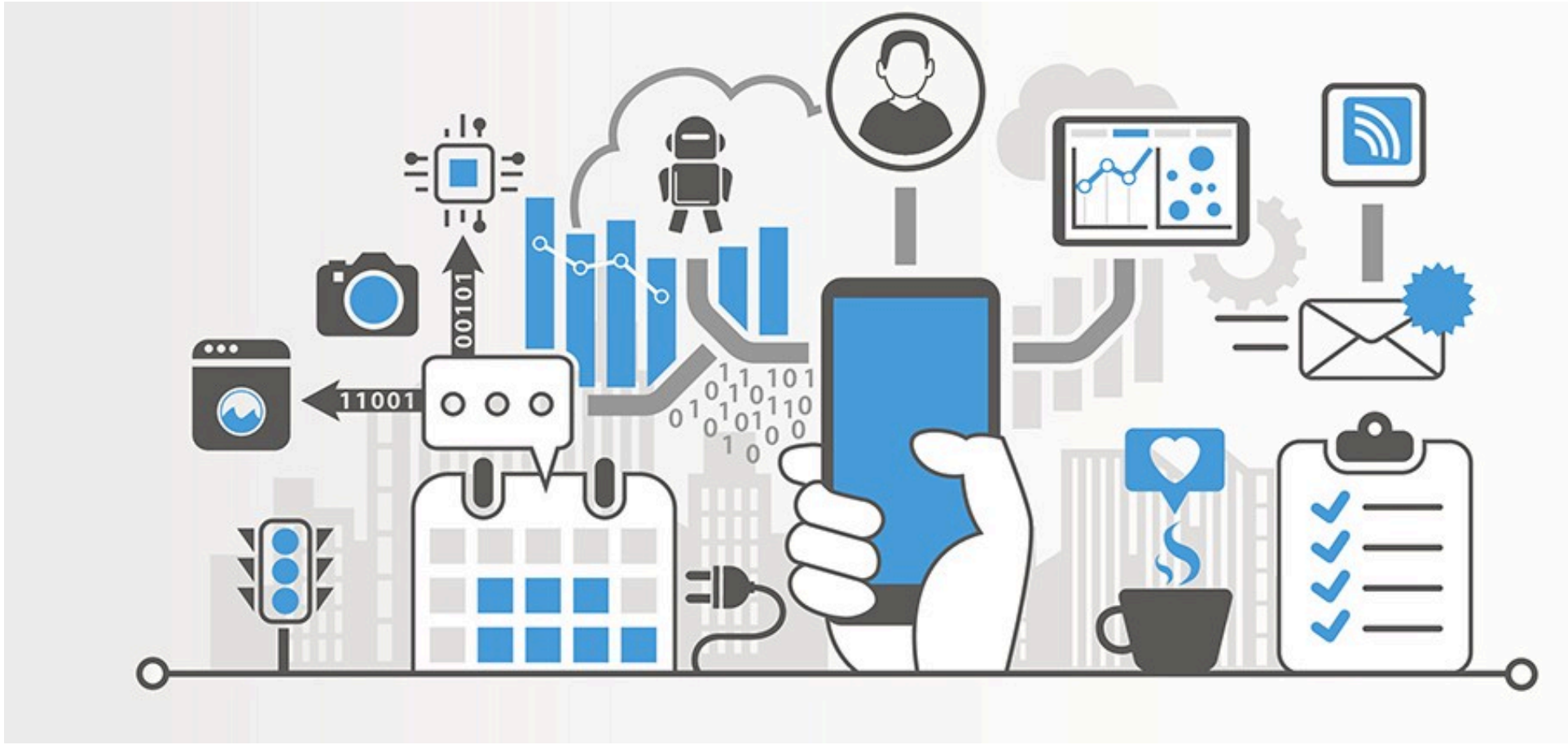
Programming for IoT Applications

Edoardo Patti

Lecture 7



COMMUNICATION PARADIGMS AND PROTOCOLS

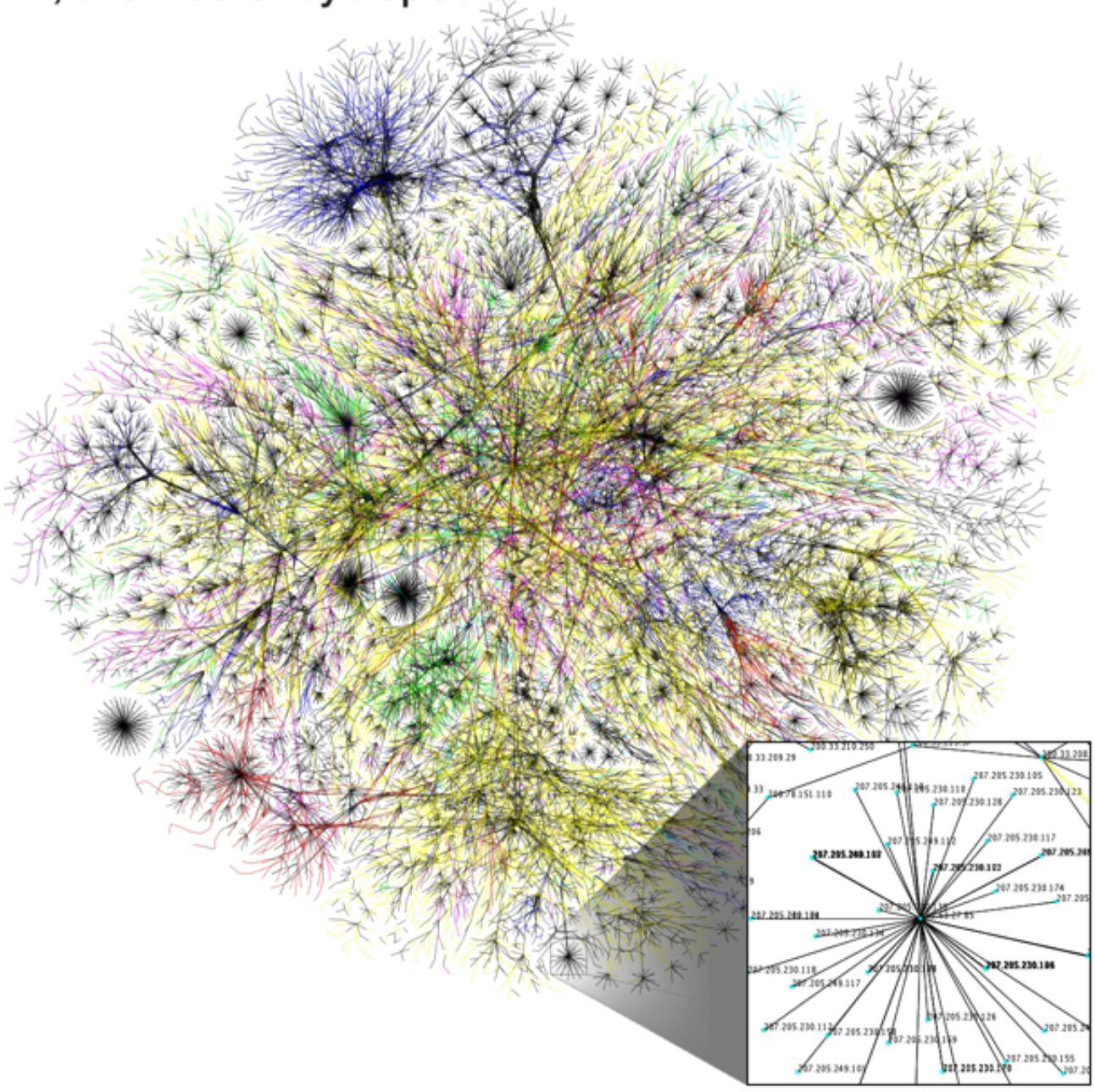
Internet

From Wikipedia, the free encyclopedia

The **Internet** (portmanteau of **interconnected network**) is the global system of interconnected computer networks that use the Internet protocol suite (TCP/IP) to link devices worldwide. It is a *network of networks* that consists of private, public, academic, business, and government networks of local to global scope, linked by a broad array of electronic, wireless, and optical networking technologies.

Internet

From Wikipedia, the free encyclopedia

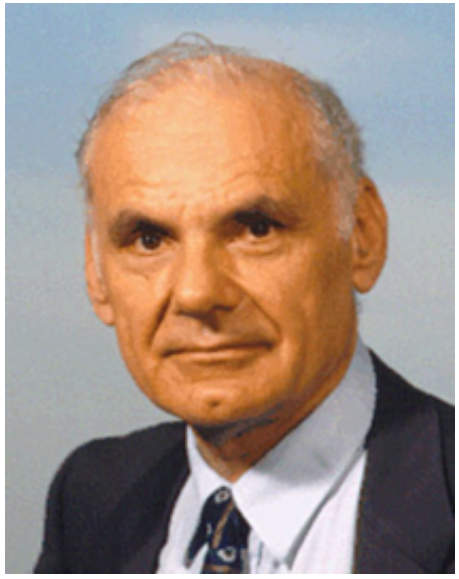


The evolution of the Internet

1965

Lawrence Roberts and Thomas Merril communicate with one another using computers connected via a low-speed dial-up telephone line in Massachusetts and Californiav, creating the first Wide Area Network and laying the groundwork for the internet.

1965



Lawrence Roberts

The evolution of the Internet

1969

ARPAnet, the first version of the internet, is created and used to link computers at UCLA and Stanford.

1965

1969

BBN ARPANET Group.

The evolution of the Internet

1965

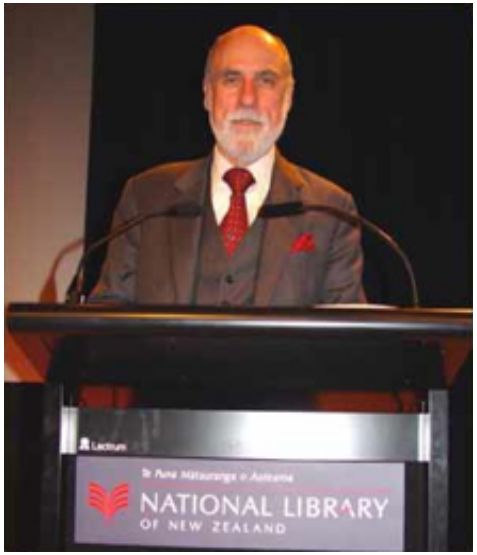
1969

1974

Robert Kahn and Vinto Cerf publish "A Protocol for Packet Network Communication" laying the groundwork for TCP/IP and much larger, interconnected computer networks. The word "internet" is first used.

1974

Robert Kahn



Vinto Cerf

The evolution of the Internet

1965

1969

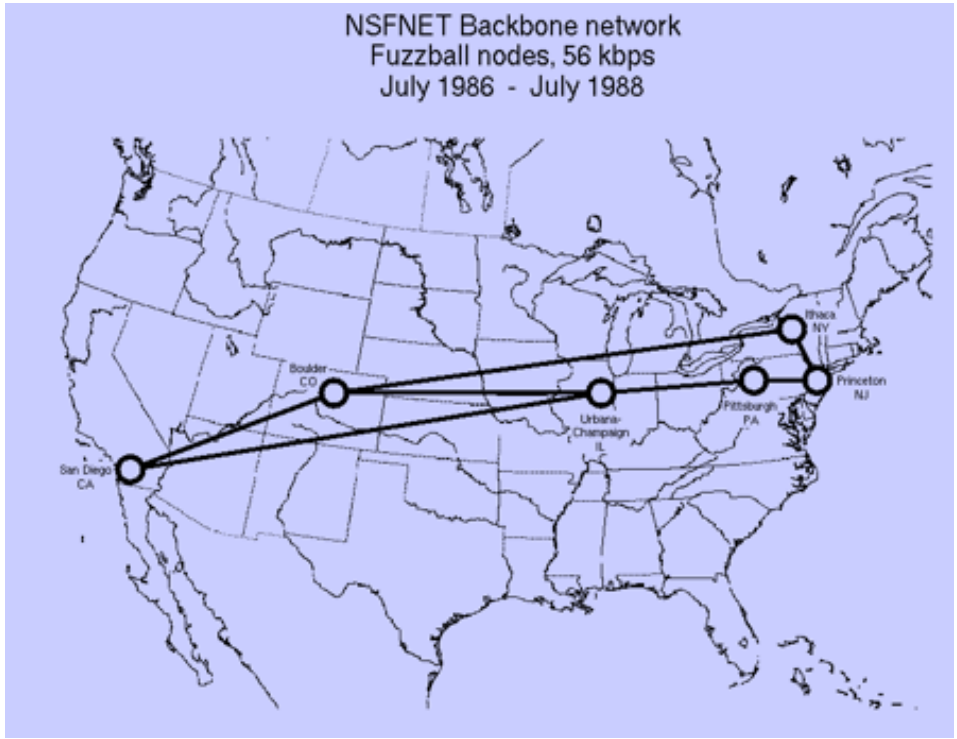
1974

1986

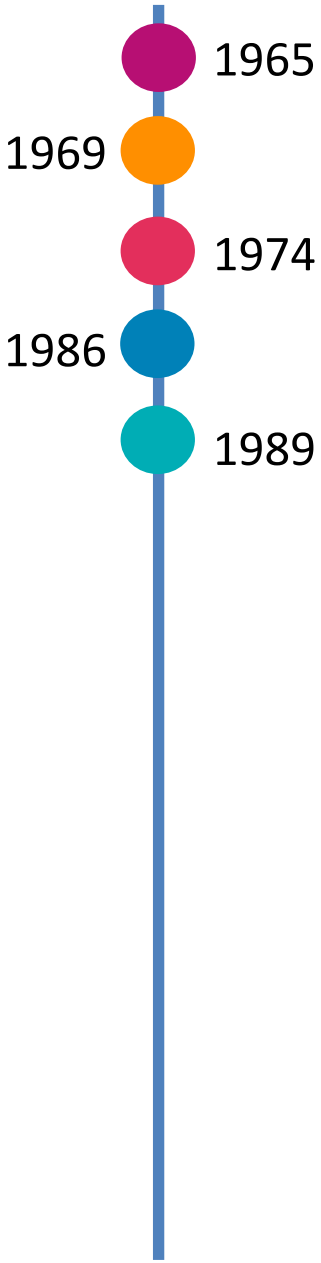
National Science Foundation Network (NSFNET)

1986

NSFNET is founded, creating the backbone and providing the investment needed to create the internet as we know it today.



The evolution of the Internet



Internet Service Provider (ISP)

The first commercial ISP, The World, launches.

1989

The evolution of the Internet

1965

1969

1974

1986

1989

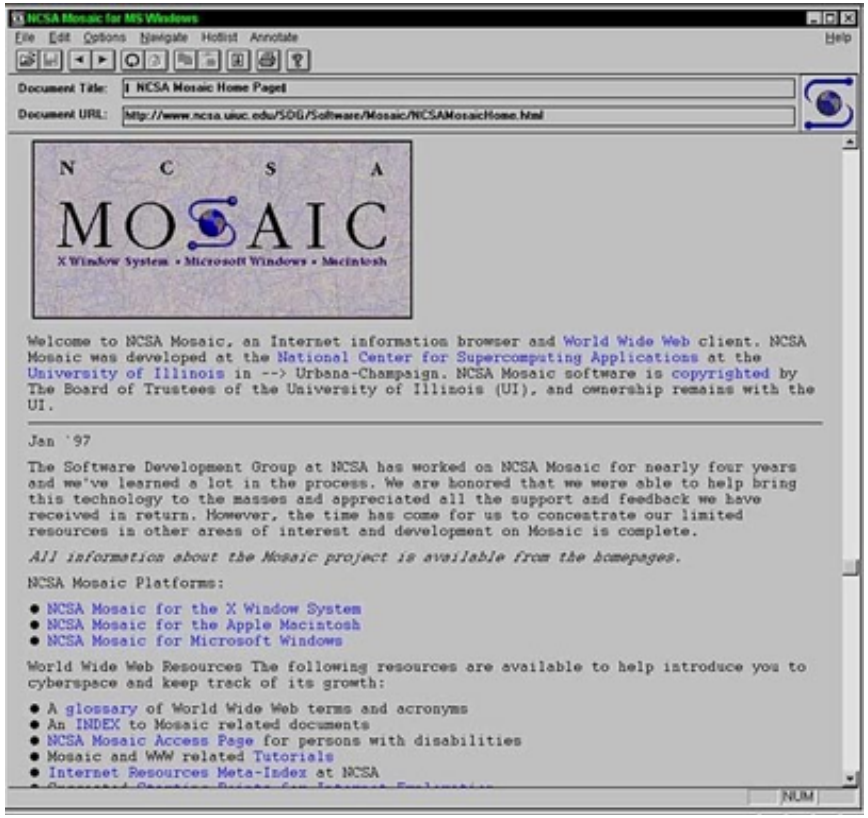
1989

1989

Tim Berners-Lee invents the world wide web and the first web browser, which opens the door for the internet to go mainstream.

Tim Berners-Lee

The evolution of the Internet



1965

1969

1974

1986

1989

1989

1993

The first web browser available to the public, Mosaic, launches.

1993

The evolution of the Internet

1965

1969

1974

1986

1989

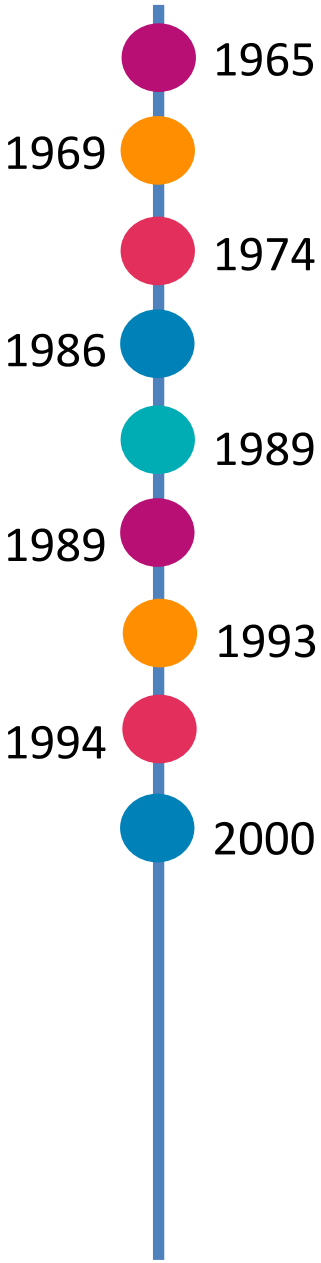
1989

1993

1994

1994 Netscape Navigator is released in stores.

The evolution of the Internet



Some 300 million people around the world are officially online.

2000

The evolution of the Internet

2004

Facebook launches and the Web 2.0 begins to grow rapidly.

