LECTURE NOTES

ON

INTERNET OF THINGS

Prepared by

Mukesh K. Sahu

Assistant Professor

School of Computing

Graphic Era Hill University (GEHU)

Haldwani, Uttarakhand-263 139

Syllabus

School of Computing

NAME OF DEPARTMENT:

Subject Name: Internet of Things **Subject Code:** TBC 505

Course Name: Bachelor of Computer Applications (BCA)

1 Contact Hours: 45 **L** 3 **T** 0 **P** 0

2 Examination Duration (Hrs): **Theory** 0 3 **Practical** 0 0

3 Relative Weightage: **CWE:** 25 **MTE:** 25 **ETE:** 50

4 Credits: 0 3

5 Semester:

*		
Autumn	**Spring**	**Both**

6 Pre-Requisite: Basic Electronics Knowledge

7 Subject Area: Electronics

8 Objective: To provide new means to understand the connection between physical devices and Internet

9 Course Outcomes: A student who successfully fulfils the course requirements will be able to:

CO1 Understand the practical aspects of internet of things and study the functional block of IOT.

CO2 Study how IOT and machine to machine communicate.

CO3 Analyse the various design challenges of IOT.

CO4 Design the various applications related projects of IOT.

CO5 Develop some application based on sensors & study various developing tools.

10 Details of the Course:

Unit No.	CONTENT	CONTACT HOURS
1	Introduction to IoT: Defining IoT, Characteristics of IoT, Physical design of IoT, Logical design of IoT, Functional blocks of IoT, Communication models & APIs	9
2	IoT & M2M: Machine to Machine, Difference between IoT and M2M, sensor node and wireless sensor network (WSN), Network & Communication aspects: Wireless medium access issues, MAC protocol survey, Routing protocols, Sensor deployment & Node discovery, Data aggregation & dissemination. Communication protocols, web connectivity using gateway, SOAP, REST, HTTP, IP addressing in IOT, Medium access control.	9
3	Data Acquiring and storage, cloud computing paradigm for data collection, storage and computing, IOT cloud-based services, sensor technology, Industrial IOT and automotive IOT, RFID Technology and wireless sensor network technology (WSN)	9
4	Challenges in IoT: Design challenges, Development challenges, Security challenges, other challenges Home automation, Industry applications, Surveillance applications, Other IoT applications.	9
5	Developing IoTs: Introduction to different IoT tools, developing applications through IoT tools, developing sensor-based application through embedded system platform, Implementing IoT concepts with python. Edge computing, difference between fog computing, edge computing & IoT, prototyping the embedded devices for IOT.	9
	TOTAL	**45**

11 Suggested Books:

Sl. NO.	NAME OF AUTHORS/BOOKS/PUBLISHERS	YEAR OF PUBLICATION
1	Vijay Madisetti, Arshdeep Bahga, “Internet of Things: A Hands-On Approach”	2014
2	Waltenegus Dargie, Christian Poellabauer, "Fundamentals of Wireless Sensor Networks: Theory and Practice	2014

UNIT-I

INTRODUCTION OF IOT

The chapter aims at explaining the fundamental concepts of the "Internet of Things" abbreviated as **IoT**. We humans, as intelligent beings, have always strived for making our lives easier, from the simplest things like a pressure cooker, to state of the art self-driving cars and UAVs used by the military for surveillance, etc. IoT is yet another technological paradigm that is working towards making our lives easier and enriching our experiences and interactions with the things around us. The concept of IoT is a little difficult to grasp because of the diversity of things involved and the plethora of current and possible applications.

Wikipedia defines IoT as "*The Internet of things (IoT) is a system of interrelated computing devices, mechanical and digital machines provided with unique identifiers (UIDs) and the ability to transfer data over a network without requiring human-to-human or human-to computer interaction*."

The definition is given by McKinsey, though small, seems more logical and makes the concept a little more understandable, "*Sensors and actuators embedded in physical objects are linked through wired and wireless networks, often using the same Internet Protocol (IP) that connects the Internet*".

IoT comprises things that have unique identities and are connected to internet. By 2020 there will be a total of 50 billion devices /things connected to internet. IoT is not limited to just connecting things to the internet but also allow things to communicate and exchange data. A dynamic global n/w infrastructure with self-configuring capabilities based on standard and interoperable communication protocols where physical and virtual things have identities, physical attributes and virtual personalities and use intelligent interfaces, and are seamlessly integrated into information n/w, often communicate

data associated with users and their environments.

Characteristics:

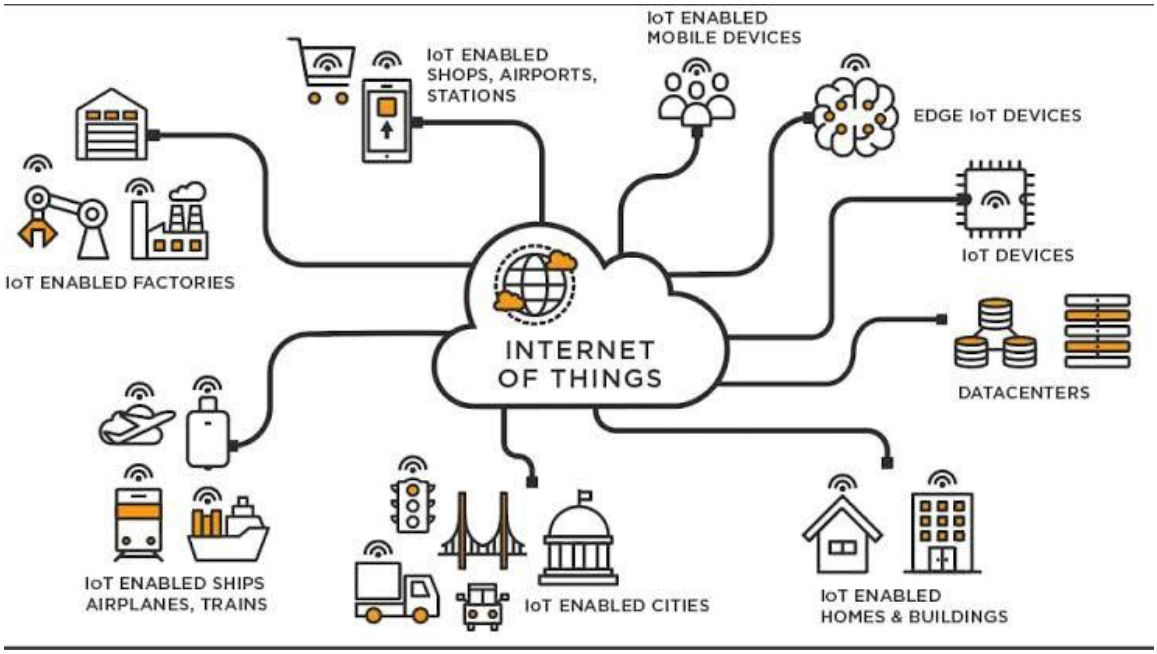
1) **Dynamic nature:** The primary activity of Internet of Things is to collect data from its environment, this is achieved with the dynamic changes that take place around the devices. The state of these devices change dynamically, example sleeping and waking up, connected and/or disconnected as well as the context of devices including temperature, location and speed. In addition to the state of the device, the number of devices also changes dynamically with a person, place and time.
2) **Self-Adapting**: IoT devices and systems may have the capability to dynamically adapt with the changing contexts and take actions based on their operating conditions, user ‘s context or sensed environment.
3) **Eg**: the surveillance system is adapting itself based on context and changing conditions.
4) **Self-configuring:** allowing a large number of devices to work together to provide certain functionality. IoT comes with the combination of algorithms and computation, software & hardware that makes it smart. Self-configuring in IoT enhances its capabilities which facilitate the things to respond in an intelligent way to a particular situation and supports them in carrying out specific tasks.
5) **Heterogeneity:** Heterogeneity in Internet of Things as one of the key characteristics. Devices in IoT are based on different hardware platforms and networks and can interact with other devices or service platforms through different networks. IoT architecture should support direct network connectivity between heterogeneous networks. The key design requirements

for heterogeneous things and their environments in IoT are scalabilities, modularity, extensibility and interoperability.

6) **Unique Identity:** Each IoT device has a unique identity and a unique identifier (IP address).
7) **Integrated into Information Network** that allow them to communicate and exchange data with other devices and systems.

Applications of IoT:

1) Home
2) Cities
3) Environment
4) Energy
5) Retail
6) Logistics
7) Agriculture
8) Industry



History of IoT

In the world of technology, one would consider the IoT as something old. Even so, the thought or vision of machines communicating with one another can be found in the works of people like Nikola Tesla's interview with Colliers in 1926[5] and literature from the early 1800s. In a way, machines can be said to be in contact since the first Telegraph in the mid-1800s. In 1900, the first Voice over Radio transmission (Wireless telegraphy) and many more inventions and innovations in the field of communication and electronics followed, laying the foundation for IoT. Amidst all the rapid advancements, the development of computers began in the 1950s. The Internet, a significant component of the IoT, started as part of a military defense project called DARPA in 1962 and evolved into ARPANET in 1969. Commercial service providers started supporting public use of ARPANET around the 1980s, which allowed the modern Internet to come into existence. Internet and communication technologies were quintessential for IoT's development. IPV6's decision to increase address space was another important component in developing a functional IoT.

Realizing the Concept The term IoT was coined in 1999 by **Kevin Ashton**, but the technology came into existence even before that. One of the first examples was a Coca Cola machine from the early 1980s [6], located at the Carnegie Melon University. Programmers there hooked up the refrigerator with some sensors and connected it to the internet so they could keep tabs on the drinks in it, preventing them from futile trips to the vending machine. Then came the toaster presented by the John Romkey at Interop in 1990[7]. It was a toaster connected to the internet that could be turned on and off from a computer. There are some pieces of evidence of the existence of elevators that could be wirelessly controlled from 1979[8], but it isn't very clear so the Cola vending machine is considered the first IoT implementation. The first corporate attention

to IoT is of LG when they announced the first internet refrigerator back in 2000. From their IoT picked up paces, and started appearing in main-stream media like The Guardian, Scientific American, etc around 2003. UN's International Telecommunications Union, ITU, published its first report on the topic *"A new dimension has been added to the world of information and communication technologies (ICTs): from any time, any place connectivity for anyone, we will now have connectivity for anything. Connections will multiply and create an entirely new dynamic network of networks – an Internet of Things"* in the year 2005. IoT rose in recognition and potential from their onwards. By 2013, IoT evolved into a system engaging many different technologies, like wireless communication, micro-electromechanical systems (MEMS), embedded systems. The support of IoT nowadays is present in nearly all automation fields from home to industry, present-day sensor networks, control systems, etc. Health & Life Style

Physical Design of IoT

Physical Design of IoT refers to IoT Devices and IoT Protocols. Devices are physical electronic components that are used to build a connection to process data, provide interfaces that offer storage, graphics and storage and also power source sometimes in the IoT system. Most of these physical components collect data from the environment and send raw data to be processed and analyzed. After analyzing it sends the information to the actuators to act accordingly. Instead of collecting data from the environment some IoT also collects user data to provide more refined performance to the users.

