Module-3

IP as the IOT Network Layer :-

The Business Case for IP

Data flowing from or to "things" is consumed controlled, or monitored by data center servers either in the cloud or in locations that may be distributed or centralized.

- Dedicated applications are then run over virtualized or traditional operating systems or on network edge platforms ; (for eg- fog computing)
- The system solutions combining various physical and data link layers call for an architectural approach with a common layer(s) independent from the lower (connectivity) and/or upper (application) layers. This is how and why the Internet Protocol (IP) suite started playing a key architectural role in the early 1990's.

The key Advantages of Internet Protocol :-

1) Open and Standards-based :- Operational technologies have often been delivered as turnkey features by vendors who may have optimized the communications through closed and proprietary networking solutions.

Versatile : A large spectrum of access technologies is available to offer connectivity of "things" in the last mile. Additional protocols and technologies are also used to transport IOT data through backhaul links and in the data center.

Ubiquitous :- All recent operating system releases, from general-purpose computers and servers to lightweight embedded systems. (Tinyos, Contiki, so on), have an integrated dual (IPv4 and IPv6) IP stack that get enhanced over time.

4] Scalable :- As the common protocol of the Internet, IP has been massively deployed and tested for robust scalability.

5] Manageable and highly secure : Communicatio infrastructure requires appropriate management and security capabilities for proper operations. Well-known network security management tools are easily leveraged with an IP network layer.

6] Stable and resilient : IP has a large and well-estab knowledge base and, more importantly, it has been used for years in critical infrastructures, such as financial an defense networks.

7] Consumer market adoption : When developing IoT solutions and products targeting the consumer market, vendors know that consumers access to applications and devices will occur predominantly over broadband and mobile wireless infrastructure

8] The innovation factor : The past two decades ha largely established the adoption of IP as a factor for increased innovation. IP is a standards-based protocol that is ubiquitous, scalable, version, isolation and stable.

⇒ Adoption or Adaptation of the Internet Protocol

The use of numerous network layer protocols in addition to IP is often a point of contention between computer networking experts.

→ Adaptation means application layered gateways (ALGs) must be implemented to ensure the translation between non-IP and IP layers.

Adoption involves replacing all non-IP layers with their IP layer counterparts, simplifying the deployment model and operation.

Supervisory control and data acquisition (SCADA) applications are typical examples of vertical market deployments that operate both the IP adaption model and the adoption model.

- We should consider the following factors when trying to determine which model is best suited for last-mile connectivity.

→ Bidirectional Versus unidirectional data flows:

While bidirectional communications are generally expected, some last-mile technologies offer optimization for unidirectional communications

For example - different classes of IOT devices, as defined in RFC 7228.

If there is only one-way communication to upload data to an application, then it is not possible to download new software or firmware to the device.

→ Overhead for last-mile communications paths:-

IP adoption implies a layered architecture with a per-packet overhead that varies depending on the IP version.

This same consideration applies to control plan traffic that is run over IP for low-bandwidth, last mile links. Routing protocol and other verbose network services may be either not be required or call for optimization.

→ Data flow model :- One benefit of the IP adoption model is the end-to-end nature of communications. Any node can easily exchange data with any other node in a network although security, privacy, and other factors may put controls and limits on the end-to-end concept.

The Need for Optimization:-

→ Constrained Nodes :-

Depending on its functions in a network, a "thing" architecture may or may not offer similar characteristics compared to a generic PC or server in an IT environment.

- IoT constrained nodes can be classified as follows:-

* Devices that are very constrained in resources, may communicate infrequently to transmit a few bytes, and may have limited security and management capabilities. (where the nodes communicate with the gateways & proxies)
* Devices with enough power and capacities to implement a stripped-down IP stack or non-IP stack (adaptation) communicate directly (gateways & proxies)
* Devices that are similar to generic PCs in terms of computing and power resources but have constrained networking capacities, such as bandwidth.

→ Constrained Networks:-

Network bandwidth capacity was restrained due to technical limitations. connections often depended on low-speed modems for transferring data.

- Constrained networks have unique characteristics and requirements. In contrast with typical IP networks, where highly stable and fast links are available, constrained networks are limited by low-power, low-bandwidth links.
- They operate between a few kbps and a few hundered kbps and may utilize a star, mesh, or combined network topologies, ensuring proper operations.
- finally, constrained nodes and networks pose major challenges for IoT connectivity in the last mile.

IP Versions → Internet Engg Task force

The IETF has been working on transitioning the Internet from IP version 4 to IP version 6.

The main driving force has been the lack of address space in IPV4 as the Internet has grown. IPv6 has a much larger range of addresses that should not be exhausted for the foreseeable future.

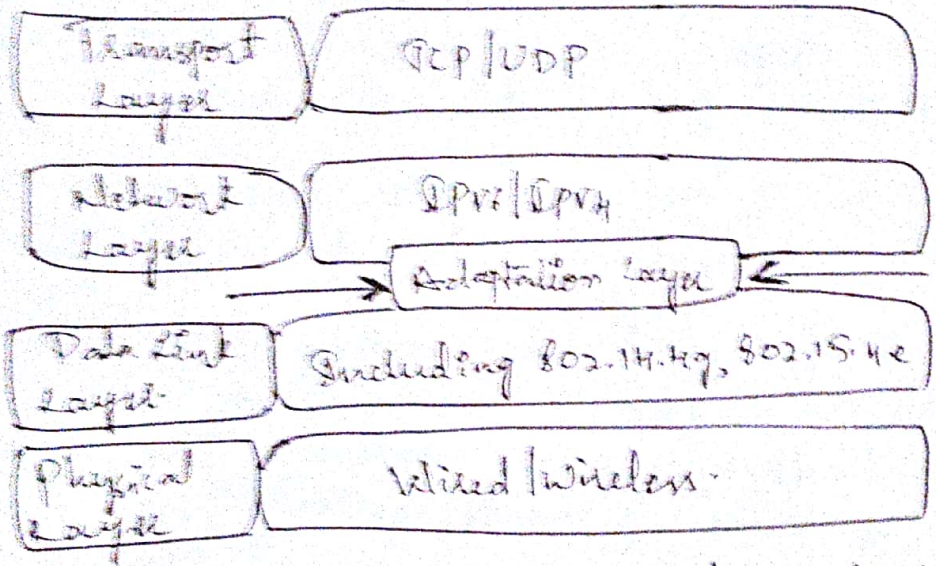
The following are some of the main factors applicable to IPV4 and IPV6 support in an IOT solution:

- **Application Protocol:** IOT devices implementing Ethernet or Wi-Fi interfaces can communicate over both IPV4 and IPV6, but the application Protocol may decide the choice of the IP version.
- **Cellular Provider and Technology:-** IOT devices with cellular modems are dependent on the generation of the cellular technology as well as the data services offered by the provider.
- **Serial communications:-** Many legacy devices in certain industries, such as manufacturing and utilities communicate through serial lines. Data is transferred using either proprietary or standards-based protocols etc.
- **IPV6 Adaptation Layer:-** IPV6-only adaption layers for same physical and data link layers for recently standardized IOT protocols support only IPV6.

While the most common physical and data link layers stipulate adaptation layer for both versions, newer technologies, such as IEEE 802.15.4, IEEE 1901.2, and ITU G.9903 etc.