1965
1969
1974
1986
1989
1989
1993
1994
2000
2004

The evolution of the Internet

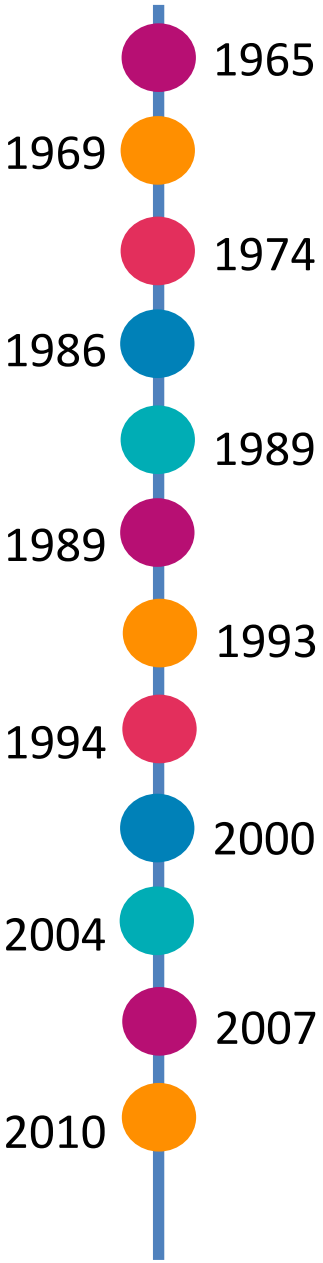
1965
1969
1974
1986
1989
1989
1993
1994
2000
2004
2007

January 9th, 2007, Steve Jobs presents the iPhone at Macworld in San Francisco

The iPhone is released, giving rise to the mobile revolution.

2007

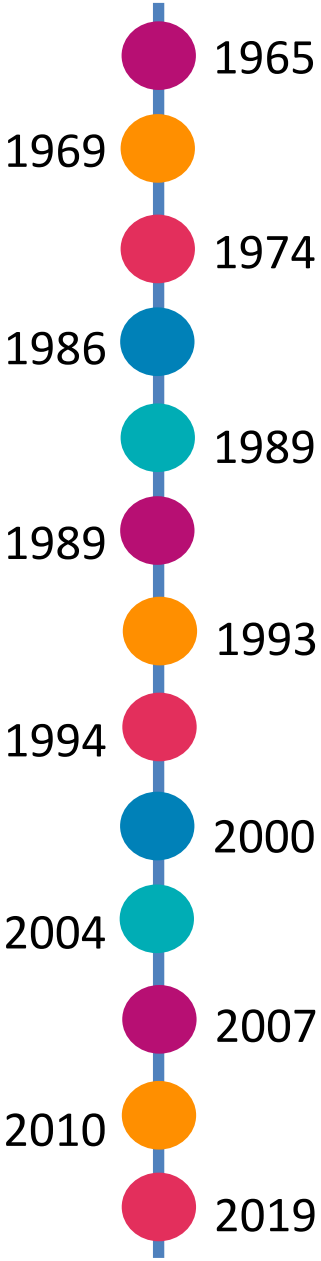
The evolution of the Internet



2010

1.966 billion people are online worldwide.

The evolution of the Internet



4.383 billion people are online.

2019

The evolution of the Internet of Things

1982

Researchers at Carnegie Mellon University connect a vending machine to the Internet, allowing them to check if the machine has cold sodas before going to purchase one. **This is often cited as one of the first IoT devices.**

The evolution of the Internet of Things

1982

1990

John Romkey, in response to a challenge, connected a **toaster** to the Internet and was able to successfully turn it on and off

The evolution of the Internet of Things

1982

1990

1995

The first version of the **GPS** satellite program is finally, a big step towards proving one of the most vital components for many IoT devices: location.

The evolution of the Internet of Things

1982

1990

1995

1998

IPv6 becomes a draft standard, enabling more devices to connect to the internet than previously allowed by IPv4. While IPv4 only provides unique identifiers for around 4.3 billion devices, IPv6 has unique identifiers for up to 2^128, or **340 undecillion devices**. (That is 340 with 36 zeroes!)

The evolution of the Internet of Things

1982

1990

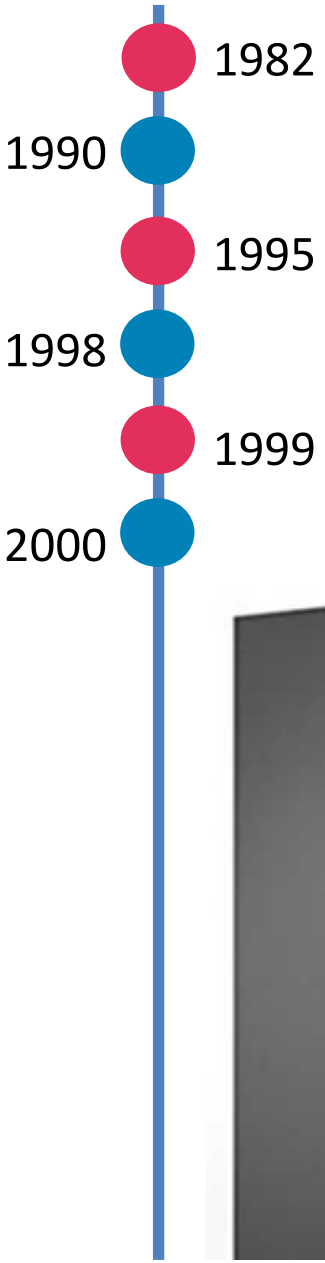
1995

1998

1999

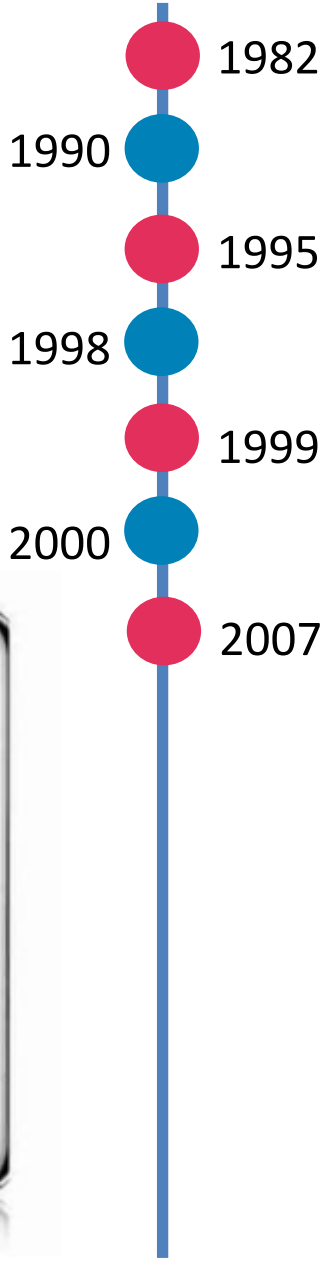
It is when the term **Internet-of-Things (IoT)** was probably first used. Kevin Ashton included it in a presentation to Proctor & Gamble executives as a way to illustrate the potential of RFID tracking technology.

The evolution of the Internet of Things



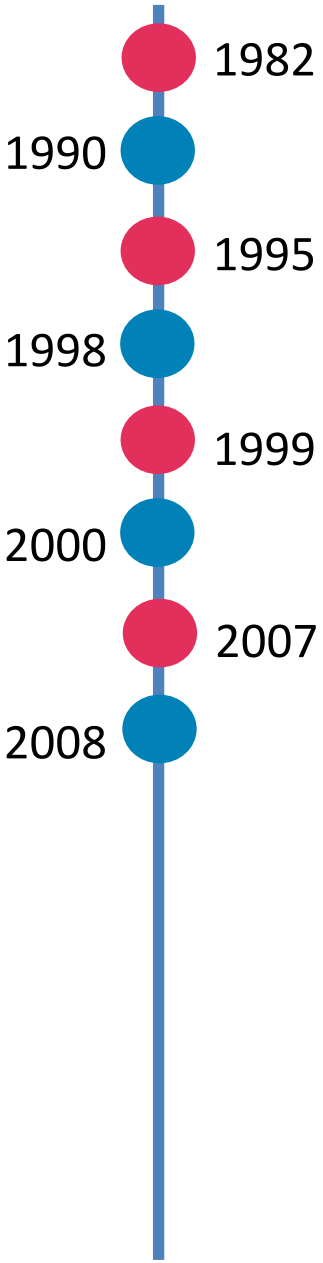
LG announces what has become one of the quintessential IoT devices: the **Internet refrigerator**. It was an interesting idea, complete with screens and trackers to help you keep track of what you had in your fridge, but its about $20,000 USD price tag did not earn it a lot of love from consumers.

The evolution of the Internet of Things



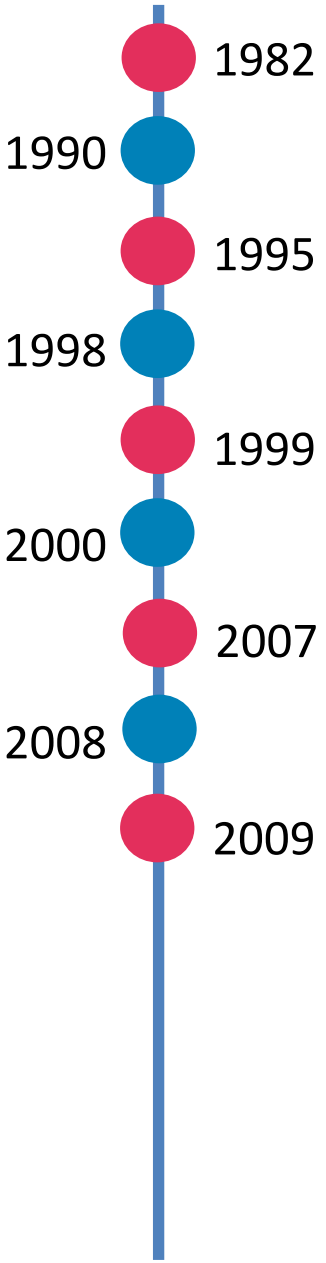
The **first iPhone** appears on the scene, offering a whole new way for the general public to interact with the world and Internet-connected devices.

The evolution of the Internet of Things



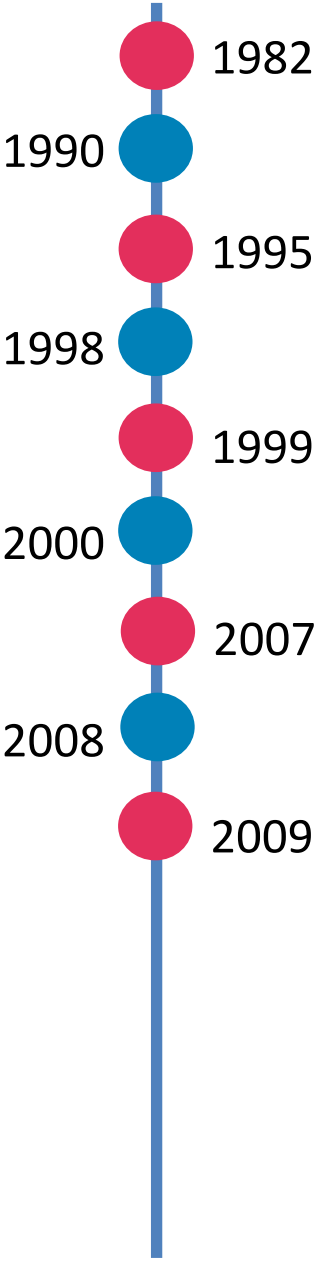
The first international IoT conference is held. The year is fitting, since it is also the first year that **the number of Internet-connected devices grew to surpass the number of humans on earth**.

The evolution of the Internet of Things



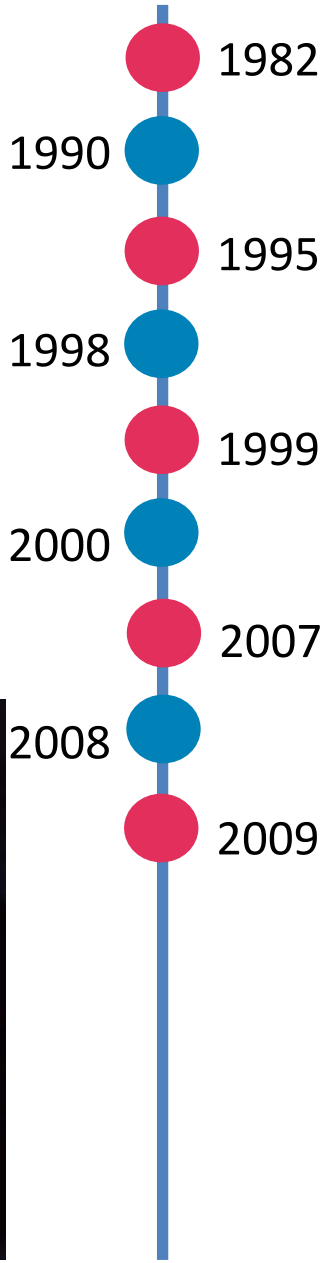
1) Google starts **self-driving car** tests.

The evolution of the Internet of Things



1) Google starts **self-driving car** tests.
2) St. Jude Medical Center releases **Internet-connected pacemakers**. St. Jude's device will go on to make yet more history by being the first IoT medical device to **suffer a major security breach in 2016** (without casualties, fortunately).

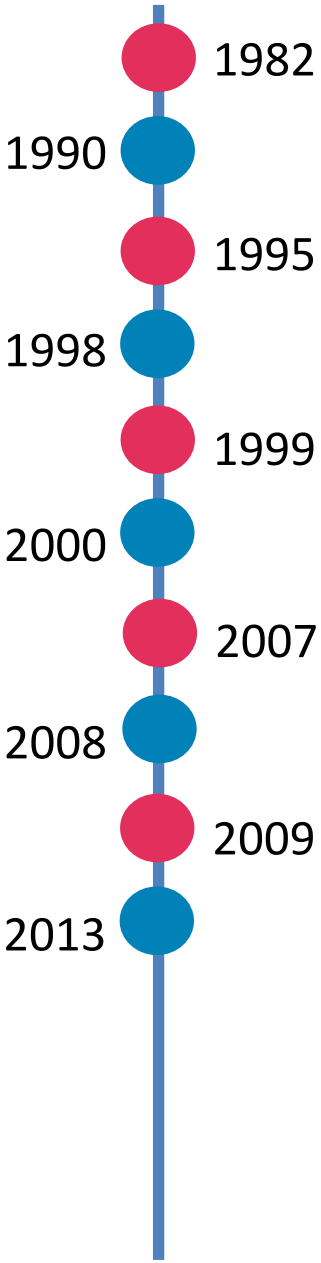
The evolution of the Internet of Things



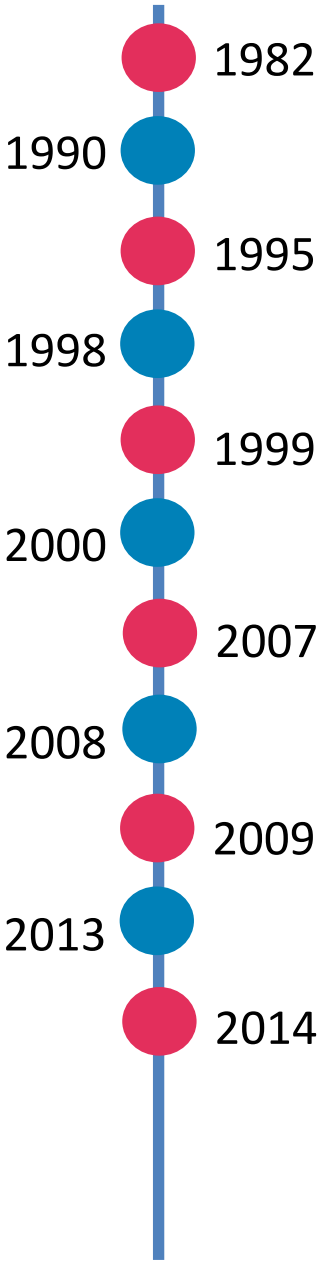
1) Google starts **self-driving car** tests.
2) St. Jude Medical Center releases **Internet-connected pacemakers**. St. Jude's device will go on to make yet more history by being the first IoT medical device to **suffer a major security breach in 2016** (without casualties, fortunately).
3) **Bitcoin** starts operation, a precursor to **blockchain** technologies that are likely to be a big part of IoT.

The evolution of the Internet of Things

Google Glass is released, a revolutionary step in IoT and wearable technology but possibly ahead of its time. It flops pretty hard. VR and AR are still in early stages

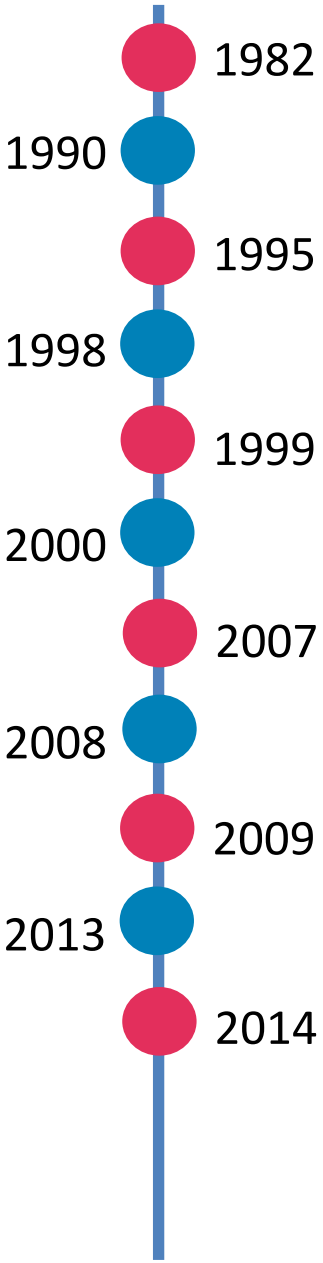


The evolution of the Internet of Things



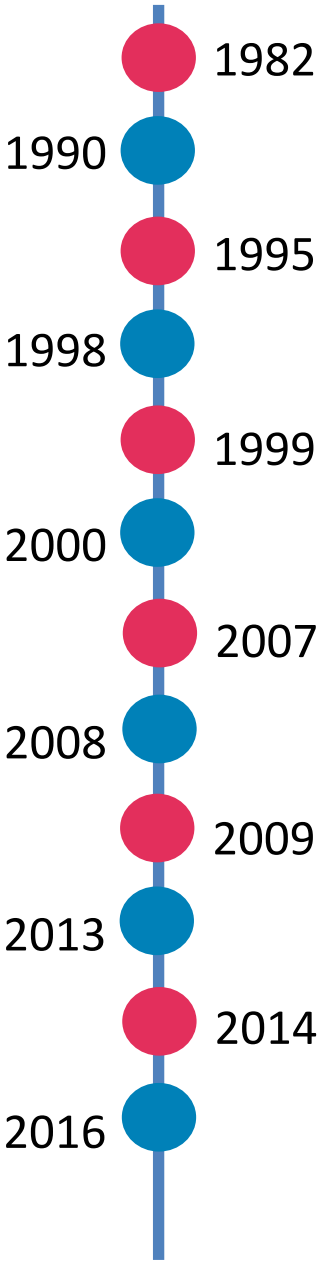
1) An **Industrial IoT** standards consortium form demonstrates the potential for IoT to **change the way any number of manufacturing and supply chain processes work**.

