[HTTP polling]

What is HTTP Polling?

HTTP Polling is a technique where a client repeatedly makes requests to the server at regular intervals to check for updates. The server responds with the latest available data, even if it hasn't changed since the previous request.

While simple to implement, HTTP Polling can be inefficient because it generates redundant requests when there is no new data to send.

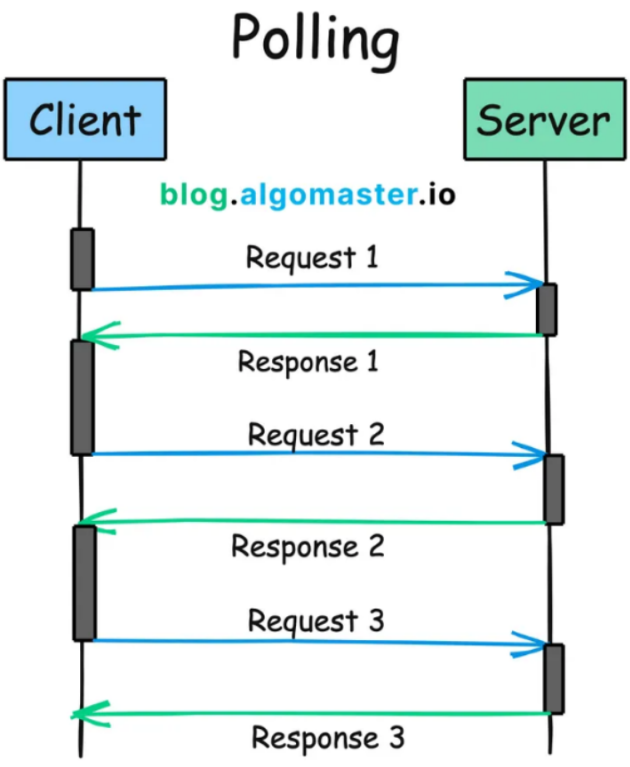
Types of Polling

1. **Short Polling:**
   - The client requests updates at regular, short intervals (e.g., every 5 seconds).
   - Increases server load due to frequent requests, especially if no new data is available.
2. **Long Polling:**
   - The server keeps the connection open until new data is available or a timeout occurs.
   - Reduces redundant requests but increases latency if updates are frequent.

Use Cases for HTTP Polling

1. **Simple Applications:**
   - Suitable for low-traffic or non-real-time systems.
2. **Legacy Systems:**
   - When other real-time techniques (e.g., WebSockets or Server-Sent Events) are unavailable.
3. **APIs Without Push Support:**
   - Clients polling REST APIs for status updates or progress tracking.

Polling: /short polling

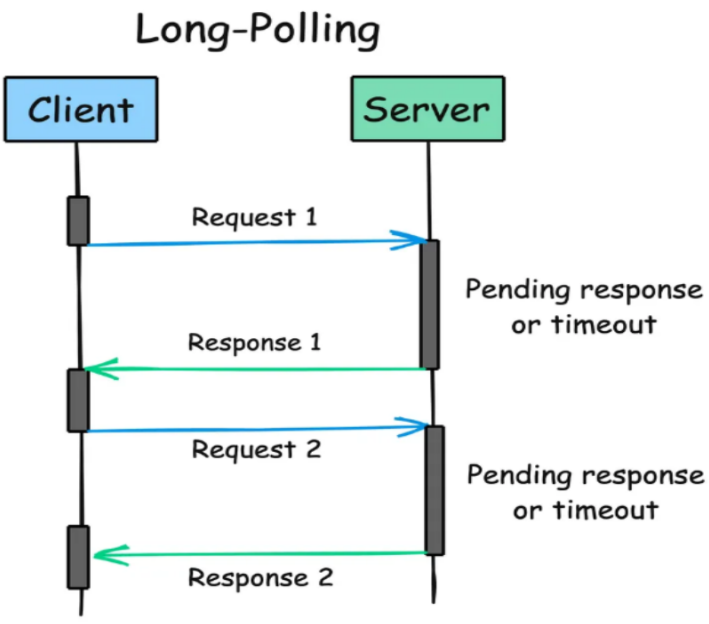


- **Repeated Requests**: The client repeatedly sends requests to the server at fixed intervals to check for updates. While this can simulate real-time updates, it is inefficient, as many requests will return no new data.
- **Latency**: Polling introduces delays because updates are only checked periodically.

```
<script>
    function pollServer() {
        fetch('/updates')
            .then(response => response.json())
            .then(data => {
                const messageList = document.getElementById('messages');
                const newMessage = document.createElement('li');
                newMessage.textContent = data.message;
                messageList.appendChild(newMessage);
            })
            .catch(error => console.error('Error polling server:', error));
    }

    // Poll the server every 5 seconds
    setInterval(pollServer, 5000);
</script>
```

Long-Polling:



- **Persistent Connection**: In long-polling, the client sends a request, and the server holds the connection open until it has data to send. Once data is sent or a timeout occurs, the connection closes, and the client immediately sends a new request.
- **Latency**: This approach reduces the frequency of requests but still suffers from higher latency compared to WebSockets since it requires the client to repeatedly send new HTTP requests after each previous request is completed.
- **Resource Usage**: Long-polling can lead to resource exhaustion on the server as it must manage many open connections and handle frequent reconnections.

```
<script>
    function pollServer() {
        fetch('/updates')
            .then(response => response.json())
            .then(data => {
                const messageList = document.getElementById('messages');
                const newMessage = document.createElement('li');
                newMessage.textContent = data.message;
                messageList.appendChild(newMessage);

                // Immediately poll again
                pollServer();
            })
            .catch(error => {
                console.error('Error polling server:', error);
                setTimeout(pollServer, 5000); // Retry after a delay
            });
    }

    // Start polling
    pollServer();
</script>
```