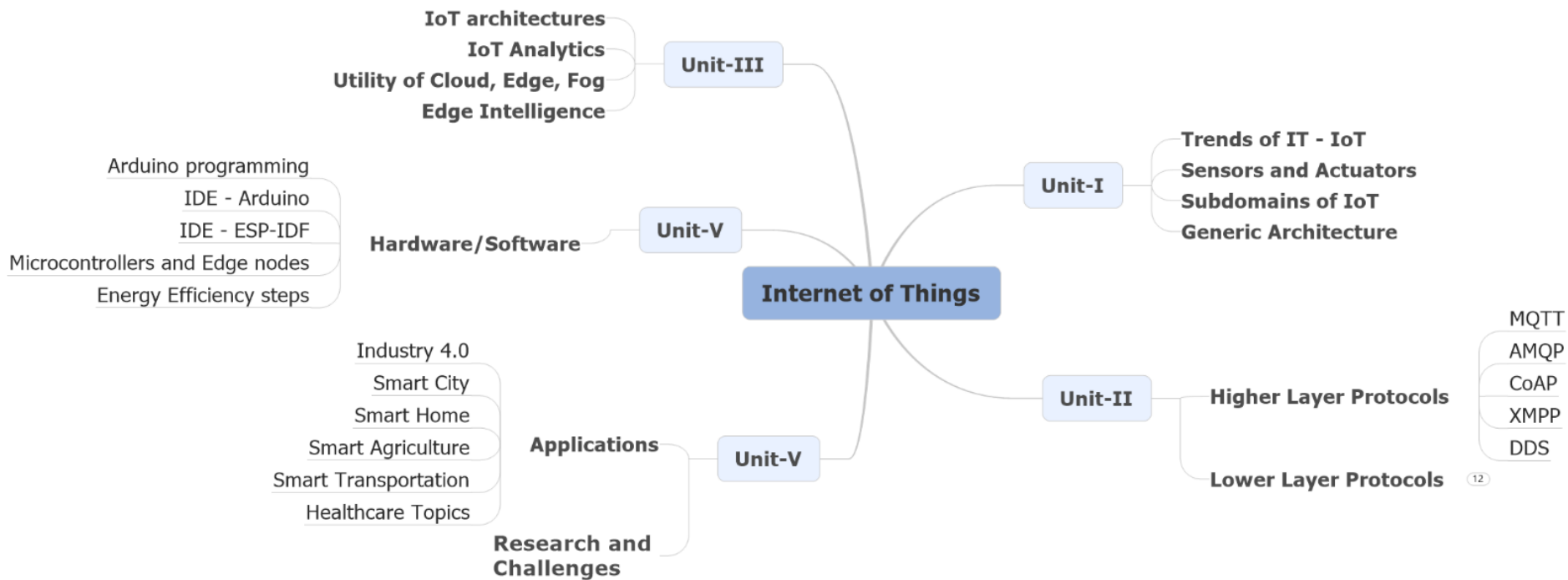


Selected Topics in Computer Architecture, Computer Networks, and Distributed Systems (Internet of Things) (IN3450)

Dr. Shajulin Benedict
shajulin@iiitkottayam.ac.in
shajulinbenedict@mytum.de

Prof. Dr. Michael Gerndt
gerndt@in.tum.de

Syllabus



•BRUSH UP on COMPUTER NETWORKS...

- Standards
- Layers
- Topologies
- Communication Aspects
 - Modulation Techniques
 - SS Techniques
 - Cellular Techniques
 - Antenna Fundamentals
 - Wave propagation and Fading Systems

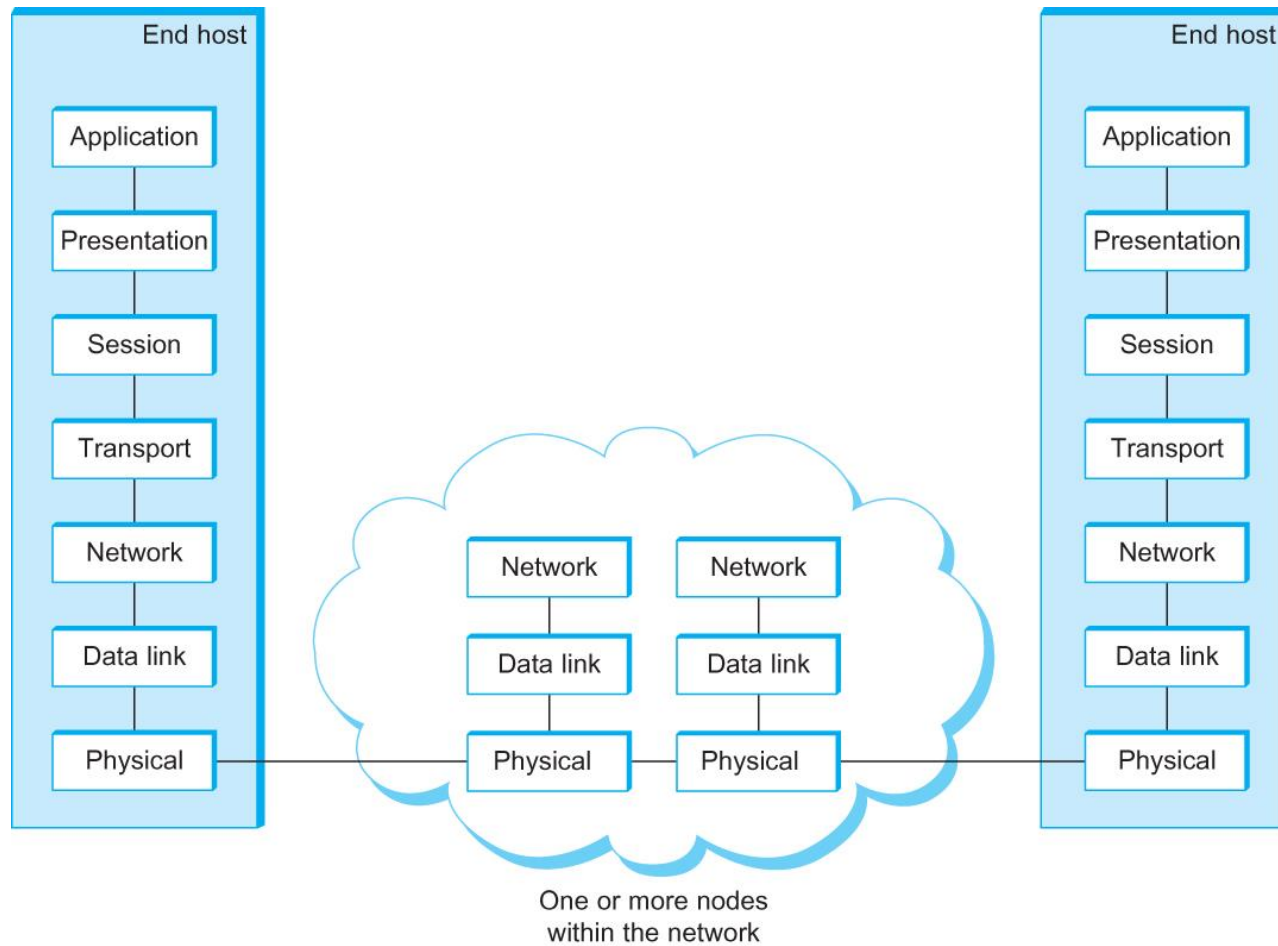
Network Architecture – In general

- A network must provide cost-effective, fair, and robust connectivity among computers/servers (preferably, large number of computers).
- To deal with the complexity, network designers have developed general blueprints (named network architectures).
- Two most widely referenced architectures
 - ISO-OSI standard architecture
 - Internet or TCP/IP standard architecture

Protocols

- A protocol defines the interfaces between the layers in the same system and with the layers of peer system.
- Protocols are the building blocks of a network architecture.
- A set of rules and guidelines for implementing network communications.

ISO-OSI Architecture



Explanation on layers

- Physical Layer

- Handles the transmission of bits over a communication link.
- A hardware part → made of chips.
- It provides the procedural interface to the transmission medium.

- Data Link Layer

- It collects a stream of bits called a **frame**.
- Network adaptors along with device driver in OS implement the protocol in this layer.
- Frames are typically delivered to the nodes rather than bits.

- Network Layer

- It handles a routing operation among nodes that are connected together.
- The unit of data exchanged between nodes in this layer is termed as **packet**.

Explanation on layers

- Transport Layer

- It implements a process-to-process channel.
- The unit of data exchanges in this layer is called a *message*.

- Session Layer

- It establishes sessions between different computers/machines.
- It also maintains the connections and terminates them.

- Presentation Layer

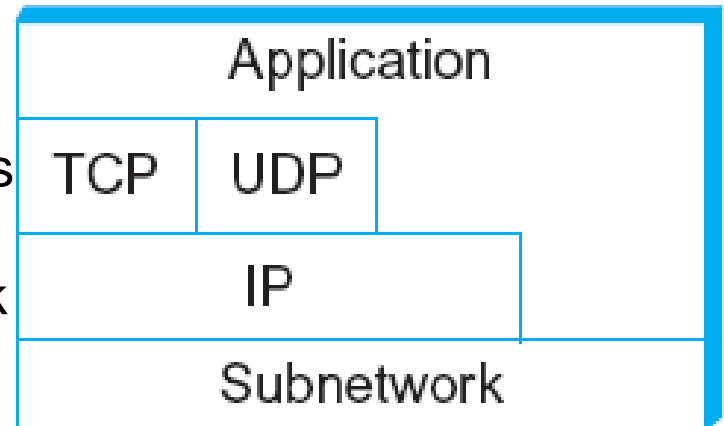
- It is concerned about the format of data exchanged between peers. (as application layers could utilize different formats).