The evolution of the Internet of Things



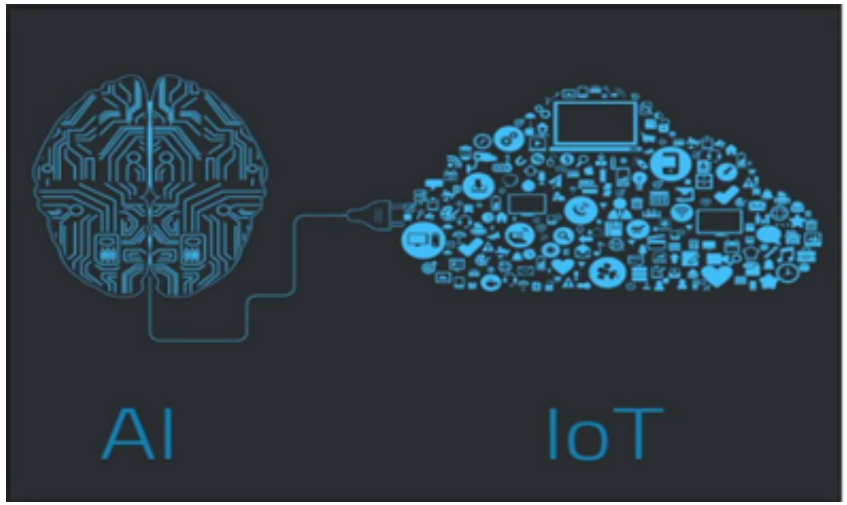
1) An **Industrial IoT** standards consortium form demonstrates the potential for IoT to **change the way any number of manufacturing and supply chain processes work**.
2) **Amazon** releases the **Echo**, paving the way for a rush into the smart home hub market.

The evolution of the Internet of Things



General Motors, Lyft, Tesla, and Uber are all testing self-driving cars. The **first massive IoT malware attack is also confirmed**, with the Mirai botnet assaulting IoT devices with manufacturer-default logins, taking them over, and using them to DDoS popular websites.

The evolution of the Internet of Things



1982

1990

1995

1998

1999

2000

2007

2008

2009

2013

2014

2016

2017-2019

IoT development gets **cheaper, easier, and more broadly-accepted**, leading to small waves of innovation all over the industry.

Self-driving cars continue to improve, **blockchains and AI begin to be integrated into IoT platforms**, and increased smartphone/broadband penetration continues to make IoT an attractive proposition for the future.

Packet switching

From Wikipedia, the free encyclopedia

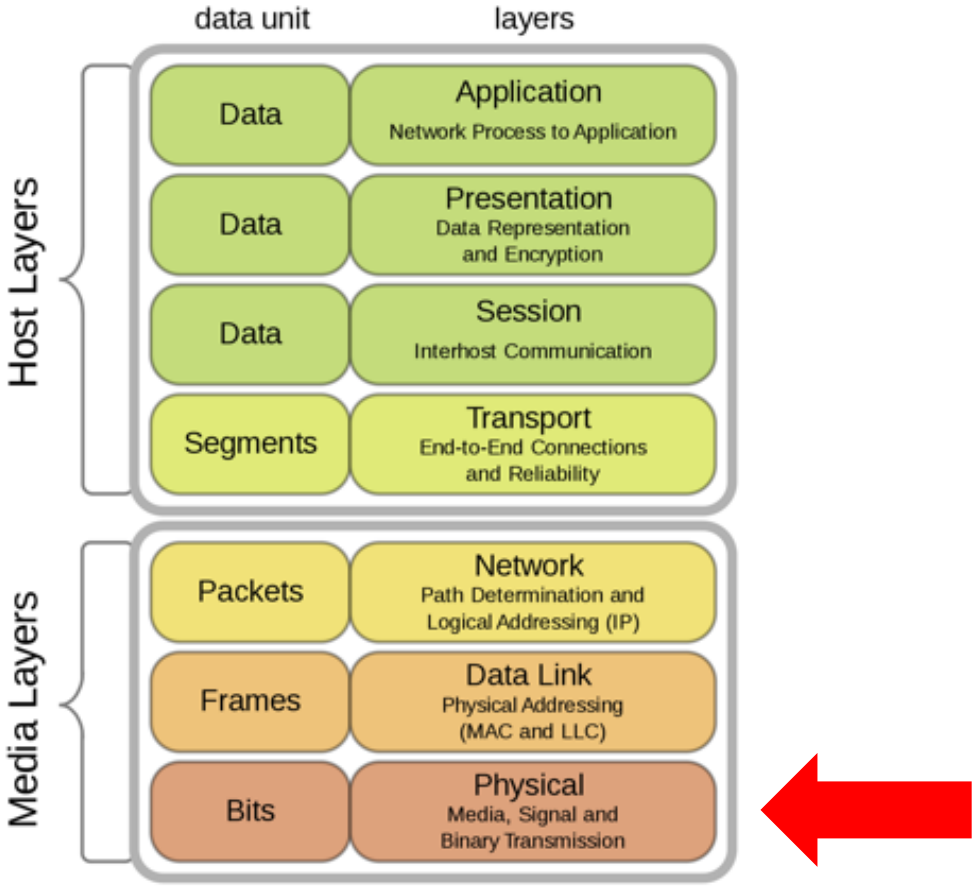
Packet switching is a method of grouping data that is transmitted over a digital network into *packets*. Packets are made of a header and a payload. Data in the header are used by networking hardware to direct the packet to its destination where the payload is extracted and used by application software. Packet switching is the primary basis for data communications in computer networks worldwide.

OSI model

From Wikipedia, the free encyclopedia

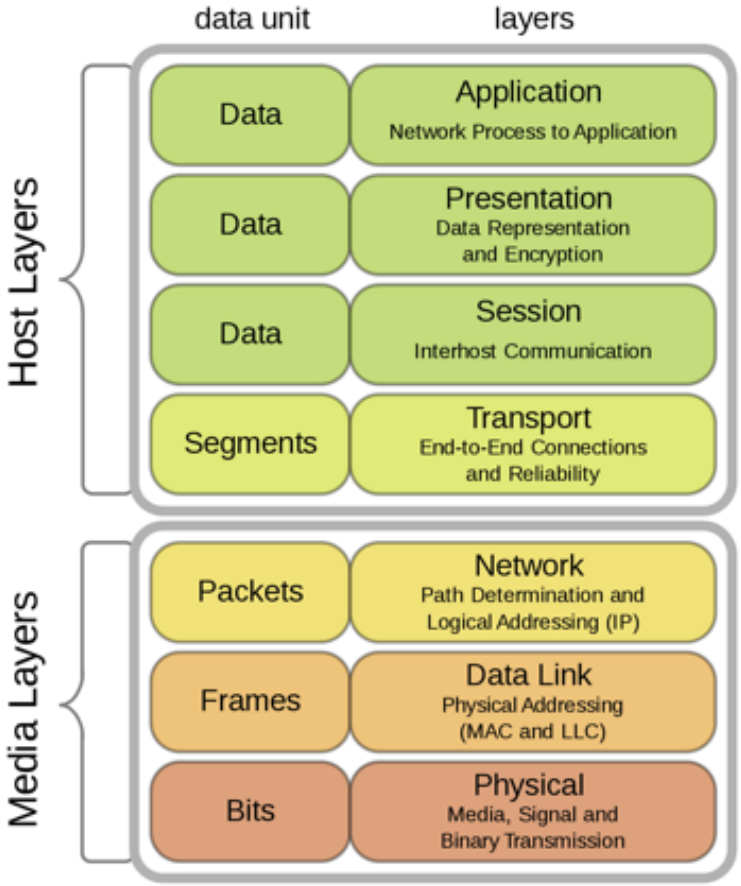
The **Open Systems Interconnection model** (**OSI model**) is a conceptual model that characterizes and standardizes the communication functions of a telecommunication or computing system without regard to its underlying internal structure and technology. Its goal is the interoperability of diverse communication systems with standard communication protocols.

The ISO\OSI Model



The **Physical Layer** is responsible for the transmission and reception of unstructured raw data between a device and a physical transmission medium. **It converts the digital bits into electrical, radio, or optical signals**.

The ISO\OSI Model



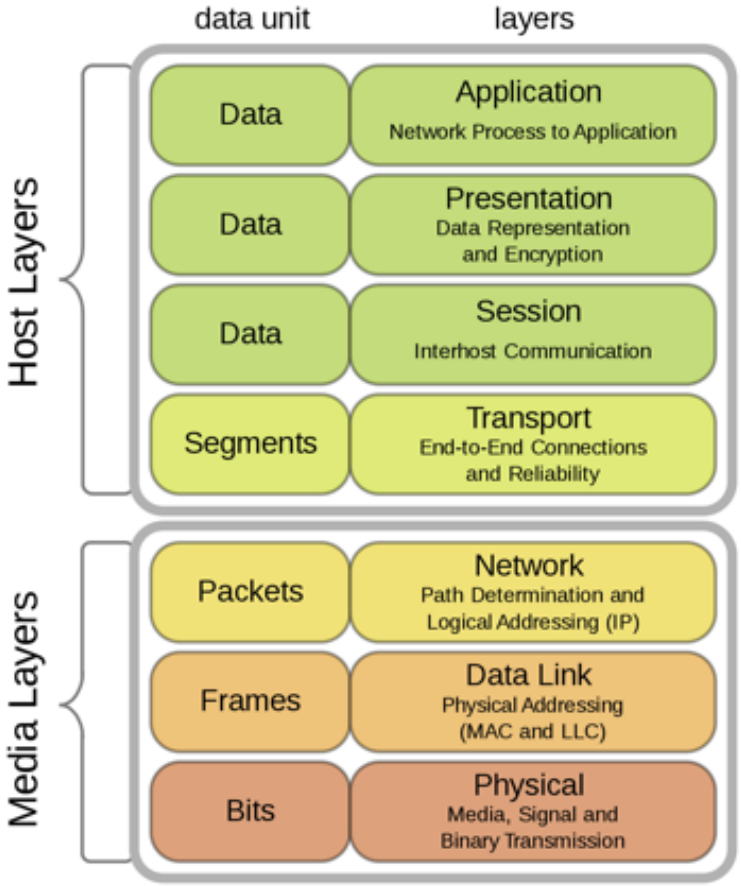
The **Data Link Layer** provides node-to-node data transfer, a link between two directly connected nodes.

It detects and possibly corrects errors that may occur in the physical layer.

It defines the protocol to establish and terminate a connection between two physically connected devices.

Protocols: MAC, LLC

The ISO\OSI Model



The **Network Layer** provides the functional and procedural means of transferring variable length data sequences (called **packets**) from one node to another connected in **"different networks"**.

A **network is a medium:**

- to which many **nodes** can be **connected**,
- on which every node has an **IP address**,
- which permits nodes to **transfer messages** each others

The **network find the way to deliver the message** to the destination node, possibly routing it through intermediate nodes

Protocols: IPv4, IPv6

Internet Protocol (IP)

From Wikipedia, the free encyclopedia

The **Internet Protocol** (**IP**) is the principal communications protocol in the Internet protocol suite for relaying datagrams across network boundaries. Its routing function enables internetworking, and essentially establishes the Internet.

IP has the task of delivering packets from the source host to the destination host solely based on the IP addresses in the packet headers. For this purpose, IP defines packet structures that encapsulate the data to be delivered. It also defines addressing methods that are used to label the datagram with source and destination information.

The first major version of IP, Internet Protocol Version 4 (IPv4), is the dominant protocol of the Internet. Its successor is Internet Protocol Version 6 (IPv6), which has been in increasing deployment on the public Internet since c. 2006.

IP address

From Wikipedia, the free encyclopedia

An **Internet Protocol address** (**IP address**) is a numerical label assigned to each device connected to a computer network that uses the Internet Protocol for communication.[1][2] An IP address serves two main functions: host or network interface identification and location addressing.

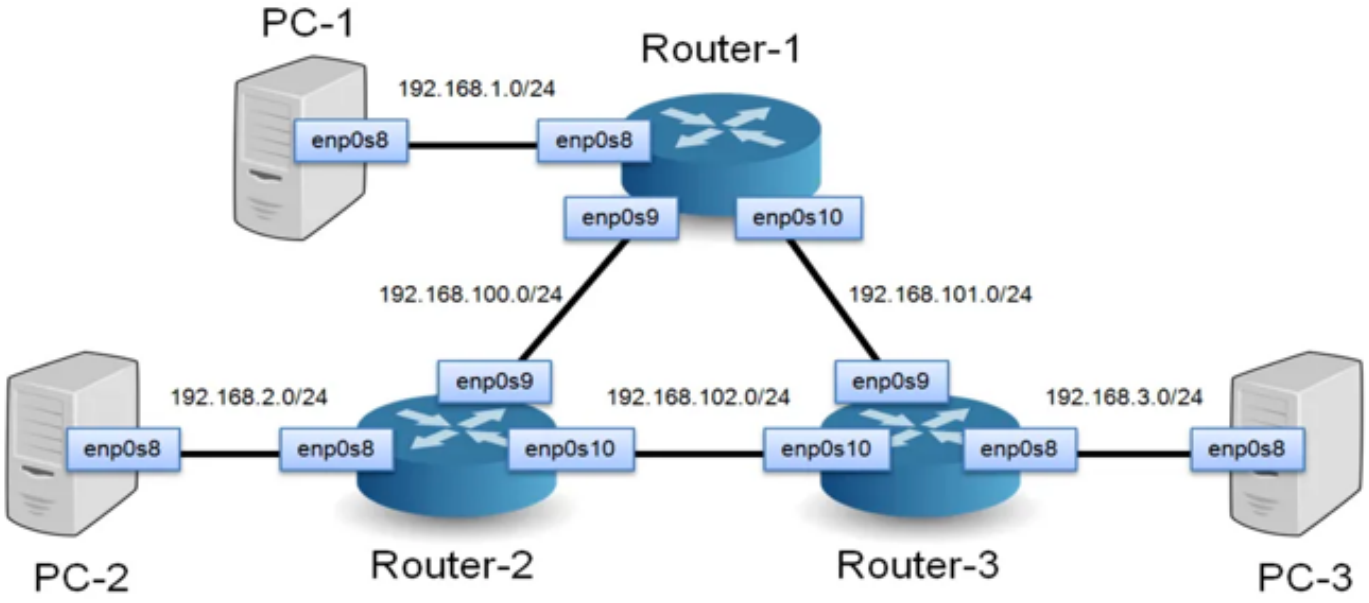
IP addresses are written and displayed in human-readable notations, such as *172.16.254.1* in IPv4, and *2001:db8:0:1234:0:567:8:1* in IPv6.

Packet Delivery

A **router** is a networking device that forwards data packets between computer networks. Routers perform the traffic directing functions on the Internet.

A router is connected to two or more data lines from different IP networks. When a data packet comes in on one of the lines, the router reads the network address information in the packet header to determine the ultimate destination.

Message delivery at the network layer is not necessarily guaranteed to be reliable

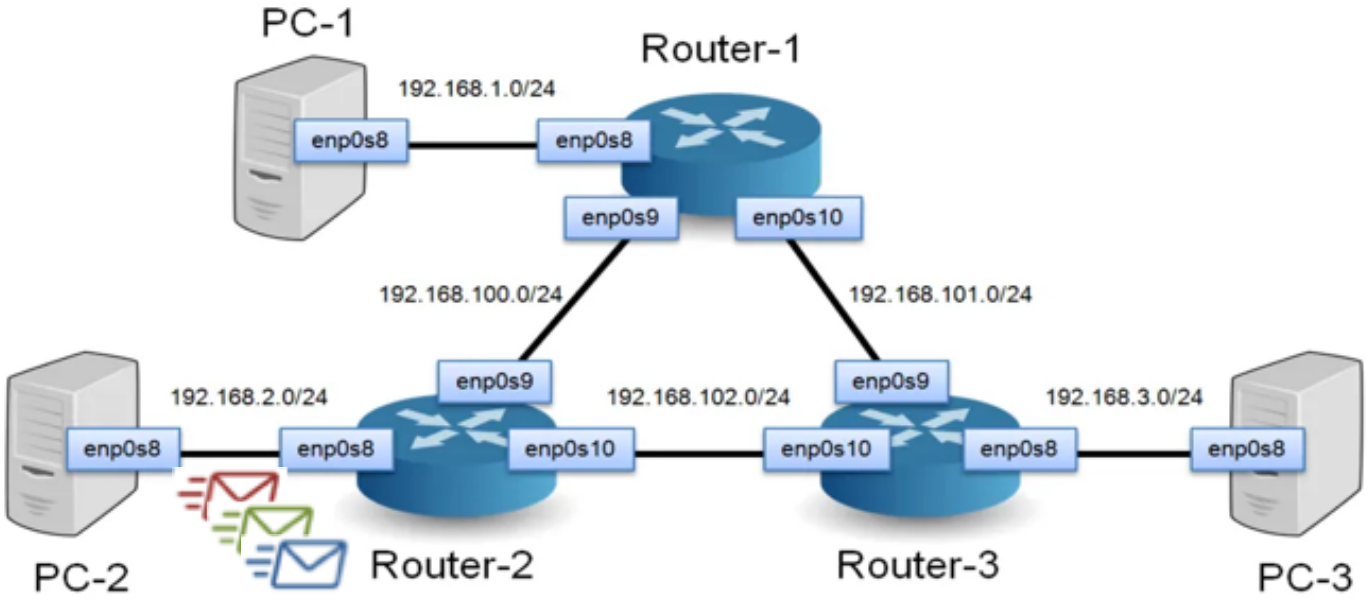


Packet Delivery

A **router** is a networking device that forwards data packets between computer networks. Routers perform the traffic directing functions on the Internet.

A router is connected to two or more data lines from different IP networks. When a data packet comes in on one of the lines, the router reads the network address information in the packet header to determine the ultimate destination.

