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

Dian Rahma L. Hayun[1)] dan Waskitho Wibisono[2)]
[1, 2)]Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember
Jl. Teknik Kimia, Gedung Teknik Informatika, Kampus ITS Sukolilo, Surabaya, 60111
e-mail: yraz.nick@gmail.com[1)], waswib@gmaill.com[2)]

ABSTRAK

Teknologi yang ada memunculkan sebuah paradigma bahwa mesin-mesin dan perangkat-perangkat pendukung yang ada harus dibangun dan dihubungkan 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.

Mekanisme Publish/Subscribe merupakan mekanisme yang sesuai untuk domain permasalahan ini, karena 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 parameter QoS untuk memenuhi kebutuhan komunikasi.

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

Oleh karena itu, penelitian ini mengusulkan mekanisme pemilihan asisten broker yang disebut child broker(s) pada area multi-zone yang bertujuan mengurangi beban kerja pada broker dari tiap zona. Hal ini dilakukan untuk meningkatkan skalabilitas dan mengurangi network traffic antar zona.

Dari hasil pengujian, 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. Metode yang diusulkan juga mampu menurunkan rata-rata penggunaan bandwidth hingga 64% sebesar 1.28 Mbps apabila dibandingkan dengan metode konvensional.

***Kata kunci**: Publish/subscribe, DDS, Brokers*

ABSTRACT

The 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 handle the 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, because it provides a mechanism to handle the scalability and data delivery procedures separately. Object Management Group Data Distribution Service is a protocol of data centric publish/subscribe with QoS parameters of communication.

To optimize DDS that can only be used for communication in one zone, several studies have been conducted, so that DDS can be implemented in multi-zone area. Some researches use a broker as a bridge between the zones. However, these studies do not discuss how to solve load balancing problem for broker to avoid failures. Therefore, this study proposes a selection method of a broker's assistant called child broker(s) in a multi-zone area aiming to reduce the load on the broker for each zone, to improve the scalability and reduce network traffic between different zones.

From the test, the proposed method decreases CPU and RAM resource usage until 50% at 10.49% and 46747902.8 bytes compared to the conventional method for 2 topic subscriptions. It also decreases the bandwidth average usage till 64% from 2 Mbps for the conventional method to 1.28 Mbps.

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

I. PENDAHULUAN

MEKANISME komunikasi *Publish/Subscribe* merupakan komunikasi tidak langsung, dimana komunikasi terjadi antar entitas dalam sebuah sistem terdistribusi melalui sebuah *perantara* antara pengirim dan penerima[1]. Terdapat tiga entitas utama pada model komunikasi ini, yaitu *broker* sebagai perantara, *publisher* sebagai penyedia data dan *subscriber* sebagai pihak yang membutuhkan data. Komunikasi antara *publisher* dan *subscriber* terjadi secara asinkron. Maknanya, komunikasi tidak terjadi secara langsung tetapi

melalui perantara *broker,* sehingga antara *publisher* dan *subscriber* tidak perlu saling mengetahui dan berhubungan secara langsung.

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 [*2*].

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.

Penelitian REVENGE menyediakan mekanisme untuk membangun *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 mengijinkan 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 [*3*]. Pada [*4*] mengajukan sebuah arsitektur untuk menghubungkan DDS dengan ESB (*Enterprise Service Bus*) yang mengijinkan sistem untuk saling bertukar data antara DDS dengan *enterprise system.* Hanya saja, fokus penelitian tersebut hanya pada poin interoperabilitas dan tidak membahas bagaimana mekanisme untuk meningkatkan skalabilitas sistem [*4*].

Oleh karena itu beberapa penelitian sebelumnya mencoba menyelesaikan permasalahan ini. Beberapa solusi DDS *broker* telah ada untuk menangani isu ini. Pada [*5*] 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 *subscribtion* dari para *subscriber* dan meneruskan *subscribtion* tersebut ke zona yang kedua jika ternyata pada zona yang sama tidak ditemukan *publisher* yang sesuai [*5*]. Sayangnya, pada penelitian tersebut 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.*

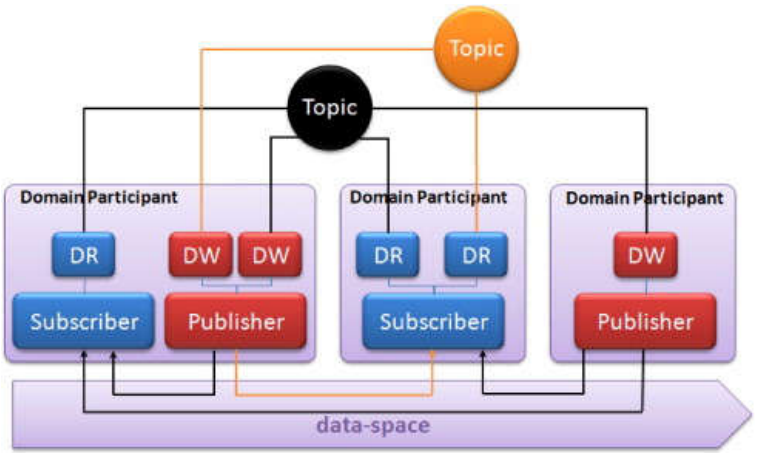
Oleh karena itu, penulis mengusulkan sebuah penelitian untuk pemilihan asisten *broker* yag disebut sebagai *child borker(s)* pada area multi-zone yang bertujuan untuk mengurangi beban kerja pada *broker* dari tiap zona yang berbeda. Hasil yang diharapkan dari penelitian ini adalah berkurangnya trafik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

