BAB 4 ANALISIS KEBUTUHAN DAN PERANCANGAN

Bab ini bertujuan untuk menjelaskan analisis kebutuhan sistem dan rancangan pengujian yang akan dilakukan pada proses pengujian. Adapun rancangan yang dijabarkan adalah perancangan arsitektur jaringan, alur kerja sistem, serta perancangan pengujian.

4.1 Ruang Lingkup

Sistem ini menggunakan distribusi pesan menggunakan metode *Publish-Subscribe* yang berjalan menggunakan protokol komunikasi *MQTT*. Sistem ini akan berjalan di dalam 1 host. Dimana dalam 1 host terdapat 4 *devices Virtual* yang berjalan menggunakan *VirtualBox*. 4 d*evices* ini terdiri dari 3 *broker* dan 1 *Load balancer*. Pada penelitian ini semua *broker* akan terhubung dan bisa saling bertukar informasi mengenai pesan dari *publisher* yang ditangani oleh setiap *broker*. Sehingga setiap *broker* akan memiliki data yang sama. Kemudian sistem ini menggunakan *load balancer* untuk meng*handle request* dari sisi *client subscriber* ke *broker*.

4.2 Analisis Kebutuhan

4.2.1 Kebutuhan Fungsional

Kebutuhan fungsional merupakan kebutuhan yang harus terpenuhi agar sistem dapat bekerja sesuai dengan tujuan awal. Berikut ini adalah beberapa penjelasan dari kebutuhan fungsional pada sistem ini.

1. *Publisher* mampu melakukan *publish* melalui *load balancer.*
2. *Subscriber* mampu melakukan *subscribe* melalui *load balancer.*
3. *Broker* melakukan penyebaran pesan ketika menerima pesan dari *publisher*.
4. Ketika salah satu *broker* mengalami kegagalan, sistem mampu tetap berjalan sesuai dengan tujuannya.

4.2.2 Kebutuhan Perangkat Keras

1. *Publisher*
    a. Perangkat Laptop
2. *Broker*
    a. Perangkat Laptop
3. *Subscriber*
    a. Perangkat Laptop
4. *Load Balancer*
    a. Perangkat Laptop

4.2.3 Kebutuhan Perangkat Lunak

1. *Host (Publisher & Subscriber)*
    a. Paho MQTT Library *Python*
    b. Wireshark
2. *Broker*
    a. Mosquitto *Broker* Server
    b. VirtualBox
    c. Pidstat
    d. Tshark
3. *Load Balancer*
    a. HAProxy
    b. VirtualBox
    c. Pidstat
    d. Tshark

4.3 Batasan Perancangan

Beberapa batasan untuk mendukung sistem ini bekerja yaitu sebagai berikut :

1. Sistem akan bekerja apabila *Broker* sudah terlebih dahulu dijalankan pada Laptop.
2. Semua *broker* dan *load balancer* berjalan dalam 1 *environment* yang sama.
3. *Broker* dan *load balancer* menggunakan virtualisasi.
4. *Publisher* dan *subscriber* yang digunakan untuk tes menggunakan *script Python*.
5. *Mosquitto* berjalan di *foreground* agar dapat menampilkan *log* yang ada sehingga performanya yang ditunjukkan berbeda ketika *mosquitto* bekerja di *background*.

4.4 Istilah

Pada perancangan akan ada istilah – istilah yang akan digunakan. Pengertian untuk setiap istilah yang digunakan kana dijabarkan pada tabel 4.1 di bawah ini.

Tabel 4.1 Tabel Istilah

Istilah	Penjelasan

Publisher	Merupakan bagian dari metode komunikasi *Publish-Subscribe* yang memiliki tugas untuk mengirimkan pesan ke *Broker.*
Broker	Merupakan bagian dari metode komunikasi *Publish-Subscribe* yang memiliki tugas untuk menerima pesan dari *Publisher* dan meneruskan pesan tersebut ke *Subscriber.*
Subscriber	Merupakan bagian dari metode komunikasi *Publish-Subscribe* yang memiliki tugas menerima pesan dari *Broker* dan melakukan *subscription* ke suatu topik untuk mendapatkan pesan.
Load balancer	Merupakan suatu *device* yang bertugas untuk mengatur beban pada *Broker*. Dimana *load balancer* sebagai gerbang pertukaran pesan antara *subscriber* dengan *broker*
Round robin	Merupakan metode *load balancing* dimana sistem distribusi beban kerja dibagikan secara bergantian.

