TESIS - BM185407

MANAJEMEN PEMBAGIAN BEBAN PADA TRANSMISI DATA BERDASARKAN ANALISA PERFORMA WIRELESS SENSOR NETWORK MENGGUNAKAN ALGORITMA MESH ROUTING

AHMAD YUSUF ARDIANSYAH

Dosen Pembimbing:
Prof. Drs.Ec. Ir. Riyanarto Sarno, M.Sc.,Ph.D

Departemen Manajemen Teknologi
Fakultas Bisnis Dan Manajemen Teknologi
Institut Teknologi Sepuluh Nopember
2019

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN TESIS

Tesis disusun untuk memenuhi salah satu syarat memperoleh gelar

Magister Manajemen Teknologi (M.MT)

di

Institut Teknologi Sepuluh Nopember

Oleh:

AHMAD YUSUF ARDIANSYAH

NRP: 09211750053029

Tanggal Ujian: 1 Juli 2019

Periode Wisuda: September 2019

Disetujui oleh:

Pembimbing:

1. **Prof. Drs. Ec. Ir. Riyanarto Sarno, M.Sc., Ph.D.**
   NIP: 19590803 198601 1 001

.....

Penguji:

1. **Dr. tech. Ir. R. V. Hari Ginardi, M.Sc.**
   NIP: 196505181992031003

.....

2. **Faizal Mahananto, S.Kom, M.Eng., Ph.D.**
   NIP: 5200201301010

.....

Kepala Departemen Manajemen Teknologi
Fakultas Bisnis dan Manajemen Teknologi

Prof. Ir. I Nyoman Pujawan, M.Eng, Ph.D, CSCP
NIP: 196912311994121076

[Halaman ini sengaja dikosongkan]

MANAJEMEN PEMBAGIAN BEBAN PADA TRANSMISI DATA BERDASARKAN ANALISA PERFORMA WSN MENGGUNAKAN ALGORITMA MESH ROUTING

Nama : Ahmad Yusuf Ardiansyah

NRP : 09211750053029

Pembimbing : Prof. Drs.Ec. Ir. Riyanarto Sarno, M.Sc., Ph.D.

ABSTRAK

Pada Umumnya, penelitian dibidang *WSN (Wireless Sensor Network)* tidak pernah membahas tentang aspek *Reliablity* pada routing jaringan dengan perangkat router yang dapat menemukan rute baru saat terjadi kerusakan. Sejauh ini, saat router bebannya berlebihan maka hanya diabiarkan saja tanpa ada respon yang memberikan kontrol, sehingga dapat menurunkan kualitas performa jaringan. Oleh karena itu, kita mengajukan dua solusi utama yang dapat memberikan kontribusi pada bidang *Computer Science* dan khususnya pada *Sensor Network.*

Dengan usulan yang diajukan ini, peneliti menggunakan *algoritma AODV routing,* dan *Mesh routing* untuk mencari alternatif rute lain saat terjadi masalah, untuk memberikan kontrol keseimbangan beban yang dijalankan pada Router. Kontrol keseimbangan ini mengatur jumlah beban yang diterima oleh Router dengan membagi beban secara bergiliran dan berurutan dari satu router ke router lainnya. Distribusi paket data dapat dikatakan berkualitas, jika data yang dikirimkan memiliki standar kualitas service atau yang disebut *Quality of Service* (QoS)

Pengujian system dibagi dua sekenario, pertama melakukan uji coba performa jaringan tanpa menerapkan balancing, dan kedua melakuan uji coba performa jaringan dengan menerapkan balancing. Dari kedua percobaan tersebut digunakan parameter QoS sebagai jaminan performa agar lebih efektif dan sesuai harapan. Pengukuran dilakukan dengan menguji parameter packet loss, *delay,* dan *throughput* mengirimakan 100 paket sejumlah karakter 235 Bytes dari titik end device ke Coordiantor melalui Router berdasarkan variasi jarak dari 0 – 100 meter.

Hasil percobaan membuktikan bahwa performa jaringan dalam menemukan alternatif rute baru telah berhasil dilakukan. Time Recovery pada saat router mati dan berhasil menemukan rute lain hanya menunggu 8,75 detik, hal ini berkaitan dengan parameter *delay*, dan *fault tolerance.* Sedangkan pada hasil penelitian penggunaan rule non-balancing system mampu memberikan hasil pengukuran delay sebesar 9,3 detik, dan paket yang loss rata-rata sebesar 37% dalam sekali pengiriman, sedangkan hasil pengujian performa jaringan yang menerapkan system balancing mampu memberikan delay lebih kecil yaitu 6,8 detik, dan packet loss yang dihasilkan rata-rata lebih kecil 20%. Artinya system WSN dengan menerapakan balancing mampu meningkatkan performa QoS sebesar 12 % dan terbukti lebih efektif, dalam perhitungannya angka 17 % ini ditunjangan dari dua

aspek pengukuran berdasarkan QoS *packet loss* dan *throughput*, hal ini membuktikan bahwa *system dynamic routing* berhasil meningkatkan kualitas reliability jaringan WSN.

Kata Kunci*: *wireless sensor, routing, QoS, balancing, monitoring, controlling, mesh, routing.

MANAGEMENT OF LOUD BALANCING FOR DATA TRANSMISSION BASED ON PERFORMANCE ANALYSIS OF WIRELESS SENSOR NETWORK USING MESH ROUTING ALGORITHM

Nama	: Ahmad Yusuf Ardiansyah
NRP	: 09211750053029
Pembimbing	: Prof. Drs.Ec. Ir. Riyanarto Sarno, M.Sc.,Ph.D.

ABSTRACT

In general, research in the field of Wireless Sensor Network (WSN) has never discussed the reliability aspect of network routing with router devices that can find new routes when damage occurs. To date, overloaded routers will be ignored without any response that gives control which can reduce the quality of network performance. Therefore, we propose research using the AODV routing, and Mesh routing algorithm to find other routes as an alternative when problems occur and using the Round Robbin based XBee algorithm on providing load balance control carried out by the router.

With this proposed thesis, researchers used the AODV routing algorithm, and Mesh routing to look for alternative other routes when problems occur. This balance control regulates the amount of load received by the Router by dividing the load in turns and sequentially from one router to another. Data package distribution can be said to be of quality, if the data sent has a service quality standard or called Quality of Service (QoS).

The system test is divided into two scenarios, first testing network performance without applying balancing, and second, testing network performance by applying balancing. From the two experiments, QoS parameters were used as collateral for performance to be more effective and as expected. Measurements were made by testing the parameters of packet loss, delay, throughput, RSSI (Receive Signal Strength Indicator), and fault Tolerance by sending 100 packets of a number of characters 235 bytes from end device to Coordinator via Router based on distance variations from 0 to 100 meters.

The experiments on the performance of non-balancing networks and balancing were conducted. Both trials used Quality of Service (QoS) parameters as a guarantee of performance to be more effective and in line with expectations. Measurements performed by testing the parameters of packet loss, delay, throughput, Receive Signal Strength Indicator (RSSI) and fault tolerance.

The experimental results prove network performance in finding alternative new routes that have been successfully carried out. Time Recovery when the router is

off and has managed to find another route that only waits for 8.75 seconds, this is related to the parameter delay, and fault tolerance. While the results of the study using non-balancing system rules are able to provide measurement results of 9.3 seconds delay, and packages that lose an average of 37% in one shipment, while the results of network performance testing using a balancing system provide a smaller delay such as 6, 8 seconds, and the resulting packet loss is on average 20%. This means that the WSN system applies balancing that can improve QoS performance by 12% and proves to be more effective, in its calculation of 17% this is supported by two aspects calculated based on QoS packet loss and throughput, this proves a dynamic routing system that increases reliability of WSN.

Keywords: ***wireless sensor, routing, qos, balancing, monitoring, controlling, mesh, routing.***

KATA PENGANTAR

Penulis mengucapkan rasa syukur yang tak berhingga kepada Allah SWT atas segala rahmat, berkah, hidayah, kesehatan dan petunjuk-Nya, sehingga penulis dapat menyelesaikan tesis yang merupakan salah satu syarat dalam menyelesaikan Program Studi Magister di Institut Teknologi Sepuluh Nopember Surabaya.

Terselesaikannya tesis beserta laporannya ini tentunya tak luput dari peran serta berbagai pihak yang telah memberikan bantuan dan dorongan semangat, baik secara langsung maupun tak langsung. Untuk itu, atas segala bantuan yang telah diberikan, penulis mengucapkan terima kasih serta penghargaan yang sebesar-besarnya antara lain kepada:

1. Bapak Dr. tech. Ir. R. V. Hari Ginardi, M. Sc. selaku dosen wali yang senantiasa memberikan bimbingan, saran, dan motivasi selama perkuliahan S2 kepada penulis.
2. Bapak Prof. Drs.Ec. Ir. Riyanarto Sarno, M.Sc., Ph.D selaku dosen pembimbing yang telah banyak meluangkan waktu, tenaga dan pikiran dalam membimbing penulis sehingga tesis ini dapat terselesaikan dengan baik.
3. Bapak Dr. tech. Ir. R. V. Hari Ginardi, M. Sc. dan Bapak Faizal Mahamamto, S.Kom., M.Eng., Ph.D. selaku dosen penguji yang telah banyak membantu penulis untuk bisa menjadi lebih baik.
4. Seluruh dosen S2 Manajemen Teknologi MMT ITS yang telah memberikan ilmu dan pengetahuan kepada penulis selama menempuh studi.
5. Kedua Orang Tua penulis yaitu bapak Edi Suyanto dan Ibu Umiyati yang senantiasa memberikan motivasi, semangat, dukungan moril dan harapan serta mendoakan penulis demi keberhasilan penulis dalam menyelesaikan studi.
6. Dewi Ayu KK, Afrianda Cahya, M. Jupri, Reza Amaliyah P dan Dike Bayu selaku teman-teman yang telah disatukan dalam ikatan takdir untuk menyelesaikan thesis bersama dengan pembimbing Bapak Prof. Drs.Ec. Ir. Riyanarto Sarno, M.Sc., Ph.D yang senantiasa memberikan motivasi, semangat, nasehat dan perhatian kepada penulis dalam menyelesaikan studi dan tesisnya.
7. Terimakasih secara khusus disampaikan penulis kepada Oxsy Giandy, Mas Odik, dan semua rekan mahasiswa S2 Manajemen Teknologi Informasi

angkatan 2017 selaku rekan seperjuangan yang telah memberikan bantuannya baik secara langsung maupun tidak langsung.

8. Terima kasih dan salam *good job* buat rekan kami, Nouval Z Kurniawan dan Muhammad Syafii karena telah membantu dalam proses pengembangan penelitian ini, dan pengambilan data pengujian yang tidak mungkin saya lakukan sendiri.
9. Terima kasih kepada Calon istriku Tanzilur Rizki Awwaliyah yang senantiasa memberikan dukungan, dan menemani dikala senang dan duka hingga penulis berhasil dipertemukan di penghujung perjuangan meniti studi S2 di MMT ITS Surabaya.

Semoga kebaikan dan bantuan yang telah diberikan kepada penulis dibalas dengan kebaikan yang lebih oleh Allah SWT. Aamin.

Tiada gading yang tak retak begitulah pribahasa yang sesuai dengan kesadaran penulis dalam pembuatan laporan tesis ini masih banyak kekurangan. Karena itu, masukan ataupun saran demi perbaikan dan penerapan tesis ini di masa mendatang tetap penulis harapkan.

Surabaya, 15 Mei 2019

Penulis

Ahmad Yusuf Ardiansyah

DAFTAR ISI

DAFTAR TABLE

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

[Halaman ini sengaja dikosongkan]

KOSA KATA PENELITIAN

NO	KOSA KATA	ARTI
1.	Load Balancing	Teknik untuk mendistribusikan beban trafik pada dua atau lebih jalur koneksi secara seimbang
2.	Non-Balancing	Suatu kondisi tidak adanya suatu pembagian beban trafik pada dua jalur atau koneksi secara simbang.
3.	QoS	Alat evaluasi kinerja distribusi antar jaringan dan merupakan sekumpulan parameter yang menunjukkan kualitas layanan suatu jaringan
4.	Routing	Proses dimana suatu item dapat sampai ke tujuan dari satu lokasi ke lokasi lain
5.	Wireless Sensor Network	Jaringan Sensor Nirkabel atau dalam banyak literatur disebut Wireless Sensor Network (WSN) adalah sebuah jaringan yang menghubungkan perangkat-perangkat seperti sensor node, router dan sink node
6.	Mesh Routing	Teknik dalam merutekan suatu jaringan menggunakan model mesh atau yang dikenal
7.	Monitoring	Aktifitas yang ditujukan untuk memberikan informasi dan memantau suatu kondisi secara nyata.
8.	Controlling	Pengendalian suatu usaha yang terdiri dari melihat bahwa segala sesuatu yang sedang dilakukan sesuai dengan rencana yang telah diadopsi, perintah yang telah diberikan, dan prinsip-prinsip yang telah ditetapkan
9.	Payload Data	Sebuah paket berisi packet header yang berisi informasi mengenai protokol tersebut (informasi mengenai jenis, sumber, tujuan, atau informasi lainnya), data yang hendak ditransmisikan
10.	Handshaking	Handshaking adalah sesi komunikasi data yang berlangsung dari mulai perencanaan komunikasi sampai dengan komunikasi tersebut selesai

11.	DH, DL	No ID perangkat Xbee dengan alamat 64-bit dan alamat 16-bit
12.	Reliability	Keandalan, adalah konsistensi dari serangkaian pengukuran atau serangkaian alat ukur
13.	Efektifitas	Suatu ukuran yang menyatakan seberapa jauh target (kuantitas, kualitas dan waktu) telah tercapai. Dimana semakin besar presentase target yang dicapai, makin tinggi
14	Digi Inc.	Perusahaan Produksi modul Xbee di Amerika
14.	Throughput	Menunjukkan besarnya paket data yang diterima pada node tujuan dibandingkan dengan waktu tempuh yang ditulis dalam satuan *bit per second (bps).*
15.	Delay	Delay adalah total waktu tunda paket yang disebabkan oleh proses transmisi dari satu titik ke titik lainnya yang mengalami keterlambatan
16.	Packet Loss	Packet Loss adalah kegagalan transmisi paket ke tujuannya. Kehilangan paket terjadi ketika satu atau lebih paket data yang melewati jaringan gagal mencapai tujuannya
17.	RSSI	RSSI adalah salah satu parameter pendukung dari QoS yang menunjukkan kekuatan sinyal yang diterima oleh perangkat dalam satuan *-dBm*
18.	Fault Tolerance	Toleransi kesalahan dalam WSN adalah kemampuan jaringan untuk mentoleransi kesalahan yang menyebabkan kegagalan layanan
19.	Skenario Penelitian	Rencana penelitian dengan memberikan ilustrasi perancanaan dari aspek waktu, kinerja, dan hasil luaran.
20.	Zigbee Protokol	Zigbee adalah sebuah protokol IEEE 802.15.4 MAC dan Layer PHY yang dirancang untuk skala yang minimalis meliputi, low bandwith, low cost, low-power, dan standar jaringan wireless

21.	XCTU	*Software* untuk melakukan konfigurasi pada produk RF keluaran Digi salah satunya XBee-pro
22.	AODV	AODV adalah *routing protocol* yang dirancang untuk jaringan *ad hoc mobile.* Teknik routing pada AODV memudahkan dalam menemukan sebuah *route*
23.	Route Discovery	*Routing discovery* adalah kemampuan routing protokol untuk membagi informasi tentang jaringan dengan node lainnya dengan menggunakan routing protokol yang digunakan
24.	Router Maintenance	*Route maintenance* adalah kondisi system melakukan perbaikan rute yang mengalami kerusakan
24.	Buffer Packer Data	Penyimpanan data pada mikrokontroller Arduino
25.	Peer to Peer	Jaringan peer-to-peer (P2P) merupakan salah satu model jaringan yang terdiri dari dua atau beberapa titik jaringan
26.	Peer to Multi Peer	Jaringan yang terhubung pada pada satu titik dan dapat terhubung banyak titik secara bersamaan.
27.	SelfHealing	Kemampuan yang dapat menyembuhkan diri sendiri, atau dapat merekondisi suatu kerusakan pada
28.	End Device	End Device adalah salah satu pilihan mode pada modul xbee yang harus selalu berinteraksi atau terhubung dengan coordinator atau router untuk dapat menerima dan mengirimkan data.
29.	Router	Router adalah mode perangkat pada modul xbee yang bertugas untuk menerima, mengirimkan dan merutekan data. Agar sebuah router dapat mengijikan router lain dan end device untuk bergabung, maka router tersebut harus terlebih dahulu bergabung dengan jaringan PAN
30.	Coordinator	Coordinator mode perangkat pada modul xbee yang bertanggung jawab untuk membangun operating channel dan PAN (Personal Area Network) ID pada sebuah jaringan

31.	Addresing	Konfigurasi pengalamatan dari suatu modul jaringan
32.	Komunikasi Serial	Model pengiriman data yang dilakukan secara urut mulai dari bit ke 1 hingga bit ke 8.
33.	Arduino	Board Mikrokontroller untuk melakukan proses sistem
34.	DHT 22	Sensor Suhu
35.	Unicast	Transmisi unicast adalah pengiriman data dari satu perangkat ke perangkat sumber tujuan lain
36.	Broadcast	Mode transmisi broadcast dalam protokol XBee dimaksudkan untuk menyebarkan paket data ke seluruh node dalam satu jaringan PAN sehingga semua node menerima data broadcast yang dikirimkan

BAB I
PENDAHULUAN

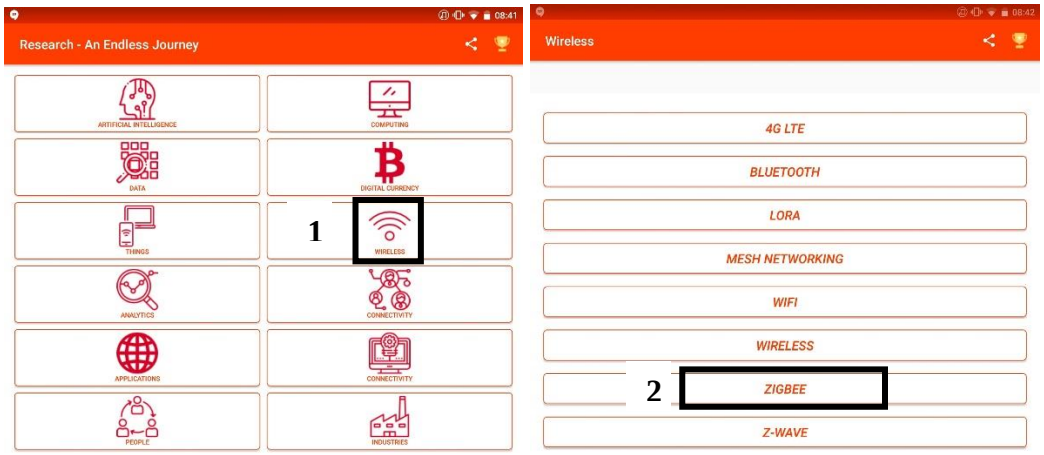
Pada bab ini akan dijelaskan beberapa hal dasar dalam pembuatan penelitian tesis yang meliputi: latar belakang, perumusan masalah, tujuan, manfaat, kontribusi penelitian, batasan masalah, dan sistematika penulisan.

1.1. Latar Belakang

Saat ini perkembangan teknologi yang semakin pesat mendorong berkembangnya perangkat elektronik, dan telekomunikasi yang berbasis nirkabel atau yang biasanya dikenal dengan WSN (*Wireless Sensor Network*) dapat diimplementasikan di berbagai bidang yang membutuhkan sitem monitoring, deteksi dan pengolahan informasi yang aktual. Teknologi nirkabel adalah teknologi yang tidak menggunakan kabel dalam proses pengiriman maupun penerimaan datanya. WSN (*Wireless Sensor Network*) adalah insfrastruktur jaringan sensor yang terhubung secara nirkabel. Sensor tersebut berfungsi untuk merasakan atau menangkap adanya perubahan di sekitar yang hasilnya dilaporkan *base station* (Suryanto and Zahra, *2015).* Pada implementasinya WSN dapat digunakan untuk pemantauan jembatan, pemantauan lingkungan, komponen pendukung dalam *smart city*, dan pemantauan deteksi kebakaran (Soijoyo and Ashari, 2017). Dalam tesis ini, kami mengajukan *Prototype* manajemen Pembagian beban yang digunakan pada transmisi data menggunakan perangkat wireless.

Pada *Wireless Sensor Network*, efisiensi *routing* protokol memainkan peran penting untuk transmisi paket data (Guo and Tang, 2010). Terbatasnya sumber daya WSN seperti menjadi salah satu tantangan dalam merancang penelitian yang memberikan kontribusi yang tepat. Semua teknologi yang menggunakan jaringan *wireless* memiliki keterbatasan antara lain, supplai energi, kapasitas penyimpanan yang terbatas, routing yang tidak *reliable* dan kurangnya pengawasan pada distribusi yang dapat mempengaruhi kualitas data yang di informasikan (Al-Karaki and Kamal, 2004).

Berdasarkan penelitian sebelumnya, maka penulis mengajukan penelitian dengan kebaruan yang fokus pada distribusi kualitas data yang memanfaatkan manajemen jaringan dengan memberikan kontribusi yang memiliki aspek, *monitoring*, *routing*, dan *controlling*, data sensor dengan memanfaatkan beberapa perangkat diantaranya end device (*transmitter*) sebagai titik pemasangan sensor, *router* sebagai jembatan penghubung antara end device dengan coordinator serta pengujian kinerja dari *load balancing* dengan *non-balancing* yang dilakukan berdasarkan pada parameter *Quality of Service* (QoS).



Gambar 1. 1 Area Teknologi Wireless

Gambar 1.1 menunjukkan sebuah perkembangan dari teknologi bidang *wireless* dimana penulis, ikut serta dalam mengembangkan penelitian pada lingkup Zigbee. Penggunaan teknologi ZigBee melalui IEEE 802.15.4 merupakan teknologi protokol komunikasi *wireless* yang memiliki keunggulan diantaranya konsumsi daya rendah, murah, memiliki *fault tolerance* yang tinggi (Hasta, Rulliyanto, 2015), dan fleksibel serta data rate yang rendah dan memberikan keandalan yang tinggi untuk kegiatan seperti pengendalian dan pemantauan (Akyildiz, I. F et al. 2002; Latré, B et al. 2006; Yanfei et al. 2009; Uikey, R et al. 2013).

Sistem ini menggunakan sensor network (Xbee module), dan sensor suhu (DHT22) kemudian dari data sensor tersebut akan dikirimkan dari titik end device ke titik coordinator menggunakan teknologi ZigBee. Penelitian ini juga mengajukan suatu topologi alternatif yaitu topologi *Mesh*. Jaringan *Wireless Mesh*

adalah teknik *networking* yang kuat untuk merutekan data. Jangkauannya diperluas dengan memungkinkan data ke *hop node ke node*. Keuntungan yang ditawarkan antara lain kehandalan jaringan, ketika ada satu node yang rusak mudah didiagnosa, memberikan jalur terbaik dalam menentukan rute pengiriman paket, dan mesh memiliki kapasitas untuk bergantian ketika satu *node* gagal atau koneksi terputus (Ashwini *et al.*, 2016).

Distribusi paket data dapat dikatakan berkualitas, jika data yang dikirimkan memiliki standar kualitas *service* atau yang disebut *QoS (Quality of Service)*. Peran QoS pada penilitian ini adalah sebagai alat evaluasi kinerja distribusi antar jaringan. Terdapat tiga parameter tentang informasi kualitas paket diantaranya, besarnya paket data yang dapat dikirimkan dan diterima oleh server (*Throughput*), total waktu pengiriman paket ke server (*Delay*), dan seberapa besar terjadi packet loss pada distribusi jaringan (*Packet Loss*).

Umumnya, metode *routing* yang digunakan pada sekala WSN pada penelitian sebelumnya hanya menggunakan *static routing* protokol yang menggunakan *single node* tanpa mengedepankan aspek *reliability*. Dalam penelitian sebelumnya juga mencoba menerapkan *dynamic routing* dengan menggunakan program *OPNET Modeler* dan *Riverbed Modeler Academic edition 17.5* namun hanya sebatas mensimulasikan berbagai topologi WSN dengan melakukan pengukuran QoS (Fanfang *et al.*, 2015), (Khalaf and Mokadem, 2017), dan (Jung and Cho, 2010). Selain itu, penelitian terkait tentang *balancing* pada WSN juga kami sertakan pada penelitian ini sebagai acuan kontribusi yang berbeda, diantaranya penelitian tentang perancangan system manajemen *load balancing* yang diterapkan pada *robot mobile* untuk mendeteksi mengurangi konsumsi bahan bakar (Jung and Cho, 2010), dan penelitian tentang perbedaan teknik *Load Balancing* pada WSN dengan alat NS2 Simulator (Shivapur, Kanakaraddi and Chikaraddi, 2015).

Oleh karena itu, pada penelitian ini kami mengusulkan kebaharuan *dynamic routing* dan *load balancing* pada WSN menggunakan algoritma *AODV routing* (Yusuf, 2014), dan *Mesh routing* (Rachman, Yanti and Hidayati, 2017). Penelitian ini sudah dilakukan dengan uji coba rangkaian hardware dan pengukuran

kinerja jaringan pada ruang outdoor dimana hasilnya terbukti efektif dalam meningkatkan *reliability* dan *effectivity* kinerja jaringan.

1.2. Rumusan Masalah

Rumusan masalah yang dikaji dalam penelitian ini dapat disebutkan pointnya sebagai berikut:

1. Apa tujuan membangun *dynamic routing* dan pengirimana paket data sensor yang terhubung dengan sistem *wireless sensor*?
2. Bagaimana proses pengiriman paket dari sensor (node sumber) ke sistem jaringan (node tujuan)*?*
3. Bagaimana meningkatkan keandalan dan efektifitas dari sistem balancing pada transimi paket data?
4. Bagaimana cara mengukur kualitas transmisi data sensor pada *wireless sensor network* ?

1.3. Tujuan

Tujuan dari penelitian ini adalah mambangun sistem *dynamic routing* dengan melakukan analisis performa jaringan terutama pada aspek *reliability* dan efektifitas jaringan, dengan mengimplementasikan sistem manajemen pembagian beban *(load balancing)* menggunakan kehandalan pencarian rute baru saat rute terjadi masalah pada jaringan *wireless.* Adanya *dynamic routing* dan *system balancing* yang dibuat pada penelitian ini menjadikan teknlogi *wireless sensor network* memiliki alternatif dalam meningkatkan kehandalan dan efektifitas kinerja jaringan yang dirancang.

1.4. Manfaat

Penelitian ini diharapkan dapat mendukung sistem *routing* dan *balancing* pada impelementasi sistem *wireless sensor network,* dengan meminimalkan penggunaan *storage* melalui system balancing dan mampu merekondisi jalur distribusi tercepat.

1.5. Kontribusi Penelitian

Kontribusi pada penelitian ini adalah merancang *dynamic routing* pada jaringan *wireless sensor* menggunakan alternatif topologi *Mesh* untuk kehandalan kapasitas secara bergantian ketika satu *node* gagal terkoneksi, dan mengatur pembagian beban pada protocol Zigbee IEEE 802.15.4. Perlu diketahui bahwa pada penelitian sebelumnya hanya membas tentang *static routing* pada *wireless sensor network* dan analisis performa pada topologi jaringan menggunakan *network simulator.*

1.6. Batasan Masalah

Untuk memfokuskan permasalahan penelitian ini, batasan masalah yang ditentukan adalah sebagai berikut:

a) prototipe modul *wireless* menggunakan Modul ZigBee Pro S2;
b) microcontroller yang dipakai Arduino Mega 2560 dan Arduino UNO;
c) sensor suhu yang dipakai adalah DHT22;
d) percobaan pengiriman paket sejumlah 100 sample paket data;
e) kriteria performa pengujian sistem dan alat berdasarkan parameter *QoS (delay), (throughput), (packet loss), Receive Signal Strength Indicator (RSSI), fault tolerance;*
f) jumlah node yang digunakan 5 node;
g) jumlah node end device 3 node;
h) jumlah node router 2 node, jumlah node coordinator 1 node;
i) batas penyimpanan router sebesar 10 paket data;
j) pembagian beban pada transmisi data dilakukan dengan cara melakukan pengalihan routing router 1 dan router 2;
k) tekanan udara, gangguan udara dan angin tidak termasuk dalam penelitian tesis ini;
l) sistem dan teknologi dirancang hanya secara *prototype*, tidak termasuk perancangan dan pengaturan kontrol lainnya;
m) dasar model jaringan menggunakan *Ad-Hoc* yang didukung dengan jaringan Mesh;

n) sistem pembagian beban (load balancing) dilakukan pada bidang pengolahan data sensor, dan manajemen antar jaringan *wireless sensor network.*

1.7. Sistematika Penulisan

Berikut ini adalah sistematika penulisan yang akan diterapkan pada proses penelitian ini :

- **Bab 1 Pendahuluan**

  Bab ini menyajikan tentang latar belakang, rumusan masalah, tujuan, manfaat, batasan masalah, kontribusi penelitian, dan sistematika penulisan.

- **Bab 2 Kajian Pustaka**

  Dalam bab ini terdapat sub bab dan landasan teori dari penelitian terdahulu yang memaparkan teori-teori yang berhubungan dengan masalah yang diteliti serta beberapa penelitian yang telah dilakukan pada penelitian-penelitian sebelumnya.

- **Bab 3 Metode Penelitian**

  Bab ini menguraikan deskripsi tentang bagaimana penelitian nantinya akan dilakukan dan menjelaskan variabel penelitian, definisi operasional, penentuan jenis sampel, jenis dan sumber data, jalannya penelitian dan alur penelitian.

- **Bab 4 Hasil Penelitian dan Pembahasan**

  Bab ini menjelaskan tentang pengumpulan data dan pengolahan data serta menguraikan tentang deskripsi objek penelitian melalui gambaran umum dan proses pengintegrasian data yang diperoleh untuk mencari makna dari hasil analisa.

- **Bab 5 Kesimpulan dan Saran**

  Bab ini menyajikan kesimpulan dan saran yang didapatkan dari pembahasan pada hasil penelitian.

BAB II
KAJIAN PUSTAKA

Pada bab ini akan dijelaskan tentang dasar teori yang digunakan dalam penelitian. Dasar teori yang digunakan antara lain mengenai definisi Jaringan Ad-Hoc, model *Ad hoc On demand Distance Vector (AODV), Quality of Service* (QoS) Performance, Konsep Pengalamatan Jaringan, dan model Mesh Routing.

2.1. Penelitian Terkait

Berdasarkan penelitian yang telah dilakukan oleh sejumlah orang terkait tentang hasil penelitian sistem *static routing* dan *dynamic routing* dapat dijadikan referensi pembanding. Dari penelitian ini diharapkan dapat memberikan kontribusi lain dari penelitian yang sudah dilakukan.

1. S. Soijoyo and A. Ashari. 2017 [2] “Analysis of Zigbee Data Transmission on Wireless Sensor Network Topology". Melakukan penelitian analisis transmisi data dengan skenario topologi menggunakan metode pengukuran QoS. Menggunakan tiga percobaan dengna topologi Star, Mesh dan Tree. Hasil dari penelitian ini menyatakan bahwa topologi Star memiliki performa stabil dalam pengukuran throughtput dan packet loss dibandingakn topologi Mesh dan Tree, hal ini dikarenakan pada topologi Star tidak terdapat perangkat Router, sehingga akurasi pada pengiriman data lebih baik

2. W. Jung and S. H. Cho. 2010 [4] "Load balancing system with sub-network management in wireless sensor networks. Menawarkan sebuah penelitian tentang system balancing yang dapat mendistribusikan trafik data yang besar dengan memperpanjang masa hidup jaringan. Sistem ini diimplementasikan pada *mobile robot.* Hasil dari penelitan tersebut memberikan evaluasi performa jaringan yang dapat mengurangi konsumsi energi.

3. N. T. Le and W. Benjapolakul. 2019 [7] "Received signal strength data of ZigBee technology for on-street environment at 2.4 GHz band and the interruption of

vehicle to link quality. Pada penelitian ini melakukan percobaan pengukuran RSSI yang dilakukan di kondisi outdoor. Jarak pengukurang yang dilakukan adala 50 – 260 m. Pada jurnal artikel tersebut membantu memberikan estimasi jarak ukur pengiriman paket data.

4. F. Z. Rachman, N. Yanti and Q. Hidayati. 2017 [8] "Implementasi Jaringan Sensor Nirkabel Zigbee Menggunakan Topologi Mesh Pada Pemantauan Dan Kendali Perangkat Ruang. Hasil dari percobaan mereka adalah membangun sebuah system terintegrasi pemantau ruangan dengan menggunakan empat modul sensor, seperti sensor PIR, sensor Asap, Finger Print, dan sensor Arus. Selain itu penggunaan topologi Mesh pada Zigbee juga terhubung pada perangkat ini, dan di akhir penelitian peneliti melakukan pengukurang Qos dengan jarak 93-100 meter.