II. KAJIAN PUSTAKA

Model komunikasi *publish/subscribe* merupakan jenis komunikasi *indirect* (tidak langsung) dalam melakukan pertukaran data. Pada model komunikasi ini, *broker*-lah yang mengatur seluruh jalannya komunikasi dan informasi ketersediaan data untuk topik atau channel tertentu. Model komunikasi ini cocok untuk digunakan pada kebutuhan system terhadap data secara *real-time,* karena proses pengiriman data nya yang asinkron.

A. Data Distribution Service

DDS 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 1 menunjukkan hubungan antar entitas dalam sebuah DDS [2].



Gambar 1. Hubungan Entitas DDS

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 *midleware* akan melakukan aksi tertentu.

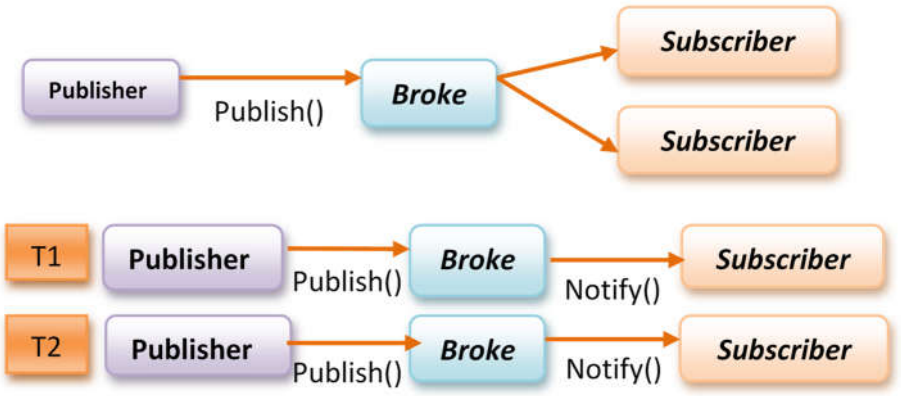
Pada dasarnya, DDS hanya diimplementasikan pada sistem pendistribusian data dimana tiap node yang akan melakukan pertukaran data berada pada satu lokasi yang sama atau terletak pada satu jaringan LAN. Namun, pada perkembangannya DDS mengijinkan terjadinya komunikasi antar jaringan yang berbeda.

B. Model Komunikasi Publish Subscribe

Publish/Susbcribe merupakan mekanisme atau paradigma komunikasi untuk sistem terdistribusi. Tiap node yang berada pada jaringan *publish/subscribe* saling berkomunikasi denan mengirimkan (*publishing*) data dan menerima (*subcribing*) 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.

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.

Ketika *subscriber* telah aktif kembali, maka dia akan mendapatkan pesan notifikasi akan adanya *event* yang baru. *Publisher* tetap bisa memproduksi *events* meskipun *susbcriber* sedang tidak aktif. Yang ketiga adalah S*ynchronization decoupling. Hal ini berarti bahwa komunikasi antara publisher dan subscriber* terjadi secara asinkronous [6]. Gambar 2 menunjukkan gambaran prinsip *time* dan *space* decoupling yang disediakan oleh mekanisme *publish/subscribe.*



Gambar 2. Prinsip *Time and Space Decoupling* Mekanisme *Publish/Subscribe*

Gambar 3. Skema Umum Mekanisme *Publish/Subscribe* [6]

Pada Gambar 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*) [7]. *Broker* 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.*

C. Event Distribution Scheme

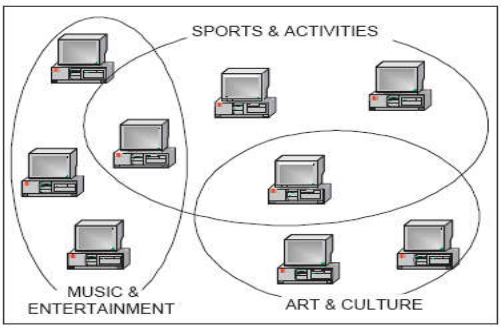
Sistem *publish/subscribe* harus mendukung komunikasi antara *publisher* dan *subscriber* yang terdistribusi letaknya dan dalam jumlah yang besar. Liu menjelaskan bahwa ketika sistem berkembang, maka secara otomatis jumlah *subscriber* dan *publisher* juga akan bertambah. Maka hal ini mendorong adanya kebutuhan terhadap suatu skema pendistribusian *event* agar data disebarkan 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 [7].

