Tutorial on:

M2M COMMUNICATIONS

Nguyen Tai Hung & Ha Duyen Trung

Claims: This tutorial has usage of the slides and materials from colleagues and other sources!

Agenda

- Internet of Things (IoT) – Short Introduction
- M2M Communications – Architecture & Concepts
- M2M Connectivity Protocols
  - Ethernet
  - Wifi
  - Zigbee
  - Lora
  - 6LoWPAN
  - Cellular
- M2M Communication Protocols
  - TCP/IP Review
  - HTTP
  - MQTT
  - CoAP
  - XMPP
  - AMQP
  - RESTful Services
- Q&A

8 July 2018

Internet of Things

- Internet → The world-wise network of interconnected computer networks, based on a standard communication protocol (TCP/IP).
- Thing → An object not precisely identifiable.
- Internet of Things → A world-wide network of interconnected objects uniquely addressable, based on standard communication protocol.
- While current Internet is a collection of rather uniform devices, IoT will exhibit much higher level of heterogeneity, as objects of totally different functionality, technology and application fields will belong to the same communication environment.

Technological Trends

- It is possible to identify five distinct macro-trends that will shape the future of IT, together with the explosion of ubiquitous devices that constitute the future of IoT:
  - **Data explosion**: explosion of the amount of data collected and exchanged. Forecasts indicate that in the year 2015 more than 220 Exabytes of data will be stored. Novel mechanisms to find, fetch, and transmit data will be needed.
  - **Decrease in energy required to operate intelligent devices**: the search will be for a zero level of entropy where the device or system will have to harvest its own energy.
  - **Miniaturization of devices**: the devices will become increasingly smaller.
  - **Autonomic management:** the devices/systems will have self-management, self-healing, and self-configuration capabilities.
  - **IPv6 as an integration layer**:

Internet of Things Enablers

- **Energy:** issues such as energy harvesting and low-power chipsets are central to the development of IoT.
- **Intelligence:** devices should have capabilities such as context-awareness and inter-machine communication etc.
- **Communication:** new, smart multi-frequency band antennas, integrated on-chip and made of new materials are the communication means that will enable the devices to communicate.
- **Integration:** integration of smart devices into packaging, or better, into the products themselves will allow a significant cost saving and increase the eco-friendliness of the products.
- **Interoperability:** protocols for interoperability have to be standardized.
- **Standards:** open standards will be the key enablers for the success of the IoT. Sustainable. Fully global, energy-efficient communication standards that are security and privacy centered and use compatible or identical protocols ate different frequencies are needed.

8 July 2018

Internet of Things Barriers

- **Governance:** without an authority, it will be impossible to have a truly global IoT.
  - No universal numbering system currently exists. EPC Global and Ubiquitous ID are two different, non-compatible ways of identifying objects.
  - There is a need of keeping governance as generic as possible. One authority per field will certainly lead to overlap, confusion and competition between standards. Example: EPC Global architecture has a “single point of failure and control” where Verisign has the records of all the numbers, and can track where any object is.
  - What would be the governance of the IoT is an open question. Will it be a state-led agency, or a group under the supervision of the UN, or an industrial consortium?
- **Privacy and Security:**
  - Public acceptance of IoT will happen only when strong security solutions are in place.
  - The standards must define different security features to provide confidentiality, integrity, or availability of services.
  - The issues related to identity of people must be dealt with in politics and legislations.

8 July 2018

Internet of Things Applications

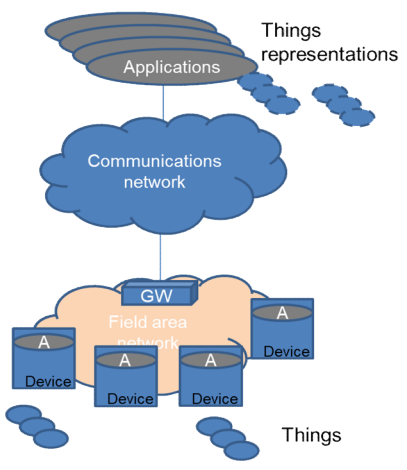
- **Manufacturing, logistics and retail sectors** → product authentication and anti-counterfeiting, next-generation industrial automation and supply chain management, inventory management, track & trace, remote maintenance, service and support.
- **Energy and utilities sectors** → smart electricity and water transmission grids, real-time monitoring of sewage systems, efficient energy and water consumption at homes enabled by connected devices to the grid.
- **Intelligent transportation systems** → support for vehicular ecosystems, use of in-vehicle sensor networks, telematics, GPS and wireless networks for developing smart vehicles, vehicle-to-vehicle and vehicle to roadside communication for collaborative road safety and efficiency, vehicle tracking, traffic data collection for traffic management etc.
- **Environment monitoring systems** → wireless sensor nodes to monitor weather, environment, civil structures, soil conditions etc.
- **Home management and monitoring** → use of sensor nodes, smart applications, wireless networks, home gateways for applications such as home security, elderly care, smart energy control etc.

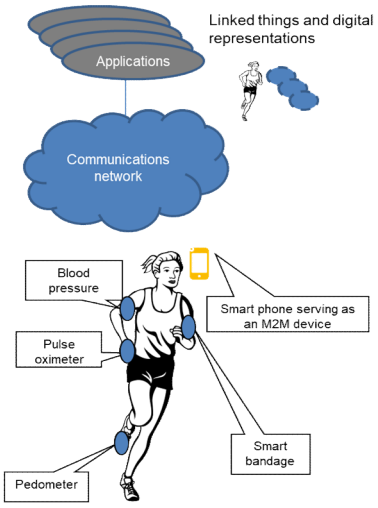
Internet of Things - Architectural Trends

- The following issues are important for IoT standardization
  - **Designing Web Services** as a common platform to publish service definitions and exchange configuration information between various hosts.
  - **Designing Messaging Services Layers** providing a basic web-services messaging framework between hosts which abstracts lower layers.
  - **Designing Common Data Exchange Formats** for sharing of structural data across different systems.
  - **Using Internet Protocol Layers or an IP proxy layer**, for connecting network nodes across multiple types of networking technologies, RF waveforms, and radio platforms.
- The architectural framework needs to incorporate all the desired aspects such as scalability, flexibility, adaptability etc.
- The components, and interfaces for various building blocks such as device interfaces, data formats, networking standards and protocols, service platforms and application interfaces are to be defined in IoT standards.

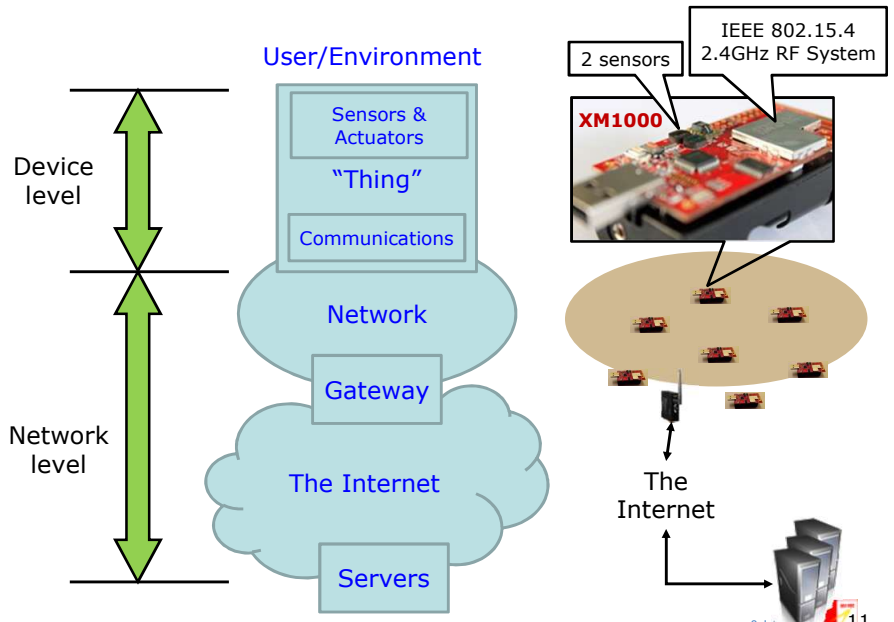
M2M Architecture & Concepts

M2M Communicational Architecture





Hardware Platform



Software Platforms and Services

- Operating Systems and execution environments
  - Contiki, TinyOS, Linus
- Relevant protocols and standards
  - 6LoWPan, CoAP, MQTT
- Architecture reference models
- ETSI M2M architecture and components
- Gateway/middleware

- Types of services
  - In conventional communication networks the target is moving bits from one place to another
  - In the IoT moving the data is not the actual goal.
    - The IoT is expected to facilitate providing meaningful information/actions.

8 July 2018 12

Communication in M2M Area Networks

- Wireless
  - 3G http://en.wikipedia.org/wiki/3G
  - 4G http://en.wikipedia.org/wiki/4G
  - 5G http://en.wikipedia.org/wiki/5G
  - IEEE 802.16p WiMAX Enhancements to Support M2M Applications http://en.wikipedia.org/wiki/IEEE_802.16
  - IEEE 802.11ah (sensor NWs, in progress) http://en.wikipedia.org/wiki/IEEE_802.11ah
  - 6LoWPAN (IPv6 Low Power Wireless) http://en.wikipedia.org/wiki/6LoWPAN
  - IEEE 802.15.4 http://en.wikipedia.org/wiki/IEEE_802.15.4
  - ZigBee http://en.wikipedia.org/wiki/ZigBee
  - Z-Wave http://en.wikipedia.org/wiki/Z-Wave
  - BT especially low energy v4.0 – v4.2 http://en.wikipedia.org/wiki/Bluetooth#Bluetooth_v4.0
  - KNX http://en.wikipedia.org/wiki/KNX_%28standard%29
  - Wireless M-BUS http://en.wikipedia.org/wiki/Meter-Bus
- Wired
  - KNX http://en.wikipedia.org/wiki/KNX_%28standard%29
  - PLC (Power Line Communication) http://en.wikipedia.org/wiki/Power-line_communication
  - M_BUS http://en.wikipedia.org/wiki/Meter-Bus

[1] p.13., [2] p.4., p.14

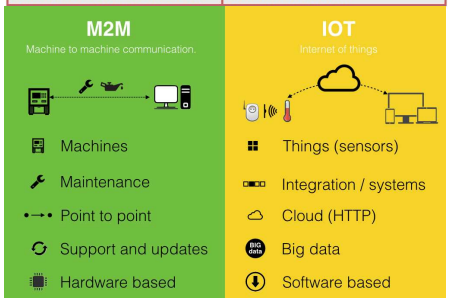
M2M Connectivity Protocols

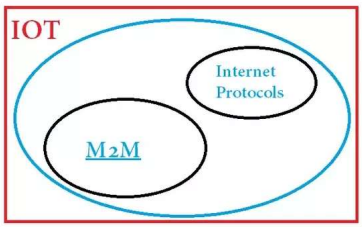
M2M/IoT Connectivity Protocols

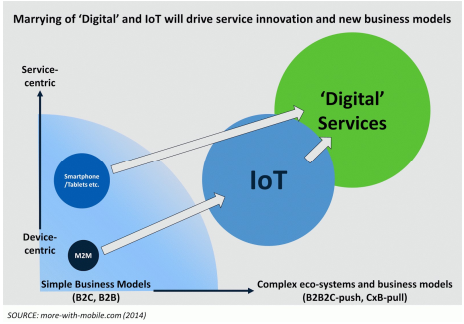
- **Wireless Personal Area Networks (WPAN)**: Bluetooth, ZigBee, Z-wave
- **Wireless Local Area Network (WLAN)**: Wifi-a/b/g/n/ah
- **Wireless Neighborhood Area Network (WNAN)**: 6LoWPAN, Zigbee
- **Wireless Wide Area Network (WWAN)**: LoRA, Cellular and Mobile Networks, NB-IoT (Narrow band IoT).

Key differences between M2M and IoT

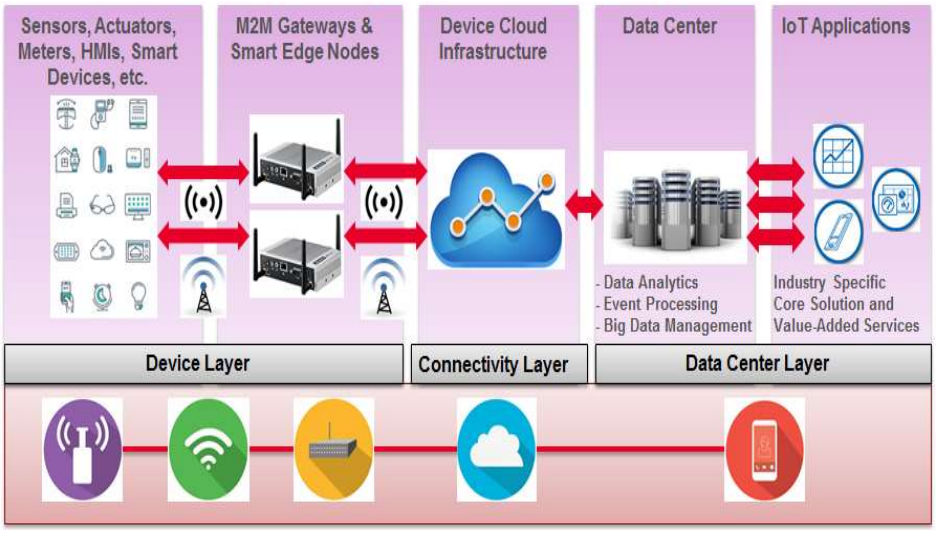
M2M	IoT
Point-to-point communication usually embedded within hardware at the customer site	Devices communicate using IP Networks, incorporating with varying communication protocols
Many devices use cellular or wired networks	Data delivery is relayed through a middle layer hosted in the cloud
Devices do not necessarily rely on an Internet connection	In the majority of cases, devices require an active Internet connection
Limited integration options, as devices must have corresponding communication standards	Unlimited integration options, but requires a solution that can manage all of the communications





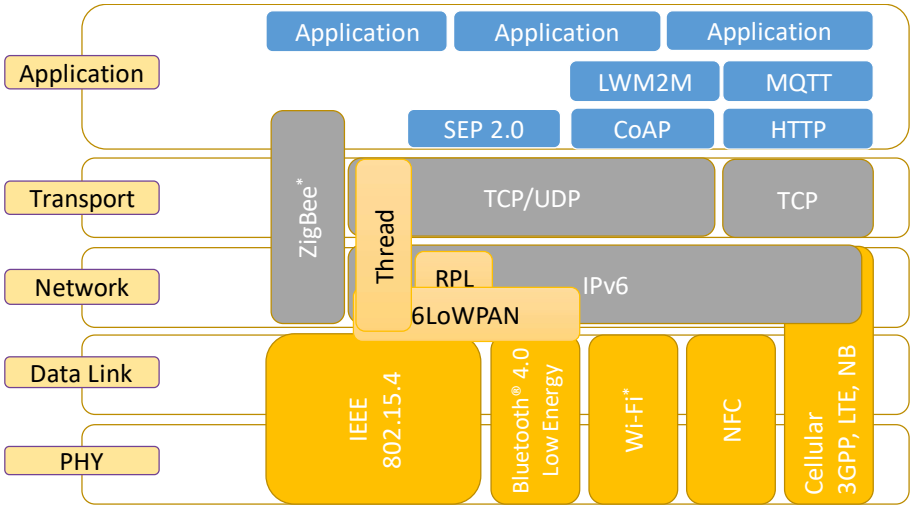


Connectivity layer in M2M/IoT reference model



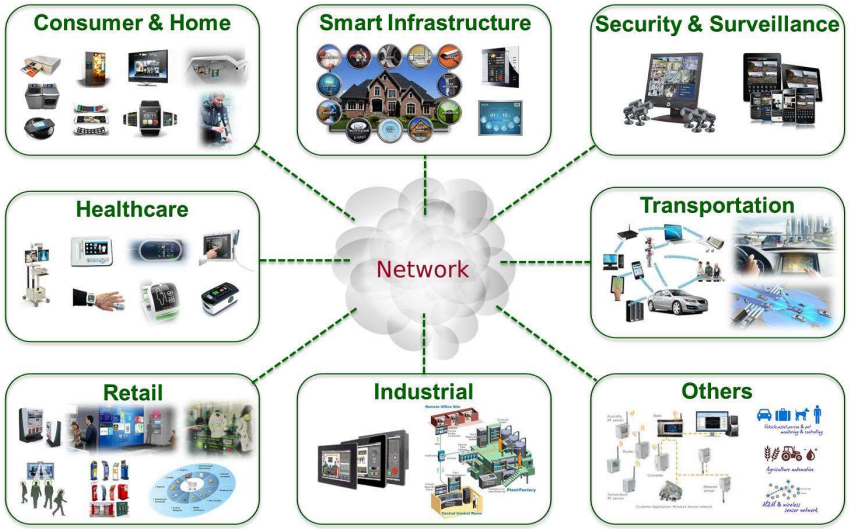
8 July 2018

Connectivity layers (cont.)



8 July 2018

Things / sensing layer

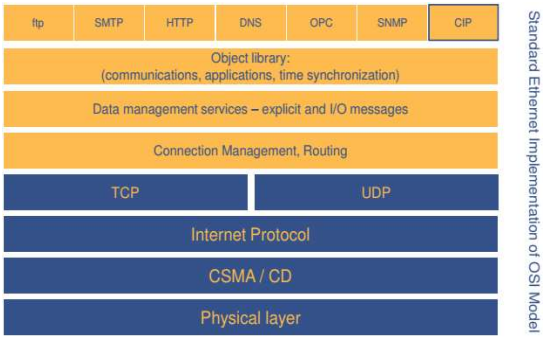


Source: https://www.electronicshub.org/internet-of-things/

Connectivity Layer: Wired/Wireless Technologies for the M2M/IoT

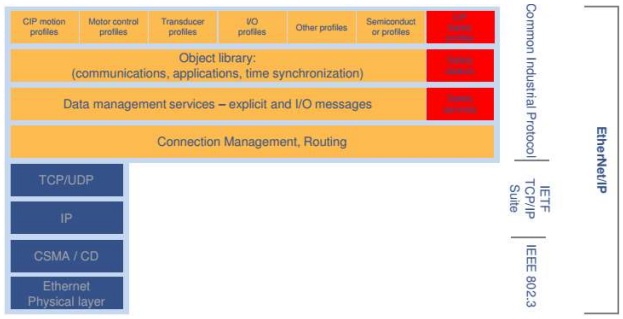
Ethernet/IP: enabling IoT in industry

- Ethernet/IP is peer to peer supporting the distributed architecture needed for IoT.
- EtherNet/IP is CIP (Common Industrial Protocol) on Ethernet.
- CIP co-exists with other TCP/IP applications
- CIP Features:
  - Open
  - Vendor independent
  - Network independence
  - Greater efficiency / flexibility
  - Seamless integration of networked devices.



EtherNet/IP: enabling IoT in industry

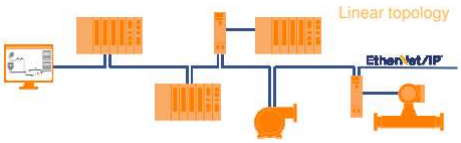
- **EtherNet/IP:**
  - Provides transparent data access from segment to segment and M2M.
  - Allows co-existence of concurrent datastreams: file transfer, video streams etc.
  - Network convergence.
  - Reduces design and operating costs.



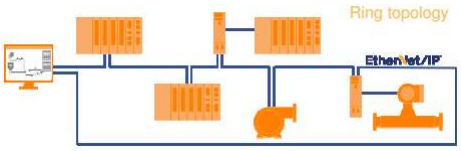
EtherNet/IP: enabling IoT in industry

EtherNet/IP flexibility:

- linear topology
  - Each device has 2 Ethernet ports and embedded switch technology
  - Simple 3 port switch can be used to connect existing 1-port Ethernet. devices

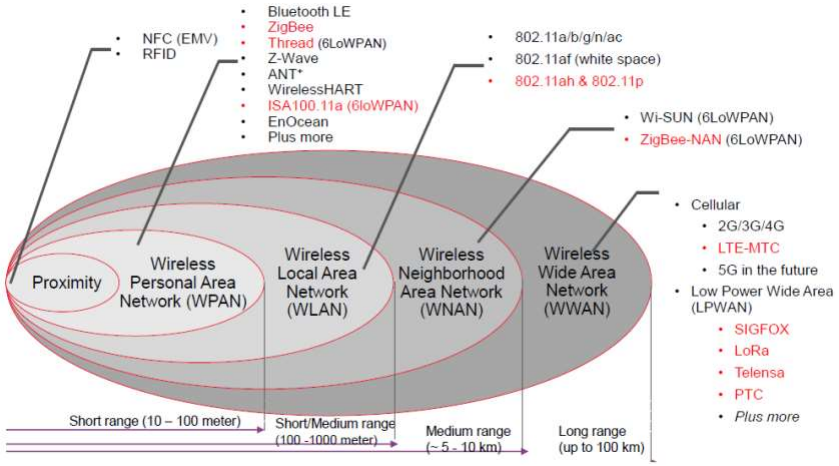


- ring topology
  - Provides fault tolerance
  - Continues to function in the event of a break.