4.5 Perancangan Sistem

4.5.1 Perancangan Perangkat Keras

Perancangan perangkat keras yang akan digunakan untuk membangung sistem yang akan diuji ini yaitu dibutuhkan *3 broker* dan *1 load balancer*. Dimana semuanya berada pada 1 *host* fisik yang sama. Sehingga dibutuhkan 1 perangkat keras yang akan bekerja sebagai *host* fisik. *Broker* dan *load balancer* berjalan secara *virtual* menggunakan aplikasi *VirtualBox* yang sudah terpasang di *host* fisik.

4.5.2 Perancangan Perangkat Lunak

Perancangan perangkat lunak dilakukan pada *publisher, broker, load balancer,* dan *subscriber*. *Publisher* akan menggunakan bahasa pemrograman *Python.* Pada perancangan ini, *publisher* merupakan *script Python* dimana ketika dijalankan maka akan membuat *request* dalam jumlah yang ditentukan. Hal ini berguna untuk melakukan *stress test* untuk mengetahui bagaimana *performa broker* ketika mendapatkan *request* dalam jumlah besar. *Broker* menggunakan aplikasi *Mosquitto* yang sudah tersedia. *Mosquitto* menggunakan bahasa C yang telah mengimplementasikan *library Paho MQTT*. Kemudian *load balancer* menggunakan aplikasi *HAProxy* yang juga menggunakan bahasa C. Setiap *broker* dan *load balancer* juga sudah terpasang *tshark* untuk mengetahui *traffic* internet yang ada pada *devices* tersebut.

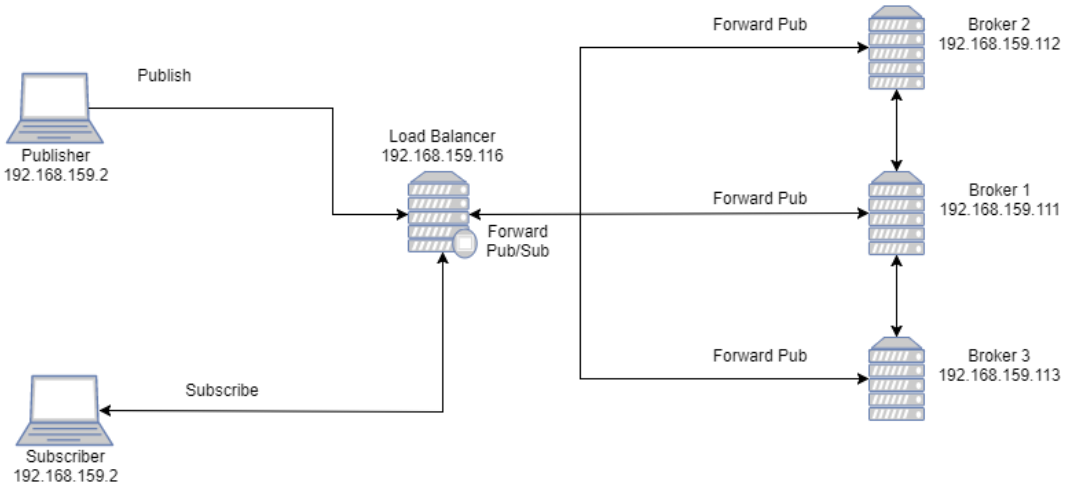
Di sisi *broker*, perancangan perangkat lunak dibuat sehingga b*roker* mampu melakukan menyebarkan pesan *Publish* sehingga ketika satu *broker* menerima pesan, maka b*roker* lain juga menerima pesan yang sama dan kemudian meneruskan pesan dari *publisher* ke *subscriber* sesuai dengan tanggung jawab

setiap *broker*. Terakhir, perancangan perangkat lunak diperlukan pada sistem *load balancer* agar sistem mampu melakukkan *load balancing* ketika menerima *subscribe* dan *publish* ke *broker* yang ada sehingga beban kerja tidak terpusat hanya pada 1 *broker* saja.

4.5.3 Perancangan Arsitektur Jaringan

Perancangan arsitektur dibutuhkan sebagai lingkungan pengujian yang akan dilakukan. Pada penelitian ini semua *devices* berada pada 1 *host* fisik yang sama. *Publisher* mengirimkan pesan topik ke *3 broker* melalui *load balancer* . Dimana pada saat proses berjalan, *broker 1*, *broker 2,* dan *broker 3* akan memiliki data yang sama karena adanya *bridging* antar *broker.* Dengan ini pesan yang diterima oleh *broker 2* akan juga dikirimkan ke *broker 1* dan *broker 2*. Sehingga semua *broker* seolah – olah mendapatkan pesan yang sama. Kemudian *subscriber* melakukan *subscription* ke *load balancer*. Disini *load balancer* bertugas meneruskan pesan *subscription* dari *subscriber* ke *broker* yang akan bertanggung jawab atas *subscriber* tersebut. Setelah *broker* mendapatkan *subscription*, maka jika ada topik yang sesuai dengan *subscription*, maka *broker* akan meneruskan pesan dari *publisher* ke *subscriber* tersebut. Selain itu *publisher* juga harus melakukan *publish* melewati *load* balancer. Dengan adanya *load balancer* ini, tanggung jawab *broker* terhadap *subscriber* dan *publisher* dibagi menjadi 3 sehingga meringankan kerja *broker*.

