Module 3

IoT Network Layer

Badharudheen P
Assistant Professor,
Dept. of CSE, MESCE, Kuttippuram

The Business Case for IP

- Data flowing from or to “things” is consumed, controlled, or monitored by data servers either in the cloud or in locations that may be distributed or centralized.
- Dedicated applications are then run over operating systems or on network edge platforms (for example, fog computing).

- These lightweight applications communicate with the data center servers.
- IP was not only preferred in the IT markets but also for the OT environment.

Advantages of IP for IoT

- Open and standards-based:
  - Devices, applications, and users can leverage a large set of devices and functionalities while guaranteeing interchangeability and interoperability, security, and management.
  - This calls for implementation, validation, and deployment of open, standards-based solutions.
  - The IETF is an open standards body that focuses on the development of the Internet Protocol suite and related Internet technologies and protocols.

Advantages of IP for IoT

- Versatile:
  - A large spectrum of access technologies are available to offer connectivity of “things”.
  - Additional protocols and technologies are also used to transport IoT data through backhaul links and in the data center.
  - Communication technologies evolve at a pace faster than the expected.
  - The layered IP architecture is well equipped to cope with any type of physical and data link layers (Ethernet, Wi-Fi, and cellular).

Advantages of IP for IoT

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

- Scalable:
  - IP has been massively deployed and tested for robust scalability. Millions of private and public IP infrastructure nodes have been operational for years.

Advantages of IP for IoT

- Manageable and Highly Secure:
  - Well-known network and security management tools are easily leveraged with an IP network layer.
- Stable and Resilient:
  - IP has a large and well-established knowledge base and it has been used for years in infrastructures, such as financial and defense networks.
  - IP has been deployed for critical services, such as voice and video.

IP Adaptation Vs IP Adoption

- **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 operations.

The Need for Optimization

- IoT networks are largely built on the IP suite.
- Challenges still exist for IP in IoT solutions.
  - To integrate with non-IP devices.
  - Also, IP is transitioning from version 4 to version 6
- So optimizations are needed at various layers of IP stack.

- **Constrained Nodes**: Nodes with limited capabilities (power, memory, processing capacity, etc.)
- **Constrained Networks**: Low-power and Lossy Networks (LLNs) with limited bandwidth.

IP Versions

- For 20+ years, the IETF has been working on transitioning the Internet from IPV4to IPV6 because of the lack of address space in IPv4.

- Today, both versions of IP run over the Internet, but most traffic is still IPv4 based.

- Techniques such as tunneling and translation need to be employed in IoT solutions to ensure interoperability between IPv4 and IPv6.

IP Versions

- Factors which decides whether IPv4, IPv6, or both can be used in an IoT solution are:

1. Application Protocol:
   - SCADA protocols and Modbus TCP standards are specified only for IPv4.
   - In HTTP/HTTPS, CoAP, MQTT, and XMPP, both IP versions are supported.

IP Versions

2. Cellular Provider and Technology:
    - IoT devices with cellular modems are dependent on the data services offered by the provider.
    - For the first three generations of data services (GPRS, Edge, and 3G), IPv4 is the base protocol version. If IPv6 is used with these generations, it must be tunneled over IPv4.
    - On 4G/LTE networks, data services can use IPv4 or IPv6 as a base protocol, depending on the provider.

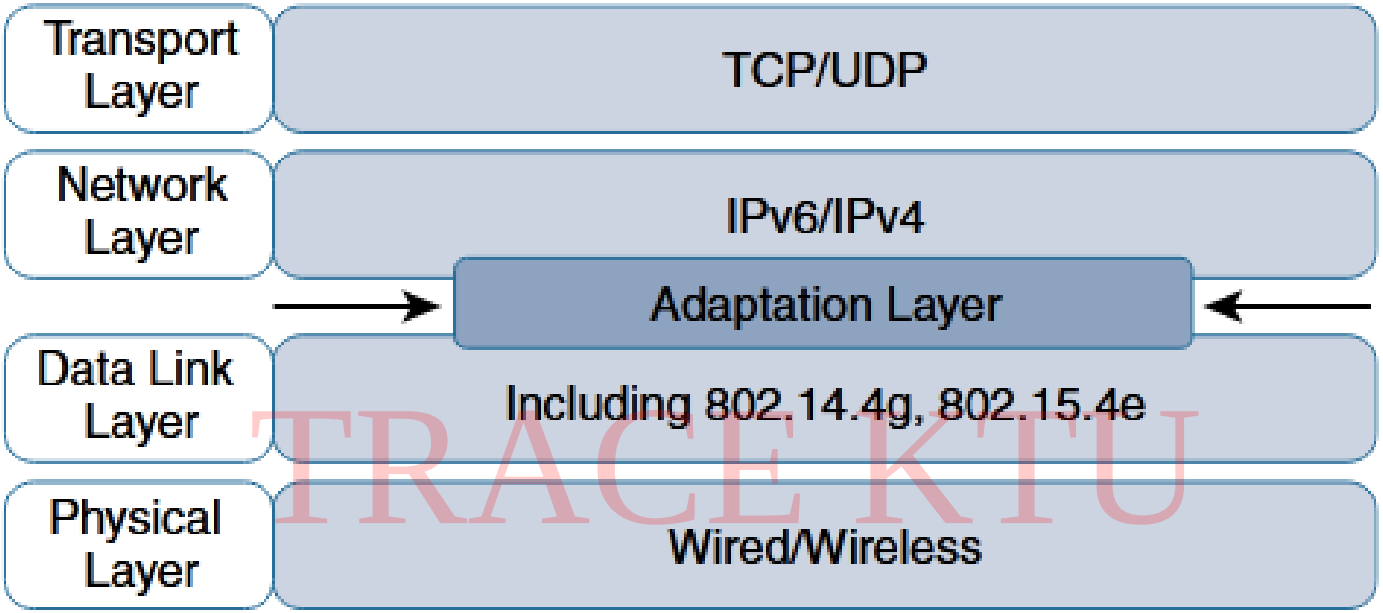
IP Versions

3. Serial Communications:
    - Many legacy devices in certain industries communicate through serial lines.
    - Here, the serial port of the legacy device will be connected to the serial port a router. This local router then forwards the serial traffic over IP to the central server for processing.
    - Encapsulation of serial protocols over IP leverages mechanisms such as raw socket TCP or UDP.
    - While raw socket sessions can run over both IPv4 and IPv6, current implementations are mostly available for IPv4 only.

IP Versions

4. IPV6 Adaptation Layer:
    - Some physical and data link layers for recently standardized IoT protocols support only IPv6.
    - While the most common physical and data link layers (Ethernet, Wi-Fi, and so on) stipulate adaptation layers for both versions.
    - Transition mechanisms such as Mapping of Address and Port using Translation (MAP-T) allow IPv4 traffic to be forwarded over an IPv6 network.
    - Such techniques enable older devices and applications to continue running IPv4 even though the network providing connectivity is IPv6.

