IPv6物聯網教育訓練

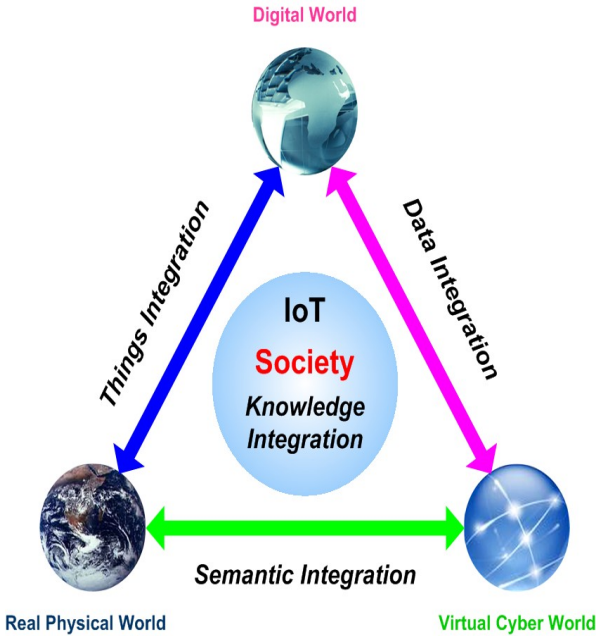
黃仁竑 教授
國立中正大學資工系

Outline

- 物聯網簡介 (Introduction to IoT)
- 物聯網重要網路協定(IoT Network Protocols)
  - IPv6
  - 6LoWPAN
  - RPL
  - CoAP
- 物聯網國際標準服務架構(ESTI M2M Service Architecture)
  - ESTI Service Capability and OM2M
  - oneM2M

物聯網簡介

- 什麼是物聯網?
  - 綜合了近幾年的發展，物聯網是要在人類真實生活世界中的物體上，植入各種感測器，使其具有智慧，可以藉由各種(無線)網路連結上網，使這些物體上的資訊可以被擷取(分享)、收集，做為智慧控制的決策依據，並能將決策結果透過遠端指令，進行智能控制，最終能提供物體與人、物體與物體、人與人之間等各種不同類型的溝通與對話，促進智慧型系統對萬物進行高效、節能、安全、環保等目標之實現。(黃仁竑, 2013)



IoT Overview

- Connect anything to the Internet
  - Anything that can be connected, will be connected
    - Cell phones, coffee makers, washing machines, headphones, lamps, wearable devices; anything you can think of.
  - Accessible from the Internet based on standard protocols
    - Not just in a closed system
- Intelligent system
  - Decision, control, integration, application
  - Smart city, smart home, smart planet, smart grid, smart transportation, …

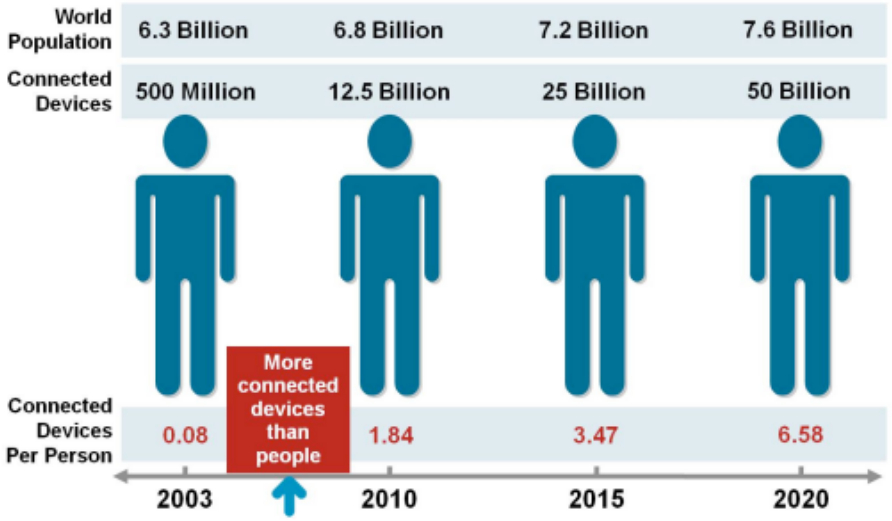
IoT Overview

- Massive (sensing) devices
  - Tens of billions of devices capturing information around us
- Connected to the Internet
  - Information is available through the Internet
  - Standardized interface
  - Semantic addressing
- Intelligent applications
  - Information is collected, processed, analyzed which …

  is used to help us derive greater knowledge, make smart decisions, build smart applications (context-aware), …

IoT Overview

- 多少物件連網?
  - 思科(Cisco IBSG)預測2015年會有250億設備，2020年會有500億設備連上網路。



by Dave Evans

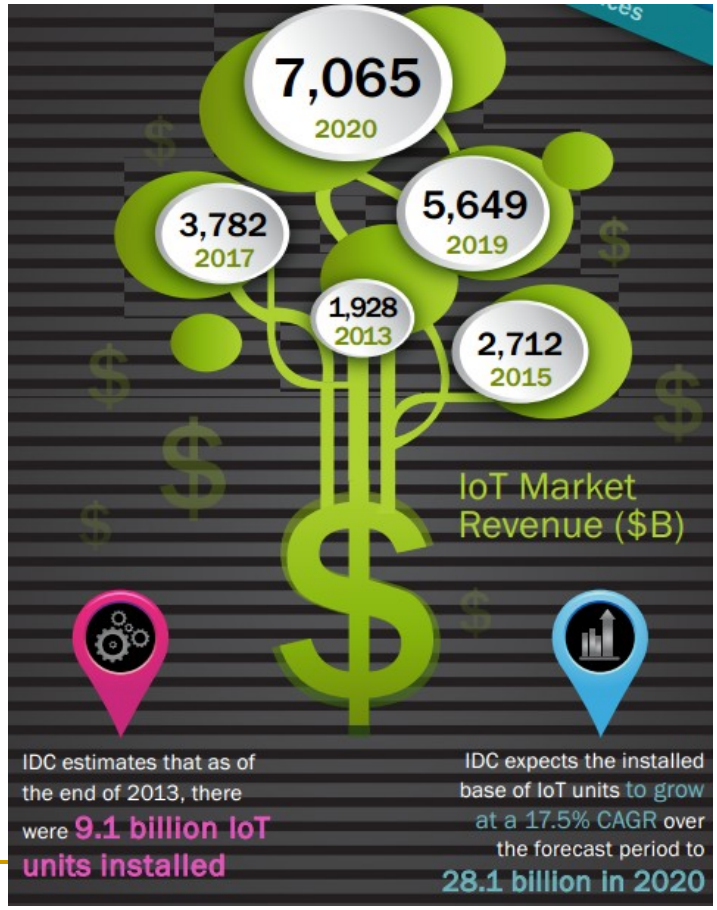
  - 高德納(Gartner)預測在2020年有260億設備連上網路
  - IDC預測在2020年有281億設備連上網路

物聯網簡介

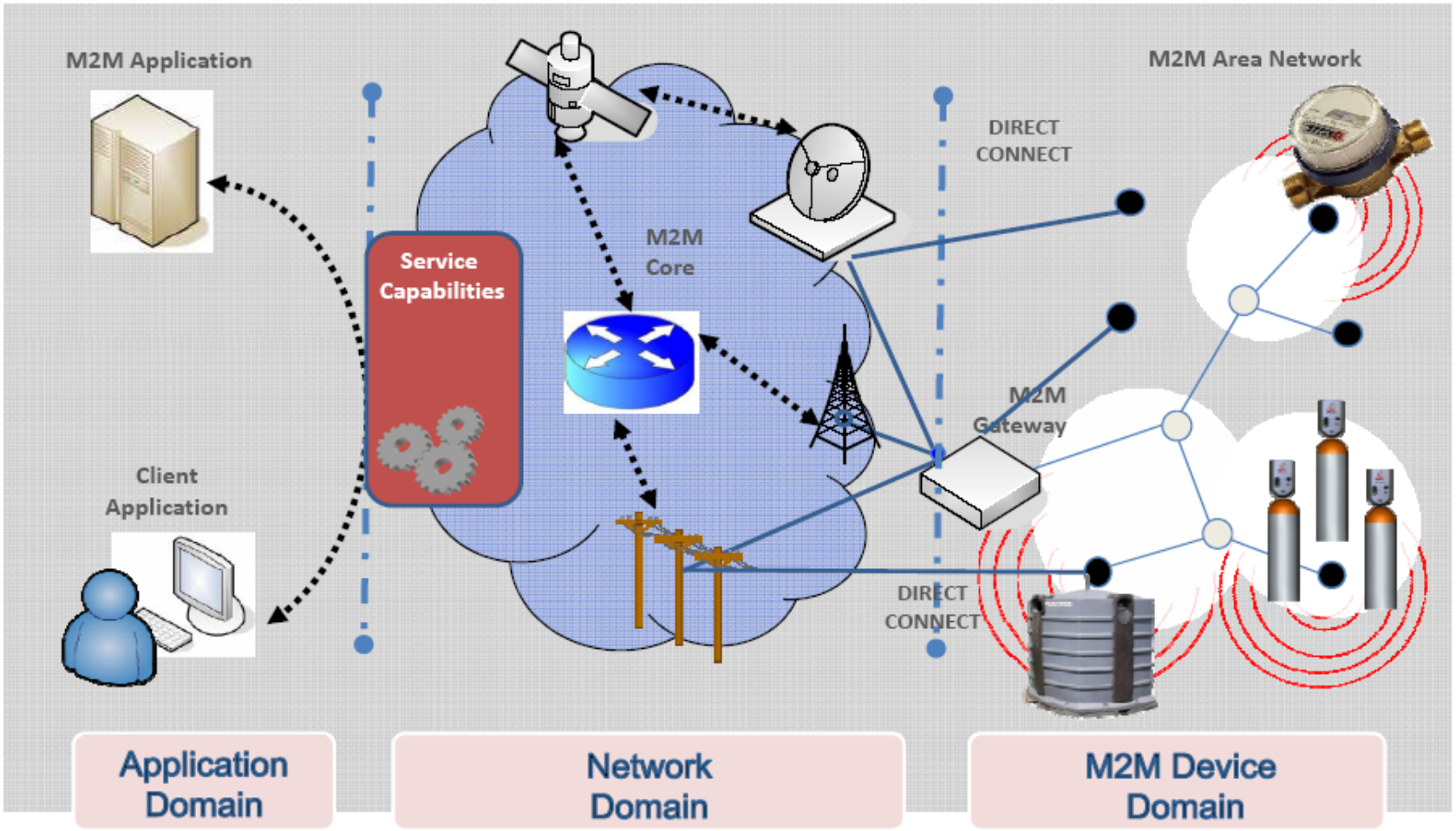
- 多少物件連網?
  - IHS Automotive對smart car進行預測，在2020年會有超過1億5千萬台汽車上Internet
  - ABI Research對smart car的預測是到2030年會有4億台vehicle搭載IoT技術
  - On World預估2020年會有超過1億個的無線燈泡連上網路
  - Acquity Group對一些裝置在未來5年的成長做出以下預測
    - Smart thermostat (智慧恆溫系統)：30%
    - Connected security system(連網保全系統)：25%
    - Smart refrigerator(智慧冰箱)：21%
    - Wearable fitness device(穿戴式健身裝置)：20%
    - Smart watch(智慧型手錶)：18%
    - Self-driving vacuum cleaner(自動掃地機)：15%
    - Wearable heads up display（抬頭顯示器）：13%
    - Smart Clothing(智慧型衣服)：10%

物聯網簡介

- 物聯網產值
  - IDC (International Data Corporation)預估
    - 2015年會達到2兆7120億
    - 2017年會達到3兆7820億
    - 2019年會達到5兆6490億
    - 2020年會達到7兆0650億
  - 高德納(Gartner)預測
    - IoT帶來的經濟附加總值
      將於2020年達1.9兆美元
  - RnRMarketResearch.com
    - 2020年會達1兆4230億

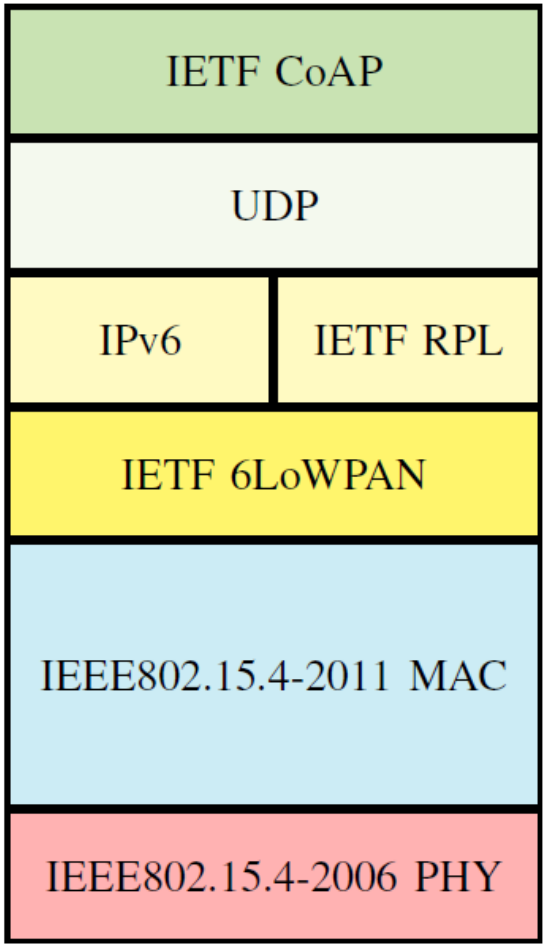


IoT Architecture



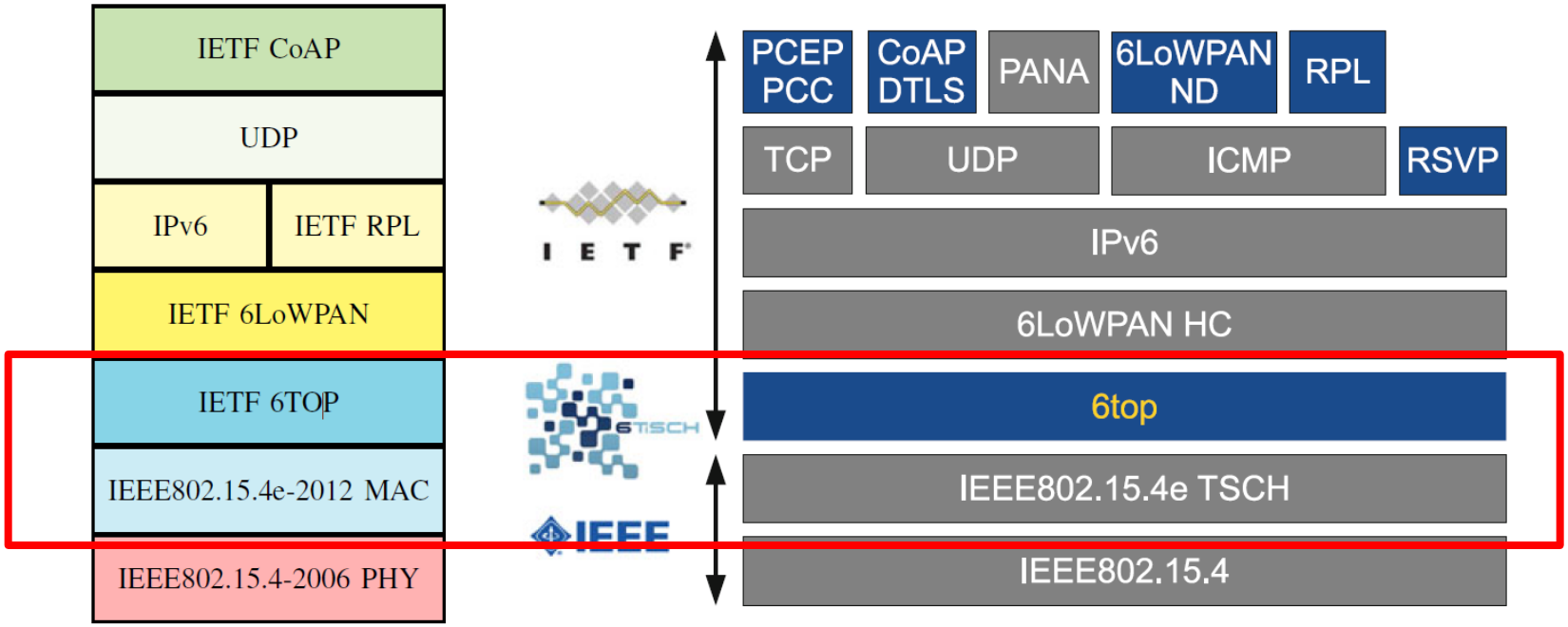
Machine Area Network

- IEEE 802.15.4 standard protocol
  - Wireless sensors (routers) may stay in high duty cycle
  - Transmit data packets on single frequency



IEEE 802.15.4e TSCH

- IPv6 over the TSCH mode of IEEE 802.15.4e (6TiSCH)



Outline

- 物聯網簡介 (Introduction to IoT)
- 物聯網重要網路協定(IoT Network Protocols)
  - IPv6
  - 6LoWPAN
  - RPL
  - CoAP
- 物聯網國際標準服務架構(ESTI M2M Service Architecture)
  - ESTI Service Capability and OM2M
  - oneM2M

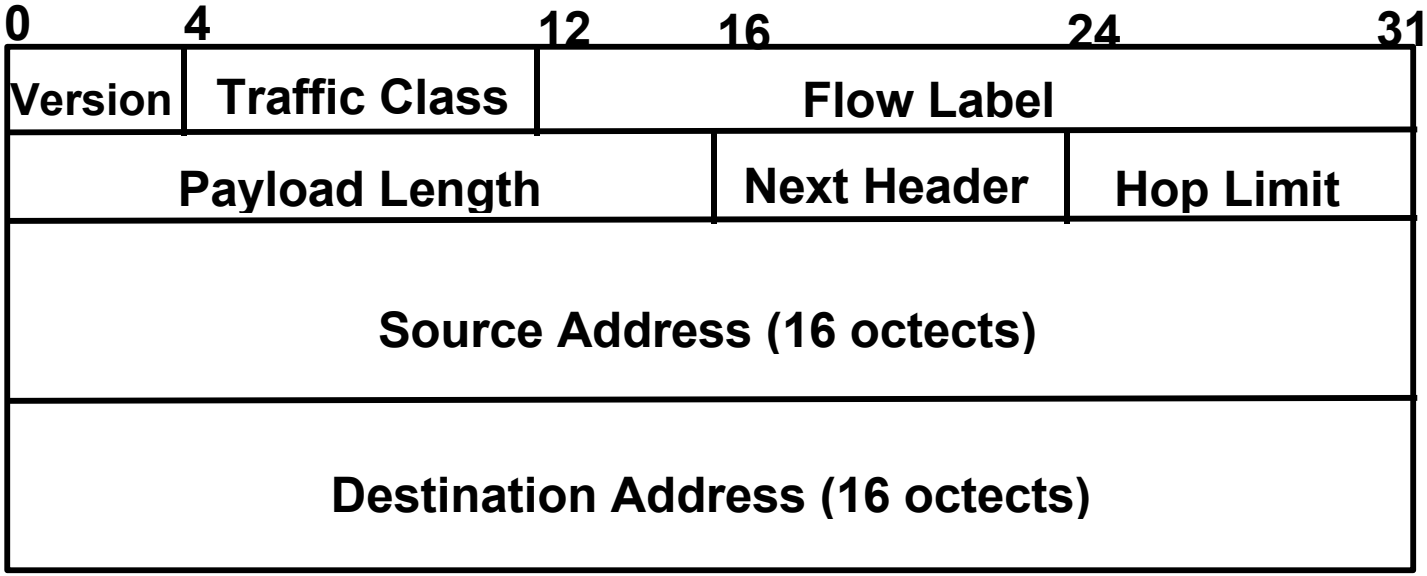
Internet Protocol Version 6 (IPv6)

Ren-Hung Hwang

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

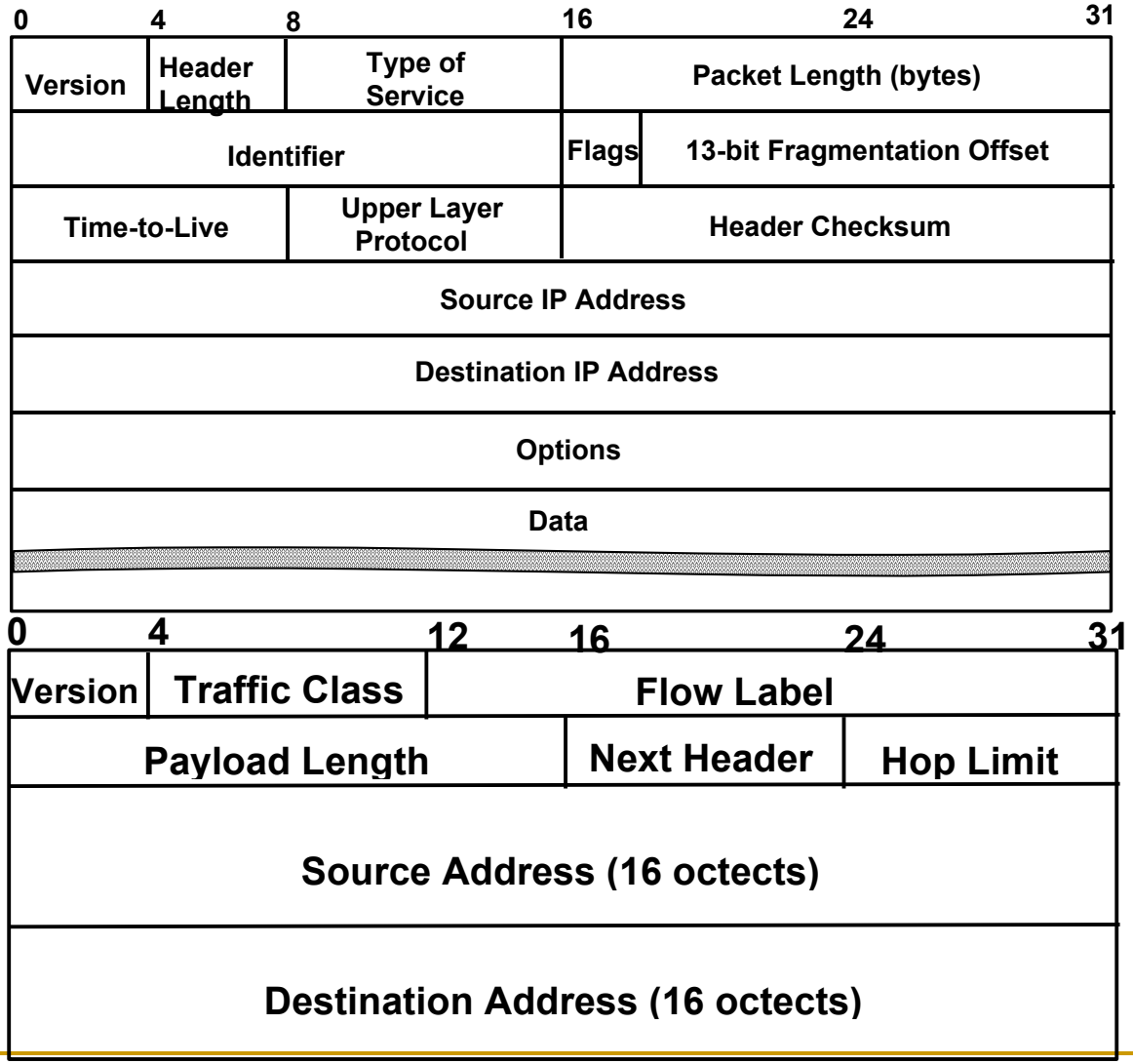
IPv6 Header



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

Changes from IPv4 (3/3)

- Checksum
  - removed to reduce processing at routers
- Fragmentation
  - Not allowed at intermediate routers

6LoWPAN(RFC 6282): IP on IEEE 802.15.4 Low-Power Wireless Networks

黃仁竑 教授

國立中正大學資工系

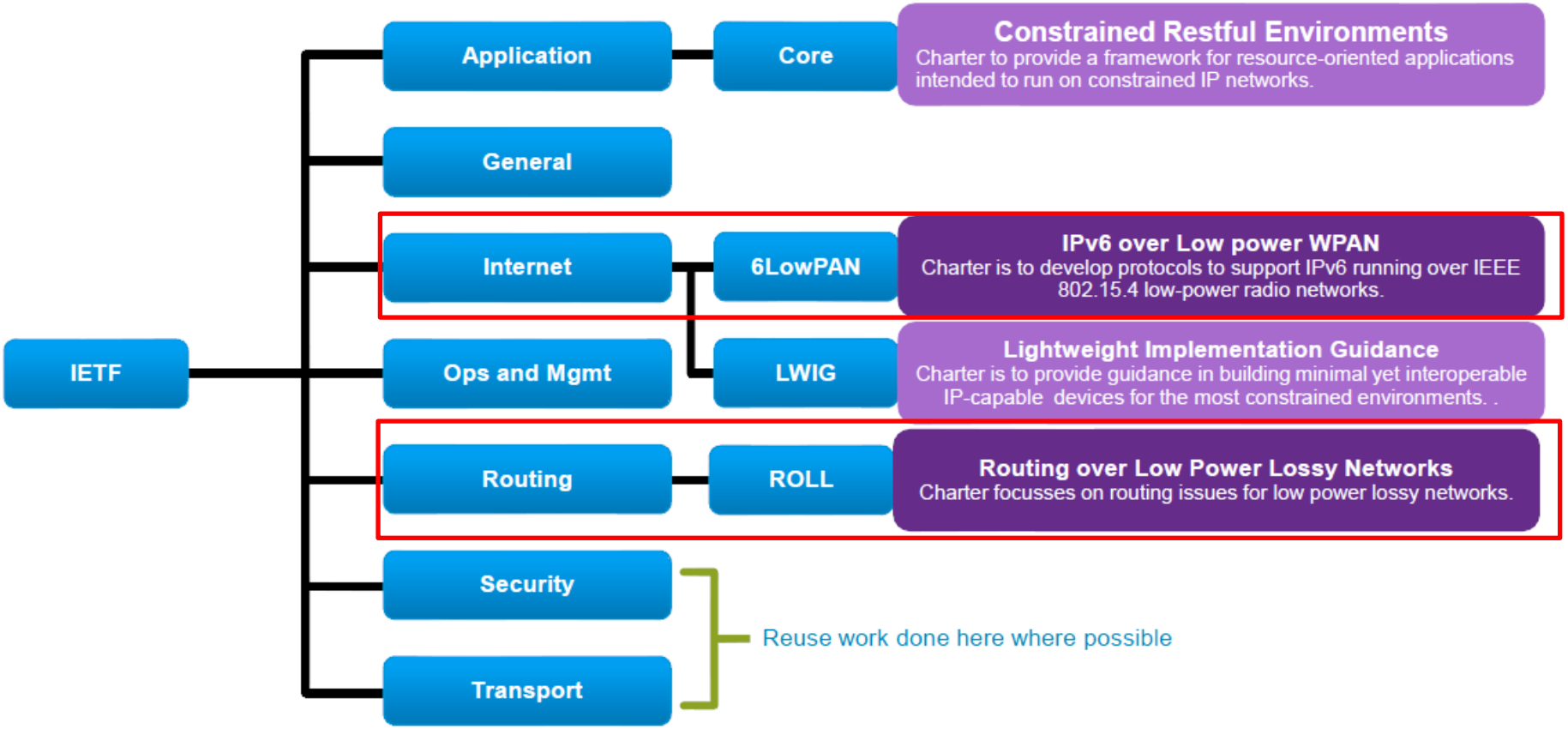
Outline

- What is 6LoWPAN?
- Motivation and Goal
- Key Elements
- Topology
- 6LoWPAN Adaptation Layer

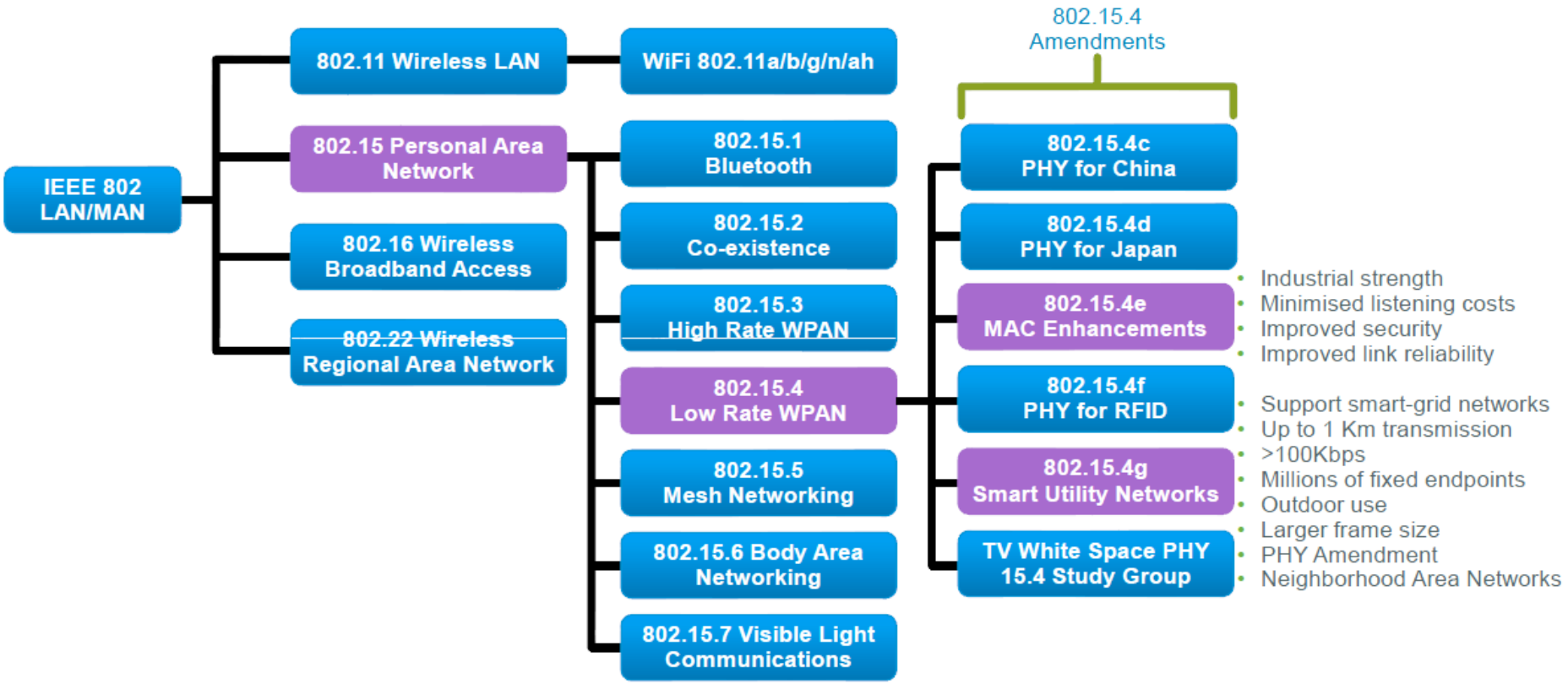
What is 6LoWPAN?

