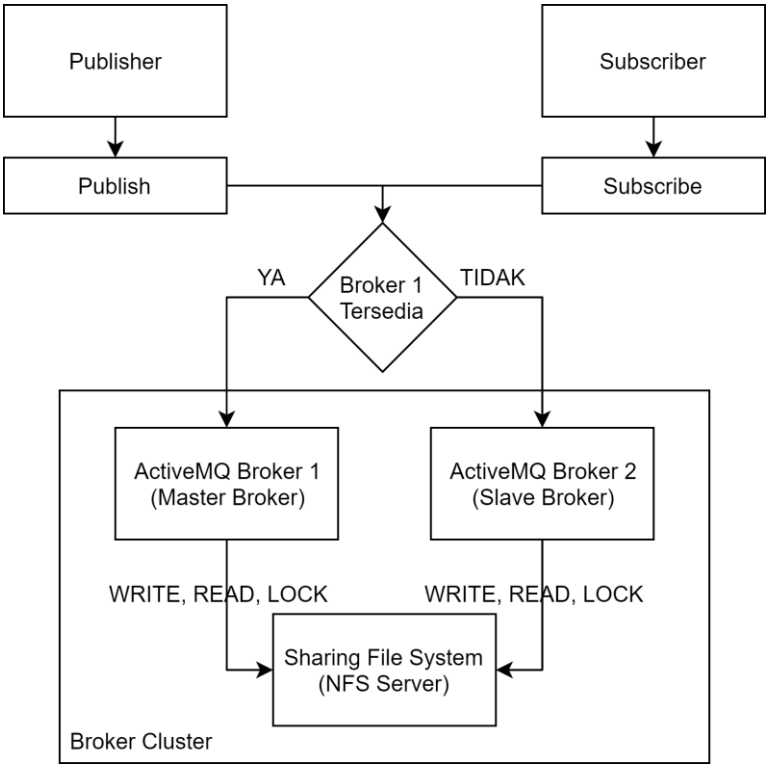
BAB 4 PERANCANGAN

Bab perancangan menjelaskan perancangan yang dilakukan untuk sistem secara detail dalam beberapa tahapan. Tahapan dimulai dari menjelaskan perancangan sistem yang dilanjutkan dengan perancangan 3 komponen utama yaitu MQTT *publisher, subscriber* dan *broker*.