Optimizing IP for IoT :-

Eg 6LoWPAN : optimizing IP for IoT Using an Adaptation Layer.



The Internet Protocol is key for a successful Internet of Things, constrained nodes and constrained networks mandate optimization at various layers and on multiple protocols of the IP architecture.

⇒ From 6LOWPAN to 6Lo

In the IP architecture, the transport of IP packets over any given Layer 1 (PHY) and Layer 2 (MAC) Protocol must be defined and documented. The model for partoping IP into lower-layer protocols is often referred to as an adaptation Layer.

The main examples of adaptation layers optimized for constrained nodes of "things" are the ones under the 6LoWPAN working group name and its successor, the 6Lo working group.

The initial focus of the 6LoWPAN wd working group was to optimize the transmission of IPv6 packets over constrained networks such as IEEE 802.15.4.

Figure (6) :- 6LoWPAN Header Stacks.

802.15.4 Header	IPV6 Header compression.	IPV6 Payload.

802.15.4 Header.	Fragment Header.	IPV6 Header compression	IPV6 payload.

802.15.4 Header.	Mesh Addressing Header.	Fragment Header	IPV6 Header compression	IPV6 Payload

The above figure (6) shows the subheaders related to compression, fragmentation and mesh addressing.

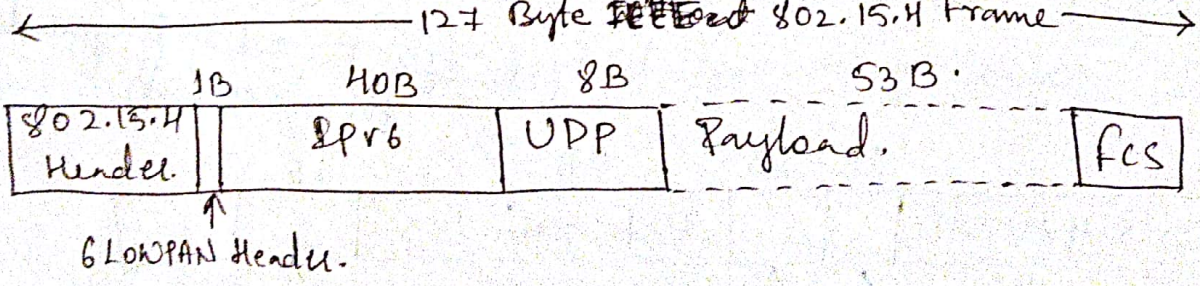
⇒ Header Compression :- .

6LoWPAN header compression is stateless, and conceptually it is not too complicated.

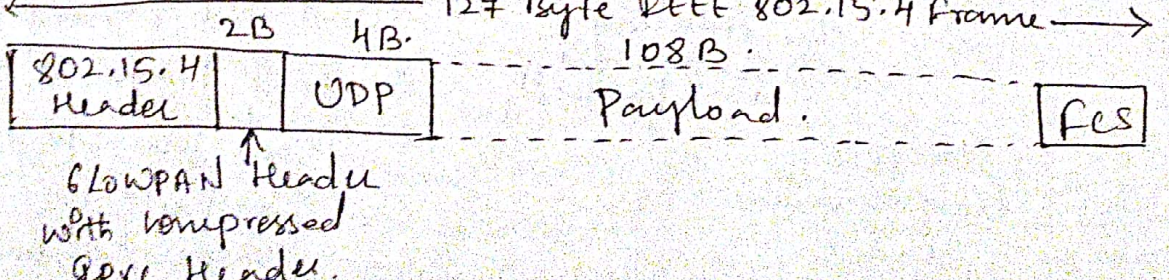
- A number of factors affect the amount of compression, such as implementation of RFC 4944 versus RFC 6922, whether UDP is included and various IPV6 addressing scenarios.

Figure (7) :- 6LoWPAN Header compression. :-

6LoWPAN without Header compression.



6LoWPAN with IPV6 and UDP Header compression



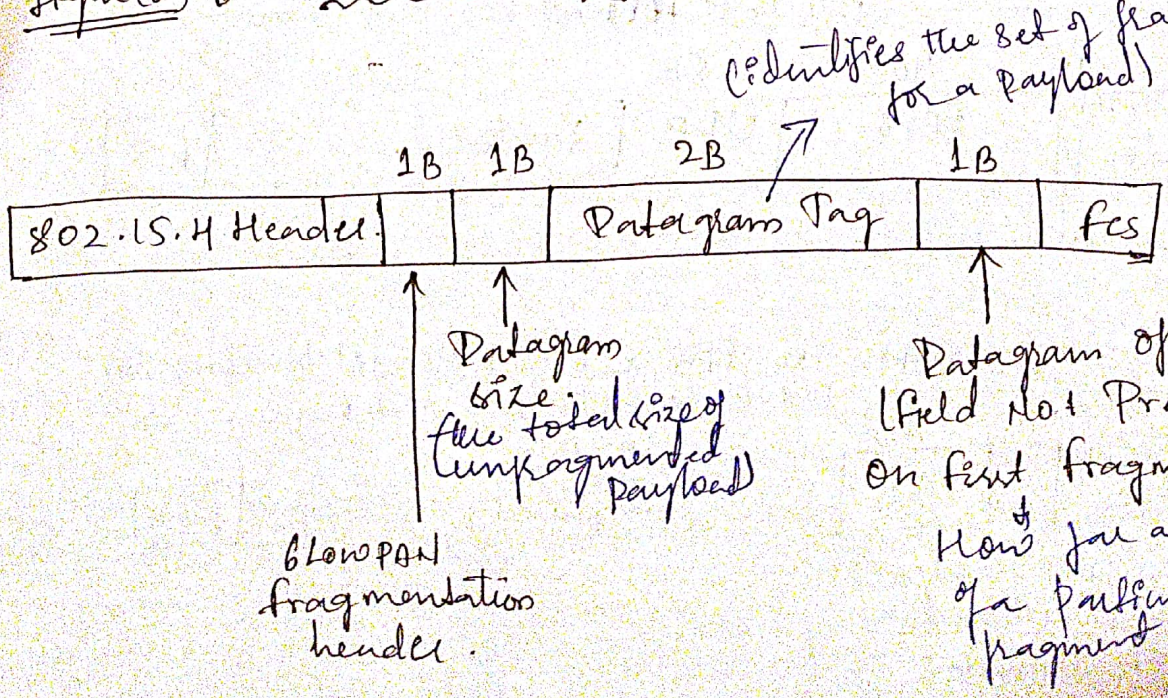
- In the figure (c), we can see a 6LOWPAN frame without any header compression enabled: The full 40-byte IPv6 header and 8-byte UDP header are visible.
- The 6LOWPAN header is only a single byte in the above half section. Notice that the UDP headers leave only 53 bytes of data payload out of the 127-byte maximum frame size in the case of IEEE 802.15.4.
- In the next bottom half figure (c), shows which frame where header compression has been enabled for a best-case scenario.
- Most importantly, the header compression has allowed the payload to more than double, from 53 bytes to 108 bytes, which is obviously much more efficient.

⇒ Fragmentation :-

The maximum transmission unit (MTU) for an IPv6 network must be at least 1280 bytes.