- 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 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 (draft-ietf-6lowpan-problem)	IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals	2007-08
RFC 4944* (draft-ietf-6lowpan-format)	Transmission of IPv6 Packets over IEEE 802.15.4 Networks	2007-09
RFC 6282 (draft-ietf-6lowpan-hc)	Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks	2011-09
RFC 6568 (draft-ietf-6lowpan-usecases)	Design and Application Spaces for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)	2012-04
RFC 6606 (draft-ietf-6lowpan-routing-requirements)	Problem Statement and Requirements for IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Routing	2012-05
RFC 6775 (draft-ietf-6lowpan-nd)	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

6LoWPAN WG Documents

RFC 7388	Definition of Managed Objects for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)	2014-10
RFC 7400	6LoWPAN-GHC: Generic Header Compression for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)	2014-11

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

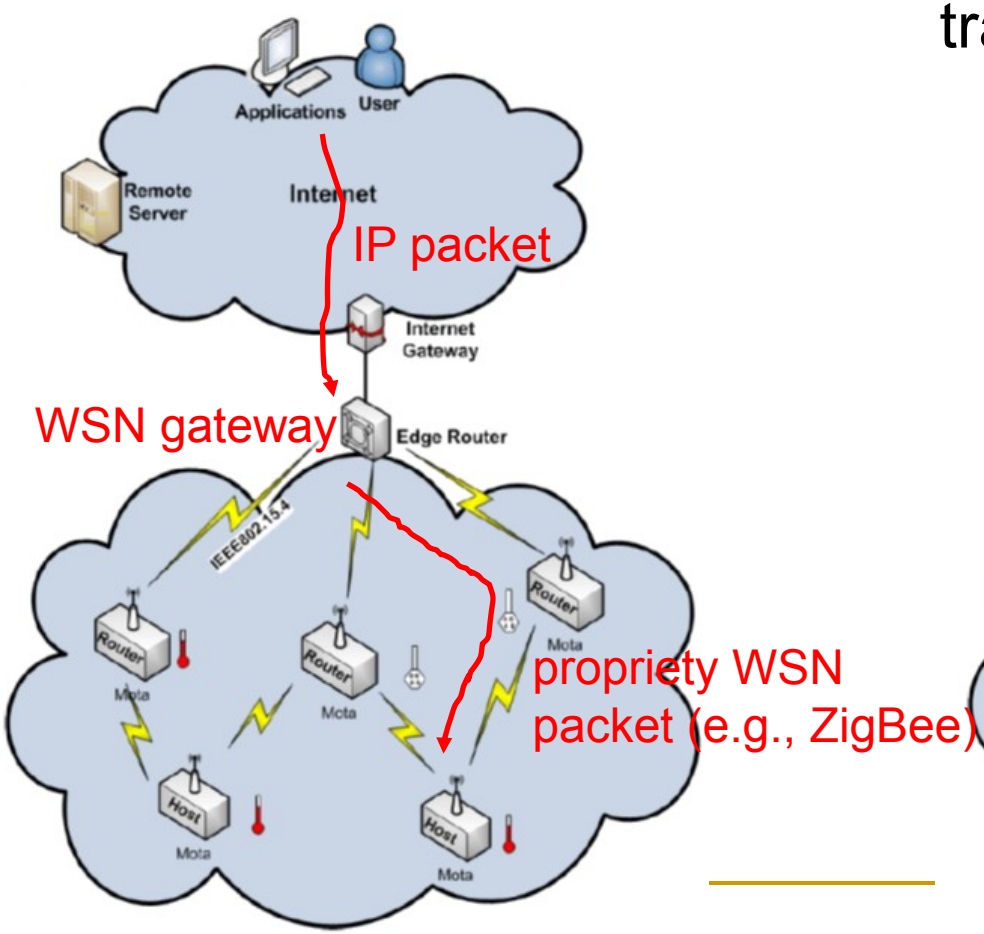
- IP, with its concept of layer 3 routing and internetwork technology, has made those debates about incompatible networks obsolete:
  - the vast majority of LANs and WANs today run IP, and many people can hardly remember which layer 2 technology their IP networks are running on.

Motivation

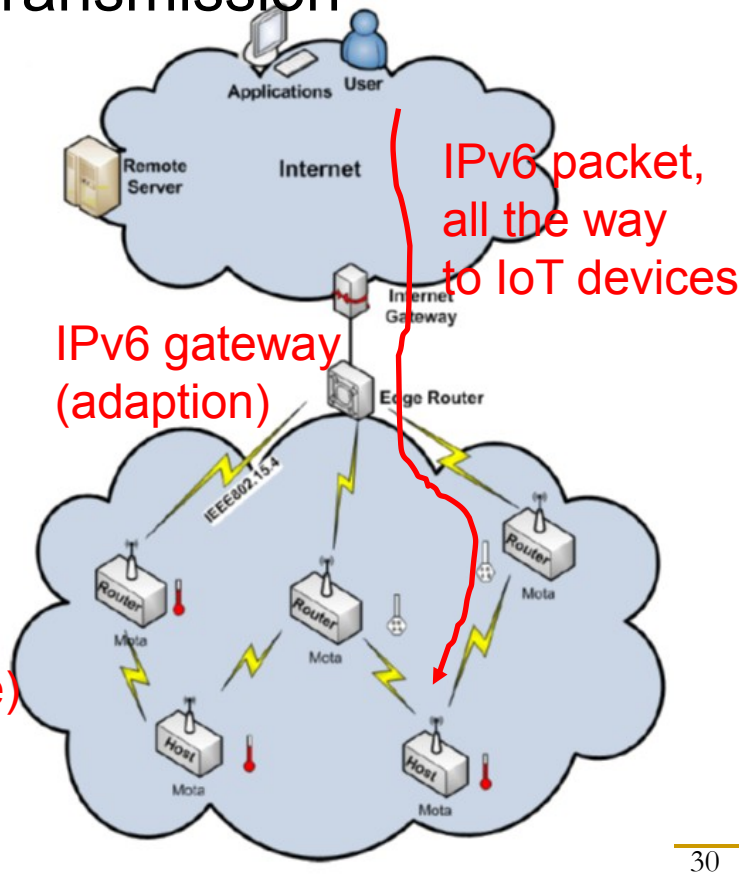
- Almost any layer 2 technology can be used and will simply extend the IP internetwork.
- The same transition to IP is now happening in the home and industrial automation worlds. 6LoWPAN and RPL have made this possible.

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 that might carry their packets
  - 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
  - Meshed
  - Routed

Star topology

- All sensor nodes can reach and are reachable from the LBR(LoWPAN Border Router)

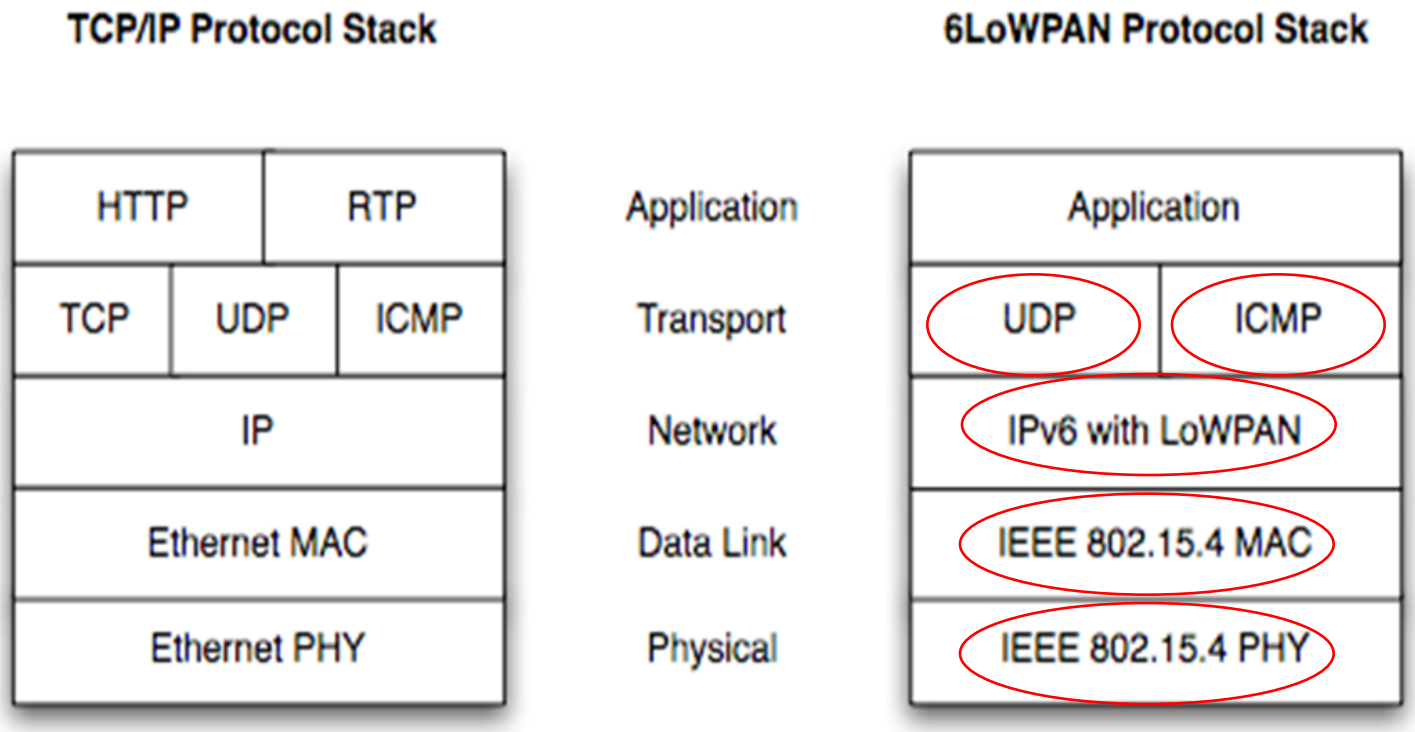
Meshed

- 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)**. The best example is RPL protocol.

6LoWPAN Protocol Stack



6LoWPAN Key Elements

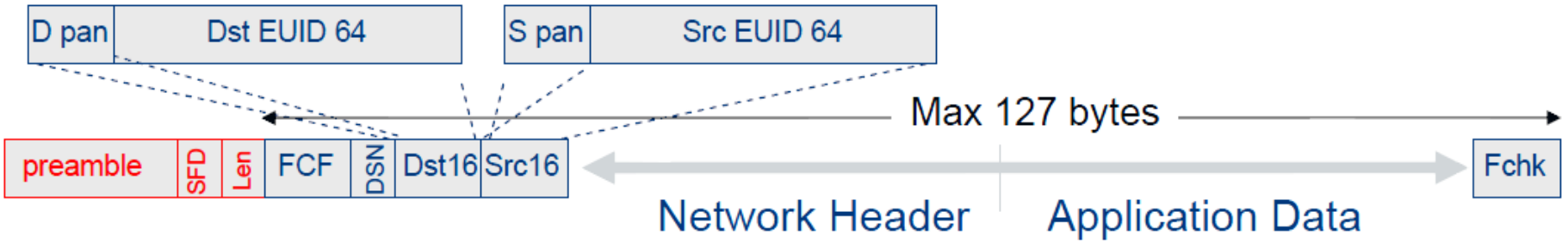
- 6LoWPAN introduces an adaptation layer between the IP stack's link and network layers to enable efficient transmission of IPv6 datagrams over 802.15.4 links
  - Provides header compression to reduce transmission overhead
  - Fragmentation to support the IPv6 minimum MTU requirement
  - Support for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

Key Concept

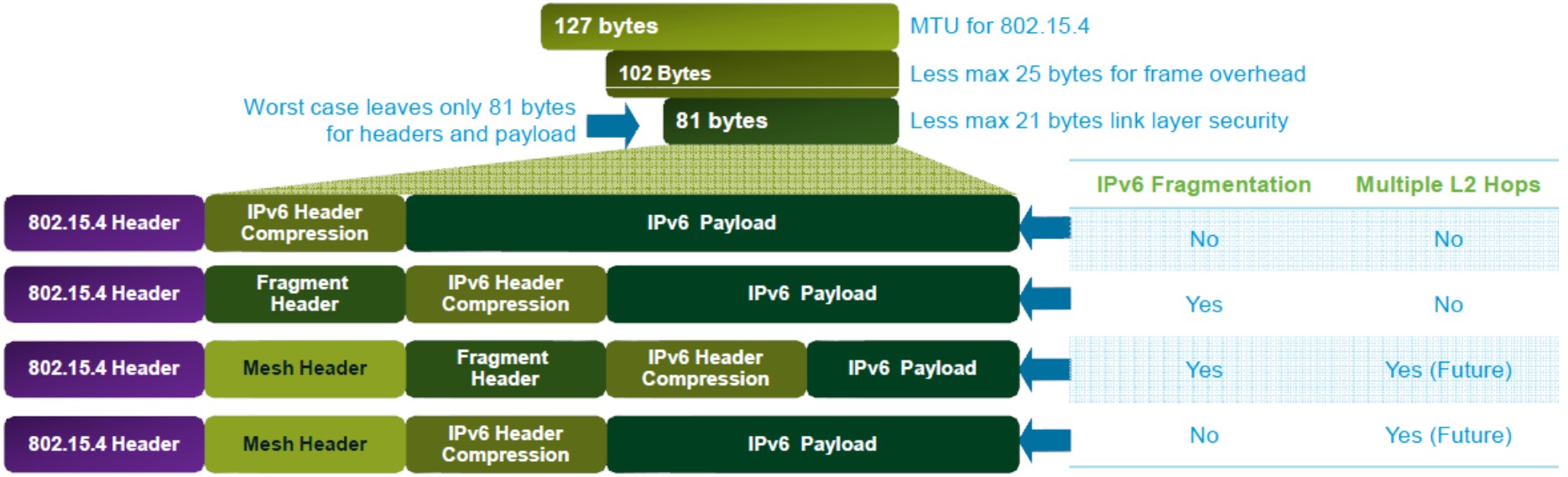
- Use of stateless or shared-context compression to elide adaptation-, network-, and transport layer header fields – compressing all three layers down to a few bytes, combined.
- It's possible to compress header fields to a few bits when we observe that they often carry common values, reserving an escape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

IEEE 802.15.4 Frame Format

Octets:2	**1**	**0/2**	**0/2/8**	**0/2**	**0/2/8**	**0/5/6/10/14**	**Variable**	**2**
Frame Control	Sequence Number	Dest. PAN Identifier	Dest. Address	Source PAN Identifier	Source Addr.	Auxiliary Security Header	Frame Payload	FCS
		Addressing fields					***MAC Payload***	**MFR**
MHR								

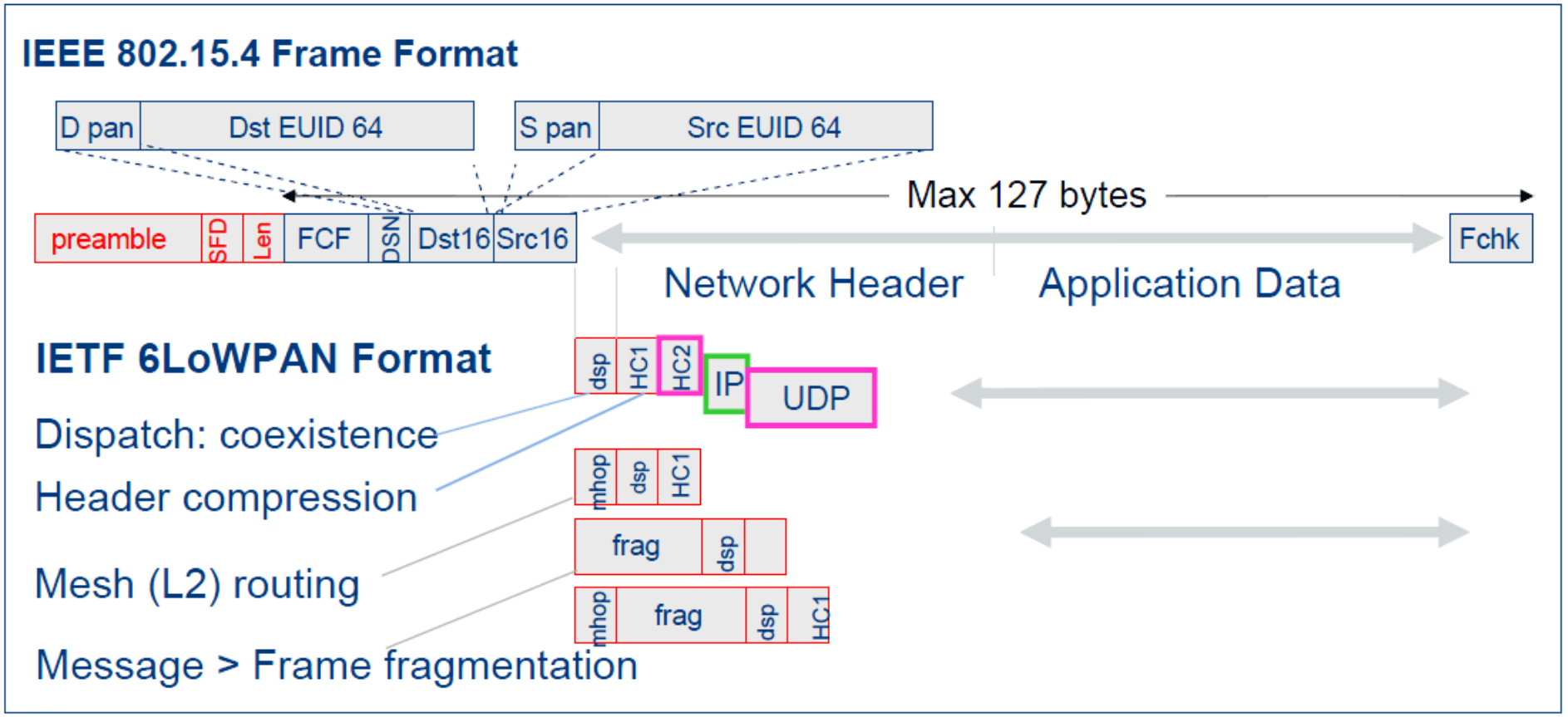


Typical 6LoWPAN Header Stacks



IPv6 Fragmentation	Multiple L2 Hops
No	No
Yes	No
Yes	Yes (Future)
No	Yes (Future)

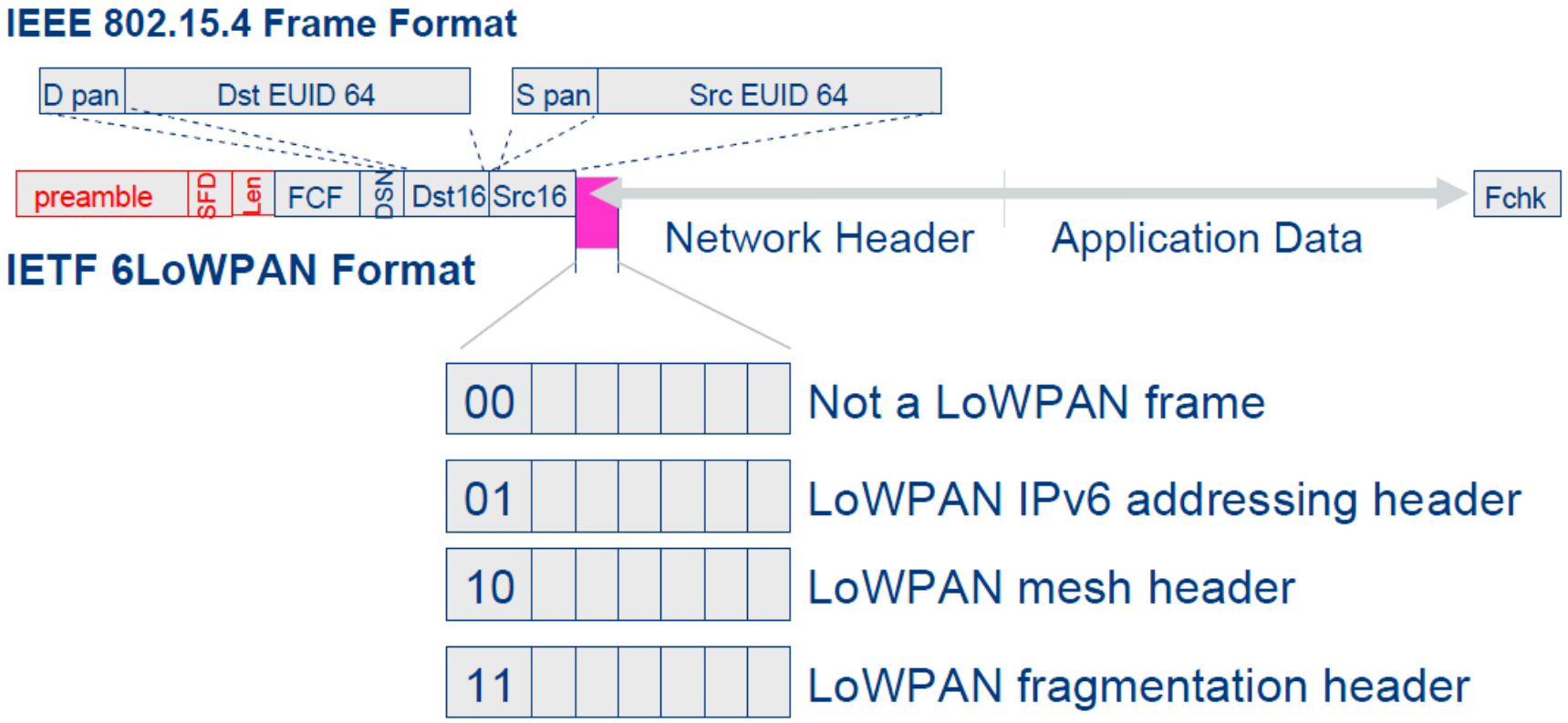
6LoWPAN Format Design



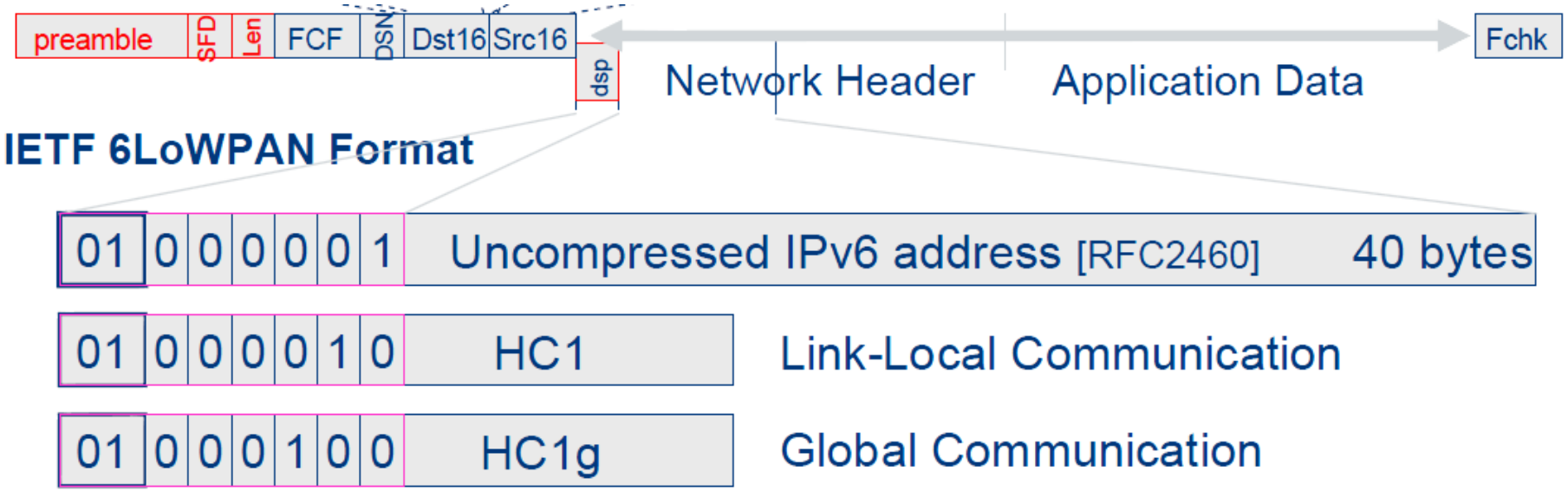
- Orthogonal stackable header format
- Almost no overhead for the ability to interoperate and scale.
- Pay for only what you use

6LoWPAN – The First Byte

- Coexistence with other network protocols over same link
- Header dispatch - understand what's coming



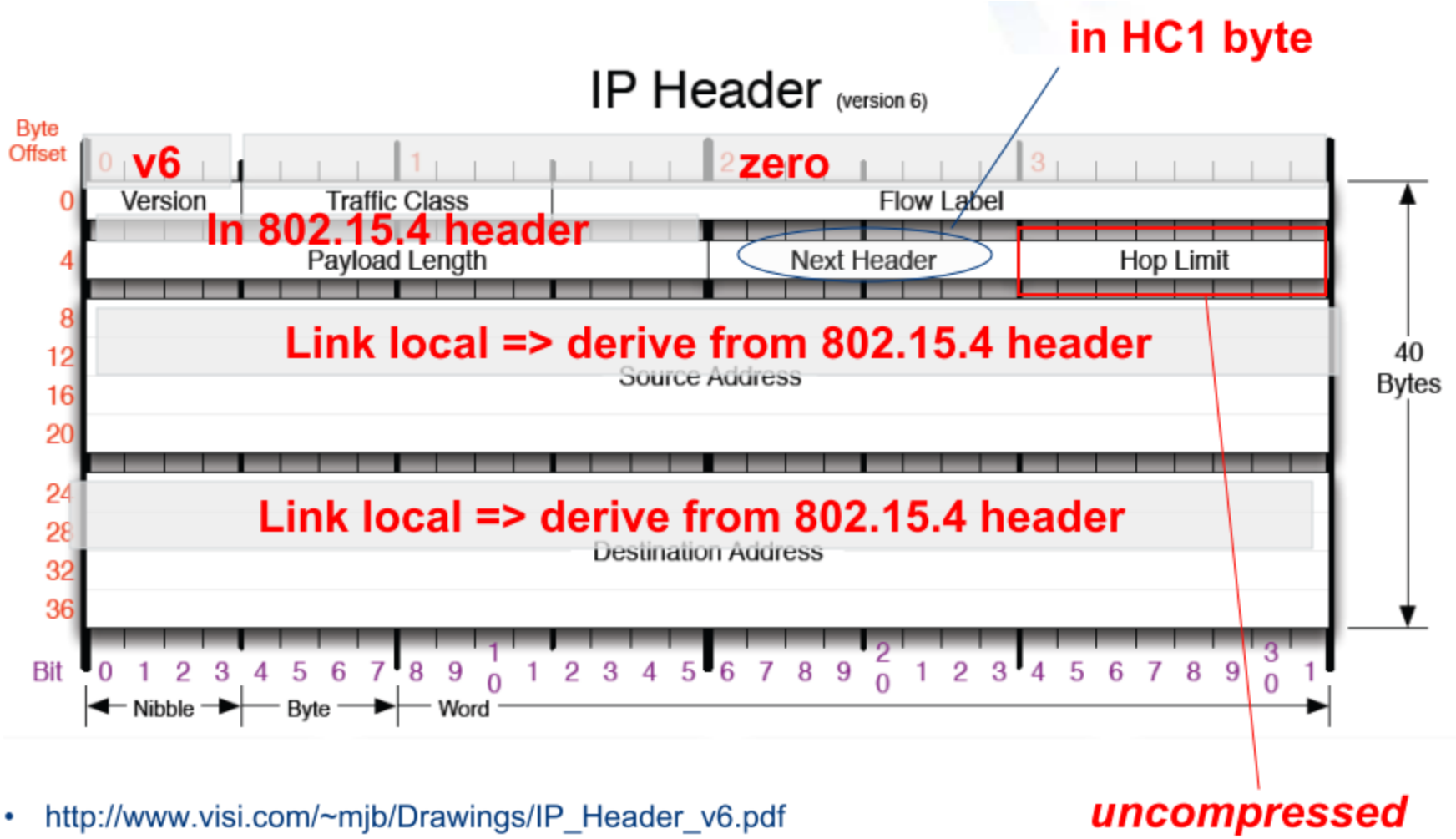
6LoWPAN – IPv6 Header



➔ *Fully compressed: 1 byte remains from uncompressed header*

Source address	: derived from link address or common prefix
Destination address	: derived from link address or common prefix
Traffic Class & Flow Label	: zero
Next header	: UDP, TCP, or ICMP
Hop Limit	: uncompressed