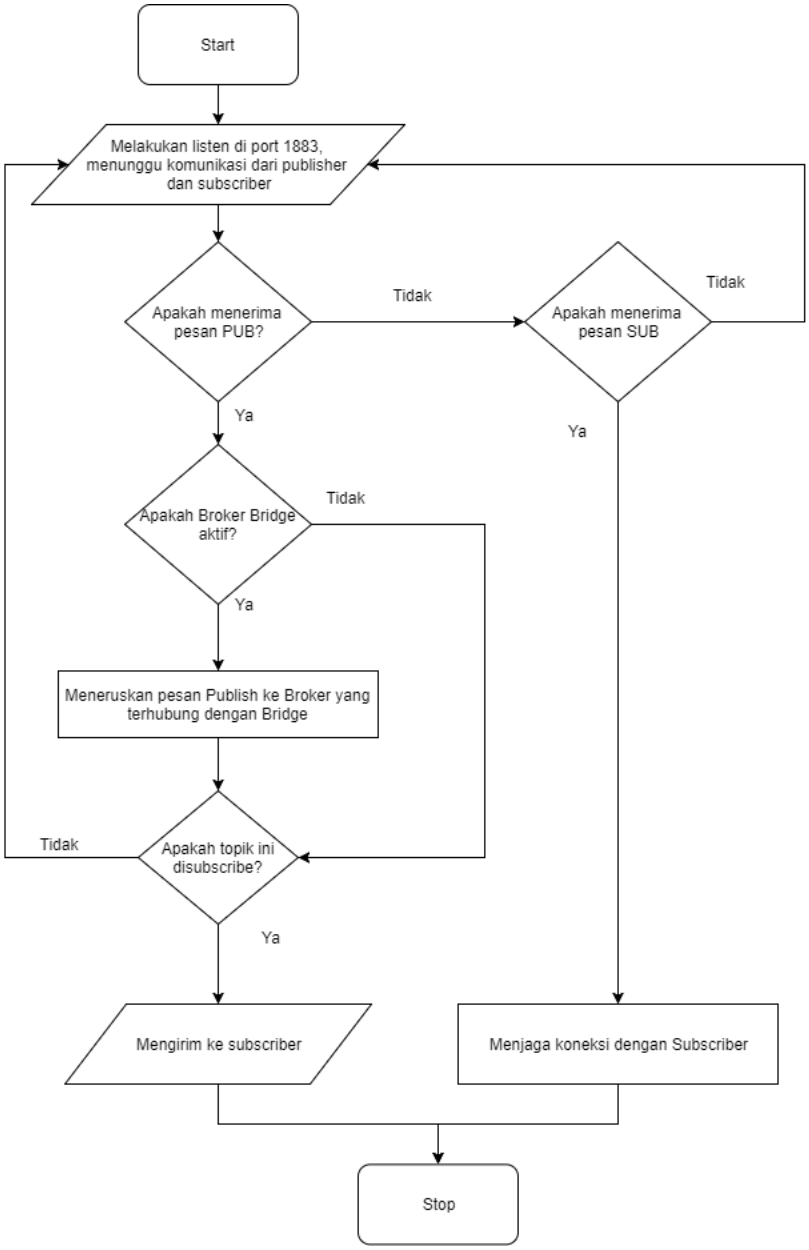
Gambar 4.1 menjelaskan arsitektur jaringan yang akan dibuat. Berdasarkan rancangan arsitektur jaringan pada gambar 4.1. Bisa dilihat bahwa *publisher* memiliki IP 192.168.159.2 yang menjalankan *script Python* untuk melakukan *publish* pesan ke *broker*. Kemudian *broker* memiliki *bridge* untuk memberikan informasi *publish* tersebut ke *broker* lain. Sehingga terjadi penyebaran pesan. *Subscriber* yang juga memliki IP 192.168.159.2 melakukan *subscribe* melalui *load balancer* untuk diarahkan ke *broker* sesuai dengan algoritma *balancing* pada *load balance*. Kemudian jika semua sudah terkoneksi, maka *broker* akan mengirimkan pesan ke *subscriber* sesuai dengan topik yang diikuti ke *load balancer* untuk diteruskan ke setiap *subscriber*.