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.

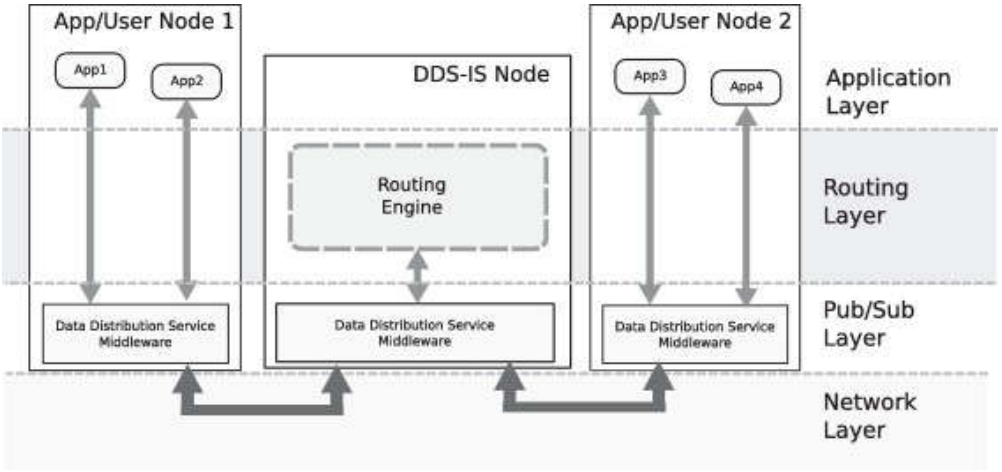
D. Semantic Overlay Network

SON merupakan sebuah pendekatan yang dilakukan untuk mengelompokkan node-node atau *peers* yang terdapat dalam satu jaringan secara tematik [*8*]. Yakni berdasarkan kesamaan topik tertentu dari data yang ada pada node-node tersebut. Contoh sederhana dari SON dapat dilihat pada Gambar 4. Pada gambar tersebut, tiap node dikumpulkan berdasarkan kesamaan dari konten mereka. Yakni *Sport &Activities, Music & Entertainment, Art & Culture.* Yang perlu digaris bawahi 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.*

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.



Gambar 4. SON pada Jaringan P2P



Gambar 5. Arsitektur DDS

E. Content Aware Bridging untuk Mekanisme Publish/Subscribe

Pada penelitian DDS-IS menjelaskan bahwa butuh adanya sebuah layanan yang meningkatkan skalabilitas DDS dan interoperabilitas dari aplikasi DDS. Pada penelitiannya, dibangun sebuah layanan yang mampu menjembatani komunikasi beberapa aplikasi DDS di dua zona yang berbeda. DDS-IS terdiri dari empat *layer*, yani *layer* aplikasi, *routing*, pub/sub, dan *network* layer [5]. Gambar 5 menunjukkan arsitektur dari DDS-IS.

Lapisan utama dari DDS-IS adalah lapisan *routing.* 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 tergabung 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 6, ketika *midleware* 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.

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 menakanisme untuk mengurangi kerja *broker* agar *broker* tidak mengalami *overload* [5].

III. METODOLOGI

Pada bab ini akan dijelaskan langkah-langkah yang dilakukan dalam penelitian, meliputi perumusan masalah, perancangan metodologi dan skema pengujian.

A. Perumusan Masalah

Tugas *broker* yang merupakan *central-node,* sebagai pengelola *subscription* dan *publication* akan menguras 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

