pada pengujian ini dibuat 1 node sebagi node server, namun pada *node client* sesuai scenario pengujian yaitu 10, 20, 30, 40 dan 50 node menyesuaikan dengan kebutuhan implementasi yang akan digunakan. Untuk membedakan antara node server dan node client bisa dilihat dari warna. Untuk node server ditandai dengan warna hijau, dan node clien ditandai dengan warna kuning. Contoh peletakan node dapat dilihat pada Gambar 5.15 :



Gambar 5. 15 Keselurhan Node Server dan Node Client

Setelah memasukan *node server* dan *node client*, langkah berikutnya untuk menguji fungsi *packet routing overhead* yaitu memasukan script, menjalankan

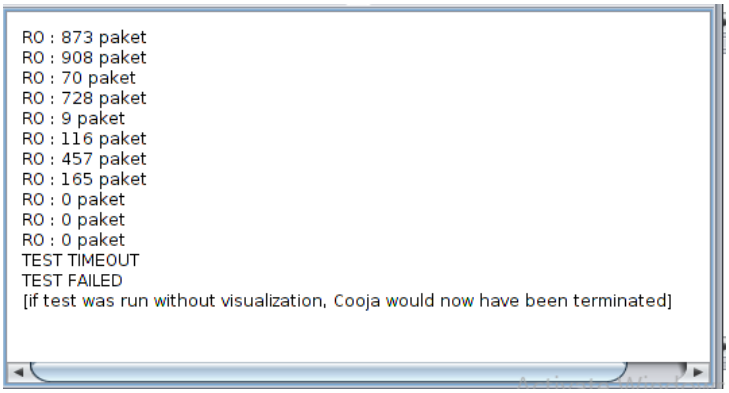
script *packet routing overhead* seperti Gambar 5.16 dan menjalankan simulasi seperti Gambar 5.17:



Gambar 5. 16 Script Fungsi dari *Routing Overhead*

Gambar 5. 17 Simulation Control

Setelah mengaktifkan script fungsi localization maka pilih menu start yang ada pada simulation control sesuai dengan gambar 5.17 untuk menjalankan script fungsi implementasi dari *packet routing overhead*. Setelah script berhasil dijalankan maka akan menghasilkan output dari fungsi packet routing overhead tersebut seperti Gambar 5.18:

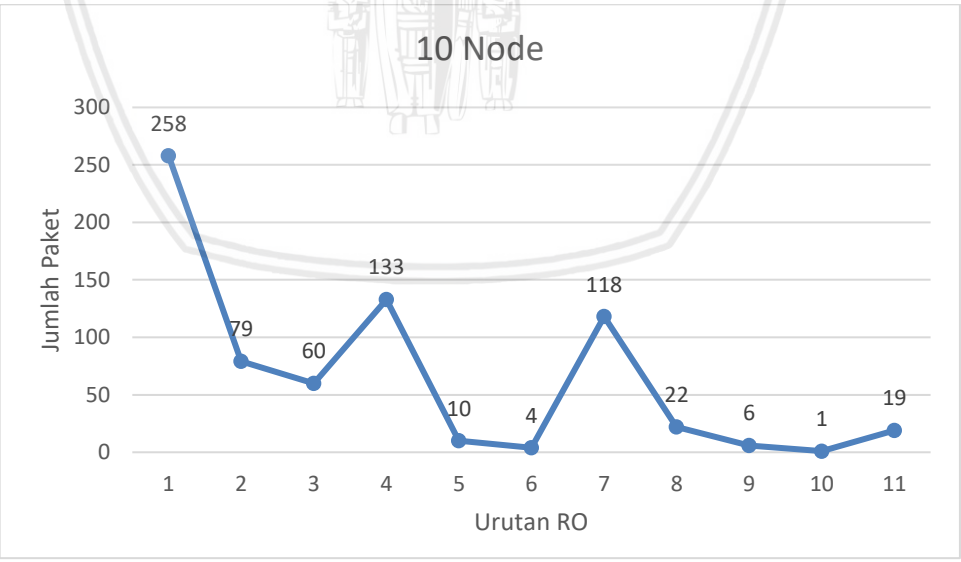


Gambar 5. 18 Output Script dari Packet Routing Overhead

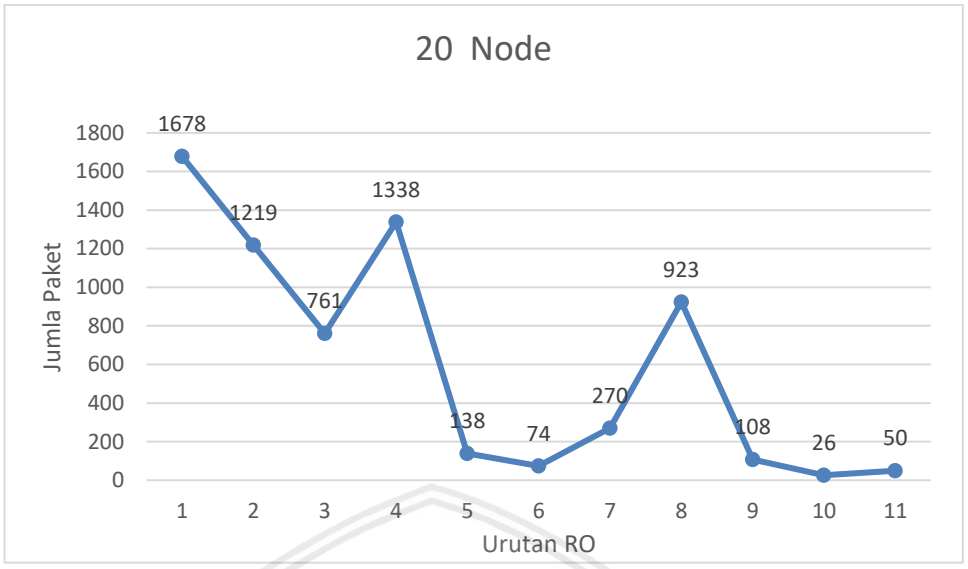
Setelah semuanya berhasil dilakukan sesuai langkah – langkah diatas sebelumnya. Maka akan keluar hasil dari script *packet routing overhead* berupa jumlah paket data dalam implementasi fungsi *packet routing overhead* sesuai pada Gambar 5.18. Jumlah *packet routing overhead* yang keluar pada script bergantung juga pada pemilihan jumlah node yang akan diuji sesuai kebetuhan yang diinginkan.

5.3.3 Hasil Pengujian *Packet Routing Overhead*

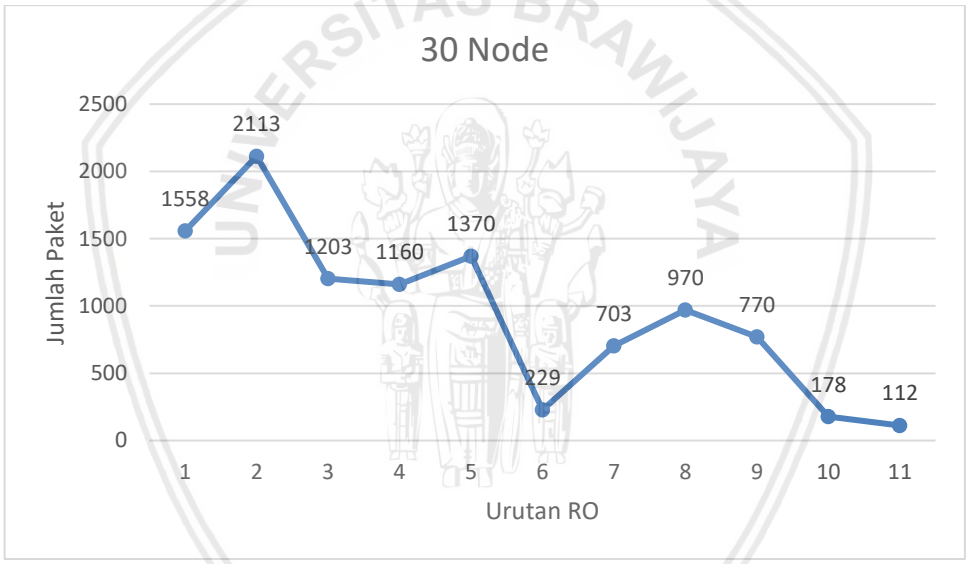
Setelah semua langkah pengujian selesai dilakukan dan sudah mendapatkan hasil berupa jumlah paket routing, lalu data tersebut diambil dan disajikan dalam bentuk grafik. Pengujian dilakukan selama 200000 ms atau 3 menit 20 detik dan data diambil per 15 detik. Gambar 5. 19 menunjukan hasil dari dari pengujian Packet Routing Overhead :