4.1 Perancangan Sistem

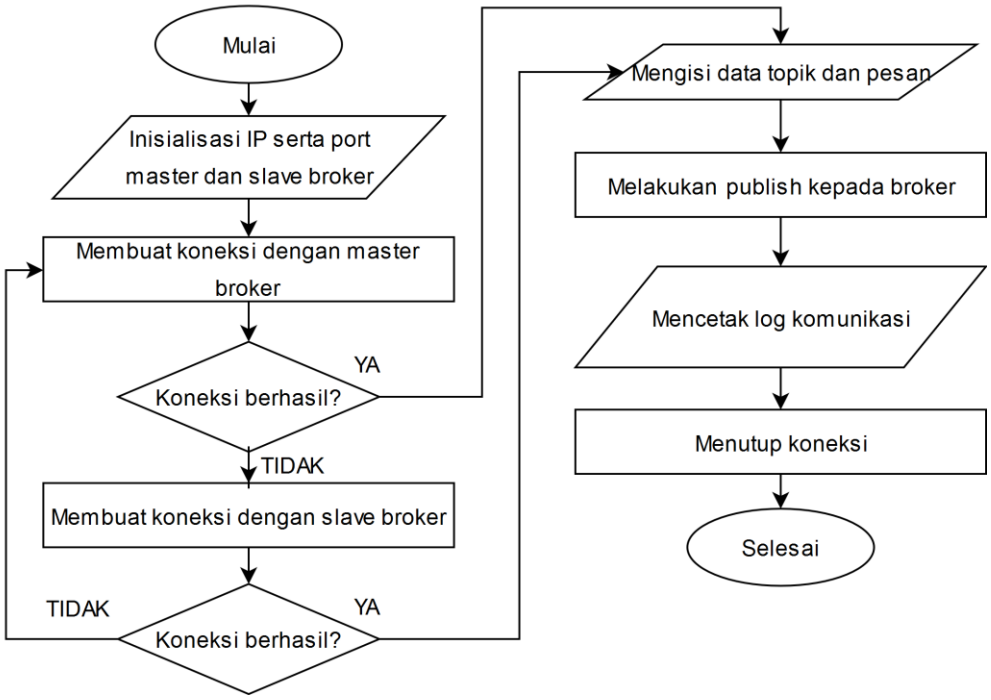


Gambar 4.1 Ilustrasi perancangan sistem

Sub bab perancangan sistem menjelaskan secara umum gambaran mengenai sistem dan komunikasi dari masing-masing komponen. Berdasarkan Gambar 4.1, perancangan sistem dilakukan dengan merancang 3 komponen utama yaitu MQTT *publisher*, *subscriber* dan *broker*. Perancangan MQTT *publisher* dilakukan dengan merancang perangkat lunak yang dapat membuat koneksi dengan salah satu MQTT *broker* yang berada dalam *broker cluster* dan melakukan *publish* pesan berbasis topik. Perancangan MQTT *subscriber* dilakukan untuk merancang perangkat lunak yang digunakan untuk proses komunikasi dengan *master* ataupun *slave broker* yang berada dalam *cluster* sehingga dapat melakukan *subscribe* pesan dan menerima pesan dari MQTT *broker*. Perangkat lunak MQTT *publisher* dan *subscriber* juga dirancang agar dapat memiliki mekanisme berpindah dari *master* kepada *slave broker* ketika terjadi kegagalan. Perancangan MQTT *broker* bertujuan untuk dapat mengimplementasikan *failover* dengan membangun *cluster* yang terdiri dari *master broker*, *slave broker* dan NFS *server*. Proses perancangan dilakukan dengan menggambarkan tahapan konfigurasi

yang perlu dilakukan. *Master broker* dirancang agar dapat menjadi *broker* utama dalam *cluster* dan memiliki cadangan *slave broker* yang berada dalam kondisi pasif menunggu *master broker* mengalami kegagalan. Komunikasi kedua broker dilakukan pada *Sharing File System* yang menggunakan mekanisme komunikasi dari *Network File System* (NFS). Perancangan MQTT *broker* juga mendefinisikan skema sistem ketika mengimplementasikan *failover* serta alur ketika melakukan proses komunikasi dengan MQTT *publisher* dan *subscriber*.

