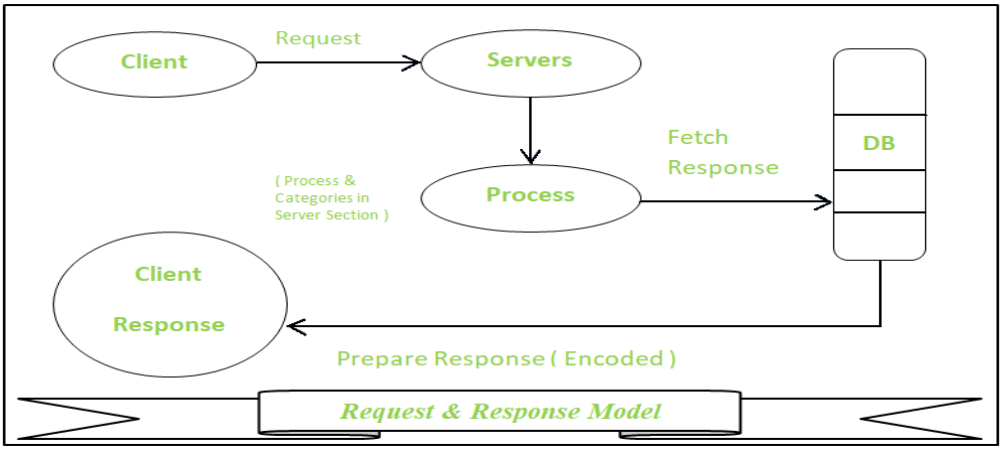
Introduction to IoT Communication Protocols

Rishabh Maheshwari
19BCY10145

1. Explore Client-server model of communication

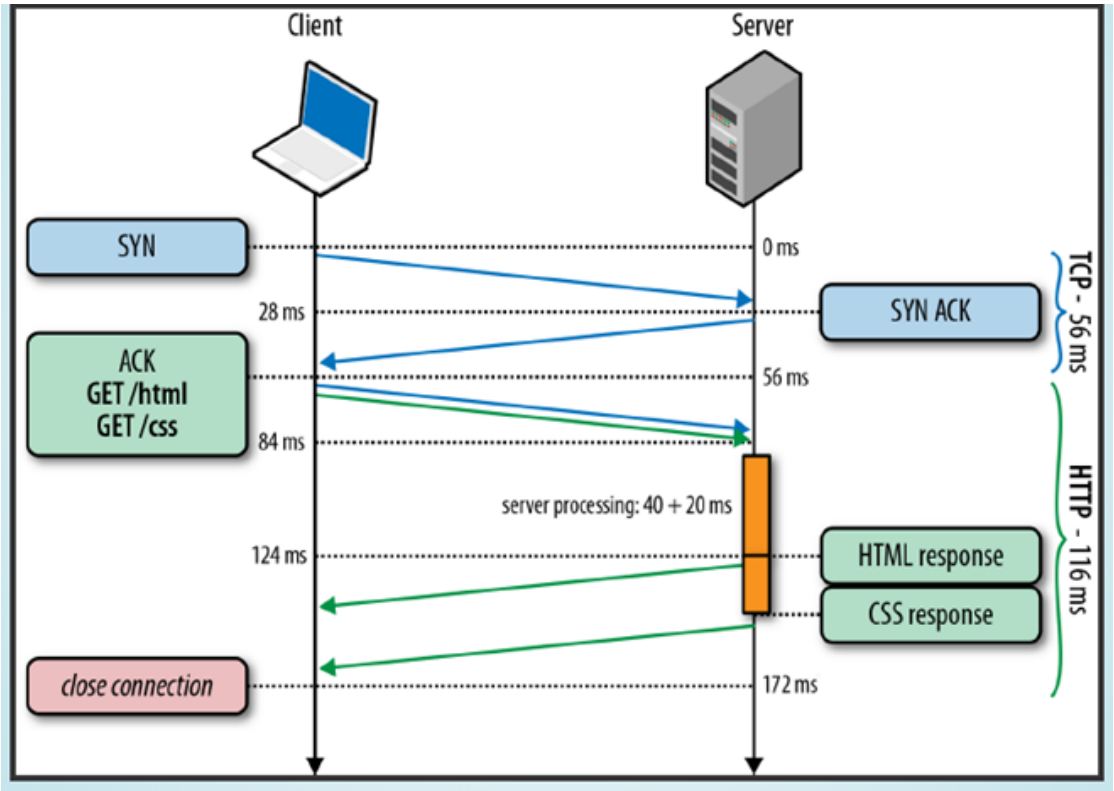
This model follows a client-server architecture.