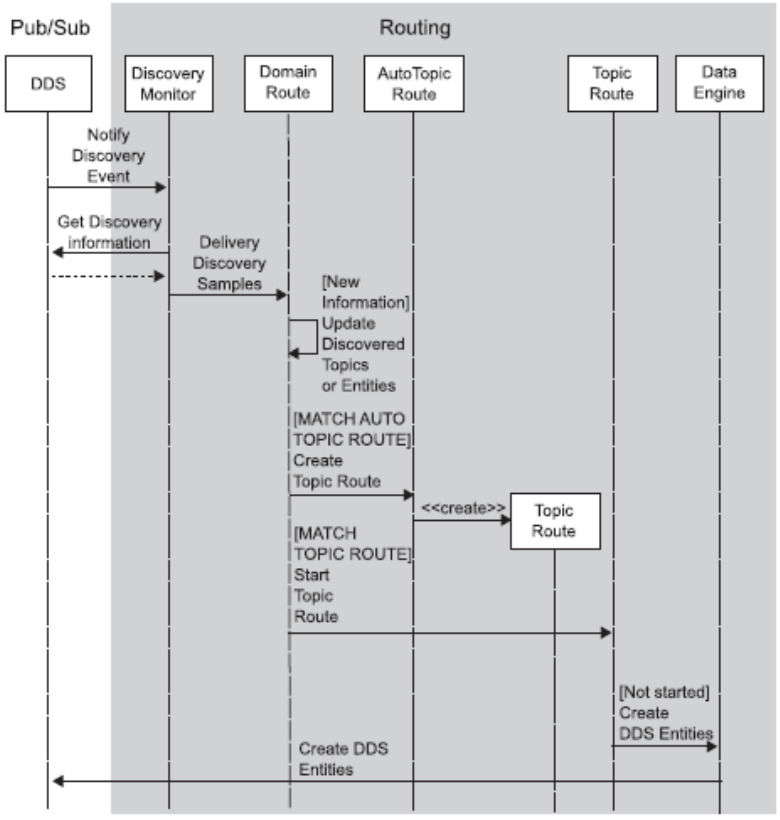
broker (child broker(s)) yang membantunya melakukan pengelolaan *subscription* dan *publication*", maka akan mengurangi beban kerja broker dan mengurangi trafik jaringan antar *broker* dan menambah skalabilitas sistem serta menghindari *overload* pada *broker.* Sehingga fokus pada penelitian masalah ini adalah bagaimana mengurangi beban kerja *broker* dengan memilih asisten broker/*child broker* yang memenuhi kriteria sebagai pembantu *broker.*

B. Perancangan Metodologi

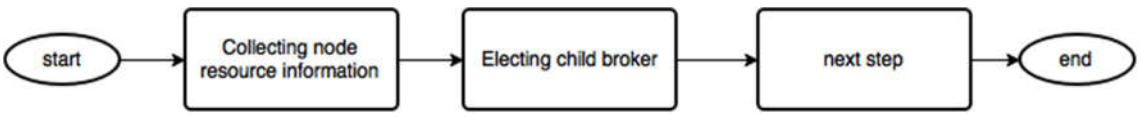
Sebuah *broker* pada mekanisme komunikasi *publish subscribe* dapat mengalami *single node failure* jika mengalami beban berlebih dalam jaringan. Hal ini mengakibatkan tidak tersampaikannya data dari *publisher* ke *subscriber* karena *broker* tidak mampu menangani *request* dan menyampaikan pesan *publishing.* 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 7.



Gambar 6. Diagram Sekuens pada Proses *Discovery Event*



Gambar 7. Konsep Kontribusi

```
{"latitude":"-5.1838",
"longitude":"140.9062",
"type":"earthquake",
"disaster_subtype":"ground_movement",
"damage":"17000 US$",
"total_death":"20011"},
```


Gambar 8. Contoh Data Pengujian

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.

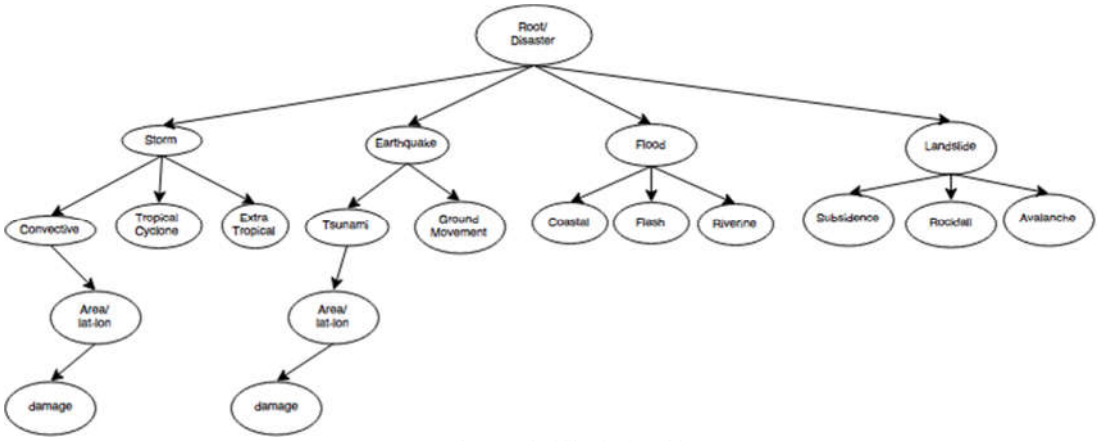
Adapun langkah-langkah yang diajukan dalam penelitian ini untuk mengoptimalkan proses *publish/subscribe* adalah sebagai berikut:

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*

Pada fase *node clustering, nodes* yang berada pada jaringan akan 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[*8*]. Pengelompokkan node dilakukan berdasarkan kedekatan topik *subscription.* Data yang akan digunakan sebagai data training pengelompokkan terdapat pada Gambar 8. 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.* 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.

Setelah melalui fase training, akan didapatkan hirarki klasifikasi seperti yang ditunjukkan pada Gambar 9.