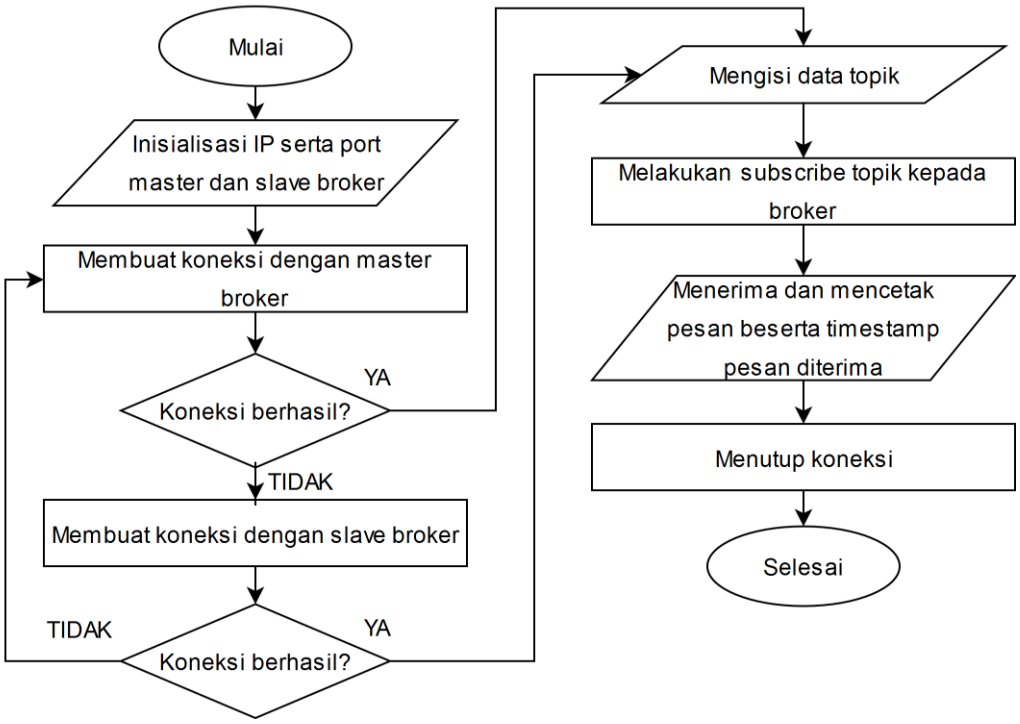
4.2 Perancangan MQTT *Publisher* dan *Subscriber*



Gambar 4.2 Perancangan MQTT *publisher*

Perancangan MQTT *publisher* dan *subscriber* digunakan sebagai dasar untuk membangun perangkat lunak yang sesuai dengan kebutuhan sistem. Gambar 4.2 dan 4.3 menjelaskan proses yang dilakukan MQTT *publisher* dan subscriber dalam sistem. Proses diawali dengan melakukan inisialisasi IP dan *port* dari *master* dan *slave broker*. Selanjutnya, MQTT *publisher* dan *subscriber* membuat koneksi dengan *master broker*. Apabila berhasil menjalin koneksi dengan *master broker*, MQTT *publisher* dan *subscriber* dapat melakukan proses komunikasi. Apabila tidak berhasil, MQTT *publisher* dan *subscriber* mencoba menjalin koneksi dengan *slave broker*. Pada Gambar 4.2, MQTT *publisher* dapat melakukan *publish* pesan setelah berhasil terhubung dengan MQTT *broker*. MQTT *publisher* memasukkan data topik dan pesan yang akan dikirimkan dalam bentuk *string* kepada MQTT *broker*. Pada Gambar 4.3, MQTT *subscriber* melakukan *subscribe* pada suatu topik dan siap menerima pesan dari MQTT *broker* sesuai dengan topik yang sudah di *subscribe* ketika telah berhasil terhubung dengan MQTT broker. Ketika ada pesan dari MQTT *broker*, MQTT *subscriber* menerima dan

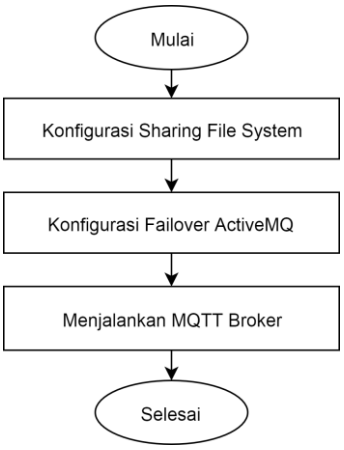
mencetak pesan. MQTT *publisher* dan *subscriber* menutup koneksi dengan MQTT *broker* ketika telah selesai melakukan proses komunikasi.