Message delivery at the network layer is not necessarily guaranteed to be reliable

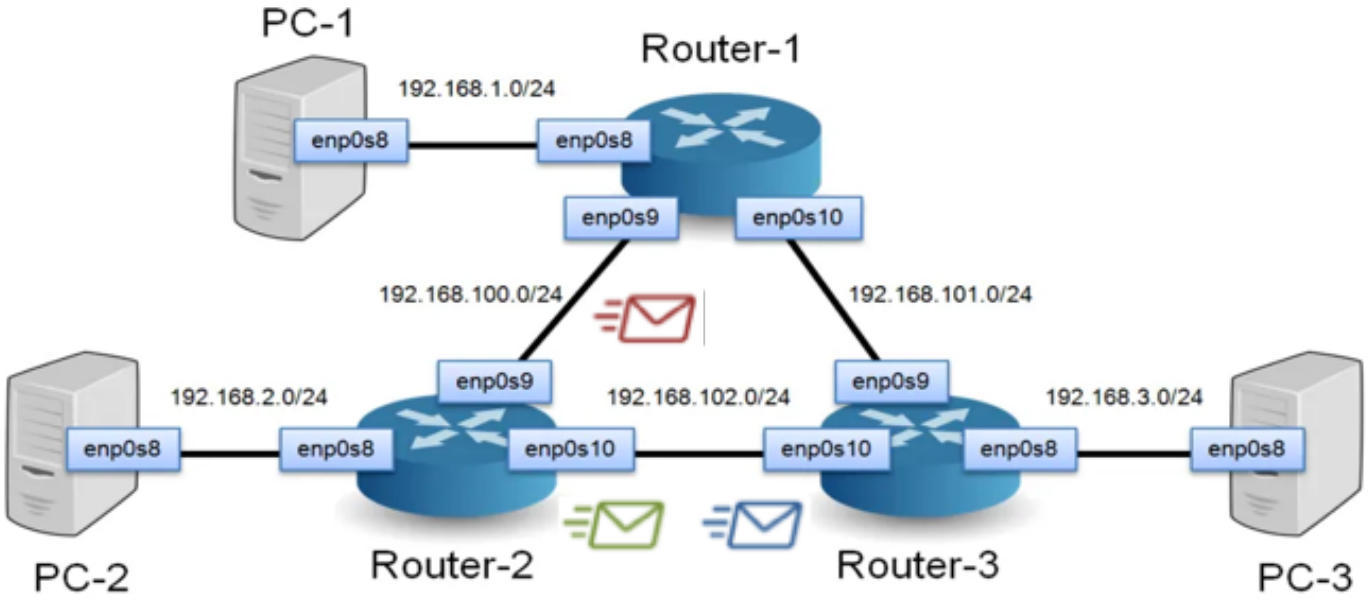


Packet Delivery

A **router** is a networking device that forwards data packets between computer networks. Routers perform the traffic directing functions on the Internet.

A router is connected to two or more data lines from different IP networks. When a data packet comes in on one of the lines, the router reads the network address information in the packet header to determine the ultimate destination.

Message delivery at the network layer is not necessarily guaranteed to be reliable

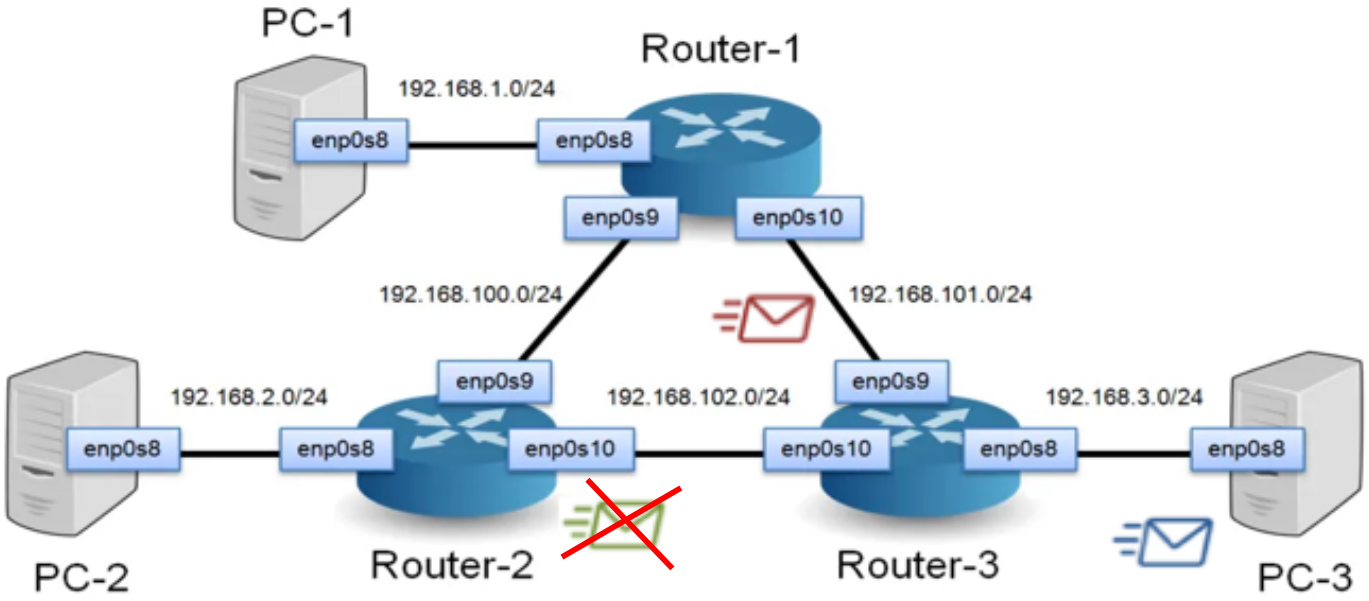


Packet Delivery

A **router** is a networking device that forwards data packets between computer networks. Routers perform the traffic directing functions on the Internet.

A router is connected to two or more data lines from different IP networks. When a data packet comes in on one of the lines, the router reads the network address information in the packet header to determine the ultimate destination.

Message delivery at the network layer is not necessarily guaranteed to be reliable

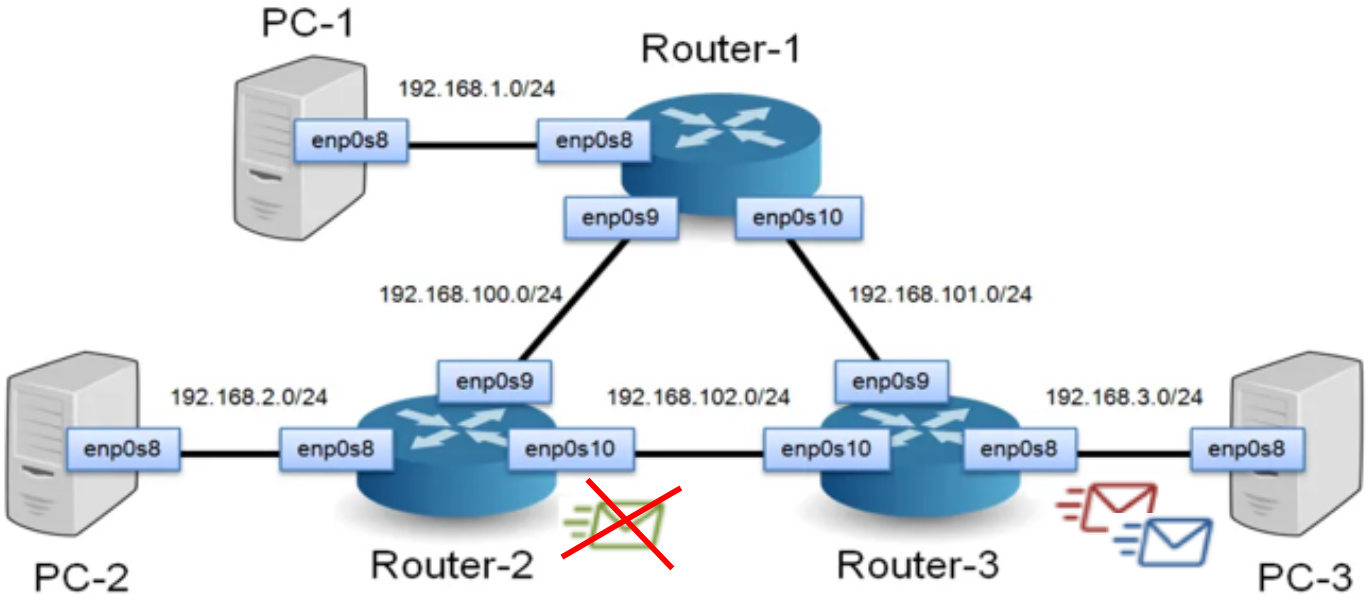


Packet Delivery

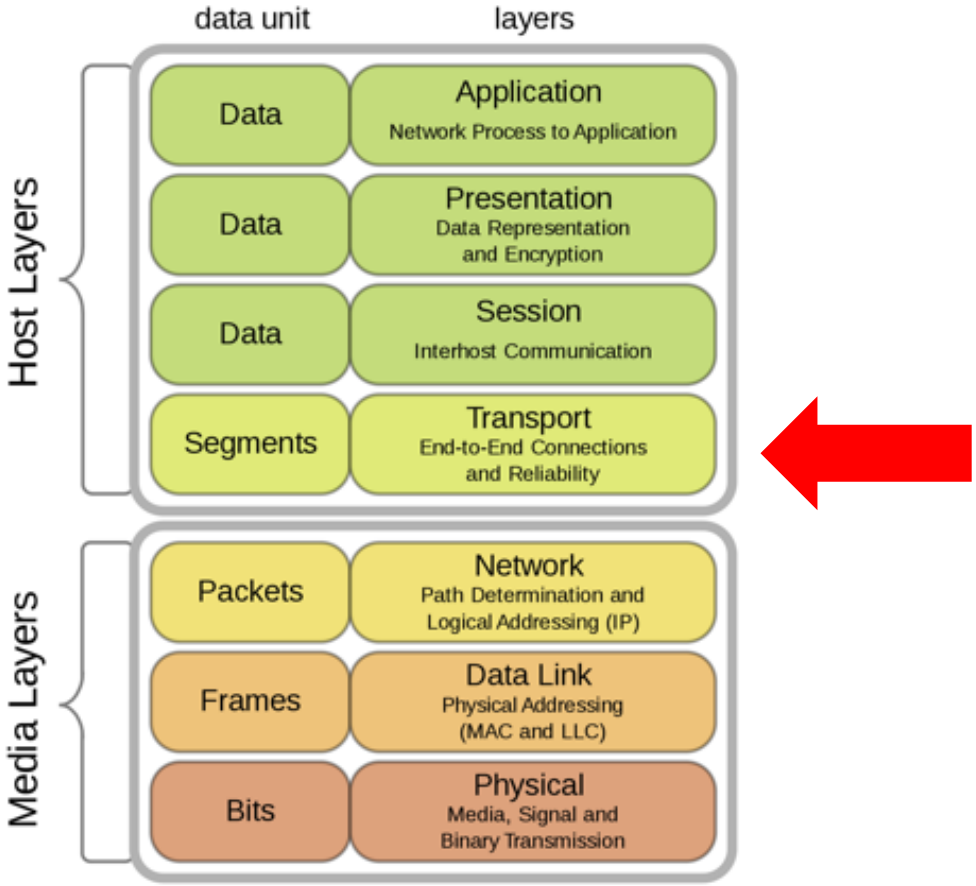
A **router** is a networking device that forwards data packets between computer networks. Routers perform the traffic directing functions on the Internet.

A router is connected to two or more data lines from different IP networks. When a data packet comes in on one of the lines, the router reads the network address information in the packet header to determine the ultimate destination.

Message delivery at the network layer is not necessarily guaranteed to be reliable



The ISO\OSI Model



The **Transport Layer** provides the functional and procedural means of transferring variable-length data sequences from a source to a destination host, while maintaining the quality of service functions.

This layer controls the reliability of a given link through flow control, segmentation/desegmentation, and error control. **Protocols are connectionless or connection-oriented**.

Protocols: UDP, TCP

User Datagram Protocol

From Wikipedia, the free encyclopedia

In computer networking, the **User Datagram Protocol** (**UDP**) is one of the core members of the Internet protocol suite. The protocol was designed by David P. Reed in 1980 and formally defined in RFC 768. With UDP, computer applications can send messages, in this case referred to as *datagrams*, to other hosts on an Internet Protocol (IP) network. Prior communications are not required in order to set up communication channels or data paths.

David P. Reed

User Datagram Protocol

From Wikipedia, the free encyclopedia

In computer networking, the **User Datagram Protocol** (**UDP**) is one of the core members of the Internet protocol suite. The protocol was designed by David P. Reed in 1980 and formally defined in RFC 768. With UDP, computer applications can send messages, in this case referred to as *datagrams*, to other hosts on an Internet Protocol (IP) network. Prior communications are not required in order to set up communication channels or data paths.

UDP uses a simple connectionless communication model with a minimum of protocol mechanisms. UDP provides checksums for data integrity, and port numbers for addressing different functions at the source and destination of the datagram. It has no handshaking dialogues, and thus exposes the user's program to any unreliability of the underlying network; there is no guarantee of delivery, ordering, or duplicate protection.

User Datagram Protocol

From Wikipedia, the free encyclopedia

In computer networking, the **User Datagram Protocol** (**UDP**) is one of the core members of the Internet protocol suite. The protocol was designed by David P. Reed in 1980 and formally defined in RFC 768. With UDP, computer applications can send messages, in this case referred to as *datagrams*, to other hosts on an Internet Protocol (IP) network. Prior communications are not required in order to set up communication channels or data paths.

UDP uses a simple connectionless communication model with a minimum of protocol mechanisms. UDP provides checksums for data integrity, and port numbers for addressing different functions at the source and destination of the datagram. It has no handshaking dialogues, and thus exposes the user's program to any unreliability of the underlying network; there is no guarantee of delivery, ordering, or duplicate protection.

Time-sensitive applications often use UDP because dropping packets is preferable to waiting for packets delayed due to retransmission, which may not be an option in a real-time system.

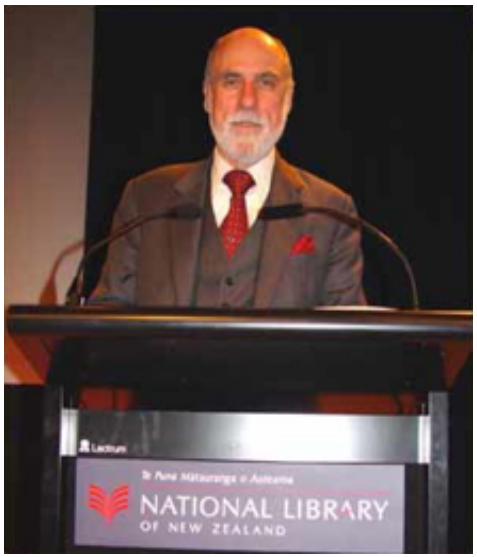
Transmission Control Protocol

From Wikipedia, the free encyclopedia

The **Transmission Control Protocol** (**TCP**) is one of the main protocols of the Internet protocol suite. It originated in the initial network implementation in which it complemented the Internet Protocol (IP). Therefore, the entire suite is commonly referred to as *TCP/IP*. TCP provides reliable, ordered, and error-checked delivery of a stream of octets (bytes) between applications running on hosts communicating via an IP network. Major internet applications such as the World Wide Web, email, remote administration, and file transfer rely on TCP.

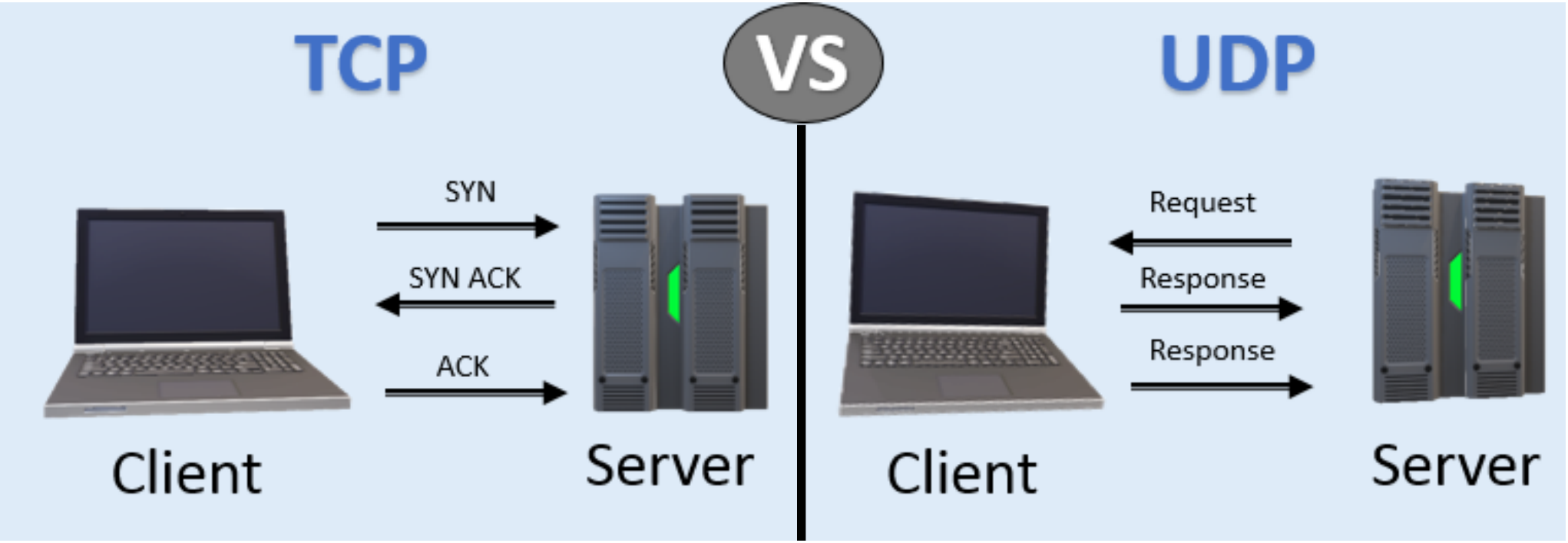
TPC is a connection-oriented protocol that establishes a connection between the two nodes before sending messages (a.k.a. **3-way handshake**).

