Connecting Smart Objects
N/W Layer and Application Layer

- **There lot** of challenges in building the IoT solutions on IP.
- Along with the integration of non-IP devices -need to deal with the limits of the device and network levels of IoT.
- Optimizations are needed at various layers of the IP stack to handle the restrictions that are present in IoT networks. The Need for Optimization

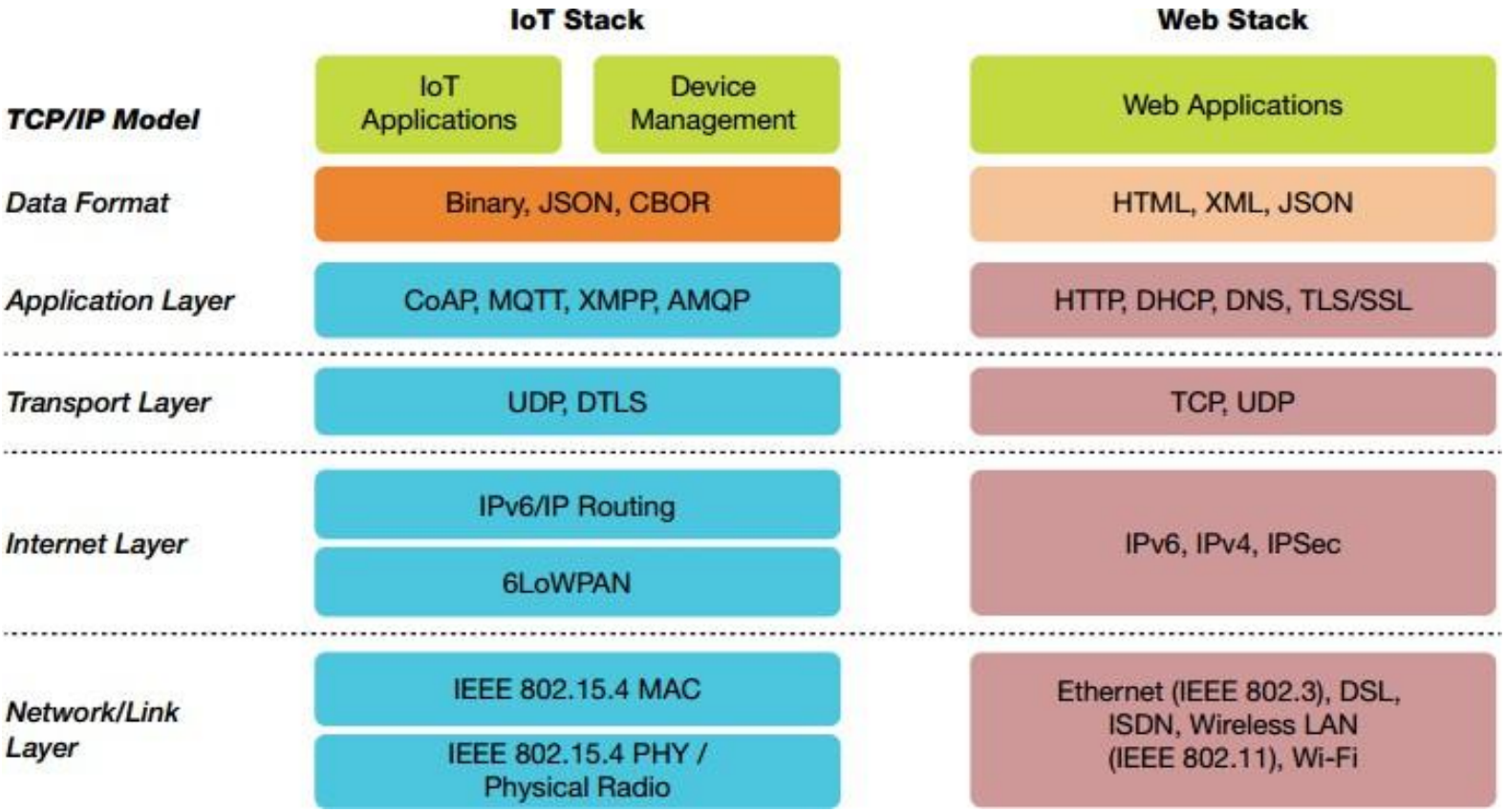
Why optimization is necessary in IP based IoT solutions?

- Constrained Nodes IoT is a platform, where 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.
- Many IoT devices are battery powered, with lifetime battery requirements varying from a few months to 10+ years.

Outline

- IPv6
- 6LoWPAN
- RPL
- SCADA,TCP,UDP
- CoAP
- MQTT
- XMPP

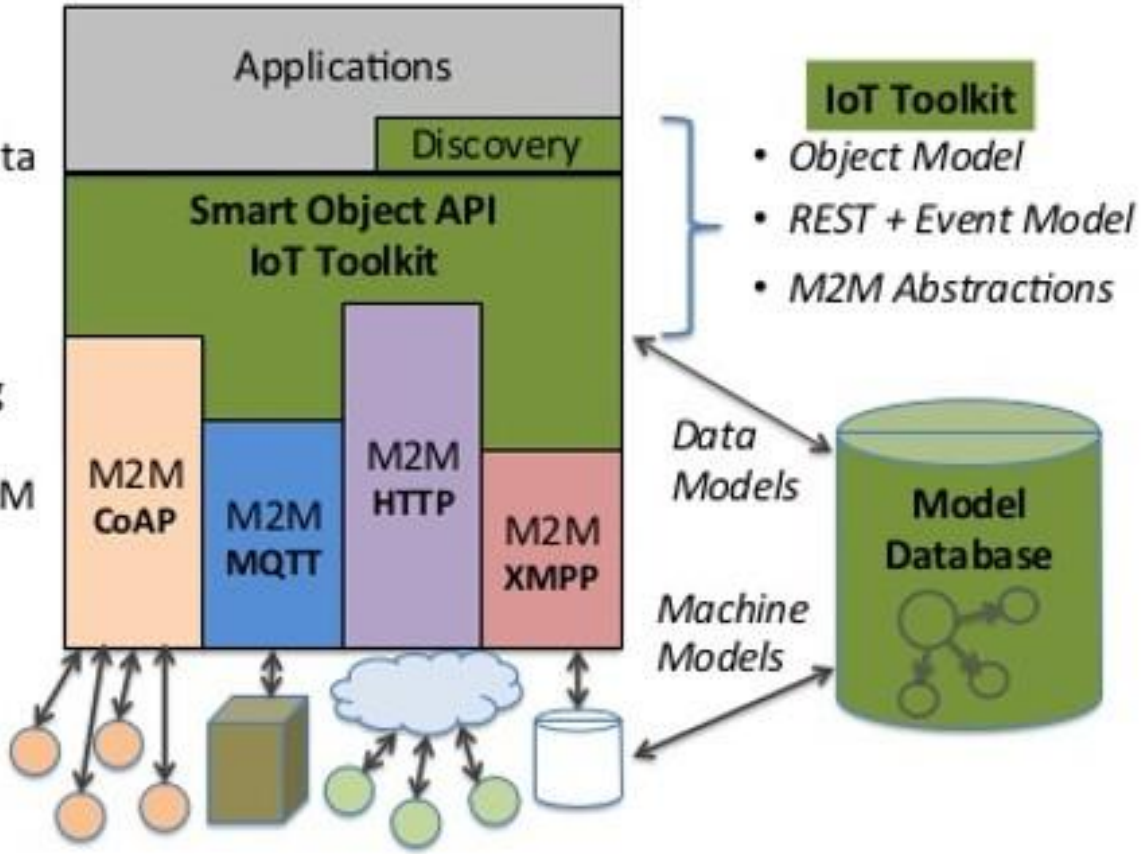
IoT vs. Internet Protocol Stack



Source
https://www.linkedin.com/pulse/emerging-open-standard-protocol-stack-iot-aniruddha-chakrabarti

IoT 2.0 Interoperability

- Easy to deploy new things and applications using data models
- Write once, run anywhere software
- **Any app** to **any thing** via **any M2M**, use-case appropriate M2M
- Network effect enabled

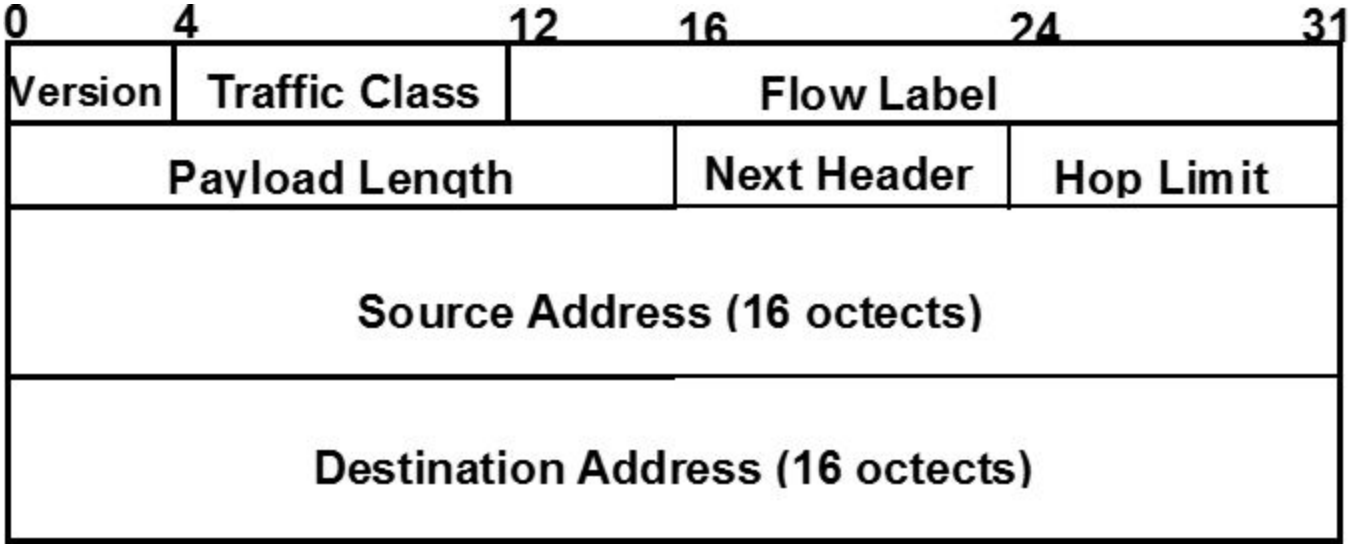


Source:
https://www.slideshare.net/michaeljohnkoster/opensourcestackforiot-131017193902phpapp02