8 July 2018

Key Enabling Wireless Technologies for IoT



[ref] M. Wang, "*Explosion of the Internet of Things: What does it mean for wireless devices*?," Keysight Technologies, 2015

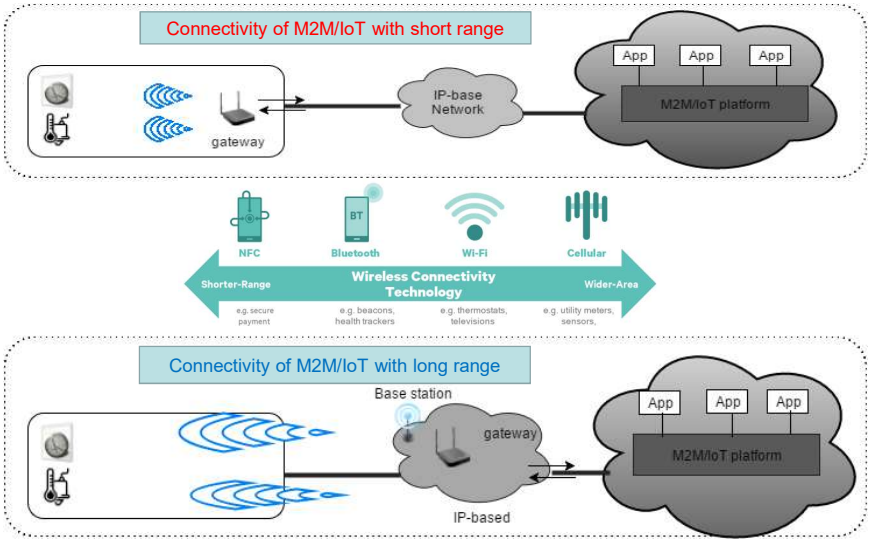
8 July 2018

Features vs transmission ranges

	Short Range				Long Range	
Features	**Bluetooth**	**802.11 (Wi-Fi)**	**802.15.4 (ZigBee/6LoWPAN)**	**RFID**	**LPWAN**	**Cellular**
security	64/128bit AES CCM	256 bits AES encryption	128 bit, AES	Low	Low	confidentiality
Latency	100ms/ <3ms (LE)	1.5ms	20ms			~90ms
Mobility	fixed	nomadic subnet roaming	Yes	Fixed	Yes	seamless global roaming
Range	10-100 meters	50-100 meters	10-200 meters	<3m	<10k	>1000m
Power Consumption	Medium Low (LE)	High	Low	Low	Very low	Medium
Battery life	Days years (LE)	Hours	Years	Years	>5	days
Max data rate	3 Mb/s 1 Mb/s (basic or LE)	22 Mb/s (802.11 g) 144 Mb/2 (802.11 n)	250Kb/s	Varies	<100 pbs	12Mb/s (4G LTE)

8 July 2018

Features vs transmission ranges



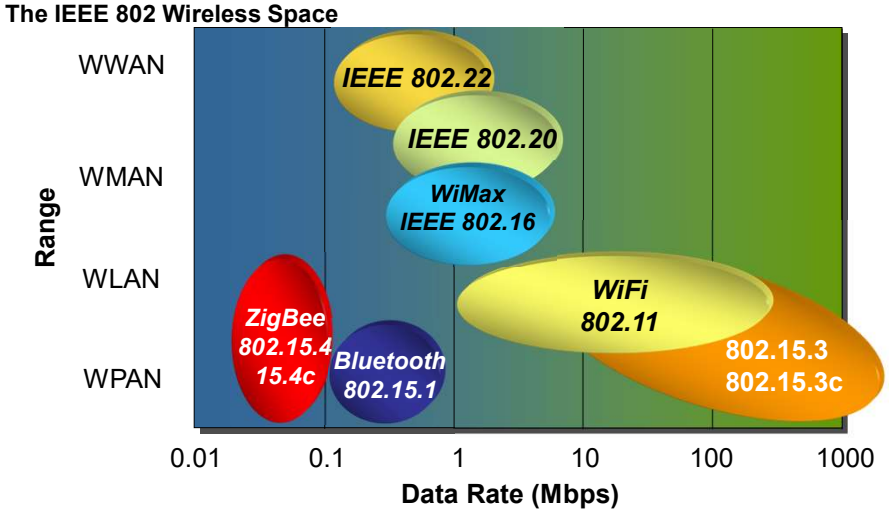
8 July 2018

Wireless Personal Area Network (WPAN)

8 July 2018

Zigbee/IEEE 802.15.4

The IEEE 802 Wireless Space



ZigBee standard uniquely fills a gap for low data rate applications

8 July 2018

Zigbee/IEEE 802.15.4

High level characteristics:

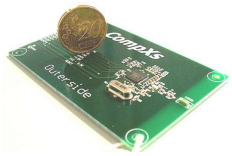
- Raw data rate: 868 Mhz - 20 kb/s,915 Mhz - 40 kb/s, 2.4 GHz - 250 kb/s.
- Range: 10 - 20 m.
- Latency: down to 15 ms.
- Channels:868/ 915Mhz & 2.4 GHz.
- Freq Bands: 2 PHYs; 868/915 Mhz, 2.4g
- Addressing: 8 bit or 64 bit IEEE.
- Channel access: CSMA – CA & slotted.

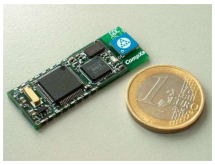
8 July 2018

Zigbee/IEEE 802.15.4

Device Type

- **Full Function Device (FFD)**
  - Network router function
  - Any Topology
- **Reduced Function Device (RFD)**
  - Easy and cheap to implement
  - Limited to star topology
- **Personal Area Network (PAN) Coordinator**
  - Maintains overall network knowledge
  - Needs most memory and computing
  - power

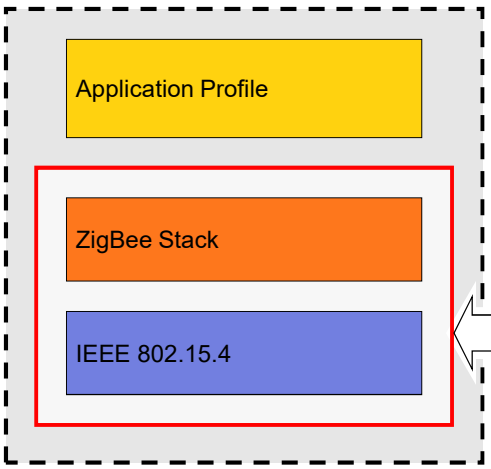




8 July 2018

Zigbee/IEEE 802.15.4

The ZigBee *Platform*

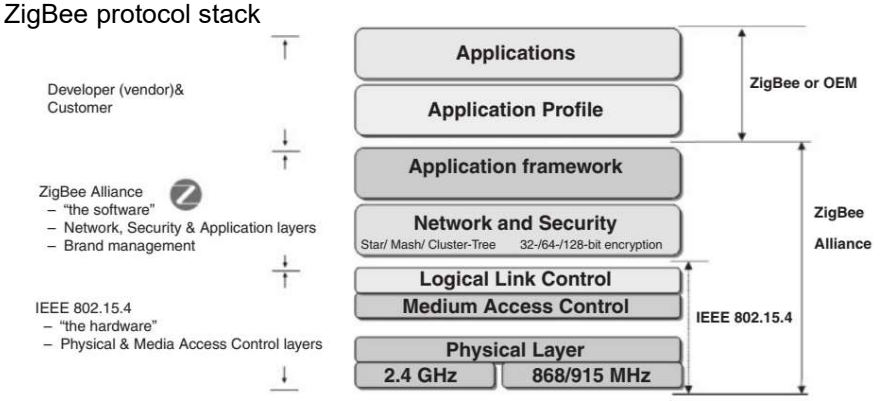


8 July 2018

Zigbee/IEEE 802.15.4

The ZigBee *Platform*

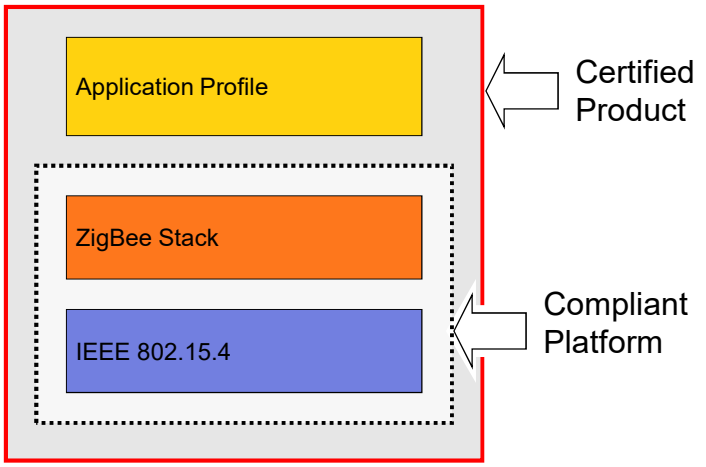
ZigBee protocol stack



8 July 2018

Zigbee/IEEE 802.15.4

The ZigBee *Product*

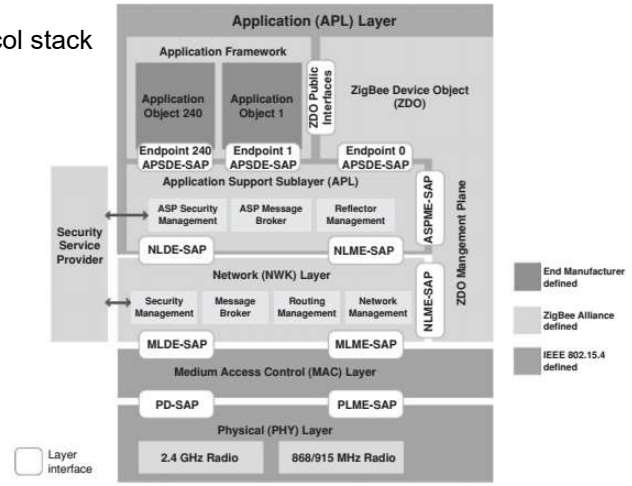


8 July 2018

Zigbee/IEEE 802.15.4

The ZigBee *Product*

ZigBee protocol stack



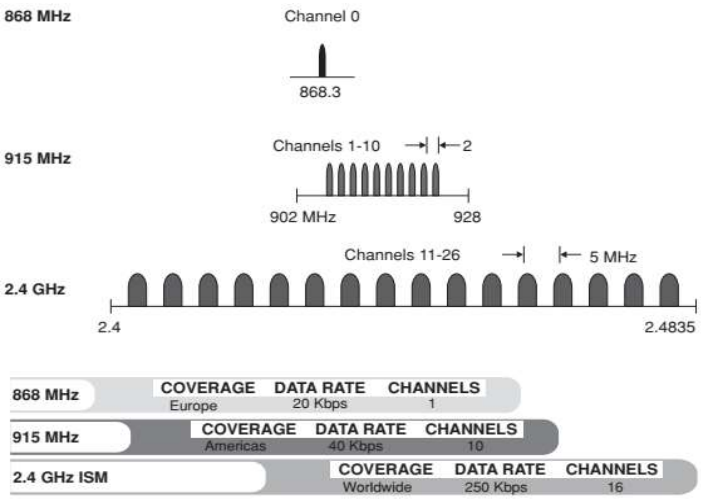
8 July 2018

Physical Layer and Channelization

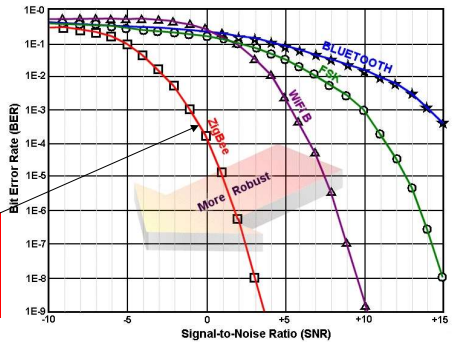
- ✓ 2 PHYs, based DSSS and differ by frequency band fundamentally.
- ✓ 868 Mhz in Europe & 915 Mhz –ISM in US and the other 2.4 Ghz.
- ✓ Another diff is the transmission rate.
- ✓ Each transmission rate has its own advantages concerned to modulation, throughput, latency etc.

- ✓ 27 channels across 3 bands.
- ✓ 1 channel between 868.0 & 868.6 Mhz.
- ✓ 10 channels between 902.8 & 928.0
- ✓ 2.4 Ghz PHY has 16 channels.
- ✓ PHY layers have other functions related to channels like scan function.

Physical Layer and Channelization



Basic Radio Characteristics

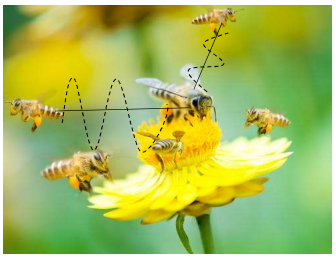


ZigBee technology relies upon IEEE 802.15.4, which has excellent performance in low SNR environments

Frequency Band	License Required?	Geographic Region	Data Rate	Channel Number(s)
868.3 MHz	No	Europe	20kbps	0
902-928 MHz	No	Americas	40kbps	1-10
2405-2480 MHz	No	Worldwide	250kbps	11-26

Modulation

- 868/915 Mhz PHY uses DSSS in which a transmitted bit is a 15 chip maximal length sequence.
- 868/915 Mhz employs Binary data and differential data encoding.
- 2.4 Ghz PHY employs a 16- ary quasi modulation technique based on DSSS.



Interference

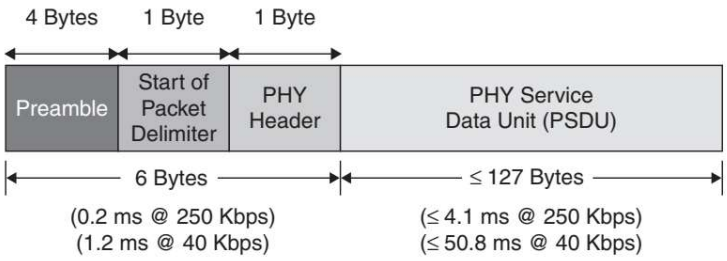
- Devices operating in 2.4 Ghz has to accept interference from other services.
- IEEE 802.15.4 expects low QoS.
- Demand: 802.15.4 needs excellent battery life.
- Why is 802.15.4 “BEST” among neighbors in 2.4 Ghz band?

Sensitivity & Range

- Receiver sensitivity range: -85dBm for 2.4 Ghz PHY and -92dBm for 868/915 Mhz.
- A star topology can provide complete home coverage.
- Mesh network topology gives the home coverage each device needs with enough power and sensitivity to communicate with nearest neighbor.

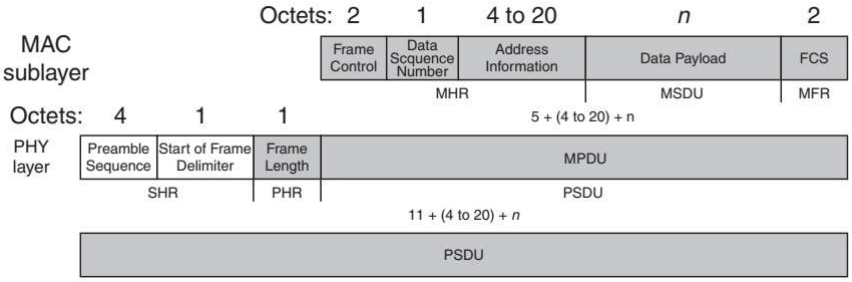
PHY Packet Structure

- PHY packet is PPDU that encompasses PSDU.
- It has a Preamble plus start of packet delimiter.
- PHY header has a 7 bit length payload that supports packet of 0-127 bytes.
- Home applications will require a packet size of 30-60 bytes.



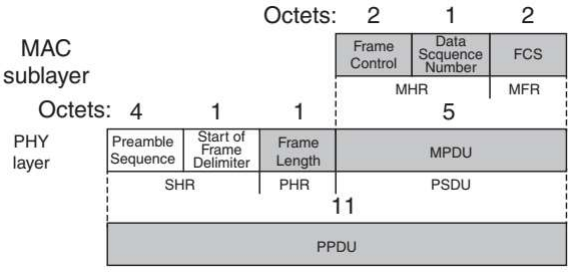
IEEE 802.15.4 frames

- Data frame format



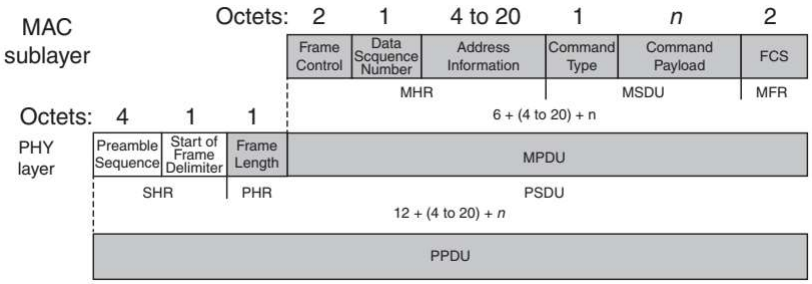
IEEE 802.15.4 frames

- Acknowledgement Frame Format



IEEE 802.15.4 frames

- **General MAC Command Frame Format:**
  - The MAC frame has a MPDU.
  - MPDU composes MHR, MSDU, MFR.
  - Size of address field 0-20B, data frame gives src & dest info.
  - IEEE 802.15.4 has 4 frame types.
  - Data and Beacon frames contain Info sent by higher layers.



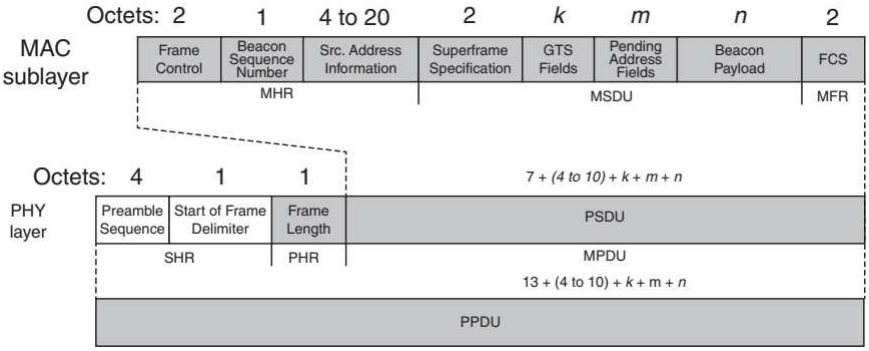
IEEE 802.15.4 frames

Super frame structure

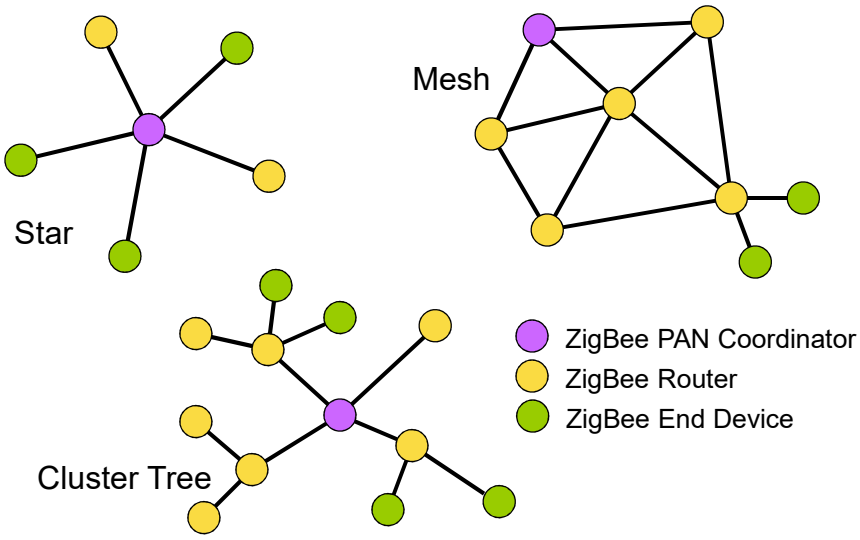
- Need for a Super frame ?
- Has dedicated network coordinator: PAN.
- PAN with its beacon plays a important role in assigning guaranteed time slots and maintaining a contention free period before next beacon.
- Coordinator's beacons: a contention-free guaranteed time service (GTS) is possible

IEEE 802.15.4 frames

- Beacon Frame format



ZigBee Network Topologies



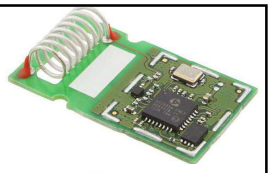
Z-wave

- This protocol was initially developed by ZenSys (currently Sigma Designs) and later was employed and improved by Z-Wave Alliance (**160 Z-Wave Alliance Members**).
- Z-wave is a low-power wireless communication protocol for Home Automation Networks (HAN) in smart homes and small-size commercial zones (together with Zigbee).
- Z-Wave covers about 30m point-to-point communication for tiny data transmission like light control, household appliance control, smart energy and HVAC, access control, wearable health care control, and fire detection.
- Z-Wave operates in ISM bands (around 900 MHz) and allows transmission rate of 40 kbps. The recent versions also support up to 200 kbps.
- Its MAC layer benefits from a collision avoidance mechanism. Reliable transmission is possible in this protocol by optional ACK messages.
- Z-wave networks include controller and slave nodes in. Routing in this protocol is performed by source routing method where a controller submits the path inside a packet.

[ref] Z-Wave "*Understanding Z-Wave Networks, Nodes & Devices*" http://www.vesternet.com/resources/technology-indepth/understanding-z-wave-networks, 2015. The online article explain about Z-Wave.
[ref] P. Amaro, R. Corteso, R. Landeck, and J. Santos, "*Implementing an Advanced Meter Reading infrastructure using a Z-Wave Compliant Wireless Sensor Network*," in Proc. of the 3rd International Youth Conference on Energetics, 2011.

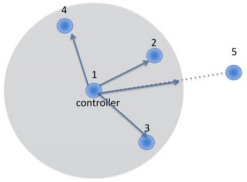
8 July 2018

Z-wave



Devices:

- Z-Wave has two basic types of device:
  - **Controllers** - devices that control other Z-Wave devices
  - **Slaves** - devices that are controlled by other Z-Wave devices.
- Very low cost and complexity (compared to WiFi, cellular)
- Low power
- Stacks are popular and available from multiple vendors
- Make your design easy - e.g. light control or HA for home automatic
- Interoperability: May be able to join existing home area network if on a compatible profile
- Can be mesh-networked without routing.