Robert Kahn



Vinto Cerf

TCP vs UDP



- **3-way Handshake** to establish a communication channel
- For each received packet, the receiver sends an **acknowledgment** (ACK) to the sender
- Each **lost packet is resent**

- **No communication channel establishment**
- **Lost packets are NOT resent**

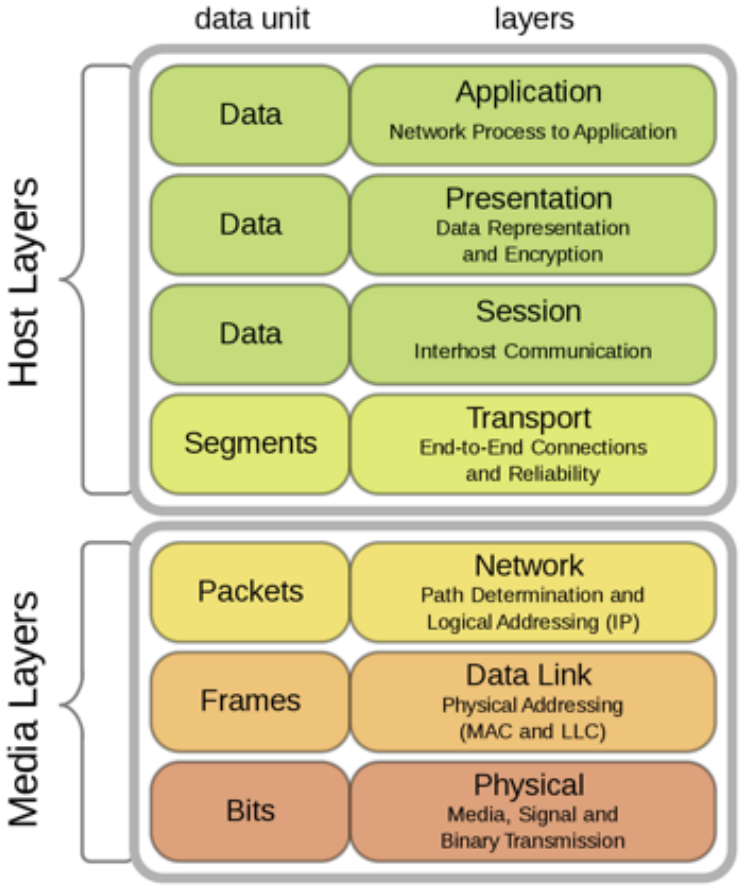
TCP vs UDP

TCP **vs** **UDP**

TCP vs UDP

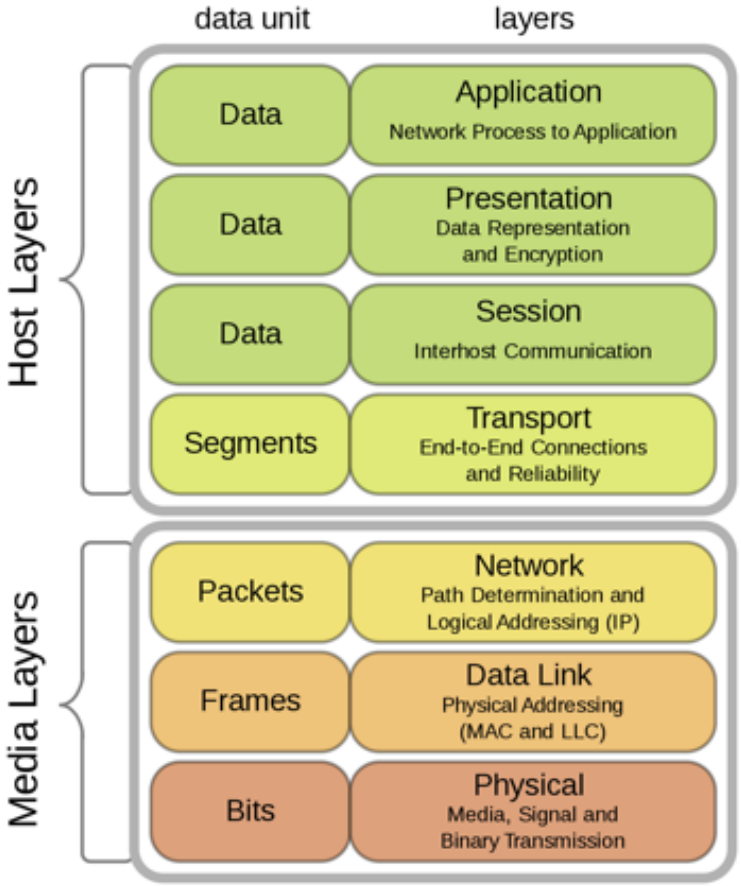
Item	TCP	UDP
Stands For	Transmission Control Protocol	User Datagram Protocol
Protocol	Connection Oriented	Connectionless
Security	Makes Checks For Errors And Reporting	Makes Error Checking But No Reporting
Data Sending	Slower	Faster
Header Size	20 Bytes	8 Bytes
Segments	Acknowledgement	No Acknowledgement
Typical Applications	- Email	- VoIP

The ISO\OSI Model



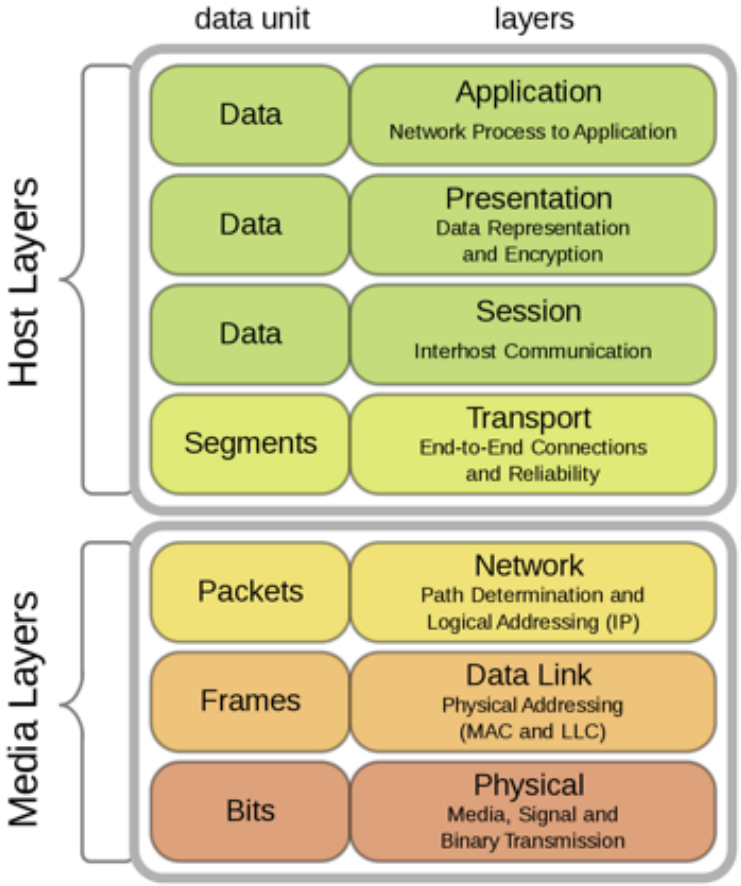
The **Session Layer** controls the dialogues (connections) between computers. It establishes, manages and terminates the connections between the local and remote application.

The ISO\OSI Model



The **Presentation Layer** provides independence from data representation by translating between application and network formats. The presentation layer transforms data into the form that the application accepts.

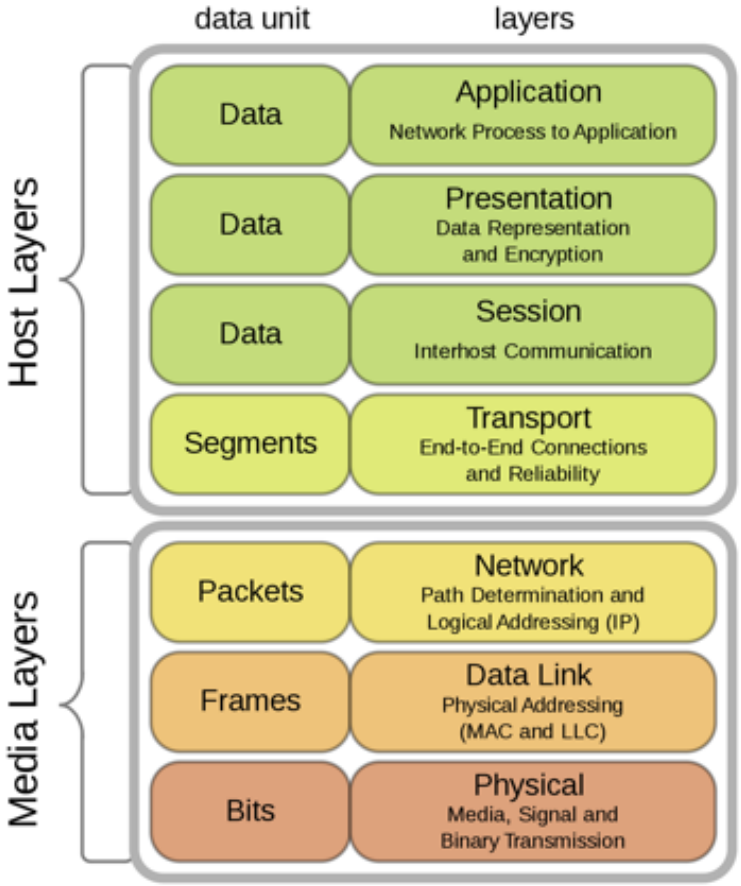
The ISO\OSI Model



The **Application Layer** is the layer closest to the end users. It interacts with software applications that implement a communicating component.
Application-layer functions typically include **identifying communication partners, determining resource availability, and synchronizing communication**

Protocols: HTTP, MQTT, etc.

The ISO\OSI Model

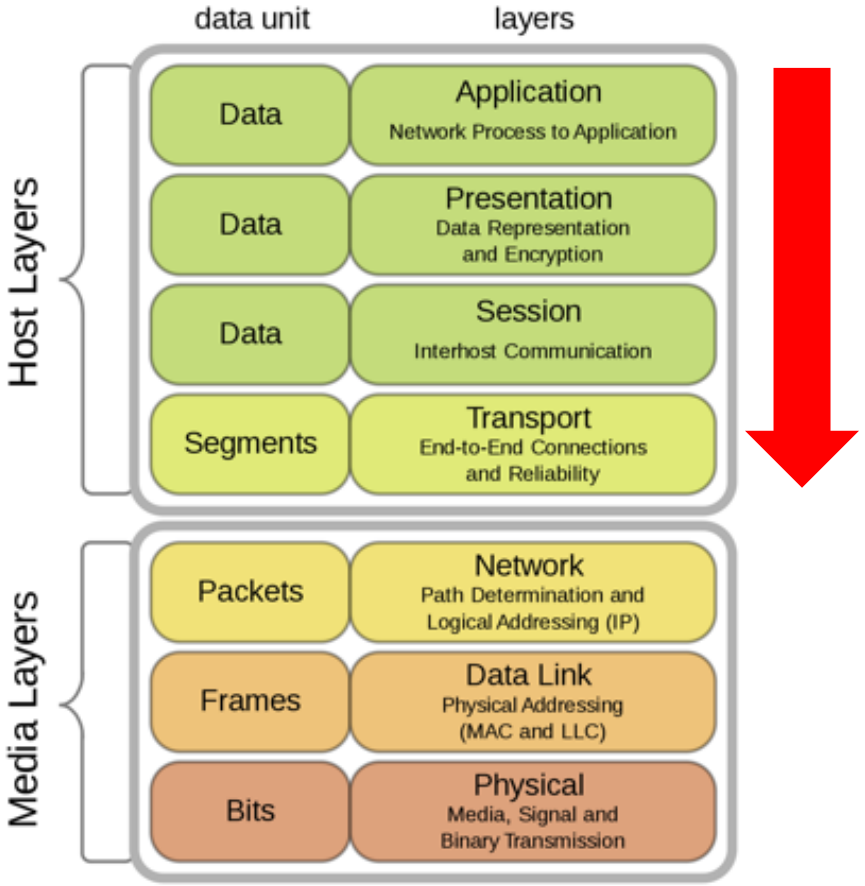


Each protocol at each layer in the stack appends additional information to the original message, i.e. Header (H) and footer

H_A Message

GET
http://myurl:8080/my/resources

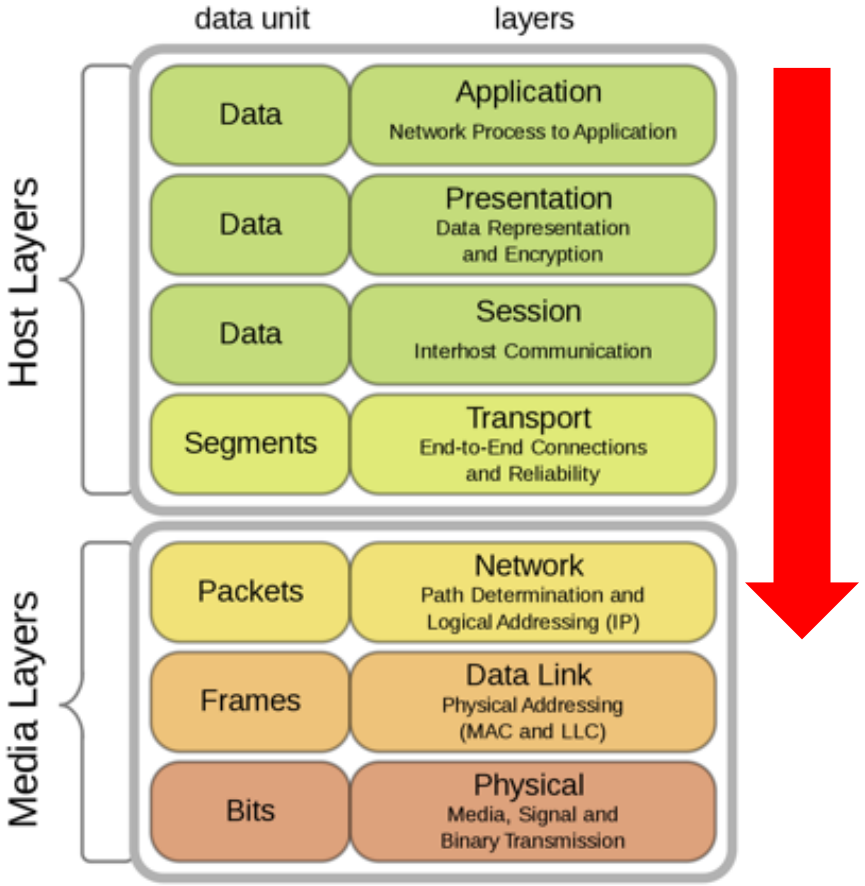
The ISO\OSI Model



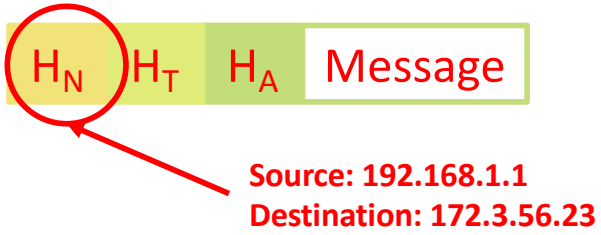
Each protocol at each layer in the stack appends additional information to the original message, i.e. Header (H) and footer

H_T H_A Message

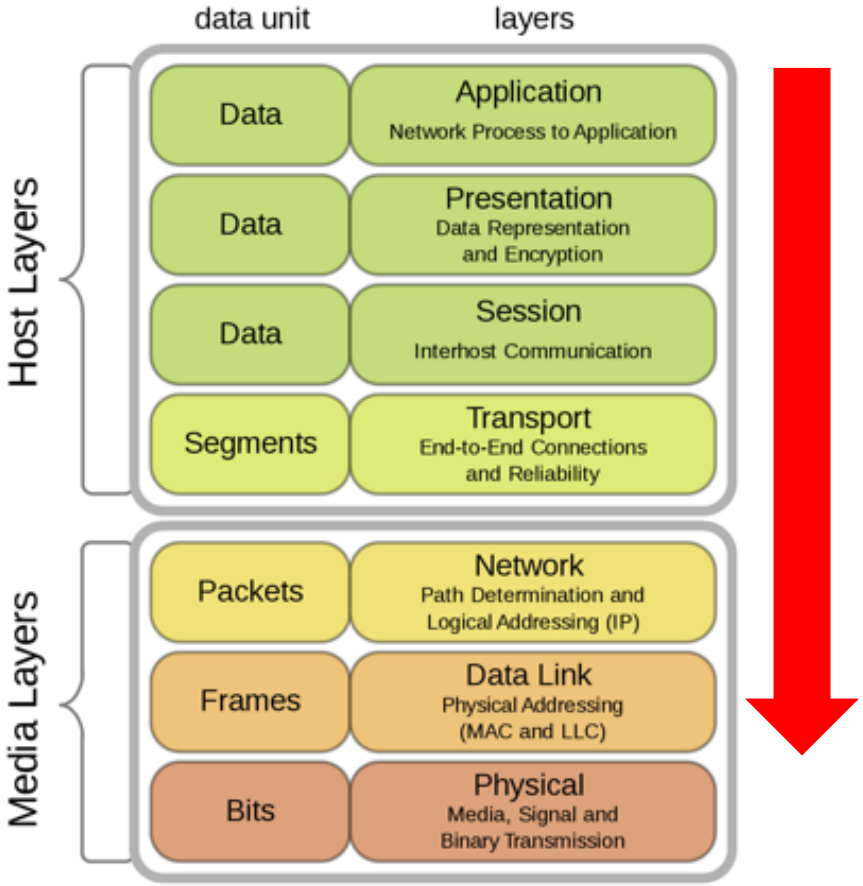
The ISO\OSI Model



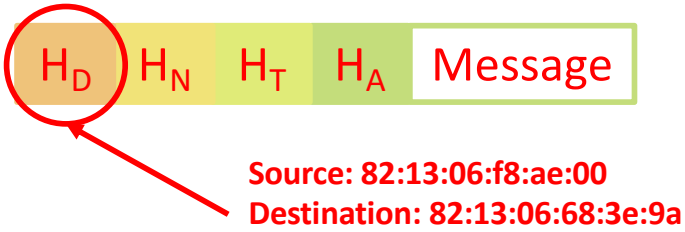
Each protocol at each layer in the stack appends additional information to the original message, i.e. Header (H) and footer



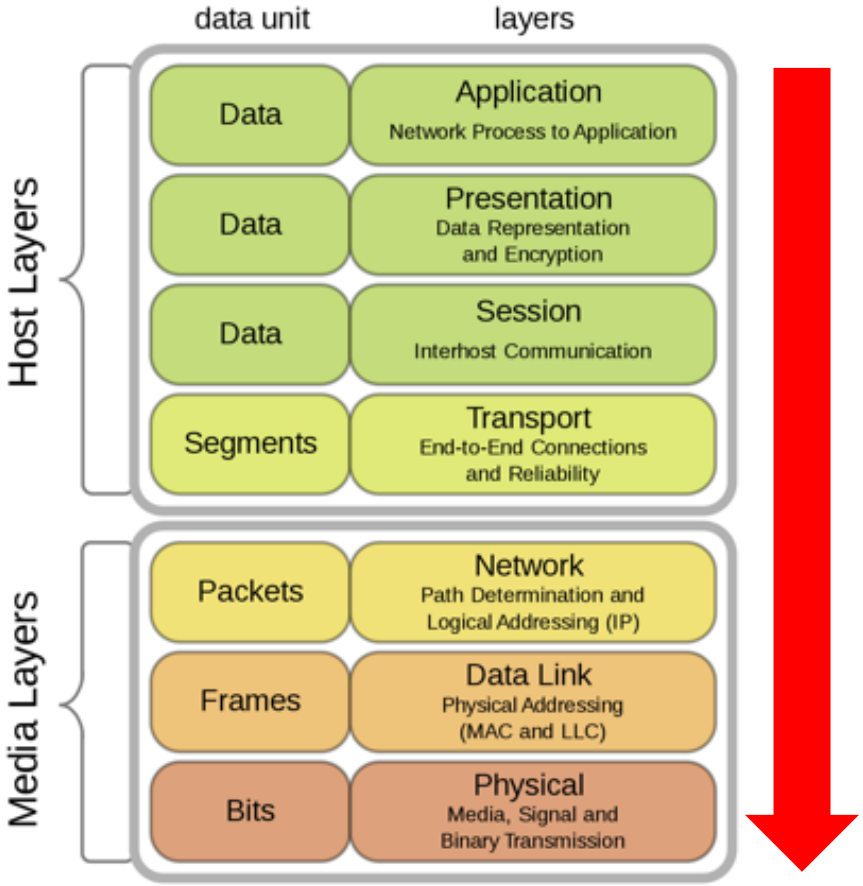
The ISO\OSI Model



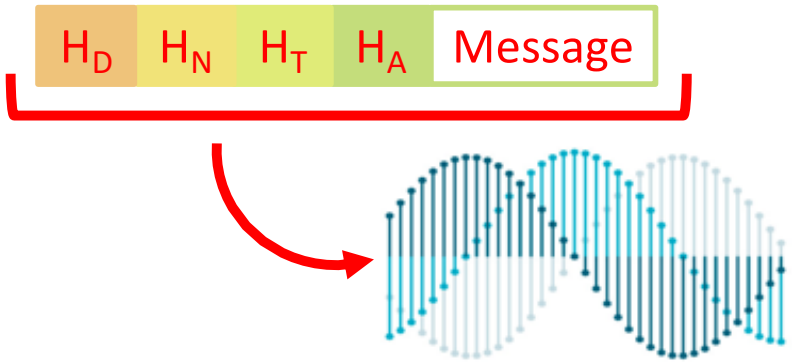
Each protocol at each layer in the stack appends additional information to the original message, i.e. Header (H) and footer