IPv6

- Problems with IPv4
  - Shortage of address space
  - Lack of Quality of Service guarantee
- New features of IPv6
  - Enlarge address space
  - Fixed header format helps speed processing/forwarding
  - Better support for Quality of Service
  - Neighbor discovery and Auto-configuration
  - Hierarchical address architecture (improved address aggregation)
  - new “anycast” address: route to “best” of several replicated servers

IPv6 Header

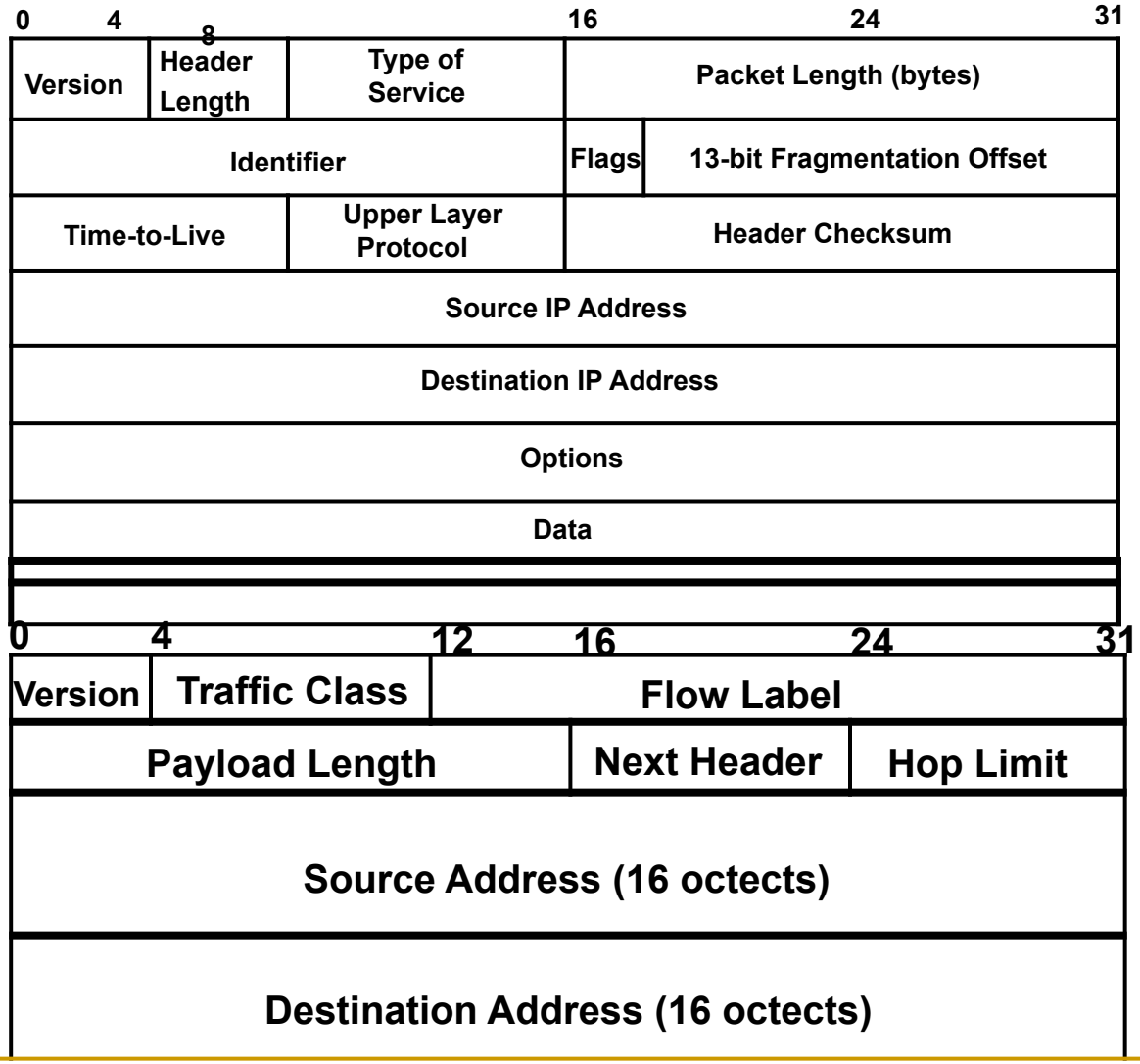

0
4
12
16
24
31
Version
Traffic Class
Flow Label
Payload Length
Next Header
Hop Limit
Source Address (16 octects)
Destination Address (16 octects)

IPv6 Header

- Version: 6
- Traffic class:
  - identify class of service
  - E.g., DiffServ (DS codepoint)
    - The 6 most-significant bits are used for DSCP
- Flow Label:
  - identify datagrams in same “flow”
- Next header:
  - identify upper layer protocol for data

)

Changes from IPv4 (1/3)



Changes from IPv4 (2/3)

- Expanded Addressing Capabilities
  - from 32 bits to 128 bits (more level and nodes)
  - improve multicast routing (“scope” field)
  - “anycast address”: send a packet to any one of a group of nodes
- Header Format Simplification
  - reduce bandwidth cost
- Extensions
  - more flexibility
- Checksum
  - removed to reduce processing at routers
- Fragmentation
  - Not allowed at intermediate routers

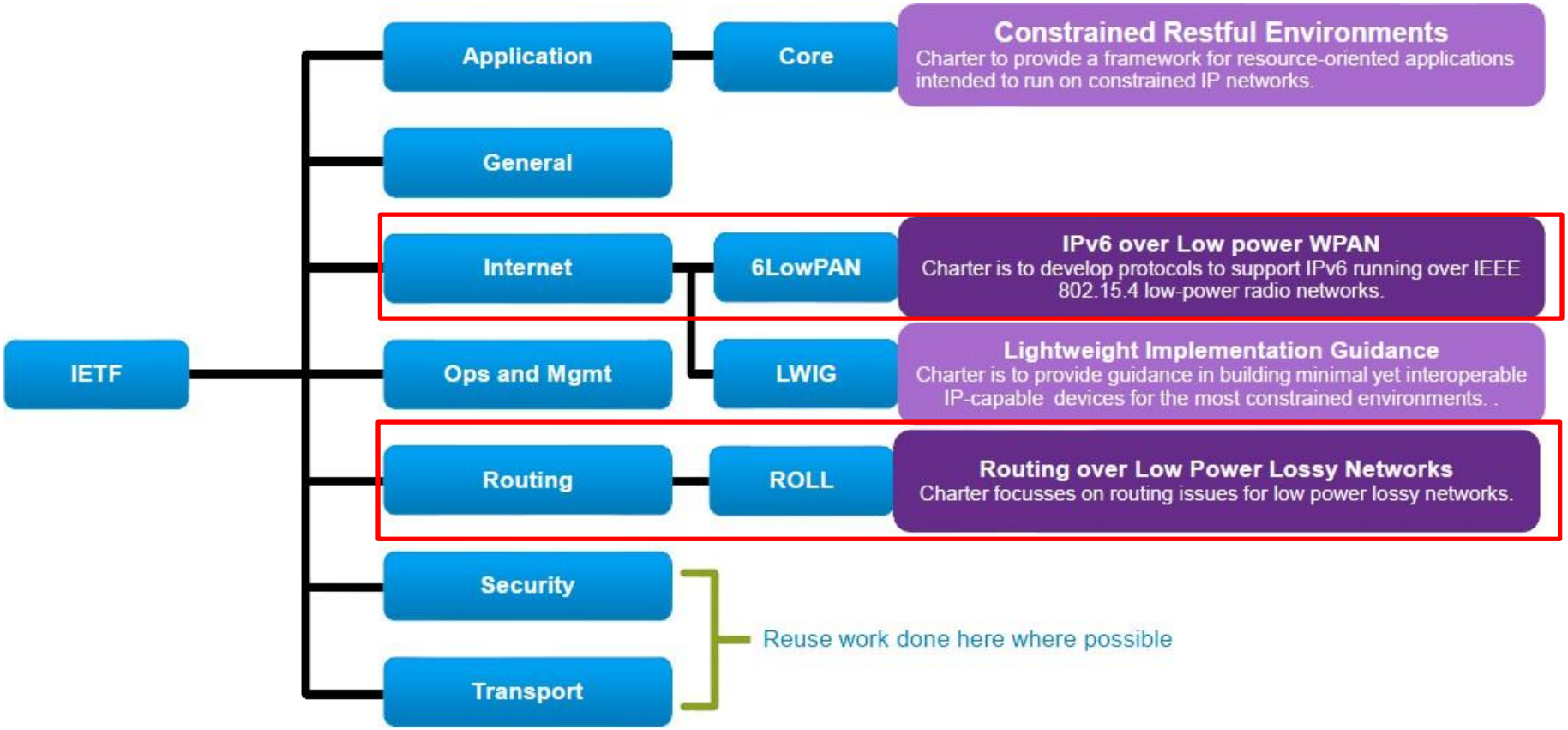
)

What is 6LoWPAN?

