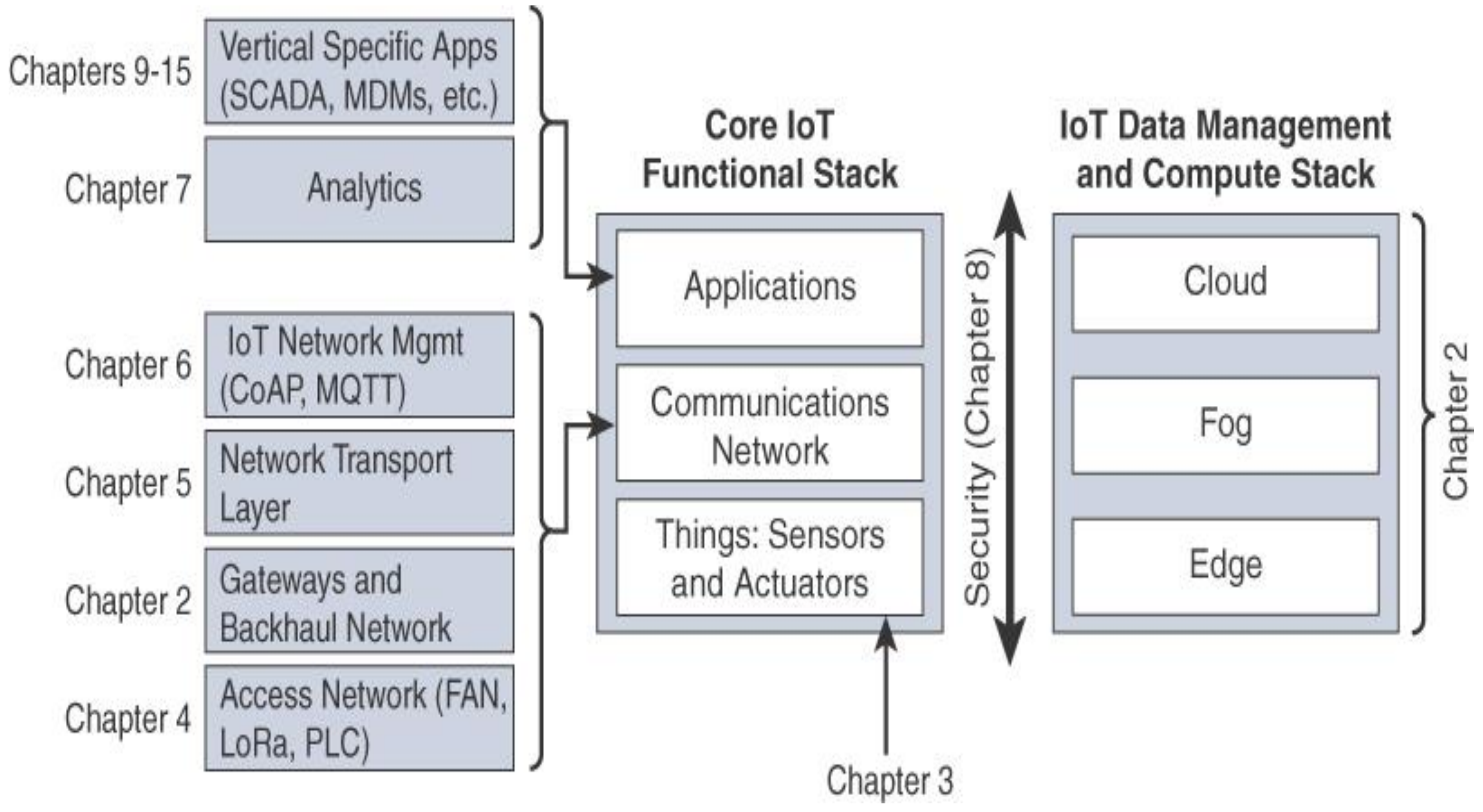
IP as the IoT Network Layer

Module 3

Contents

The Simplified IoT Architecture



Chapters 9-15
Vertical Specific Apps (SCADA, MDMs, etc.)
Chapter 7
Analytics
Chapter 6
IoT Network Mgmt (CoAP, MQTT)
Chapter 5
Network Transport Layer
Chapter 2
Gateways and Backhaul Network
Chapter 4
Access Network (FAN, LoRa, PLC)
Core IoT Functional Stack
Applications
Communications Network
Things: Sensors and Actuators
Chapter 3
Security (Chapter 8)
IoT Data Management and Compute Stack
Cloud
Fog
Edge
Chapter 2

The Business Case for IP

❑**Key Advantages of Internet Protocol**

- **Open and standards-based**
  - guaranteeing interchangeability and interoperability, security, and management
- **Versatile**
  - Physical and data link layers such as Ethernet, Wi-Fi, and cellular are widely adopted, the history of data communications demonstrates that no given wired or wireless technology fits all deployment criteria

Contd..

- **Ubiquitous**
  - All recent operating system releases, from general-purpose computers and servers to lightweight embedded systems (TinyOS, Contiki, and so on), have an integrated dual (IPv4 and IPv6) IP stack that gets enhanced over time
  - In addition, IoT application protocols in many industrial OT solutions have been updated in recent years to run over IP.

Contd..

- **Scalable**
  - As the common protocol of the Internet, IP has been massively deployed and tested for robust scalability.
  - Millions of private and public IP infrastructure nodes have been operational for years
- **Manageable and highly secure**
  - Communications infrastructure requires appropriate management and security capabilities for proper operations
  - Well-understood network management and security protocols, mechanisms, and toolsets that are widely available

Contd..

- **Stable and resilient**
  - IP has been around for 30 years, and it is clear that IP is a workable solution.
  - IP has a large and well-established knowledge base and, more importantly, it has been used for years in critical infrastructures, such as financial and defence networks.
  - In addition, IP has been deployed for critical services, such as voice and video, which have already transitioned from closed environments to open IP standards

Contd..

- **Consumers' 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.
  - The main consumer devices range from smart phones to tablets and PCs.
  - The common protocol that links IoT in the consumer space to these devices is IP

Contd..

- **The innovation factor**
  - The past two decades have largely established the adoption of IP as a factor for increased innovation.
  - IP is the underlying protocol for applications ranging from file transfer and e-mail to the World Wide Web, e-commerce, social networking, mobility, and more.
  - Even the recent computing evolution from PC to mobile and mainframes to cloud services are perfect demonstrations of the innovative ground enabled by IP.

Adoption or Adaptation of the Internet Protocol

- How to implement IP in data center, cloud services, and operation centers hosting IoT applications may seem obvious, but the adoption of IP in the last mile is more complicated and often makes running IP end-to-end more difficult
- *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

Contd..

- The IP adaptation versus adoption model still requires investigation for particular last-mile technologies used by IoT.
- You should consider the following factors when trying to determine which model is best suited for last-mile connectivity:
- **Bidirectional versus unidirectional data flow**

Contd..

- **Overhead for last-mile communications paths**
  - IP adoption implies a layered architecture with a per-packet overhead that varies depending on the IP version
  - IPv4 has 20 bytes of header at a minimum, and IPv6 has 40 bytes at the IP network layer.
  - For the IP transport layer, UDP has 8 bytes of header overhead, while TCP has a minimum of 20 bytes.
  - If the data to be forwarded by a device is infrequent and only a few bytes, you can potentially have more header overhead than device data

Contd..

- **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
  - However, in many IoT solutions, a device's data flow is limited to one or two applications.
  - In this case, the adaptation model can work because translation of traffic needs to occur only between the end device and one or two application servers.

Contd..

- **Network diversity:**
  - One of the drawbacks of the adaptation model is a general dependency on single PHY and MAC layers.
  - For example, ZigBee devices must only be deployed in ZigBee network islands.
  - Therefore, a deployment must consider which applications have to run on the gateway connecting these islands and the rest of the world.
  - This is not a relevant consideration for the adoption model.

The Need for Optimization

- The Internet of Things will largely be built on the Internet Protocol suite.
- However, challenges still exist for IP in IoT solutions.
- In addition to coping with the integration of non-IP devices, you may need to deal with the limits at the device and network levels that IoT often imposes.
- Therefore, optimizations are needed at various layers of the IP stack to handle the restrictions that are present in IoT networks.

Contd..

- **Constrained Nodes**
- In IoT solutions, different classes of devices coexist.
- 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.
- Another limit is that this network protocol stack on an IoT node may be required to communicate through an unreliable path. Even if a full IP stack is available on the node, this causes problems such as limited or unpredictable throughput and low convergence when a topology change occurs.

- Power consumption is a key characteristic of constrained nodes. Many IoT devices are battery powered
- 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:**
  - This drives the need for the IP adaptation model, where nodes communicate through gateways and proxies.
- **Devices with enough power and capacities to implement a stripped-down IP stack or non-IP stack:**
  - In this case, you may implement either an optimized IP stack and directly communicate with application servers (adoption model) or go for an IP or non-IP stack and communicate through gateways and proxies (adaptation model).

- **Devices that are similar to generic PCs in terms of computing and power resources but have constrained networking capacities, such as bandwidth:**
  - These nodes usually implement a full IP stack (adoption model), but network design and application behaviors must cope with the bandwidth constraints.

Constrained Networks

- In the early years of the Internet, network bandwidth capacity was restrained due to technical limitations.
- Connections often depended on low-speed modems for transferring data.
- However, these low-speed connections demonstrated that IP could run over low-bandwidth networks
- However, high-speed connections are not usable by some IoT devices in the last mile.
- The reasons include the implementation of technologies with low bandwidth, limited distance and bandwidth due to regulated transmit power, and lack of or limited network services
- A constrained network can have high latency and a high potential for packet loss

- Constrained networks have unique characteristics and requirements
- Constrained networks are limited by low-power, low-bandwidth links
- With a constrained network, in addition to limited bandwidth, it is not unusual for the packet delivery rate (PDR) to oscillate between low and high percentages.
- Large bursts of unpredictable errors and even loss of connectivity at times may occur.
- You have to consider the power consumption in battery-powered nodes

IP Versions

- For 20+ years, 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.
- Today, both versions of IP run over the Internet, but most traffic is still IPv4 based

Contd…

- IoT should support both IPv4 and IPv6 versions concurrently.
- Techniques such as tunneling and translation need to be employed in IoT solutions to ensure interoperability between IPv4 and IPv6
- A variety of factors dictate whether IPv4, IPv6, or both can be used in an IoT solution.
- Most often these factors include a legacy protocol or technology that supports only IPv4.
- Newer technologies and protocols almost always support both IP versions.

Contd…

- 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 dictate the choice of the IP version
- For example, SCADA protocols such as DNP3/IP (IEEE 1815), Modbus TCP, or the IEC 60870- 5-104 standards are specified only for IPv4
- The selection of the IP version is only dependent on the implementation.

Contd…

- **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.
- For the first three generations of data services—GPRS, Edge, and 3G—IPv4 is the base protocol version.
- Consequently, 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

Contd…

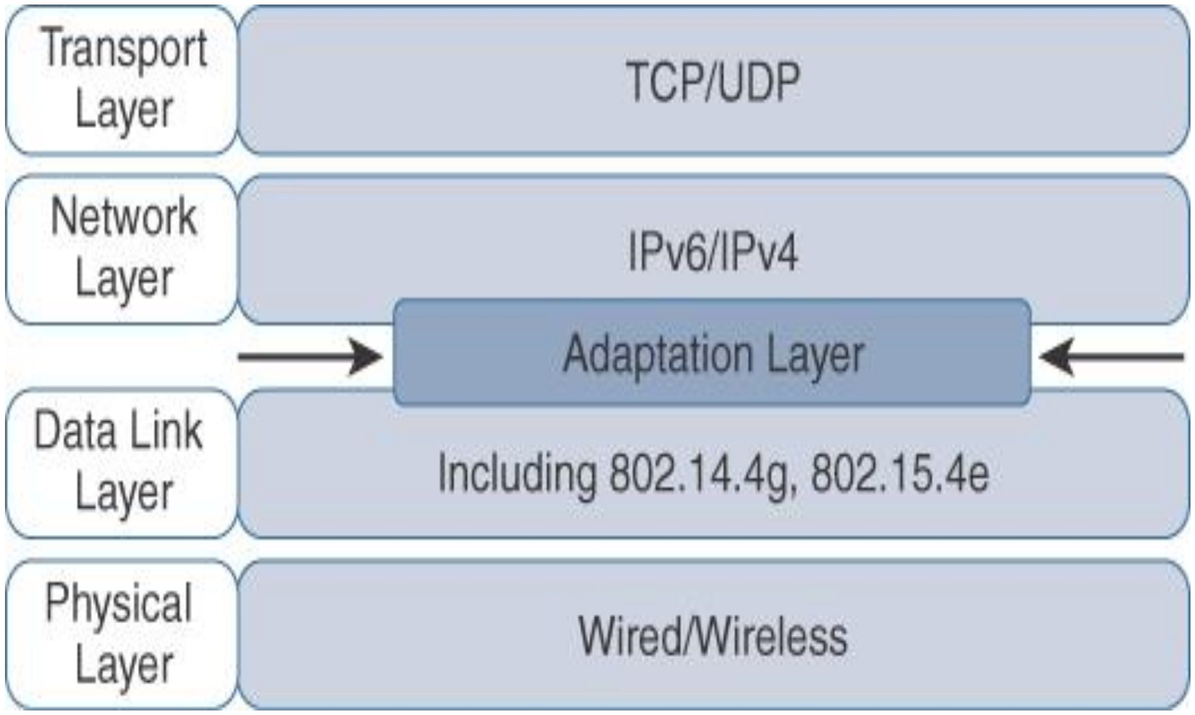
- **Serial Communications**
- Many legacy devices in certain industries, such as manufacturing and utilities, communicate through serial lines
- Data is transferred using either proprietary or standards-based protocols, such as DNP3, Modbus, or IEC 60870-5-101
- Communicating this serial data over any sort of distance could be handled by an analog modem connection
- 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.

