BAB 6 PENGUJIAN

Bab pengujian menjelaskan secara detail tentang pengujian dari sistem yang telah diimplementasikan. Pengujian didasarkan pada kebutuhan, perancangan dan implementasi sistem yang sudah dilakukan. Proses pengujian dilakukan untuk menguji sistem yang diimplementasikan dengan skenario yang telah didefinisikan. Skenario pengujian digunakan untuk menguji nilai parameter yang telah didefinisikan untuk menilai keberhasilan implementasi sistem. Parameter uji yang didefinisikan meliputi pengujian *availability, latency* dan *packet loss.* Pengujian dilakukan para jaringan lokal dan perangkat keras dari MQTT *publisher*, *subscriber* dan *broker* terhubung dengan *access point* secara *wireless*. Hasil pengujian dari sistem yang diimplementasikan akan dibandingkan dengan sistem protokol MQTT pada umumnya yang menggunakan *Mosquitto*. Perbandingan dilakukan untuk mengetahui sejauh mana performa sistem yang diimplementasikan dengan *Mosquitto* sebagai tolak ukur.

6.1 Pengujian *Availability*

6.1.1 Skenario Pengujian

Pengujian ini dilakukan untuk menguji nilai parameter uji *availability* berdasarkan *downtime* dari sistem yang diimplementasikan. Nilai *downtime* diharapkan menjadi gambaran *availability* sistem dengan menunujukkan seberapa lama layanan MQTT *broker* pada penelitian ini terhenti ketika mengalami kegagalan dan melakukan *failover*. Pengukuran *downtime* dilakukan karena keterbatasan waktu dalam pengujian dimana pengukuran *availability* harus dilakukan dalam waktu yang lama seperti 1 tahun pengujian. Pengujian dilakukan dengan metode *planned downtime* ketika MQTT *publisher* mengirimkan pesan kepada MQTT *subscriber* secara berulang-ulang. *Master broker* akan dinonaktifkan ditengah-tengah proses komunikasi. Pengujian melihat nilai *downtime* dalam detik ketika *master broker* berpindah kepada *slave broker*. Pengujian dilakukan sebanyak lima kali percobaan. Hasil pengujian ditunjukkan dalam Tabel 6.1.

Tabel 6.1 Pengujian *availability*

Pengujian Ke -	*Down Timestamp*	*Up Timestamp*	*Downtime* (s)
1	1507743408477	1507743432563	24.086
2	1507743757038	1507743781140	24.102
3	1507743880882	1507743901818	20.936
4	1507743977507	1507744003949	26.442
5	1507744079618	1507744105685	26.067
Rata-Rata			24.3266

6.1.2 Pembahasan

Tabel 6.1 menunjukkan nilai rata-rata *downtime* adalah 24.3266 detik dari 5 percobaan. Implementasi *failover* terbukti dapat meminimalisir *downtime* ketika terjadi kegagalan pada *master broker*. Hal ini menandakan protokol MQTT dapat mengadopsi *failover* dan dapat mengurangi resiko berhenti nya *service* atau *downtime* yang lama. *Downtime* yang terjadi bisa dianggap cepat dan memiliki potensi untuk memenuhi batasan sistem yang *High-Availability*. Sistem dikatakan *High-Availability* apabila memiliki persentase *availability* sebesar 99.999% dan mengalami *downtime* 5 menit dalam setahun (Thulin, 2004). Apabila *downtime* yang terjadi berkisar 20 detik, maka sistem dapat mentolerir *down* sebanyak 12 kali dalam setahun. Nilai *downtime* yang didapatkan merupakan waktu yang dibutuhkan *slave broker* untuk mendeteksi kegagalan pada *master broker* dan menjalankan *ActiveMQ*. *Slave broker* mendeteksi kegagagalan atau *failure detection* dipengaruhi oleh *cost* dalam hal *processing delay* dari sistem. Karena untuk mendeteksi kegagalan pada *master broker*, *slave broker* mengirimkan *packet* NFSV4 *Call* kepada *master broker* untuk mendapatkan *lock* pada *sharing file system*. Sementara lama waktu yang dibutuhkan *slave broker* menjalankan *ActiveMQ* dipengaruhi nilai *overhead* ketika mengimplementasikan sistem. Nilai *overhead* didefinisikan sebagai penggunaan *resource* ketika mengimplementasikan sistem dalam hal penggunaan CPU dan *memory*. Data penggunaan CPU dan *memory* ketika mengimplementasikan sistem ditunjukkan pada Tabel 6.2.

