



TESIS - KI142502

# **OPTIMASI PEMILIHAN CHILD *BROKER(S)* PADA MODEL KOMUNIKASI *PUBLISH/SUBSCRIBE* PADA PROTOKOL *DATA DISTRIBUTION SERVICE* DI AREA *MULTI-ZONE***

Dian Rahma Latifa Hayun  
NRP. 511320151

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

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

*[Halaman ini sengaja dikosongkan]*



TESIS - KI142502

***OPTIMIZATION OF CHILD BROKER(S) ELECTION IN  
PUBLISH/SUBSCRIBE COMMUNICATION MODEL IN DATA  
DISTRIBUTION SERVICE PROTOCOL IN MULTI-ZONE AREA***

Dian Rahma Latifa Hayun  
NRP. 5113201051

SUPERVISOR

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

MAGISTER PROGRAMME

INFORMATICS ENGINEERING DEPARTMENT

FACULTY OF INFORMATION TECHNOLOGY

SEPULUH NOPEMBER INSTITUTE OF TECHNOLOGY

SURABAYA

2016

*[Halaman ini sengaja dikosongkan]*

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

oleh:

Dian Rahma Latifa Hayun  
Nrp. 5113201051

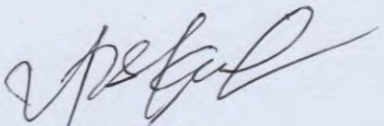
Dengan judul :

Optimasi Pemilihan Child Broker(S) Pada Model Komunikasi Publish/Subscribe Pada  
Protokol Data Distribution Service(Dds) Di Area Multi-Zone

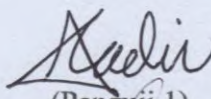
Tanggal Ujian : 22-6-2016  
Periode Wisuda : 2015 Genap

Disetujui oleh:

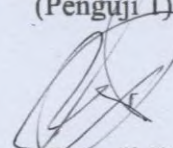
Waskitho Wibisono, S.Kom, M.Eng, Ph.D  
NIP. 197410222000031001

  
(Pembimbing 1)

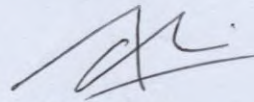
Dr.Eng. Radityo Anggoro, S.Kom, M.Sc  
NIP. 1984101620081210002

  
(Penguji 1)

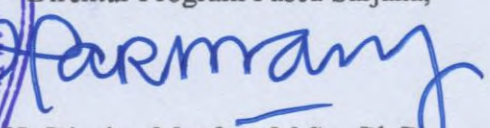
Tohari Ahmad, S.Kom, MIT, Ph.D  
NIP. 197505252003121002

  
(Penguji 2)

Baskoro Adi Pratomo, S.Kom, M.Kom  
NIP. 198702182014041001

  
(Penguji 3)



Direktur Program Pasca Sarjana,  
  
Prof.Ir.Djauhar Manfaat, M.Sc., Ph.D  
NIP. 196012021987011001

*[Halaman ini sengaja dikosongkan]*

# **Optimasi Pemilihan Child Broker(s) pada Model Komunikasi *Publish/Subscribe* pada Protokol *Data Distribution Service* di Area Multi-Zone**

Nama Mahasiswa : Dian Rahma Latifa Hayun

NRP : 5113 201 049

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

## **Abstrak**

Teknologi yang ada kini memunculkan sebuah paradigma bahwa mesin-mesin dan perangkat-perangkat pendukung yang ada harus dibangun dan dihubungkan sedemikian rupa agar tiap komponen dalam sistem dapat berinteraksi satu sama lain secara bebas dan lepas (*loosely coupled*) dan dapat menangani tantangan skalabilitas. Proses bisnis yang terjadi harus didistribusikan kepada beberapa *backend*, agar sistem tahan terhadap kegagalan atau perubahan yang terjadi.

Mekanisme komunikasi *Publish/Subscribe* merupakan mekanisme yang sesuai untuk domain permasalahan ini, karena mekanisme ini menyediakan fitur untuk menangani skalabilitas dan prosedur pengiriman data secara terpisah. OMG (*Object Management Group*) DDS (*Data Distribution Service*) merupakan sebuah spesifikasi standard dari *data centric publish/subscribe middleware* dengan banyak parameter QoS untuk memenuhi kebutuhan komunikasi yang ada.

Untuk mengoptimalkan kinerja DDS yang hanya bisa digunakan untuk komunikasi satu zona saja, maka beberapa penelitian telah dilakukan dengan menemukan mekanisme agar DDS dapat diimplementasikan pada area *multi-zone*. Beberapa penelitian yang ada menggunakan *broker* sebagai jembatan komunikasi antar zona. Akan tetapi penelitian-penelitian tersebut tidak membahas bagaimana mekanisme yang dilakukan agar node yang digunakan sebagai *broker* tersebut tidak mengalami kegagalan karena *overload*.

Oleh karena itu, penelitian ini mengusulkan sebuah mekanisme pemilihan asisten *broker* yang disebut *child broker(s)* pada area *multi-zone* yang bertujuan untuk mengurangi beban kerja pada *broker* dari tiap zona yang berbeda. Hal ini dilakukan

untuk meningkatkan skalabilitas dan mengurangi *network traffic* antar zona yang berbeda.

Dari hasil pengujian yang dilakukan, didapatkan penghematan penggunaan sumber daya CPU dan RAM pada *broker* yang mencapai 50% sebesar 10.49% dan 46747902.8 dibanding dengan metode tanpa pemilihan *child broker* secara acak untuk skenario 2 topik subskripsi. Didapatkan pula, metode yang diusulkan juga mampu menurunkan rata-rata penggunaan *bandwidth* hingga 64% sebesar 1.28 Mbps apabila dibandingkan dengan metode tanpa pemilihan *child broker*.

**Kata kunci:** *Publish/subscribe, DDS, Broker, OMG* .



***Optimization of Child Broker(s) Election in Publish/Subscribe  
Communication Model in Data Distribution Service Protocol(DDS) in  
Multi-Zone Area***

Student Name : Dian Rahma Latifa Hayun  
NRP : 5113 201 051  
Supervisor : Waskitho Wibisono, S.Kom, M.Eng., Ph.D.

**Abstract**

The existing technology has now led to a paradigm that machines and supporting devices should be constructed and connected in such a way that each component in the system can interact each other in a freely and loosely coupled way, and can handle the challenges of scalability. The business processes must be distributed to the multiple back end, so that the system is resistant to failures or changes.

Publish/Subscribe is an appropriate paradigm for this problem domain, because it provides a mechanism to handle the scalability features and data delivery procedures separately. OMG(Object Management Group) DDS(Data Distribution Service) is a standard specification of a data centric publish/subscribe middleware with QoS parameters to meet the needs of communication.

To optimize the performance of DDS that can only be used for communication in one zone, several studies have been conducted to find a mechanism, so that DDS can be implemented in a multi-zone area. Some existing research using a broker as a bridge of communication between the zones. However, these studies do not discuss how the mechanism is carried out so that the node used as the broker does not experience failures due to overload.

Therefore, this study proposes a mechanism for the selection of a broker's assistant called child broker(s) in a multi-zone area which aims to reduce the work load on the broker so feach different zone. It is done to improve the scalability and reduce network traffic between different zones.

From the test result, the proposed method decrease CPU and RAM resource usage untill 50% at 10.49% and 46747902.8 bytes compared to the conventional method without electing *child broker* for 2 topic subscriptions. The proposed method also decrease the bandwidth average usage till 64% from 2 Mbps for the conventional method without electing child broker to 1.28 Mbps.

**Keywords:** *Publish/subscribe, DDS, Broker, OMG*

## KATA PENGANTAR

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

Alhamdulillahirabil'amin, segala puji bagi Allah SubhanahuWata'alla, yang telah melimpahkan rahmat dan hidayah-Nya sehingga penulis bisa menyelesaikan Tesis yang berjudul "***OPTIMASI PEMILIHAN CHILD BROKER(S) PADA MODEL KOMUNIKASI PUBLISH/SUBSCRIBE PADA PROTOKOL DATA DISTRIBUTION SERVICE DI AREA MULTIZONE***" dengan baik.

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

1. Allah SWT atas limpahan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan Tesis ini dengan baik.
2. Kedua orang tua penulis, Bapak Moh Duhru dan Ibu Titi Nurhayati, serta kakak-kakakku, Dani, Rijal, Rika dan Reni, yang telah memberikan dukungan moral, spiritual dan material, semangat, perhatian, selalu setia dan sabar dalam menghadapi curhatan dari penulis, serta selalu memberikan doa yang tiada habisnya yang dipanjatkan untuk penulis.
3. Bapak Waskitho Wibisono, S.Kom., M.Eng., Ph.D. selaku dosen pembimbing yang telah memberikan kepercayaan, dukungan, bimbingan, nasehat, perhatian, serta semua yang telah diberikan kepada penulis.
4. Bapak Waskitho Wibisono, S.Kom, M.Eng., Ph.D, selaku ketua jurusan dan selaku dosen wali penulis, serta Bu Dr. Chastine Fatichah, S.Kom., M.Kom., dan segenap dosen Teknik Informatika yang telah memberikan ilmunya kepada penulis.
5. Pak Yudi, Pak Sugeng, Mbak Lina, Mas Kunto dan segenap staf Tata Usaha yang telah memberikan segala bantuan dan kemudahan kepada penulis selama menjalani kuliah di Teknik Informatika ITS.
6. Siska Arifiani yang telah banyak membantu penulis dalam pengurusan administrasi sidang dan wisuda pasca sarjana, semoga Allah yang membalas kebaikan kamu.
7. Ratri, Dika, Yustiana, Margareta, Virky, Nurina dan Faiza, terima kasih atas dukungan dan doa dari kalian.
8. Misbach, Annisa, dan Zarkasi, yang menjadi teman seperjuangan dan diskusi penulis selama di perkuliahan pasca sarjana.
9. Teman-teman dan admin Lab Pasca Sarjana Teknik Informatika ITS.
10. Seluruh teman Pasca Sarjana Teknik Informatika ITS angkatan 2013.
11. Juga tidak lupa kepada semua pihak yang belum sempat disebutkan satu per satu yang telah membantu terselesaikannya Tesis ini.

Kesempurnaan tentu sangat jauh tercapai pada Tesis ini, maka penulis mengharapkan saran dan kritik yang membangun dari pembaca.

Surabaya, Juli 2016

Penulis

*[Halaman ini Sengaja dikosongkan]*

## DAFTAR ISI

Lembar Pengesahan .....	v
Abstrak .....	vii
Abstract .....	ix
KATA PENGANTAR .....	xi
DAFTAR ISI .....	xiii
Daftar Gambar .....	xv
DAFTAR TABEL .....	xvii
BAB I PENDAHULUAN .....	1
1.1. Latar Belakang .....	1
1.2. Perumusan Masalah .....	3
1.3. Tujuan .....	4
1.4. Manfaat .....	4
1.5. Kontribusi Penelitian .....	4
1.6. Batasan Masalah .....	4
1.7. Sistematika Penulisan .....	4
BAB II KAJIAN PUSTAKA DAN DASAR TEORI .....	7
2.1. Data Distribution Service .....	7
2.2. Mekanisme Publish/Subscribe .....	8
2.2.1. Arsitektur Jaringan Publish/Subscribe .....	9
2.2.2. Event Distribution Scheme .....	10
2.2.3. Layanan Content-Aware Bridging untuk Mekanisme Publish/Subscribe .....	10
2.2.4. Semantic Overlay Network(SON) .....	13
BAB III METODOLOGI PENELITIAN .....	17
3.1. Perumusan Masalah .....	17
3.2. Studi Literatur .....	17
3.3. Pemetaan Skenario dan Konsep Kerja .....	18
3.3.1. Data Sampel Uji Coba .....	20
3.3.2. Fase Training .....	25
3.3.3. ManageEngine OpManagers .....	29
3.4. Implementasi .....	30
3.4.1. Pemrograman Node.js .....	31
3.4.2. Mosca <i>Middleware</i> .....	33

3.5	Lingkungan Pengujian .....	34
3.5.1.	Skenario Pengujian .....	35
3.5.2	Evaluasi Pengujian.....	35
BAB IV .....		37
HASIL DAN PEMBAHASAN .....		37
4.1.	Uji Coba .....	37
4.1.1.	Lingkungan Uji Coba.....	37
4.1.2.	Skenario Pengujian .....	38
4.1.3.	Parameter Pengujian .....	38
4.2.	Hasil Pengujian dan Analisis .....	39
4.2.1.	Hasil Pengujian Parameter Penggunaan CPU <i>Broker</i> .....	39
a.	Skenario 2 Topik Subskripsi .....	39
b.	Skenario 4 Topik Subskripsi .....	40
4.2.2.	Hasil Pengujian Parameter Penggunaan RAM Broker .....	41
a.	Skenario 2 Topik Subskripsi .....	41
b.	Skenario 4 Topik Subskripsi .....	42
4.2.3.	Hasil Pengujian Parameter Penggunaan <i>Bandwidth</i> .....	43
a.	Skenario 2 Topik Subskripsi .....	43
b.	Skenario 4 Topik Subskripsi .....	44
4.2.4.	Hasil Pengujian Parameter <i>Latency</i> .....	45
a.	Skenario 2 Topik Subskripsi .....	46
b.	Skenario 4 Topik Subskripsi .....	46
BAB V KESIMPULAN DAN SARAN .....		49
5.1.	Kesimpulan .....	49
5.2.	Saran.....	49
BIODATA PENULIS .....		53

## Daftar Gambar

Gambar 2. 1. Entitas dalam DDS(Jose, et al, 2012) .....	7
Gambar 2. 2. Time dan Space Decoupling pada Publish/Subscribe.....	8
Gambar 2. 3. Skema Umum Mekanisme <i>Publish/Subscribe</i> (Esposito, et al, 2013).....	9
Gambar 2. 4. Arsitektur DDS .....	11
Gambar 2. 5. Komponen Lapisan <i>Routing</i> pada DDS-IS(Jose, et al, 2012).....	12
Gambar 2. 6. Diagram Sekuens pada Proses <i>Discovery Event</i> .....	13
Gambar 2. 7. SON pada Jaringan P2P .....	14
Gambar 2. 8. SON pada Studi Kasus Bencana Alam .....	15
Gambar 2. 9. <i>Tree</i> yang dibentuk oleh SON(Vazirgiannis, et al, 2005).....	15
 Gambar 3. 1. Konsep Kontribusi .....	19
Gambar 3. 2. Tahap Pemilihan <i>Child Broker</i> .....	20
Gambar 3. 3. Contoh Data Sampel Uji Coba.....	23
Gambar 3. 4. Proses Pembentukan SON .....	25
Gambar 3. 5. dKlasifikasi Hierarki .....	26
Gambar 3. 6. Kecenderungan Data Atribut terhadap Waktu(s) sebelum Normalisasi ...	28
Gambar 3. 7. Grafik Kecenderungan Data yang telah Ternormalisasi .....	29
Gambar 3. 8. <i>Monitoring Dashboard</i> ManageEngine OpManager .....	30
Gambar 3. 9. Pengaturan <i>threshold</i> pada ManageEngine OpManager.....	31
Gambar 3. 10. Arsitektur Rancangan Sistem Scalable DDS .....	32
Gambar 3. 11. Contoh Pemrograman Node.js “Hello World!” .....	33
Gambar 3. 12. Hasil <i>Compile</i> Program “Hello World!” .....	33
Gambar 3. 13. Contoh Program pada Mosca Menggunakan Mongo .....	34
 Gambar 4. 2. Hasil Pengujian Penggunaan CPU Broker untuk 2 Topik .....	40
Gambar 4. 3. Hasil Pengujian Penggunaan CPU Broker untuk 4 Topik .....	41
Gambar 4. 4. Hasil Pengujian Penggunaan RAM Broker untuk 2 Topik.....	42
Gambar 4. 5 Hasil Pengujian Penggunaan RAM Broker untuk 4 Topik.....	43
Gambar 4. 6 Hasil Pengujian Penggunaan <i>Bandwidth Broker</i> untuk 2 Topik.....	44
Gambar 4. 7. Hasil Pengujian Penggunaan <i>Bandwidth Broker</i> untuk 4 Topik.....	45
Gambar 4. 8. Hasil Pengujian <i>Latency</i> untuk 2 Topik.....	46
Gambar 4. 9. Hasil Pengujian <i>Latency</i> untuk 4 Topik .....	47

*[Halaman ini sengaja dikosongkan]*



## DAFTAR TABEL

Tabel 3. 1. Tabel Data Training .....	26
Tabel 3. 2 Hasil Regresi Data Sebelum Proses Normalisasi .....	27
Tabel 3. 3. Hasil Regresi Setelah Normalisasi.....	28
Tabel 3. 4. Koefisien Regresi.....	29

*[Halaman ini sengaja dikosongkan]*

# BAB I

## PENDAHULUAN

Pada Bab 1 dijelaskan tentang studi dasar dalam pembuatan proposal penelitian yang terdiri dari latar belakang permasalahan, rumusan masalah, tujuan, manfaat, kontribusi penelitian serta batasan permasalahan dalam penelitian.