Optimizing IP for IoT

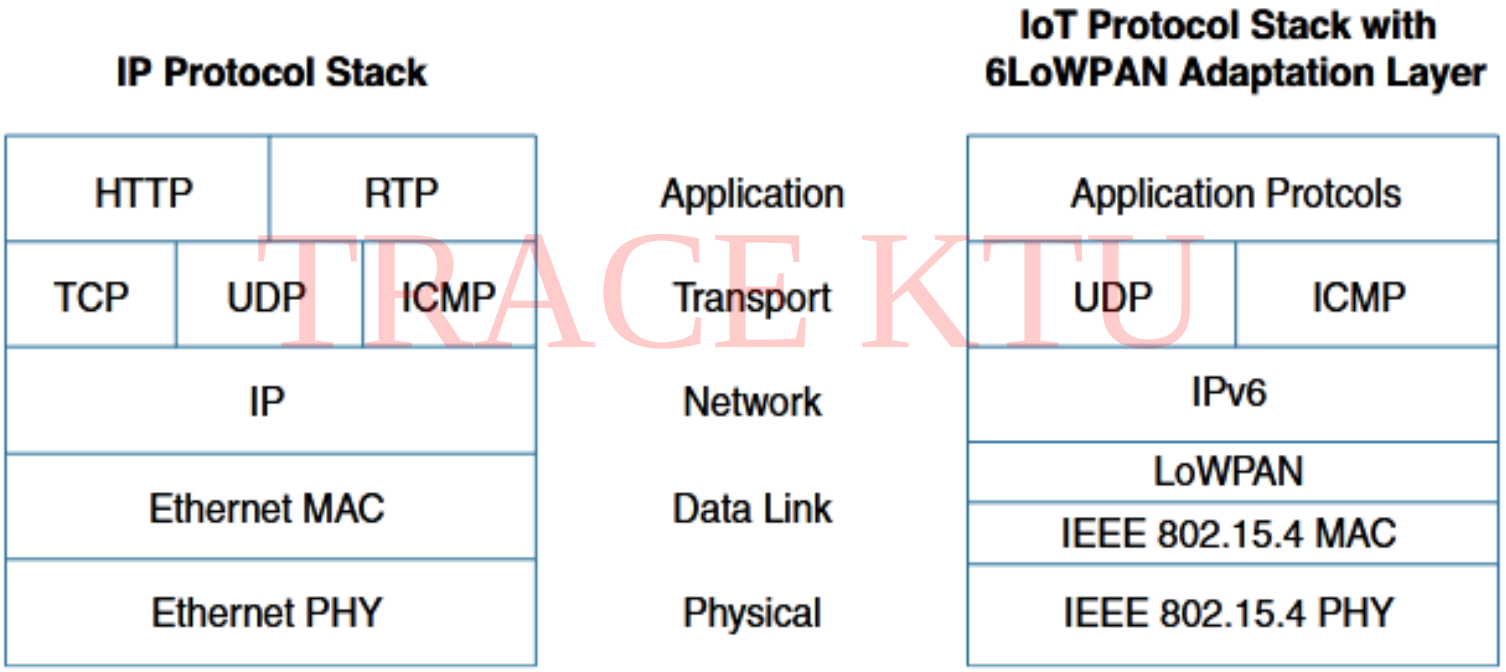


Optimizing IP for IoT Using an Adaptation Layer

The function of adaptation layer is to pack IP into lower layer protocols such as Layer 1 and 2 (ie., PHY, MAC).

From 6LoWPAN to 6Lo

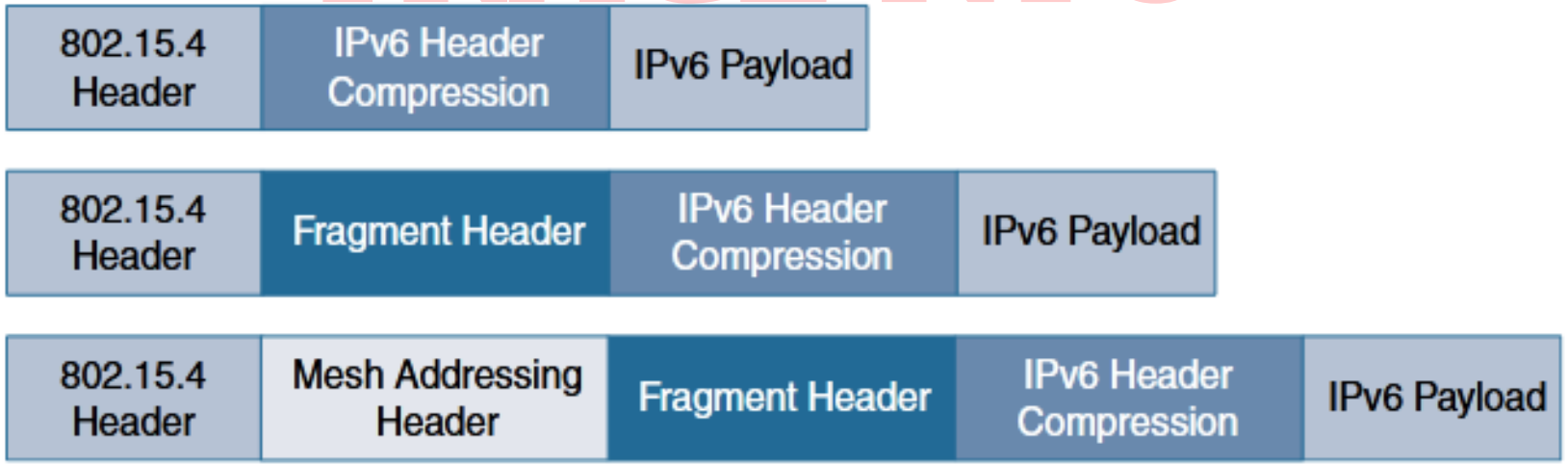
- The main examples of adaptation layers optimized for constrained nodes are 6LoWPAN and its successor 6Lo.



Comparison of an IoT Protocol Stack Utilizing 6LoWPAN and an IP Protocol Stack

From 6LoWPAN to 6Lo

- The 6LoWPAN defines frame headers for the capabilities of header compression, fragmentation, and mesh addressing.
- Depending on the implementation, all, none, or any combination of these capabilities and their corresponding headers can be enabled.

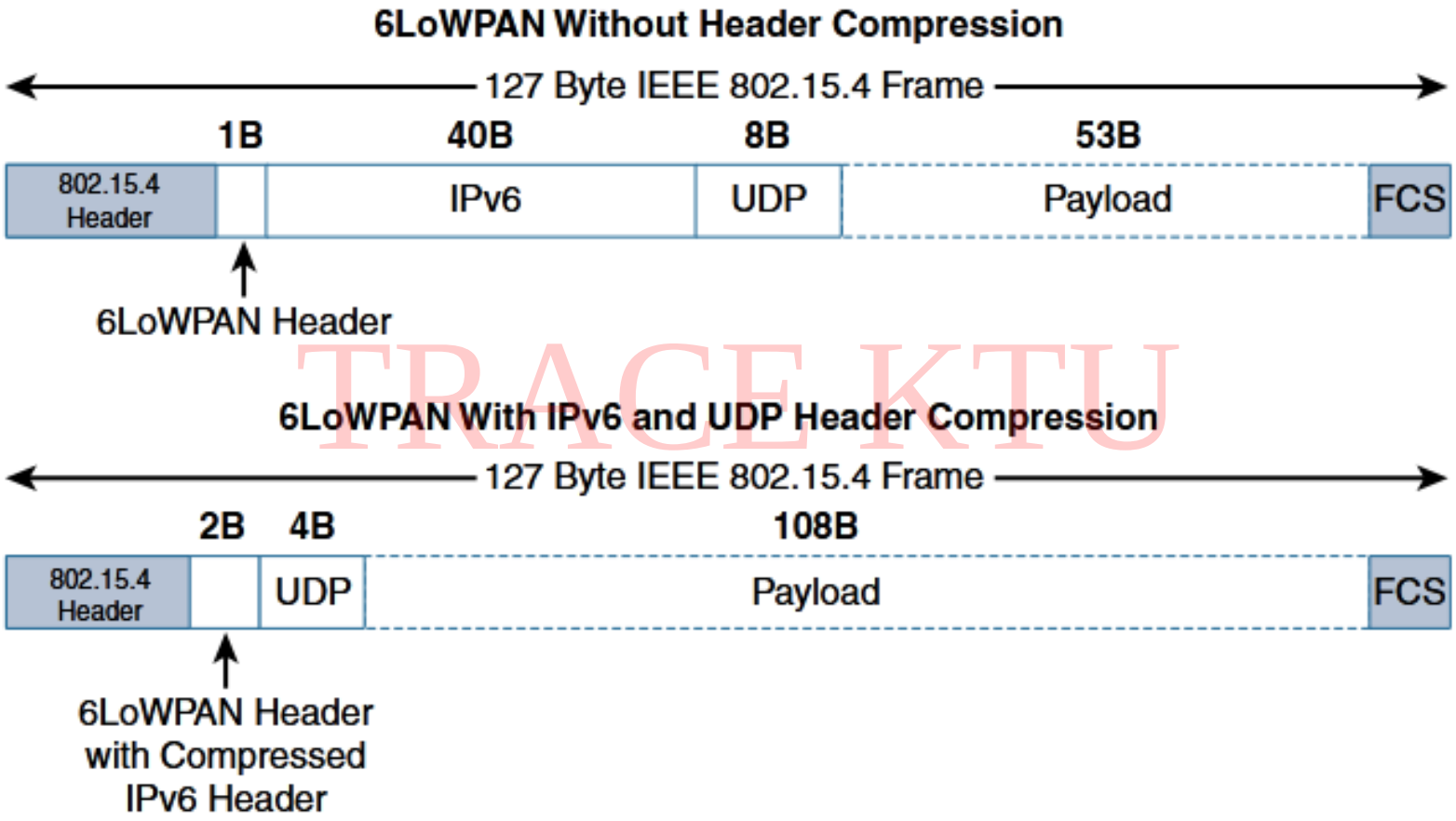


Header Compression

- It shrinks the size of IPv6's 40-byte headers and UDP's 8-byte headers down as low as 6 bytes.
  - Defined only for an IPv6 header and not IPv4.

- At a high level, 6LoWPAN works by taking advantage of shared information known by all nodes from their participation in the local network.
- In addition, it omits some standard header fields by assuming commonly used values.

