System Design Fundamentals

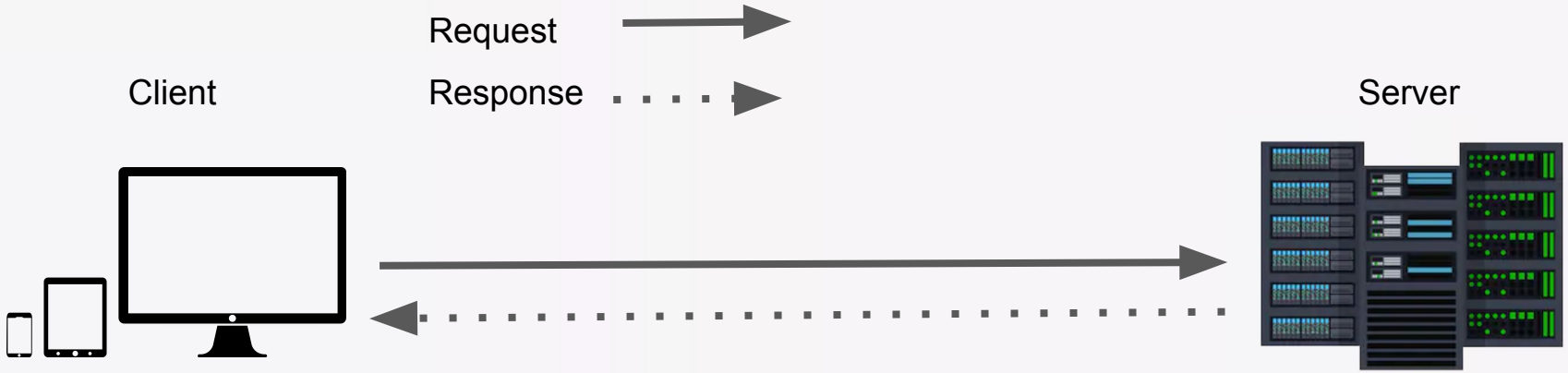
Network Layer

Network Responses

- HTTP Response
- AJAX Polling
- Long Polling (Hanging Get)
- Web sockets (Full Duplex)
- Server Sent Events (SSE)