- The term MTU defines the size of the largest protocol data unit that can be passed.

Figure (d) :- 6LOWPAN fragmentation Header



→ In the figure (d), the 6LOWPAN fragmentation header field itself uses a unique bit value to identify that the subsequent fields behind it are fragment fields as opposed to another capability such as header compression.

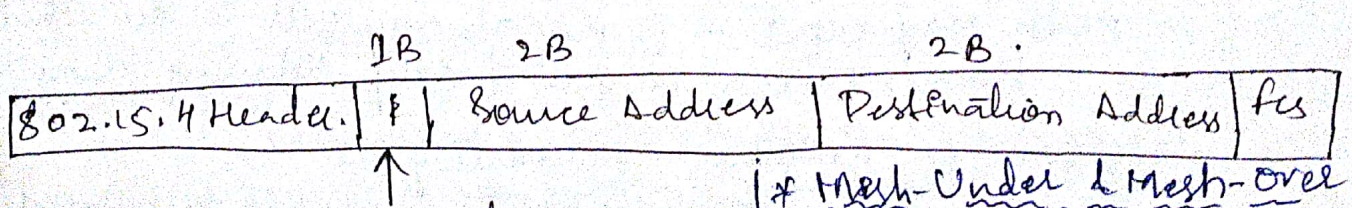
In the fragmentation header for an IPv6 payload being only 4 bytes long. The remainder of the fragments have a 5-byte header field so that the appropriate offset can be specified.

→ Mesh Addressing :-

The purpose of the 6LOWPAN mesh addressing function is to forward packets over multiple hops. Three fields are defined for this header; Hop Limit, Source Address, and Destination Address.

- The hop limit for mesh addressing also provides an upper limit on how many times the frame can be forwarded. Each hop decrements this value by 1 as it is forwarded. Once the value hits 0, it is dropped and no longer forwarded.
- The Source Address and Destination Address fields for mesh addressing are IEEE 802.15.4 addresses indicating the endpoints of the IP hop.

→ Figure(e) :- 6LOWPAN Mesh Addressing Header.



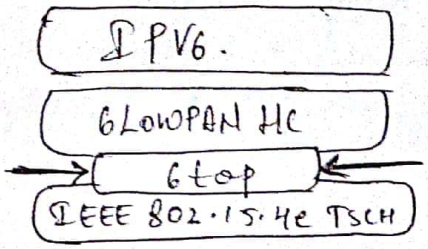
6LOWPAN Mesh Addressing Header including Hop count.

- Hop limit – how many times the frame can be forwarded.
- Each hop decrements this value by 1 as it is forwarded.
- Once the value hits 0, it is dropped & no longer forwarded

* Mesh-Under & Mesh-Over Routing :- IEEE 802.15.4, IEEE 15.4g, & IEEE 1901.2a supports the MAC & operate at the Phy and data link layers.
- mesh-under, the routing of packets is handled at the 6LOWPAN adaption layer
- "mesh-over" or "route-over" utilizes IP routing to getting packets to their destination

⇒ **6TiSCH** :- [Time-Slotted channel Hopping], is an add-on to the Media Access Control (MAC) portion of the IEEE 802.15.4 Standard, with direct inheritance from other Standard. Devices implementing IEEE 802.15.4e TSCH, communicate by the Time Division Multiple Access (TDMA) Schedule

Fig (4) :- Location of 6TiSCH's 6top Sublayer.



- Schedules in 6TiSCH are broken down into cells. A cell is simply a single element in the TSCH schedule that can be allocated for unidirectional or bidirectional communications is open for communication between the specific nodes.
- The 6TiSCH architecture defines four schedule management mechanisms;

* **Static Scheduling** :- All nodes in the constrained network share a fixed schedule. Cells are shared, and nodes contend for slot access in a slotted aloha manner.

  Slotted aloha over a shared is a basic protocol for sending data using time slot boundaries when communicating over a shared medium.

  The drawback with static scheduling is that nodes may expect a packet at any cell in the schedule. Hence, Energy is wasted idly listening across the cell.

* **Neighbor-to-neighbor scheduling** :-

  A schedule is established that correlates with the observed number of transmissions between nodes. Cells in this schedule can be added or deleted as traffic requirements and bandwidth needs change.

* Remote monitoring and scheduling management :—
Time slots and other resource allocations are handled by a management entity that can be multiple hops away. This scheduling mechanism provides quite a bit of flexibility and control in allocating cells for communication between nodes.

* Hop-by-Hop scheduling :—
A node reserves a path to a destination node multiple hops away by requesting the allocation of cells in a schedule at each intermediate node in the path.

⇒ There are three 6TiSCH forwarding models :—

* Track forwarding (TF) :—
This is the simplest and fastest forwarding model. A "track" is this model is a unidirectional path between a source and a destination.
This track is constructed by pairing bundles of receive cells in a schedule with a bundle of receive cells set to transmit.

* Fragment forwarding (FF) :—
This model takes advantage of 6LoWPAN fragmentation to build a Layer 2 forwarding table.
The 6LoWPAN sublayer learns the next-hop selection of this first fragment, which is then applied to all subsequent fragments of that packet.
Otherwise, IPv6 packets undergo hop-by-hop reassembly.
This increases latency and can be power and CPU-intensive for a constrained node.

* IPv6 forwarding (6F) :— This model forwards traffic based on its IPv6 routing table. Flows of packets should be prioritized by traditional QoS (Quality of Service) and RED (Random Early detection) operations.

→ RPL :-

The new distance vector routing protocol was named the IPv6 Routing Protocol for Low power and Lossy networks (RPL).

The RPL specification was published as RFC 6550 by the ROLL (Routing over Low-power and Lossy networks) working group.

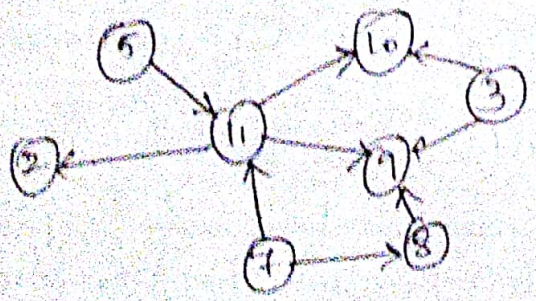
- In an RPL network, each node acts as a router and becomes part of a mesh network. Routing is performed at the IP layer.
- To cope with the constraints of computing and memory that are common characteristics of constrained nodes, the protocol defines two modes.

* Storing mode :- All nodes contain the full routing table of the RPL domain. Every node knows how to directly reach every other node.

* Non-storing mode :- Only the border router(s) of the RPL domain contain(s) the full routing table.
All other nodes in the domain only maintain their list of parents and use this as a list of default routes toward the border router.

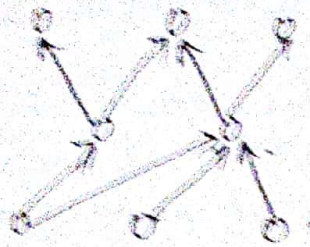
→ RPL is based on the concept of a directed acyclic graph (DAG). A DAG is a directed graph where no cycles exist.

→ Fig (g) - Example of a Directed Acyclic Graph (DAG).



