INTERNET OF THINGS

UNIT – 1

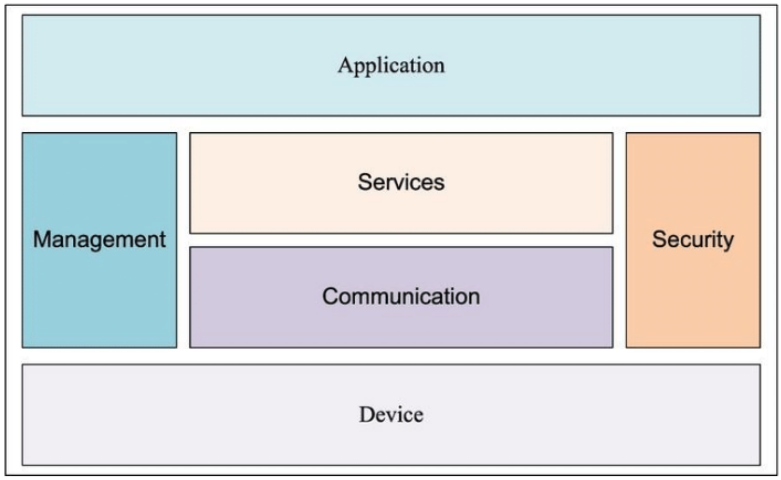
Logical design of IoT

Logical design of an IoT system refers to an abstract representation of the entities and process without going into low level specification of the implementations.

Logical design of IoT contains the following parameters.

1) IoT functional block diagram
2) IoT Communication models

IoT Functional block diagram



An IoT system comprises of a number of functional blocks that provide the system the capabilities for identification , sensing, actuation ,communication and Management.

The function blocks are described as follows

Devices: An IoT system comprises of the devices that provide sensing, actuation, monitoring and control function

Communication: communication block handle the communication systems

Services : An IoT system uses various types of IoT services such as services for device monitoring ,device control services ,data publishing services and services for device discovery.

Management: Functional blocks provide various functions to govern the IoT system

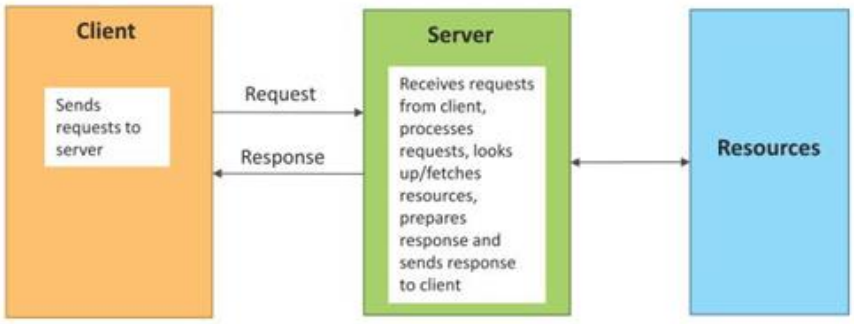
Security: Security functional block security IoT system and by providing functions such as application authorization message and content integrity and data security.

Application: IoT application provides and interface that the user can used to control and monitor

various aspects of the IoT system. Application also allow users to view the system status and view or analyze the processed to data.

IoT communication models

Request-Response communication Prorocol



This communication model consists of two parties:

I) Client ii) server

When the client requires any data, it sends a request to the server

The server upon receiving the request decides how to respond, fetch the data and sends its response to the client.

Each request-response pair is independent as it is a stateless model.

Ex: Online search engine: when we want to search anything we type a keyword in search bar it fetches all the data relevant to the word we typed and finds all the related websites, images,links on our screen.

Publish-Subscribe communication Protocol

This communication model consists of three parties:

1) Publisher re data producers or data generators
2) subscriber subscribers are data consumers and
3) Brokers are data mangers

Publishers: They produce data related to various topics let us say sports, fashion, science and technology anything where publishers are not aware of subscribers.