### 1.1. Latar Belakang

Keberadaan teknologi internet telah membawa perubahan terhadap skala dari sistem pendistribusian data. Teknologi yang ada kini memunculkan sebuah paradigma bahwa mesin-mesin dan perangkat-perangkat pendukung yang ada harus dibangun dan dihubungkan sedemikian rupa agar tiap komponen dalam sistem dapat berinteraksi satu sama lain secara bebas dan lepas (*loosely coupled*) dan dapat menangani tantangan skalabilitas. Proses bisnis yang terjadi harus didistribusikan kepada beberapa *backend*, agar sistem tahan terhadap kegagalan atau perubahan yang terjadi.

Mekanisme komunikasi *Publish/Subscribe* merupakan mekanisme yang sesuai untuk domain permasalahan ini, karena mekanisme ini menyediakan fitur untuk menangani skalabilitas dan prosedur pengiriman data secara terpisah. Maknanya, komunikasi antara pengguna yang meminta data dengan penyedia data terjadi secara asinkronous.

Akan tetapi mekanisme *publish/subscribe* selain harus mampu menghadapi tantangan skalabilitas juga harus mampu memenuhi syarat interoperabilitas yakni mampu berinteraksi dengan aplikasi lain melalui suatu protokol tertentu yang telah disepakati bersama dan dapat diandalkan ketika berada dalam lingkungan yang heterogen dan dinamis. Singkatnya, perkembangan teknologi kini membutuhkan sebuah mekanisme *publish/subscribe* yang memenuhi beberapa QoS (*Quality of Service*). OMG (*Object Management Group*) DDS (*Data Distribution Service*) merupakan sebuah spesifikasi standard dari *data centric publish/subscribe middleware* dengan banyak kebijakan QoS untuk memenuhi kebutuhan komunikasi yang terjadi.

Pada umumnya, DDS hanya bekerja pada satu zona saja, dimana sumber data (*publisher*) dan *subscriber* berada pada lokasi jaringan yang sama. DDS menggunakan teknik *multicast* untuk menemukan *endpoint* (sebagai contoh: *publisher* atau *subscriber*) dalam sebuah sistem. Jika ternyata suatu *endpoint* terletak pada suatu

jaringan yang terisolasi atau *private* yang tidak mendukung *multicast*, maka *endpoint* tersebut tidak akan ditemukan oleh *endpoint* yang lain. Disamping itu, karena adanya *firewall* atau NAT(*Network Address Translation*), meski *endpoint* ditemukan bisa jadi tidak terjadi pertukaran data.

Corradi dan Foschini menyediakan mekanisme REVENGE(Corradi, et al, 2010) yang merupakan *middleware* untuk mendistribusikan berita antar klien yang heterogen yang memenuhi prinsip ketersediaan dan reliabilitas. REVENGE memiliki dua kontribusi yaitu ruting untuk memfasilitasi pengiriman data antar *mobile-nodes*. REVENGE juga memungkinkan tiap node untuk saling berkomunikasi dengan node yang berada pada domain DDS yang berbeda. Tetapi mekanisme ini harus terikat dengan topik tertentu sehingga membatasi fleksibilitas sistem. Park mengajukan sebuah arsitektur untuk menghubungkan DDS dengan ESB(*Enterprise Service Bus*) yang memungkinkan sistem untuk saling bertukar data antara DDS dengan *enterprise system*(Park, et al, 2011). Hanya saja, fokus penelitian tersebut hanya pada poin interoperabilitas dan tidak membahas bagaimana mekanisme untuk meningkatkan skalabilitas sistem.

Oleh karena itu beberapa penelitian sebelumnya mencoba menyelesaikan permasalahan ini. Beberapa solusi DDS *broker* telah ada untuk menangani isu ini. Pada (Jose, et al, 2012) peneliti menghadirkan sebuah mekanisme untuk menghubungkan DDS yang berada pada dua jaringan zone yang berbeda. Tiap zona memiliki satu *broker* yang digunakan sebagai jembatan penghubung antara dua zona tersebut. Pada penelitian tersebut, *broker* bertugas untuk menangani permintaan *subscription* dari para *subscriber* dan meneruskan *subscription* tersebut ke zona yang kedua jika ternyata pada zona yang sama tidak ditemukan *publisher* yang sesuai.

Sayangnya, pada (Jose, et al, 2012) tidak membahas kebijakan komunikasi antar *broker* dan manajemen pengiriman data. Sehingga memungkinkan terjadinya beban jaringan yang terjadi antar broker. Selain itu, pada penelitian ini hanya terbatas pada dua jaringan/zona yang berbeda, sedangkan pada implementasinya dibutuhkan komunikasi dalam skala WAN(*Wide Area Network*). Hal ini berarti membutuhkan lebih banyak *broker* sehingga membutuhkan kebijakan tersendiri untuk melakukan pengaturan komunikasi antara *broker*, *subscriber*, dan *publisher*. Ditambah lagi, *broker* yang menjadi *central-node* akan semakin sibuk karena memiliki fungsional tambahan,

sehingga dibutuhkan mekanisme pemilihan asisten *broker* sebagai pengganti *main broker* jika mengalami *node-failure*.

Oleh karena itu, penulis mengusulkan sebuah penelitian untuk pemilihan asisten *broker* yang disebut sebagai *child broker(s)* pada area multi-zone yang bertujuan untuk mengurangi beban kerja pada *broker* dari tiap zona yang berbeda. Hasil yang diharapkan dari penelitian ini adalah berkurangnya trafik jaringan antar *broker*, menghindari terjadinya *single node failure* dan peningkatan skalabilitas pada DDS dengan pendekatan *single overlay network*.

## **1.2. Perumusan Masalah**

Tugas *broker* yang merupakan *central-node*, sebagai pengelola *subscription* dan *publication* akan mengurus sumber daya yang dimiliki oleh *node broker*. Ditambah lagi dalam konsep multi-zone, *broker* juga bertugas sebagai jembatan antar satu zona dengan zona yang lain. Yang ini berarti menambah beban kerja *broker*. Sehingga hipotesa yang diusulkan untuk mengatasi permasalahan tersebut adalah “seandainya tiap *broker* memiliki asisten *broker(child broker(s))* yang membantunya melakukan pengelolaan *subscription* dan *publication*”, maka akan mengurangi beban kerja broker dan mengurangi trafik jaringan antar *broker* dan menambah skalabilitas sistem serta menghindari *overload* pada *broker*.

Rumusan masalah pada penelitian ini dapat dijabarkan menjadi beberapa poin rumusan berikut:

1. Bagaimana mengurangi beban kinerja *main broker* pada mekanisme *publish/subscribe* dengan memilih *child broker* yang akan membantu melakukan pengelolaan terhadap publikasi dan subskripsi yang terjadi.
2. Bagaimana mekanisme untuk menentukan *child broker* yang akan membantu *main broker* sehingga *main broker* akan terhindar dari beban berlebih.
3. Bagaimana menentukan jumlah *child broker* yang harus dipilih agar pertukaran data dan komunikasi yang terjadi antara *publisher* dan *subscriber* menjadi optimal.

### 1.3. Tujuan

Tujuan dari penelitian ini adalah menghasilkan sebuah perbaikan mekanisme komunikasi antara *publisher*, *subscriber* dengan *broker* dalam domain DDS pada area *multi-zone network* agar mengurangi beban kerja *broker* sebagai *central-node*.

### 1.4. Manfaat

Karena adanya pengurangan beban kerja pada *main broker*, maka manfaat yang dihasilkan dari penelitian ini adalah mengurangi kepadatan jaringan dan meningkatkan skalabilitas sistem.

### 1.5. Kontribusi Penelitian

Kontribusi penelitian ini adalah adanya pemilihan asisten *broker* sebagai pembantu *main broker* dalam melakukan pengelolaan *publishing* dan *subscribing* yang dilakukan oleh *publisher* dan *subscriber* untuk mengurangi beban kerja *main broker*.

### 1.6. Batasan Masalah

Batasan masalah pada penelitian ini adalah:

1. Penerapan solusi hanya pada domain permasalahan DDS
2. Pengujian dilakukan pada studi kasus bencana alam
3. Implementasi menggunakan bahasa pemrograman Java dan Javascript
4. Socket middleware yang digunakan adalah Node.js
5. Middleware yang digunakan adalah Mosca dan freedomjs
6. IDE yang digunakan adalah WebStorm
7. *Subscriber* berupa desktop client, web client ataupun server
8. Node yang dapat dipilih untuk menjadi *child broker* adalah node yang berupa server atau memiliki konfigurasi web server
9. Penelitian ini tidak mencakup mekanisme untuk menghindari atau menangani kondisi *failure* pada *subscriber node*

### 1.7. Sistematika Penulisan

Pada Bagian ini dipaparkan secara singkat sistematika penulisan bab-bab yang terdapat pada laporan Tesis ini untuk memudahkan pembacaan. Sistematika tersebut disusun sebagai berikut:

1. Bab I Pendahuluan. Bab ini berisi pendahuluan yang menjelaskan latar belakang, perumusan masalah, tujuan, hipotesis, batasan masalah, tujuan, manfaat, kontribusi penelitian dan sistematika penulisan terkait dengan penelitian.

2. Bab II Kajian Pustaka dan Dasar Teori. Bab ini berisi tinjauan pustaka yang mengacu pada dasar-dasar teori terkait penelitian, *publish/subscribe*, *data distribution service*, *single overlay network*, *OMG*, *event dissemination*.
3. Bab III Metodologi Penelitian. Bab ini berisi tahapan penelitian meliputi analisis dan rancangan sistem yang dikembangkan, skenario pengujian dan analisis pengujian terhadap penelitian ini.
4. Bab IV Hasil Penelitian dan Pembahasan. Bab ini memaparkan tentang tahapan pengujian dan pembahasan hasil yang dilakukan. Tahapan pengujian meliputi skenario pengujian, variabel pengujian, dan lingkungan pengujian. Pembahasan hasil pengujian meliputi analisis hasil yang didapat dari pengujian secara nyata.
5. Bab V Kesimpulan dan Saran. Bab ini memuat kesimpulan yang dapat diambil dari penelitian yang dilakukan, serta disampaikan juga hal-hal yang perlu dikembangkan dari penelitian yang telah dilakukan untuk penelitian lebih lanjut.

*[Halaman ini sengaja dikosongkan]*



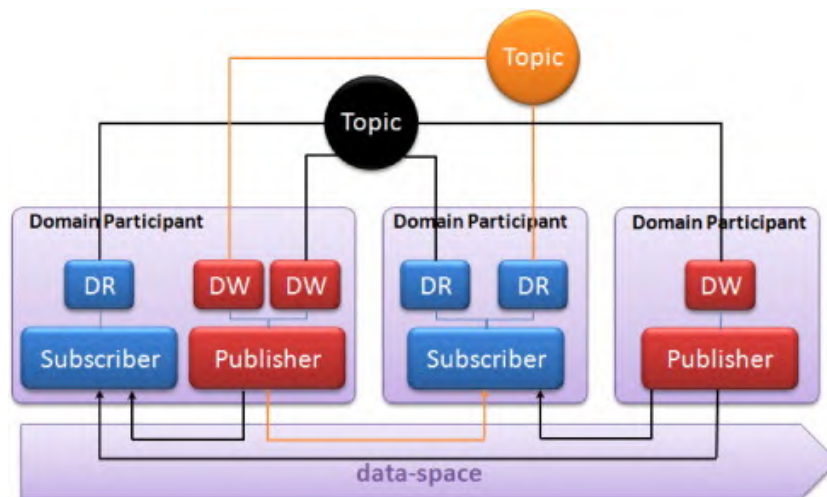
## BAB II

### KAJIAN PUSTAKA DAN DASAR TEORI

Pada Bab 2 ini dijelaskan dasar teori dan kajian pustaka yang terkait dengan penelitian. Dasar teori dan kajian pustaka yang dijelaskan meliputi DDS, mekanisme *publish/subscribe*, *event dissemination*, *Semantic Overlay Network*.

#### 2.1. Data Distribution Service

DDS (OMG, 2006) merupakan sebuah model pemrograman *Data-Centric Publish Subscribe*(DCPS) dan sebuah API untuk mengembangkan aplikasi yang terdistribusi. Berdasarkan DCPS, *publishers* dan *subscribers* saling bertukar data dengan secara sederhana melakukan pembacaan dan penulisan data yang ditandai dengan “topik” dan sebuah “key”. Dalam DDS dikenal istilah *data-space*, yakni memori yang terdistribusi yang konten tiap memori atau tempat penyimpanan tersebut diakses dan diperbaharui oleh sebuah entitas *endpoint* yang tiap dari entitas tersebut memiliki topik yang berbeda, yang disebut sebagai *DataWriter*(DW) dan *DataReader*(DR). Gambar 2.1. menunjukkan hubungan antar entitas dalam sebuah DDS.



**Gambar 2. 1. Entitas dalam DDS(Jose, et al, 2012)**

Masing-masing DW/DR memiliki QoS dan *data-chace* masing-masing. Sebagai contoh jika terdapat DR yang ingin meminta data dari DR, maka masing-masing dapat menentukan waktu maksimum pengiriman data (*Deadline* QoS) atau melakukan penyaringan data sesuai topik masing-masing. Jika *middleware* mendapati adanya satu

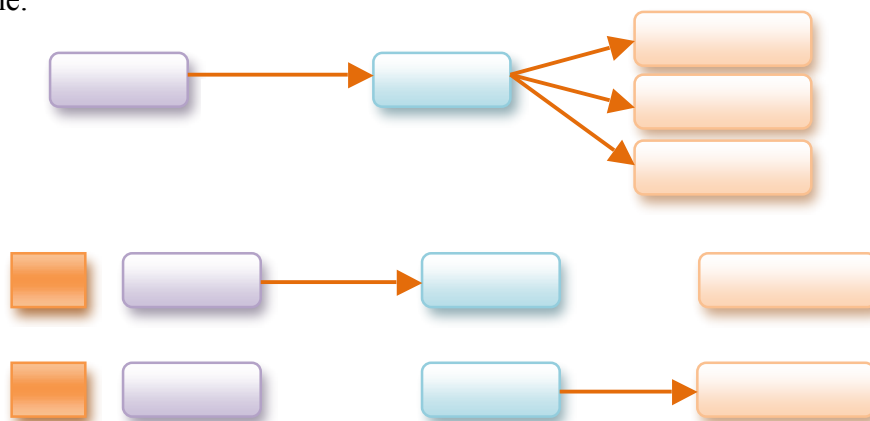
pelanggaran dari *rule* yang telah disepakati, maka *middleware* akan melakukan aksi tertentu.

Pada dasarnya, DDS hanya diimplementasikan pada sistem pendistribusian data dimana tiap node yang akan melakukan pertukaran data berada pada satu lokasi yang sama atau terletak pada satu jaringan LAN. Namun, pada perkembangannya DDS memungkinkan terjadinya komunikasi antar jaringan yang berbeda.

## 2.2. Mekanisme Publish/Subscribe

Esposito menjelaskan bahwa *Publish/Subscribe* merupakan mekanisme atau paradigma komunikasi untuk sistem terdistribusi. Tiap node yang berada pada jaringan *publish/subscribe* saling berkomunikasi dengan mengirimkan (*publishing*) data dan menerima (*subscribing*) data secara anonim. Pada umumnya, properti yang dibutuhkan oleh sebuah *publisher* agar dapat berkomunikasi dengan sebuah *subscriber* adalah nama dan definisi data. *Publisher* tidak memerlukan informasi apapun tentang *subscriber*, begitu juga sebaliknya (Esposito, et al, 2013).

Mekanisme *publish/subscribe* menganut 3 prinsip. Yang pertama adalah *space decoupling*. Prinsip pertama ini bermakna bahwa interaksi yang terjadi antara *pulisher* dan *subscriber* terjadi secara anonim. Antara *publisher* dan *susbcriber* tidak saling mengetahui informasi tentang satu sama lain. Yang kedua adalah *time decoupling*. Hal ini berarti antara *publisher* dan *subscriber* tidak harus aktif dalam waktu yang sama. *Publisher* dapat melakukan publikasi beberapa *event* meski para *subscribers* tidak sedang online.



**Gambar 2. 2. Time dan Space Decoupling pada Publish/Subscribe**

Ketika *subscriber* telah aktif kembali, maka dia akan mendapatkan pesan notifikasi akan adanya *event* yang baru. *Publisher* tetap bisa memproduksi *events* meskipun *subscriber* sedang tidak aktif. Yang ketiga adalah *Synchronization decoupling*. Hal ini berarti bahwa komunikasi antara *publisher* dan *subscriber* terjadi secara asinkronous (Esposito, et al, 2013).