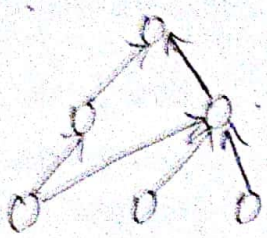
A basic RPL process involves building a destination oriented directed acyclic graph (DODAG). A DODAG is a DAG rooted to one destination.

Fig (4) - DAG and DODAG comparison.



DAG.

DAG Roots.



DODAG.

In the above fig, compares a DAG and a DODAG. We can see that a DAG has multiple roots, whereas the DODAG has just one.

In a DODAG, each node maintains up to three parents that provide a path to the root, Typically, one of these parents is the preferred parent, which means it is the preferred next hop for upward routes toward the root.

⇒ Authentication and Encryption on constrained Nodes:-

Here we have two security: ACE and DICE

→ ACE:- Like the ROLL working group, the Authentication and Authorization for constrained Environments (ACE) working group is tasked with evaluating the applicability of existing authentication and authorization protocols and documenting their suitability for certain constrained-environment use cases.

→ DICE :- Datagram Transport Layer Security

The DTLS in constrained Environments (DICE) working group focuses on implementing the DTLS transport layer security protocol in these environments.

The first task of the DICE working group is to define an optimized DTLS profile for constrained nodes.

Profiles and compliances :—

* **Internet Protocol for Smart Objects (IPSO) Alliance**

In 2008, the IPSO alliance has had its objective evolve and established. The alliance initially focused on promoting IP as the premier solution for smart objects communications.

The IPSO Alliance does not define technologies, as that is the role of the IETF and other standard organizations, but it documents the use of IP-based technologies for various IOT use cases and participates in educating the industry.

* **Wi-SUN Alliance :—**

The Wi-SUN Alliance is an example of efforts from the industry to define a communication profile that applies to specific physical and data link layer protocols.

* **Thread :-**

A group of companies involved with smart object solutions for consumers created the Thread Group.

This group has defined an IPV6-based wireless profile that provides the best way to connect more than 250 devices into a low-power, wireless mesh networks.

* **IPV6 Ready Logo :-**

The IPV6 Ready logo program has established conformance and interoperability testing programs with the intent of increasing user confidence when implementing IPV6.

The IPV6 core and specific IPV6 components, such as DHCP, IPsec, and customer edge router certifications, are in place.

Chapter-2 :-
Application Protocols for IOT.

The Transport Layer

— With the TCP/IP protocols, two main protocols are specified for the transport layer :-

* **Transmission Control Protocol (TCP) :-**
This connection oriented protocol requires a session to get established between the source and destination before exchanging data.

* **User Datagram Protocol (UDP) :-** With this connectionless protocol, data can be quickly sent between source & destination — but with no guarantee of delivery.
This is analogous to the traditional mail system, in which a letter is mailed to a destination.

— TCP is the main protocol used at the transport layer. This is largely due to its inherent characteristics, such as its ability to transport large volumes of data into smaller sets of packets.

— In contrast, UDP is most often used in the context of network services, such as its ability Domain Name System (DNS), Network Time Protocol (NTP), Simple Network Management Protocol (SNMP), and Dynamic Host Control Protocol (DHCP), or for real-time data traffic, including voice and video over IP.

— While the use of TCP may not strain generic computer platforms and high-data-rate networks, it can be challenging and is often overkill on constrained IOT devices and networks.

IoT Application Layer Protocols

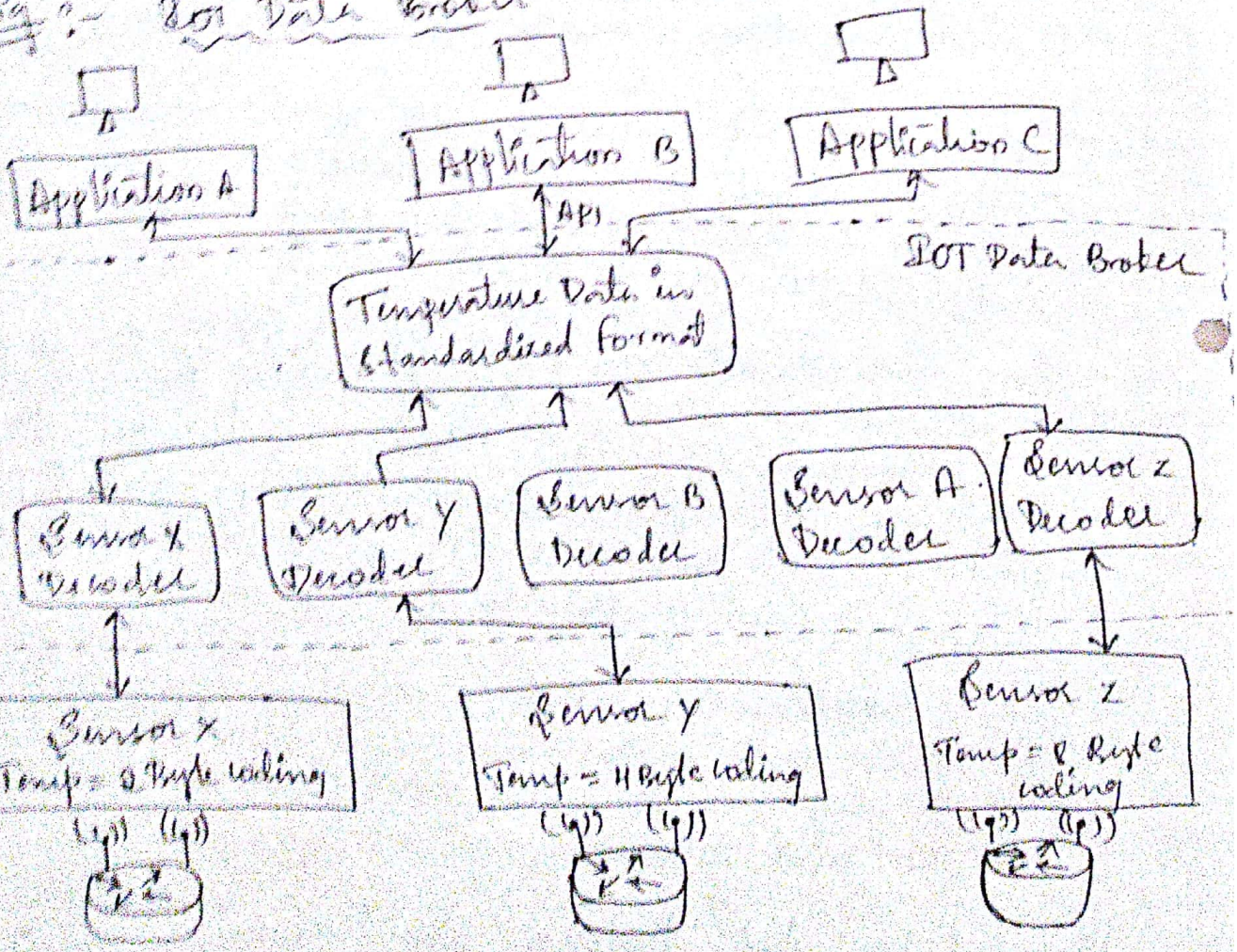
There are various means for transporting data protocols across a network. The following are the categories of IoT application protocols and their transport methods are covered in the following section.