Contd…

- **IPv6 Adaptation Layer**
- IPv6-only adaptation layers for some physical and data link layers for recently standardized IoT protocols support only IPv6.
- This means that any device implementing a technology that requires an IPv6 adaptation layer must communicate over an IPv6-only subnetwork.
- This is reinforced by the IETF routing protocol for LLNs, RPL, which is IPv6 only

Optimizing IP for IoT

- While 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



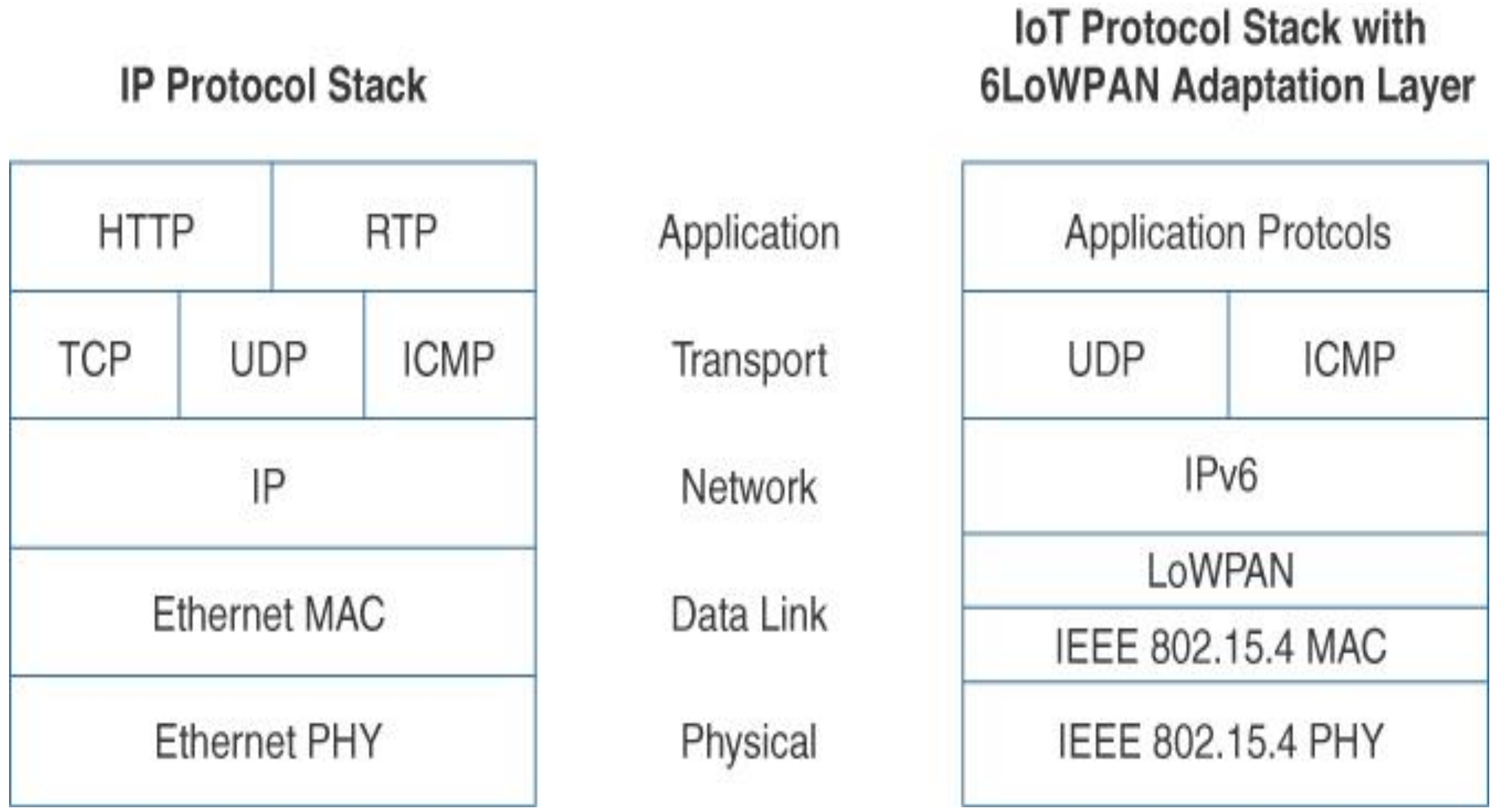
From 6LoWPAN to 6Lo

- 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 packaging IP into lower-layer protocols is often referred to as an *adaptation layer.*
- Unless the technology is proprietary, IP adaptation layers are typically defined by an IETF working group and released as a Request for Comments (RFC).
- An RFC is a publication from the IETF that officially documents Internet standards, specifications, protocols, procedures, and events.
- For example, RFC 864 describes how an IPv4 packet gets encapsulated over an Ethernet frame, and RFC 2464 describes how the same function is performed for an IPv6 packet.

Contd…

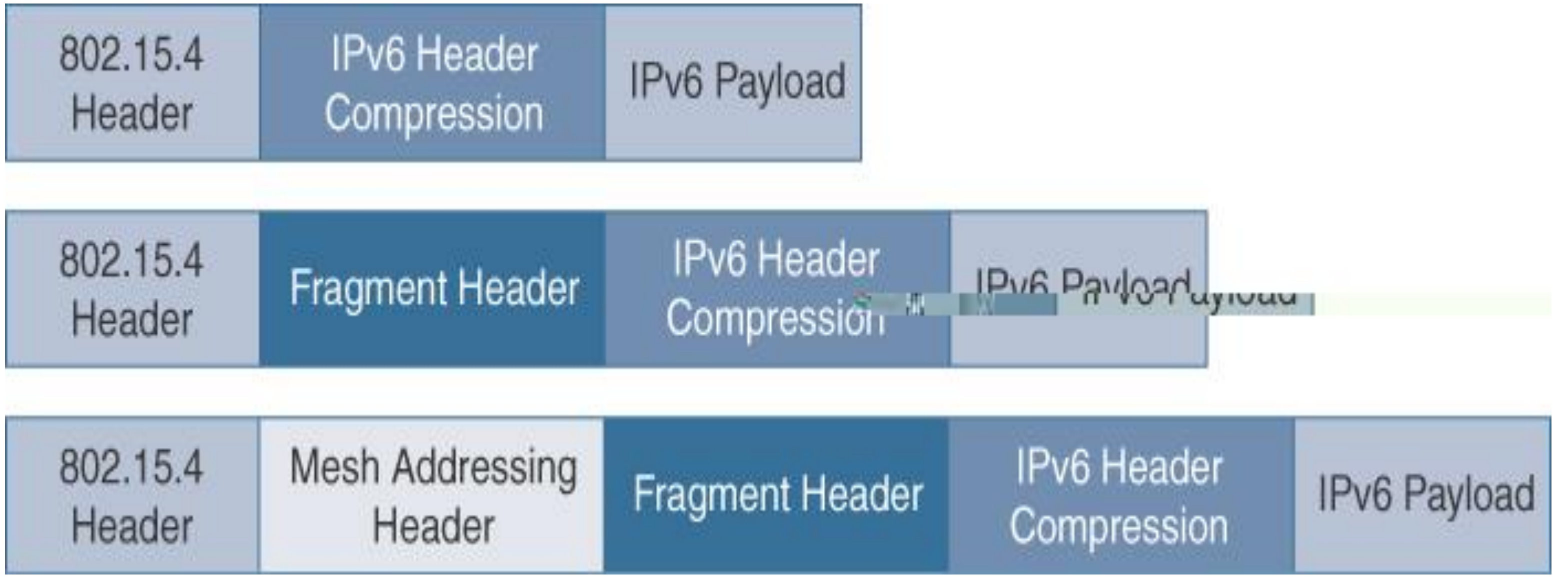
- IoT-related protocols follow a similar process.
- The main difference is that an adaptation layer designed for IoT may include some optimizations to deal with constrained nodes and networks
- The main examples of adaptation layers optimized for constrained nodes or "things" are the ones under the 6LoWPAN working group and its successor, the 6Lo working group.
- The initial focus of the 6LoWPAN working group was to optimize the transmission of IPv6 packets over constrained networks such as IEEE 802.15.4

IoT protocol stack using the 6LoWPAN adaptation layer



Contd…

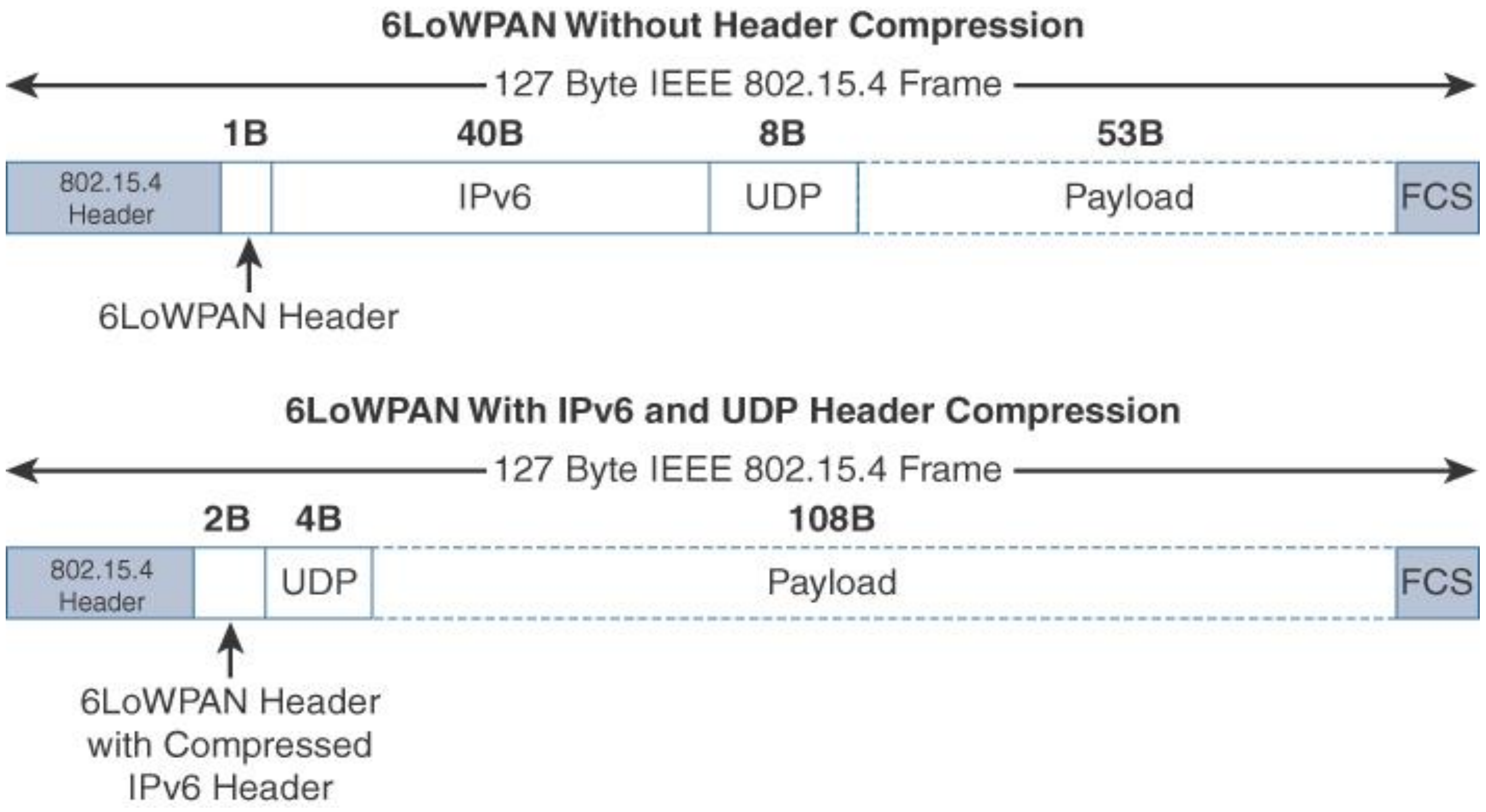
- The 6LoWPAN working group published several RFCs, but RFC 4994 is foundational because it defines frame headers for the capabilities of header compression, fragmentation, and mesh addressing



Header Compression

- IPv6 header compression for 6LoWPAN was defined initially in RFC 4944 and subsequently updated by RFC 6282.
- This capability shrinks the size of IPv6's 40-byte headers and User Datagram Protocol's (UDP's) 8-byte headers down as low as 6 bytes combined in some cases.
- Note that header compression for 6LoWPAN is only defined for an IPv6 header and not IPv4.
- The 6LoWPAN protocol does not support IPv4

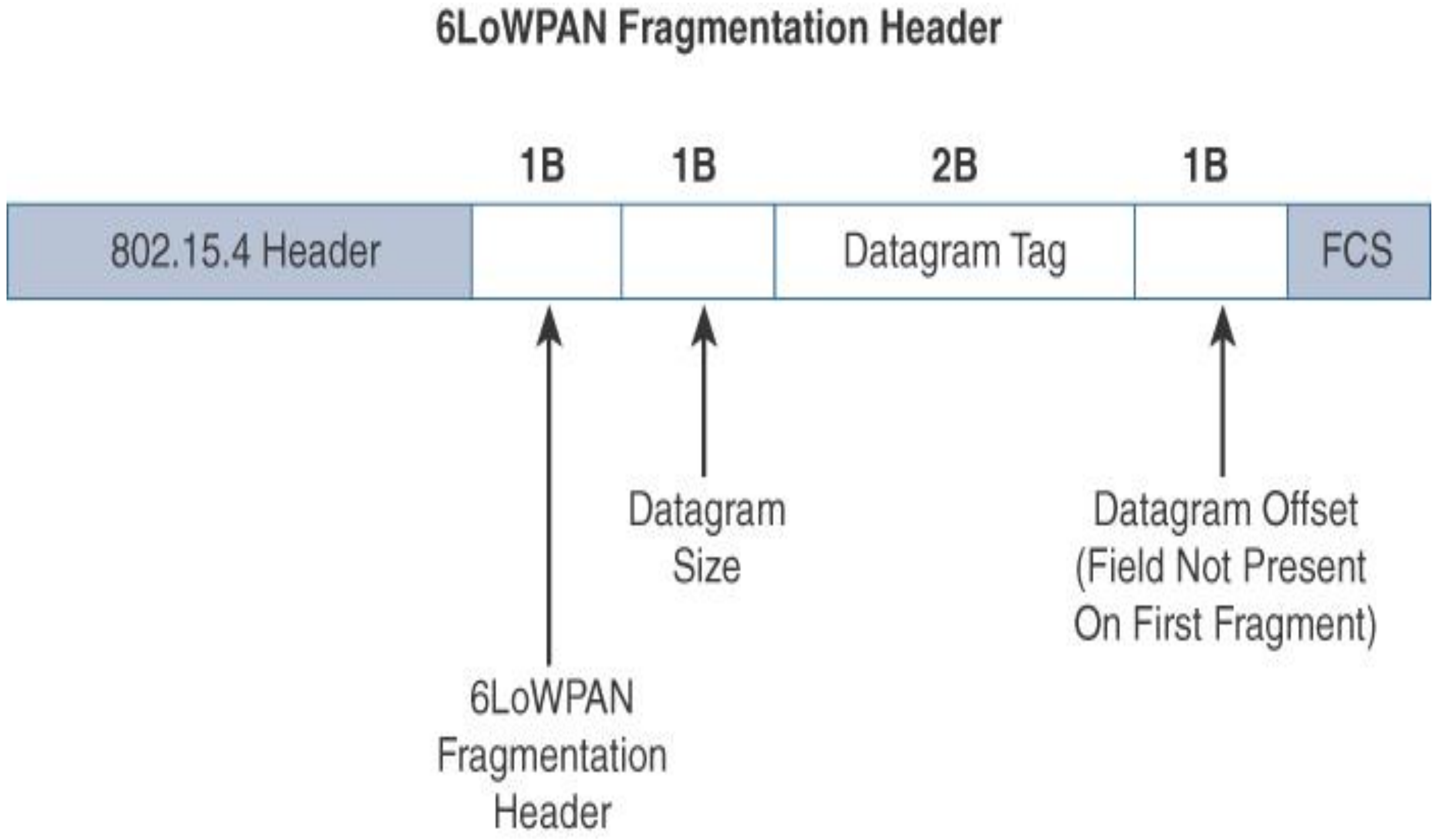
- At a high level, 6LoWPAN works by taking advantage of shared information known by all nodes from their participation in the local network.
- In addition, it omits some standard header fields by assuming commonly used values


6LoWPAN Without Header Compression
127 Byte IEEE 802.15.4 Frame
1B
40B
8B
53B
802.15.4 Header
IPv6
UDP
Payload
FCS
6LoWPAN Header
6LoWPAN With IPv6 and UDP Header Compression
127 Byte IEEE 802.15.4 Frame
2B
4B
108B
802.15.4 Header
UDP
Payload
FCS
6LoWPAN Header with Compressed IPv6 Header

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.
- For IEEE 802.15.4, 127 bytes is the MTU
- You can see that this is a problem because IPv6, with a much larger MTU, is carried inside the 802.15.4 frame with a much smaller one.
- To remedy this situation, large IPv6 packets must be fragmented across multiple 802.15.4 frames at Layer 2.