Header Compression

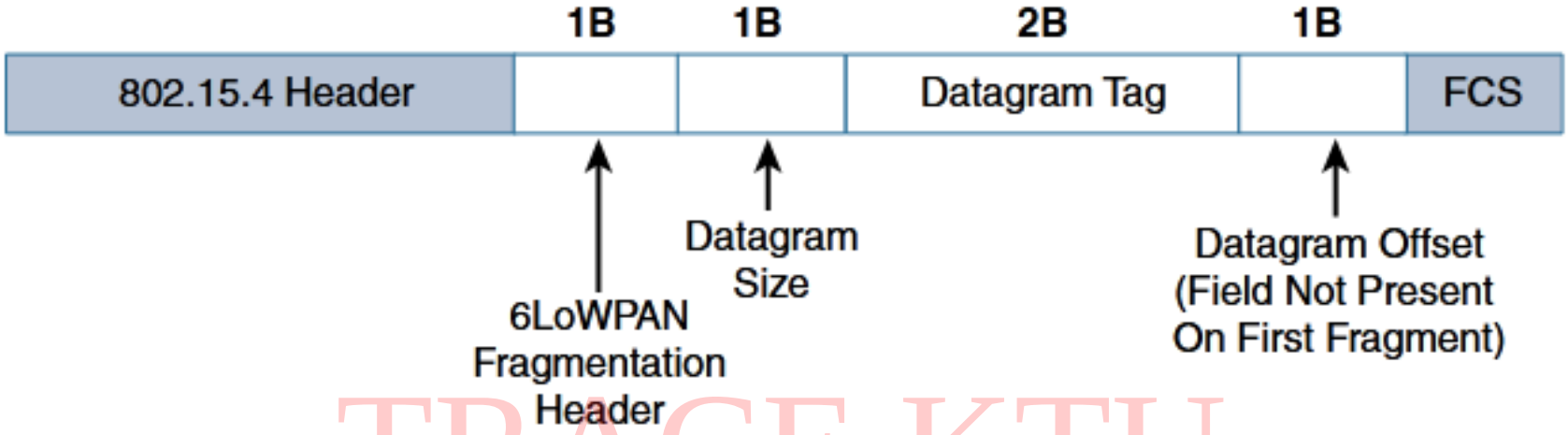


Fragmentation

- In general, the maximum transmission unit (MTU) for an IPv6 network must be at least 1280 bytes.
- For IEEE 802.15.4, 127 bytes is the MTU.

- So large IPv6 packets must be fragmented across multiple 802.15.4 frames at Layer 2.
- The fragment header is composed of three primary fields:
  - Datagram Size
  - Datagram Tag
  - Datagram Offset

Fragmentation

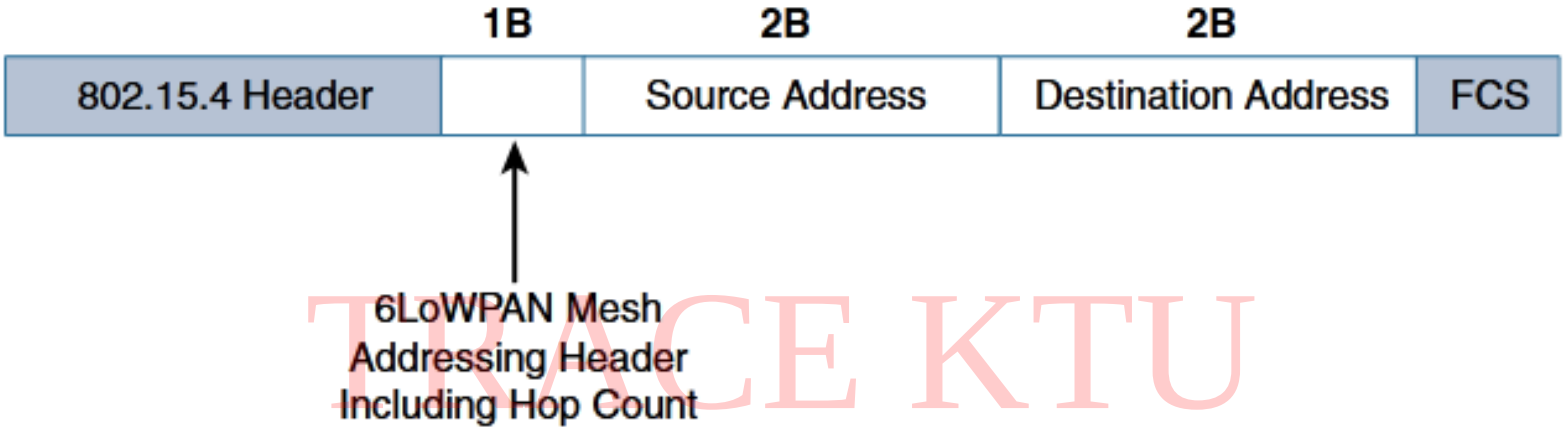


- The 1-byte Datagram Size field specifies the size of the payload.
- Datagram Tag identifies the set of fragments for a payload.
- Datagram Offset field specifies how far into a payload a particular fragment occurs.

Mesh Addressing

- The purpose is to forward packets over multiple hops.
- Three fields are defined for this header:
  - Hop Limit, Source Address, and Destination Address.
- The hop limit 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 indicates the endpoints of an IP hop.

Mesh Addressing



Note: The mesh addressing header is used in a single IP subnet and is a Layer 2 type of routing known as mesh-under.

Profiles and Compliances

- Some of the main industry organizations working on profile definitions for IoT constrained nodes and networks are:
  - Internet Protocol for Smart Objects (IPSO) Alliance
  - Wi-Sun Alliance
  - Thread
  - IPv6 Ready Logo.

Application Protocols for IoT

- IoT application protocols are dependent on the characteristics of the lower layers.
  - For eg:- application protocols used for traditional networks are not well suited for constrained nodes and networks.
- Besides HTTP, other protocols that are suitable for IoT are
  - Message Queue Telemetry Transport (MQTT),
  - Constrained Application Protocol (CoAP),
  - WebSocket,
  - Extensible Messaging and Presence Protocol (XMPP),
  - Advanced Message Queuing Protocol (AMQP).

The Transport Layer

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

- **User Datagram Protocol (UDP)**: With this connectionless protocol, data can be quickly sent between source and destination - but with no guarantee of delivery. This is analogous to the traditional mail delivery system, in which a letter is mailed to a destination. Confirmation of the reception of this letter does not happen until another letter is sent in response.

The Transport Layer

- TCP is the main protocol used at the transport layer of internet because of:
  - Reliability, flow control mechanism, retransmission of lost packets and reassembly of packets in order.
- In contrast, UDP is mostly used in the services such as 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.
- Note:- The selection of protocol depends on the performance and scalability of IoT constrained devices and networks.

IoT Application Transport Methods

- There are various methods for transporting IoT application protocols across a network.
    1. Application layer protocol not present
    2. Supervisory control and data acquisition (SCADA)
    3. Generic web-based protocols
    4. IoT application layer protocols