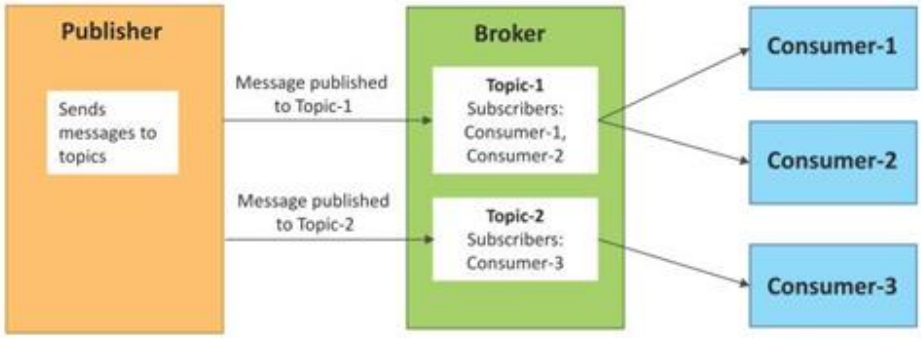
Broker: Forms groups and categorizes of this data according to various topics be it sports, fashion etc

borker Identifies the subscribers who has subscribed to the topic 1 then send data related to topic 1 to those subscribers and so on. Publishers and subscribers are connected by the broker, which acts as the connection link.

Subscribers: subscribe various topics according to their interest and brokers/managers provide data to

them depending upon their choice of subscription.

Ex: Social media platforms work on this model where we subscribe some channels, like someone's pages or follow someone on social media platform then all the data related to that channel or pages they all appear on our timeline this is how they work because the broker are aware that we have liked that persons page or subscribe that persons channel.



Push-Pull communication model

This communication model consists of two parties:

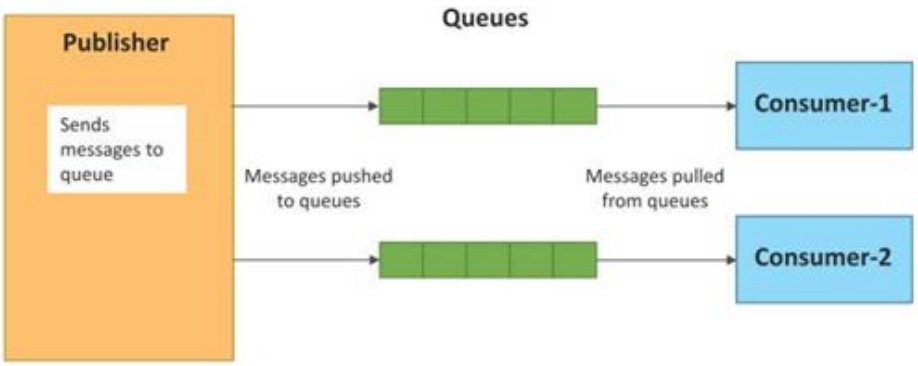
I) Data producer 2) Data consumers

Data producers generate data related to various topics and push this data into queues (waiting list)

Data consumers pull data out of these queues

Producers don't require awareness of consumers.

Ex" Social media platforms we scroll down various topics like videos, images, news items them one to our timeline you can visualize as a queue. This queue acts as a buffer or decoupling unit between the data producers and data consumers.if we subscribe some channel then all the data related to that channel comes into timeline from they are placed in the queue.

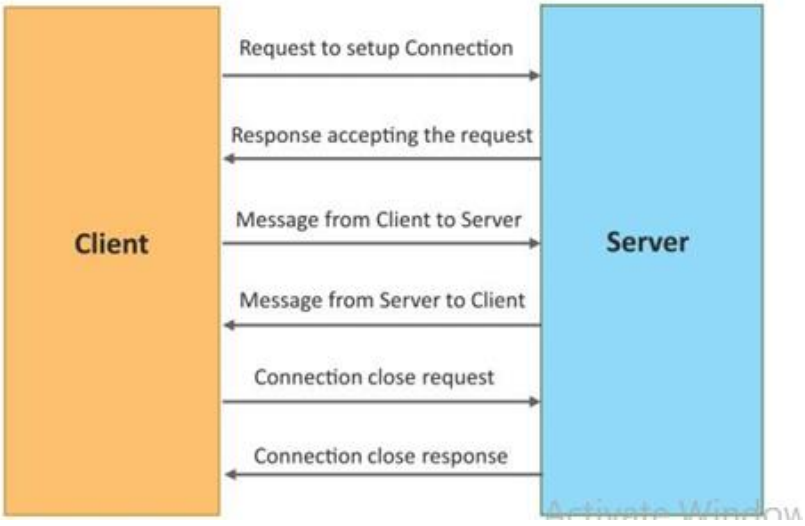


Exclusive pair

This communication model consists of two parties:

1) client 2) server

Exclusive pair is a bidirectional, fully duplex communication model with persistent connections between client and server, remaining open until the client requests closure. Both client and server can exchange messages after the connection is established. It's a stateful communication model, with the server aware of all open connections.