Contd…


6LoWPAN Fragmentation Header
1B
1B
2B
1B
802.15.4 Header
Datagram Tag
FCS
Datagram Size
Datagram Offset (Field Not Present On First Fragment)
6LoWPAN Fragmentation Header

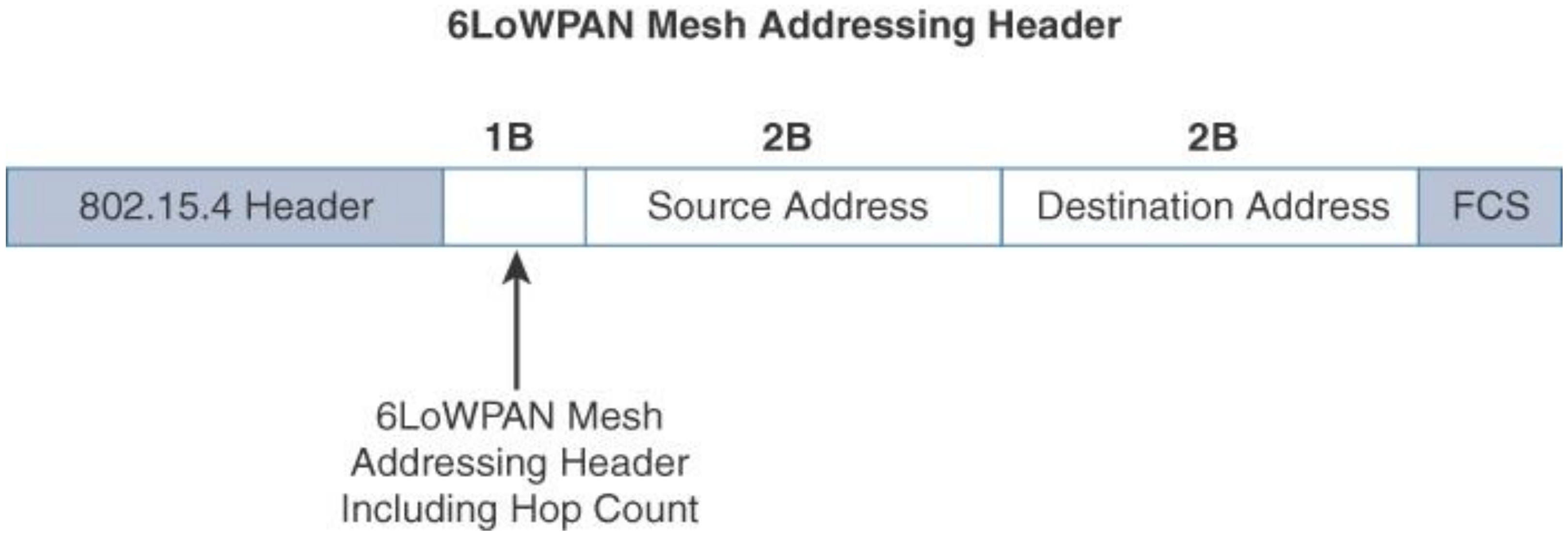
Contd…

- The fragment header utilized by 6LoWPAN is composed of three primary fields:
- Datagram Size, Datagram Tag, and Datagram Offset.
- The 1-byte Datagram Size field specifies the total size of the unfragmented payload.
- Datagram Tag identifies the set of fragments for a payload.
- Finally, the Datagram Offset field delineates how far into a payload a particular fragment occurs

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
- Hop limit for mesh addressing 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 an IP hop

Contd…


6LoWPAN Mesh Addressing Header
1B
2B
2B
802.15.4 Header
Source Address
Destination Address
FCS
6LoWPAN Mesh Addressing Header Including Hop Count

Mesh-Under Versus Mesh-Over Routing

- For network technologies such as IEEE 802.15.4, IEEE 802.15.4g, and IEEE 1901.2a that support mesh topologies and operate at the physical and data link layers, two main options exist for establishing reachability and forwarding packets.
- With the first option, mesh-under, the routing of packets is handled at the 6LoWPAN adaptation layer.
- The other option, known as "mesh-over" or "route-over," utilizes IP routing for getting packets to their destination

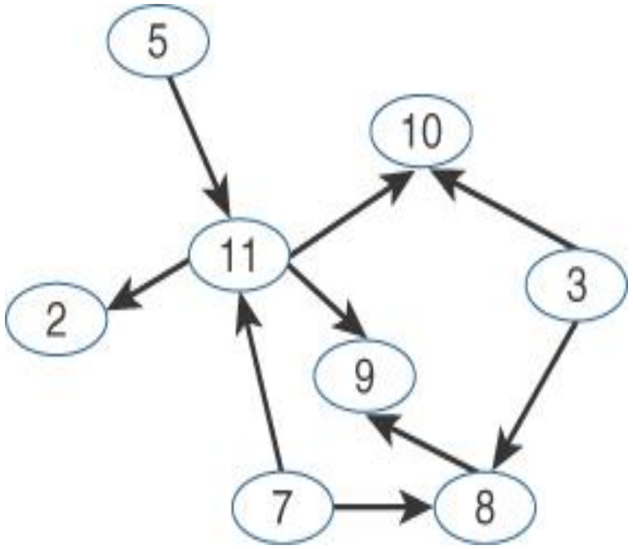
RPL

- The IETF chartered the RoLL (Routing over Low-Power and Lossy Networks) working group to evaluate all Layer 3 IP routing protocols and determine the needs and requirements for developing a routing solution for IP smart objects
- This new distance-vector routing protocol was named the IPv6 Routing Protocol for Low Power and Lossy Networks (RPL).
- In an RPL network, each node acts as a router and becomes part of a mesh network.
- Routing is performed at the IP layer. Each node examines every received IPv6 packet and determines the next-hop destination based on the information contained in the IPv6 header

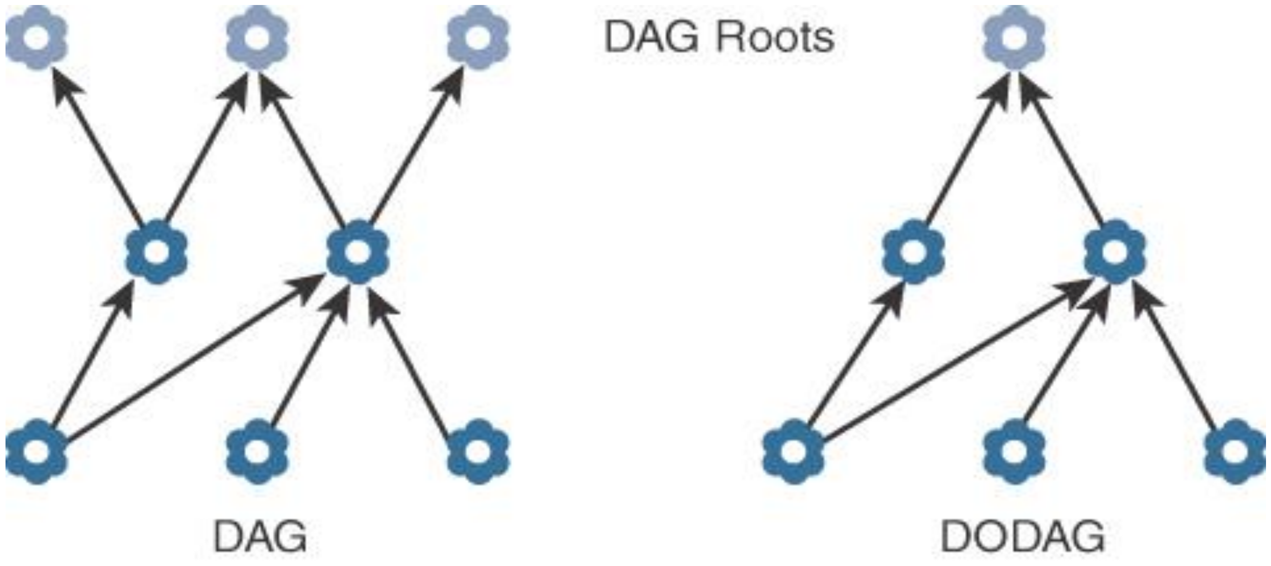
Contd…

- 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.
  - This abbreviated routing table saves memory space and CPU.

- RPL is based on the concept of a directed acyclic graph (DAG).
- A DAG is a directed graph where no cycles exist.
- This means that from any vertex or point in the graph, you cannot follow an edge or a line back to this same point. All of the edges are arranged in paths oriented toward and terminating at one or more root nodes



- A basic RPL process involves building a destination-oriented directed acyclic graph (DODAG).
- A DODAG is a DAG rooted to one destination.
- In RPL, this destination occurs at a border router known as the DODAG root.
- *DAG and DODAG Comparison*



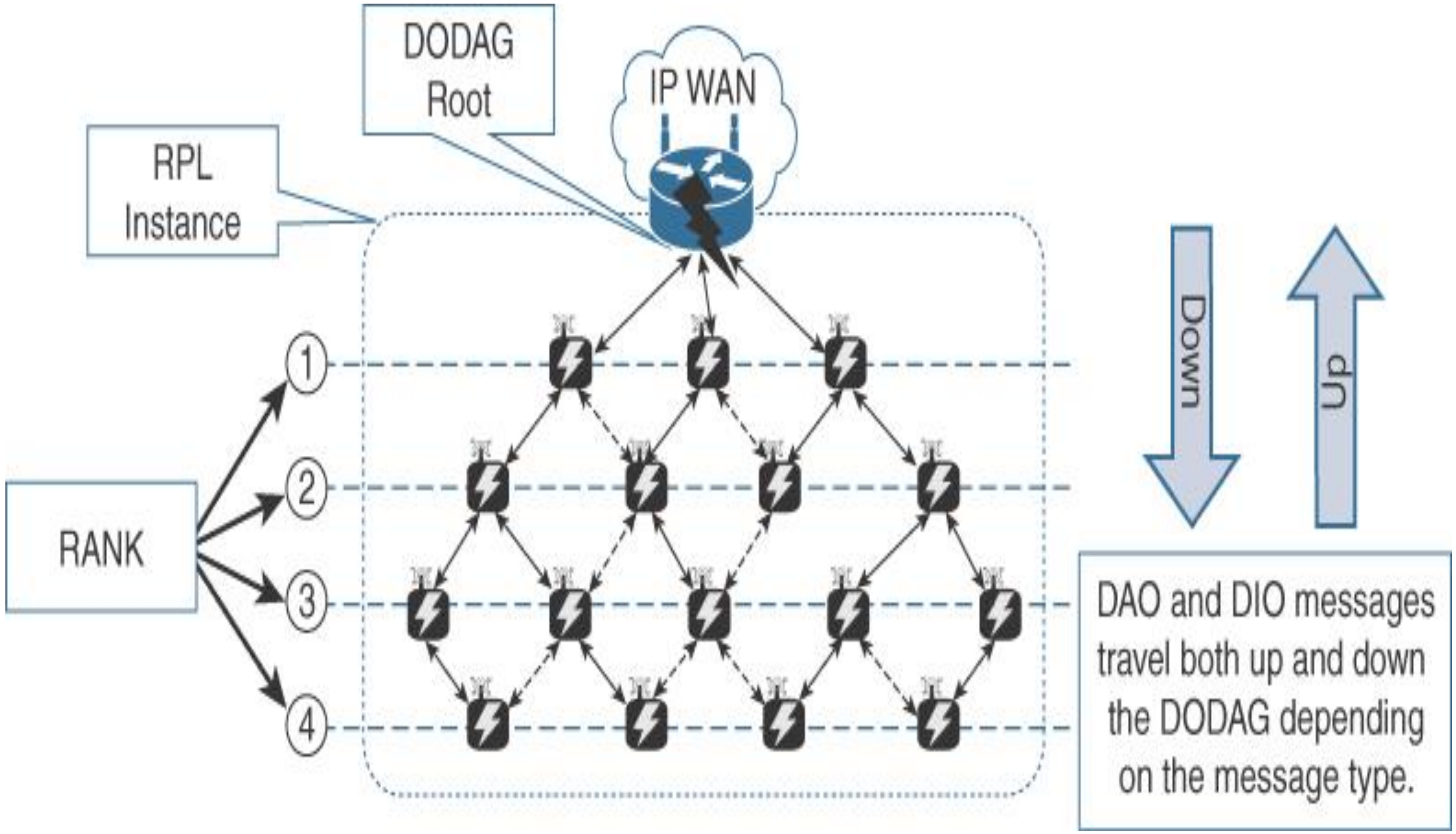
- In a DODAG, each node maintains up to three parents that provide a path to the root.
- Typically, one of these parents is the preferred parent, which means it is the preferred next hop for upward routes toward the root
- The routing graph created by the set of DODAG parents across all nodes defines the full set of upward routes.
- RPL protocol implementation should ensure that routes are loop free

- Upward routes in RPL are discovered and configured using DAG Information Object (DIO) messages.
- Nodes listen to DIOs to handle changes in the topology that can affect routing.
- The information in DIO messages determines parents and the best path to the DODAG root
- Nodes establish downward routes by advertising their parent set toward the DODAG root using a Destination Advertisement Object (DAO) message.
- DAO messages allow nodes to inform their parents of their presence and reachability to descendants

- In the case of the non-storing mode of RPL, nodes sending DAO messages report their parent sets directly to the DODAG root (border router), and only the root stores the routing information.
- The root uses the information to then determine source routes needed for delivering IPv6 datagrams to individual nodes downstream in the mesh.