Gambar 2.2. menunjukkan gambaran prinsip *time* dan *space decoupling* yang disediakan oleh mekanisme *publish/subscribe*.

### 2.2.1. Arsitektur Jaringan Publish/Subscribe

Pada Gambar 2.3. terlihat bahwa arsitektur dasar dari *publish/subscribe* terdiri dari 3 komponen. Yaitu *subscriber*, *broker*, dan *publisher*. *Subscriber* adalah pihak yang meminta data tertentu dengan memanggil fungsi *subscribe()*. Subskripsi yang dilakukan oleh *subscriber* tersebut ditujukan kepada *broker* untuk dikelola dan diteruskan kepada penyedia data yang mempublikasikan datanya (*publisher*) (RTI, 2012). *Broker* (Esposito, et al, 2013) setidaknya melakukan beberapa tugas berikut.

- Menyimpan subskripsi
- Menerima dan menampung *event* yang dipublikasikan oleh *publisher*
- Menghubungkan antara *subscriber* dengan *publisher* sebagai penyedia data yang memenuhi syarat yang diberikan oleh *subscriber*.



Gambar 2. 3. Skema Umum Mekanisme *Publish/Subscribe* (Esposito, et al, 2013)

### 2.2.2. Event Distribution Scheme

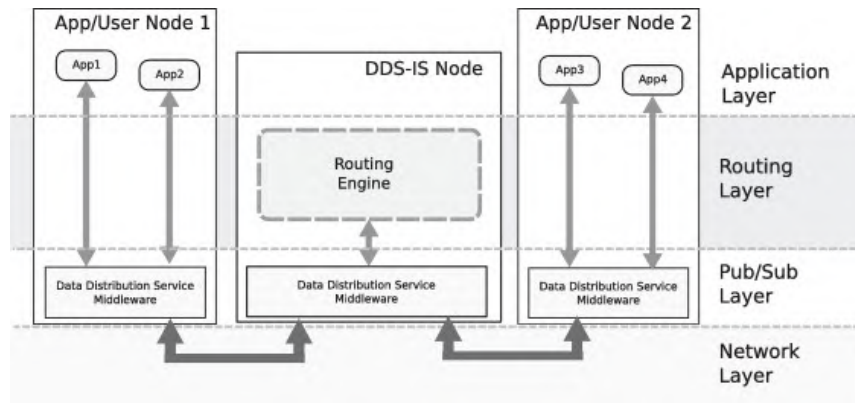
Sistem *publish/subscribe* harus mendukung komunikasi antara *publisher* dan *subscriber* yang terdistribusi letaknya dan dalam jumlah yang besar. Liu menjelaskan bahwa ketika sistem berkembang, maka secara otomatis jumlah *subscriber* dan *publisher* juga akan bertambah. Maka hal ini mendorong adanya kebutuhan terhadap suatu skema pendistribusian *event* agar data disebarikan secara merata dan tepat pada sasaran. Pada umumnya, skema ini dilakukan dengan teknik *multicast* dengan menyebarkan pesan dari satu *broker* kepada *subscriber* yang mensubskripsi data pada topik tertentu (Liu, et al, 2003).

Pentingnya skema ini adalah guna memastikan bahwa *event* bisa diterima masing-masing *subscriber* atau *publisher*. Harus ada kepastian bahwa sebuah *subscriber* akan menemukan paling tidak satu *publisher* yang memenuhi kriteria data yang ia minta.

### 2.2.3. Layanan Content-Aware Bridging untuk Mekanisme Publish/Subscribe

DDS merupakan sebuah *middleware* yang digunakan untuk mengatasi pertukaran data pada sistem terdistribusi yang bersifat *real-time*. Dua permasalahan utama yang muncul ketika DDS diimplementasikan adalah heterogenitas dan skalabilitas sistem. Seiring dengan berkembangnya kebutuhan sistem yang semakin kompleks, bertambahnya jumlah *subscriber* dan *publisher* yang bisa jadi terletak di beberapa zona yang berbeda membutuhkan sebuah scenario baru DDS yang mendukung area atau ruang lingkup yang lebih besar.

Dari permasalahan di atas, pada (Jose, et al, 2012) menjelaskan bahwa butuh adanya sebuah layanan yang meningkatkan skalabilitas DDS dan interoperabilitas dari aplikasi DDS. Pada penelitian (Jose, et al, 2012), dibangun sebuah layanan yang mampu menjembatani komunikasi beberapa aplikasi DDS di dua zona yang berbeda. DDS-IS (Jose, et al, 2012) terdiri dari empat *layer*, yani *layer* aplikasi, *routing*, pub/sub, dan *network layer*. Gambar 2.4. menunjukkan arsitektur dari DDS-IS.

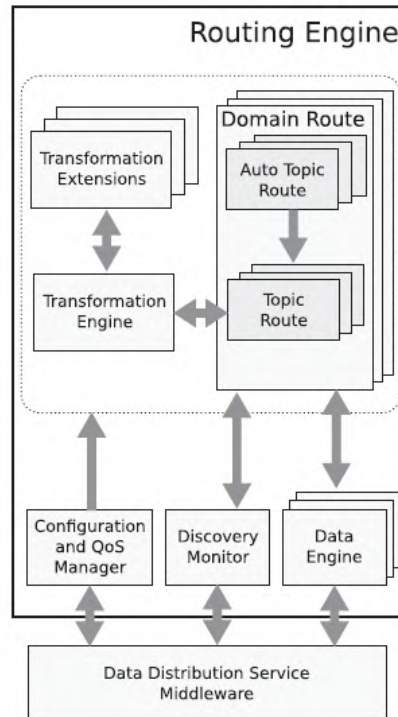


**Gambar 2. 4. Arsitektur DDS**

Lapisan utama dari DDS-IS adalah lapisan *routing*. Seperti yang tampak pada Gambar 2.5., lapisan *routing* ini terdiri dari beberapa komponen. Komponen-komponen utama dari lapisan *routing* tersebut adalah sebagai berikut:

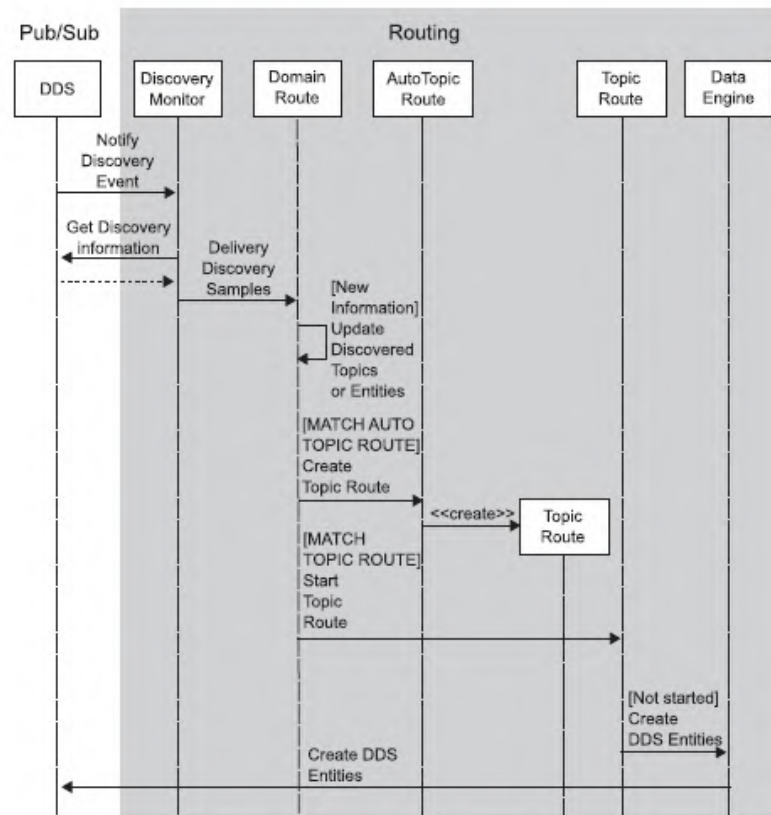
- *Discovery Monitor*: Komponen ini bertugas untuk mendeteksi adanya *subscriber* atau *publisher* baru yang bergabung dalam sistem. Setiap kali ada perubahan, maka komponen ini akan mengirimkan notifikasi ke *Domain Route*.
- *Domain Route*: *Domain route* berisi informasi *routing* antara dua zona DDS yang berbeda.
- *Topic Route*: Komponen ini merupakan sebuah pemetaan antara subskripsi dan publikasi yang sedang berlangsung.

Pada DDS-IS setidaknya ada dua langkah utama yang dilakukan untuk menangani komunikasi antara dua DDS yang berada di dua zona yang berbeda. Yang pertama adalah *event discovery*. Seperti yang tampak pada Gambar 2.6., ketika *middleware* DDS menemukan adanya *event* baru yang muncul, maka DDS tersebut akan mengirimkan kepada *Discovery Monitor*. *Discovery Monitor* selanjutnya akan mengambil informasi dari DDS mengenai *event* yang baru saja muncul, kemudian meneruskannya ke *Domain Route*. *Domain Route* kemudian akan memeriksa apakah *event* yang baru muncul ini memiliki topik yang sama dengan *event-event* yang sudah ada sebelumnya atau tidak. Jika didapati topiknya sama, maka akan langsung meneruskannya ke *Data Engine*. Tetapi jika *event* memiliki topik yang baru, maka *Domain Route* akan menginstansiasi sebuah *Topic Route* yang baru.



**Gambar 2. 5. Komponen Lapisan *Routing* pada DDS-IS(Jose, et al, 2012)**

Sedangkan langkah utama kedua yang dilakukan adalah melakukan fungsionalitas untuk menjembatani antar *event* dari *subscriber* dengan *publisher* yang terletak di dua zona yang berbeda. DDS akan menghubungi *Data Engine*. Kemudian *Data Engine* dari zona tersebut menghubungi *Data Engine* di zona kedua, agar dihubungkan dengan *Topic Route* yang sesuai dengan *event* baru yang muncul. Jika ternyata *publisher* untuk *event* yang diminta dari zona pertama ditemukan, maka data itu akan di publikasikan dan dialirkan kepada zona pertama melalui *Data Engine* tersebut. Sayangnya pada penelitian ini tidak membahas tentang bagaimana mekanisme untuk mengurangi kerja *broker* agar *broker* tidak mengalami *overload*.



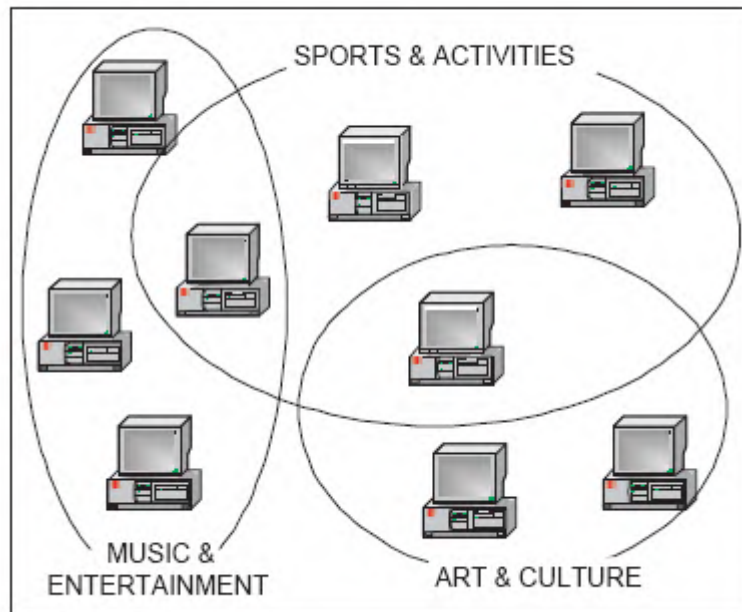
**Gambar 2. 6. Diagram Sekuens pada Proses *Discovery Event***

#### 2.2.4. Semantic Overlay Network(SON)

SON merupakan sebuah pendekatan yang dilakukan untuk mengelompokkan node-node atau *peers* yang terdapat dalam satu jaringan secara tematik (Doulkeridis, et al, 2010). Yakni berdasarkan kesamaan topik tertentu dari data yang ada pada node-node tersebut. Contoh sederhana dari SON dapat dilihat pada Gambar 2.7. Pada gambar tersebut, tiap node dikumpulkan berdasarkan kesamaan dari konten mereka. Yakni *Sport & Activities*, *Music & Entertainment*, *Art & Culture*. Yang perlu digaris bawah adalah bisa jadi satu node dalam sebuah SON juga menyimpan data untuk SON yang lain. Misalnya, sebuah node dapat masuk dalam dua kategori, *Sport & Activities* dan *Music & Entertainment*. Contoh yang lain adalah pada studi kasus pengelompokkan data bencana alam. Sebagai contoh yang ditunjukkan pada Gambar 2.8. Terdapat 4 SON, yakni gempa bumi, tsunami, kebakaran hutan, dan gunung meletus. Memungkinkan adanya node yang menjadi anggota tiap SON tersebut menyimpan data



dari SON yang lain. Misalkan pada SON gunung meletus, terdapat node memiliki data temperatur. Yang data ini juga dibutuhkan pada SON kebakaran hutan. Atau misalkan pada SON gempa bumi, terdapat node yang menyimpan data besaran getaran yang dibutuhkan oleh node yang berada di SON tsunami.

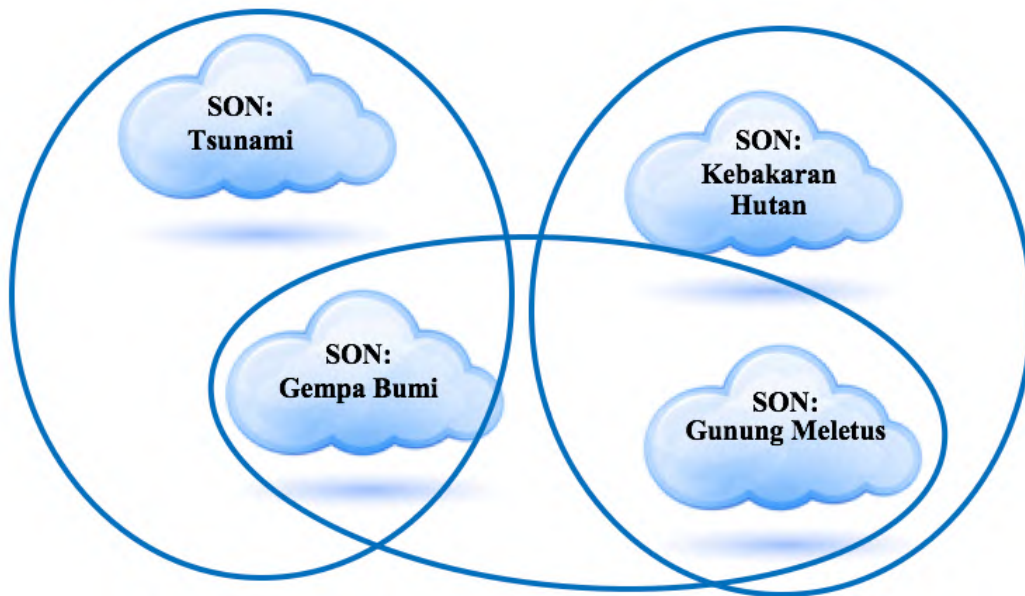


**Gambar 2. 7. SON pada Jaringan P2P**

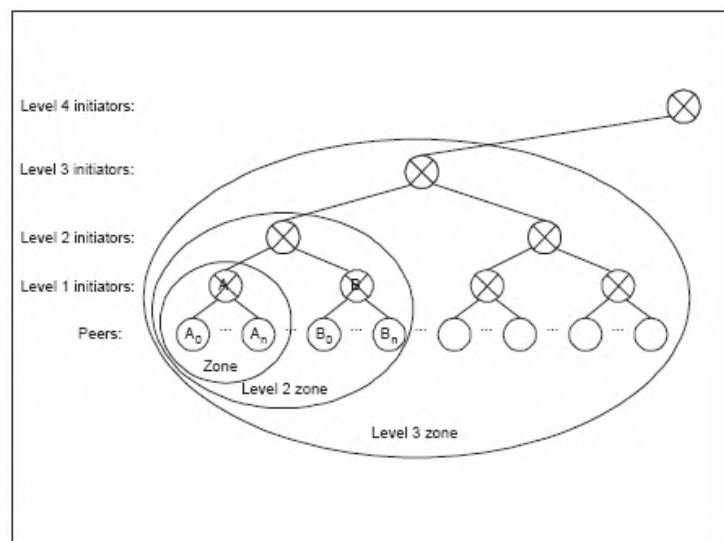
Konsep SON ini cocok apabila diterapkan pada mekanisme *publish/subscribe*. Keuntungan dari penggunaan SON ini adalah dapat mempercepat proses *matching event* karena node-node yang memiliki kesamaan topik dikelompokkan dalam sebuah grup sendiri.