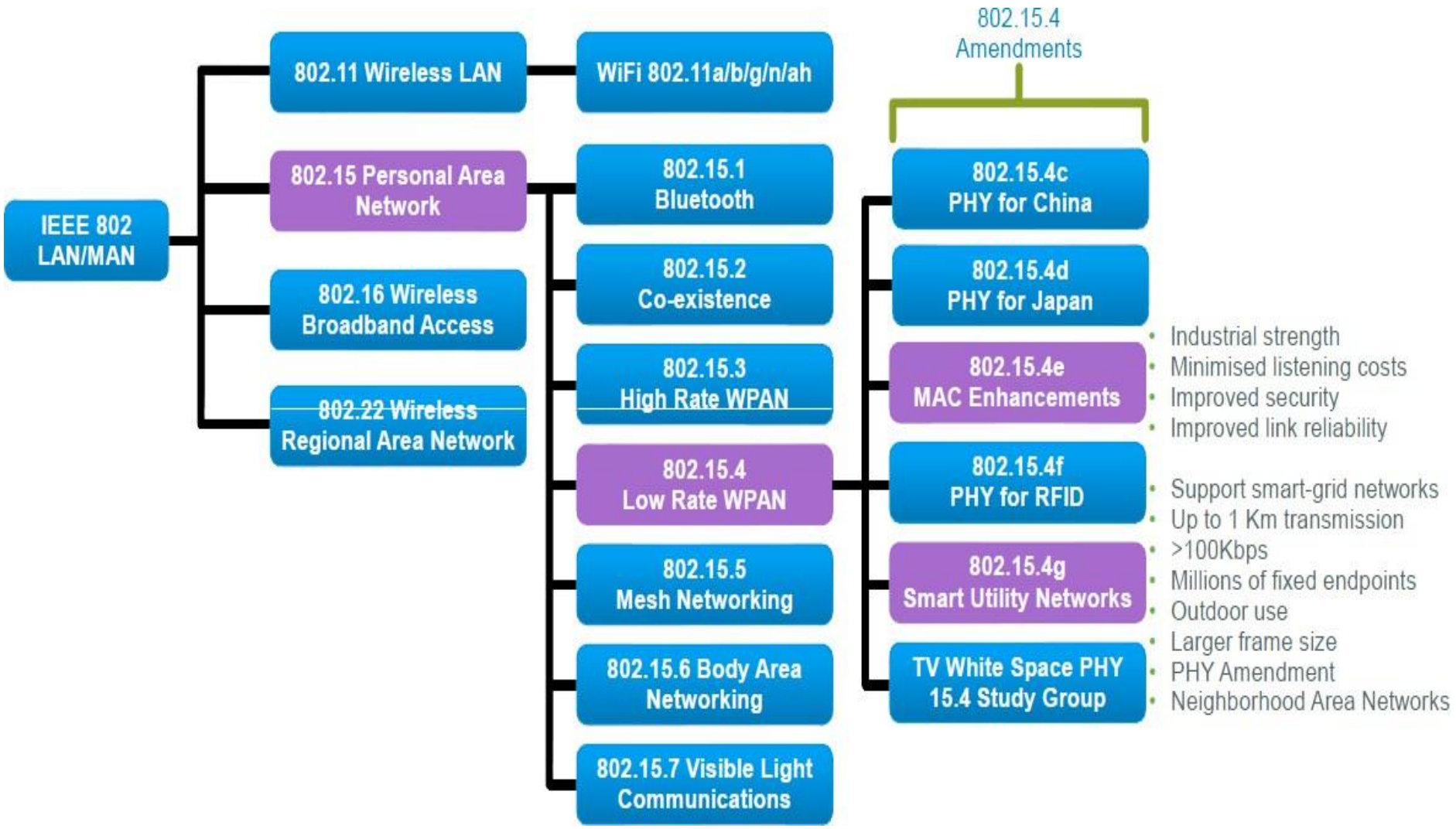
- 6LoWPAN is an acronym of IPv6 over Low power Wireless Personal Area Networks.
- It is designed by the 6LoWPAN working group in IETF (Internet Engineering Task Force).
- RFC 4919 (6LoWPAN Overview, Assumptions, Problem Statement, and Goals) included a detailed review of requirements, which were released in 2007.

IETF Low Power Lossy Network

Related Working Groups



IEEE Wireless Standards



6LoWPAN WG Documents

draft-ietf-6lowpan-btle-11	Transmission of IPv6 Packets over BLUETOOTH Low Energy	2012-10-12
RFC 4919	IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals	2007-08
RFC 4944*	Transmission of IPv6 Packets over IEEE 802.15.4 Networks	2007-09
RFC 6282	Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks	2011-09
RFC 6568	Design and Application Spaces for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)	2012-04
RFC 6606	Problem Statement and Requirements for IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Routing	2012-05
RFC 6775	Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)	2012-11 new

*RFC 4944 (Proposed Standard) Updated by RFC 6282, RFC 6775

Motivation

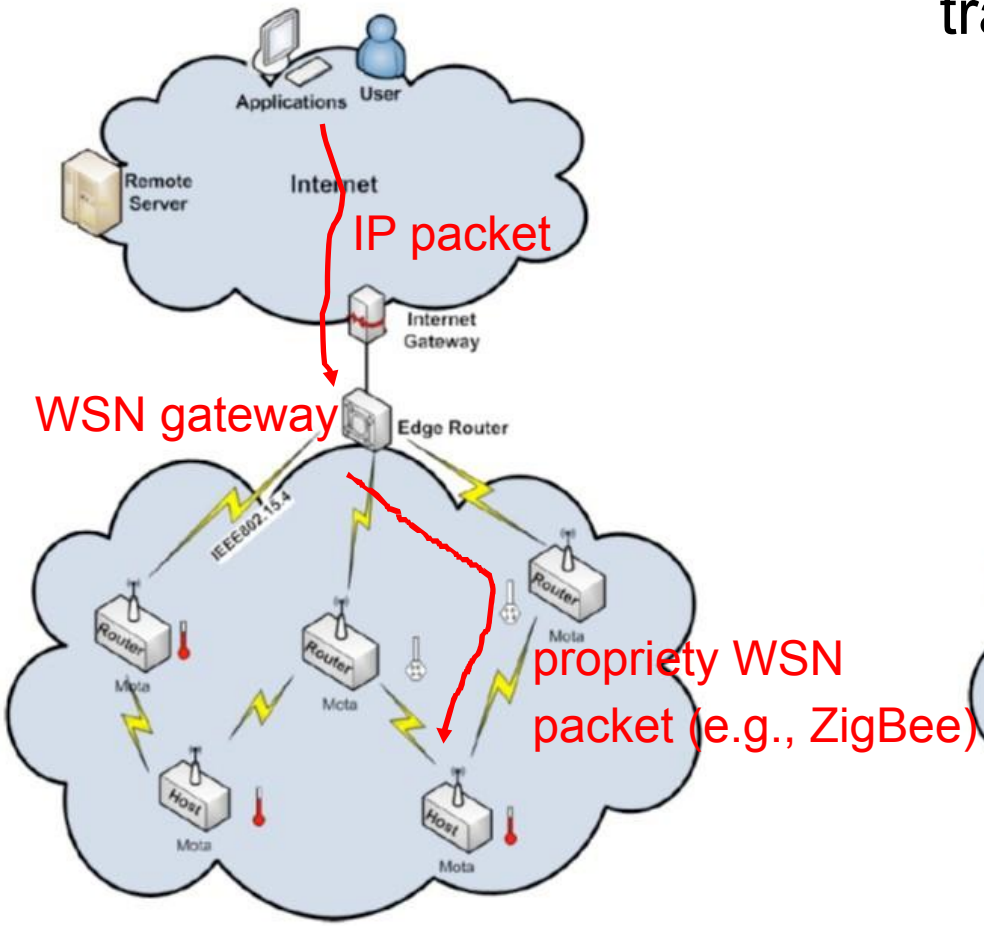
- Traditionally, battery-powered networks or low-bitrate networks, such as most fieldbus networks or 802.15.4 were considered **incapable of running IP**.
- In the home and industrial automation networks world, the situation compares to the situation of corporate LANs in the **1980**s:
  "should I run Token-Ring, ATM or IPX/SPX?"
  translates to "should I run ZigBee, LON or KNX?"