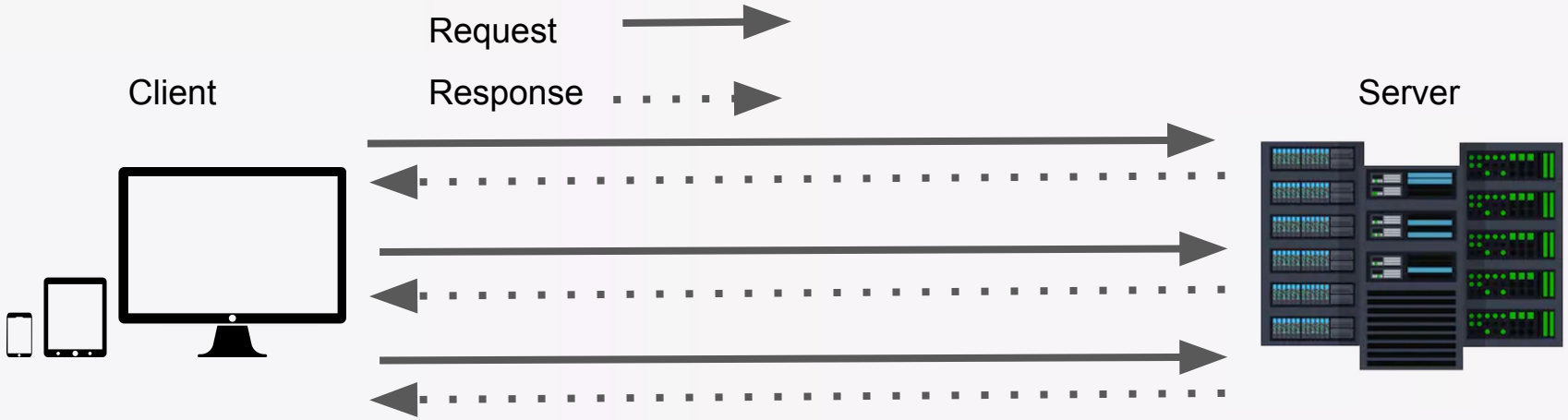
HTTP Response

1. Client opens connection and post request.
2. Server sends response back to client on opened request.



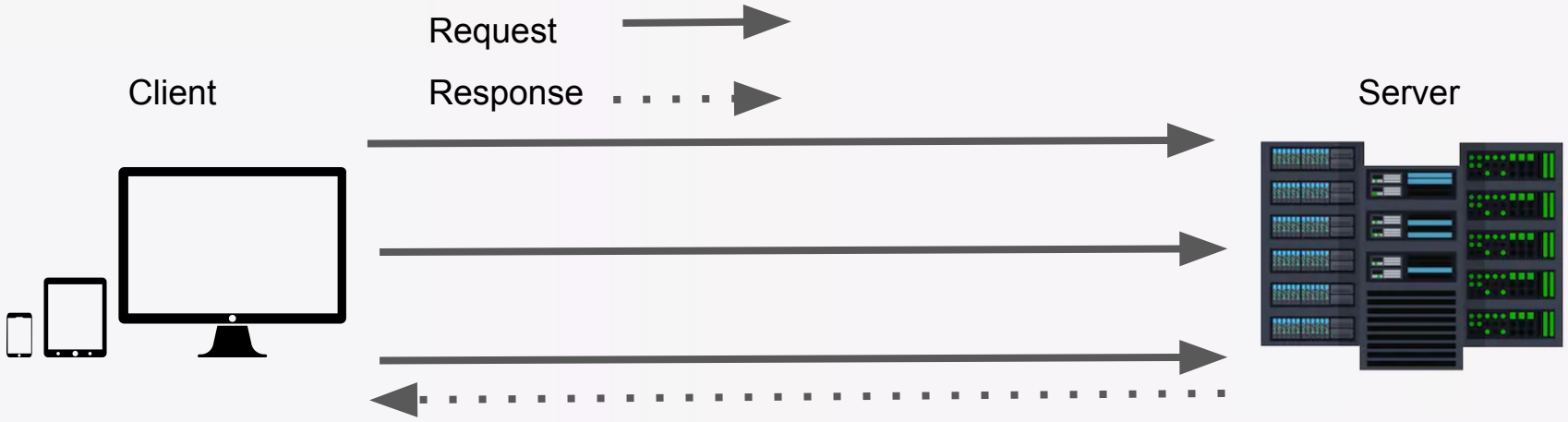
AJAX Polling

1. Client opens connection and requests data using regular HTTP.
2. Requested web page **sends requests to the server at regular intervals**.
3. Server calculates the response and sends it back, like regular HTTP.
4. Client repeats the above 3 steps periodically to get updates from the server.



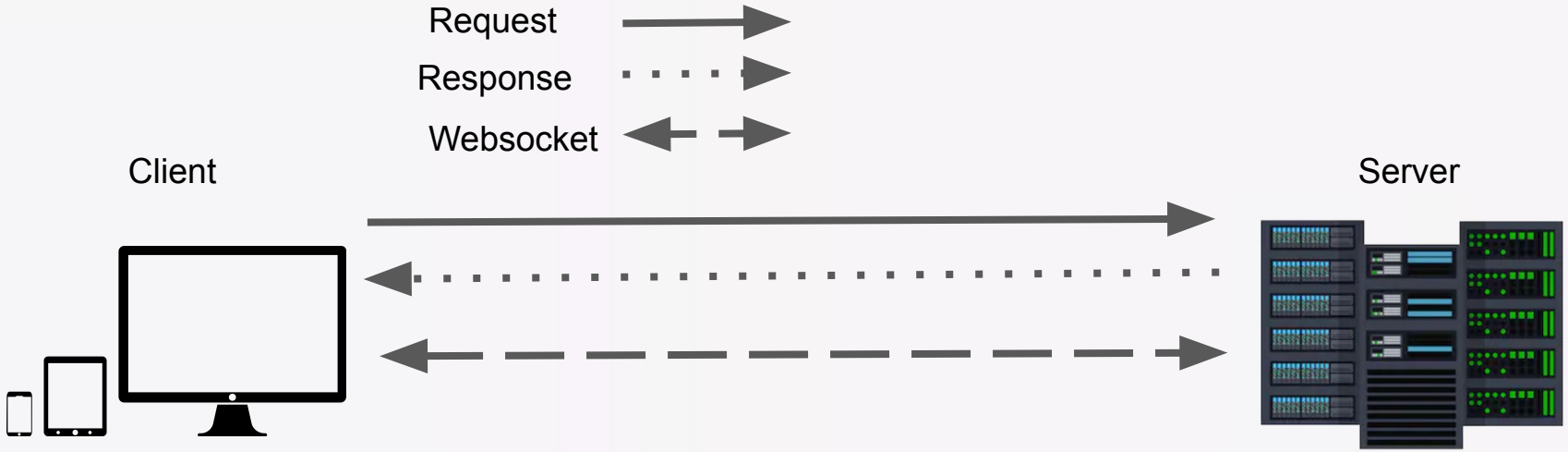
Long Polling (Hanging Get)