1) Things in IoT:

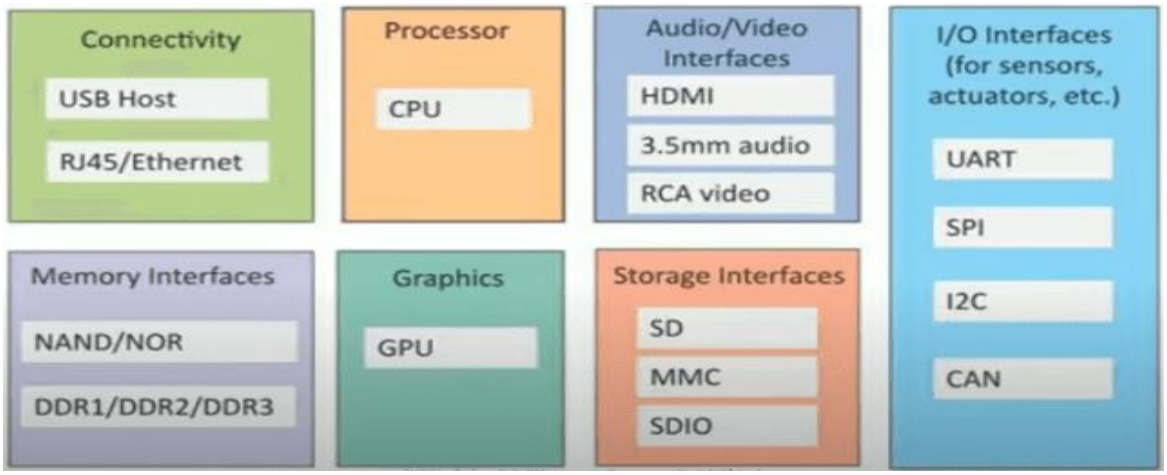
Basically, Things refers to IoT Devices which have unique identities and can perform remote sensing, actuating and monitoring capabilities. Things are is main part of IoT Application. IoT Devices can be various type, Sensing Devices, Smart Watches, Smart Electronics appliances, Wearable Sensors,

Automobiles, and industrial machines. These devices generate data in some forms or the other which when processed by data analytics systems leads to useful information to guide further actions locally or remotely.

For example, Temperature data generated by a Temperature Sensor in Home or other place, when processed can help in determining temperature and take action according to users. Above picture, shows a generic block diagram of IoT device. It may consist of several interfaces for connections to other devices. IoT Device has I/O interface for Sensors, Similarly for Internet connectivity, Storage and Audio/Video. IoT Device collect data from on-board or attached Sensors and Sensed data communicated either to other device or Cloud based sever. Today many cloud servers available for especially IoT System. These Platform known as IoT Platform. Actually, these cloud especially design for IoT purpose. So here we can analysis and processed data easily.

The things in IoT refers to IoT devices which have unique identities and perform remote sensing, actuating and monitoring capabilities. IoT devices can exchange data with other connected devices applications. It collects data from other devices and process data either locally or remotely.

An IoT device may consist of several interfaces for communication to other devices both wired and wireless. These includes I/O interfaces for sensors, Processor, Interfaces for internet connectivity, memory, storage interfaces and audio/video interfaces.



Connectivity

Devices like USB hosts and ETHERNET are used for connectivity between the devices and the server.

Processor

A processor like a CPU and other units are used to process the data. these data are further used to improve the decision quality of an IoT system.

Audio/Video Interfaces

An interface like HDMI and RCA devices is used to record audio and videos in a system.

Input/Output interface

To give input and output signals to sensors, and actuators we use things like UART, SPI, CAN, etc.

Storage Interfaces

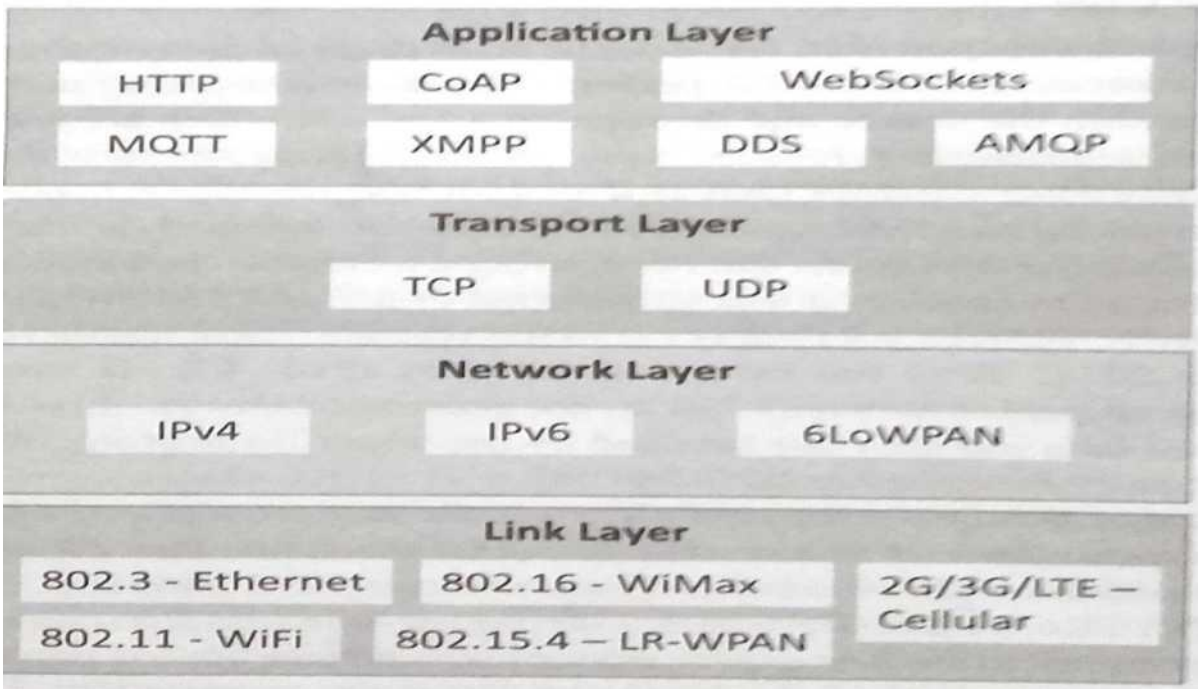
Things like SD, MMC, and SDIO are used to store the data generated from an IoT device.