- Application Layer

- It standardizes the common type of exchanges.
- The transport layer and the higher layers typically run only on end-hosts and not on the intermediate switches and routers

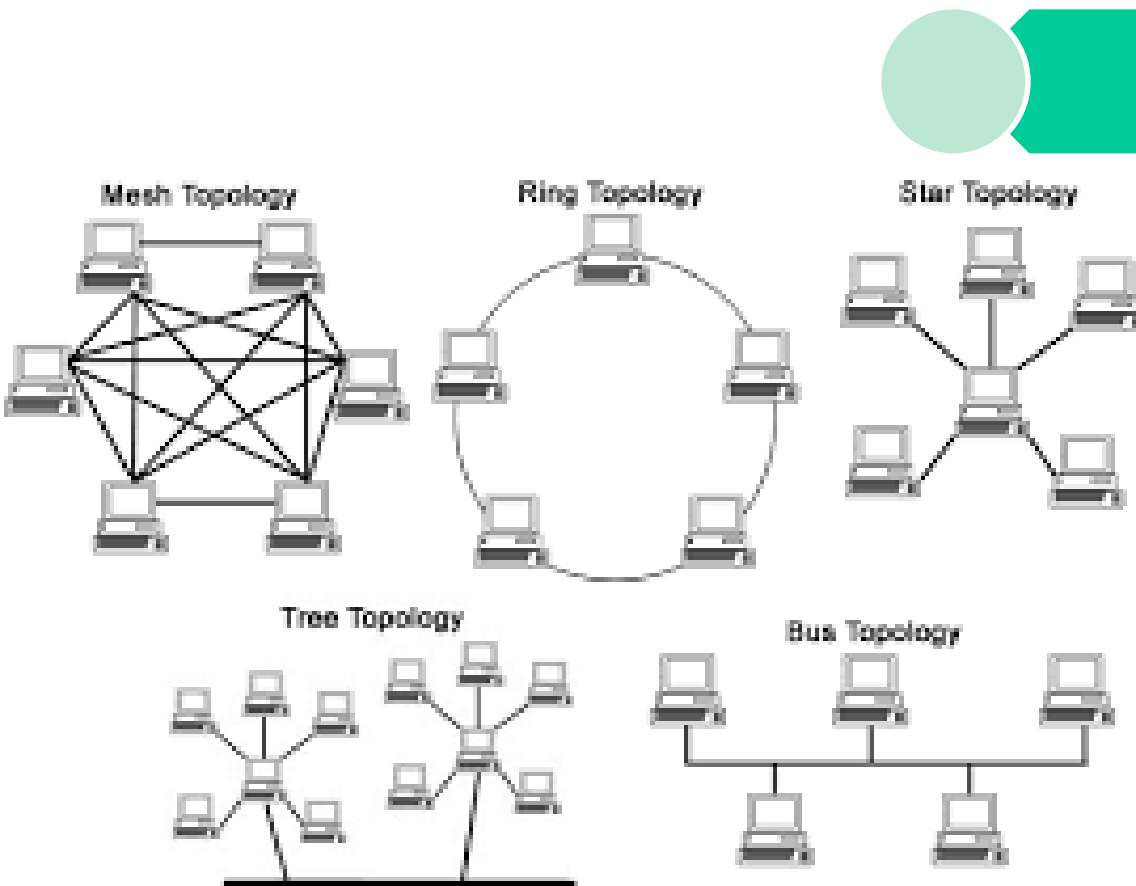
Internet or TCP/IP Architecture

- 4 layers are available.
- Subnetwork includes the lower layers
 - DLL and PL.
- Defined by Internet Engineering Task Force (IETF).



- The internet is based on TCP/IP...
(http://en.wikipedia.org/wiki/OSI_model#Comparison_with_TCP.2FIP_model)
- Most often, IoT applications are designed based on TCP/IP standards.

Network Topologies – in Computers

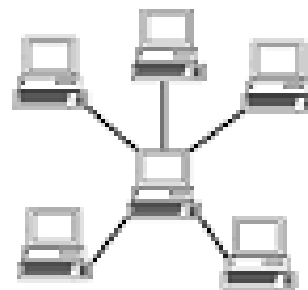


computerhope.com



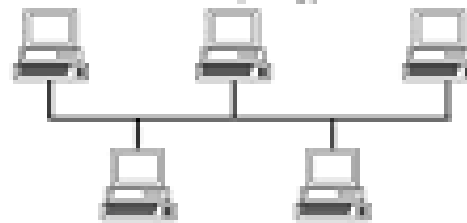
Bus

Star Topology



Star

Bus Topology



Ring

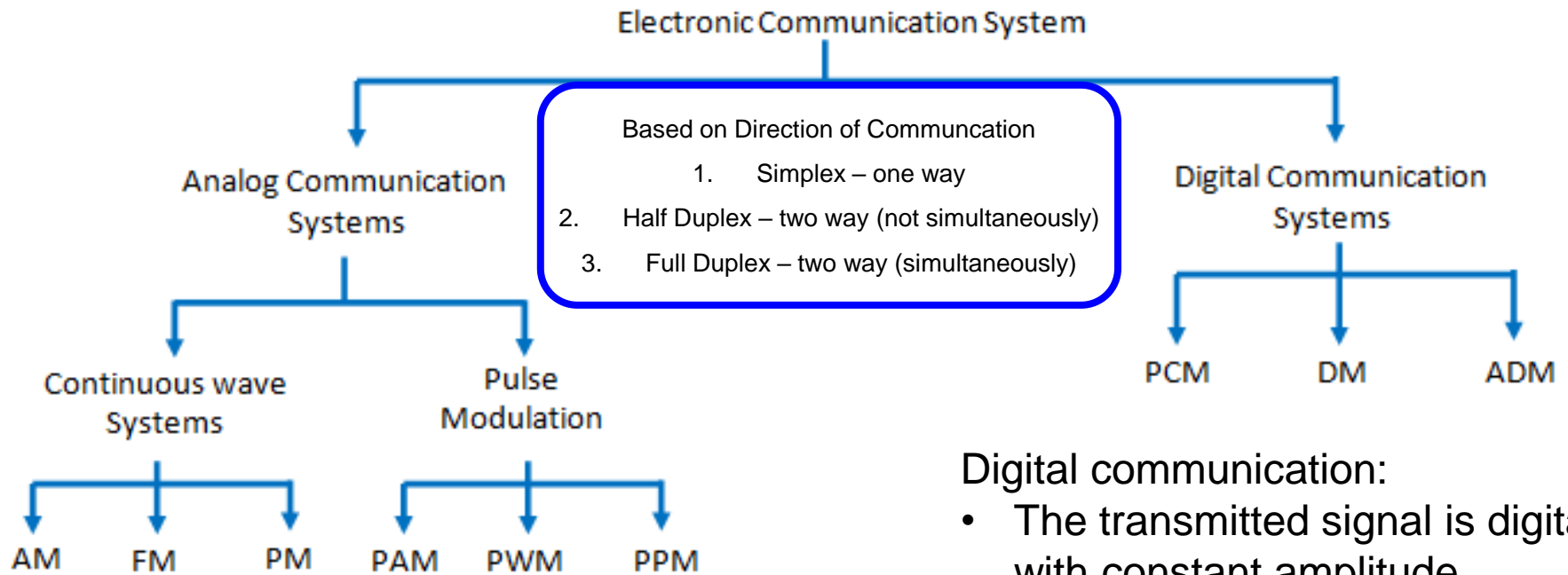
Tree

Mesh



Hybrid

Transmissions – Communication Aspects



Continuous wave – carrier is continuous; Adv: simple;
Disadv: noise
Pulse modulation – carrier is based on pulses.

Digital communication:

- The transmitted signal is digital with constant amplitude.
- Adv: noise could be avoided
- Adv: repeaters can be employed for regenerating signals
- Disadv: bit rates become higher.
- Disadv: requires synchronization

electronicspost.com/describe-the-classification-of-electronic-communication-system/

Spread Spectrum Technology

- SS technology was initially designed for military applications.
- It is a technique in which the signals generated with a certain bandwidth are spread across a wider bandwidth.
- For instance, LoRa based IoT sensors utilize SS techniques.



Direct Sequence

- Here, information is spread across a single BW.
- i.e., Each bit is represented by multiple bits in transmitted signal.
- Chipping code

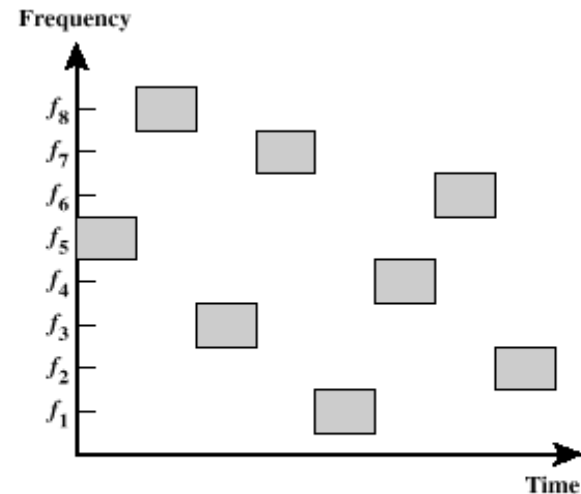
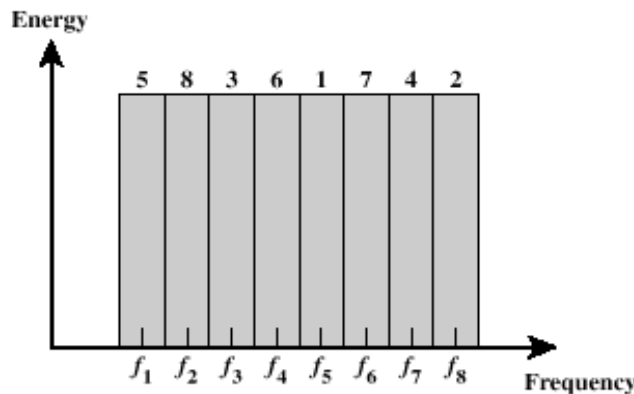