Gambar 9. Klasifikasi Hierarki

TABEL I
DATA TRAINING

No	CPU	GFlop	RAM Speed	CPU Util (%)	Link (Mbps)	Data (Mb)	Waktu (s)
1	CPU A	1.38×10^{-5}	7.50×10^{-6}	27.7	10	5	0.52
2		1.38×10^{-5}	7.50×10^{-6}	33.8	10	10	1.1
3		1.38×10^{-5}	7.50×10^{-6}	34	10	20	2.2
4		1.38×10^{-5}	7.50×10^{-6}	34.7	10	50	5
5		1.38×10^{-5}	7.50×10^{-6}	35.2	10	100	11.3
6		1.38×10^{-5}	7.50×10^{-6}	32.2	20	5	0.27
7		1.38×10^{-5}	7.50×10^{-6}	32.5	20	10	0.55
8		1.38×10^{-5}	7.50×10^{-6}	33	20	20	1.1
9		1.38×10^{-5}	7.50×10^{-6}	33.2	20	50	2.6
10		1.38×10^{-5}	7.50×10^{-6}	33.9	20	100	5.2
11	CPU B	1.18×10^{-5}	7.60×10^{-6}	29	10	5	0.5
12		1.18×10^{-5}	7.60×10^{-6}	31.1	10	10	1.3
13		1.18×10^{-5}	7.60×10^{-6}	31.7	10	20	2
14		1.18×10^{-5}	7.60×10^{-6}	32.2	10	50	5.1
15		1.18×10^{-5}	7.60×10^{-6}	32.3	10	100	11
16		1.18×10^{-5}	7.60×10^{-6}	30.3	20	5	0.25
17		1.18×10^{-5}	7.60×10^{-6}	30.1	20	10	0.57
18		1.18×10^{-5}	7.60×10^{-6}	31	20	20	1.1
19		1.18×10^{-5}	7.60×10^{-6}	31.2	20	50	2.6
20		1.18×10^{-5}	7.60×10^{-6}	31.7	20	100	5.1
21	CPU C	1.34×10^{-5}	7.51×10^{-6}	28.9	10	5	0.51
22		1.34×10^{-5}	7.51×10^{-6}	30.3	10	10	1
23		1.34×10^{-5}	7.51×10^{-6}	30.9	10	20	2
24		1.34×10^{-5}	7.51×10^{-6}	31.4	10	50	5
25		1.34×10^{-5}	7.51×10^{-6}	31.7	10	100	10.7
26		1.34×10^{-5}	7.51×10^{-6}	29.7	20	5	0.27
27		1.34×10^{-5}	7.51×10^{-6}	30.5	20	10	0.56
28		1.34×10^{-5}	7.51×10^{-6}	30.7	20	20	1.1
29		1.34×10^{-5}	7.51×10^{-6}	30.3	20	50	2.6
30		1.34×10^{-5}	7.51×10^{-6}	31.9	20	100	5.3

TABEL II
HASIL REGRESI DATA SEBELUM PROSES NORMALISASI

Regression Statistics	
Multiple R	0.948839331
R Square	0.900296076
Adjusted R Square	0.879524425
Standard Error	1.120879604
Observations	30

TABEL III
HASIL REGRESI SETELAH NORMALISASI

Regression Statistics	
Multiple R	0.998898128
R Square	0.997797471
Adjusted R Square	0.99733861
Standard Error	0.025639265
Observations	30

Fase kedua yakni 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 dengen kedekatan tema atau konten subskripsi.

Pada fase ini digunakan 3 buah CPU, dengan nilai atribut seperti yang ditampilkan pada Tabel I yakni data, waktu pengiriman data dalam jaringan, penggunaan RAM dan CPU. Dari data pada Tabel I dilakukan proses analisis regresi berganda untuk mendapatkan hubungan antara variabel-variabel tersebut dengan variabel dependen (Waktu).

Hasil dari proses regresi dapat dilihat pada Tabel II. Dari proses tersebut didapatkan standar eror yang tinggi. Hal ini menunjukkan bahwa akurasi data kurang baik apabila dijadikan sebagai standar.

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 mentranformasi data tersebut. Setelah proses normalisasi data, dilakukan kembali proses regresi sehingga menghasilkan standar eror yang rendah seperti terlihat pada Tabel III.