Motivation

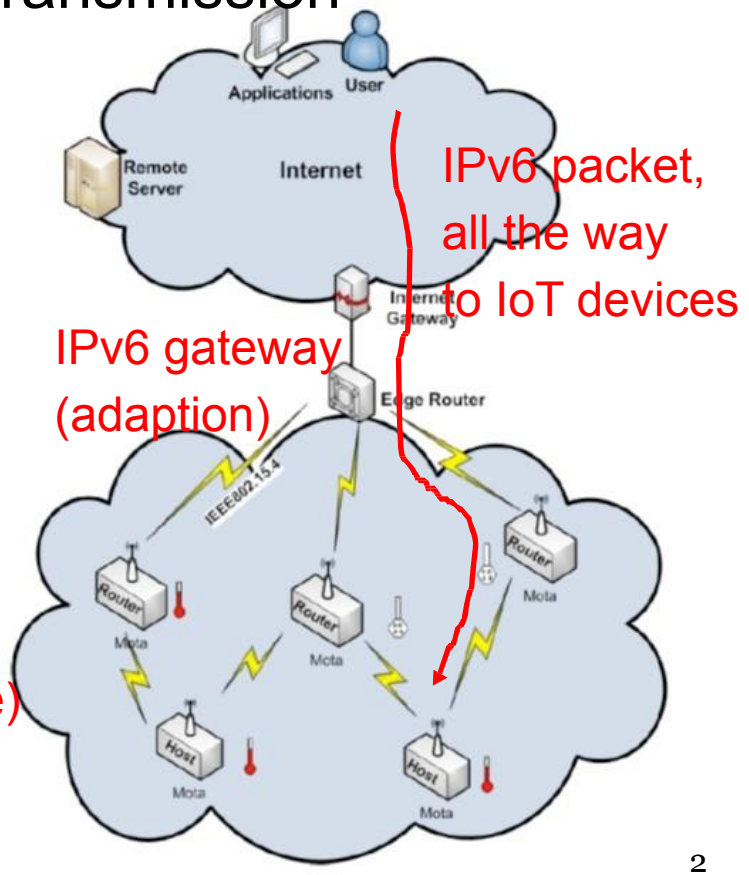
- Almost any layer 2 technology can be used and will simply extend the IP internetwork.
- The same transition to IP is now happening in the home and industrial automation worlds. 6LoWPAN and RPL have made this possible.

Goal of 6LowPAN

- Traditional way: 2-stage



- End-to-end IP transmission



Constraints of LoWPAN

- Low-cost nodes communicating over multiple hops to cover a large geographical area
- Operate unattended for years on modest batteries.
- Capabilities are more limited
  - small frame sizes, low bandwidth, and low transmit power, limited memory and compute power.
- Proprietary protocols and link-only solutions, presuming that IP was too memory and bandwidth-intensive

Key Factors for IP over 802.15.4

- Header
  - Standard IPv6 header is 40 bytes [RFC 2460]
  - Entire 802.15.4 MTU is 127 bytes [IEEE 802.15.4]
  - Often data payload is small
- Fragmentation
  - Interoperability means that applications need not know the constraints of physical links
  - IP packets may be large, compared to 802.15.4 max frame size
  - IPv6 requires all links support 1280 byte packets [RFC 2460]

Key Factors for IP over 802.15.4

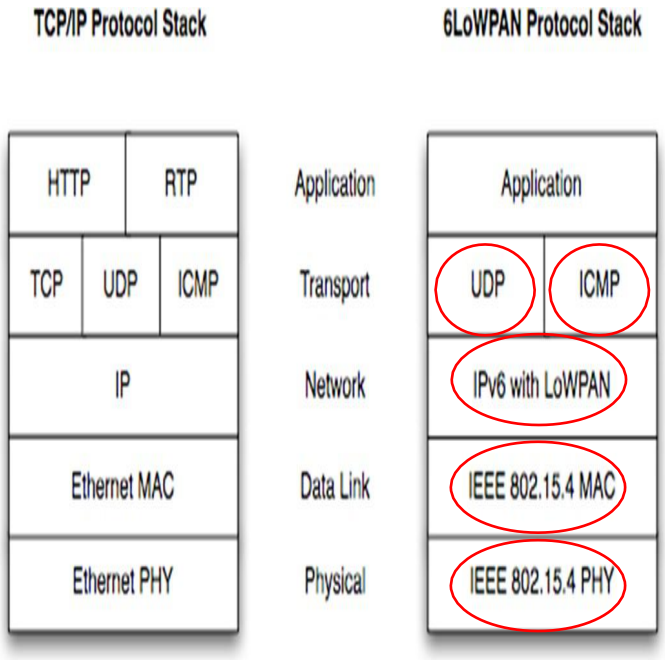
- Allow link-layer mesh routing under IP topology
  - 802.15.4 subnets may utilize multiple radio hops per IP hop
  - Similar to LAN switching within IP routing domain in Ethernet
- Allow IP routing over a mesh of 802.15.4 nodes
  - Options and capabilities already well-defined
  - Various protocols to establish routing tables

Topology

6LoWPAN network can be organized around three topologies:

- Star topology- All sensor nodes can reach and are reachable from the LBR(LoWPAN Border Router)
- Meshed-Nodes are organized at Layer 2 in order to relay frames toward the destination. From point of view of the Internet, a meshed network is similar to an Ethernet network where every node shares the same prefix.6LoWPAN refers to that technology as **mesh- under (MU)**.
- Routed-Nodes act as routers and forward packets toward the destination. Nodes acting as a router inside the LoWPAN network and not directly connected to the Internet are called LoWPAN routers(LRs). 6LoWPAN refers to that technology as **route- over (RO)**.- RPL protocol.

6LoWPAN Protocol Stack



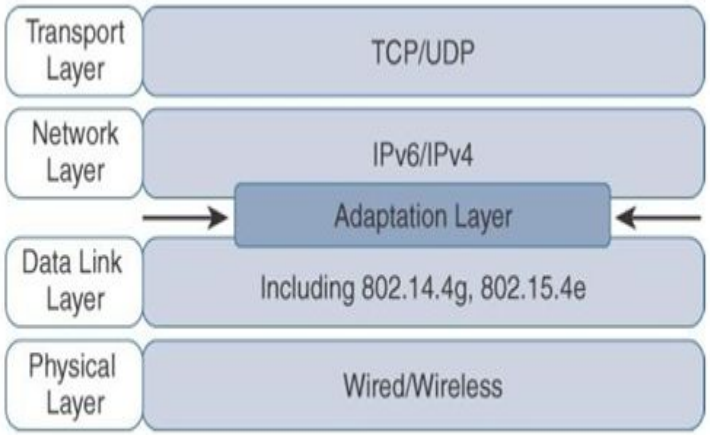


Figure 2.12: Optimizing IP for IoT Using an Adaptation Layer

6LoWPAN Key Elements

- 6LoWPAN introduces an adaptation layer between the IP stack's link and network layers to enable efficient transmission of IPv6 datagrams over 802.15.4 links
  - Provides header compression to reduce transmission overhead
  - Fragmentation to support the IPv6 minimum MTU requirement
  - Support for layer-two forwarding to deliver and

IPv6 datagram over multiple radio hops

- 6LoWPAN turns IEEE 802.15.4 into the next IP-enabled link
- Provides open-systems based interoperability among low-power devices over IEEE 802.15.4
- Provides interoperability between low-power devices and existing IP devices, using standard routing techniques
- Paves the way for further standardization of communication functions among low-power IEEE 802.15.4 devices

Key Concept

- Use of stateless or shared-context compression to elide adaptation, network, and transport layer header fields
  - Compressing all three layers down to a few bytes.
- It's possible to compress header fields to a few bits when we observe that they often carry common values, reserving an escape value for when less-common ones appear.

- 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.
- IoT devices with application protocols defined by the IETF, such as HTTP/HTTPS, CoAP, MQTT, and XMPP, both IP versions are supported. The selection of the IP version is only dependent on the implementation.

Cellular Provider and Technology:

- IoT devices with cellular modems are dependent on the generation of the cellular technology and 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.

Serial Communications:

- Data is transferred using either proprietary or standards based protocols, such as DNP3, Modbus, or IEC 60870-5-101.
- Old days it was handled by an analog modem connection.
- connect the serial port of the legacy device to a nearby serial port on a piece of communications equipment, typically a router.
- The 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.
- Raw socket sessions can run over both IPv4 and IPv6, current implementations are mostly available for IPv4 only.

IPv6 Adaptation Layer:

- IPv6-only adaptation layers for some physical and data link layers for recently standardized IoT protocols support only IPv6.
- Most common physical and data link layers (Ethernet, Wi-Fi, and so on) stipulate adaptation layers for both versions, newer technologies, such as IEEE 802.15.4 (Wireless Personal Area Network), IEEE 1901.2, and ITU G.9903 (Narrowband Power Line Communications) only have an IPv6 adaptation layer specified.
- It is reinforced by the IETF routing protocol for LLNs, RPL, which is IPv6 only.

Routing What is RPL?

- The IETF Routing Over Low-power and Lossy networks (ROLL) Working Group was formed in 2008
  - to create an IP level routing protocol adapted to the requirements of mesh networking for IoT/M2M
- The first version of RPL (Routing Protocol for Low-power and lossy networks) was finalized in April 2011
- Current standard: RFC 6550 (March 2012)
  - based on distance vector algorithms

Functionality of RPL

- RPL specifies a routing protocol specially adapted for the needs of IPv6 communication over “low-power and lossy networks” (LLNs), supporting
  - peer to peer traffic (point to point) (P2P)
  - point to multipoint (P2MP) communication: from a central server to multiple nodes on the LLN
  - multipoint to point (MP2P) communication
- The base RPL specification is optimized only for MP2P traffic or P2MP, and P2P is optimized only through use of additional mechanisms.

Topology

- RPL organizes a topology as a Directed Acyclic Graph (DAG) that is partitioned into one or more Destination Oriented DAGs (DODAGs), one DODAG per sink.
- A RPLInstanceID identifies a set of one or more Destination Oriented DAGs (DODAGs).
- The set of DODAGs identified by a RPLInstanceID is called a RPL Instance. All DODAGs in the same RPL Instance use the same OF.