Frequency hopping

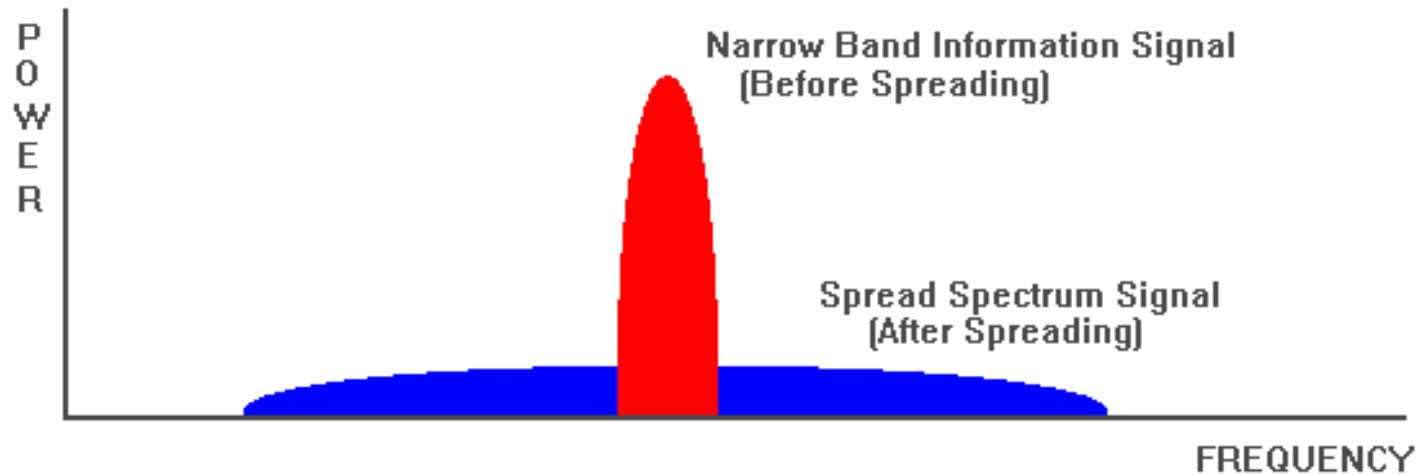
- Divides the larger BW to smaller channels.
- Signal broadcasts over seemingly random series of frequencies.
- Developed before DS-SS

FH-SS

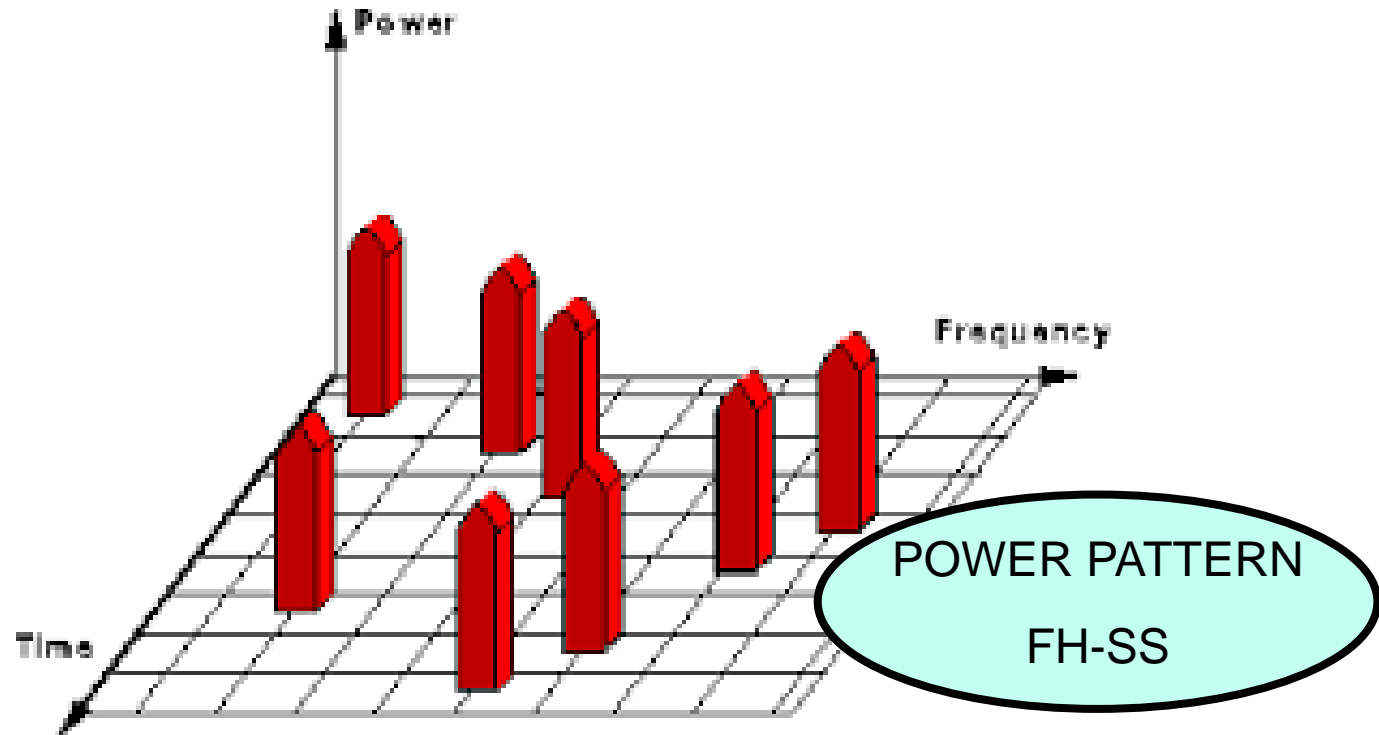
- Signal is broadcast over a large random series of radio frequencies.
- i.e., the information hops from frequency to frequency at fixed intervals.
 - Transmitter operates in one channel at a time.
 - At each successive interval, a new carrier frequency is selected.



DS-SS – Power Pattern

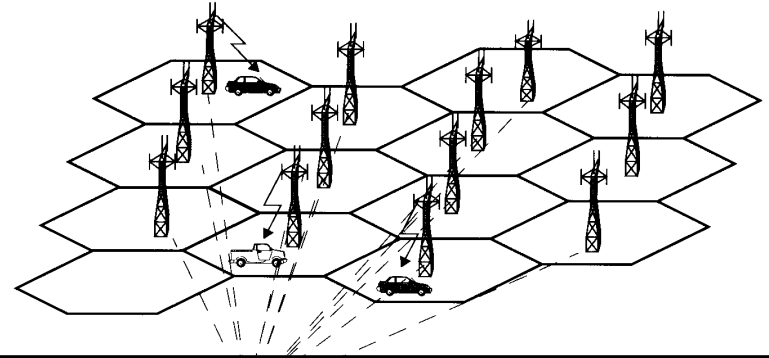


FH-SS – Power Pattern



Cellular telephone systems

- As like smartphones, cellular IoT devices can also rely on cellular telephone systems (such as 2G, 3G, 4G, 5G).
- Cellular telephone systems accommodate more users.
- It relies on the freq. reuse concept.
- A sophisticated handoff switching technique enables a uninterrupted call when the user moves from one cell to the other.
- It consists of mobile stations, base stations, and mobile switching centre.



Mobile Switching Centres

Cellular Telephone System

Types of Networks

LAN

WAN

MAN

PAN

CAN

SAN

WLAN

Antenna Fundamentals

- An antenna is a device that radiates radio waves when supplied with electric power, and/or a device that converts radio waves into electric power.
- Antenna fundamentals...
 - Types – based on spreading signals (omni-, semi-, uni-directional)
 - Types – based on structure and functionality
 - Wire antennas
 - Aperture antennas (waveguide based, horn type)
 - Reflector antennas (satellite-based)
 - Lens antennas (Typically, used for high frequency communications)
 - Array antennas (Yagi-uda, slotted waveguide)

Wave propagation effects / Noise

- Wireless transmission ***distorts*** any transmitted signal
- Sources of signal distortion in wireless channels
 - Attenuation – energy is distributed to larger areas with increasing distance. (gradual loss of signal over distance)
 - Reflection/refraction – wave hits smooth boundary and the wave gets reflected.
 - Diffraction – wave starts “new wave” from a sharp edge
 - Scattering – multiple reflections at rough surfaces
 - Doppler fading – shift in frequencies (loss of center)
- The signal at the receiver is the superposition of multiple and delayed copies of the same signal.

Fading

- Fading is often the reason when the transmitted signal is not received correctly in wireless medium.
- It is caused due to multipath propagation or shadowing from obstacles – scattered or reflected waves.
- 2 types
 - **slow fading** –the amplitude and phase of multi-path signals are in coherent. Hence, fading is slow.
 - **Fast fading** – the coherence time of a channel is small relative to the delay requirement of an application.