Gambar 4.1 Arsitektur Jaringan

4.5.4 Perancangan Alur Sistem di Sisi *Broker*

Broker menggunakan aplikasi *Mosquitto*. *Mosquitto* sendiri pada kajian pustaka dijelaskan adalah suatu aplikasi *broker MQTT* yang dikembangan menggunakan library *Paho MQTT*. Gambar 4.2 akan menjelaskan alur sistem di sisi *broker*. Gambar 4.2 menjelaskan bahwa *broker* menunggu koneksi di port 1883, kemudian mengecek jenis pesan yang diterima melalui port 1883. Jika tipenya PUB (*Publish)* maka mengecek topik dari *publisher* dan daftar *subscriber*. Jika topik tersebut sesuai, maka dikirim ke *subscriber* melalui *load balancer*. Jika tipe pesan SUB, maka memasukkan ke dalam daftar *subscriber* dan kembali dilakukan pengecekkan topik yang diterima dengan daftar *subscriber* yang terbaru.

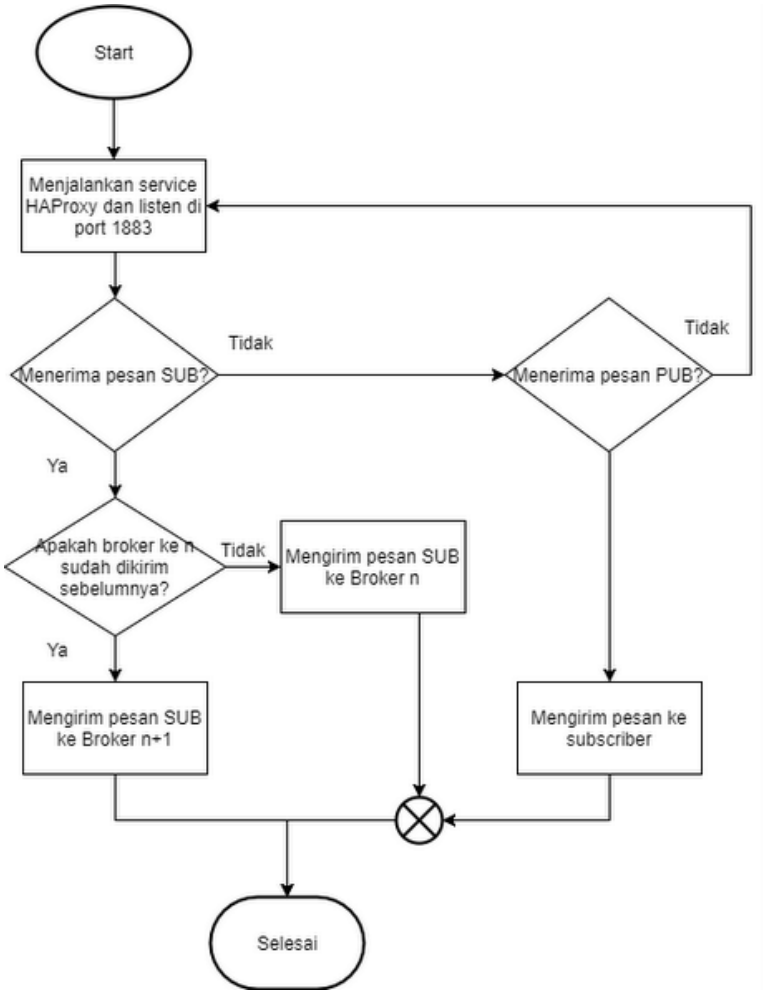


Gambar 4.2 Diagram Alir Kerja Sistem di Sisi *Broker*

4.5.5 Perancangan Alur Sistem di Sisi *Load Balancer*

Untuk melakukan *subscription*, *subscriber* harus melalui *load balancer* terlebih dahulu. *Load balancer* menggunakan aplikasi *HAProxy* dengan mode distribusi *round robin*. Sehingga distribusi *subscriber* akan dibuat secara bergantian antara *broker 1*, *broker 2,* dan *broker 3*. Mekanisme ini akan dijelaskan dalam bentuk diagram alir kerja sistem. Diagram ini akan menjelaskan alur logika yang akan diimplementasikan. Pada gambar 4.3, *Load Balancer* melakukan proses *listen* di port 1883 yang sudah dikonfigurasi dalam *HAProxy*. Kemudian jika menerima pesan ke port 1883 yang menggunakan *TCP*. Maka *Load Balancer* meneruskan pesan tersebut ke *broker* yang sebelumnya belum digunakan. Jika sebelumnya *Load Balancer* sudah meneruskan pesan ke *broker* tersebut. Maka ketika ada

pesan baru, *Load Balancer* akan meneruskan pesan tersebut ke *broker* yang lain. Hal ini dilakukan secara terus menerus secara bergantian. Hal ini dikarenakan *Load Balancer* menggunakan *Balance Mode Round Robin*.