- The **client**, when required, requests the information from the server. This request is usually in the encoded format.
- This model is stateless since the data between the requests is not retained and each request is independently handled.
- The server Categories the request, and fetches the data from the database and its resource representation. This data is converted to response and is transferred in an encoded format to the client. The client, in turn, receives the response.
- On the other hand —
  In **Request-Response** communication model client sends a request to the server and the server responds to the request. When the server receives the request it decides how to respond, fetches the data retrieves resources, and prepares the response, and sends it to the client.



2. Explore HTTP and HTTPS communication protocols

The Hypertext Transfer Protocol is an application protocol for distributed, collaborative, hypermedia information systems that allows users to communicate data on the World Wide Web. HTTP was invented alongside HTML to create the first interactive, text-based web browser: the original World Wide Web. Today, the protocol remains one of the primary means of using the Internet.



HTTP data rides above the TCP protocol, which guarantees reliability of delivery, and breaks down large data requests and responses into network-manageable chunks. TCP is a “connection” oriented protocol, which means when a client starts a dialogue with a server the TCP protocol will open a connection, over which the HTTP data will be reliably transferred, and when the dialogue is complete that connection should be closed. All of the data in the HTTP protocol is expressed in human-readable ASCII text.

While HTTPS prevents communication sniffing, data manipulation and offers verification of your peer. This should make you feel safer, since even when your router is compromised, your bank account login is still safe. Also, HTTPS verifies the identity of the server, so that you can be sure who you are communicating with, and in case of a crime happening, you know who to sue.

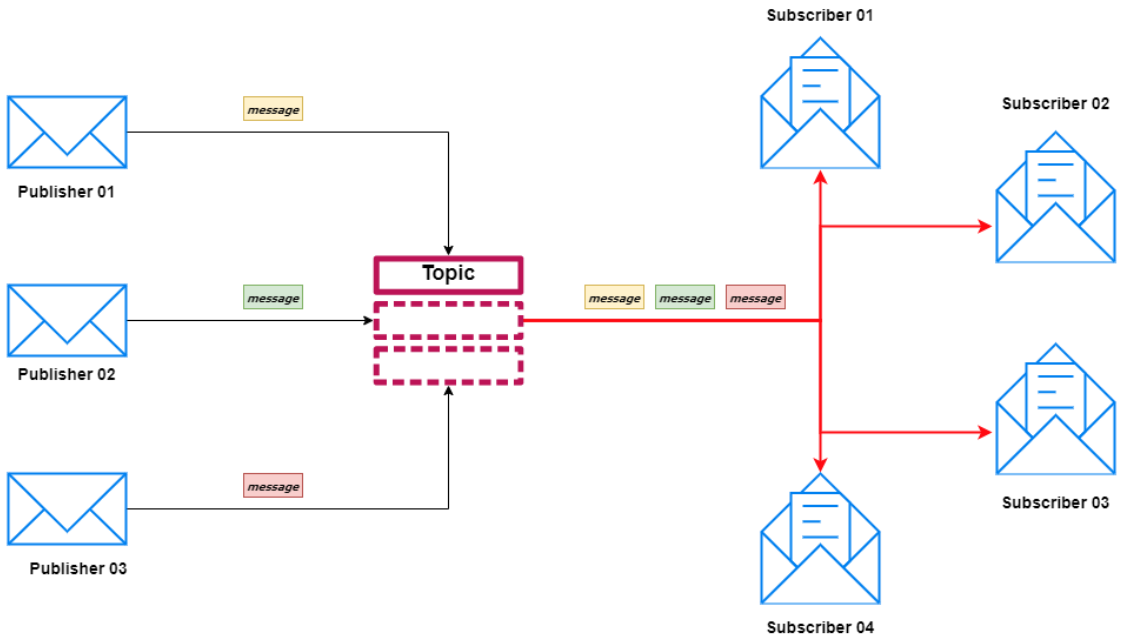
The core problem with HTTPS is that a fundamental component of it is the host name. Browsers compare the host name that you entered into the address bar with the host name that the server's certificate has been issued to. If these don't match, the browser will not connect to the server, except when your user wades through hardly visible links to finally add an exception for your device. And it will tell your users that it is highly insecure what he is currently doing along the way.