WSN Requirements

Sensor Network requirements

- i) Quality of service
- ii) Fault tolerant
- iii) Lifetime
- iv) Scalability
- v) Wide range of densities
- vi) programmability
- vii) Maintainability

Required Mechanisms

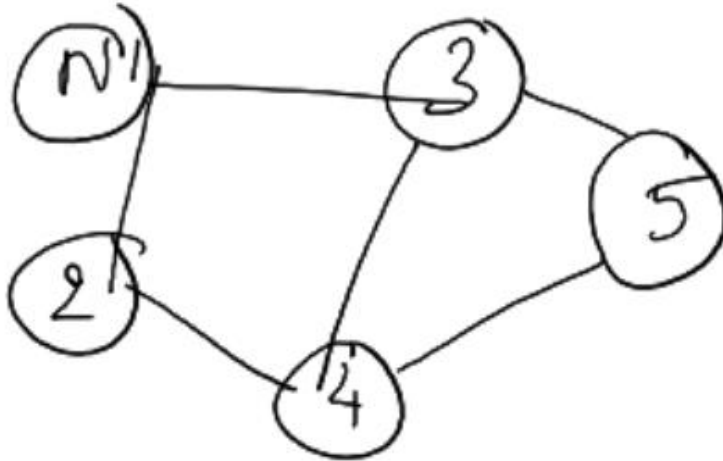
Required mechanisms:

- i) Multi-hop wireless communication
- ii) Energy efficient operation
- iii) Auto-configuration
- iv) collaboration and in-network processing
- v) Data centric operations.
- vi) Locality
- vii) Exploit tradeoffs
 - accuracy vs. cost
 - energy vs. cost

IoT Connectivity and Nodes

- IoT connectivity is similar to the normal internet connectivity.
- Some terms:
 - IoT LAN
 - Local connection of IoT nodes/things
 - IoT WAN
 - Connection of various different networks
 - IoT Node
 - It is a device connected to other nodes (Either via. LAN or WAN)
 - IoT Gateway
 - A device connecting an IoT LAN to a WAN to an internet/cloud.
 - IoT proxy
 - It performs application layer functions between IoT nodes and other entities
 - Caches pages/data/ requests (acts as intermediate client for nodes). Eg. Restricts access to a few sites or few nodes.

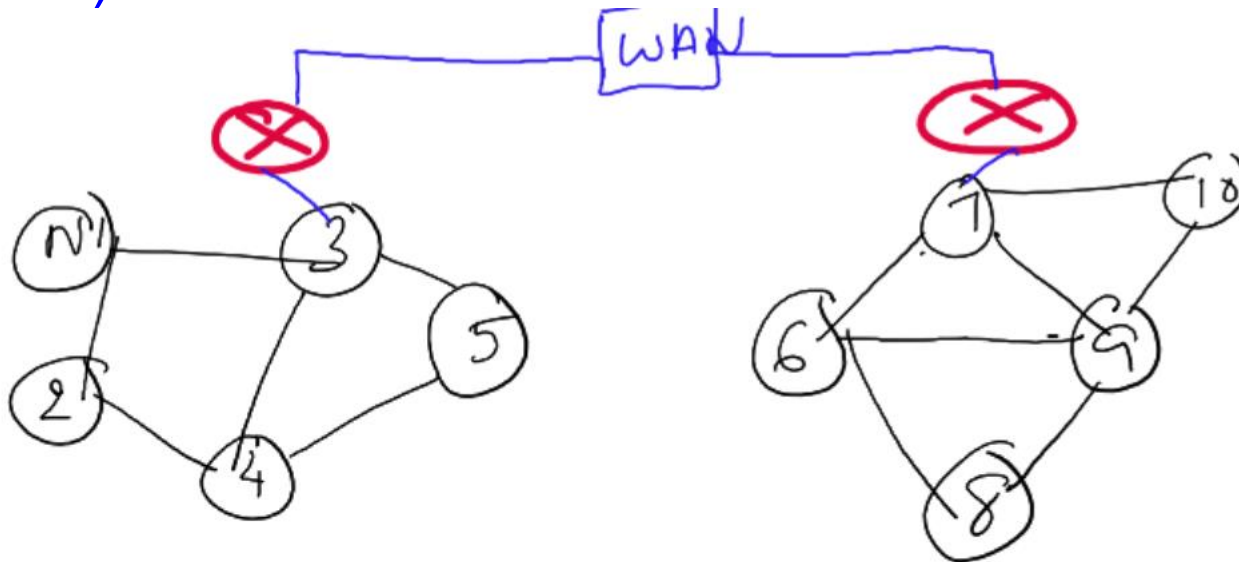
IoT LAN



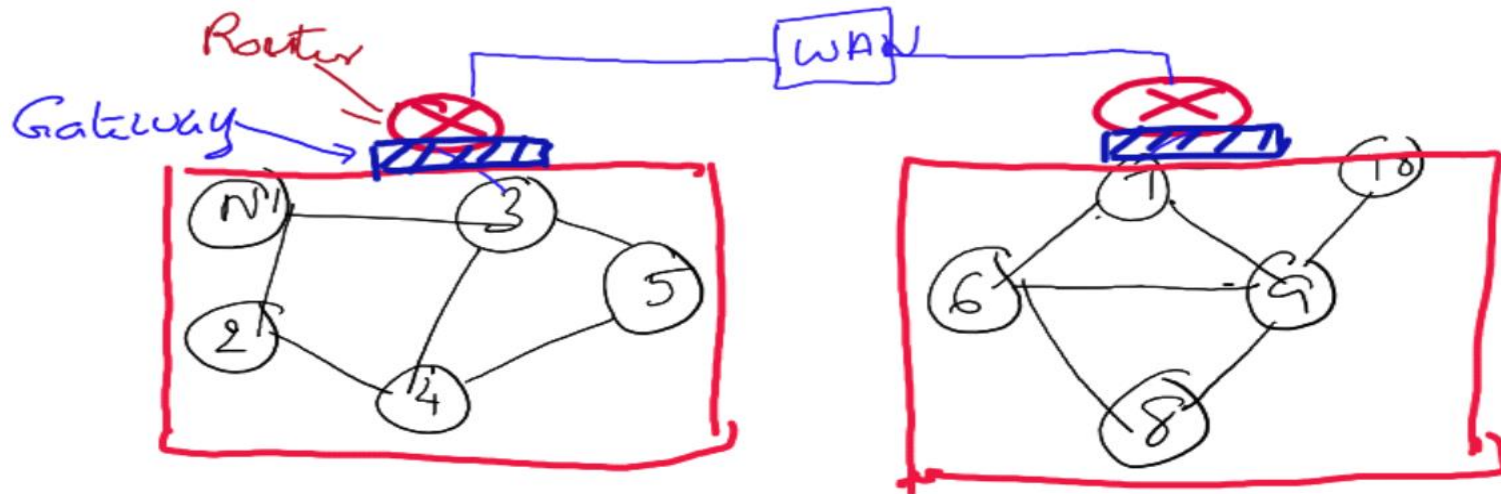
- Here, within a LAN, each sensor node has unique local addresses.

IoT WAN

- Here, nodes have unique addresses on its own network (within a router/gateway)
- The same local unique addresses may be repeated on different LANs (under the purview of other gateways)
- Thus, address allotment can be efficiently done (i.e., reused).



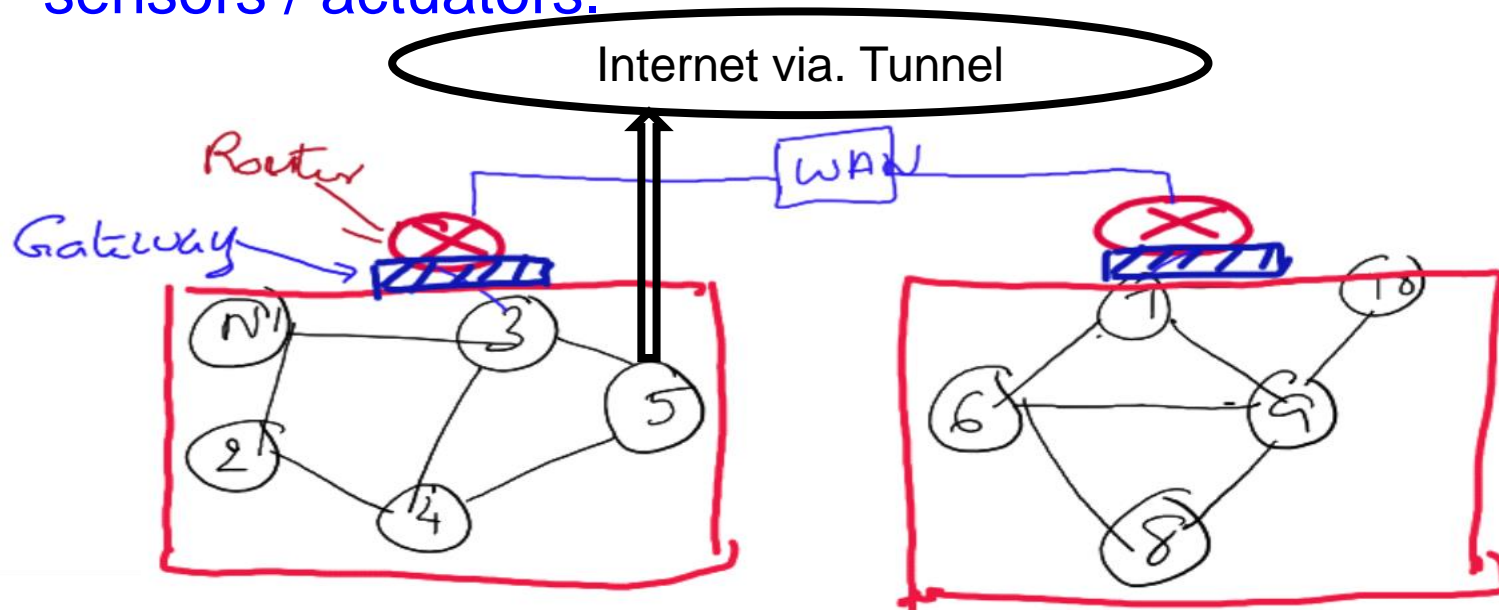
IoT Gateways (Prefix Allotment)



- Compared to normal networks, IoTs have more number of gateways.
- Gateways are utilized to conserve IP addresses.
- As seen in figure, the network connected to the internet has routers with their set of addresses and ranges.
- Generally, gateways preprocess data before they are sent to cloud/internet.

Mobility and Direct Internet connectivity

- Nodes may be directly connected to internet via tunneling mechanism.
- Tunnels are created to remote anchor points (nodes that are GPS enabled).
- For eg. We need to connect to cloud services from sensors / actuators.



HIGHER LAYER PROTOCOLS...