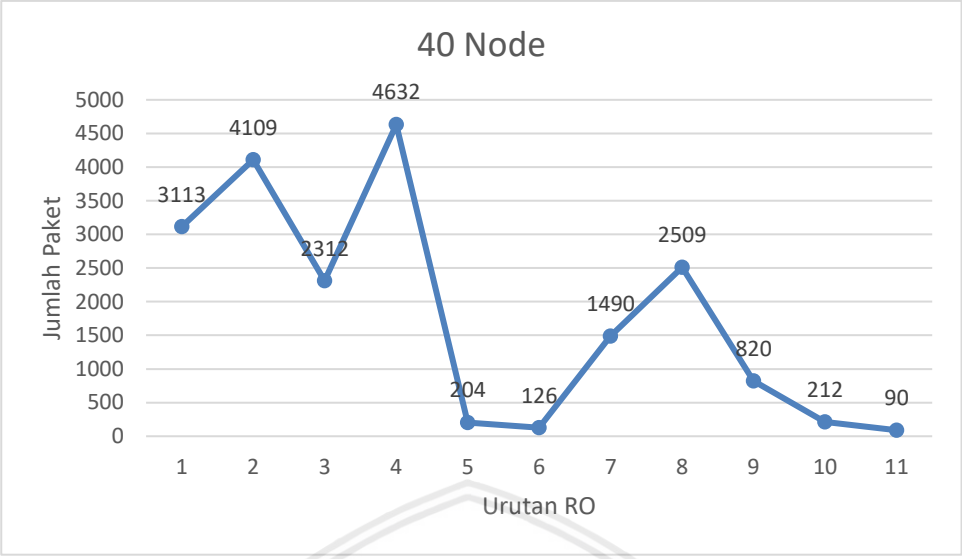
(a) Pengujian dengan 10 Node



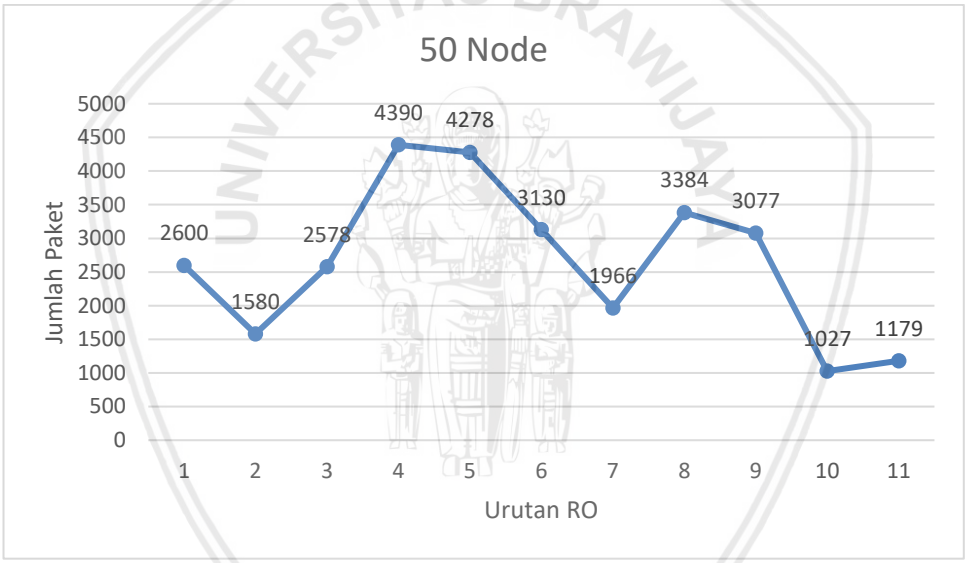
(b) Pengujian dengan 20 Node



(c) Pengujian dengan 30 Node



(d) Pengujian dengan 40 Node



(e) Pengujian dengan 50 Node

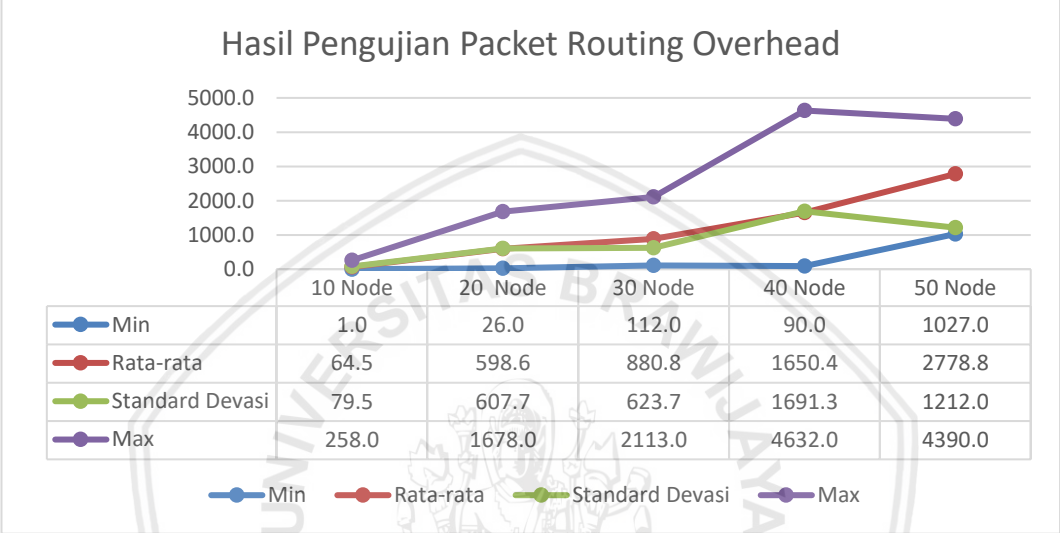
Gambar 5. 19 Grafik Pengujian *Routing Overhead*

Berdasarkan hasil pengujian pada 10 , 20, 30, 40 dan 50 Node, dapat dilihat bahwa grafik pengujian Routing Overhead memiliki kondisi yang beragam. Pada 10, 20 ,40 dan 50 Node. Dapat diamati pada urutan RO ke-4, jumlah paket RO cenderung naik drastis dari urutan sebelumnya, dan pada akhir simulasi jumlah paket RO cenderung menurun karena seluruh node yang ada pada jaringan telah berhasil mengenali nodenya satu sama lain. Pada pengujian 10, 20, 30 dan 40 Node jumlah paket RO hanya berkisar di angka puluhan yang menandakan seluruh node telah mengenali satu sama lain jadi tidak memerlukan paket routing lagi. Sedangkan pada pengujian 50 Node, terjadi anomaly data yaitu akhir pengujian masih banyak paket RO yang ditransmisikan, hal ini disebabkan semakin banyak

node pada suatu topologi semakin lama juga waktu yang dibutuhkan untuk node itu mengenali satu node dengan yang lainnya.