Pada mekanisme *publish/subscribe* keberadaan *Semantic Overlay Network* dapat digunakan sebagai cara untuk menentukan node-node yang dapat membantu proses diseminasi *event* dari *subscriber* untuk mengurangi beban pada *main broker*. Beban *main broker* akan dapat berkurang jika ada node yang dapat membantunya untuk mengelola subskripsi dan publikasi. Pemilihan node(*Child broker(s)*) bisa dilakukan dengan menelusuri *tree* yang dibentuk oleh SON seperti yang terlihat pada Gambar 2.9.





**Gambar 2. 8. SON pada Studi Kasus Bencana Alam**



**Gambar 2. 9. Tree yang dibentuk oleh SON(Vazirgiannis, et al, 2005)**

## **BAB III**

### **METODOLOGI PENELITIAN**

Pada penelitian ini disusun sebuah metode untuk memilih *child broker* sebagai *node* untuk membantu *main broker* dalam melayani *subscribe* maupun *publish data* pada mekanisme komunikasi *publish/subscribe*. Untuk memfokuskan tujuan penelitian ini, maka disusun beberapa langkah penelitian sebagai berikut:

1. Perumusan Masalah
2. Studi Literatur
3. Pemetaan Skenario dan Konsep Kerja
4. Desain Framework dan Implementasi
5. Pengujian dan Evaluasi
6. Penyusunan Buku Tesis

#### **3.1. Perumusan Masalah**

Masalah yang dihadapi dalam penelitian ini adalah pemilihan *child broker* sebagai pembantu *main broker* pada mekanisme *publish/subscribe*. Terdapat kemungkinan apabila *main broker* mengalami *overload*, maka akan terjadi *single node failure*. Dalam penelitian ini, penulis memiliki pendapat bahwa untuk mengurangi kemungkinan terjadinya *single node failure*, maka dibutuhkan pemilihan node lain sebagai asisten dari *main broker* tersebut. Semakin banyak pembantu *main broker* dalam melayani proses *subscribe* dan *publish* pada mekanisme komunikasi *publish/subscribe*, maka akan lebih cepat pesan tersampaikan, mengurangi *delay* dan *throughput*.

Maka, untuk mengukur hipotesa penelitian ini, dilakukan beberapa hal yaitu persiapan lingkungan *publish subscribe*, persiapan data sampel untuk uji coba menggunakan data sample bencana alam, pengukuran sumber data broker dan node yang berada dalam sistem pada fase *training*, dan pemilihan *child broker* sebagai asisten dari *main broker*.

#### **3.2. Studi Literatur**

Pada tahap studi literatur, dilakukan pencarian referensi pustaka dan pembelajaran yang menjadi dasar pijakan bagi penelitian dan dapat dipertanggungjawabkan. Daftar referensi tersebut dapat diperoleh dari berbagai sumber

seperti buku, jurnal atau artikel ilmiah. Topik yang diambil dalam studi literatur adalah topik yang terkait dengan penelitian yang diajukan. Diantaranya mekanisme *publish/subscribe*, konsep dasar *Data Distribution Service*, *Semanti Overlay Network*, *Node Criteria* pada jaringan.

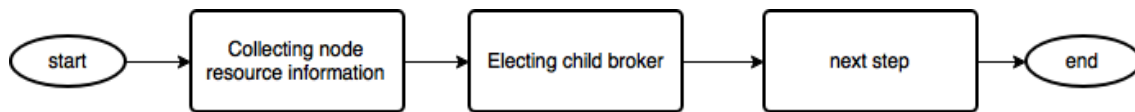
### 3.3. Pemetaan Skenario dan Konsep Kerja

Pada penelitian ini dirancang tahapan-tahapan untuk pemetaan skenario dan konsep kerja sebagai bentuk solusi terhadap permasalahan yang diangkat. Diharapkan dengan melakukan analisa permasalahan, penyusunan konsep kerja dan pemetaan skenario mampu mengurangi permasalahan yang luput pada tahap desain solusi dan implementasi.

Pada titik permasalahan yang diangkat pada penelitian ini, sebuah *broker* pada mekanisme komunikasi *publish subscribe* dapat mengalami *single node failure* jika mengalami beban berlebih dalam jaringan. Hal ini mengakibatkan tidak tersampainya data dari *publisher* ke *subscriber* karena *broker* tidak mampu menangani *request* dan menyampaikan pesan *publishing*. Misalnya, *broker* harus bertugas untuk menampung *event* yang dimunculkan oleh *subscriber* maupun *publisher*. Di satu sisi, *broker* juga harus melakukan pencarian *publisher* yang memenuhi kriteria subskripsi. Di sisi lain, *broker* ini juga yang bertugas untuk melakukan *forwarding event* dari satu zona ke zona lain. Hal ini akan memberikan beban berat bagi broker, dan memungkinkan terjadinya *overload* pada *broker* yang bisa mengantarkan pada *single-node-failure*. Jika *broker* ini mengalami *fail*, maka otomatis kegagalan pengiriman dan penerimaan data akan terjadi pada suatu zona DDS.

Dalam penelitian ini mengusulkan adanya mekanisme pemilihan asisten *broker* yang bisa membantunya dalam mengelola subskripsi dan publikasi. Dengan demikian, maka yang diharapkan adalah beban kinerja *broker* berkurang, trafik jaringan antar zona juga berkurang. Ditambah lagi hal ini akan meningkatkan *throughput* serta mengurangi latensi karena sistem semakin *reliable*.

Pemilihan asisten *broker/ child broker* meliputi pengumpulan informasi sumber daya dari *node* yang tergabung dalam jaringan dan keputusan memilih *child broker* sebagai asisten dari *main broker* seperti yang ditunjukkan pada Gambar 3.1.



**Gambar 3. 1.Konsep Kontribusi**

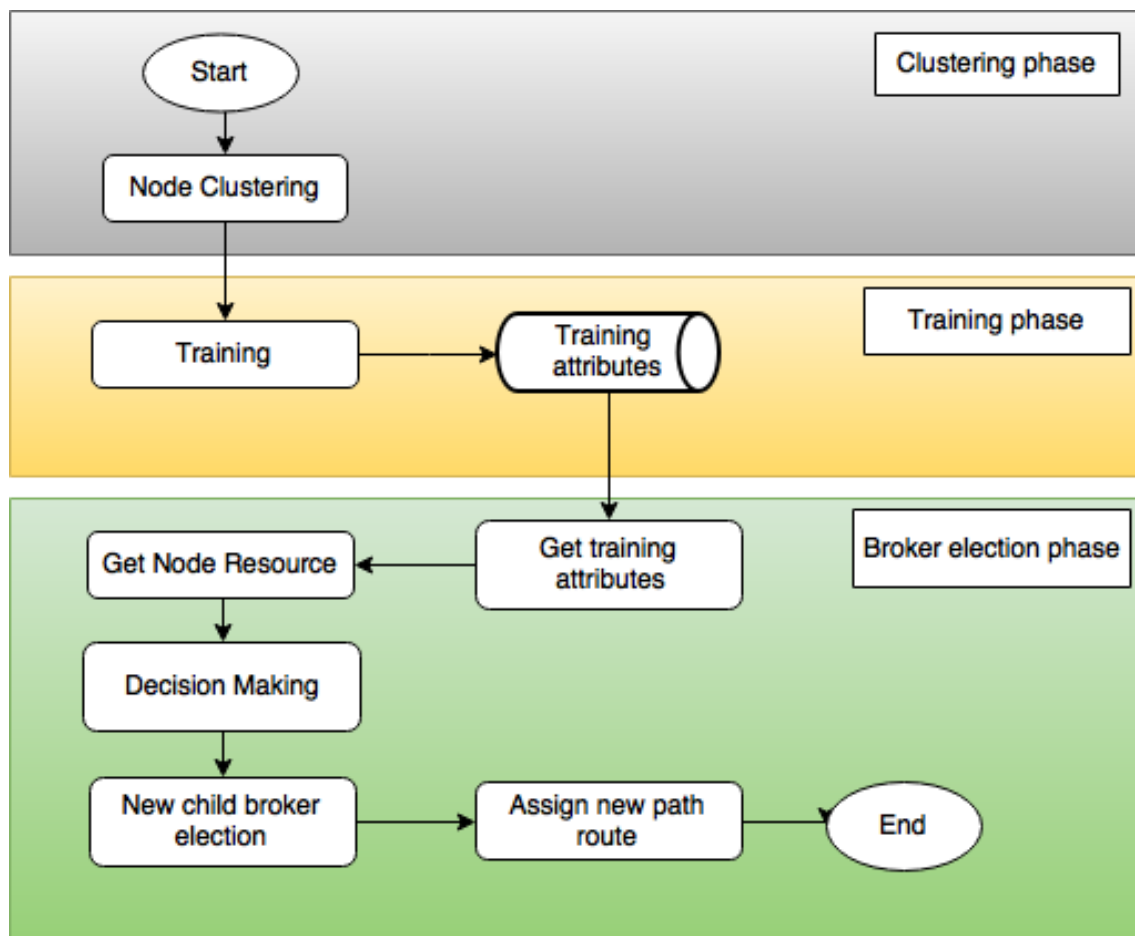
Untuk memilih sebuah *child broker* sebagai asisten dari *main broker*, dilakukan pengukuran sumber daya. Adapun parameter yang digunakan dalam proses pengukuran sebagai dasar yang digunakan untuk memutuskan node mana yang akan menjadi *child broker* adalah sebagai berikut:

- a. Penggunaan RAM
- b. Response Time
- c. Kesibukan CPU
- d. Kecepatan pengiriman data

Hasil dari pengukuran sumber daya dengan menggunakan keempat parameter di atas adalah terpilihnya sebuah node sebagai *child broker* yang memiliki waktu terkecil. Sehingga *main broker* dapat memindahkan beberapa bebannya ke node yang terpilih.

Adapun langkah-langkah yang diajukan dalam penelitian ini untuk mengoptimalkan proses *publish/subscribe* dalam suatu sistem jaringan dapat dilihat pada Gambar 3.2. dengan penjelasan sebagai berikut:

1. *Node Clustering* berdasarkan kedekatan topik
2. *Child Broker Election* berdasarkan 4 kriteria, yakni penggunaan RAM, *response time*, kesibukan CPU dan kecepatan pengiriman data.
3. *Assign Path Route* untuk sebuah *subscribe topic*



**Gambar 3. 2. Tahap Pemilihan *Child Broker***

### 3.3.1. Data Sampel Uji Coba

Dalam penelitian ini, dibutuhkan data sampel uji coba untuk 2 tujuan. Yang pertama adalah sebagai data training untuk membangkitkan formula untuk penghitungan standar kompetensi broker untuk memilih node lain sebagai *child broker*. Yang kedua adalah sebagai data pengujian. Data sample uji coba yang digunakan adalah data bencana alam yang meliputi *field latitude*, *longitude*, *disaster\_type*, *disaster\_subtype*, *place*, *total\_death*, dan *damage*. Sampel data ini diambil dari situs CRED (Centre for Research on the Epidemiology of Disasters).

Data tersebut disimpan dalam *database* dan akan di *publish* oleh *publisher* dengan format JSON. Data dalam bentuk format JSON yang dapat dilihat pada Gambar 3.3. ini yang digunakan sebagai data training dan data pengujian pada lingkungan uji coba yang telah disediakan.

<pre> {"item": [{"latitude":"-2.396", "longitude":"68.1679", "type":"earthquake", "disaster_subtype":"ground_movement", "damage":"230000 \$US", "total_death":"17001"},  {"latitude":"-5.1838", "longitude":"140.9062", "type":"earthquake", "disaster_subtype":"ground_movement", "damage":"17000 US\$", "total_death":"20011"},  {"latitude":"-13.3865", "longitude":"-69.8724", "type":"earthquake", "disaster_subtype":"tsunami", "damage":"670000 \$US", "total_death":"32091"},  {"latitude":"-29.6625", "longitude":"-72.1722", "type":"earthquake", "disaster_subtype":"tsunami", "damage":"76300 \$US", "total_death":"12989"},  {"latitude":"-13.2829", " longitude":"-70.7397", </pre>
---

<pre> "type":"flood", "disaster_subtype":"coastal_flood", "damage":"42750 \$US", "total_death":"169"},  {"latitude":"-30.6613", "longitude":"-71.6762", "type":"flood", "disaster_subtype":"coastal_flood", "damage":"121272 \$US", "total_death":"210"},  {"latitude":"4.5734", "longitude":"124.9909", "type":"flood", "disaster_subtype":"flash_flood", "damage":"521086 \$US", "total_death":"3143"},  {"latitude":"49.3752", "longitude":"155.2986", "type":"flood", "disaster_subtype":"flash_flood", "damage":"78850 \$US", "total_death":"3270"},  {"latitude":"-41.6766", "longitude":"-74.0854", "type":"flood", "disaster_subtype":"riverine_flood", "damage":"386347 \$US", </pre>
--

	<pre> "total_death":"120"},  {"latitude":"0.8544", "longitude":"122.4858", "type":"flood", "disaster_subtype":"riverine_flood", "damage":"868230 \$US", "total_death":"378"},  {"latitude":"20.0507", "longitude":"145.5402", "type":"landslide", "disaster_subtype":"rockfall", "damage":"38728 \$US", "total_death":"102"},  {"latitude":"-6.7995", "longitude":"-76.8565", "type":"landslide", "disaster_subtype":"rockfall", "damage":"50000 \$US", "total_death":"1209"},  {"latitude":"-1.0045", "longitude":"-15.9845", "type":"landslide", "disaster_subtype":"avalanche", "damage":"50000 \$US ", "total_death":"3142"}]]} </pre>
--	--

**Gambar 3. 3. Contoh Data Sampel Uji Coba**



### 3.3.1 Fase *Node Clustering*

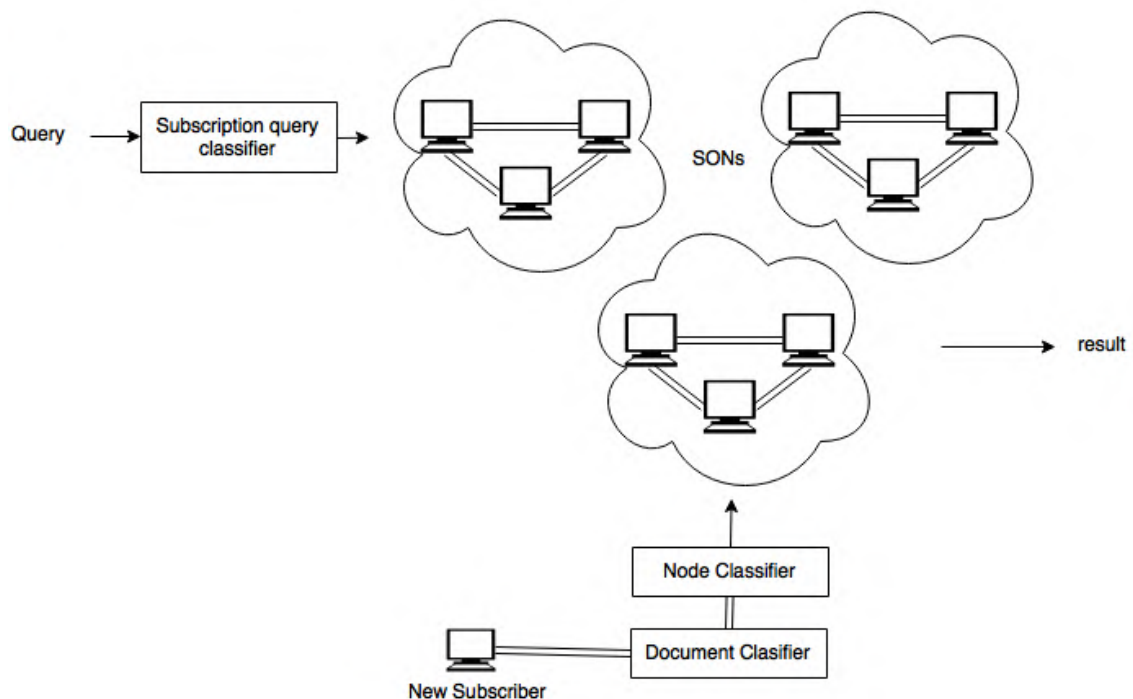
Fase pertama yang dilakukan adalah *clustering node* dari node-node yang tergabung pada sistem jaringan *publish/subscribe*. Node-node tersebut dikelompokkan berdasarkan kedekatan topik *subscribe* yang disampaikan ke *broker*. Pengelompokkan node dilakukan dengan membentuk sebuah *Semantic Overlay Network (SON)* pada kelompok *subscriber*. Pada sistem SON dilakukan sebuah pendekatan untuk mengelompokkan node-node atau *peers* yang terdapat dalam satu jaringan secara tematik (Doulkeridis, et al, 2010).