1. Application layer protocol not present

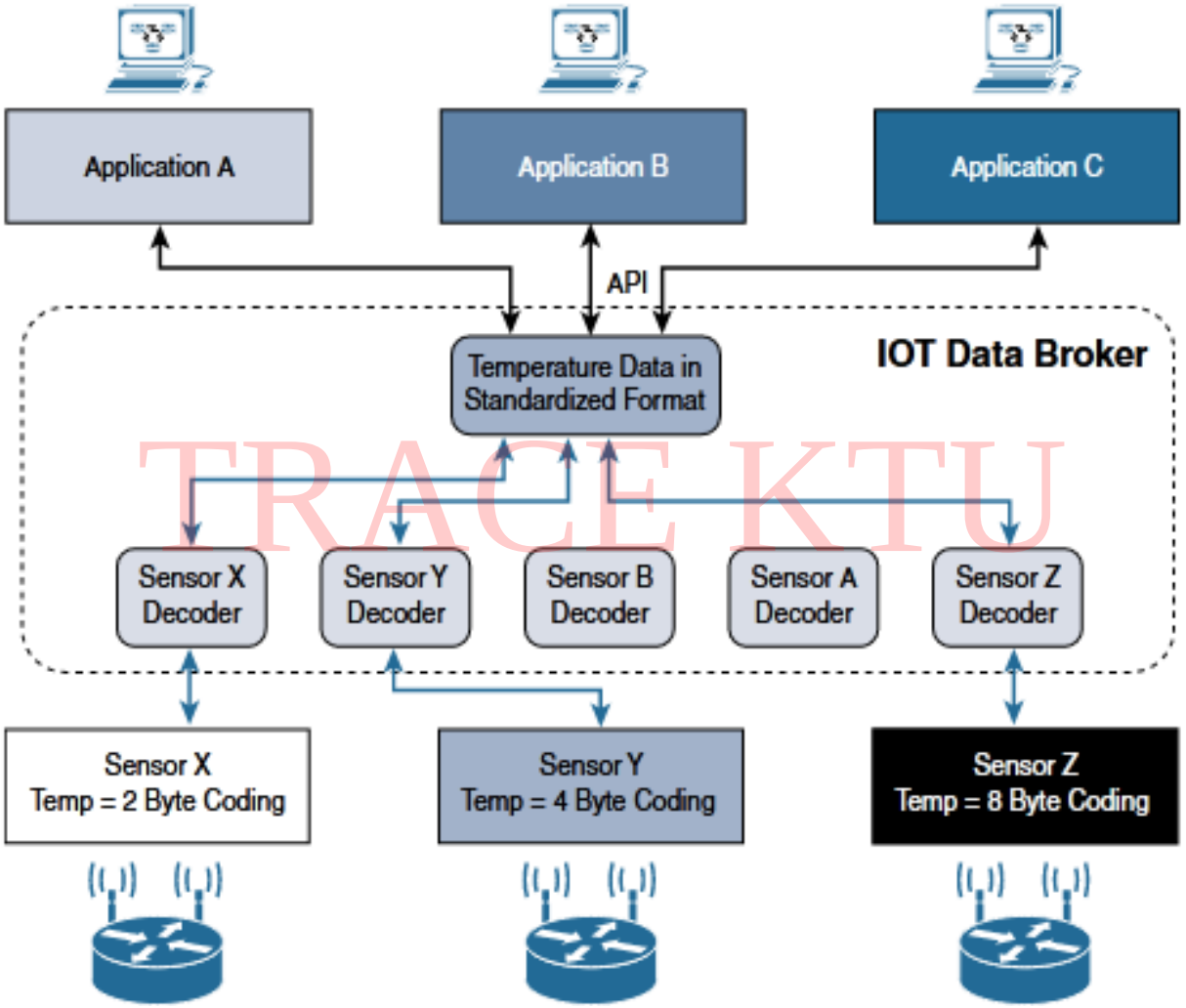
- Class 0 devices send or receive only a few bytes of data.
- For many reasons, such as processing capability, power constraints, and cost, these devices do not implement a fully structured network protocol stack or even an application layer protocol.
- Here the data payload is directly transported on top of the lower layers without the use of TCP/IP.
  - No application layer protocol is used.
  - So the sensors will report data in varying formats and decoding this data will be vendor specific.

Application layer protocol not present

- The decoding of these data will be difficult if the number of sensors increases.
  - Since the encoding scheme is different.

- 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.

Application layer protocol not present



2. Supervisory Control and Data Acquisition (SCADA)

- It is a control system architecture comprising computers, networked data communications and GUIs for high-level supervision of machines and processes.
- It is an automation control system that was initially implemented without IP over serial links, before being adapted to Ethernet and IPv4.
- SCADA systems collect sensor data and telemetry from remote devices, while also providing the ability to control them.
- In the initial stage of SCADA, the networking were serial link-based, such as RS-232 and RS-485.

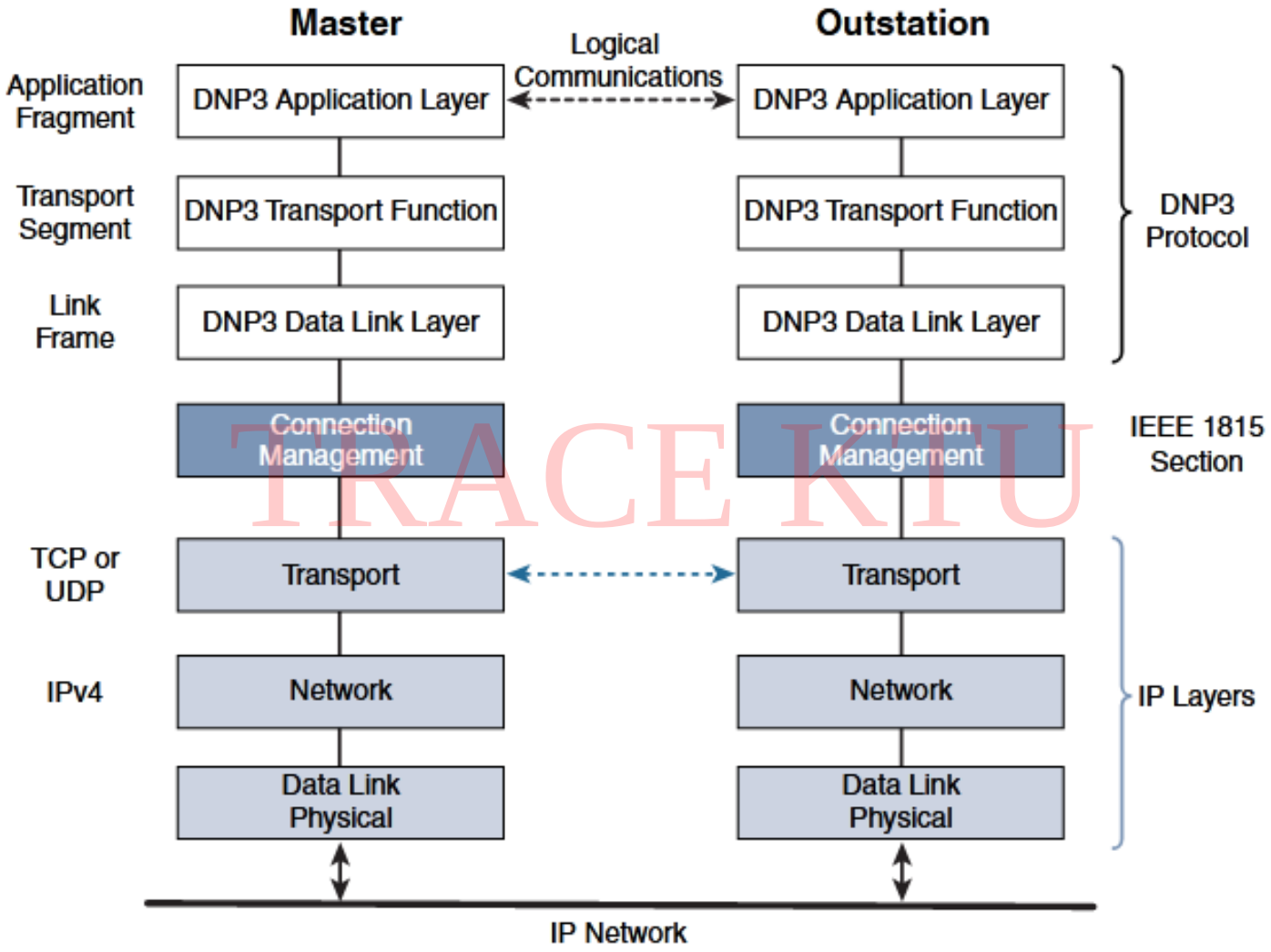
Adapting SCADA for IP

- Now days, the SCADA systems are connected with several open networks and allow the transmission of data over internet using TCP/IP or UDP.
- Examples of SCADA protocols and their port numbers:
  - DNP3 (Distributed Network Protocol 3) uses TCP or UDP on port 20000 for transporting messages over IP.
  - The Modbus messaging service uses TCP with port 502.
  - IEC (International Electrotechnical Commission) 60870-5-104 uses ethernet service using IPv4 with port 2404.
  - DLMS (Device Language Message Specification) runs over TCP/IP with port number 4059.

Adapting SCADA for IP

- DNP3 is based on a master/slave relationship.
- Master device refers to a powerful computer located in the control center.
- Slave is a remote device with computing resources found in a location such as a substation.
- DNP3 slaves are otherwise know as outstations.
- Outstations monitor and collect data from devices that indicate their state and take measurements.
  - This data is then transmitted to the master.
- The master also issues control commands, such as to start a motor or reset a circuit breaker, etc.

Adapting SCADA for IP



Protocol Stack for Transporting Serial DNP3 SCADA over IP

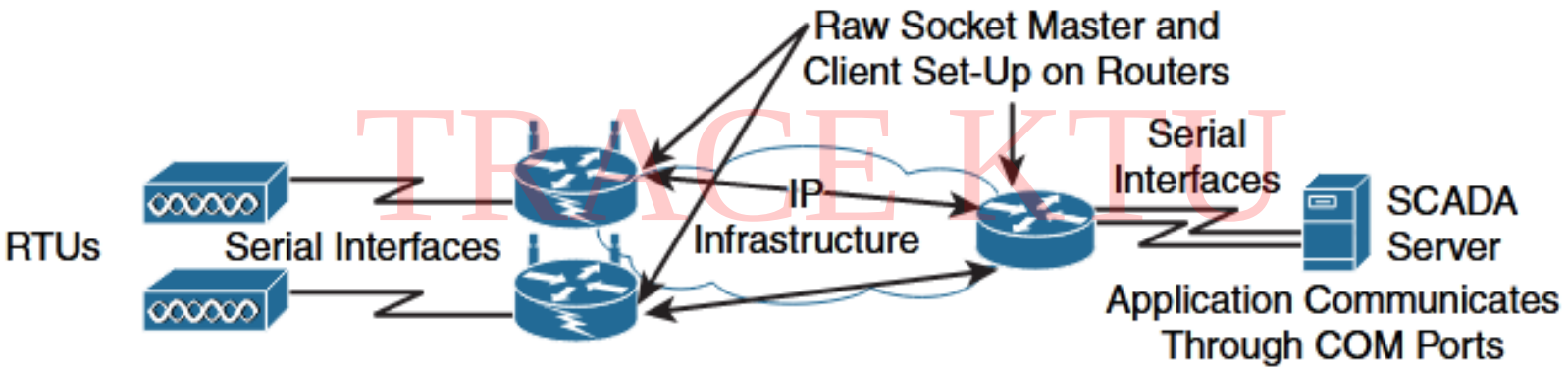
Adapting SCADA for IP

- Transport of the serial protocol over IP can be achieved either by:
  - Tunneling using raw sockets over TCP or UDP (In modern application servers)
  - Installing an intermediate device that performs protocol translation between the serial protocol and its IP implementation (In older servers).