- When a node A sends a packet to a node B within the RPL domain, the packet first follows the graph up to the root where the routing information is stored.
- At this point, the graph root inspects the destination, consults its routing table that contains the path to the destination (obtained from DAO messages received).
- The root “source -routes” the packet to its destination using a specific routing header for IPv6

RPL Control Messages

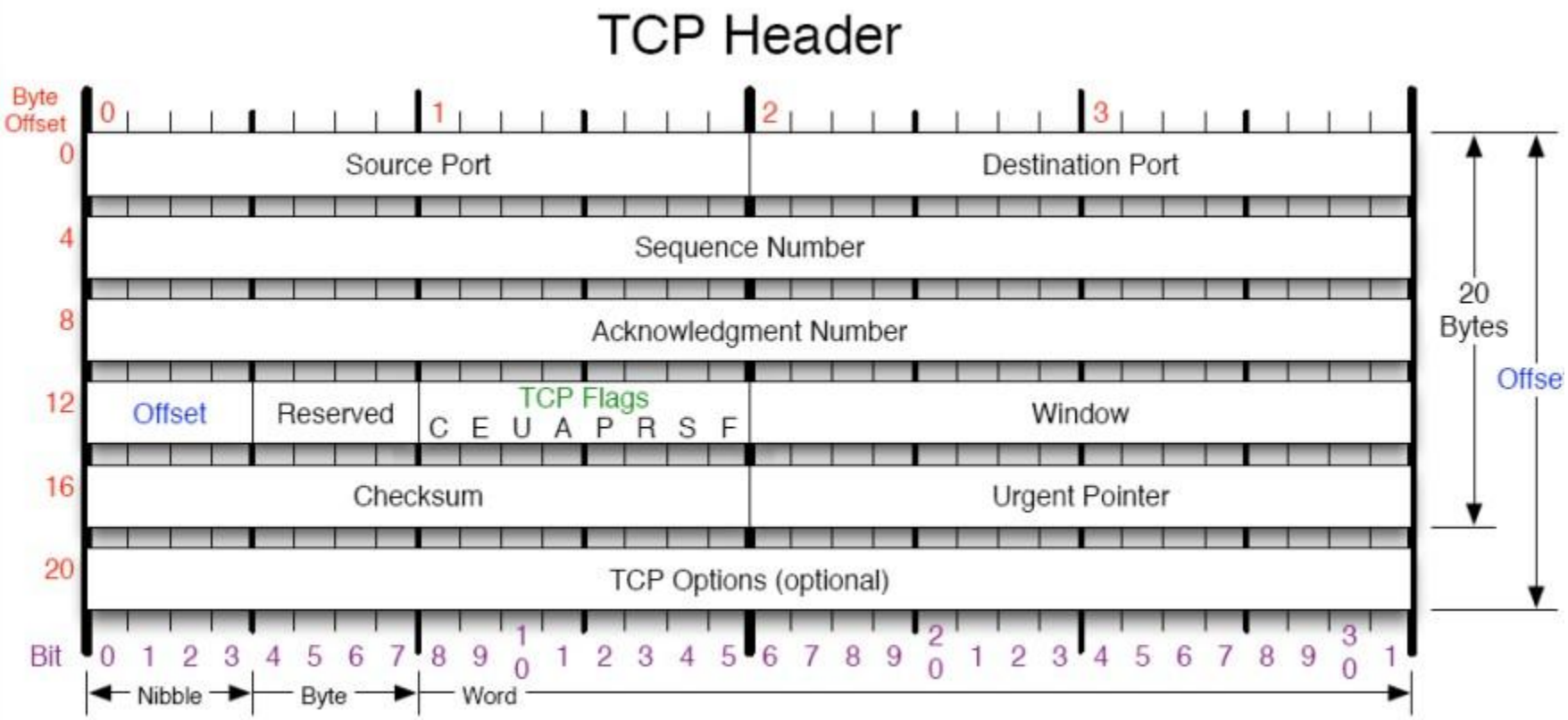
- RPL Control message are ICMPv6 messages
  - DAG Information Object (DIO) - carries information that allows a node to discover an RPL Instance, learn its configuration parameters and select DODAG parents
  - DAG Information Solicitation (DIS) - solicit a DODAG Information Object from a RPL node
  - Destination Advertisement Object (DAO) - used to propagate destination information upwards along the DODAG.
  - DAO-ACK: Destination Advertisement Object Acknowledgement

+ The 4 secured versions

Control Message Exchange

- Each DODAG, uniquely identified by RPLInstanceID and DODAGID, is incrementally built from the root to the leaf nodes.
  - RPL nodes send DIOs periodically via link-local multicasts.
  - Joining nodes may request DIOs from their neighbors by multicasting DIS (DODAG Information Solicitation) .
  - DTSN (Destination Advertisement Trigger Sequence Number) is a 8-bit unsigned integer set by the issuer of the message. In the storing mode, increasing DTSN is to request updated DAOs from child nodes.

TCP Header


TCP Header
Byte Offset
0
4
8
12
16
20
0
1
2
3
Source Port
Destination Port
Sequence Number
Acknowledgment Number
Offset
Reserved
TCP Flags
C E U A P R S F
Window
Checksum
Urgent Pointer
TCP Options (optional)
20 Bytes
Offse
Bit
0 1 2 3 4 5 6 7 8 9 10 1 2 3 4 5 6 7 8 9 20 1 2 3 4 5 6 7 8 9 30 1
Nibble
Byte
Word

Compressing UDP Checksum

- UDP checksum is mandatory with IPv6
- In RFC 6282, an endpoint MAY elide the UDP Checksum if it is authorized by the upper layer.
  - Tunneling: tunneled Protocol Data Unit (PDU) possesses its own addressing, security and integrity check
  - Message Integrity Check: e.g., IPsec Authentication Header

Application Layer protocols

- CoAP- The Constrained Application Protocol (CoAP) is defined by IETF CoRE WG for the manipulation of resources on a device that is on the constrained IP networks.
- CoAP is
  - A RESTful protocol
  - Both synchronous and asynchronous
  - For constrained devices and networks
  - Specialized for M2M applications
  - Easy to proxy to/from HTTP
- CoAP is not
  - A replacement for HTTP
  - General HTTP compression
  - Separate from the web

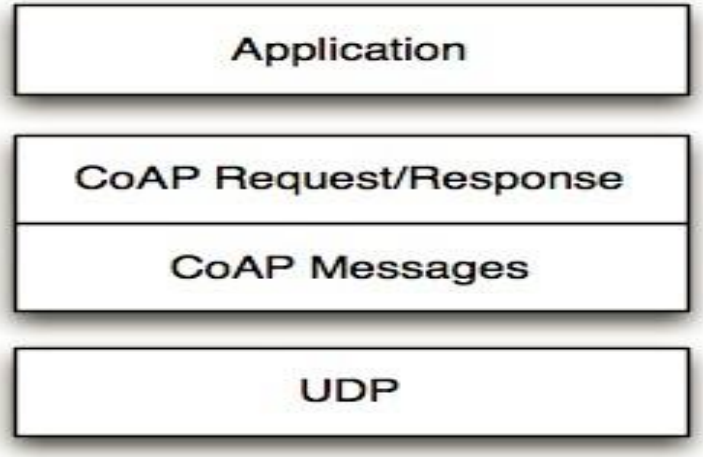
CoRE Documents

Number	Title	Date	Status
RFC 7252 RFC 7959	The Constrained Application Protocol (CoAP)	June 2014 August 2016	Proposed Standard
RFC 7390	Group Communication for the Constrained Application Protocol (CoAP)	October 2014	Experimental
RFC 7641	Observing Resources in the Constrained Application Protocol (CoAP)	September 2015	Proposed Standard
RFC 7650	A Constrained Application Protocol (CoAP) Usage for REsource LOcation And Discovery (RELOAD)	September 2015	Proposed Standard
RFC 8323	CoAP (Constrained Application Protocol) over TCP, TLS, and WebSockets	February 2018	Proposed Standard

Application Scope of CoAP

- CoAP targets the type of operating environments defined in the ROLL and 6LOWPAN working groups which have additional constraints compared to normal IP networks, but the CoAP protocol will also operate over traditional IP networks.
- This includes applications to monitor simple sensors (e.g. temperature sensors, light switches, and power meters), to control actuators (e.g. light switches, heating controllers, and door locks), and to manage devices.

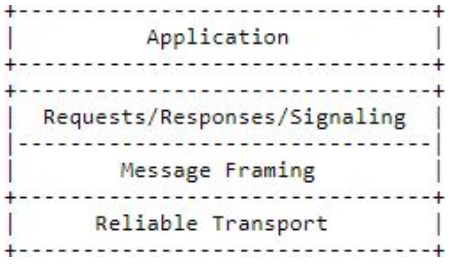
CoAP RESTful Applications



Source: IETF IPv6 WG

CoAP vs. HTTP

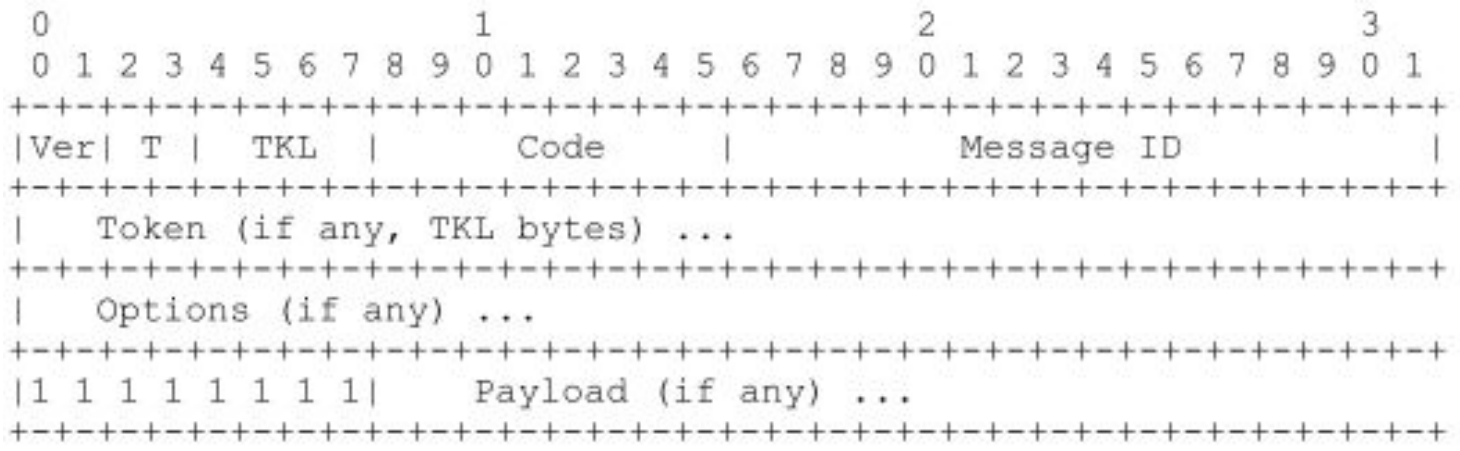
- Like HTTP, the CoAP is a way of structuring REST communications but optimized for M2M applications.
- TCP and HTTP are considered too heavy for 6LowPAN devices such as sensors. CoAP is thus based on UDP and a compressed simplified message exchange.
  - NB: RFC 8323 extends CoAP over TCP/TLS



