**PENGEMBANGAN WEB (TEORI)**

**LAPORAN EKSPERIMEN CHOOSING BETWEEN LONG POLLING VS WEBSOCKETS VS SERVER-SENT EVENTS FOR REAL-TIME COMMUNICATION**

*Laporan ini disusun untuk memenuhi tugas 4 mata kuliah Pengembangan Web (Teori)*



Disusun oleh kelompok B4:

Asri Husnul Rosadi 221524035

Faris Abulkhoir 221524040

Mahardika Pratama 221524044

Muhamad Fahri Yuwan 221524047

Najib Alimudin Fajri 221524053

Sarah 221524059

Septyana Agustina 221524060

Dosen Pengampu:

Joe Lian Min, M.Eng.

**JURUSAN TEKNIK KOMPUTER DAN INFORMATIKA**

**PROGRAM STUDI D4 TEKNIK INFORMATIKA**

**POLITEKNIK NEGERI BANDUNG**

**2024**

# **DAFTAR ISI**

[DAFTAR ISI i](#_Toc177290826)

[A. IDENTIFIKASI PROBLEM 1](#_Toc177290827)

[B. DESKRIPSI PROBLEM 1](#_Toc177290828)

[C. METODOLOGI EKSPERIMEN 2](#_Toc177290829)

[D. PELAKSANAAN EKSPERIMEN 3](#_Toc177290830)

[E. ANALISIS HASIL EKSPERIMEN 4](#_Toc177290831)

# **IDENTIFIKASI PROBLEM**

Real-time communication (komunikasi waktu nyata) semakin penting dalam pengembangan aplikasi web modern, khususnya untuk aplikasi yang memerlukan pembaruan data secara langsung, seperti notifikasi, chat, atau monitoring data. Ada beberapa teknik yang umum digunakan untuk mengimplementasikan komunikasi real-time, termasuk Long Polling, Server-Sent Events (SSE), dan WebSockets. Masing-masing teknik ini memiliki kelebihan dan kekurangan dalam hal performa, efisiensi, dan kompleksitas. Tujuan eksperimen ini adalah untuk mengevaluasi efektivitas masing-masing metode dalam skenario real-time communication, membandingkan bagaimana mereka menangani frekuensi pembaruan data secara berkala.

# **DESKRIPSI PROBLEM**

Eksperimen ini mengevaluasi tiga teknik komunikasi real-time:

1. **Long Polling**: Teknik polling di mana klien melakukan permintaan HTTP ke server, dan server menunda respons hingga data tersedia. Setelah data dikirim, klien segera melakukan permintaan ulang.
2. **Server-Sent Events (SSE)**: Server mengirimkan data secara langsung ke klien melalui koneksi HTTP yang terbuka. Klien menerima pembaruan secara terus-menerus tanpa perlu membuat permintaan baru.
3. **WebSockets**: WebSockets menyediakan koneksi dua arah yang terbuka antara klien dan server, memungkinkan data dikirim dan diterima secara real-time tanpa perlu membuat permintaan HTTP berulang.

Metode ini dibandingkan dalam hal:

* Latency (Waktu Tunda): Seberapa cepat data sampai ke klien.
* Efisiensi Koneksi: Penggunaan bandwidth dan jumlah permintaan yang diperlukan.
* Skalabilitas: Seberapa baik teknik ini bekerja saat jumlah klien meningkat.

# **METODOLOGI EKSPERIMEN**

**Variabel Terikat:**

* Waktu respons data yang diperbarui (latency).
* Efisiensi dalam menggunakan bandwidth dan resource server.

**Variabel Bebas:** Teknik komunikasi yang digunakan: Long Polling, SSE, WebSockets.

**Metode Pengukuran:**

* **Latency**: Diukur berdasarkan waktu antara data yang diperbarui di server hingga diterima di klien.
* **Efisiensi Koneksi**: Dilihat dari jumlah permintaan HTTP (untuk Long Polling) atau ukuran pesan data (untuk SSE dan WebSockets).

**Skenario Pengujian:** Pengujian dilakukan dengan pembaruan data setiap 5 detik di server, dan klien menerima pembaruan tersebut menggunakan masing-masing teknik.

**Alat dan Teknologi:**

* + **Express.js**: Untuk menyajikan server yang mendukung berbagai teknik real-time communication, yaitu Long Polling, Server-Sent Events (SSE), dan WebSockets.
  + **Node.js**: Untuk mengembangkan server backend yang mendukung komunikasi real-time.
  + **HTML dan JavaScript**: Untuk implementasi klien yang terhubung ke server menggunakan masing-masing teknik komunikasi real-time.
  + **WebSocket Protocol**: Untuk implementasi komunikasi dua arah yang efisien antara klien dan server.
  + **Browser dengan Developer Tools**: Untuk memonitor permintaan dan respons HTTP, serta memverifikasi penggunaan WebSocket dan Server-Sent Events (SSE).

