Communication Protocols

What is the communication protocol?

A communication protocol is a rule system that allows two or more entities in the communication system to transmit information through any type of change in physical quantities. The protocol defines the rules, syntax, semantics and timing of communication and possible error recovery methods. These protocols can be implemented using hardware, software, or a combination of both.

Example: HyperText Transfer Protocol(HTTP), WiFi

Push and Pull Model

1. Push: In the push protocol, the client opens the connection with the server and keeps it always active. The server will use this single connection to send (push) all-new events to the client each time. In other words, the server pushes new events to the client. This method reduces the server load.

   Example: Suppose your mobile phone is always connected to the mobile network. We can judge the connectivity by the signal bar on the phone screen. When the caller makes a call, the network will send the call to your mobile phone through the active connection that your mobile phone already has. This is a push.

2. Pull/Polling: In the pull protocol, the client periodically connects to the server, checks and gets (pulls) the latest events, and then closes the connection and disconnects from the server. The client repeats the entire process to receive updates on new events. In this mode, the client will periodically pull recent events from the server. It is the default method of HTTP communication.

   Example: When we are waiting for a specific show on the TV, we will switch ON the TV repeatedly and check if our particular show has started, and then close it again if it hasn't. This is pulling.

The pull Model is more commonly used than the push Model.

The flow of data from Client -> Server: Pull Model
The flow of data from Server -> Client: Push Model

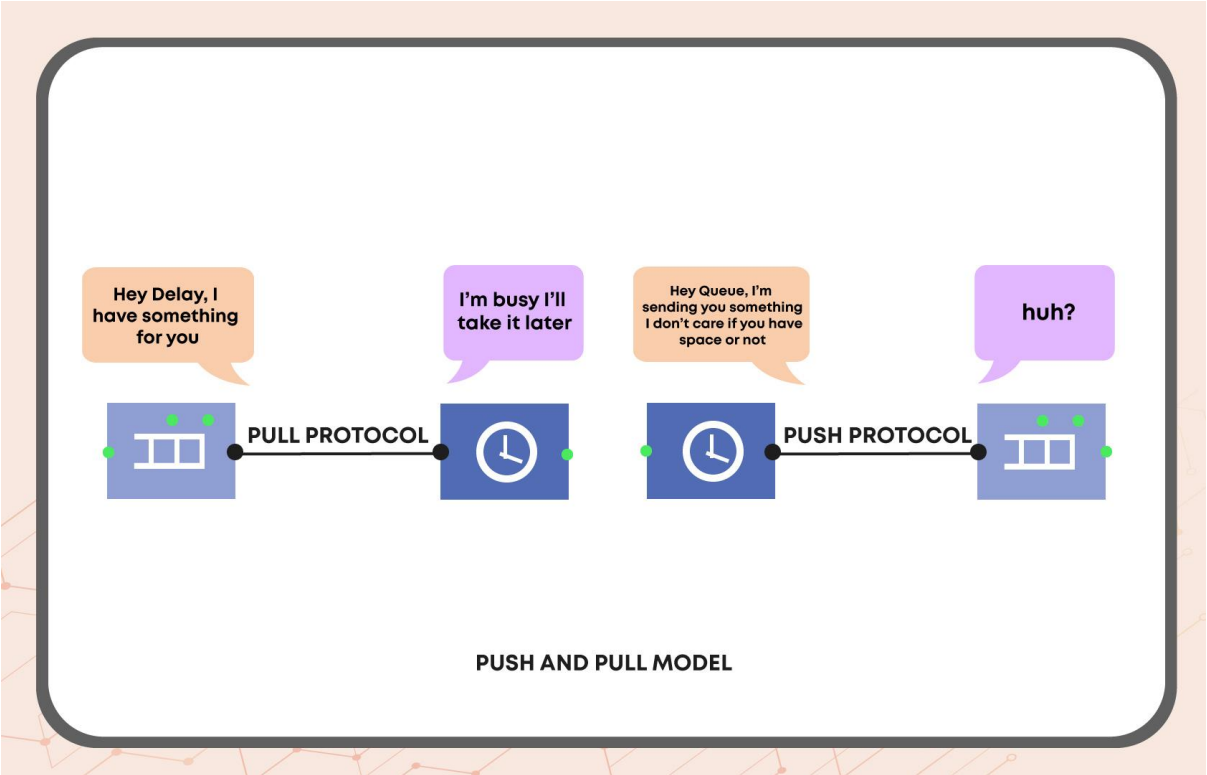

Fig: Push and Pull Model

Steps of Client-Server Communication

1. The client initiates the communication and waits for the response.
2. Computation and processing at the server end.
3. The server sends back the response.

When to use Push and Pull Protocols:

- The push approach is better in a low write rate and a large number of client use cases. (a celebrity makes a status update on Instagram and has millions of followers; so we use a push strategy for the user's news feed to update)
- The poll approach is better in a high write rate and a low number of client use cases. (a web crawler can poll for updates on wiki pages to update its index for search engines)

3. Long Polling: To overcome the deficiency of the Polling protocol, web application developers can implement a technique called HTTP long polling, in which the client polls the server for new information. The server keeps the request open until new data is available. Once available, the server will respond and send further

information. When the client receives new information, it immediately sends another request and repeats the operation. This effectively simulates the insert function of the server.

Challenges of long polling:

1. Ordering of the message cannot be guaranteed when multiple connections are established.
2. Message delivery cannot be guaranteed, which may lead to message loss.

The steps of long polling are:

1. The client sends the request.
2. If the data is not available, the server will wait for the computation. This causes a delayed response.
3. The client gets free once the response is obtained.

4. Socket Connection: A socket is an endpoint of a two-way connection link between two servers/nodes over the network. The socket is identified by a port number so that the TCP layer can identify the application to which the data will be sent. Each TCP connection can be uniquely identified by its two endpoints. Sockets are usually used for interaction between client and server. The plug has a typical flow of events. In the connection-oriented client-server model, the socket in the server process waits for a request from the client. To this end, the server first sets (binds) an address, which the client can use to find the server. It is used for applications that need a dedicated connection that remains open so that the data can flow in proper order securely and immediately from both sides.

5. Server-Sent Events(SSE): In the Server-Sent Events, the client subscribes to the server "stream", and the server will send a message ("stream of events") to the client until the server or client closes the stream. The server decides when and what to send to the client, for example, once the data changes. SSE is usually used to send continuous data streams or message updates to the browser client. In short, the server sending event is the time when the update is sent from the server (not pulled or requested) to the browser.

Polling	Socket Connection	Server-Sent Events
Client requests the Server	The socket on the server process waits for requests from a client.	The server sends the message to the Client
One way connection	Two-way connection	One way connectionS
Short-Lived Connection	Long-Lived Connection	Long-Lived Connection