Gambar 4.3 Diagram Alir Kerja Sistem di Sisi *Load Balancer*

4.6 Perancangan Pengujian

Perancangan pengujian bertujuan untuk mempersiapkan pengujian yang bertujuan untuk mengukur kemampuan sistem dalam menghadapi berbagai macam situasi yang ada. Kemudian perancangan pengujian ini juga dibuat untuk mengetahui performa sistem berdasarkan beberapa aspek yang ingin dicapai. Pengujian dilakukan menggunakan *script Python* yang akan berjalan memanfaatkan *multi threading* dan *library* Paho *MQTT*. Dari perancangan pengujian ini maka akan bisa dicapai poin – poin yang akan diukur. Semua pengujian dilakukan ketika *broker* menjalan *mosquitto* pada *foreground*. Karena dengan ini bisa dirasakan dampak penggunaan *resource CPU* yang ada.

Tabel 4.2 Tabel Rancangan Pengujian

No	Nama Pengujian	Tujuan
1	Pengujian waktu untuk melakukan penyebaran pesan antar *broker* menggunakan *bridge*	Mendapatkan nilai waktu yang dibutuhkan oleh *broker* yang menerima pesan dari *publisher* untuk dibagikan pesan ke *broker* lainnya yang terhubung dengan *bridge*. Sehingga didapatkan nilai waktu yang dibutuhkan *broker* untuk menyebarkan pesan.
2	Pengujian CPU *Usage* ketika melakukan penyebaran pesan antar *broker* menggunakan *bridge*	Mendapatkan nilai besar utilisasi CPU setiap *broker* ketika menyebarkan dan menerima pesan. Dimana pengujian ini akan lebih menitik beratkan pada *broker* yang berperan sebagai *bridge* dan harus meneruskan pesan ke semua *broker* yang ada.
3	Pengujian waktu yang dibutuhkan *broker* untuk menangani *request subscribe*	Mendapatkan nilai dari waktu yang dibutuhkan oleh *broker* untuk memproses dan menangani semua pesan *subscribe* yang ada. Pengujian ini dilakukan dengan 2 variabel pembanding, yaitu ketika sistem menggunakan *load balancer* dan tidak menggunakan *load balancer*. Dengan parameter uji yaitu jumlah *subscriber* dengan tujuan untuk mendapatkan waktu yang dibutuhkan setiap *broker* dan keseluruhan sistem untuk menangani *request* dari *subscriber*.
4	Pengujian CPU *Usage Broker* saat menangani *request subscribe*	Mendapatkan nilai *CPU Usage broker* saat menangani pesan *subscribe* yang ada ketika *subscriber*. Dilakukan perbandingan ketika *subscriber* melakukan *request* melalui *load balancer* dan ketika tidak melakukan *load balancer* dengan pembagian jumlah *request* yang merata.
5	Pengujian delay waktu ketika *broker* meneruskan pesan *publish* ke *subscriber*	Mendapatkan nilai waktu durasi *publish* ketika *broker* mengirimkan pesan yang sesuai dengan topik yang diikuti oleh *subscriber* ketika sistem menggunakan *load balancer* dan tidak menggunakan *load balancer*.
6	Pengujian untuk mengetahui *Resource* CPU *Broker* dalam	Mendapatkan nilai *CPU Usage* yang dibutuhkan *broker* ketika melakukan proses

	menangani banyak *client* secara pararel dalam melakukan proses *publish-subscribe*	*publish – subscribe*. Kemudian dari pengujian ini bisa dibandingkan *resource CPU* ketika jumlah *broker* bertambah dan beban kerja dibagi berdasarkan jumlah *broker* aktif.
7	Pengujian integritas data	Melakukan validasi integritas data yang dikirimkan oleh *publisher* ke *subscriber.*
8	Pengujian rekoneksi ketika *broker* mengalami kegagalan	Menguji kemampuan sistem dalam melakukan rekoneksi ketika terjadi kegagalan pada salah satu *broker* yang sedang terkoneksi dengan *subscriber*.