5.4 Analisis dari Pengujian *Packet Routing Overhead*

Pada bab ini, pengujian dilakukan dengan mengambil keseluruhan data pengujian *Packet Routing Overhead* pada 10, 20, 30, 40 dan 50 Node, lalu data tersebut diambil dan disajikan dalam bentuk grafik. Hasil pengujian dapat dilihat pada grafik di Gambar 5.20:



	10 Node	20 Node	30 Node	40 Node	50 Node
Min	1.0	26.0	112.0	90.0	1027.0
Rata-rata	64.5	598.6	880.8	1650.4	2778.8
Standard Devasi	79.5	607.7	623.7	1691.3	1212.0
Max	258.0	1678.0	2113.0	4632.0	4390.0

Gambar 5. 20 Grafik Hasil Pengujian *Routing Overhead*

Dari Gambar 5.20 dapat dilihat bahwa hasil pengujian *routing overhead* dengan menggunakan 10, 20, 30, 40 dan 50 Node. Berdasarkan grafik hasil pengujian diatas, dapat dilihat trend nilai *routing overhead* seiring dengan bertambahnya jumlah node. Routing Overhead adalah jumlah paket routing yang dijalankan pada keselurhan node. Pengujian menunjukan trend nilai *routing overhead* yang meningkat seiiring dengan bertambah jumlah node, hal ini disebabkan karena node harus mempertahankan kerja *routing* dan banyaknya rute yang diambil suatu paket yang dikirimkan dari node sumber ke node tujuan.