5. D. Fanfang, Z. Dahai, B. Zhiqian, C. Xin, D. Xinzhou and S. Shenxing. 2015 [9] "ZigBee Wireless Networking for the Intelligent Protection Center Research. Peneliti melakukan Analisa system Protection Center yang didesain dan disimulasikan menggunakan teknologi Zigbee. Hal ini membukti *channel wireless* dari Zigbee memberikan solusi cepat dan nyaman dalam hal *recovery of power communication.*

6. H. Kumbhar, "Wireless Sensor Network using Xbee on Arduino Platform An experimental study. 2016 [10]. Dalam penelitan tersebut, diajukan implementasi praktis membuat WSN dengan topologi Mesh. Dengan menggunakan *microcontroller* Arduino WSN ini dirancang sebagai system pembacaan suhu secara real time dan dipasang di berbagai loakasi serta dikirimkan ke base station.

7. D. S. Yun and S. H. Cho. 2008 [11] "A Data Transmission Method in ZigBee Networks Using Power Efficient Device. Pada penelitian ini, mengajukan permasalahn pada alamat ZED pada saat transmisi. Pada ZED fitur SM (Sleep Mode) diaktifkan, system ini mendukung metode *reducting power comsuption*

yang ditawarkan dapat mengurangi tingkat kepadatan pada ZED, dalam metode ini diharapakan dapat mengirimakan data secara efisien dan reliability.

8. R. Piyare and S.-r. Lee 2013 [12] "Performance Analysis of XBee ZB Module Based Wireless Sensor Networks. Pada penelitian tersebut, melakukan analisa performa dari topologi yang berbeda dengan Xbee. Dari dua hasil sekenario yang dirancang, bahwa pengirimana data ZED secara langsung ke ZC jauh lebih cepat dibandingkan dengan adanya sebuah ZR pada sebuah jaringan ZigBee yang multihop. Sehingga pada hasilnya, peneliti menggunakan fitur SM (Sleep Mode) pada ZED (ZigBee End Device) untuk meningkatkan masa hidup jaringan ZigBee.

2.2. Perangkat Keras Pendukung

Dalam sistem ini, kami mengintegrasikan modul perangkat keras mikrokontroler (Ardiansyah, Sarno and Giandi, 2018)Arduino Uno ATmega328, Sensor DHT 22 (Konsentrasi Suhu Temperature), Modul ZigBee SC Pro 2, dan Adapter Zigbe yang semua sensor ini akan melekat dalam mikrokontroler Arduino Uno ATmega328 menggunakan analog atau port digital. Unit hardware yang terhubung ke modem nirkabel ZigBee.

2.2.1 Sensor Suhu DHT 22

DHT 22 adalah sensor yang dapat melakukan pengukuran suhu dan kelembaban pada satu waktu (Saptadi, 2014). Dilengkapi dengan dua modul sensor secara terpisah yakni pengukur suhu *Termocouple* dan *Humidity* namun menjadi satu dalam satu produk. Sensor ini terhubung ke Arduino melalui komunikasi serial dan menggunakan tipe data digital (Ardiansyah, Sarno and Giandi, 2018).

2.2.2 Arduino Mega 2560

Arduino mega 2560 merupakan keluaran single board *microcontroller* dari keluarga Arduino. Arduino Mega pada penelitian ini digunakan sebagai proses pada pengolahan data sensor, *wireless sensor network*, dan sebagai pusat kontrol sistem deteksi kebakaran. Arduino Mega memiliki spesifikasi antara lain (Giandi and Sarno, 2018). Komunikasi USART (*Universal Synchronous Asynchronous*

Receiver Transmitter), dan SPI *(Serial Peripheral Interface).* Modul ini juga telah dilengkapi sejumlah 54 pin digital dan 16 pin analog yang dapat digunakan sebagai input atau output.

2.2.3 Modul Xbee Pro 2

ZigBee merupakan standar jaringan nirkabel yang ditujukan untuk remote control dan aplikasi sensor yang cocok untuk operasi di lingkungan indoor dan outdoor dan lokasi terpencil yang sulit dijangkau. Beberapa kelebihan protokol ZigBee ialah biaya rendah, handal, baterai tahan lama, *high security, self-healing properties, large number of nodes supported*, pengoperasian lebih mudah, pengiriman terjamin, optimasi rute (Culler D et al 2004; Ergen S.C 2004).

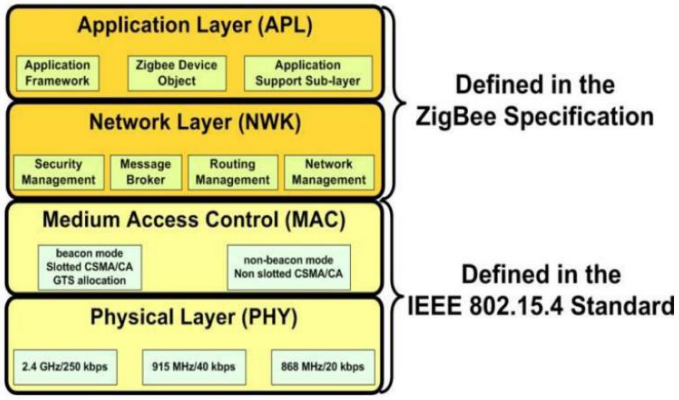
2.3. Aplikasi Wireless Sensor Network (WSN)

Menurut Roy *et al*. (2009) *wireless mesh network* adalah salah satu tipe jaringan dimana setiap *node* pada jaringan dapat berkomunikasi dengan banyak *node* lainnya (Fuad, 2015). *Wireless sensor network* (WSN) merupakan jaringan *wireless* yang menggunakan sensor untuk memonitor fisik atau kondisi lingkungan sekitar. *Wireless sensor* biasanya digunakan untuk fungsi monitoring yaitu mengukur suatu besaran fisis misal: suhu, tekanan, kelembaban, dan lain-lain dan mengirimkan datanya kepada sebuah data *concentrator*. Berdasarkan data yang terkumpul tersebut, kemudian data bisa ditampilkan dalam bentuk grafik, diambil keputusan tertentu berdasarkan *event-trigger*. Serta fungsi control yaitu pengontrolan pada WSN dan umumnya dilakukan pada penggunaan WSN dengan skala kecil serta fungsi kontrolnya terbatas (Iqbal, 2015).

2.3.1 Zigbee Protocol

Zigbee adalah sebuah protokol IEEE 802.15.4 MAC dan Layer PHY yang dirancang untuk skala yang minimalis meliputi, low bandwith, low cost, low-power, dan standar jaringan wireless (Piyare and Lee, 2013). Jaringan mesh memberikan kemampuan kehandalan yang tinggi, dan cakupan jarak yang jauh dengan power yang rendah dan daya tahan yang lama. Contoh penerapan jaringan mesh biasanya diterapkan pada system yang menerapkan monitoring, controlling,

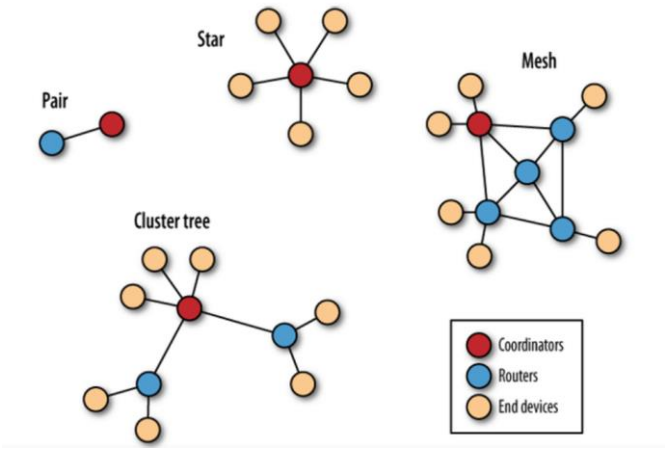
pada aplikasi system wireless. Zigbee dioperasikan di kawasan industri, pendidikan dan bidang kesehatan yang menjangkau pada frekuensi; 868 MHZ di Eropa, 915MHZ di kedua benua Australia dan Amerika, dan 2.4 GHZ band international. Adapun data transmission Zigbee berkisar dari 20kb/s sampai 250kb/s dalam gelombang radio 868 MHZ sampai 2.4 GHZ , dan Zigbee mendukung network layer peer to peer, cluster trees, dan jaringan mesh.



Gambar 2. 1 Arsitektur Zibee (Yusuf, 2014)

2.3.2 Topologi WSN

Seiring dengan kemajuan teknologi WSNs, perubahan besar terjadi dalam jaringan distribusi juga. Banyak bentuk jaringan telah diperkenalkan untuk dicapai tindakan yang berbeda dan mereka memiliki kelebihan dan keterbatasan yang berbeda. Di sini kita merangkum beberapa topologi terkenal yang digunakan dalam WSNs (Sharma, Verma and Sharma, 2013).



Gambar 2. 2 Topologi Jaringan WSN

2.3.2.1 Peer to Peer Networks

Topologi ini bersifat ad-hoc dan mendukung komunikasi node to node. Satu simpul dapat berkomunikasi dengan node lain secara langsung selama mereka berada dalam jangkauan. Jika mereka tidak dalam jangkauan, satu node dapat berkomunikasi melalui node lain juga. Itu Aspek penting dari jaringan peer to peer ini adalah organisasi diri dan penyembuhan diri mereka sendiri. Topologi ini dapat diimplementasikan dalam pemantauan industri atau pelacakan inventaris karena data lewat langsung dari satu node ke node lainnya dalam keadaan sekecil mungkin waktu. Node bertindak tidak hanya sebagai penerima atau pengirim, tetapi juga sebagai router dalam jenis ini topologi jaringan (Sharma, Verma and Sharma, 2013). Gambar 2.2 menunjukkan jaringan *peer to peer.*

2.3.2.2 Star Networks

Tidak seperti jaringan *peer to peer*, dalam jaringan bintang, node tidak dapat berkomunikasi dengan satu sama lain. Jaringan ini memiliki koordinator yang berfungsi sebagai simpul kepala dan yang lainnya node terhubung dengannya. Jadi semua data pada awalnya harus pergi ke coordinator. Lebih besar jaringan, node sensor biasanya ditenagai oleh baterai tetapi coordinator diaktifkan langsung oleh listrik untuk menopang kebutuhan komunikasi dengan node sensor lainnya data tidak terputus. Meskipun node sensor menghubungkan diri mereka ke coordinator, mereka adalah operator independen (Sharma, Verma and Sharma, 2013). Gambar 2.2 menunjukkan jaringan *Star Network.*

2.3.2.3 Mesh Networks

Jaringan mesh memungkinkan node untuk berkomunikasi dari satu ujung ke ujung lainnya dengan melompat melalui node lain, yang membuatnya dapat memiliki kemampuan *self-healing* dengan kata lain sembuh sendiri. Setiap node terhubung dengan node lain yang tersedia di area yang sama. Topologi mesh bisa sangat mahal karena strukturnya yang inklusif. Ini adalah struktur paling kompleks di antara topologi jaringan (Sharma, Verma and Sharma, 2013). Gambar 2.2 menunjukkan jaringan *Mesh Network.*

2.3.2.4 Cluster Tree

Topologi pohon sering disebut bintang hibrida dan topologi jaringan *peer-to-peer*. Hub pusat terhubung dengan node lain serta simpul root dan dengan demikian membuat situasi seperti topologi pohon. Node terhubung dengan hub pusat dan kemudian dari hub pusat, sinyal menuju ke simpul root. Jadi ketika sebuah simpul baru ingin bergabung dengan jaringan ini, ia harus terhubung hub pusat pertama (Sharma, Verma and Sharma, 2013). Gambar 2.2 menunjukkan contoh topologi jaringan pohon untuk memudahkan pemahaman konsep.

2.3.3 Mode Operation Xbee

Terdapat tiga mode operasi dalam penggunaan Xbee, dari ketika mode ini disesuaikan dengan kebutuhan dasar pembentukan jaringan. Berikut jenis tiga mode operasi;

- ***Coordinator***

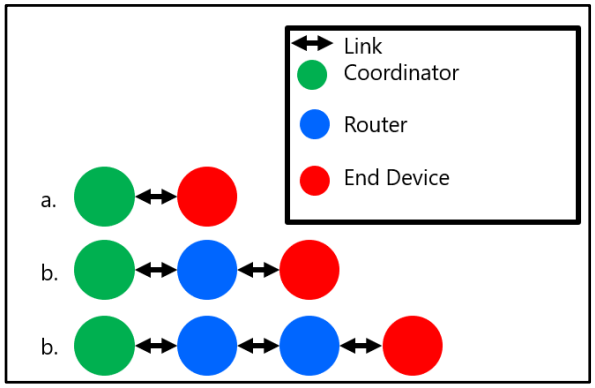
  Coordinator bertanggung jawab untuk membangun operating *channel* dan PAN *(Personal Area Network)* ID pada sebuah jaringan. Coordinator dapat membentuk suatu jaringan dengan mengijinkan router dan end device untuk bergabung dalam jaringan tersebut. Setelah jaringan terbentuk, fungsi coordinator seperti router (dapat berpartisipasi dalam routing paket dan menjadi sumber atau tujuan untuk paket data).

- ***Router***

  Adalah sebuah node yang bertugas untuk menerima, mengirimkan dan merutekan data. Agar sebuah router dapat mengijikan router lain dan end device untuk bergabung, maka router tersebut harus terlebih dahulu bergabung dengan jaringan PAN. Selain itu, router juga dapat berfungsi merutekan data atara PAN satu dengan yang lainnya. Router dapat berpartisipasi dalam routing paket dan menjadi sumber atau tujuan untuk paket data.

- ***End device***

  End device harus selalu berinteraksi atau terhubung dengan coordinator atau *router* untuk dapat menerima dan mengirimkan data. *End device* juga dapat menjadi sumber atau tujuan untuk paket data tetapi tidak bisa untuk menentukan rute paket data.



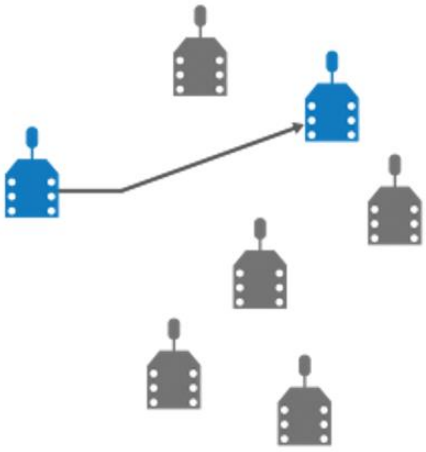
Gambar 2. 3 Dasar Topologi Jaringan Protokol Zigbee

2.3.4 Zigbee Data Transmission

Semua paket data Zigbee dikirimkan menggunkan perangkat keras wireless dan application layer addressing field (Chaitanya, 2007). Paket data Zigbee dapat di kirimkan sebagai mode pengiriman *unicast* atau *broadcast*.

2.3.3.1 Unicast

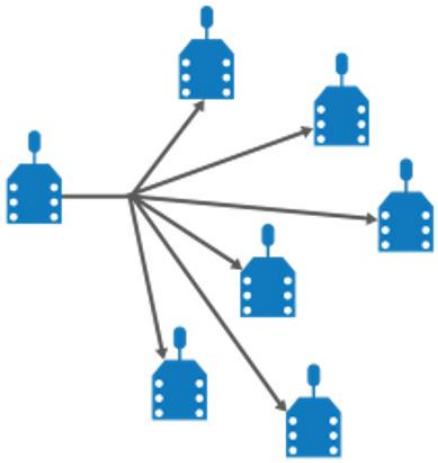
Transmisi unicast adalah pengiriman data dari satu perangkat ke perangkat sumber tujuan lain. Unicast Mode adalah satu-satunya mode yang mendukung pengulangan. Dalam mode ini modul receiver mengirim paket *acknowledgement* (ACK) RF ke pemancar transmitter. Jika modul pengirim tidak menerima ACK, maka paket tersebut akan dikirim ulang hingga tiga kali atau sampai ACK diterima. Berikut ini adalah gambar contoh untuk setting perangkat XBee yang melakukan pengiriman data secara unicast antara router dan coordinator.



Gambar 2. 4 Unicast Transmission

2.3.3.2 Broadcast

Seluruh node dalam satu jaringan PAN sehingga semua node menerima data *broadcast* yang dikirimkan. Untuk *broadcast* mode, di salah satu node yang akan membroacast informasi, nilai DL diubah menjadi "0xFFFF" dan nilai DH menjadi "0". Sedangkan pada node yang lain DL nya diisi dengan SL dari node yang mem-broadcast informasi. Konfigurasi XBee Model transmisi *broadcast* ditunjukkan pada gambar 2.5.



Gambar 2. 5 Broadcast Transmission

2.4. Quality of Service (QoS)

QoS merupakan sekumpulan parameter yang menunjukkan kualitas layanan suatu jaringan dan kemampuan jaringan tersebut dalam menjalankan aplikasi-aplikasi dengan kinerja sesuai dengan yang dibutuhkan. Dengan mengetahui *QoS* kita dapat mengetahui kondisi jaringan dan menyesuaikan jaringan dengan aplikasi yang akan digunakan (Fuad, 2015).

Beberapa parameter QoS antara lain:

- *Throughput*

  Menunjukkan besarnya paket data yang diterima pada *node* tujuan dibandingkan dengan waktu tempuh yang ditulis dalam satuan *bit per second (bps)* (Misra *et al.* 2009).

  $$\text{throughput} = \frac{total\ received\ bit}{end\ time - start\ time} \quad \text{..... (1)}$$

- *Delay*

  Merupakan selang waktu antara mulai dikirimkannya paket data sampai paket diterima di node tujuan (Szigeti dan Hattingh 2004).

  delay $= \frac{\sum(received\ time - send\ time)}{received\ packet}$ .....(2)

- *Packet Loss Ratio (PLR)*

  Yaitu banyaknya data yang hilang pada suatu proses pengiriman data ke node tujuan (Hanzo dan Tafazolli 2007).

  PLR $= \frac{sent\ packet - arrived\ packet}{total\ packet} X\ 100\ \%$ .....(3)

2.5. Arsitektur Xbee Pro 2

Perangkat ini memiliki 20 pin dengan fungsi sebagai *transceiver* dan *receiver*. Untuk koneksi minimum, dibutuhkan pin *VCC, GND, DOUT & DIN*. Sedangkan untuk dapat melakukan *update firmware*, dibutuhkan koneksi pin *VCC, GND, DIN, DOUT, RTS & DTR. VCC* dan *GND* untuk tegangan suplai, *DOUT* merupakan pin Transmit *(TX), DIN* merupakan pin Receive *(RX), RESET* merupakan pin reset *XBee PRO* dan yang terakhir adalah *PWMO/RSSI* merupakan indikator bahwa ada penerimaan data yang biasanya dihubungkan ke led yang di-drive oleh transistor. Untuk mengaktifkan XBee dibutuhkan supply tegangan sebesar 3.3 V.

2.5.1 Konsep Pengalamatan Pada Xbee Pro 2

Terdapat dua teknik pengalamatan dalam konfigurasi Xbee Pro 2 yakni pertama penggunaan alamat 64-bit dan alamat 16-bit. *Source Address* unik 64-Bit IEEE ditetapkan oleh pabrik dan dapat dibaca dengan perintah *SL (Serial Number Low)* dan *SH (Serial Number High)*. Sedangkan pengalamatan 16-bit harus dikonfigurasi secara manual.

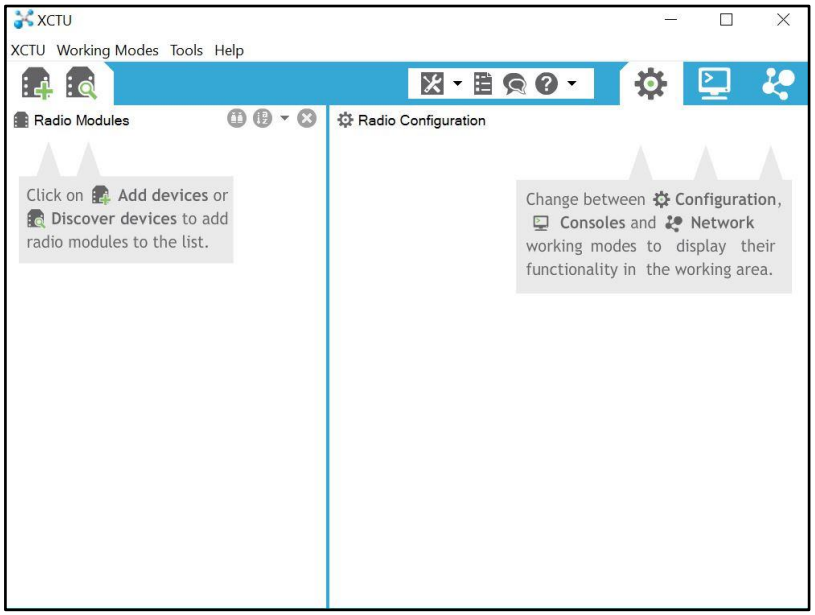
Pada mode pengalamatan 16-bit digunakan range alamat dari *0 – 0xFFFF*. Modul akan menggunakan alamat 64-bit jika nilai pada *Source Address DL (mode alamat 16-bit)* adalah *"0xFFFF" atau "0xFFFE"*. Untuk mengirimkan paket ke

spesifik modul menggunakan alamat 64-bit, *Destination Address (DL+DH)* dari pengirim harus disesuaikan dengan *Source Address (SL + SH)* dari modul tujuan. Sedangkan untuk mengirimkan paket data menggunakan alamat 16-bit, pada sisi *Destination Address Low (DL)* disesuaikan dengan *Source Address (SL)* pada modul tujuan dan untuk nilai *Destination Address High (DH)* diatur '0'. (Anonim, 2015).

2.5.2 Framework XCTU

Digi mengembangkan *X-CTU* yang merupakan perangkat lunak yang digunakan untuk konfigurasi dan pengujian pada produk *RF Digi*. Banyak fitur yang disediakan dalam *software* ini untuk melakukan konfigurasi pada produk RF keluaran Digi salah satunya XBee-pro.

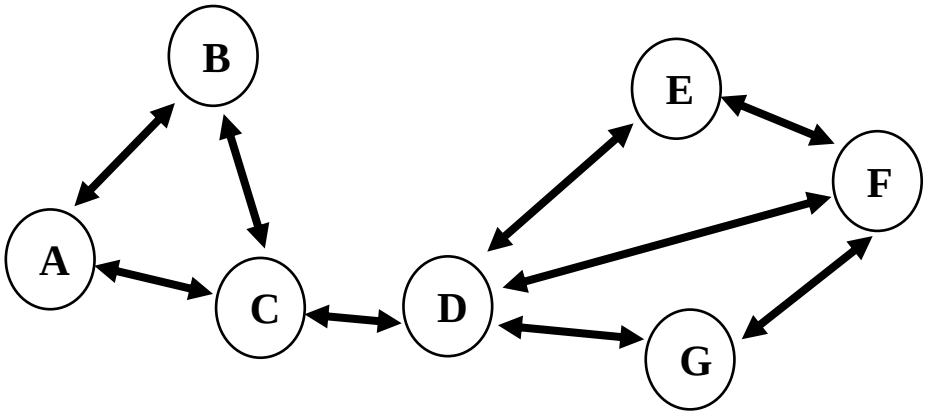
- Support seluruh produk Digi.
- Jendela terminal yang terintegrasi.
- *Upgrade firmware* modul RF.
- Menampilkan ASCII dan *hexadecimal* karakter di jendela terminal.
- Menyimpan dan mengambil konfigurasi modul umum digunakan (profil).
- Secara otomatis mendeteksi jenis modul.



Gambar 2. 6 Tampilan Konfigurasi XCTU Xbee

2.6. Jaringan Wireless Ad-Hoc

Jaringan wireless dibagi menjadi dua model, yaitu jaringan *wireless fixed* dan *mobile*. Pada jaringan *fixed wireless* tidak mengukung *mobility,* dan kebanyakan adalah *point to point.* Berbeda dengan *mobile* wireless yang dibutuhkan oleh pengguna agar bisa bergerak. Pada jaringan fixed wireless tidak memiliki insfrastruktur, jaringan ini disebut dengan jarinan *ad hoc* (Somprakash, Amitava and Saha, 2003). Jaringan *ad hoc* datapat diartikan sebagai suatu jaringan tanpa infrastruktur dimana masing-masing *node* adalah suatu *router* bergerak yang dilengkapi dengan *transceiver* wireless. Pesan yang dikirim dalam jaringan ini akan berada diantara dua *node* dalam cakupan transmisi yang secara tidak langsung dihubungkan oleh *multiple hop* melalui beberapa *node* perantara (Somprakash, Amitava and Saha, 2003). Gambar 2.7 menunjukkan *node* C dan *node* F berada di luar cakupan transmisi satu dengan lainnya, tetapi masih dapat berkomunikasi melalui perantara *node* D dalam *multiple hop*.

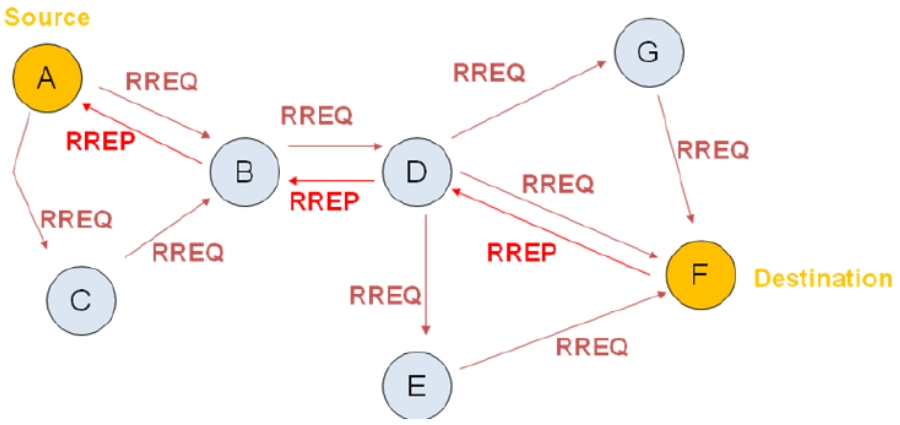


Gambar 2. 7 Struktur Dasar Jaringan Ad Hoc

2.7. AODV (Ad-Hoc Distance Vector)

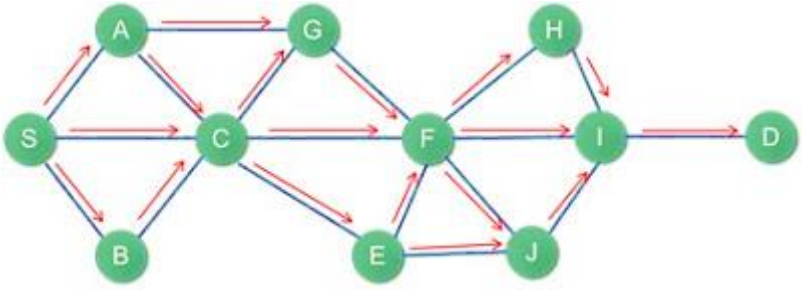
Ad hoc On demand Distance Vector (AODV) sangat populer apabila diterapkan ada jaringan *wireless sensor* (Kim, Moon and Cho, 2009). AODV adalah *routing protocol* yang dirancang untuk jaringan *ad hoc mobile.* Teknik routing pada AODV memudahkan dalam menemukan sebuah *route.* Dapat menggunakan pesan hello untuk koneksi informasi dan dapat juga memberikan infromasi kesalahan dalam *link route*. Pada AODV memiliki komponen utama seperti *Sequence number, RREP, RREQ, hop count, Hello message, precursor limits* (Karthikeyan, 2010).

Penggunaan *Sequence number* mendukung untuk mendeteksi data yang sudah kedaluarsa, sehingga hanya *routing* terbaru yang digunakan.



Gambar 2. 8 AODV Routing

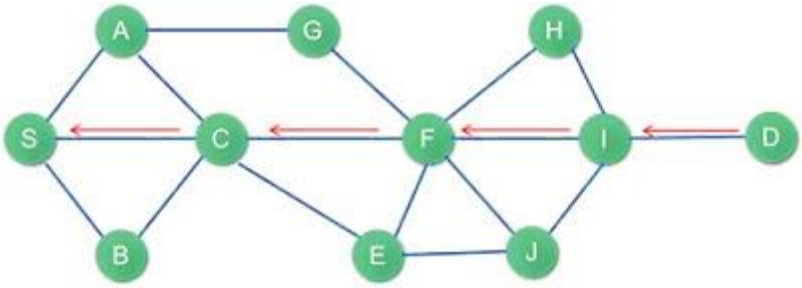
Pada gambar 2.8 melakukan proses pengiriman paket dari sumber *node Source "A"* ingin mengirim paket data ke *node* tujuan Destination "F" tetapi tidak memiliki rute ke F dalam table routing-nya, maka pencarian rute harus dilakukan oleh node "B". Selanjutnya paket data *buffer* disimpan selama penemuan rute. Sumber node "A" melakukan *broadcast* dalam hal ini RREQ *(Route Request)* melalui jaringan. AODV membangun rute baru menggunakan 2 sinyal yakni RREQ (*Route Request*) dan RREP (*Route Replay*).



Gambar 2. 9 Pengiriman Route Request (RREQ)

Berdasarkan gambar 2.9 ketika sumber *node* membutuhkan rute ke tujuan yang belum memiliki rute, maka rute tersebut menyiarkan permintaan (RREQ) paket melalui jaringan. Disaat *node* tujuan telah menerima paket *request* dengan

memperbaharui informasi untuk node sumber, selanjutnya *node* tujuan membuat pointer ke *node* sumber dalam bentuk table *routing*.



Gambar 2. 10 Pengiriman Route Replay (RREP)

Sebuah node menerima RREQ dapat mengirim rute balasan (RREP) seperti pada gambar 2.10 jika dalam frekuensi radio yang sama dengan spesifikasi alamat tujuan maka alamat tujuan dapat dikenali, jika tidak suatu node tidak dikenali, maka akan melakukan rebroadcast ulang. Node tetap melacak sumber RREQ alamat IP dan *Broadcast* ID. Hal ini dimaksud sebagai tugas update informasi routing untuk tujuan pemilihan rute yang lebih baik. Selama rute tetap aktif, hal ini akan terus dipertahankan. Sebuah rute dianggap aktif selama ada paket data secara berkala melakukan aktifitas transmisi.

Keuntungan utama dari protokol ini adalah bahwa rute yang dibangun pada permintaan dan nomor urut tujuan digunakan dalam menemukan rute tujuan. Sambungan konfigurasi delay lebih rendah, komunikasi sepanjang link lebih efektif, karena tidak memerlukan lalu lintas tambahan. Selain itu jarak *vector routing* sederhana, dan tidak memerlukan banyak memori atau perhitungan (Yohanes, 2015).

Kekurangan dari protokol AODV adalah protokol ini membutuhkan lebih banyak waktu untuk membangun sambungan, dan komunikasi awal untuk mendirikan sebuah rute lebih berat dari beberapa pendekatan lain.

2.7.1 Route Discovery

Routing discovery adalah kemampuan routing protokol untuk membagi informasi tentang jaringan dengan node lainnya dengan menggunakan routing protokol yang digunakan. Pemilihan jalur terbaik pada setiap jaringan terdapat pada

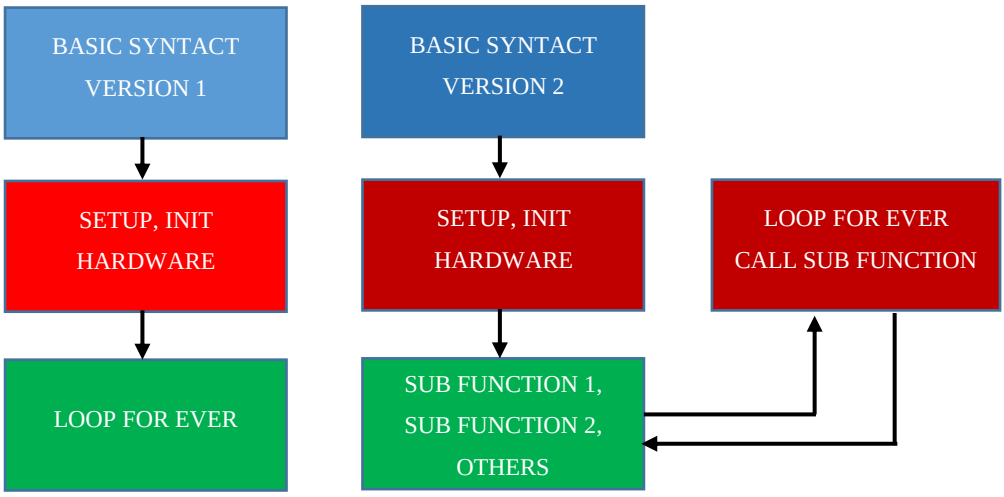
table routing dengan menggunakan routing dinamik (Sepriwono, 2018). Routing discovery dapat dipengaruhi oleh berbagai factor seperti timer, jumlah, posisi, dan pergerakan node.

2.7.2 Route Maintenance

Route maintenance adalah kondisi system melakukan perbaikan rute yang mengalami kerusakan. Permasalahan ini biasanya jika sebuah *link* ke hop *(destination node)* berikutnya tidak dapat terdeteksi dengan menggunakan metode penemuan rute, maka link tersebut akan mengirimkan pesan *route error (RRER)* ke node tetangganya sampai tersebar ke *source node.*

2.8. Software Arduino (Pemrograman C)

Bahasa C adalah bahasa pemrograman yang dapat dikatakan berada di antara bahasa level rendah dan level tinggi. Bahasa level rendah artinya bahasa yang berorientasi pada mesin dan bahasa level tinggi berorientasi pada manusia. Bahasa level rendah, misalnya bahasa *assembler*, bahasa ini ditulis dengan sandi yang dimengerti oleh mesin saja, oleh karena itu hanya digunakan bagi yang memprogram mikroprosesor.



