What is Long Polling in System Design?

When developing a web application that handles real-time data, it's important to consider how to efficiently deliver the data to the client. One option for this is long polling. In this blog, we'll explore how long polling works and the features it offers. Let's get started!

What is long polling?

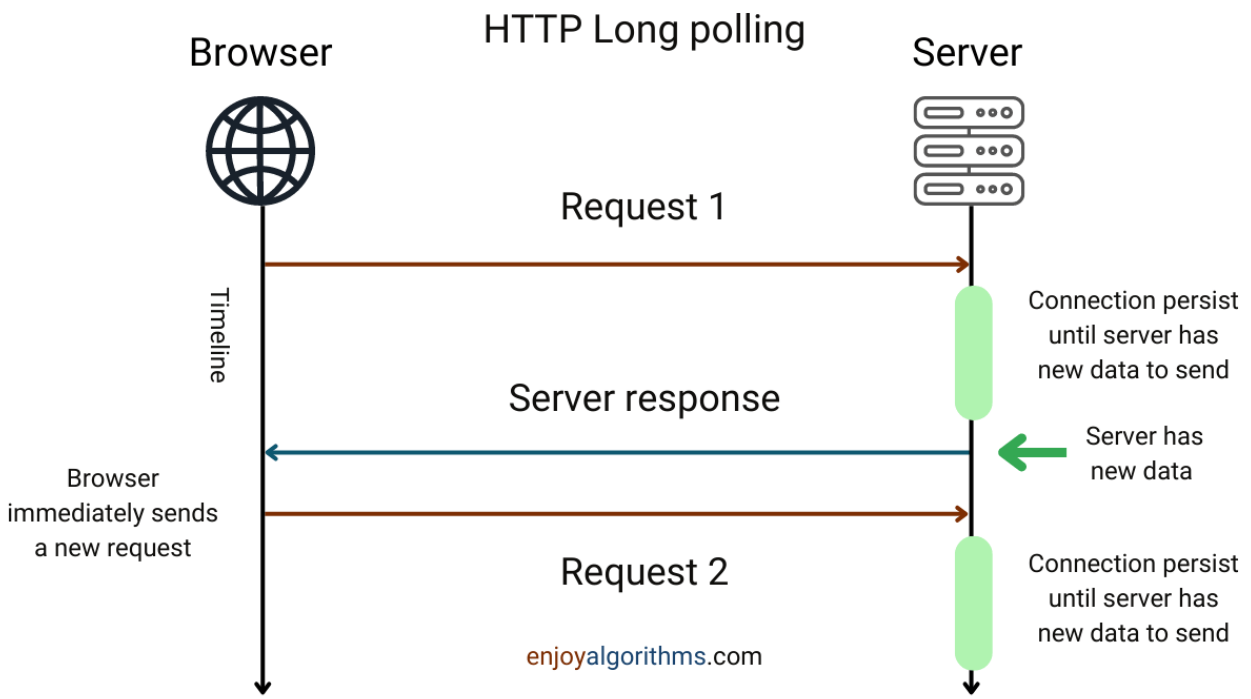
Let’s first look at what polling is and how it extends to Long polling. Polling is a technique that allows servers to send information to clients. On another side, Long polling is a variation of traditional polling that enables the server to send data to the client whenever it becomes available. The client sends a request for information to the server, but the server may not immediately respond. Instead, it waits until the data is available and then sends a complete response to the client.

Long Polling is an efficient way to send information to a client because it reduces the number of HTTP requests required to transmit the same amount of data. However, it can be a challenge for the server to manage unfulfilled client requests and handle new information that becomes available before the client has made a new request.

One advantage of Long Polling is that it is part of the HTTP protocol, which makes it widely accepted and generates less bandwidth than short polling. Overall, Long Polling is a reliable way to keep clients updated with new information in real-time. Here is the basic process of an application that uses HTTP Long Polling:

1. The client sends an HTTP request and waits for a response from the server.
2. When an update is available, the server provides the client with a complete response.

3. The client then sends a new long-poll request immediately or after a pause to allow for an appropriate latency period.
4. A timeout is set for each long-poll request. If the connection is lost due to a timeout, the client must re-establish the connection regularly.



How does long polling work?

Long Polling is a more efficient version of traditional polling because it reduces the number of requests that need to be made to the server. In traditional polling, each request requires establishing a new connection, parsing HTTP headers, issuing a new data query, and generating and delivering a response. The connection is then terminated and resources are cleaned up.

Long Polling allows the server to keep a client's connection open for as long as possible, only responding when data becomes available or a timeout threshold is reached. This avoids the need to repeat the process for each client until new data becomes available, which helps to conserve resources.

In Long Polling, most of the work is done on the server side. On the client

side, only one request to the server needs to be managed. When the client receives a response, it can make a new request and repeat the process as needed. The main difference between basic polling and Long Polling from the client's perspective is that a client performing basic polling may intentionally leave a small time window between each request to reduce server load. It may also handle timeouts differently than a server that does not support Long Polling.

For example, with Long Polling, the client may be configured to allow a longer timeout when waiting for a response. This is done to avoid problems with communication with the server, which may be identified by a timeout period.

Apart from these considerations, there isn't much else that a client needs to do that isn't already covered by basic polling. The server, on the other hand, must manage the state of multiple unresolved connections. When multiple servers and load balancers are used, it may be necessary to create solutions to preserve session state. The server must also gracefully handle connection timeout issues, which are more common with Long Polling than with purpose-built protocols.

Considerations and challenges when using long-polling

Using HTTP long Polling to build real-time interactivity in an application comes with various considerations, both in terms of development and scaling. Some questions to consider include:

- How will you manage the real-time backend as usage grows?
- Will long Polling automatically re-establish connections when mobile devices switch between networks or lose connections, or when the IP address changes?
- Can you manage the message queue and catch up on missed messages with long Polling?
- Does long Polling provide load balancing and failover support across multiple servers?

Keep in mind that long Polling is simply an improvised version of the underlying request-response mechanism, which adds complexity to its implementation. As a result, these are important factors to consider when using long Polling in your application.

You'll have to develop your communication management system when building a real-time application with HTTP long polling for server push. This means that you'll be responsible for updating, maintaining, and scaling your backend infrastructure.

Message ordering and delivery

Long Polling can pose challenges in terms of reliable message ordering. This is because it's possible for multiple HTTP requests from the same client simultaneously. For example, if a client has two browser tabs open consuming the same server resource or if the client implementation uses more than one connection at a time, there may be no guarantee that duplicate data will be written only once.

Another issue is that a server may send a response, but network or browser issues may prevent the message from being successfully received. If a message receipt confirmation process is not implemented, a subsequent call to the server may result in missed messages.

Performance and scaling

One challenge of using Long Polling is that it can be difficult to scale due to its complexity. Maintaining the session state for a given client requires either sharing it among all servers behind a load balancer or routing subsequent client requests within the same session to the same server that processed the original request. This can result in uneven distribution of load and potentially overload a single server in a cluster.

Conclusion

HTTP Long Polling is a way for the server to send data to the client independently, which allows the server to push data to the client in real-

time as it becomes available. However, it is most effective when the messages from the server are rare and not too frequent.

Thanks Suyash Namdeo, for his contribution in creating the first version of this content. Please write in the message below if you want to share more insight. Enjoy learning. Enjoy algorithms!