Gambar 4.3 Perancangan MQTT *subscriber*

4.3 Perancangan MQTT *Broker*

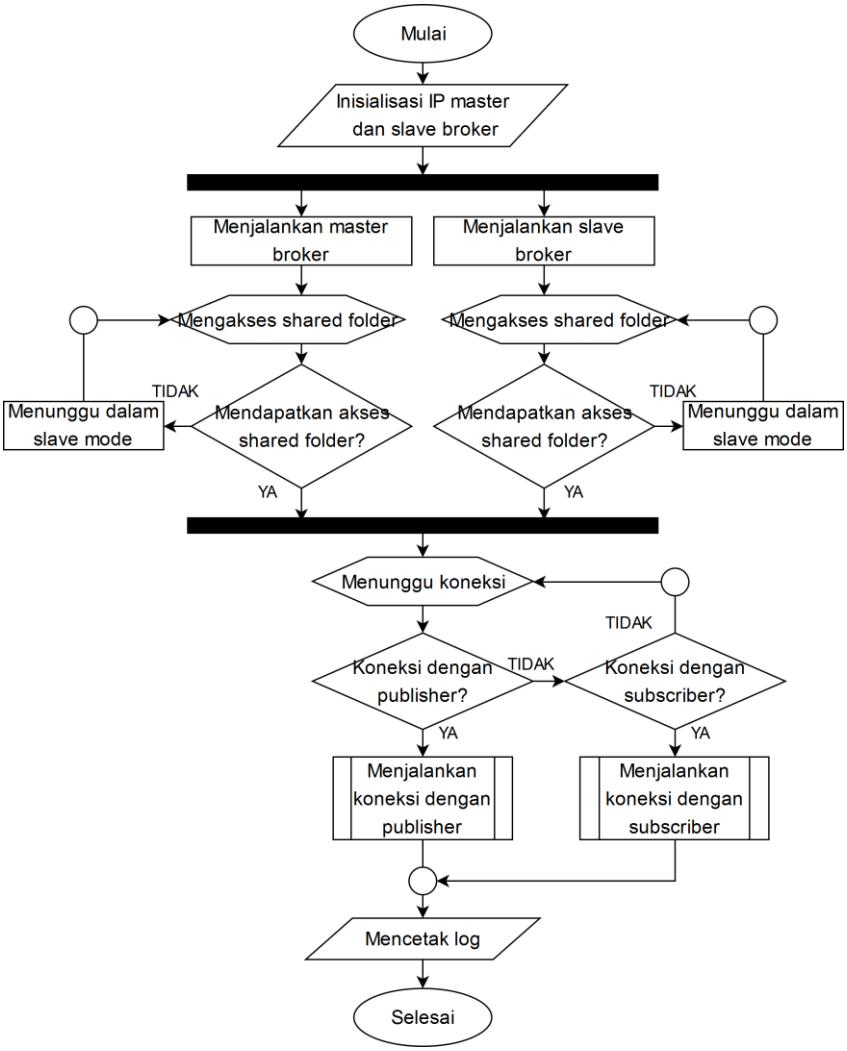
Perancangan MQTT *broker* dilakukan untuk mendefinisikan proses yang dijalankan agar dapat memenuhi kebutuhan sistem. Kebutuhan sistem MQTT *broker* adalah MQTT *broker* dapat membentuk *broker cluster* yang terdiri dari *master* dan *slave broker* serta mengimplementasikan *failover*. MQTT *broker* juga harus dapat menjadi jembatan komunikasi antara MQTT *publisher* dengan MQTT *subscriber*. Perancangan MQTT *broker* ditunjukkan pada Gambar 4.4.



Gambar 4.4 Perancangan MQTT *broker*

Proses konfigurasi *Sharing File System* dilakukan untuk membangun redundansi data pada *broker cluster* dan menjadi media komunikasi dari kedua MQTT *broker* untuk melakukan *fault detection*. *Konfigurasi Sharing File System* menggunakan *tools Network File System* (NFS) karena adanya fitur *file locking* yang digunakan untuk menentukan broker yang menjadi *master* dan *slave*. Selanjutnya, MQTT *broker* melalui tahapan untuk mengkonfigurasikan *failover broker ActiveMQ*. Konfigurasi *ActiveMQ* dilakukan dengan membangun *broker cluster* yang memanfaatkan *Sharing File System* yang telah dibuat sebelumnya. Tahapan konfigurasi *ActiveMQ* juga melakukan konfigurasi agar MQTT *broker* dapat menjembatani komunikasi antara MQTT *publisher* dan *subscriber*. MQTT *broker* yang telah dikonfigurasikan akan dijalankan pada tahapan selanjutnya dan digambarkan skema *broker cluster*, *failover* dan komunikasi berdasarkan konfigurasi.