Gambar 2. 11 Struktur Dasar Program Arduino

Struktur dasar dari Program Arduino seperti gambar 2.11 merupakan jenis program yang memilik struktur dasar. Struktur ini seperti layak nya susunan program, ada global *variable*, ada setup yang akan di akses sekali, kemudian ada

program yang berulang ulang atau terus menerus di jalankan sampai *power supply* di matikan. Pada posisi global variabel, semua variabel yang dideklarasikan dapat digunakan di semua fungsi yang kita buat. Pada bagian Setup Init merupakan fungsi yang berfungsi untuk mendeklarasikan pin in/out yang langsung terhubung dengan pin in/out hardware mikroporsesor. Biasanya di fungsi ini dideklarasikan pula mode suatu perangkat sensor akan dijadikan sebagai input atau output. Dan pada bagian fungsi loop, adalah tempat system dijalankan secara urut dari baris pertama pada fungsi loop hingga pada baris akhir.

BAB III
METODOLOGI PENELITIAN

Bab ini menjelaskan tentang metodologi yang akan dilalui pada penelitian ini. Adapun alur penelitiannya terdiri dari lima tahap, meliputi: (1) studi literatur, (2) analisis permasalahan, (3) rancangan penelitian, (4) implementasi penelitian, (5) pengujian dan analisis hasil, (6) penulisan laporan penelitian. Ilustrasi alur metodologi penelitian ditujukkan pada Gambar 3.1. Penjelasan dari tahapan metode penelitian dan diterangkan secara terperinci pada sub-bab berikutnya.

3.1. Studi Literatur

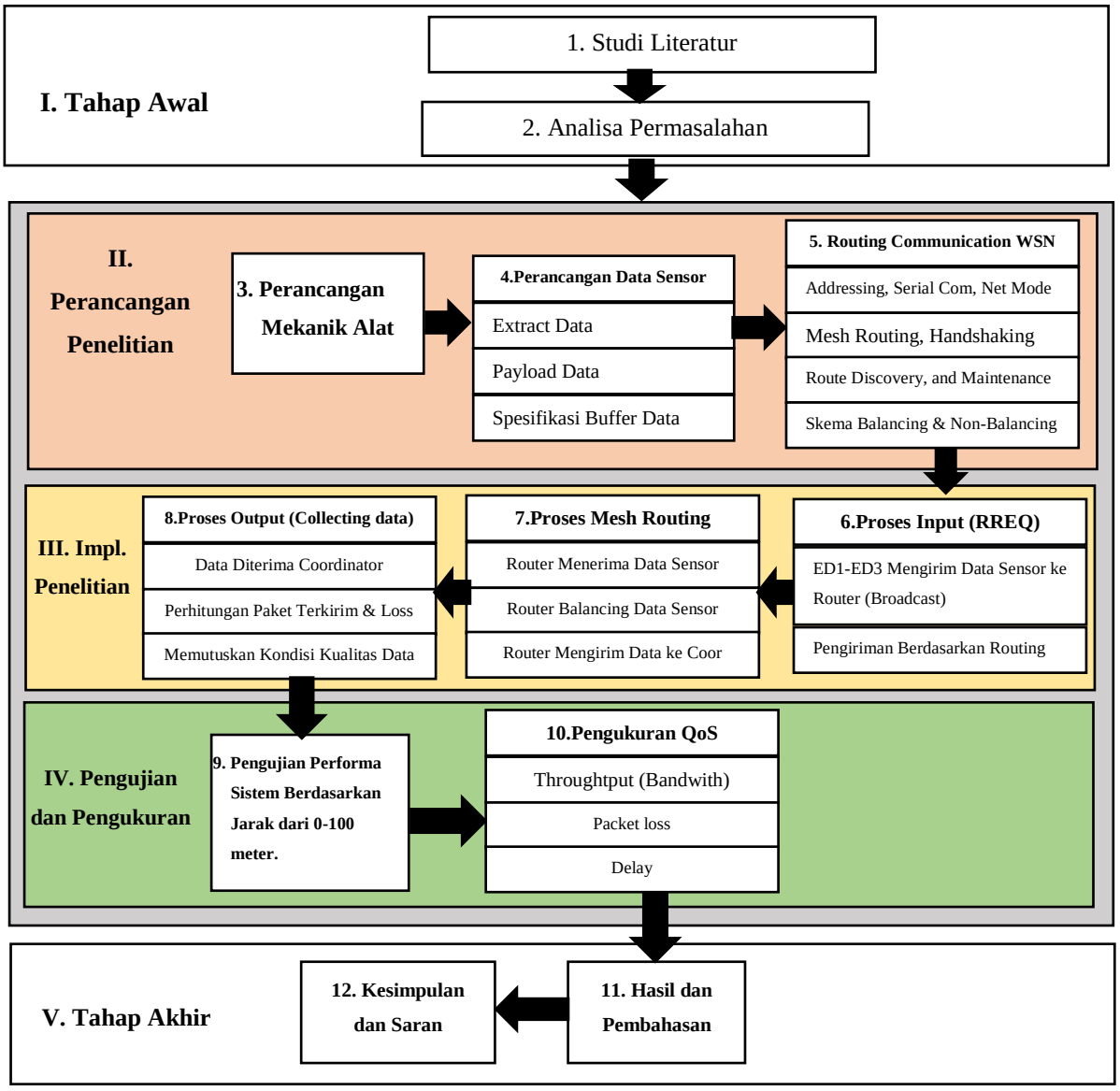
Penelitian diawali dengan melakukan kajian yang berkaitan dengan topik penelitian. Referensi yang digunakan dalam penelitian ini berasal dari jurnal, konferensi, dan buku yang berkaitan dengan *sistem routing*, *sistem balancing*, wsn dan manajemen *wireless sensor network* yang datanya diambil dari sensor atau pengukuran.

3.2. Analisis Permasalahan

Latar belakang permasalahan penelitian ini, Umumnya, metode *routing* yang digunakan pada skala WSN pada penelitian sebelumnya hanya menggunakan *static routing* protokol yang menggunakan *single node* tanpa mengedepankan aspek *reliability*. Dalam penelitian sebelumnya juga mencoba menerapkan *dynamic routing* dengan menggunakan program *OPNET Modeler* dan *Riverbed Modeler Academic edition 17.5* namun hanya sebatas mensimulasikan berbagai topologi WSN dengan melakukan pengukuran QoS (Gautam and Sen, 2015), (Fanfang *et al.*, 2015), (Khalaf and Mokadem, 2017). Mengapa perlu adanya manajemen *wireless sensor network? node* dalam sebuah jaringan memiliki batasan dari aspek penyimpanan, sumber daya listrik, dan keterbatasan dalam hal komputasi secara otomatis (Augusto, 2013).

3.3. Perancangan Penelitian

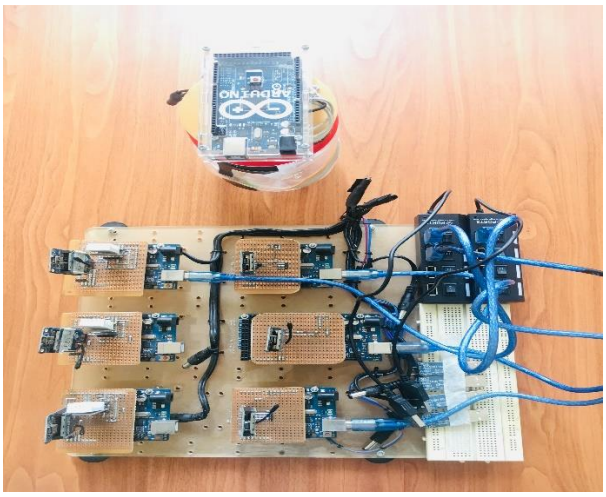
Pada penelitian ini terdapat lima bagian perancangan yang mengacu dari gambar 3.1 dimana diantaranya yakni, pertama melakukan study literatur dan mencari rumusan maasalah dari penelitian ini. Kedua perancangan mekanik alat/desain *prototipe* ruangan, setelah itu melakukan extraksi pada perancangan data sensor, dan melakukan perancangan routing. Ketiga implementasi alat dan system balancing, Keempat melakukan uji coba performa jaringan dan Kelima menuliskan hasil kesimpulan dari penelitian kedalam bentu laporan.



Gambar 3. 1 Diagram Tahap Penelitian

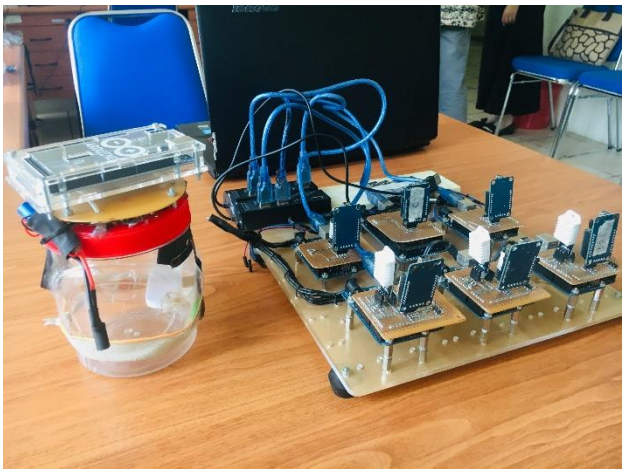
3.3.1 Perancangan Mekanik Alat

Ditahap awal dilakukan pembuatan maket prototipe dibentuk kotak persegi dan dibuat dari bahan akrilik sebagai alas tempat modul kontroler dengan ukuran 30 x 30 cm dan bagian atas diletakkan komponen sensor dan modul utama. Maket ini dirancang sebagai tempat peletakan dari alat penelitian, seperti sensor, *Xbee, Arduino,* dan *power USB extended.* Berikut ini perancangan mekanik alat yang divisualisasikan.



Gambar 3. 2 Maket Tampak Dari Atas

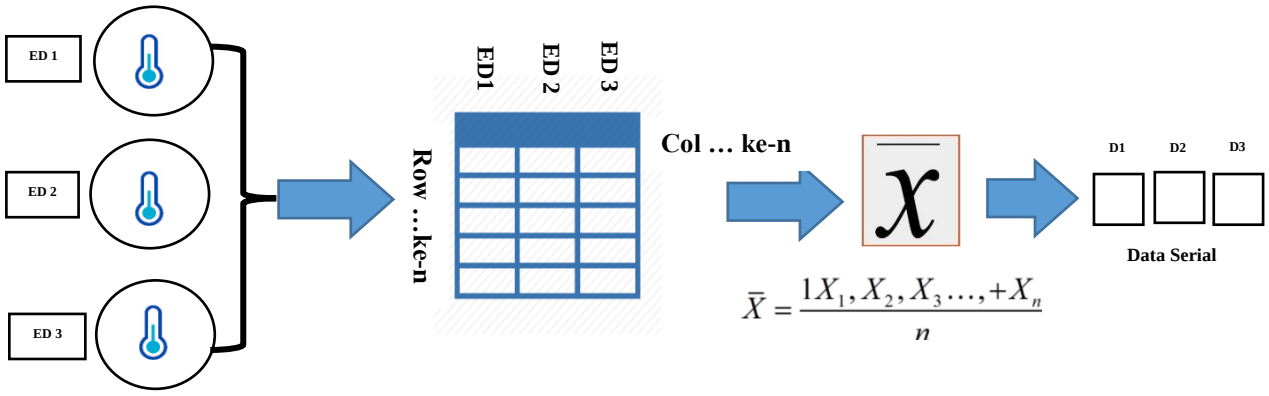
Pada gambar 3.2 dan gambar 3.3 memberikan visualisasi tentang desain alat yang digunakan sebagai tempat meletakkan komponen seperti sensor suhu, modul *Xbee, Arduino,* dan *power usb extended.* Termasuk juga pada gambar 3.3 yang terlihat.



Gambar 3. 3 Maket Tampak Dari Samping

3.3.2 Perancangan Data Sensor

Pada perancangan data sensor ini, peneliti mengambil data sensor dari data primer pengukuran secara langsung menggunakan perangkat sensor suhu, Data sensor di dapatkan secara otomatis disaat perangkat Arduino terhubung secara aktif dengan perangkat sensor. Banyaknya data sensor yang diperolah mengharuskan sistem melakukan perhitungan komputasai terhadap ratusan data dalam waktu yang singkat. Hal ini dapat memperlambat kinerja sistem dalam melakukan proses pengolahan data. Perancangan data sensor ini bertujuan mendapatkan data tunggal atau data serial karena pengiriman data jenis ini dilakukan satu per satu melalui perhitungan matematis. Ketika data tunggal sudah diperoleh selanjutnya data ini akan dikirimkan ke perangkat router sebagai perantaranya. Berikut ilustrasi perancangan data sensor sesuai dengan gambar 3.4.



Gambar 3. 4 Tahap Perancangan Data Sensor

Keterangan: ED1 = Kelompok data dari End Device 1

ED2 = Kelompok data dari End Device 2

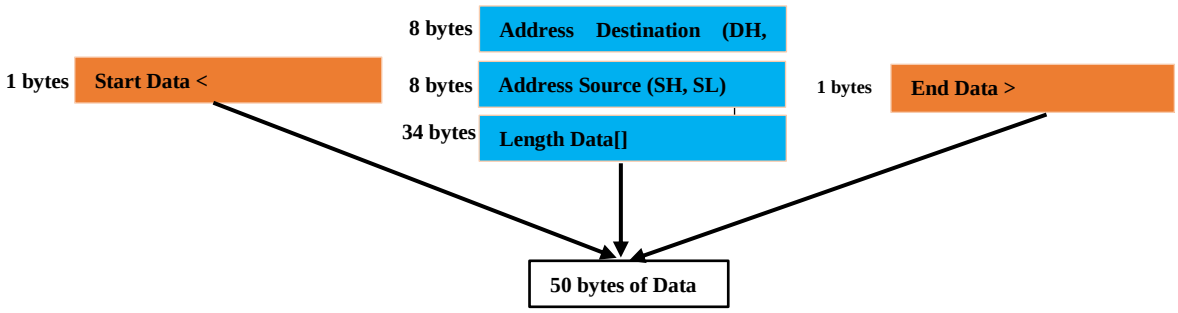
ED3 = Kelompok data dari End Device 3

Data sensor yang diperoleh dari pengukuran masih dalam kategori data mentah yang harus diolah lebih lanjut agar lebih sederhana. Dari sejumlah data yang direkam oleh sistem, dilakukanlah penjumlahan data dari baris pertama kolom

pertama sampai dengan baris akhir kolom akhir, dimana hasil akhirnya didapatkan total nilai / ∑ dari tiga sensor. Selanjutnya sistem melakukan perhitungan *mean* (rata-rata) dari total nilai yang didapatkan sebelumnya guna mencari nilai akhir dari tahap perancangan data sensor.

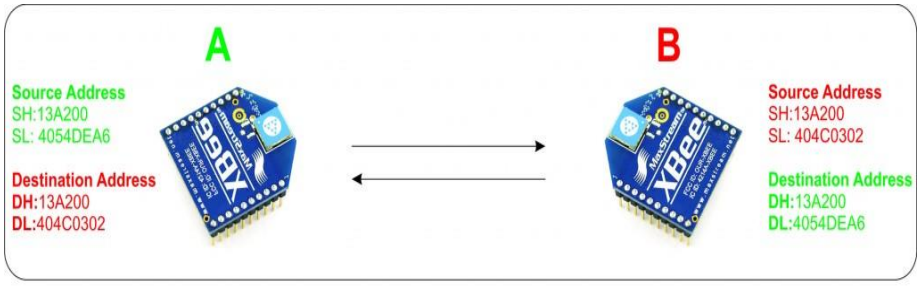
3.3.2.1 Payload Paket Data

Protokol data merupakan aspek penting dalam mendukung kinerja pada saat melakukan pengiriman data dan pengukuran. Di dalam protocol data terdapat *payload* paket data yang meng-enkapsulasi semua karakter mulai dari data sensor dari ketiga end device, variabel *dependen* dan *independent*, variabel global, library dari arduino, variable alamat pada *Xbee Router* dan coordinator yang disimpan pada *microcontroller arduino* dalam bentuk struktur paket data. Struktur ini juga untuk mempermudah proses pengiriman data, monitoring data, dan juga memberikan informasi tentang jumlah karakter yang dikirim dalam satu waktu.



Gambar 3. 5 Payload Paket Data

Pada gambar 3.5 telah dideklarasikan total ukuran *payload* dalam satu paket sebesar 50bytes, dimana 50bytes ini terdiri dari pembagian variabel data sensor dari end device, variabel global penyimpan alamat *Xbee Router*, panjang data karakter, dan penanda awal maupun akhir yang berupa tanda <>. Dengan adanya perancangan *payload* paket data memberikan kemudahan dalam menitoring paket dan memberikan informasi identitas paket, waktu pengiriman paket, nomor urut paket, dan total karakter yang dikonversikan ke dalam bit/karakter. Semua *payload* paket data disimpan dalam buffer *microcontroller* ardiuno.



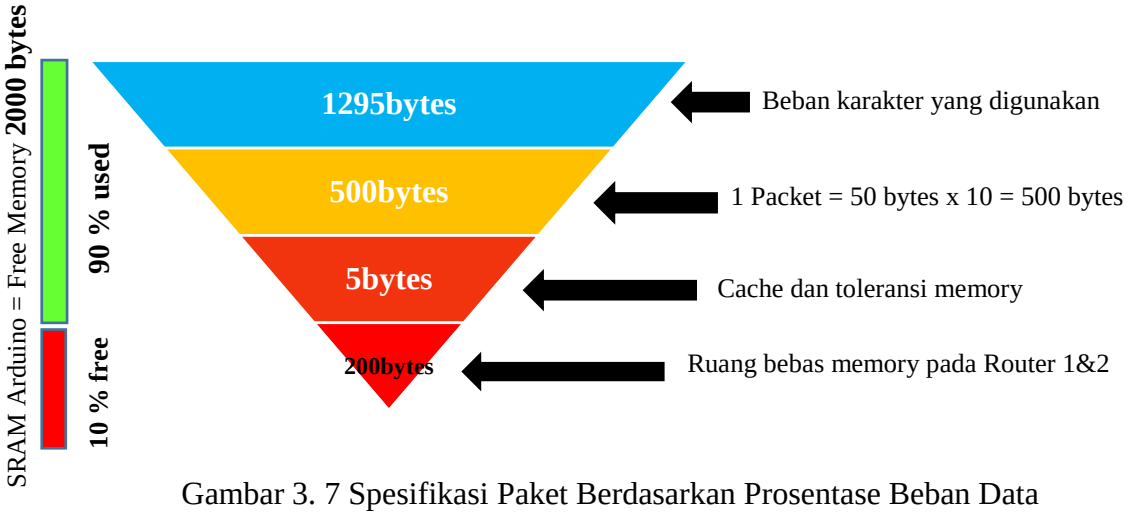
Gambar 3. 6 Ilustrasi Alamat pada Xbee

Pada gambar 3.6 panjang data pada protocol data ini sebesar 50 Bytes yang terdiri dari *header, net address, data, dan tail*, biasanya Xbee hanya mampu mengirimakan maksimum 84 Bytes dalam sekali pengiriman (Hartawan and Desnanjaya, 2018). Gambar 3.7 memberikan ilustrasi alamat pada masing-masing Xbee yang termasuk kedalam sruktur data *payload* yang disimpan pada fungsi *buffer di microcontroller* dengan tujuan mengontrol *payload* saat proses pengiriman data. Pada router 1 dan router 2, mampu menyimpan sampai 35 paket data. Namun sesuai dengan kebutuhan dasar system balancing yang diusulkan dalam penelitian ini, router 1 dan router 2 dibatasi dengan maksimal 10 paket yang diterima sebelum melakukan switching ke router berikutnya. Artinya 10 paket data jika dikonversikan dengan protokol data maka hasilnya sebagai berikut;

- 100 packet x 50 Bytes = 5000 Bytes
- 5000 Bytes = 40.000 bit

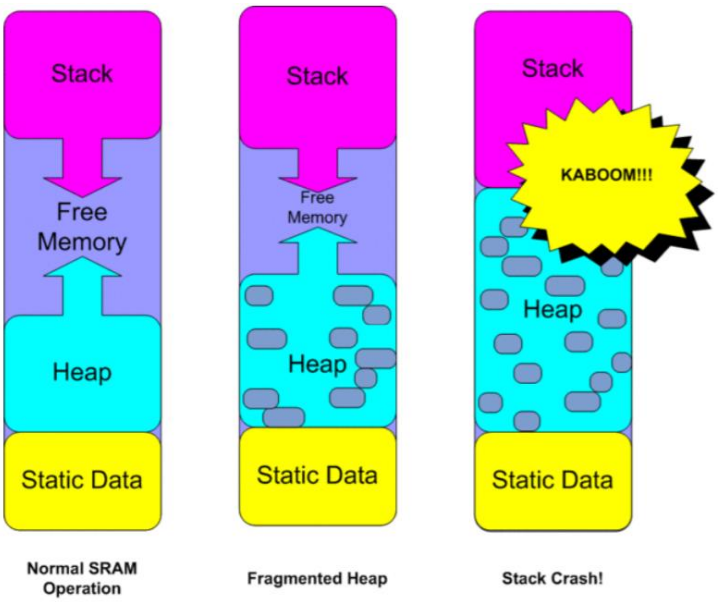
3.3.2.2 Spesifikasi Kebutuhan Buffer Packet Data

Dalam melakukan proses pengiriman paket data, perlu diperhatikan kebutuhan dasar dalam menentukan kapasitas memory yang dibutuhkan untuk perancangan data sensor beserta paketnya. Hal ini mengingatkan pada standart dari perangkat wireless yang memiliki keterbatasan memory, sehingga perlu dilakukan perhitungan spesifikasi kebutuhan buffer paket data yang digunakan, masalah ini diharapkan mampu menjawab pada sub bab sebelumnya tentang mengapa pada router hanya dibatasi 10 paket data. Berikut ilustrasinya;



Gambar 3. 7 Spesifikasi Paket Berdasarkan Prosentase Beban Data

Pada gambar 3.7 adalah gambaran dari total storage yang digunakan dalam penelitian ini, jika mengacu pada Xbee router, maka storge yang tersedia hanya 2000 bytes sesuai dengan yang ada pada *microcontroller arduino uno.* Penggunaan beban terbagi menjadi 3 bagian. Bagian pertama adalah penggunaan pada arsitektur program secara keseluruhan, penggunaan kedua yakni seperti pada struktur *payload* gambar 3.5 dibutuhkan sejumlah 50 bytes untuk 1 paket, kemudian dalam pengiriman paket data pada router nantinya akan dikontrol maksimal 10 paket, hal ini bertujuan agar system transmisi berjalan stabil, tidak sampai terjadi *collision.*



Gambar 3. 8 Perbedaan Penggunaan Memory Arduino (Earl, 2018)

Kondisi penggunaan memory diilustrasikan dengan tiga kejadian sesuai pada gambar 3.8, dimana pada kondisi *Normal SRAM Operation,* kapasitas memory masih memiliki ruang bebas 40-50% dari total memory. Hal ini sangat direkomendasikan dari pakar proyek elektronika yang menggunakan modul *microkrontroller arduino* karena untuk menghindari *crash* saat running system. Selanjutnya pada kondisi *Fragmented heap* kapasitas memory memiliki ruang bebas 10-20 % dari total memory, dimana pada kondisi ini sesekali system akan terjadi *hang* pada saat running yang berkepanjangan, hal ini dikarenakan ruang komputasi pada microkontroller sudah mendekati batas maksimal. Dan pada kondisi *Stack Crash* kapasitas memory berada diantara 2-5 % atau bahkan tersisa 1 % saja, tentu hal ini akan menyebabkan mudahnya system terjadi *crash* dan *stack* sehingga akan menurunkan performa dari system tersebut, kondisi yang seperti sangat tidak direkomendasikan oleh pengguna modul *microkrontroller arduino.*

3.3.3 Perancangan Wireless Communication

Perancangan *wireless communication* menjelaskan bagian dan aspek pendukug dalam membangun komunikasi media wireless dalam hal ini protokol *Zigbee*. Dalam melakukan perancangan komunikasi wireless terdapat dasar-dasar yang harus dilakukan dalam mengatur komunikasi yang diingingkan. Berikut aspek pendukung dalam merancang komunikasi wireless zigbee, Pengalamatan (*addressing*), protokol komunikasi, *mode network* dan *mode operation* metode routing dalam menghubungkan titik dalam jaringan tersebut, topologi jaringan, wireless *data transmission* dan skenario komunikasi wireless yang di rencanakan.

3.3.3.1 Addressing

Alamat perangkat XBee mirip dengan alamat pos dan email untuk orang. Beberapa alamat sifatnya unik, seperti alamat email, tetapi yang lain tidak. Misalnya, beberapa orang dapat hidup secara bersamaan alamat pos. Setiap perangkat XBee dikenal oleh beberapa alamat berbeda, yang masing-masing memiliki tujuan. Terdapat tiga tipe dalam pengalamatan Xbee yang sesuai dengan table 3.1.

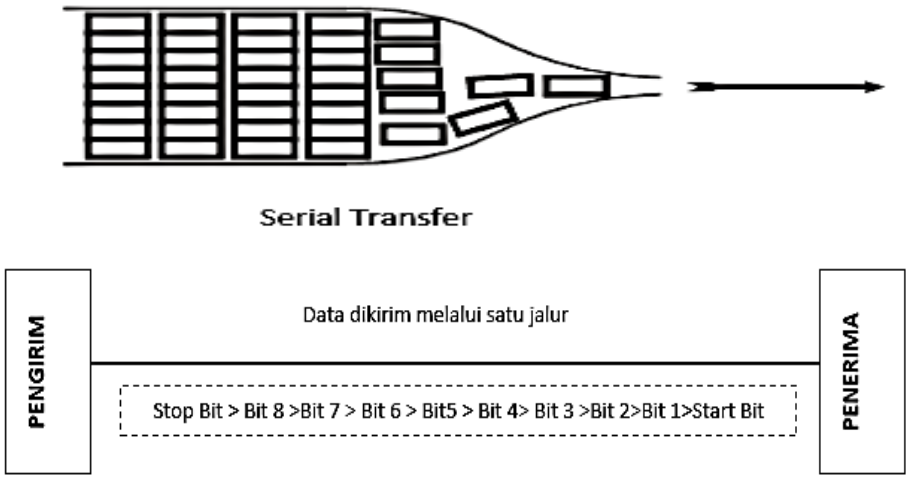
Table 3. 1 Tipe Pengalamatan Zigbee Network

Type	Example	Unique
64-bit	0013A200403E0750	Yes, always and everywhere
16-bit	23F7	Yes, but only within a network
Node identifier	Xbee's radio	Uniqueness not guaranteed

Semua perangkat ZigBee memiliki dua alamat yang berbeda, yaitu alamat 64-bit dan alamat 16-bit. Source Address unik 64-Bit IEEE ditetapkan oleh pabrik dan dapat dibaca dengan perintah SL (*Serial Number Low*) dan SH (*Serial Number High*). Sedangkan pengalamatan 16-bit harus dikonfigurasi secara manual. Pada mode pengalamatan 16-bit digunakan *range* alamat dari *0 – 0xFFFF*. Modul akan menggunakan alamat 64-bit jika nilai pada *Source Address DL* (mode alamat 16-bit) adalah "*0xFFFF*" atau "*0xFFFE*". Untuk mengirimkan paket ke spesifik modul menggunakan alamat 64-bit, *Destination Address* (DL+DH) dari pengirim harus disesuaikan dengan *Source Address* (SL + SH) dari modul tujuan. Sedangkan untuk mengirimkan paket data menggunakan alamat 16-bit, pada sisi *Destination Address Low* (DL) disesuaikan dengan Source Address (SL) pada modul tujuan dan untuk nilai *Destination Address High* (DH) diatur '0'. Paket data XBee dapat dikirim secara *Unicast* atau *Broadcast.*

3.3.3.2 Serial Interfacing

Aspek penting lainnya dalam merancang komunikasi baik *wireless* ataupun *wired* adalah tipe komunikasi yang digunakan, dalam dunia elektronika dikenal dengan tipe pengiriman serial atau pararel dalam hal ini (SPI dan UART). Modul XBee dapat beroperasi sebagai perangkat yang berdiri sendiri atau dapat dipasang ke perangkat yang cerdas, Saat beroperasi sebagai perangkat yang berdiri sendiri, modul XBee hanya mengirim data sensor ke node pusat. Ketika modul XBee terhubung ke perangkat cerdas (seperti komputer, Arduino, atau *Raspberry* Pi) menggunakan komunikasi serial.



Gambar 3. 9 AT Serial Communication (Muamar and Radiannor, 2018)

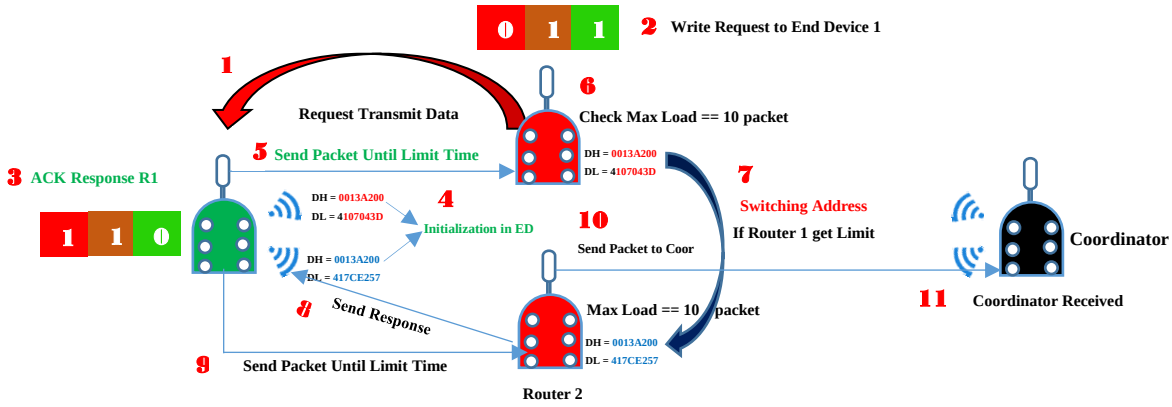
Pada gambar 3.9 penggunaan port serial atau terminal seri adalah port yang menggunakan teknik *interfacing* secara seri. Dalam teknik ini masing-masing bit data dikirim secara berurutan (serial), sehingga dalam satu detak (satuan waktu) hanya 1bit data yang dikirim, lalu data berikutnya sampai semua data yang akan dikirim sudah terkirim. Serial port memberikan kecepatan yang sangat lambat dan telah digantikan oleh USB dan *interface* lain yang lebih cepat untuk koneksi perangkat dengan komputer desktop. Meski masih banyak digunakan dalam akuisisi data, serial port tidak lagi ditemukan pada komputer baru.

3.3.3.3 Handshaking Communication

Handshaking adalah sesi komunikasi data yang berlangsung dari mulai perencanaan komunikasi sampai dengan komunikasi tersebut selesai. Proses *handshaking*, diawali oleh proses pra komunikasi yaitu proses perncarian host tujuan (destination) oleh host yang bertindak sebagai pengirim. Proses ini diakhiri dengan kesepakatan kedua belah pihak untuk melaksanakan pertukaran data *(connection establish)* Proses selanjutnya adalah proses *connection establish*, merupakan proses inti yaitu proses pengiriman informasi berupa request dan tanggapan antara kedua belah pihak (Muamar and Radiannor, 2018).

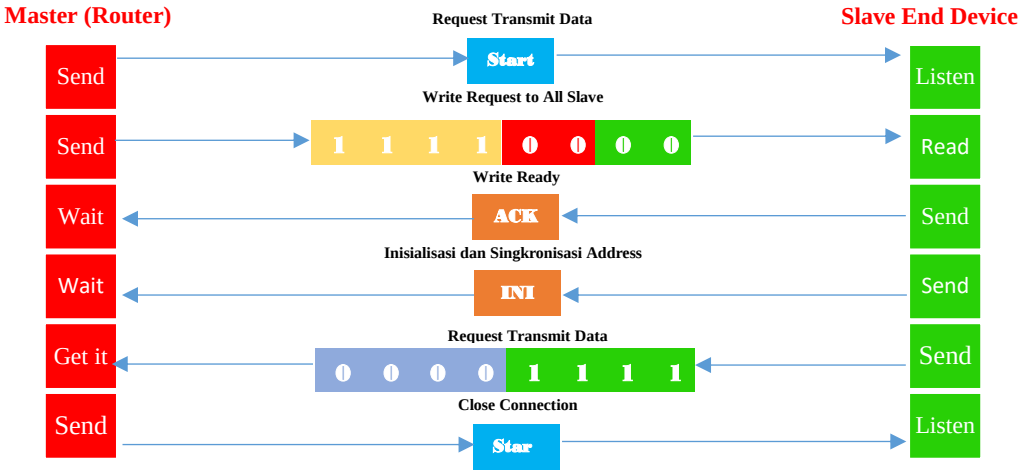
Seperti pada beberapa jaringan, setiap node jaringan membutuhkan untuk memiliki ID yang unik untuk efisiensi dalam komunikasi data. Node ini

merepresentasikan kemampuannya untuk dapat mengenali node tujuan. Dalam waktu yang sama pula, node tersebut harus dapat mengenali *broadcast* ID yang tidak berfungsi sebagaimana yang telah ditentukan di awal kesepatakan komunikasi, dengan kata lain node ini juga harus mampu untuk mengenali ID yang bukan tujuan prioritasnya (Ahmed, 2012).



