Long Polling, Websockets, Server Side Events

Background

In a decent amount of systems design interview questions, there is a need to provide real time updates from a database or server back to a client. While there are multiple options for doing this, each has their own benefits and drawbacks.

These things come up in chat applications, ride sharing applications (to continually see where your driver is located), and even things like notification services.

Polling (Naive Implementation)

Make a request to the server every few seconds from the client asking for new data.

This is bad because you will overload your servers with unnecessary requests!

Long Polling

Make a typical HTTP request to the server for data, if the server does not have new data to respond with it keeps the request open, and eventually responds once new data is present. At this point, the client will issue another long polling request. If the connection times out, also send another request.

Pros:

- Easy to implement, universal support

Cons:

- Only one directional, any information sent from client to server must be done through a normal HTTP request
- Excess load on server by constantly tearing down connections HTTP connections and re-establishing them
- Possible race condition of multiple requests from same client receiving messages out of order (two tabs open, deduplication needed)

WebSockets

WebSockets are a fully bidirectional communication channel between clients and servers. WebSockets are each registered to a port, so a given application service can have around 65,000 active connections before needing to be horizontally scaled.

Pros:

- Bidirectional communication in real time (as opposed to near real time)
- Lower network overhead due to headers only being sent once

Cons:

- Lesser support for them on older browsers
- Thundering herd problem due to overhead of establishing a websocket connection if the number of application servers changes
- May be overkill for infrequent data changes

Server Sent Events

One way connection from servers to clients, client-side code registers for events from the server. Good for things like stock ticker updates or notifications.

Pros:

- Unlike long polling use persistent HTTP connection as opposed to killing and reconnecting
- Unlike websockets re-establish failed connections automatically

Cons:

- Single directional communication
- A lot of concurrent connections can add load on server if they are not needed

Conclusion

While generally it is going to be enough in systems design interviews to mention using websockets for most real time applications, it is important to be able to distinguish some of the subtle differences between them and long polling or server side events. In actual tech companies, these latter two are still frequently used due to their widespread support, ease of understanding, and lower overhead.