MQTT
SMQTT
CoAP
AMQP
XMPP
DDS

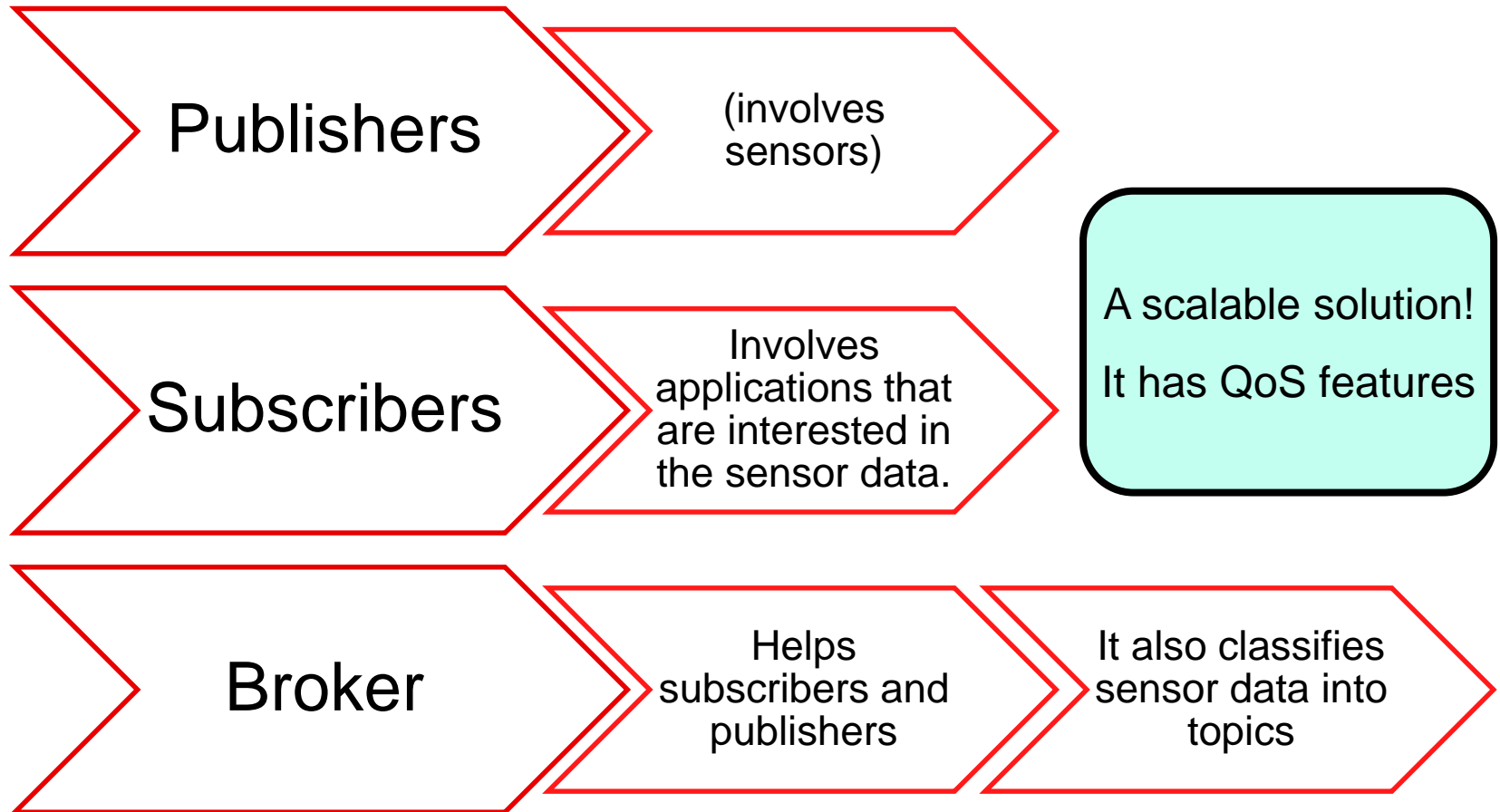
MQTT

- Message Queue Telemetry Transport
 - It is a messaging protocol.
 - It is widely used for M2M communications.
 - It is useful for transporting telemetry data such as sensor data.
- It falls under ISO standard's OASIS standard.
 - Organization for the Advancement of Structured Information **Standards (OASIS)** is framed by a group of consortium – focuses on framing open standards.
- It is a publish/subscribe-based lightweight messaging protocol.
- It is designed to offer connectivity between applications and middleware; and, networks and communications.
- NOTE: the messages are not transferred to end-to end devices – the messages are published to **topics**.

MQTT

- In fact, MQTT was introduced in 1999 by IBM and it was standardized in 2013.
- Several versions exists
 - V3.1.1 (commonly utilized), V5 (currently in limited use).
- MQTT Advantages
 - It is designed for remote connections
 - It utilizes limited bandwidth (typically, message payload is less than 256MB)
 - Small code footprint
- Who uses MQTT?
 - Facebook messenger uses MQTT for online chat.
 - AWS IoT uses MQTT; Instagrams direct message function...
 - Microsoft Azure IoT uses MQTT
 - Everything IoT platform uses MQTT for M2M communications

Three Principal MQTT components










MQTT Brokers

- Message broker

- It is responsible to control the publishing messages and subscribing messages.
- E.g., HiveMQ, Eclipse Mosquitto
- 2 types based on messages and configurations
 - private broker
 - permits only known set of devices.
 - It is also named as self-hosted brokers.
 - We need to setup the broker with static IPs.
 - Public broker
 - – permits messages from any set of devices.
 - - Don't let us to configure anything on their devices
- The brokers are responsible to ensure security by authenticating publishers and subscribers.

MQTT brokers

Name	Broker Address	TCP Port
 Eclipse	mqtt.eclipse.org	1883
 Mosquitto	test.mosquitto.org	1883
 HiveMQ	broker.hivemq.com	1883
 Flespi	mqtt.flespi.io	1883
 Dioty	mqtt.dioty.co	1883
 Fluux	mqtt.fluux.io	1883
 EMQX	broker.emqx.io	1883

<https://mntolia.com/10-free-public-private-mqtt-brokers-for-testing-prototyping/>

Crucial Characteristics of MQTT Brokers

- QoS Support

- We need to ensure whether QoS support is provided by the MQTT broker.
- i.e., the message may be lost in the broker.

- Opensource

- Opensource enables better scope for future developments. But, the challenge would be to ensure if the broker has support for long years.

- Performance

- Latency – message delivery time
- Throughput – no.of messages processed within a specified time.
- Scalability – more number of resources with increasing messages.
- Availability – Broker availability in a consistent manner.

Crucial Characteristics of MQTT Brokers

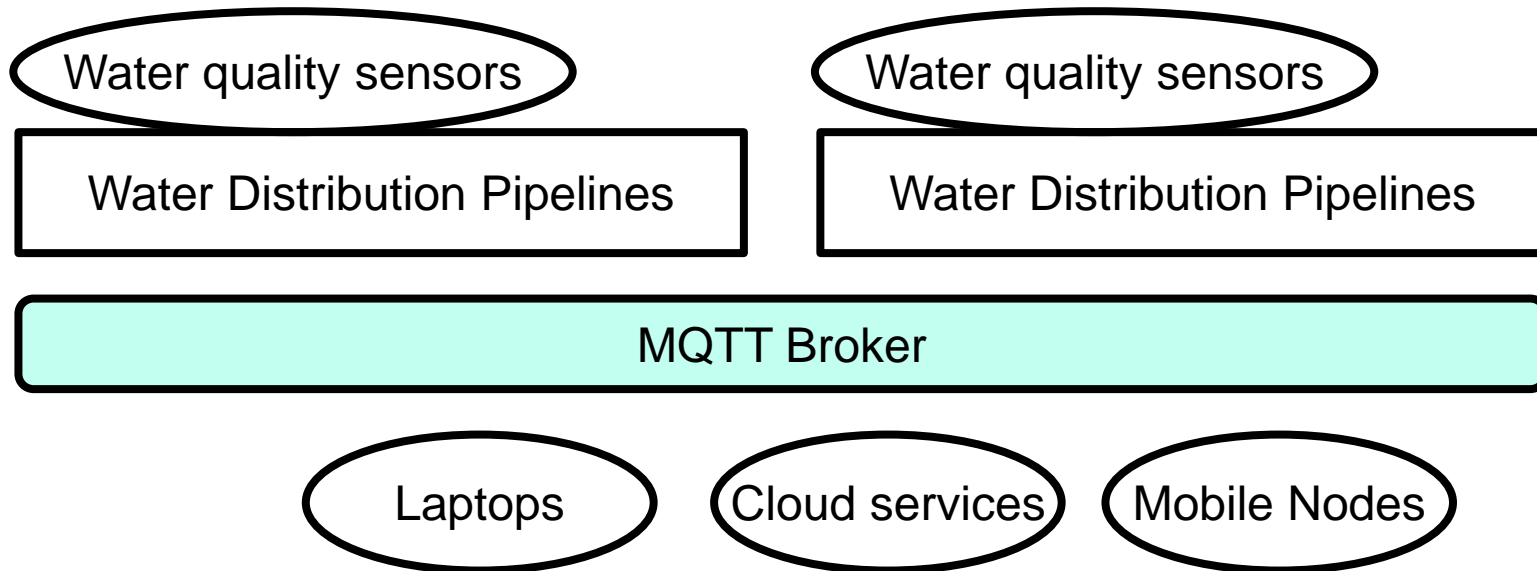
- Interoperability

- Backend integration with other software or services.
- Integration with the other MQTT brokers.

- Security

- In public/private brokers, enabling proper authentication methods.
- For example, token authentication methods.
- Avoiding any attacks such as DoS attacks.
- Providing custom authentication methods (but, still the messages are properly reached the clients).