Gambar 3. 10 Handshaking Master – Slave

Peran teknik *handshaking* disini sangat penting, dengan teknik ini pengkondisian transmisi data dapat dikontrol dengan teratur. Teknik *handshaking* ini telah diterapkan pada system balancing dimana router sebagai master nya dan perangkat end device sebagai slave nya. Pada gambar 3.10 adalah gambaran umum handshaking dijalankan mewakili penggunaan tiga mode Xbee dari perangkat end device ke perangkat coordinator.



Gambar 3. 11 Proses Handshaking

Dua proses awal ini (pra komunikasi dan *connection establish*) dapat disebut dengan proses pembentukan koneksi. Artinya, untuk melakukan komunikasi, perangkat yang dituju harus menerima koneksi awalan terlebih dahulu sebelum mengirimkan atau menerima data.

Proses *handshaking* sesuai dengan gambar 3.11 diawali dengan router melakukan permintaan (request) ke host tujuan untuk mengirimkan paket data dalam hal ini adalah perangkat end device 1, 2, dan 3. Disaat yang bersamaan router melakukan indentifikasi *network* ke end device yang berada dalam *channel* yang sama. Setelahnya router melakukan *request* dengan mengirimkan format *request* dalam bentuk *interface serial*. Selanjutnya end device melakukan inisialisasi *request* dengan *router*, di proses ini end device melakukan konfirmasi dan kesepakatan antara router dan end device dengan mengenali *ID broadcast* dan *MAC network*, dalam Xbee dikenal dengan *Serial High* (SL) + *Serial Low* (SL), dan *Destination Address High* (DH) + *Destination Address Low* (DL). Dan proses terakhir dimana pada alamat ID end device yang sudah ditentukan mengirimkan paket data sesuai dengan spesifikasi dari router, pada gambar 3.11 hanya menggambarkan satu kali siklus saat system ini melakukan handshaking transmisi data.

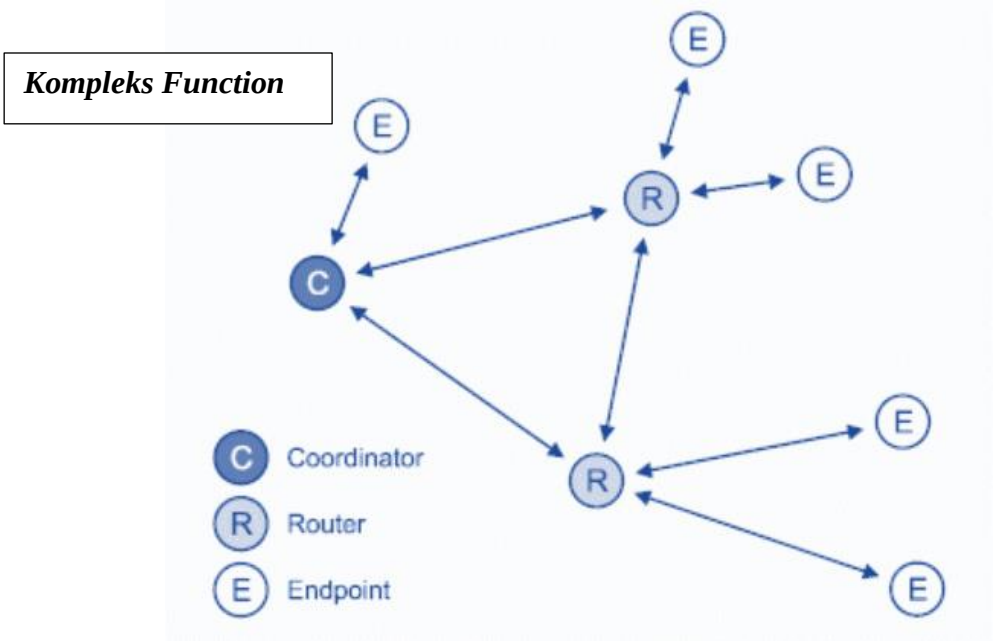
3.3.4 Perancangan Routing Wireless Sensor Network

Perancangan penelitian pada arsitektur *wireless sensor network dibagi* menjadi tiga bagian. Pertama penulis melakukan table Routing Xbee, kedua melakukan proses routing jaringan mesh.

3.3.4.1 Zigbee Mesh Routing Protocol

Ada banyak protocol routing untuk WSN, namun hal terpenting dalam routing protocol khuhusnya pada protocol Zigbee adalah kebutuhan dasar jaringan agar dapat memperpanjang *lifetime of network*, dan dapat dengan mudah diimpelementasikan dalam node yang menggunakan teknologi saat ini (Mihajlov and Bogdanoski, 2011). Struktur protocol routing yang digunakan dalam penelitian ini adalah *Zigbee Mesh routing* protocol berdasarkan routing AODV. Routing protocol AODV *(Ad hoc On demand Distance Vector)* adalah routing protocol yang

dirancang untuk jaringan *ad hoc mobile* yang termasuk kategori reactive routing protocol. Routing protocol ini menyimpan informasi routing seputar path, host yang aktif dan disimpan di semua node. Didalam AODV, ketika node asal ingin mengirim packet ke tujuan namun tidak ada route yang tersedia, node tersebut akan memulai proses *route discovery*. AODV memiliki komponen utama seperti, *sequence number, ID number, RREP, RREQ, hop count, data message*. Komponen tersebut sangat mendukung pada saat kondisi routing melakukan *route discovery*, dan *route maintenance*. Berikut ini kami berikan ilustrasi penerapan dari *route discovery* dan *route maintenance.*

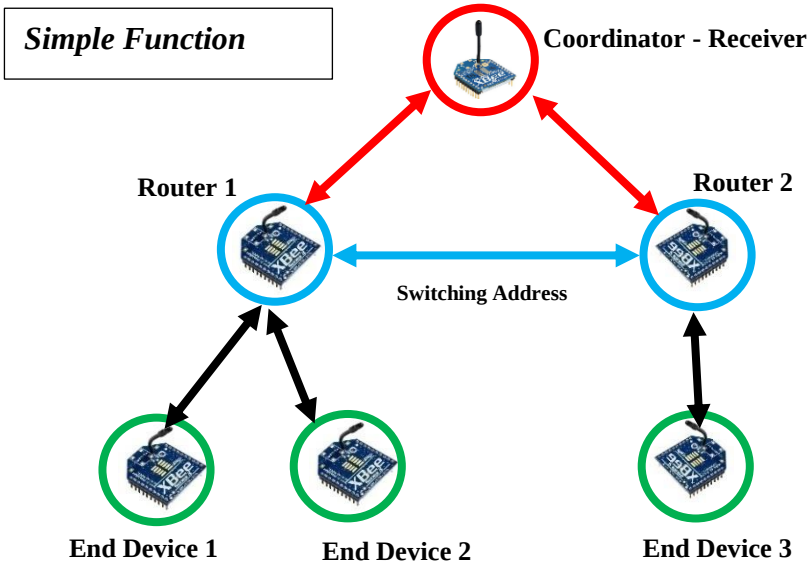


Gambar 3. 12 Skema Zigbee Mesh Routing (Bricker and Harris, 2007)

Gambar 3.12 adalah Skema Zigbee Mesh Routing ini mengilustrasikan desain jaringan dengan menggunakan *multi router*. Terdapat tiga perangkat Zigbee yakni, *Coordinator*, *Router*, dan endpoint atau end device. Topologi Mesh routing memberikan keuntungan dari sisi kehandalan (reliability) routing discovery (memperbaiki rute yang rusak), dan menajemen traffic distribusi data yang lebih stabil, hal ini dikarenakan adanya router yang bertugas sebagai mengatur dan melakukan kontrol terhadap akses pada end device dan coordinator.

3.3.4.2 Mesh Routing Rule

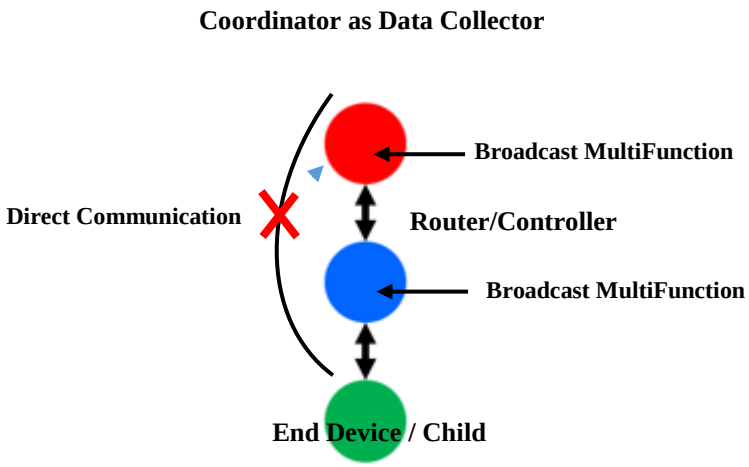
Pada gambar 3.13 merupakan skema dari routing Xbee yang masing masing menggunakan satu sumber end device dan dua sumber end device. Disaat kondisi *router* 1 *off*, maka *route* yang akan dilalui dari data end device dikirim ke *router* 2, dan *router* 3 selanjutnya dikirim menuju coordinator. Dalam hal ini *routing protocol* sangat diperlukan dalam komunikasi jaringan mesh, Nomor di setiap node diibaratkan sebagai suatu alamat yang dapat dikenali oleh *node* yang lainnya. Dari nomor tersebut *node* dapat mengirim *request* (*RREQ*) atau menerima *response (RREP).*



Gambar 3. 13. Skema Mesh Routing

Pada gambar 3.13 *routing protocol* mencoba untuk menerapkan *self-healing* pada jaringan mesh yang menggunakan tiga sumber data dikirim dengan route yang berbeda. Terdapat dua *router*, tiga *source* end device dan satu coordinator. Pada coordinator bertugas sebagai control pusat yang menerima *request* dan *respon* dari *router* dan *source*, sedangkan pada *router* memiliki tugas sebagai penghubung antara coordinator dan s*ource*, dan pada *source* hanya melakukan *request* pengiriman data dari sensor untuk disampaikan ke router kemudian ke coordinator. Masing masing garis penghubung memiliki rute yang sesuai dengan penomoran skenario routing. Routing ini memudahkan dalam menentukan pilihan rute yang dilalui oleh paket data. Perbedaan *Mesh routing*

dengan star routing, ataupun dengan pair routing, adalah pada bagian kontrol jaringan yang menggunakan router. Berikut ilustrasi hirarki komunikasi mesh routing.



Gambar 3. 14 Hirarki Mesh Transmission

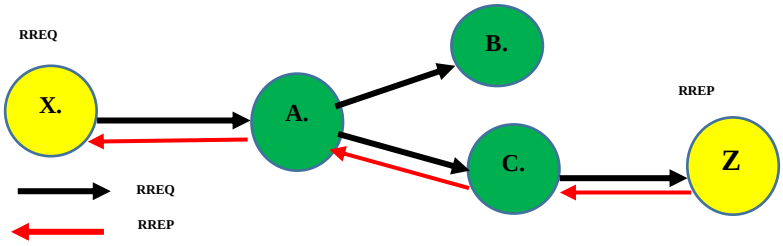
Berdasarkan pada gambar 3.14, bahwa pada *Mesh Routing* terdapat tingkatan dalam berkomunikasi (Sharma, Verma and Sharma, 2013). Sebagai contoh, jika end device ingin berbicara langsung dengan coordinator, maka ini tidak diperbolehkan, sebab pada *child* atau end device hanya mengenal rute yang ditujukan pada controller yaitu router. Pada konfigurasi XCTU Xbee diperkenankan menyetting alamat dengan dua pilihan, *0X000 address* atau *0xFFFF address*. Jika tujuan dari komunikasinya untuk menghubungkan jalur komunikasi *peer to peer* maka menggunakan alamat *0x0000*, dimana alamat ini secara default membangun komunikasi antara end device dan coordinator, dengan kata lain saat end device mengirimkan data maka data tersebut akan sampai ke alamat coordinator secara default, walaupun tanpa menginisialisasi alamat dari kedua perangkat tersebut. Dan jika pada konfigurasi XCTU perangkat end device menyetting alamat *0XFFFF* maka mode komunikasinya menjadi mode *broadcast*, ini artinya jika end device mengirimakan data maka semua perangkat yang berada dalam satu *channel* dan alamat jaringan yang sama, maka data ini akan dapat diterima oleh router maupun coordinator, sekalipun ada perangkat end device tambahan, maka *perangkat* end device pun dapat mendengar komunikasi ini.

Table 3. 2 Mesh Routing

NO	SKENARIO	SOURCE	KONDISI ALTERNATIF ROUTING
1	Node *Router* 1 Off	Source 1	*Source 1 -> router 2 -> coordinator*
4.	Node *Router* 2 Off	Source 2	*Source 2 -> router 1 -> coordinator*

3.3.4.3 Route Discovery

Route discovery adalah kondisi system menemukan informasi rute baru. Dalam melakukan *route discovery* node asal melakukan *broadcast route request (RREQ)* ke tetangga terdekat untuk mengetahui nomor *sequence* ID tujuan.



Gambar 3. 15 Route Discovery

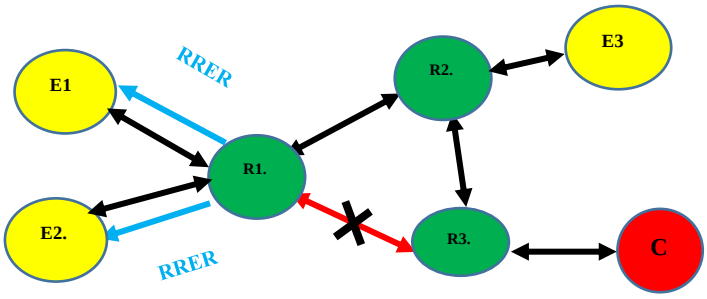
Pada gambar 3.15 memberikan ilustrasi tentang *route discovery* dengan menggunakan metode AODV, dimana tanda *X (source node)* dan tanda Z adalah *(destination note).* Ketika route tujuan menerima *request* dari node asal, node tersebut memeriksa nomor ID untuk dilakukan verifikasi ulang. Untuk memastikan bahwa paket tersebut valid, node tujuan membalas paket RREQ dengan *replay route* (RREP) paket. Berikut ini adalah langkah dalam menemukan rute dalam zigbee routing.

1. Node X need a route to Z
2. Create a router request (RREQ)
   - Set X's ID as source, set Z's ID as destination, X's length[] or type data, X's data
3. Node X broadcast RREQ to neighbors
4. Node A receives RREQ
   - Makes a reverse route entery for X
   - If stil hasn't route to Z, so it reboardcasts RREQ

5. Node C receives RREQ
    - Makes a reverse route entery for X
    - It has route to Z,
    - Broadcast route to Z in RREQ
6. Node Z receives request from X's paket.
7. Node Z send reply to update route information

3.3.4.4 Route Maintenance

Route maintenance adalah kondisi system melakukan perbaikan rute yang mengalami kerusakan. Permasalahan ini biasanya jika sebuah *link* ke hop *(destination node)* berikutnya tidak dapat terdeteksi dengan menggunakan metode penemuan rute, maka link tersebut akan mengirimkan pesan *route error (RRER)* ke node tetangganya sampai tersebar ke *source node*. Berikut ini adalah langkah dalam melakukan *route maintenance* dalam *routing zigbee.*



Gambar 3. 16 Route Maintenance

Pada gambar 3.16 adalah ilustrasi dari skema *route maintenance* dengan menggunakan metode AODV, dimana tanda X *(source node)* dan tanda Z adalah *(destination note).* Saat node tidak dapat meneruskan paket data, maka node tersebut akan mengirimkan pesan *route error* ke alamat asal pengiriman yaitu E1 dan E2. Saat E1 dan E2 menerima *feedback route error* selanjutnya dilakukan pengecekan dan konfirmasi terhadap *route* baru yakni R2. Setelah node E1 dan E2 memverifikasi alamat node baru system menghapus route R3, dan paket dapat dikirimkan melalui *route* yang baru.

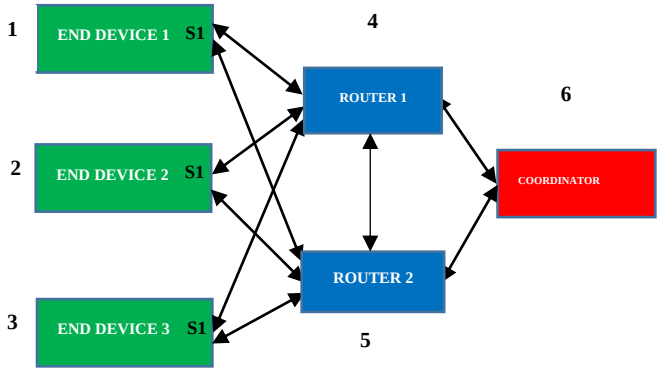
1. Link between R1 and R3 are break

2. Node R1 invalidates route to R3 in route table
3. Node R1 creates Route Error message
    - List all destination that are update unreachable
    - Send to near neightbors
4. Node ED1, and ED2 receive RERR message
    - Check whether R1 is its next hop on route to R3
    - Deletes routes to R3
5. Rediscover route if still needed

3.3.5 Rule Load Balancing

Loab balancing adalah salah satu teknik untuk mendistribusikan beban paket data pada dua atau lebih jalur koneksi secara seimbang dan bergantian. Teknik ini digunakan dalam menerapkan system load balancing disini adalah dengan membagi beban secara bergiliran dan berurutan dari satu router ke router lainnya. Di lain pendapat load balancing merupakan teknik untuk mendistribusikan beban trafik pada dua atau lebih jalur koneksi secara seimbang, agar trafik dapat berjalan optimal, memaksimalkan *throughput*, memperkecil waktu tanggap dan menghindari overload pada salah satu jalur koneksi (Affandi and Setijadi, 2012).

Skenario pertama berdasarkan gambar 3.17 dimana aturan system yang digunakan adalah dengan memastikan bahwa kondisi kedua router aktif. Hal pertama yang dipersiapkan end device sebelum mengirimkan paket data adalah mengecek status router. Berikut ilustrasi model jaringan rule balancing yang dirancang dalam penelitian ini.



Gambar 3. 17.Topologi Rule Balancing Dua Router

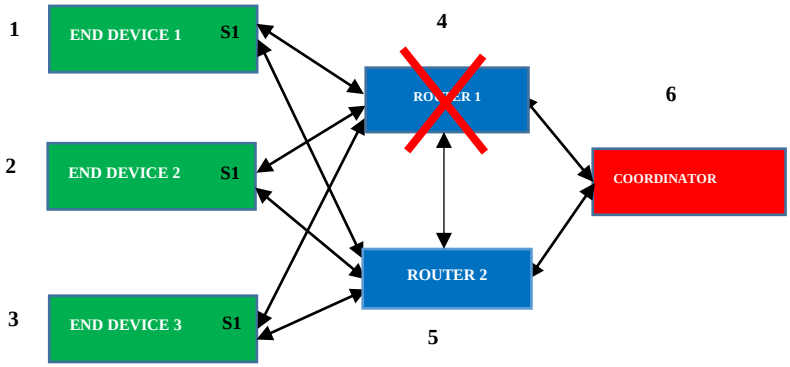
Secara sederhana system yang kita sebut dengan rule balancing bekerja apabila penyimpanan pada router 1 sudah menerima 10 paket data. Sistem balancing secara otomatis akan membagi beban berdasarkan jumlah paket maksimal yang diterimanya. Jumlah beban paket yang diatur pada router adalah 10 paket dengan maksimal 84 Bytes karakter yang dapat sekali terkirim antar Xbee (Hartawan and Desnanjaya, 2018).

Saat routing 1 aktif pengiriman paket data terhubung ke router 1 dan router 2. Jumlah maksimum beban paket yang diatur pada router disesuaikan dengan jumlah maksimum paket dalam satu kali pengiriman maksimum 10 paket, dimana jumlah paket tersebut jika berdasarkan jumlah karakter setara dengan 500 Bytes atau 4800 bits. Maksimal packet pada router 1 dan router 2 diatur dengan jumlah 10 paket yang dikirimkan dari end device menuju coordinator melalui kedua router tersebut. Secara sederhana sistem balancing akan bekerja apabila penyimpanan pada router 1 sudah menerima 10 paket data. Selanjutnya Router 1 meminta request pada end device untuk melakukan *Switching Address* (SA) ke router 2. Tujuan dari SA ini adalah mengalihkan rute yang akan dilalui end device dari router 1 berganti ke router 2. Pengalihan routing ini digunakan sebagai pengalihan transmisi dari semua end device untuk menuju ke routing yang direkomendasikan.

3.3.6 Rule Non-Balancing

Rule non-balancing adalah kondisi tidak adanya suatu pembagian beban transmisi paket data pada dua jalur atau koneksi secara seimbang dan bergantian. Pengiriman paket data non *balancing* ini menerapkan teknik *self-healing* dari *Zigbee Mesh Routing Protocol*, rule non-balancing adalah suatu kondisi tidak adanya suatu pembagian beban trafik pada dua jalur simbang. Peran router disni bertugas sebagai penghubung, dan kontrol dalam mengatur *route* transmisi (Mahmoud, 2013). Pada gambar 3.18 ED1, ED2, ED3 mengirimkan paket data melalui router 1, dan router 2. Pada saat kondisi router 1, dan router 2 menerima paket yang dikirimkan, router 1 dan router 2 hanya menerima paket data dari end device selama proses pengiriman tanpa melakukan proses *limit* beban. Jadi agar router 1 dapat mengetahui alamat dari router 2, mode pengiriman pada kedua router ini diatur sebagai komunikasi *point to poin* dan *point to multipoint* atau dalam

komunikasi jaringan Zigbee komunikasi *Unicast*, dan *Broadcast.* Berikut ilustrasinya dari rule non-balancing yang menggunakan dua router.



Gambar 3. 18 Topologi Rule Non-Balancing Dua Router

Dikondisi ini diasumsikan router 1 mengalami gangguan atau error, sehingga pengiriman paket data harus dialihkan ke router 2 untuk meneruskan pingiriman paket data dari router 1, hal ini dikarenakan untuk menghindari *overload* pada buffer arduino yang digunakan untuk memproses system routing. Sistem *routing* akan memberikan respon dengan mencari route lain yang dapat dilalui. Hanya terdapat dua router yang digunakan dalam implementasi penelitian ini sesuai dengan batasan masalah yang diusulkan. Namun sebagai bahan pendukung dalam penelitian ini, disajikan pula hasil ekperimen menggunakan simulasi dengan kondisi alternati router yang ditambahkan didalamnya, karena jika hanya menggunakan dua router, maka routing tidak memiliki alternatif lain untuk menemukan rute lain saat terjadi masalah dan saat terjadi *overload* pada pengiriman paket. Sehingga dari penelitian ini juga diterapkan dalam bentuk simulasi yang memberikan gambaran saat menggunakan dua router, tiga router dan empat router, agar bisa memberikan skenario tambahan pada penelitian ini saat mengharuskan tambahan routing sebagai alternatif lain.

3.3.7 Skenario Rule System Pembagian Beban (Load Balancing Xbee)

Sistem sekenario ini memberikan informasi tentang *routing* pada system yang terdiri dari penerapan *balancing* maupun *non-balancing*. Informasi *routing* yang disampaikan melalui table ini mewakili kondisi saat pengujian jaringan berdasarkan kondisi router off dan kondisi maksimal beban paket yang diterima.

Maksimal beban yang diterima oleh masing-masing router tidak melebihi 500 karakter (Mahmoud, 2013). Rule system *non-balancing* bekerja pada kondisi router yang mengalamai gangguan sehingga mengharuskan berpindah ke jalur yang lain. Sebaliknya saat kedua router aktif maka rule system balancing yang akan digunakan. Berikut rule table nya.

Table 3. 3 Rule Non-Balancing

No	Rule	Routing	Route Status	Router Maintenance
1	Non -Balancing	Routing 1	Router 1 Off	*Source ED1,2,3 -> router 2 -> coordinator*
		Routing 2	Router 2 Off	*Source ED1,2,3 -> router 1 -> coordinator*

Pada table 3.3 *non-balancing* adalah teknik pembagian beban tanpa menggunakan batas penyimpanan. Sesuai dengan table 3.3 tedapat dua rute yang bisa dilalui paket data yakni routing 1 dan routing 2. Pada routing 1 semua paket data dikirim dari end device menuju coordinator melalui router 2. Pada routing 2 semua paket data dikirimkan dari end device sampai ke *coordinaor* melalui router 1. Agar kondisi non-balancing berjalan efektif, maka masing-masing routing perlu mematikan perangkat router. Hal ini diasumsikan seperti menerapakan *route maintenance* pada jaringan *Zigbee Mesh routing* (Mahmoud, 2013).

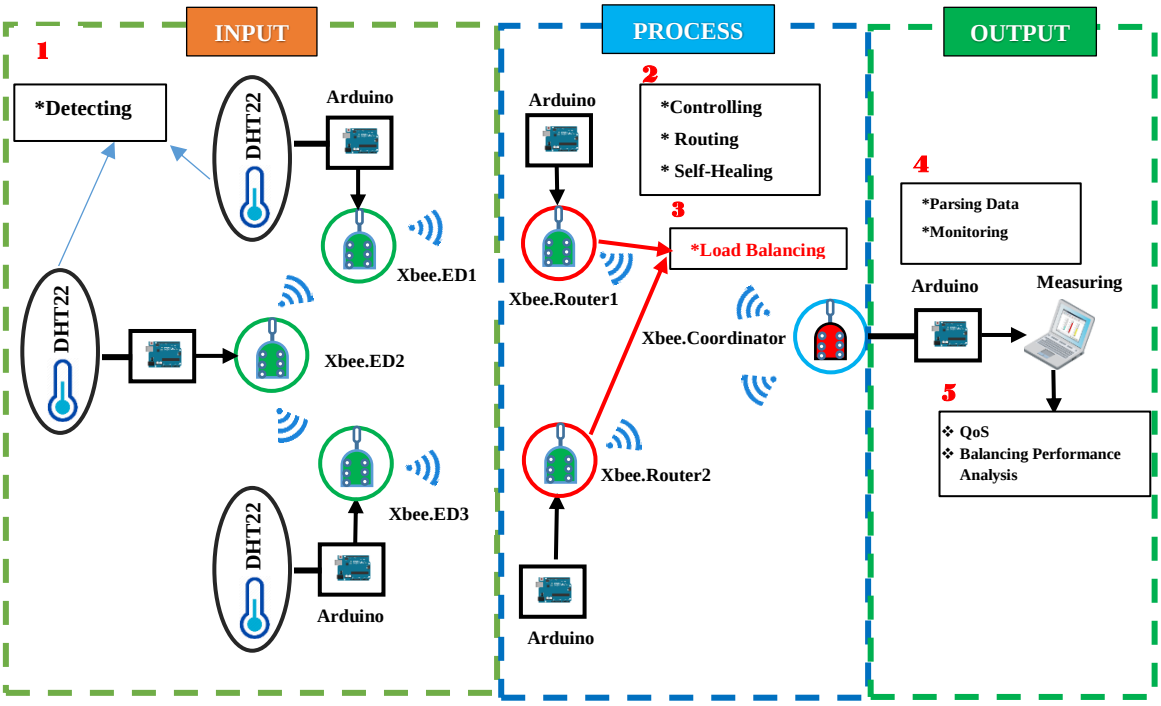
Table 3. 4 Rule Balancing

No	Rule	Routing	Router Max Load	Router Maintenance
1	Balancing	Routing 1	Router 1 = 10 Packet	*Source ED1,2,3 -> router 1 -> coordinator*
		Routing 2	Router 2 = 10 Packet	*Source ED1,2,3 -> router 2 -> coordinator*

Rule balancing adalah teknik membagi beban secara bergiliran dan berurutan dari router 1 ke router 2 (Zhang and Chen, 2014). Berdasarkan table 3.4 dua router mewakili masing-masing Routing. *System Balancing* akan bekerja apabila jumlah paket yang diterima oleh router 1 sudah mencapai batas maksimum, fungsi *switching address* pada router berfungsi untuk memberikan respon pada router 2 untuk meneruskan paket data yang diterima dari router 1. Nilai maksimum yang diperlukan router 1 dan router 2 adalah 10 paket data.

3.3.8 Perancangan Desain Sistem

Perancangan desain sistem ini terdapat tiga alur sistem. Pertama, sistem melakukan pengiriman paket data suhu dari ED1, ED2, dan ED3. Ketiga end device tersebut mendeteksi kondisi yang berbeda sesuai dengan konsentrasi output sensor. Paket data yang dikirim diklasifikasikan kedalam beberapa kelompok sesuai dengan output data sensor. Output data sensor suhu dikelompokkan dengan menggunakan *raw data*. Tujuan dari mengelompokkan output data sensor adalah mencari nilai rata-rata dari setiap sensor sebelum menuju proses selanjutnya, sehingga dapat meringankan beban Xbee dalam proses distribusi paket data ke *router*. Kedua, semua informasi paket data sensor diterima oleh *router* dimana pada proses ini sistem melakukan rekondisi dan pengecekan jalur terbaik dalam distribusi data. Pengecekan dan rekondisi jalur terbaik berdasarkan skenario *routing wireless sensor* digunakan sebagai acuan dalam distribusi data pada Router. Pada router dilakukan proses manajemen jaringan, seperti monitoring status *node*, cek status data sensor, balancing paket dan melakukan *backup* pada salah satu *node* yang tidak tersambung. Selanjutnya data diterima oleh coordinator dan diproses untuk menampilkan *monitoring interface* dari proses manajemen yang dilakukan. Ketiga, data sensor yang dikirim melalui router dan coordinator akan dilakukan uji kualitas paket yang dikirim menggunakan tiga parameter QoS.



Gambar 3. 19 Arsitektur Hardware

Gambar 3.19 adalah ilustrasi arsitektur *hardware* yang sudah diimplementasikan pada penelitian ini. Secara keseluruhan arsitektur ini mewakili bagian terpenting dalam setiap prosesnya. Arsitektur ini dibagi tiga proses bagian, proses pertama yaitu mendeteksi dan mengirimkan data, kedua proses *routing* dan *balancing*, dan ketiga adalah proses pengukuran performa.

Pada gambar 3.19 proses input menggunakan tiga end device sebagai media *wireless* yang langsung terhubung pada Arduino dan sensor DHT22. Setiap data input yang diterima *microcontroller* arduino dikirimkan dengan menggunakan komunikasi serial. Panjang data dalam sekali pengiriman sudah ditentukan dan dibatasi. Ukuran panjang data [*leghtdatastore*] sudah ditentukan pada masing-masing end device. Jumlah karakter data yang digunakan adalah 34 Bytes, dan 50 Bytes dari router ke coordinator, karena perangkat Xbee hanya dapat mengirimkan paket data dalam sekali transmisi sejumlah 256 Bytes (Mahmoud, 2013), maka perlu adanya penjadwalan waktu kirim antar perangkat Xbee (Zhang and Chen, 2014). Penjadwalan pada system ini bertujuan untuk mengatur dan mengontrol saat proses transmisi data antar perangkat Xbee agar tidak terjadi *collision* (Sabilla, Sarno and Effendi, 2018).

Pada bagian kedua adalah inti proses yang kami lakukan dalam mengembangan kebaharuan system pada perangkat router. Digunakan dua modul Xbee sebagai router untuk mengontrol system balancing, hal ini telah mewakili fungsi router sebagai perangkat *fungsi full* atau (FFD) (Mahmoud, 2013). Jika dalam penelitian sebelumnya yakni (Soijoyo and Ashari, 2017), (Rachman, Yanti and Hidayati, 2017), dan (Piyare and Lee, 2013) telah dibahas dan digunakan perangkat Xbee router untuk transmisi data, namun penelitian meraka belum membahas secara detail performa dari perangkat router yang mampu melakukan *balancing* dan *self-healing*. Oleh karena itu pada bagian Process, kami melakukan tiga skenario perangkat router yakni sebagai *routing*, *self-healing, dan load balancing*.