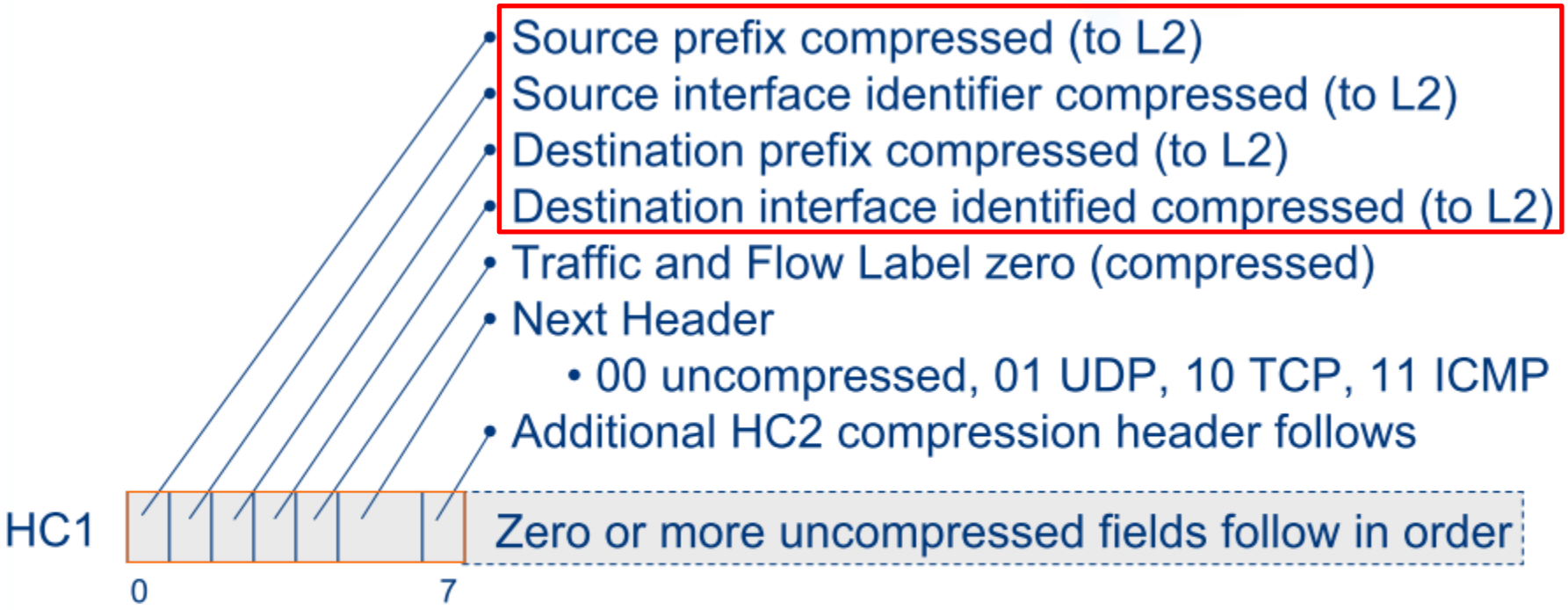
IPv6 Header Compression



- http://www.visi.com/~mjb/Drawings/IP_Header_v6.pdf

HC1 Compressed IPv6 Header [4944]

For Link-Local Communication



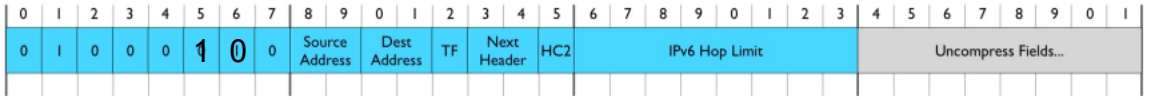
- Efficient communication with link-local IPv6 addresses
- IPv6 address <prefix64 || interface id> for nodes in 802.15.4 subnet derived from the link address.
  - PAN ID maps to a unique IPv6 prefix
  - Interface identifier generated from EUI-64 or short address
- Hop Limit is the only incompressible IPv6 header field

LOWPAN_HC1 (common compressed header encoding)

- The address fields encoded by "HC1 encoding" are interpreted as follows:
  - Source/Destination Prefix compression
    - PI(0): Prefix carried in-line (Section 10.3.1).
    - PC(1): Prefix compressed (link-local prefix assumed).
  - Source/Destination Interface ID compression
    - II(0): Interface identifier carried in-line (Section 10.3.1).
    - IC(1): Interface identifier elided (derivable from the corresponding link-layer address).

HC1g Compressed IPv6 Header[4944]

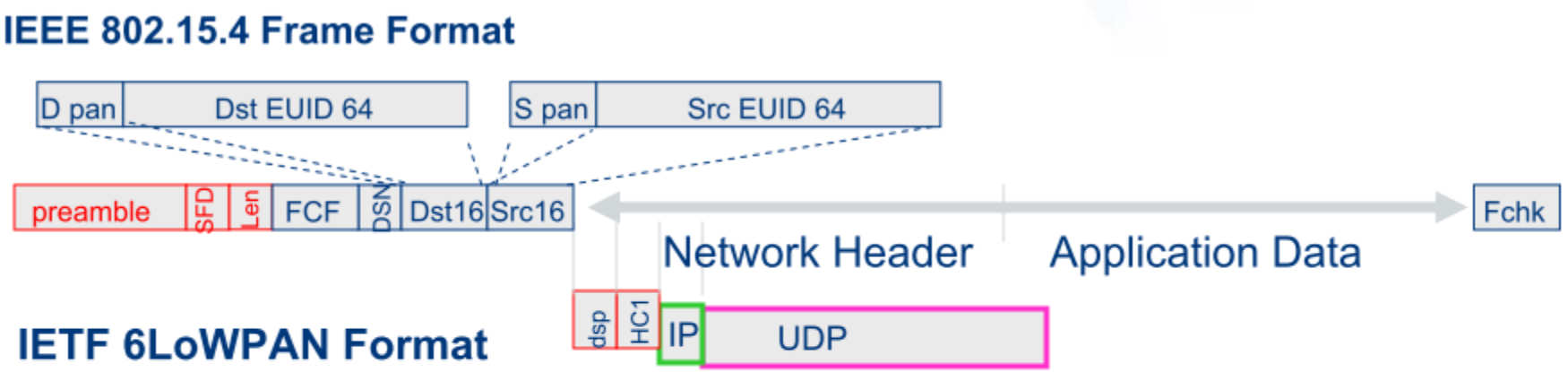
For Global Communicatior



- Source prefix compressed (common prefix assumed)
- Destination prefix compressed (common prefix assumed)
- Traffic and Flow Label zero (compressed)
- Next Header if the next header is compressed?
- Additional HC2 compression header follows

- Efficient communication with routable IPv6 addresses
- PAN ID maps to a unique IPv6 prefix
  - Common prefix derived from PAN ID
  - Interface identifier generated from EUI-64 or short address
- Hop Limit is the only incompressible IPv6 header field

6LoWPAN – Compressed / UDP or ICMP



Dispatch: Compressed IPv6

HC1: Source & Dest Local, next hdr=UDP

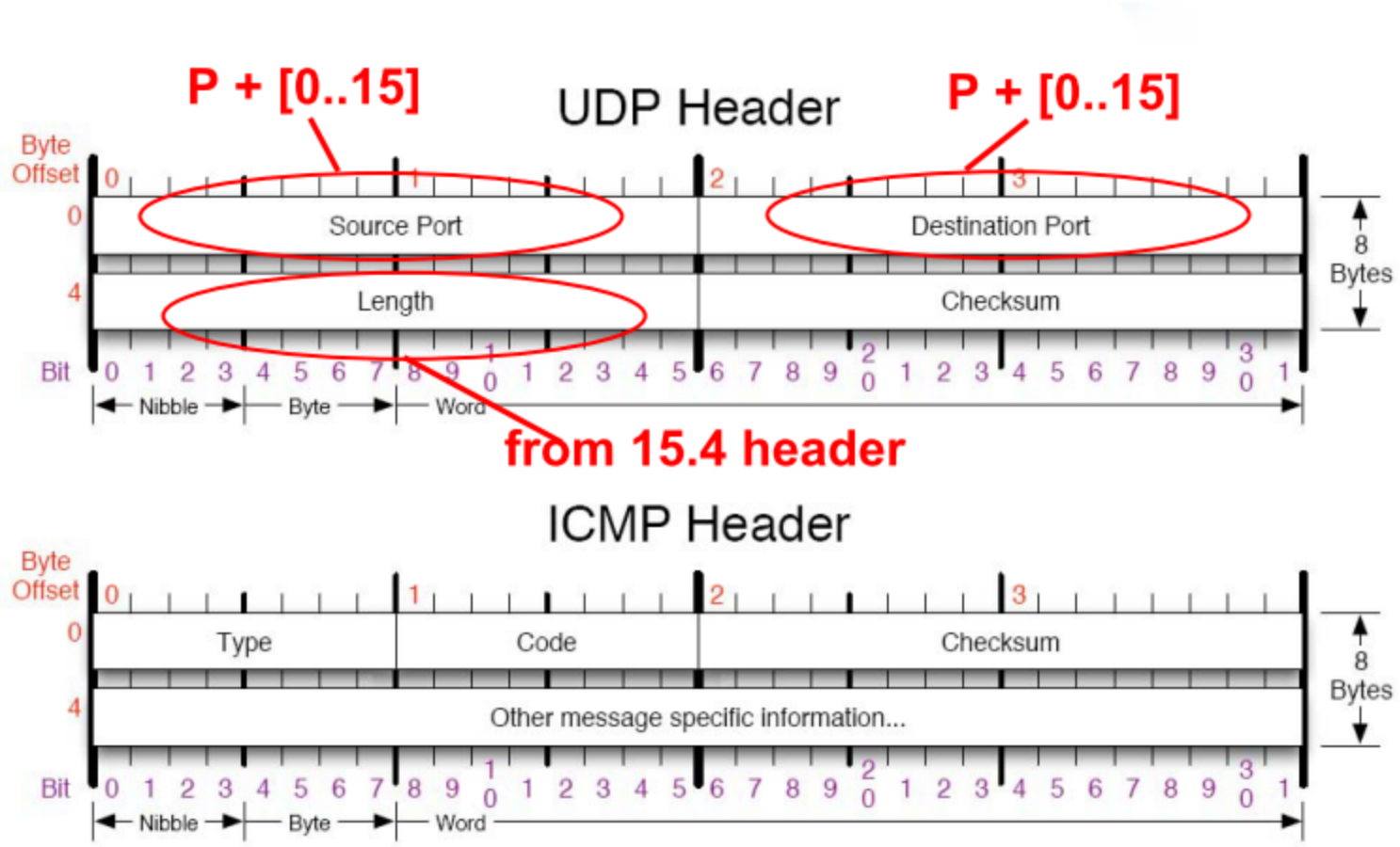
IP: Hop limit

UDP: 8-byte header (uncompressed)

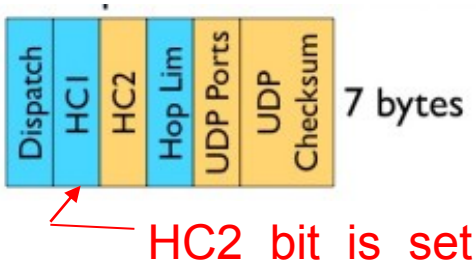
HC2 bit is not set

ICMP: 8-byte header (uncompressed)

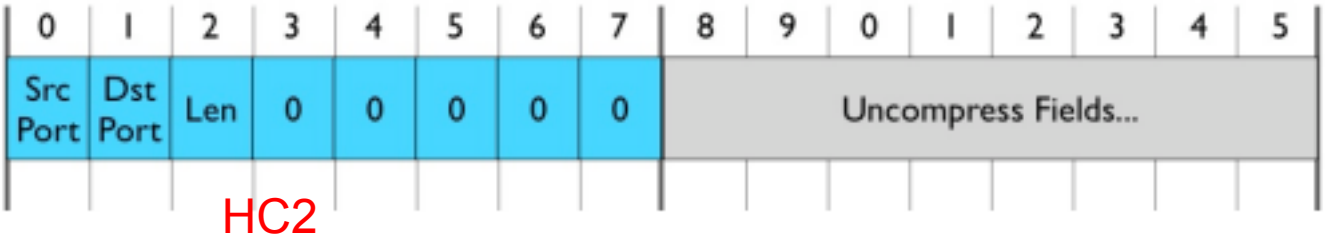
L4 – UDP/ICMP Headers (8 bytes)



6LoWPAN – Compressed / Compressed UDP

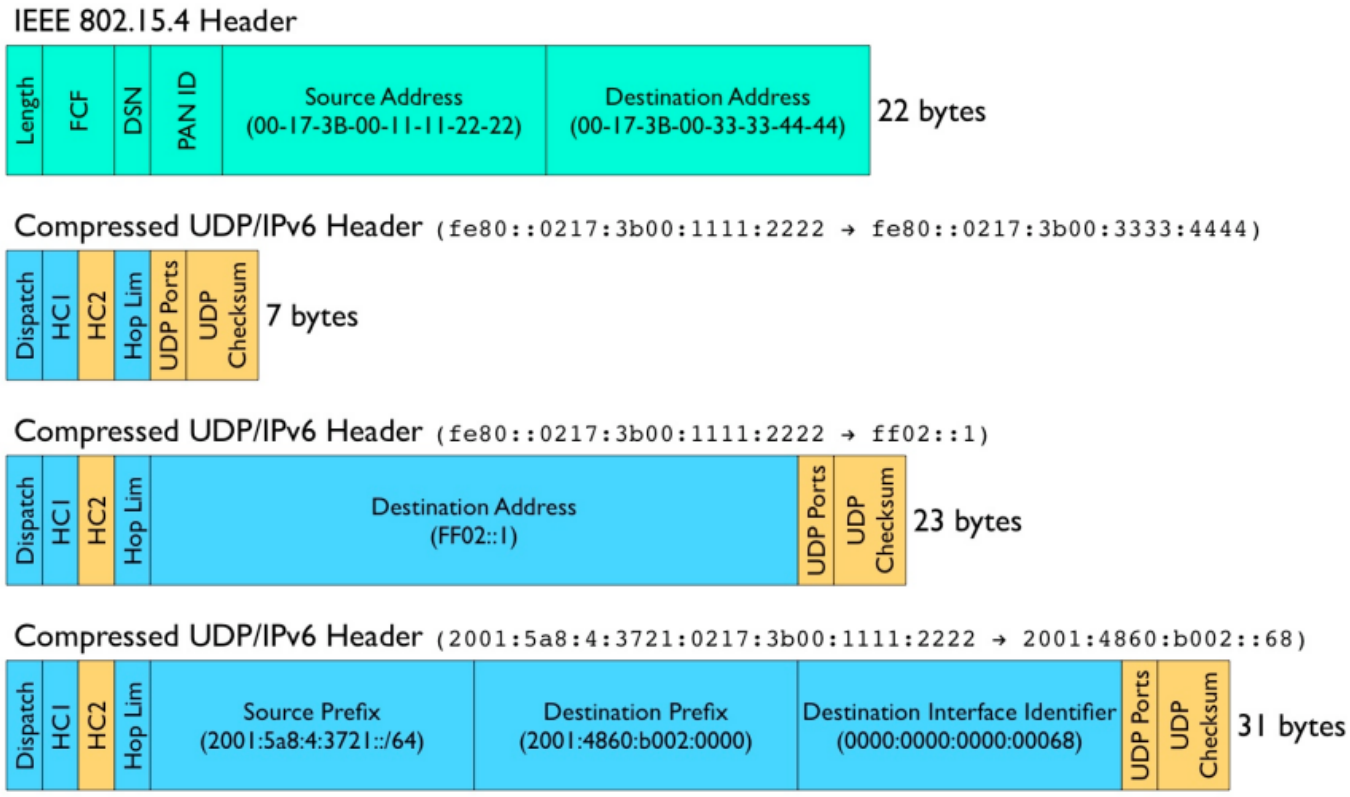


IP: 3 bytes
UDP: 4 bytes (compressed indicator, ports, checksum)

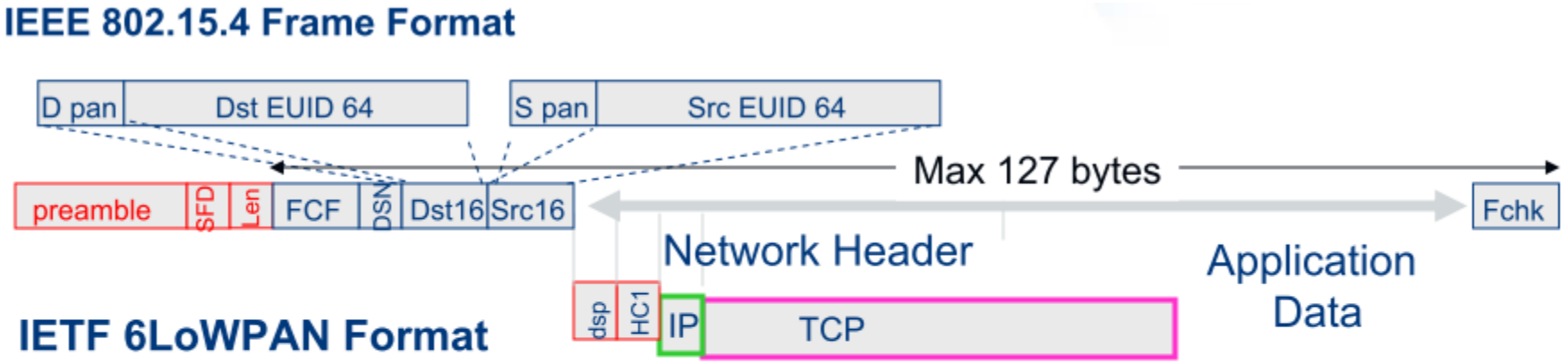


6LoWPAN introduces a range of well-known ports (61616 – 61631).
When ports fall in the well-known range, the upper 12 bits may be elided.
HC2 also allows elision of the UDP Length, as it can be derived from the IPv6 Payload Length field.

6LoWPAN IPv6/UDP Compression Examples

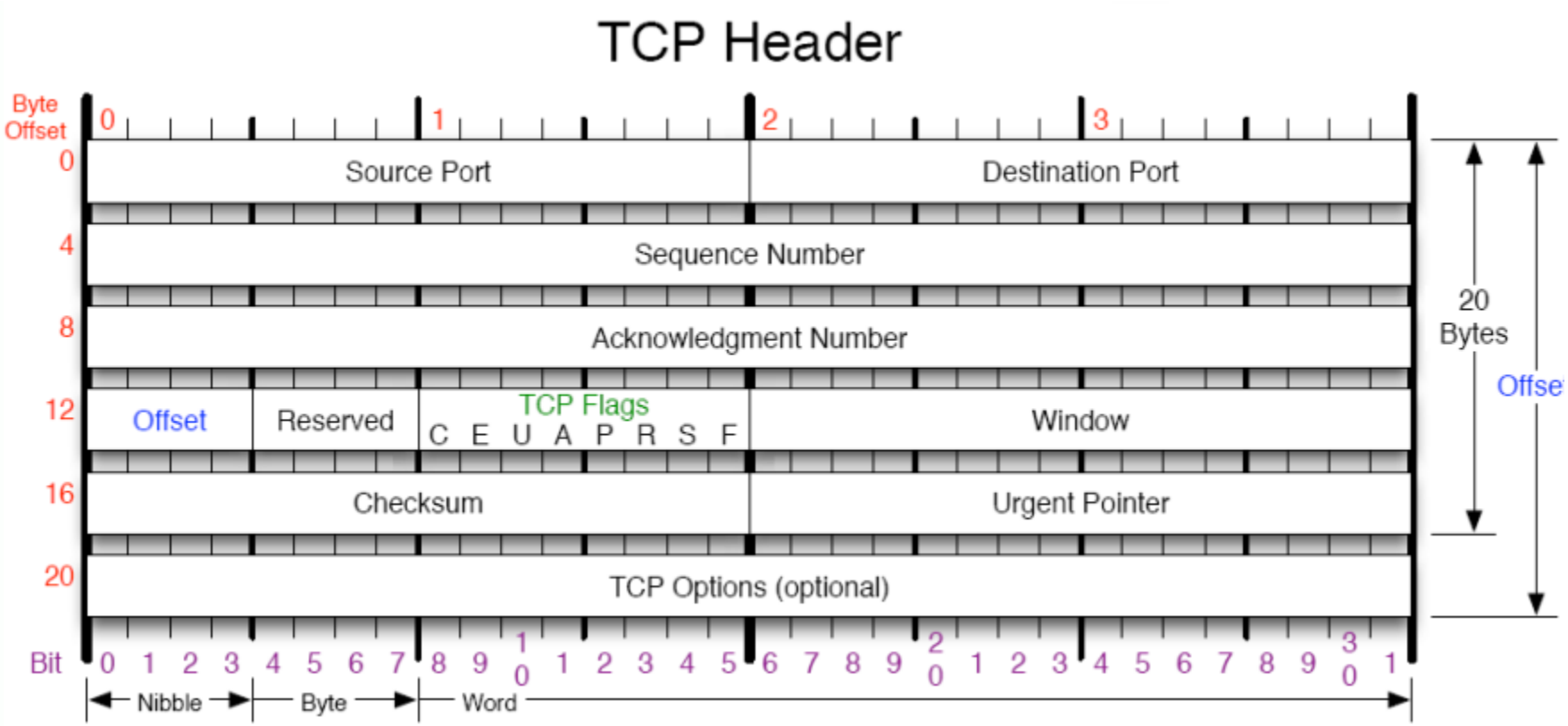


6LoWPAN – Compressed / TCP



Dispatch: Compressed IPv6
HC1: Source & Dest Local, next hdr=TCP
IP: Hops Limit
TCP: 20-byte header

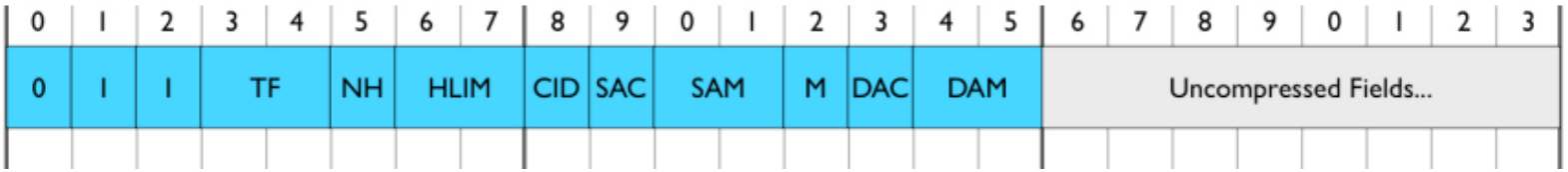
TCP Header


TCP Header
Byte Offset
0
1
2
3
Source Port
Destination Port
4
Sequence Number
8
Acknowledgment Number
12
Offset
Reserved
TCP Flags
C E U A P R S F
Window
16
Checksum
Urgent Pointer
20
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

LOWPAN_IPHC, NHC [RFC 6282]

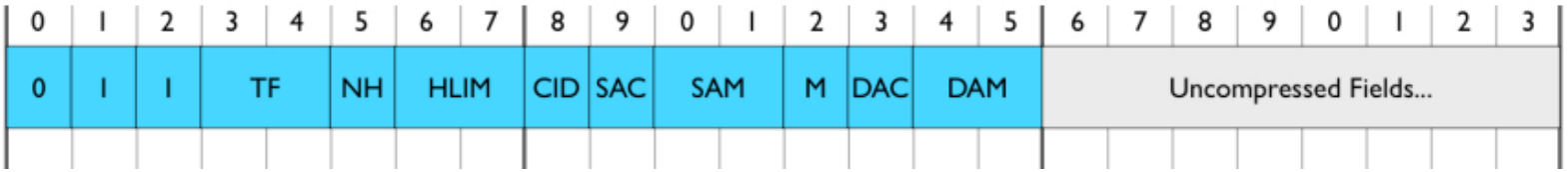
- Common case assumption
  - Version is 6
  - Traffic Class and Flow Label are both zero
  - Payload Length can be inferred from lower layers
  - Hop Limit will be set to a well-known value
  - Source addresses is formed using the link-local prefix or a small set of routable prefixes
  - Addresses assigned to 6LoWPAN interfaces are formed with an IID derived directly from either the 64-bit extended or the 16-bit short IEEE 802.15.4 addresses.

6LoWPAN Improved IPv6 Header Compression [RFC 6282]



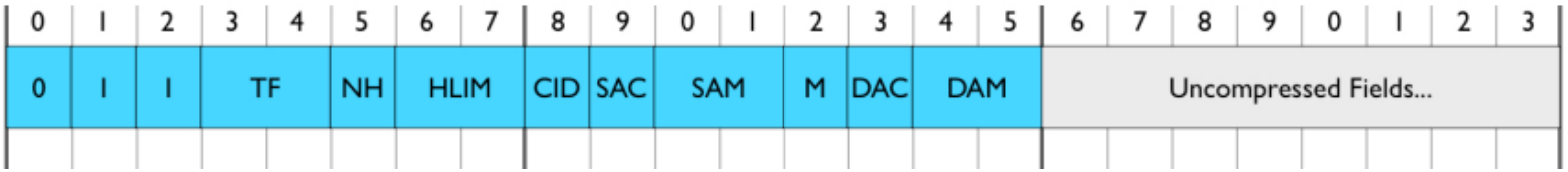
- IPHC
  - TF: Traffic Class and Flow Label to be individually compressed
    - 2-bit Explicit Congestion Notification (ECN) and 6-bit Differentiated Services Code Point (DSCP)
    - 00: ECN + DSCP + 4-bit Pad + Flow Label (4 bytes)
    - 01: ECN + 2-bit Pad + Flow Label (3 bytes), DSCP is elided.
    - 10: ECN + DSCP (1 byte), Flow Label is elided.
    - 11: Traffic Class and Flow Label are elided.
  - NH: Next Header
    - 0: Full 8 bits for Next Header are carried in-line.
    - 1: The Next Header field is compressed and the next header is encoded using LOWPAN_NHC

6LoWPAN Improved IPv6 Header Compression [RFC 6282]



- IPHC
  - HLIM: Hop Limit compression when common values
    - 00: The Hop Limit field is carried in-line.
    - 01: The Hop Limit field is compressed and the hop limit is 1.
    - 10: The Hop Limit field is compressed and the hop limit is 64.
    - 11: The Hop Limit field is compressed and the hop limit is 255.
  - Context Identifier(CID): Makes use of shared-context to elide the prefix from IPv6 addresses (two additional 4-bit fields)
    - 0: No additional 8-bit Context Identifier Extension is used (either Source Address Compression (SAC) or Destination Address Compression (DAC)).
    - 1: An additional 8-bit Context Identifier Extension field immediately follows the Destination Address Mode (DAM) field.