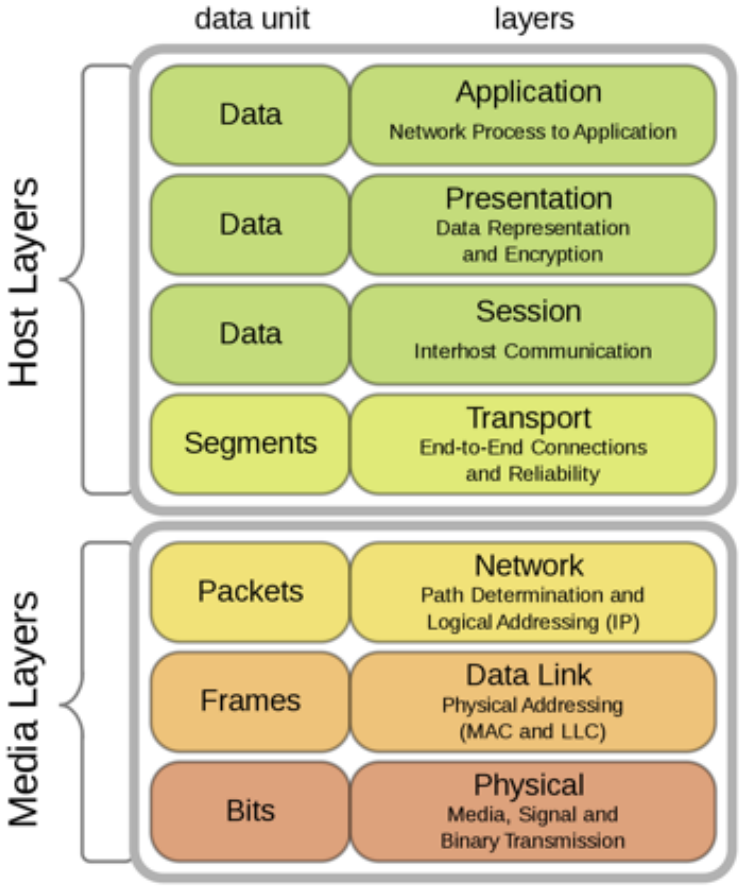
The ISO\OSI Model



Each protocol at each layer in the stack appends additional information to the original message, i.e. Header (H) and footer



The ISO\OSI Model

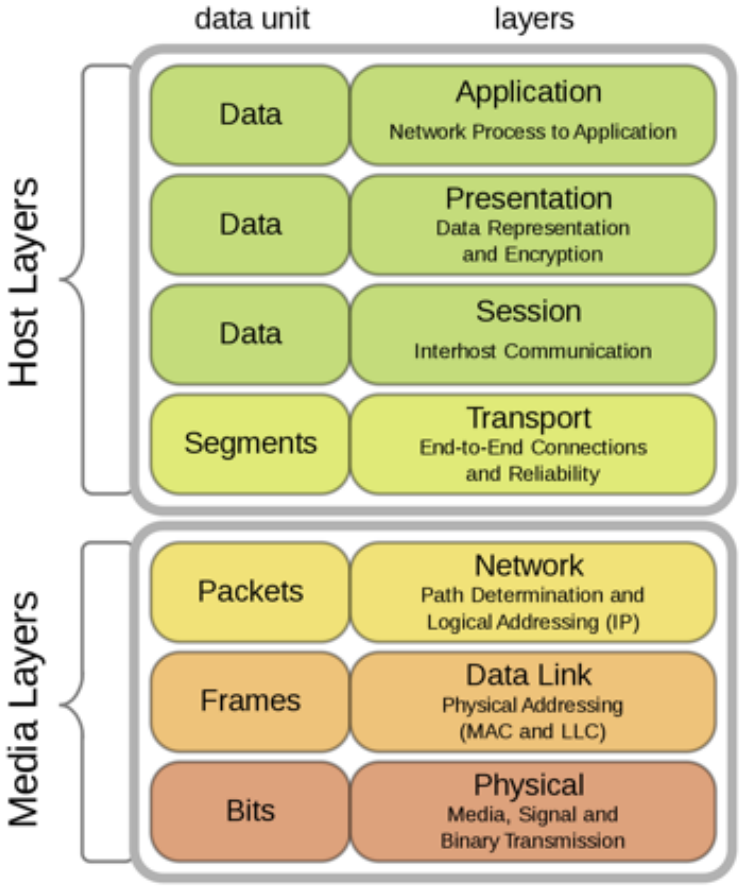


Most of the **protocols are not lightweight**, thus are not suitable for those IoT devices with limited computational resources (i.e. cpu and ram) that can be battery supplied.

New protocols have been proposed for these devices:

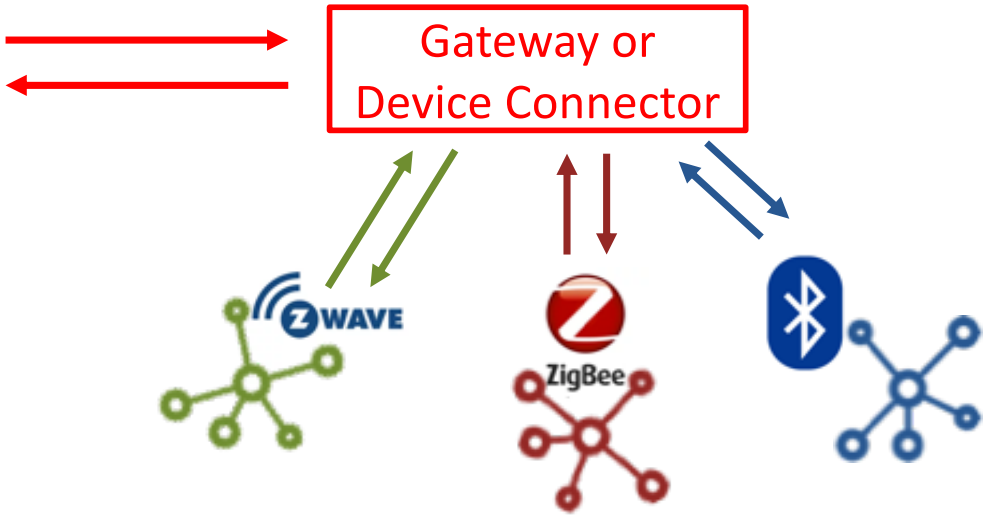
- Bluetooth
- ZigBee
- Z-wave
- 6lowpan

The ISO\OSI Model

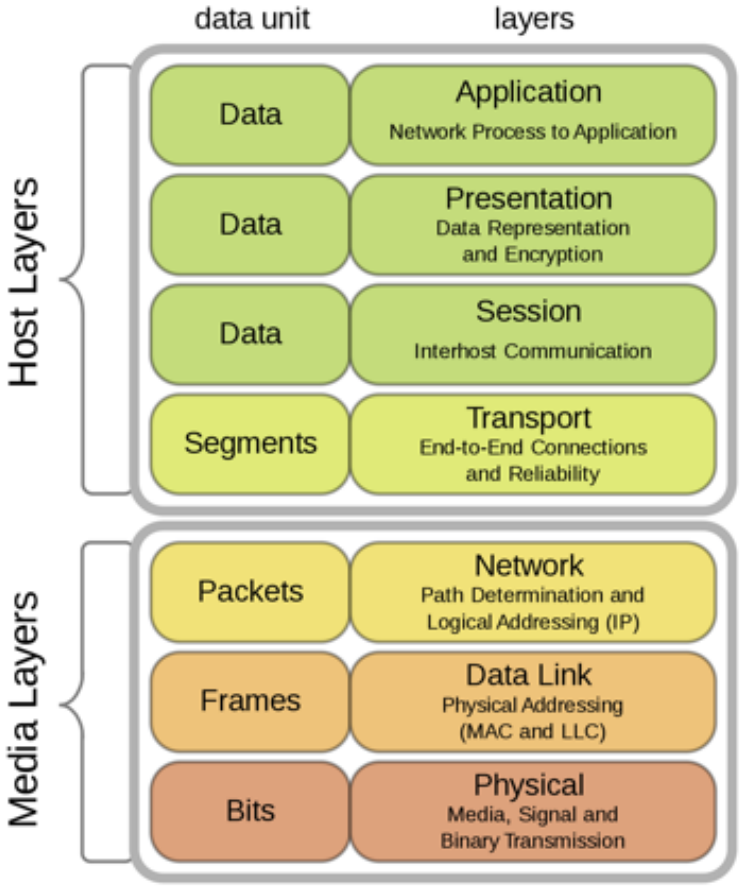


Hardware/software **components** are needed **to allow the interoperability** among them and **to provide Internet connection** acting as bridges.

Such components have different names, e.g. **Gateways** or **Device Connectors**

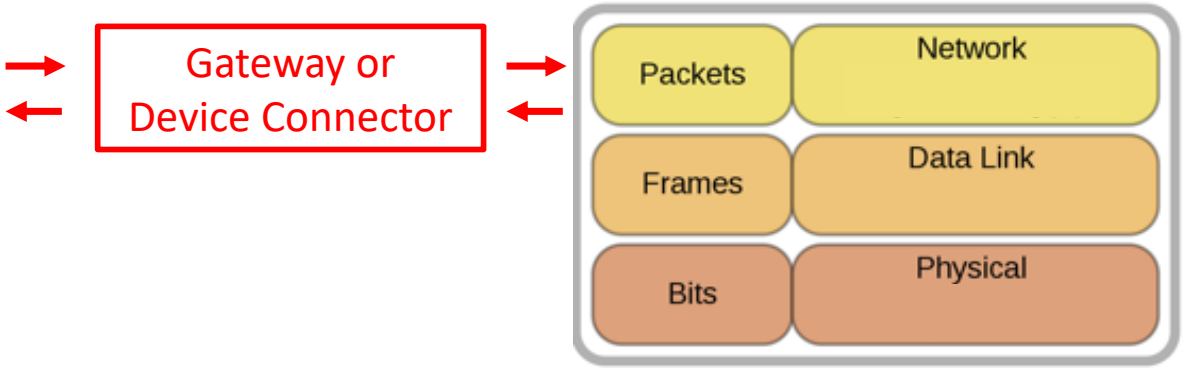


The ISO\OSI Model

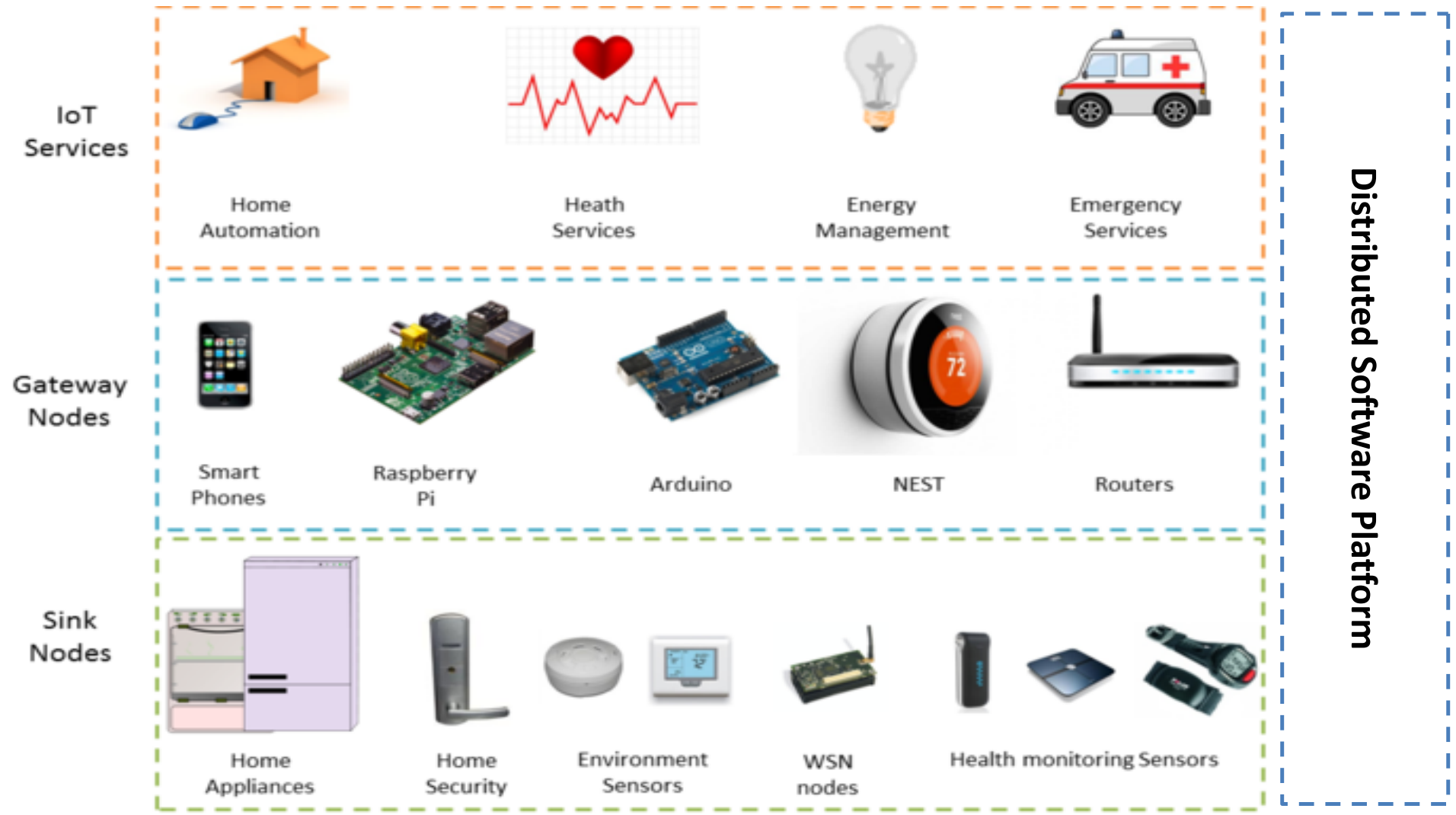


Hardware/software **components** are needed **to allow the interoperability** among them and **to provide Internet connection** acting as bridges.

Such components have different names, e.g. **Gateways** or **Device Connectors**

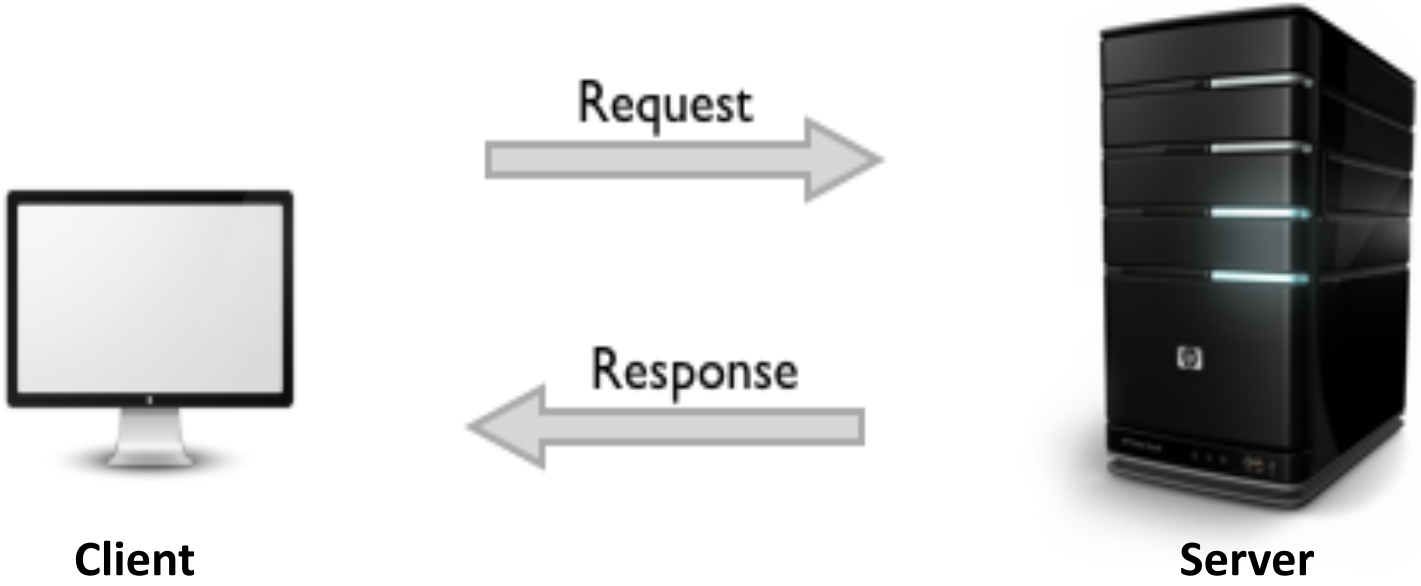


IoT Systems


IoT Services
Home Automation
Heath Services
Energy Management
Emergency Services
Gateway Nodes
Smart Phones
Raspberry Pi
Arduino
NEST
Routers
Sink Nodes
Home Appliances
Home Security
Environment Sensors
WSN nodes
Health monitoring Sensors
Distributed Software Platform

Communication paradigms

Request/Response is a **synchronous** communication paradigm. The client requests for some data and the server responds to the request.



Communication paradigms

Publish/subscribe is an **asynchronous** communication paradigm. It allows the development of loosely-coupled event-driven architectures. It removes the dependencies between producer and consumer of information.