Bagian ketiga adalah proses *Output*, dimana dilakukan pengukuran performa WSN dengan menggunakan parameter QoS yang terdiri dari parameter *Throughput*, *Packet Loss*, dan *Delay* (Sugeng *et al.*, 2015). Selain itu proses akhir pada system ini terletak pada Xbee coordinator yang membangun jaringan Zigbee

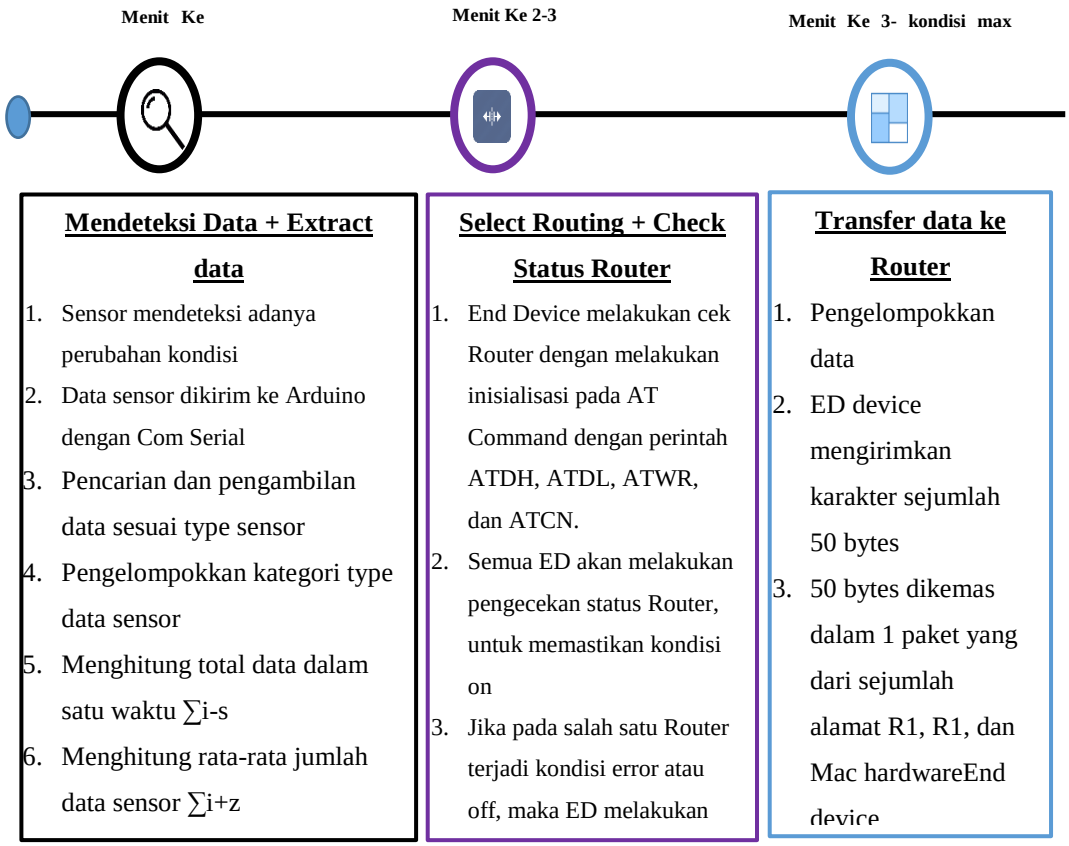
dengan memindai ID PAN pada channel yang tersedia. Pada gambar 3.19 menjelaskan skema komunikasi antar Xbee, dimana coordinator berperan dalam melakukan *request* dan *respon* kepada end device melalui router.

3.3.9 Skenario Penelitian

Sebelum melakukan implementasi penelitian sebaiknya perlu memaparkan sebuah skenario penelitian. Dalam skenario penelitian materi yang dibahas adalah penjelasan detil mengenai langkah demi langkah alur sistem dalam melakukan proses yang dimulai dari menit pertama sampai menit akhir. Berikut ini terdapat tiga skenario penelitian yang nantinya akan dilakukan dalam penelitian ini,

3.3.8.1 Skenario Input (End Nodes)

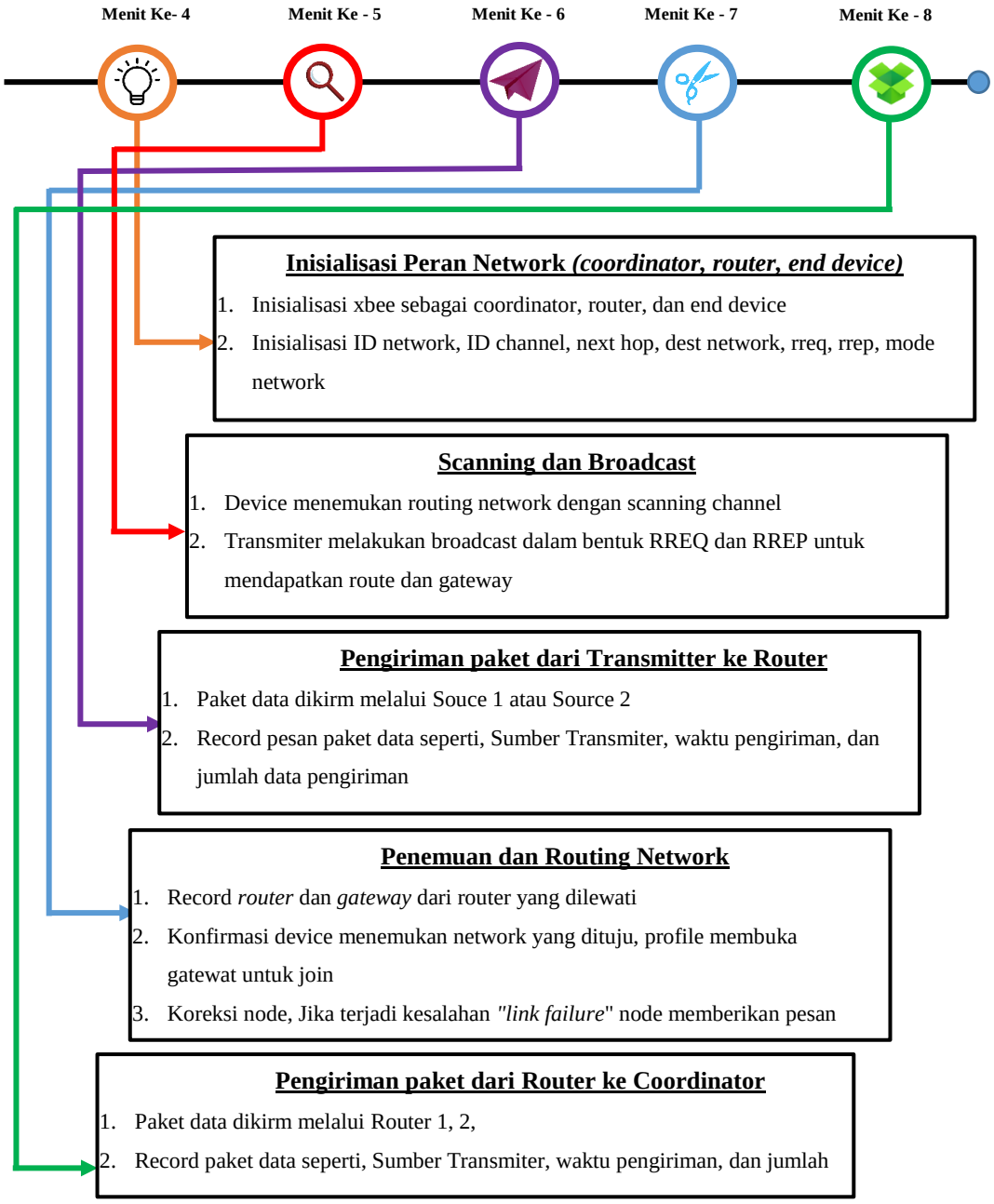
Diawali dengan skenario input terdapat tiga langkah sistem dalam melakukan proses input, pada menit pertama sistem mendeteksi data, menit ke dua sistem melakukan klarifikasi data, dan menghitung *raw data* yang diterima dari



Gambar 3. 20 Skenario Input (End Nodes)

3.3.8.2 Skenario Proses (Routing)

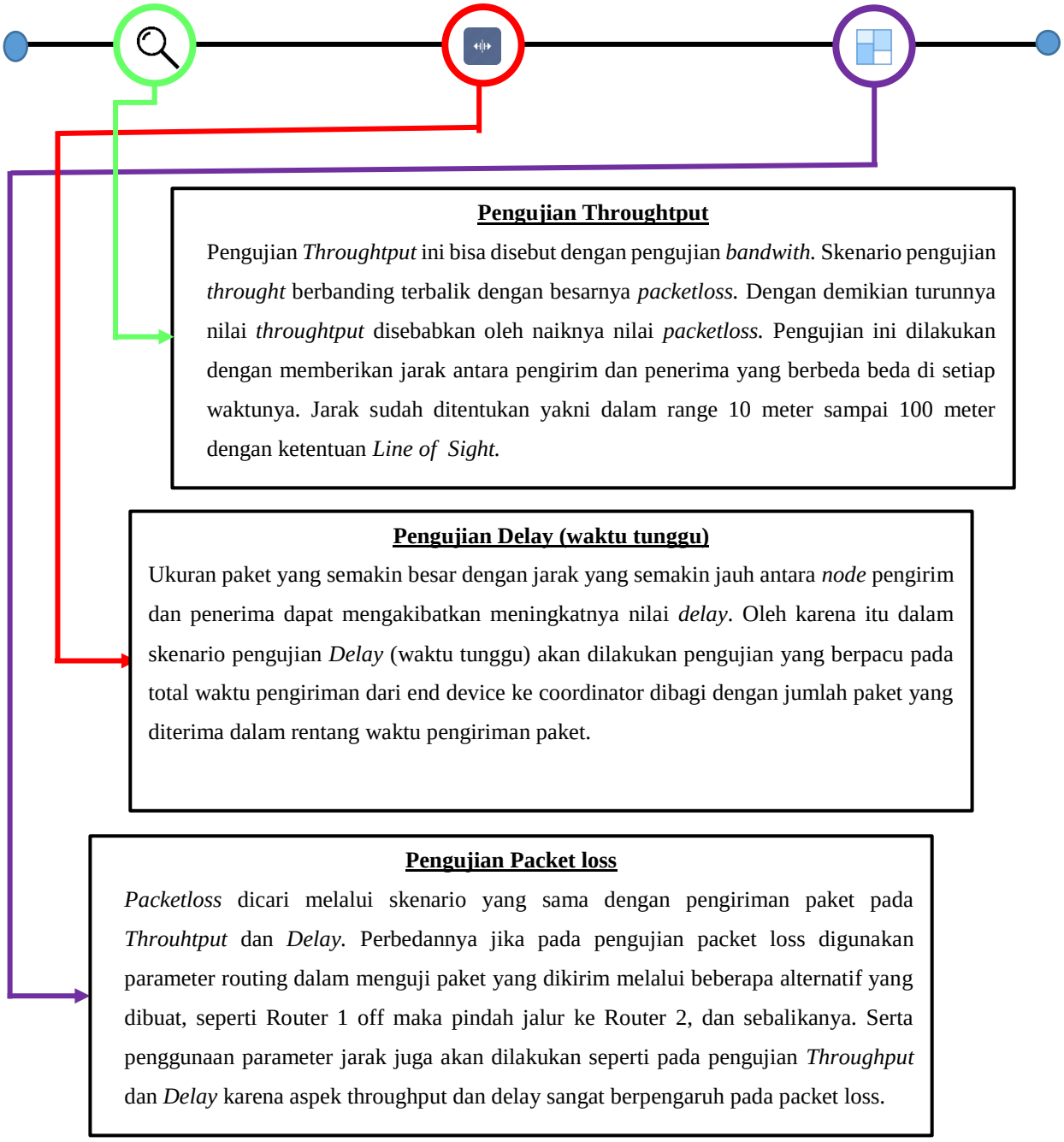
Skenario kedua adalah proses Routing, dimana pada skenario ini terdapat lima langkah proses untuk melakukan persiapan dan pengiriman pada Router. Dimulai router melakukan request ke pada ketiga end device sampai melakukan proses *route discovery*, dan *route maintenance* saat terjadi masalah. Berikut adalah gambar 3.21 skenario proses dari router.



Gambar 3. 21 Skenario Proses (Routing)

3.3.8.3 Skenario Ouput (Performa Testing)

Pada gambar 3.22 merupakan skenario ketiga yakni proses performa testing atau bagian pengujian kualitas data sensor yang telah melewati proses routing. Paket data yang akan dikirimkan ke coordinator dikumpulkan dan diparsing untuk diketahui total paket yang berhasil terkirim dan total paket yang hilang.



Gambar 3. 22 Skenario Output (Performa Testing)

3.4. Quality of Service Method Performance

QoS atau disebut dengan *service quality* adalah sebuah mekanisme yang mengizinkan aplikasi atau layanan jaringan dapat beroperasi sesuai dengan harapan. Service *quality* ini memberikan jaminan performa di dalam jaringan. Dalam menjamin performa jaringan yang lebih unggul, diperlukan parameter jaringan untuk dapat mengevaluasi setiap persyaratan yang diperlukan dalam skenario, berikut ini acuan beberapa parameter performa dalam *Quality of Service*, *Delay Tolerance* (delay), *Loss Tolerance* (Packet Loss), *Capacity* (*Throughtput*), *RSSI* dan *Fault Tolerance* (Kumbhar, 2016).

3.4.1 Packet Loss

Packet Loss adalah kegagalan transmisi paket ke tujuannya. Kehilangan paket terjadi ketika satu atau lebih paket data yang melewati jaringan gagal mencapai tujuannya.

$$\text{Packet Loss} = \left(\frac{\sum_{i=T}^{i=T_{t+1}} D_i}{\sum_{i=T}^{i=T_{t+1}} S_i} \right) x100; 0 \le t \le T \quad (1)$$

Dimana ∑Di adalah jumlah paket yang hilang, dan ∑Si adalah jumlah paket yang sukses terkirim, dan Tt adalah *sampling time (s).* Untuk dapat mengetahui jumlah paket yang hilang dalam satu kali pengiriman maka paket perlu di kalikan dengan nilai 100 %, hal ini merupakan standar nilai dari packet loss.

3.4.2 Delay

Delay adalah total waktu tunda paket yang disebabkan oleh proses transmisi dari satu titik ke titik lainnya yang mengalami keterlambatan. Penundaan ini disebabkan oleh waktu pemrosesan yang dibutuhkan oleh router dalam menangani antrian pengiriman paket di sepanjang jaringan.

$$\text{Delay} = \frac{\sum_{i=T}^{i=T_{t+1}} RT_i - \sum_{i=T}^{i=T_{t+1}} ST_i}{\sum_{i=T}^{i=T_{t+1}} RP_i}; 0 \le t \le T \quad (2)$$

Dimana ∑RTi adalah jumlah paket yang diterim dalam satu waktu, ∑STi adalah jumlah paket yang dikirim dalam satu waktu dan ∑RPi adalah jumlah paket yang diterima. Agar dapat mengetahui selisih antara waktu pengiriman paket dan waktu penerimaan paket maka ketiga formula tersebut dilakukan pengurangan kemudian dibagi dengan jumlah paket yand diterima.

3.4.3 Throughtput

Menunjukkan besarnya paket data yang diterima pada node tujuan dibandingkan dengan waktu tempuh yang ditulis dalam satuan *bit per second (bps).*

$$\text{Throughput} = \frac{\sum \frac{i = T_{t+1}}{i = T} P_i}{T}; 0 \leq t \leq T \qquad (3)$$

Dimana ∑Pi adalah jumlah paket yang diterima dalam satuan (bit), T adalah total waktu pengiriman, total waktu pengiriman tersebut dapat diketahui dengan cara mencari selisih antara waktu terima dan waktu kirim. Berdasarkan Xbee RF Modules oleh Digi International nilai *throughput* pada jaringan Zigbee bernilai antara 5 Kbps sampai 35 Kbps.

3.4.4 Receive Signal Strength Indicator (RSSI)

RSSI adalah salah satu parameter pendukung dari QoS yang menunjukkan kekuatan sinyal yang diterima oleh perangkat dalam satuan *-dBm*. Pengukuran RSSI dilakukan dengan mengirimkan jumlah paket dengan ukuran paket data sama, yaitu 84 Bytes menggunakan *software* XCTU. RSSI dalam beberapa penelitian *network performance* (Sabilla, Sarno and Effendi, 2018) .

3.4.5 Fault Tolerance (Recovery Time for Router)

Toleransi kesalahan dalam WSN adalah kemampuan jaringan untuk mentoleransi kesalahan yang menyebabkan kegagalan layanan. Ini adalah aspek ketahanan jaringan. Service ini berfokus pada pengukuran waktu deteksi kegagalan node atau mengamati *recovery time* saat terjadi kegagalan route.

BAB IV
HASIL PENELITIAN DAN PEMBAHASAN

Pada penelitian ini dilakukan analisis performa transmisi data dengan menerapkan dua skenario yaitu, skenario pertama melakukan analisis terhadap pengiriman data tanpa menerapkan sistem *load balancing (non-balancing).* Skenario kedua melakukan analisis terhadap transmisi data dengan menerapkan *system load balancing*. Di bagian ini kita akan mengetahui perbandingan performa jaringan sebelum menggunakan *balancing* dan sesudah menggunakan *balancing*. Hasil akhir pada penelitian ini adalah perbandingan analisa performa antara *dynamic routing* dan *balancing* yang diusulkan, dengan hasil penelitian sebelumnya.

4.1. Parameter Pengujian

Dalam penelitian ini, ada pencapaian utama yakni membangun *system dynamic* routing dengan system pembagian beban berdasarkan protokol Zigbee serta melakukan analisis performa dari *system wireless sensor network* tersebut. Pada penelitian ini terdapat parameter pengujian yang digunakan sebagai kelengkapan SOP yang perlu disampaikan, antara lain;

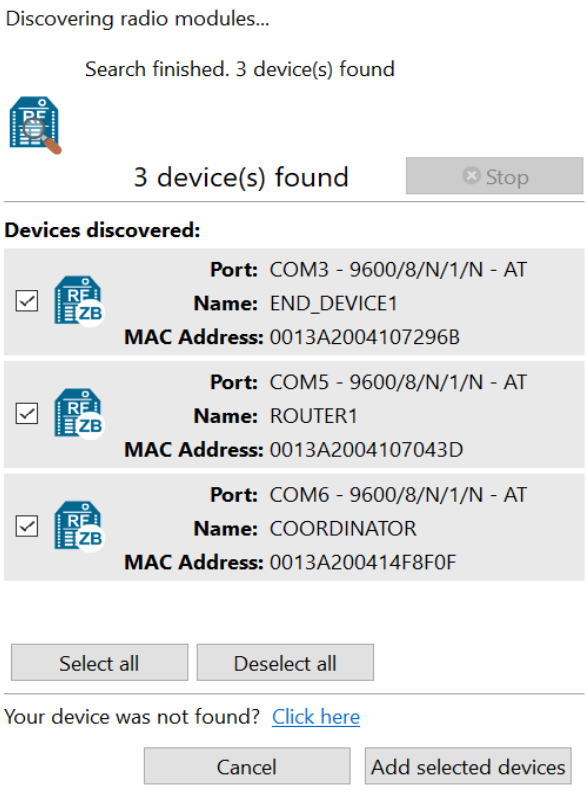
Table 4. 1 Parameter Pengujian

NO	PARAMETER	SPESIFIKASI	FUNGSI
1.	Modul Xbee	Pro S2	Modul Wireless
2.	Xbee Shield	V1.2	Device Pendukung
3.	Arduino	Uno R3	Microprosesor Sistem
4.	XCTU	Versi 6.4.0	Konfigurasi Interface Xbee
5.	Packet Payload	50 Bytes	Deklarasi Karakter 1 Paket Data
6.	Sample of Packets	100 Bytes	Uji Coba Pengiriman Paket
7.	Rx Timeout	1000 ms	Toleransi Waktu Terima Paket
8.	TX Interval	1000 ms	Toleransi Waktu Kirim Paket
9.	Baudrate	9600	Kecepatan Transmisi Serial Com

10.	Mode	AT Mode	Mode Operation for Xbee
13.	Pengujian Outdoor	LOS (100m)	Analisa Qos, Non-Balancing & Balancing
14.	Model Jaringan	Mesh Topology	Analisa Aspek Reliability
15.	Maket Alat	30 cm x 30 cm	Tempat Hardware dan Sensor

4.2. Konfigurasi Dasar Xbee Node by Node Berdasarkan Mesh Routing

Bentuk dasari dari model topologi Mesh adalah berawal dari *jaringan peer to peer* atau dua perangkat sensor saling berkomunikasi dan saling memberikan *request-response*. Sebelum menerapkan algoritma *mesh routing* pada system maka perlu dilakukan konfigurasi via software XCTU dahulu. Berikut ini adalah pengujian dari ketiga mode *wireless* yang di konfigurasi.



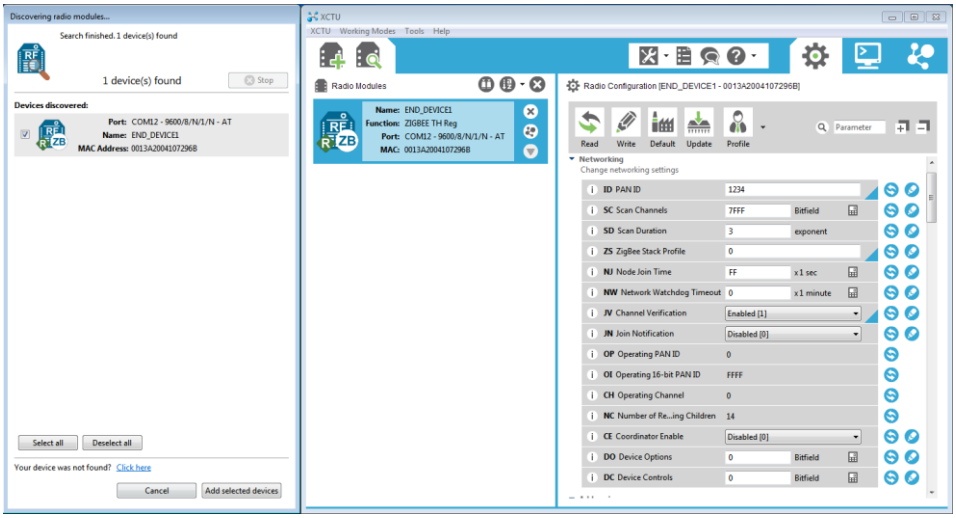
Gambar 4. 1 Konfigurasi Xbee Via XCTU

Gambar 4.1 adalah konfigurasi dari 3 mode operasi perangkat yang digunakan dalam modul wireless sensor. Masing – masing dari perangkat Xbee

menggunakan komunikasi serial dari laptop. *MAC address* pada setiap modul sensor wireless ini didapatkan secara otomatis saat *power usb* tersambung antara Xbee dan Laptop / PC.

4.2.1 Konfigurasi End Device

Perangkat pertama adalah end device, perangkat ini berfungsi sebagai titik yang terhubung secara langsung dengan perangkat sensor, dan perangkat yang berperan sebagai pengirim data sensor dalam system ini. Adapun konfigurasi Xbee S2 sebagai end device disambungkan pada aplikasi *XCTU* yang sudah tersambung dengan COM port serial 5. Selanjutnya mengatur beberapa parameter *device* seperti *Baud*;9600, *Flow Control*; None, *Data Bits*;8, *Parity; None*, *Stop Bits*;1

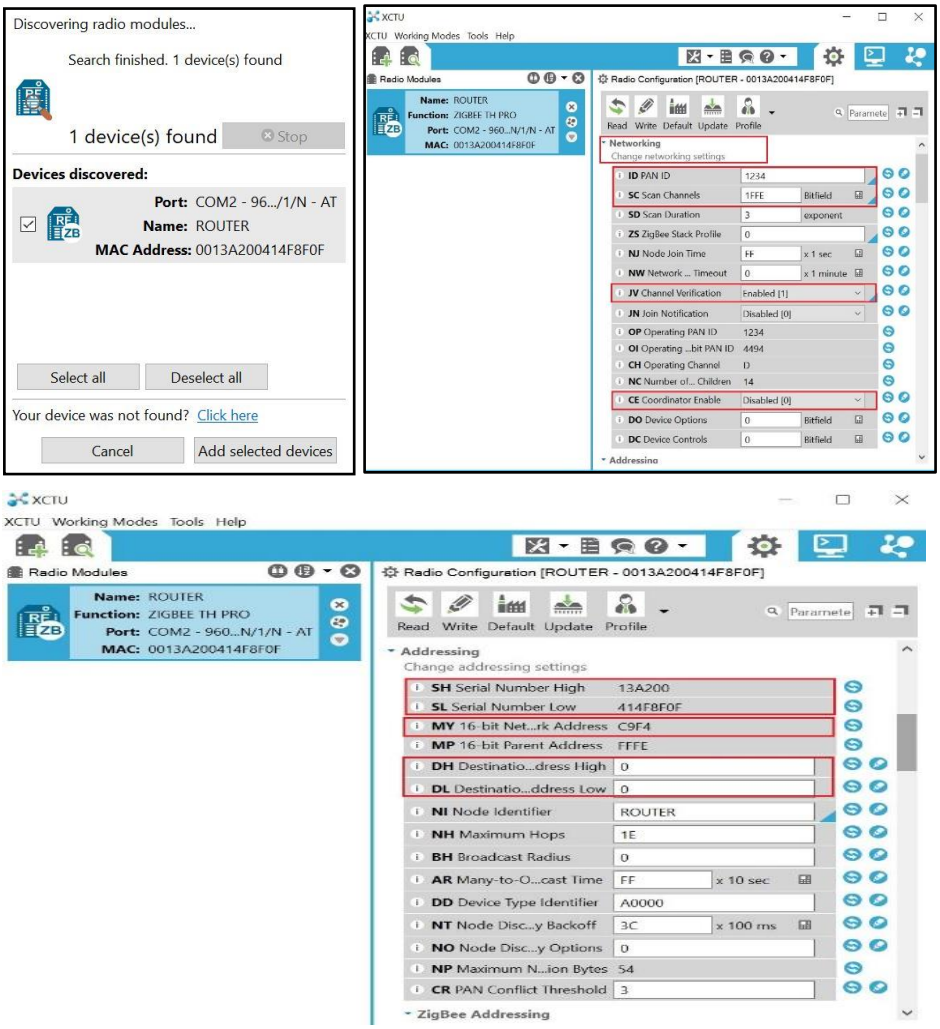


Gambar 4. 2 Konfigurasi End Device dan Port Parameter

4.2.2 Konfigurasi Router

Perangkat kedua adalah Router, perangkat ini berperan sebagai master dalam system ini. Perangkat master selain sebagai perantara dalam jaringan, dia juga bertugas dalam mengontrol lalu lintas transmisi data yang bersumber dari end device. Konfigurasi *Xbee S2* disambungkan pada aplikasi *XCTU* yang sudah tersambung dengan *COM port serial* 2 sebagai Router. Selanjutnya mengatur beberapa parameter *device* seperti *Baud;9600, Flow Control; None, Data Bits;8, Parity; None, Stop Bits*;1.

Pada gambar 4.3 terdapat beberapa parameter yang harus sama agar Xbee dapat berkomunikasi yaitu: *Channel, PAN ID, Destination Address High*. MY merupakan alamat dari device Xbee yang digunakan.

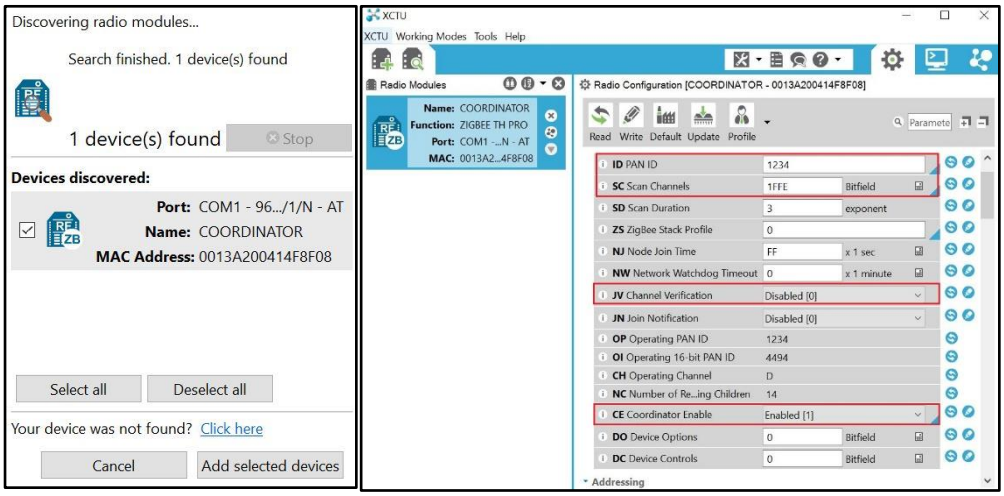


Gambar 4. 3 Konfigurasi Router dan Port Parameter

4.2.3 Konfigurasi Coordinator

Perangkat ketiga adalah coordinator, perangkat ini berfungsi sebagai *data collector* di system. Selain sebagai pusat kontrol jaringan, Coordinator juga berfungsi sebagai server yang menyimpan semua *log* dan *history* pengiriman dari titik end device dan *Router* 1 maupun *Router* 2. Konfigurasi Xbee S2 sebagai coordinator disambungkan pada aplikasi *XCTU* yang sudah tersambung dengan

COM port serial 1. Selanjutnya mengatur beberapa parameter *device* seperti *Baud*;9600, *Flow Control*; None, *Data Bits*;8, *Parity; None*, *Stop Bits*;1, sama halnya seperti *Router* dan end device.



Gambar 4. 4 Konfigurasi Networking Node Coordinator

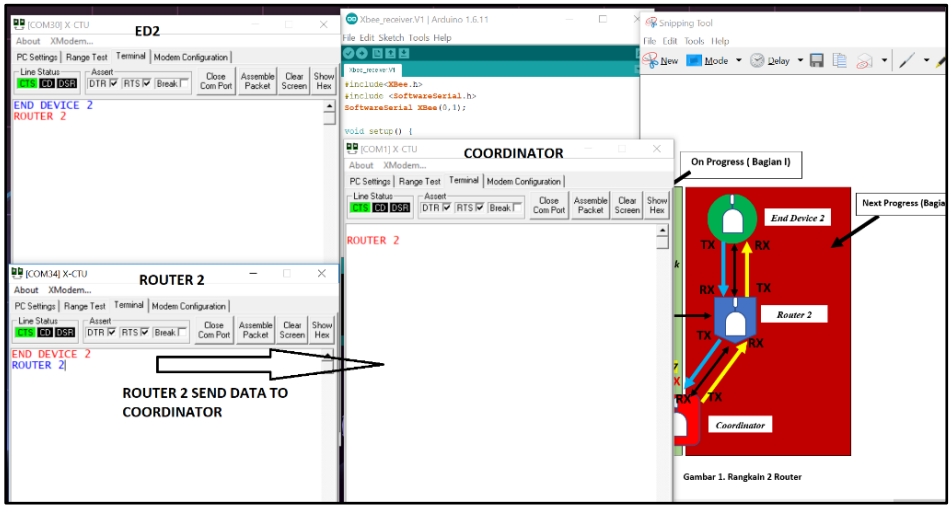
Pada Gambar 4.4 parameter konfigurasi awal agar Xbee dapat berkomunikasi dengan mengatur parameter: *Channel, PAN ID, Destination Address High*, jika pada Router parameter JV *(Join Verification)* disetting 1, maka pada coordinator disetting 0, tujuan dari diset 0 pada coordinator karena perangkat ini sudah memiliki identitas sebagai CE (*Coordinator Enable*) sebagai indentitas induk yang tidak memerlukan verification lagi.

Parameter kedua adalah melakukan konfigurasi pada *Addressing.* Pada konfigurasi ini alamat Xbee didefinisikan. Pada table 3.1 dijelaskan *Source Address* unik 64-Bit IEEE ditetapkan oleh pabrik dan dapat dibaca dengan perintah *SL (Serial Number Low) dan SH (Serial Number High).* Sedangkan pengalamatan 16-bit harus dikonfigurasi secara manual. Pada mode pengalamatan 16-bit digunakan range alamat dari *0 – 0xFFFF*. Modul akan menggunakan alamat 64-bit jika nilai pada *Source Address DL* (mode alamat 16-bit) adalah *"0xFFFF"* atau "0xFFFE". Untuk mengirimkan paket ke spesifik modul menggunakan alamat 64-bit, *Destination Address (DL+DH)* dari pengirim harus disesuaikan dengan *Source Address (SL + SH*) dari modul tujuan.

Gambar 4. 5 Konfigurasi Addressing Pada Coordinator

4.3. Pengujian Peer to Peer Via XCTU (Coordinator ke Router)

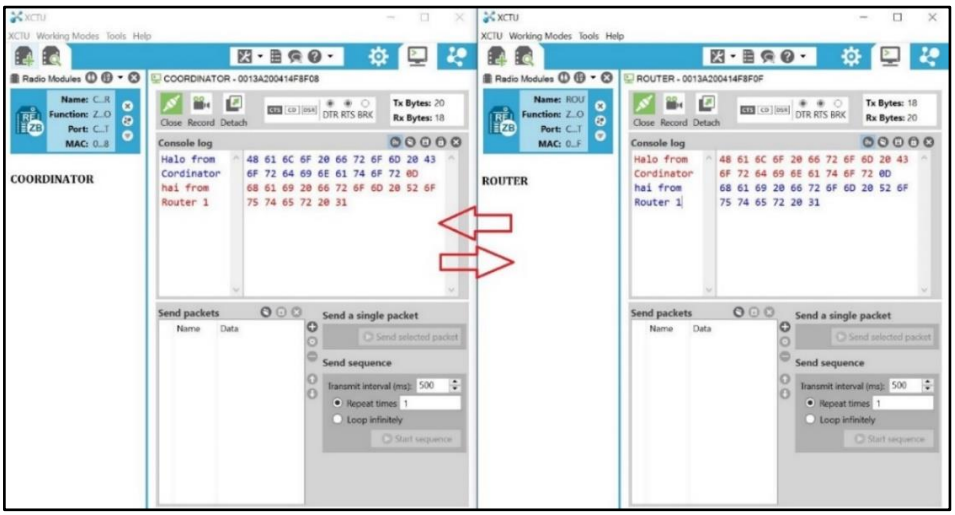
Pengujian ini dilakukan antara end device *2* dan coordinator yang melakukan tes koneksi secara *peer to peer*, dalam gambar via XCTU tepatnya di tab "Terminal", dimana ini adalah *serial interface* untuk tes koneksi *Xbee.*



Gambar 4. 6 Pengujian Komunikasi Peer to Peer Coordinator dan Router

Pengujian ini dilakukan agar dapat mengamati kinerja dari komunikasi *serial* sebagai pondasi jaringan sederhana sebelum merancangan model jaringan *Mesh* nya. Terdapat dua uji coba pada tahap ini, pertama dilakukan pengiriman karakter melalui *Interface* terminal aplikasi XCTU dari coordinator *(Transmitter)*

ke *Router (Receiver),* dan kedua dilakukan pengiriman karakter dari Coordinator ke Router. Berikut uji coba yang telah dilakukan.