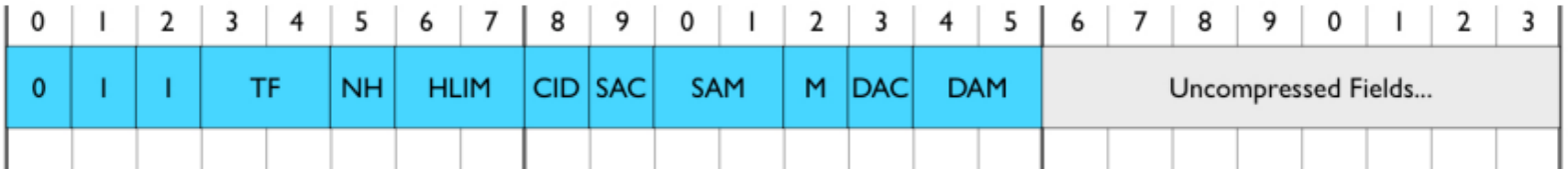
6LoWPAN Improved IPv6 Header Compression [RFC 6282]



- IPHC
  - Source Address Compression (SAC) indicates whether stateless compression is used
    - 0: Source address compression uses stateless compression.
    - 1: Source address compression uses stateful, context-based compression.
  - Source Address Mode (SAM) indicates whether the full Source Address is carried inline, upper 16 or 64-bits are elided, or the full Source Address is elided.
    - If SAC=0:
      - 00: 128 bits
      - 01: 64 bits (network prefix elided, use link-local prefix)
      - 10: 16 bits (first 112 bits elided); 0000:00ff:fe00:XXXX
      - 11: 0 bits.

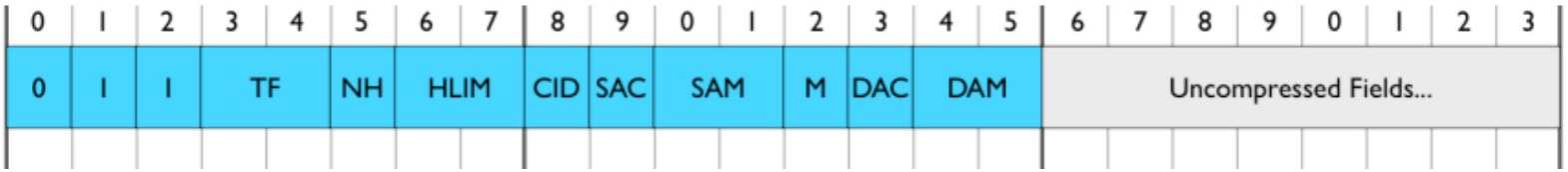
Zigbee short address

6LoWPAN Improved IPv6 Header Compression [RFC 6282]

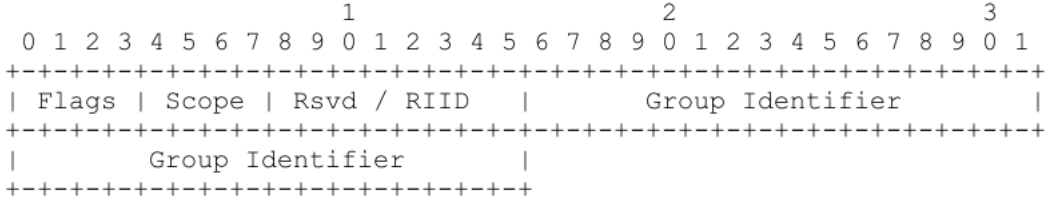


- IPHC
  - Source Address Mode (SAM)
    - If SAC=1:
      - 00: The UNSPECIFIED address, ::
      - 01: 64 bits. (other address bits are derived using context information)
      - 10: 16 bits. (using context information + 0000:00ff:fe00:XXXX)
      - 0 bits. The address is fully elided and is derived using context information and the encapsulating header (e.g., 802.15.4 or IPv6 source address)
  - M: Supports multicast addresses most often used for IPv6 ND and SLAAC.
    - 0: Destination address is not a multicast address.
    - 1: Destination address is a multicast address.

6LoWPAN Improved IPv6 Header Compression [RFC 6282]

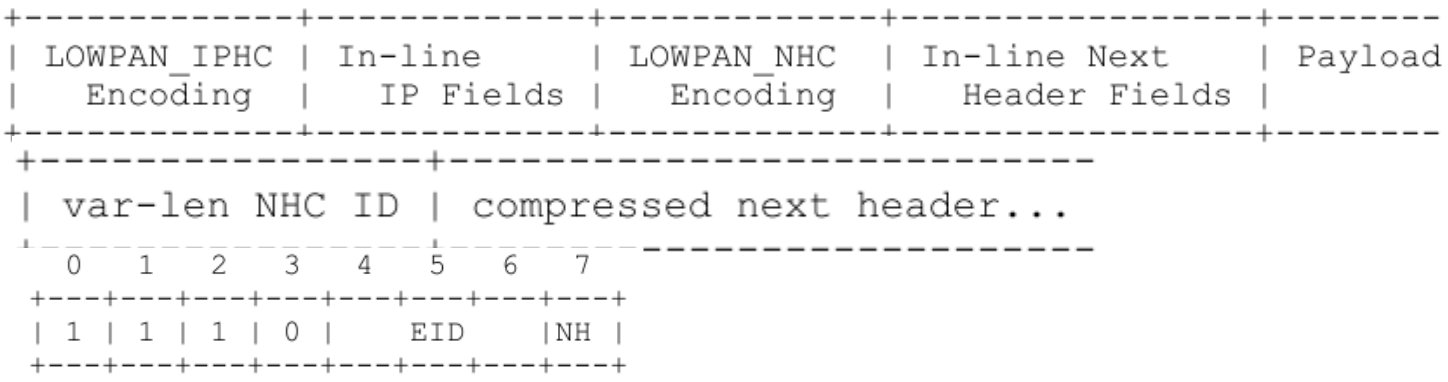


- IPHC
  - M:
    - If M=1 and DAC=0, DAM:
      - 00: 128 bits. The full address is carried in-line.
      - 01: 48 bits. The address takes the form ffXX::00XX:XXXX:XXXX.
      - 10: 32 bits. The address takes the form ffXX::00XX:XXXX.
      - 11: 8 bits. The address takes the form ff02::00XX.
    - If M=1 and DAC=1, DAM:
      - 00: 48 bits. This format is designed to match Unicast-Prefix-based IPv6 Multicast Addresses as defined in [RFC3306] and [RFC3956]. The multicast address takes the form ffXX:XXLL:PPPP:PPPP:PPPP:PPPP:XXXX:XXXX.
      - 01, 10, 11: reserved



Next Header Compression (NHC)

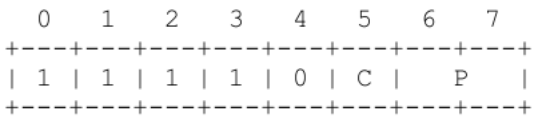
- NH=1 in IPHC indicates the use of LOWPAN_NHC



EID: IPv6 Extension Header ID:
- 0: IPv6 Hop-by-Hop Options Header [RFC2460]
- 1: IPv6 Routing Header [RFC2460]
- 2: IPv6 Fragment Header [RFC2460]
- 3: IPv6 Destination Options Header [RFC2460]
- 4: IPv6 Mobility Header [RFC6275]
- 5: Reserved
- 6: Reserved
- 7: IPv6 Header (NH must be 0; MUST be encoded using LOWPAN_IPHC)

NH: Next Header:
- 0: Full 8 bits for Next Header are carried in-line.
- 1: Next Header is elided, next header is encoded using LOWPAN_NHC

UDP NHC Format



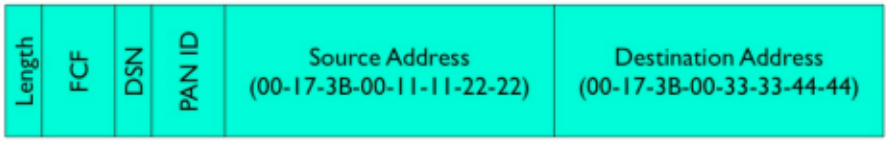
- C: Checksum:
  - 0: All 16 bits of Checksum are carried in-line.
  - 1: All 16 bits of Checksum are elided. The Checksum is recovered by recomputing it on the 6LoWPAN termination point.
- P: Ports:
  - 00: All 16 bits for both Source Port and Destination Port are carried in-line.
  - 01: All 16 bits for Source Port are carried in-line. First 8 bits of Destination Port is 0xf0 and elided. The remaining 8 bits of Destination Port are carried in-line.
  - 10: First 8 bits of Source Port are 0xf0 and elided. The remaining 8 bits of Source Port are carried in-line. All 16 bits for Destination Port are carried in-line.
  - 11: First 12 bits of both Source Port and Destination Port are 0xf0b and elided. The remaining 4 bits for each are carried in-line.

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
- A decompressor that expands a 6LoWPAN packet with the C bit set MUST compute the UDP Checksum on behalf of the source node and place that value in the restored UDP header as specified in the incumbent standards [RFC0768], [RFC2460].

Improved UDP/IPv6 Header Compression Examples

IEEE 802.15.4 Header - 22 bytes

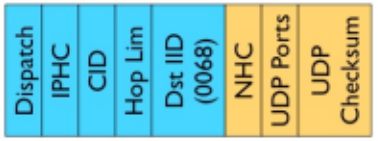


Compressed UDP/IPv6 Header (`fe80::0217:3b00:1111:2222 → fe80::0217:3b00:3333:4444`)

Traffic Class, Flow Label, Payload Length, Next Header, Hop Limit, and link-local prefixes for the IPv6 Source and Destination addresses are all elided.

Compressed UDP/IPv6 Header (`fe80::0217:3b00:1111:2222 → ff02::1`)

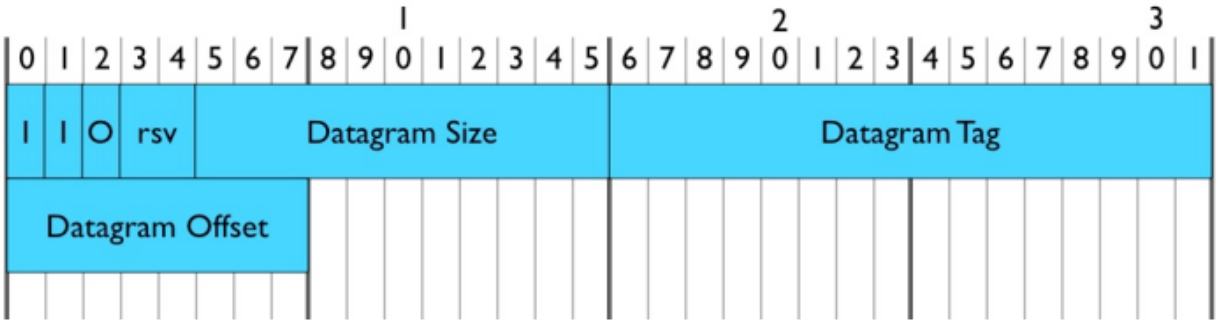
Compressed UDP/IPv6 Header (`2001:5a8:4:3721:0217:3b00:1111:2222 → 2001:4860:b002::68`)



NHC header defines a new variable length Next Header identifier, allowing for future definition of arbitrary next header compression encodings.

Fragmentation

- Datagram Size(11): total size of the unfragmented payload
- Datagram Tag(16): ID of the fragmented packet
- Datagram Offset(8): in units of 8-byte chunks



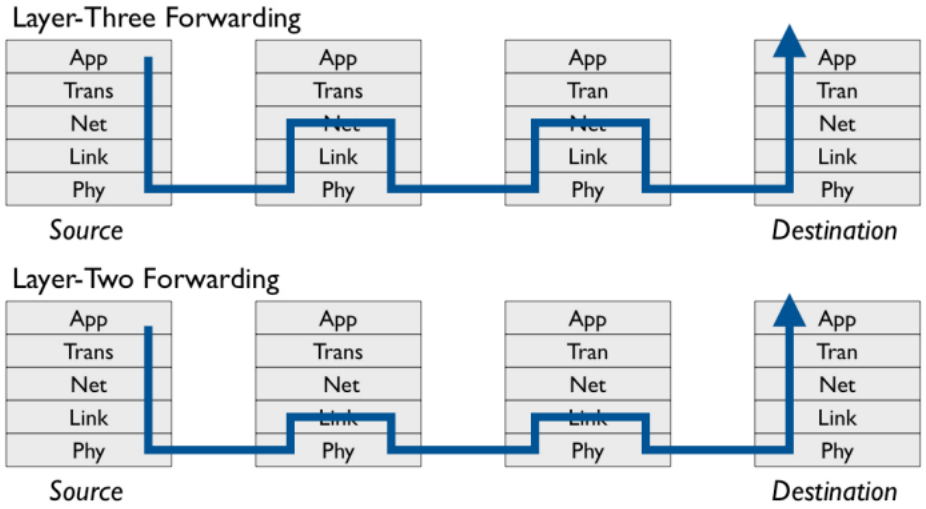
The header type is only two bits.
The third bit is used to compress the datagram offset on the first fragment as it is always zero.
The fragment header is 4 bytes for the first fragment and 5 bytes for all subsequent fragments.

Mesh Addressing Header

- Hop Limit
- Source Address, and Destination Address: IEEE 802.15.4 link addresses and may carry either a short or extended address.
  - S/D: short or full address of source/destination address

Routing Issues

- A LoWPAN network isn't typically a single broadcast domain.
  - May not able to perform some IPv6 link functions, such as ND, DAD
- Zigbee routing: Mesh Under vs. Route Over



Addressing and Autoconfiguration

- Address autoconfiguration
  - Mesh under: link-local (64 bits) scope covers an entire LoWPAN
  - Route over: a link-local address (16 bits) is sufficient to communicate with nodes in direct radio communication
- Autoconfiguration should configure interface addressing using a common prefix so that 6LoWPAN header compression can elide the prefix.

Conclusion

- 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

RPL: The IP routing protocol designed for low power and lossy networks [RFC 6550]

黃仁竑 教授

國立中正大學資工系

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

Working Items of ROLL WG

- Protocol work
  - http://datatracker.ietf.org/doc/draft-ietf-roll-rpl/
  - RPL is designed to support different LLN application requirements
    - RFC 5548 - Routing requirements for Urban LLNs
    - RFC 5673 - Routing requirements for Industrial LLNs
    - RFC 5826 - Routing requirements for Home Automation LLNs
    - RFC 5867 - Routing requirements for Building Automation LLNs
- Routing metrics
  - http://tools.ietf.org/id/draft-ietf-roll-routing-metrics/
- Security Framework
  - http://tools.ietf.org/id/draft-ietf-roll-security-framework/
- The Trickle Algorithm (RFC 6206): adjustable transmission window scheme
- Terminology
  - http://tools.ietf.org/id/draft-ietf-roll-terminology/
- Applicability statement
  - http://tools.ietf.org/id/draft-ietf-roll-applicability-ami/

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.

Functionality of RPL

- RPL expects an external mechanism to access and transport some control information, referred to as the "RPL Packet Information", in data packets.
- RPL provides a mechanism to disseminate information over the dynamically formed network topology.
  - To reduce the number of messages sent on the network, a trickle algorithm may limit the number of periodic messages that are sent. [RFC6206]

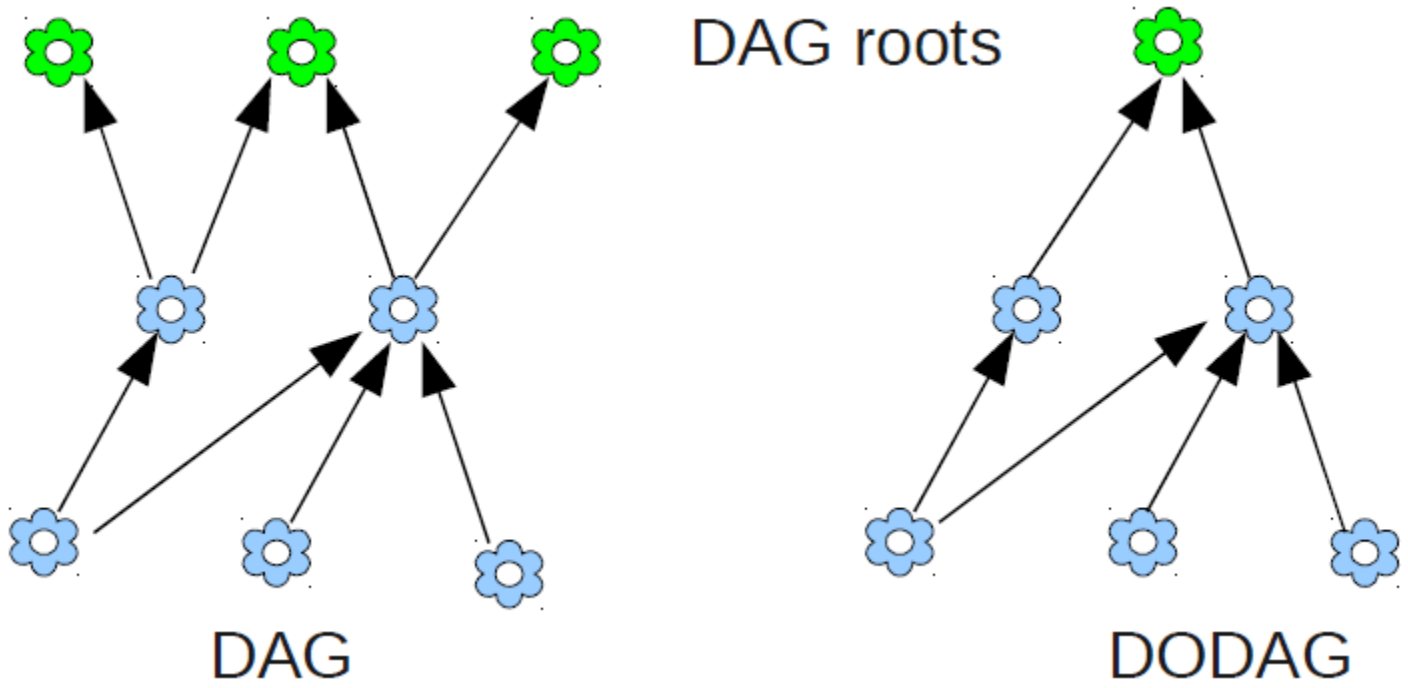
Functionality of RPL

- In some applications, RPL assembles topologies of routers that own independent prefixes.
- RPL also introduces the capability to bind a subnet together with a common prefix and to route within that subnet.
- RPL may disseminate IPv6 Neighbor Discovery (ND) information such as the [RFC4861] Prefix Information Option (PIO) and the [RFC4191] Route Information Option (RIO).

Terminology

- DAG: Directed Acyclic Graph
- DAG root: A DAG root is a node within the DAG that has no outgoing edge.
- Destination-Oriented DAG (DODAG): A DAG rooted at a single destination
- DODAG root: A DODAG root is the DAG root of a DODAG; it may act as a border router for the DODAG.

DAG and DODAG



Terminology

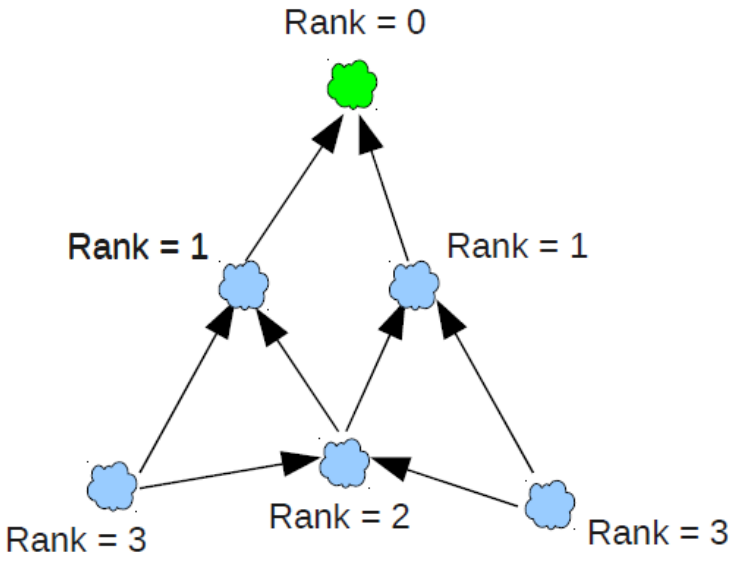
- Virtual DODAG root: A Virtual DODAG root is the result of two or more RPL routers, for instance, 6LoWPAN Border Routers (6LBRs), coordinating to synchronize DODAG state and act in concert as if they are a single DODAG root (with multiple interfaces), with respect to the LLN.

Terminology

- Up: Up refers to the direction from leaf nodes towards DODAG roots, following DODAG edges.
- Down: Down refers to the direction from DODAG roots towards leaf nodes, in the reverse direction of DODAG edges.

Terminology

- Rank: A node's Rank defines the node's individual position relative to other nodes with respect to a DODAG root. Rank strictly increases in the Down direction and strictly decreases in the Up direction. The exact way Rank is computed depends on the DAG's Objective Function (OF).



Terminology

- Objective Function (OF): An OF defines how routing metrics, optimization objectives, and related functions are used to compute Rank.
  - Currently, two objective functions are defined
    - OF0: based on hop counts (no routing metrics)
    - Minimum rank with hysteresis objective function (MRHOF)
      - The rank computation is based on metrics (e.g. link quality) contained in DIO messages.
      - MRHOF works only for additive metrics

Terminology

- Routing Metrics and constraints
  - LLN requires a sophisticated routing metric strategy driven by type of data traffic.
  - A metric is a scalar quantity used as input for best path selection.
  - A constraint, on the other hand, is used as an additional criterion to prune links or nodes that do not meet the set of constraints.
  - Metrics and constraints can be node or link based.
    - Examples of node level metrics are node state attribute, node energy state etc., while link level metrics can be latency, reliability, link color etc.

Terminology

- Objective Code Point (OCP): An OCP is an identifier that indicates which Objective Function the DODAG uses.
- RPLInstanceID: A RPLInstanceID is a unique identifier within a network. DODAGs with the same RPLInstanceID share the same Objective Function. multi-topology routing (MTR)

Terminology

- RPL Instance: A RPL Instance is a set of one or more DODAGs that share a RPLInstanceID.
- DODAGID: identifier of a DODAG root.
- DODAG Version: a specific iteration ("Version") of a DODAG with a given DODAGID
- DODAGVersionNumber: a sequential counter that is incremented by the root to form a new Version of a DODAG. A DODAG Version is identified uniquely by the (RPLInstanceID, DODAGID, DODAGVersionNumber) tuple

Terminology

- Grounded: A DODAG is grounded when the DODAG root can satisfy the Goal. (DAG's root is a border router)
- Floating: A DODAG is floating if it is not grounded. (a subDAG's root may not be a border router)
- DODAG parent: one of the immediate successors of the node on a path towards the DODAG root.
- Sub-DODAG: The sub-DODAG of a node is the set of other nodes whose paths to the DODAG root pass through that node.

Terminology

- Local DODAG: Local DODAGs contain one and only one root node, and they allow that single root node to allocate and manage a RPL Instance, identified by a local RPLInstanceID, without coordination with other nodes.
- Global DODAG: A Global DODAG uses a global RPLInstanceID that may be coordinated among several other nodes.

Terminology

- DIO: DODAG Information Object
- DAO: Destination Advertisement Object
- DIS: DODAG Information Solicitation
- CC: Consistency Check

RPLinstanceID

- Multiple concurrent instances of RPL may operate in a given network, each of them is characterized by a unique RPLinstanceID.
  - Below, we describe the behavior of an individual RPL instance.
  - A RPL instance defines Optimization Objective when forming paths towards roots based on one or more metrics
    - Metrics may include both Link properties (Reliability, Latency) and Node properties (Powered on not)

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.

Destination Oriented Direct Acyclic Graphs (DODAGs)

- Each DODAG is a directed graph with no cycles and with a single root node to optimization objectives specified by an Objective Function (OF)
  - the OF is designated by the OCP field of DIO (DODAG Information Object)
    - the OF computes the “rank” measuring the “distance” between the node and the DODAG root, and
    - also defines the parent node selection policy
  - bidirectional connectivity must be verified before accepting a router as a parent
- OFs are defined in other companion documents

Overview of DODAG Construction

- Some nodes are configured to be DODAG roots, with associated DODAG configurations.
- Nodes advertise their presence, affiliation with a DODAG, routing cost, and related metrics by sending link-local multicast DIO messages to all-RPL-nodes.
- Nodes listen for DIOs and use their information to join a new DODAG (thus, selecting DODAG parents), or to maintain an existing DODAG, according to the specified Objective Function and Rank of their neighbors.
- Nodes provision routing table entries, for the destinations specified by the DIO message, via their DODAG parents in the DODAG Version. Nodes that decide to join a DODAG can provision one or more DODAG parents as the next hop for the default route and a number of other external routes for the associated instance.
  - chooses parents that minimize path cost to the DODAG root

DODAG Construction (2nd view)

- The root starts advertising the information about the graph using the DIO message.
- The neighboring nodes of the root will receive and process DIO message potentially from multiple nodes and makes a decision based on certain rules (according to the objective function, DAG characteristics, advertised path cost and potentially local policy) whether to join the graph or not.
- Once the node has joined a graph it has a route toward the graph (DODAG) root.
- The graph root is termed as the ‘parent’ of the node.

DODAG Construction (2nd view)

- The node computes the ‘rank’ of itself within the graph, which indicates the “coordinates” of the node in the graph hierarchy.
- If configured to act as a router, it starts advertising the graph information with the new information to its neighboring peers.
- If the node is a “leaf node”, it simply joins the graph and does not send any DIO message.
- The neighboring peers will repeat this process and do parent selection, route addition and graph information advertisement using DIO messages.
- This rippling effect builds the graph edges out from the root to the leaf nodes where the process terminates.

DODAG Construction (2nd view)

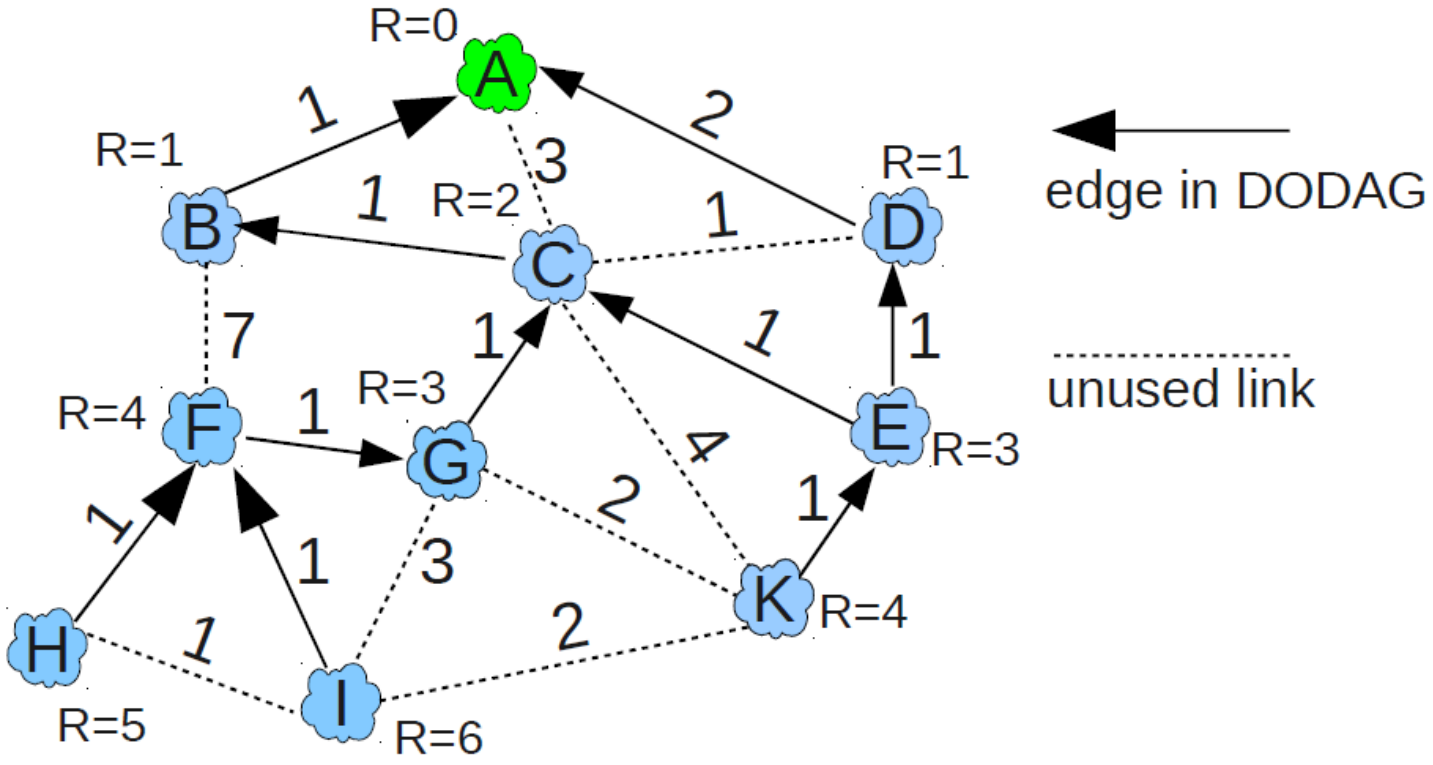
- In this formation each node has a routing entry towards its parent (or multiple parents depending on the objective function) in a hop-by-hop fashion and the leaf nodes can send a data packet all the way to root of the graph by just forwarding the packet to its immediate parent.
- This model represents a MP2P (Multipoint-to-point) forwarding model where each node of the graph has reach-ability toward the graph root. This is also referred to as UPWARD routing.