- For storing mode, each node keeps track of the routing information that is advertised in the DAO messages.
- While this is more power- and CPU-intensive for each node, the benefit is that packets can take shorter paths between destinations in the mesh.
- The nodes can make their own routing decisions; in non-storing mode, on the other hand, all packets must go up to the root to get a route for moving downstream

RPL Overview


DODAG Root
IP WAN
RPL Instance
RANK
1
2
3
4
Down
Up
DAO and DIO messages travel both up and down the DODAG depending on the message type.

Objective Function (OF)

- An objective function (OF) defines how metrics are used to select routes and establish a node's rank
- For example, nodes implementing an OF based on RFC 6719's Minimum Expected Number of Transmissions (METX) advertise the METX among their parents in DIO messages.

Rank

- The rank is a rough approximation of how "close" a node is to the root and helps avoid routing loops and the count-to-infinity problem.
- Nodes can only increase their rank when receiving a DIO message with a larger version number.
- However, nodes may decrease their rank whenever they have established lowercost routes.

Metrics

- RPL defines a large and flexible set of new metrics and constraints for routing in RFC 6551.
- Developed to support powered and battery-powered nodes, RPL offers a far more complete set than any other routing protocol.
- Some of the RPL routing metrics and constraints defined in RFC 6551 include the following:
- **Expected Transmission Count (ETX)**
- Assigns a discrete value to the number of transmissions a node expects to make to deliver a packet.

Contd…

- **Hop Count:**
  - Tracks the number of nodes traversed in a path. Typically, a path with a lower hop count is chosen over a path with a higher hop count.
- **Latency:**
  - Varies depending on power conservation. Paths with a lower latency are preferred.
- **Link Quality Level:**
  - Measures the reliability of a link by taking into account packet error rates caused by factors such as signal attenuation and interference.
- **Throughput:**
  - Provides the amount of throughput for a node link

Contd…

- **Link Color:**
  - Allows manual influence of routing by administratively setting values to make a link more or less desirable. These values can be either statically or dynamically adjusted for specific traffic types.
- **Node State and Attribute:**
  - Identifies nodes that function as traffic aggregators and nodes that are being impacted by high workloads.
- **Node Energy:**
  - Avoids nodes with low power, so a battery-powered node that is running out of energy can be avoided and the life of that node and the network can be prolonged.

Application Protocols for IoT

THE TRANSPORT LAYER

- This section reviews the selection of a protocol for the transport layer as supported by the TCP/IP architecture in the context of IoT networks.
- With the TCP/IP protocol, 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 and destination—but with no guarantee of delivery.

Contd…

- 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 addition, it ensures reassembly in a correct sequence, flow control and window adjustment, and retransmission of lost packets.
- These benefits occur with the cost of overhead per packet and per session, potentially impacting overall packet per second performances and latency.

Contd…

- In contrast, UDP is most often used in the context of network services, such as 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.
- In these cases, performance and latency are more important than packet retransmissions because re-sending a lost voice or video packet does not add value.
- When the reception of packets must be guaranteed error free, the application layer protocol takes care of that function

Contd…

- When considering the choice of a transport layer by a given IoT application layer protocol, it is recommended to evaluate the impact of this choice on both the lower and upper layers of the stack
- For example, most of the industrial application layer protocols, are implemented over TCP, while their specifications may offer support for both transport models. The reason for this is that often these industrial application layer protocols are older and were deployed when data link layers were often unreliable and called for error protection

Contd...

- The use of TCP may not be good when an IoT device needs to send only a few bytes of data per transaction.
- When using TCP, each packet needs to add a minimum of 20 bytes of TCP overhead, while UDP adds only 8 bytes.
- TCP also requires the establishment and potential maintenance of an open logical channel
- IoT nodes may also be limited by the intrinsic characteristics of the data link layers. For example, low-power and lossy networks (LLNs), may not cope well with supporting large numbers of TCP sessions

Contd…

- Multicast requirements are also impacted by the protocol selected for the transport layer.
- With multicast, a single message can be sent to multiple IoT devices.
- This is useful in the IoT context for upgrading the firmware of many IoT devices at once.
- Multicast utilizes UDP exclusively.

IoT Application Transport Methods

- Because of the diverse types of IoT application protocols, there are various means for transporting these protocols across a network.
- Sometimes you may be dealing with legacy utility and industrial IoT protocols that have certain requirements, while other times you might need to consider the transport requirements of more modern application layer protocols

Categories of IoT application protocols

- **Application layer protocol not present**
  - In this case, the data payload is directly transported on top of the lower layers. No application layer protocol is used
- **Supervisory control and data acquisition (SCADA)**
  - SCADA is one of the most common industrial protocols in the world, but it was developed long before the days of IP, and it has been adapted for IP networks.

Categories of IoT application protocols

- **Generic web-based protocols**
  - Generic protocols, such as Ethernet, Wi-Fi, and 4G/LTE, are found on many consumer- and enterprise-class IoT devices that communicate over non-constrained networks

- **IoT application layer protocols**
  - IoT application layer protocols are devised to run on constrained nodes with a small compute footprint and are well adapted to the network bandwidth constraints
  - Example - Message Queuing Telemetry Transport (MQTT) and Constrained Application Protocol (CoAP)

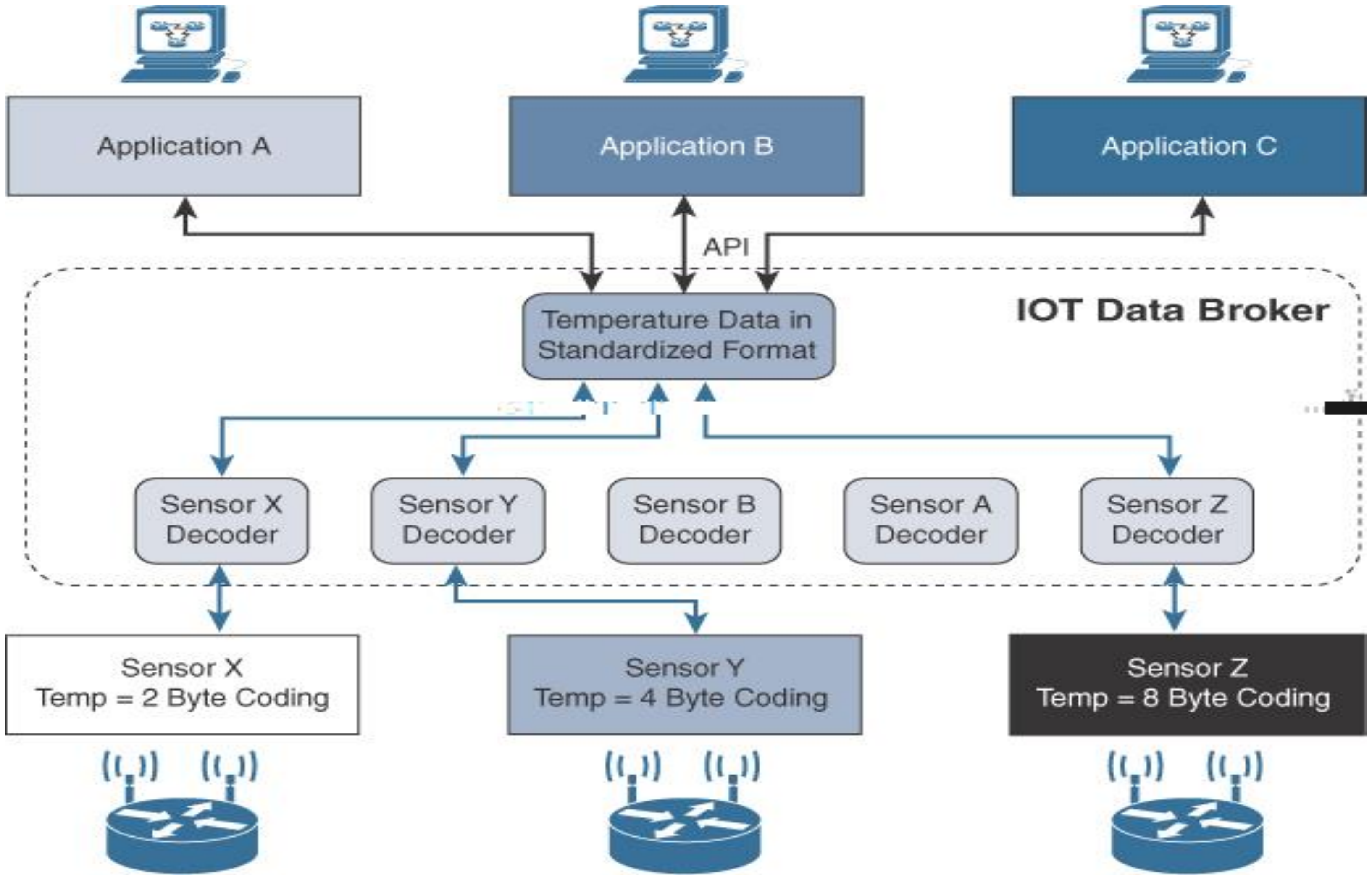
Application Layer Protocol Not Present

- Devices defined as class 0 send or receive only a few bytes of data
- Processing capability, power constraints, and cost, these devices do not implement a fully structured network protocol stack, such as IP, TCP, or UDP, or even an application layer protocol
- Class 0 devices are usually simple smart objects that are severely constrained.
- Implementing a robust protocol stack is usually not useful and sometimes not even possible with the limited available resources

Contd…

- For example, consider low-cost temperature and relative humidity (RH) sensors sending data over an LPWA LoRaWAN infrastructure
- Temperature is represented as 2 bytes and RH as another 2 bytes of data.
- Therefore, this small data payload is directly transported on top of the LoRaWAN MAC layer, without the use of TCP/IP.

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
- Sensors X, Y, and Z are all temperature sensors, but their output is encoded differently.
- The IoT data broker understands the different formats in which the temperature is encoded and is therefore able to decode this data into a common, standardized format.
- Applications A, B, and C can access this temperature data without having to deal with decoding multiple temperature data formats.

- In summary, while directly transporting data payload without a structured network stack clearly optimizes data transmission over low-data-rate networks, the lack of a data model implies that each application needs to know how to interpret the data-specific format.
- This becomes increasingly complex for larger networks of devices with different data payload formats.
- Furthermore, it makes the IoT application environment challenging in terms of evolution, development, interoperability, and so on, and often calls for structured data models and data broker applications

SCADA-supervisory control and data acquisition

- Designed decades ago, SCADA is an automation control system that was initially implemented without IP over serial links, before being adapted to Ethernet and IPv4.

- At a high level, SCADA systems collect sensor data and telemetry from remote devices, while also providing the ability to control them.

- Used in today's networks, SCADA systems allow global, real-time, data-driven decisions to be made about how to improve business processes.

SCADA-supervisory control and data acquisition

- SCADA networks can be found across various industries, but mainly concentrated in the utilities and manufacturing /industrial verticals.
- Within these specific industries, SCADA commonly uses certain protocols for communications between devices and applications.
- For example, Modbus and its variants are industrial protocols used to monitor and program remote devices via a master/slave relationship.

SCADA-supervisory control and data acquisition

- Modbus is also found in building management, transportation and energy applications.

- The DNP3 and International Electro technical Commission (IEC) 60870-5-101 protocols are found mainly in the utilities industry, along with DLMS/COSEM and ANSI C12 for advanced meter reading (AMR).

SCADA-supervisory control and data acquisition

- Adapting SCADA for IP:
- In the 1990s, the rapid adoption of Ethernet networks in the industrial world drove the evolution of SCADA application layer protocols.

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

SCADA-supervisory control and data acquisition

- This included assigning TCP/UDP port numbers to the protocols, such as the following:

1. DNP3 (adopted by IEEE 1815-2012) specifies the use of TCP or UDP on port 20000 for transporting DNP3 messages over IP .

2. The Modbus messaging service utilizes TCP port 502.

3. IEC 60870-5-104 is the evolution of IEC 60870-5-101 serial for running over Ethernet and IPv4 using port 2404.

4. DLMS User Association specified a communication profile based on TCP/IP in the DLMS/COSEM Green Book (Edition 5 or higher), or in the IEC 62056-53 and IEC 62056-47 standards, allowing data exchange via IP and port 4059.

SCADA-supervisory control and data acquisition

- These legacy serial protocols have adapted and evolved to utilize IP and TCP/UDP as both networking and transport mechanisms.

- This has allowed utilities and other companies to continue leveraging their investment in equipment and infrastructure, supporting these legacy protocols with modern IP networks

SCADA-supervisory control and data acquisition

- These legacy serial protocols have adapted and evolved to utilize IP and TCP/UDP as both networking and transport mechanisms.