Pada penelitian ini dilakukan pengelompokkan node berdasarkan kedekatan topik *subscription*. Data yang akan digunakan sebagai data training pengelompokkan terdapat pada Gambar 3.5. Terdapat beberapa topik yang digunakan dalam data sampel uji coba, *earthquake, flood, landslide* dan *storm*. Pada tiap data sampel terdapat *field* data yang digunakan untuk melakukan klasifikasi kedekatan node berdasarkan topik *subscription*. Field tersebut adalah *type, disaster\_subtype, latitude, longitude, damage, dan total\_death*.

Pada SON tidak fokus pada *layer* fisik koneksi antar node dalam jaringan. Tetapi SON akan membentuk sebuah *layer* bayangan yang merepresentasikan hubungan antar node berdasarkan kedekatan topik mereka. Tiap node pada SON terhubung dengan node tetangga yang dinotasikan dengan  $(n_i, n_j, l)$ , dimana  $l$  merupakan flag penanda hubungan antara node  $n_i$  dan  $n_j$ . Terdapat 3 fungsi utama pada SON, yakni:

1.  $Join(n_i, l)$  dimana satu atau lebih node  $(n_i, n_j, l)$  bergabung dalam sebuah overlay network  $(ON_l)$ .
2.  $Search(r, l)$  yang akan mengembalikan satu atau beberapa informasi node pada  $ON_l$  sesuai dengan *request*  $r$  yang diberikan.
3.  $Leave(n_i, l)$  yang akan menghapus semua link pada  $ON_l$  yang melibatkan node  $n_i$ .

Tujuan dari penggunaan SON adalah untuk mengelompokkan node-node pada sisi subscriber agar dapat memudahkan *broker* untuk memilih *child broker* yang akan diseleksi berdasarkan kedekatan topik subskripsi baru yang masuk ke dalam sistem *publish subscribe*. Node pada kelompok *subscriber* ini akan bergabung pada SON yang sesuai berdasarkan klasifikasi data yang dimiliki oleh node tersebut. Pada Gambar 3.4. menunjukkan proses pembentukan SON.



**Gambar 3. 4. Proses Pembentukan SON**

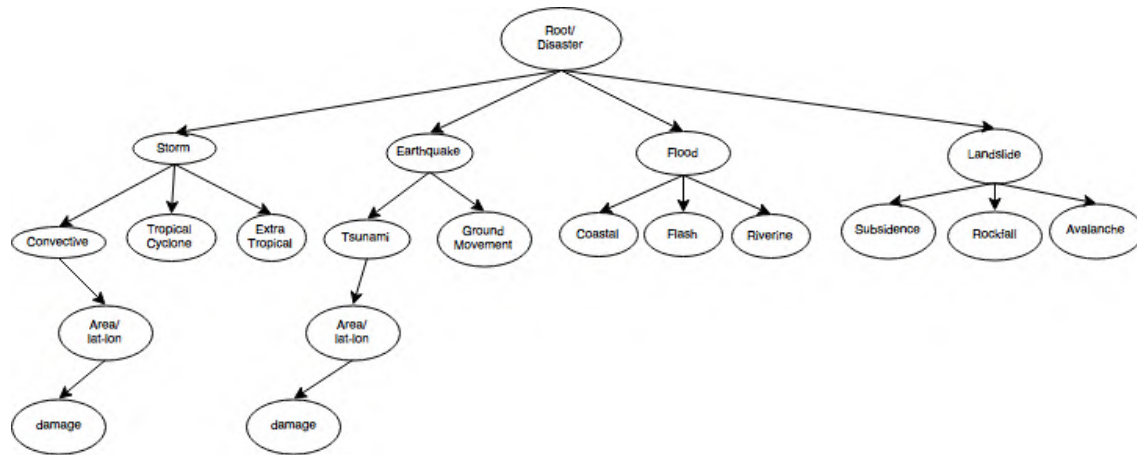
Pada Gambar 3.4. tersebut terlihat bahwa apabila sebuah node subscriber akan bergabung dengan SONs terkait, harus melewati proses klasifikasi dokumen yang merupakan proses klasifikasi data yang dimiliki oleh node tersebut. Untuk proses klasifikasi digunakan 4 kata kunci sebagai batasan. Dimana kata kunci tersebut diambil dari field data yang didefinisikan pada Gambar 3.3. Node tersebut akan menghubungi seluruh node hingga menemukan SONs yang sesuai. Informasi node tersebut akan disimpan pada node *classifier*. Gambar 3.5. menunjukkan hierarki klasifikasi SON pada penelitian ini sesuai dengan sampel data pada Gambar 3.3.

### 3.3.2. Fase Training

Fase *training* merupakan fase dimana akan dilakukan beberapa percobaan untuk mendapatkan sekumpulan data inisial atau data awal. Data inisial atau data awal ini yang akan digunakan untuk membentuk sebuah formula yang berfungsi sebagai formula penghitungan *threshold*/batas dimana sebuah *broker* membutuhkan asisten *broker* dan menentukan node mana yang dapat terpilih sebagai *child broker*. Fase ini dilakukan setelah terbentuknya cluster node sesuai dengan kedekatan tema atau konten subskripsi.

Pada fase ini digunakan 3 buah CPU, dengan nilai atribut seperti yang ditampilkan pada Tabel 3.1. yakni data, waktu pengiriman data dalam jaringan,

penggunaan RAM dan CPU. Dari data pada Tabel 3.1 dilakukan proses analisis regresi berganda untuk mendapatkan hubungan antara variabel-variabel tersebut dengan variabel dependen(Waktu).



**Gambar 3. 5. Klasifikasi Hierarki**

Hasil dari proses regresi dapat dilihat pada Tabel 3.2. Dari proses tersebut didapatkan standar error yang tinggi. Hal ini menunjukkan bahwa akurasi data kurang baik apabila dijadikan sebagai standar. Ditambah lagi pada Gambar 3.6. adalah grafik yang menunjukkan bahwa data-data tersebut memiliki hubungan eksponensial dan belum terdistribusi secara normal.

**Tabel 3. 1. Tabel Data Training**

No	CPU	GFlop	RAM Speed	CPU Util(%)	Link(Mbps)	Data(Mb)	Waktu(s)
1	CPU A	1.38E-05	7.50E-06	27.7	10	5	0.52
2		1.38E-05	7.50E-06	33.8	10	10	1.1
3		1.38E-05	7.50E-06	34	10	20	2.2
4		1.38E-05	7.50E-06	34.7	10	50	5
5		1.38E-05	7.50E-06	35.2	10	100	11.3
6		1.38E-05	7.50E-06	32.2	20	5	0.27
7		1.38E-05	7.50E-06	32.5	20	10	0.55
8		1.38E-05	7.50E-06	33	20	20	1.1
9		1.38E-05	7.50E-06	33.2	20	50	2.6
10		1.38E-05	7.50E-06	33.9	20	100	5.2

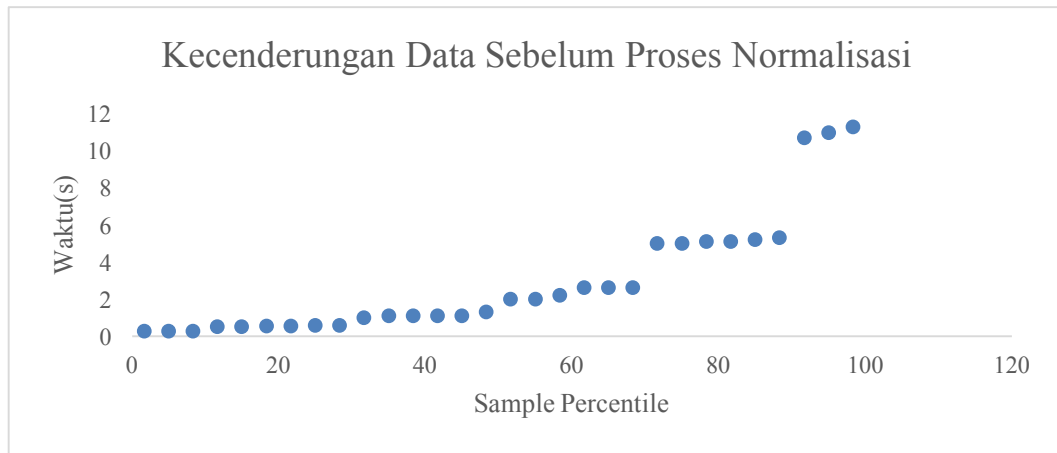
11	CPU B	1.18E-05	7.60E-06	29	10	5	0.5
12		1.18E-05	7.60E-06	31.1	10	10	1.3
13		1.18E-05	7.60E-06	31.7	10	20	2
14		1.18E-05	7.60E-06	32.2	10	50	5.1
15		1.18E-05	7.60E-06	32.3	10	100	11
16		1.18E-05	7.60E-06	30.3	20	5	0.25
17		1.18E-05	7.60E-06	30.1	20	10	0.57
18		1.18E-05	7.60E-06	31	20	20	1.1
19		1.18E-05	7.60E-06	31.2	20	50	2.6
20		1.18E-05	7.60E-06	31.7	20	100	5.1
21	CPU C	1.34E-05	7.51E-06	28.9	10	5	0.51
22		1.34E-05	7.51E-06	30.3	10	10	1
23		1.34E-05	7.51E-06	30.9	10	20	2
24		1.34E-05	7.51E-06	31.4	10	50	5
25		1.34E-05	7.51E-06	31.7	10	100	10.7
26		1.34E-05	7.51E-06	29.7	20	5	0.27
27		1.34E-05	7.51E-06	30.5	20	10	0.56
28		1.34E-05	7.51E-06	30.7	20	20	1.1
29		1.34E-05	7.51E-06	30.3	20	50	2.6
30		1.34E-05	7.51E-06	31.9	20	100	5.3

**Tabel 3. 2 Hasil Regresi Data Sebelum Proses Normalisasi**

<i>Regression Statistics</i>	
Multiple R	0.948839331
R Square	0.900296076
Adjusted R Square	0.879524425
Standard Error	1.120879604
Observations	30

Untuk mendapatkan standar eror yang kecil dan meningkatkan akurasi, maka dibutuhkan proses normalisasi data agar mentransformasikan bentuk eksponensial

menjadi linear. Untuk menjadikan data normal digunakan fungsi logaritma untuk mentransformasi data tersebut.



**Gambar 3. 6. Kecenderungan DataAtribut terhadap Waktu(s) sebelum Normalisasi**

Setelah proses normalisasi data, dilakukan kembali proses regresi sehingga menghasilkan standar error yang rendah seperti terlihat pada Tabel 3.3. dan sebaran data yang normal atau linear seperti yang terlihat pada Gambar 3.7.

**Tabel 3. 3. Hasil Regresi Setelah Normalisasi**

<i>Regression Statistics</i>	
Multiple R	0.998898128
R Square	0.997797471
Adjusted R Square	0.99733861
Standard Error	0.025639265
Observations	30

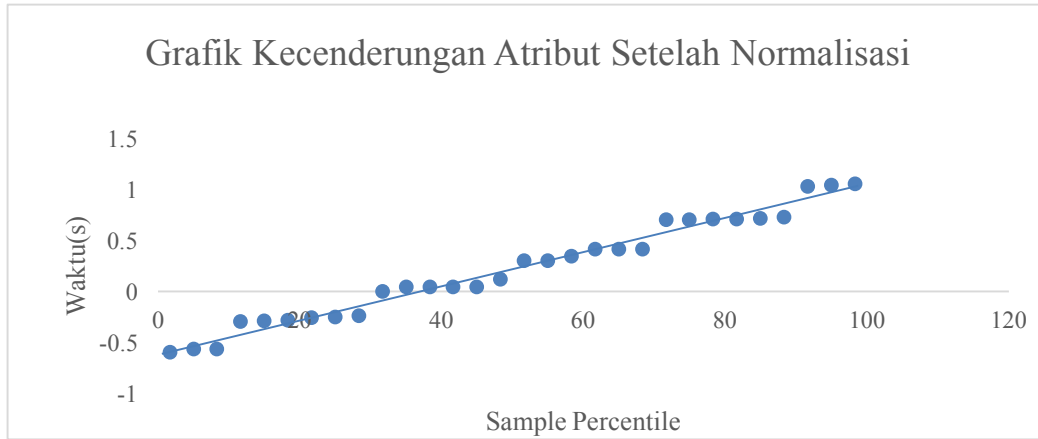
Dari Tabel 3.3. didapatkan bahwa Multiple R tinggi yakni sebesar 0.99889 yang menunjukkan bahwa atribut tersebut berpengaruh kuat terhadap variabel waktu(konvergen).

Dari Tabel 3.4. yang merupakan hasil proses regresi didapatkan nilai koefisien dari masing-masing atribut.

- GFlop = -0.383238623
- RAM Speed = -0.383238623
- CPU Util = 0.440351744

d. Link = -0.978304621

e. Data = 0.982207119



**Gambar 3. 7. Grafik Kecenderungan Data yang telah Ternormalisasi**

Sehingga didapatkan sebuah formula untuk menghitung besaran waktu yang dibutuhkan yang lengkapnya ditunjukkan pada Persamaan 1.

$$Waktu = 0.982207119 * Data - 0.383238623 * GFlop - 0.383238623 * RAMSpeed - 0.978304621 * Link + 0.440351744 * CPUUtil - 16.48429187 \text{ (Persamaan 1)}$$

**Tabel 3. 4. Koefisien Regresi**

	<i>Coefficients</i>
Intercept	-16.48429187
GFlop	-0.383238623
RAM Speed	-2.728405832
CPU Util	0.440351744
Link	-0.978304621
Data	0.982207119

### 3.3.3. ManageEngine OpManagers

ManageEngine OpManagers merupakan aplikasi yang digunakan untuk melakukan pengawasan terhadap sumber daya dari node-node yang tergabung dalam sebuah jaringan. Aplikasi ini dapat digunakan untuk mengetahui kesibukan CPU, penggunaan RAM, trafik, *bandwith monitoring* dan beberapa parameter lain. Pada

Gambar 3.8. terlihat contoh *monitoring dashboard* dari node-node yang berada dalam jaringan.

Pada aplikasi ManageEngine OpManagers dapat ditentukan *threshold*/batasan nilai untuk mengindikasikan nilai krisis dari sumber daya yang dipilih. Misalkan penetapan nilai *threshold* untuk CPU dan Memori seperti yang terlihat pada Gambar 3.9. Nilai *threshold* yang diberikan didapatkan dari penghitungan data *training* yang dilakukan di fase *training*.



**Gambar 3. 8. Monitoring Dashboard ManageEngine OpManager**

### 3.4. Implementasi

Penelitian sebelumnya di DDS-IS(Jose, et al, 2012) telah dibangun sebuah layanan yang mampu menjembatani komunikasi beberapa aplikasi DDS di dua zona yang berbeda. Pada DDS-IS terdapat *event discovery* untuk menangkap *subscriptions event* dari *subscribers*. Dari *event discovery* akan dilanjutkan ke lapisan routing, dimana pada lapisan tersebut terdapat *Data Engine/Broker* yang bertugas untuk menemukan dimana *publisher* yang memiliki data tersebut. Sayangnya, pada DDS-IS belum terdapat mekanisme untuk mengatasi *single-node-failure*, yakni permasalahan dimana *data engine/broker* mengalami *overload*.