- Application layer protocol not present
- Supervisory control and data acquisition
- Generic web-based protocols
- IoT application layer protocols.

Application Layer Protocol not Present

IETF RFC 7228 device defined as zero send or receive only a few bytes of data. Zero devices are usually simple in many aspects that are severely constrained.

Eg:- IoT Data Broker



- In the fig as shown, the different kinds of temperature sensors from different manufacturer. These sensors will report temperature data in varying formats.

If we scale the scenario and across hundreds or thousands of sensors, the problem of allowing various application to receive and interpret temperature values delivered in different formats becomes increasingly complex. The solution to this problem is to use an IOT data broker.

An IOT data broker is a piece of middleware that standardizes sensor output into a common format that can then be retrieved by authorized applications.

- In the fig, Sensors X, Y and Z are all temperature sensors, but their output is encoded differently.

Application A, B and C can access this temperature data without having to deal with decoding multiple temperature data formats.

⇒ Supervisory control and data acquisition (SCADA) :—

It is combined with the fact that IP is the de-facto standard for standard for computer networking in general, older protocols that connected sensors and actuators have evolved and adapted themselves to utilize IP.

At a high level, SCADA systems collect sensor data and telemetry from remote devices, while also providing the ability to control them. In today's networks SCADA systems allow global, real-time, data-drive decisions to be made about how to improve business processes.

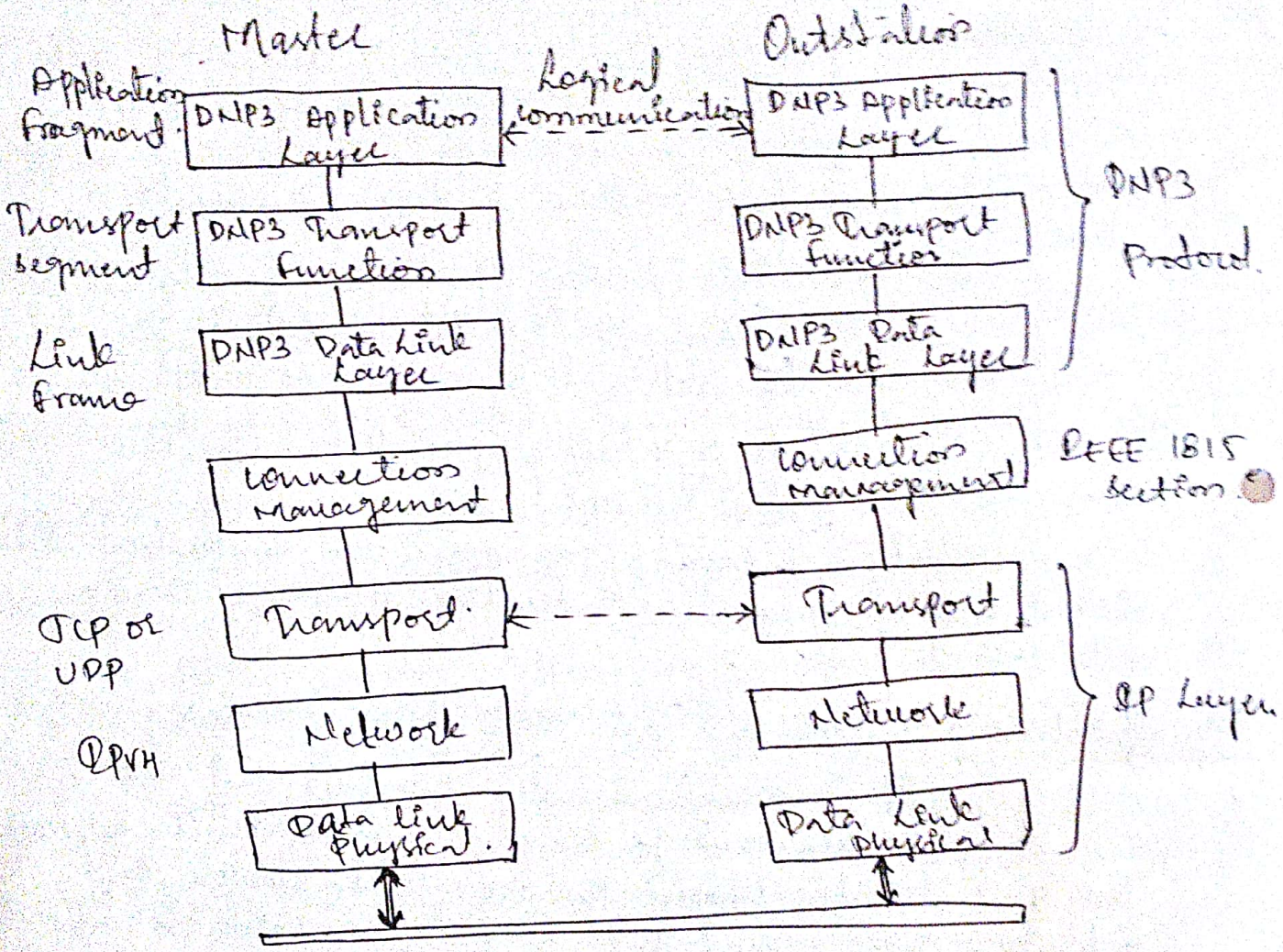
* Adapting SCADA for IP :—

In the 1990s, the rapid adoption of Ethernet networks in the industrial world drove the evolution of SCADA application layer protocol.

To further facilitate the support of legacy industrial protocols over IP networks, protocol specifications were updated and published, documenting the use of IP for each protocol.

Eg :- Protocol Stack for Transporting Serial DNP3 SCADA over IP.

DNP3 → Distributed Network Protocol



- Like many of the other SCADA protocols, DNP3 is based on a master/slave relationship.

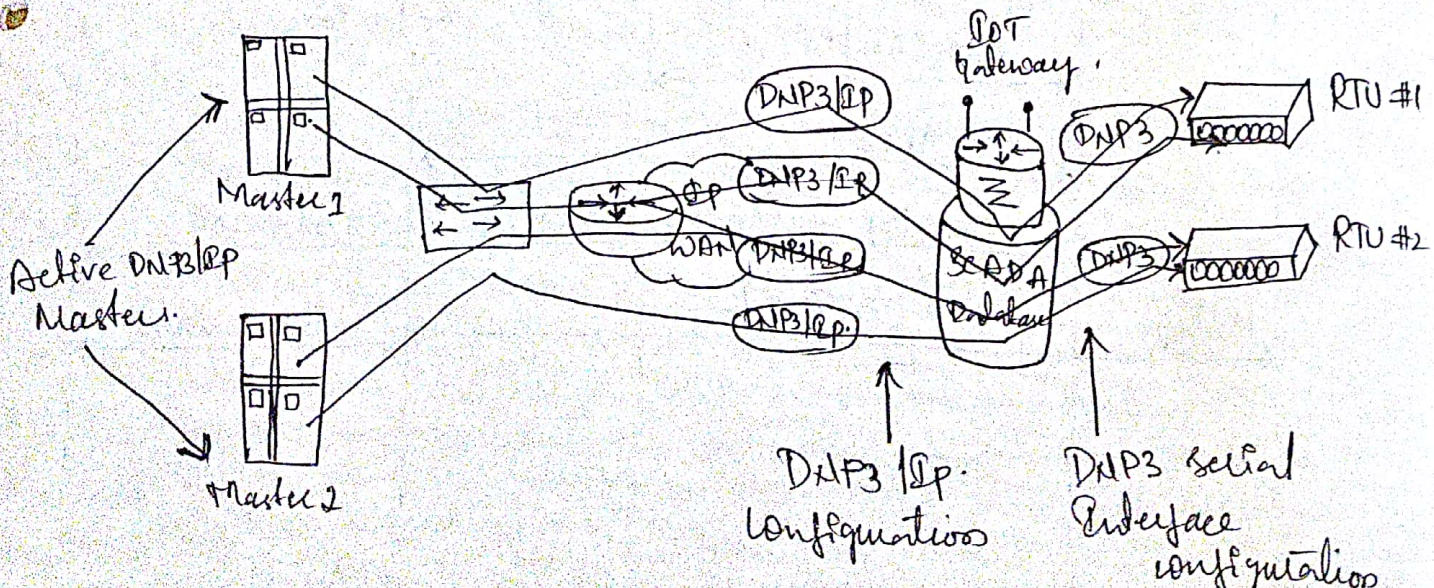
The term master in this case refers to what is typically a powerful computer located in the control center of a utility, and a slave is a remote device with computing resources found in a location such as a plant or motor or less a substation. DNP3 refers to slaves specifically as outstations.