Tabel 6.2 Data penggunaan CPU dan *memory*

Jumlah MQTT *Publisher*	*Memory Usage Mosquitto* (MB)	CPU *Usage Mosquitto*	*Memory Usage ActiveMQ* (MB)	CPU *Usage ActiveMQ*
20	3.34	0.43%	101.99	24.73%
40	3.34	0.71%	102.23	50.05%
60	3.34	1.00%	102.31	65.80%
80	3.34	2.43%	103.56	88.88%
100	3.34	3.43%	106.24	109.72%

Tabel 6.3 Data penggunaan CPU dan *memory* dalam kondisi *idle*

Jumlah MQTT *Publisher*	*Memory Usage Mosquitto* (MB)	CPU *Usage Mosquitto*	*Memory Usage ActiveMQ* (MB)	CPU *Usage ActiveMQ*
20	3.34	0.00%	101.73	0.67%
40	3.34	0.33%	102.12	0.83%
60	3.34	0.33%	102.14	1.00%
80	3.34	0.50%	103.40	1.17%
100	3.34	0.50%	105.95	2.00%

Hasil Tabel 6.2 didapatkan ketika melakukan proses komunikasi antara MQTT *publisher* dan *subscriber* pada sistem yang diimplementasikan. *ActiveMQ* menggunakan CPU dan *memory* yang lebih besar ketika menjalankan proses komunikasi bila dibandingkan dengan *Mosquitto*. Tabel 6.3 menunjukkan nilai *overhead* ketika dalam kondisi *idle*. Nilai overhead ketika idle didapatkan setelah MQTT publisher dan subscriber selesai melakukan komunikasi. Dalam kondisi idle, penggunaan CPU dan *memory ActiveMQ* masih lebih besar dibandingkan dengan *Mosquitto*. Hal itu disebabkan karena adanya komunikasi antara *master* dan *slave broker* meskipun dalam kondisi *idle* atau tidak menangani komunikasi dari MQTT *publisher* dan *subscriber*. Penggunaan resource *ActiveMQ* mempengaruhi nilai *downtime* ketika terjadi kegagalan pada *master broker*. Hal itu disebabkan karena *slave broker* membutuhkan waktu yang lebih banyak untuk menjalankan perangkat lunak *ActiveMQ* yang menggunakan CPU dan *memory* yang besar.

6.2 Pengujian *Latency*

6.2.1 Skenario Pengujian

Pengujian ini dilakukan untuk menguji nilai parameter *latency* pada sistem yang diimplementasikan. Nilai *latency* diharapkan dapat menjadi gambaran terkait *cost* dalam hal *processing delay* pada *availability* sistem. Pengujian dilakukan dengan menggunakan *thread* sebanyak 20, 40, 60, 80 dan 100 ketika melakukan komunikasi antara MQTT *publisher* menuju MQTT *subscriber*. Pengukuran *latency* dilakukan dengan menghitung *timestamp* ketika pesan diterima MQTT *subscriber* dikurangi dengan *timestamp* ketika pesan terkirim dari MQTT *publisher*. Perangkat keras MQTT *publisher* dan *subscriber* disinkronisasi menggunakan *Network Time Protocol* (NTP) sehingga kedua perangkat keras yang digunakan memiliki waktu yang sama. Hasil pengujian ditunjukkan dalam Tabel 6.4.

Tabel 6.4 Pengujian *latency* MQTT *publisher*

Pengujian Ke-	Jumlah MQTT *Publisher*	*Latency Mosquitto* (s)	*Latency ActiveMQ* (s)
1	20	0.2027	0.37
2	40	0.3099	0.8253
3	60	0.5639	1.4087
4	80	0.8145	1.5114
5	100	1.5096	1.7662
Rata-Rata (s)		0.6801	1.1763