Other things like DDR and GPU are used to control the activity of an IoT system.

2) IoT Protocols:

a) Link Layer: Protocols determine how data is physically sent over the networks physical layer

or medium. Local network connect to which host is attached. Hosts on the same link exchange data packets over the link layer using link layer protocols. Link layer determines how packets are coded and signaled by the h/w device over the medium to which the host is attached.



Protocols:

- 802.3-Ethernet: IEEE802.3 is collection of wired Ethernet standards for the link layer. Eg: 802.3 uses co-axial cable; 802.3i uses copper twisted pair connection; 802.3j uses fiber optic connection; 802.3ae uses Ethernet over fiber.
- 802.11-WiFi: IEEE802.11 is a collection of wireless LAN (WLAN) communication standards including extensive description of link layer. Eg: 802.11a operates in 5GHz band, 802.11b and 802.11g operates in 2.4GHz band, 802.11n operates in 2.4/5GHz band, 802.11ac operates in 5GHz band, 802.11ad operates in 60Ghzband.
- 802.16 - WiMax: IEEE802.16 is a collection of wireless broadband standards including exclusive description of link layer. WiMax provide data rates from 1.5 Mb/s

to 1Gb/s.

- 802.15.4-LR-WPAN: IEEE802.15.4 is a collection of standards for low rate wireless personal area network (LR-WPAN). Basis for high level communication protocols such as ZigBee. Provides data rate from 40kb/s to250kb/s.
- 2G/3G/4G-Mobile Communication: Data rates from 9.6kb/s(2G) to up to100Mb/s(4G).

B) **Network/Internet Layer:** Responsible for sending IP datagrams from source n/w to destination n/w. Performs the host addressing and packet routing. Datagrams contains source and destination address.

Protocols:

- IPv4: Internet Protocol version4 is used to identify the devices on a n/w using a hierarchical addressing scheme. 32-bit address. Allows total of 2^{32} addresses.
- **IPv6:** Internet Protocol version6 uses 128-bit address scheme and allows 2^{128} addresses.
- 6LOWPAN: **(IPv6 over Low power Wireless Personal Area Network) operates in 2.4 GHz frequency range and data transfer 250 kb/s.**

C) **Transport Layer:** Provides end-to-end message transfer capability independent of the underlying n/w. Set up on connection with ACK as in TCP and without ACK as in UDP. Provides functions such as error control, segmentation, flow control and congestion control.

Protocols:

- **TCP:** Transmission Control Protocol used by web browsers (along with HTTP and

HTTPS), email (along with SMTP, FTP). Connection oriented and stateless protocol. IP Protocol deals with sending packets, TCP ensures reliable transmission of protocols in order. Avoids n/w congestion and congestion collapse.

- **UDP:** User Datagram Protocol is connectionless protocol. Useful in time sensitive applications, very small data units to exchange. Transaction oriented and stateless protocol. Does not provide guaranteed delivery.

D) **Application Layer:** Defines how the applications interface with lower layer protocols to send data over the n/w. Enables process-to-process communication using ports.

Protocols:

- **HTTP:** Hyper Text Transfer Protocol that forms foundation of WWW. Follow request- response model Stateless protocol.
- **CoAP:** Constrained Application Protocol for machine-to-machine (M2M) applications with constrained devices, constrained environment and constrained n/w. Uses client- server architecture.
- **WebSocket:** allows full duplex communication over a single socket connection.
- **MQTT:** Message Queue Telemetry Transport is light weight messaging protocol based on publish-subscribe model. Uses client server architecture. Well suited for constrained environment.
- **XMPP:** Extensible Message and Presence Protocol for real time communication and streaming XML data between network entities. Support client-server and server-server communication.
- **DDS:** Data Distribution Service is data centric middleware standards for device-to-

device or machine-to-machine communication. Uses publish-subscribe model.

- **AMQP:** Advanced Message Queuing Protocol is open application layer protocol for business messaging. Supports both point-to-point and publish-subscribe model.

LOGICAL DESIGN of IoT

Refers to an abstract represent of entities and processes without going into the low level specifies of implementation. The "Logical Design" of IoT is the framework or the imaginary ideal design in which the components including software and the hardware components will be laid out. It doesn't go into the depth of describing how each component will be built with low-level programming specifics.