- Outstations monitor and collect data from devices that indicate their state, such as whether a circuit breaker is on or off, and take measurements, including voltage, current, temperature and so on.

- Connection management links the DNP3 layer with the IP layer. In addition to the configuration parameters and methods necessary for implementing the network connection.
- The DNP3 endpoints or devices are not aware of the underlying IP transport that is occurring.
- The master side initiates connections by performing a TCP active open. The outstation listens for a connection request by performing a TCP passive open.
- Dual end-point is defined as a process that can both listen for connection requests and perform an active open on the channel if required.
- Keepalive messages are implemented as DNP3 data link layer status requests. If a response is not received to a keepalive message, the connection is deemed broken, and the appropriate action is taken.

* SCADA Protocol Translation :- (RTU - Remote Terminal Units)

Fig :- DNP3 Protocol Translation

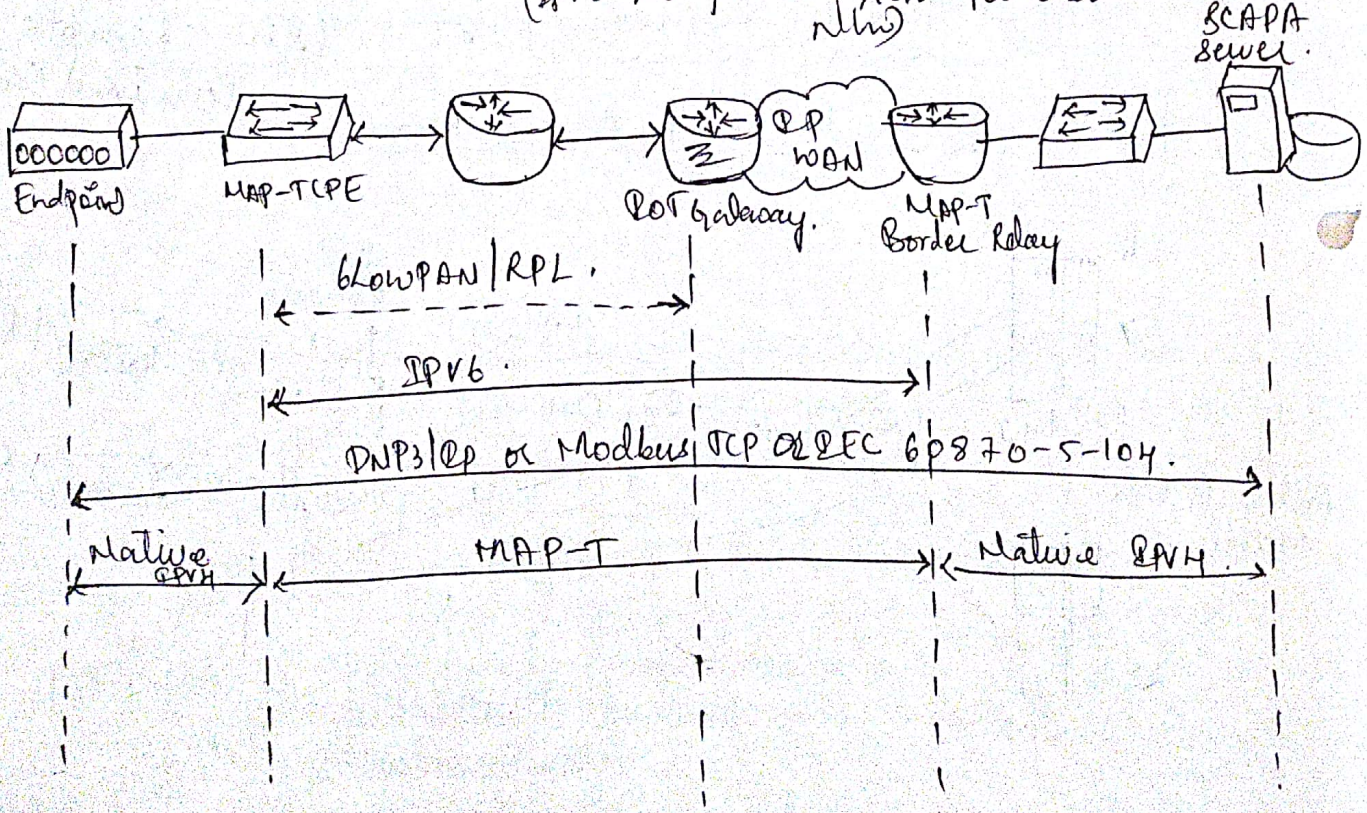


- The above figure shows two serially connected DNP3 RTUs and two master applications supporting DNP3 over IP that control and pull data from the RTUs.
- The IOT gateway in the figure performs a protocol translation function that enables communication between the RTUs and servers, despite the fact that a serial connections is present on one side and an IP connection is used on the other.
- By running protocol translation, the IOT gateway connected to the RTUs in fig - is implementing a computing function close to the edge of the network. Adding computing functions close to the edge helps scale distributed intelligence in IOT networks.

SCADA Transport over LLNs with MAP-T :-

Figure : DNP3 Protocol over 6LOWPAN Networks with MAP-T.

(IPv6 Lowpower wireless personal area Nlw)



- The above figure depicts a scenario in which a legacy endpoint is connected across an LLN running 6LOWPAN to an IP-capable SCADA server.

- The legacy endpoint could be running various industrial and SCADA protocols, including DNP3/IP, Modbus/TCP, or IEC 60870-5-104.

In this scenario, the legacy devices and the SCADA server support only IPV4, and IPV6 is being used for connectivity to the endpoint.

- In this situation, the end points/devices, the SCADA server support only IPV4, but the network in the middle supports only IPV6.
- The IPV4 endpoints on the left side is connected to a Customer Premise Equipment (CPE) device. The MAP-T CPE has an IPV6 connection to the RPL mesh.

On the right side, a SCADA server with native IPV4 support connects to a MAP-T border gateway.