Gambar 4. 7 Komunikasi Serial dari Coordinator ke Router

Gambar 4. 8 Uji Coba Komunikasi Antar Xbee

- **Analisa➡** Pada table 4.2 coordinator melakukan pengiriman kata *"Halo from Coordinator",* jika dipecah menjadi karakter maka kata tersebut mengandung sejumlah 20 karakter. Sesuai dengan teknik pengiriman serial yang melakukan pengiriman dari bit 1 sampai 8.

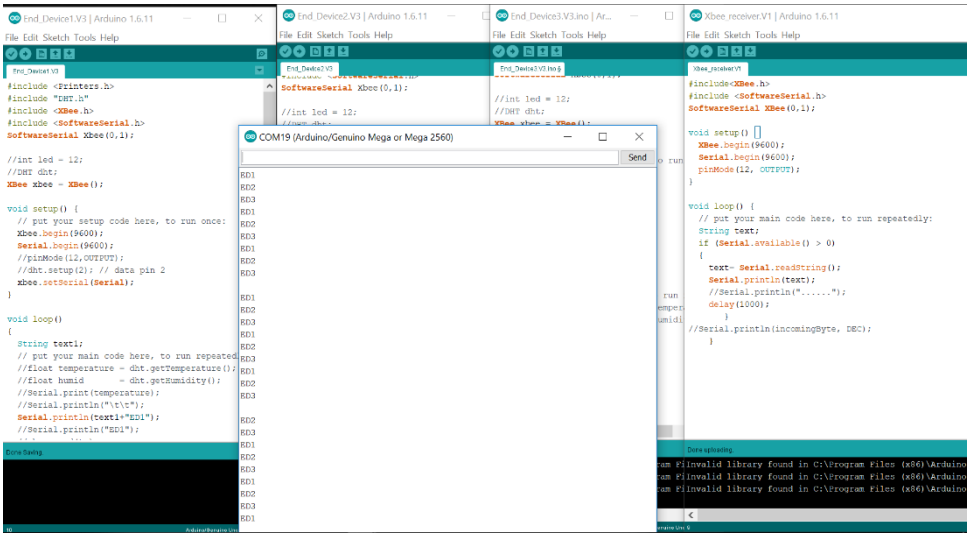
Table 4. 2 Hasil Konversi Pengiriman Karakter

H	a	l	o		f	r	o	m		C	o	r	d	i	n	a	t	o	r
Convert to ASCII Code																			
48	**61**	**6C**	**6F**	**20**	**66**	**72**	**6F**	**6D**	**20**	**43**	**6F**	**72**	**64**	**69**	**6E**	**61**	**74**	**6F**	**72**

Protokol *Xbee* mencari alamat tujuan pengiriman, dalam hal ini mengidentifikasi parameter *ID, CH, SH, SL, DH, dan DL*, setelah parameter ini sama, maka Xbee siap untuk melakukan komunikasi dengan perangkat Xbee lainnya. Terlihat pada tampilan *TX Bytes: 20,* artinya Xbee coordinator mengirimkan karakter sejumlah 20 ke pada Router.

4.4. Pengujian Komunikasi Peer to Peer Via Arduino

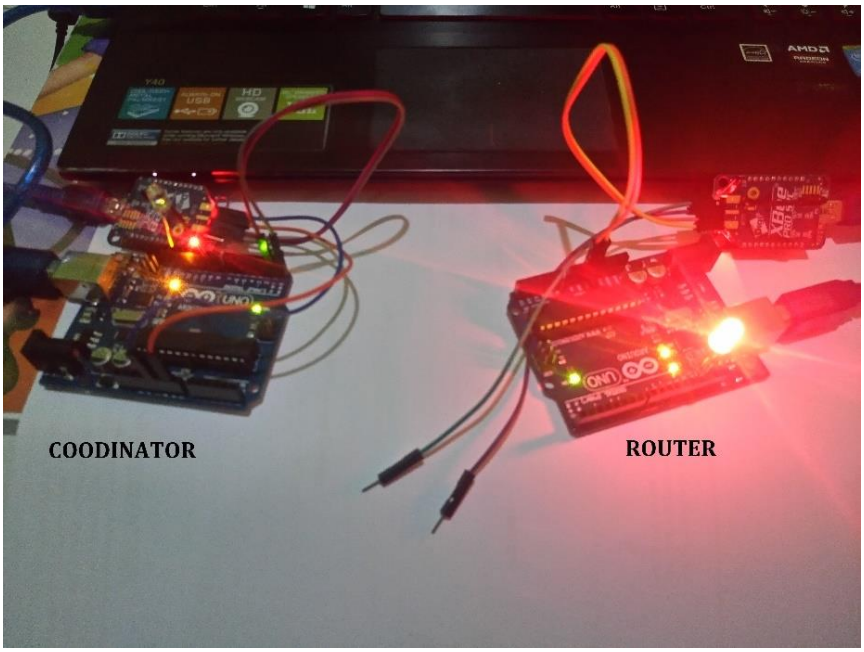
Pengujian ini melibatkan tiga end device dan satu coordinator, dimana pada percobaan dasar ini adalah melakukan pengiriman *peer to peer* langsung menuju ke coordinator dengan menggunakan serial arduino by "C" programming



Gambar 4. 9 Interface Programming Komunikasi Peer to Peer Via Arduino

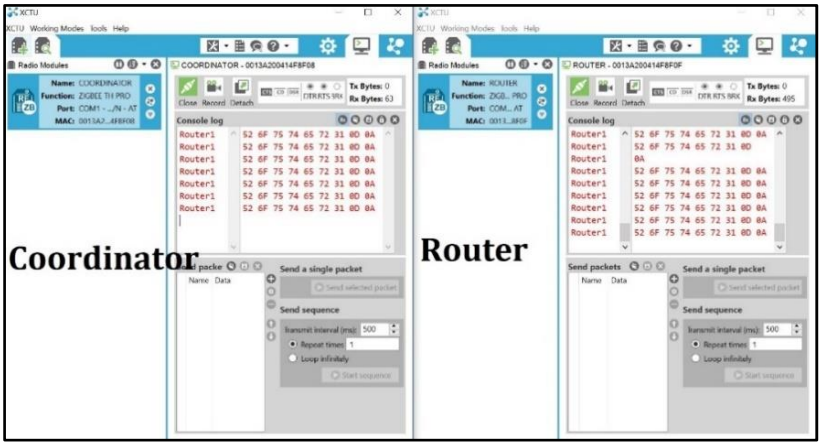
Analisa➡ Pengiriman secara *peer to peer* pada gambar 4.9 menjelaskan bahwa pengiriman ini dengan menggunakan objek string untuk memastikan bahwa *routing* dan komunikasi dari end device ke coordinator sudah tersambung. Pengujian komunikasi ini bertujuan memastikan selain Xbee dapat

berkomunikasi antar Xbee namun dapat berkomunikasi dengan Arduino. Pengujian ini terdapat dua macam, pertama pengujian komunikasi secara internal artinya komunikasi yang dilakukan antara Xbee dengan Arduino, kedua adalah pengujian external artinya komunikasi yang dikirimkan dari Arduino-Xbee A ke Arduino-Xbee B melalui jaringan wireless Xbee, berikut gambar uji cobanya.



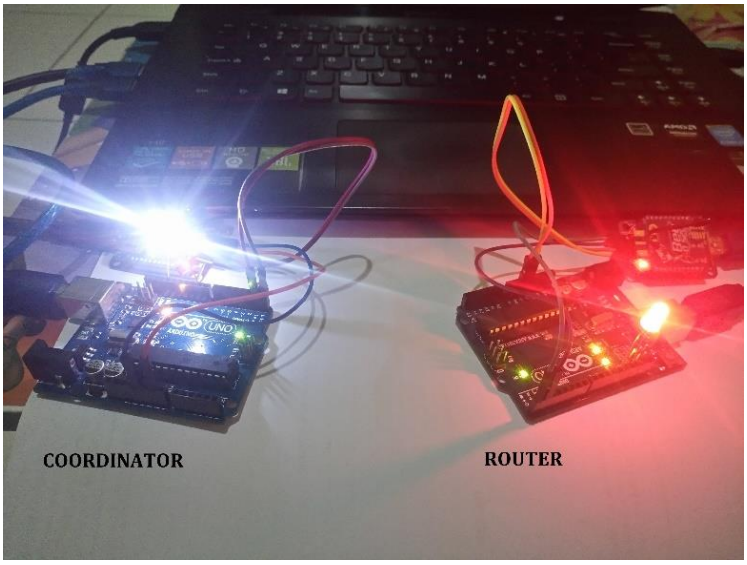
Gambar 4. 10 Uji Coba Internal antar Xbee Router – Arduino

Pada gambar 4.10 telah dilakukan uji coba awal jaringan *peer to peer* antara *Router* sebagai *transmitter* dan Coordinator sebagai *receiver*. Agar dapat melakukan pengiriman internal Pin RX pada Xbee dihubungkan ke Pin TX Arduino. Pin TX pada Arduino melakukan pengiriman serial ke Xbee dan diterima oleh pin RX Xbee. Hal ini hanya menggunakan dua Xbee yang tersambung langusng dengan Arduino untuk memastikan koneksi dari satu titik ke titik berikutnya agar tetap tersambung.



Gambar 4. 11 Coba Komunikasi Serial Internal Xbee-Arduino

Pada serial monitor XCTU pengiriman karakter berhasil dilakukan dari Arduino ke Xbee, tampak bahwa dari Arduino mengirimkan pesan router1, dan Xbee Coordinator menerima karakter router1.



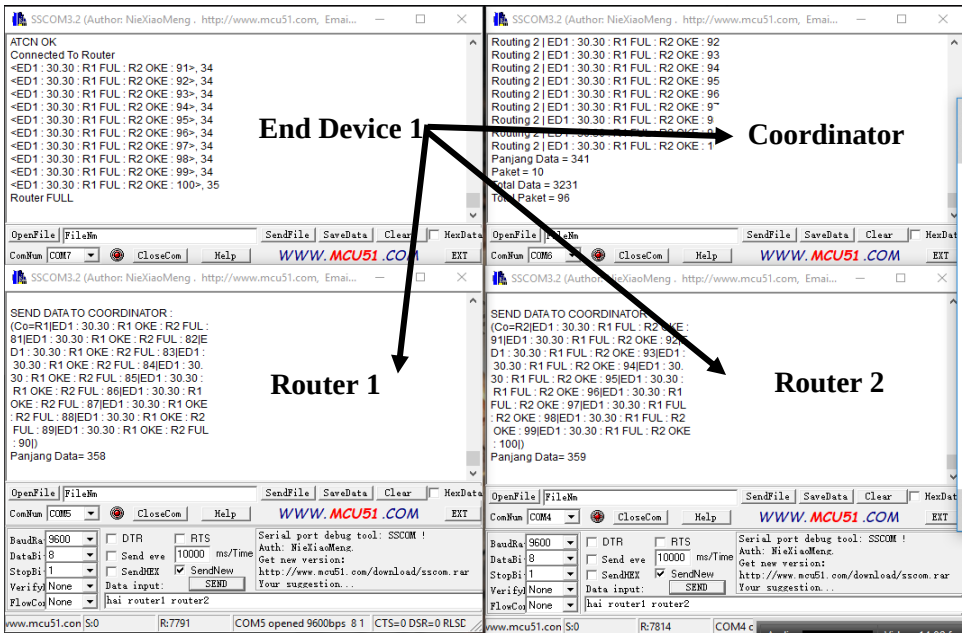
Gambar 4. 12 Uji Rangkaian Pengiriman antara Router dan Coordinator

Pin TX pada Xbee dihubungkan ke Pin TX Arduino. Pin TX pada Arduino melakukan pengiriman serial ke Xbee, selanjutnya dari pin TX Xbee mengirim ke alamat *ID, CH, SH, SL, DH, dan DL* yang satu network. Pada gambar 4.12 penulis membuat sebuah visualisasi dalam bentuk rangkaian Router dan Coordinator dengan LED sebagai indikator karena modul yang dimiliki terbatas. Visualisasi

rangkaian Router saat melakukan pengiriman paket led merah akan menyala, dan saat Xbee Coordinator menerima paket dari Router led putih akan menyala, visualisasi rangkaian seperti ini penulis rancang sebagai ganti dari uji coba awal sebelum menggunakan rangkain seperti arsitektur hardware seperti pada gambar 3.19.

4.5. Pengujian Komunikasi Peer to Multi Peer Via Serial Interface

Pengujian peer to multi peer ini dengan menggunakan software *Serial Monitor*. Pengujian ini menggunakan satu end device, dua router dan satu coordinator. Pada proses pengujian ini dilakukan penerapan mesh routing secara sederhana sebelum membentuk titik yang lebih kompleks dengan tiga end device, adapun tujuan yang dikirimkan dari titik ED1 menyebar Router1,2 dan Coordinator.



Gambar 4. 13 Interface Serial Peer to Multi Peer

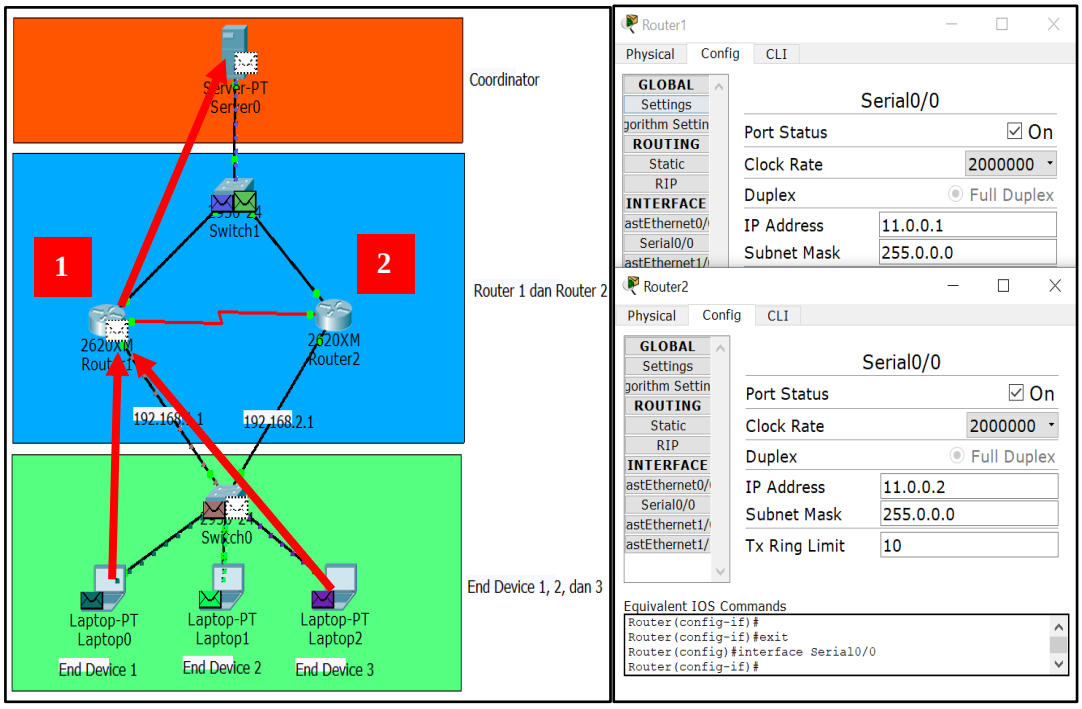
Analisa➡ Gambar 4.13 adalah proses transmit paket data dengan *multi-channel*. Terlihat pada serial coordinator, paket yang dikirimkan berhasil diterima sejumlah 96 paket dari total 100 paket yang dikirimkan, dengna kata lain dalam siklus mengirimkan 100 sample paket terjadi 4 paket yang hilang *(packet loss)*. Hal ini telah berhasil diimplementasikan pengujian komunikasi secara *peer to multi peer*.

4.6. Pengujian Dengan Softwere Simulasi Jaringan

Pengujian selanjutnya adalah dengan melakukan uji coba menggunakan aplikasi *Cisco Packet Tracer*. Aplikasi ini digunakan sebagai simulator dalam penelitian ini dengan tujuan untuk memberikan gambaran bagaimana peran router dalam melakukan kontrol dan pengalihan routing disaat kondisi *load balancing maupun non-balancing.* Di sub bab selanjutnya akan dibahas kondisi dimana system hanya memiliki satu dua router, tiga router, dan empat router. Tentu dalam berbagai kondisi ini memberikan analisa routing dan skenario yang berbeda.

4.6.1 Simulasi Dengan Dua Router

Simulasi ini melibatkan tiga end device, dua router dan 1 coordinator. Perangkat end device bertugas mengirimkan data sensor yang dikelola dengan menggunakan mikrokontroller arduino yang terhubung dengan komunikasi serial komputer, sedangkan perangkat router bertugas mengatur *routing* komunikasi antara end device, coordinator, dan router lainnya.



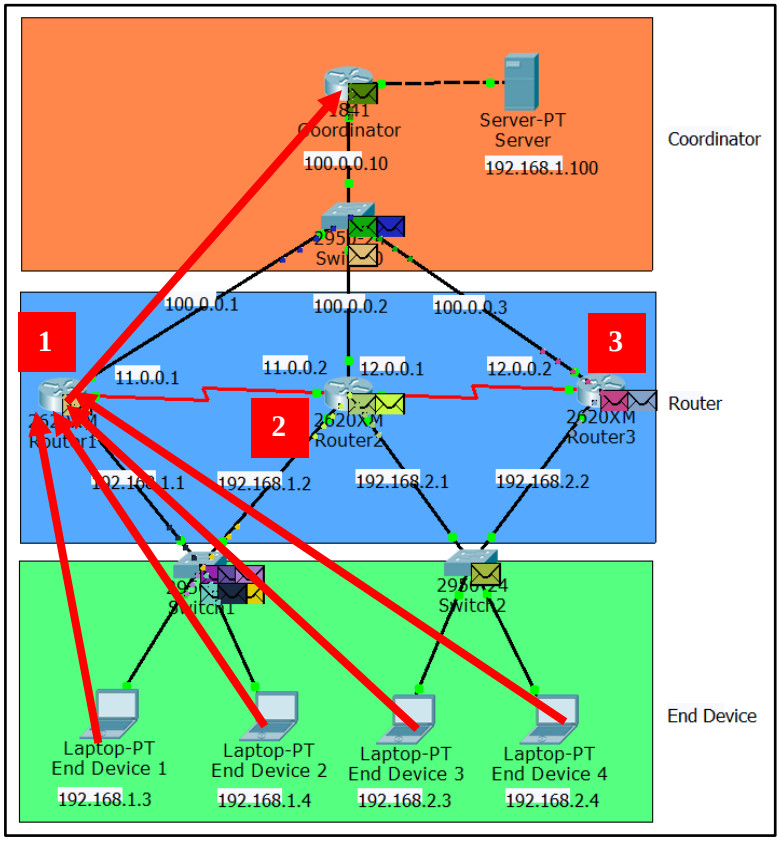
Gambar 4. 14 Dua Router Aktif | Gambar 4. 15 Konfigurasi 2 Router Aktif

Pada bagian ini hal yang paling ditekankan adalah melakukan analisis saat menggunakan dua router yang masih aktif dan hanya satu router saja yang aktif.

Pada gambar 4.14 kedua router disetting aktif kondisi ini memberikan keuntungan akses routing tambahan. Hal yang dapat di amati dari kondisi ini adalah pengiriman paket data yang dikirimkan melalui router satu dan router satu sebagai alternatif routernya, dan jika terdapat satu router yang mati /off maka pada simulasi ini hanya terdapat satu routing yang dapat dilalui paket data.

4.6.2 Simulasi Dengan Tiga Router

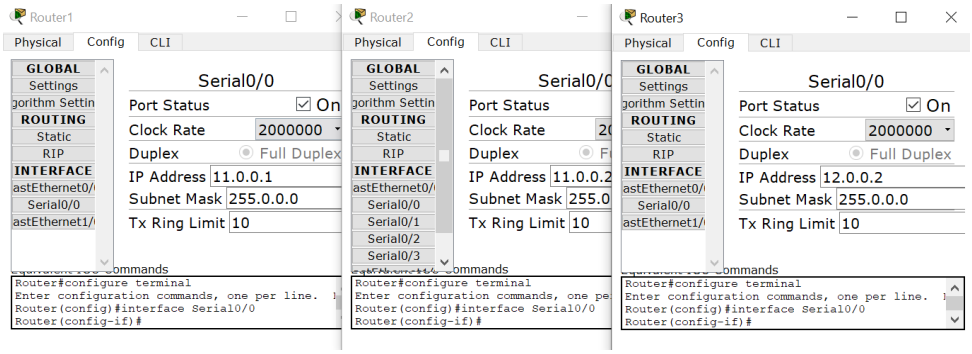
Pada bagian ini simulasi disetting dengan tiga router. Secara tugas dari masing masing perangkat sama seperti pada sub bab yang sudah dijelaskan diatas. Hanya pada kondisi tiga router ini, alternatif routing atau jalur lintasnya bertambah satu, yang awalnya hanya memiliki dua routing, sekarang memiliki alternatif tiga routing.



Gambar 4. 16 Simulasi Kondisi Tiga Router Aktif, Skenario Pertama

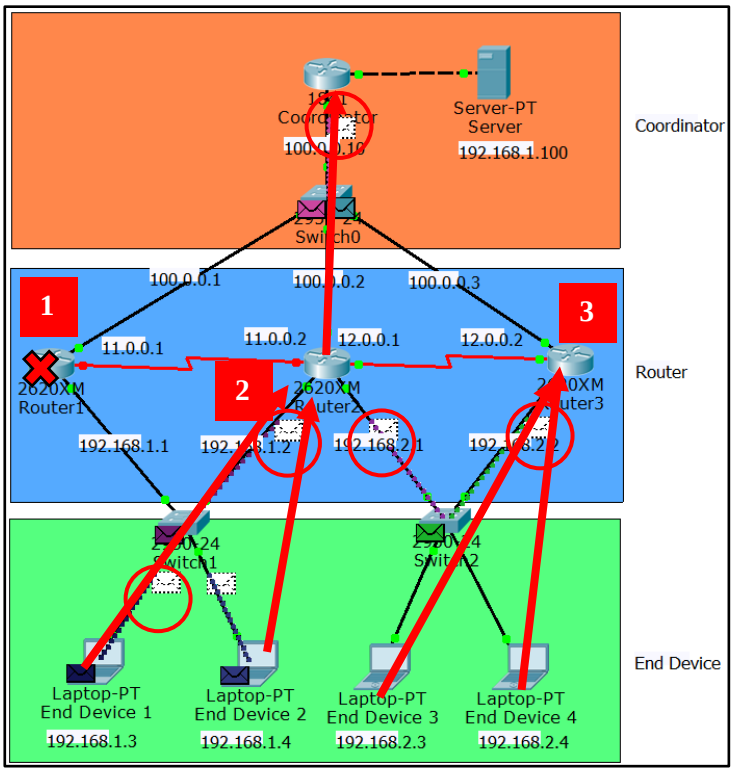
Kondisi saat end device mengirimkan paket akan secara berurutan dalam proses transmisinya, diawali dari end device 1 mengirimkan paket ke router 1, kemudian end device 2 mengirimkan paket ke router 1, lalu end device 3 mengirimkan paket

ke router 1, dan end device 4 mengirimkan paket ke router 1. Kondisi ini terus berlangsung hingga mencapai batas maksimum penyimpanan pada masing-masing router.



Gambar 4. 17. Konfigurasi Tiga Router Aktif

Namun kita coba dengan melakukan setting dua router yang aktif, routing manakah yang akan dilalui? Jawabannya ada dua alternatif, pertama paket data yang dikirimkan akan sampai pada router 2 (routing 2), dan alternatif kedua paket data akan dikirimkan menuju router 3 (routing 3).

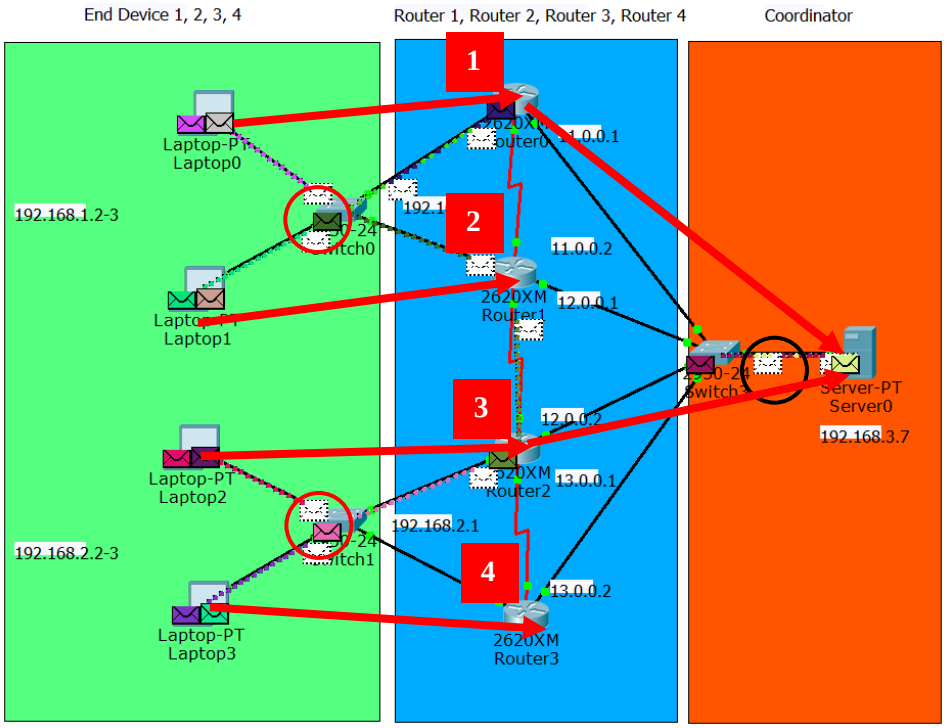


Gambar 4. 18 Simulasi Dua Router, Skenario Kedua (Satu Router Off)

Jika diperhatikan dari gambar 4.18 walaupun kondisi system memiliki tiga router, namun disaat router 1 terjadi masalah dan berujung off, maka jalur yang dapat dilewati oleh paket data akan dialihkan ke router 2 dan router 3. Gambar lingkaran adalah ilustrasi paket data yang tersebar mengikuti rute yang menuju pada router yang masih aktif.

4.6.3 Simulasi Dengan Empat Router

Di bagian ini simulasi akan menggunakan empat router. Empat router pada skenario pertama akan disetting aktif semua. Selanjutnya di skenario kedua akan dimatikan 1 router, dan diskenario ketiga akan dimatikan 2 router.

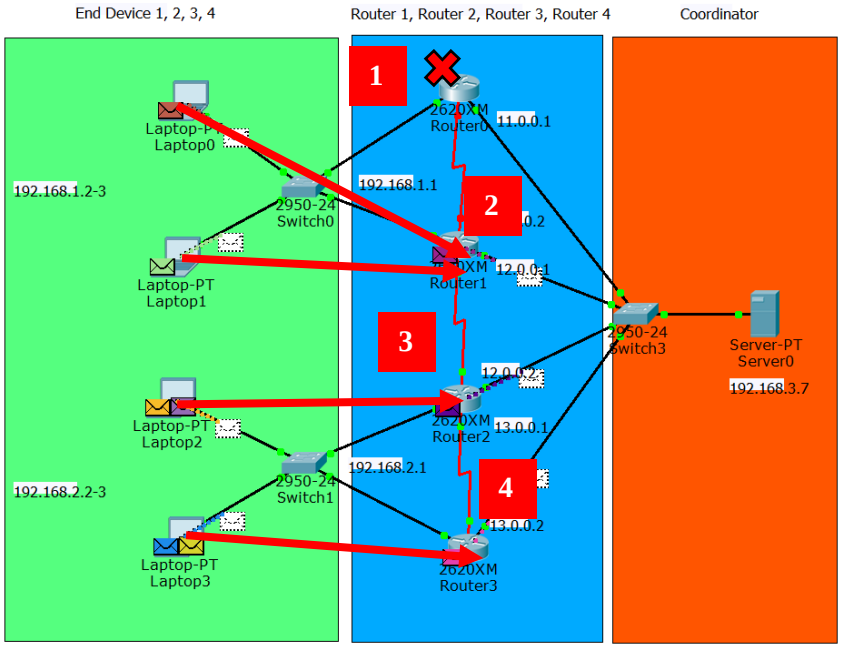


Gambar 4. 19 Simulasi Kondisi Empat Router Aktif

Skenario pertama adalah kondisi disaat empat router aktif, di kondisi ini simulasi routing lebih bervariasi. Paket data ditransmisikan dari empat end device melawati routing 1 menuju ke router 1. Masih pada skenario 1, disaat router 1 berhasil menerima semua paket yang dikirimkan dari end device menuju ke router 1. Pada skenario kedua kondisi router 1 dinonaktifkan, jalur yang menghubungkan dari end device menuju dari router 1 tidak terhubung, ilustrasi pada gambar 4.21

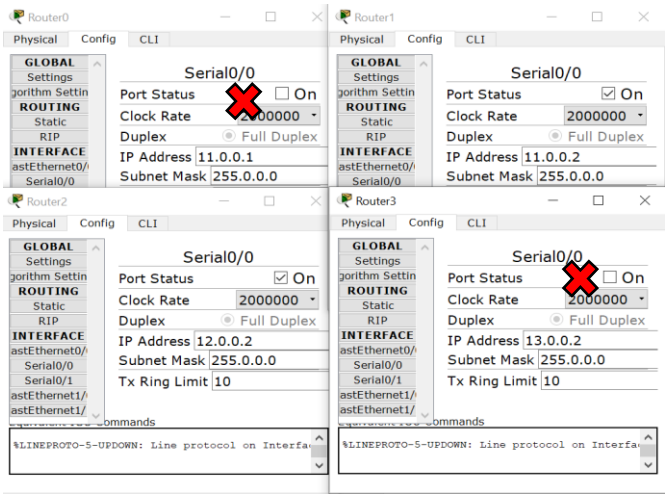
Gambar 4. 20 Konfigurasi Skenario Kedua (Satu Router Off)

Dilanjutkan pada percobaan simulasi menggunakan empat router yang menggunakan skenario kedua, dimana kondisi router 1 dimatikan. Terlihat pada gambar 4.20 konfigruasi router 1 telah dimatikan. Simulasi pada skenario ini menguji transmisi paket data untuk dapat mengalihkan rute disaat terjadi gangguan.



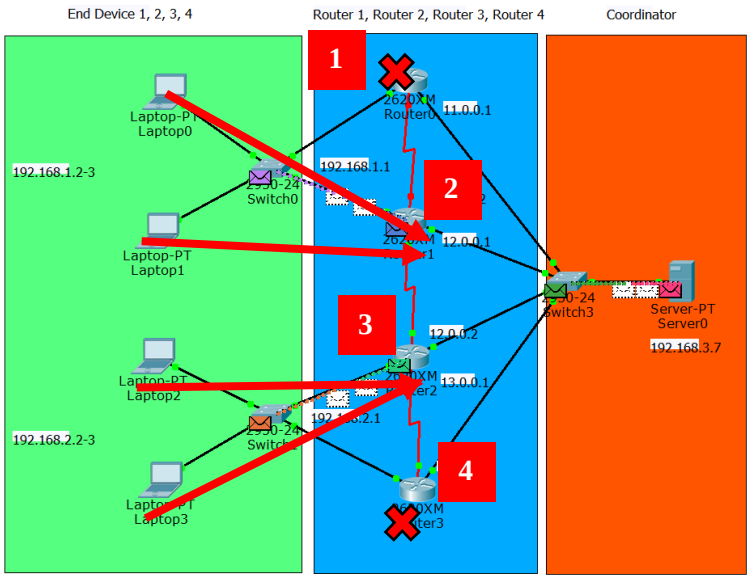
Gambar 4. 21 Simulasi Empat Router Skenario Kedua (Satu Router Off)

Simulasi pada skenario kedua ini memiliki tingkat kerumitan dari pada simulasi menggunakan dua router dan tiga router. Penggunaan alamat routing pada router harus benar benar sesuai dengan alamat router yang dituju. Di simulasi ini terdapat tiga rute yang dapat dilalui, pertama paket dapat dapat dikirimkan menuju router 2 (routing 2), kemudian terdapat alternatif pilihan router 3 (routing 3), dan router 4 (routing 4).