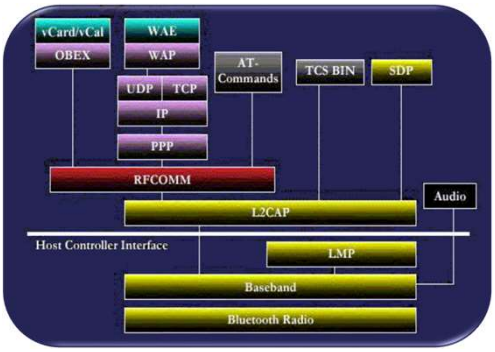
➢Uses sub-1GHz band and the Alliance Recommendation ZAD12837, "Z-Wave transceivers - Specification of Spectrum Related Components"
Link: http://z-wavealliance.org/z-wave-oems-developers/

8 July 2018

Bluetooth/IEEE 802.15.1

- Wireless Personal Area Networks (WPAN)
- Design goal
  - Cable replacement
  - Low cost
  - Low power
  - Small size
  - For mobile devices
- Standard: IEEE 802.15.1
- Classes
  - Class 1 (100mW, 100m range)
  - **Class 2 (2.5mW, 10m range)**
  - Class 3 (1mW, 1m range)
- RF
  - ISM band between 2.4-2.485GHz
  - Frequency hopping over 79 channels, 1600 hops/second

Bluetooth Protocol Stack



8 July 2018

Bluetooth Version

Version	Data rate	Feature
1.2	721 kb/s	
2.0 + EDR	3 Mb/s	Enhanced Data Rate (EDR)
3.0 + HS	24 Mb/s	High-Speed
4.0	1 Mb/s (BLE)	Bluetooth Low Energy (BLE)

8 July 2018

Wireless Local Area Network (WLAN)

Wifi and Low-power Wifi/IEEE 802.11

- The IEEE802.11 standard, known as Wifi, released in 1997, designed without IoT in mind.
- Provide high throughput to a limited number of devices, located indoor at short distances
- Due to its large energy consumption, interference issues, poor mobility and roaming support, Wifi has not been applied into M2M/IoT use cases.

Wifi / IEEE 802.11 Family

Protocol	Release	Freq. (GHz)	Data Rate (Mbit/s) (Typical / Max)	Range (m) (Indoor/outdoor)
A	Sep 1999	5 / 3.7	20 / 54	35 / 120
B	Sep 1999	2.4	5.5 / 11	35 / 140
G	Jun 2003	2.4	22 / 54	38 / 140
N	Oct 2009	2.4 / 5	110+ / 300+	70 / 250

Wifi and Low-power Wifi/IEEE 802.11

New

- In 2010, IEEE802.11ah Task Group (TGah) is formed, called Low-power Wifi.
- IEEE802.11ah extends the application area of Wifi network to meet the IoT requirements (large number of devices, coverage range, and energy constrains) in many IoT applications (e.g., smart grids, industrial automation, environmental monitoring, etc.).
- An access point (AP) has to cover hundreds, or thousands of devices (sensors, actuators), which periodically transmit short packets.
- IEEE 802.11ah draft 1.0 was conducted in September 2013.
- The standardization completed approximately by 2016.
- The IEEE 802.11ah standard is one of the candidate standards for IoT and Machine to Machine (M2M) applications (which is still in its preliminary stage of development???)

E. Khorov, A. Lyakhov, A. Krotov, and A. Guschin, "A survey on ieee 802.11 ah: An enabling networking technology for smart cities," *Computer Communications*, vol. 58, pp. 53–69, 2015.
T. Adame, A. Bel, B. Bellalta, J. Barcelo, and M. Oliver, "Ieee 802.11 ah: the wifi approach for m2m communications," *Wireless Communications, IEEE*, vol. 21, no. 6, pp. 144–152, 2014.

IEEE 802.11ah

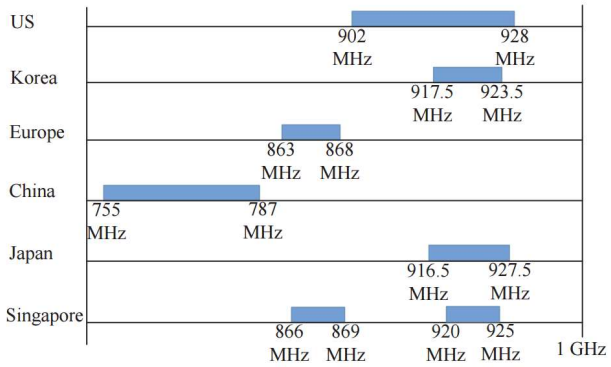
- IEEE 802.11ah using sub 1-GHz frequency to transfer data.
- It's a new standard for IoT, because it can transfer a long range more than 1000 meter, least 100 kbps data rate, and compatible with 802.11 WLAN legacies.
- The common standards we use everyday:
  - IEEE 802.11n → 2.4GHz
  - IEEE 802.11ac → 5GHz
- IEEE 802.11ah advantages:
  - Low power consumption
  - Long battery life
  - Burst data transmission
  - Easy Setup
  - High Transmission range
  - On-Hop reach
  - Reliability.

IEEE 802.11ah

Free space path loss equation

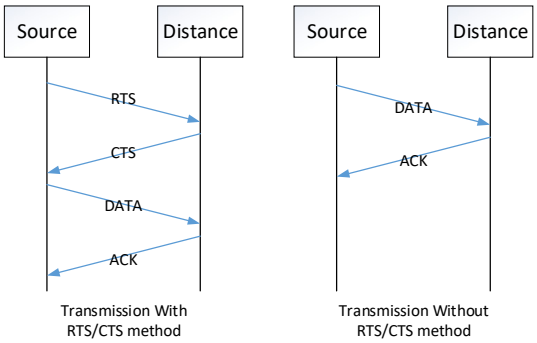
$$P_r = P_t - 20 * \log \frac{4\pi f d}{c}$$

- P_r = Received signal strength [dBm]
- P_t = Transmitter output signal strength [dBm]
- f = frequency [Hz]
- c = speed of light
- d = distance from transmitter



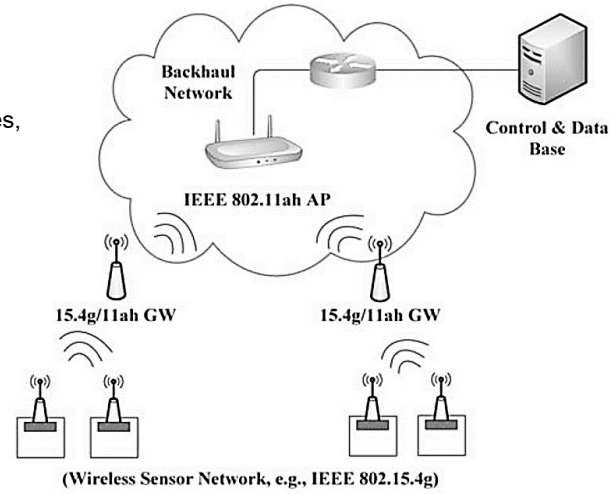
IEEE 802.11ah

- The basic access to the medium in IEEE 802.11ah is using the DCF based on CSMA/CA scheme.
- IEEE 802.11 MAC defined two Coordination Function :
  - Distributed Coordination Function (DCF)
  - Point Coordination Function (PCF)
- The DCF scheme can use two different mechanisms:
  - two-way handshaking technique known as basic access mechanism.
  - four-way handshaking technique known as RTS/CTS method.

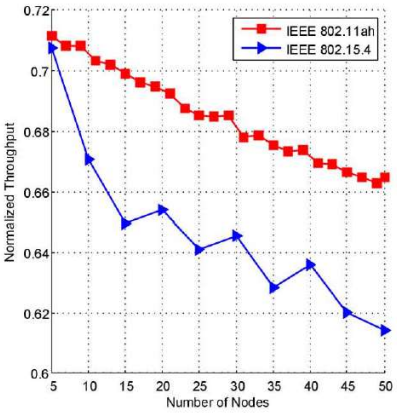


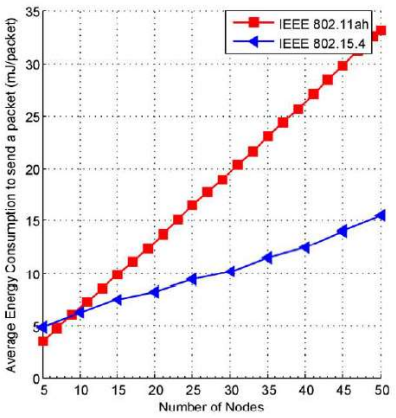
IEEE 802.11ah

- IEEE 802.11ah use cases, backhaul networks for sensor and meter



Comparison between IEEE 802.11ah and IEEE 802.15.4





Wireless Neighborhood Area Network (WNAN)

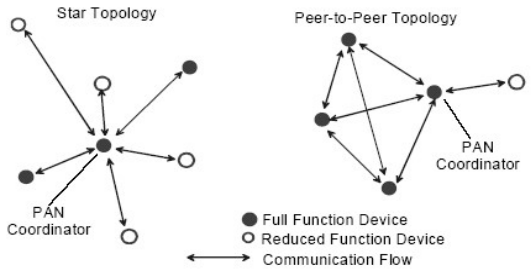
6LoWPAN

Overview

- *6LoWPAN* is an acronym of IPv6 over Low power Wireless Personal Area Networks.
- A simple low throughput wireless network comprising typically low cost and low power devices.
- Common topologies include – star, mesh, and combinations of star and mesh.
- The Phy and MAC layers *conform* to IEEE 802.15.4-2003 standard

LoWPAN - A different *beast* of networks compared to traditional networks !

LoWPAN architecture



Octets: 2	1	0/2	0/2/8	0/2	0/2/8	variable	2
Frame control	Sequence number	Destination PAN identifier	Destination address	Source PAN identifier	Source address	Frame payload	FCS
		Addressing fields					
MHR						MAC payload	MFR

8 July 2018

6LoWPAN

Characteristics

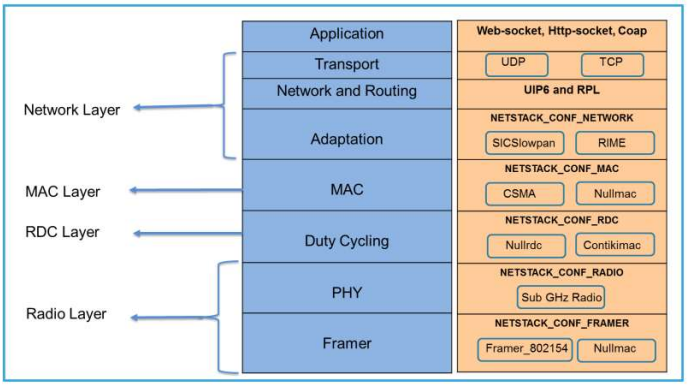
- Small packet size (127Byte)
- 16-bit short or IEEE 64-bit extended media access control addresses
- Low bandwidth. (250kbps)
- Low power, typically battery operated
- Relatively low cost

8 July 2018

6LoWPAN

6LoWPAN Supports:

1. Header Compression
2. Fragmentation and Re-assembly
3. Routing
4. Neighbor discovery and Multicast support



IP Stack Architecture in Contiki Based 6LoWPAN Systems.

8 July 2018

Problem of LoWPAN

- No method exists to make IP run over IEEE 802.15.4 networks
  - Worst case .15.4 PDU 81 octets, IPv6 MTU requirements 1280 octets
- Stacking IP and above layers “as is” may not fit within one 802.15.4 frame
  - IPv6 40 octets, TCP 20 octets, UDP 8 octets + other layers (security, routing, etc) leaving few bytes for data
- Not all adhoc routing protocols may be immediately suitable for LoWPAN
  - DSR may not fit within a packet, AODV needs more memory, etc
- Current service discovery methods “bulky” for LoWPAN
  - Primarily XML based that needs computing, more memory, etc
- Limited configuration and management necessary
- Security for multi hop needs to be considered

8 July 2018

Challenges of LoWPAN

Impact Analysis	Addressing	Routing	Security	Network management
Low power (1-2 years lifetime on batteries)	Storage limitations, low overhead	Periodic sleep aware routing, low overhead	Simplicity (CPU usage), low overhead	Periodic sleep aware management, low overhead
Low cost (<$10/unit)	Stateless address generation	Small or no routing tables	Ease of Use, simple bootstrapping	Space constraints
Low bandwidth (<300kbps)	Compressed addresses	Low routing overhead	Low packet overhead	Low network overhead
High density (<2-4? units/sq ft)	Large address space – IPv6	Scalable and routable to *a node*	Robust	Easy to use and scalable
IP network interaction	Address routable from IP world	Seamless IP routing	Work end to end from IP network	Compatible with SNMP, etc

Goals

- Define adaptation (frag/reassembly) layer to match IPv6 MTU requirements
- Specify methods to do IPv6 stateless address auto configuration
- Specify/use header compression schemes.
- Specify implementation considerations and best methods of an IPv6 stack
- Methods for meshing on LoWPAN below IP*

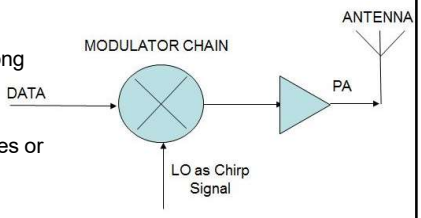
Not currently in charter

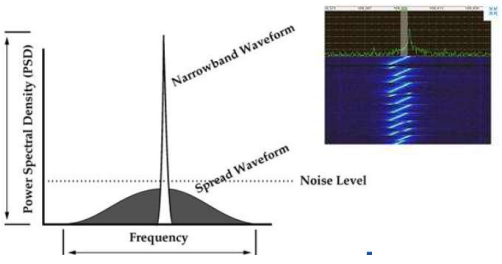
- Use/adapt network management technologies for LoWPANs
- Specify encoding/decoding (or perhaps new protocols) for device discovery mechanisms
- Document LoWPAN security threats

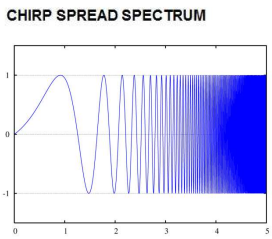
Wireless Wide Area Network (WWAN)

LoRa (Long Range)

- LoRa is the proprietary physical layer or the wireless modulation technique utilized to create long Range Communication Link
- Based on chirp spread spectrum (CSS) modulation
- A single gateway or base station can cover entire cities or hundreds of square kilometers.







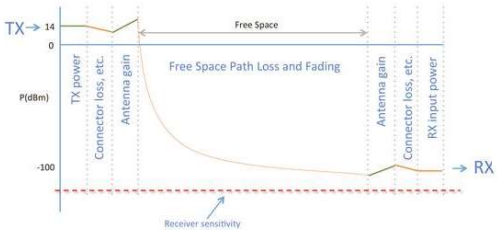
Chirp signal

LoRa (Long Range)

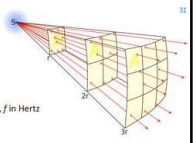
- Operates in the license-free ISM bands all around the world: 433, 868, 915 Mhz
- Data rates are defined that range from 300bps to 5.5kbps
  - ✓ With two high-speed channels at 11kbps and 50kbps (FSK modulation)
- Supports: secure bi-directional communication, mobility and localization

Link budget & Free-space path loss:

Link Budget



Free Space Path Loss



$$FSPL = \left(\frac{4\pi d}{\lambda}\right)^2 = \left(\frac{4\pi df}{c}\right)^2$$ d in meters, f in Hertz

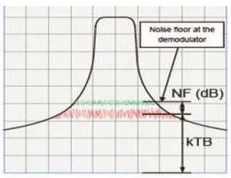
$$FSPL(dB) = 20\log_{10}(d) + 20\log_{10}(f) - 147.55$$

- Double the distance: *6dB* path loss (free space)
- Difference in FSPL between 868Mhz and 2.4Ghz: almost *9dB*

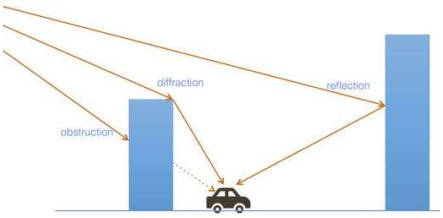
LoRa (Long Range)

- **Multipath fading effect**
  - Free space only exists in free space
  - Fresnel zone: depend on heights of the transmitter and receiver
  - Fading margin
- **Sensitivity**
  - Minimum detectable signal
  - Thermal nose
  - NF: receiver noise figure (in dB)
  - SNR: signal-to-noise ratio

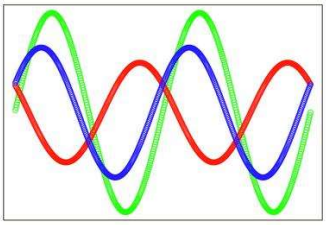
$$Sensitivity(dBm) = -174 + 10\log_{10}(BW) + NF + SNR$$



Wireless reach in reality

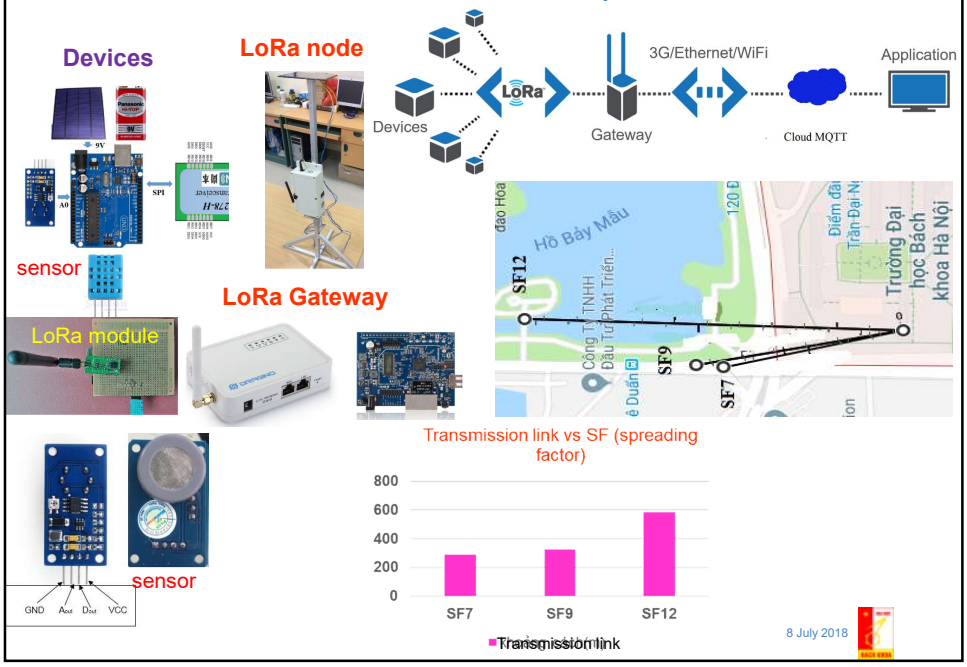


Multipath



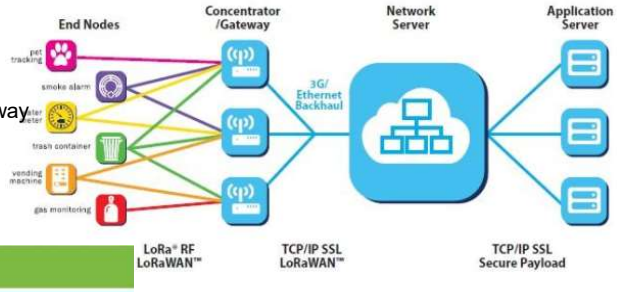
- Direct signal
- Reflected signal
- Combined direct and reflected

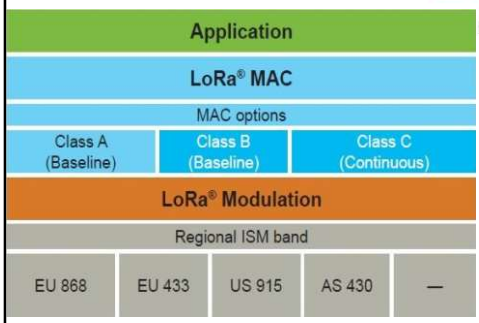
LoRa: Transmission link vs SF experiment



LoRaWAN (Long Range Wide Area Network)

- Star of Stars Topology
- Nodes connect to multiple gateways.
- No handover needed from gateway to gateway
- Nodes are asynchronous and communicate when they have data to send



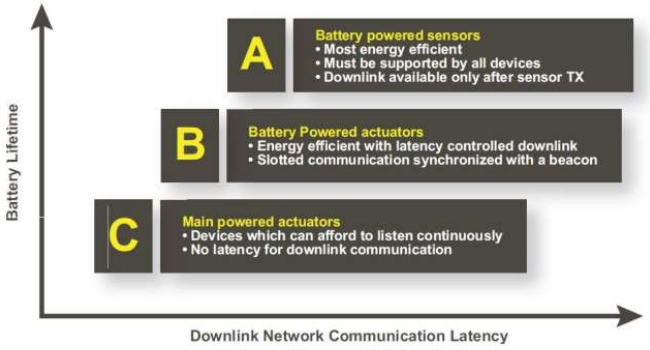


- Device Classes-All nodes are not equal
- Division based on tradeoff between communication latency versus battery life time.
  - Class A -Battery powered sensors
  - Class B- Battery powered actuators
  - Class C- Main powered actuators

8 July 2018

LoRaWAN (Long Range Wide Area Network)

- Class A -Battery powered sensors
- Class B- Battery powered actuators
- Class C- Main powered actuators



8 July 2018

Cellular and mobile network technologies

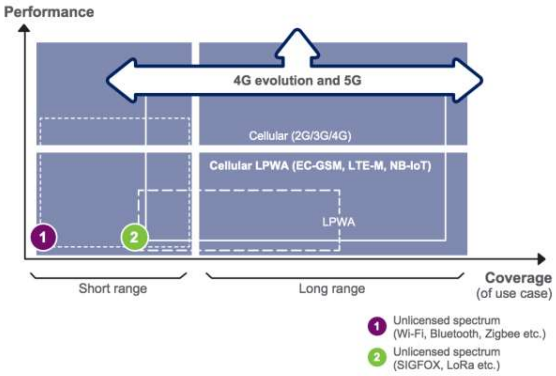
- IoT/M2M applications may find cellular networks to be the practical connectivity technology of choice.
- M2M applications are expected to become important sources of traffic (and revenues) for cellular data networks.
- IoT/M2M traffic has specific characteristics that relate to the priority of the data being communicated, the size of the data, the real-time streaming needs, degrees of mobility.

- Cellular/mobile networks are characterized by varying capacity, bandwidth, link conditions, link utilization, and overall network load, which affect ability to reliably transfer such M2M data.
- Cost-effectively utilize cellular technologies need to consider for a broad set of applications: 3G, LTE,…

8 July 2018

Key performance requirements for the cellular M2M/IoT

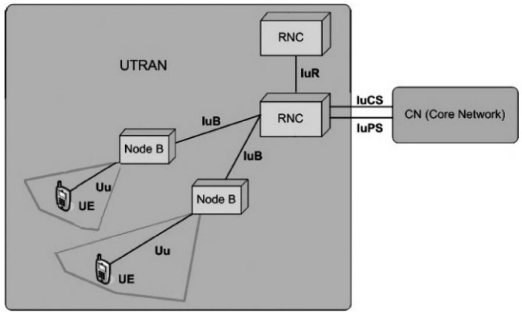
1. Long battery life.
2. Low device cost.
3. Low deployment cost.
4. Extended coverage.
5. Support for a massive number of devices.
6. Acceptable latency.