DODAG Construction (2nd view)

- Each node in the graph has a ‘rank’ that is relative and represents an increasing coordinate of the relative position of the node with respect to the root in graph topology.
- The notion of “rank” is used by RPL for various purposes including loop avoidance. The MP2P flow of traffic is called the ‘up’ direction in the DODAG.

DODAG Example

- Each node has a set of parent nodes
- A node has no knowledge about children → ONLY upward routes



RPL Identifiers

- RPLInstanceID
- DODAGID: RPLInstanceID and DODAGID uniquely identifies a single DODAG in the network
- DODAGVersionNumber: RPLInstanceID, DODAGID, and DODAGVersionNumber uniquely identifies a DODAG Version.
- Rank: defining individual node positions with respect to the DODAG root

Routing Loop Detection

- If a node receives a packet flagged as moving in the Upward direction, and if that packet records that the transmitter is of a lower (lesser) Rank than the receiving node, then the receiving node is able to conclude that the packet has not progressed in the Upward direction and that the DODAG is inconsistent.

Downward Routes

- RPL uses Destination Advertisement Object (DAO) messages to establish Downward routes. (for P2MP or P2P) Two modes:
  - Storing (fully stateful)
    - packet may be directed Down towards the destination by a common ancestor of the source and the destination
  - Non-Storing (fully source routed)
    - packet will travel all the way to a DODAG root before traveling Down.(因為只有root存routing info.)

A mixed mode of operation is not allowed.

Downward Routing

- Each RPL instance supporting download traffic selects one of the two models
  - "storing" model: nodes maintain routing tables
    - DAO message, including the prefixes and addresses reachable by the sending node, are sent to the parents.
    - Parents store the preferred downward routes and propagate aggregated DAOs upward.
  - "nonstoring" model: nodes use default routing upward and source routing downward
    - All downward traffic includes a source routing header specifying each hop along the path.
    - Intermediary routers don't store any routing information .
    - The DAG root calculate an optional hop by hop source routing path for each advertised destination. (IPv6 routing extensions)
    - Messages will be much longer.
    - P2P traffic is always routed to the DAG root.

Storing Mode

- DAO messages are used to advertise prefix reachability towards the leaf nodes in support of the ‘down’ traffic.
- DAO carrys prefix information, valid lifetime and other information about the distance of the prefix.
- As each node joins the graph it will send DAO message to its parent set. Alternately, a node or root can poll the sub-dag for DAO message through an indication in the DIO message.
- As each node receives the DAO message, it processes the prefix information and adds a routing entry in the routing table.

Storing Mode

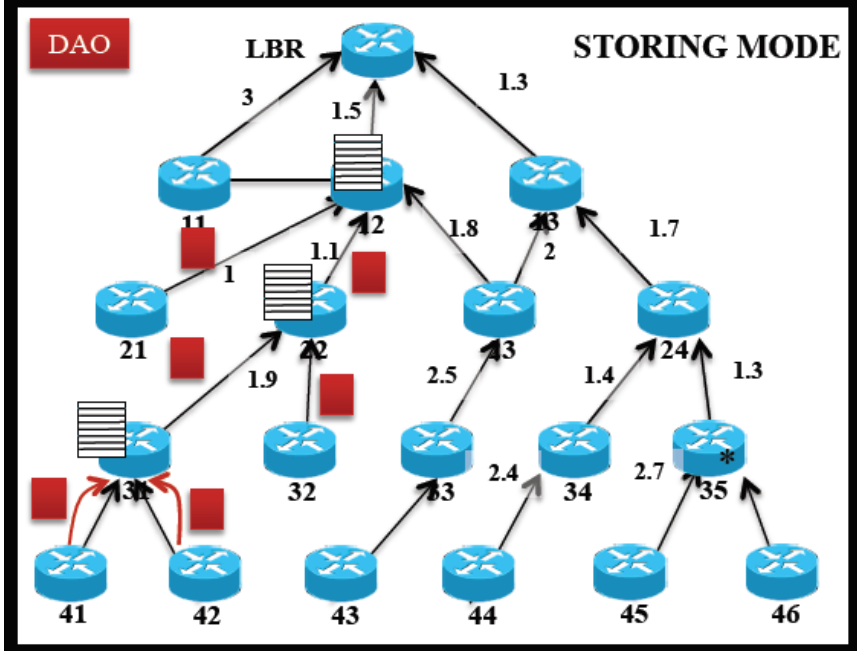
- It optionally aggregates the prefix information received from various nodes in the sub-dag and sends a DAO message to its parent set.
- This process continues until the prefix information reaches the root and a complete path to the prefix is setup.

Non-storing Mode

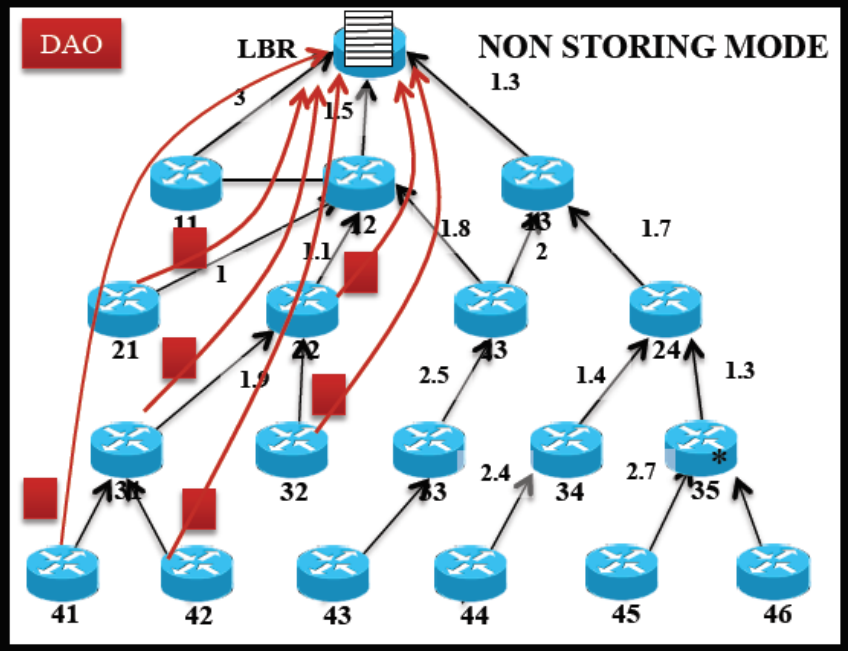
- When a node A sends a packet to a node B within the RPL domain, the packet first follows the graph up to the root where the routing information is stored.
- At this point, the graph root inspects the destination, consults its routing table that contains the path to the destination (obtained from DAO messages received).
- The root “source -routes” the packet to its destination using a specific routing header for IPv6 (called RH4).

Two modes of Operation

- Two modes of operations: storing mode and non storing modes



Unicast to DAO parents



Unicast to DODAG Root (not processed by intermediate nodes)

P2P Routing

- When a node sends a packet to another node within the LLN network, the packet travels ‘up’ to a common ancestor at which point it is forwarded in the ‘down’ direction to the destination.

Local DODAGs Route Discovery

- Optionally, a RPL network can support on-demand discovery of DODAGs to specific destinations within an LLN.

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.

Build a DODAG

- RPL nodes, starting by the DODAG root, advertise their presence, affiliation with a DODAG, routing cost, and related metrics by sending link-local multicast DIOs to the all-RPL-nodes address.
  - The DODAG root advertises predefined rank Root_Rank (= MinHopRankIncrease).
- Nodes use the received DIO information to join a new DODAG and select their parents in the DODAG according to the objective function and the rank of their neighbors.
  - Nodes may select one or more DODAG parents.
  - Two ranks r_1 and r_2 are considered the same if

$$\left\lfloor \frac{r_1}{\text{MinHopRankIncrease}} \right\rfloor = \left\lfloor \frac{r_2}{\text{MinHopRankIncrease}} \right\rfloor$$

- Expanding: After hearing DIO messages,
  - New nodes attached the DODAG by selecting parents, and then
  - Start to advertise DIO messages with the corresponding RPLInstanceID and DODAGID.

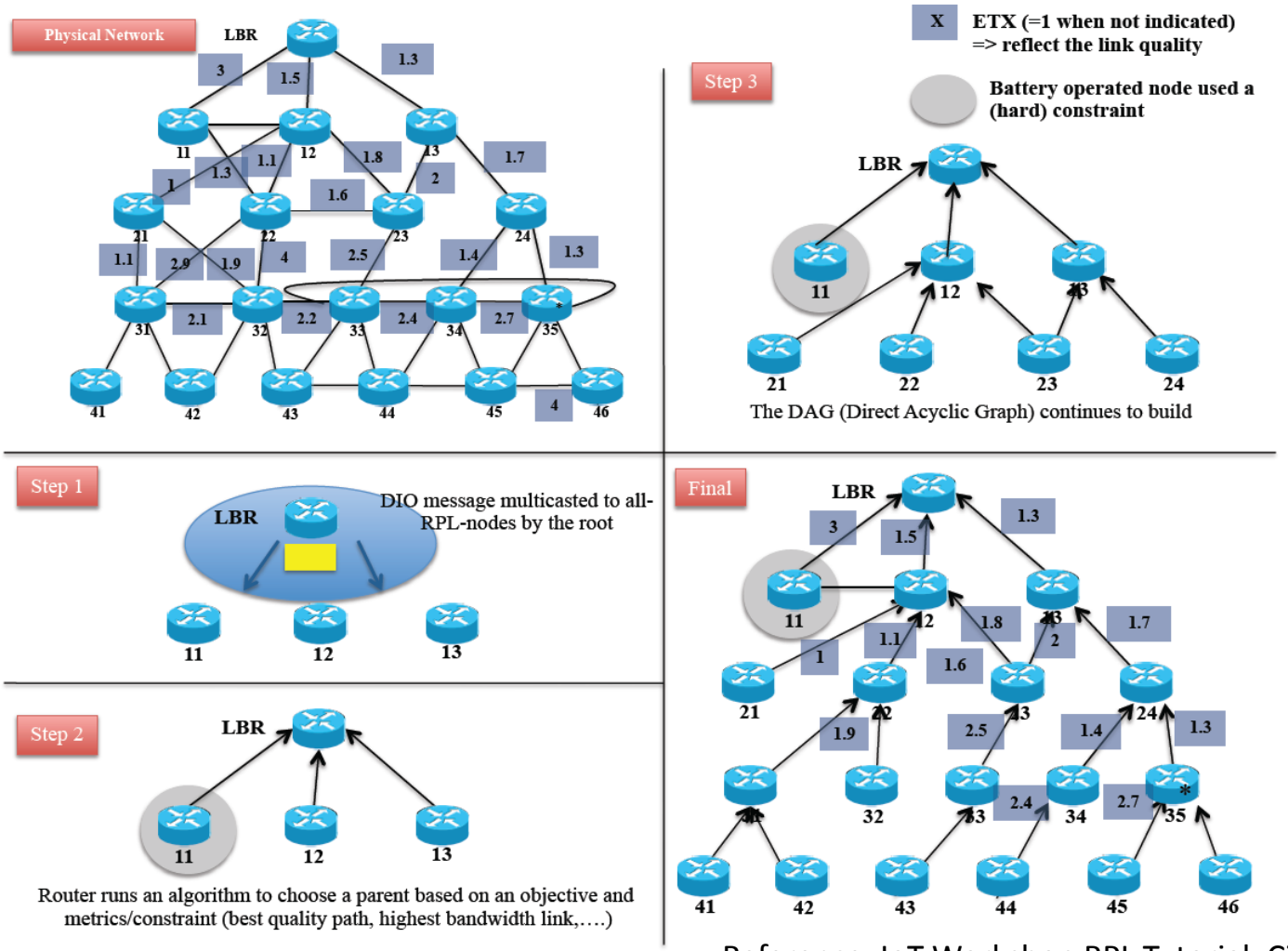
Routing Metrics in LLNs

Node Metrics	Link Metrics
Node State and Attributes Object Purpose is to reflects node workload (CPU, Memory…) "O" flag signals overload of resource "A" flag signal node can act as traffic aggregator	**Throughput Object** Currently available throughput (Bytes per second) Throughput range supported
Node Energy Object "T" flag: Node type: 0 = Mains, 1 = Battery, 2 = Scavenger "I" bit: Use node type as a constraint (include/ exclude) "E" flag: Estimated energy remaining	**Latency** Constraint - max latency allowable on path Metric - additive metric updated along path
Hop Count Object Constraint - max number of hops that can be traversed Metric - total number of hops traversed	**Link Reliability** Link Quality Level Reliability (LQL) 0=Unknown, 1=High, 2=Medium, 3=Low Expected Transmission Count (ETX) (Average number of TX to deliver a packet)
Object can be used as metric and/or constraint - metric can be additive/max/..	**Link Colour** Metric or constraint, arbitrary admin value

Specified in draft-ietf-roll-routing-metrics

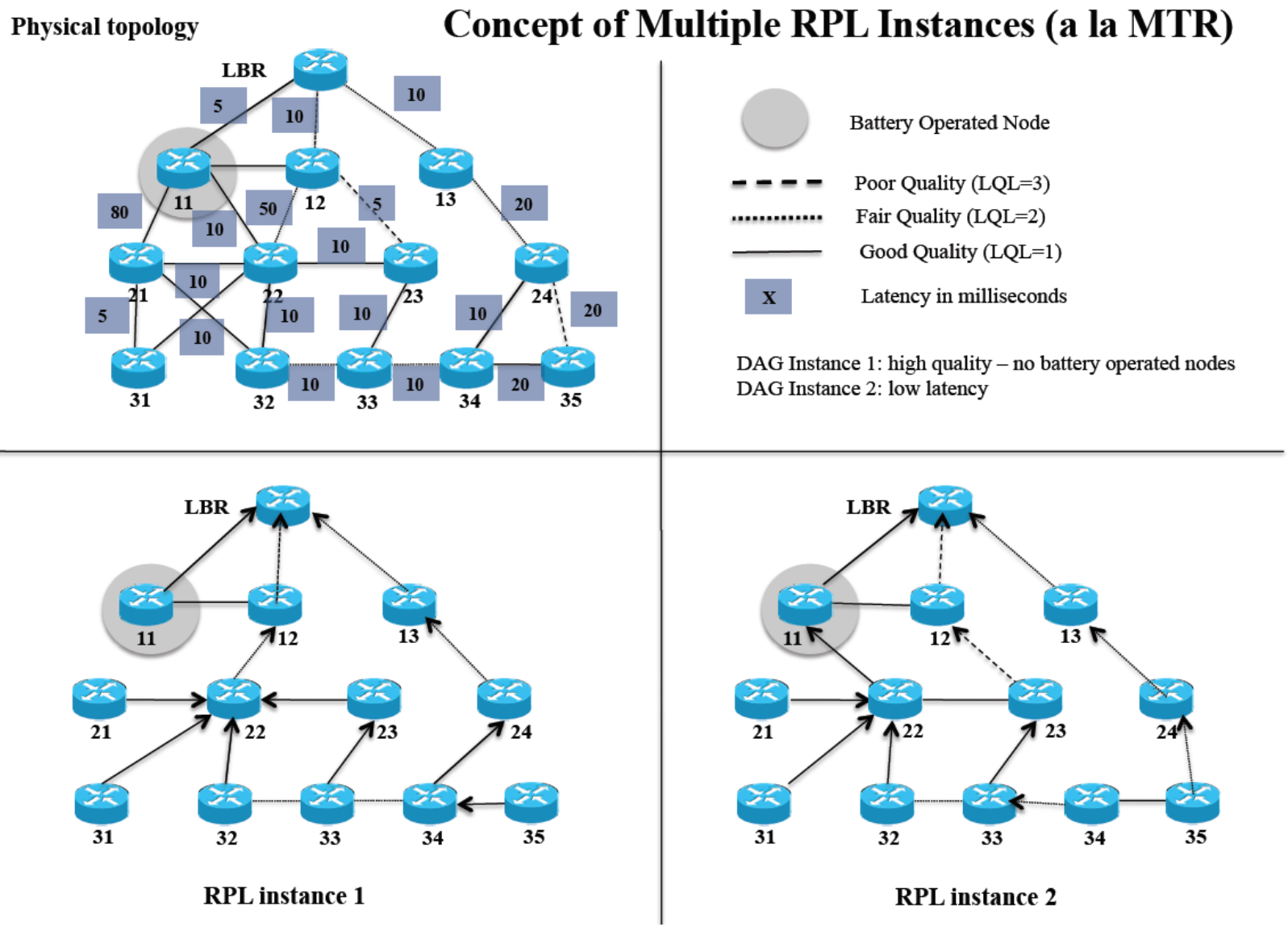
Reference: IoT Workshop RPL Tutorial, Cisco Systems

Building a DAG-Upward Routing

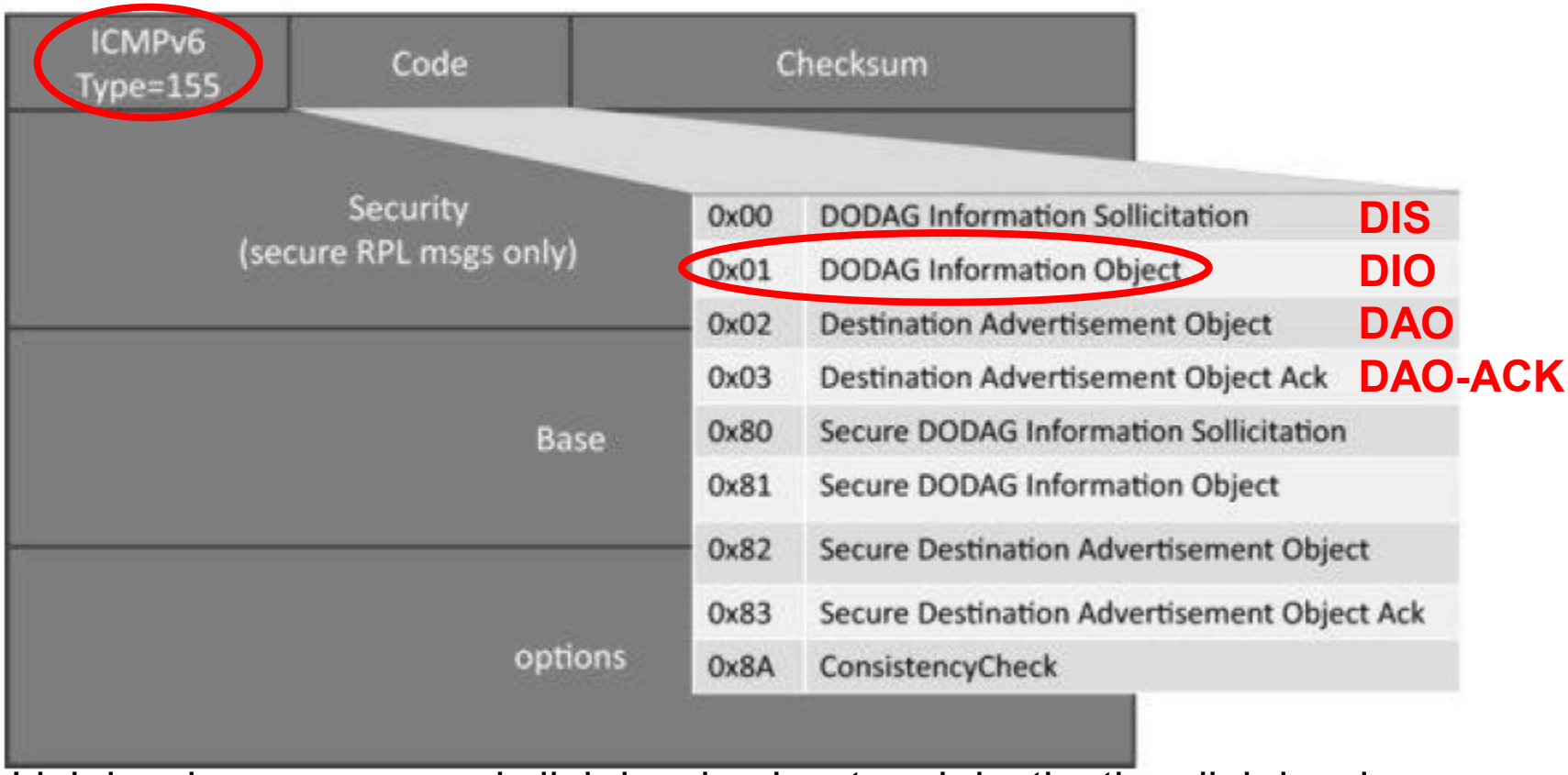


Reference: IoT Workshop RPL Tutorial, Cisco Systems

Multiple Instances of RPL



ICMPv6 RPL Control Message

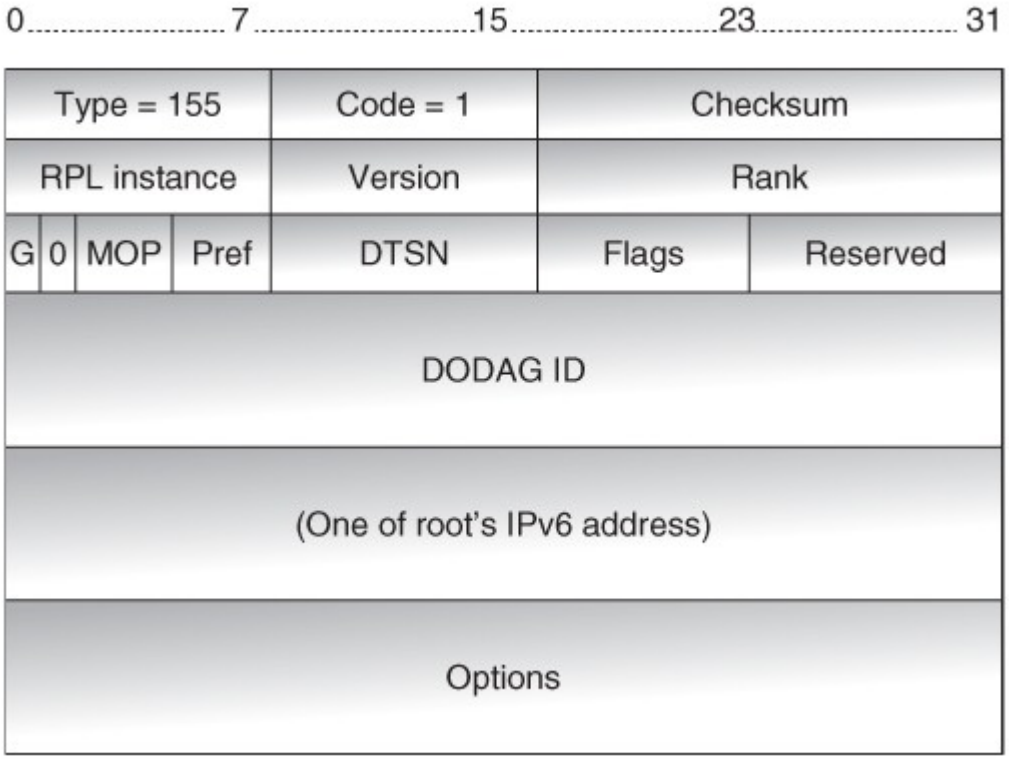


Link-local scope: source is link-local unicast and destination=link-local unicast or all-RPL-nodes(FF02::1) (for all RPL messages except DAO/ DAO-ACK in non storing mode, DIO replies to DIS)

Reference: Figure 12.7: Structure of ICMPv6 RPL control message

DODAG Information Object (DIO)

All nodes except “leaves” generate DIO periodically (controlled by Trickle).
A node uses the DIO messages received from its neighbor to determine their rank.
A node will select a set of possible parents and a preferred parent.



Trickle Timer

- RPL uses an adaptive timer mechanism called the “trickle timer”
  - The algorithm treats building of graphs as a consistency problem and makes use of trickle timers to decide when to multicast DIO messages.
  - The interval of the trickle timer increases as the network stabilizes
  - Inconsistency events: loop, join, move, etc
    - As inconsistencies are detected, the nodes reset the trickle timer and send DIOs more often.

Trickle Algorithm

- Configuration parameters
  - Imin: minimum interval size (in some unit of times)
  - Imax: maximum interval size (the base-2 log(max/min)
    - $interval = 2^{Imax} \times Imin$
  - K: redundancy constant
    - a protocol SHOULD set k and Imin such that Imin is at least two to three times as long as it takes to transmit k packets

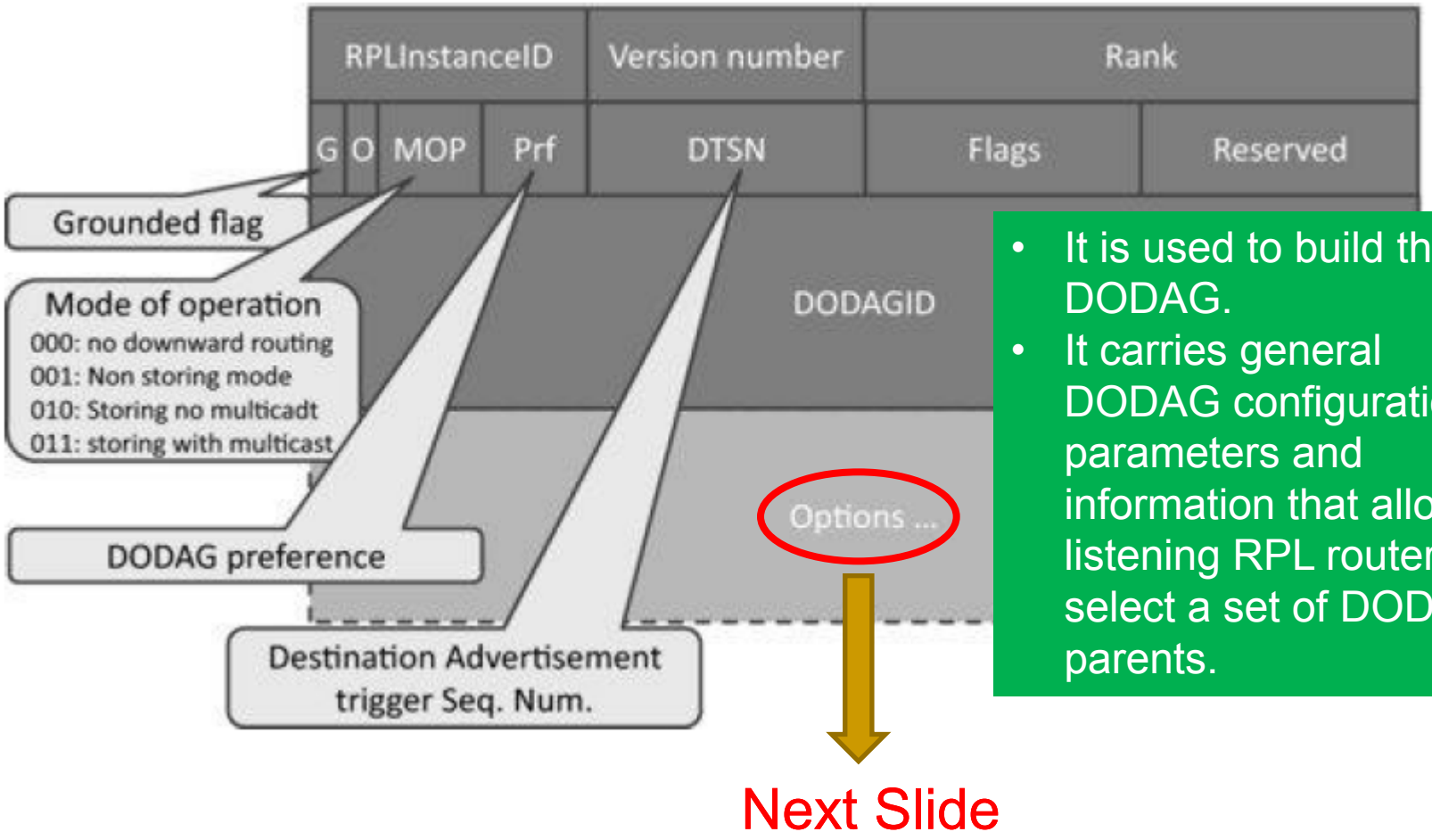
Trickle Algorithm

- Node operation parameters
  - I: the current interval size
  - t: a time within the current interval; time to send a control packet
  - c: a consistent counter

Trickle Algorithm

1. Sets I to a value in the range of [Imin, Imax]
2. Starts an interval; resets c to 0; sets t to a random number from the range [I/2, I)
3. If receives a consistent transmission, c++
4. At time t, transmits a control packet iff c<k
5. When I expires, I=Max(2xI, Imax)
6. If receives an inconsistent transmission and I>Imin, resets I=Imin, starts a new interval; resets c to 0; t=random[I/2, I)

DODAG Information Object (DIO)



- It is used to build the DODAG.
- It carries general DODAG configuration parameters and information that allows listening RPL routers to select a set of DODAG parents.