- This has allowed utilities and other companies to continue leveraging their investment in equipment and infrastructure, supporting these legacy protocols with modern IP networks

SCADA-supervisory control and data acquisition

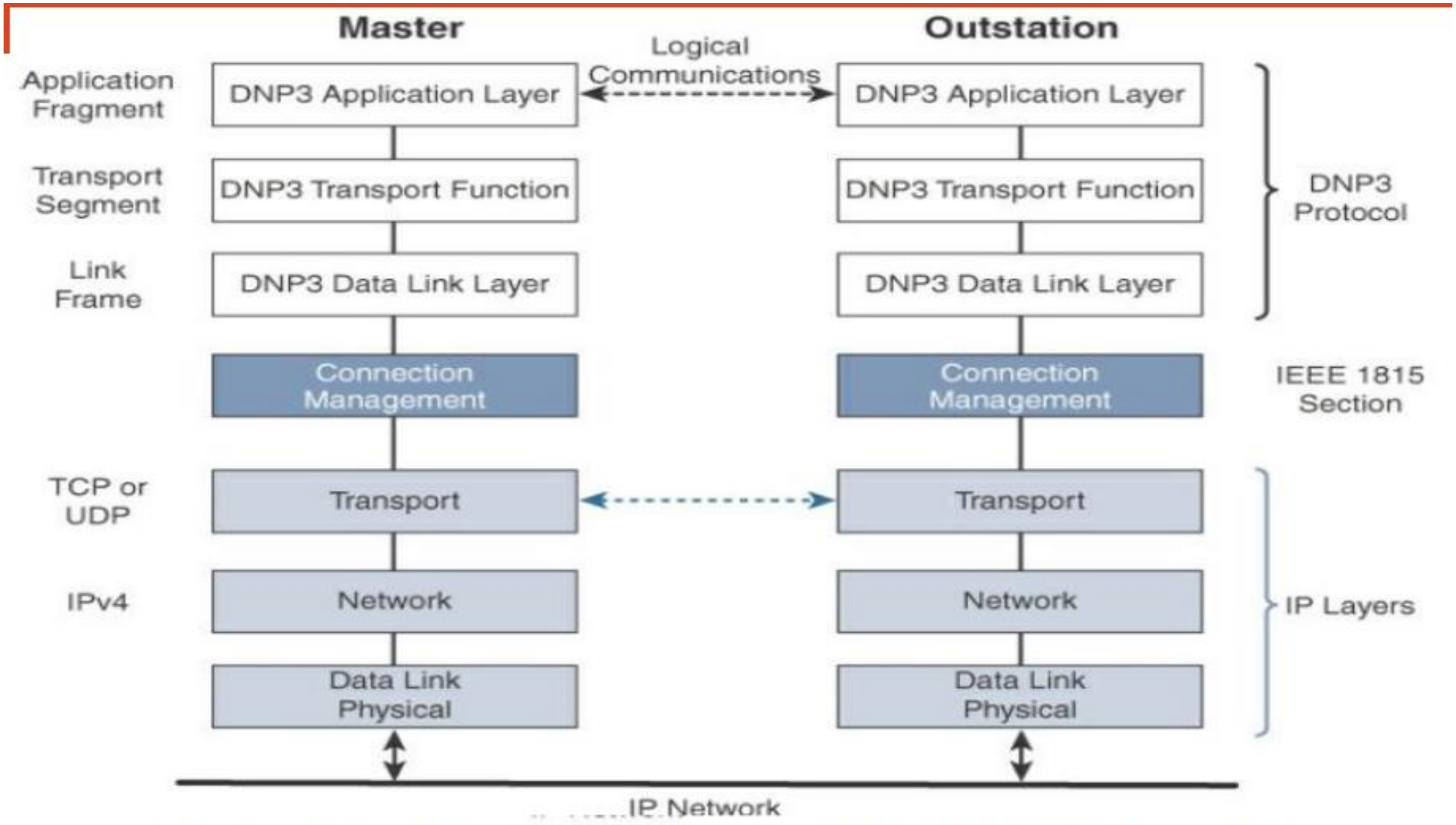
- DNP3 is based on a master/slave relationship.

- The term master refers to 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 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..

SCADA-supervisory control and data acquisition



Protocol Stack for Transporting Serial DNP3 SCADA over IP

SCADA-supervisory control and data acquisition

- The IEEE 1815-2012 specification describes how the DNP3 protocol implementation must be adapted to run either over TCP (recommended) or UDP .

- This specification defines connection management between the DNP3 protocol and the IP layers, as shown in Figure.

- Connection management links the DNP3 layers with the IP layers in addition to the configuration parameters and methods necessary for implementing the network connection.

SCADA-supervisory control and data acquisition

- The IP layers appear transparent to the DNP3 layers as each piece of the protocol stack in one station logically communicates with the respective part in the other.

- This means that the DNP3 endpoints or devices are not aware of the underlying IP transport that is occurring

SCADA-supervisory control and data acquisition

- In Figure, 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 endpoint is defined as a process that can both listen for connection requests and perform an active open on the channel if required.

SCADA-supervisory control and data acquisition

- Master stations may parse multiple DNP3 data link layer frames from a single UDP datagram, while DNP3 data link layer frames cannot span multiple UDP datagrams.
- Single or multiple connections to the master may get established while a TCP keepalive timer monitors the status of the connection.
- 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-supervisory control and data acquisition

- **Tunneling Legacy SCADA over IP Networks:**

- Deployments of legacy industrial protocols, such as DNP3 and other SCADA protocols, in modern IP networks call for flexibility when integrating several generations of devices or operations that are tied to various releases and versions of application servers.

- Transport of the original serial protocol over IP can be achieved either by tunneling using raw sockets over TCP or UDP or by installing an intermediate device that performs protocol translation between the serial protocol version and its IP implementation

SCADA-supervisory control and data acquisition

- A raw socket connection simply denotes that the serial data is being packaged directly into a TCP or UDP transport.

- A socket in this instance is a standard application programming interface (API) composed of an IP address and a TCP or UDP port that is used to access network devices over an IP network.

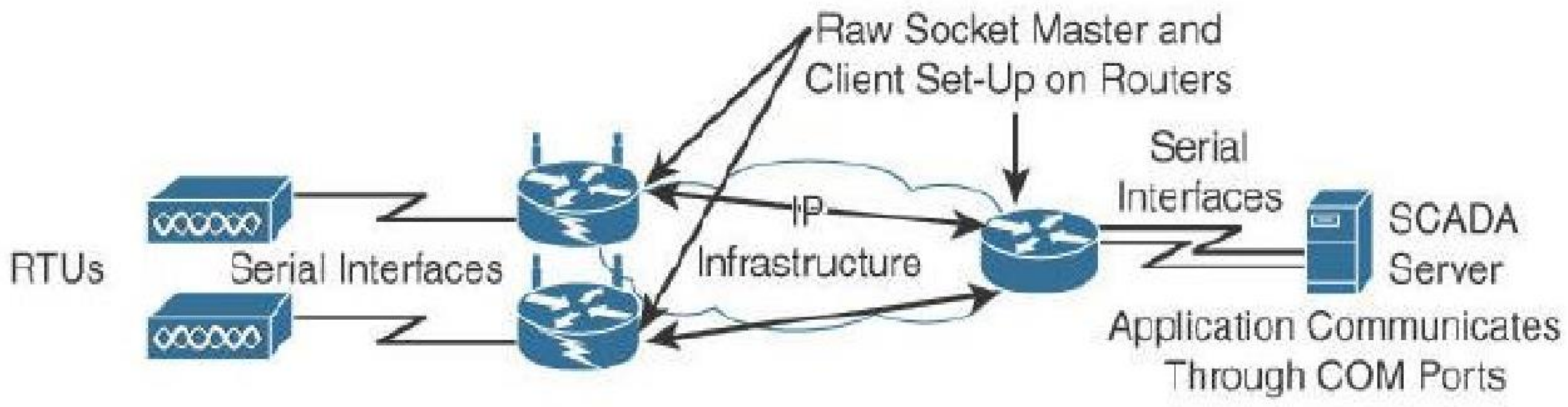
- Figure details raw socket scenarios for a legacy SCADA server trying to communicate with remote serial devices.

SCADA-supervisory control and data acquisition

- In all the scenarios in Figure, that routers connect via serial interfaces to the remote terminal units (RTUs), which are often associated with SCADA networks.
- An RTU is a multipurpose device used to monitor and control various systems, applications, and devices managing automation.
- From the master/slave perspective, the RTUs are the slaves. Opposite the RTUs in each Figure scenario is a SCADA server, or master, that varies its connection type.

SCADA-supervisory control and data acquisition

- Raw Socket TCP or UDP Scenarios for Legacy Industrial Serial Protocols


Raw Socket Master and Client Set-Up on Routers
Serial Interfaces
SCADA Server
RTUs
Serial Interfaces
IP Infrastructure
Application Communicates Through COM Ports

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

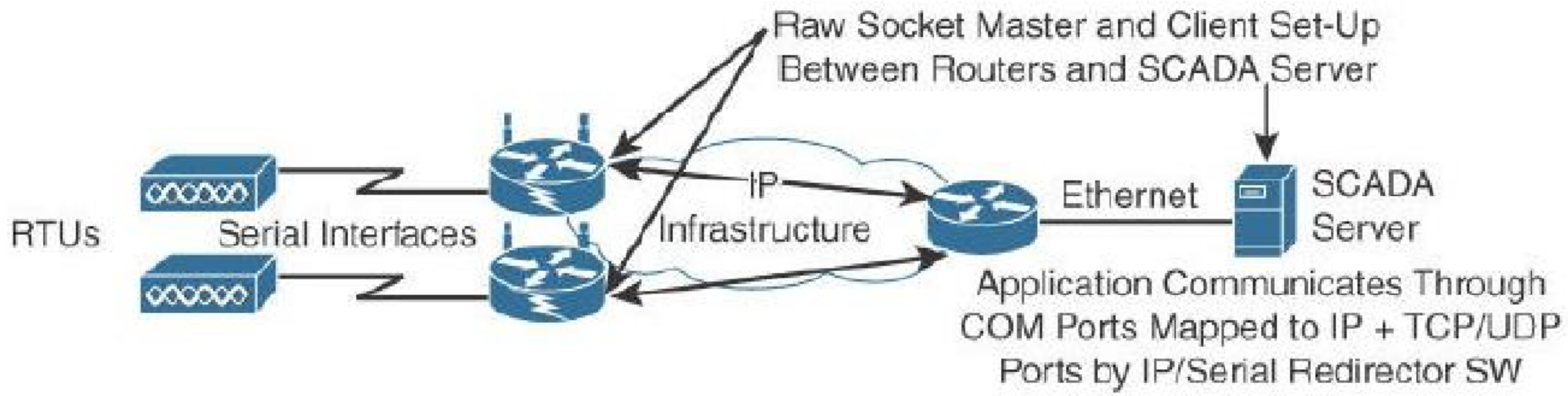
SCADA-supervisory control and data acquisition

- In Scenario A in Figure, both the SCADA server and the RTUs have a direct serial connection to their respective routers.

- The routers terminate the serial connections at both ends of the link and use raw socket encapsulation to transport the serial payload over the IP network.

SCADA-supervisory control and data acquisition

- Raw Socket TCP or UDP Scenarios for Legacy Industrial Serial Protocols



Raw Socket Master and Client Set-Up
Between Routers and SCADA Server
RTUs
Serial Interfaces
IP
Infrastructure
Ethernet
SCADA
Server
Application Communicates Through
COM Ports Mapped to IP + TCP/UDP
Ports by IP/Serial Redirector SW

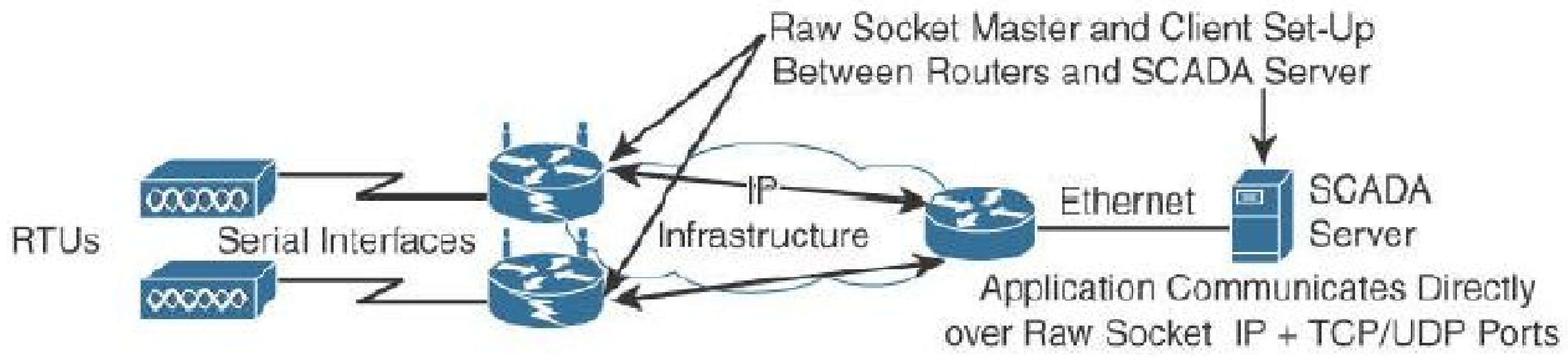
Scenario B: Raw Socket between Router and SCADA Server – no SCADA application change on server but IP/Serial Redirector software and Ethernet interface to be added

SCADA-supervisory control and data acquisition

- Scenario B has a small change on the SCADA server side.
- A piece of software is installed on the SCADA server that maps the serial COM ports to IP ports.
- This software is commonly referred to as an IP/serial redirector.
- The IP/serial redirector in essence terminates the serial connection of the SCADA server and converts it to a TCP/IP port using a raw socket connection.