**Langkah-langkah Eksperimen:**

1. **Setup Lingkungan:**

* Kembangkan server Express.js yang menyediakan tiga endpoint berbeda: satu untuk Long Polling, satu untuk SSE, dan satu lagi untuk WebSockets. Setiap endpoint mengirimkan pembaruan data sederhana setiap 5 detik.
* Buat halaman HTML sederhana dengan tiga tombol untuk menghubungkan ke server menggunakan Long Polling, SSE, dan WebSockets.

1. **Implementasi Long Polling, SSE, dan WebSockets:**

* **Long Polling:** Klien mengirim permintaan HTTP ke server, dan server menahan respons hingga data baru tersedia. Setelah respons diterima, klien segera mengirim permintaan baru.
* **SSE:** Server mengirimkan data secara otomatis ke klien melalui koneksi HTTP yang terbuka.
* **WebSockets:** Menerapkan koneksi dua arah yang terbuka di antara klien dan server, memungkinkan data dikirim dan diterima secara real-time.

1. **Pengujian Koneksi dan Latency:**

* Uji setiap metode dengan mengukur waktu tunda (latency) dari saat data dikirim oleh server hingga diterima oleh klien.
* Gunakan Developer Tools pada browser untuk memantau jumlah permintaan HTTP, melihat detail aliran data pada setiap metode, serta mencatat ukuran data yang dikirim dan respons yang diterima.
* Pantau perbedaan penggunaan bandwidth dan jumlah koneksi yang dibuka dengan setiap metode.

1. **Simulasi Beban dengan Banyak Klien:**

* Jalankan beberapa klien secara bersamaan untuk melihat bagaimana masing-masing teknik menangani peningkatan jumlah klien yang memerlukan pembaruan real-time.
* Bandingkan performa server dan klien saat menggunakan Long Polling, SSE, dan WebSockets di bawah beban tinggi.

# **PELAKSANAAN EKSPERIMEN**

**Tahap 1: Setup Lingkungan**

* 1. **Aplikasi Server:** Aplikasi Express.js diatur untuk menangani Long Polling, SSE, dan WebSockets. Pembaruan data dilakukan setiap 5 detik, yang akan dikirimkan ke klien menggunakan tiga metode yang berbeda.
  2. **Aplikasi Klien:** HTML sederhana dengan JavaScript yang terhubung ke server untuk masing-masing metode real-time communication. Klien akan menampilkan data terbaru yang diterima di layar.

**Tahap 2: Pengujian Kinerja**

1. **Long Polling:** Klien mengirim permintaan HTTP setiap 5 detik, server merespons setelah data baru tersedia.
2. **Server-Sent Events (SSE):** Server mengirim data baru ke klien melalui koneksi HTTP yang terbuka setiap 5 detik.
3. **WebSockets:** WebSockets tetap terhubung sepanjang waktu, dan server mengirimkan pembaruan data secara langsung ke klien setiap 5 detik.

# **ANALISIS HASIL EKSPERIMEN**

**Hasil Eksperimen:**

* 1. **Long Polling:**
* **Latency**: Terdapat sedikit waktu tunda karena klien harus terus menerus membuka koneksi baru setelah menerima pembaruan. Pada koneksi lambat atau tidak stabil, waktu tunda bisa lebih besar.
* **Efisiensi Koneksi**: Tidak efisien karena setiap pembaruan memerlukan pembukaan dan penutupan koneksi HTTP baru. Ini meningkatkan overhead, terutama saat ada banyak klien.
* **Skalabilitas**: Beban pada server meningkat dengan jumlah klien karena setiap klien membuat banyak permintaan HTTP.
  1. **Server-Sent Events (SSE):**
* **Latency**: Waktu tunda lebih kecil dibandingkan Long Polling karena koneksi tetap terbuka, dan server dapat mengirimkan pembaruan data seketika tanpa perlu menunggu permintaan baru dari klien.
* **Efisiensi Koneksi**: Lebih efisien daripada Long Polling, karena hanya satu koneksi HTTP dibuka selama sesi berlangsung.
* **Skalabilitas**: Lebih mudah diskalakan daripada Long Polling, tetapi masih menggunakan protokol HTTP, yang bisa menjadi masalah pada koneksi yang sangat lambat atau dengan banyak klien.
  1. **WebSockets:**
  + **Latency:** Waktu tunda paling rendah karena WebSockets menggunakan koneksi dua arah yang terbuka, memungkinkan server mengirimkan data dengan segera.
  + **Efisiensi Koneksi:** Paling efisien karena koneksi tetap terbuka, dan data dapat dikirim baik oleh klien maupun server tanpa overhead HTTP.
  + **Skalabilitas:** Skalabilitas tinggi, cocok untuk aplikasi dengan banyak klien yang memerlukan komunikasi dua arah secara simultan.

**Analisis Keseluruhan**

* + **Long Polling**: Meskipun mudah diimplementasikan, Long Polling kurang efisien dalam hal waktu tunda dan penggunaan sumber daya. Setiap pembaruan memerlukan permintaan baru, yang tidak ideal untuk skenario dengan pembaruan frekuensi tinggi.
  + **Server-Sent Events (SSE)**: Lebih efisien daripada Long Polling dan cukup cocok untuk aplikasi di mana hanya server yang mengirim data ke klien (misalnya notifikasi, feed data). Namun, SSE hanya mendukung satu arah komunikasi dan menggunakan protokol HTTP.
  + **WebSockets**: Solusi terbaik untuk komunikasi real-time yang membutuhkan pembaruan cepat dan interaksi dua arah. WebSockets sangat efisien dalam hal bandwidth dan latency, cocok untuk aplikasi seperti chat, game online, dan dashboard real-time.

