UNIT I FUNDAMENDALS OF IOT

Definition and Characteristics of IoT, Sensors, Actuators, Physical Design of IoT – IoT Protocols, IoT communication models, IoT Communication APIs, IoT enabled Technologies – Wireless Sensor Networks, Cloud Computing, Embedded Systems, IoT Levels and Templates, Domain Specific IoTs – Home, City, Environment, Energy, Agriculture and Industry.

LOGICAL DESIGN of IoT

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

1) IoT Functional Blocks
2) IoT Communication Models
3) IoT Comm. APIs

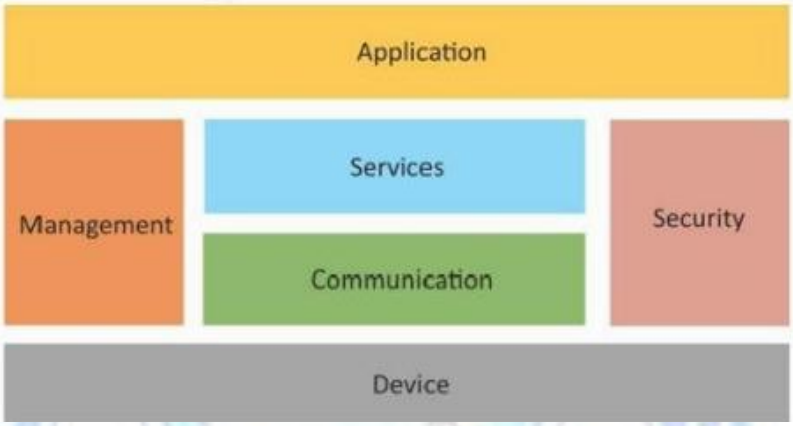
1) IoT Functional Blocks:

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.

- **Device**: An IoT system comprises of devices that provide sensing, actuation, monitoring and control functions.
- **Communication**: handles the communication for IoT system.
- **Services**: for device monitoring, device control services, data publishing services and services for device discovery. 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**: Provides various functions to govern the IoT system. • Security: Secures IoT system and priority functions such as authentication, authorization, message and context integrity and data security.
- **Application**: IoT application provide an interface that the users can use to control and monitor various

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

Figure: IoT functional block

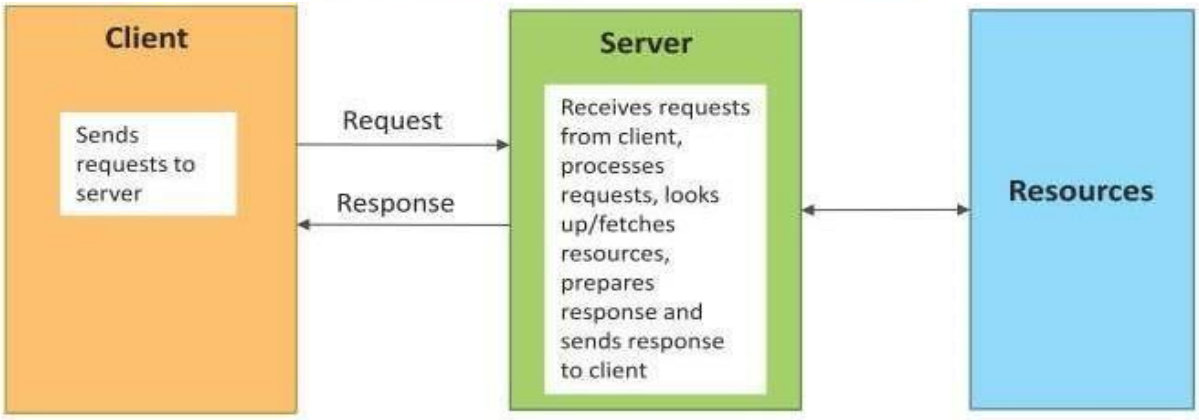


2) IoT Communication Models:

1) Request-Response 2) Publish-Subscribe 3) Push-Pull 4) Exclusive Pair

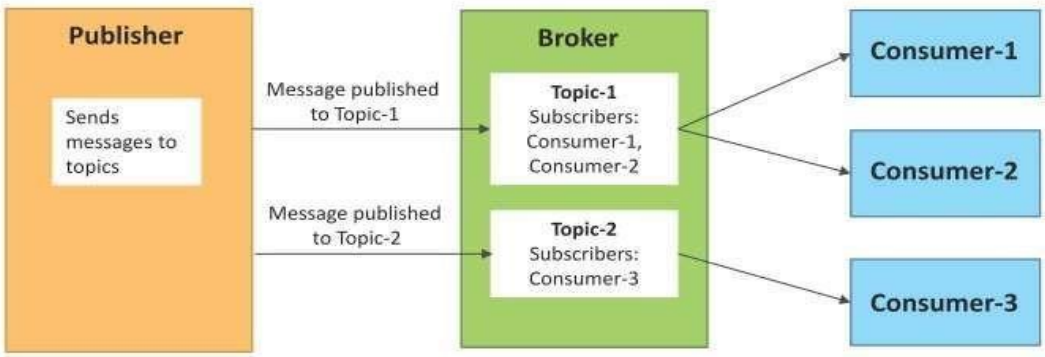
1. Request–Response Communication Model:

- Request-response is a Communications model in which the client sends request to the server and the server responds to the requests.
- when the server receives a request, it decides how to respond, if it shows the data retrieved resources definitions for the response, and then send the response to the client.
- Access to response model is a stateless communication model and each request response per is independent of others the crime and server interactions in the request response model.

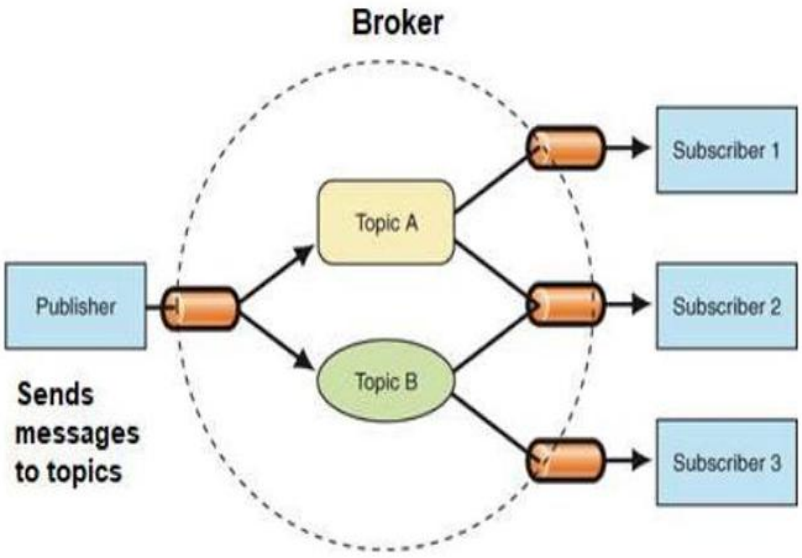


2. Publish–Subscribe Communication Model:

- Publish - Subscribe is a communication model that involve Publishers, brokers and consumers.
- Publishers are the source of data. Publishers send the data to the topics which is managed by the broker. Publishers are not aware of the consumer.
- Consumers Subscribe to the topic which are managed by the broker. When the broker receives the data for a topic from the publisher, it sends the data to all the subscribed consumers