Gambar 4. 22 Konfigurasi Skenario Ketiga (Dua Router Off)

Pada gambar 4.23 skenario ketiga dimana kondisi router 1 dan router 4 dimatikan. Hal ini membuat rute ke routing 1 dan 4 tidak bisa dilalui harus dialihkan ke routing 2 dan 3. Dalam percobaan ini telah berhasil dilakukan dengan menguji pengiriman



Gambar 4. 23 Simulasi Empat Router Skenario Ketiga (Dua Router Off)

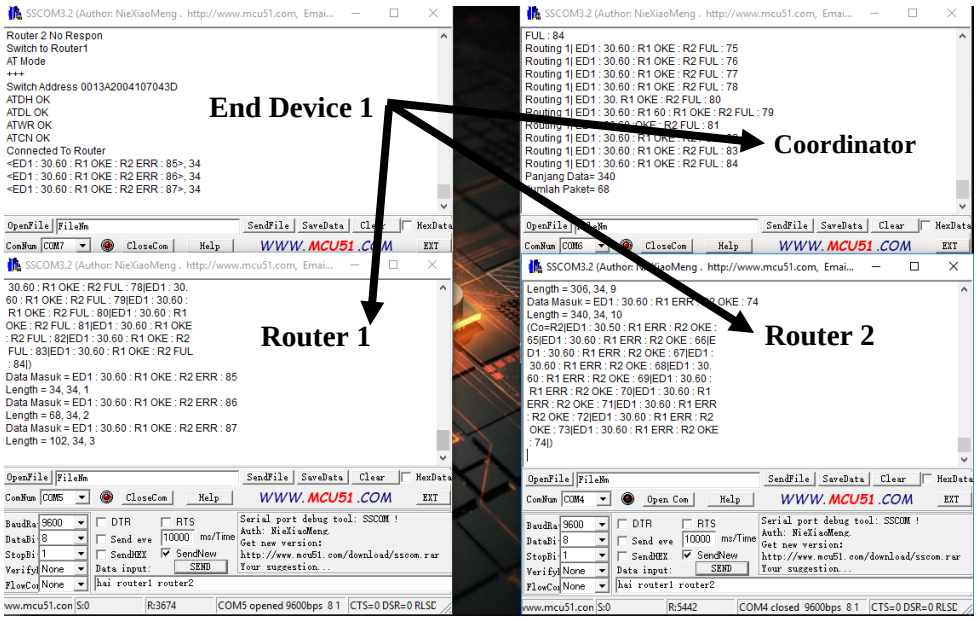
Paket dari end device 1, 2, 3, dan 4. Semua paket didistribusikan secara berurutan dari router 2 menuju ke router 3 dan selanjutnya dikirimkan ke coordinator. Jika diperhatikan pada gambar 4.23 paket tidak melewati rute pada routing 1 dan routing 4 karena pada kondisi skenario ini router 1 dan router 4 dimatikan, sehingga pengiriman paket tidak bisa melewati.

4.7. Arduino Test

Uji coba keseluruhan system pada penelitian ini diproses pada *microcontroller arduino*. Deklarasi panjang data dan variable, fungsi mengolah jadwal pengiriman (*handshaking*) paket data antar Xbee, serta menjalankan fungsi routing dan balancing menggunakan komunikasi serial.

4.7.1 Self-Healing (Rerouting)

Dalam pengujian ini terdapat dua skenario yang sudah diimplemtasikan yaitu pertama implementasi *self-healing* dan kedua system balancing. *Self-Healing* system berfungsi sebagai alternatif routing saat terjadi masalah pada router, dan system balancing sebagai alternatif *workload* saat terjadi beban berlebih. Uji coba ini telah berhasil diimplementasikan dan dijalankan menggunakan *microcontroller* arduino.



Gambar 4. 24 Self-Healing to Router 1

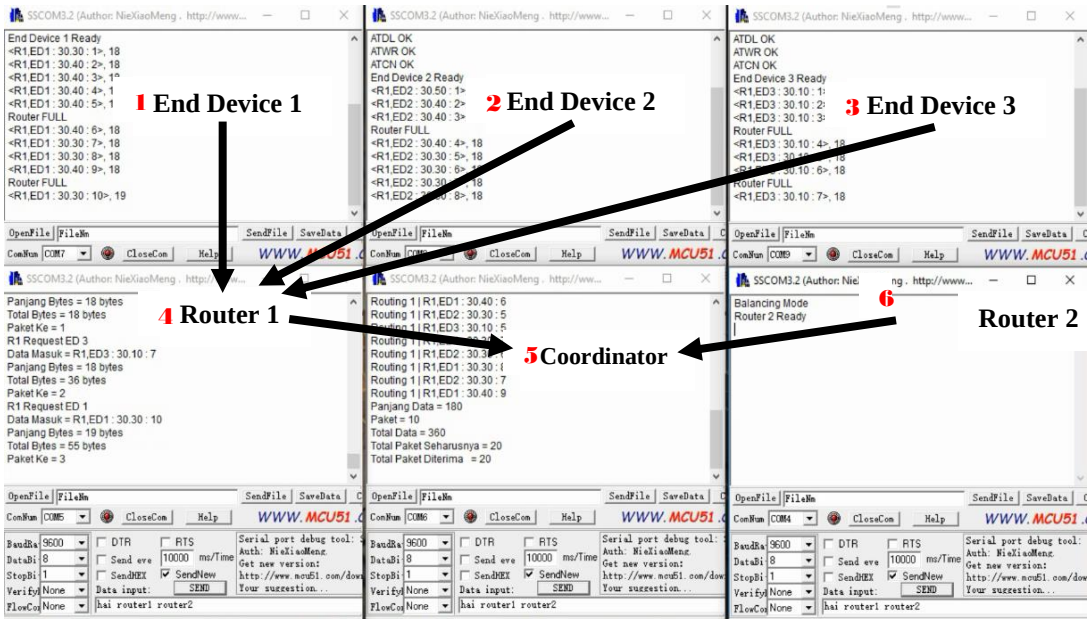
Analisa ➡ Berdasarkan hasil uji coba *self-healing* pada gambar 4.24 terdapat empat *interface serial* yang melakukan transmisi data, end device *1 (ED1) , router 1, router 2,* dan coordinator. Di menit pertama semua perangkat aktif dan mendapatkan sumber daya laptop. Proses pertama yang dilakukan router 1 adalah melakukan *request* pada tiga ED dengan prioritas urutan pertama adalah dari ED1. Selantjutnya ED1 mengirimkan sejumlah paket sesuai dengan permintaan Router 1. Selama proses pengiriman data berlangsung dari ED1, router 1 bertugas untuk mengontrol dan memonitoring paket yang dikirimkan sampai dengan batas rule yang ditentukan. Di proses ini router 2, dan coordinator berada dalam kondisi *idle* dan menunggu respon dari Router 1 saat terjadi gangguan atau Router 1 *off*. Saat kondisi router 1 *off* tindakan pertama yang dilakukan adalah memberikan respon ke pada router 2 dengan mengambil alih proses penerimaan data dari ED 1. Proses pengambil alihan ini dilakukan dengan cara mendeteksi tidak adanya koneksi dari Router 1 yang tersambung dengan *microcontroller*. Saat kondisi router 1 *off* system akan melakukan *self-healing* dengan cara mencari rute baru yang masih aktif hal ini berdasarkan *rule system* yang telah dibuat. Kondisi ini membuat router 2 memberikan request kepada ED2, dan ED3 untuk meneruskan paket yang dikirimkan.

Mekanisme dalam pergantian rute ini dilakukan dengan cara mengganti alamat jaringan dari Router 1 yang kita sebut dengan alamat (DH+DL) ke alamat jaringan router 2, atau bisa dikatakan system yang telah mendeteksi terjadinya *self-healing* maka akan melakukan *"Switching Address"* ke alamat router yang aktif. Selanjutnya pada proses tersebut pengiriman paket diteruskan oleh ED2, dan ED 3 yang melalui Routing 2, kondisi rute telah berganti yang awalnya paket dikirimkan melalui Routing 1 beralih ke Routing 2.

Selanjutnya perhatikan pada gambar 4.24 pada bagian ED 1 terlihat pengiriman yang melewati routing 1, mengalami gangguan *(router off)* sehingga memicu respon "*Router 2 No Respon*" sehingga dilakukan *switching address* ke alamat router 1. Selanjutnya router 1 meneruskan paket ke Coordinator, dimana saat uji coba yang terdokumentasi, Coordinator berhasil menerima 68 paket, dari pengiriman yang ke 8 dengan jumlah 80 paket data. Satu kali pengiriman paket berjumlah 10 paket. Jadi 8 kali pengiriman paket sudah berjumlah 80 paket.

4.7.2 Sistem Pembagian Beban (Pengujian Balancing)

Uji coba *balancing* bertujuan untuk membuktikan aspek efektifitas dari kualitas paket yang terjaga, Uji coba ini telah berhasil diimplementasikan dan dijalankan menggunakan microcontroller arduino, seperti yang ditunjukkan pada gambar 4. 25.



Gambar 4. 25 Balancing Router

Analisa :➡ Sistem Pembagian beban atau yang disebut dengan *Balancing System* adalah system yang dirancang untuk mengontrol *workload (beban kerja). Workload* disini direpresentasikan dengan beban paket data yang tersimpan pada *microprosesor*. *Workload* yang berlebihan akan menurunkan performa jaringan pada media penyimpanannya, sehingga perlu adanya mekanisme kontrol pada perangkat wireless. Berdasarkan hasil uji coba *balancing system* pada gambar 4.25, ED1, ED2, dan ED2 menerima respon dari router 1 yang berupa keterangan *"Router Full",* artinya *system balancing* telah aktif dengan membatasi penerimaan jumlah paket yang diterima oleh router 1. Jumlah paket awal yang sudah ditentukan pada prosedur penelitian ini adalah maksimum 10 paket dengan pertimbangan menjaga kestabilan router dan keterbatasan spesifikasi perangkat *microcontroller*.

Secara mekanisme pada *balancing system*, satu kali siklus transmisi membawa sebanyak 10 paket data dari semua proses pengiriman yang dilakukan

ED secara berurutan. Sepuluh paket data ini akan disimpan sementara pada *buffer* router sebelum dikirimkan ke Coordinator. Jadi saat router 1 sudah menerima 10 paket, maka system akan memberikan respon atau pemberitahuan ke pada ED bahwa router 1 sudah mencapai batas maksimum paket siklus transmisi yang pertama. Selanjutnya router 1 memberikan *request* ke router 2 untuk melakukan *switching address* dengan tujuan router 2 memberikan tugas selanjutnya pada ED untuk meneruskan paket nya melalui jalur routing 2. Disaat bersamaan router 1 meneruskan paketnya tersebut ke coordinator. Di posisi ini paket yang dikirimkan dari ED disimpan sementara di router 2. Penyimpanan paket sementara pada router 2 hanya dibatasi sepuluh paket, Saat router 2 sudah menerima sejumlah 10 paket yang berasal dari tiga ED, maka Router 2 memberikan respon ke pada router 1 untuk mengganti ke alamat Routing 1, dan siklus selanjutnya router 1 yang kemudian mengambil alih peran *master* untuk *request* penerimaan paket dari ketiga ED, dari siklus proses point 6 yang tinjukkan pada gambar 4.25 adalah proses dimana semua paket dari router 1 dan router 2 dikirimkan ke coordinator.

4.8. Performance Analysis of Qos Parameter

Hasil pengujian performa QoS ini memberikan bukti pendukung untuk menjamin performa jaringan yang lebih unggul, berikut ini acuan beberapa parameter performa dalam *Quality of Service, Delay Tolerance(delay), Loss Tolerance (Packet Loss), Capacity (Throughtput),* dan *RSSI.*

Table 4. 4 Performance Analyst based on QoS Parameters

No	Rule	Range	End Device	Router 1	Router 2	COOR	Throughput (KBps)	PacketLoss (%)	Delay (second)	RSSI (-dBm)	Recovery Time
1	Non-Balancing	10 meter	100 packet	92	85	82	60,7	18%	6,6	-56	12s
		50 meter	100 packet	80	78	75	55,6	25%	7,2	-75	12s
		70 meter	100 packet	75	60	51	37,8	49%	10,6	-86	13s
		100 meter	100 packet	58	50	45	33,3	55%	12,0	-90	13s
2	Balancing	10 meter	100 packet	95	91	87	64,4	13%	6,2	-49	4s
		50 meter	100 packet	88	82	82	60,7	18%	6,6	-58	3s
		70 meter	100 packet	85	75	74	54,8	26%	7,3	-65	3s
		100 meter	100 packet	76	65	78	57,8	22%	6,9	-85	5s

Keterangan table 4.4 :

Pada table ini akan diberikan penjelasan tentang variabel yang digunakan salam pengukuran penelitian ini. Mulai rule/ aturan pengujian, range/jarak, end device, router 1 dan 2, coordinator, dan aspek pendukung qualitas QoS.

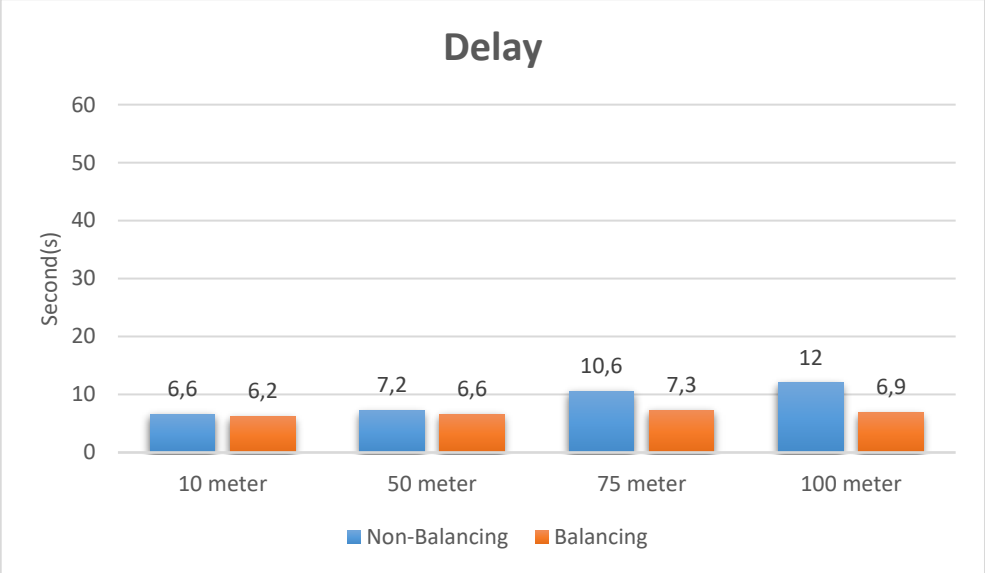
• Rule	: Aturan uji coba system transmisi paket data
• Range	: Aturan uji coba system transmisi paket data
• End Device	: Jarak yang diskalakan dengan ukuran meter. Perangkat WSN yang bertugas mengirimkan data sensor yang dibungkus dalam 1 paket data dengan ukuran 50 bytes
• Router 1 &2	: Perangkat WSN sebagai *controller routing*, dan penghubung dari perangkat end device dan coordinator
• Coordinator	: Perangkat WSN sebagai pusat jaringan dan membangun operating channel dan PAN (Personal Area Network) ID pada sebuah jaringan
• Throughput	: Menunjukkan besarnya paket data yang diterima pada node tujuan dibandingkan dengan waktu tempuh yang ditulis dalam satuan *bit per second (bps).*
• Packet Loss	: Packet Loss adalah kegagalan transmisi paket ke tujuannya
• Delay	: Delay adalah total waktu tunda paket yang disebabkan oleh proses transmisi
• RSSI	: RSSI adalah salah satu parameter pendukung dari QoS yang menunjukkan kekuatan sinyal yang diterima oleh perangkat dalam satuan -dBm
• Recovery Time	: Recovery time adalah jeda waktu pemulihan saat terjadi node dan routing terjadi masalah

Berdasarkan table 4.4 hasil uji coba dan analisa QoS terbagi menjadi dua rule system. Pertama pada rule *non-balancing* menghasilkan rata-rata nilai *packet loss* sebesar 37% dengan rata-rata delay sebesar 1.7 second, sedangkan untuk standar *throughput* Xbee sendiri masih dalam kategori rendah. Jika dibandingakan dengan hasil pengukuran balancing menghasilkan rata-rata nilai packet loss sebesar

20 %, dengan rata-rata delay sebesar 2,3 second. Dan jika diamati lebih detail hasil pengukuran RSSI pada rule balancing memberikan selisih nilai -12dBm yang relatif lebih baik kekuatan sinyalnya.

4.8.1 Delay

Dari hasil uji coba Delay pada gambar 4.26 Bertujuan mengetahui waktu selisih antara waktu pengiriman paket data dan waktu penerimaan paket data. Berikut gambar grafik Delay;



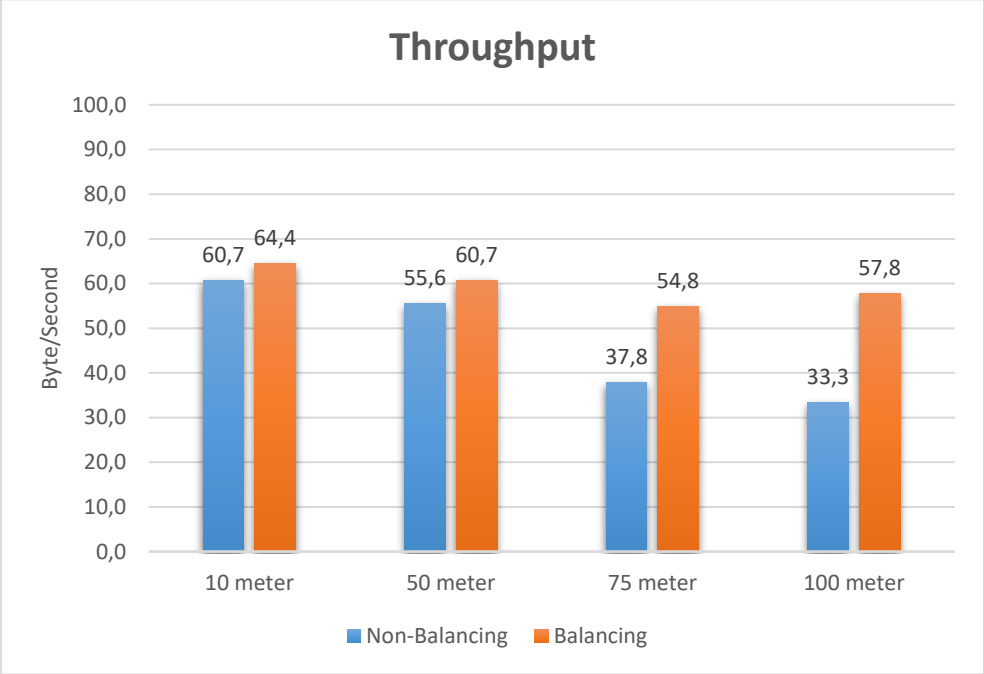
Gambar 4. 26 Delay vs Distance

Analisa ➡ Kriteria yang diharapkan dari pengujian delay adalah mendapatkan nilai waktu yang paling kecil nilainya, karena nilai terkecil, transmisi data lebih baik. Analisa delay menunjukkan bahwa waktu tunda proses rule **non-balancing rata-rata** sebesar 9,1 dan delay pada **balancing rata-rata** 6,8. Artinya dari kedua rule performance diatas menunjukkan bahwa pada rule non-balancing delay nya lebih lambat jika dibandingakan dengan rule balancing.

4.8.2 Throughput

Dari hasil uji coba *throughput* pada gambar 4.27 merupakan hasil pengukuran pada nilai rata-rata *throughput* dengan pengaturan jarak yang berbeda. Dari pengukuran ini bertujuan untuk mengetahui *bandwidth* aktual (aktual) yang

diukur dalam waktu tertentu dan dalam kondisi jaringan tertentu yang digunakan untuk mentransfer file dengan ukuran tertentu. Berikut gambar grafik *throughput* ;

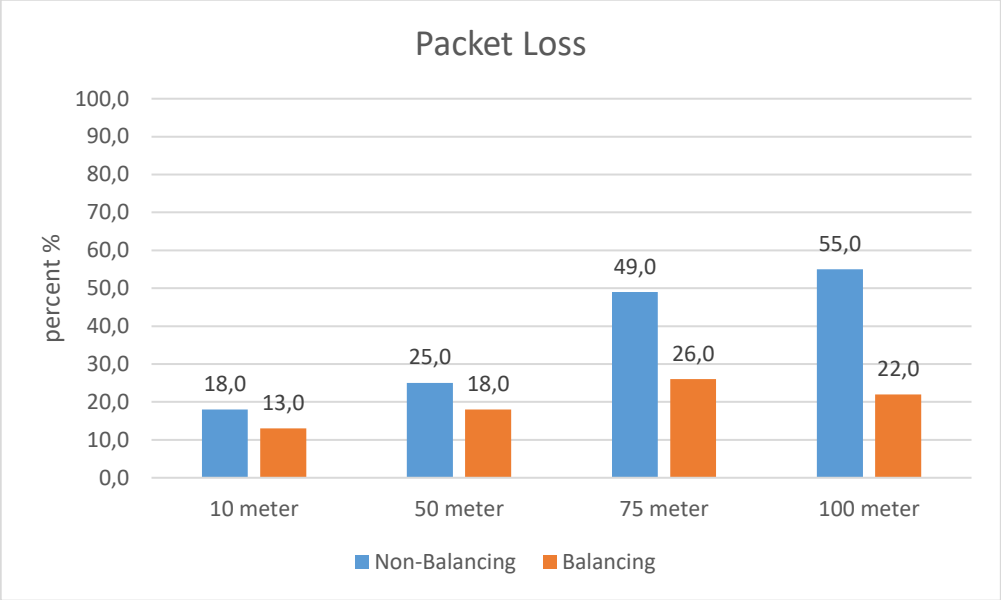


Gambar 4. 27 Throughput vs Distance

Analisa ➡ Kriteria yang diharapkan dari pengukuran ini adalah nilai throughput yang semakin besar merepresentasikan alokasi *bandwith* untuk mentrasfer data lebih besar ukurnnya. Analisa *Throughput* menunjukkan bahwa alokasi bandwith rule **rata-rata non-balancing** sebesar 46,9 dan bandwith pada **balancing rata – rata** 59,4. Artinya dari kedua rule performance diatas menunjukkan bahwa pada rule non-balancing menunjukkan *Throughput* nya kecil jika dibandingkan dengan rule balancing, karena dari rule balancing paket yang ditransmisikan melalui router akan dikontrol untuk meminimalisir paket yang loss.

4.8.3 Packet Loss

Dari hasil uji coba *packet loss* pada gambar 4.28 merupakan hasil pengukuran berdasarkan selisih dari jumlah paket yang berhasil dikirim dengan total paket keseluruhan yang terkirim. Dari pengukuran ini bertujuan untuk mengetahui hilang selama proses transmisi paket data. Berikut gambar grafik *packet loss*;

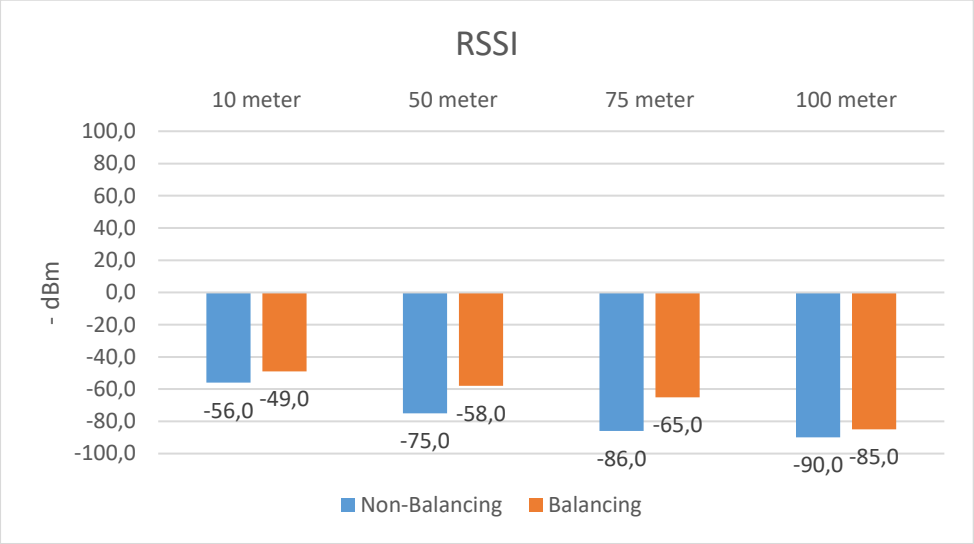


Gambar 4. 28 Packet Loss vs Distance

Analisa ➡ Kriteria yang diharapkan dari pengukuran ini adalah nilai packet loss dengan representasi nilai % yang kecil menunjukkan jumlah paket yang hilang atau rusak sedikit jumlahnya. Dalam pengukuran packet loss ini juga salah satu point pendukung dari aspek *reliability*. Jika nilai *reliability* relatif kecil, maka hal tersebut merepresentasikan performa jaringan yang tidak direkomendasikan. Analisa *packet loss* menunjukkan **rata-rata non-balancing** sebesar 37 % dan pada **balancing rata – rata** menunjukkan angka 20 %. Artinya dari kedua rule performance diatas menunjukkan bahwa pada rule **balancing** menunjukkan hasil pengukuran *packet loss* yang lebih kecil jika dibandingkan dengan **rule balancing**. Selisih nilai packet loss dari pengukuran diatas sebesar 20 %. Hal ini menunjukkan peningkatan aspek reliability yang signifikan. Dari analisa ini jarak juga memiliki pengaruh dalam proses transmisi paket data. Semakin jauh jarak antara titik pengiriman dan titik penerima, maka memberikan respon yang lama dan penurunan performa *packet loss.*

4.8.4 Receive Signal Strength (RSSI)

Dari hasil uji coba *packet loss* pada gambar 4.29 merupakan hasil pengukuran yang menunjukkan kekuatan sinyal yang diterima oleh perangkat dalam satuan -*dBm.* Berikut gambar grafik *RSSI;*

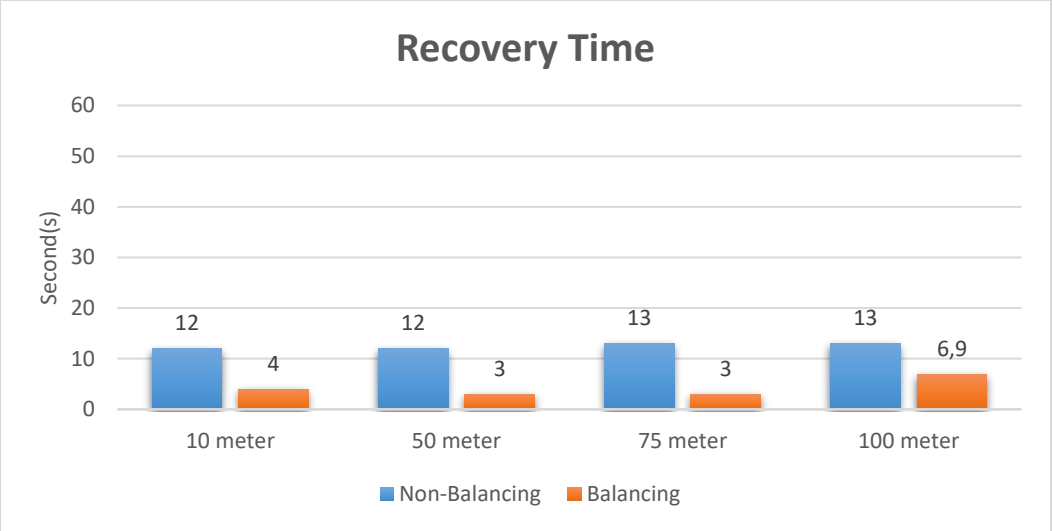


Gambar 4. 29 RSSI vs Distance

Analisa ➡ Kriteria yang diharapkan dari pengukuran ini adalah nilai *-dbm* yang semakin mendekati dengan nilai "0". Jadi jika nilai RSSI menunjukkan hasil minus yang semakin besar, merepresentasikan penyebaran sinyal radio perangkat *wireless sensor* semakin tidak direkomendasikan.

4.8.5 Recovery Time

Toleransi kesalahan dalam WSN adalah kemampuan jaringan untuk mentoleransi kesalahan yang menyebabkan kegagalan layanan. Ini adalah aspek ketahanan jaringan. Service ini berfokus pada pengukuran waktu deteksi kegagalan node atau mengamati *recovery time* saat terjadi kegagalan route.

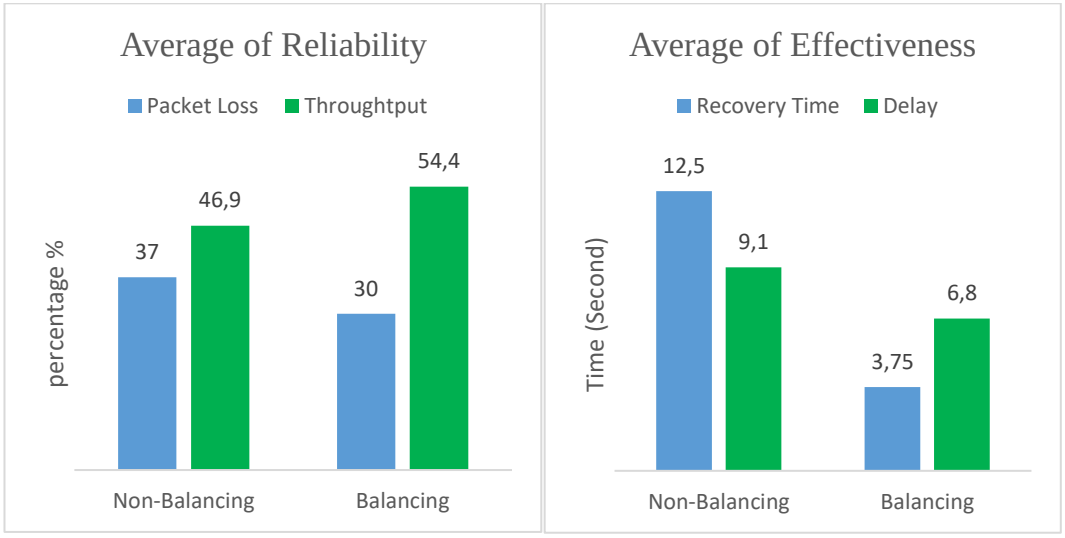


Gambar 4. 30 Recovery Time Vs Distance

Analisa ➡ Kriteria yang diharapkan dari pengukuran ini adalah nilai *recovery time (rt)* yang semakin mendekati dengan nilai 0 *second*. Jadi semakin cepat waktu pemulihan dalam memperbaiki masalah pada route yang rusak / error, maka tingkat efektifitas dari kinerja rule tersebut semakin baik dan direkomendasikan. Jika diperhatikan nilai *rt* dari kedua rule tersebut, rule balancing memiliki hasil pengukuran yang lebih baik dalam pemulihan saat rute terjadi kerusakan.

4.9. Analisis Data Pengujian Performa *Quality of Service* (QoS)

Hasil pengujian performa QoS ini memberikan bukti pendukung untuk menjamin performa jaringan yang lebih unggul, berikut ini acuan beberapa parameter performa dalam *Quality of Service*, *Delay Tolerance(delay)*, *Loss Tolerance (Packet Loss)*, *Capacity (Throughtput)*, dan *RSSI*. Berdasarkan table 4.4 hasil uji coba dan analisa QoS terbagi menjadi dua rule system. Pertama pada rule *non-balancing* menghasilkan rata-rata nilai *packet loss* sebesar 37 % dengan rata-rata delay sebesar 9,1 second, sedangkan untuk standar *throughput* Xbee sendiri masih dalam kategori rendah. Jika dibandingakan dengan hasil pengukuran balancing menghasilkan rata-rata nilai packet loss sebesar 20 %, dengan rata-rata delay sebesar 6,8 second. Dan jika diamati lebih detail hasil pengukuran RSSI pada rule balancing memberikan selisih nilai -12dBm yang relatif lebih baik kekuatan sinyalnya.



Gambar 4. 31 Perbandingan Hasil Rata-Rata Reliability dan Efektifitas

Hasil data perbandingan pada gambar 4.31 terdapat dua aspek utama yang menjadi acuan performa jaringan yaitu aspek *Reliability* dan *Effectiveness.* Aspek *Reliability* berdasarkan acuan kuantitas paket, sedangkan aspek *Effectiveness* berdasarka acuan waktu, sedangkan parameter RSSI (-dBm) tidak kita tampilkan dalam bentuk grafik, karena memiliki satuan pengukuran yang berbeda. Perhitungan yang kita sajikan dalam bentuk grafik ini adalah perhitungan secara rata-rata dari kedua parameter. Hasil yang didapatkan berdasarkan aspek *Reliability* memiliki selisih 17 % berdasarkan aspek *packet loss* baik secara ∑*PacketReceived* maupun ∑*PacketLoss.* Perhitungan ini didapatkan dari:

- **Average of Realiability**
  - Packet Loss

    $= \sum paket\ loss\ non\ balancing - \sum paket\ loss\ balancing$

    $= \mathbf{37}\ \% - \mathbf{20}\ \% = \mathbf{17}\ \%$
  - Throughput

    $= \sum throughput\ balancing - \sum throughput\ non\ balancing$

    $= \mathbf{54,4 - 46,9 = 12,6}\ \boldsymbol{bps}$

Sedangkan selisih *throughput* berdasarkan perhitungan diatas adalah 12,6 bps.