**DiskUtilization**

Monitor Name: Disk Utilization      SNMP OID: .13.6.1.2.1.25.2.3.1.6

Interval (Minutes): 60      Units: Percentage

Store Data: ☒ Yes, I need graphs    ☐ No, just alert

Threshold details

Consecutive times: 1

	Condition	Threshold Value	Message
Attention	>		\$MONITOR is \$CURREI
Trouble	>		\$MONITOR is \$CURREI
Critical	>		\$MONITOR is \$CURREI
Rarm	<=		\$MONITOR is now bac

Cancel    Save

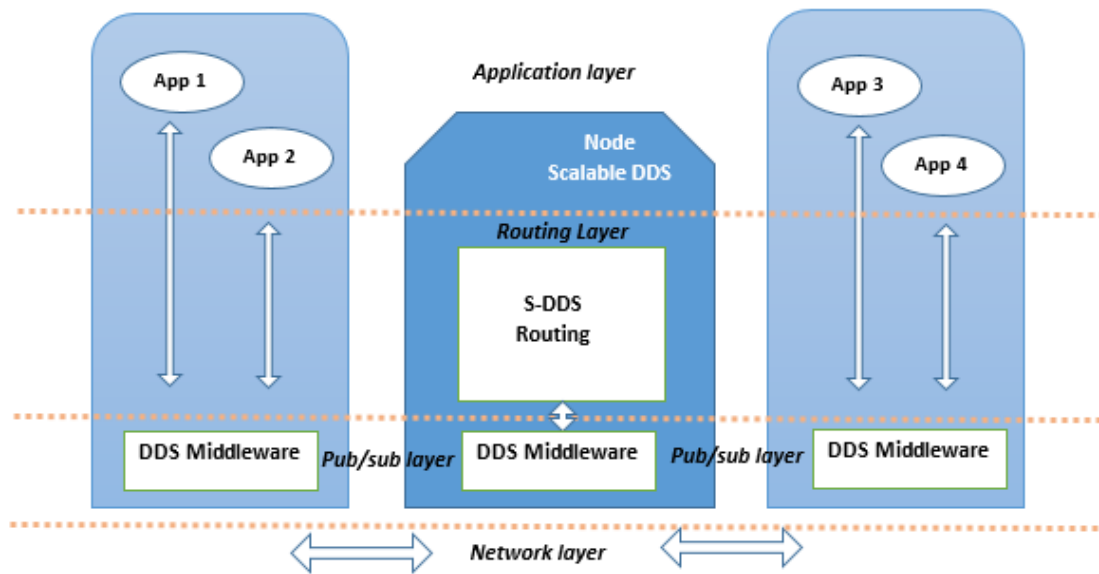
**Gambar 3. 9. Pengaturan *threshold* pada ManageEngine OpManager**

Sehingga pada implementasi penelitian ini ditambahkanlah satu langkah yakni *broker election* untuk menentukan node mana yang akan menjadi asisten *broker/child broker* ketika suatu *broker* mengalami *overload*. Adapun rancangan arsitektur sistem yang diusulkan dapat dilihat pada Gambar 3.10.

#### **3.4.1. Pemroraman Node.js**

Node.js merupakan sebuah *platform* yang digunakan untuk aplikasi yang berjalan secara *real time*. Node.js diimplementasikan di atas engine V8 javaScript yang selalu mengikuti perkembangan ECMAScript sehingga mampu berjalan di semua *browser*.





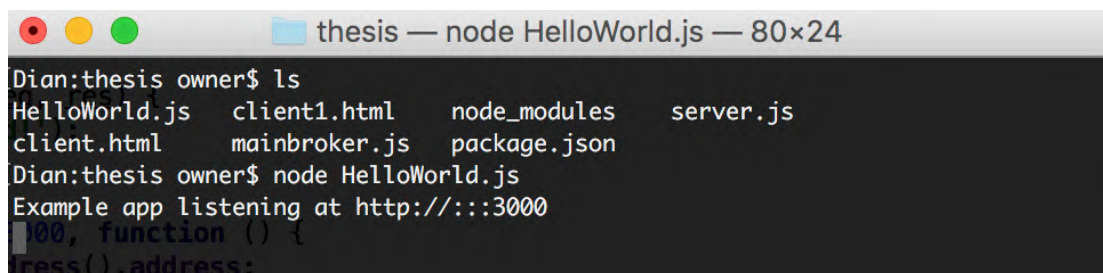
**Gambar 3. 10. Arsitektur Rancangan Sistem *Scalable DDS***

Penggunaan Node.js dalam implementasi penelitian ini adalah berdasarkan kemampuannya dalam menangani sebuah proses *event-driven* yang mendukung aplikasi *real-time*. Kelebihan penggunaan Node.js adalah penerapan teknik *Non-blocking* dimana Node.js akan mengeksekusi proses secara independen. Artinya Node.js akan mengeksekusi sebuah operasi tanpa menunggu operasi sebelumnya selesai terlebih dahulu. Hal ini tentu membuat aplikasi berjalan lebih cepat dan efisien karena mampu menghemat memori dan mengoptimalkan penggunaan prosesor. Ditambah lagi database NoSQL seperti MongoDB yang digunakan pula dalam implementasi dan Couch DB mendukung langsung Bahasa pemrograman JavaScript sehingga *interfacing* dengan *database* dilakukan lebih mudah.

Pada Gambar 3.11 merupakan contoh pemrograman sederhana menggunakan Node.js yang dapat diimplementasikan melalui IDE WebStorm. Jika program tersebut dijalankan melalui *console* maka akan menghasilkan keluaran seperti pada Gambar 3.12.



**Gambar 3. 11. Contoh Pemrograman Node.js “Hello World!”**



**Gambar 3. 12. Hasil *Compile* Program “Hello World!”**

Pada pemrograman Node.js mengenal adanya Node Package Manager(NPM). NPM ini digunakan untuk kebutuhan instalasi modul-modul terkait dengan system yang akan dibangun. Modul-modul tersebut ter-embed di dalam package Node.js tetapi harus diinstal terlebih dahulu agar dapat dimanfaatkan. Pada implementasi penelitian ini, digunakan beberapa modul Node.js diantaranya adalah sebagai berikut:

- a. Modul Express
- b. Modul MySQL
- c. Modul MongoDB
- d. Modul Socket.io
- e. Modul Async
- f. Modul Mosca

#### **3.4.2. Mosca Middleware**

Mosca merupakan salah satu modul dalam Node.js yang merupakan modul *broker*. Komunikasi *publish/subscribe* yang melibatkan *broker* sebagai pengatur *event* *publish* dan *subscribe* dibangun di atas modul Mosca ini. Mosca dibangun di atas protocol MQTT yang merupakan salah satu protocol handal dalam komunika-

*publish/subscribe*. Meskipun Mosca merupakan MQTT *broker* tetapi dia juga mendukung *broker* lain seperti RabbitMQ dengan protocol AMQP, Zero MQ, juga mendukung MongoDB, Redis dan Mosquitto.

Pada Gambar 3.13 dapat dilihat contoh sederhana program Mosca dengan menggunakan Mongo. Untuk dapat menggunakan Mongo, terlebih dahulu harus *install* modul tersebut pada sistem.

### 3.5 Lingkungan Pengujian

Sistem *publish-subscribe* yang dibangun ini memiliki 3 komponen utama, yakni *publisher* yang berlaku sebagai sumber data, *subscriber* yang berlaku sebagai pihak yang membutuhkan data terkait dan *broker* yang dibangun dengan *middleware* Mosca. Data yang digunakan adalah data sample bencana alam dengan jenis *Earthquake*, *Flood*, *Storm* dan *Landslide*. Pada penelitian ini menggunakan Mosca sebagai *middleware* untuk membangun sistem *publish subscribe* dan komunikasi antar *broker*.



```
1  /**
2   * Created by owner on 24/2/16.
3   */
4
5   var mosca = require('mosca')
6
7   var ascoltatore = {
8     //using ascoltatore
9     type: 'mongo',
10    url: 'mongodb://localhost:27017/mqtt',
11    pubsubCollection: 'ascoltatori',
12    mongo: {}
13  };
14
15  var moscaSettings = {
16    port: 1883,
17    backend: ascoltatore,
18    persistence: {
19      factory: mosca.persistence.Mongo,
20      url: 'mongodb://localhost:27017/mqtt'
21    }
22  };
23
24  var server = new mosca.Server(moscaSettings);
25  server.on('ready', setup);
26
27  server.on('clientConnected', function(client) {
28    console.log('client connected', client.id);
29  });
30
31  // fired when a message is received
32  server.on('published', function(packet, client) {
33    console.log('Published', packet.payload);
34  });
35
36  // fired when the mqtt server is ready
37  function setup() {
38    console.log('Mosca server is up and running')
39  }
```

Gambar 3. 13. Contoh Program pada Mosca Menggunakan Mongo

### **3.5.1. Skenario Pengujian**

Tujuan dilakukannya skenario pengujian adalah untuk membuktikan kebenaran hipotesa pada rumusan masalah. Yakni penghematan sumber daya *broker* dan mengurangi kemungkinan terjadinya *single node failure*. Uji coba dilakukan pada lingkungan *test bed* yang telah dibangun yang terdiri dari 3 *subnet* dengan 1 *broker* untuk masing-masing *subnet* dan dengan variasi jumlah *subscriber* dan *publisher*.

### **3.5.2 Evaluasi Pengujian**

Pada tahap evaluasi pengujian ditujukan untuk mendapatkan data empiris dari hasil uji coba dengan skenario yang telah ditetapkan dengan parameter uji coba yang meliputi *latency*, *bandwidth*, serta penggunaan CPU dan RAM.

*[Halaman ini sengaja dikosongkan]*

## BAB IV

### HASIL DAN PEMBAHASAN

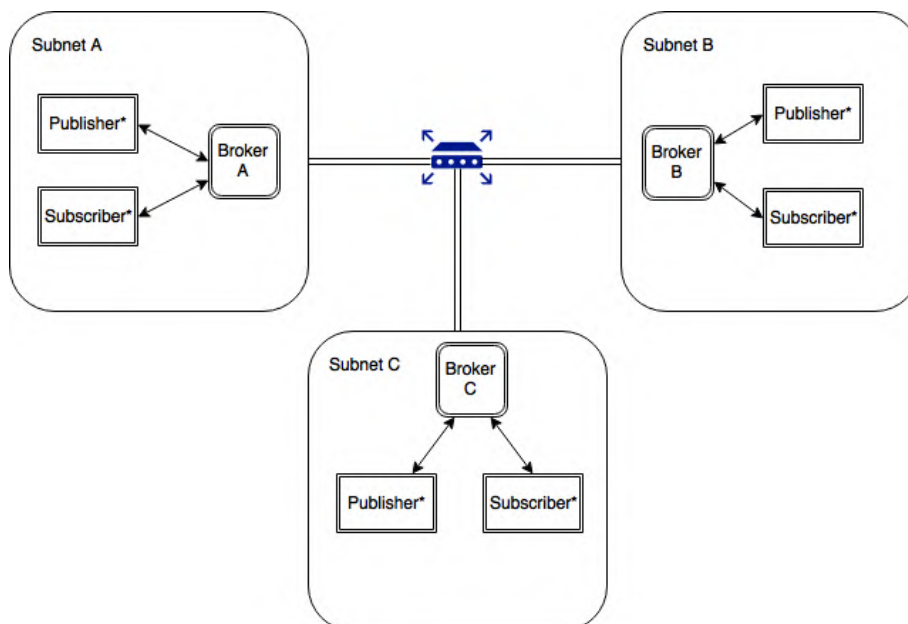
Tahap pengujian dilakukan untuk membuktikan kebenaran hipotesa serta untuk menjawab pertanyaan penelitian yang diuraikan pada Bab II.

#### 4.1. Uji Coba

Tahapan uji coba pada penelitian ini meliputi penentuan skenario pengujian, variable pengujian, lingkungan pengujian, dan perbandingan dengan metode konvensional. Tahap pertama menentukan kasus uji coba yang dilakukan dan parameter yang ditentukan. Tahap kedua adalah menentukan lingkungan pengujian yang membahas topologi, dan spesifikasi dari sumber daya yang digunakan. Tahap terakhir adalah pengujian dan perbandingan dengan metode konvensional.

##### 4.1.1. Lingkungan Uji Coba

Pada penelitian ini, metode yang diusulkan yang merupakan solusi dari permasalahan utama yang diungkap dari penelitian ini dievaluasi pada lingkungan uji coba dengan topologi seperti yang ditunjukkan pada Gambar 4.1.



Pada lingkungan pengujian, terdapat komponen untuk *broker*, *publisher* serta *subscriber*. Disamping itu juga terdapat sebuah *router* yang digunakan untuk

menghubungkan komponen tersebut dalam jaringan. Adapun spesifikasi perangkat keras yang berperan sebagai *broker* memiliki *processor* Intel Core i5 2.6GHz, *memory* RAM 8 GB 1600 MHz DDR3, dengan sistem operasi OSX 10.11.4 (El Capitan). Sedangkan untuk spesifikasi node yang berperan sebagai *publisher/subscriber* memiliki *memory* RAM 1GB dengan sistem operasi Ubuntu 10.04 yang di virtualisasikan dengan Oracle Virtual Box. Keseluruhan node memiliki konfigurasi web server yang sama.

#### 4.1.2. Skenario Pengujian

Skenario pengujian ini dilakukan untuk melihat hasil dari penerapan konsep pemilihan asisten *broker* dan pengaruhnya terhadap parameter pengujian yang meliputi penggunaan CPU *broker*, RAM *broker*, *bandwidth* dan *latency*.

Secara umum, pada pengujian dilakukan dua buah skenario. Skenario pertama yakni tanpa adanya pemilihan asisten/*child broker* dan skenario kedua dengan menggunakan sistem pemilihan *child broker*. Untuk mendapatkan hasil penelitian yang objektif, maka kedua skenario tersebut menggunakan topologi jaringan yang sama yang ditunjukkan pada Gambar 4.1. Pada kedua skenario tersebut dilakukan skema pengujian sebagai berikut:

- a. Perbedaan jumlah *subscriber* dengan jumlah *subscription* yang sama
- b. Perbedaan jumlah *publisher* dengan jumlah topik yang sama
- c. Penambahan topik *subscription*

#### 4.1.3. Parameter Pengujian

Adapun untuk parameter pengujian yang digunakan pada penelitian ini agar dapat dijadikan sebagai referensi pembandingan untuk metode yang diusulkan.

- a. Penggunaan CPU *broker*

Pengujian parameter ini ditujukan untuk melihat penggunaan sumber daya CPU pada *broker* selama terjadi *event publish subscribe*. Pengamatan uji coba ini dipengaruhi oleh jumlah *subscriber*.

- b. Penggunaan RAM *broker*

Pengujian parameter ini ditujukan untuk melihat penggunaan sumber daya Memori RAM selama terjadi *event publish subscribe*.

- c. Penggunaan *Bandwidth*

Pengujian parameter ini ditujukan untuk melihat besarnya *bandwidth* selama proses *event publish subscribe* berlangsung. Akan dibandingkan penggunaan bandwidth antara tanpa metode yang diusulkan dengan metode yang diusulkan.

d. *Latency*

Parameter pengujian ini digunakan untuk melihat lama waktu yang dibutuhkan agar data dari *publisher* sampai ke *subscriber*.

#### **4.2. Hasil Pengujian dan Analisis**

Tujuan dari pengujian adalah untuk mendapatkan data empiris hasil implementasi pemilihan *child broker* yang mampu menurunkan penggunaan CPU dan RAM pada *broker*. Pemilihan node sebagai *child broker* didasarkan pada informasi yang telah terbentuk pada fase *clustering* dan *training*. Data hasil pengujian didapatkan dari pengamatan terhadap metode yang diusulkan dengan mekanisme pemilihan *child broker* yang memiliki sumber daya yang terbaik yang didapat dari formula regresi pada fase *training* dan dengan metode pemilihan *child broker* secara acak serta dengan metode tanpa mekanisme pemilihan *child broker*.

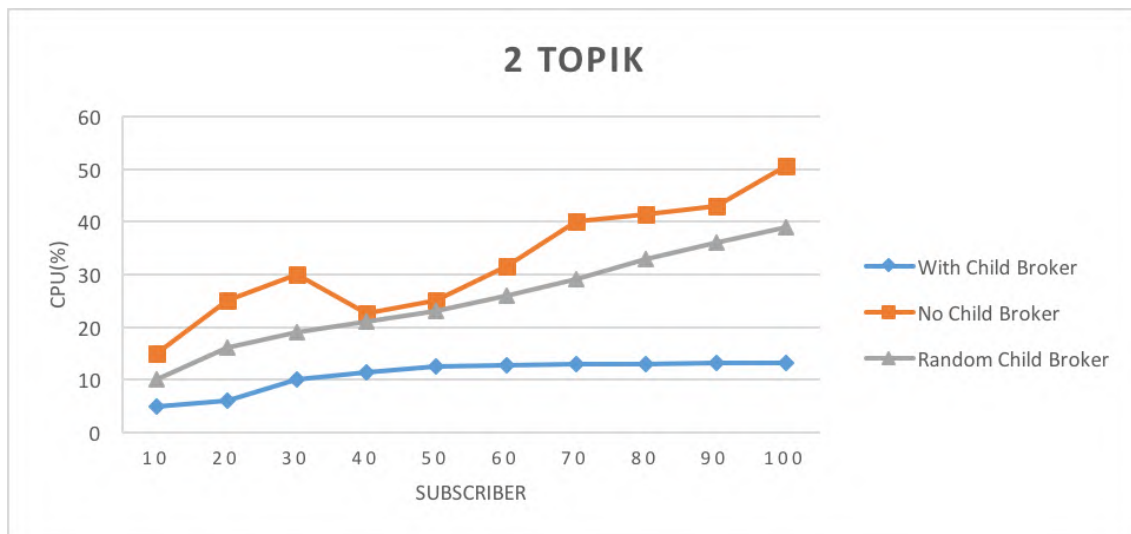
##### **4.2.1. Hasil Pengujian Parameter Penggunaan CPU Broker**

Pada pengujian parameter penggunaan CPU pada *broker* ini digunakan alat bantu Network Traffic Generator(WAN Killer) untuk membebani interface *broker* dengan *traffic load* tertentu dan Prime95 untuk memberikan kesibukan pada node. Ukuran data yang digunakan untuk seluruh topik disamakan dengan ukuran data sebesar 5 KB berupa teks. Pada pengujian untuk parameter CPU ini dibagi lagi menjadi 2 skema dengan perbedaan jumlah topik subskripsi.

a. **Skenario 2 Topik Subskripsi**

Pada pengujian parameter CPU dengan 2 Topik Subskripsi ini, diberikan topik Earthquake dan Flood dengan *event publish* diberikan tiap interval 1 menit dan untuk mendapatkan hasil objektif, pengujian dilakukan sebanyak 10 kali untuk masing-masing metode. Jumlah *publisher* yang digunakan adalah sebanyak 10. Metode yang digunakan sebagai perbandingan adalah metode sistem *publish subscribe* tanpa pemilihan *child broker*, dan sistem *publish subscribe* dengan pemilihan *child broker* secara acak.