Next Slide

Reference: Figure 12.8: RPL DIO base object (followed by options)

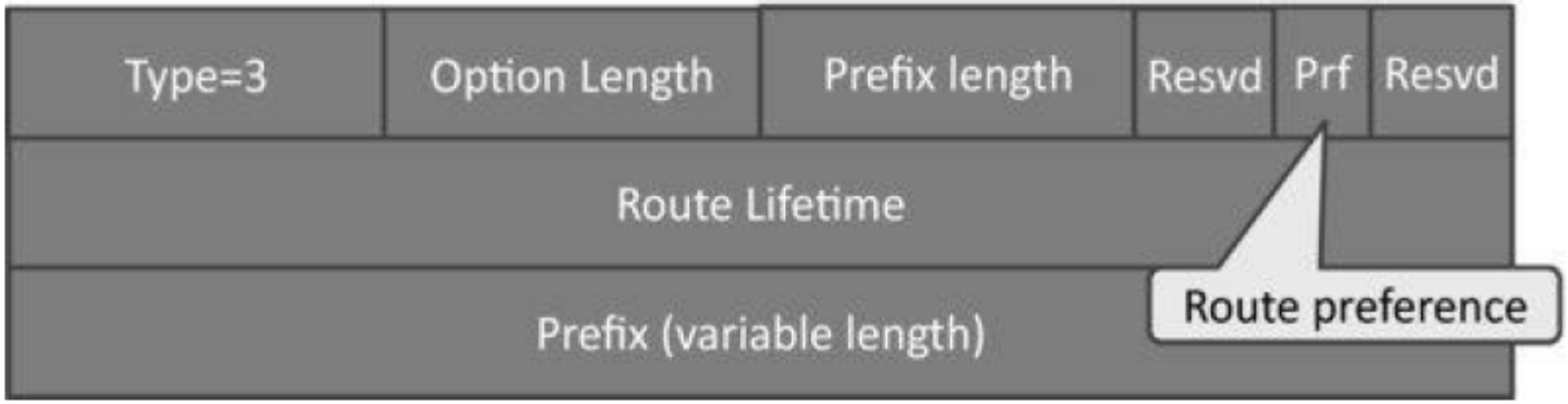
Options of RPL DIO

Option:	Type	Length	Data …

- Option types
  - 0x02: metric container option
    - Estimate the cost to reach destinations
  - 0x03: routing information option
    - Contains the same fields as the IPv6 neighbor discovery route information option
  - 0x04: DODAG information option
    - Constrain the rank a node can advertise when reattaching to a DODAG, or
    - The default lifetime of all RPL routes
  - 0x08: prefix information option
    - Contains the same fields as the IPv6 neighbor discovery prefix option

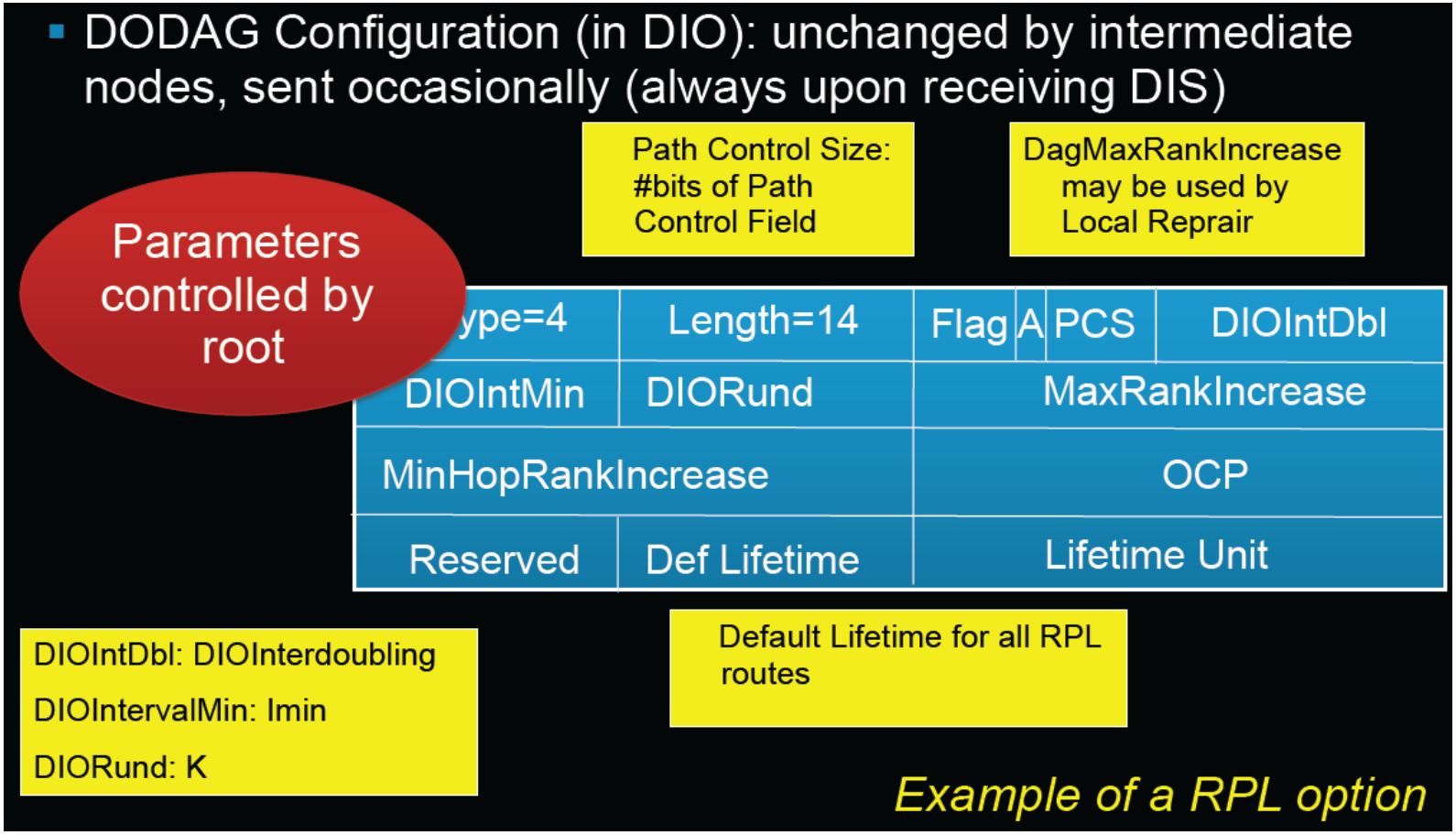
Routing Information Option (Type=3)

- RPL nodes send DIOs periodically via link-local multicasts
- Joining nodes may request DIOs from their neighbors by multicasting DIS



Reference: Figure 12.9: RPL Route Information option (Type=3)

DODAG Information Option (Type=4)



Reference: Figure 12.8: RPL DIO base object (followed by options)

RPL DIS Message

- Base Format:

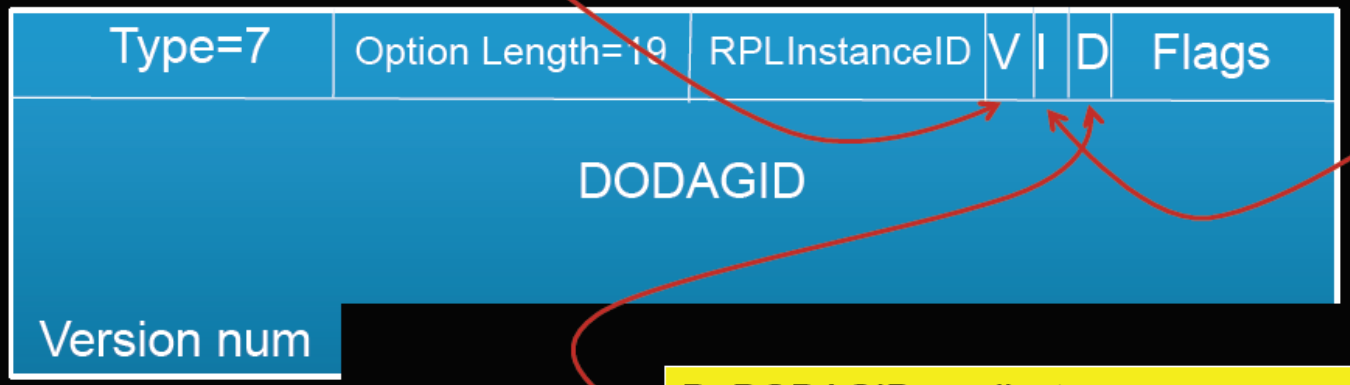
- Allows for predicate to solicit replies from subset of nodes

V: Version predicate

Receiver DOAGVersion=Version?

I: InstanceID predicate

Receiver RPLInstanceID=RPLInstanceID ?



D: DODAGID predicate

Receiver DODAGID=DODAGID?

- Used to solicit a DIO from a RPL node in the vicinity
- Ability to add filtering to the request to limit the number of replies (use of predicates)

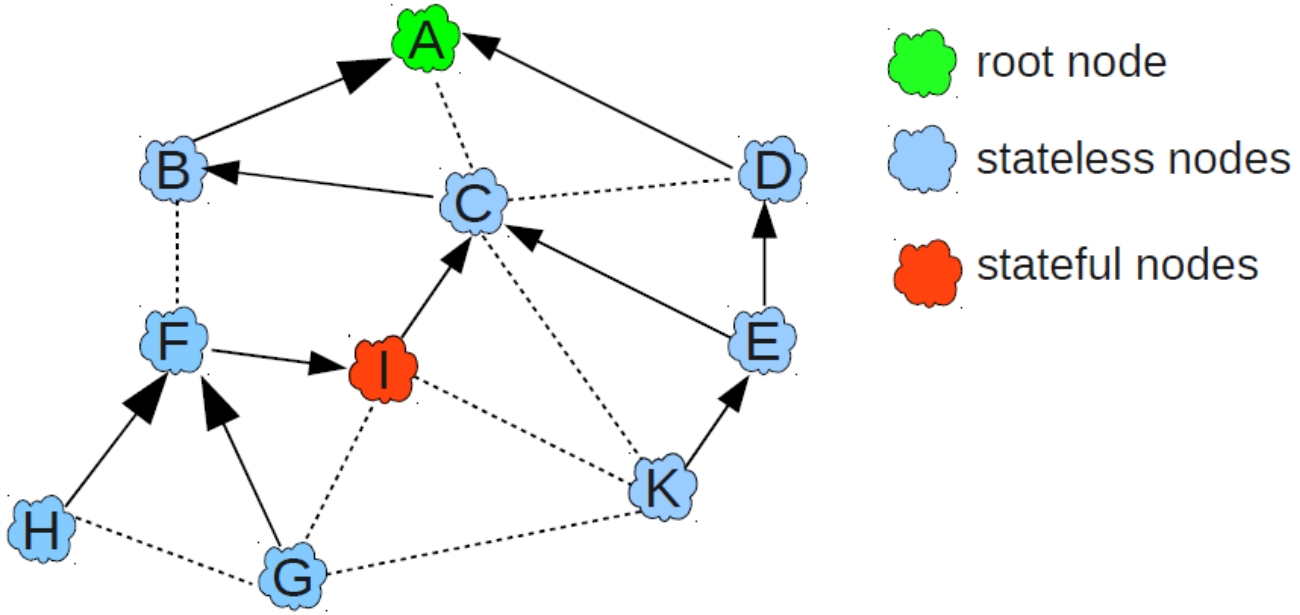
Node reset trickle timer when all predicates are true.

Downward Routes and Destination Advertisement

- Nodes inform parents of their presence and reachability to descendants by sending a DAO message
- Node capable of maintaining routing state →aggregate routes
- Node incapable of maintaining routing state → attach a next-hop address to the reverse route stack contained within the DAO message

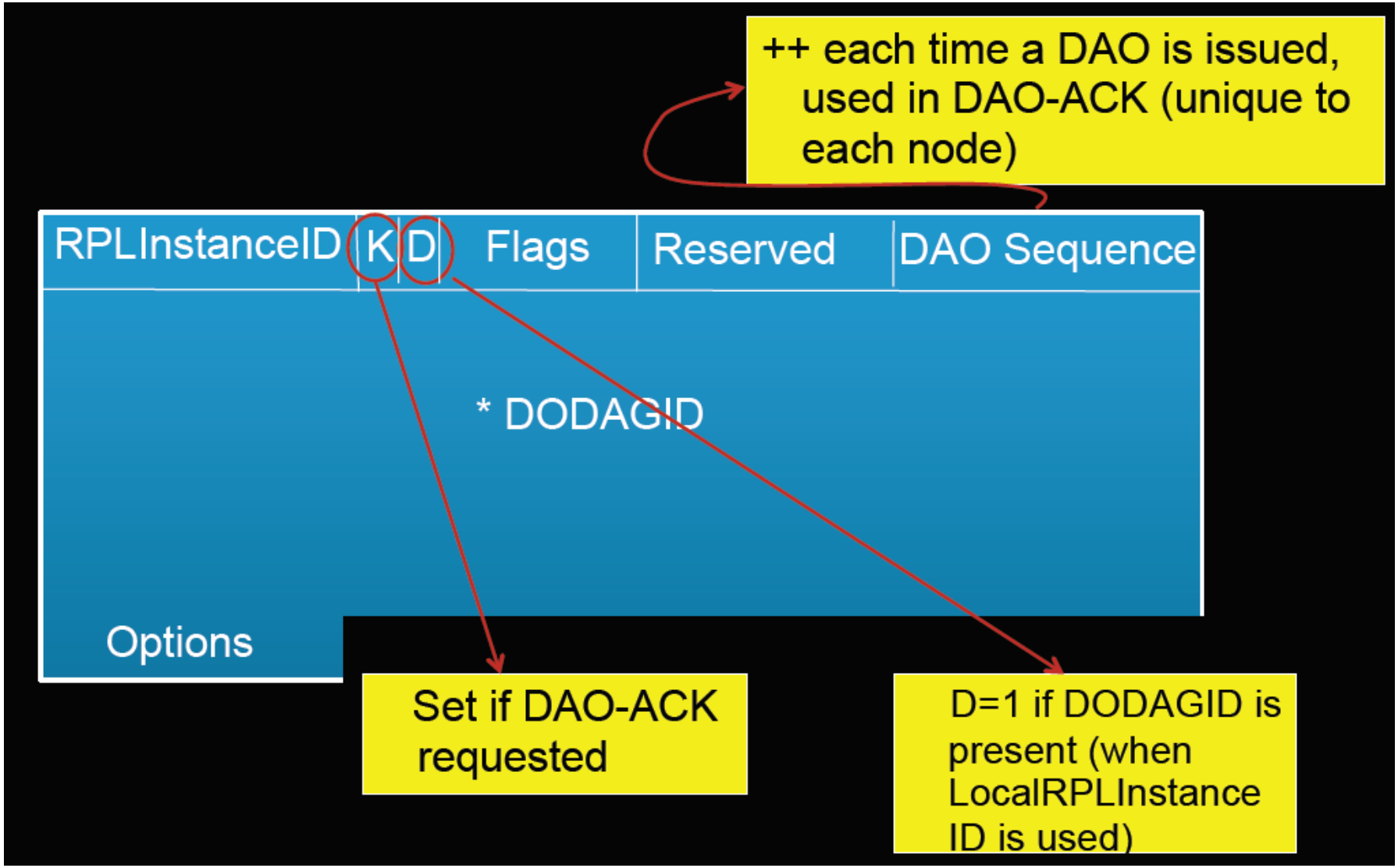
Destination Advertisement - Example

- H sends a DAO message to F indication the availability of H, F adds the next-hop and forwards the message to I
- G sends a DAO message to F indication the availability of G, F adds the next-hop and forwards the message to I
- F sends a DAO message to I indication the availability of F
- I aggregates the routes and sends a DAO advertising (F-I)



Source: Siarhei Kuryla, Networks and Distributed Systems seminar, March 2010

DAO Message


++ each time a DAO is issued, used in DAO-ACK (unique to each node)
RPLInstanceID
K
D
Flags
Reserved
DAO Sequence
* DODAGID
Options
Set if DAO-ACK requested
D=1 if DODAGID is present (when LocalRPLInstance ID is used)

RPL and 6LoWPAN

- 6LoWPAN
  - Mesh-under: single broadcast domain
- RPL
  - Route-over: places all routing functions at the network layer
    - 6LoWPAN routers operate as IPv6 routers
    - Border routers operate as RPL DODAG roots

Conclusion

- Optimized for many-to-one and one-to-many traffic patterns
- Routing state is minimized: stateless nodes have to store only instance(s) configuration parameters and a list of parent nodes
- Takes into account both link and node properties when choosing paths
- Link failures does not trigger global network re-optimization

ND in a route-over topology

- ND bootstrapping process allows hosts to attach to a LoWPAN without the need to participate in routing, thus reducing complexity.
- 6LRs (6lowpan Routers) respond to Router Solicitation (RS) messages with Router Advertisement (RA) messages.
- RAs contain the needed prefix and context information for a node to discover the LoWPAN and autoconfigure its addresses.
- In a LoWPAN, neighbor information is maintained by having nodes register with their default next-hop routers.

Route Prefix

- A LoWPAN functions properly only when its prefix information and the set of compression contexts (if any), used for further compressing addresses, is in sync for all nodes in the LoWPAN.
- In a route-over LoWPAN the link is non-transitive, thus every 6LR in the LoWPAN needs a fresh set of prefix and context information. This information is then included in the RA sent in response to an RS from a neighboring node.
- This is achieved in a LoWPAN by using the multihop prefix distribution mechanism

CoAP: Constrained Application Protocol

黃仁竑 教授
國立中正大學資工系

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.

What CoAP is (and is not)

- 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 WG Documents

draft-ietf-core-block-14	Blockwise transfers in CoAP	2013-10-21
draft-ietf-core-coap-18	Constrained Application Protocol (CoAP)	2013-06-28
draft-ietf-core-groupcomm-16	Group Communication for CoAP	2013-10-02
draft-ietf-core-observe-11	Observing Resources in CoAP	2013-10-15
RFC 6690 (draft-ietf-core-link-format)	Constrained RESTful Environments (CoRE) Link Format	2012-08

Constrained IP Networks

- A constrained IP network has limited packet sizes, may exhibit a high degree of packet loss, and may have a substantial number of devices that may be powered off at any point in time but periodically "wake up" for brief periods of time.
- These networks and the nodes within them are characterized by severe limits on throughput, available power, and particularly on the complexity that can be supported with limited code size and limited RAM per node.
- Low-Power Wireless Personal Area Networks (LoWPANs) are an example of this type of network. Constrained networks can occur as part of home and building automation, energy management, and the Internet of Things.

Source: IETF CoRE WG

Devices on Constrained Networks

- The general architecture consists of nodes on the constrained network, called Devices, that are responsible for one or more Resources that may represent sensors, actuators, combinations of values or other information.
- Devices send messages to change and query resources on other Devices.
- Devices can send notifications about changed resource values to Devices that have subscribed to receive notification about changes.
- A Device can also publish or be queried about its resources.

Source: IETF IPv6 WG

Application Scope of CoAP

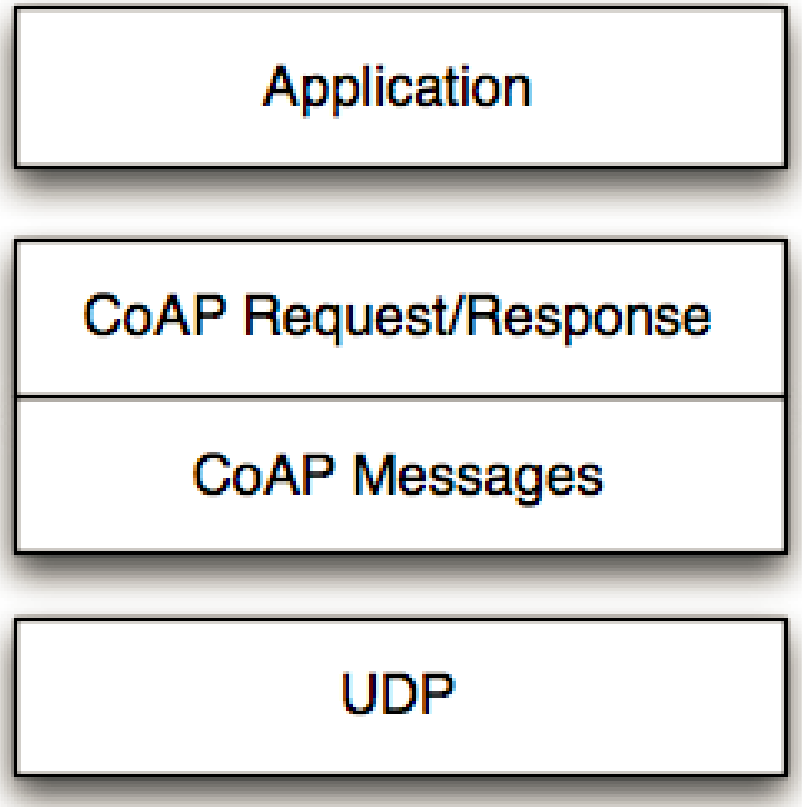
- CoAP targets the type of operating environments defined in the ROLL and 6LOWPAN working groups which have additional constraints compared to normal IP networks, but the CoAP protocol will also operate over traditional IP networks.
- This includes applications to monitor simple sensors (e.g. temperature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

Source: IETF IPv6 WG

CoAP vs. HTTP

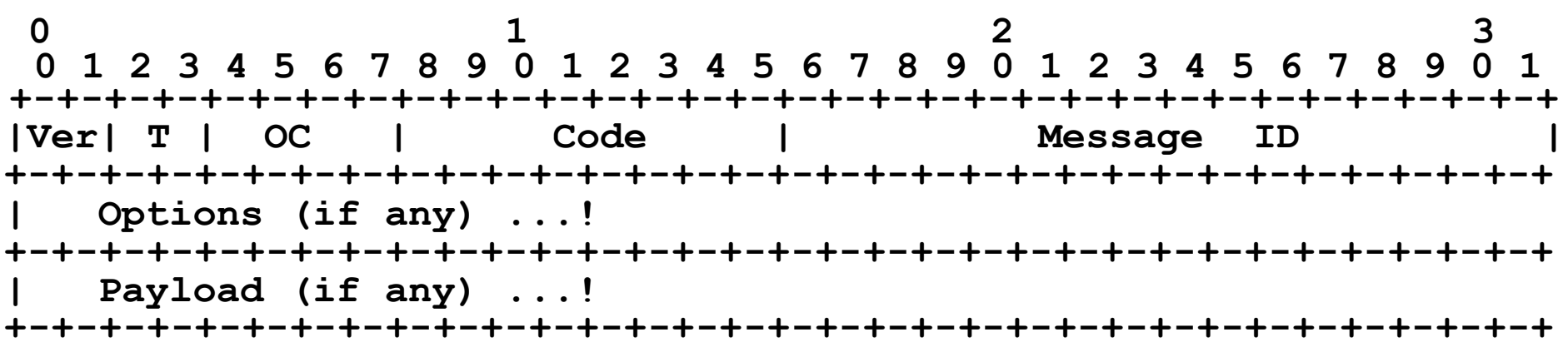
- Like HTTP, the CoAP is a way of structuring REST communications but optimized for M2M applications.
- TCP and HTTP are considered too heavy for 6LowPAN devices such as sensors. CoAP is thus based on UDP and a compressed simplified message exchange.

CoAP RESTful Applications



Source: IETF IPv6 WG

CoAP Message Format



Ver - Version (1)

T - Transaction Type

- CON – Confirmable
- NON - Non-Confirmable
- ACK – Acknowledgement
- RST - Reset

OC - Option Count, number of options after this header

Code - Request Method (1-10) or Response Code (40-255)

Message ID - Identifier for matching responses

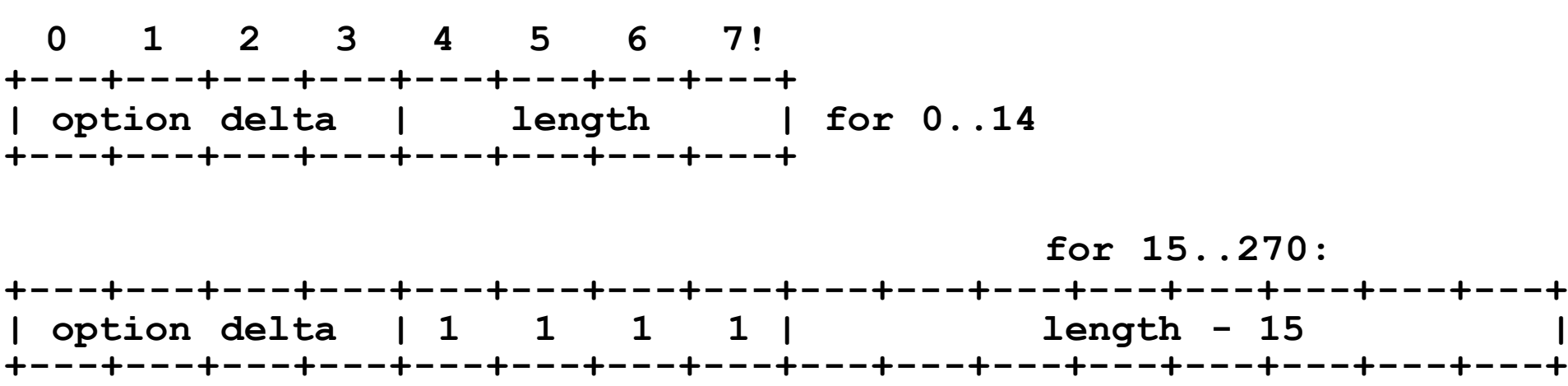
CoAP Code and Message ID

- Code: compressed from HTTP text representation (3 numbers) into one byte
  - HTTP requests =>first 3 bits 000; next five bits 0~32 (1: GET; 2: POST; 3:PUT; 4:DELETE etc.)
  - HTTP responses=>first 3 bits 001-101 (1~5) representing the first number of 1xx: informational, 2xx: success, 3xx: redirection, 4xx: client error, 5xx: server error; xx represented by next five bits 00001~01111 (1~15 used only; e.g. with HTTP response 201 is represented as 010-00001; HTTP response 400 is represented as 100-00000 etc.)
- Message ID: used in the acknowledgment process to tie a request with a response.

CoAP Option Count (OC)

- OC is a 4-bit field specifying the number of options.
- Options are stored as TLV (Type Length Value).
- Options are always sent in the same numerical order based on the type and the type is encoded as an *option delta*.
- For examples, if a message contains options types 1, 5, 6, 7 and 11, the option types sent will be 1, 4, 1, 1 and 4.

CoAP Options



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

Examples of Option types

No.	C	U	N	R	Name	Format	Length	Default
1	x			x	If-Match	opaque	0-8	(none)
3	x	x			Uri-Host	string	1-255	(see below)
4				x	ETag	opaque	1-8	(none)
5	x				If-None-Match	empty	0	(none)
7	x	x			Uri-Port	uint	0-2	(see below)
8				x	Location-Path	string	0-255	(none)
11	x	x		x	Uri-Path	string	0-255	(none)
12					Content-Format	uint	0-2	(none)
14		x			Max-Age	uint	0-4	60
15	x	x		x	Uri-Query	string	1-255	(none)
16				x	Accept	uint	0-2	(none)
19	x	x			Token	opaque	1-8	(empty)
20				x	Location-Query	string	0-255	(none)
35	x	x			Proxy-Uri	string	1-1034	(none)

4 (Etag) entity tag: proxy can assign entity tags to responses it sends to a client

14 (Max-Age) gives the maximum duration in seconds for which the answer may be cached.

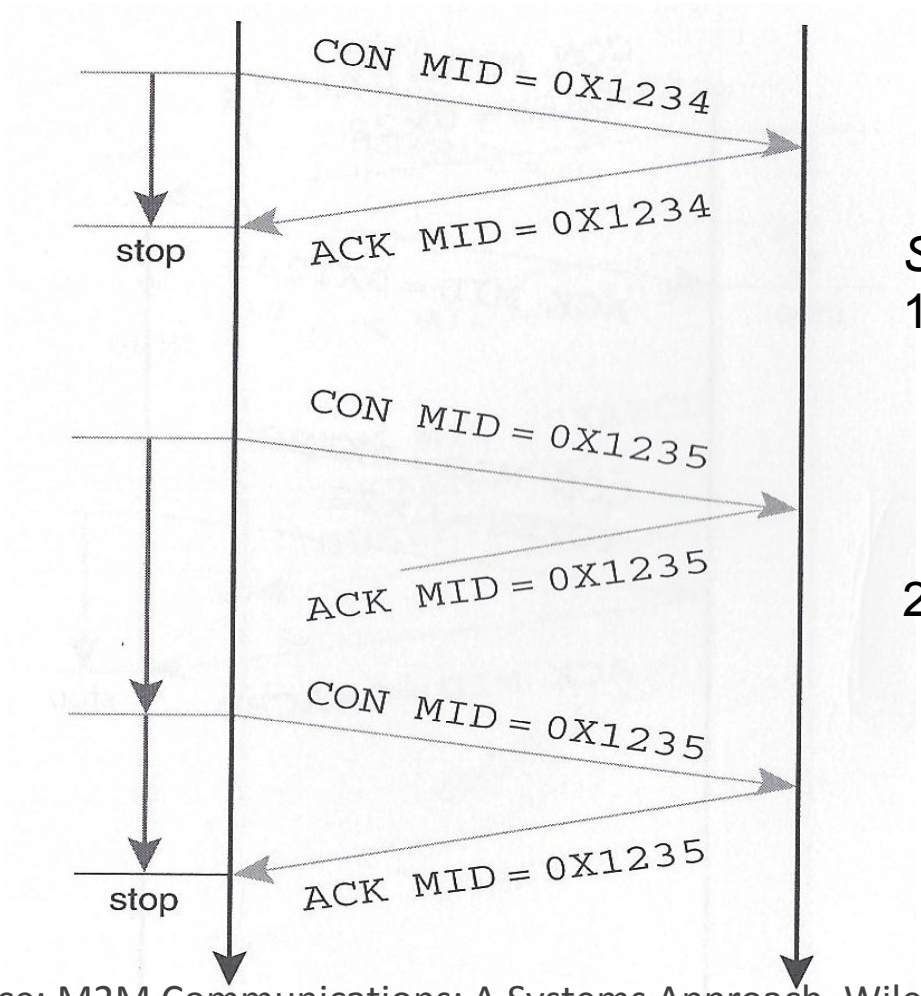
19 (Token) is used to match a response with a request.