- **Average of Effectiveness**
  - Recovery Time

    $= \sum recovery\ time\ non\ balancing - \sum recovery\ time\ balancing$

    $= \mathbf{12,5 - 3,75 = 8,75}$ second
  - Delay

    $= \sum delay\ non\ balancing - \sum delay\ balancing$

    $= \mathbf{9,1 - 6,8 = 2,3}$ second

Hal ini menunjukkan bahwa system *balancing* memberikan kontribusi peningkatan pada aspek *Reliability.* Kemudian hasil yang didapatkan berdasarkan aspek *Effectiveness* menyatakan bahwa rule *balancing* lebih unggul pada parameter *"delay"* dan *"recovery time"* dibandingkan dengan rule non *balancing*, sebab pada saat menggunakan rule *non balancing* system membutuhkan waktu tambahan

sebesar 12,5 detik untuk melakukan *switching address* pada saat router terjadi masalah. Namun pada parameter "*recovery time" rule balancing* lebih cepet dibandingkan *non-balancing*, karena router pada rule *balancing* tidak perlu melakukan reset ulang pada *AT command* sehingga dapat mempersingkat waktu transmisi paket.

4.10. Comparison Non-Balancing and Balancing Measurement

Uji coba *non-balancing* ini berdasarkan pada aspek QoS yaitu *recovery time*, dan pada uji coba *balancing* berdasarkan aspek *reliability* yaitu menjaga kuantitas jumlah paket untuk meminimalisir jumlah paket yang hilang semakin banyak. Dalam uji coba ini kami melakukan pengiriman 100 paket dari titik end device ke coordinator dengan jarak awal 10 hingga 100 meter, kemudian kami secara bertahap memindahkan end device lebih jauh dari coordinator hingga sambungan terputus. Saat paket tiba pada router 1 kami mencoba mematikan router 1. Disaat inilah kami mengukur *recovery time* perangkat router untuk melakukan pencarian alternatif rute lain saat router mati, sedangkan pada uji coba balancing kami tidak melakukan pengukuran *recovery time* pada router, dikarenakan system balancing tidak akan bekerja apabila ada salah satu router yang mati. Berikut ini adalah table pengujian performa jaringan yang ditinjau dari aspek *RSSI, reliability, dan duration*. Berdasarkan hasil percobaan table 4.4 menghasilkan dua analisa performa jaringan dari penerapan *non-balancing* dan *balancing*, adapun hasil yang diperoleh dari penggunaan rule *system non-balancing* memberikan rata-rata waktu pengiriman paket dari titik end device sampai ke titik coordinator sebesar 9 sampai 10 menit.

Perbedaan dari hasil pengujian table 4.4 terletak $\sum$*PacketLoss* dimana pada penggunaan rule *balancing* memberikan jumlah paket lebih efektif dibandingkan dengan rule *non-balancing* yang mengalami banyak packet loss dikarena tidak adanya kontrol pada router. Berdasarkan standar parameter packet loss pada QoS jika packet loss terjadi sampai >25% termasuk dalam kategori "Poor" (Sugeng *et al.*, 2015).

[Halaman ini sengaja dikosongkan]

KESIMPULAN DAN SARAN

5.1. Kesimpulan

Berdasarkan hasil pengujian *dynamic routing* dan *system balancing* pada WSN yang menggunakan algoritma *AODV routing*, dan Mesh routing, kita membagi dalam dua kategori kesimpulan, yakni kesimpulan berdasarkan proses system secara keseluruhan, dan kesimpulan berdasarkan analisa pengujian, berikut pembagiannya.

5.1.1 Analisa Pengujian, dan Implementasi Penelitian

Sistem dynmic routing terbukti efektif dalam meningkatkan kualitas reliability jaringan WSN dibandingkan dengan menggunakan static routing yang tidak mampu memberikan alternatif rute lain saat router terjadi masalah. Dampak dari router yang tidak mampu memberikan alternatif rute saat terjadi masalah akan semakin memicu terjadinya packet loss. Oleh karena itu system dynamic routing yang diusulkan mampu meredam *packet loss* sebesar 17% , dimana perhitungan 17 % ini ditunjang dari dua aspek pengukuran berdasarkan QoS *packet loss* dan *throughput*, hal ini membuktikan bahwa system *dynamic routing* berhasil meningkatkan kualitas reliability jaringan WSN.

Penerapan sistem *load balancing* memberikan peningkatan pada aspek *throughput (capacity)*, dibandingkan dengan *non-balancing*. Kinerja *non-balancing* hanya meneruskan paket data saja tanpa adanya kontrol dan *self-healing* pada router, oleh karena itu saat router bebannya berlebihan maka diabiarkan saja tanpa ada respon sehingga dapat menurunkan kualitas performa jaringan. Dari hasil penelitian yang diusulkan mampu meningkatkan efektifitas jaringan pada aspek *recovery time* sebesar 8,75 *second* dan aspek *delay* sebesar 2,3 *second*.

5.2. Saran

Dalam penelitian akhir tesis ini memiliki beberapa saran yang bisa dikembangkan agar lebih kompleks dan efisien, antara lain :

1. Mengembangkan sejumlah titik tambahan pada sisi end device maupun router untuk bisa menjangkau kapasitas yang lebih kompleks dan luas.
2. Membangun sistem routing dengan mengutamakan konsumsi power yang digunakan agar terlihat sensor mana yang paling banyak bekerja.
3. Membangun sebuah *history* berupa *log* dari data sensor yang diterima dengan membuat *database*.
4. Data akhir yang diterima oleh *collector* (coordinator) bisa di bawa *ke fog computing* atau *cloud*, sehingga *history* berupa *log* bisa diakses dan di monitoring secara luas.

DAFTAR PUSTAKA

Affandi, A. and Setijadi, E. (2012) 'Rancang Bangun Server Learning Management System Menggunakan Load Balancer dan Reverse Proxy', 1.

Ahmed, M. (2012) 'Handshaking Problem Associated with Addressing Scheme for the Nodes of a Wireless Sensor Network', *International Journal of Advancements in Research & Technology*, 1(5), pp. 1–5.

Al-Karaki, J. N. and Kamal, A. E. (2004) 'Routing Techniques in Wireless Sensor Networks: A Survey', *IEEE Wireless Communications*, 11(6), pp. 6–27. doi: 10.1109/MWC.2004.1368893.

Anonim (2015) 'Modul 1 Komunikasi Nirkabel Menggunakan Modul RF X-Bee', in *Modul Praktikum Jaringan Telekomunikasi 2*. Komunikasi. Surabaya: Politeknik Elektronika Negeri Surabaya - PENS, pp. 1–11.

Ardiansyah, Y., Sarno, R. and Giandi, O. (2018) 'Rain Detection System for Estimate Weather Level Using Mamdani fuzzy Inference System', in *2018 International Conference on Information and Communications Technology, ICOIACT 2018*, pp. 848–854. doi: 10.1109/ICOIACT.2018.8350711.

Ashwini, V. S. *et al.* (2016) 'Architecting Wireless Sensor Mesh Networks for Connected Home and Home Security- using Raspberry pi B + and Zigbee', 5(4), pp. 245–253.

Bricker, G. and Harris, D. (2007) *Getting Started with XBee RF Modules, Network*.

Chaitanya, S. M. (2007) *ANALYSIS OF POWER CONSUMPTION OF AN END DEVICE IN A ZIGBEE MESH NETWORK*. The University of North Carolina at Charlotte.

Earl, B. (2018) *Memories of an Arduino, https://learn.adafruit.com/memories-of-an-arduino 2*.

Fanfang, D. *et al.* (2015) 'ZigBee Wireless Networking for the Interlligent Protection Center Research', *Preprints of the 5th international conference on Electric Utility Deregulation and Restructuring and Power Technologies*, pp. 7–10. doi: 10.1109/DRPT.2015.7432668.

Fuad, M. (2015) *'Rancang Bangun Wireless Sensor Network Berbasis Protokol Zigbee Dan Gsm Untuk Sistem Pemantauan Polusi Udara'*. Tesis. Sekolah Pascasarjana Institut Pertanian Bogor.

Gautam, G. and Sen, B. (2015) 'Design and Simulation of Wireless Sensor Network Topologies Using the ZigBee Standard', *International Journal of Computer Applications*, 113(16), pp. 14–16. doi: 10.5120/19910-2018.

Giandi, O. and Sarno, R. (2018) 'Prototype of Fire Symptom Detection System', in *2018 International Conference on Information and Communications Technology, ICOIACT 2018*, pp. 489–494. doi: 10.1109/ICOIACT.2018.8350730.

Guo, L. and Tang, Q. (2010) ‘An Improved Routing Protocol in WSN with Hybrid Genetic Algorithm’, *NSWCTC 2010 - The 2nd International Conference on Networks Security, Wireless Communications and Trusted Computing*, 2, pp. 289–292. doi: 10.1109/NSWCTC.2010.202.

Hartawan, I. N. B. and Desnanjaya, I. G. M. N. (2018) ‘Analisis Kinerja Protokol Zigbee Di Dalam Dan Di Luar Ruangan Sebagai Media Komunikasi Data Pada Wireless Sensor Network’, *Jurnal RESISTOR (Rekayasa Sistem Komputer)*, 1(2), pp. 65–72. doi: 10.31598/jurnalresistor.v1i2.320.

Hasta, Rulliyanto (2015) ‘Aplikasi Teknologi Komunikasi Wireless Berbasis Zigbee Pada Sistem Kontrol Dan Monitoring Ruangan Kelas’, 18(1), pp. 70–81.

Iqbal, M. (2015) *Rancang Bangun Wireless Sensor Network Berbasis Topologi Tree-Like Mesh Untuk Sistem Pemantauan Polusi Udara*. Tesis. Edited by S. P. Institut.

Jung, W. and Cho, S. H. (2010) ‘Load balancing system with sub-network management in wireless sensor networks’, *Proceedings - 2010 2nd IEEE International Conference on Network Infrastructure and Digital Content, IC-NIDC 2010*, (September), pp. 639–643. doi: 10.1109/ICNIDC.2010.5657859.

Karthikeyan, R. (2010) ‘Routing Strategy Selection for Zigbee Mesh Networks’, *International Journal of Communications, Network and System Sciences*, 03(07), pp. 608–611. doi: 10.4236/ijcns.2010.37081.

Khalaf, A. A. M. and Mokadem, M. S. A. (2017) ‘Effects of ZigBee Component Failure on the WSN Performance with Different Topologies Ashraf’, *Proceedings of the International Conference on Microelectronics, ICM*, (December), pp. 9–12. doi: 10.1109/ICM.2016.7847894.

Kim, Y., Moon, I. and Cho, S. (2009) ‘A comparison of improved AODV routing protocol based on IEEE 802.11 and IEEE 802.15. 4’, *Journal of Engineering Science and*, 4(2), pp. 132–141. Available at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.698.3192&rep=rep1&type=pdf.

Kumbhar, H. (2016) ‘Wireless Sensor Network using Xbee on Arduino Platform An experimental study’, *International Conference on Computing Communication Control and automation (ICCUBEA)*. doi: 10.1109/ICCUBEA.2016.7860081.

Mahmoud, K. H. (2013) *Data Collection and Processing from Distributed System of Wireless Sensors*. Masaryk Unviversity. Available at: http://is.muni.cz/th/395772/fi_m/MasterThesis.pdf.

Mihajlov, B. and Bogdanoski, M. (2011) ‘Overview and Analysis of the Performances of ZigBee- based Wireless Sensor Networks’, *International Journal of Computer Applications*, 29(12), pp. 28–35. doi: 10.5120/3704-5138.

Muamar, R. and Radiannor, R. (2018) *Handshaking, Serial dan Pararel Interfacing*. Yogyakarta.

Piyare, R. and Lee, S. (2013) ‘Performance Analysis of XBee ZB Module Based Wireless Sensor Networks’, *International Journal of Scientific & Engineering Research*, 4(4), pp. 1615–1621.

Rachman, F. Z., Yanti, N. and Hidayati, Q. (2017) ‘Implementasi Jaringan Sensor Nirkabel Zigbee Menggunakan Topologi Mesh Pada Pemantauan Dan Kendali Perangkat Ruang’, *Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK)*, 4(3), pp. 201–206. doi: 10.25126/jtiik.201743438.

Sabilla, S. I., Sarno, R. and Effendi, Y. A. (2018) ‘Optimizing time and cost using goal programming and FMS scheduling’, *2018 International Conference on Information and Communications Technology, ICOIACT 2018*, 2018-Janua, pp. 244–249. doi: 10.1109/ICOIACT.2018.8350727.

Saptadi, A. H. (2014) ‘Perbandingan Akurasi Pengukuran Suhu dan Kelembaban Antara Sensor DHT11 dan DHT22’, *Jurnal Infotel*, 6(2), pp. 49–55.

Sepriwono, R. (2018) *Analisis Perbandingan Waktu Konvergensi Routing Protokol Proaktif DSDV Terhadap Routing Protokol Proaktif OLSR Di Jaringan Bergerak AD HOC*. Universitas Sanata Dharma Yogyakarta.

Sharma, D., Verma, S. and Sharma, K. (2013) ‘Network Topologies in Wireless Sensor Networks: A Review’, *International Journal of Electronics and Communication Technology*, 4(3), pp. 93–97. doi: 2230-7109.

Shivapur, S., Kanakaraddi, S. G. and Chikaraddi, A. K. (2015) ‘Load Balancing Techniques in Wireless Sensor Networks : A Comparative Study’, 14(2), pp. 218–223.

Soijoyo, S. and Ashari, A. (2017) ‘Analysis of Zigbee Data Transmission on Wireless Sensor Network Topology’, *International Journal of Advanced Computer Science and Applications*, 8(9), pp. 145–151. doi: 10.14569/ijacsa.2017.080921.

Somprakash, B., Amitava, M. and Saha, D. (2003) ‘Location Management in Wireless Data Networks’, in *Database*. Artech House. Boston, London, p. 223. Available at: http://www.cse.wustl.edu/~jain/cse574-06/ftp/wireless_location.pdf.

Sugeng, W. *et al.* (2015) ‘The Impact of QoS Changes Towards Network Performance’, *International Jurnal of Computer Networks and Communications Security*, 3(2), pp. 48–53.

Suryanto, Y. I. and Zahra, A. A. (no date) ‘Analisis Kinerja Zigbee (802 . 15 . 4) Wsn Pada Topologi Tree Dan Star Mode Non Beacon Menggunakan Network Simulator 2’.

Yohanes, A. A. (2015) *Analisis Kecepatan Inisialisasi Jaringan Ad Hoc Pada Routing Protocol AODV, OLSR, dan ZRP Dengan NS2*. Universitas Sanata Dharma Yogyakarta.

Yusuf, O. (2014) *Determination of Optimal Power for ZigBee-Based Wireless Sensor Networks*. University of Windsor Windsor, Canada.

Zhang, L. and Chen, F. (2014) ‘A Round-robin Scheduling Algorithm of Relay-

nodes in WSN Based on Self-adaptive Weighted Learning for Environment Monitoring', *Journal of Computers*, 9(4), pp. 830–835. doi: 10.4304/jcp.9.4.830-835.

LAMPIRAN I

I. Source Code (End Device)

```
 #include <SoftwareSerial.h>
#include "DHT.h"
#include <stdlib.h>

SoftwareSerial xbee (3, 2);

#define router1H      "0013A200"
#define router1L      "4107043D"
#define router2H      "0013A200"
#define router2L      "417CE257"
#define broadcastH      "0"
#define broadcastL      "FFFF"

#define pinMQ2    A1
#define pinDHT    A0
#define DHTType DHT22 // DHT 22  (AM2302)
DHT dht(pinDHT, DHTType); // Initialize DHT sensor for normal 16mhz
Arduino

namespace{
  #define ED 3

#if ED == 1
  #define ED 1
  #define antri 1000
#elif ED == 2
  #define ED 2
  #define antri 2000
#elif ED == 3
  #define ED 3
  #define antri 3000
#endif
}

#define MAXDATA 50
#define MAXPAKET 100
#define tunggu 400

// TIMER SECOND
int MENIT, JAM, count, paketsend, datasend = 0;
int DETIK;
```

```
unsigned long previousMillis = 0;     // will store last time LED was
updated
const long interval = 1000;           // interval at which to blink
(milliseconds)
int timeout,jeda,jedanow = 0;

// VARIABEL GLOBAL
String Router1="";
String Router2="";
int readData;
String incomingByte; //Variable to store the incoming byte

float Temp;// = 0.0;
float Smoke = 0.0;

char buffTemp[5]="0";
char buffSmoke[5]="0";
char DataKirim [MAXDATA];

String outTemp="";
String outSmoke="";

bool Route=false;
bool Status=false;
int value;
bool SerialOK = false;
unsigned int Rout;

template<typename T, size_t N> size_t ArraySize (T (&) [N]){return N; }

void setup()
{
  Serial.begin(9600);
  xbee.begin(9600);
  dht.begin();

//  SerialOK = true;
  Serial.println("End Device Set Broadcast");
//  Rout = 1;
  SwitchRouter(broadcastH, broadcastL); delay(2000);
//  Serial.println("Cek Router 2");
//  Rout = 2;
//  SwitchRouter(router2H, router2L, Rout); delay(1000);
//  delay(tunggu);
  Status = false;
  timeout = 0;
  Serial.print("End Device ");
```

```
  Serial.print(ED);
  Serial.println(" Ready");
}

void loop()
{
  if(SerialOK == true)
  { SendData(); SerialOK = false; }

  if((xbee.available()>2)/* && (SerialOK == true)*/)
  {
    //delay(tunggu);
    incomingByte = "";
    incomingByte = xbee.readString();
    Status = true; timeout = 0; CekInput();
  }

void SendData()
{
  if(paketsend == MAXPAKET) { paketsend = 0; }
  paketsend++;
  Temp = dht.readTemperature();
  outTemp = dtostrf(Temp, 2, 2, buffTemp); // change float to string
  sprintf(DataKirim, "<R%d,ED%d : %s : %d>", Rout, ED,
outTemp.c_str(), paketsend);
//  delay(antri);
  xbee.print(DataKirim);
  String datanya = DataKirim; datasend =+ datanya.length();
  Serial.print(DataKirim); Serial.print(", "); Serial.println(datasend-2);
}
void CekInput()
{
  if((incomingByte.indexOf("FULL") >= 0))
  {
  }
  else if(incomingByte.indexOf("NEXT") >= 0)
  {
    if(int(incomingByte[3])-48 == ED)
    { Status = true; SerialOK = true; Rout = int(incomingByte[5])-48; }
  }
  else if(incomingByte.indexOf("READY") >= 0)
  {
    if(incomingByte[1] == '1')
    { Router1 = "R1 OKE";}
    else {Router2 = "R2 OKE";}
    Serial.println("Router Ready"); Status = true; SerialOK = true;}
  else
```

```
  {
    incomingByte="";
  }
  incomingByte=""; timeout = 0;
}
void SwitchRouter(String addrSH, String addrSL)
{
  timeout = 0;
  int n = 0;
  String atcom="";
  Serial.println("AT Mode");

  xbee.print("+++");
  while (xbee.available() == 0){
    timeout=timeout+1; delay(100);
    if(timeout == 10) { xbee.print("+++"); }
  }
  while (xbee.available() > 0) {
    atcom = "";
    atcom = xbee.readString(); Serial.println("+++ ");
  }
  delay(300);
  timeout = 0;
  Serial.println("Switch Address " + addrSH + addrSL);
  xbee.println("ATDH" + String(addrSH));
  while (xbee.available() == 0) {
    timeout=timeout+1; delay(100);
    if(timeout == 20) { xbee.println("ATDH" + String(addrSH));}
//    else (Serial.println(timeout));
  }
  while (xbee.available() > 0) {
    atcom = "";
    atcom = xbee.readString();
//    if (atcom.indexOf("OK") >= 0) {
    Serial.print("ATDH ");  Serial.println(atcom);
//    else { delay(1000); xbee.println("ATDH" + String(addrSH)); } timeout
= 0;
  }

  delay(300);
  timeout = 0;
  xbee.println("ATDL" + String(addrSL));
  while (xbee.available() == 0) {
    timeout=timeout+1; delay(100);
    if(timeout == 2000) { xbee.println("ATDL" + String(addrSL)); }
  }
  while (xbee.available() > 0) {
```

```
  atcom = "";
  atcom = xbee.readString();
  Serial.print("ATDL ");  Serial.println(atcom); timeout = 0;
 }

 delay(300);
 timeout = 0;
 xbee.println("ATWR");
 while (xbee.available() == 0) {
  timeout=timeout+1; delay(100);
  if(timeout == 2000) { xbee.println("ATWR"); }
 }
 while (xbee.available() > 0) {
  atcom = "";
  atcom = xbee.readString();
  Serial.print("ATWR ");  Serial.println(atcom); timeout = 0;
 }

 delay(300);
 timeout = 0;
 xbee.println("ATCN");
 while (xbee.available() == 0) {
  timeout=timeout+1; delay(100);
  if(timeout == 2000) { xbee.println("ATCN"); }
 }
 while (xbee.available() > 0) {
  atcom = "";
  atcom = xbee.readString();
  Serial.print("ATCN ");  Serial.println(atcom); timeout = 0;
 }
}
```

II. Source Code (Router)

```
#include <SoftwareSerial.h>
#include <stdio.h>
#include <string.h>

namespace{
#define maxpaket 10

#if maxpaket == 10
  #define MAXLENGTH 360
#elif maxpaket == 20
  #define MAXLENGTH 900
#elif maxpaket == 30
  #define MAXLENGTH 1055
#endif
}

#define Routerr 1
boolean Balancing = true;

#define coordH      "0013A200"
#define coordL      "414F8F0F"
#define broadcastH      "0"
#define broadcastL      "FFFF"

#define msglength 40

SoftwareSerial xbee (3, 2);

//Variables
bool started, ended = false;
char incomingByte ;
char msg[msglength];
byte bitPosition;

String datamasuk;
int lengthdata, paket, EndDev,pengiriman = 0;
String value, StatusRouter ="";
bool OKE,StatRouter;
unsigned int timeout = 0;
bool flagBoleh,full;

void setup()
{
  delay(2000);
  Serial.begin(9600);
  xbee.begin(9600);
```

```
//  Serial.println("Router Set Broadcast");
//  SwitchRouter(broadcastH, broadcastL); delay(2000);

  if(Balancing == 1)
  { Serial.println("Balancing Mode"); }
  else
  { Serial.println("No Balancing Mode"); }

  char addd[10];
  sprintf(addd, "Router %d Ready", Routerr);
  Serial.println(addd);

  OKE = false;
  StatRouter = false;
  if(Routerr == 2)
  {
    flagBoleh = false;
  }
  else
  { flagBoleh = true; }
  delay(1000);
}

void loop()
{
  if (xbee.available() > 0)
  {
    flagBoleh=false;
    timeout = 0;
    incomingByte = xbee.read(); delay(1);
    ReadData();
    if(started && ended)
    {
      value = String (msg);
//      Serial.println(value);

      if (int(value[1])-48 != Routerr)
      {
        EndDev == int(value[5])-48;
      }

      if(value.indexOf("FULL")>0)
      {
        Serial.print("R"); Serial.print(value[1]); Serial.println(" FULL");
        flagBoleh = true;
        delay(12000);
```

```
//        Serial.println(value);
        if(int(value[1])-48 != Routerr)
        {
//          flagBoleh = true;
          paket = 0;
          OKE = true; StatRouter = true;
        }
      }

      if(value.indexOf("NEXT")>0)
      {
//              Serial.print("R");  Serial.print(value[value.length()-5]);  Serial.println("
Request");
        if(int(value[value.length()-5])-48 != Routerr)
        {
//          Serial.println(value[value.length()-5]);
          paket = 0;
          OKE = false;
          StatRouter = false;
        }
      }
    }
    delay(800);
  }

  if(flagBoleh)
  { ReqED();}
  else if (full)
  { waiting(150,1);}
  else
  {value=""; waiting(80,1);}

}

void waiting(const int timee, const int i)
{
  timeout = 0;
  while (xbee.available() == 0)
  {
    timeout=timeout+1; delay(100);
    if(timeout == timee)
    {
      switch (i)
      {
        case 1:
          Serial.print("R"); Serial.print(Routerr);Serial.println(" Switch");
          flagBoleh = true;
```

```
        break;
       case 2:
        Serial.println("Status Router False");
        break;
       case 3:
        Serial.print("ED ");Serial.print(EndDev);Serial.println(" No Respon");
        break;
     }
     break;
   }
  }
 return;
}

void ReqED()
{
 if((paket <= maxpaket-1))// and (OKE == true))
 {
  flagBoleh = true;
  EndDev+=1;
  if(EndDev > 3) EndDev = 1;
  delay(1000);
  char next[11];
  sprintf(next,"<ED%dR%dNEXT>", EndDev, Routerr);
  xbee.print(next);
  Serial.print("R");Serial.print(Routerr);     Serial.print("     Request     ED     ");
Serial.println(EndDev);
  timeout = 0;
  waiting(30,3);
  while (xbee.available() > 0)
  {
   incomingByte = xbee.read(); delay(1);
   ReadData();
   timeout = 0;
  }

  value="";
  if(started && ended)
  {
   value = String (msg);
   Serial.print("Data Masuk = "); Serial.println(value);
   if (int(value[1])-48 == Routerr)
   {
    if ((value[3] == 'E') and (value[4] == 'D'))
    {
     paket++;
     datamasuk += value + "|";
```

```
        lengthdata += value.length();
        Serial.print("Panjang Bytes = "); Serial.print(value.length()); Serial.println("
bytes");
        Serial.print("Total Bytes = "); Serial.print(lengthdata); Serial.println("
bytes");
        Serial.print("Paket Ke = "); Serial.println(paket);
       }
      }

     if(value.indexOf("NEXT")>0)
     {
       if(int(value[value.length()-5])-48 != Routerr)
       {
         value.remove(0);
         if(Routerr != 1)
         {flagBoleh=false;}
       }
     }

     bitPosition = 0;
     msg[bitPosition] = '\0';
     started = false;
     ended = false;
     delay(500);
    }
  }
  else if((paket >= maxpaket-1))
  {
    flagBoleh = false;
    full = true;
    if (Balancing)
    {
      char full[8];
      sprintf(full,"<R%dFULL>", Routerr);
      xbee.println(full); delay(10);
    }
    delay(1000);
    SendData(); delay(4000);
    datamasuk = "";
    lengthdata = 0;
    OKE = false; StatRouter = false;
    paket = 0;
  }
}

void ReadData()
{
```

```
  if(incomingByte == '<')
  {
    started = true;
    bitPosition = 0;
    msg[bitPosition] = '\0';
  }
  else if(incomingByte == '>')
  {
    ended = true;
  }
  else
  {
    if(bitPosition < msglength)
    {
      msg[bitPosition] = incomingByte;
      bitPosition++;
      msg[bitPosition] = '\0';
    }
  }
}

void SendData()
{
  pengiriman+=maxpaket;
  Serial.println(); Serial.println("SEND DATA TO COORDINATOR :");
  char datakirim[MAXLENGTH];
  sprintf(datakirim, "(Co=R%d|%s)", Routerr, datamasuk.c_str()); delay(500);
//  Serial.print(sizeof(datakirim)); delay(1000);
  char buffer[40] = {0};
  size_t len = strlen(datakirim);
  size_t blen = sizeof(buffer)-1;
  size_t i = 0;

  for(i == 0; i <= len/blen; ++i)
  {
    memcpy(buffer, datakirim + (i*blen), blen);
//    puts(buffer);
    Serial.println(buffer); delay(600);
    xbee.print(buffer);
  }
  if (len % blen)
  {
    memcpy(buffer, datakirim + (len - len % blen), blen);
    //puts(buffer);
//    Serial.println(buffer);
  }
  Serial.print("Panjang Data= ");Serial.println(len);
```

```
  Serial.print("Total Paket Terkirim = "); Serial.println(pengiriman);

//  Serial.print("Paket= ");Serial.println(buffer);
}

void SwitchRouter(String addrSH, String addrSL)
{
  timeout = 0;
  int n = 0;
  String atcom="";
  Serial.println("AT Mode");

  xbee.print("+++");
  while (xbee.available() == 0){
    timeout=timeout+1; delay(100);
    if(timeout == 10) { xbee.print("+++"); }
  }
  while (xbee.available() > 0) {
    atcom = "";
    atcom = xbee.readString(); Serial.println("+++ ");
    //Serial.println(atcom);
//    if (atcom != "OK") { delay(1000); xbee.print("+++"); }
//    else {}
//      Serial.print("+++ "); Serial.println(atcom); timeout = 0;
  }
  delay(300);
  timeout = 0;
  Serial.println("Switch Address " + addrSH + addrSL);
  xbee.println("ATDH" + String(addrSH));
  while (xbee.available() == 0) {
    timeout=timeout+1; delay(100);
    if(timeout == 20) { xbee.println("ATDH" + String(addrSH));}
//    else (Serial.println(timeout));
  }
  while (xbee.available() > 0) {
    atcom = "";
    atcom = xbee.readString();
//    if (atcom.indexOf("OK") >= 0) {
    Serial.print("ATDH ");  Serial.println(atcom);
//    else { delay(1000); xbee.println("ATDH" + String(addrSH)); } timeout = 0;
  }

  delay(300);
  timeout = 0;
  xbee.println("ATDL" + String(addrSL));
  while (xbee.available() == 0) {
    timeout=timeout+1; delay(100);
```

```
  if(timeout == 2000) { xbee.println("ATDL" + String(addrSL)); }
 }
 while (xbee.available() > 0) {
  atcom = "";
  atcom = xbee.readString();
  Serial.print("ATDL ");  Serial.println(atcom); timeout = 0;
 }

 delay(300);
 timeout = 0;
 xbee.println("ATWR");
 while (xbee.available() == 0) {
  timeout=timeout+1; delay(100);
  if(timeout == 2000) { xbee.println("ATWR"); }
 }
 while (xbee.available() > 0) {
  atcom = "";
  atcom = xbee.readString();
  Serial.print("ATWR ");  Serial.println(atcom); timeout = 0;
 }

 delay(300);
 timeout = 0;
 xbee.println("ATCN");
 while (xbee.available() == 0) {
  timeout=timeout+1; delay(100);
  if(timeout == 2000) { xbee.println("ATCN"); }
 }
 while (xbee.available() > 0) {
  atcom = "";
  atcom = xbee.readString();
  Serial.print("ATCN ");  Serial.println(atcom); timeout = 0;
 }
}
```

III. Source Code (Coordinator)

```
/*
* PIN XBEE - ARDUINO MEGA
*      5V  - 5V
*     GND  - GND
*      RX  - 19
*      TX  - 18
*/

#include <stdio.h>
#include <string.h>

//Variables
String dataIn = "";
String dt[20] = "";
int lengthIn = 0;
bool started= false;//True: Message is started
bool ended = false;//True: Message is finished
bool statuss = false;
bool databaru = false;
String incomestring;
int k, pengiriman = 0;

void setup()
{
  Serial.begin(9600);
  Serial3.begin(9600);
  delay(2000);
  Serial.println("Coordinator Ready");
  statuss = true;
}

void loop()
{
  if(Serial3.available())
  {
    incomestring = Serial3.readString();
// ------------------------------------------------- Cek Data Selain Input Dari Router
    if(incomestring.indexOf("<") >= 0 )
    {
      incomestring.remove(0);
      databaru = true;
    }
    if((incomestring.indexOf("NEXT") >= 0 ))
    {
      databaru = false; incomestring.remove(0);
    }
```

```
// ------------------------------------------------- Data Yang Diambil
    if (incomestring[0] == '(')
    {   dataIn =""; databaru = false; started = true;}
    if (incomestring[incomestring.length() - 1] == ')') // Character terakhir String
adalah \0
    { databaru = false; ended = true; }

    if (started)
    {
      dataIn += incomestring;
      if(ended)
      {
//          Serial.println("Packet With End");
        dataIn.remove(dataIn.length() - 2);
        parsingData();
//          statuss = false;
      }
      if (databaru)
      {
//          Serial.println("Packet Without End");
        parsingData();
//          statuss = false;
      }
    }
  }
  else
  {incomestring = "";}
}

void parsingData()
{
  int i, j, m = 0;
  dt[j]="";
  dataIn.remove(0, 4);
  Serial.print("Data: "); Serial.println(dataIn);
    for(i = 3; i <= dataIn.length(); i++)
    {
      if (dataIn[i] == '|')
      {
        j++;
        dt[j]="";
      }
      else
      {
        dt[j] = dt[j] + dataIn[i];
      }
```

```
    }
    char Output[100];
    for(i = 0; i <= j; i++)
    {
      sprintf(Output, "Routing %c | %s", dataIn[1], dt[i].c_str());
      Serial.println(Output); m=m+dt[i].length();
      delay(10);
    }
   lengthIn=lengthIn+m-1;
   k = k+j+1;
   Serial.print("Panjang Data = "); Serial.println(m-1);
   Serial.print("Paket = "); Serial.println(j+1);
   Serial.print("Total Data = "); Serial.println(lengthIn);
//   Serial.print("Total Paket Seharusnya = "); Serial.println(pengiriman);
   Serial.print("Total Paket Diterima   = "); Serial.println(k);

   memset(dt, 0, sizeof(dt));
   memset(Output, 0, sizeof(Output));
   if(k + 10 >= 100)
   { k = 0; lengthIn = 0;}
   started = false;
   ended = false;
   databaru = false;
   dataIn = "";
}
```

BIOGRAFI PENULIS

A. Yusuf Ardiansyah is currently a teacher and working toward the Master of Technology Management degree from Institut Teknologi Sepuluh Nopember. He received his bachelor's degree in Computer Engineering from Electronic Engineering Polytechnic Institute of Surabaya (PENS), Surabaya, Indonesia in 2012. He has been working as Teacher in Computer and Network Engineering Department at SMK Rajasa Surabaya, Indonesia. His research interest includes Wireless Sensor Network, Internet of Thing, Network Analyst and Computer Networks.

E-mail : yusuf.17092@mhs.its.ac.id, yusufardian.tanz@gmail.com