⇒ Generic Web-Based Protocols :-

Web-based protocols have become common in consumer and enterprise applications and services. Hence it makes sense to try to leverage these protocols when developing IOT applications, services and devices in order to ease the integration of data and devices from Prototyping to production.

The HTTP/HTTPS client/server model serves as the foundation for the world wide web. Recent evolutions of embedded web server software with advanced features are now implemented with very little memory.

- Interactions between real-time communication tools powering collaborative applications, such as voice and video, instant messaging, chat rooms and IOT devices are also merging.

This is driving the need for simpler communication systems between people and IOT devices. One protocol that addresses this need is Extensible Messaging and Presence Protocol (XMPP).

→ IoT Application Layer Protocols :-

When considering constrained networks and/or a large-scale deployment of constrained nodes, verbose web-based and data model protocols may be too heavy for IoT applications.

The two most popular protocols that suited for constrained network and nodes are CoAP and MQTT.

fig:- Example of a High-Level IoT protocol stack for CoAP and MQTT.

CoAP.	MQTT.
UDP	TCP.
IPV6.	
6LOWPAN	
802.15.4 MAC.	
802.15.4 PHY	

Here CoAP messaging model is primarily designed to facilitate the exchange of messages over UDP between endpoints, including the secure transport protocol Datagram Transport Layer Security (DTLS)

- In the above figure - CoAP and MQTT are naturally at the top. This sample IoT stack, based on an IEEE 802.15.4 mesh network.

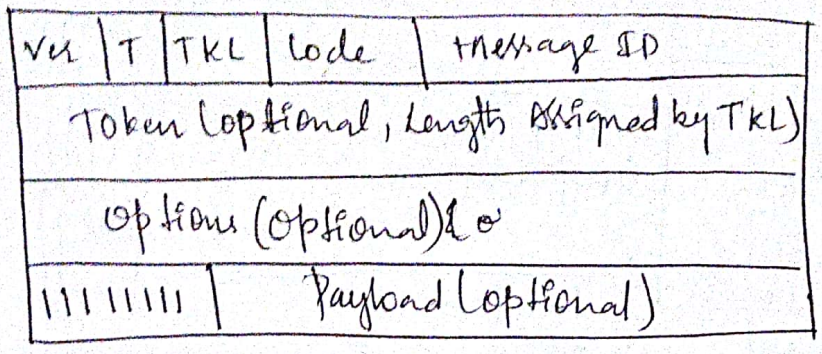
CoAP :-

Constrained Application Protocol (CoAP) resulted from the IETF constrained RESTful Environments (CoRE) working group's effort to develop a generic framework for resource oriented applications targeting constrained nodes and networks.

- The CoAP framework defines simple and flexible ways to manipulate sensors and actuators for data or device management.

→ RFC 6690: Constrained RESTful Environment (CoRE) Link format.
→ RFC 7252: The constrained Application Protocol (CoAP)
→ RFC 7641: Observing Resources in the constrained Application Protocol (CoAP)
→ RFC 7959: Block-wise Transfer in the CoAP
→ RFC 8075: Guidelines for Mapping Implementation HTTP to CoAP

Fig :- CoAP Message Format

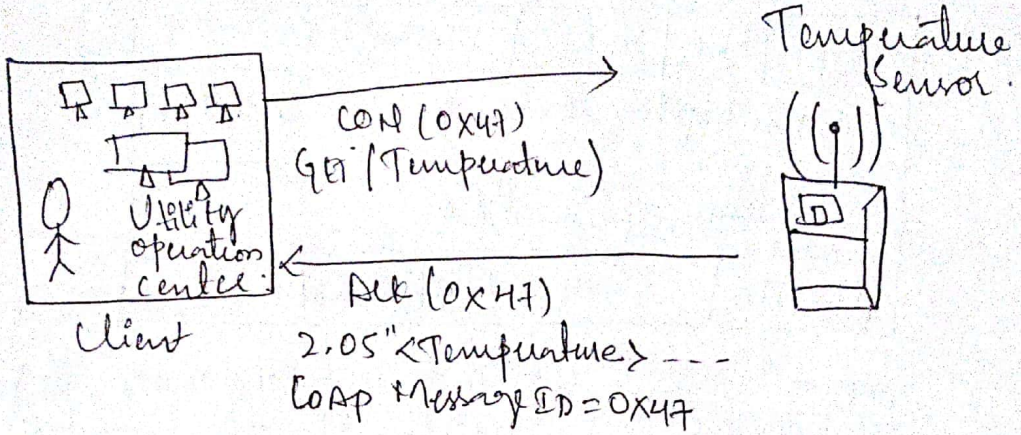


- The above figure represents the CoAP message format is relatively simple and flexible. It allows CoAP to deliver low overhead, which is critical for constrained networks, while also being easy to parse and process for constrained devices.

* Ver (Version) :- Identifies the CoAP version
* T (Type) :- Defines one of the following four message types : Confirmable (CON), Non-Confirmable (NON), Acknowledgement (ACK), or Reset (RST).
* TKL (Token Length) :- Specifies the size (0-8 Bytes) of the Token field.
* Code :- Indicates the request method for a request message and a response code for a response message.
* Message ID :- Detects message duplication and used to match ACK and RST message types to CON and NON message types.
* Token :- With a length specified by TKL, correlates requests and responses.
* Options :- Specifies option number, length and option value.
* Payload :- Carries the CoAP application data. This field is optional, but when it is present, a single byte of all 1s (0xFF) precedes the payload.

→ CoAP defines four types of messages, CON, NON, ACK and RST. Method codes and response codes included in some of these messages make them carry requests or responses.

Fig :- CoAP Reliable Transmission Example.



- The above figure shows a utility operations center on the left, acting as the CoAP client, with CoAP server being a temperature sensor on the right of the figure. The communication between the client and server uses a CoAP message ID of 0X47. The CoAP message ID ensures reliability and is used to detect duplicate message.

Message Queuing Telemetry Transport (MQTT).

At the end of the 1990's, engineers were looking for a reliable, lightweight, and cost-effective protocol to monitor and control a large number of sensors and their data from a central server location.

They have introduced a client/server and publish/subscribe framework based on the TCP/IP architecture.

An MQTT client can act as a publisher to send data to an MQTT server acting as an MQTT message broker.