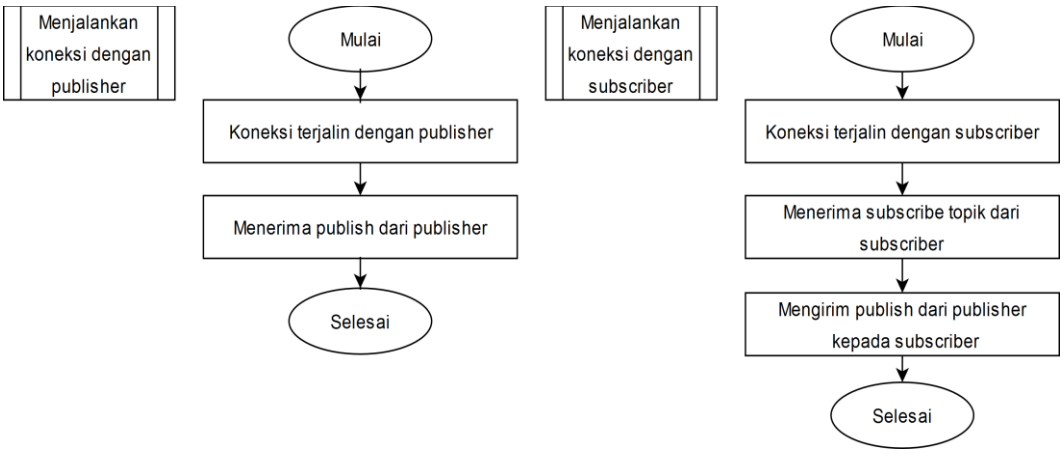
4.3.1 Menjalankan MQTT *Broker*



Gambar 4.5 Diagram alir MQTT *broker*

Gambar 4.6 menjelaskan proses yang dilakukan oleh *broker cluster master* dan slave *broker* dalam sistem. Proses diawali dengan menjalankan *service* dari *master* dan *slave broker* pada *Raspberry*. *Master* dan slave *broker* melakukan perulangan untuk mengakses direktori *Sharing File System* yang sudah ditentukan lokasinya. MQTT *broker* yang mendapatkan akses *Sharing File System* akan menjadi *master broker* dan siap menerima koneksi. Apabila tidak berhasil mendapatkan akses dari *Sharing File System*, maka MQTT *broker* harus menunggu dalam *slave mode* dan melakukan perulangan untuk mendapatkan akses dari *Sharing File System*. *Master broker* yang sudah aktif dapat menerima koneksi dari MQTT *publisher* atau *subscriber*. Ketika ada permintaan koneksi dari MQTT *publisher*, maka MQTT *broker* akan menjalankan sub proses dengan MQTT *publisher*. Ketika ada permintaan koneksi dari MQTT *subscriber*, maka MQTT *broker* menjalankan sub proses dengan MQTT *subscriber*. Apabila tidak ada permintaan koneksi dengan MQTT *publisher* atau MQTT *subscriber*, maka *master broker* tetap aktif dan siap menerima koneksi.