MQTT connection - Example



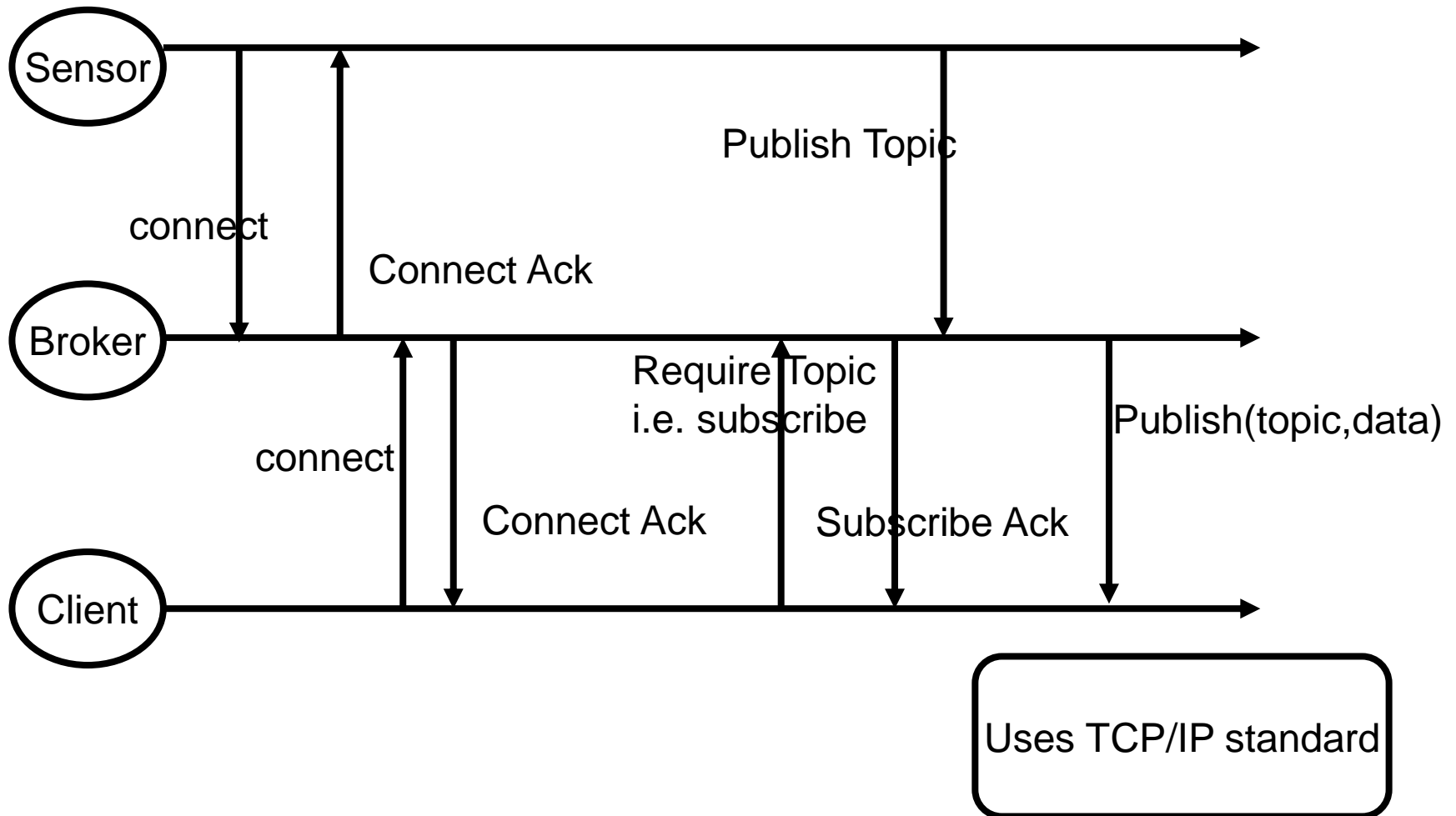
- Figure describes the publish-subscribe model...
- WQ. sensor publishes the pH or similar values.
- MQTT broker will receive subscriptions from various services
- Now, MQTT broker publishes sensor data to different subscribers.

MQTT Methods

- Connect
 - Helps to connect sensors with the server
- Disconnect
 - Helps to disconnect sensors with the server
- Subscribe
 - Helps to subscribe sensor data for application services
- Unsubscribe
 - Helps to unsubscribe sensor data for application services
- Publish
 - Helps sensor to submit or publish sensor data to brokers so that they can be fetched by subscribing applications.

MQTT – Connection Approach

- Publish using topics...



MQTT Topics

- MQTT topics are a form of addressing that allows MQTT clients to share information.
- A topic is a string (UTF-8 string) that can have more hierarchy levels which are separated by a slash. For eg.
 - house/bath-room/waterquality
 - For sending wq data of bath room No.1.
- Topics are of two kinds
 - Exact topic
 - A string of information that discusses about an exact location
 - Eg. house/bath-room/waterquality
 - Wildcard topic
 - A string of information that discusses about multiple locations.
 - Wild card topics are of two types: Single level wildcard and Multi-level wildcard.
 - Eg. **house/+waterquality** --- (which deals with house/bath-room/waterquality or house/kitchen/waterquality)
 - Or, Eg. house/# --- multilevel wildcard (deals with all topics under the topic house/)

MQTT Topics

- Topics can be named by **your own names** without specific restrictions.
- However, a topic starting with '\$' symbol is restricted for other uses.
- They are often utilized to represent some status of messages. For e.g.,
 - \$SYS/broker/clients/connected
 - \$SYS/broker/clients/disconnected
 - \$SYS/broker/clients/total
 - \$SYS/broker/messages/sent
 - \$SYS/broker/uptime

SMQTT

- Secure MQTT
- It is an extension of MQTT.
- It is similar to `::→ http and shhttp`.
- SMQTT uses encryption based on lightweight encryption.
- The SMQTT algorithm consists of four stages:
 - Setup
 - Encryption
 - Publish
 - Decryption

Phases or Stages of SMQTT

- Setup phase

- The publishers and subscribers register to the broker and get a master secret key according to the choice of the developer's encryption algorithm.

- Encryption phase

- During the encryption phase, the data is encrypted by the BROKER.

- Publish phase

- The encrypted data are sent to the subscribers.

- Decryption phase

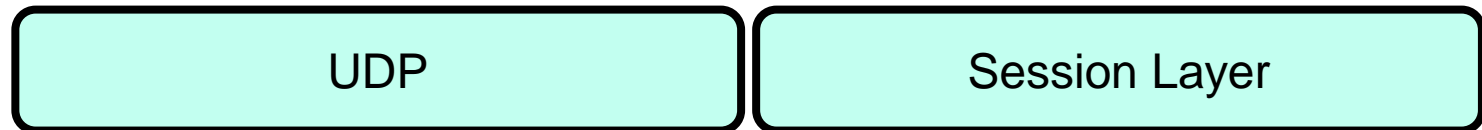
- Here, the subscribers are responsible to decrypt the data using the master secret key.

CoAP

- Constrained Application Protocol
- Mostly, it is utilized for web transfers within constrained nodes (ie. Power conscious nodes)
- Mostly, Machine to Machine communications
 - Eg. Smart energy, building automation, smart manufacturing
- CoAP operates in a Request-Response model (ie. Client-server model) between endpoints.
- CoAP works only on UDP based communications (Hence, on asynchronous machines).

CoAP

- CoAP mostly operates in session layer. But, it could also be utilized in the application layer.
 - Typically, it is utilized to translate to HTTP so that the integration with internets is easier.
 - NOTE: CoAP is a session layer protocol (But, a thin difference between the session layer and application layer).



CoAP Nodejs example

```
### Opens UDP server
var coap    = require('coap')
    , server = coap.createServer() -----> OPENS A UDP server

server.on('request', function(req, res) {    -> REQ-RESPONSE MODEL
    res.end('Hello ' + req.url.split('/')[1] + '\n')
})
```

```
### Sends coap message to the server
// the default CoAP port is 5683
server.listen(function() {
    var req = coap.request('coap://localhost/Mess')

    req.on('response', function(res) {
        res.pipe(process.stdout)
        res.on('end', function() {
            process.exit(0)
        })
    })
    req.end()
})
```

Node-Coap library

Points to Observe:

1. Port number
2. Request-Response model
3. Request is given to coap://xxxxxx
4. Two parts: Req/Res; and, Messaging

<https://github.com/mcollina/node-coap>

CoAP

- CoAP is designed by IETF → by the Constrained RESTfull Environment (CoRE working group)
- The objective of the working group is to provide a lightweight **REST-based** http interface.
- Note: REST is a standard **interface between HTTP** client and servers.
- REST is good for normal internet communications.
- But, CoAP meets the needs of REST on IoT communications with reduced power consumptions.

SOAP vs. REST

SOAP

REST

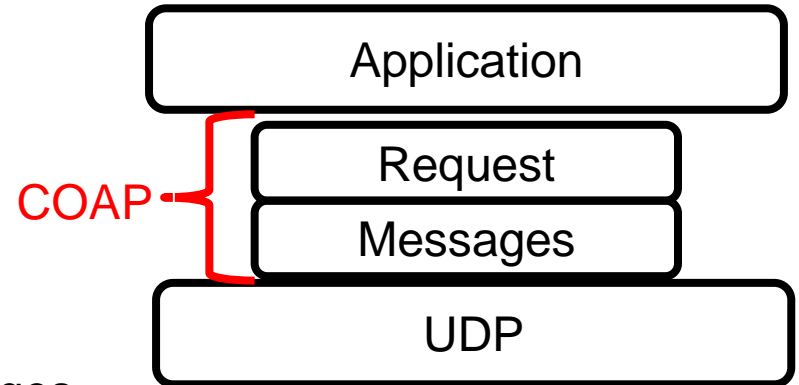
{"city":"Mumbai","state":"Maharashtra"}

```
<?xml version="1.0"?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV
    ="http://www.w3.org/2001/
12/soap-envelope"
  SOAP-ENV:encodingStyle
    =" http://www.w3.org/2001
12/soap-encoding">
  <soap:Body>
    <Demo.guru99WebService
      xmlns="http://tempuri.or
g/">
      <EmployeeID>int</Emplo
      yeeID>
    </Demo.guru99WebServic
    e>
  </soap:Body>
</SOAP-ENV:Envelope>
```

SOAP needs more bandwidth...!