8 July 2018

Cellular and mobile network technologies

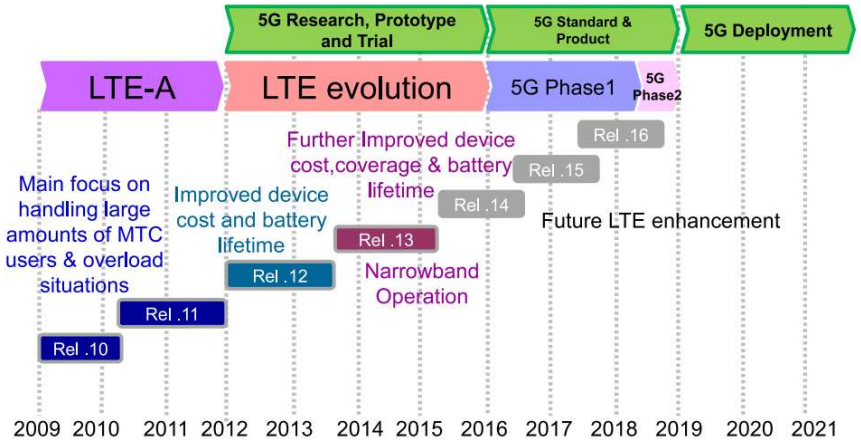
- **Universal Mobile Telecommunications System (UMTS) / 3G**
  - Support voice and data (IP) based on the GSM standard developed by the 3GPP
  - Can carry many traffic types from real-time circuit switched to IP-based packet switched.
  - Can support video over the data (IP) capability, mobility at the PHY level with the MIPv6 mechanisms.
  - The challenge of 3G system is related to bandwidth availability.



8 July 2018

3GPP MTC Standardization

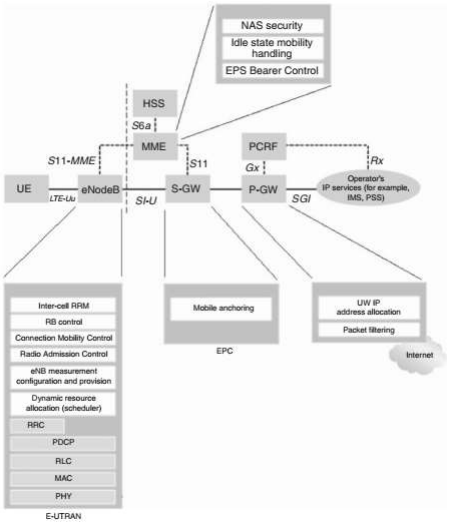
- Machine-Type Communications (MTC): The IoT connectivity solution referred to cellular - based machine-to-machine (M2M).



M. R. Palattella, et.al. **"Internet of Things in the 5G Era: Enablers, Architecture, and Business Models,"** IEEE Journal on Selected Areas in Communications (Volume: 34, Issue: 3, March 2016).

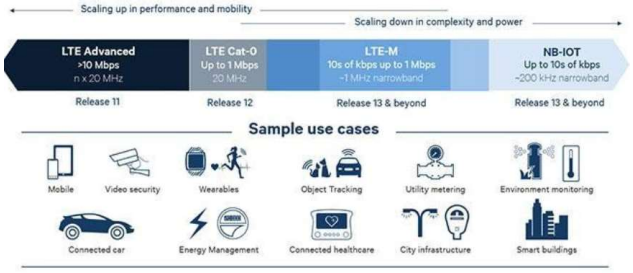
Cellular and mobile network technologies

- **LTE**
  - Aa architecture framework and a set of additional mechanisms providing seamless IP connectivity between UE and the packet (IPv4, IPv6) data network.
  - Support *only* packet-switched services.
  - The key element provided by LTE/SAE is the EPS (evolved packet system).
  - EPS provides the user with IP connectivity to a packet data network for accessing the Internet, as well as for supporting services such as streaming video.



NB-IoT (Narrow band IoT)

- Technology standardized by the 3GPP standards.
- Narrow band Technology designed for IoT which can be deployed in GSM and LTE Spectrum
- NB-IoT technology is an LPWAN standard designed for cellular based IoT: devices require very low data rate and very low battery power in the coverage area of the mobile communication.
- Standardization of NB-IoT completed with Release 13 of 3GPP published on 22 June 2016.
- NB-IoT majorly focuses on low-cost, low battery power, and a large number of things connected in an indoor coverage area. It can be deployed to in-band spectrum which is allocated to long-term evolution (LTE) or stand-alone spectrum.



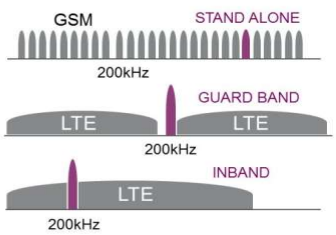
The first NB-IoT shield for Arduino: supported by T-Mobile

NB-IoT (Narrow band IoT)

Transmission schemes:

- Minimum system **bandwidth** for both downlink and uplink – 180 kHz
- GSM carrier frequency of 200 kHz
- 1 PRB (Physical Resource Block) inside an LTE carrier/ guard-band could be replaced by NB-IoT carrier.
- 12 subcarriers of 15 kHz in downlink using OFDM and 3.75/15 kHz in uplink using SC-FDMA.

NB-IoT

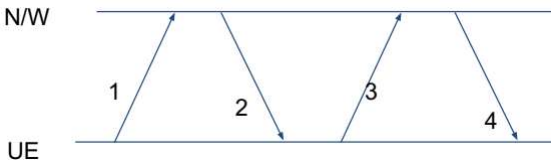


- Highest modulation scheme: QPSK
- Minimal noise level inside a single narrowband
- Receiver does not require to add processing gain to decode
- Simple and inexpensive transceiver design

REDUCED POWER AND LOW COST

NB-IoT (Narrow band IoT)

- Physical Channel/ Resource mapping:
  - Extensive reuse of current LTE (Long Term Evolution) specifications
  - Few changes to physical channels used in LTE
- Random Access :
  - **Contention** based algorithm similar to LTE [Ref]



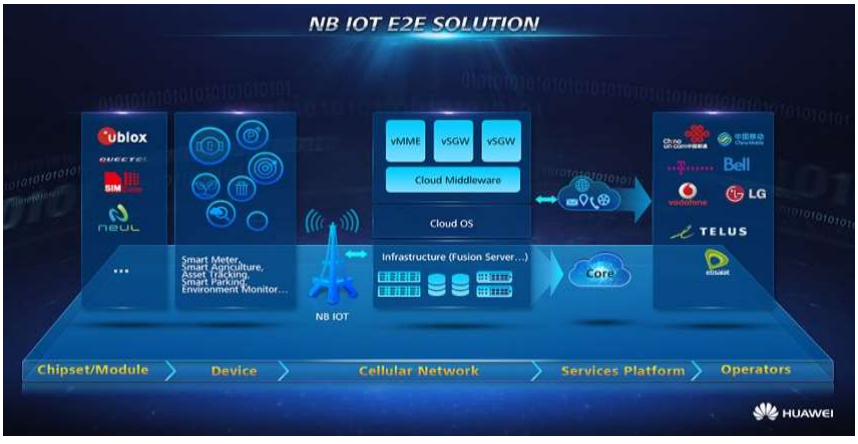
1. Preamble
2. Response containing advance command and scheduling of the uplink resources for the UE to use
3. Identity to the network
4. Contention resolution message.

- To enable low-complexity UE implementation, NB-IoT allows only one **HARQ** process in both downlink and uplink, and allows longer UE decoding time.
- Asynchronous, adaptive HARQ procedure is adopted to support scheduling flexibility.

[Ref] *A Primer on 3GPP Narrowband Internet of Things (NB-IoT)* IEEE Communications Magazine (Volume: 55, Issue: 3, March 2017)

8 July 2018

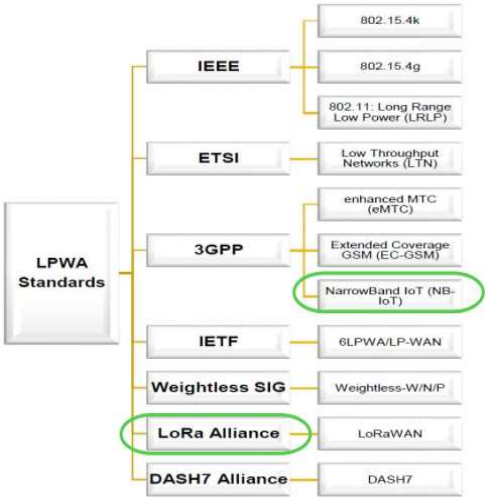
NB-IoT (Narrow band IoT)



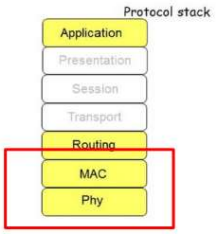
Source: NB-IoT, Accelerating Cellular IoT, Huawei

8 July 2018

LoRa vs. NB-IoT



LPWA standards and their developing organizations



- **LoRa / LoRaWAN**
  - Set up by industrial consortia
  - Unlicensed
- **NB-IoT : Narrow band IoT**
  - Set up by 3gpp
  - licensed

8 July 2018

LoRa vs. NB-IoT

	SIGFOX	LoRa	clean slate cIoT	NB LTE-M Rel. 13	LTE-M Rel. 12/13	EC-GSM Rel. 13	5G (targets)
Range (outdoor) MCL	<13km 160 dB	<11km 157 dB	<15km 164 dB	<15km 164 dB	<11km 156 dB	<15km 164 dB	<15km 164 dB
Spectrum Bandwidth	Unlicensed 900MHz 100Hz	Unlicensed 900MHz <500kHz	Licensed 7-900MHz 200kHz or dedicated	Licensed 7-900MHz 200kHz or shared	Licensed 7-900MHz 1.4 MHz or shared	Licensed 8-900MHz 2.4 MHz or shared	Licensed 7-900MHz shared
Data rate	<100bps	<10 kbps	<50kbps	<150kbps	<1 Mbps	10kbps	<1 Mbps
Battery life	>10 years	>10 years	>10 years	>10 years	>10 years	>10 years	>10 years
Availability	Today	Today	2016	2016	2016	2016	beyond 2020

8 July 2018

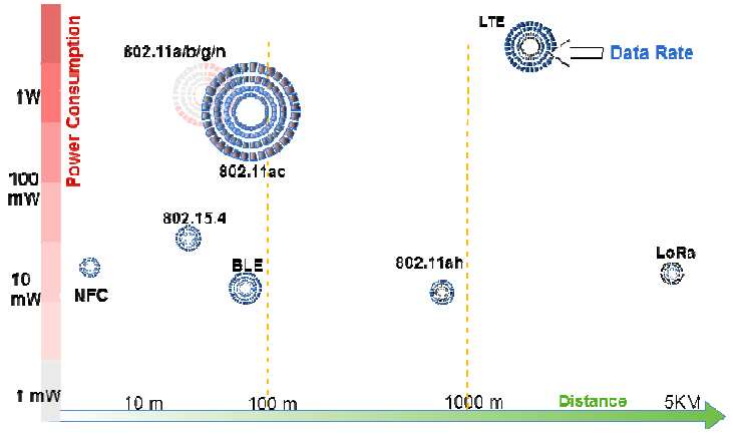
LoRa vs. NB-IoT

Feature	LoRa	NB-IoT
Licensed/Unlicensed Spectrum	Unlicensed Band	Licensed Band
Reuse of Cellular Network	No	Yes
Development Status	Existing	Yet to develop
Modulation	SS chirp	QPSK
Bandwidth	500 Hz - 125 KHz	180 KHz
Data Rate	290 bps- 50 kbps	250 kbps max
Device cost/ complexity	1-5 $ (Ref- LPWA survey)	< 5$ per module (Ref-IETF)
Latency and Battery Lifetime	> 10 years	<10 seconds, >10 years battery (Ref-IETF)
Type of Standard	Proprietary	open

8 July 2018

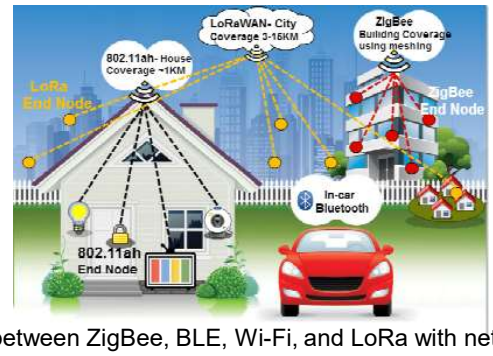
Capacity vs. IoT Requirement

Power Consumption, Distance Coverage in Meters, and Data Rate



8 July 2018

Network Size Capabilities for IoT Networks



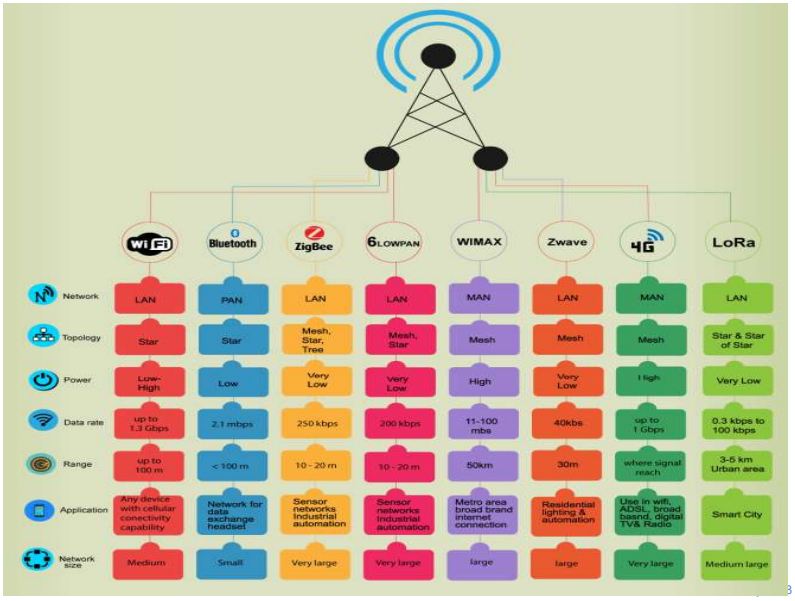
Wireless Network Size Comparison

Comparison between ZigBee, BLE, Wi-Fi, and LoRa with network sizes.

Technology	Network Size
ZigBee	Approximately up to 65,000 nodes
Bluetooth	Eight nodes per network/piconet
Wi-Fi (802.11a/ac)	2007 associated with an AP
Wi-Fi 802.11ah	Approximately 8000 nodes
LoRa	LoRa gateway must have a very high capacity or capability to receive messages from a very high volume of End nodes

8 July 2018

Wireless Technologies Comparison for IoT Networks



M2M Communication Protocols

M2M/IoT Communication Protocols

- TCP/IP Review
- HTTP
- MQTT
- CoAP
- XMPP
- AMQP
- RESTful API

Communication Protocols

TCP/IP Revisited

What is a Protocol ?

Q:- So what is a protocol, anyway?

Communication Protocols

Definition

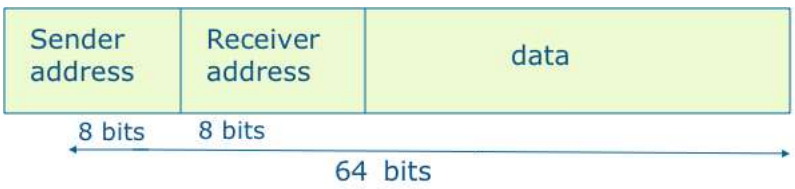
- Protocol is a set of rules that govern all aspect of data communication between computers on a network.
- These rules include guidelines that regulate the following characteristics of a network: access method, allowed physical topologies, types of cabling, and speed of data transfer.
- A protocol defines what, how, when it communicated.
- The key elements of a protocol are syntax, semantics and timing.
- Protocols are to computers what language is to humans. Since this article is in English, to understand it you must be able to read English. Similarly, for two devices on a network to successfully communicate, they must both understand the same protocols.

Key Elements of protocol

i) **Syntax**

The structure or format of the data.

Eg. A simple protocol;



Elements of protocol

ii) Semantics

- Refers to the meaning of each section of bits.
- how is a particular pattern to be interpreted, and what action is to be taken based on that interpretation.

Eg. Does an address identify the route to be taken or the final of the message?

Elements of protocol

iii) Timing

Refers to two characteristics:

a. When data to be sent
b. How fast it can be sent

Eg. If a sender produces data at 100 Mbps but the receiver can process data at only 1 Mbps, the transmission will overload the receiver and data will be largely lost.

Characteristics of protocol

a) **Direct / Indirect**

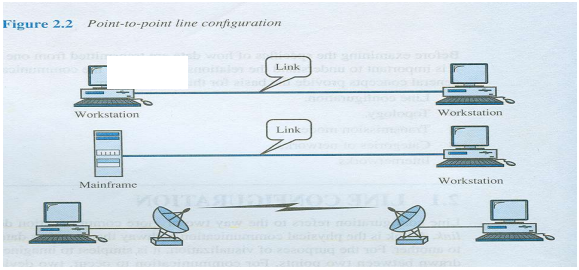
- communication between two entities maybe direct or indirect.

i) point-to-point link

- connection provides a dedicated link between two devices
- the entities in these systems may communicate directly that is data and control information pass directly between entities with no intervening active agent.

8 July 2018

Characteristics of protocol



ii) multipoint link

- connection more than two devices can share a single link
- The entities must be concerned with the issue of access control and making the protocol more complex.

8 July 2018

Common protocol used

Protocol	Acronym	Remarks
Point To Point	PPP	Used to manage network communication over a modem
Transfer/Transmission Control Protocol	TCP / IP	Backbone protocol. The most widely used protocol.
Internetwork package exchange	IPX	Standard protocol for Novell NOS
NetBIOS extended user interface	NetBEUI	Microsoft protocol that doesn't support routing to other network. Running only Windows-based clients.
File transfer Protocol	FTP	used to send and received file from a remote host
Simple mail Transfer protocol	SMTP	Used to send Email over a network
Hyper text transfer protocol	HTTP	Used for Internet to send document that encoded in HTML
Apple Talk	Apple Talk	Protocol suite to network Macintosh computer and a peer-to-peer network protocol
OSI Model	OSI Layers	A way of illustrating how information functions travels through network of its 7 layers.

8 July 2018

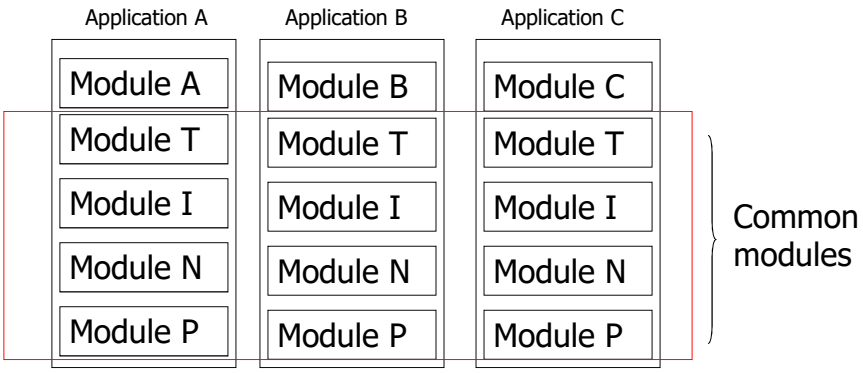
Key Elements of a Protocol

- Syntax – data block format
- Semantics - control info. & error handling
- Timing - speed matching & sequencing

8 July 2018

(Cont.)

- Most of the network apps share some common modules



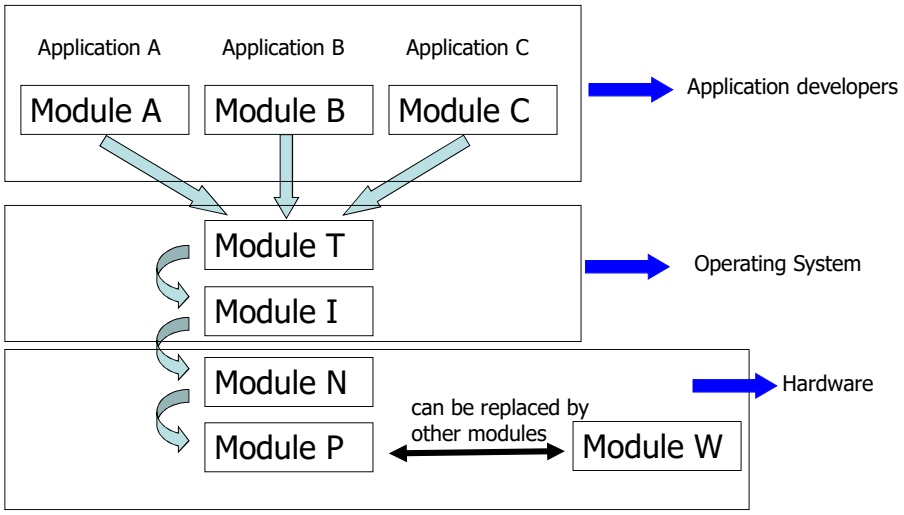
8 July 2018

(Cont.)

- A complex task is broken into subtasks: modular design
  - Each subtask is implemented separately as a layer, arranged in a vertical stack
- Each layer performs a related subset of the functions required to communicate with another system.
  - It relies on the next lower layer to perform more primitive functions and to conceal the details of those functions.
  - It provides services to the next higher layer.
  - Layers should be defined so that changes in one layer do not require changes in other layers.
- So, instead of using a single complex protocol, it's more flexible to implement a stack of protocols!
  - Reduce the design and development workload significantly!