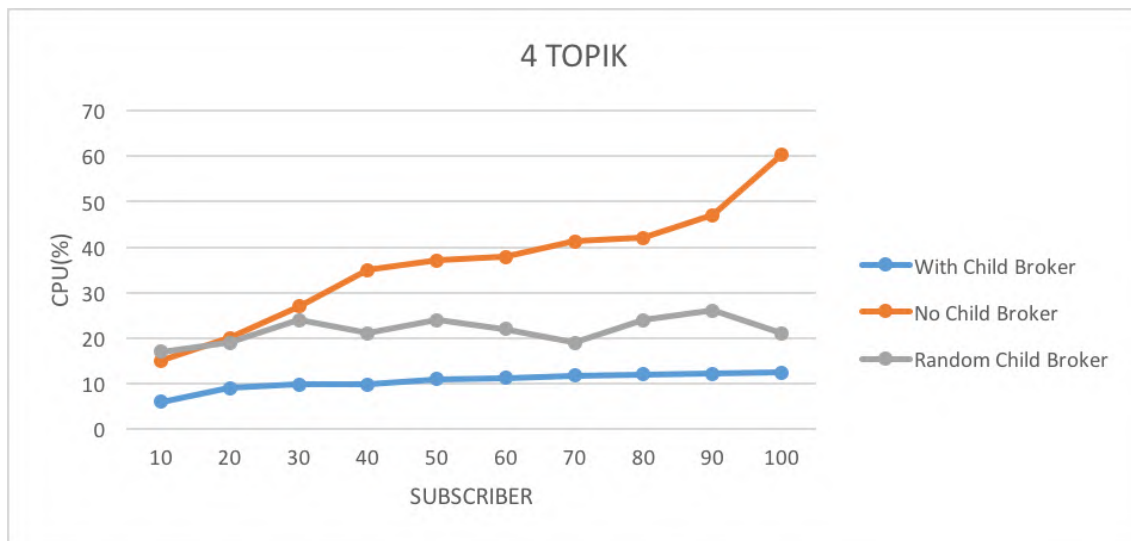
**Gambar 4. 2. Hasil Pengujian Penggunaan CPU Broker untuk 2 Topik**

Dari grafik pada Gambar 4.2. terlihat hasil penggunaan CPU pada metode yang diusulkan memiliki *range* penggunaan CPU paling rendah dengan rata-rata penggunaan CPU sebesar 11% dibanding dengan metode tanpa pemilihan child broker dengan rata-rata hampir 3 kali lipat dari metode yang diusulkan sebesar 32.4%. Hasil metode yang diusulkan juga lebih baik dibanding metode dengan pemilihan *child broker* secara acak dengan rata-rata 2 kali dari metode yang diusulkan sebesar 25.2%.

#### **b. Skenario 4 Topik Subskripsi**

Pada pengujian parameter CPU dengan 4 Topik Subskripsi ini, diberikan topik *Earthquake*, *Flood*, *Storm* dan *Landslide* dengan *event publish* diberikan tiap interval 1 menit dan untuk mendapatkan hasil objektif, pengujian dilakukan sebanyak 10 kali untuk masing-masing metode. Jumlah *publisher* yang digunakan adalah sebanyak 10. Metode yang digunakan sebagai perbandingan adalah metode sistem *publish subscribe* tanpa pemilihan *child broker*, dan sistem *publish subscribe* dengan pemilihan *child broker* secara acak.

Dari grafik pada Gambar 4.3. terlihat hasil penggunaan CPU pada metode yang diusulkan memiliki *range* penggunaan CPU paling rendah dengan rata-rata penggunaan CPU sebesar 10.49% dibanding dengan metode tanpa pemilihan *child broker* dengan rata-rata 3 kali lipat dari metode yang diusulkan sebesar 36.24%. Hasil metode yang diusulkan juga lebih baik dibanding metode dengan pemilihan *child broker* secara acak dengan rata-rata 2 kali dari metode yang diusulkan sebesar 21.27%.



**Gambar 4. 3. Hasil Pengujian Penggunaan CPU Broker untuk 4 Topik**

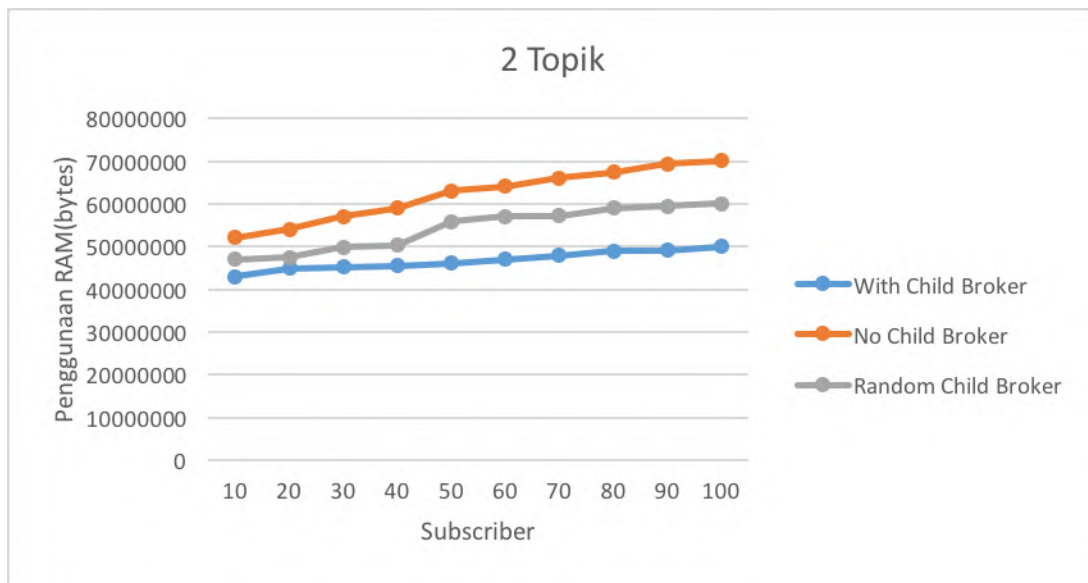
Penghematan penggunaan CPU pada metode yang diusulkan ini terjadi karena adanya mekanisme pemilihan *child broker*. Sehingga apabila penggunaan sumber daya pada *broker* tidak lebih besar dari salah satu node saja untuk topik tertentu pada sistem *publish subscribe*, maka *broker* tersebut akan menugaskan node yang memiliki sumber daya yang lebih besar dari node tersebut sebagai *child broker* dengan formula regresi yang telah dipaparkan pada Bab 3.

#### **4.2.2. Hasil Pengujian Parameter Penggunaan RAM Broker**

Hasil pengujian parameter penggunaan RAM bertujuan untuk mengetahui rata-rata beban RAM pada *broker*. Pada pengujian ini digunakan alat bantu Network Traffic Generator(WAN Killer) untuk membebani *interface broker* dengan *traffic load* tertentu dan Prime95 untuk memberikan kesibukan pada node. Ukuran data yang digunakan untuk seluruh topik disamakan dengan ukuran data sebesar 5 KB berupa teks. Pada pengujian untuk parameter RAM ini dibagi lagi menjadi 2 skema dengan perbedaan jumlah topik subskripsi.

##### **a. Skenario 2 Topik Subskripsi**

Pada pengujian parameter RAM dengan 2 Topik Subskripsi ini, diberikan topik *Earthquake* dan *Flood* dengan *event publish* diberikan tiap interval 1 menit dan untuk mendapatkan hasil objektif, pengujian dilakukan sebanyak 10 kali untuk masing-masing metode. Jumlah *publisher* yang digunakan adalah sebanyak 10. Metode yang digunakan sebagai perbandingan adalah metode sistem *publish subscribe* tanpa pemilihan *child broker*, dan sistem *publish subscribe* dengan pemilihan *child broker* secara acak.



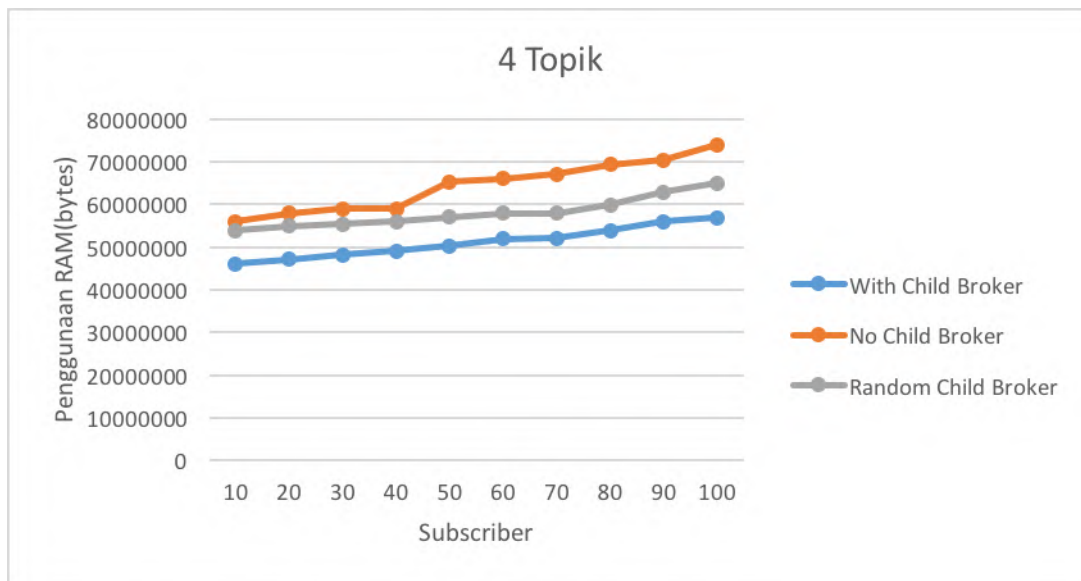
**Gambar 4. 4. Hasil Pengujian Penggunaan RAM Broker untuk 2 Topik**

Dari grafik pada Gambar 4.4. terlihat hasil penggunaan RAM pada metode yang diusulkan memiliki *range* penggunaan RAM paling rendah dengan rata-rata penggunaan RAM sebesar 46747902.8 bytes dibanding dengan metode tanpa pemilihan *child broker* dengan rata-rata sebesar 62197733.5 bytes. Hasil metode yang diusulkan juga lebih baik dibanding metode dengan pemilihan *child broker* secara acak dengan rata-rata sebesar 54303819.5 bytes.

#### **b. Skenario 4 Topik Subskripsi**

Pada pengujian parameter RAM dengan 4 Topik Subskripsi ini, diberikan topik *Earthquake, Flood, Landslide* dan *Storm* dengan *event publish* diberikan tiap interval 1 menit dan untuk mendapatkan hasil objektif, pengujian dilakukan sebanyak 10 kali untuk masing-masing metode. Jumlah *publisher* yang digunakan adalah sebanyak 10. Metode yang digunakan sebagai perbandingan adalah metode sistem *publish subscribe* tanpa pemilihan *child broker*, dan sistem *publish subscribe* dengan pemilihan *child broker* secara acak.

Dari grafik pada Gambar 4.5 terlihat hasil penggunaan RAM pada metode yang diusulkan memiliki range penggunaan RAM paling rendah dengan rata-rata penggunaan RAM sebesar 51122635.8 bytes dibanding dengan metode tanpa pemilihan *child broker* dengan rata-rata sebesar 64372747.3 bytes. Hasil metode yang diusulkan juga lebih baik dibanding metode dengan pemilihan *child broker* secara acak dengan rata-rata sebesar 58032160.6 bytes.



**Gambar 4. 5 Hasil Pengujian Penggunaan RAM Broker untuk 4 Topik**

Penghematan beban RAM pada metode yang diusulkan ini terjadi karena adanya mekanisme pemilihan *child broker*. Sehingga apabila penggunaan sumber daya pada *broker* tidak lebih besar dari salah satu node saja untuk topik tertentu pada sistem *publish subscribe*, maka *broker* tersebut akan menugaskan node yang memiliki sumber daya yang lebih besar dari node tersebut sebagai *child broker* dengan formula regresi yang telah dipaparkan pada Bab 3. Adapun beban RAM pada *broker* di metode yang diusulkan masih belum bisa memiliki perbedaan cukup jauh dikarenakan, *broker* masih harus menjalankan fungsi *update* informasi sumber daya node yang tergabung pada jaringan yang sama dengan *broker*.

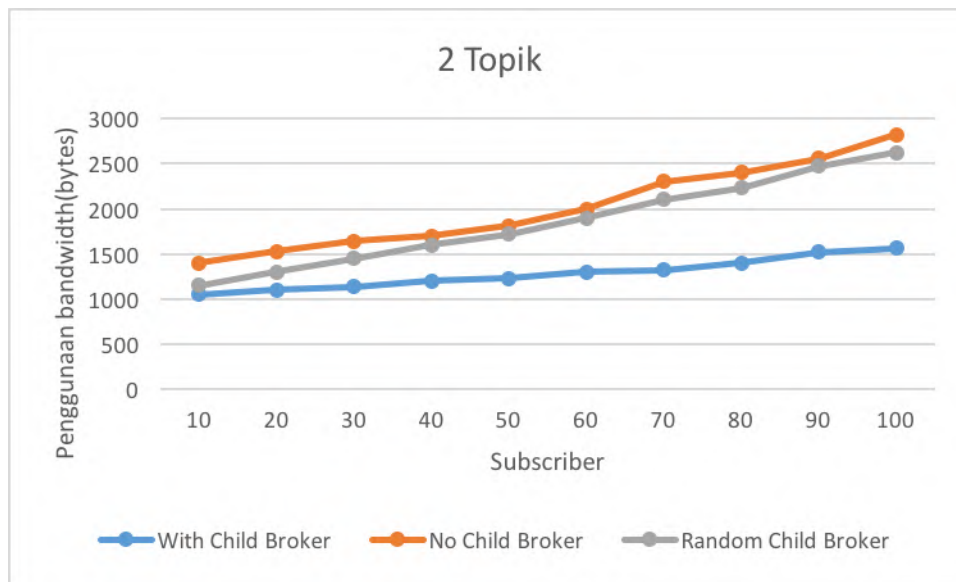
#### **4.2.3. Hasil Pengujian Parameter Penggunaan *Bandwidth***

Hasil pengujian parameter penggunaan *bandwidth* bertujuan untuk mengetahui rata-rata penggunaan *bandwidth* pada ketiga *broker*. Ukuran data yang digunakan untuk seluruh topik disamakan dengan ukuran data sebesar 5 KB berupa teks. Pada pengujian untuk parameter *bandwidth* ini dibagi lagi menjadi 2 skema dengan perbedaan jumlah topik subskripsi.

##### **a. Skenario 2 Topik Subskripsi**

Pada pengujian parameter penggunaan *bandwidth* dengan 2 Topik Subskripsi ini, diberikan topik *Earthquake* dan *Flood* dengan *event publish* diberikan tiap interval 1 menit dan untuk mendapatkan hasil objektif, pengujian dilakukan sebanyak 10 kali untuk masing-masing metode. Jumlah *publisher* yang digunakan adalah sebanyak 10.

Metode yang digunakan sebagai perbandingan adalah metode sistem *publish subscribe* tanpa pemilihan *child broker*, dan sistem *publish subscribe* dengan pemilihan *child broker* secara acak.