- A raw socket connection denotes that the serial data is being packaged directly into a TCP or UDP transport.
  - A socket is a standard API composed of an IP address and a TCP or UDP port number.

Adapting SCADA for IP

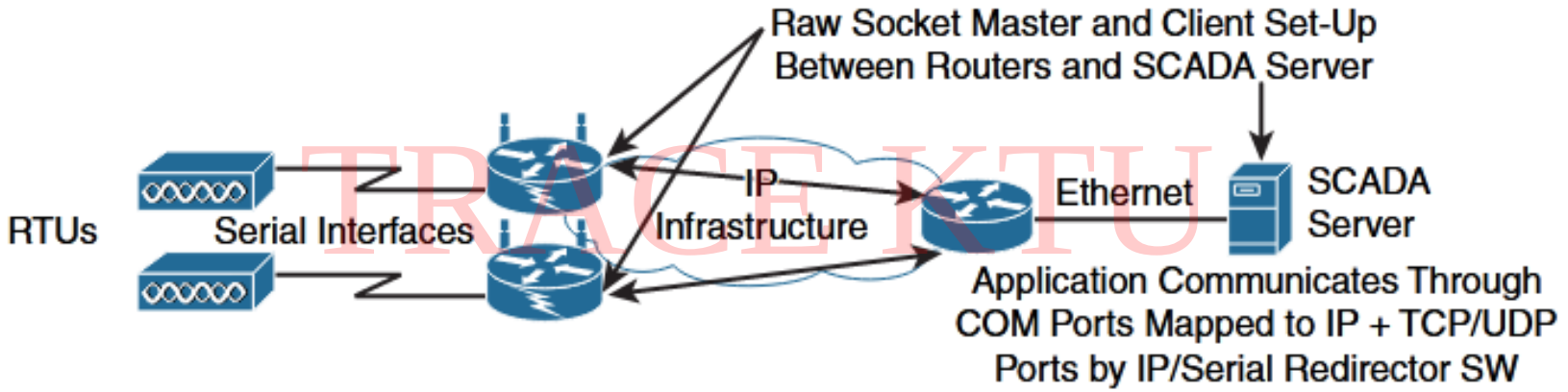
Scenario A



Scenario A: Raw Socket between Routers – no change on SCADA server

Adapting SCADA for IP

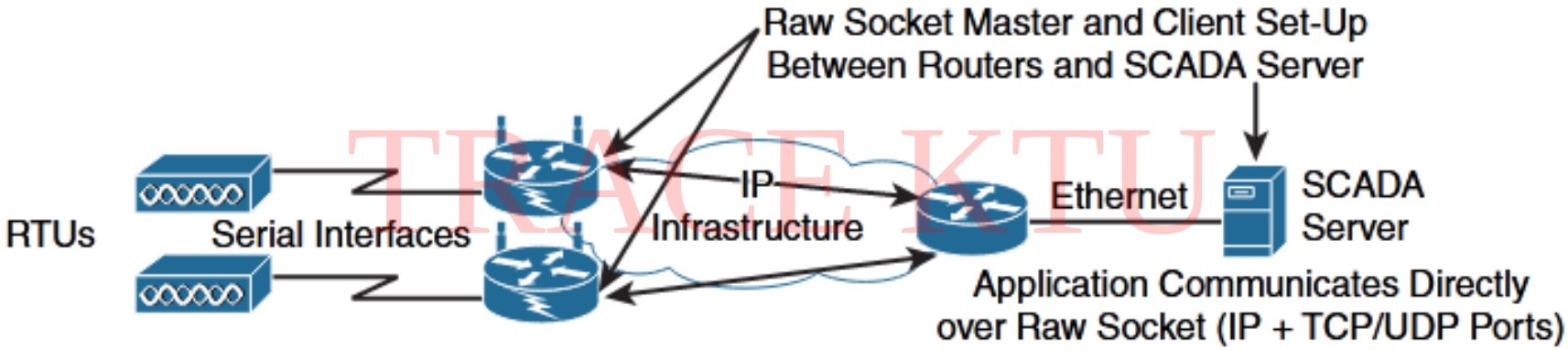
Scenario B



Scenario B: Raw Socket between Router and SCADA Server – no SCADA application change on server but IP/Serial Redirector software and Ethernet interface to be added

Adapting SCADA for IP

Scenario C



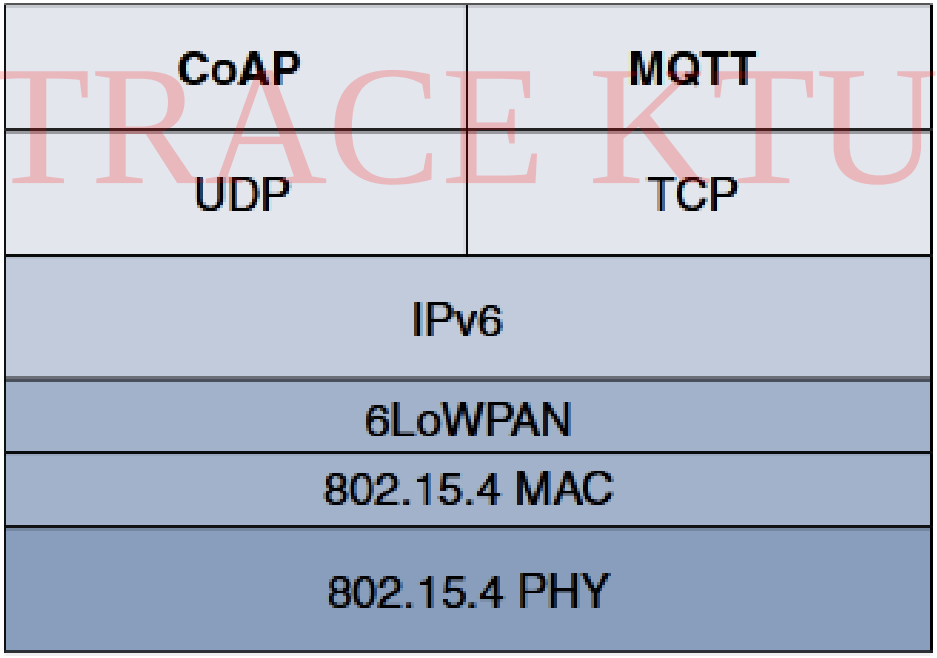
Scenario C: Raw Socket between Router and SCADA Server – SCADA application knows how to directly communicate over a Raw Socket and Ethernet interface

3. Generic Web-Based Protocols

- IoT greatly benefits from the existing web-based protocols such as HTTP/HTTPS, XMPP (Extensible Messaging and Presence Protocol).
- However, to fully address constrained devices and networks, optimized IoT protocols are required.
- The implementation of web services can be either at client or server side.
- IoT devices that only push data to an application may need to implement web services on the client side.
- IoT devices, such as a video surveillance camera, may have web services implemented on the server side.
- Note:- Some applications are collaborative.

4. IoT Application Layer Protocols

- Two of the most popular application layer protocols are CoAP and MQTT.
  - Lightweight protocols for constrained nodes and networks.