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

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

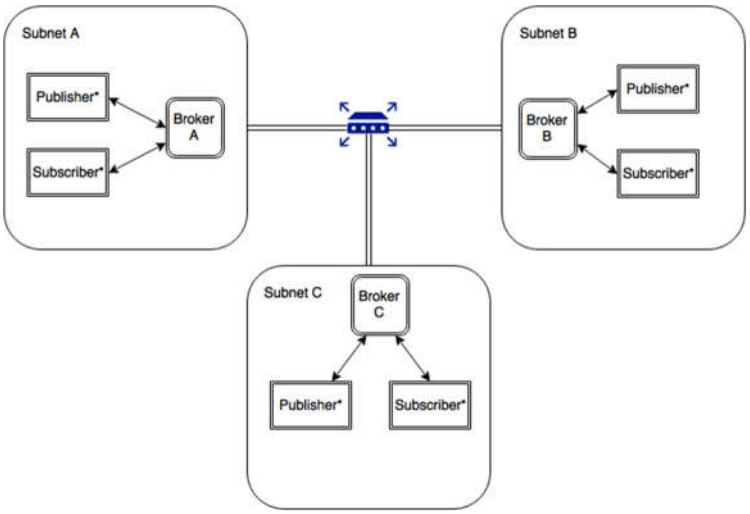
a. GFlop = -0.383238623
b. RAM Speed = -0.383238623
c. CPU Util = 0.440351744
d. Link = -0.978304621
e. Data = 0.982207119

sehingga didapatkan sebuah formula untuk menghitung besaran waktu yang dibutuhkan yang ditunjukkan pada (1).

$$Waktu = 0.982207119\ x\ Data - 0.383238623\ x\ GFlop - 2.728405832\ x\ RAMSpeed - 0.978304621\ x\ link + 0.982207119\ x\ Data \quad (1)$$

TABEL IV
KOEFISIEN REGRESI

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



Gambar 10. Topologi Jaringan Pengujian

C. Skema 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, bandwith* 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 10. 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*

Adapun unuk parameter pengujian yang digunakan pada penelitian ini agar dapat dijadikan sebagai referensi pembanding 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.*

IV. 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.*

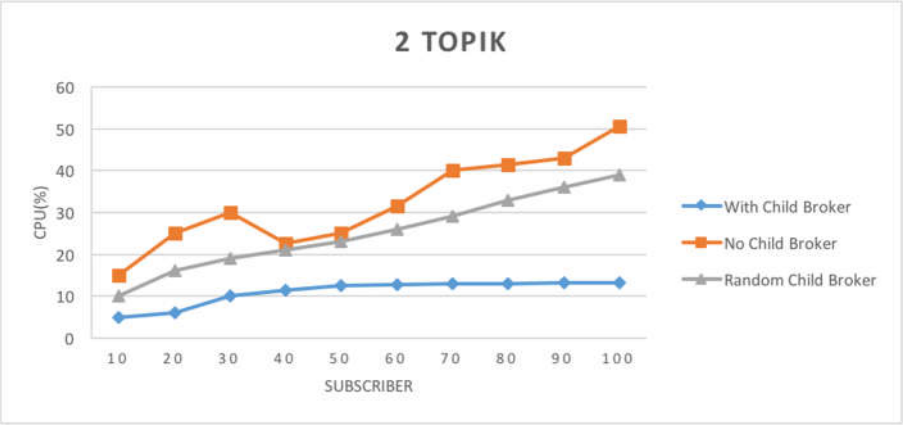
A. Hasil Pengujian Parameter Penggunaan CPU Broker

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.

- *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.

Dari grafik pada Gambar 11 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%.



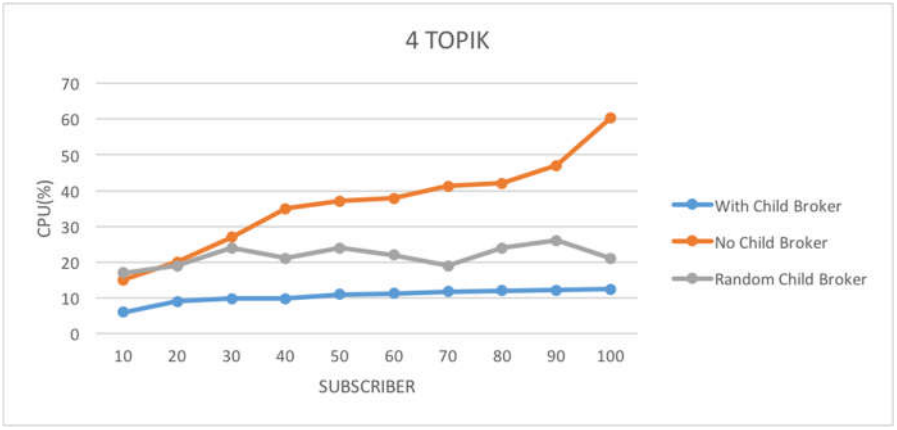
Gambar 11. Hasil Pengujian Penggunaan CPU Broker untuk 2 Topik