3. Explore Pub-sub model of communication

With the popularity of decoupled
and microservices-based applications, proper communication between components and services is crucial for overall application functionality. Pub/Sub messaging helps with this in two crucial ways:

Allowing developers to create decoupled applications easily with a reliable communication method

Enabling users to create event-driven architectures easily



A pub/sub model allows messages to be broadcasted asynchronously across multiple sections of the applications.

The core component that facilitates this functionality is something called a Topic. The publisher will push messages to a Topic, and the Topic will instantly push the message to all the subscribers. This is what differentiates the Pub/Sub model from traditional message brokers, where a message queue will batch individual messages until a user or service requests these messages and retrieves them.

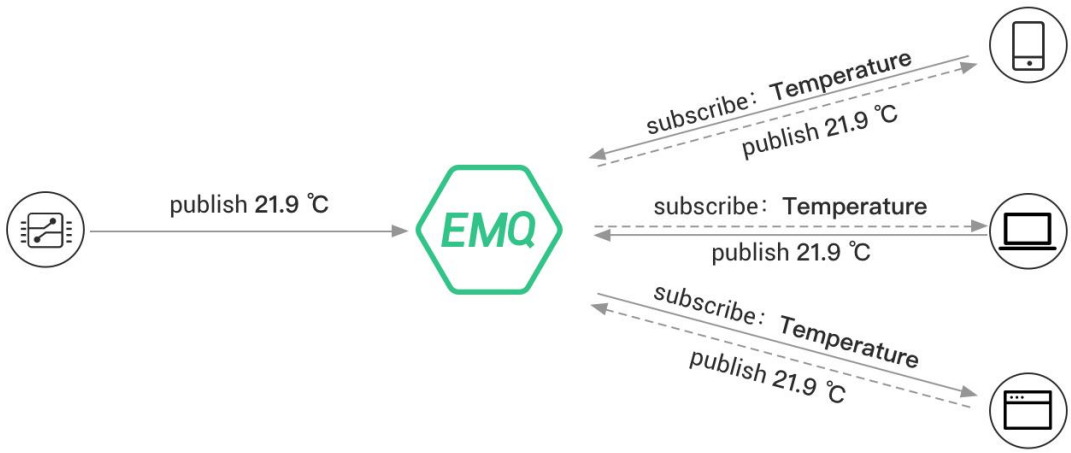
Whatever the message is in the Pub/Sub model, it will be automatically pushed to all the subscribers. The only exception is user-created policies for subscribers that will filter out messages.

4. Explore MQTT Protocol

MQTT is a lightweight message protocol for sending simple data flows from sensors to applications and middleware. It works on top of the TCP/IP network for supplying reliable yet

simple streams of data. Though it may work with any network that provides ordered, lossless, & bi-directional connections. The MQTT protocol comprises 3 key elements: subscriber, publisher, and a broker.

It separates the client (publisher) that sends the message from the client (subscriber) that receives the message. The publisher and the subscriber do not need to establish direct contact. We can either let multiple publishers publish messages to one subscriber, or let multiple subscribers receive messages from one publisher at the same time. The essence of it is that an intermediate role called a broker is responsible for all message routing and distribution. The traditional client-server model can achieve similar results, but it cannot be as simple and elegant as the publish-subscribe model.



The advantage of the publish-subscribe model is the decoupling of publishers and subscribers. This decoupling is manifested in the following two aspects:

- Spatial decoupling. Subscribers and publishers do not need to establish a direct connection, and new subscribers do not need to modify the publisher's behavior when they want to join the network.

- Time decoupling. Subscribers and publishers do not need to be online at the same time. Even if there are no subscribers, it does not affect publishers to publish messages