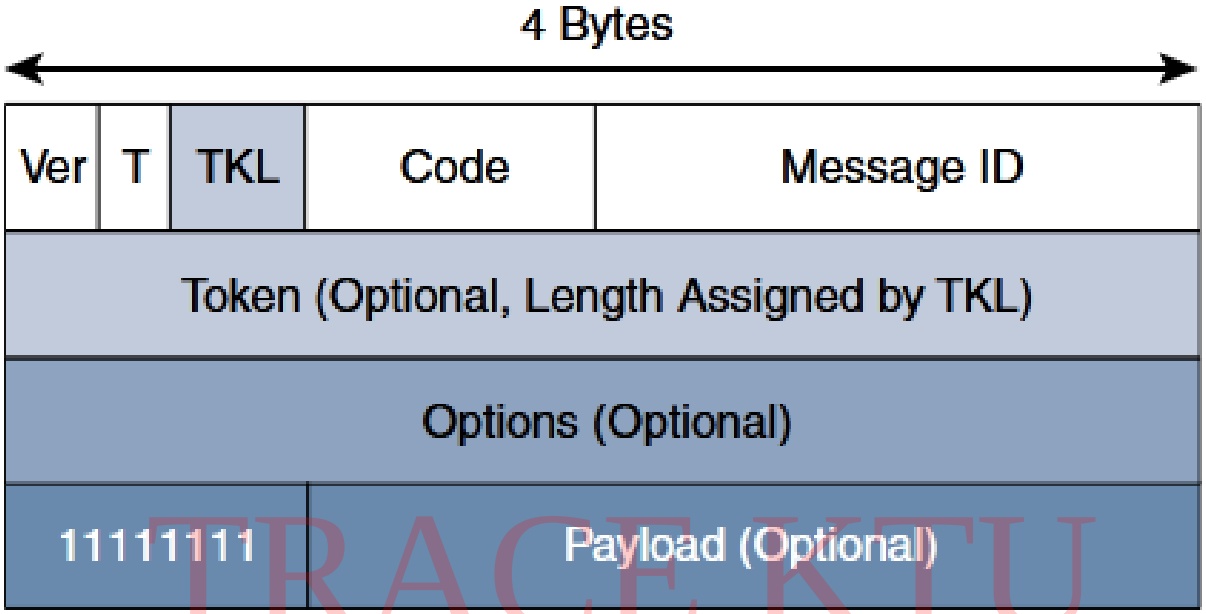


CoAP

- The CoAP framework defines simple and flexible ways to manipulate sensors and actuators for data or device management.
- The CoAP messaging model is primarily designed to exchange the messages over UDP between endpoints, including the Datagram Transport Layer Security (DTLS).

- Four security modes are defined:
  - NoSec, PreSharedKey, RawPublicKey, and Certificate.

CoAP Message Format



- CoAP message is composed of a short fixed-length Header field (4 bytes), a variable-length but mandatory Token field (0–8 bytes), Options fields if necessary, and the Payload field.

CoAP Message Format

- Ver (Vesion): Identifies the CoAP version – 2 bits
- T (Type): Defines one of the following message types: Confirmable (CON), Non-confirmable (NON), Acknowledgement (ACK), or Reset (RST) – 2 bits
- TKL (Token Length): Specifies the size of the Token field – 4 bits
- Code: Indicates the request code for a request message and a response code for a response message (GET, POST, Not Found, Forbidden, etc.) – 8 bits
- Message ID: Detects message duplication and used to match ACK and RST message types to Con and NON message types – 16 bits

CoAP Message Format

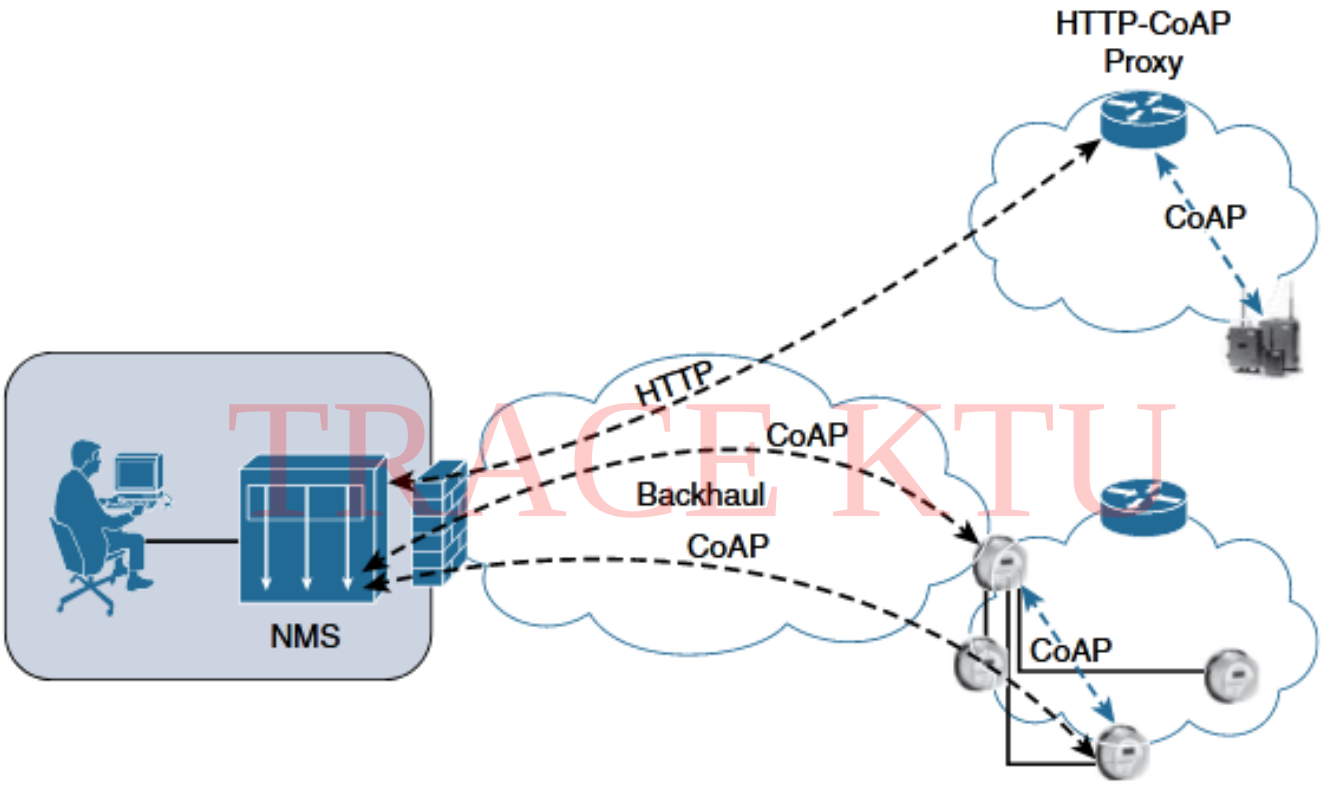
- Token: With a length specified by TKL, correlates requests and responses. Every request carries a token (but it may be zero length) whose value was generated by the client. The server must echo every token value without any modification back to the client in the corresponding response.
- Options: Specifies option number, length, and option value. Capabilities provided by the Options field include specifying the target resource of a request and proxy functions.
- 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. The purpose of this byte is to delineate the end of the Options field and the beginning of Payload.

CoAP Communication

- CoAP can run over IPv4 or IPv6.

- It is recommended that the message fit within a single IP packet and UDP payload to avoid fragmentation.
- While most sensor and actuator traffic utilizes small-packet payloads, some use cases, such as firmware upgrades, require the capability to send larger payloads.
- CoAP doesn't rely on IP fragmentation but defines a pair of Block options for transferring multiple blocks of information from a resource in multiple request/response pairs.

CoAP Communication



CoAP Communication

- Just like HTTP, CoAP is based on the REST architecture, but with a “thing” acting as both the client and the server.
- Here, a client requests an action via a method code on a server resource.
- A uniform resource identifier (URI) localized on the server identifies this resource.
- The server responds with a response code that may include a resource representation.
- The CoAP request/response semantics include the methods GET, POST, PUT, and DELETE.

CoAP Communication

```
coap-URI = "coap:" "//" host [":" port] path-abempty ["?" query]
coaps-URI = "coaps:" "//" host [":" port] path-abempty  ["?" query]
```

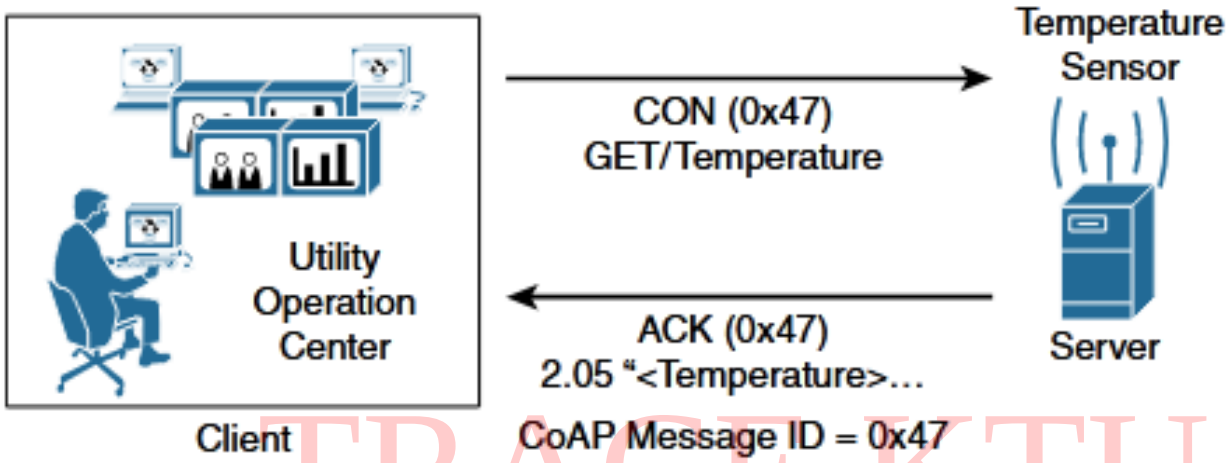
CoAP URI Format

- CoAP defines four types of messages: confirmable, non-confirmable, acknowledgement, and reset.
- Method codes and response codes are used to carry requests or responses.
- CoAP code, method and response codes, option numbers, and content format have been assigned by IANA as Constrained RESTful Environments (CoRE) parameters.