8 July 2018

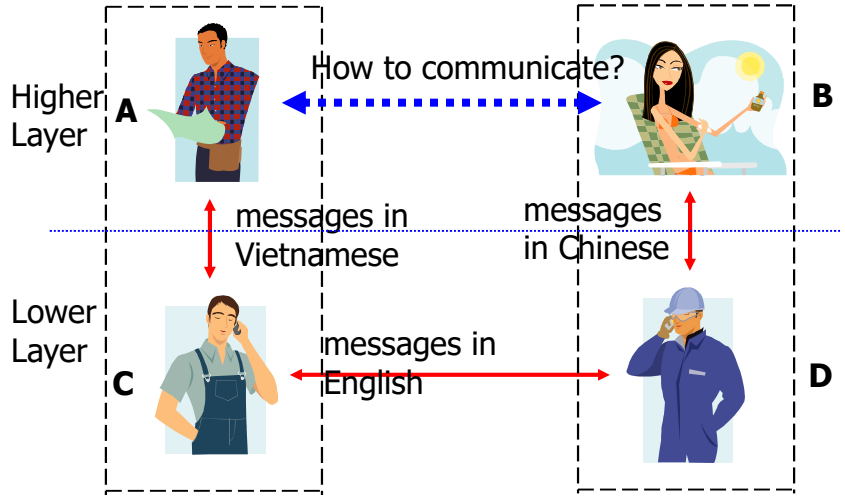
Vertical Stack



Example

- E.g., A can only speak Vietnamese, B can only speak Chinese, how can A communicate with B?
  - A finds a translator C, who can speak Vietnamese and English
  - B finds a translator D, who can speak Chinese and English
- Two layers:
  - Higher layer: A and B
  - Lower layer: C and D

A Two-layer example



Lower layer provides services to the next higher layer.

8 July 2018

Protocol Architecture

- Tasks of communications are broken up into modules
  - Each module (or layer) can have its own protocol
- In very general terms, communications can be said to involve three components: applications, computers, and networks.
- For example, file transfer could use three modules (or layers)
  - File transfer application
  - Communications service module
  - Network access module
- The stack of protocols is called “Protocol Stack”
  - Or Protocol Architecture

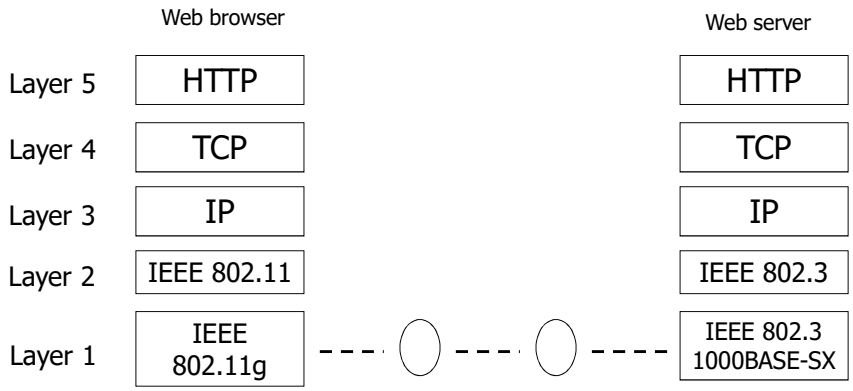
8 July 2018

TCP/IP Protocol Architecture

- Developed by the US Defense Advanced Research Project Agency (DARPA) for its packet switched network (ARPANET)
- Used by the global Internet
- It consists of a large collection of protocols that have been issued as Internet standard by the Internet Architecture Board (IAB).
  - Check http://www.ietf.org/rfc/rfc2026.txt
- The TCP/IP protocol architecture organizes the communication task into five relatively independent layers:
  - Layer 5: Application layer
  - Layer 4: Transport layer, or Host to host (TCP belongs to this layer)
  - Layer 3: Internet layer, or Network layer (IP belongs to this layer)
  - Layer 2: Network access layer, or Link layer
  - Layer 1: Physical layer
- Remark: Each layer can have lots of different protocols!

Example

- World Wide Web
  - Replies on the HTTP protocol

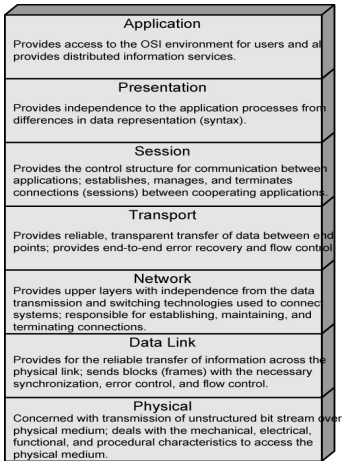


Standardized Protocol Architectures

- Required for devices to communicate
- Vendors have more marketable products
- Customers can insist on standards based equipment
- Two standards:
  - OSI Reference model
    - Never lived up to early promises
  - TCP/IP protocol suite
    - Most widely used

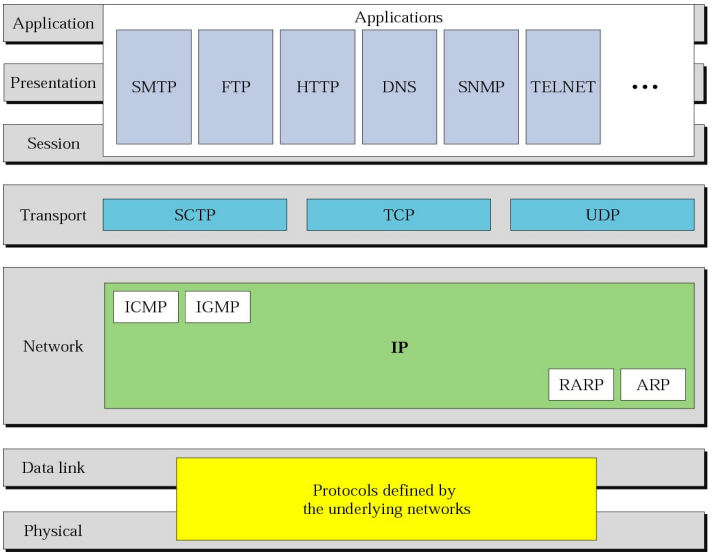
Protocol Architecture

Q:- What is a protocol architecture?



Ans:- The software structure that implements the communications function. Typically, the protocol architecture consists of a layered set of protocols, with one or more protocols at each layer.

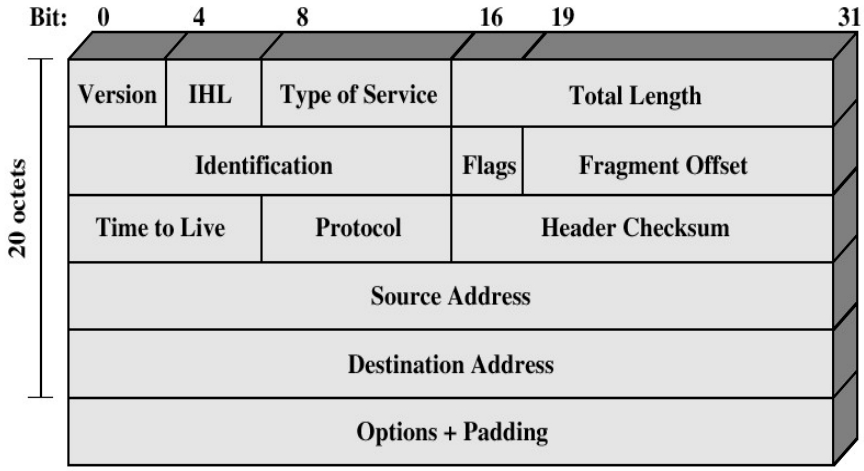
Basics: TCP/IPv4 Protocol Suite



Internet Protocol (IP)

- Features:
  - Layer 3 (Network layer)
  - Unreliable, Connectionless, Datagram
  - Best-effort delivery
- Popular version: IPv4
- Major functions
  - Global addressing
  - Datagram lifetime
  - Fragmentation & Reassembly

IPv4 Header



IPv4 Companion Protocols (1)

- ARP: Address Resolution Protocol
  - Mapping from IP address to MAC address
- ICMP: Internet Control Message Protocol
  - Error reporting & Query
- IGMP: Internet Group Management Protocol
  - Multicast member join/leave
- Unicast Routing Protocols (Intra-AS)
  - Maintaining Unicast Routing Table
  - E.g. RIP, OSPF (Open Shortest Path First)

IPv4 companion protocols (2)

- Multicast Routing Protocols
  - Maintaining Multicast Routing Table
  - E.g. DVMRP, MOSPF, CBT, PIM
- Exterior Routing Protocols (Inter-AS)
  - E.g. BGP (Border Gateway Protocol)
- Quality-of-Service Frameworks
  - Integrated Service (ISA, IntServ)
  - Differentiated Service (DiffServ)

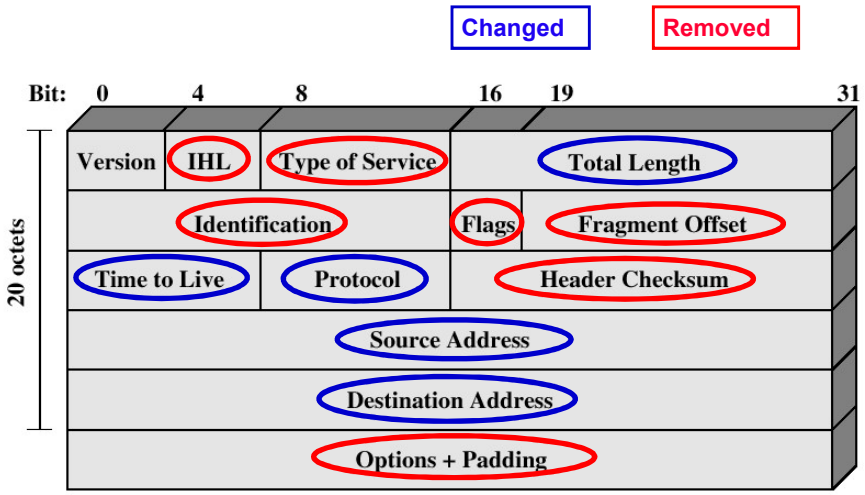
Why IPv6?

- Deficiency of IPv4
- Address space exhaustion
- New types of service → Integration
  - Multicast
  - Quality of Service
  - Security
  - Mobility (MIPv6)
- Header and format limitations

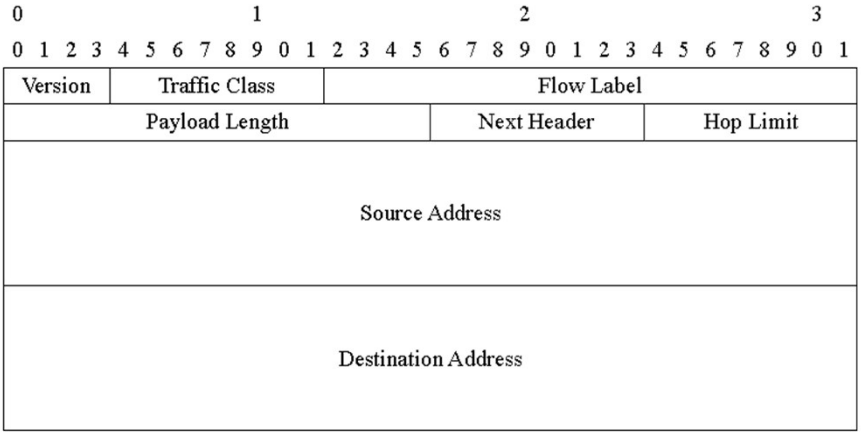
Advantages of IPv6 over IPv4

- Larger address space
- Better header format
- New options
- Allowance for extension
- Support for resource allocation
- Support for more security
- Support for mobility

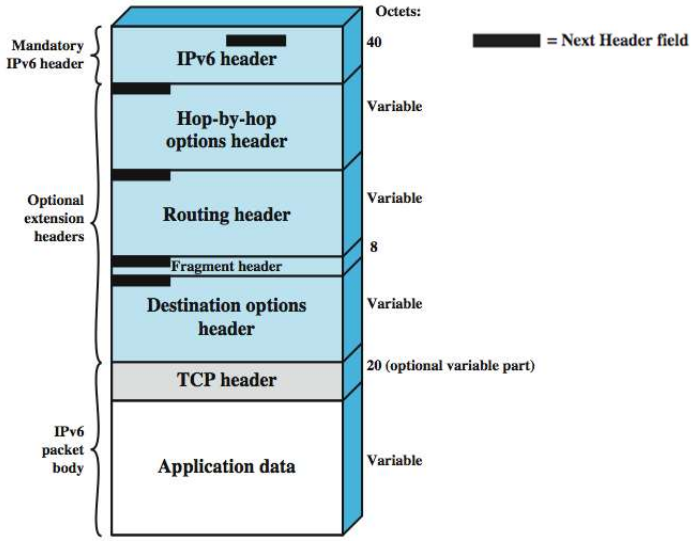
Header: from IPv4 to IPv6



IPv6 Header Format



IPv6 Packet (PDU) Structure



Traffic Class

- The 8-bit field in the IPv6 header is available for use by originating nodes and/or forwarding routers to identify and distinguish between different classes or priorities of IPv6 packets.
  - E.g., used as the codepoint in DiffServ
- General requirements
  - Service interface must provide means for upper-layer protocol to supply the value of traffic class
  - Value of traffic class can be changed by source, forwarder, receiver
  - An upper-layer protocol should not assume the value of traffic class in a packet has not been changed.

IPv6 Flow Label

- Related sequence of packets
- Needing special handling
- Identified by src & dest addr + flow label
- Router treats flow as sharing attributes
  - E.g. path, resource allocation, discard requirements, accounting, security
- May treat flows differently
  - Buffer sizes, different forwarding precedence, different quality of service
- Alternative to including all info. in every header

Payload Length

- 16-bit unsigned integer. Length of the IPv6 payload, i.e., the rest of the packet following this IPv6 header, in octets.
- Note that any extension headers present are considered part of the payload, i.e., included in the length count.

Extension Header Order

Order	Header Type	Next Header Code
1	Basic IPv6 Header	
2	Hop-by-Hop Options	0
4	Routing header	43
5	Fragment header	44
6	Authentication header	51
7	Encapsulation Security Payload header	50
8	Destination Options	60
9	Mobility header	135
	No Next header (Null)	59
	Upper layer: TCP, UDP, ICMP	6, 17, 58

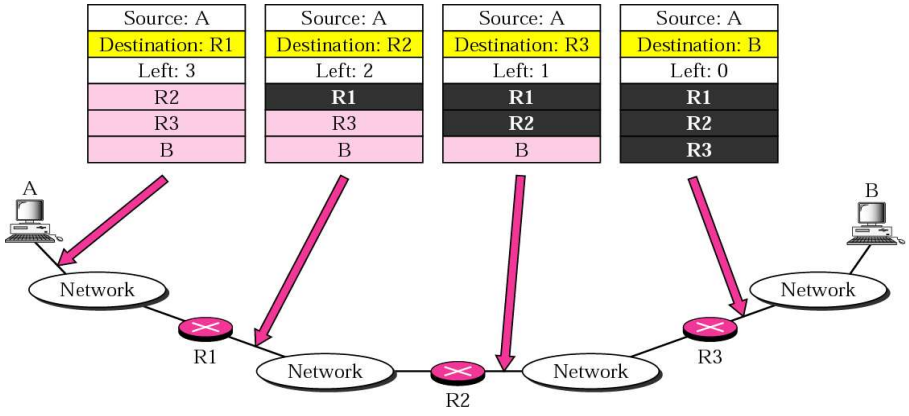
Hop-by-Hop Options

- Must be examined by every router
  - Specifies discard/forward handling
- Options
  - Pad1
  - PadN
  - Jumbo payload
  - Router alert (can be used for RSVP)

Routing Header

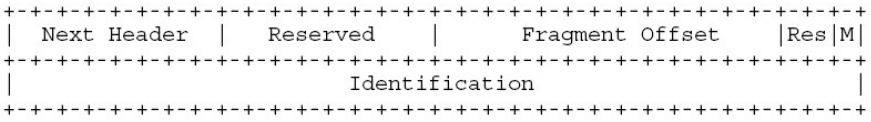
- List of one or more intermediate nodes to visit
- Header includes
  - Next Header
  - Header extension length
  - Routing type (e.g. type 0 = Source Routing)
  - Segments left

Source Routing Example



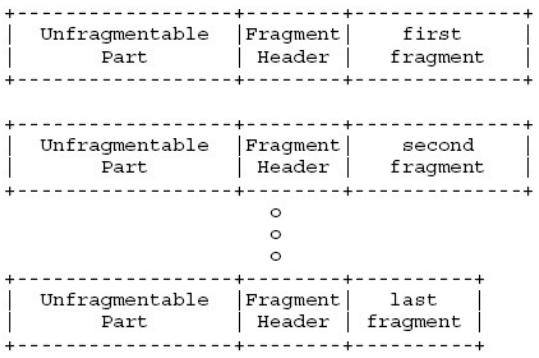
Fragment Header (1)

- Fragment Offset: 8-bit unsigned integer
  - The offset, in 8-octet units, of the data following this header, relative to the start of the Fragmentable Part of the original packet
  - Unfragmentable part: IPv6 header + any extension headers that must be processed by nodes en route



Fragment Header (2)

- M flag: 1=more fragments, 0=last fragment
- Identification: combined with the src & dest addr uniquely identifies the original packet



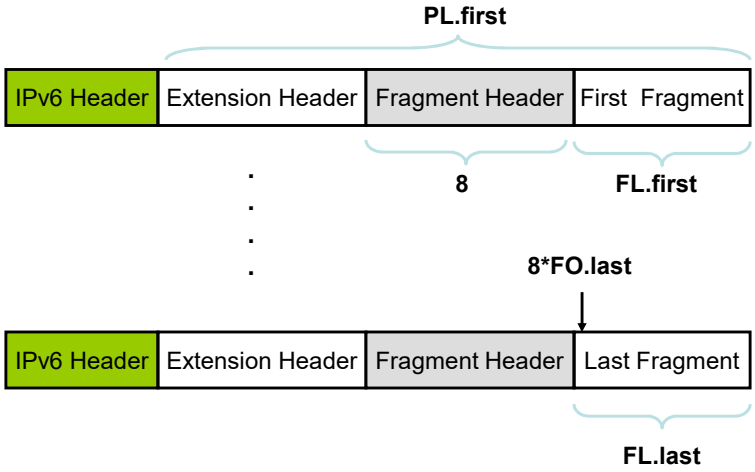
Reassembly (1)

- The Unfragmentable Part of the reassembled packet consists of all headers up to, but not including, the Fragment header of the first fragment packet
- The Next Header field of the last header of the Unfragmentable Part is obtained from the Next Header field of the first fragment's Fragment header
- The Payload Length of the reassembled packet is computed from the length of the Unfragmentable Part and the length and offset of the last fragment.

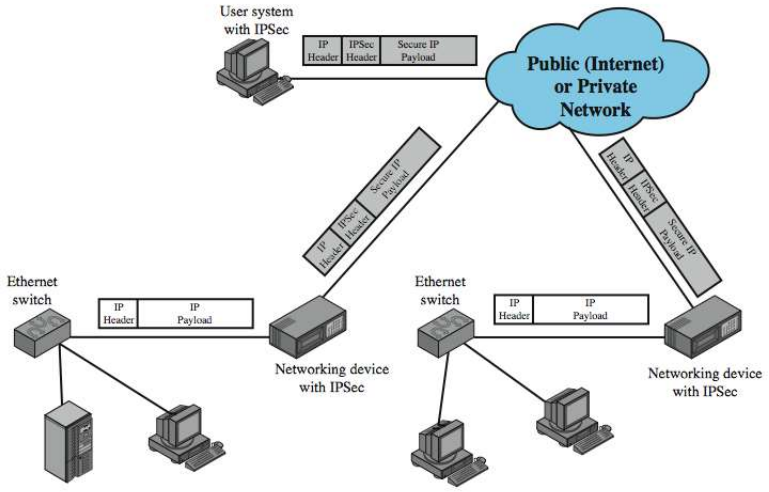
Reassembly (2)

- PL.orig = PL.first - FL.first - 8 + (8 * FO.last) + FL.last
  - PL.orig = Payload Length field of reassembled packet.
  - PL.first = Payload Length field of first fragment packet.
  - FL.first = length of fragment following Fragment header of first fragment packet.
  - FO.last = Fragment Offset field of Fragment header of last fragment packet.
  - FL.last = length of fragment following Fragment header of last fragment packet.