Hasil pada pengujian 10, 20, 30 dan 40 Node menunjukan hasil standar deviasi hanya tidak bernilai jauh dibandingkan rata-rata nya, hal ini berarti bahwa simpangan data yang baik pada pengujian 10, 20, 30, 40 Node. Namun pada pengujian 50 Node standar deviasi dan rata-rata berjarak cukup jauh dimana nilai rata-ratanya adalah 2778 sedangkan nilai standar deviasinya adalah 1212, hal ini dikarenakan nilai routing overhead yang besar dikarenakan pada routing 50 Node membutuhkan paket routing yang lebih banyak untuk mengenali satu node dengan yang lain sehingga mengakibatkan nilai routing overhead antara satu sama lain memiliki jarak yang besar. Berdasarkan hasil tersebut, terbukti bahwa dengan banyaknya jumlah node pada suatu jaringan maka akan semakin besar routing overheadnya dan membuat komunikasi antara node kurang efisien karena banyaknya rute yang diambil.

5.5 Hasil dan Pengujian *Packet Delivery Ratio*

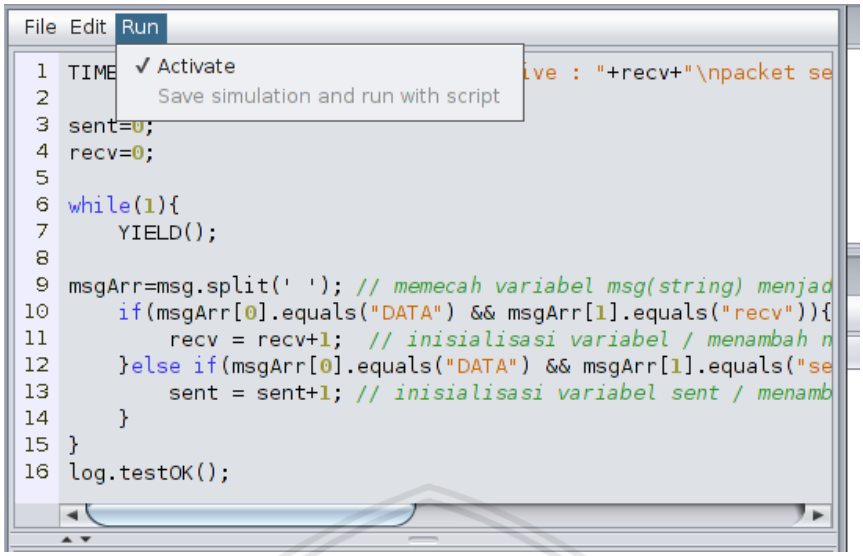
Pada bab ini, pengujian dilakukan dengan mengambil keseluruhan data pengujian Packet Delivery Ratio pada 10, 20, 30, 40 dan 50 Node. Sebelum memulai menjalankan proses pengujian pada *Packet Delivery Ratio*, ada beberapa tahapan yang harus diketahui antara lain tujuan dari pengujian Packet Delivery Ratio, prosedur pengujian dan yang terakhir hasil pengujian *Packet Delivery Ratio.* lalu data tersebut diambil dan disajikan dalam bentuk grafik.

5.5.1 Tujuan Pengujian *Packet Delivery Ratio*

Pengujian *Packet Delivery Ratio* bertujuan untuk mengetahui perbandingan antara banyaknya paket yang diterima dengan banyaknya paket yang dikirim secara keseluruhan.

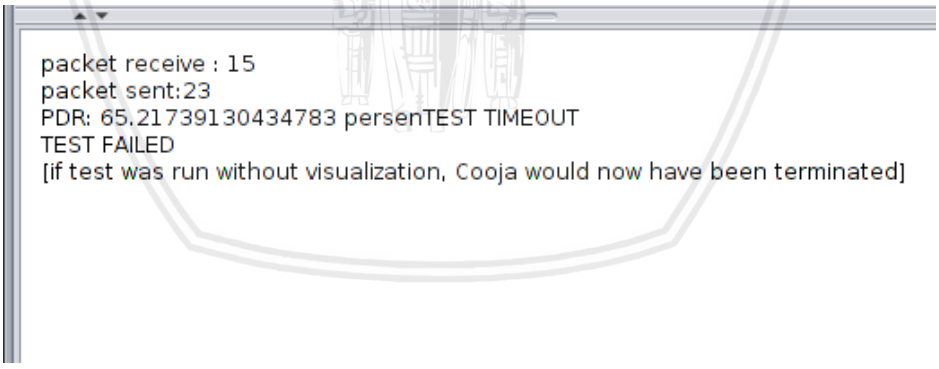
5.5.2 Prosedur Pengujian *Packet Delivery Ratio*