1) IoT Functional Blocks

2) IoT Communication Models

3) IoT Comm. APIs

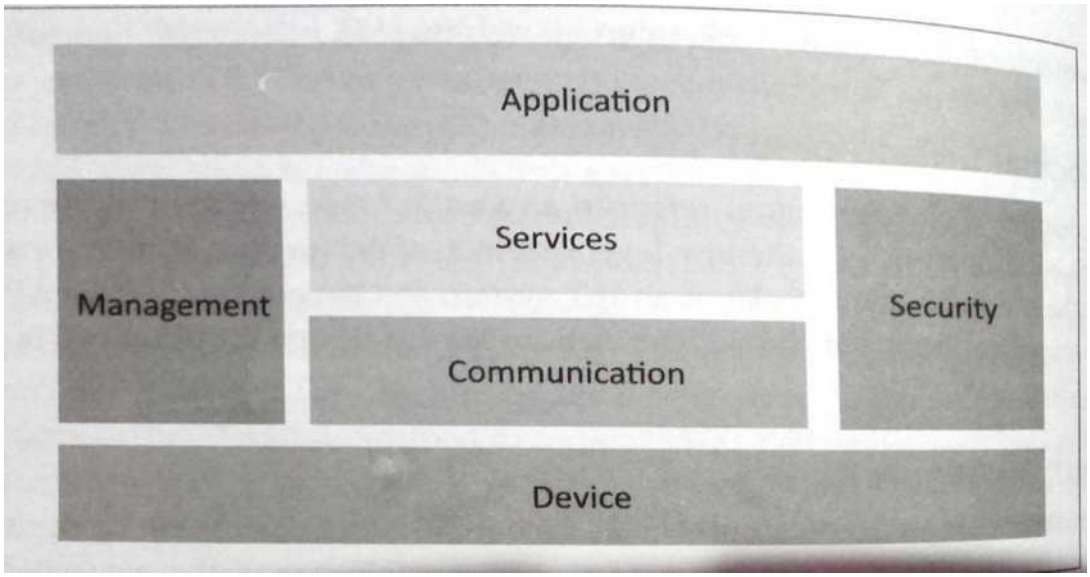
1) IoT Functional Blocks:

What is a functional block?

An IoT system consists of a number of functional blocks that provide the system with the capabilities for identification, sensing, actuation, communication and management. The function of the Communication functional block in short Handles the communication for the IoT system.

IoT systems include several functional blocks such as Devices, communication, security, services, and application. The functional blocks provide sensing, identification, actuation, management, and communication capability. These functional blocks consist of devices that handle the communication between the server and the host, enable monitoring control

functions, manage the data transfer, secure the IoT system using authentication and different functions, and provide an interface for controlling and monitoring various terms. Provide the system the capabilities for identification, sensing, actuation, communication and management.



- **Device**: An IoT system comprises of devices that provide sensing, actuation, monitoring and control functions.
- **Communication or connectivity:** handles the communication for IoT system. The communication block contains different protocols (wired or wireless) through which the data moves from devices to Internet and from Internet to devices. The services block acts as a middleware which can provide services like device identification, device discovery, or data processing and analysis.
- **Services:** for device monitoring, device control services, data publishing services and services for device discovery.
- **Management:** Provides various functions to govern the IoT system. The management block allows us to manage the other blocks like device, services, communication, application, and

security.

- **Security:** Secures IoT system and priority functions such as authentication authorization, message and context integrity and data security. The security block provides different security services like confidentiality, integrity, availability, and authentication.
- **Application:** IoT application provide an interface that the users can use to control and monitor various aspects of IoT system. The users of an IoT application interacts with the application block. It provides user interface with which the user can access the data sent by the sensors, perform operations on that like aggregation, simplification, etc. and visualize that data. The application interface can also provide control functions for controlling the sensors, actuators or functionality of the application.

2) IoT Communication Models:

1) Request-Response

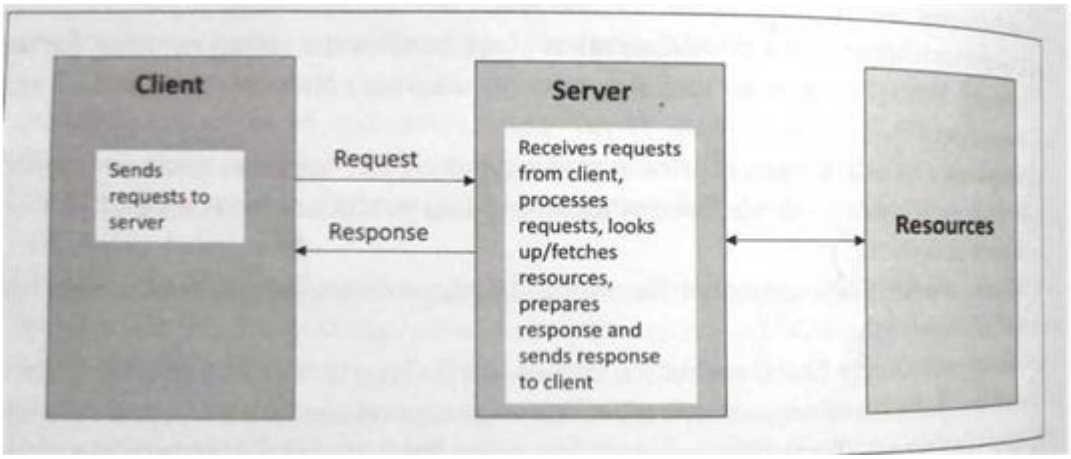
2) Publish-Subscribe

3)Push-Pull

4) Exclusive Pair

1. **Request-Response Model:**

The request-response model can be illustrated as shown below:



The two main entities in request-response model are client and server. The client can be a web application, mobile application, etc. The client may be a browser requesting web pages or accessing an email. Each access of a resource will be treated as request. The server accepts the requests from the clients, processes them and sends back responses to the clients. While processing the requests, the server might access additional resources like file, databases, etc. The request-response model is stateless, i.e., the requests in a session are not related to each other with respect to a server. Each request is treated as a new one and is completely unrelated to the previous requests.