Reassembly (3)



IPsec Scenario



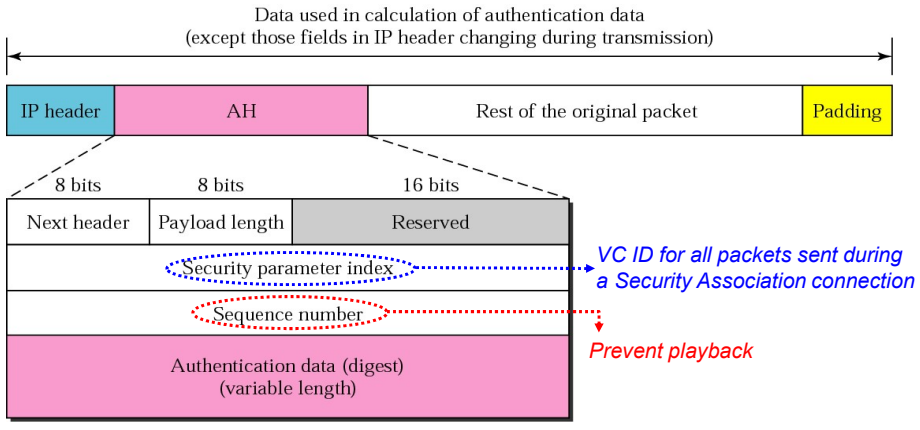
IPSec Benefits

- Provides strong security for external traffic
- Resistant to bypass
- Below transport layer hence transparent to applications
- Can be transparent to end users
- Can provide security for individual users if needed

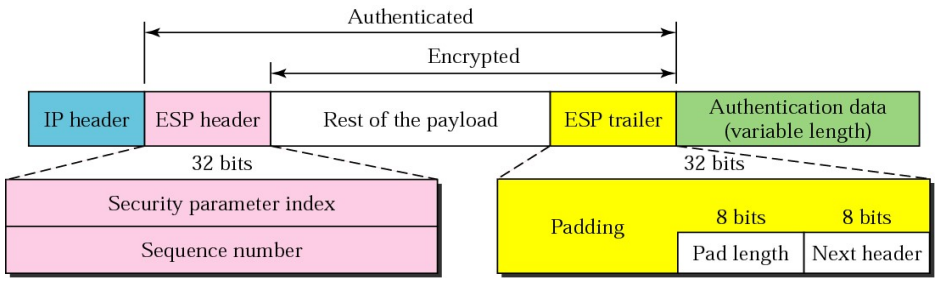
IPSec Functions

- Authentication Header
  - for authentication/integrity only
- Encapsulating Security Payload (ESP)
  - for authentication/integrity/encryption (privacy)
- A key exchange function
  - Manual or automated
- VPNs usually need combined function

Authentication Header



Encapsulating Security Payload



IPv6 Addressing Model

- Addresses are assigned to interfaces, not hosts
- Interface expected to have multiple addresses
- Addresses have scope
  - Link-Local
  - Site-Local → Unique Local
  - Global

Text Representation of Address

- Colon-Hex
  - 3ffe:3600:2000:0800:0248:54ff:fe5c:8868
- Compressed Format:
  - 3ffe:0b00:0c18:0001:0000:0000:0000:0010
  - becomes 3ffe:b00:c18:1::10

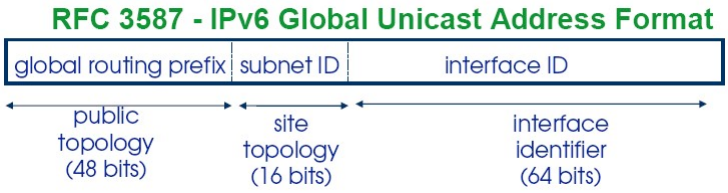
Address Type Prefixes

Address type	Binary prefix
IPv4-compatible	0000...0 (96 zero bits)
global unicast	001
link-local unicast	1111 1110 10
site-local unicast	1111 1110 11
multicast	1111 1111

- all other prefixes reserved (approx. 7/8ths of total)
- anycast addresses allocated from unicast prefixes

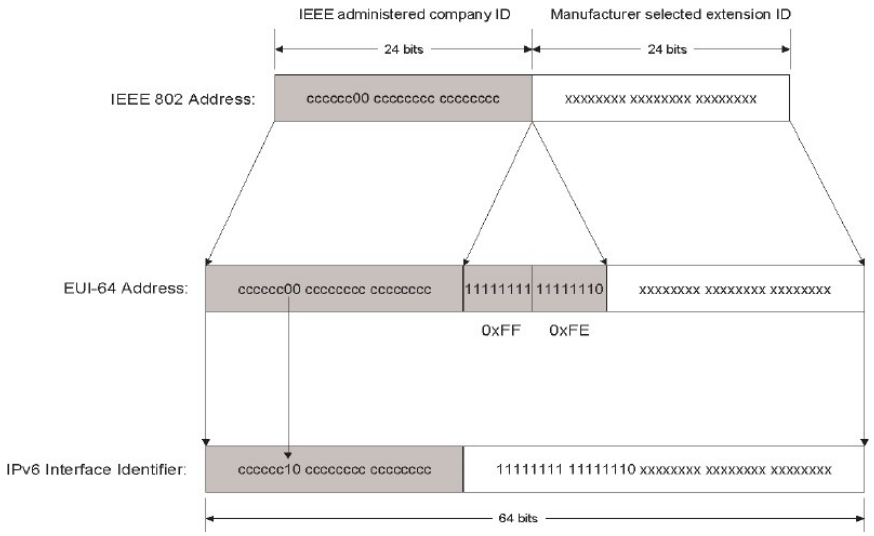
Global Unicast Address

- Global routing prefix
  - A (typically hierarchically-structured) value assigned to a site (a cluster of subnets/links)
- Subnet ID
  - An identifier of a subnet within the site
- Interface ID
  - Constructed in Modified EUI-64 format



8 July 2018

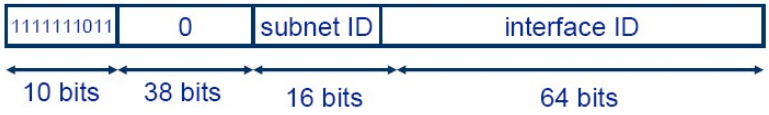
IEEE 802 → IPv6 Interface ID



8 July 2018

Site-Local Address

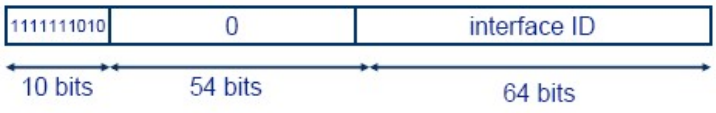
- Meaningful only in a single site zone, and may be re-used in other sites
- Equivalent to the IPv4 private address space
- Address are not automatically configured and must be assigned
- Prefix= FEC0::/48



8 July 2018

Link-Local Address

- Meaningful only in a single link zone, and may be re-used on other links
- Link-local addresses for use during auto-configuration and when no routers are present
- Required for Neighbor Discovery process, always automatically configuration
- An IPv6 router never forwards link-local traffic beyond the link
- Prefix= FE80::/64



8 July 2018

Special IPv6 Address

- Loopback address (0:0:0:0:0:0:0:1 or ::1)
  - Identify a loopback interface
- IPv4-compatible address (0:0:0:0:0:0:w.c.x.z or ::w.c.x.z)
  - Used by dual-stack nodes
  - IPv6 traffic is automatically encapsulated with an IPv4 header and send to the destination using the IPv4 infrastructure
- IPv4 mapped address (0:0:0:0:0:FFFF:w.c.x.z or ::FFFF:w.c.x.z)
  - Represent an IPv4-only node to an IPv6 node
  - Only use a single listening socket to handle connections from client via both IPv6 and IPv4 protocols.
  - Never used as a source or destination address of IPv6 packet
  - Rarely implemented

8 July 2018 

Address Autoconfiguration (1)

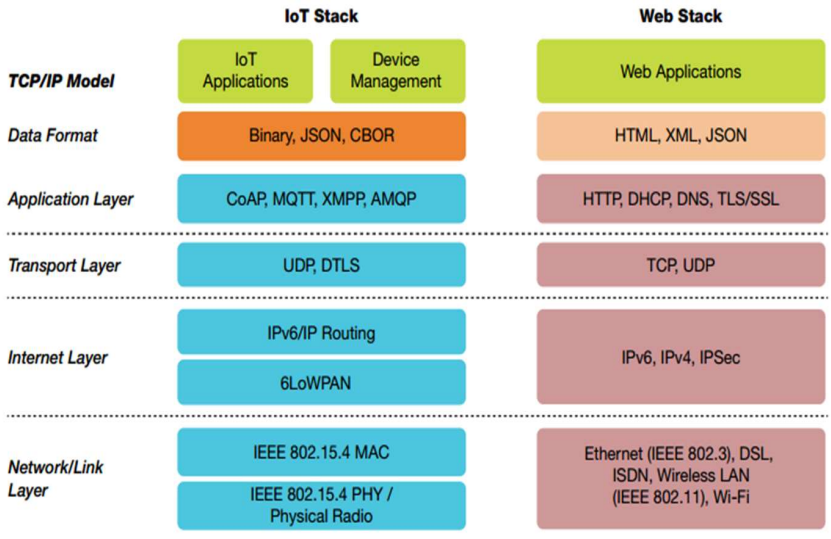
- Allow plug and play
- BOOTP and DHCP are used in IPv4
- DHCPng will be used with IPv6
- Two Methods: Stateless and Stateful
- Stateless:
  - A system uses link-local address as source and multicasts to "All routers on this link"
  - Router replies and provides all the needed prefix info
  - All prefixes have a associated lifetime
  - System can use link-local address permanently if no router

8 July 2018 

Address Autoconfiguration (2)

- Stateful:
  - Problem w/ stateless: Anyone can connect
  - Routers ask the new system to go DHCP server (by setting managed configuration bit)
  - System multicasts to "All DHCP servers"
  - DHCP server assigns an address

IoT Protocols Stack vs. TCP/IP Stack



M2M Commnnication Protocols: what is it?

- Application layer protocols that facilitate the transfer of messages between two devices.
- An important distinction between connectivity and communication protocols is that M2M connectivity protocols facilitated linking of the devices and reside on the link layer, while M2M communication protocols facilitate messaging between devices and reside on the application layer. These protocols allow devices to inter-operate with each other. They do this by standardizing the key aspects of naming, messaging, and controls.
  - Standardized naming means defining how each device will be referred, recognized, or found.
  - Standard messaging determines how each message is formatted, so that all the devices can understand it.
  - Standard controls enable the devices to control the flow of communication or quality of service, or to define the type of a message.
- **Request-Response Model**

  In this model, the client node sends a request to the central server node to initiate a connection, and the server responds in a predefined pattern depending on the protocol.
- **Publish-Subscribe Model**

  All the clients during or after connection establishment define what type of messages they are interested in, generally called topics or classes. Whenever a client, a publisher sends a message of a specific topic, the message is forwarded to all those clients or subscribers who have expressed interest in the topic in question.

8 July 2018

Potential Protocols

Features: Constrained Application Protocol

Attribute	Support
Communication Pattern	PubSub & Req-Resp
Transport Protocol	UDP
Reliability/QoS	Yes
Security	Reslies on DTLS
Distributed or Operation/Centralized	Decentralised
Service/Node discovery	YES
Messages and Payload Support	Text, XML

Features: Hypertext Transport Protocol

Attribute	Support
Communication Pattern	Request-Response
Transport Protocol	TCP
Reliability/QoS	NIL, Relies on TCP.
Security	NIL, SSL/TLS
Distributed Operation/Centralized	Centralised
Service/Node discovery	YES
Messages and Payload Support	TXT, XML, HTML etc. Etc.

Features: Extensible Messaging and Presence Protocol

Attribute	Support
Communication Pattern	PubSub & Req-Resp
Transport Protocol	TCP
Reliability/QoS	Via TCP
Security	TLS/SASL
Distributed Operation/Centralized	Both
Service/Node discovery	Yes
Messages and Payload Support	EXI, XML

Features: Data Distribution Service

Attribute	Support
Communication Pattern	Bus like, Publish Subscribe
Transport Protocol	TCP, UDP
Reliability/QoS	Yes
Security	Yes (DDS Security)
Distributed Operation/Centralized	Distributed
Service/Node discovery	Yes
Messages and Payload Support	Yes, User-defined data structures.

Features: Message Queue Telemetry

Attribute	Support
Communication Pattern	Req-Response
Transport Protocol	TCP
Reliability/QoS	YES
Security	Yes, DTLS
Distributed Operation/Centralized	Centralised
Service/Node discovery	No
Messages and Payload Support	Binary. Payload agnostic.

Features: Advanced Message Queuing Protocol

Attribute	Support
Communication Pattern	Point-to-point, Publish Subscribe
Transport Protocol	TCP
Reliability/QoS	Yes
Security	Yes(TLS, SASL)
Distributed Operation/Centralized	Centralised
Service/Node discovery	No Direct Method.
Messages and Payload Support	Yes. (JSON, XML etc.)

Communication Protocols

HTTP: Hyper Text Transfer Protocol

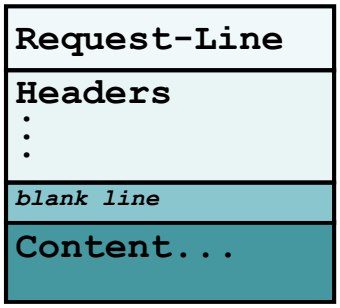
HyperText Transfer Protocol (HTTP)

- HTTP is the protocol that supports communication between web browsers and web servers.
- A “Web Server” is a HTTP server
- Most clients/servers today speak version 1.1, but 1.0 is also in use.
  - RFC 1945 (HTTP 1.0)
  - RFC 2616 (HTTP 1.1)
- “HTTP is an application-level protocol with the lightness and speed necessary for distributed, hypermedia information systems.”
- Transport Independence
  - The HTTP protocol generally takes place over a TCP connection,
  - but the protocol itself is not dependent on a specific transport layer.
- HTTP has a simple structure:
  - client sends a request
  - server returns a reply.
- HTTP can support multiple request-reply exchanges over a single TCP connection.
- The “well known” TCP port for HTTP servers is port 80.
  - Other ports can be used as well...

8 July 2018

HTTP 1.0+ Request

- Lines of text (ASCII).
- Lines end with CRLF `"\r\n"`
- First line is called "Request-Line"



Request Line

***Method URI HTTP-Version*\r\n**

- The request line contains 3 *tokens* (words).
- space characters " " separate the tokens.
- Newline (\n) seems to work by itself
  - but the protocol requires CRLF

Request Method

- The Request Method can be:

GET	**HEAD**	**DELETE**
PUT	**POST**	**TRACE**
		OPTIONS

future expansion is supported

- GET, HEAD and POST are supported everywhere (including Lab 2!).
- HTTP 1.1 servers often support PUT, DELETE, OPTIONS & TRACE.

8 July 2018

Methods

- GET: retrieve information identified by the URI.
  - Typically used to retrieve an HTML document
- HEAD: retrieve meta-information about the URI.
  - used to find out if a document has changed
- POST: send information to a URI and retrieve result.
  - used to submit a form
- PUT: Store information in location named by URI.
- DELETE: remove *entity* identified by URI.
- TRACE: used to trace HTTP forwarding through proxies, tunnels, etc.
- OPTIONS: used to determine the capabilities of the server, or characteristics of a named resource.

URI: Universal Resource Identifier

- URIs defined in RFC 2396.
- Absolute URI:
  - **`scheme://hostname[:port]/path`**
  - **`http://www.cse.unr.edu:80/~mgunes/cpe401`**
- Relative URI:
  - **`/path`**
  - **`/blah/foo`** ← No server mentioned

8 July 2018

The Header Lines

- Request Headers provide information to the server about the client
  - what kind of client
  - what kind of content will be accepted
  - who is making the request
- Each header line contains
  - an attribute name followed by a ":" followed by a space and the attribute value.
- There can be 0 headers (HTTP 1.0)
- HTTP 1.1 requires a `Host:` header

Example HTTP Headers

```
Accept: text/html
Host: www.cse.unr.edu
From: mgunes@cse.unr.edu
User-Agent: Mozilla/4.0
Referer: http://www.unr.edu/
```


Example POST Request

```
POST /~mgunes/cpe401/grades.cgi HTTP/1.1
Accept: */*
Host: www.cse.unr.edu
User-Agent: SecretAgent V2.3
Content-Length: 35
Referer: http://www.unr.edu/

stuid=6660182722&item=test1&grade=99
```

- A POST request includes some *content (some data)* after the headers (after the blank line).
- There is no format for the data (just raw bytes).
- A POST request must include a Content-Length line in the headers:

Content-length: 35

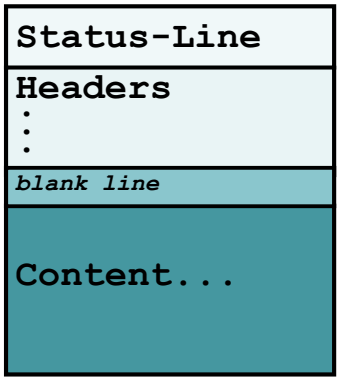
Example GET Request

```
GET /~mgunes/cpe401/lab1.htm HTTP/1.1
Accept: */*
Host: www.cse.unr.edu
User-Agent: Internet Explorer
From: mgunes@cse.unr.edu
Referer: http://www.unr.edu/
```

← There is a blank line here!

HTTP Response

- ASCII Status Line
- Headers Section
- Content can be anything (not just text)
  - typically an HTML document or some kind of image.



Response Status Line

HTTP-Version *Status-Code* *Message*

- *Status Code* is 3 digit number (for computers)
  - **1xx** Informational
  - **2xx** Success
  - **3xx** Redirection
  - **4xx** Client Error
  - **5xx** Server Error
- Message is text (for humans)

8 July 2018

HTTP response message

status line (protocol status code status phrase)

header lines

data, e.g., requested HTML file

```
HTTP/1.1 200 OK
Connection close
Date: Thu, 06 Aug 1998 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Mon, 22 Jun 1998 …...
Content-Length: 6821
Content-Type: text/html

data data data data data ...
```

8 July 2018

HTTP response status codes

In first line in server->client response message.
A few sample codes:

200 OK
- request succeeded, requested object later in this message

301 Moved Permanently
- requested object moved, new location specified later in this message (Location:)

400 Bad Request
- request message not understood by server

404 Not Found
- requested document not found on this server

505 HTTP Version Not Supported

Communication Protocols

MQTT: Message Queuing Telemetry Transport

Overview

- Message Queuing Telemetry Transport (old acronym) since the 1990's
- "Telemetry" is from the Greek remote measure
- Created by Andy Stanford-Clark (IBM) and Alan Nipper - now part of OASIS
- Version 3.1 released royalty free in 2010 (IBM)
- Originally built for oil pipeline monitoring over satellite connections.
- Satellites appropriate because pipelines are remote
- Built for a proprietary imbedded system now shifting to IoT
- You can send anything as a message. Up to 256 MB.
- Built for unreliable networks
- Enterprise scale implementations down to hobby projects
- Decouples readers and writers
- Message have a topic, quality of service and retain status associated with them.

Overview (cont.)

- MQTT Runs over TCP or TLS. May use Websockets from within a browser.
- MQTT–SN uses UDP packets or serial communication rather than TCP. MQTT-SN may run over Bluetooth Low Energy (BLE).
- Open, industry agnostic, no polling. What does it mean to be open?
- Hierarchical topic namespace and subscriptions with wildcards. MQTT-SN has simpler topics.
- As soon as you subscribe you may receive the most recently published message. One message per topic may be retained by the broker. This featuer provides for devices that transmit messages only on occasion. A newly connected subscriber does not need to wait. Instead, it receives the most recent message.

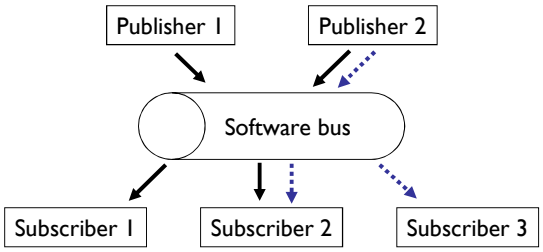
In short:

"MQTT is a Client Server publish/subscribe messaging transport protocol. It is light weight, open, simple, and designed so as to be easy to implement. These characteristics make it ideal for use in many situations, including constrained environments such as for communication in Machine to Machine (M2M) and Internet of Things (IoT) contexts where a small code footprint is required and/or network bandwidth is at a premium."

Publish/Subscribe Mechanism

- Achieved by ***publish/subscribe*** paradigm
  - Idea: Entities can publish data under certain names
  - Entities can subscribe to updates of such ***named data***
- Conceptually: Implemented by a software bus
  - Software bus stores subscriptions, published data; names used as filters; subscribers notified when values of named data changes

- Variations
  - ***Topic-based*** P/S – inflexible
  - ***Content-based*** P/S – use general predicates over named data

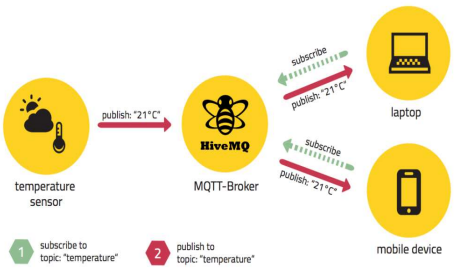


MQTT Operation

- 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
- The use of TCP/IP to provide basic network connectivity
- Three qualities of service for message delivery:
  - "At most once", where messages are delivered according to the best efforts of the underlying TCP/IP network. Message loss or duplication 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.

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

MQTT: Sensor readings with pub/sub



Decoupled in space and time.
The clients do not need each others
IP address and port (space) and
They do not need to be running at
the same time (time).

The broker's IP and port must be known by clients.

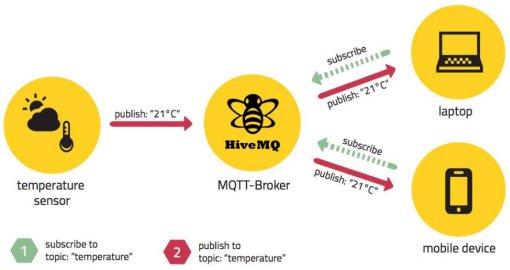
Namespace hierarchy used for topic filtering.

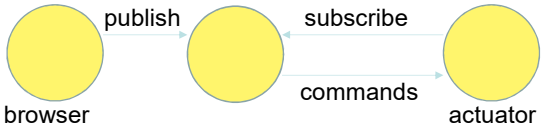
It may be the case that a published
message is never consumed by any
subscriber.

8 July 2018

Actuators too!

In the picture, replace the laptop with an actuator, subscribing to a command topic, say, device3/commands.
Replace the sensor with a browser sending commands to device3/commands.





If my toaster is a command subscriber,
I can control it over the web!

8 July 2018

MQTT Client

- A client may publish or subscribe or do both.
- An MQTT client is any device from a micro controller up to a full fledged server, that has an MQTT library running and is connecting to an MQTT broker over any kind of network. (from HiveMQ MQTT Essentials)
- A client is any device that has a TCP/IP stack and speaks MQTT. MQTT-SN does not require TCP.
- Client libraries widely available (Android, Arduino, iOS, Java, Javascript, etc.)
- No client is connected directly to any other client

MQTT Broker

- The broker is primarily responsible for receiving all messages, filtering them, decide who is interested in it and then sending the message to all subscribed clients. (From HiveMQ MQTT Essentials)
- May authenticate and authorize clients.
- Maintains sessions and missed messages
- Maintains a hierarchical namespace for topics and allows subscribers (but not publishers) to use wildcards (+ and #).

MQTT Message Format

- The message header for each MQTT command message contains a fixed header.
- Some messages also require a variable header and a payload.
- The format for each part of the message header:

bit	7	6	5	4	3	2	1	0
byte 1	Message Type				DUP flag	QoS level		RETAIN
byte 2	Remaining Length							

— DUP: Duplicate delivery
— QoS: Quality of Service
— RETAIN: RETAIN flag
  —This flag is only used on PUBLISH messages. When a client sends a PUBLISH to a server, if the Retain flag is set (1), the server should hold on to the message after it has been delivered to the current subscribers.
  —This allows new subscribers to instantly receive data with the retained, or Last Known Good, value.