- *Skenario 4 Topik Subskripsi*

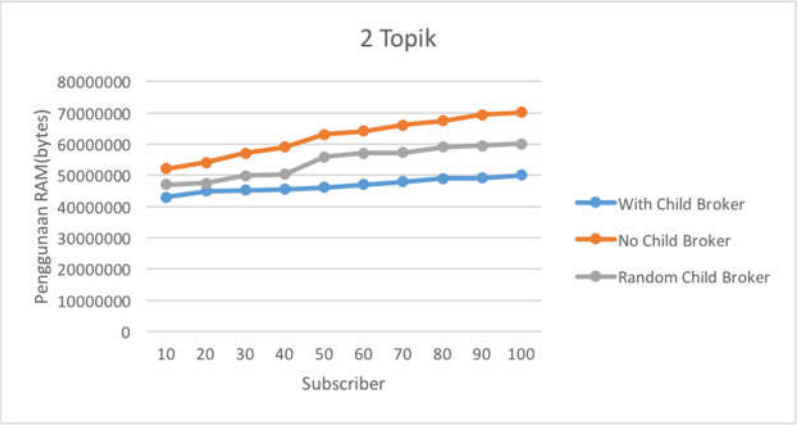
Pada pengujian parameter CPU dengan 4 Topik Subskripsi ini, diberikan topik *Earthquake*, *Flood*, *Storm* dan *Landslide* dengan *event publish* diberikan tiap interval 1 menit dan untuk mendapatkan hasil objektif, pengujian dilakukan sebanyak 10 kali untuk masing-masing metode. Jumlah *publisher* yang 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 12. 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%.

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 pada (1).



Gambar 12. Hasil Pengujian Penggunaan CPU Broker untuk 4 Topik



Gambar 13. Hasil Pengujian Penggunaan RAM Broker untuk 2 Topik

B. 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.

- *Skenario 2 Topik Subskripsi*

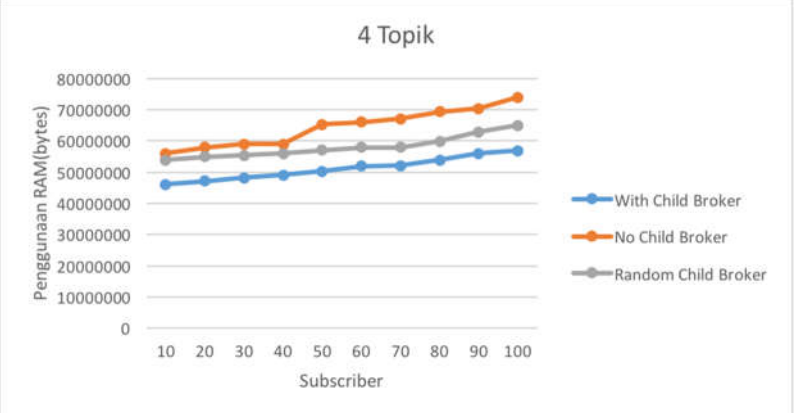
Pada pengujian parameter RAM dengan 2 Topik Subskripsi ini, diberikan topik *Earthquake* dan *Flood* dengan *event publish* diberikan tiap interval 1 menit dan untuk mendapatkan hasil objektif, pengujian dilakukan sebanyak 10 kali untuk masing-masing metode. Jumlah *publisher* yang digunakan adalah sebanyak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

Dari grafik pada Gambar 13 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.

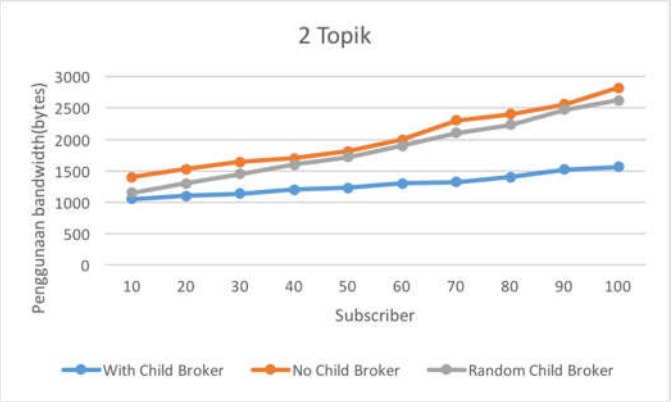
- *Skenario 4 Topik Subskripsi*

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 14 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 14. Hasil Pengujian Penggunaan RAM Broker untuk 4 Topik



Gambar 15. Hasil Pengujian Penggunaan *Bandwidth Broker* untuk 2 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 pada (1). 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.*

C. 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.

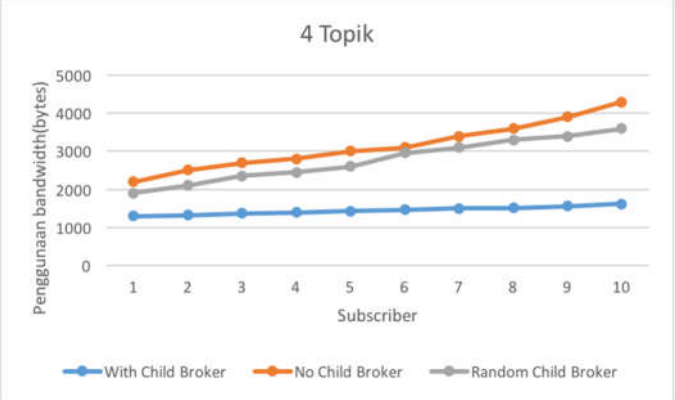
- *Skenario 2 Topik Subskripsi*

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.