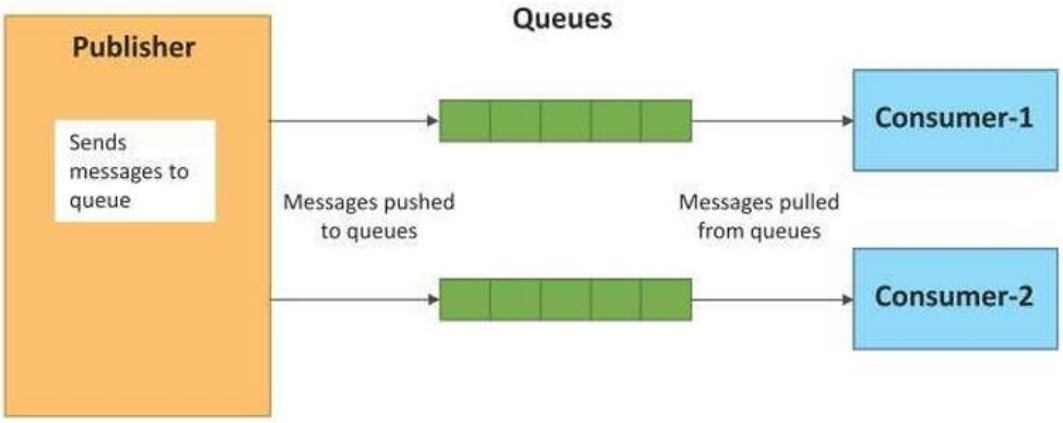


Publish–Subscribe Communication Model.



3. Push–Pull Communication Model:

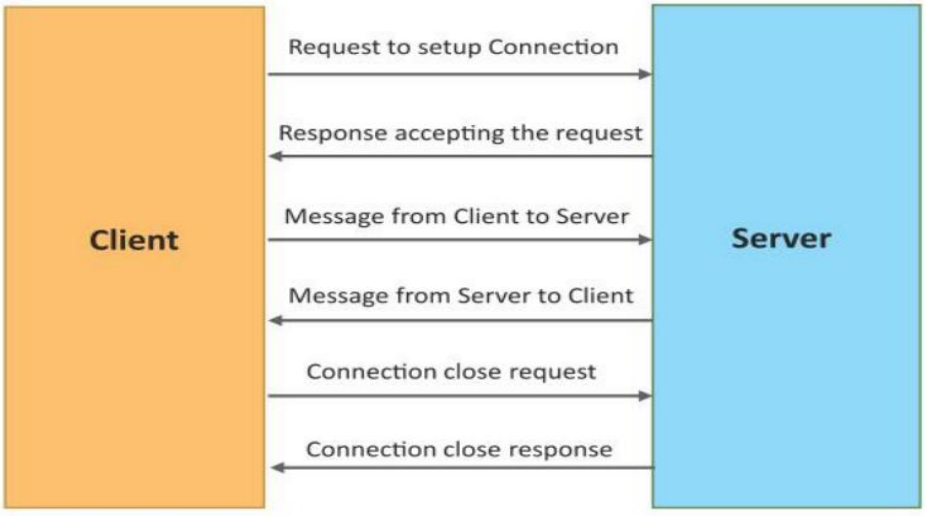
- Push–Pull is a communication model in which the data producers push the data to queues and the consumers pull the data from the queues. Producers do not need to be aware of the consumers.
- Queues help in decoupling the messaging between the producers and consumers.
- Queues also act as a buffer which helps in situations when there is a mismatch between the rate at which the producers push data and the rate at which the consumers pull data.



Push-Pull Communication Model

4. Exclusive Pair Communication Model:

- Exclusive Pair is a bidirectional, fully duplex communication model that uses a persistent connection between the client and the server.
- Once the connection is set up it, remains open until the client sends a request to close the connection.
- Client and server can send messages to each other after connection setup. Fig shows the block diagram of Exclusive Pair Communication Model