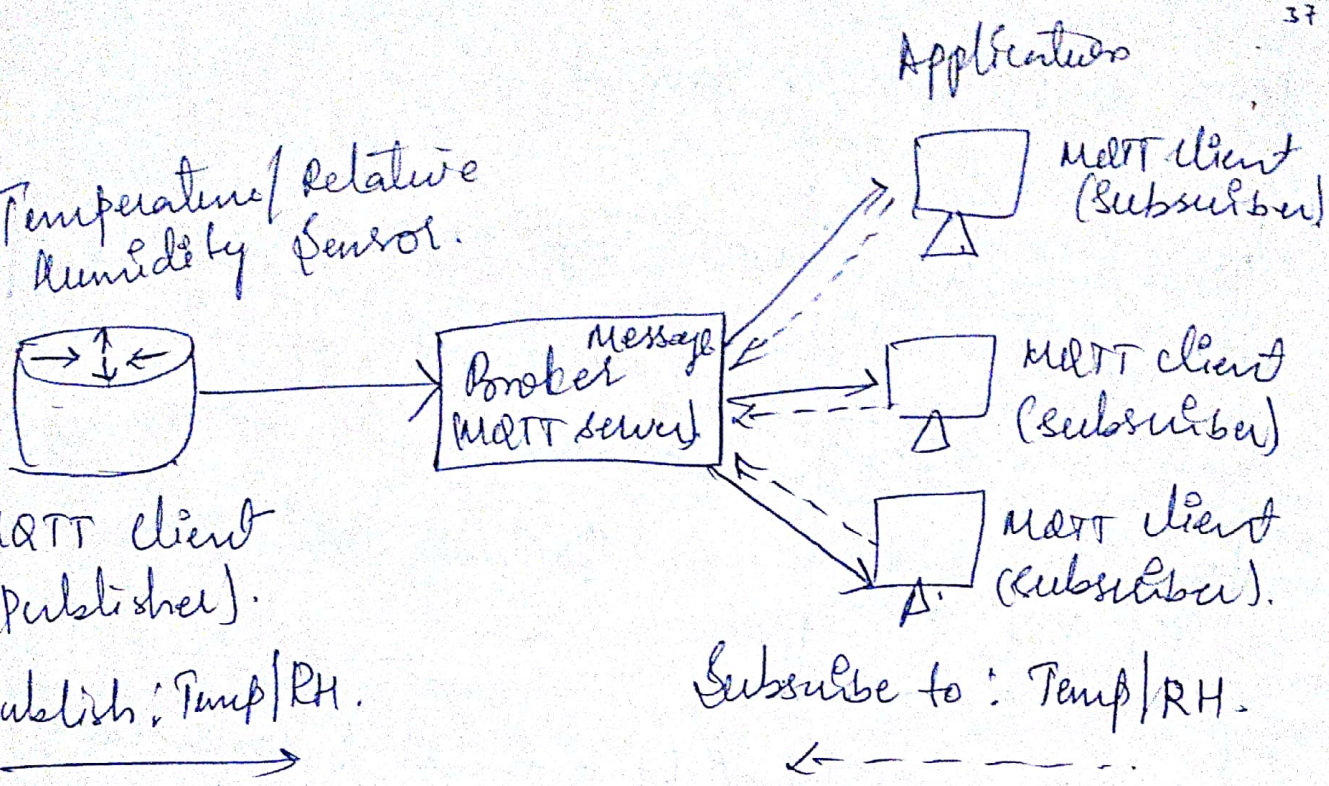


Fig: MQTT publish/subscribe framework.

In the figure, the MQTT client (publisher) on the left side is a temperature (Temp) and relative humidity (RH) sensor that publishes its Temp/RH data.

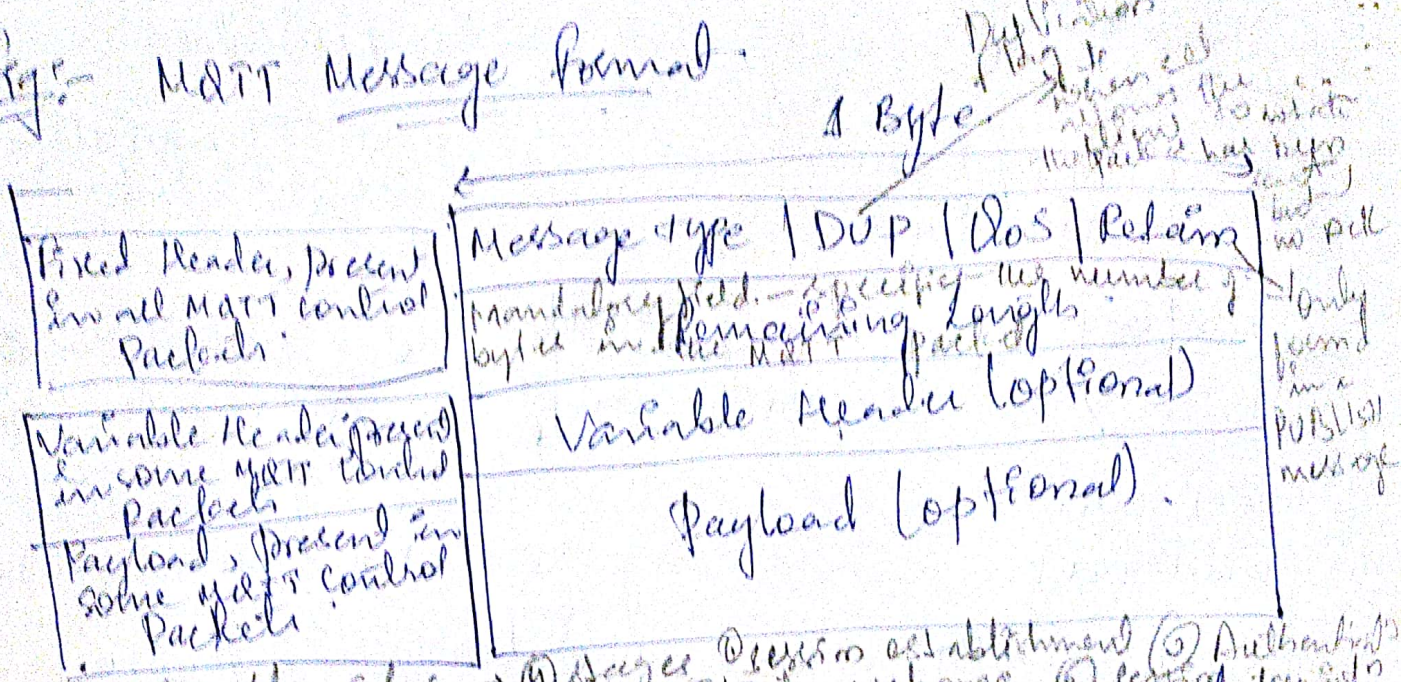
← The MQTT server (or message broker) accepts the msg who connect'n along with application messages, such as Temp/RH data, from the publisher.

It also handles the subscription & Unsubscription process and pushes the application data to MQTT clients acting as subscriber.

← The application on the right side – is an MQTT client subscriber to the Temp/RH data being generated by the publisher or sensor on the left

– MQTT conducted packets runs over a TCP Transport using Port 1883. TCP ensures an ordered, lossless stream of bytes btw the MQTT client and the MQTT server.

Fig:- MQTT Message format.



MQTT session has 4 stages ① Session establishment ② Authentication ③ Data exchange ④ Session termination.

→ MQTT is a light-weight protocol because each control packet consists of a 2-byte fixed header with optional variable header fields and optional payload.

→ Compared to the CoAP message format, the MQTT contains a smaller header of 2 bytes compared to 4 bytes for CoAP.

① Message Type - MQTT packet which identifies the kind of message. There are 14 different types of control packets are specified in MQTT

Message type	Value	Flow	Description
CONNECT	1	client to server	Request to connect
CONNACK	2	server to client	connect acknowledgment
PUBLISH	3	C→S, S→C	Publish msg
PUBACK	4	C→S, S→C	Publish acknowledgement
PUBREC	5	C→S, S→C	Publish received
PUBREL	6	C→S, S→C	Publish release
PUBCOMP	7	C→S, S→C	Publish complete.