CoAP Methods

- CoAP makes use of GET, PUT, POST, and DELETE methods in a similar manner to HTTP.
- New methods can be added, and do not necessarily have to use requests and responses in pairs.
  - For example: OBSERVE (embedded in GET method)

CoAP Message Format



Ver - Version (1) 2-bit

T - Transaction Type 2-bit

- CON (0) – Confirmable
- NON (1) – Non-Confirmable
- ACK (2) – Acknowledgement
- RST (3) – Reset

Token Length (TKL): 4-bit

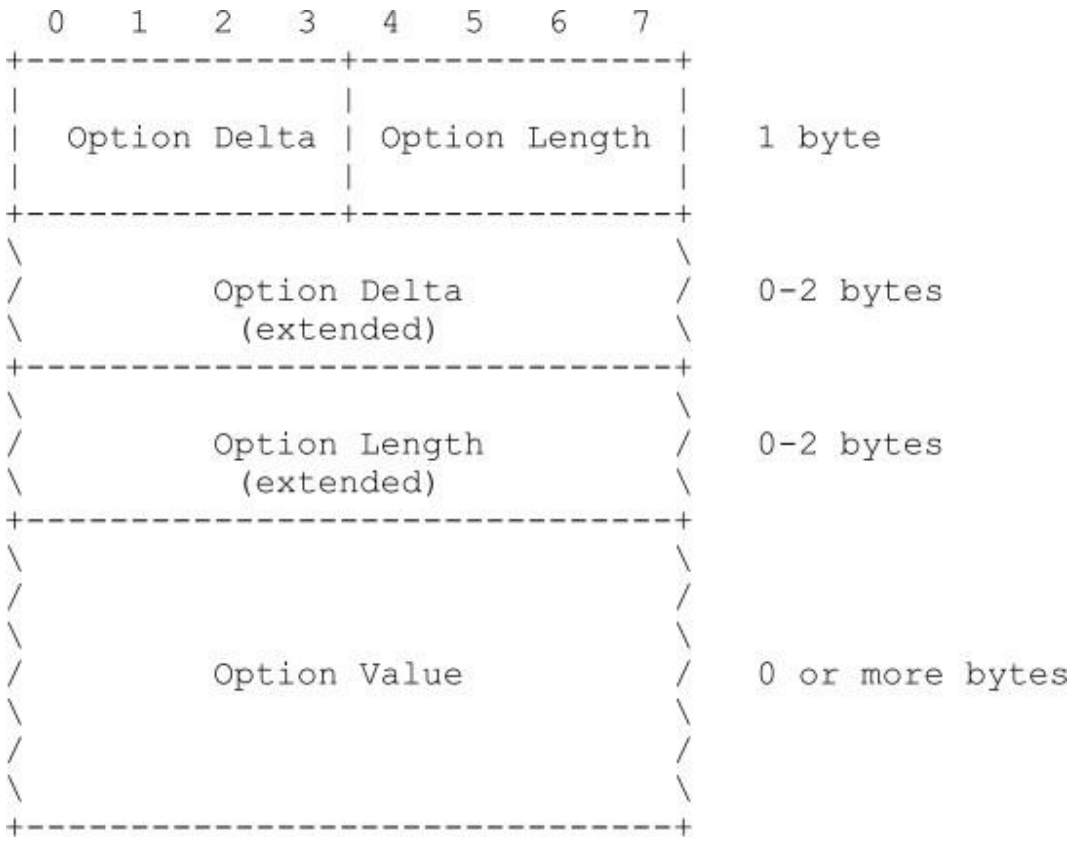
Code – 3-bit class, 5-bit detail ("c.dd“)

Class: request (0), success response (2), client error response (4), server error response (5) (see next page for details)

Message ID - Identifier for matching responses

Token - used to correlate requests and responses.

CoAP Options



Option Delta - Difference between this option type and the previ
Length - Length of the option value (0-270)
Value - The value of Length bytes immediately follows Length

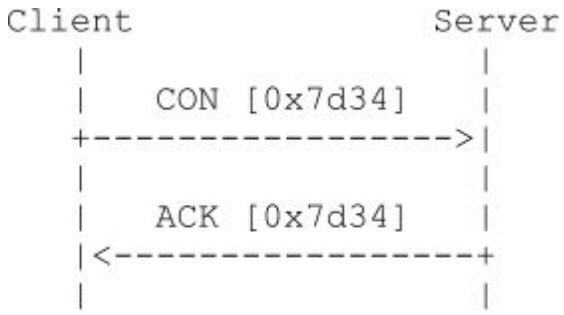
CoAP URI

- coap-URI = "coap:" "//" host [":" port] path-abempty ["?" query]
  - coap://example.com:5683/~sensors/temp.xml
  - coap://EXAMPLE.com/%7Esensors/temp.xml

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

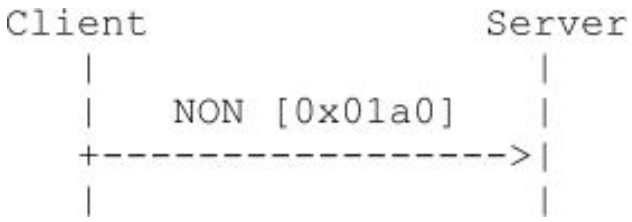
Messaging Model

- Reliable Message Transmission:
  - Reliability is provided by marking a message as Confirmable (CON).
  - A Confirmable message is retransmitted using a default timeout and exponential back-off between retransmissions, until the recipient sends an Acknowledgement message.

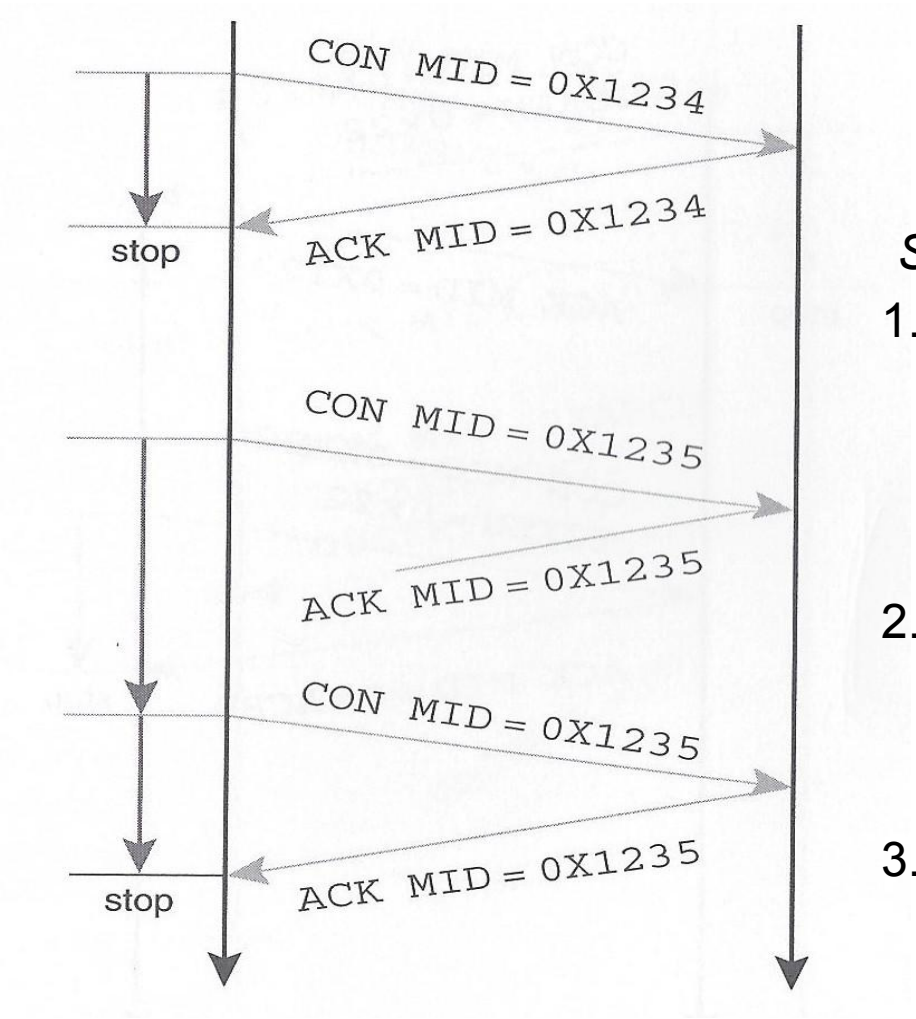


Messaging Model

- Unreliable Message Transmission:
  - A message that does not require reliable transmission can be sent as a Non-confirmable message (NON).
  - These are not acknowledged.