The communication is based on Topics

(i.e. a label to identify a communication

flow)

Communication paradigms

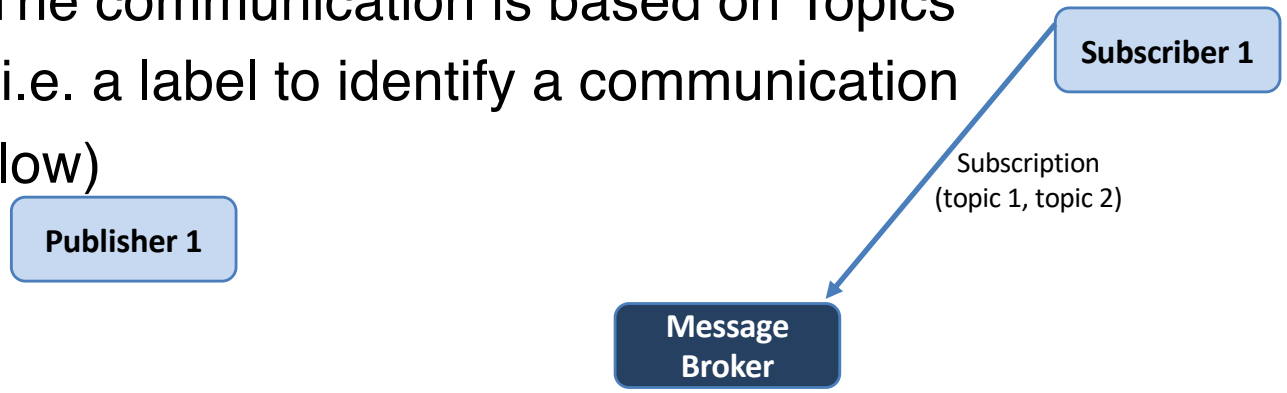
Publish/subscribe is an **asynchronous** communication paradigm. It allows the development of loosely-coupled event-driven architectures. It removes the dependencies between producer and consumer of information.

The communication is based on Topics (i.e. a label to identify a communication flow)

Communication paradigms

Publish/subscribe is an **asynchronous** communication paradigm. It allows the development of loosely-coupled event-driven architectures. It removes the dependencies between producer and consumer of information.

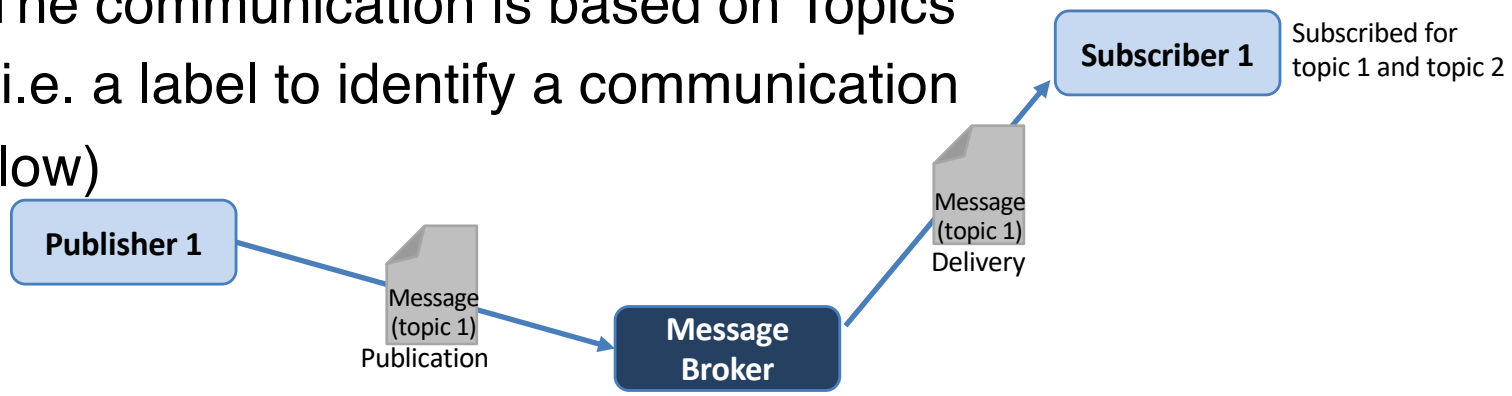
The communication is based on Topics (i.e. a label to identify a communication flow)



Communication paradigms

Publish/subscribe is an **asynchronous** communication paradigm. It allows the development of loosely-coupled event-driven architectures. It removes the dependencies between producer and consumer of information.

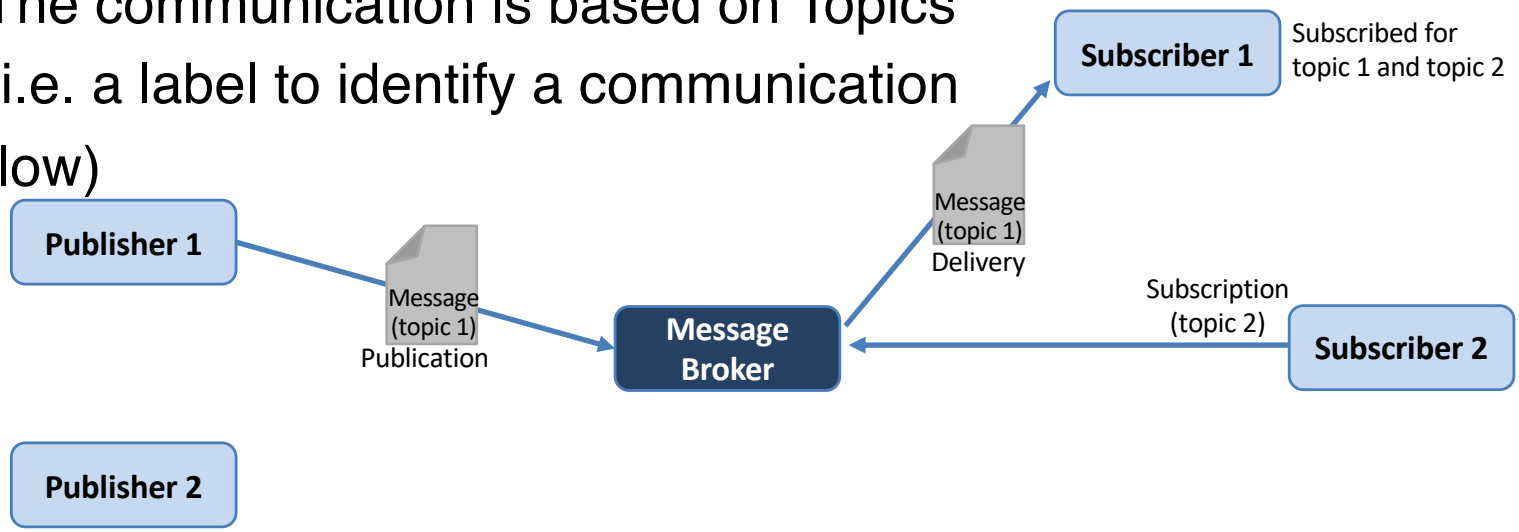
The communication is based on Topics (i.e. a label to identify a communication flow)



Communication paradigms

Publish/subscribe is an **asynchronous** communication paradigm. It allows the development of loosely-coupled event-driven architectures. It removes the dependencies between producer and consumer of information.

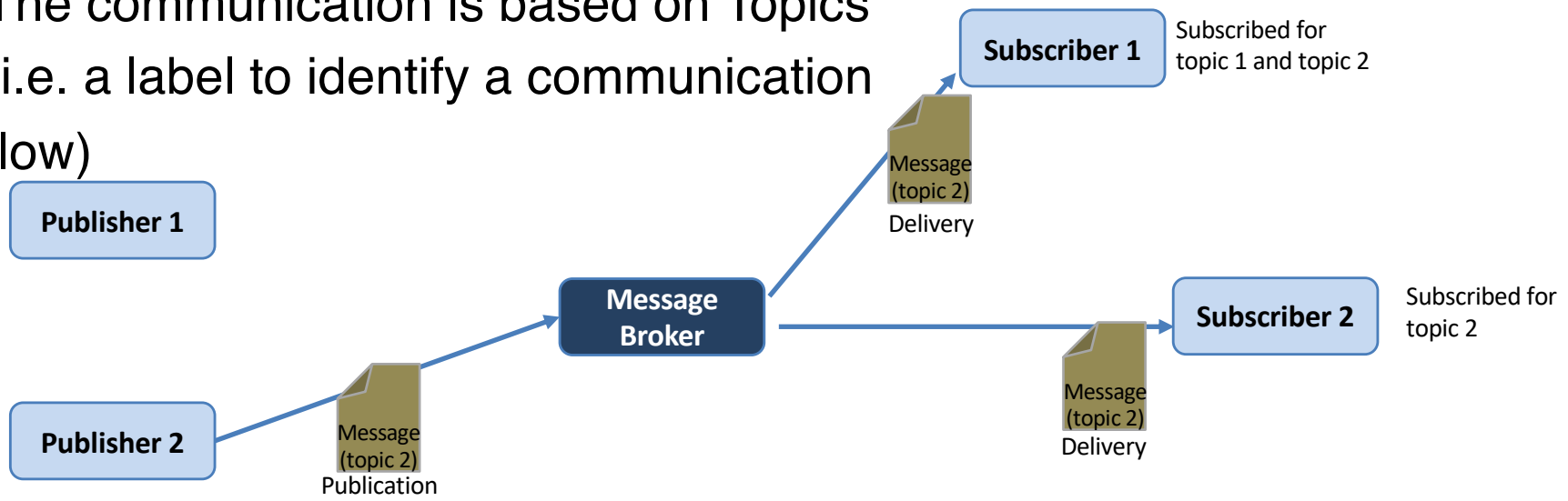
The communication is based on Topics (i.e. a label to identify a communication flow)



Communication paradigms

Publish/subscribe is an **asynchronous** communication paradigm. It allows the development of loosely-coupled event-driven architectures. It removes the dependencies between producer and consumer of information.

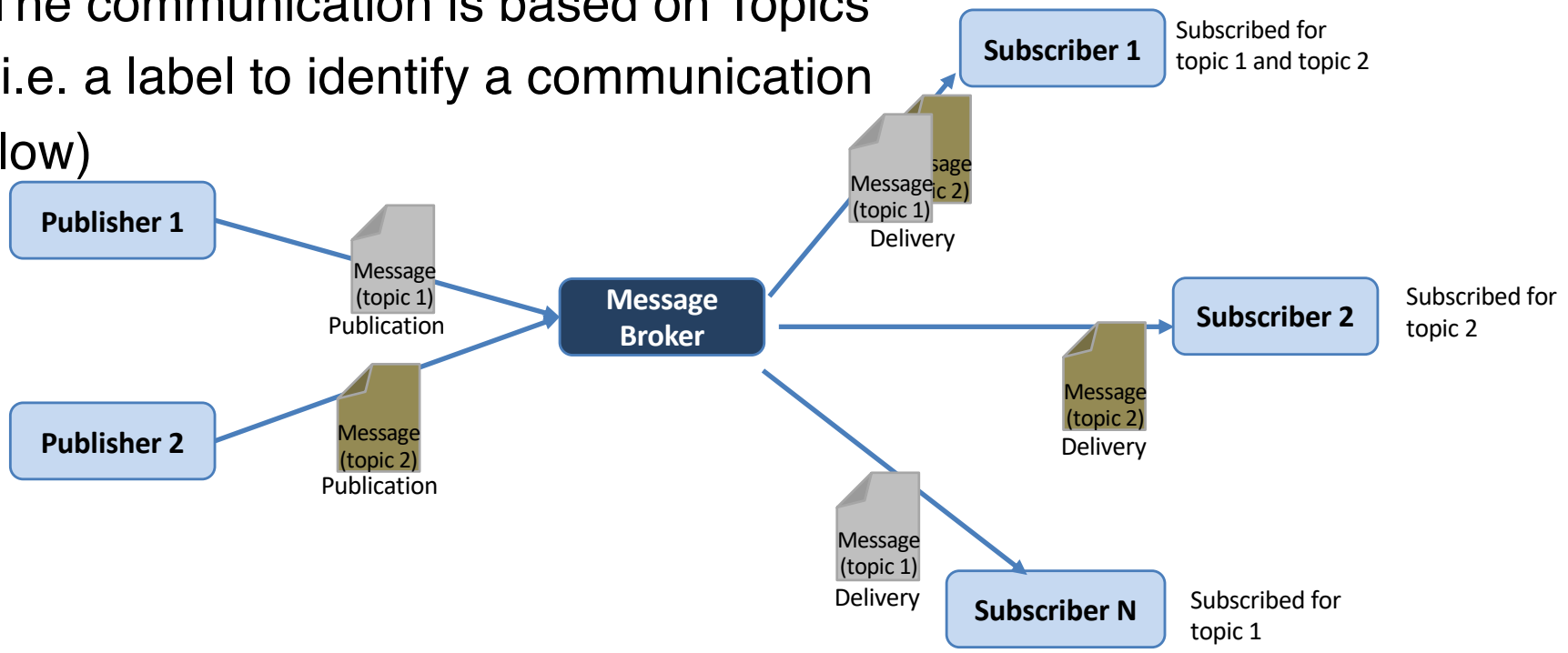
The communication is based on Topics (i.e. a label to identify a communication flow)



Communication paradigms

Publish/subscribe is an **asynchronous** communication paradigm. It allows the development of loosely-coupled event-driven architectures. It removes the dependencies between producer and consumer of information.

The communication is based on Topics (i.e. a label to identify a communication flow)



Messaging protocols for IoT systems

These two communication paradigms are implemented in many **Machine-to-Machine** (M2M) protocols suitable for IoT systems, addressing different requirements (e.g. data collection in constrained network, near-real-time data transmission, etc.)

Messaging protocols for IoT systems

These two communication paradigms are implemented in many **Machine-to-Machine** (M2M) protocols suitable for IoT systems, addressing different requirements (e.g. data collection in constrained network, near-real-time data transmission, etc.)

Those widely accepted and emerging are:

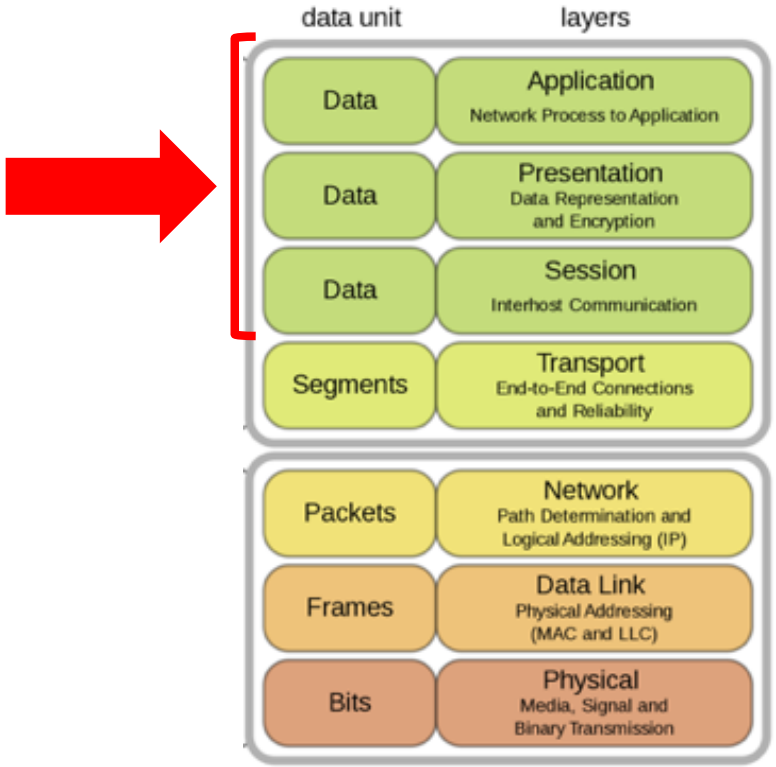
- **HTTP**
- **CoAP**
- **MQTT**
- **AMQP**

Messaging protocols for IoT systems

These two communication paradigms are implemented in many **Machine-to-Machine** (M2M) protocols suitable for IoT systems, addressing different requirements (e.g. data collection in constrained network, near-real-time data transmission, etc.)

Those widely accepted and emerging are:

- **HTTP**
- **CoAP**
- **MQTT**
- **AMQP**

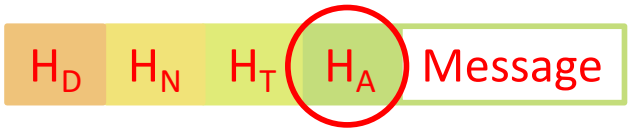


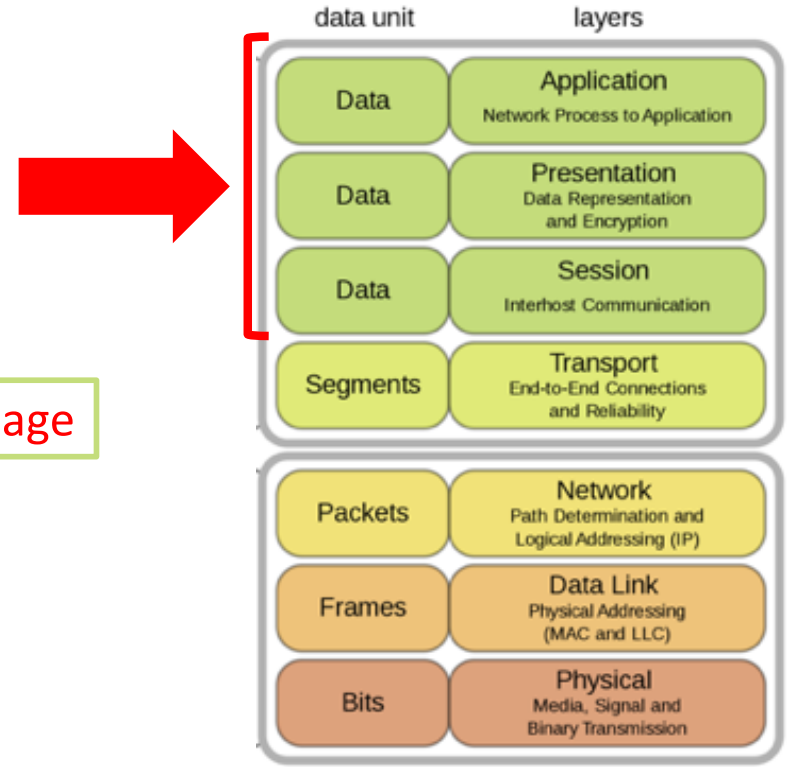
Messaging protocols for IoT systems

These two communication paradigms are implemented in many **Machine-to-Machine** (M2M) protocols suitable for IoT systems, addressing different requirements (e.g. data collection in constrained network, near-real-time data transmission, etc.)