CoAP Communication

- While running over UDP, CoAP offers a reliable transmission of messages when a CoAP header is marked as "confirmable."
- In addition, CoAP supports basic congestion control with a default time-out, simple stop and wait retransmission with exponential back-off mechanism, and detection of duplicate messages through a message ID.
- If a request or response is tagged as confirmable, the recipient must explicitly either acknowledge or reject the message, using the same message ID,
- If a recipient can't process a non-confirmable message, a reset message is sent.

CoAP Communication



The client sends a GET confirmable message to get the temperature from the sensor. The temperature sensor reply with an ACK message referencing the message ID of 0x47. In addition, this ACK message piggybacks a successful response to the GET request itself. This is indicated by the 2.05 response code followed by the requested data.

CoAP Communication

- CoAP supports data requests sent to a group of devices by the use of IP Multicast.
- For that, it requires the use of all-CoAP-node multicast addresses.
- For IPv4 it is 224.0.1.187, and for IPv6 it is FF0X::FD and the port numbers are 5683 for coap and 5684 for coaps.
- Therefore, endpoints can find available CoAP services through multicast service discovery.
  - A typical use case for multicasting is deploying a firmware upgrade for a group of IoT devices, such as smart meters.

CoAP Communication

- Services from a CoAP server can either be discovered by learning a URI in a namespace or through the "All CoAP nodes" multicast address.
- When utilizing the URI scheme for discovering services, the default port 5683 is used for non-secured CoAP, or coap, while port 5684 is utilized for DTLS-secured CoAP, or coaps.
- The CoAP server must be in listening state on these ports.

Message Queuing Telemetry Transport (MQTT)

- At the end of the 1990s, engineers from IBM and Arcom 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.
- Their research resulted in the development and implementation of the Message Queuing Telemetry Transport (MQTT) protocol that is now standardized by the Organization for the Advancement of Structured Information Standards (OASIS).

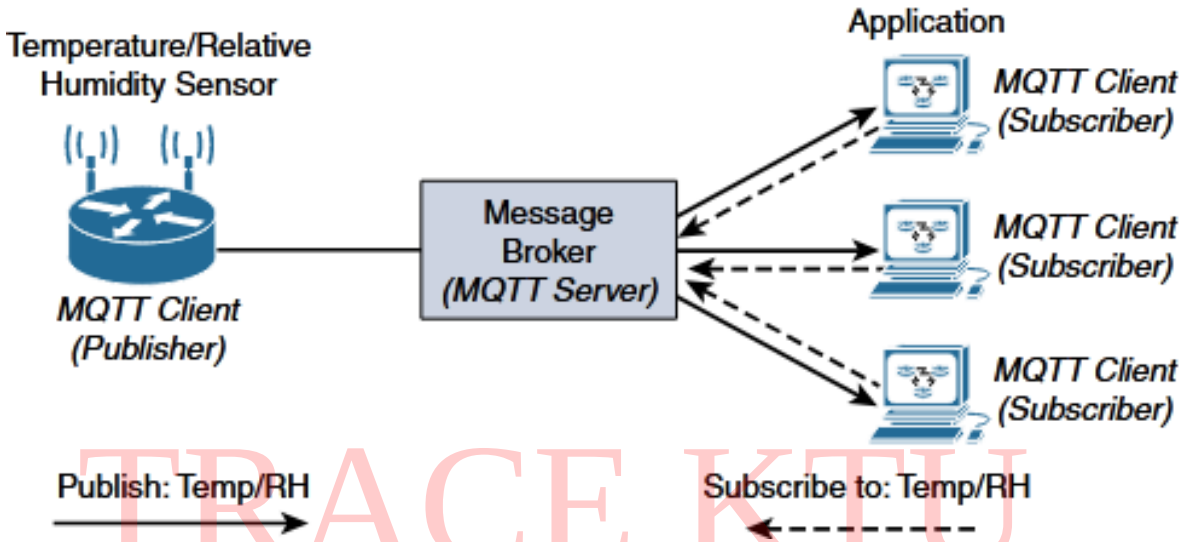
MQTT

- It is a lightweight, publish-subscribe, machine to machine network protocol for message queuing service.

- It is designed for connections with remote locations that have devices with resource constraints or limited network bandwidth, such as in the Internet of Things (IoT).

- It must run over a transport protocol that provides ordered, lossless, bi-directional connections.

MQTT Publish/Subscribe Framework



The MQTT client on the left side is a temperature and relative humidity (RH) sensor that publishes its Temp/RH data. The MQTT server (or message broker) accepts the network connection along with application messages, such as Temp/RH data, from the publishers. It also pushes the application data to MQTT clients acting as subscribers.

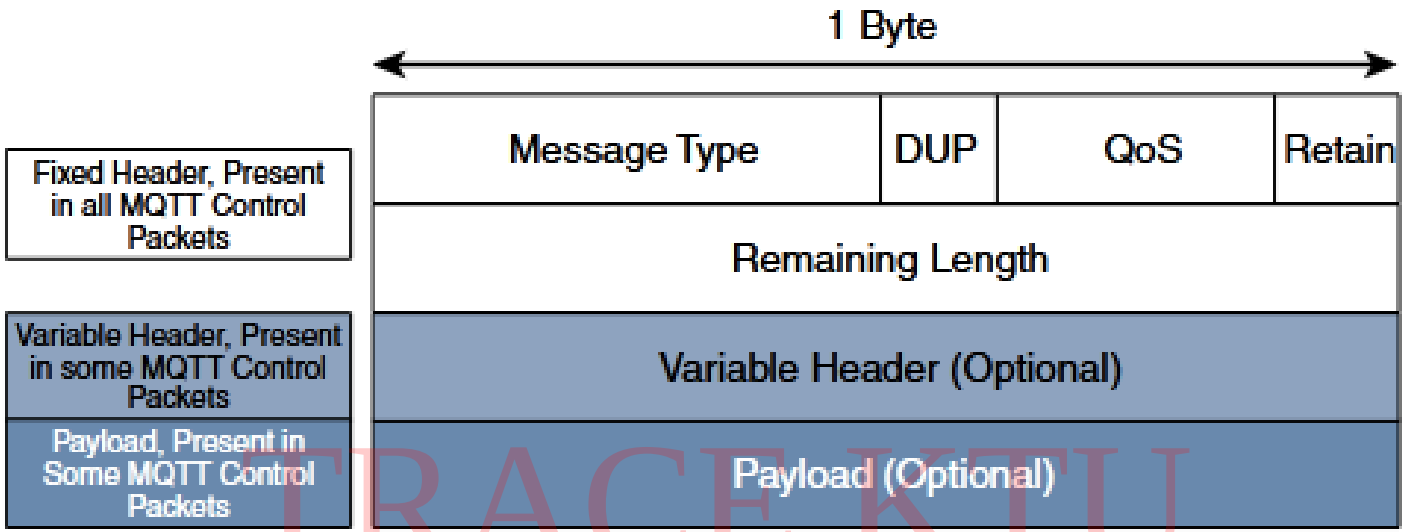
MQTT Publish/Subscribe Framework

- Here clients can subscribe to all data or specific data from the information tree of a publisher.
- The presence of a message broker decouples the data transmission between publisher and subscriber.
  - In fact, publishers and subscribers do not even know about each other.
- A benefit this decoupling is that the MQTT message broker ensures that information can be buffered and cached in case of network failures.
- This also means that publishers and subscribers do not have to be online at the same time.

MQTT Publish/Subscribe Framework

- MQTT control packets run over a TCP transport using port 1883.
- Optionally, MQTT can be secured using TLS on port 8883.
- 14 different types of control packets are specified in MQTT version 3.1.1.
- MQTT is a lightweight protocol because each control packet consists of a 2-byte fixed header with optional variable header fields and optional payload.
  - A control packet can contain a payload up to 256 MB.

MQTT Message Format



- Message Type identifies the kind of MQTT packet within a message (14 different types of control packets). Each of them has a unique value that is coded into the Message Type field. Values 0 and 15 are reserved.