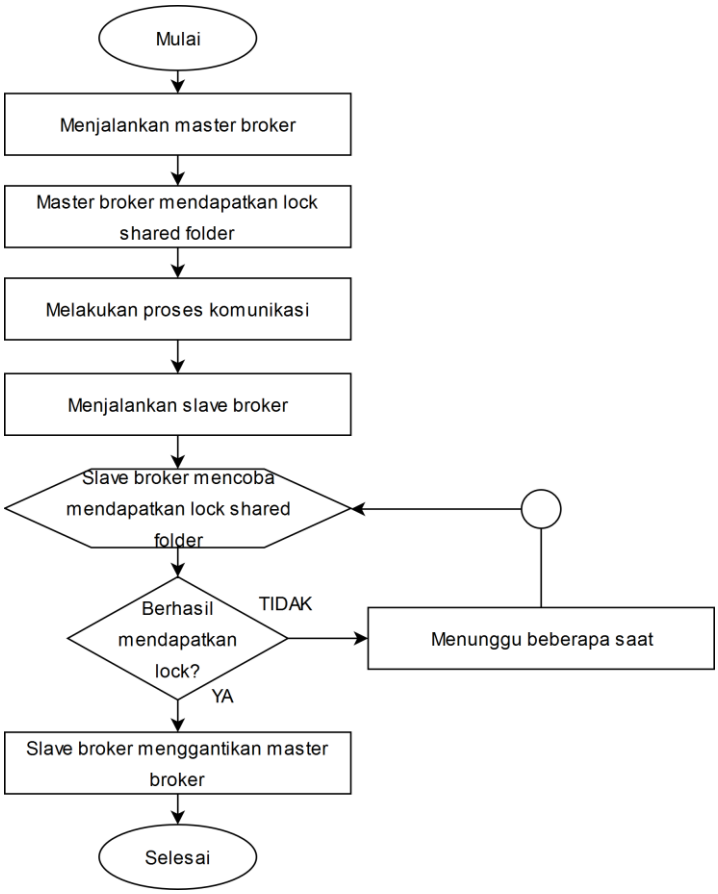
Gambar 4.7 menggambarkan sub proses dari MQTT *broker* ketika melakukan komunikasi dengan MQTT *publisher* dan *subscriber*. Ketika ada permintaan koneksi dari MQTT *publisher*, *master broker* menjalin koneksi dengan MQTT *publisher*. Selanjutnya, MQTT *broker* menerima pesan yang di *publish* dan menyimpannya untuk dapat dikirimkan kepada MQTT *subscriber*. Ketika ada koneksi dari MQTT *subscriber*, *master broker* menjalin koneksi dengan MQTT *subscriber*. Selanjutnya, MQTT *broker* menerima *request* topik dari MQTT *subscriber*. Ketika MQTT *publisher* mengirimkan pesan pada topik dari MQTT *subscriber*, MQTT *broker* mengirimkan pesan tersebut kepada MQTT *subscriber*.



Gambar 4.6 Diagram alir komunikasi MQTT *broker*

Gambar 4.8 menunjukkan diagram alir ketika mengimplementasikan *failover*. Proses *failover* diawali dengan *master broker* yang terlebih dahulu dijalankan agar *master broker* mendapatkan *lock* direktori *Sharing File System*. Selanjutnya, *master broker* siap menerima koneksi dari MQTT *publisher* dan *subscriber*. Kemudian, *slave broker* mulai dijalankan dan memasukki *slave mode* untuk menunggu *master broker* tidak aktif. *Slave broker* akan mencoba mengakses

direktori *Sharing File System* secara berulang-ulang untuk mengetahui kondisi dari *master broker*. Ketika *slave broker* tidak berhasil mendapatkan *lock*, *slave broker* menunggu beberapa saat untuk mencoba mengakses direktori *lock* kembali. Ketika *service master broker* terhenti, *master broker* akan melepas *lock* pada direktori *Sharing File System* sehingga *slave broker* mengetahui bahwa telah terjadi kendala pada *master broker* dan mendapatkan *lock* dari direktori *Sharing File System*. Pada saat itu *slave broker* akan mengimplementasikan *failover* dan menggantikan *master broker* yang terhenti.



Gambar 4.7 Diagram alir *failover*