SCADA-supervisory control and data acquisition

- Raw Socket TCP or UDP Scenarios for Legacy Industrial Serial Protocols


Raw Socket Master and Client Set-Up
Between Routers and SCADA Server
RTUs
Serial Interfaces
IP
Infrastructure
Ethernet
SCADA
Server
Application Communicates Directly
over Raw Socket IP + TCP/UDP Ports

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

SCADA-supervisory control and data acquisition

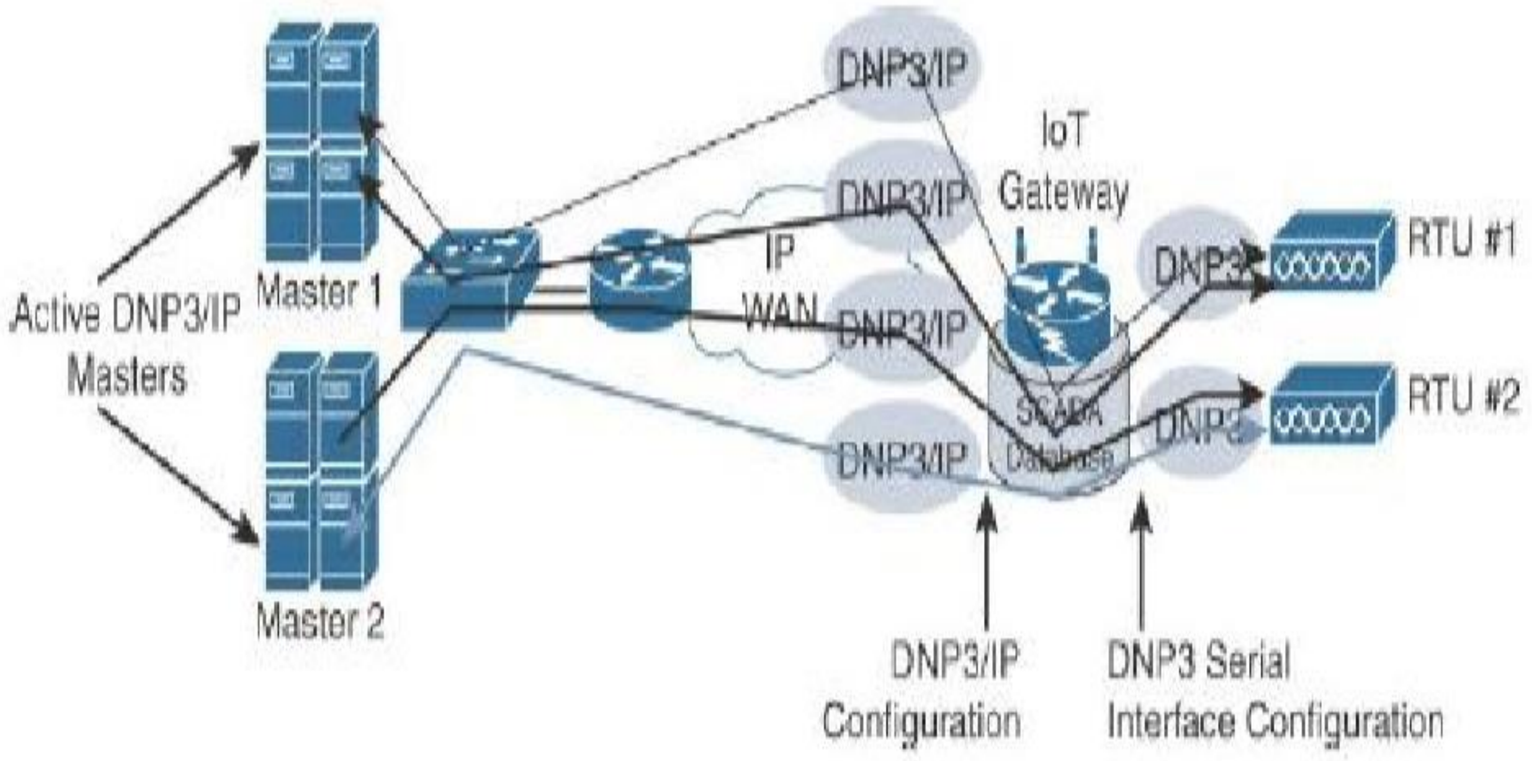
- In Scenario C in Figure, the SCADA server supports native raw socket capability.

- Unlike in Scenarios A and B, where a router or IP/serial redirector software has to map the SCADA server's serial ports to IP ports, in Scenario C the SCADA server has full IP support for raw socket connections.

SCADA-supervisory control and data acquisition

- **SCADA Protocol Translation:**
- An alternative to a raw socket connection for transporting legacy serial data across an IP network is **protocol translation.**
- With protocol translation, the legacy serial protocol is translated to a corresponding IP version.
- For example, Figure shows two serially connected DNP3 RTUs and two master applications supporting DNP3 over IP that control and pull data from the RTUs.

SCADA-supervisory control and data acquisition



DNP3 Protocol Translation

SCADA-supervisory control and data acquisition

- The IoT gateway in this figure performs a protocol translation function that enables communication between the RTUs and servers, despite the fact that a serial connection 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 Figure 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-supervisory control and data acquisition

- This can be accomplished by offering computing resources on IoT gateways or routers, as shown in this protocol translation example.

- Alternatively, this can also be performed directly on a node connecting multiple sensors.

- In either case, this is referred to as fog computing.

SCADA-supervisory control and data acquisition

- **SCADA Transport over LLNs with MAP-T:**
- Due to the constrained nature of LLNs, the implementation of industrial protocols should at a minimum be done over UDP .
- This in turn requires that both the application servers and devices support and implement UDP .
- While the long-term evolution of SCADA and other legacy industrial protocols is to natively support IPv6, but most of the industrial devices supporting IP today support IPv4 only.

SCADA-supervisory control and data acquisition

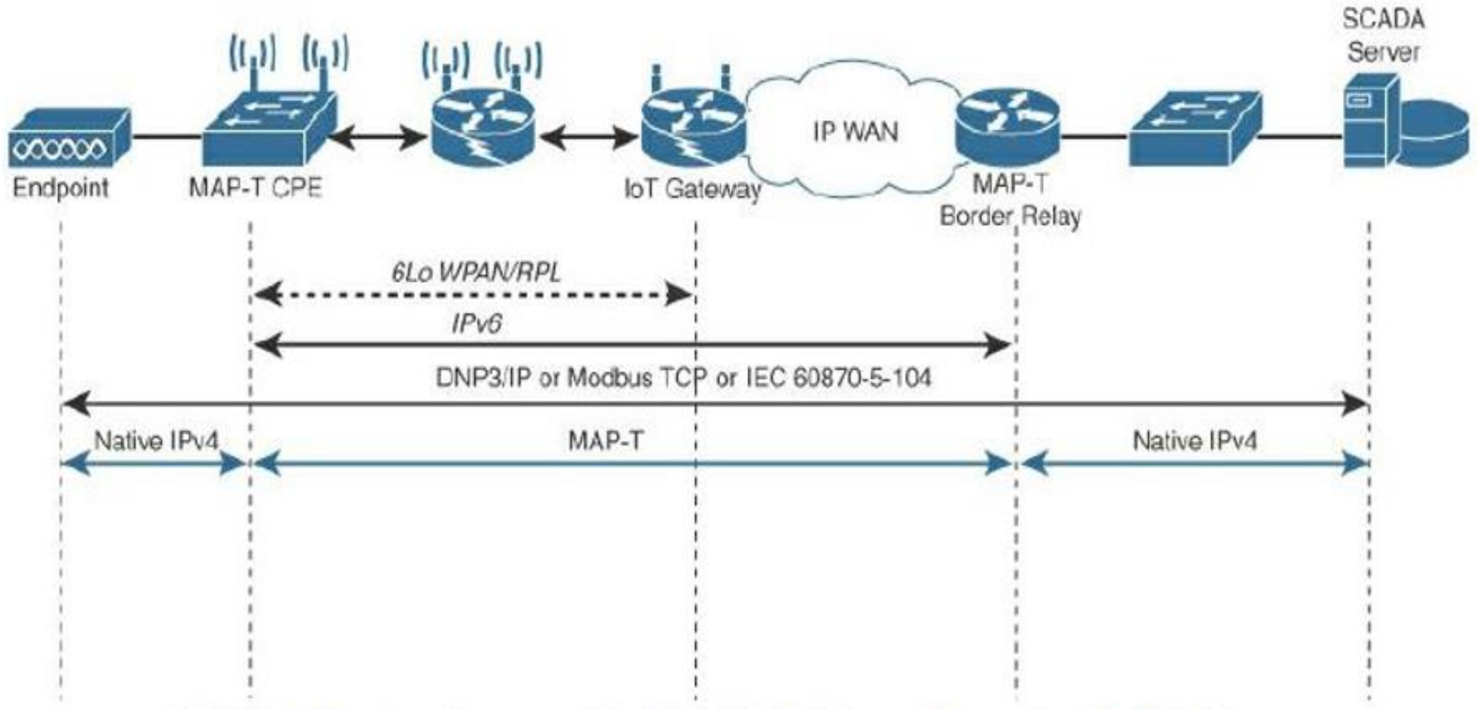
- When deployed over LLN subnetworks that are IPv6 only, a transition mechanism, such as MAP-T (Mapping of Address and Port using Translation, RFC 7599), needs to be implemented.

- This allows the deployment to take advantage of native IPv6 transport transparently to the application and devices.

SCADA-supervisory control and data acquisition

- 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 (typical in the industry today).
- However, IPv6 (with 6LoWPAN and RPL) is being used for connectivity to the endpoint

SCADA-supervisory control and data acquisition



DNP3 Protocol over 6LoWPAN Networks with MAP-T

SCADA-supervisory control and data acquisition

- 6LoWPAN is a standardized protocol designed for constrained networks, but it only supports IPv6.
- In this situation, the end devices, the endpoints, and the SCADA server support only IPv4, but the network in the middle supports only IPv6.
- The solution to this problem is to use the protocol known as MAP-T makes the appropriate mappings between IPv4 and the IPv6 protocols.
- This allows legacy IPv4 traffic to be forwarded across IPv6 networks.

SCADA-supervisory control and data acquisition

- The IPv4 endpoint on the left side is connected to a Customer Premise Equipment (CPE) device.
- The MAP-T CPE device 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.
- The MAP-T CPE device and MAP-T border gateway are responsible for the MAP-T conversion from IPv4 to IPv6.

Generic Web-Based Protocols

- Web-based protocols have become common in consumer and enterprise applications and services
- On non-constrained networks, such as Ethernet, Wi-Fi, or 3G/4G cellular, where bandwidth is not perceived as a potential issue, data payloads including XML or JavaScript Object Notation (JSON), can be transported over HTTP/HTTPS or WebSocket.
- This allows implementers to develop their IoT applications in contexts similar to web applications

Contd…

- 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 (in the range of tens of kilobytes in some cases).
- This enables the use of embedded web services software on some constrained devices
- When considering web services implementation on an IoT device, the choice between supporting the client or server side of the connection must be carefully made

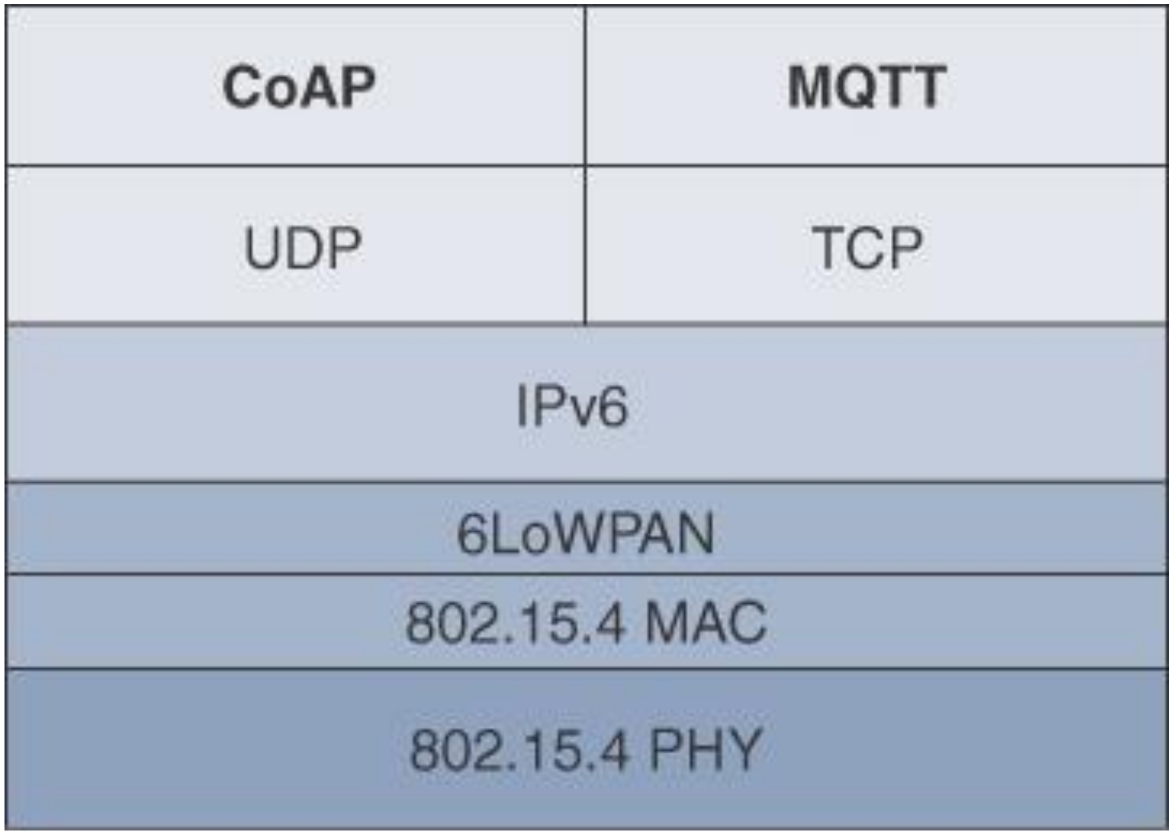
Contd…

- IoT devices that only push data to an application (for example, an Ethernet- or Wi-Fi-based weather station reporting data to a weather map application or a Wi-Fi–enabled body weight scale that sends data to a health application) may need to implement web services on the client side.
- The HTTP client side only initiates connections and does not accept incoming ones

IoT Application Layer Protocols

- When considering constrained networks and/or a large-scale deployment of constrained nodes, web-based and data model protocols may be too heavy for IoT applications.
- To address this problem, the IoT industry is working on new lightweight protocols that are better suited to large numbers of constrained nodes and networks.
- Two of the most popular protocols are CoAP and MQTT

Application Layer Protocols

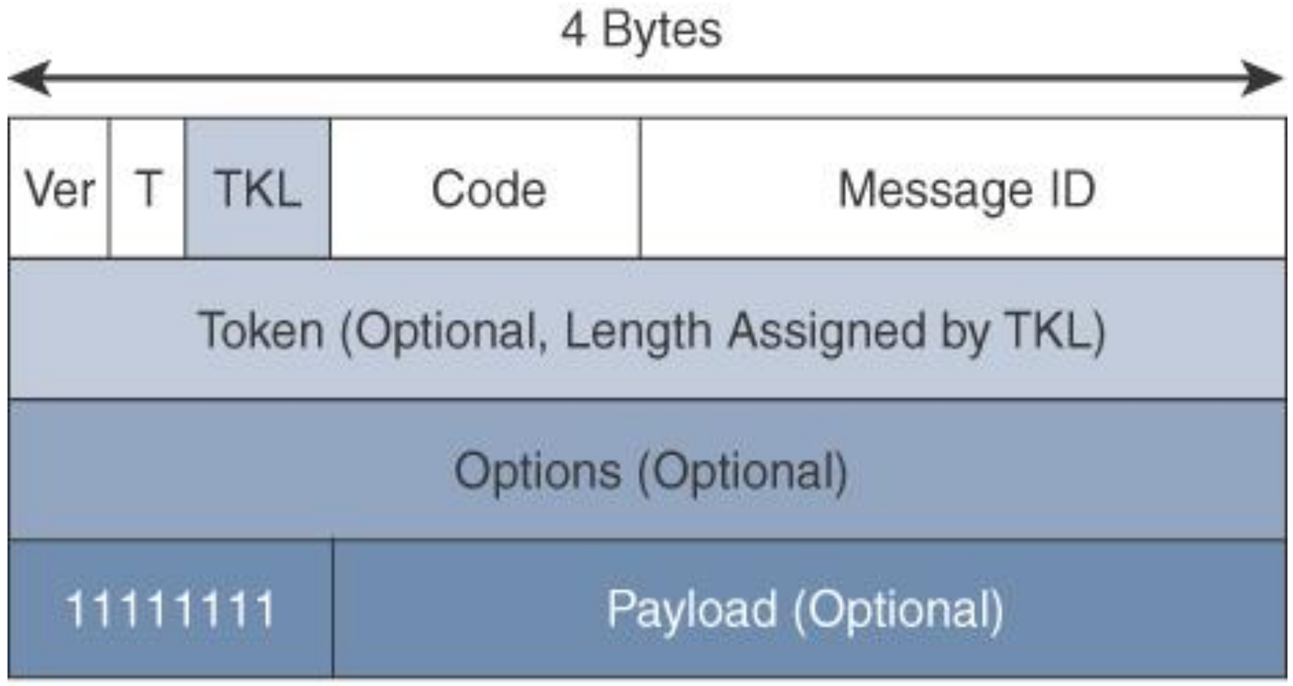


CoAP

- The 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).
- RFC 7252 provides more details on securing CoAP with DTLS.
- It specifies how a CoAP endpoint is provisioned with keys and a filtering list.
- Four security modes are defined: NoSec, PreSharedKey, RawPublicKey, and Certificate.
- The NoSec and RawPublicKey implementations are mandatory.