Example is HTTP. HTTP operates as a query-response protocol between a client and a server. A web browser can be the client, and an application on a computer that supports a website can be the server. The client(browser) submits an HTTP request to the server and the server will return a response back to the client.

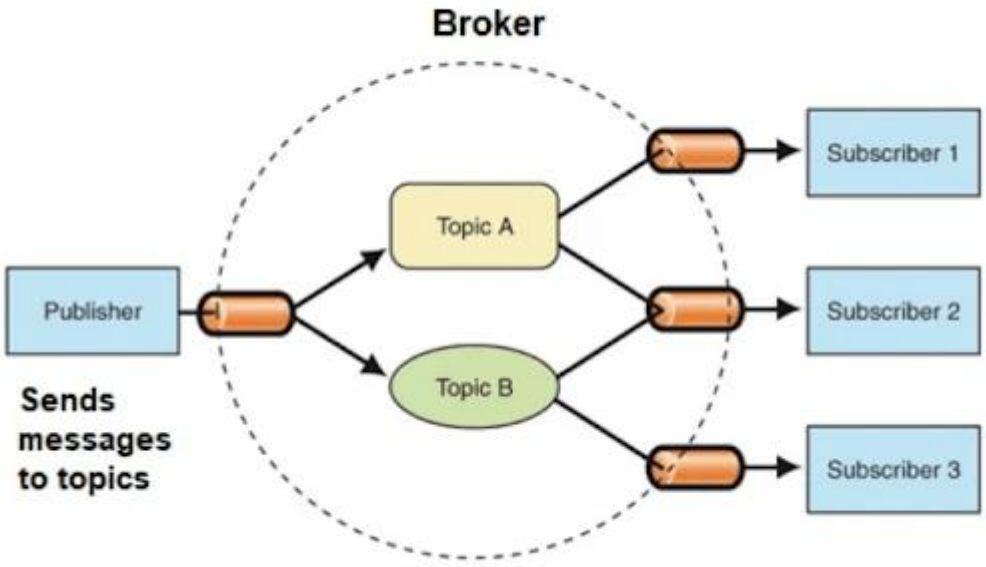
2. **Publish-Subscribe Model** The three main entities in publish-subscribe model are publisher, consumer, and broker.

Publishers are the source of data. It sends the data to the topic which are managed by the broker. They are not aware of consumers.

Consumers subscribe to the topics which are managed by the broker.

Brokers responsibility is to accept data from publishers and send it to the appropriate consumers. The broker only has the information regarding the consumer to which a particular topic belongs to which the publisher is unaware of.

The publisher always publishes messages at a pre-defined interval. The sensors in IoT can be thought of as publishers. The publishers publish data as topics. The broker is an entity that maintains different topics to which consumers subscribe. The broker is generally a server. The published messages of publishers are maintained by the broker. The consumers consume the data published by the publishers. A consumer is generally the IoT application through which the users interact. A consumer can subscribe to one or more topics maintained by a broker.

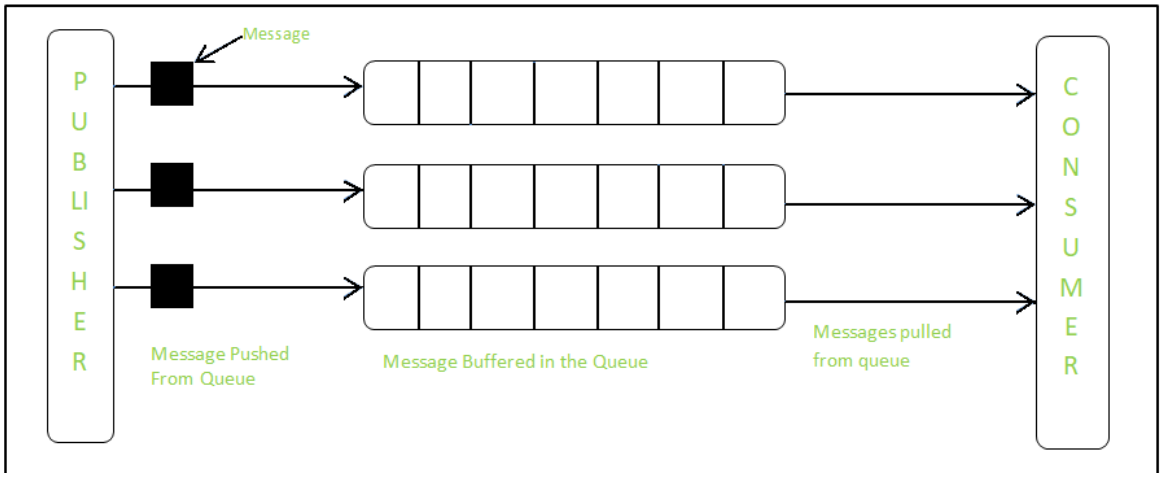


Involves publishers, brokers and consumers. Publishers are source of data. Publishers send data to the topics which are managed by the broker. Publishers are not aware of the consumers. Consumers subscribe to the topics which are managed by the broker. When the broker receives data for a topic from the publisher, it sends the data to all the subscribed consumers.

Example(s):- MQTT (Message Queue Telemetry Transport), AMQP (Advanced Message Queue Protocol), DDS (Data Distribution Service).

3. **Push-Pull Model:** in which data producers push data to queues and consumers pull data from the queues. Producers do not need to aware of the consumers. Queues help in decoupling the message between the producers and consumers. The push-pull model constitutes data publishers, data consumers, and data queues. Publishers and Consumers are not aware of each other.