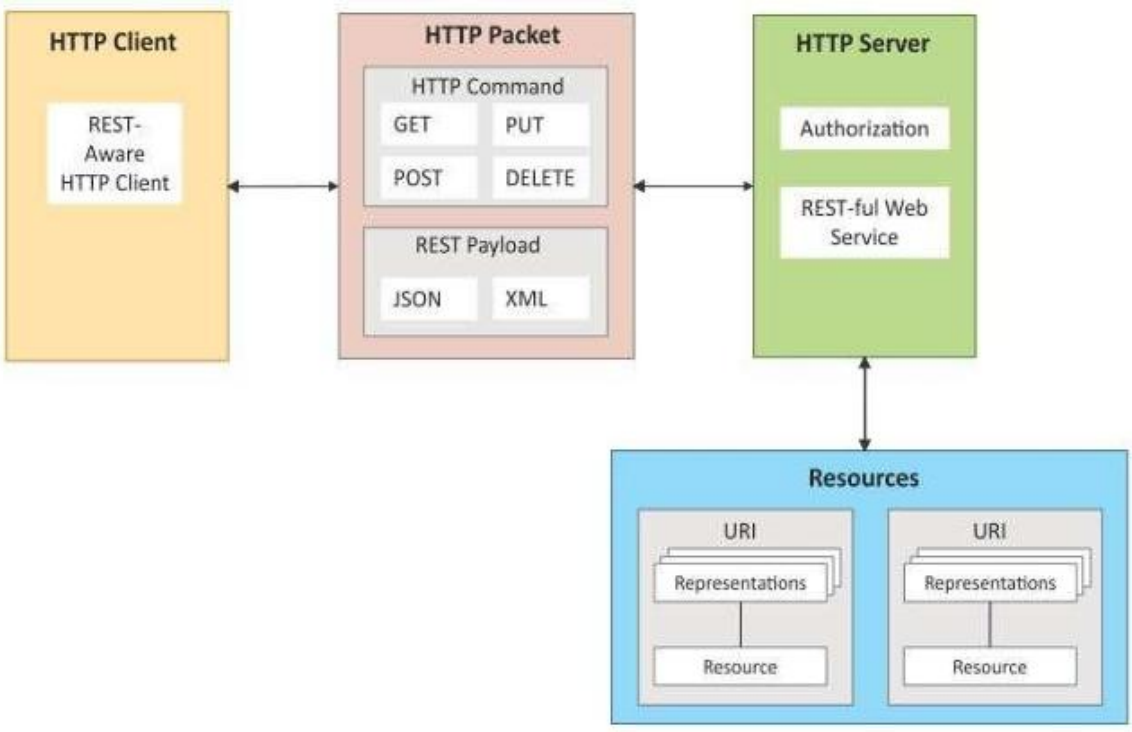


3)IoT Communication APIs:

- REST based communication APIs (Request-Response Based Model)
- WebSocket based Communication APIs (Exclusive Pair Based Model)

a) REST based communication APIs:

- Representational State Transfer (REST) is a set of architectural principles by which you can design web services and web APIs that focus on a system's resources and how resource states are addressed and transferred.
- REST APIs follow the request–response communication model. REST architectural constraints apply to the components, connectors and data elements within a distributed hypermedia system.