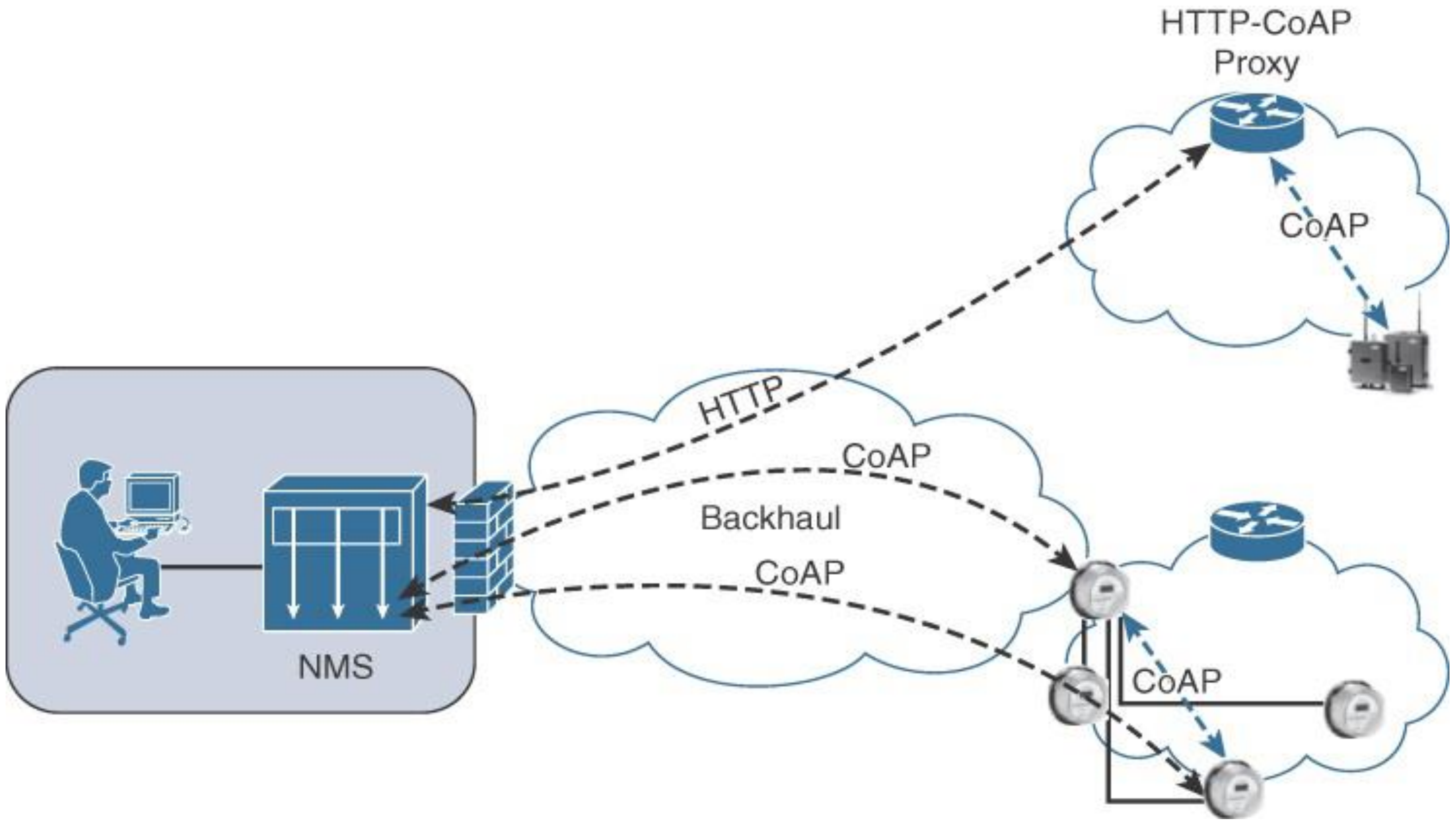
CoAP

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

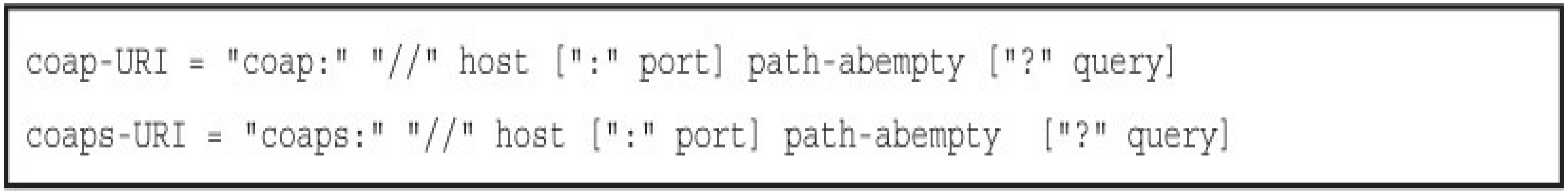


CoAP Message Field	Description
Ver (Version)	Identifies the CoAP version.
T (Type)	Defines one of the following four message types: Confirmable (CON), Non-confirmable (NON), Acknowledgement (ACK), or Reset (RST). CON and ACK are highlighted in more detail in Figure 6-9.
TKL (Token Length)	Specifies the 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. For example, in Figure 6-9, GET is the request method, and 2.05 is the response code. For a complete list of values for this field, refer to RFC 7252.
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. 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 Communications in IoT Infrastructures