Fungsi *Packet Delivery Ratio* merupakah salah satu fungsi uji implementasi yang dilakukan terkahir setelah menguji implementasi fungsi *Localization* dan juga fungsi dari *Packet Routing Overhead*. Langkah – langkah yang dilakukan implementasi dari fungsi *Packet Delivery Ratio* ini sama dengan langkah sebelumnya saat melakukan implementasi dari fungsi *Localization* dan juga *Packet Routing Overhead*. Hal yang pertama dilakukan yaitu menentukan jenis node sesuai kebutuhan yaitu node server dan juga node client. Setelah itu memasukan jumlah node client sesuai kebutuhan yang akan digunakan pada implementasi fungsi dari *Packet Delivery Ratio.* Setelah memasukan node server dan juga node client seperti halnya pada *Localization* dan juga Routing *Overhead*, maka pilihlah script sesuai kebutuhan mana yang akan dijalankan. Sesuai nama implementasi diatas yaitu implementasi dalam menguji fungsi *Packet Delivery Ratio,* maka *script* yang akan digunakan yaitu script dari fungsi *Packet Delivery Ratio*. Untuk penjelasan bisa dilihat pada Gambar 5.21 :



Gambar 5. 21 Script Fungsi dari Packet Delivery Ratio

Gambar 5.21 adalah gambar script dari fungsi *packet delivery ratio*. Setelah melakukan langkah – langkah yang sudah disebutkan pada penjelasan sebelumnya, pada bagian script ini akan dijalankan pada simulator agar mengetahui hasil dari fungsi *packet delivery ratio.* Sama halnya dengan fungsi dari *localization* dan juga packet *routing overhead*, maka untuk memulai menjalankan script pilih run selanjutnya *activate* untuk mengaktifkan fungsi script diatas. Setelah script aktif pilih menu *start* pada kolom *simulation control* lalu script siap untuk dijalankan. Setelah script berjalan maka akan menghasilkan keluaran data seperti Gambar 5.22:

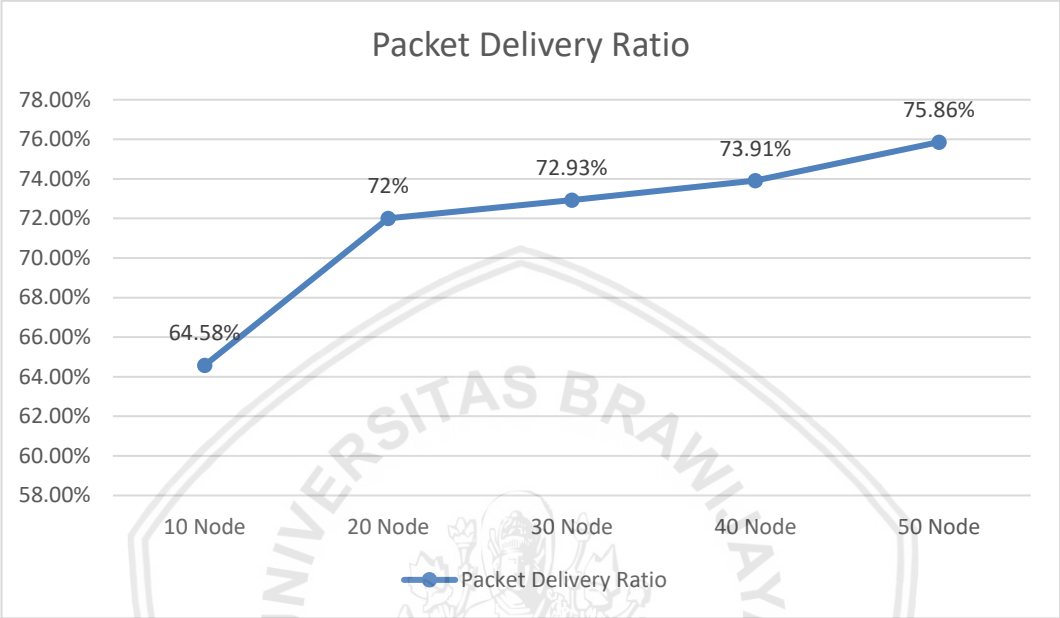


Gambar 5. 22 Output dari Script Fungsi Packet Delivery Ratio

Gmbar diatas hasil keluaran dari *fungsi packet delivery ratio.* Yang menghasilkan keberhsilan data yang diterima dari keseluruhan node. Namun data yang dikirimkan juga bisa ditampilkan pada output gambar 5.22 dan juga presentasi keberhasilan data yang diterima. Dari gambar diatas bisa dilihat bahwa prsentase *packet delivery ratio* mencapai 65% dengan jumlah paket data yang diterima 15 dan paket terkirim sebanyak 23.