6 (Observe) is used to receive regularly updated values from the server.

23, 27 (Block2, Block1) is used to transfer blocks of responses

C=Critical, U=Unsafe, N=NoCacheKey, R=Repeatable

Example 1 of CoAP Requests



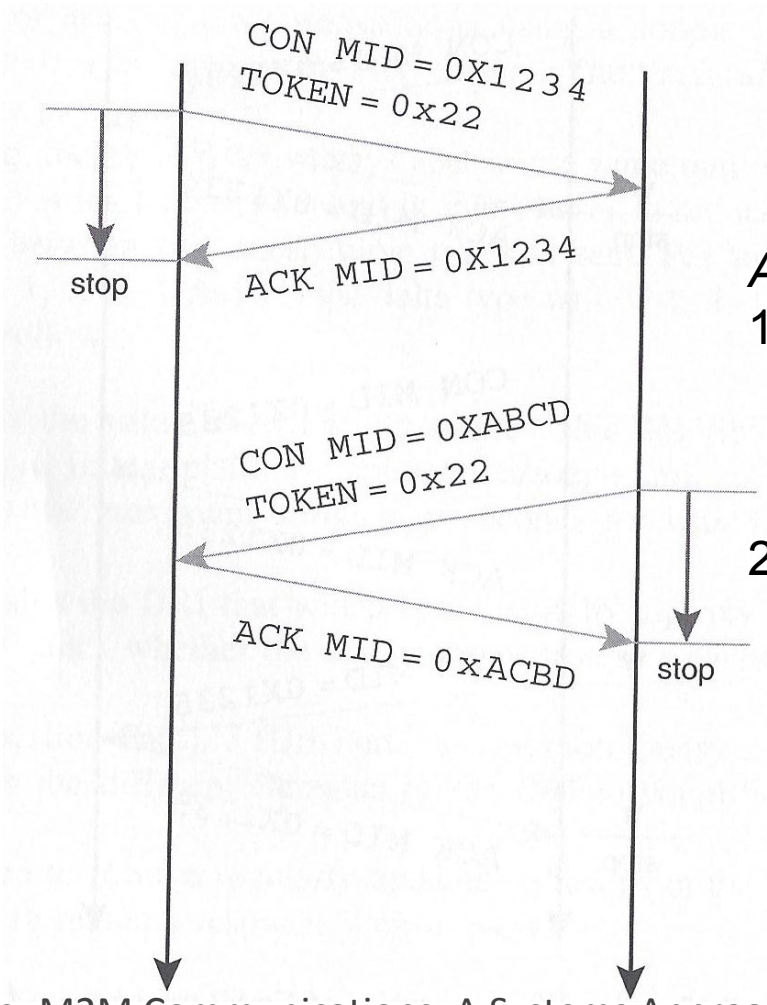
Synchronous Message Exchange

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.

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

Example 2 of CoAP Requests



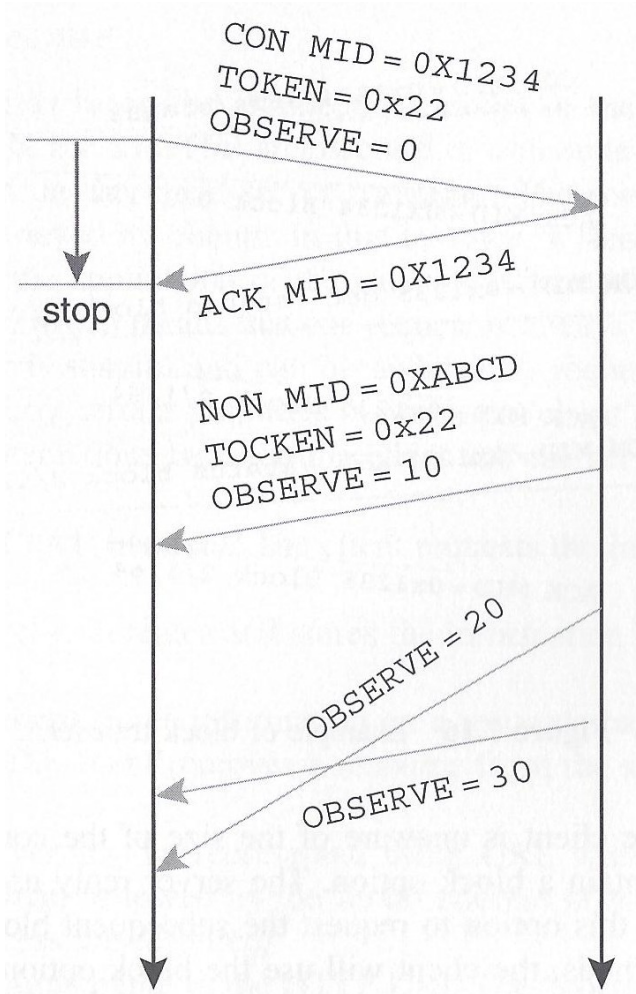
Asynchronous Message Exchange

1. A CONfirmable message with TOKEN option can be acknowledged immediately without a response.

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

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

Example 3 of CoAP Requests



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.

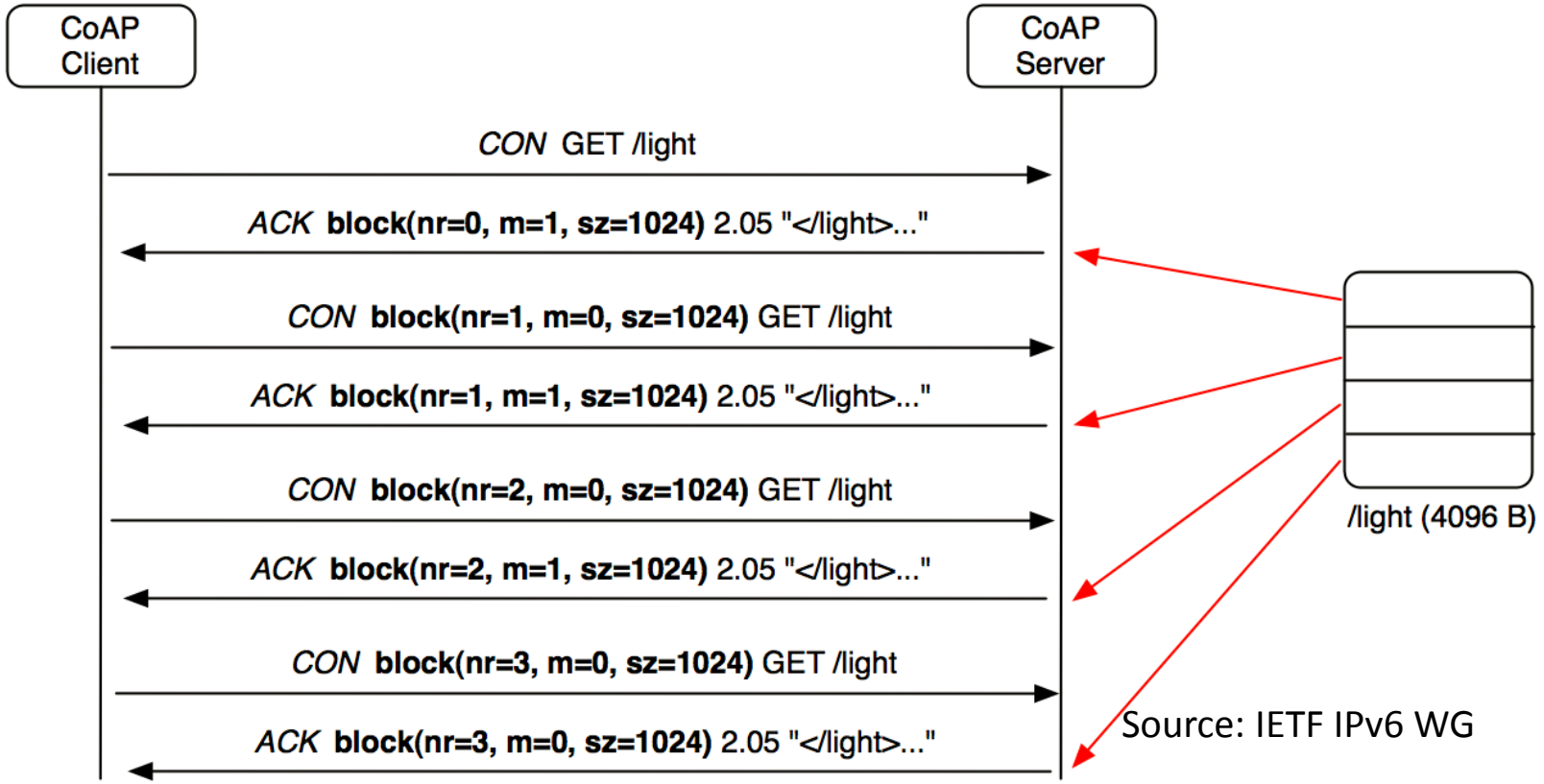
3. OBSERVE is the response will be increased to indicate the order of the response.

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

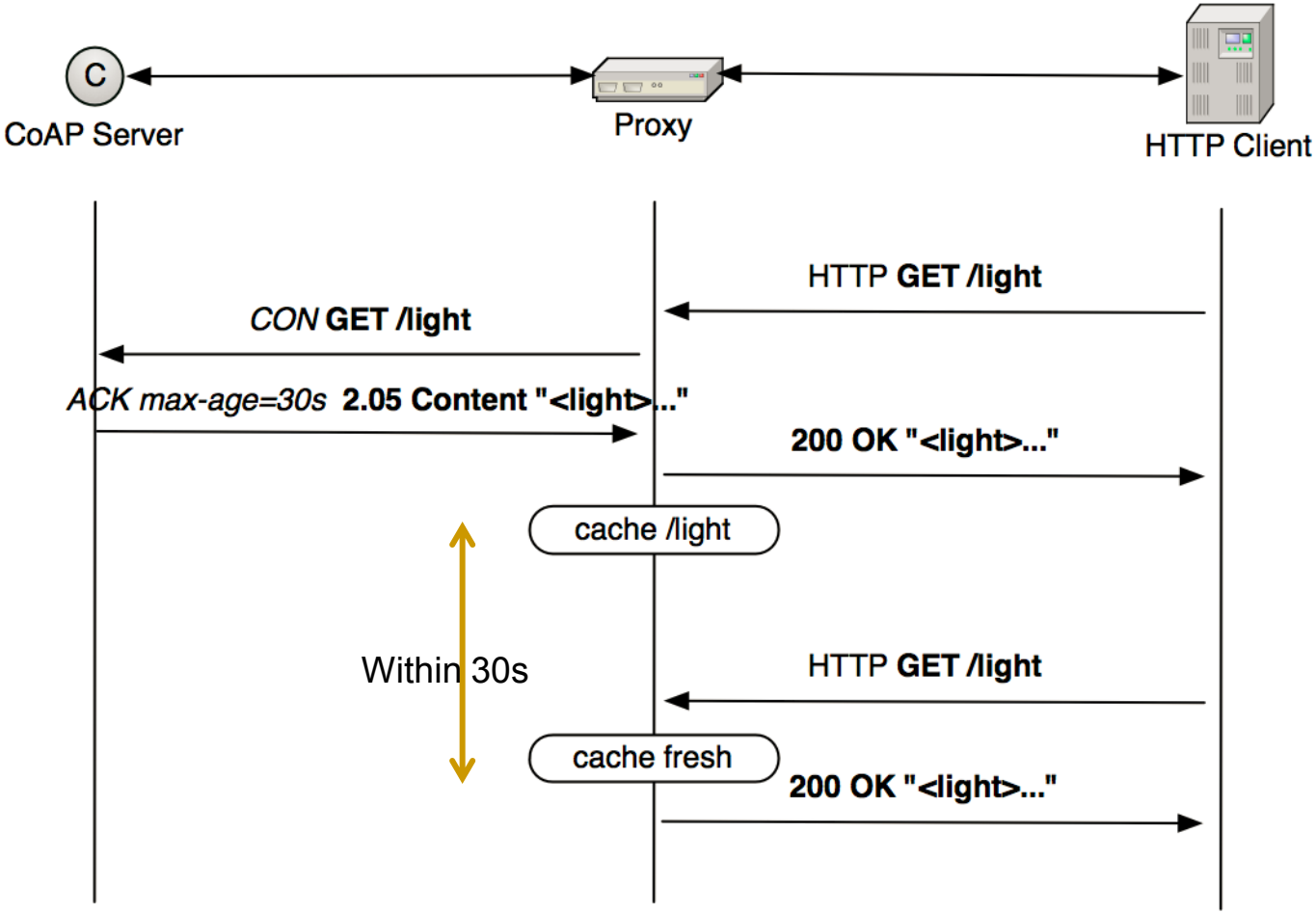
Example 4 of CoAP Requests



Block Transfer from Server to Client

1. A CONfirmable message from Client to get information.
2. Server indicates it has block of information to send.
3. Client then asks for more blocks of information.

Proxying and Caching



Source: IETF IPv6 WG

CoAP Caching Model

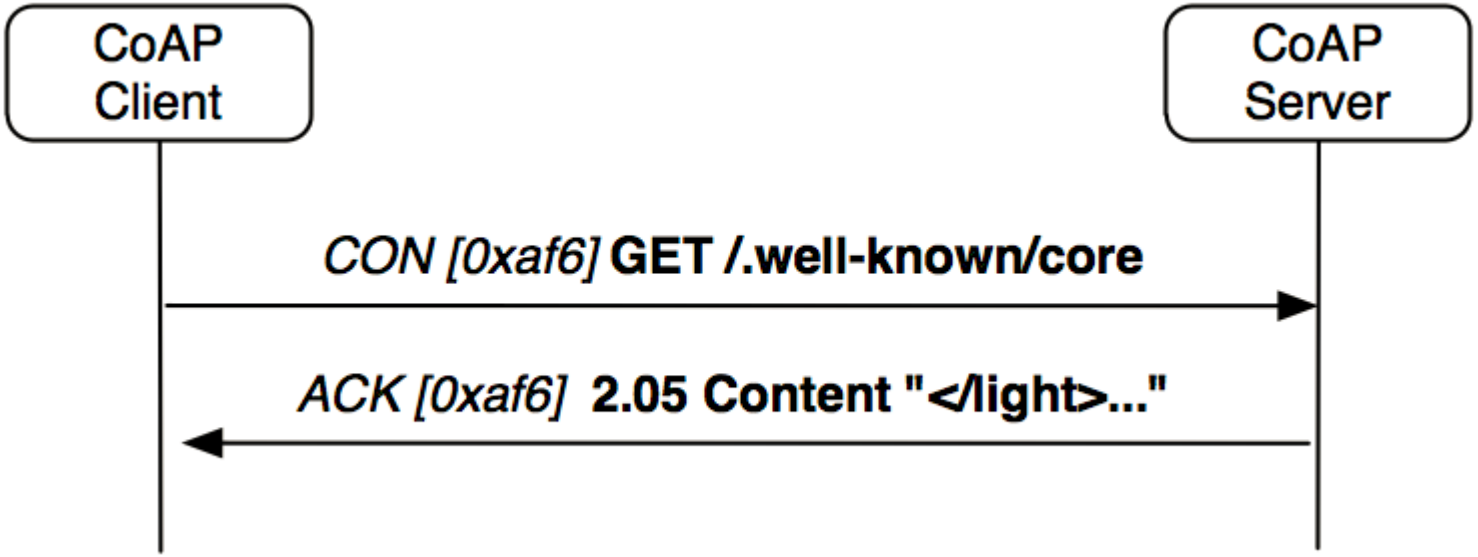
Cacheability determined by response code

- Freshness model
  - Max-Age option indicates cache lifetime
- Validation model
  - Validity checked using the Etag Option (http://en.wikipedia.org/wiki/HTTP_ETag)

CoAP Resource Discovery

- Resource Discovery with CoRE Link Format
  - Discovering the links hosted by CoAP servers
  - GET /.well-known/core
  - Returns a link-header style format based on RFC5988 including URL, relation, type, interface, content-type etc.
  - See draft-ietf-core-link-format

Example of Resource Discovery



</light>;rt="Illuminance";ct=0,
</s/maastr.xml>;title="Maastricht weather";ct=1,
</s/maastr/temp>;title="Temperature in Maastrich";ct=1,
</s/oulu.xml>;title="Oulu weather";ct=1,
</s/oulu/temp>;title="Temperature in Oulu";ct=1,
</s/temp>;rt="Temperature";ct=0

Source: IETF IPv6 WG

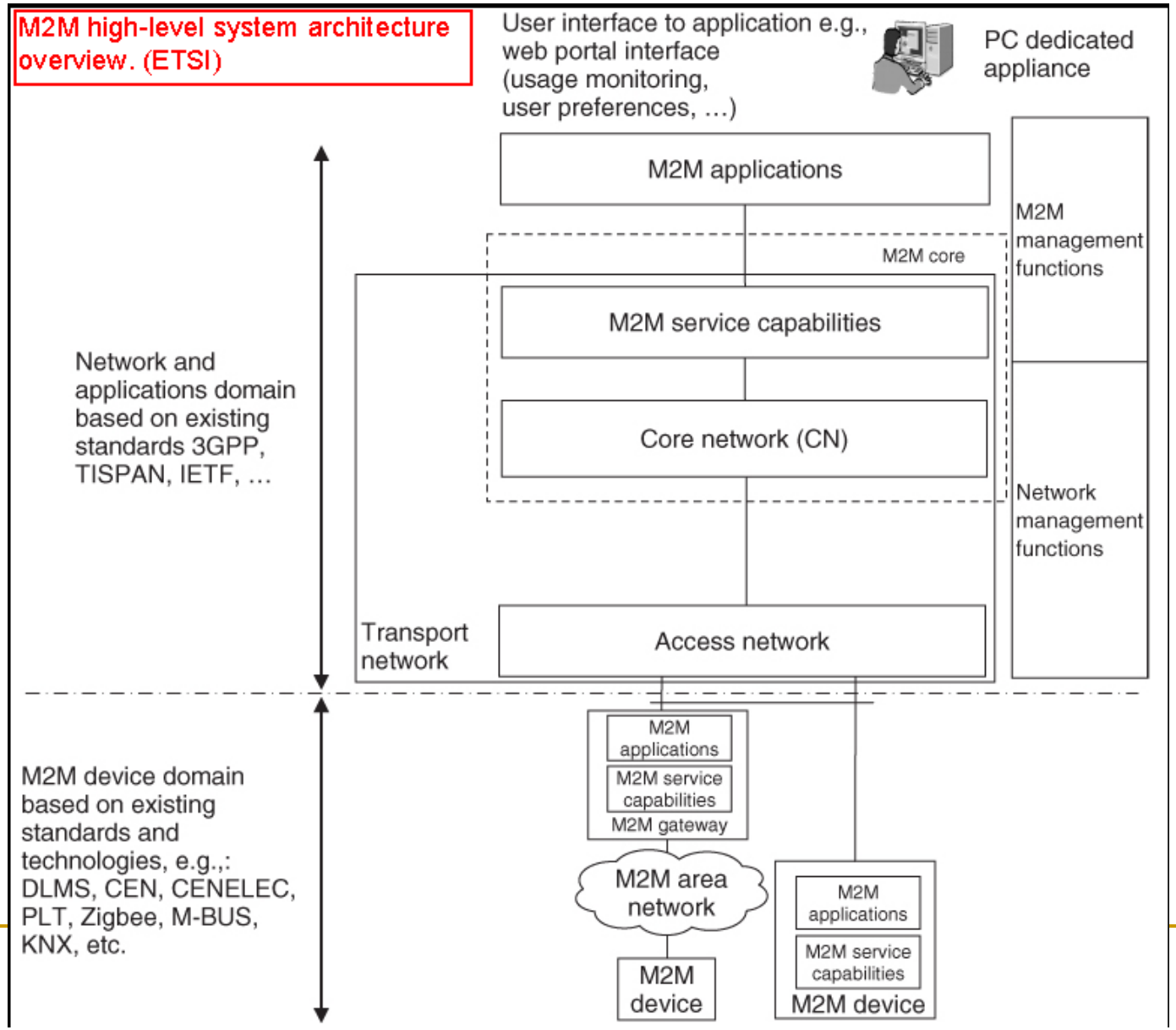
Summary

- CoAP is applicable to any IP networks.
- Open source software available for these protocols.
  - http://coapy.sourceforge.net/index.html
  - CoAPy: Constrained Application Protocol in Python

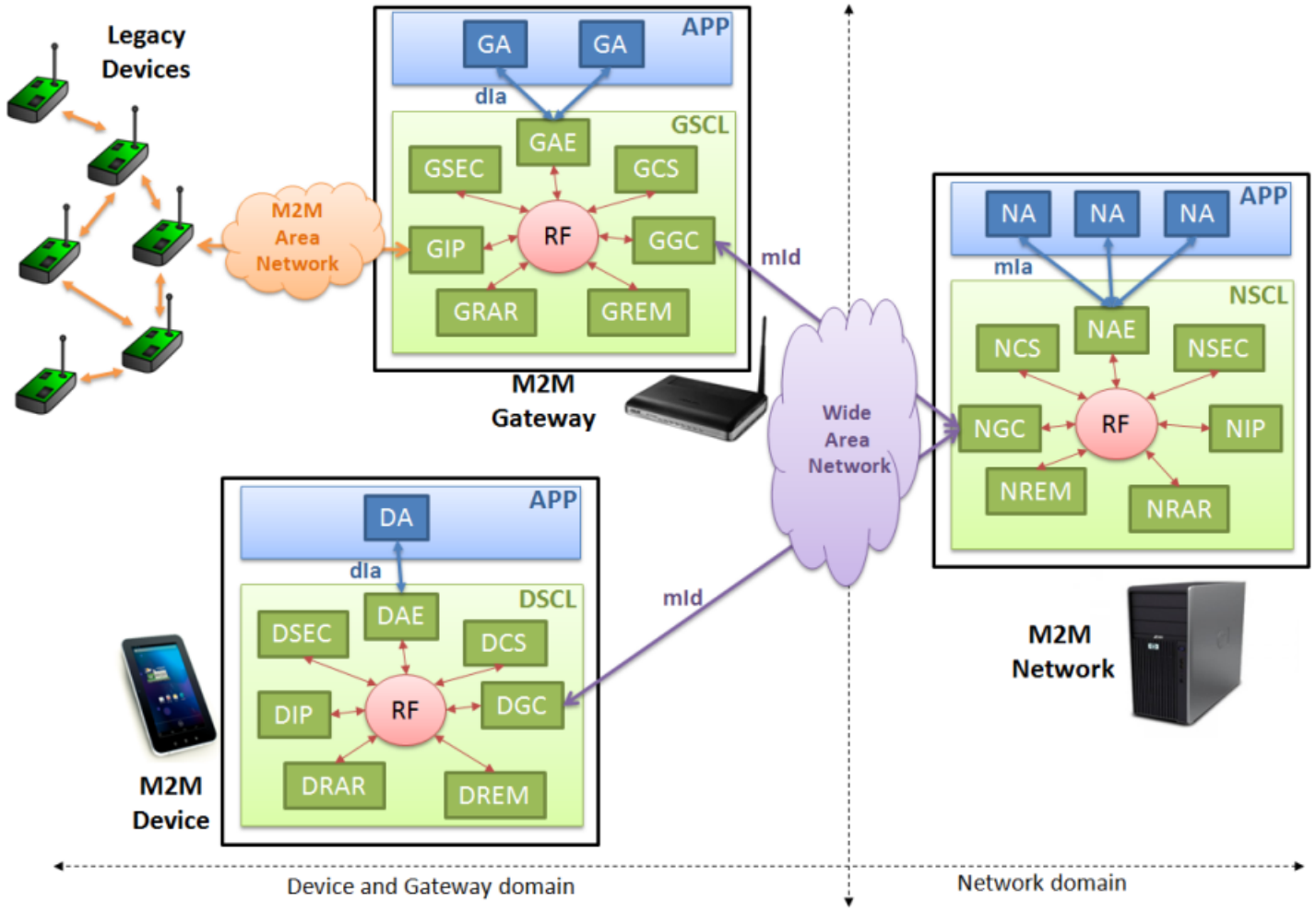
Outline

- 物聯網簡介 (Introduction to IoT)
- 物聯網重要網路協定(IoT Network Protocols)
  - IPv6
  - 6LoWPAN
  - RPL
  - CoAP
- 物聯網國際標準服務架構(ESTI M2M Service Architecture)
  - ESTI Service Capability and OM2M
  - oneM2M

M2M High Level System Architecture

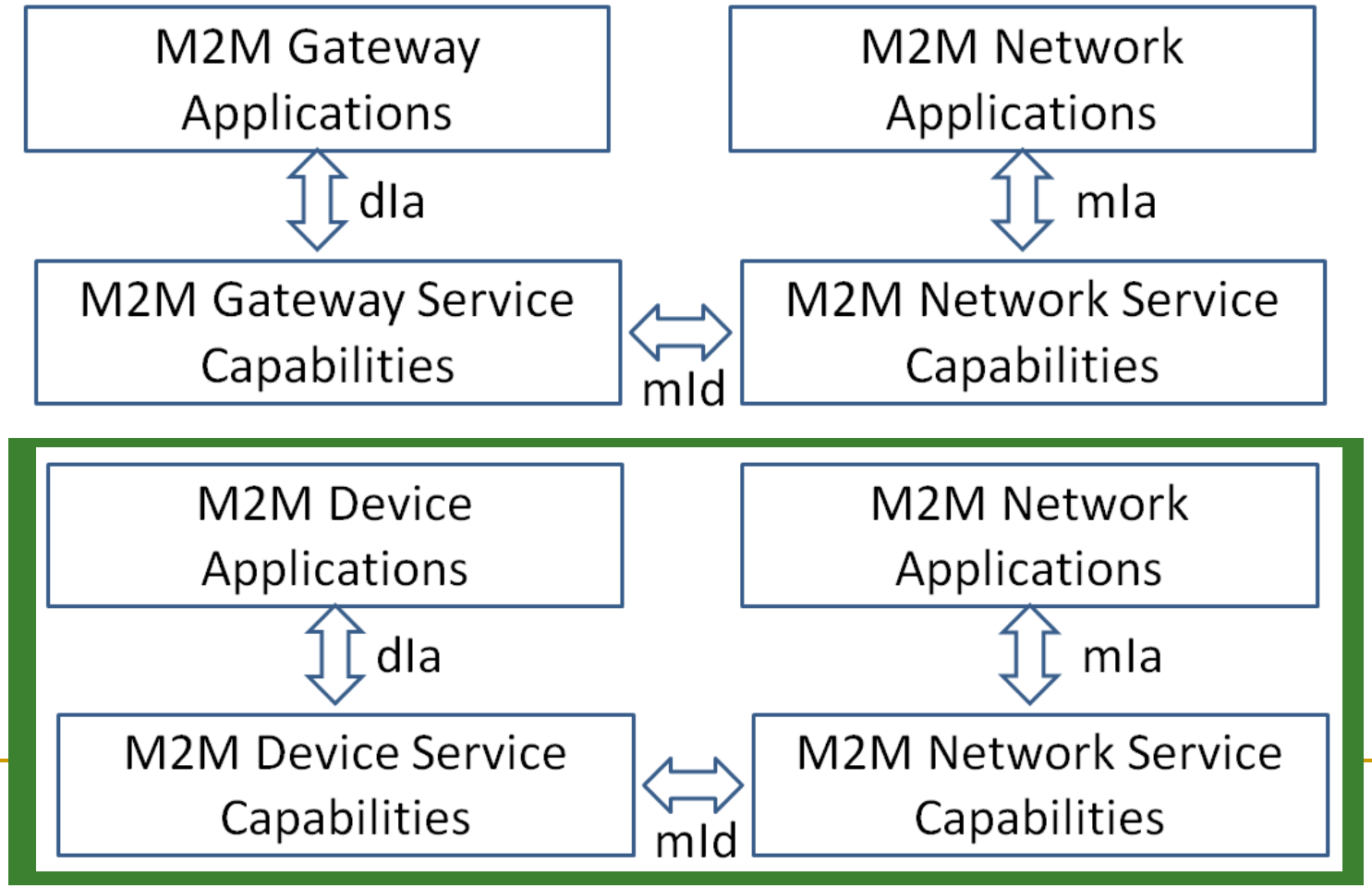


M2M Service Capabilities Framework



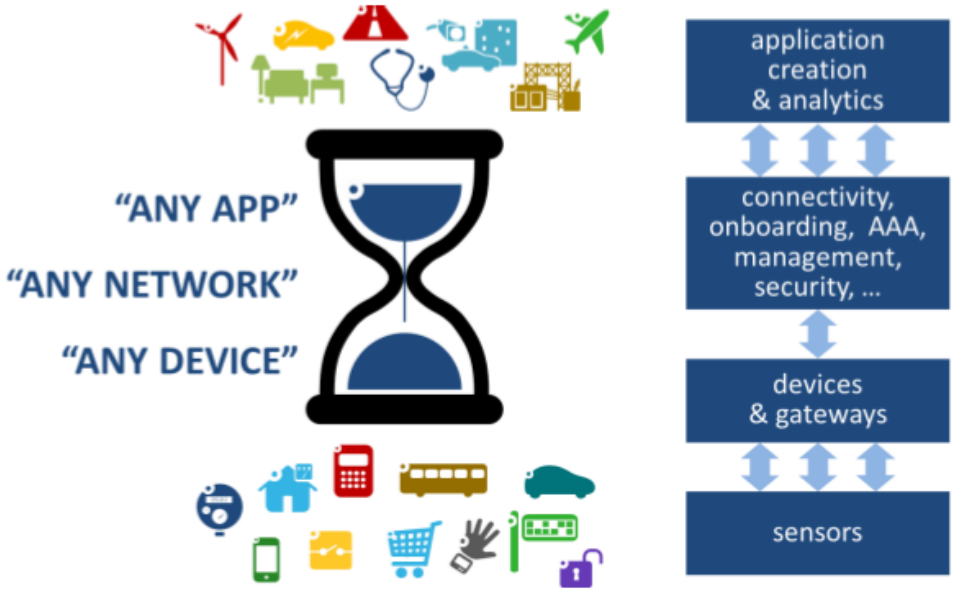
M2M Service Capabilities Framework

- M2M service capabilities can reside on the device (via dIa), the gateway (via dIa) or the network (via mIa).