CoAP Layer

- CoAP lies between the transport layer and the application layer.
- CoAP has two sublayers:
 - Request Response
 - Deals with real communications
 - Reliability message
 - Deals with duplication of messages
- CoAP includes reliability messages as they operate using UDPs (unreliable protocol).



CoAP messages

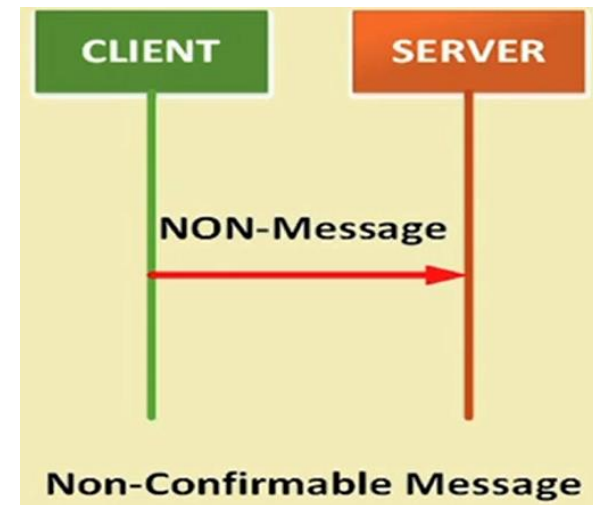
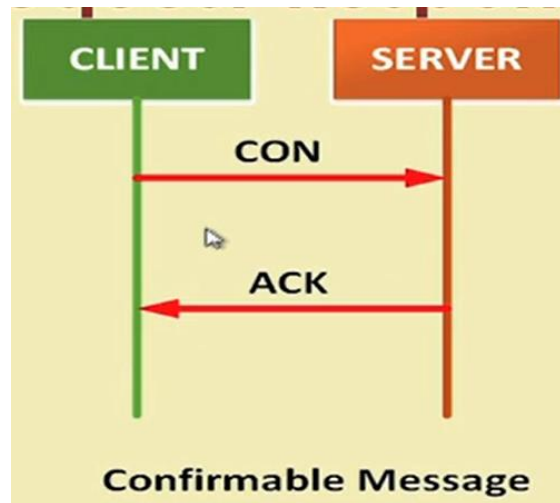
- CoAP message types

- Confirmable message

- Here, a message is sent and the ACK is received between power constrained machines/nodes.
 - Similar to http, CoAP also has GET, PUSH, DELETE, UPDATE..

- Non-confirmable message

- Vice-versa – no ACK messages are involved in communications.



CoAP messages

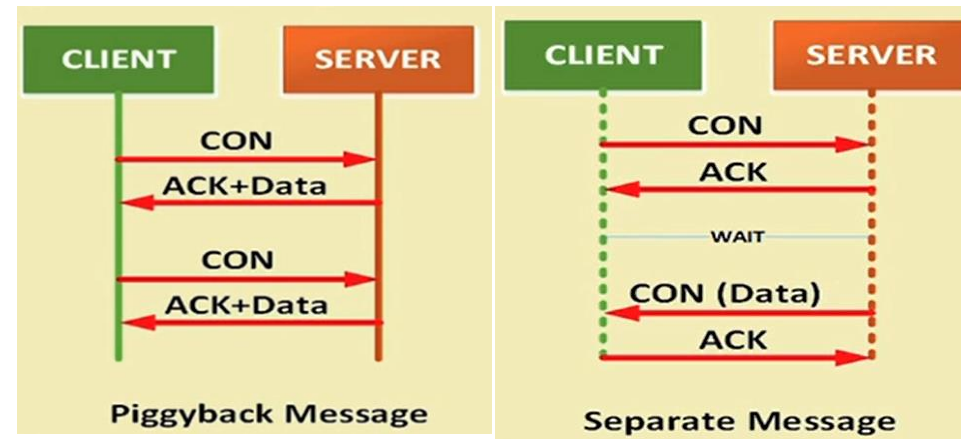
- CoAP messages

- Piggyback

- Here, along with the ACK message the data is also sent to the client.

- Separate

- Here, ACK message and the data messages are sent separately by the server to the client.



- Similar to HTTP, CoAP utilizes GET, PUT, PUSH, DELETE message requests to retrieve, create, late, and delete messages.

XMPP

- XMPP – Extensible Messaging and Presence Protocol.
- Developed in 1999 by the Jabber open source community (now, with XMPP Foundation).
- XMPP is originally conceived under the name Jabber.
- It is targeted for **instant messaging** applications.
- It is a communication protocol extended from XML.
- It is useful for
 - instant messaging,
 - multi-party chat, and
 - routing of XML data.

XMPP

- Open standard protocol
 - – no permissions are required for implementing this protocol-based IoT applications.
 - Or, no permissions are required for extending the standard.
- It enables the near-real-time exchange of structured data between any two or more network entities.
- In general, XMPP is used for unstructured data.

XMPP XML Payload -- EXAMPLE

```
<?xml version='1.0'?>
```

```
<stream:stream
```

```
  from='example.com'
```

```
  id='someid'
```

```
  xmlns='jabber:client'
```

```
  xmlns:stream='http://etherx.jabber.org/streams'
```

```
  version='1.0'>
```

```
<!-- Encryption, Authentication, and Resource binding omitted -->
```

```
<message from='test@gmail.com '
```

```
  to='test@gmail.com '
```

```
  xml:lang='en'>
```

```
  <body>Body part of XMPP goes here!</body>
```

```
</message>
```

```
<message from='test@gmail.com'
```

```
  to='test@gmail.com '
```

```
  xml:lang='en'>
```

```
  <body>Body part of XMPP goes here!</body>
```

```
</message>
```

```
</stream:stream>
```

```
</stream>
```

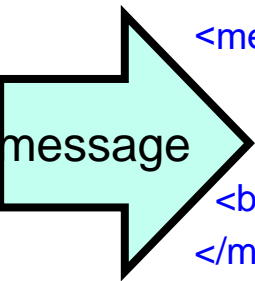
RFC6120 and RFC6121

Under IETF

Default port: 5222

Points to observe:

1. Designed using XML **stream** tags
2. Messages are sent using message tag



XMPP Payload parts

- Connection via. streams
- Messaging via. tags
- Any clients could connect to XMPP servers

- <stream:stream
- from='client-a@myxmppserver.com'
- to='myxmppserver.com'
- version='1.0'
- xmlns='jabber:client'
- >

Client A
Connected to
XMPP server

- <stream:stream
- from='client-b@myxmppserver.com'
- to='myxmppserver.com'
- version='1.0'
- xmlns='jabber:client'
- >

Client A
Connected to
XMPP server

XMPP Payloads

- Client A sends near-real time message to Client B
 - <message to=' **client-b**@myxmppserver.com '>
 - <body>Enjoy the day! </body>
 - </message>
- Client B receives the instant message from Client A
 - <message from=' **client-a**@myxmppserver.com '>
 - <body>Enjoy the day! </body>
 - </message>
- Client A terminates the stream
 - </stream:stream>

XMPP – contd.

- It uses client server architecture for near-real time instant messaging.
- A **decentralized model** – ie. No central server is required (as like brokers in MQTT).
 - i.e., any node can perform as server and any node can act as client for an application.
 - This feature improves the security of the communications.
- XMPP is highly interoperable (i.e., easily communicates with the other protocols).
- It uses gateways (other higher layers) to communicate with the other internet protocols.
- With XMPP, we could communicate with
 - Servers, messaging services such as Yahoo chat, Or, other intranets via TCP.

XMPP technologies - Extensions

<https://xmpp.org/extensions/>
Around 200 extensions....!

- Core

- Information about the core XMPP technologies for XML streaming

- Jingle

- XMPP used for multimedia signaling (voice, video, file transfer) <https://xmpp.org/extensions/xep-0166.html>

- Multi-user chat

- XMPP used for multiparty communications.

- PubSub

- XMPP used for alerts and notifications

- BOSH

- XMPP used for binding with HTTP

Drawbacks of XMPP

- It doesn't offer any QoS to communications.
- It doesn't provide end-to-end security (or, encryption methods).
- Obviously, due to these limitations, XMPP was not so successful in industrial practices.

AMQP

- Advanced Message Queuing Protocol.
- It was originated in 2003.
- It is an open standard application protocol which follows the publish/subscribe model.
- It is used for passing messages between applications and/or organizations (i.e., **pt to pt communication**).
- RabbitMQ (a message broker) supports AMQP and MQTT protocols.