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

Communication protocols

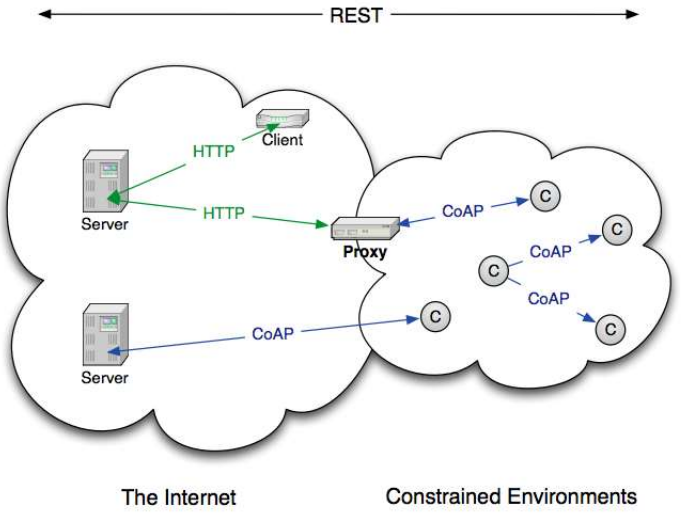
CoAP: Constrained Application Protocol

CoAP (Constrained Application Protocol)

- **CoAP** is a standardized request response protocol. It's creators redesign functions from HTTP taking into consideration, the target device's CoAP will be used on. That is embedded devices with limited processing power, storage, and bandwidth.
- **CoAP** also supports publish-subscribe style communication and resource discovery. It uses UDP as the transport protocol and has a dual layered structure. The transactional layer takes care of message handling between nodes, and the request response layer manages resources and transmits messages. This enables efficient congestion control and reliability over UDP.
- Few other features of **CoAP**:
  - A key IoT standard. Supported in Java, C, Python, C#, Go, etc.
  - Open IETF standard since June 2014
  - Based on web standards, easily integrates with HTPP. Is not simply a compressed version of HTTP.
  - Built for small, constrained, imbedded, occasionally sleeping devices. Why sleep?
  - Some built-in reliabilty over UDP/IP.
  - May also run over UDP/6LoWPan.
  - Use on low power, low bandwidth, lossy networks

8 July 2018

The CoRE (Constrained RESTful Environment) Architecture



8 July 2018

Who uses the CoAP?

- Open Mobile Alliance M2M
- IPSO Alliance (IP for Smart Objects)
- European Telecom Standards Institute M2M / OneM2M
- Lighting systems for smart cities
- Device management for network operators.
- Copper is a Firefox plugin – treat devices as REST services
- Main Java project on github : Californium

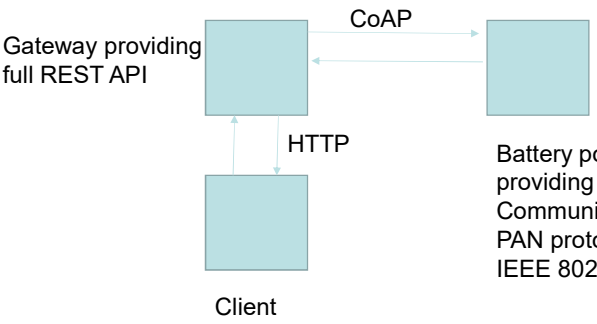
8 July 2018

Recall the typical HTTP Interaction

- Connection oriented and synchronous (blocking)
- TCP 3 way handshake with server
- HTTP GET /kitchen/light
- HTTP response with headers and {"setting" : "dim" }
- TCP 2 way termination
- Too much work for simple IoT applications
- CoAP is not a general replacement for HTTP
- CoAP does not support all features of HTTP

CoAP is based on REST

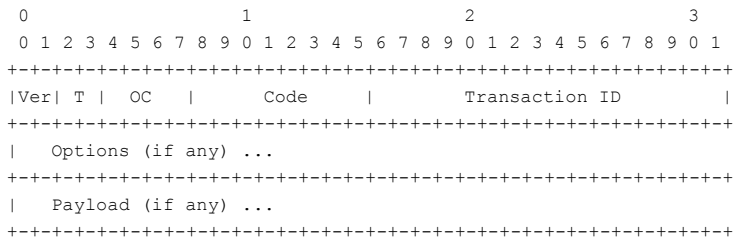
CoAP provides a request/response RESTful interaction like HTTP. Smaller messages than HTTP and with very low overhead. BLE (Bluetooth Low Energy) nodes, for example, have limited memory and storage. Sensors and actuators on BLE nodes are simply CoAP REST resources. For example, to obtain a current temperature, send a GET request.
To turn on/off or toggle LEDs we use PUT requests.



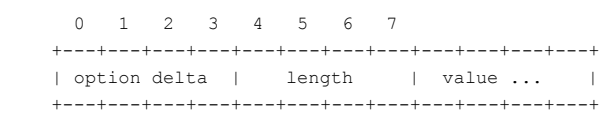
REST API Design Principles

Principle	Implementation
Use a constrained user interface. The verbs are polymorphic.	HTTP GET, POST, DELETE, PUT
Use standard status codes. Don't make things up.	HTTP codes
Use URI's for nouns.	Naming. Identification of resources. Well designed URI's pointing to a resource - protocol and location.
Negotiate representations.	JSON or XML messages or Image or …
HATEOAS (Hypertext as the Engine of application state)	Messages returning pointers or links for further discovery.
Statelessness	Simple request/response required. No conversational state. Easy to scale. The request must contain all that is required for the reply to be computed.

The Header



Typical Option:



Ver - Version (1)

T - Transaction Type (Confirmable, Non-Confirmable, Acknowledgement, Reset)

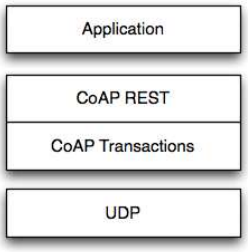
OC - Option Count, number of options after this header

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

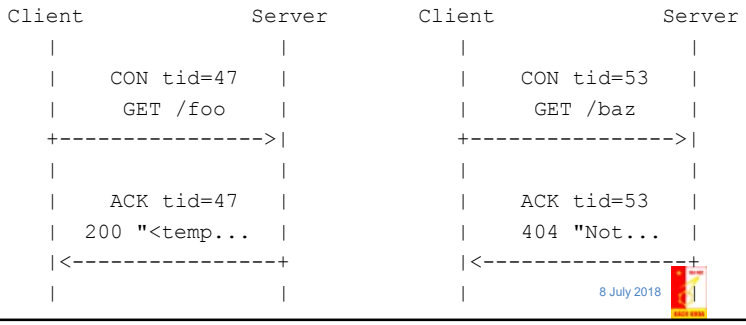
Transaction ID - Identifier for matching responses

The Transaction Model

- Transport
  - CoAP is defined for UDP
- Transaction
  - Single message exchange between end-points
  - CON, NON, ACK, RST
- REST
  - Piggybacked on transaction messages
  - Method, Response Code and Options (URI, content-type



Synchronous Transaction



8 July 2018

CoAP Messages

- Has a scheme coap://
- Has a well known port.
- GET, POST, PUT, DELETE encoded in binary (1 == GET)
- Block transfer support.
- Confirmable messages requires an ack with message ID. The message ID of the ack matches the message ID of the confirmable message.
- Non-confirmable messages do not require an ack. Less reliable.
- Responses are matched with requests via the clieny generated Token.
- Example:

CoAP Client CoAP Server

----> CON {id} GET /basement/light Confirmable request has an ID

<---- ACK {id} 200 Content {“status” : “on”} Piggy back response and same ID

8 July 2018

CoAP Uses Timeouts over UDP

CoAP Client CoAP Server

----> CON {id} GET /basement/light lost request

timeout -- --> CON {id} GET /basement/light finally arrives

<---- ACK {id} 200 Content {“status” : “on”}

The {id} allows us to detect duplicates.

CoAP Request/Acknowledge/Callback

CoAP Client CoAP Server

----> CON {id} PUT /basement/cleanFloor Token: 0x22 Needs time

<---- ACK {id} I am on it!

<----- CON {newID} 200 Content /basement/cleanFloor Token: 0x22 Done

----> ACK {newID}

In this example, the same token is used to identify this request and the service response. The id’s are used at the message level.

CoAP Publish/Subscribe

CoAP Client CoAP Server

----> CON {id} GET /basement/light Observe: 0 Token: 0x22

<---- ACK 200 {id} Observer: 27 Token 0x22

<---- CON 200 Observe: 28 Token: 0x22 {“light” : ”off”}

-----> ACK Token: 0x22

<---- CON 200 Observe: 28 Token: 0x22 {“light” : ”on”}

:

:

Block transfer is similar. We may request a transfer (one block at a time).

CoAP Resource Discovery

- Not the same as service discovery. Service discovery is at a lower level. At low levels, we don’t even know if services are available or how they communicate.
- We might register a printer, for example, with a discovery service and find it later on the fly.
- With resource discovery, we know we are looking for web resources.
- Links are returned. HATEOAS.
- Links may include a rel attribute – providing semantics.
- A well known resource is used to discover other resources.
- Perform a GET on the well known resource. Returned content is a list of links with REL attributes.
- Resource directories may be used to register resources. Registrations are simply POSTs with links. PUTs are used for updates. GETs for discovery.

Communication Protocols

XMPP: Extensible Messaging & Presence Protocol

Where are we now?

Internet Protocol Suite

We ar

HTTP, Websockets, DNS, XMPP, MQTT, CoAP	Application layer
TLS, SSL	Application Layer (Encryption)
TCP, UDP	Transport
IP(V4, V6), 6LowPAN	Internet Layer
Ethernet, 802.11 WiFi, 802.15.4	Link Layer

Who Uses XMPP?

- Cisco Webex
- WhatsApp uses a trimmed down version
- Google's Firebase and Google's Android
- Adopted by Sensor Andrew
- So, what do messaging systems have in common with IoT?

What is XMPP?

XMPP was originally a text-based messaging protocol for messaging between devices over a network using the XML document format. XMPP is being explored for Internet of Things by the XMPP standards foundations for access control and sensor data. Its strong point is it's very familiar device at server naming approach. Besides this, it provides end to end security, provisioning, and publish subscribe style communication. It is slow, but secure and uses TCP as the transport protocol.

8 July 2018

XMPP From IETF

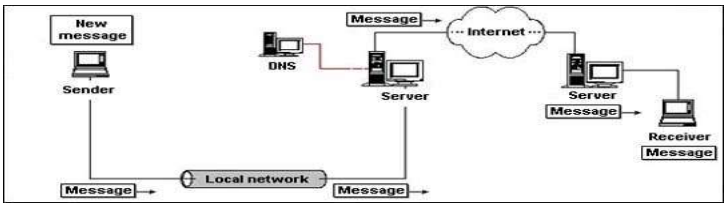
- In IM, the central point of focus is a list of one's contacts or "buddies" (in XMPP this list is called a "roster").
- The purpose of using such an application is to exchange relatively brief text messages with particular contacts in close to real time -- often relatively large numbers of such messages in rapid succession, in the form of a one-to-one "chat session" The catalyst for exchanging messages is "presence" -- i.e., information about the network availability of particular contacts (thus knowing who is online and available for a one-to-one chat session).
- Presence information is provided only to contacts that one has authorized by means of an explicit agreement called a "presence subscription".
- Thus at a high level XMPP needs to be able to complete the following use cases:
  - Manage items in one's contact list (list is maintained on the server)
  - Exchange messages with one's contacts
  - Exchange presence information with one's contacts (send communication status to the server)
  - Manage presence subscriptions to and from one's contacts

8 July 2018

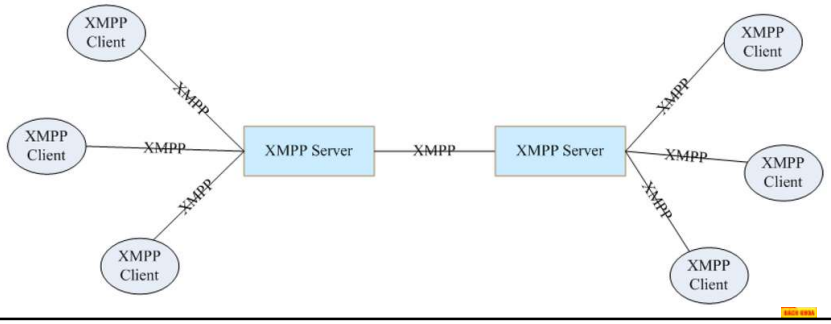
XMPP Standard

- IETF Standard since 2004
- Open and free – supported by the XMPP Standards Foundation (XSF)
- Created by Jermy Miller in 1998 – originated as Jabber - multi user chat
- Has been extended in many ways over the years, supports Pub/Sub
- Designed for client server instant messaging but with no central authority
- Architecture similar to email – anyone can run an XMPP server
- May be isolated on a local intranet or may be public
- All about sending XML messages over a network to do a wide variety of things. An XMPP message may be used to communicate between people (a chat application) or might be used by a sensor to report on its state.

SMTP Architecture



XMPP Architecture



XMPP Basic connection

- Client initiates a TCP connection
- Client sends presence information to the server
- The client requests and receives its roster
- The client interacts with roster members
- The client disconnects

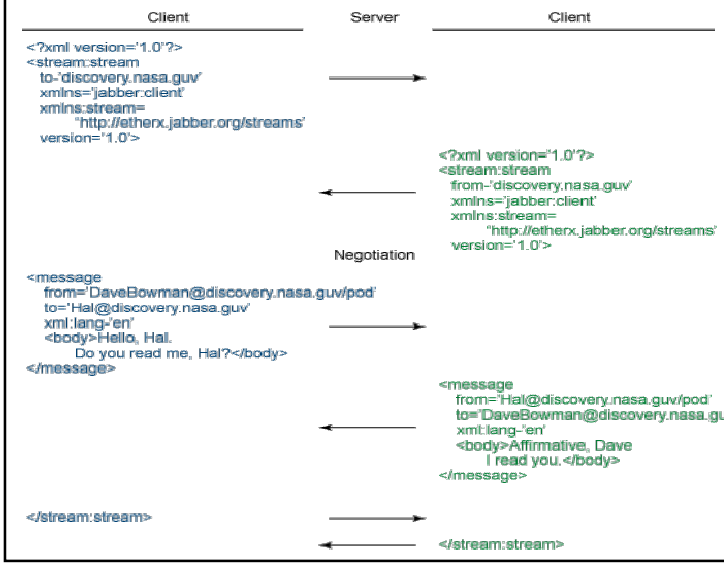
This is all done with standard XML messages using the XMPP vocabulary and grammar.

Naming Things

- A lot like an email address , XMPP uses Jabber ID's (JIDs)
- All of these JIDs could be logged on at the same time.
- mm6@andrew.cmu.edu Called a "bare JID"
- mm6@andrew.cmu.edu/mobile Includes a resource. This is called a "full JID"
- mm6@andrew.cmu.edu/tablet Resource used for message delivery
- mm6@andrew.cmu.edu/auto Resource typically assigned by server
- On Whatsapp [phone number]@s.whatsapp.net
- You may only subscribe to bare JIDS.
- A resource identifies a particular client belonging to the user
- A user can log in to an XMPP server multiple times, and in this case, the resource can denote a location. For example, the sample user might have a JID for his main terminal (DavidBowman@discovery.nasa.guv/termina and another JID (session) from an EVA pod (DavidBowman@discovery.nasa.guv/eva_pod1). So, a particular location can be targeted or left absent to reach the user at whichever location he happens to be logged in.

8 July 2018

Simplified Discussion (example)



Note: the XML is being transferred in pieces. The TCP connection only closes at the end.

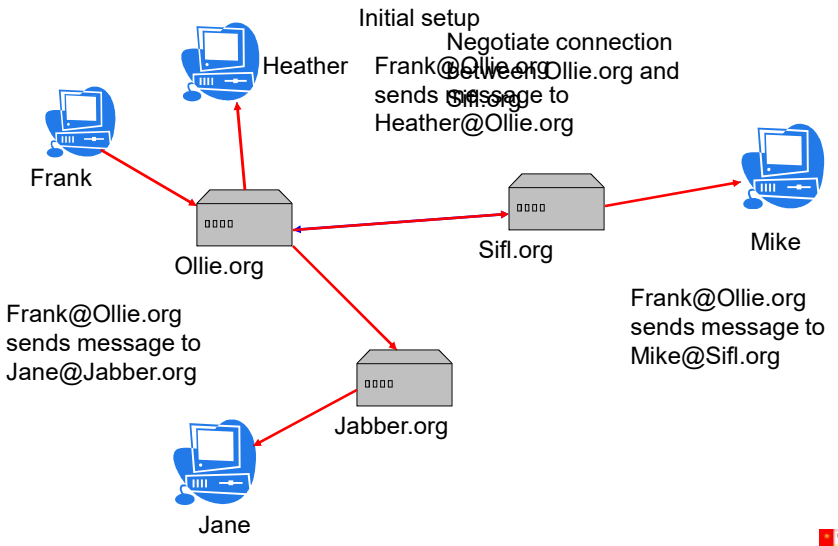
Note: there are two XML documents involved.

Would this work over websockets?

Sure. It involves a bidirectional conversation.

8 July 2018

Generic XMPP Message Routing



8 July 2018

XMPP mechanics

- XML-Stanzas
  - An element of communication
  - 3 types
    - message
    - IQ (Information Query)
    - presence
- XML-Streams
  - Entire communication between client and server
  - Contains unbounded number of XML-Stanzas

Common Attributes

- Attributes common to Message, Presence, and IQ stanzas
  - to
  - from
  - id
  - type

8 July 2018

Message Elements

- Used when one client "talks" to another
- Push mechanism similar to email
- Should have 'to' attribute
- Possible types (defined by type attribute)
  - Chat
  - Groupchat
  - Error
  - Headline
  - Normal
- Child elements
  - Subject
  - Body
  - Thread

Presence Elements

- Used to inform users of who is online
- Pub-Sub mechanism
- Types
  - unavailable
  - subscribe
  - unsubscribe
  - probe
- Child elements
  - show
  - status

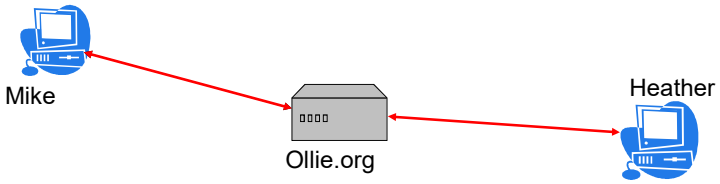
8 July 2018

Information Query (IQ) Elements

- Request-Response mechanism similar to HTTP
- Interactions tracked by 'id'
- Type (required)
  - get
  - set
  - result
  - error
- Semantics defined very broadly by RFC
  - Example use is to retrieve rosters

8 July 2018

XMPP and TCP

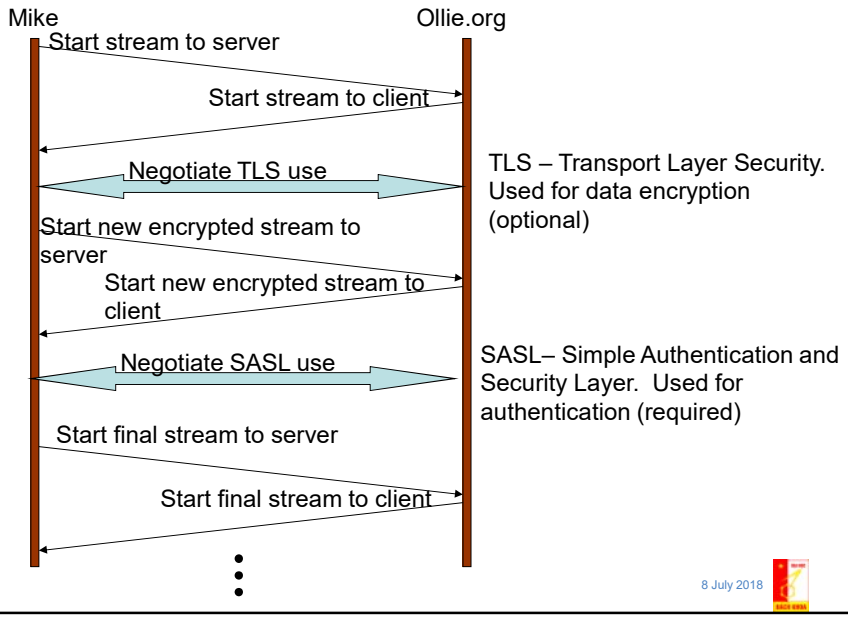


Mike wants to sign on:
1. Establish TCP connection
2. Establish XML streams

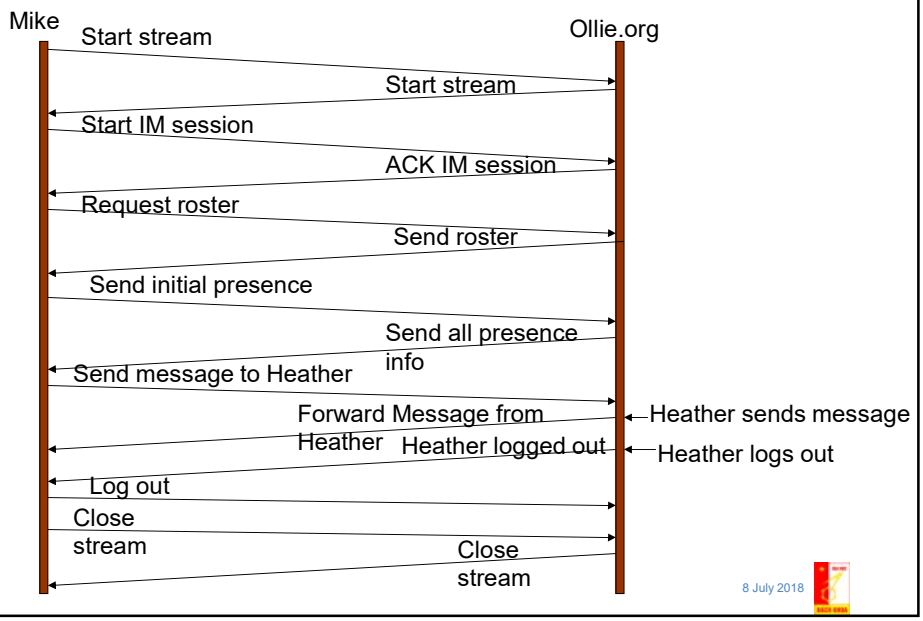
XML stream

One TCP connection per stream
One stream from client to server, one from server to client
TCP connection stays active for entire length of stream

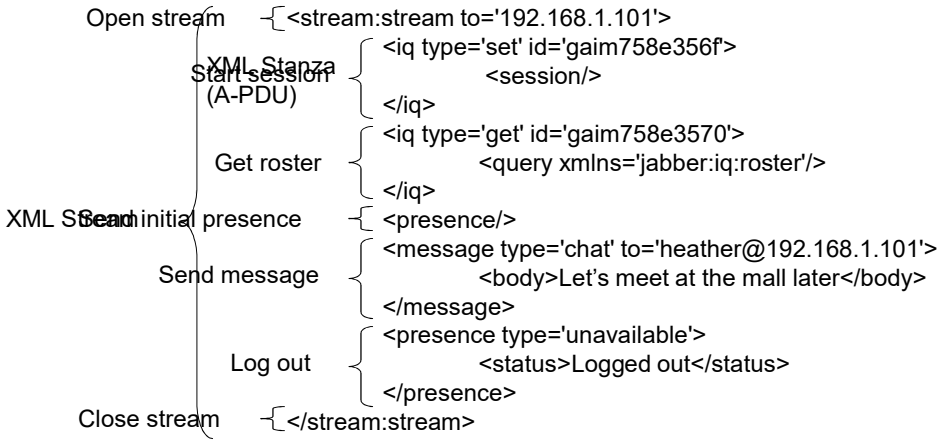
XML stream establishment



XMPP-IM conversation



Simple Client-to-Server Stream (document view)



Simple Client-to-Server Stream (document view)

Open stream
```
<stream:stream>
```
Offer services
```
<stream:features>
        <bind xmlns="xmpp-bind"/>
        <session xmlns="xmpp-session"/>
</stream:features>
```
Start session
```
<iq type="result" to="mike">
        <session xmlns="xmpp-session"/>
</iq>
```
Send roster
```
<iq type="result" to="mike">
        <query xmlns="jabber:iq:roster">
                <item jid="heather" subscription="both"/>
        </query>
</iq>
```
Send contact's presence
```
<presence type="unavailable" from="heather" to="mike">
        <status>Logged out</status>
</presence>
```
Send presence update
```
<presence from="heather" to="mike"/>
```
Send message
```
<message type="chat" to="mike" from="heather">
        <body>Ok.  See you at 8</body>
</message>
```
Send presence update
```
<presence type="unavailable" from="heather" to="mike">
        <status>Logged out</status>
</presence>
```
Close stream
```
</stream:stream>
```

8 July 2018

With Things we need Discovery[1]

- A naming service provides service references if you have the name in hand. Example: Phone book, the Domain Name Service (DNS)
- A directory service provides service references if you have attributes in hand. Example: Google search, Lightweight Directory Access Protocol (LDAP)
- A discovery service is a directory service that allows registration, de-registration, and lookUp of services in a spontaneous network – where clients and services may come and go. Example: You have printer attributes and discover a printer.
- Discovery may be done with a directory server or be serverless.
- In serverless discovery, participants collaborate to implement a distributed directory service. Two main models:
  - push model : services regularly advertise their services (multicast).
  - pull model: clients multicast their queries.