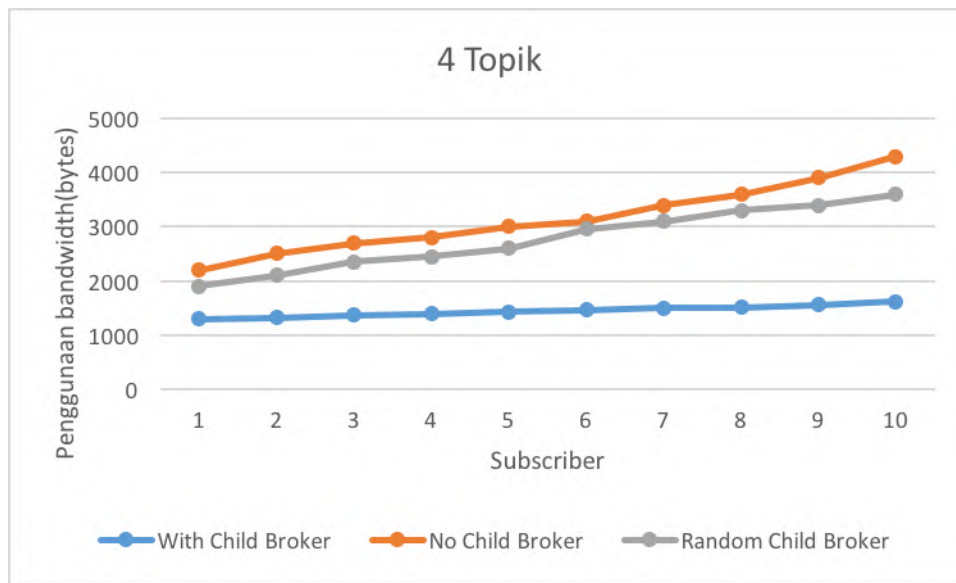


**Gambar 4. 6 Hasil Pengujian Penggunaan *Bandwidth Broker* untuk 2 Topik**

Dari grafik pada Gambar 4.6. terlihat hasil penggunaan *bandwidth* pada metode yang diusulkan memiliki penggunaan *bandwidth* paling rendah dengan rata-rata sebesar 1.28Mbps dibanding dengan metode tanpa pemilihan *child broker* dengan rata-rata hampir dua kali lipat sebesar 2Mbps. Hasil metode yang diusulkan juga lebih baik dibanding metode dengan pemilihan *child broker* secara acak dengan rata-rata sebesar 1.85Mbps.

#### **b. Skenario 4 Topik Subskripsi**

Pada pengujian parameter penggunaan bandwidth dengan 4 Topik Subskripsi ini, diberikan topik *Earthquake*, *Flood*, *Landslide* dan *Storm* dengan event *publish* diberikan tiap interval 1 menit dan untuk mendapatkan hasil objektif, pengujian dilakukan sebanyak 10 kali untuk masing-masing metode. Jumlah *publisher* yang digunakan adalah sebanyak 10. Metode yang digunakan sebagai perbandingan adalah metode sistem *publish subscribe* tanpa pemilihan *child broker*, dan sistem *publish subscribe* dengan pemilihan *child broker* secara acak.



**Gambar 4. 7. Hasil Pengujian Penggunaan *Bandwidth Broker* untuk 4 Topik**

Dari grafik pada Gambar 4.7. terlihat hasil penggunaan *bandwidth* pada metode yang diusulkan memiliki penggunaan *bandwidth* paling rendah dengan rata-rata sebesar 1.4Mbps dibanding dengan metode tanpa pemilihan *child broker* dengan rata-rata lebih dari dua kali lipat sebesar 3.15Mbps. Hasil metode yang diusulkan juga lebih baik dibanding metode dengan pemilihan *child broker* secara acak dengan rata-rata 2 kali lipat sebesar 2.8Mbps.

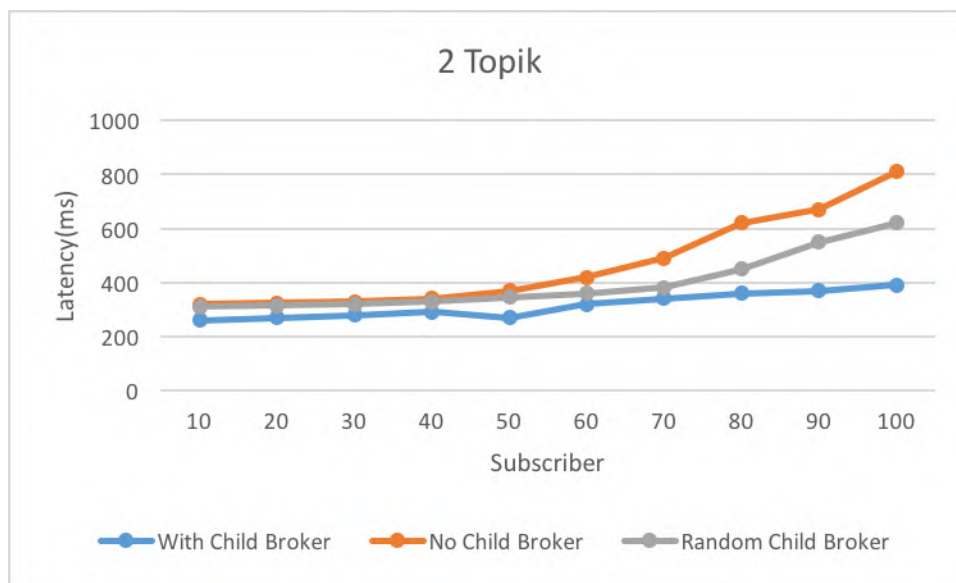
Pada grafik di Gambar 4.6. maupun Gambar 4.7. menunjukkan adanya peningkatan rata-rata penggunaan *bandwidth* ketiga *broker* dikarenakan jumlah *subscriber* yang semakin meningkat dengan kelipatan 10 dan jumlah data yang dipertukarkan semakin banyak. Adapun pada metode pemilihan *child broker* secara acak maupun tanpa pemilihan *child broker* penggunaan *bandwidth* relatif lebih tinggi dikarenakan *broker* utama terus melayani *event subscription* dari node-node *subscriber* yang ada pada sistem tersebut.

#### **4.2.4. Hasil Pengujian Parameter *Latency***

Hasil pengujian parameter penggunaan *latency* bertujuan untuk mengetahui rata-rata penggunaan *latency* pada ketiga *broker*. Ukuran data yang digunakan untuk seluruh topik disamakan dengan ukuran data sebesar 5 KB berupa teks. Pada pengujian untuk parameter *latency* ini dibagi lagi menjadi 2 skema dengan perbedaan jumlah topik subskripsi.

**a. Skenario 2 Topik Subskripsi**

Pada pengujian parameter *latency* dengan 2 Topik Subskripsi ini, diberikan topik *Earthquake* dan *Flood* dengan *event publish* diberikan tiap interval 1 menit dan untuk mendapatkan hasil objektif, pengujian dilakukan sebanyak 10 kali untuk masing-masing metode. Jumlah *publisher* yang digunakan adalah sebanyak 10. Metode yang digunakan sebagai perbandingan adalah metode sistem *publish subscribe* tanpa pemilihan *child broker*, dan sistem *publish subscribe* dengan pemilihan *child broker* secara acak.



**Gambar 4. 8. Hasil Pengujian *Latency* untuk 2 Topik**

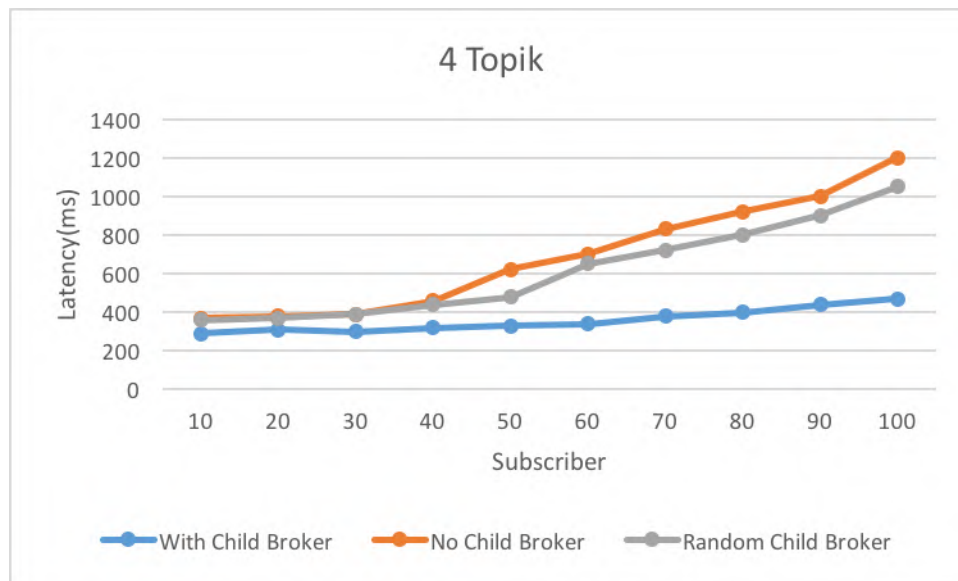
Dari grafik pada Gambar 4.8. terlihat hasil pengujian parameter *latency* pada metode yang diusulkan memiliki *latency* paling rendah dengan rata-rata sebesar 315ms dibanding dengan metode tanpa pemilihan *child broker* dengan rata-rata sebesar 469.5ms. Hasil metode yang diusulkan juga lebih baik dibanding metode dengan pemilihan *child broker* secara acak dengan rata-rata 398ms.

**b. Skenario 4 Topik Subskripsi**

Pada pengujian parameter *latency* dengan 4 Topik Subskripsi ini, diberikan topik *Earthquake*, *Flood*, *Landslide* dan *Storm* dengan *event publish* diberikan tiap interval 1 menit dan untuk mendapatkan hasil objektif, pengujian dilakukan sebanyak 10 kali untuk masing-masing metode. Jumlah *publisher* yang digunakan adalah sebanyak 10. Metode yang digunakan sebagai perbandingan adalah metode sistem *publish subscribe*



tanpa pemilihan *child broker*, dan sistem *publish subscribe* dengan pemilihan *child broker* secara acak.



**Gambar 4. 9. Hasil Pengujian *Latency* untuk 4 Topik**

Dari grafik pada Gambar 4.9. terlihat hasil pengujian parameter *latency* pada metode yang diusulkan memiliki *latency* paling rendah dengan rata-rata sebesar 358ms dibanding dengan metode tanpa pemilihan *child broker* dengan rata-rata hampir dua kali lipat sebesar 687ms. Hasil metode yang diusulkan juga lebih baik dibanding metode dengan pemilihan *child broker* secara acak dengan rata-rata 616ms. Pada pengujian *latency* ini, waktu *delay* yang dibutuhkan oleh metode yang diusulkan untuk memasukkan sebuah node ke dalam SON tertentu sesuai konten dapat diabaikan karena waktu yang dibutuhkan tidak terlalu berpengaruh kepada penambahan *latency* dengan rata-rata sebesar 3ms.

Rendahnya nilai *latency* pada metode yang diusulkan ini terjadi karena adanya mekanisme pemilihan *child broker*. Sehingga apabila penggunaan sumber daya pada *broker* tidak lebih besar dari salah satu node saja untuk topik tertentu pada sistem *publish subscribe*, maka *broker* tersebut akan menugaskan node yang memiliki sumber daya yang lebih besar dari node tersebut sebagai *child broker* dengan formula regresi yang telah dipaparkan pada Bab 3. Mekanisme ini membuat *event subscription* yang dikeluarkan oleh *subscriber* lebih cepat untuk ditangani.



*[Halaman ini Sengaja Dikosongkan]*

## **BAB V**

### **KESIMPULAN DAN SARAN**

Pada Bab ini dijelaskan kesimpulan akhir yang didapatkan dari penelitian yang telah dilakukan dan juga dipaparkan saran-saran yang bersifat membangun untuk penelitian selanjutnya di masa yang akan datang.

#### **5.1. Kesimpulan**

Pengujian dan analisis yang telah dilakukan menghasilkan beberapa kesimpulan penelitian sebagai berikut:

1. Metode yang diusulkan pada penelitian ini, yakni dengan pemilihan *child broker* sebagai asisten *broker* utama terbukti mampu menghemat sumber daya *broker* sehingga mampu menghindari *single node failure*.
2. Penambahan jumlah topik subskripsi serta jumlah *subscriber* cenderung menambah beban *broker* baik dari penggunaan CPU, RAM, *bandwidth* maupun *latency*.
3. Nilai parameter pengujian CPU, RAM, *bandwidth* dan *latency* terbukti selalu lebih kecil dibanding dengan metode konvensional (tanpa pemilihan *child broker*, dan pemilihan *child broker* secara acak).
4. Dari hasil pengujian yang dilakukan, didapatkan penghematan penggunaan sumber daya CPU dan RAM pada *broker* yang mencapai 50% sebesar 10.49% dan 46747902.8 dibanding dengan metode tanpa pemilihan *child broker* secara acak untuk skenario 2 Topik subskripsi. Didapatkan pula, metode yang diusulkan juga mampu menurunkan rata-rata penggunaan *bandwidth* hingga 64% sebesar 1.28 Mbps apabila dibandingkan dengan metode tanpa pemilihan *child broker*.

#### **5.2. Saran**

Topik optimasi sumber daya pada *broker* pada titik *load balancing* pada mekanisme komunikasi *publish subscribe* masih menjadi topik yang menarik. Terlebih jika optimasi ini dilakukan pada sisi *subscriber* dengan membuat *subscriber* juga turut berperan aktif dalam mekanisme tersebut. Oleh karena itu, dibutuhkan penelitian tambahan agar terdapat mekanisme *load balancing* pula di sisi *subscriber* agar node yang terpilih sebagai asisten *broker* tidak mengalami kegagalan.

Penggunaan SON pada mekanisme komunikasi *peer to peer* telah banyak dikembangkan. Hanya, untuk mekanisme komunikasi *publish subscribe* masih

membutuhkan penelitian yang lebih dalam, karena algoritmanya yang menggunakan *tree*. Sehingga dibutuhkan penelitian lebih lanjut, sedalam apa *tree* yang harus dibentuk agar pembentukan *tree* SON tidak malah membebani sistem itu sendiri.

## DAFTAR PUSTAKA

- JP Molina, & G.Pardo C, 2012. A Content-aware Bridging Service for Publish-Subscribe Environments. Science Direct: Journal of System and Softwares, pp. 108-124.*
- A. Corradi, L. Foschini, L. Nardelli, 2010. A DDS-compliant infrastructure for fault-tolerant and scalable data dissemination. IEEE Symposium on Computers and Communications (ISCC) (2010), pp. 489–495*
- Y. Park, D. Chung, D. Min, E. Choi, 2011. Middleware integration of DDS and ESB for interconnection between real-time embedded and enterprise systems. Convergence and Hybrid Information Technology (2011), pp. 337–344*
- RTI, 2012. Real-Time Innovations (RTI) DDS Data Distribution Service. <http://www.rti.com/>.*
- OMG, 2006. Data-Distribution Service for Real-Time Systems (DDS). v1.2. Tech. Rep., OMG. <http://www.omg.org/cgi-bin/doc?formal/07-01-01.pdf>.*
- Esposito, et al, 2013. Survey on Reliability in Publish/Subscribe Service. ACM: Journal Computer Networks: The International Journal of Computer and Telecommunications Networking, pp 1318-1343.*
- Liu, et al, 2003. Survey of Publish Subscribe Event System. In Indiana Computer Science Technical Report TR 574, 2003.*
- Doulkeridis, et al, 2010. Distributed Semantic Overlay Networks. Citeseer.*

*[Halaman ini Sengaja Dikosongkan]*

## BIODATA PENULIS



Penulis, Dian Rahma Latifa Hayun, lahir di kota Madiun pada tanggal 22 Oktober 1991. Penulis adalah anak kelima dari lima bersaudara dan dibesarkan di kota Magetan, Jawa Timur.

Penulis menempuh pendidikan formal di SD Negeri PurwodadiI (1997-2003) SMPN 1 Barat (2003-2006), dan SMA Negeri 1 Maospati (2006-2009).

Pada tahun 2009-2013, penulis melanjutkan pendidikan S1 di Jurusan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember Surabaya, Jawa Timur. Pada tahun 2013-2016, penulis melanjutkan pendidikan Magister S2 di jurusan yang sama, yaitu Jurusan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember Surabaya, Jawa Timur.

Di Jurusan Teknik Informatika, penulis mengambil bidang minat *Net Centric Computing*. Penulis juga aktif dalam organisasi kemahasiswaan seperti: Lembaga Pers Mahasiswa dan Keluarga Muslim Informatika sebagai sekretaris departemen. Penulis juga pernah menjadi juara 1 untuk lomba Karya Tulis Ilmiah di Gelar Karya Mahasiswa (GEMASTIK) 2011, serta Juara 2 pada ITS Expo tahun 2011 bidang pengembangan permainan. Penulis dapat dihubungi melalui alamat email [dian.rahma.lh@gmail.com](mailto:dian.rahma.lh@gmail.com)

*[Halaman ini Sengaja Dikosongkan]*