CoAP URI format



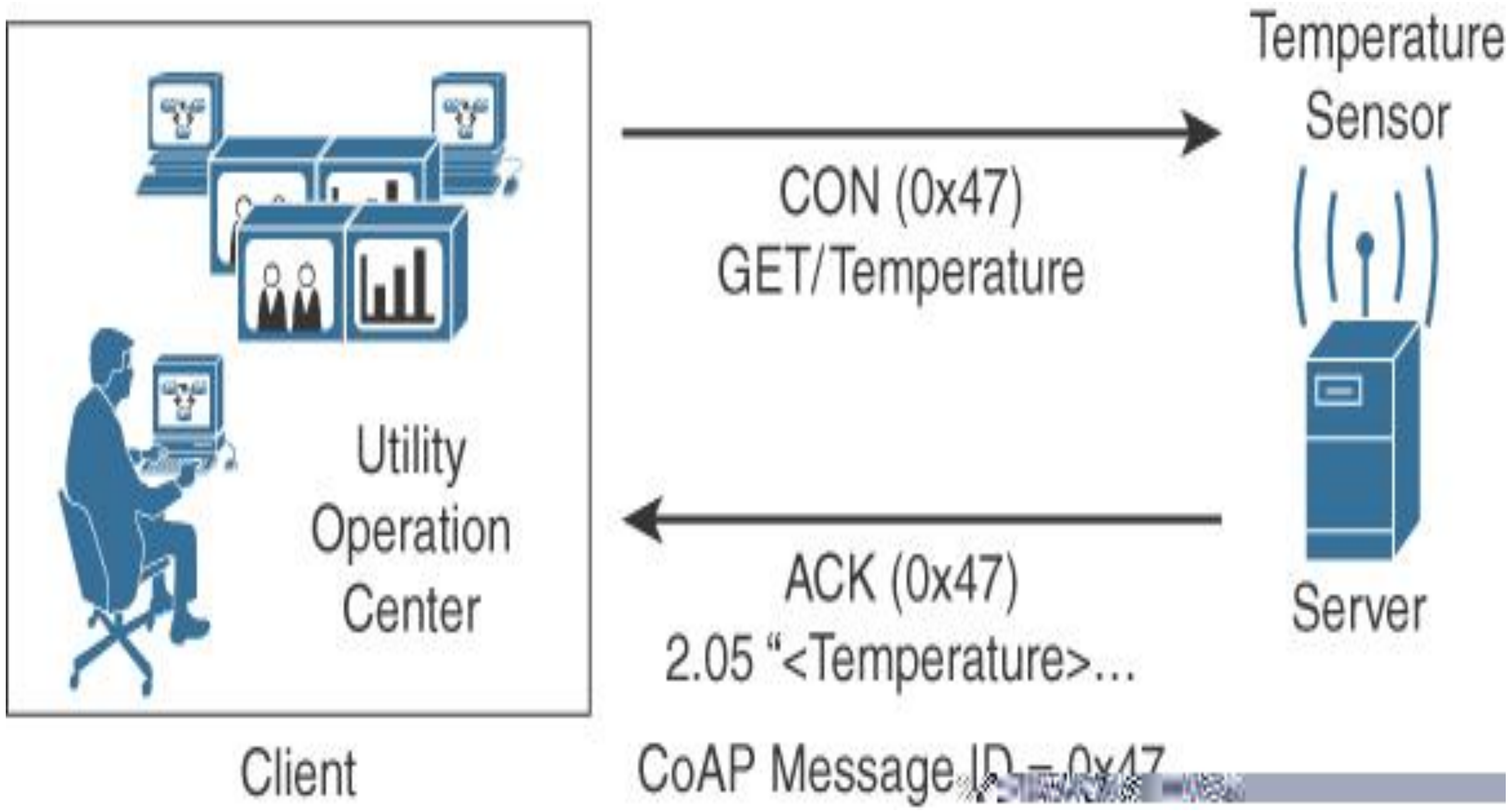
```
coap-URI = "coap:" "//" host [":" port] path-abempty ["?" query]

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

- CoAP defines four types of messages:
  - confirmable, non-confirmable, acknowledgement, and reset.
  - Method codes and response codes included in some of these messages make them carry requests or responses.

CoAP Reliable Transmission Example

- 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 nonconfirmable message, a reset message is sent


Temperature Sensor
CON (0x47)
GET/Temperature
Utility Operation Center
ACK (0x47)
2.05 "<Temperature>…
Server
Client
CoAP Message ID = 0x47

- A utility operations center on the left, acting as the CoAP client, with the 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 messages
- The client sends a GET message to get the temperature from the sensor.
- Notice that the 0x47 message ID is present for this GET message and that the message is also marked with CON.
- A CON, or confirmable, marking in a CoAP message means the message will be retransmitted until the recipient sends an acknowledgement (or ACK) with the same message ID

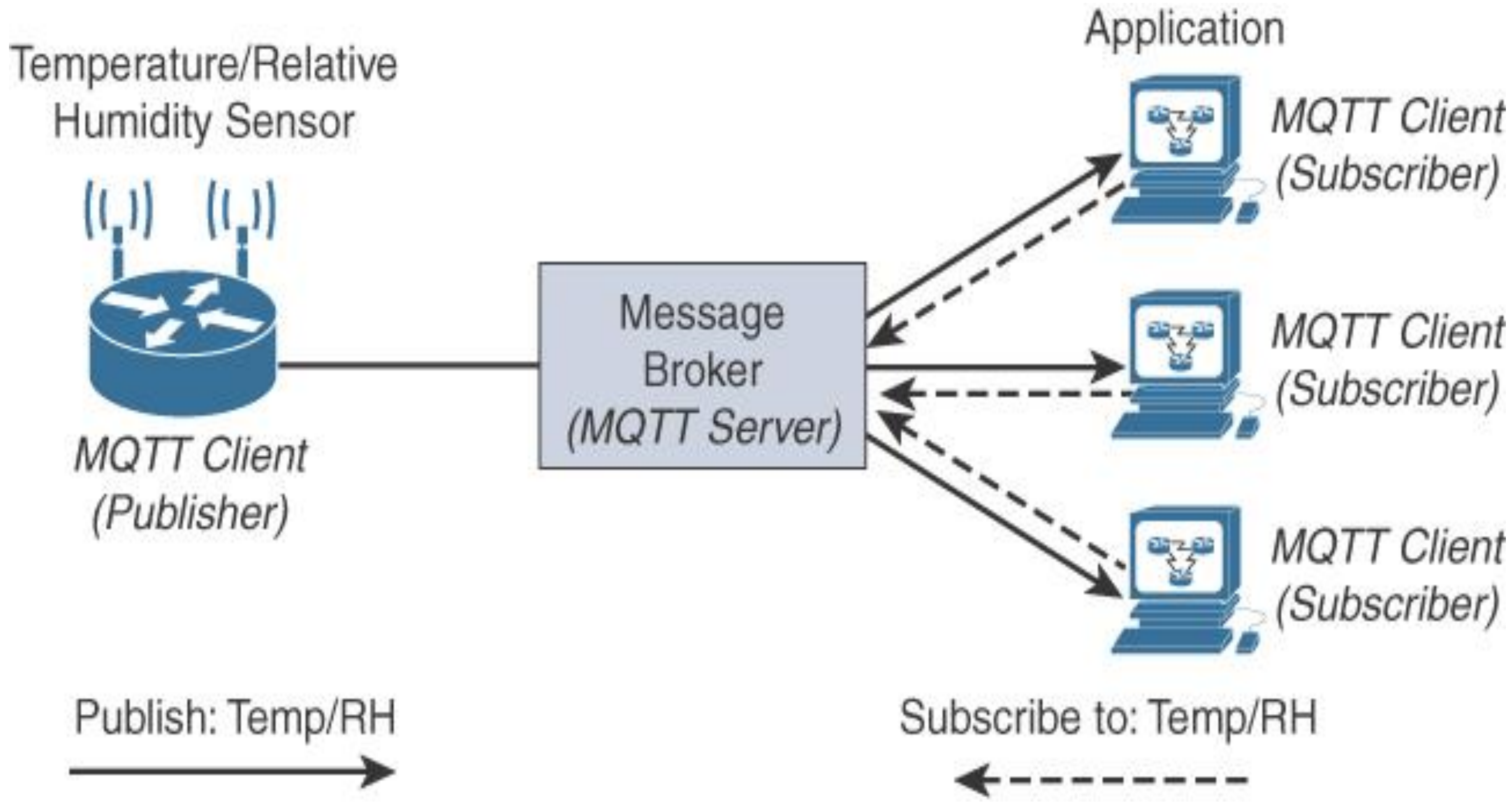
Message Queuing Telemetry Transport (MQTT)

- At the end of the 1990s, engineers from IBM and Arcom (acquired in 2006 by Eurotech) 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, as typically used by the oil and gas industries.
- 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

- Considering the harsh environments in the oil and gas industries, an extremely simple protocol with only a few options was designed, with considerations for constrained nodes, unreliable WAN backhaul communications, and bandwidth constraints with variable latencies.
- These were some of the rationales for the selection of a client/server and publish/subscribe framework based on the TCP/IP architecture

MQTT Publish/Subscribe Framework



Contd…

- An MQTT client can act as a publisher to send data (or resource information) to an MQTT server acting as an MQTT message broker
- The MQTT client 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 network connection along with application messages, such as Temp/RH data, from the publishers.
- It also handles the subscription and unsubscription process and pushes the application data to MQTT clients acting as subscribers

Contd…

- The application on the right side of Figure is an MQTT client that is a subscriber to the Temp/RH data being generated by the publisher or sensor on the left.
- This model, where subscribers express a desire to receive information from publishers.
- A great example is the collaboration and social networking application Twitter

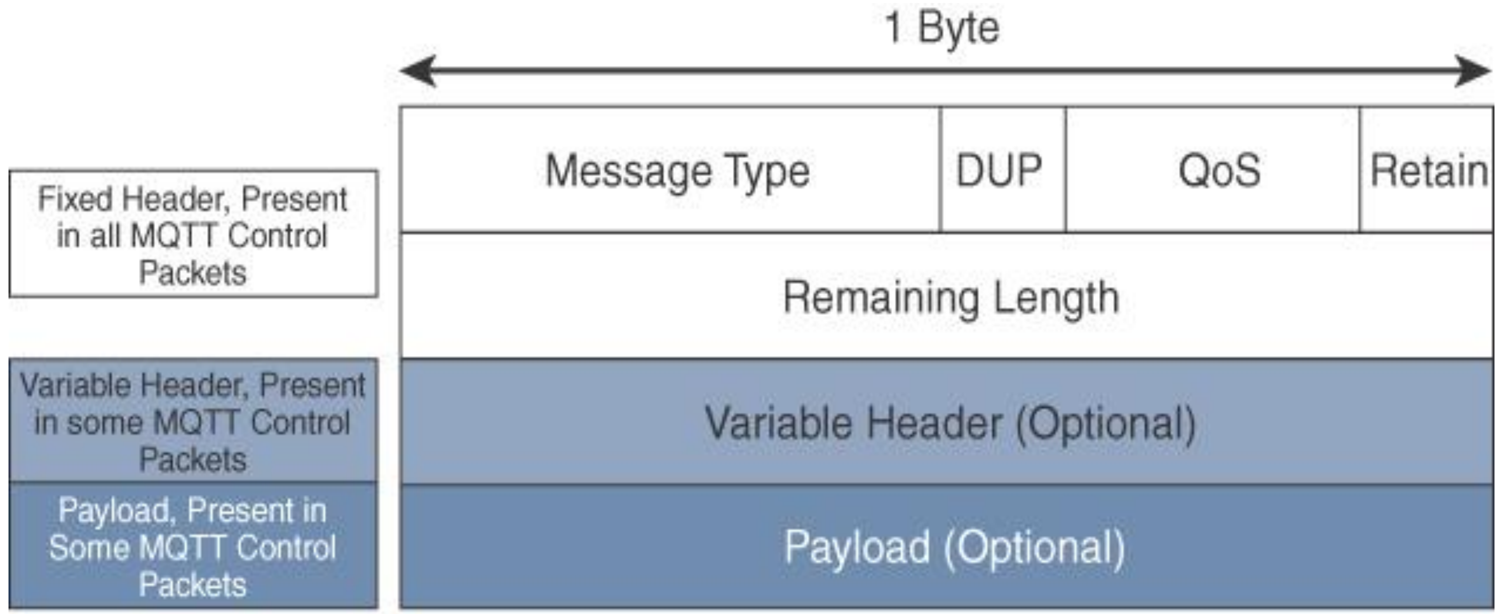
Contd…

- With MQTT, clients can subscribe to all data (using a wildcard character) or specific data from the information tree of a publisher.
- In addition, the presence of a message broker in MQTT decouples the data transmission between clients acting as publishers and subscribers.
- In fact, publishers and subscribers do not even know (or need to know) about each other.
- A benefit of having 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

Contd…

- MQTT control packets run over a TCP transport using port 1883.
- TCP ensures an ordered, lossless stream of bytes between the MQTT client and the MQTT server.
- Optionally, MQTT can be secured using TLS on port 8883
- MQTT is a lightweight protocol because each control packet consists of a 2-byte fixed header with optional variable header fields and optional payload
- Control packet can contain a payload up to 256MB

MQTT Message Format


1 Byte
Message Type
DUP
QoS
Retain
Remaining Length
Variable Header (Optional)
Payload (Optional)
Fixed Header, Present in all MQTT Control Packets
Variable Header, Present in some MQTT Control Packets
Payload, Present in Some MQTT Control Packets

- The first MQTT field in the header is Message Type, which identifies the kind of MQTT packet within a message.
- Fourteen different types of control packets are specified in MQTT version 3.1.1.
- Each of them has a unique value that is coded into the Message Type field

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
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

- The next field in the MQTT header is DUP (Duplication Flag). This flag, when set, allows the client to notate that the packet has been sent previously, but an acknowledgement was not received
- The QoS header field allows for the selection of three different QoS levels
- The next field is the Retain flag. Only found in a PUBLISH message, the Retain flag 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

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

- Subscriptions to resources generate SUBSCRIBE/SUBACK control packets, while unsubscription is performed through the exchange of UNSUBSCRIBE/UNSUBACK control packets
- PINGREQ/PINGRESP control packets are used to validate the connections between the client and server. Similar to ICMP pings that are part of IP, they are a sort of keepalive that helps to maintain and check the TCP session

- Securing MQTT connections through TLS is considered optional because it calls for more resources on constrained nodes.
- When TLS is not used, the client sends a clear-text username and password during the connection initiation.
- MQTT server implementations may also accept anonymous client connections (with the username/password being "blank").
- When TLS is implemented, a client must validate the server certificate for proper authentication.
- Client authentication can also be performed through certificate exchanges with the server, depending on the server configuration.

Three levels of MQTT QoS

- **QoS 0**
- This is a best-effort and 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.
- The message arrives at the receiver either once or not at all.

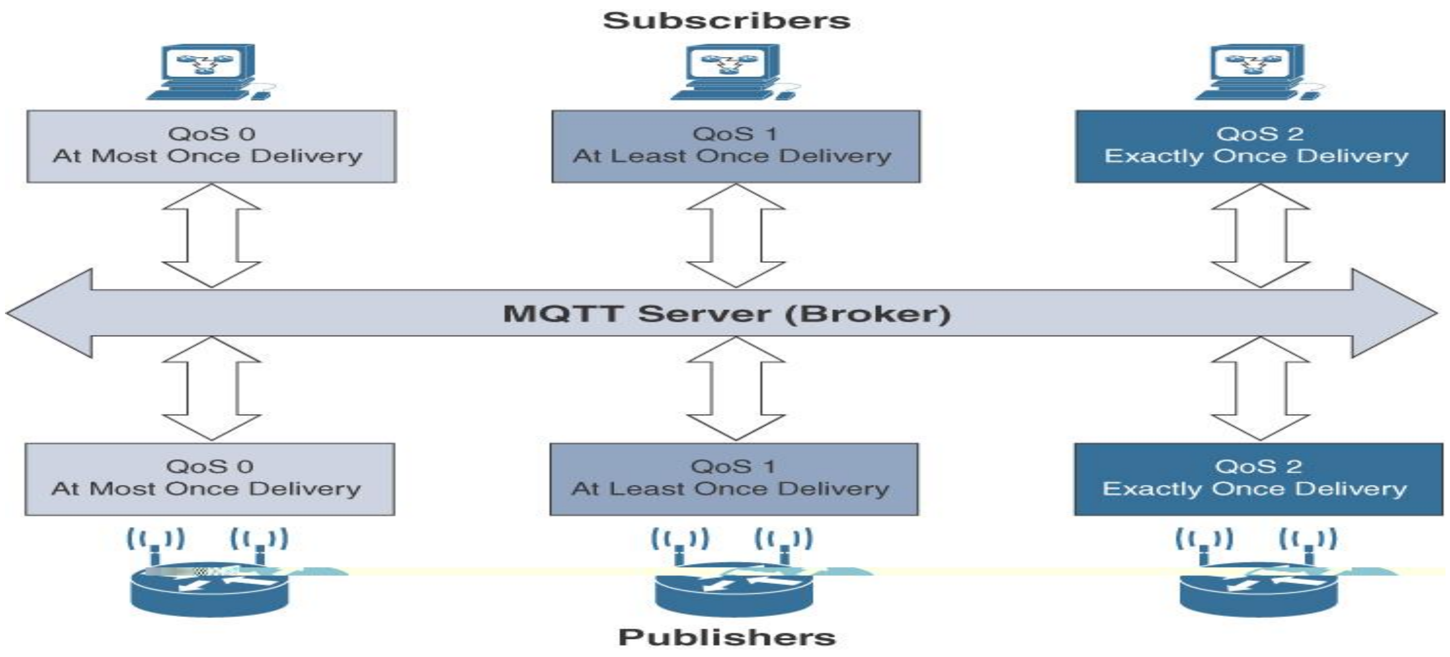
Contd…

- **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.
- This level guarantees "at least once" delivery

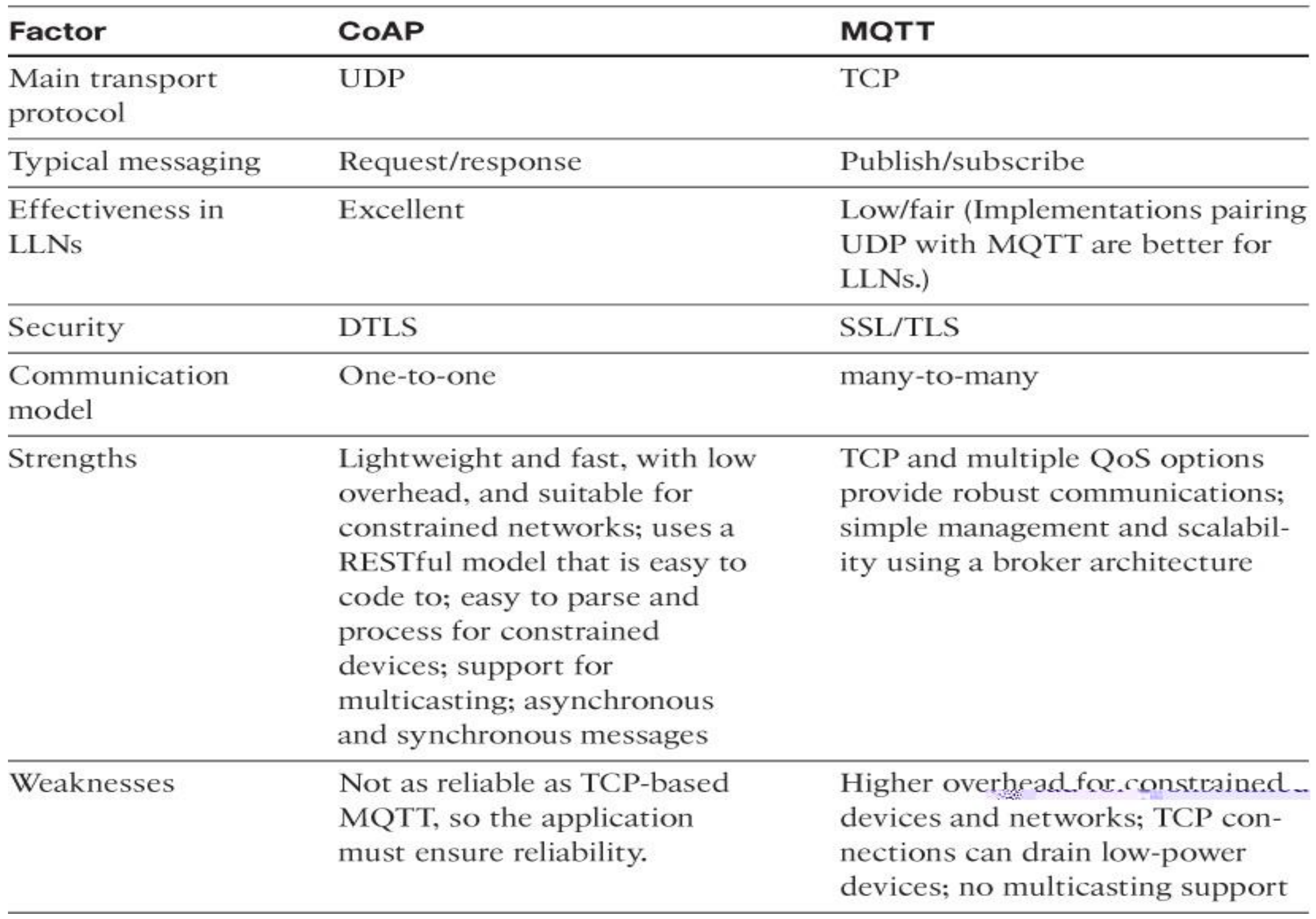
Contd…

- **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 QoS level because each packet contains an optional variable header with a packet identifier.
- Confirming the receipt of a PUBLISH message requires a two-step acknowledgement process.
- The first step is done through the PUBLISH/PUBREC packet pair, and the second is achieved with the PUBREL/PUBCOMP packet pair.
- 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

- The QoS process is symmetric in regard to the roles of sender and receiver, but 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



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; uses a RESTful model that is easy to code to; easy to parse and process for constrained devices; support for multicasting; asynchronous and synchronous messages	TCP and multiple QoS options provide robust communications; simple management and scalability using a broker architecture
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