Example 1 of CoAP Requests

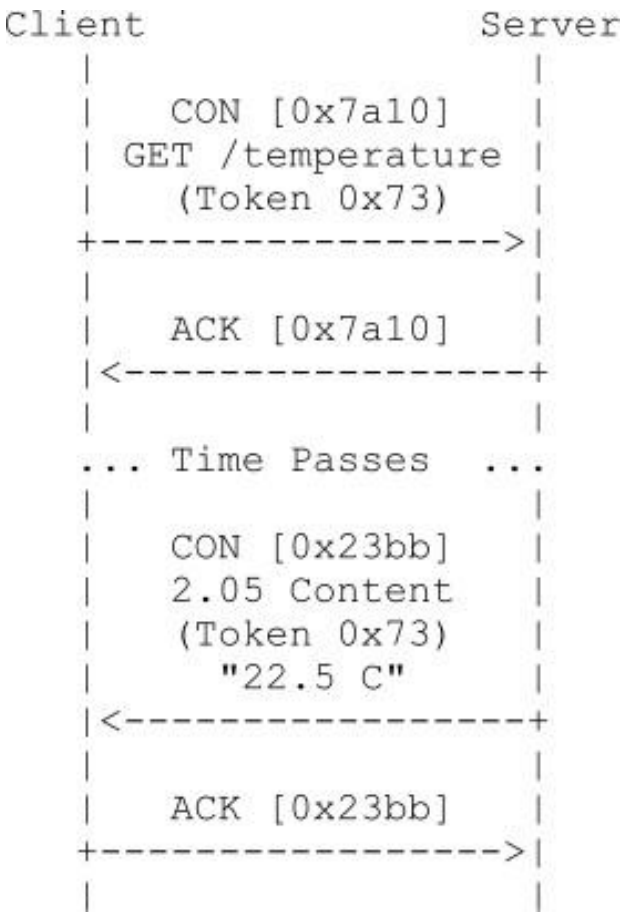


Synchronous Message Exchange

1. A CONfirmable message followed by ACKowledgement piggybacked with the response in the same Message ID (MID).
2. When ACKnowledgment was lost, Client's timer expires and it resends the message.
3. Exponential back-off between retransmissions.

Source: M2M Communications: A Systems Approach, Wiley, 2012

Example 2a of CoAP Requests

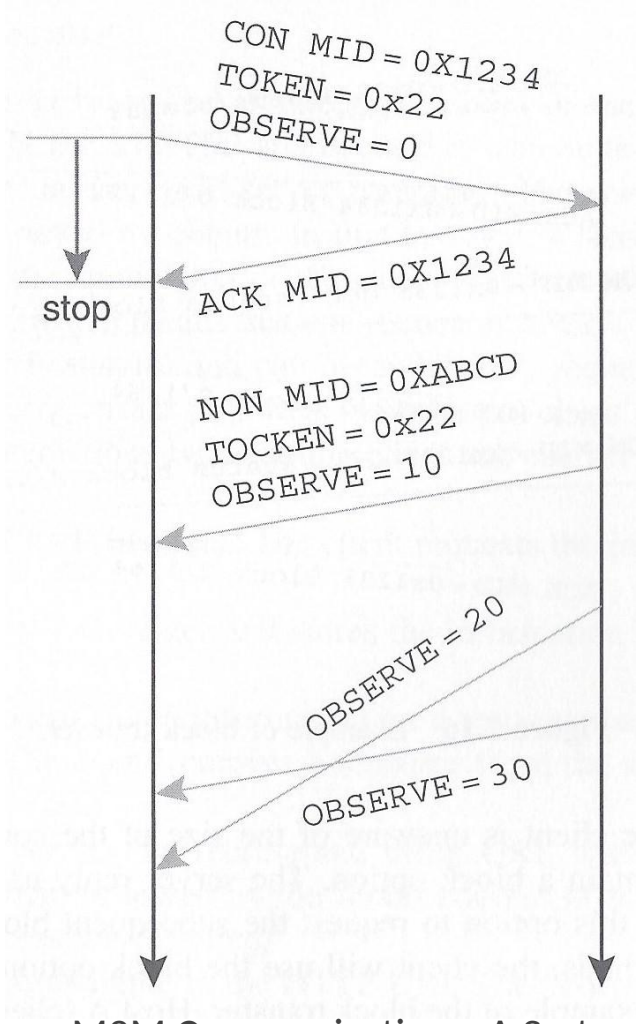


Asynchronous Message Exchange

1. A CONfirmable message with TOKEN option can be acknowledged immediately with an Empty Acknowledgement.

2. When the response is available, it can be returned in a new CON message with the same TOKEN ID.

Example 3a of CoAP Requests (OBSERVE)



Periodic response from a server

1. A CONfirmable message from the client contains OBSERVE option asking periodic responses from the server.
2. The server send NON responses with the same TOKEN ID.
3. OBSERVE is the response will be increased to indicate the order of the response.
4. The client will ignore OBSERVE=20 since it arrives later than OBSERVE=30.
5. Either client or server can terminate the process.

Source: M2M Communications: A Systems Approach, Wiley, 2012

- **Open source software available**
- **http://coapy.sourceforge.net/index.html CoAPy: Constrained Application Protocol in Python**

MQTT

- MQTT = MQ Telemetry Transport
- MQTT protocol is a lightweight publish/subscribe protocol flowing over TCP/IP for remote sensors and control devices through low bandwidth, unreliable or intermittent communications.
- MQTT was developed by Andy Stanford-Clark of IBM, and Arlen Nipper of Cirrus Link Solutions more than a decade ago.

- ISO/IEC 20922:2016 Message Queuing Telemetry Transport (MQTT) v3.1.1. (Draft of v5.0 published in July, 2017)
- Client libraries are available in almost all popular languages now.

  https://eclipse.org/paho/
- The current MQTT specification is available at:
  - http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html
  - http://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html

Projects that Implement MQTT

- Amazon Web Services announced Amazon IoT based on MQTT in 2015
- Microsoft Azure IoT Hub uses MQTT as its main protocol for telemetry messages
- Node-RED supports MQTT nodes as of version 0.14
- Facebook has used aspects of MQTT in Facebook Messenger for online chat
- The EVRYTHNG IoT platform uses MQTT as an M2M protocol for millions of connected products.

Features of MQTT

- It supports publish/subscribe message pattern to provide one-to-many message distribution and decoupling of applications
- A messaging transport that is agnostic to the content of the payload
- Three qualities of service (QoS) for message delivery:
  - "At most once" , where messages are delivered according to the best efforts of the operating environment. Message loss can occur.This level could be used, for example, with ambient sensor data where it does not matter if an individual reading is lost as the next one will be published soon after.
  - "At least once", where messages are assured to arrive but duplicates may occur.
  - "Exactly once", where message are assured to arrive exactly once.This level could be used, for example, with billing systems where duplicate or lost messages could lead to incorrect charges being applied.
- A small transport overhead and protocol exchanges minimized to reduce network traffic.

MQTT Pub/Sub Protocol

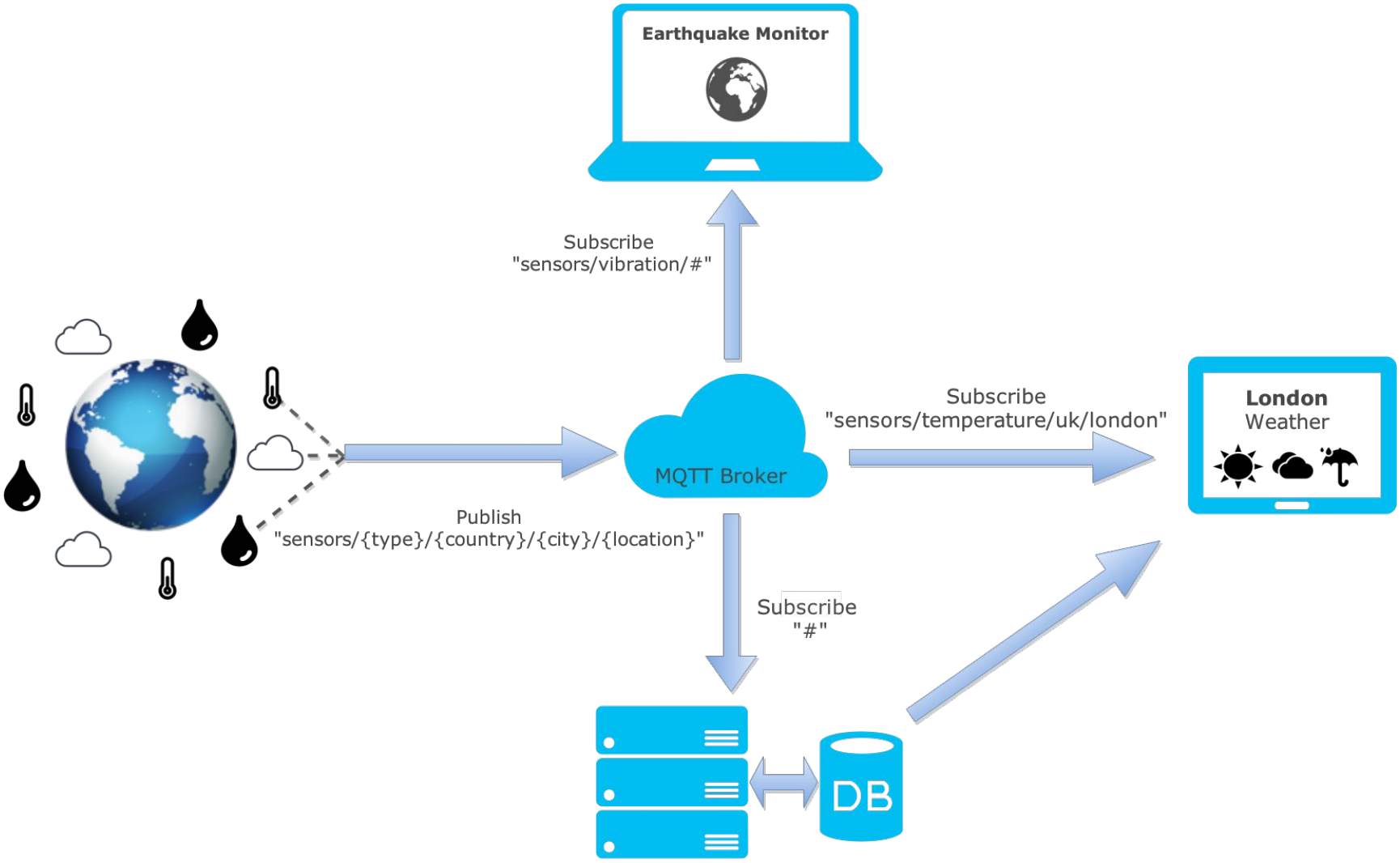
- MQ Telemetry Transport (MQTT) is a lightweight broker-based publish/subscribe messaging protocol.
- MQTT is designed to be open, simple, lightweight and easy to implement.
  - These characteristics make MQTT ideal for use in constrained environments, for example in IoT.
    - Where the network is expensive, has low bandwidth or is unreliable
    - When run on an embedded device with limited processor or memory resources;
- A small transport overhead (the fixed-length header is just 2 bytes), and protocol exchanges minimized to reduce network traffic