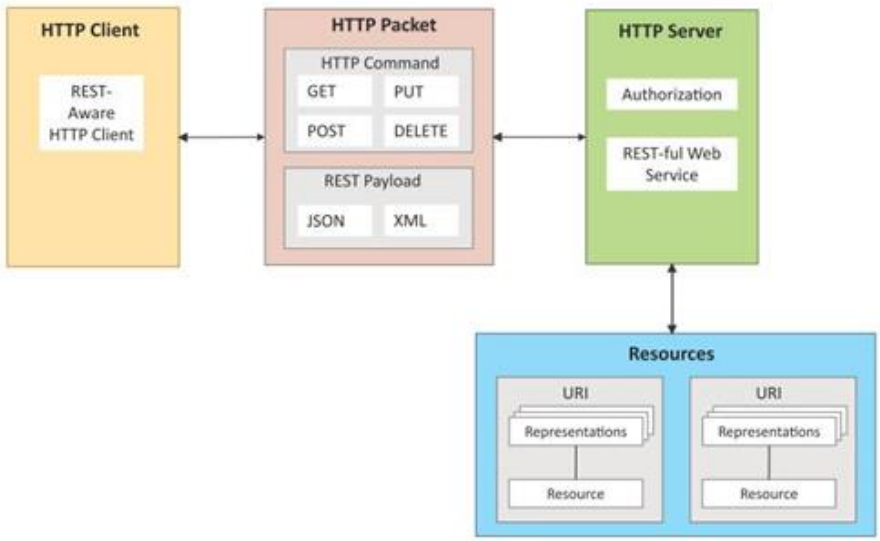


IoT communication APIs

REST-based communication API

REST-based communication API follows representational state transfer principles, focusing on system resources, their states, and addresses for transfer. REST APIs adhere to the request-response

communication model, applying architectural constraints to components, connectors, and data elements.



Client server:

Separation of concerns in client-server architecture ensures that the client focuses on user interface while the server handles data storage, enabling independent deployment and updates for each component.

Stateless:

Each request from client to server must contain all the information necessary to understand the request , and cannot take advantage of any stored context on the server

Catchable:

By categorizing response data as catchable or non-catchable, the cache allows the reuse of data for similar requests, reducing redundancies and enhancing efficiency and scalability.

Layered system:

System constraints limit component behavior to interacting only with immediate layers, meaning a client cannot discern if it's connected directly to the end server or an intermediary. By enabling intermediaries to respond to requests rather than the tender server, system scalability can be enhanced

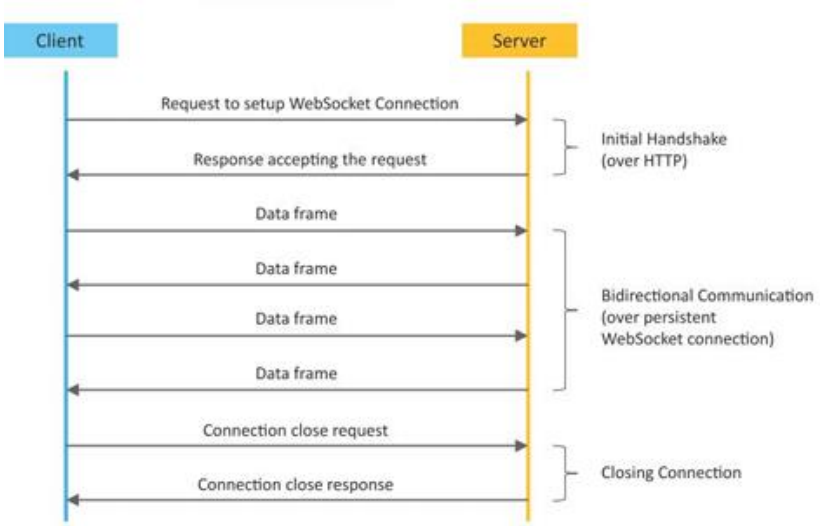
Uniform interface:

The uniform interface constraint mandates consistent communication between client and server, where resources are identified in requests and separate from the representations returned to the client. When the client holds a representation of a resource, it contains all necessary information to update

or delete the resource.

Code on demand: Service can provide executable code script for clients to execute in their context.

Web Socket based communication API:



The Web Socket API enables bidirectional, full-duplex communication between client and server without requiring new connections for every message.

It starts with the client sending an HTTP connection setup request, recognized by the server as an upgrade request, followed by a handshake upon protocol support, enabling smooth data transmission. This decreases network traffic and latency by removing connection setup overhead for each message.

Difference between REST API and Web socket API

REST API	Web Socket
Stateless	Statefull
Request-response communication model	Full duplex
Each request involves setting up a new TCP connection	Single TCP connection
Header overload	No header overload
Not suitable for real time applications	suitable for real time