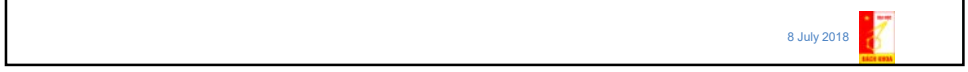
XMPP and Thing Discovery (1)

- During production of a Thing, decisions have to be made whether the following parameters should be pre-configured, manually entered after installation or automatically found and/or created by the device if possible (zero-configuration networking):
  - Address and domain of XMPP Server.
  - JID of the Thing.
  - JID of Thing Registry, if separate from the XMPP Server.
  - JID of Provisioning server, if different from the Registry and XMPP server
- A "provisioning server" may be needed if the device needs special configuration parameters from that server. Perhaps with user involvement.

8 July 2018

XMPP and Thing Discovery (2)

- If the address of an XMPP Server is not preconfigured, the thing must attempt to find one in its local surroundings. This can be done using one of several methods:
  - Dynamic Host Configuration Protocol (DHCP)
    - Server returns IP address as needed – device issues a query to a well known broadcast address. Response may include DNS location.
  - Multicast Domain Naming Service (mDNS)
    - small network, no DNS server, P2P
    - multicast a query message with a name. The server with that name responds (broadcasts) its IP address. Machines may update caches. Build a directory of name, ip mappings
  - Simple Service Discover protocol SSDP/UPnP (Universal Plug n Play)
    - No DNS or DHCP, Uses UDP and multicast addresses
    - A UPnP compatible device from any vendor can dynamically join a network, obtain an IP address, announce its name, advertise or convey its capabilities upon request, and learn about the presence and capabilities of other devices.[1] Uses XML messages.
- The XMPP server and registry are found and the Thing registers itself with the following XMPP message:



XMPP and Thing Registration

```
<iq type='set'
    from='thing@example.org/imc'
    to='discovery.example.org'
    id='1'>
  <register xmlns='urn:xmpp:iot:discovery'>
    <str name='SN' value='394872348732948723'/>
    <str name='MAN' value='www.ktc.se'/>
    <str name='MODEL' value='IMC'/>
    <num name='V' value='1.2'/>
    <str name='KEY' value='4857402340298342'/>
  </register>
</iq>
```

Suppose a sensor is registered. How do we read from it?

Reading sensor data

- Request to a Thing – an AMR device is used for Automatic Meter Reading

```
<iq type='get' from='client@clayster.com/amr'
               to='device@clayster.com' id='S0001'>
    <req xmlns='urn:xmpp:iot:sensordata' seqnr='1' momentary='true'/>
</iq>
```

- Response from a Thing – I got your request

```
<iq type='result' from='device@clayster.com'
                  to='client@clayster.com/amr' id='S0001'>
    <accepted xmlns='urn:xmpp:iot:sensordata' seqnr='1'/>
</iq>
```

8 July 2018

Data arrives from a sensor

```
<message from='device@clayster.com'
         to='client@clayster.com/amr'>
  <fields xmlns='urn:xmpp:iot:sensordata' seqnr='1' done='true'>
    <node nodeId='Device01'>
      <timestamp value='2013-03-07T16:24:30'>
        <numeric name='Temperature' momentary='true'
                 automaticReadout='true' value='23.4' unit='°C'/>
        <numeric name='load level' momentary='true'
                 automaticReadout='true'
                 value='75' unit='%'/>
      </timestamp>
    </node>
  </fields>
</message>
```

8 July 2018

Communication Protocols

AMQP: Advanced Message Queuing Protocol

The 'Advanced Message Queuing Protocol' ('AMQP') is an open standard application layer protocol for message-oriented middleware.

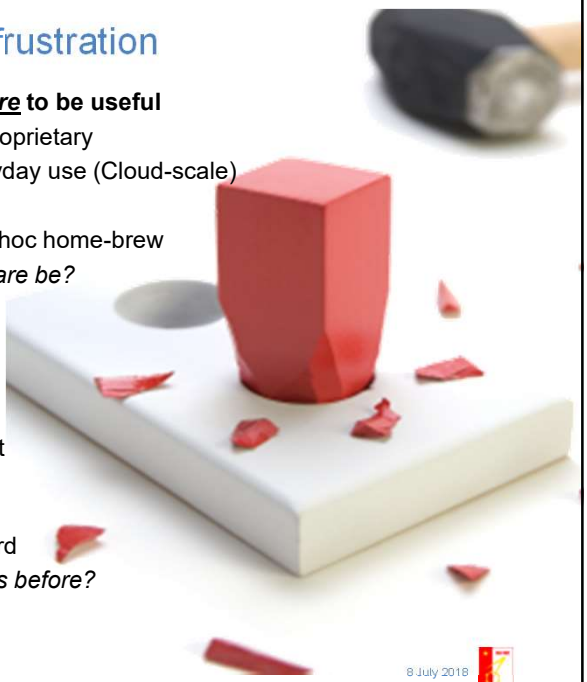
The defining features of AMQP are message orientation, queuing, routing (including point-to-point and Publish/subscribe|publish-and-subscribe), reliability and security.

Unlike JMS, which merely defines an API, AMQP is a **Wire protocol|wire-level protocol**

AMQP Motivation

AMQP was born of frustration

- **MOM needs to be ***everywhere*** to be useful**
  - dominant solutions are proprietary
    - too expensive for everyday use (Cloud-scale)
    - they don't interoperate
  - has resulted in lots of ad-hoc home-brew
    - *how hard can middleware be?*
- Middleware Hell
  - 100's of applications
  - 10,000's of links
  - every connection different
  - massive waste of effort
- The Internet's missing standard
  - *Why has no one done this before?*



The AMQP Working Group

- Set up by JPMorgan in 2006
  - Goal to make Message Oriented Middleware pervasive
  - Make it practical, useful, interoperable
  - Bring together users and vendors to solve the problem
- We say AMQP is an "**Internet Protocol for Business Messaging**" so end users feel a connection to the technology.
  - AMQP aspires to define MOM

Cisco Systems
Credit Suisse
Deutsche Börse Systems
Envoy Technologies
Goldman Sachs
iMatix
IONA (a Progress company)
JPMorgan Chase
Microsoft
Novell
Rabbit Technologies
Red Hat
Solace Systems
Tervela
TWIST
WSO2
29West

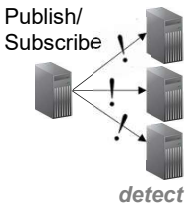
Ubiquitous => Unencumbered

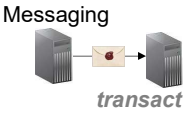
- **AMQP Intellectual Property Policy**
  - **Unambiguous Right to Implement**
    - The Authors each **hereby grants to you a worldwide, perpetual, royalty-free**, non-transferable, nonexclusive license to (i) copy, display, **distribute and implement the Advanced Messaging Queue Protocol** ("AMQP") Specification and (ii) the Licensed Claims that are held by the Authors, all for the purpose of implementing the Advanced Messaging Queue Protocol Specification.
    - "Licensed Claims" means those claims of a **patent or patent application**, throughout the world, excluding design patents and design registrations, owned or controlled, or that can be sublicensed without fee and in compliance with the requirements of this Agreement, by an Author or its affiliates **now or at any future time and which would necessarily be infringed by** implementation of the Advanced Messaging Queue Protocol Specification.
  - **The License is attached to the AMQP Specification itself**
    - You get the rights when you download it!

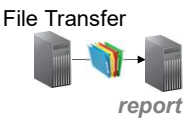
AMQP Functionality

AMQP 1.0 Covers…

- Queuing with strong Delivery Assurances
- Event distribution with Flexible Routing
- Large Message capability (gigabytes)
- Global Addressing Scheme (email-like)
- Meet common requirements of mission-critical systems





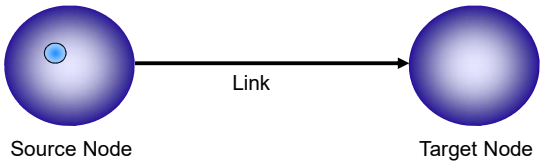


The AMQP Network

An AMQP Network consists of **Nodes** and **Links**.

A **Node** is a named source and/or sink of **Messages**.

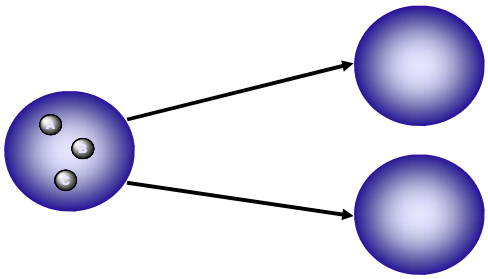
Messages travel between **Nodes** along named, unidirectional **Links.**



Types of Links

Destructive: the transfer along the link removes the message from the source

Non- Destructive: the message remains at the source node, and is "copied" to the destination.



Messages

Messages consist of parseable **Properties** and an opaque **Body**.

Messages may not be mutated by the AMQP Network.

The network carries information about the Message in **Headers** and **Footers**.

Header and Footer values may be modified in the Network.

Message Identity

A Message is assigned a globally unique identifier.

Nodes which perform transformations are creating new Messages, with new ids.

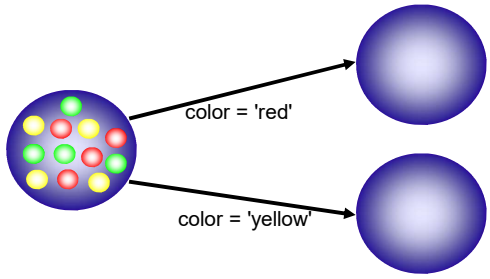
Only one “copy” of a Message can ever exist at a Node.

Filters

Each **Link** may have a **Filter** which evaluates which messages may travel along it.

Filters are evaluated against immutable properties of the Message.

Filter syntax determined by the Filter Type, e.g. SQL, XQuery



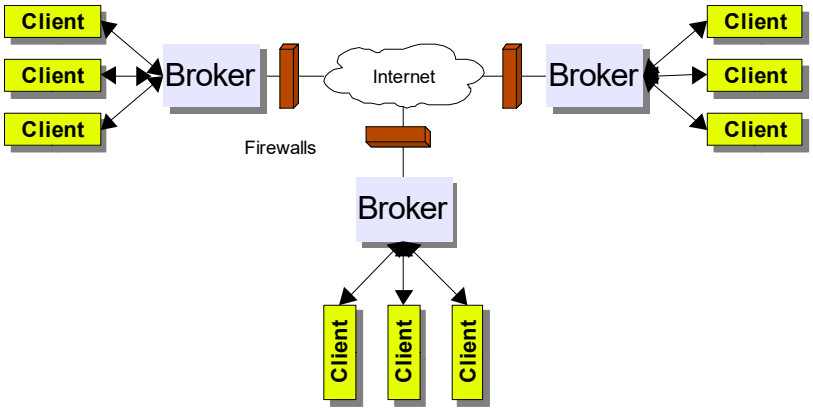
8 July 2018

AMQP is an Overlay Network

- Broker
  - Applications Connect to a Broker to participate in the AMQP network
  - The Connection is used to establish a Session
    - Sessions provide state between Connections, establish identity, ease failover
  - Connections are further subdivide into Channels
    - Multiple threads of control within an Application can share one Connection
- Queues
  - Applications logic interacts ONLY with Queues
  - Queues have well known Names == Addressable
  - Applications do not need to know how messages get in/out of Queues
  - Queues can be smart, they are an extension point
  - Applications will assign implied semantics to Queues (e.g. “StockOrderQueue”)
- Links
  - Links move Messages between Queues and/or Applications
  - Contain Routing and Predicate Evaluation Logic – similar to Complex Event Processing

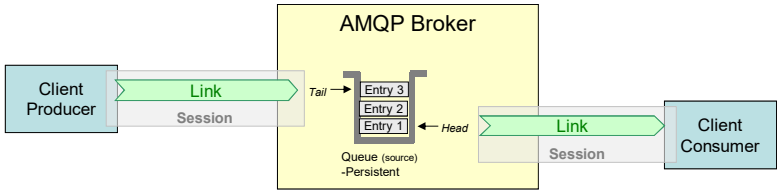
8 July 2018

Inter-Network Connectivity



Some Typical Usage Patterns

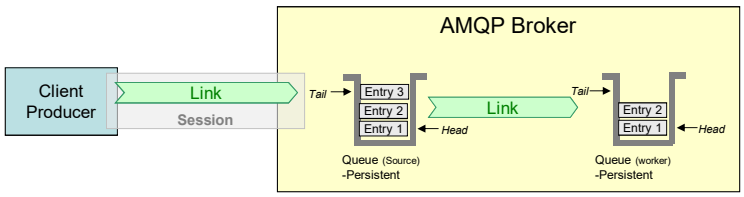
Point-to-point Queue Delivery



Highlights:
- Only "Source" queue is required and can be read directly by consumer over Link (i.e. dedicated consumer Worker queue and bridging between Source and Worker unnecessary).

8 July 2018

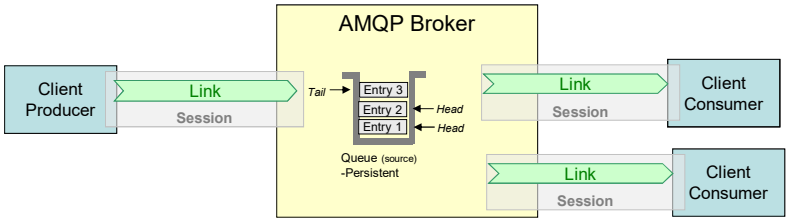
Abstracted Point-to-point Queue



Highlights:
- One Queue performs the role of holding the "Well Known" name for the outside world.
- All messages are automatically forward on to the real worker queue.
- Allows internal topology to change without the outside world seeing (this PO Box)

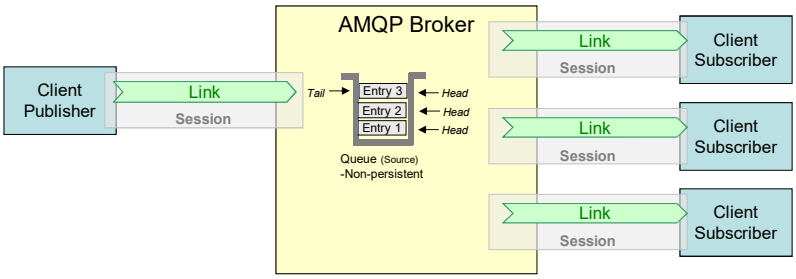
8 July 2018

Load-Balanced Point-to-point Queue Delivery



8 July 2018

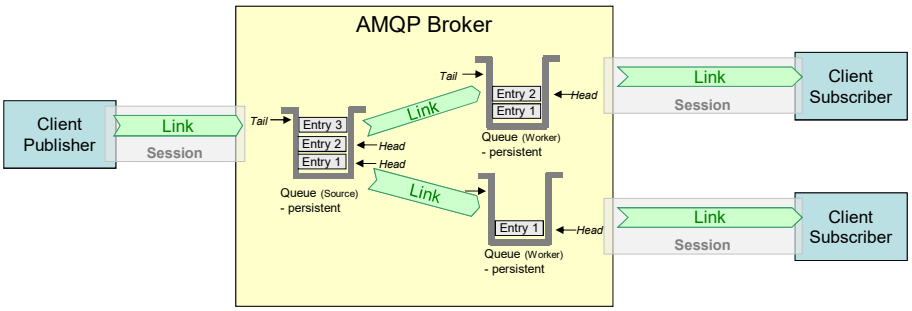
Dynamic (non-persistent) Pub/Sub Delivery



Highlights:
- Messages are "garbage collected" in an implementation specific manner after delivery.
- AMQP makes some guarantees about how long messages are valid for.

8 July 2018

Durable (persistent) Pub/Sub Delivery



Sample Message Queues
(a partial list…)

- **RabbitMQ**
- The most popular due to multiple operating system support (even Windows!) and a plethora of clients and tools that support RabbitMQ.
- RabbitMQ is written in Erlang and implements a common standard: the Advanced Message Queuing Protocol (AMQP).
- RabbitMQ queues messages on a central server, making it easy to deploy but a bit more interesting to scale.

- **ActiveMQ**
- Apache ActiveMQ is a message broker written in Java together with a full **Java Message Service (JMS)** client.
- Designed to communicate over a number of protocols such as AMQP, STOMP (a the Simple Text Orientated Messaging Protocol.) and OpenWire (native binary wire formant of ActiveMQ) together with supporting a number of different language specific clients.

- **ZeroMQ**
- Typically deployed for clustered systems and/or supercomputing.
- It is a sockets library for messaging and is extremely fast. It gives you sockets that carry atomic messages across various transports like in-process, inter-process, TCP, and multicast.
- You can connect sockets N-to-N with patterns like fanout, pub-sub, task distribution, and request-reply.
- **ZeroMQ does not follow a broker pattern** like RabbitMQ or ActiveMQ, meaning ZeroMQ does not run on a single server or cluster of servers (P2P)
- A wealth of information about ZeroMQ is available in "Code Connected Volume 1 - Professional Edition" by Pieter Hinjens, the CEO of iMatrix.

8 July 2018

- **Zaqar (ex Marconi)**
- Marconi is a new OpenStack project to create a multi-tenant cloud queuing service.
- The aim is to create an **open alternative** to Amazon's SQS (producer-consumer) and SNS (pub-sub) services, for use in applications that run on OpenStack clouds.
- Marconi is currently in development.

8 July 2018

- **IronMQ ("The Message Queue For the Modern Cloud")**
- An easy-to-use highly available message queuing service.
- IronMQ is for the person that doesn't want to manage their own queue servers. They provide an endpoint where you create queues and messages with a highly available backend.
- IronMQ uses HTTPS to transport messages instead of AMQP.
- Create a Queue, Post, Receive messages from your application to your named queue all via **REST**-based with JSON bodies and use OAuth2 for authentication.
- http://www.iron.io/mq

8 July 2018

- **Many more…**
- Kafka
- Amazon Simple Queue Service (SQS)
- Microsoft Azure Queues / ServiceBus
- …

8 July 2018

Communication Protocols

RESTful Services

8 July 2018

- **RESTFUL SERVICES** is not really a protocol but an architectural style. It was first introduced by Roy Fielding in 2000 [15], and it is being widely used ever since.
- REST uses the HTTP methods GET, POST, PUT, and DELETE to provide a resource-oriented messaging system where all actions can be performed simply by using the synchronous request/response HTTP commands. It uses the built-in accept header of HTTP to indicate the format of the data that it contains.
- The content type can be XML or JSON (JavaScript Object Notation) and depends on the HTTP server and its configuration. REST is already an important part of the M2M Communication because it is supported by all the commercial M2M cloud platforms.
- Moreover it can be implemented in smartphone and tablet applications easily because it only requires an HTTP library which is available for all the Operative Systems (OS) distributions.

8 July 2018

REST API Design Principles

Principle	Implementation
Use a constrained user interface. The verbs are polymorphic.	HTTP GET, POST, DELETE, PUT
Use standard status codes. Don't make things up.	HTTP codes
Use URI's for nouns.	Naming. Identification of resources. Well designed URI's pointing to a resource - protocol and location.
Negotiate representations.	JSON or XML messages or Image or …
HATEOAS (Hypertext as the Engine of application state)	Messages returning pointers or links for further discovery.
Statelessness	Simple request/response required. No conversational state. Easy to scale. The request must contain all that is required for the reply to be computed.

The End!

(Questions & Answers)