Source: MQTT V3.1 Protocol Specification, IBM,
http://public.dhe.ibm.com/software/dw/webservices/ws-mqtt/mqtt-v3r1.html

Suitable for Constrained Networks

- Protocol compressed into bit-wise headers and variable length fields.
- Smallest possible packet size is 2 bytes
- Asynchronous bidirectional “push” delivery of messages to applications (no polling)
- Client to server and server to client
- Supports always-connected and sometimes-connected models
- Provides Session awareness
- Configurable keep alive providing granular session awareness
- “Last will and testament” enable applications to know when a client goes offline abnormally
- Typically utilizes TCP based networks e.g. ,Webscokets
- Tested on many networks

Broker/Publish/Subscribe



https://zoetrope.io/tech-blog/brief-practical-introduction-mqtt-protocol-and-its-application-iot/

MQTT Message Format

- Structure of an MQTT Control Packet
  - The message header for each MQTT command message contains a fixed header.
  - Some messages also require a variable header and a payload.

Fixed header, present in all MQTT Control Packets
Variable header, present in some MQTT Control Packets
Payload, present in some MQTT Control Packets

XMPP?

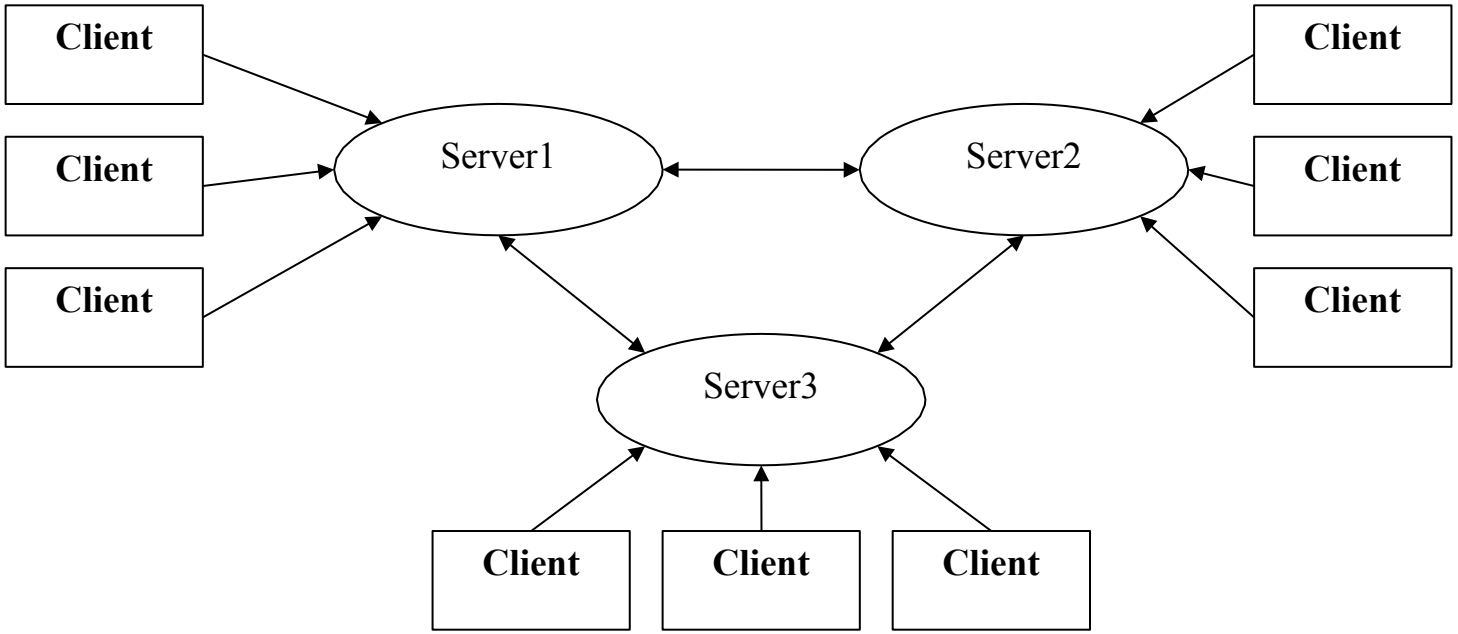
- eXtensible Messaging and Presence Protocol Bi-directional streaming XML
- Core: IETF RFC 6120, 7590, 6121
- Extensions: XMPP Standards Foundation (XSF)
  - Membership-based
  - Elected technical council
  - Unit of work: XMPP Extension Protocol (XEP)
  - Process: Experimental, Proposed, Draft, Final
- Goals:
  - Simple clients
  - Federate everything

What is XMPP ?

- The eXtensible Messaging and Presence Protocol (XMPP) is a TCP communications protocol based on XML that enables near-real-time exchange of structured data between two or more connected entities.
- Out-of-the-box features of XMPP include presence information and contact list maintenance.
- Due in part to its open nature and XML foundation, XMPP has been extended for use in publish-subscribe systems
  - Perfect for IoT applications.

https://www.infoworld.com/article/2972143/internet-of-things/real-time-protocols-for-iot-apps.html

XMPP Architecture

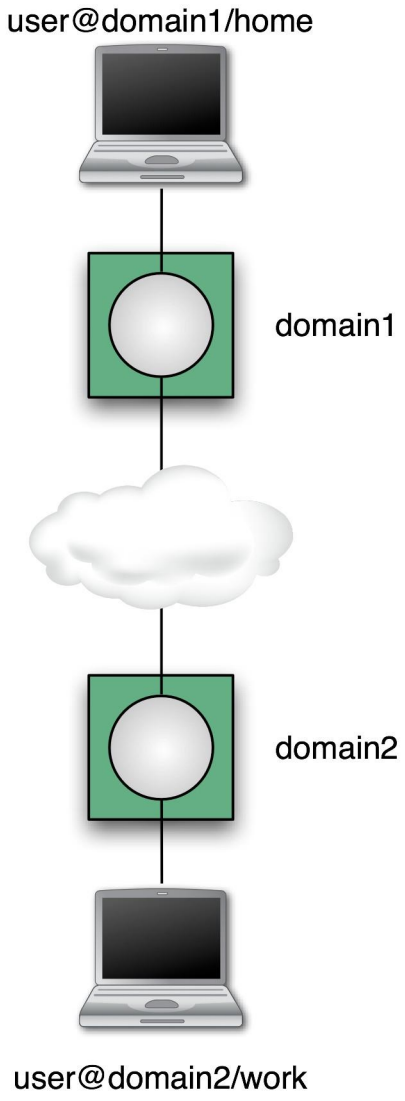


Server<->Server: Port 5269

Client<->Server: Port 5222

XMPP Architecture

- Addressing Scheme:
`node@domain/resource`
  - JID = Jabber ID
  - Node: identity, e.g. user name
  - Domain: DNS domain name
  - Resource: device identifier
  - `node@domain` identifies a person
- Client talks to "local" server
  - Wherever the user account is hosted
  - Tied to directory if desired
  - Organizational policy enforced
- Servers talk to other servers
  - DNS lookup on domain portion of address
  - Dialback, MTLS for security
  - One connection for many conversations



Advantages of XMPP

- The primary advantage is XMPP's decentralized nature.
- XMPP works similar to email, operating across a distributed network of transfer agents rather than relying on a single, central server or broker (as CoAP and MQTT do).
- As with email, it's easy for anyone to run their own XMPP server, allowing device manufacturers and API operators to create and manage their own network of devices.
- And because anyone can run their own server, if security is required, that server could be isolated on a company intranet behind secure authentication protocols using built-in TLS encryption.

Disadvantages of XMPP

- One of the largest flaws is the lack of end-to-end encryption.
- Another downside is the lack of Quality of Service (QoS).