5.6 Hasil dan Analisis dari Pengujian *Packet Delivery Ratio*

Pada bab ini, pengujian dilakukan dengan mengambil keseluruhan data pengujian Packet Delivery Ratio pada 10, 20, 30, 40 dan 50 Node, lalu data tersebut diambil dan disajikan dalam bentuk grafik. Hasil pengujian dapat dilihat pada grafik di Gambar 5.23:



Gambar 5. 23 Grafik Hasil Pengujian *Packet Delivery Ratio*

Dari Gambar 5.23 dapat dilihat bahwa hasil pengujian *Packet Delivery Ratio* dengan menggunakan 10, 20, 30, 40 dan 50 Node berturut-turut adalah 64.58%, 72%, 72.93%, 73.01% dan 75.56%. Dari hasil pengujian diatas, dapat dilihat bahwa semakin banyak node pada suatu jaringan, semakin besar pula *Packet Delivery Ratio*-nya yang menunjukan keberhasilan packet dikirm dan dan sampai ke tujuannya. Packet Delivery Ratio (PDR) dapat menunjukkan tingkat keberhasilan sebuah protokol routing karena semakin tinggi nilai PDR salah satunya disebabkan oleh berhasilnya sebuah protokol routing dalam melakukan pencarian dan pemeliharaan rutenya. Terbukti bahwa dari grafik diatas semakin banyak node pada suatu jaringan, semakin besar pula *Packet Delivery Ratio*-nya, hal ini disebabkan karena *link breakage* yang menyebabkan antrian yang lama pada node sehingga terjadi packet loss saat pengiriman data dan berpengaruh pada nilai packet delivery ratio yang semakin kecil.

BAB 6 PENUTUP

6.1 Kesimpulan

Kesimpulan pada penelitian "Analisis Kinerja *Routing Protocol Low Power And Lossy Network* Pada Contiki Cooja Dengan Menggunakan Metode *Time Of Arrival*" Adalah sebagai berikut :

1. Analisis Kinerja *Routing Protocol Low Power And Lossy Network* Pada Contiki Cooja Dengan Menggunakan Metode *Time Of Arrival* dapan dijalankan dengan bantuan simulator Contiki cooja dengan membangun lingkungan simulasi yang terdiri dan node client dan server serta menggunakan script editor untuk mengukur perfoma protocol RPL yang berupa *localization*, *packet routing overhead* dan *packet delivery ratio.*
2. Pengujian *localization* menggunakan metode estimasi jarak *Time Of Arrival (TOA)* yang dimana pada metode ini, sebuah sensor mengukur jarak dengan sensor yang lain dengan mengestimasi delay propagasi sinyal dalam udara bebas, dimana sinyal-sinyal radio ditransmisikan dalam kecepatan cahaya yang konstan. Hal ini dibuktikan dengan hasil pengujian dimana apabila komunikasi antara node client dan node server memakan waktu lama, maka dapat diketahui jarak antar node tersebut semakin jauh.
3. Pengujian *Routing Overhead* menunjukan trend nilai *routing overhead* yang meningkat seiiring dengan bertambah jumlah node, hal ini disebabkan karena node harus mempertahankan kerja *routing* dan banyaknya rute yang diambil suatu paket yang dikirimkan dari node sumber ke node tujuan.
4. Pengujian *Packet Delivery Ratio* menunjukan bahwa semakin banyak node pada suatu jaringan, semakin besar pula *Packet Delivery Ratio*-nya yang menunjukan keberhasilan packet dikirm dan dan sampai ke tujuannya. Hal ini disebabkan karena *link breakage* yang menyebabkan antrian yang lama pada node sehingga terjadi packet loss saat pengiriman data dan berpengaruh pada nilai packet delivery ratio yang semakin kecil.

6.2 Saran

Setelah penulis melakukan penelitian, ada beberapa saran yang dapat digunakan untuk penelitian lebih lanjut yaitu :

1. Menggunakan protocol WSN lain selain RPL seperti CoAP atau MQTT
2. Mencoba dengan lingkungan jaringan dengan node yang bergerak mobile seperti MANET atau VANET.