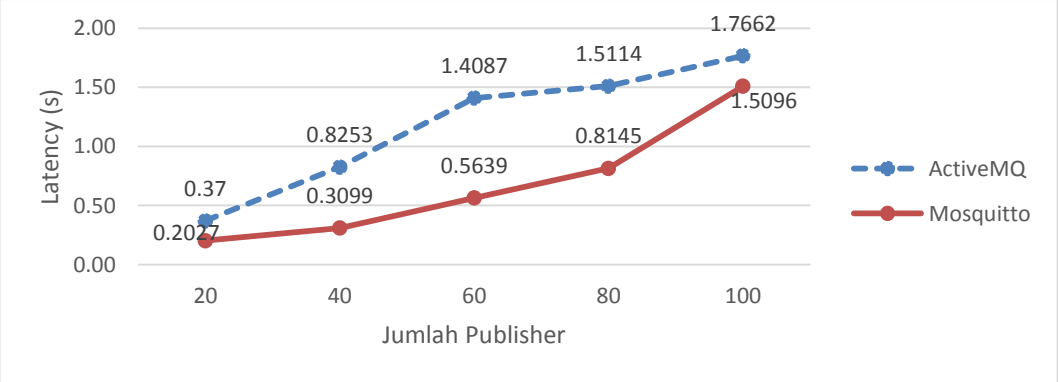
Pengujian *latency* juga dilakukan dengan jumlah *thread* yang sama ketika digunakan pada sisi MQTT *subscriber*. Hasil pengujian ditunjukkan dalam Tabel 6.5.

Tabel 6.5 Pengujian *latency* MQTT *subscriber*

Pengujian Ke-	Jumlah MQTT *Subscriber*	*Latency Mosquitto* (s)	*Latency ActiveMQ* (s)
1	20	0.031	0.048
2	40	0.0642	0.0926
3	60	0.0783	0.1189
4	80	0.1099	0.1450
5	100	0.1374	0.1739
Rata-Rata (s)		0.0842	0.1157

6.2.2 Pembahasan

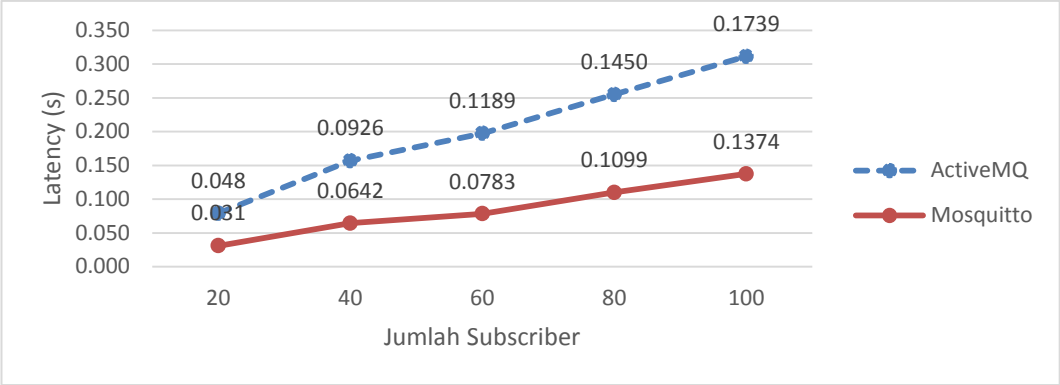
Gambar 6.1 menunjukkan hasil pengujian nilai parameter *latency* pada sisi MQTT *publisher*. *Mosquitto* mendapatkan hasil pengujian *latency* yaitu 0.2027 s, 0.3099 s, 0.5639 s, 0.8145 s dan 1.5096 s. Sementara *ActiveMQ* memiliki hasil 0.37 s, 0.8253 s, 1.4087 s, 1.5114 s dan 1.7662 s.



Gambar 6.1 Hasil pengujian *latency* MQTT *publisher*

Berdasarkan Gambar 6.2, hasil pengujian *Mosquitto* pada sisi MQTT *subscriber* didapatkan sebesar 0.031 s, 0.0642 s, 0.0783 s, 0.1099 s dan 0.1374 s. Sementara hasil dari *ActiveMQ* didapatkan nilai 0.048 s, 0.0926 s, 0.1189 s, 0.1450 s dan 0.1739 s.