   Publishers publish the message/data and push it into the queue. The consumers, present on the other side, pull the data out of the queue. Thus, the queue acts as the buffer for the message when the difference occurs in the rate of push or pull of data on the side of a publisher and consumer.



Queues help in decoupling the messaging between the producer and consumer. Queues also act as a buffer which helps in situations where there is a mismatch between the rate at which the producers push the data and consumers pull the data.

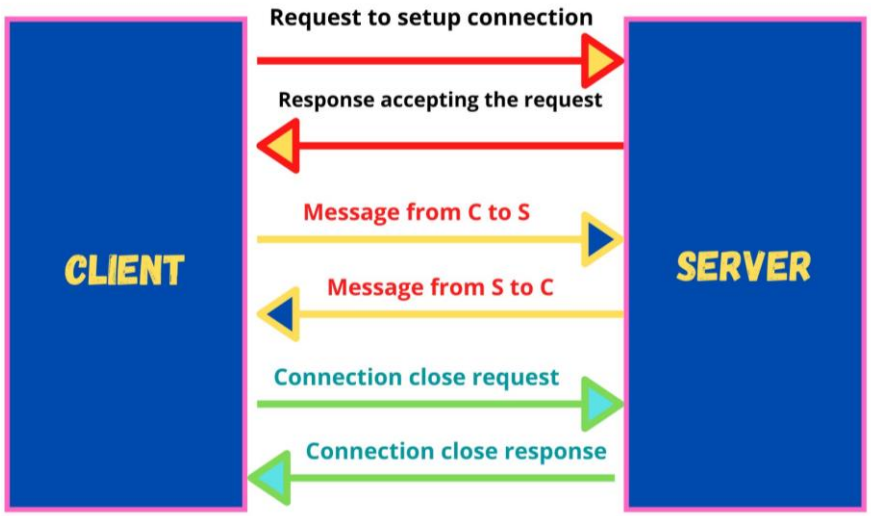
At times, there might be some mismatch in the push-pull rates. Queues act as a buffer which helps in situations when there is a mismatch between the rate at which the producers push data and the rate at which the consumer pull data. So they work as a buffer and flow control mechanisms whenever there is any mismatch in the push-pull rates.

4. **Exclusive Pair**

Exclusive Pair is the bi-directional model, including full-duplex communication among client and server. The connection is constant and remains open till the client sends a request to close the connection.

The Server has the record of all the connections which has been opened. This is a state-full connection model and the server is aware of all open connections. WebSocket based communication API is fully based on this model.

WebSocket based communication API is fully based on this model.


Request to setup connection
Response accepting the request
Message from C to S
Message from S to C
Connection close request
Connection close response
CLIENT
SERVER

3) IoT Communication APIs:

a) REST based communication APIs (Request-Response Based Model)

b) 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 we can design web services and web APIs that focus on a system ‘s resources and have resource states are addressed and transferred.

The REST architectural constraints: Fig. shows communication between client server with REST APIs.

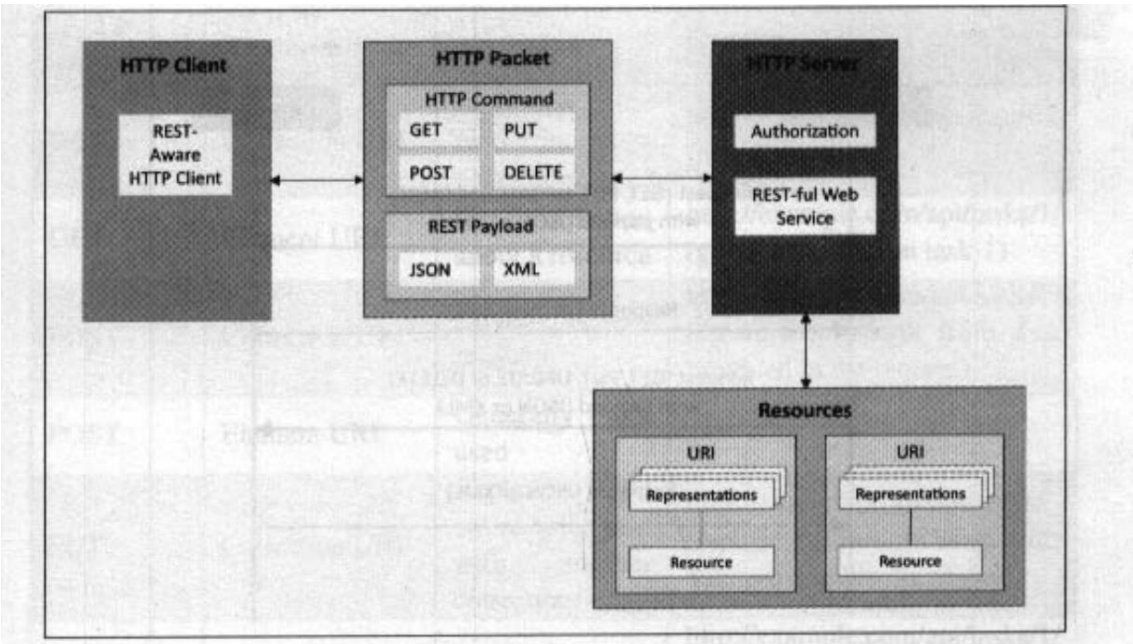


Figure 1.11: Communication with REST APIs

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:

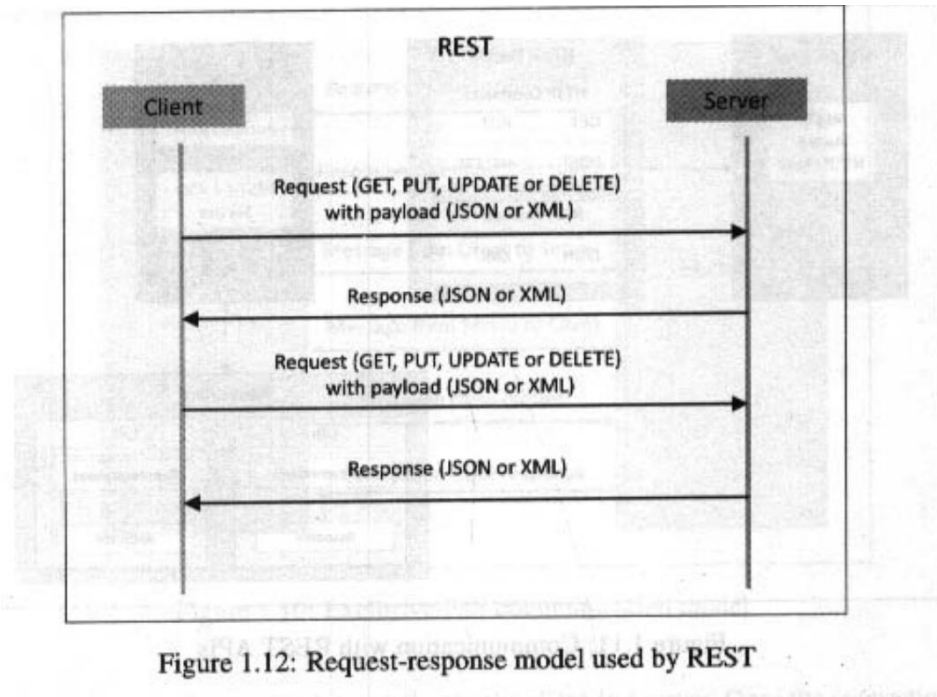


Figure 1.12: Request-response model used by REST

RESTful web service 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**: WebSocket Based Communication APIs: WebSocket APIs allow bi-directional, full duplex communication between clients and servers. WebSocket APIs follow the exclusive pair.

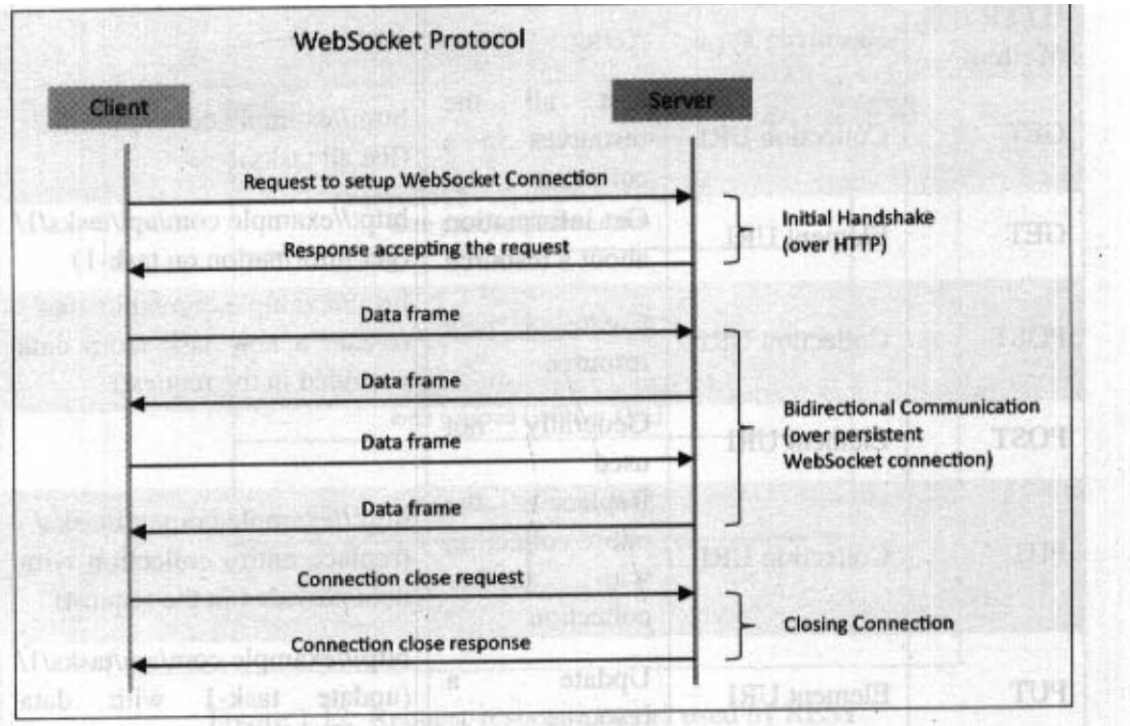


Figure 1.13: Exclusive pair model used by WebSocket APIs

The **WebSocket** enables communication between clients and a remote server. It provides a security model similar to the one used in Web browsers. It executed over TCP and is adapted for applications that use browsers and must interact with remote hosts. WebSocket runs on the TCP socket for web services and browsers, however it does not implement reliability mechanisms on its own. If necessary, securing of sessions can be done using the WebSocket

on TLS/SSL. WebSocket messages contain 2 bytes of overload during all the time of the session as in indicated by the studies, the header information is repeated on HTTP polling (in REST) as increasing in data transmission rate, latency increased too. Compared, to half-duplex HTTP polling, WebSocket is supposed to reduce latency to tree-to-one. WebSocket is not recommended for limited resource devices and its architecture is not suitable for IOT applications. This pit that, it's secure, perfect for real-time communication and it can used for good messaging systems if used with WAMP. So, we can consider it among the best protocols working on TCP.