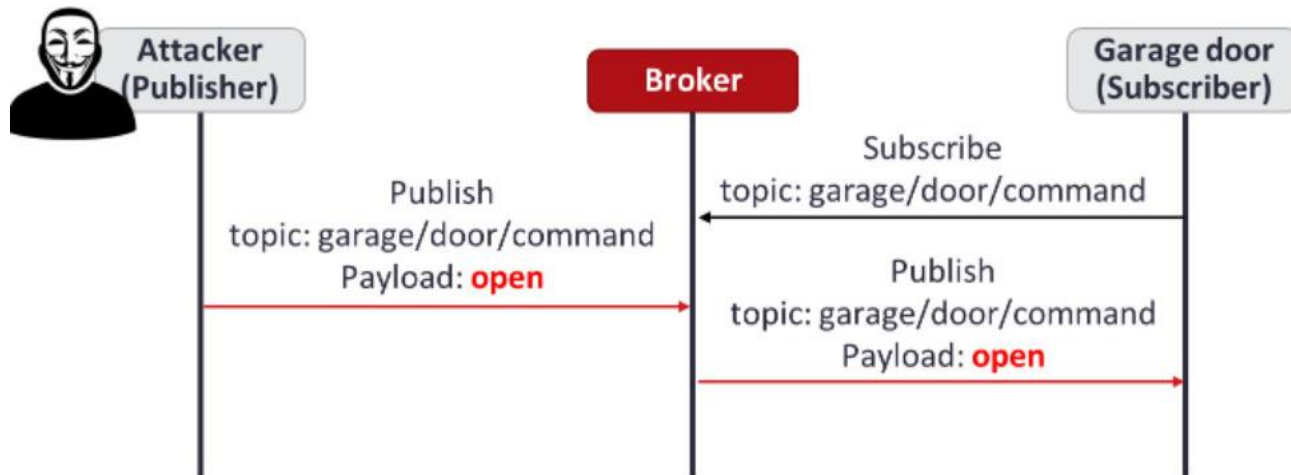
ROUTED Communication while following publish/subscribe

And, involving brokers

Analogous – Buffae system with name specified in dishes

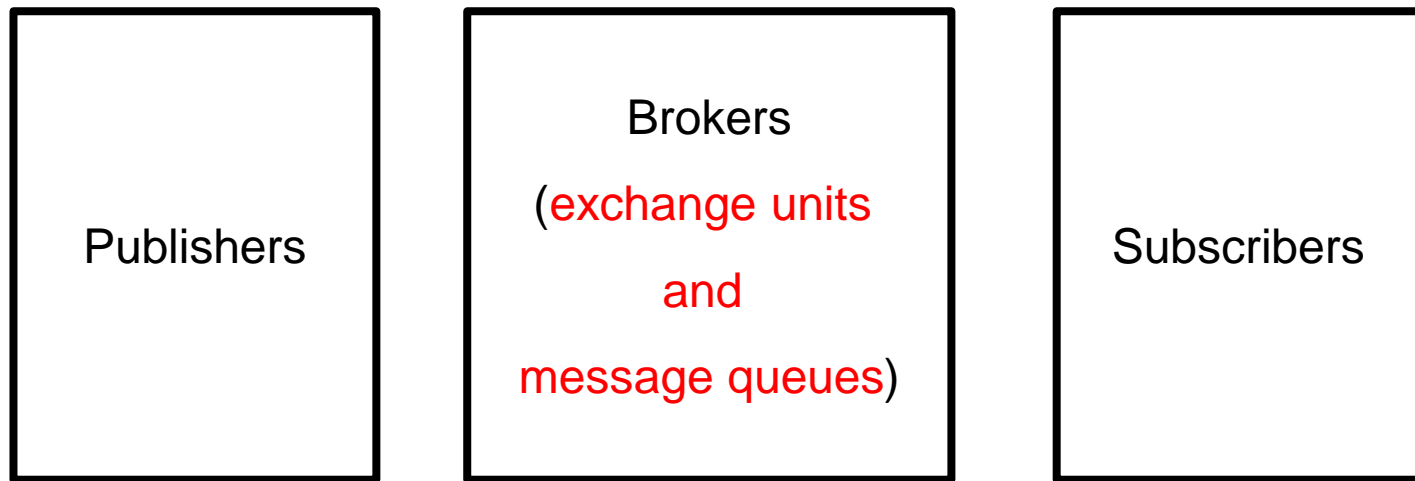
Drawbacks of Open Publishers and MQTTs

- There are typically around 47000 exposed MQTT brokers.
- If the publishers are not configured properly, they send risky topics.
- These topics could be subscribed using wildcards #



<https://www.txone-networks.com/blog/content/MQTT-Series-2-Potential-risks-of-exposed-MQTT-brokers>

AMQP architecture



- Difference between MQTT and AMQP is in the architecture of brokers.
- Here, brokers include two components -- exchange units and queues.
- Exchange units include routing mechanisms to ensure message delivery. (Hence, it achieves better QoS)

AMQP Exchange Methods

- **Default Exchange**
 - Routes message to a queue.
- **Direct Exchange**
 - Routes message to a queue (using routing key).
- **Fanout Exchange**
 - Routes message to more no.of queues (using publish/subscribe)
- **Topic Exchange – A default method**
 - Routes message to a queue (like a topic – ie., a pattern)
- **Header Exchange**
 - Routes message to a queue (based on header filters)

AMQP

- Here, Message Delivery guarantees are classified in three types
 - At-most-once – each message is delivered once or never.
 - At-least-once – each message is certain to be delivered; but, may do so multiple times.
 - Exactly-once – each message will always certainly arrive and do so only once.
- (AS SIMILAR TO MQTT)...

AMQP

- It is a **binary application layer protocol**.
- Basic unit of data is called as **FRAME** (different from data link layer) – Here, it is in Application layer.
- AMQP assures good QoS.
- In fact, AMQP emerged from the financial sector for assuring complete transactions.

DDS - Protocol

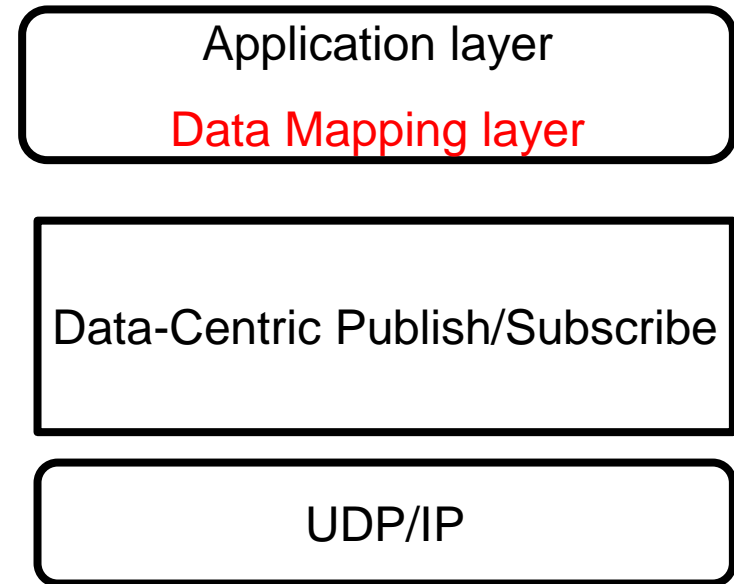
- Data Distribution Service protocol
- It is a data-centric messaging protocol.
 - i.e. it tries to provide i) what to describe? ii) who to describe? And iii) the current state of the description.
 - E.g., organizing calendars by a group of people might **need an architecture** to clearly express the state of data rather than an application.

DDS-Protocol

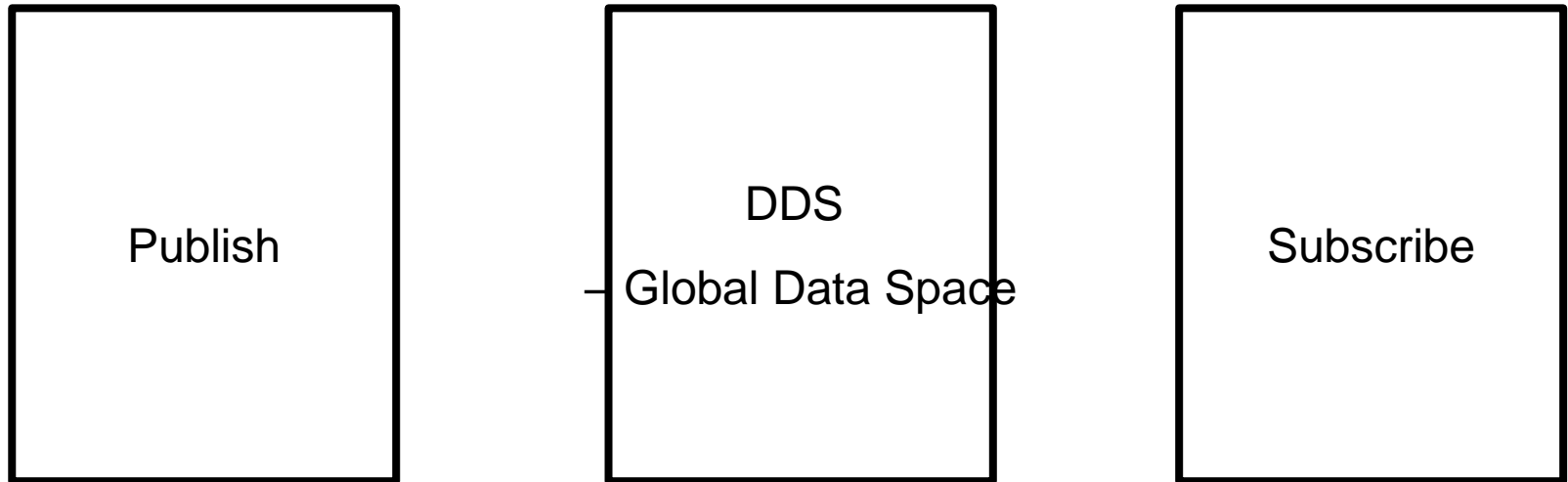
- It is also a service-level protocol in the application layer of IoT stack.
 - i.e, it may need to reconstruct the state of the previous messages and provide the apt content to the intended receiver.
- It applies a broker-less architecture in IoT.
- It is a protocol designed for M2M communications (especially, to stream data between powerful machines).

DDS Protocol Stack

- It utilizes UDP protocol to publish/subscribe messages.
- There are two versions:
 - DDSv1.2
 - DDSv2.1
- DDS V1.2 API standard is language independent, OS and HW architecture independent.
- DDS V2.1 is a standard wire protocol which allows interoperability between different implementations of S standards.



DDS Working Methodology



- DDS utilizes distributed GDS (Global Data Space).
- GDS specifications make it fully distributed.
- By this, it attempts to avoid any single point of failure or bottleneck while delivering messages.
- In DDS protocol architecture, applications can autonomously and asynchronously read/write data in GDS.