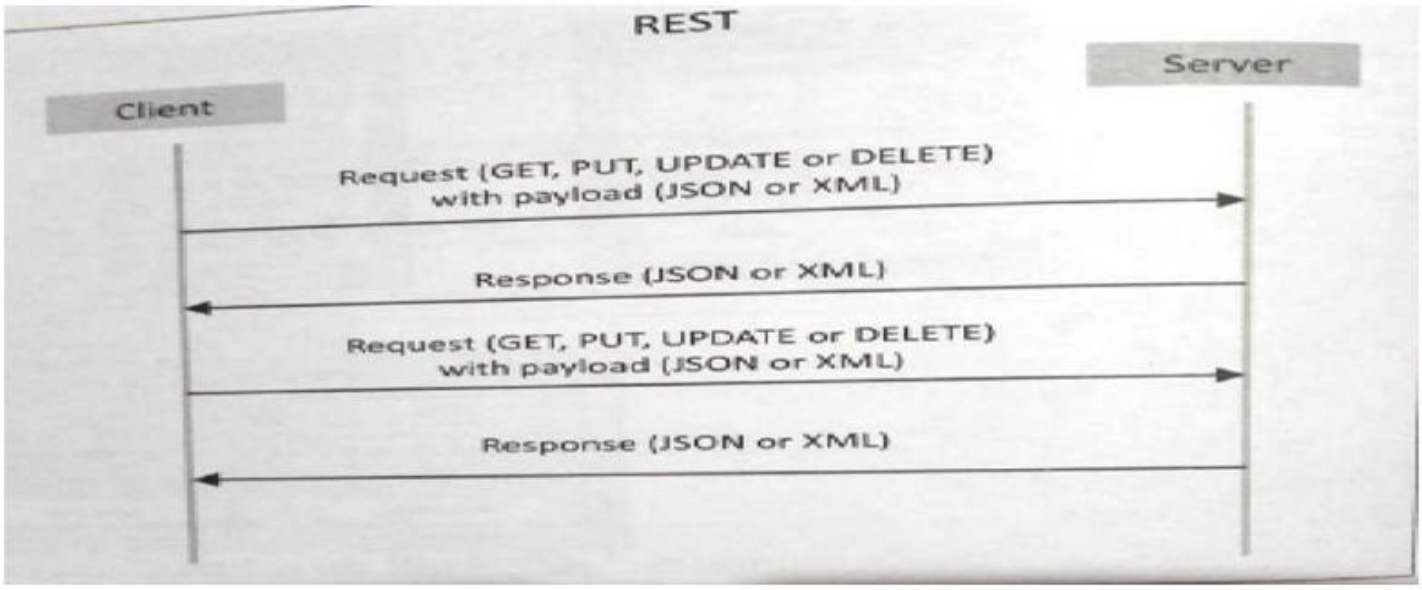
- **Client-Server**: The principle behind client-server constraint is the separation of concerns. Separation allows client and server to be independently developed and updated.
- **Stateless**: Each request from client to server must contain all the info. Necessary to understand the request, and cannot take advantage of any stored context on the server.
- **Cache-able**: Cache constraint requires that the data within a response to a request be implicitly or

explicitly labeled as cache-able or non-cacheable. If a response is cache-able, then a client cache is given the right to reuse that response data for later, equivalent requests.

- **Layered System**: constraints the behavior of components such that each component cannot see beyond the immediate layer with which they are interacting.
- **User Interface**: constraint requires that the method of communication between a client and a server must be uniform.
- **Code on Demand**: Servers can provide executable code or scripts for clients to execute in their context. This constraint is the only one that is optional.

Request-Response model used by REST:

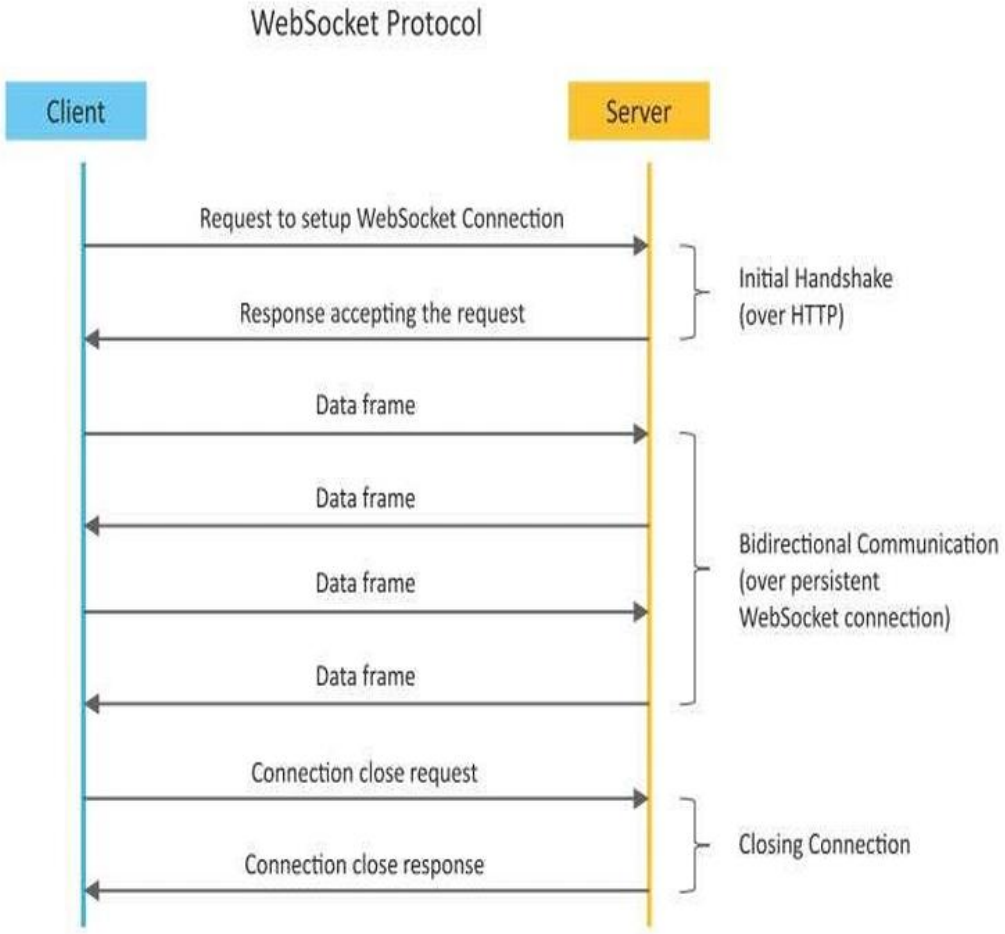
RESTful webservice is a collection of resources which are represented by URIs. RESTful web API has a base URI (e.g: http://example.com/api/tasks/). The clients and requests to these URIs using the methods defined by the HTTP protocol (e.g: GET, PUT, POST or DELETE). A RESTful web service can support various internet media types.



b) WebSocket Based Communication APIs:

- WebSocket APIs allow bi-directional, full duplex communication between clients and servers. WebSocket APIs follow the exclusive pair communication model.
- Unlike request response API allow full duplex communication and do not require new connection to be set up for each message to be sent.

- WebSocket communication begins with connection setup request send by the client to be server. The request is sent over http and the server interprets it as an upgrade request,
- If the server support protocol response to the website handshake response after the connection setup the client and the server can send data or messages to each other in full duplex model.
- WebSocket API reduce network traffic and latency as there is no overhead for connection setup and determination records to each message.



Difference between REST and Web Socket-based Communication APIs:

	WebSockets	HTTP
Communication Model	Full-duplex, bidirectional	Request-response
Connection	Persistent, long-lived connection	Short-lived, stateless connections
Latency	Low latency due to the open connections	Higher latency, connection setup for each request
Data Transfer	Efficient for real-time data streaming	Suited for transferring documents and resources
Overhead	Lower overhead due to fewer handshakes	Higher overhead due to multiple requests/responses
Scalability	Horizontally scalable for a large number of concurrent connections	Scalable, but new connections need to be established for each request
Security	Supports secure connections via WSS (WebSocket Secure)	Supports HTTPS for secure data transmission

S.NO.	REST API	WEB SOCKET API
1.	It is Stateless protocol. It will not store the data.	It is Stateful protocol. It will store the data.
2.	It is Uni-directional. Only either server or client will communicate.	It is Bi-directional. Messages can be received or sent by both server or client.
3.	It is Request-response model.	It is Full duplex model.
4.	HTTP request contains headers like head section, title section.	It is suitable for real-time applications. It does not have any overhead.
5.	New TCP connection will be set up for each HTTP request.	Only Single TCP connection.
6.	Both horizontal and vertical scaling (we can add many resources and number of users both horizontally and vertically).	Only vertical scaling (we can add resources only vertically).
7.	It depends upon the HTTP methods to retrieve the data..	It depends upon the IP address and port number to retrieve the data
8.	It is slower than web socket regarding the transmission of messages.	web socket transmits messages very fastly than REST API.
9.	It does not need memory or buffers to store the data.	It requires memory and buffers to store the data.