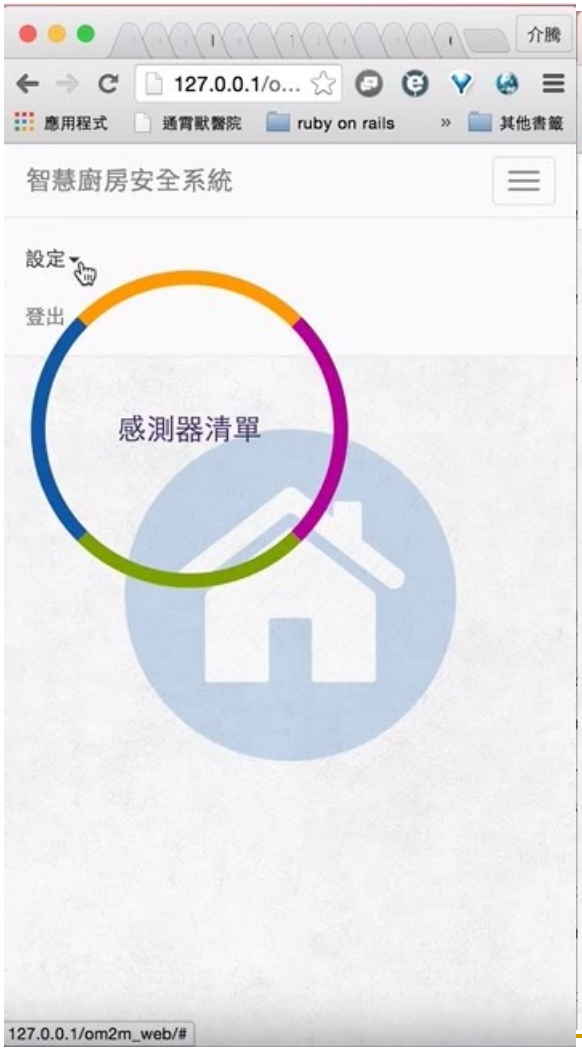


Eclipse OM2M Project

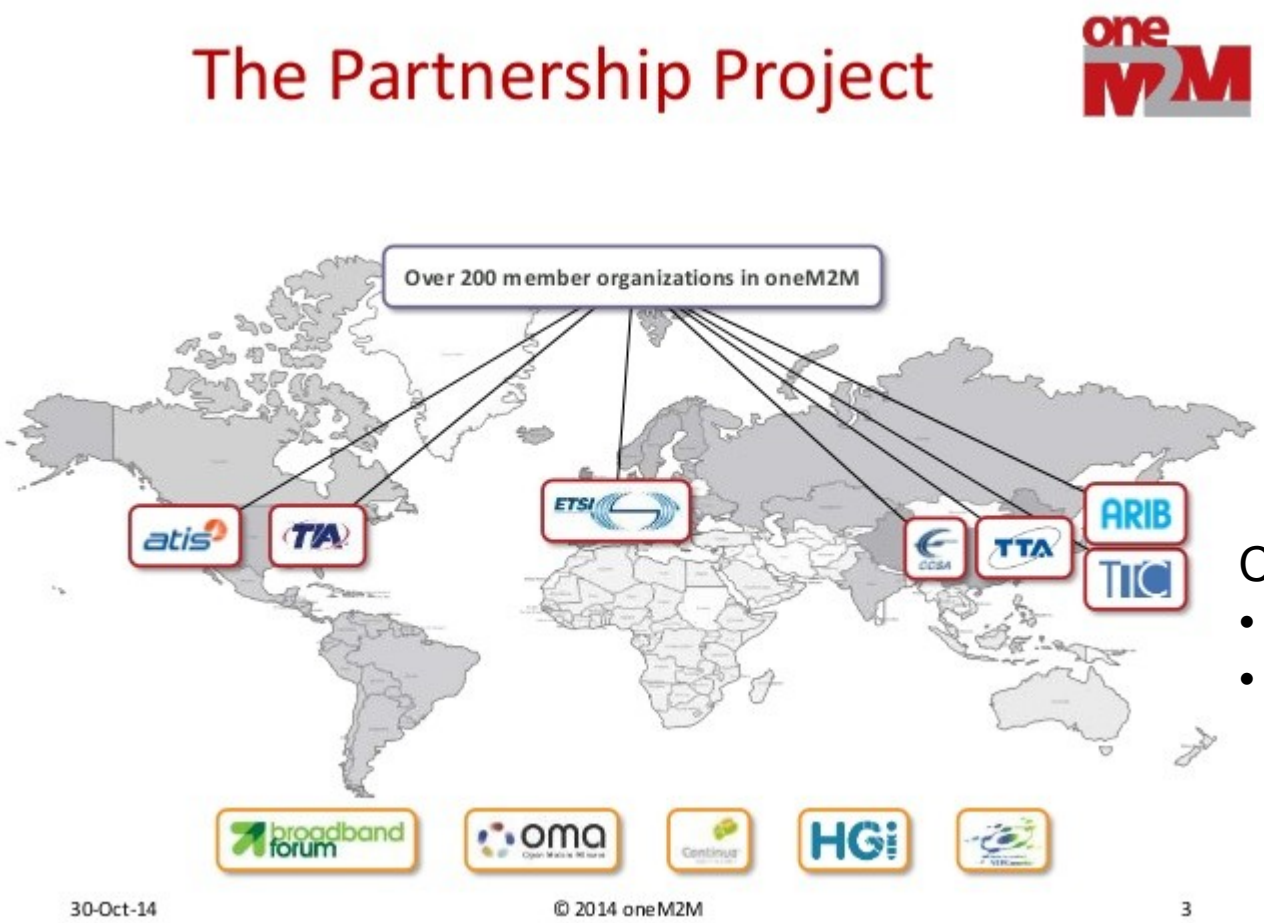
- Release 0.8.0 (April 8, 2015)
  - Implement the SmartM2M standard. (ETSI service capabilities framework)
- Release 1.0.0
  - Implement the oneM2M standard.



Eclipse OM2M Implementation



OneM2M



Chunghwa Telecom
- HTC
- III

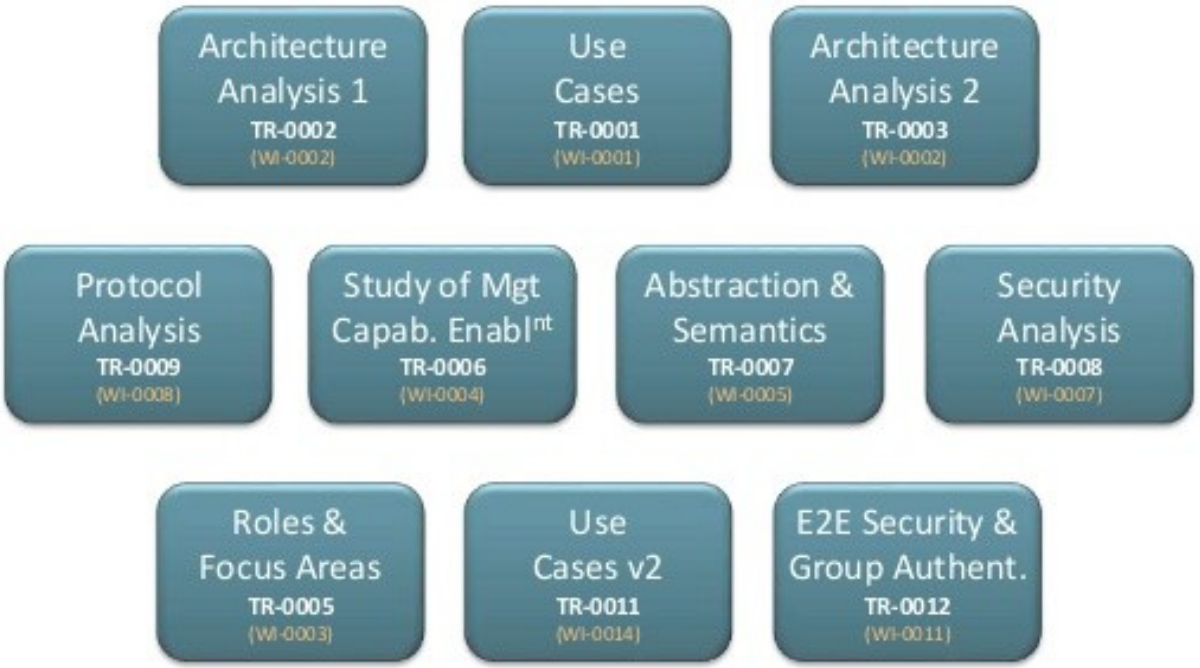
Technical Specifications

ftp://ftp.onem2m.org/Work Programme/

oneM2M Release 1 specifications

Reference	Version	Title	Date
TS 0001	1.6.1	Functional Architecture	01/2015
TS 0002	1.0.1	Requirements	01/2015
TS 0003	1.0.1	Security Solutions	01/2015
TS 0004	1.0.1	Service Layer Core Protocol Specification	01/2015
TS 0005	1.0.1	Management Enablement (OMA)	01/2015
TS 0006	1.0.1	Management Enablement (BBF)	01/2015
TS 0008	1.0.1	CoAP Protocol Binding	01/2015
TS 0009	1.0.1	HTTP Protocol Binding	01/2015
TS 0010	1.0.1	MQTT Protocol Binding	01/2015
TS 0011	1.2.1	Common Terminology	01/2015

Technical Reports

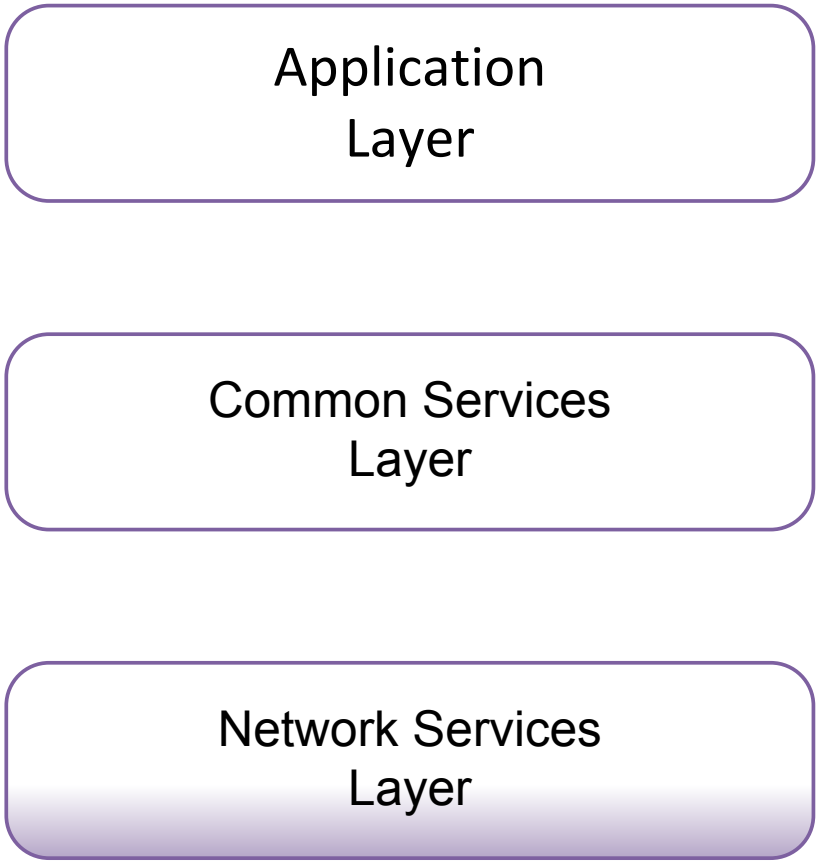


ftp://ftp.onem2m.org/Work Programme/

Technical Reports

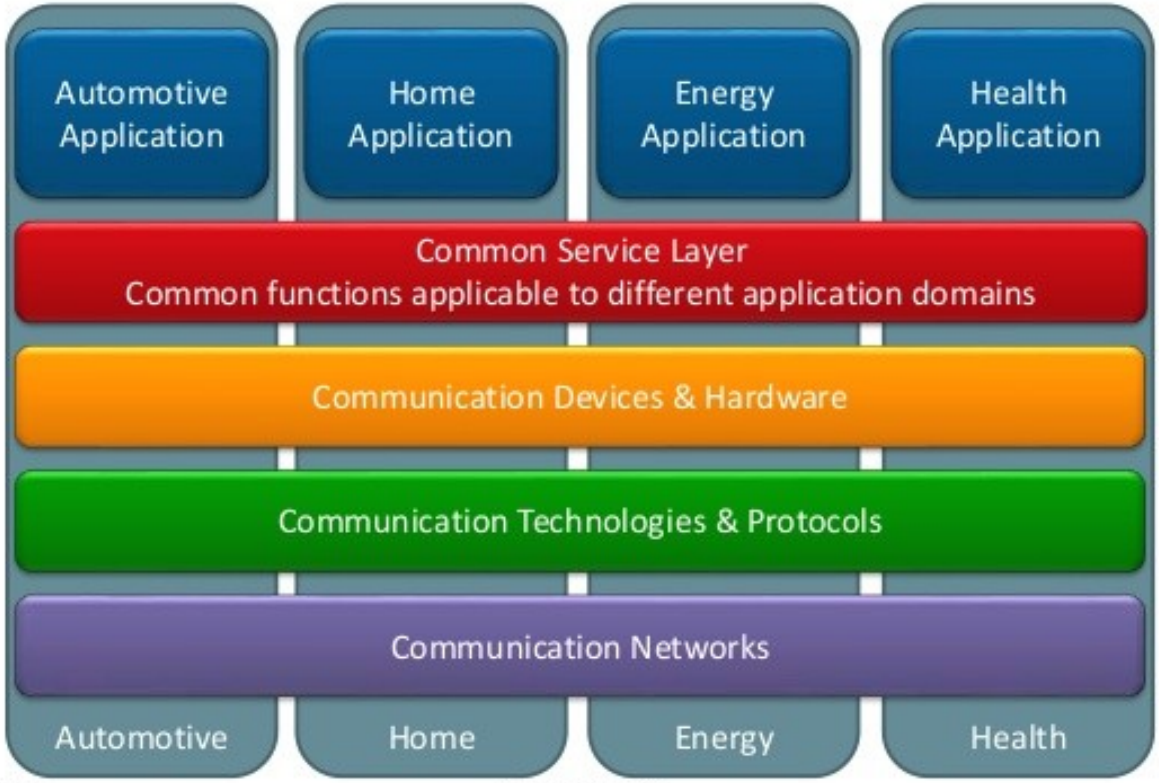
Reference	Version	Title	Date
TR 0001	0.0.5	oneM2M Use Cases Collection	11/2013
TR 0002	0.2.0	Architecture Analysis - Part 1: Analysis of architectures proposed for consideration by oneM2M	10/2013
TR 0003	0.5.0	Architecture Analysis - Part 2: Study for the merging of architectures proposed for consideration by oneM2M	10/2013
TR 0006	0.5.1	Study of Management Capability Enablement Technologies for Consideration by oneM2M	12/2013
TR 0008	1.0.0	Analysis of Security Solutions for the oneM2M System	07/2014
TR 0009	0.7.0	Protocol Analysis	11/2014

OneM2M Architecture Model

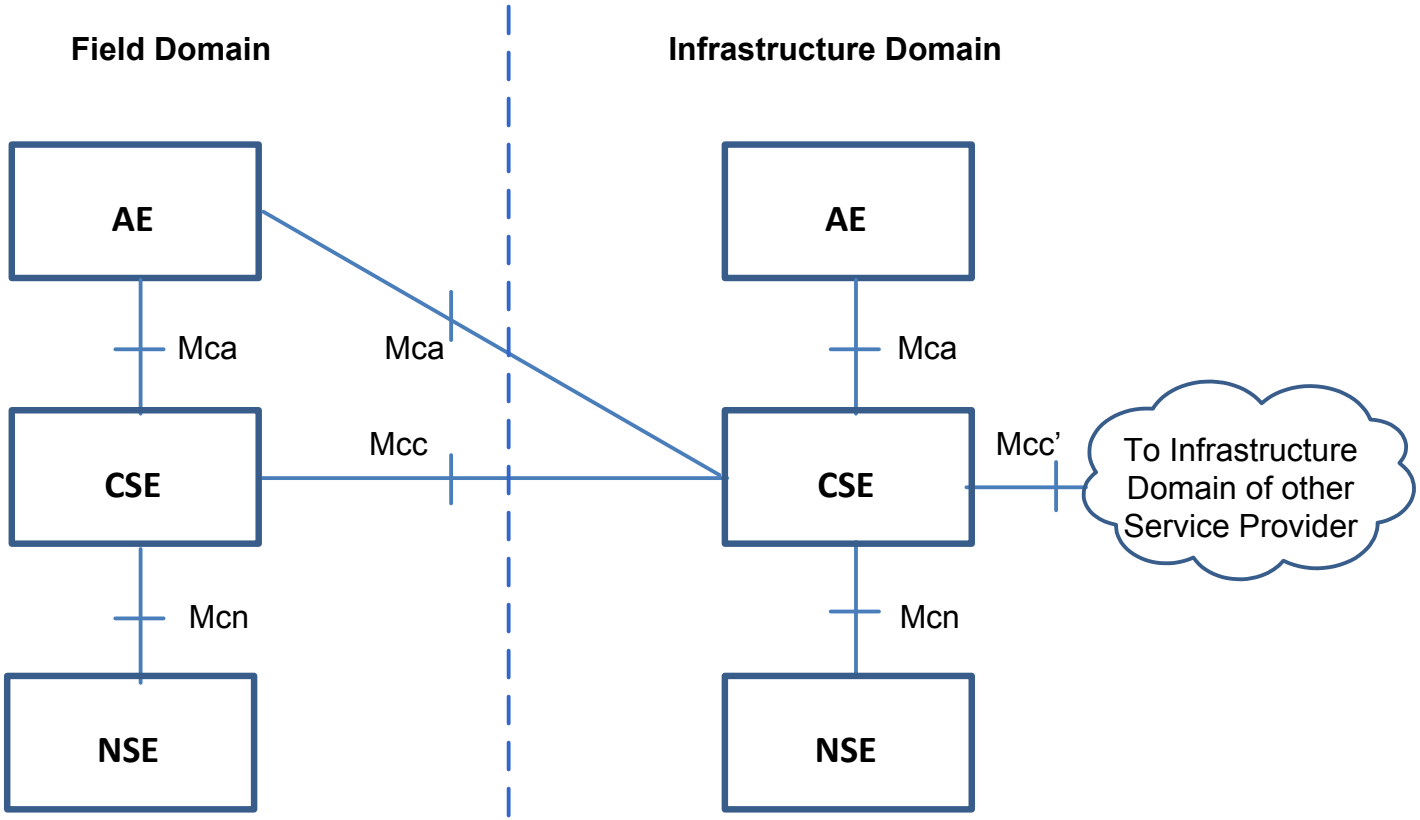


OneM2M

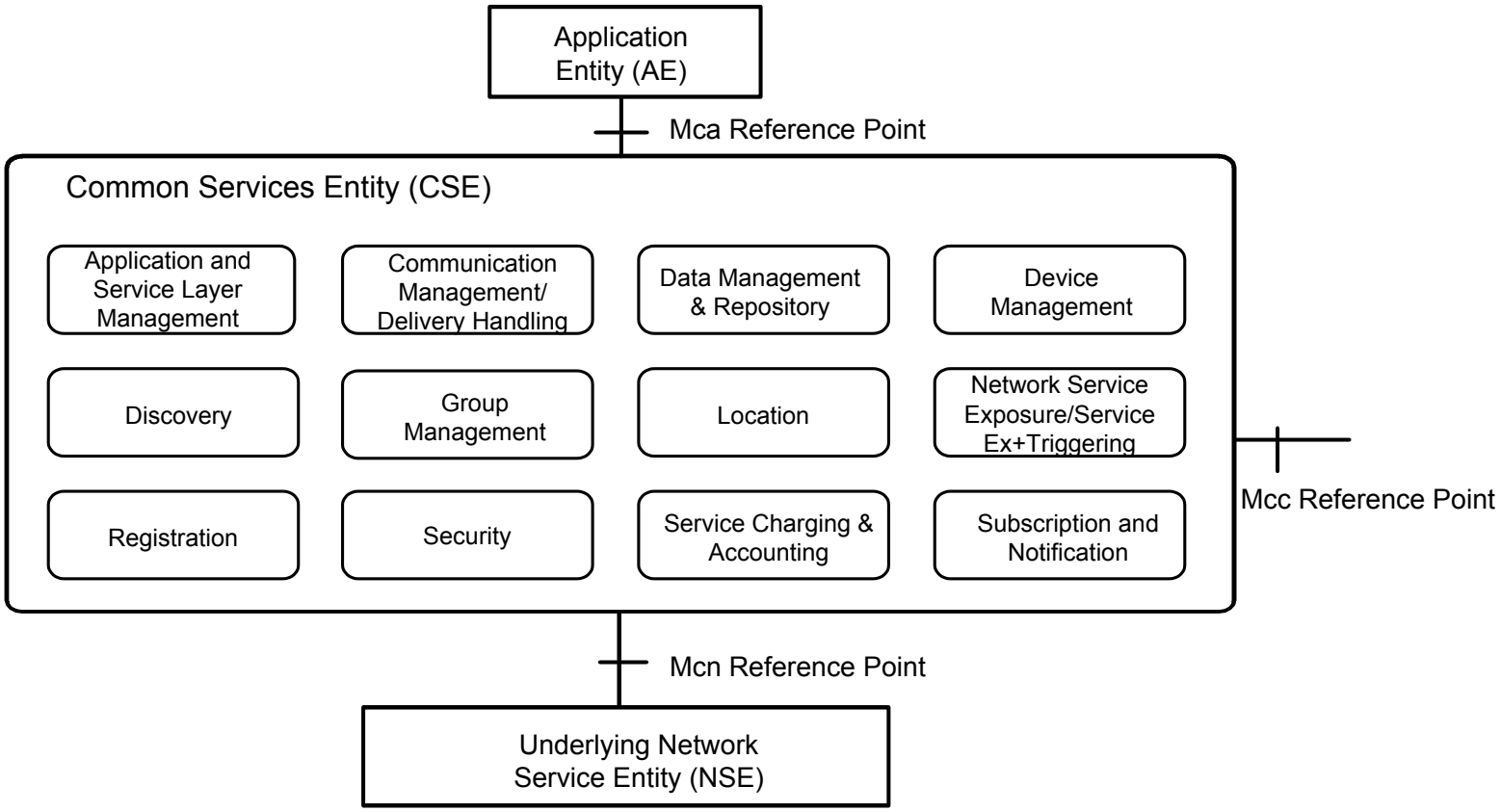
The Common Service Layer



OneM2M Functional Architecture



Common Services Functions



Communication Protocols

Reuse IP-based existing protocols

XML or JSON Content serialization

HTTP Example

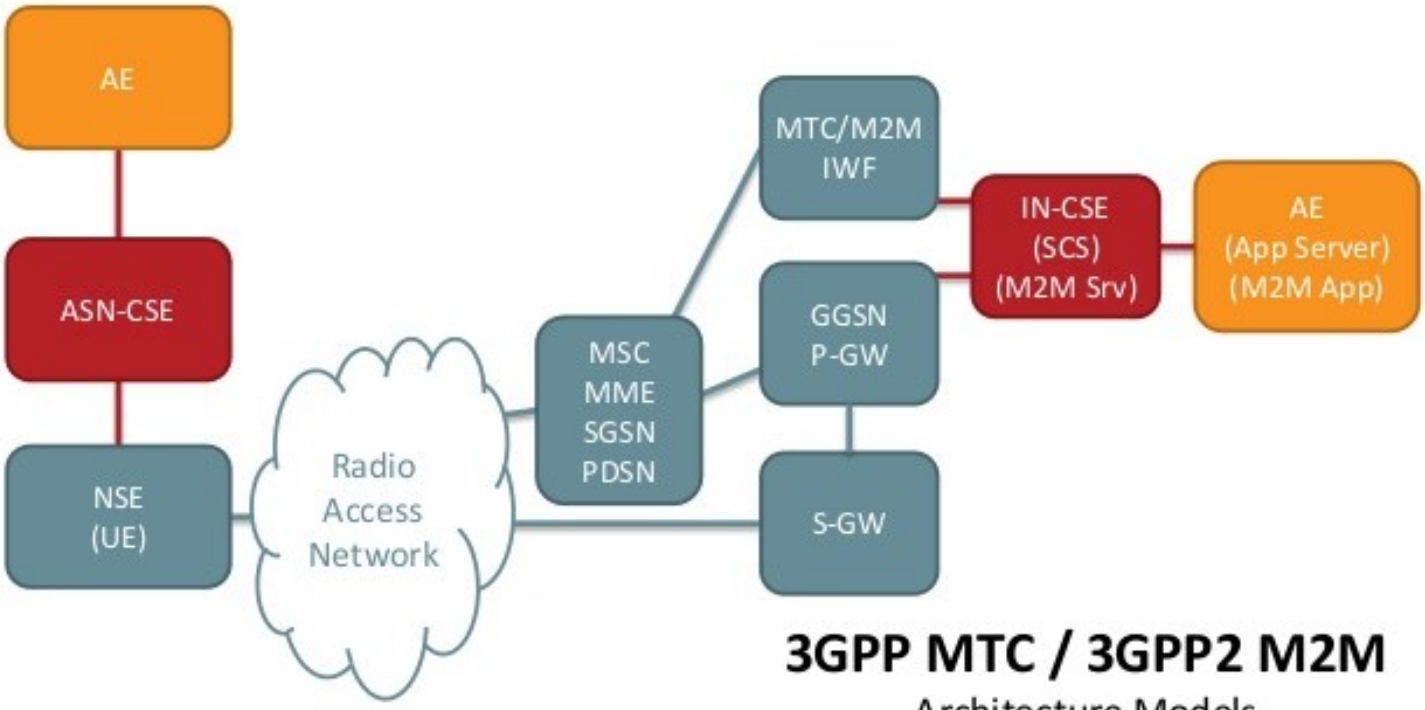
REQUEST

```
GET http://provider.net/home/temperature HTTP/1.1
Host: provider.net
From: //provider.net/CSE-1234/WeatherApp42
X-M2M-RI: 56398096
Accept: application/onem2m-resource+json
```

RESPONSE

```
HTTP/1.1 200 OK
X-M2M-RI: 56398096
Content-Type: application/onem2m-resource+json
Content-Length: 107
{"typeOfContent":"application/json",
"encoding":1,
"content": "{'timestamp':1413405177000,'value':25.32}"
}
```

Interworking – 3GPP/3GPP2



ASN-CSE CSE which resides in the Application Service Node
IN-CSE CSE which resides in the Infrastructure Node

Q&A