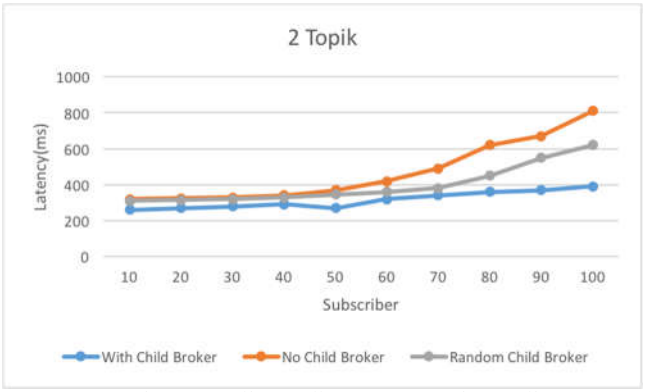
Dari grafik pada Gambar 15 terlihat hasil penggunaan *bandwidth* pada metode yang diusulkan memiliki penggunaan *bandwidth* paling rendah dengan rata-rata sebesar 1.28Mbps dibanding dengan metode tanpa pemilihan child broker dengan rata-rata hampir dua kali lipat sebesar 2Mbps. Hasil metode yang diusulkan juga lebih baik dibanding metode dengan pemilihan *child broker* secara acak dengan rata-rata sebesar 1.85Mbps.

- *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 16. Hasil Pengujian Penggunaan *Bandwidth Broker* untuk 4 Topik



Gambar 17. Hasil Pengujian *Latency* untuk 2 Topik

Dari grafik pada Gambar 16 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 15 maupun Gambar 16 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.

D. 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.

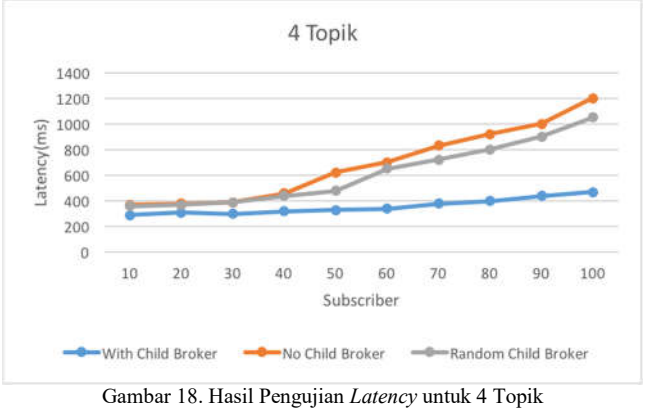
- *Skenario 2 Topik Subskripsi*

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.

Dari grafik pada Gambar 17 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.

- *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 18. Hasil Pengujian *Latency* untuk 4 Topik

Dari grafik pada Gambar 18 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 pada (1). Mekanisme ini membuat *event subscription* yang dikeluarkan oleh *subscriber* lebih cepat untuk ditangani.

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.

A. Kesimpulan

Pengujian dan analisis yang telah dilakukan menghasilkan beberapa kesimpulan penelitian yaitu 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.* Penambahan jumlah topik subskripsi serta jumlah *subscriber* cenderung menambah beban *broker* baik dari penggunaan CPU, RAM, *bandwidth* maupun *latency.* 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). 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*.

B. 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

[1] G. Coulouris, et al, "Characterization of Distributed System" dalam *"Distributed System Concept and Design",* edisi kelima, Boston, Amerika Serikat, 2012, bab1, hal. 2-20.
[2] J.M. Schlesselman, B. Farabaugh, G. P. Castellote. 2004. OMG Data Distribution Service: Architectural Update. In Military Communication Conference, 2013 on (pp 961-967). IEEE.
[3] A. Corradi, L.Foschini, L. Nardelli. 2010. A DDS-Compliant Infrastructure for fault Tolerant and Scalable data Dissemination. In IEEE Symposium on Computers and Communications (ISSC) on (pp 489-495). IEEE.
[4] Y. Park, D. Chung, D. Min, E. Choi. 2011. Middleware Integratuo of DDS and ESB for Interconnection between Real-Time Embedded and Enterprise System. In Convergence and Hybrid Informastion Technology on (pp 337-344). IEEE.
[5] L. V. Jose, J.P. Molina, G.P. Castellote. 2012. A Content-Aware Bridging Service for Publlish Subscribe Environment. Journal of System and Softwares on (pp 489-495). Science Direct.
[6] Esposito. 2013. Survey on Reliability in Publish/Subscribe Service. Journal of Computer Network on (pp 1318-1343). ACM.
[7] Liu, et al. 2003. Survey of Publish Subscribe Event System. Tidak dipublikasikan.
[8] Doulkediris, et al. 2010. Distributed Semantic Overlay Network. Tidak dipublikasikan.