BAB V
KESIMPULAN, IMPLIKASI, DAN REKOMENDASI

5.1 Kesimpulan

Berdasarkan hasil temuan yang sudah di jelaskan di bab sebelumnya, maka dapat disimpulkan bahwa

1. Dari hasil pengujian hipotesis dihasilkan bahwa terdapat perbedaan yang signifikan antara performa uWebSocket dan WS dalam aspek *Time Behavior*. uWebSocket unggul dibandingkan WS dalam berbagai aspek seperti *Time Behavior*, *Capacity*, dan *Reliability*. uWebSocket memiliki waktu koneksi yang lebih cepat sebesar 7.74 ms dibandingkan WS dengan 550.97ms dan durasi sesi yang lebih cepat sebesar 0.01054 detik dibandingkan WS dengan 0.55384 detik pada 3000 pengguna virtual. Sementara WS menunjukkan performa yang lebih baik dalam durasi pengiriman pesan dengan 0.94721 detik, sementara uWebSocket dengan durasi pengiriman pesan 2.514121 detik pada 3000 pengguna virtual. uWebSocket.js juga lebih efisien dalam mengirim dan menerima pesan sebanyak 891.550 pesan dibandingkan WS yang mengirim dan menerima 582.389. uWebSocket mampu menginisiasi sesi WebSocket dengan 891.551 sesi, sementara WS hanya mampu mencatat 582.389 sesi. Selain itu, uWebSocket.js mempertahankan tingkat kesuksesan yang lebih stabil pada status "101" (protokol switch) dengan hampir 100% pada setiap pengujian.
2. Dalam hal *Resource Utilization*, uWebSocket.js menunjukkan penggunaan CPU yang lebih efisien dibandingkan dengan WS pada semua tingkat beban. uWebSocket.js menggunakan CPU secara lebih rendah dan konsisten sebesar 12.8% sementara WS sebesar 22% pada 50 pengguna virtual. meskipun jumlah pengguna virtual meningkat. Sebaliknya, WS menunjukkan penggunaan CPU yang lebih tinggi secara konsisten, dengan perbedaan yang semakin jelas seiring dengan peningkatan jumlah pengguna virtual, mengindikasikan efisiensi yang lebih rendah dalam penggunaan sumber daya.

5.2 Implikasi

Berdasarkan hasil perbandingan performa antara WS dan uWebSocket.js, menunjukkan bahwa uWebSocket.js sesuai untuk aplikasi dengan beban tinggi dan efisiensi, Dengan waktu koneksi dan durasi sesi yang lebih cepat serta kemampuan untuk mengirim dan menerima pesan secara lebih efisien, uWebSocket.js sangat cocok untuk aplikasi perpesanan yang menghadapi beban tinggi dan memerlukan pengelolaan sumber daya yang optimal. Keunggulan uWebSocket.js dalam hal Time Behavior dan Capacity menjadikannya pilihan ideal untuk aplikasi yang memerlukan latensi rendah dan throughput tinggi. WS menunjukkan performa yang lebih baik dalam durasi pengiriman pesan pada beban rendah, sehingga lebih sesuai untuk aplikasi dengan volume pesan yang tidak terlalu tinggi. Meskipun WS mengalami penurunan performa pada beban tinggi, jika aplikasi lebih fokus pada komunikasi yang efisien pada skala kecil hingga menengah, WS bisa menjadi pilihan yang memadai. Pemilihan antara WS dan uWebSocket.js harus didasarkan pada kebutuhan spesifik aplikasi. Jika efisiensi dalam pengelolaan sumber daya dan keandalan pada beban tinggi adalah prioritas, maka uWebSocket.js adalah pilihan yang lebih baik. Namun, jika aplikasi lebih fokus pada komunikasi pada skala kecil hingga menengah dengan durasi pengiriman pesan yang lebih baik pada beban rendah, WS bisa menjadi alternatif yang sesuai.

5.3 Rekomendasi

Berikut adalah beberapa saran dan rekomendasi berdasarkan proses dan hasil penelitian untuk studi mendatang.

1. Sebaiknya, pengujian dilakukan dengan skenario yang lebih terbatas namun dilakukan beberapa kali untuk setiap skenario agar mendapatkan hasil yang lebih konsisten dan akurat.
2. Pengujian sebaiknya dilakukan menggunakan server *cloud* atau server lokal yang baru dan bersih dari aplikasi yang berjalan di latar belakang untuk memastikan kondisi pengujian yang optimal.