Jumlah *thread* mempengaruhi nilai *latency* yang didapatkan. Semakin banyak penggunaan *thread* mengakibatkan semakin banyak koneksi untuk melakukan *publish* atau *subscribe*. Jumlah koneksi yang banyak menyebabkan jaringan semakin terbebani. Hasil yang didapatkan menunjukkan semakin banyak jumlah *thread* menyebabkan peningkatan nilai *latency*. Hasil *ActiveMQ* didapatkan lebih besar dibandingkan *Mosquitto*. Hal itu disebabkan karena perangkat keras MQTT *broker* yaitu Raspberry Pi 3 Model B memiliki *resource* yang terbatas. Sementara *ActiveMQ* menggunakan *resource* yang besar dalam hal penggunaan CPU dan *memory* seperti ditunjukkan pada Tabel 6.2. Ketidaksesuaian *overhead ActiveMQ* dengan kemampuan perangkat keras menyebabkan terjadinya penurunan kecepatan ketika melakukan proses komunikasi.



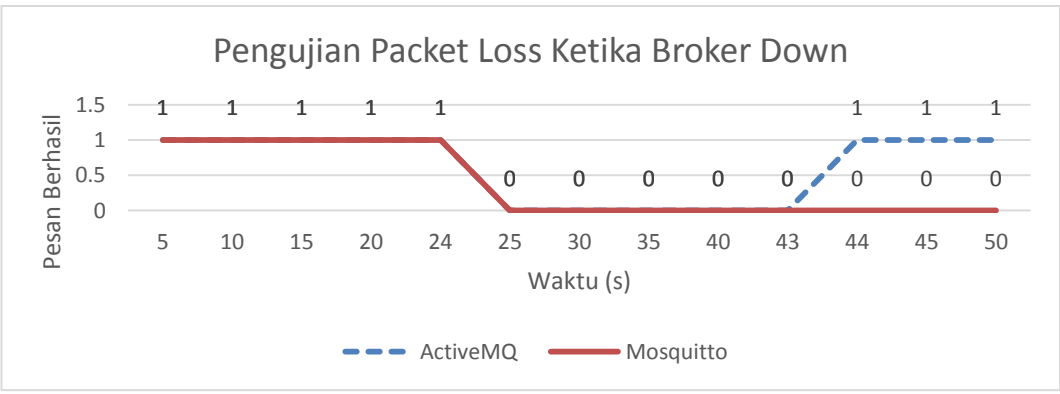
Gambar 6.2 Hasil pengujian *latency* MQTT *subscriber*

6.3 Pengujian *Packet Loss* Ketika MQTT *Broker Down*

6.3.1 Skenario Pengujian

Pengujian ini dilakukan untuk menguji nilai parameter *packet loss* ketika MQTT *broker down* pada sistem yang diimplementasikan. Pengujian dilakukan dengan mengirimkan pesan dari MQTT *publisher* kepada MQTT *subscriber* setiap detiknya selama 50 detik. Ditengah proses komunikasi, MQTT *broker* akan dimatikan. Pengujian melihat pesan yang berhasil diterima pada *ActiveMQ* dan *Mosquitto* setiap detiknya ketika berkomunikasi dan terjadi *down* pada MQTT *broker*.

6.3.2 Pembahasan



Gambar 6.3 Hasil pengujian *packet loss*

Gambar 6.3 menunjukkan jumlah pesan yang berhasil dari detik 1 hingga 50 pada *ActiveMQ* dan *Mosquitto*. Ketika MQTT *broker* dihentikan pada detik 24, Pengiriman pesan pada *Mosquitto* terhenti karena MQTT *publisher* kehilangan koneksi dengan MQTT *broker* dan tidak bisa melakukan *publish*. Sehingga jumlah pesan berhasil pada *Mosquito* mulai detik 25 hingga detik 50 terhenti. Sementara *ActiveMQ* menggunakan *failover* dan berpindah kepada *slave broker* ketika *master broker* terhenti. Sehingga proses pengiriman pesan pada *ActiveMQ* terjadi packet loss pada detik 25 hingga *slave broker* tersedia pada detik 44. Dapat diartikan bahwa pesan yang hilang berjumlah 20 pesan dan menunjukkan

downtime yang terjadi adalah selama 20 detik. MQTT *publisher* dapat melanjutkan proses *publish* ketika *slave broker* telah menggantikan *master broker*. Sementara pada sistem satu MQTT *broker* seperti *Mosquitto*, proses komunikasi tidak dapat dilanjutkan ketika MQTT *broker* terhenti. MQTT *publisher* dan *subscriber* tidak dapat tersambung pada MQTT *broker*. Sehingga jumlah pesan berhasil dari *Mosquitto* dimulai detik 24 hingga 50 berjumlah 0.