**Pros dan Cons**

* 1. **Long Polling**

**Pros:**

* **Kompatibilitas yang Luas**: Long Polling didukung oleh hampir semua browser dan protokol HTTP, menjadikannya mudah diterapkan di lingkungan yang berbeda.
* **Kesederhanaan**: Implementasinya sederhana, karena menggunakan permintaan HTTP standar yang dikirim berulang kali dari klien ke server.
* **Fallback Solution**: Dapat digunakan sebagai solusi cadangan ketika WebSockets atau SSE tidak tersedia, terutama di jaringan yang tidak mendukung komunikasi persisten.

**Cons:**

* **Overhead Tinggi**: Setiap kali permintaan dikirim, server harus membuat dan menyelesaikan koneksi baru, menyebabkan overhead jaringan dan penundaan tambahan.
* **Inefisien pada Jaringan**: Penggunaan bandwidth lebih tinggi karena koneksi HTTP berulang-ulang, bahkan ketika tidak ada data baru untuk dikirim.
* **Latensi Lebih Tinggi**: Karena klien harus terus-menerus mengirim permintaan baru setelah menerima data, latensi bisa meningkat, terutama dalam skenario yang memerlukan komunikasi real-time cepat.
  1. **WebSockets**

**Pros:**

* **Komunikasi Dua Arah (Full Duplex)**: WebSockets memungkinkan komunikasi dua arah yang persisten antara klien dan server, ideal untuk aplikasi real-time seperti obrolan, game online, atau pelacakan langsung.
* **Efisiensi Bandwidth**: Setelah koneksi WebSocket dibuka, tidak ada overhead koneksi HTTP tambahan, yang mengurangi penggunaan bandwidth dan latensi.
* **Real-Time Communication**: Data dikirim dan diterima segera setelah tersedia, menghasilkan komunikasi real-time yang efisien.

**Cons:**

* **Kompleksitas Implementasi**: Implementasi WebSocket lebih kompleks dibandingkan dengan Long Polling atau SSE, karena memerlukan pengelolaan koneksi persisten.
* **Firewall dan Proxy**: WebSockets mungkin terblokir oleh beberapa firewall atau proxy yang tidak mendukung komunikasi persisten, sehingga memerlukan konfigurasi khusus.
* **Kebutuhan Sumber Daya Server**: WebSockets memerlukan lebih banyak sumber daya server untuk mempertahankan banyak koneksi terbuka secara simultan, terutama dalam skala besar.
  1. **Server-Sent Events (SSE)**

**Pros:**

* **Komunikasi Satu Arah yang Efisien**: SSE menyediakan aliran data satu arah dari server ke klien melalui koneksi HTTP yang terbuka. Ideal untuk aplikasi yang hanya membutuhkan pembaruan data dari server.
* **Efisiensi Lebih Tinggi Dibandingkan Long Polling**: Tidak seperti Long Polling, SSE mempertahankan koneksi terbuka, sehingga mengurangi overhead jaringan dan meningkatkan efisiensi.
* **Kelebihan Native HTTP**: SSE menggunakan protokol HTTP standar, sehingga memanfaatkan header HTTP dan caching yang ada tanpa masalah kompatibilitas besar.

**Cons:**

* **Tidak Mendukung Komunikasi Dua Arah**: Tidak seperti WebSockets, SSE hanya mendukung pengiriman data dari server ke klien. Jika komunikasi dua arah dibutuhkan, SSE tidak cukup.
* **Dukungan Browser Terbatas**: Beberapa browser lama dan klien mungkin tidak mendukung SSE dengan baik. Namun, sebagian besar browser modern sudah mendukungnya.
* **Batas Koneksi Per Domain**: SSE terikat dengan batasan jumlah koneksi yang dapat dibuka oleh klien per domain, terutama di browser.

**Kesimpulan**

* **Long Polling** lebih cocok untuk aplikasi dengan frekuensi pembaruan rendah atau jika lingkungan pengembangan tidak mendukung SSE atau WebSockets.
* **Server-Sent Events (SSE)** ideal untuk aplikasi yang memerlukan pembaruan satu arah dari server ke klien dengan latensi rendah.
* **WebSockets** direkomendasikan untuk aplikasi dengan kebutuhan komunikasi dua arah dan pembaruan real-time, karena menawarkan kinerja dan efisiensi terbaik di antara ketiga metode.

**Rekomendasi:** Untuk aplikasi yang memerlukan pembaruan real-time dengan frekuensi tinggi, WebSockets adalah pilihan terbaik. Namun, jika kebutuhan hanya berupa pembaruan satu arah, SSE bisa menjadi pilihan yang lebih sederhana dan mudah diimplementasikan.