MQTT Message Format

Message Type	Value	Flow	Description
CONNECT	1	Client to server	Request to connect
CONNACK	2	Server to client	Connect acknowledgement
PUBLISH	3	Client to server Server to client	Publish message
PUBACK	4	Client to server Server to client	Publish acknowledgement
PUBREC	5	Client to server Server to client	Publish received
PUBREL	6	Client to server Server to client	Publish release
PUBCOMP	7	Client to server Server to client	Publish complete
SUBSCRIBE	8	Client to server	Subscribe request
SUBACK	9	Server to client	Subscribe acknowledgement
UNSUBSCRIBE	10	Client to server	Unsubscribe request

MQTT Message Format

Message Type	Value	Flow	Description
UNSUBACK	11	Server to client	Unsubscribe acknowledgement
PINGREQ	12	Client to server	Ping request
PINGRESP	13	Server to client	Ping response
DISCONNECT	14	Client to server	Client disconnecting

- DUP (Duplication Flag), when set, allows the client to notate that the packet has been sent previously, but an acknowledgement was not received.
- QoS header field allows for the selection of three different QoS levels (QoS 0, QoS 1, QoS 2).

MQTT Message Format

- Retain flag only found in a PUBLISH message. It notifies the server to hold onto the message data. This allows new subscribers to instantly receive the last known value without having to wait for the next update from the publisher.

- Remaining Length field specifies the number of bytes in the MQTT packet following this field. This is the last mandatory field.

MQTT Communication

- MQTT sessions between each client and server consist of four phases:
  - Session establishment, authentication, data exchange, and session termination.
- Each client connecting to a server has a unique client ID, which allows the identification of the MQTT session between both parties.
- When the server is delivering an application message to more than one client, each client is treated independently.

MQTT Communication

- Subscriptions to resources generate SUBSCRIBE/SUBACK control packets, while un-subscription is performed through the exchange of UNSUBSCRIBE/UNSUBACK control packets.

- Graceful termination of a connection is done through a DISCONNECT control packet, which also offers the capability for a client to reconnect by re-sending its client ID to resume the operations.

MQTT Communication

- A message broker uses a topic string or topic name to filter messages for its subscribers.
- When subscribing to a resource, the subscriber indicates the one or more topic levels that are used to structure the topic name.
- The forward slash (/) in an MQTT topic name is used to separate each level within the topic tree and provide a hierarchical structure to the topic names.
  - For ex: *adt/lora/adeunis/0018B2000000023A*.

MQTT Communication

- Using wildcard symbols, clients can subscribe to a single topic or multiple topics at once.
- The '#' is a multilevel wildcard character that matches any number of levels within a topic.
- For example, subscribing to adt/lora/adeunis/# enables the reception of the whole sub-tree.
- The '+' is a wildcard character that matches only one topic level.
- For example, adt/lora/+ allows access to adt/lora/adeunis/ and adt/lora/abeeway but not to adt/lora/adeunis/0018B20000000E9E.

MQTT Communication

- PINGREQ/PINGRESP control packets are used to validate the connections between the client and server.
- The MQTT protocol offers three levels of quality of service (QoS).
- QoS for MQTT is implemented when exchanging application messages with publishers or subscribers.
- **QoS 0**: This is an unacknowledged data service referred to as "at most once" delivery. The publisher sends its message one time to a server, which transmits it once to the subscribers. No response is sent by the receiver, and no retry is performed by the sender.

MQTT Communication

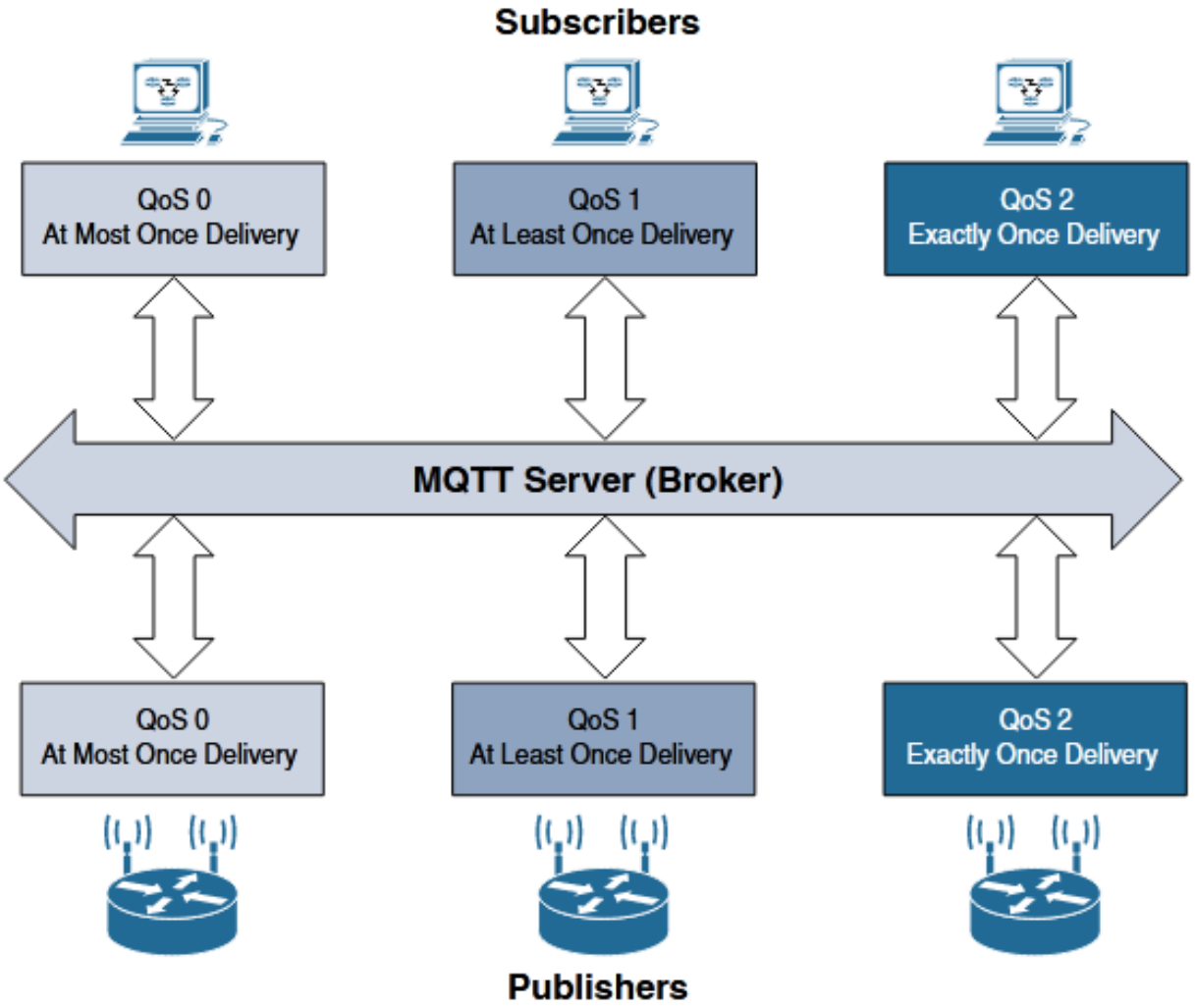
- **QoS 1**: This QoS level ensures that the message delivery between the publisher and server and then between the server and subscribers occurs at least once. In PUBLISH and PUBACK packets, a packet identifier is included in the variable header. If the message is not acknowledged by a PUBACK packet, it is sent again.

MQTT Communication

- **QoS 2**: This is the highest QoS level, used when neither loss nor duplication of messages is acceptable. There is an increased overhead associated with this, because each packet contains an optional header with a packet identifier. Confirming the receipt of a PUBLISH message requires a two-step acknowledgement process. This level provides a "guaranteed service" known as "exactly once" delivery, with no consideration for the number of retries as long as the message is delivered once.

- Note: The QoS process is symmetric. Two separate transactions exist. One transaction occurs between the publishing client and the MQTT server, and the other transaction happens between the MQTT server and the subscribing client.

MQTT Communication



MQTT QoS Flows

Comparison Between CoAP and MQTT

Factor	CoAP	MQTT
Main transport protocol	UDP	TCP
Typical messaging	Request/response	Publish/subscribe
Effectiveness in LLNs	Excellent	Low/fair (Implementations pairing UDP with MQTT are better for LLNs.)
Security	DTLS	SSL/TLS
Communication model	One-to-one	many-to-many
Strengths	Lightweight and fast, with low overhead, and suitable for constrained networks	TCP and multiple QoS options provide robust communications
Weaknesses	Not as reliable as TCP-based MQTT, so the application must ensure reliability.	Higher overhead for constrained devices and networks; TCP connections can drain low-power devices; no multicasting support