DAFTAR PUSTAKA

(n.d.). Retrieved from http://www.contiki-os.org/start.html

Alrajeh, N. A., Bashir, M., & Shams, B. (2013). Localization Techniques in Wireless Sensor Networks. *International Journal of Distributed Sensor Networks*.

Danuansa, R., Yulianto, F. A., & Prabowo, S. (2017). Analisis Performansi dan Simulasi Security Protocol TinySec dan LLSP pada Wireless Sensor Network . *e-Proceeding of Engineering*.

Dhamayanti, Y., & Hendrantoro, G. (2013). Analisis Perbandingan Kinerja Protokol Dynamic Source Routing dan Ad hoc On-demand Distance Vector pada Mobile Ad Hoc Network untuk Sistem Komunikasi Taktis Kapal Perang. *JURNAL ILMIAH ELITE ELEKTRO*.

Empriantomo, C. A., Kristalina, P., & Pratiarso, A. (2012). Eksplorasi Beberapa Skema Lokalisasi Range Free Pada Jaringan Sensor Nirkabel . *Jurnal Elektro PENS, I*(1).

Garg, N. (2015). *Network Simulators: A Case Study*. Haryana: University College Kurukshetra.

Hendrantoro, Y. D. (2013). Analisis Perbandingan Kinerja Protokol Dynamic Source Routing dan Ad hoc Network untuk Sistem Komunikasi Taktis Kapal Perang. *Jurnal Ilmiah Elite Elektro, 4*.

Hendrawan, I. N. (2018). Analisis Kinerja Protokol Routing RPL pada Simulator Cooja. *Jurnal Sistem dan Infomartika, 12*.

Imaduddin, M. I. (2015). Analisis Dan Modifikasi Routing Protocol RPL Untuk Mobile Wireless Sensor Network Berbasis IPv6 Pada Contiki OS. *Universitas Telkom*.

Kevin Roussel, Y.-Q. S. (2016). *Using Cooja for WSN Simulations: Some New Uses and Limits*.

Kristalina, P. (2013). Fundamental Teknik Lokalisasi pada Jaringan Sensor Nirkabel.

Kristalina, P., Wirawan, & Hendrantoro, G. (2010). Estimasi Lokasi Relatif Sensor pada Jaringan Sensor Nirkabel Menggunakan Metode Maximum Likelihood Estimation dan Cramer-Rao Bound . *Industrial Electronic Seminar*(Politeknik Elektronika Negeri Surabaya).

Prasetya, B., Renyati, Tatang, U. A., Arseno, D., & Budianto. (2008). PENENTUAN POSISI USER PADA SISTEM KOMUNIKASI SELULER DENGAN METODA TIME OF ARRIVAL (TOA) DAN TIME DIFFERENCE OF ARRIVAL (TDOA) . *Seminar Nasional Informatika 2008*.

Puspitasari, N. F. (2010). Analisis RSSI (Receive Signal Strength Indicator) Terhadap Ketinggian Perangkat Wi-Fi Di Lingkungan Indoor. *Jurnal Ilmiah Dasi*.

Putra, A., Yulianto, F. A., & Herutomo, A. (2015). Analisis Performansi dan Perbandingan Routing ProtocolOLSRdan ZRP pada Vehicular Ad Hoc Network. *e-Proceeding of Engineering, II*, 6285-6292.

Rianda, A., Munadi, R., & Negara, R. M. (2016). Analisis Performansi Routing Protocol OLSR Dan AOMDV Pada Vehicular Ad Hoc Network (VANET). *Jurnal Nasional Teknik Elektro, V*, 87.

Toscano, E., & Bello, L. L. (2012). Comparative assessments of IEEE 802.15.4/ZigBee and 6LoWPAN for. *University of Catania*.

Union, I. T. (2018). *Comparison Of Time-Difference-Of-Arrival And Angle-Of-Arrival Methods Of Signal Geolocation.* Geneva: SM Series.

Wibowo, A. P., Zuliansyah, M., & Rakhmatsyah, A. (2008). Analisis Strategi Proactive Routing Protocol Dan Reactive Routing Protocol Pada Mobile Ad Hoc Network Strategy Analysis Proactive Routing Protocol And Reactive Routing Protocol In Mobile Ad Hoc Network. *Universitas Telkom*.