1. Client makes initial request using regular HTTP and await response.
2. **Server delays its response until an update is available or a timeout has occurred.**
3. When an update is available, the server sends a full response to the client.
4. **Client typically sends a new long-poll request, either immediately upon receiving a response or after a pause to allow an acceptable latency period.**
5. Each Long-Poll request has a timeout. Client has to reconnect periodically after the connection is closed due to timeouts.



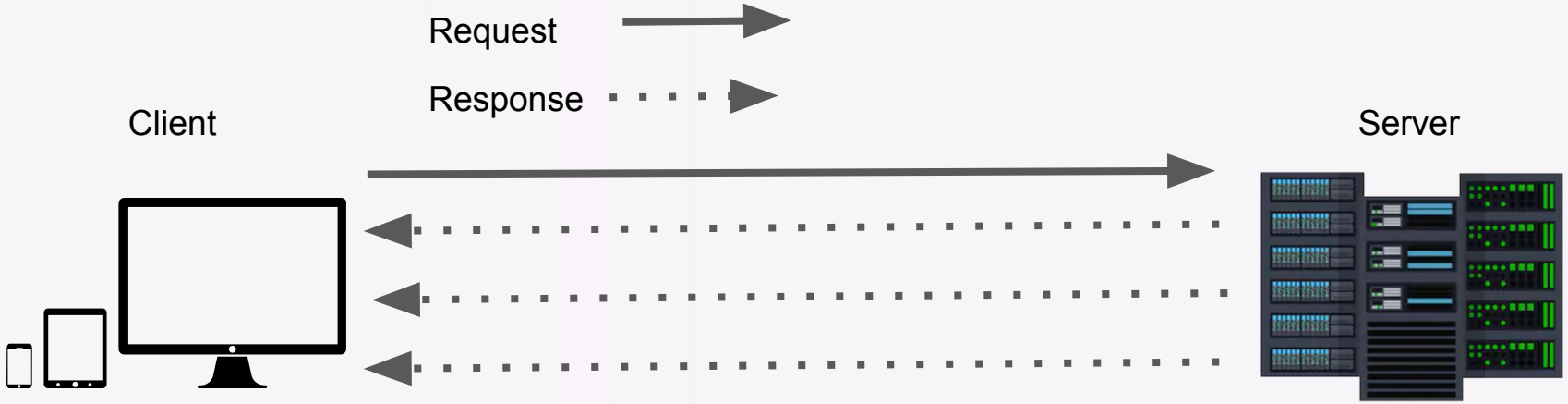
Web sockets (Full Duplex)

1. Client makes initial request for WebSocket handshake using regular HTTP and await response.
2. Server respond with acknowledging the request
3. Websocket Protocol



Server Sent Events (SSE)

1. Client requests data from a server using regular HTTP.
2. Requested web page opens a connection to the server.
3. **Server pushes the data to the client whenever there's new information available.**



Push vs Pull

Pull : Fanout-on-load
(Client updates itself)

- Hard to find the right pull cadence
- Most of the time pull requests are empty responses
- Waste of resources

Push : Fanout-on-write
(Server updates client)

- Resource hogging on server side
- Unnecessary bandwidth consumption.

‘Push to notify’ and ‘Pull for serving’

SOAP vs REST

SOAP

- Standardized ✓
- Language, platform, and transport independent ✓
- SOAP uses XML for all messages ✗
- Works well in distributed enterprise environments ✓
- Built-in error handling ✓
- Automation when used with certain language products ✓

REST

- Smaller learning curve ✓
- REST requires use of HTTP ✗
- Fast (no extensive processing required) ✓
- Closer to other Web technologies in design philosophy ✓

System Design Fundamentals

Network Layer