Those widely accepted and emerging are:

- **HTTP**
- **CoAP**
- **MQTT**
- **AMQP**





HTTP - Hyper Text Transport Protocol

- HTTP is predominantly a **web messaging protocol** that **exploits** the **TCP** protocol as a default transport protocol and TLS/SSL for security.
- The communication between client and server is **connection-oriented**.
- HTTP uses Universal Resource Identifier (**URI**)
- HTTP supports **request/response RESTful** Web Services.
- HTTP is a **text-based protocol** and it does not define the size of header and message payloads

CoAP - Constrained Application Protocol

- **CoAP is a lightweight M2M protocol** that supports both request/response and a variant of publish/subscribe.
- CoAP is developed to **interoperate with HTTP** and RESTful Web Services through simple gateways (**request/response**).
- CoAP uses **URI** to request/send data (**request/response**).
- CoAP uses **URI instead of topics** (**publish/subscribe**).
  - Publisher publishes data to the URI and subscriber subscribes to a particular resource indicated by the URI. When a publisher publishes new data to the URI, then all the subscribers are notified about the new value as indicated by the URI

CoAP - Constrained Application Protocol

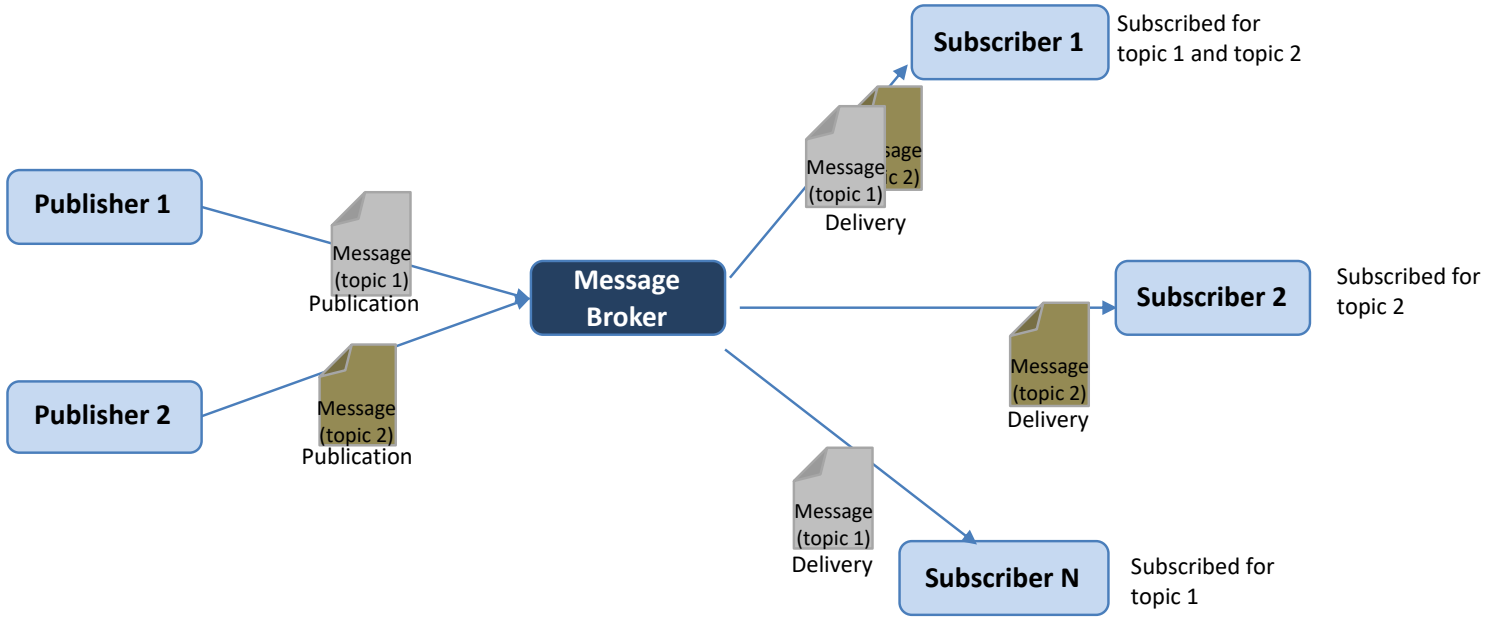
- CoAP is a binary protocol and requires fixed **header of 4-bytes with small message payloads** (depending on servers).
- CoAP **uses UDP** as a transport protocol and DTLS for security.
- The communication between client and server is **connectionless datagrams** (i.e. less reliability). CoAP messages can be **confirmable** or **non-confirmable**.
  - **Confirmable** messages must be **acknowledged** by the receiver with an ACK and non-confirmable messages are not.

MQTT - Message Queuing Telemetry Transport Protocol

- **MQTT** is one of the oldest lightweight M2M protocols (i.e. 1999) for **communications in constrained networks**.
- MQTT is a **very basic messaging protocol** offering only a few control options.

MQTT - Message Queuing Telemetry Transport Protocol

- MQTT client publishes messages to an MQTT broker, which are subscribed by other clients or may be retained for the future subscription. Every message is published to a topic. Clients can subscribe to multiple topics and receives every message published to the each topic.



MQTT - Message Queuing Telemetry Transport Protocol

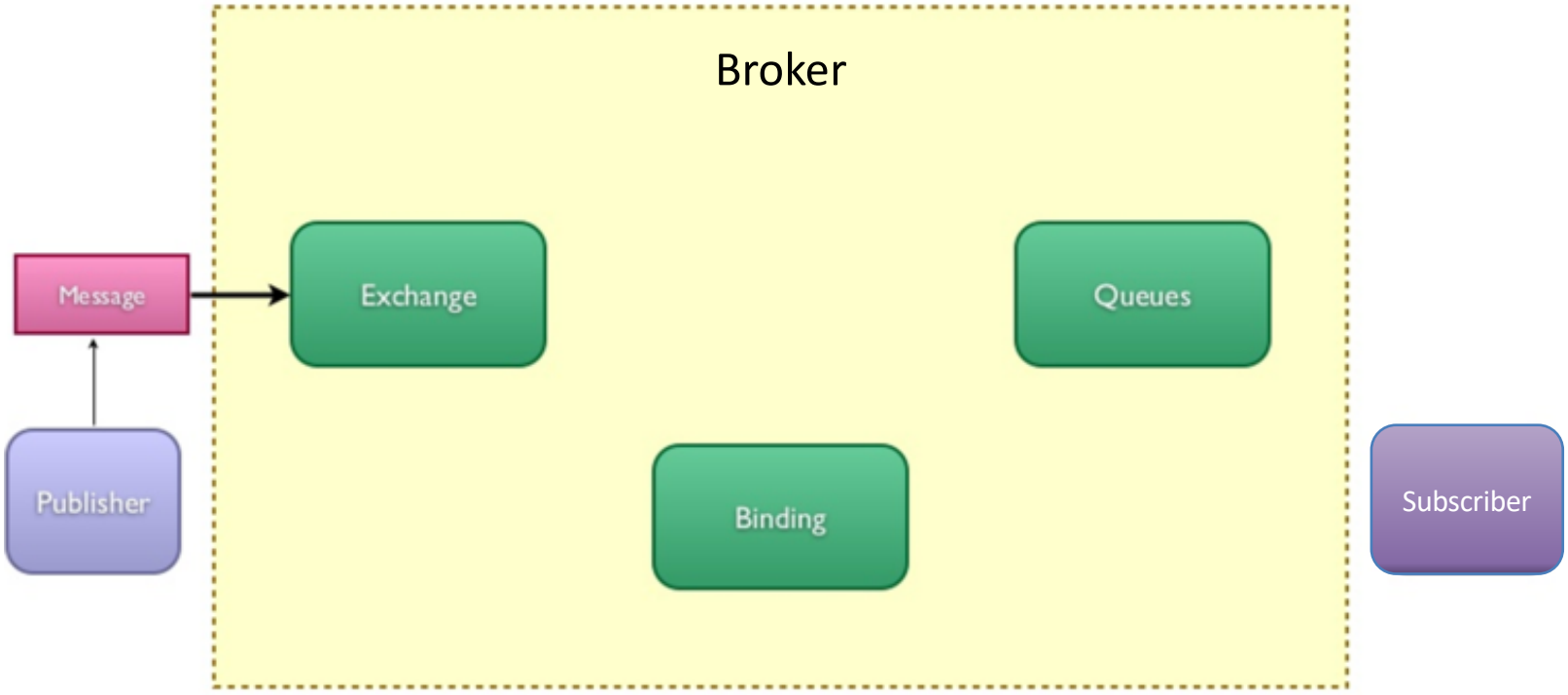
- MQTT is a binary protocol and normally requires **fixed header of 2-bytes** with small message **payloads up to** maximum size of **256 MB**.
- MQTT uses **TCP** as a transport protocol and TLS/SSL for security.
- Communication between clients and broker is **connection-oriented**.
- MQTT provides **three levels of** Quality of Service (**QoS**) for reliable delivery of messages.
- MQTT is most **suitable for large networks of small devices** that need **to be monitored or controlled from a back-end server** on the Internet.

AMPQ – Advanced Message Queue Protocol

- **AMQP** is a lightweight M2M protocol **designed for reliability, security, provisioning and interoperability**.
- AMQP supports both request/response and publish/subscribe.

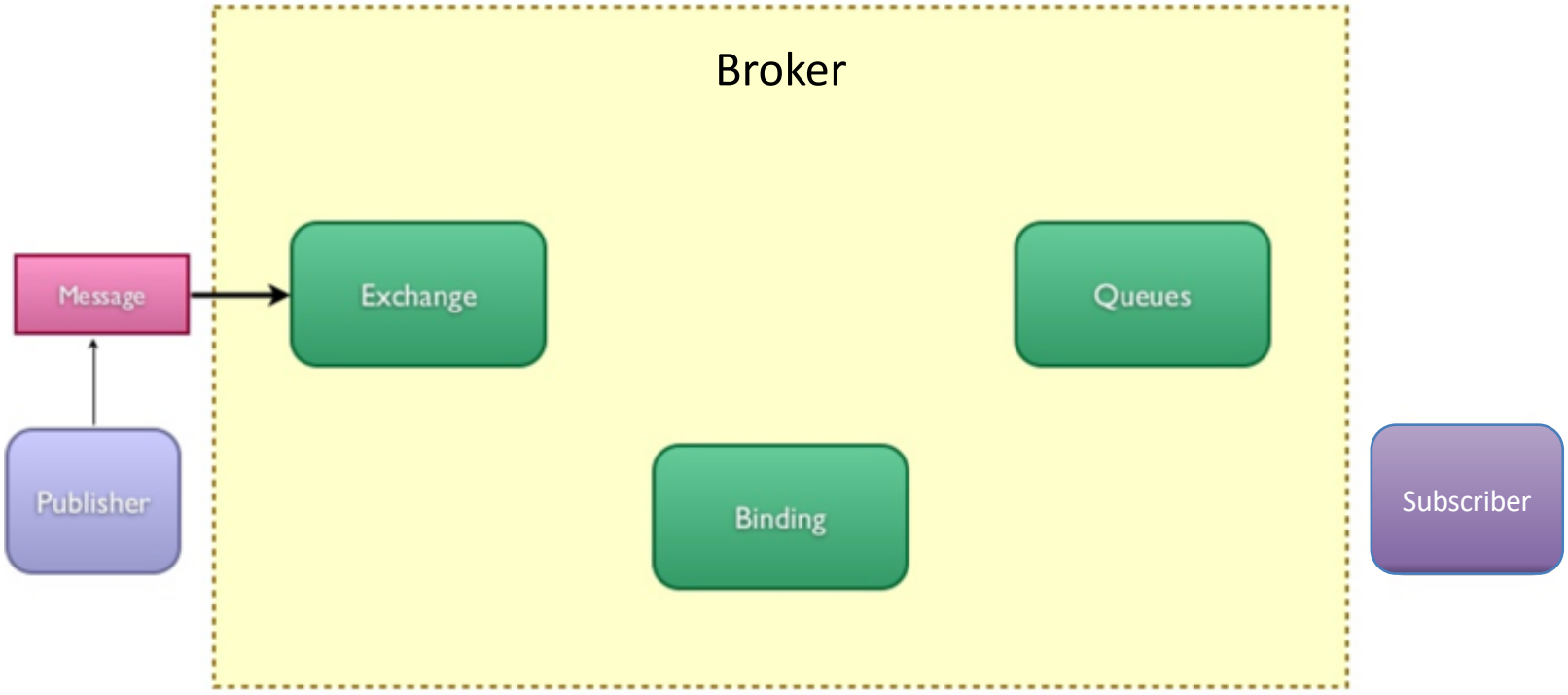
AMPQ – Advanced Message Queue Protocol

- AMQP communication system requires that:
  1. the publisher creates an “exchange” with a given name and then broadcasts that name.



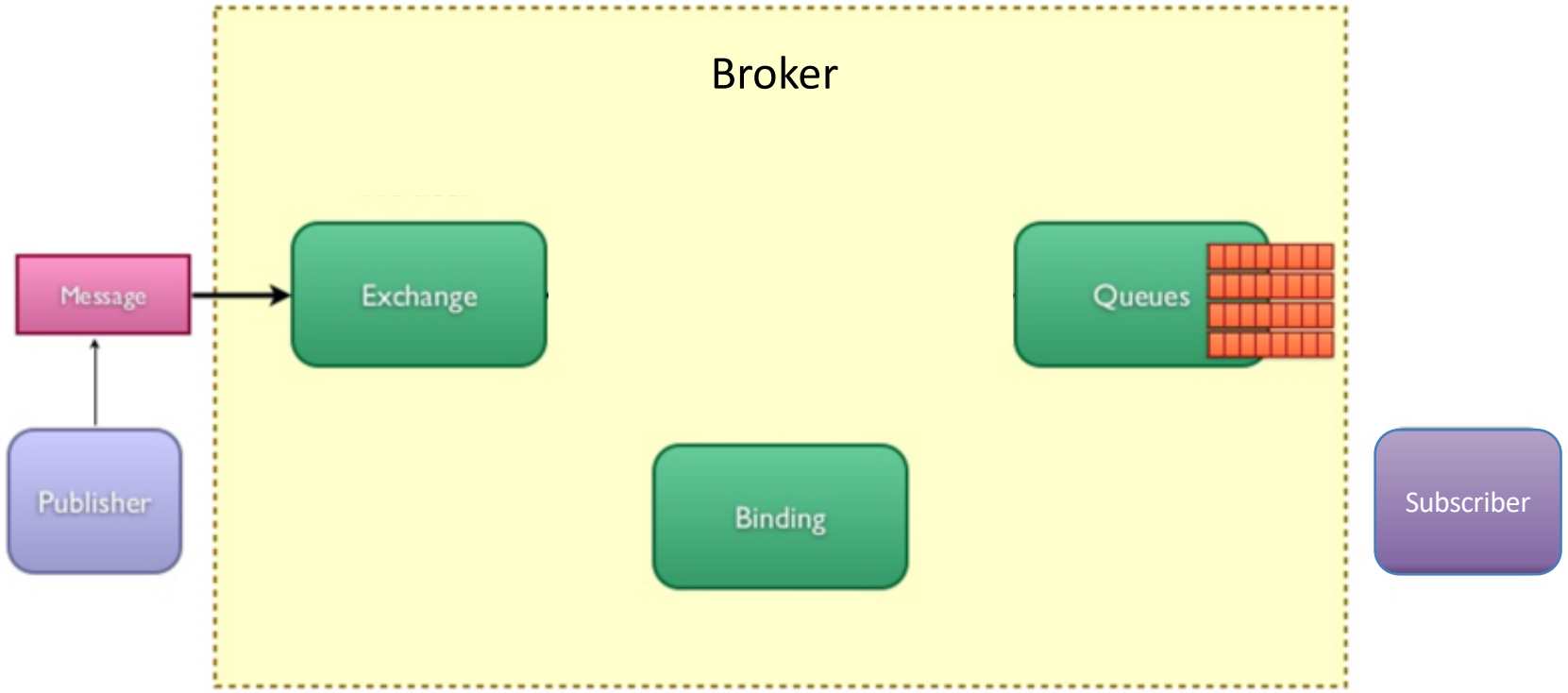
AMPQ – Advanced Message Queue Protocol

- AMQP communication system requires that:
  2. publishers and subscribers use the name of this exchange to discover each other.



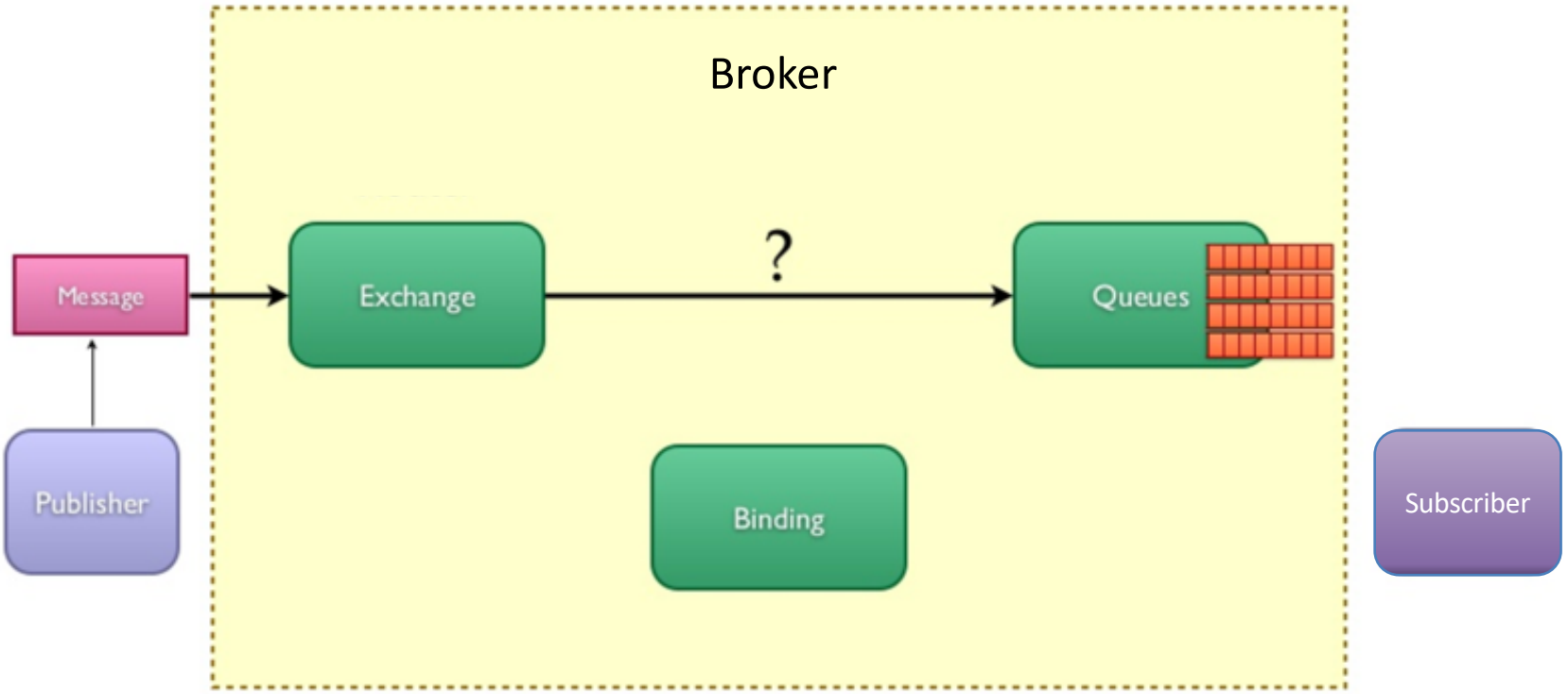
AMPQ – Advanced Message Queue Protocol

- AMQP communication system requires that:
  3. the subscriber creates a “queue” and attaches it to the exchange at the same time.



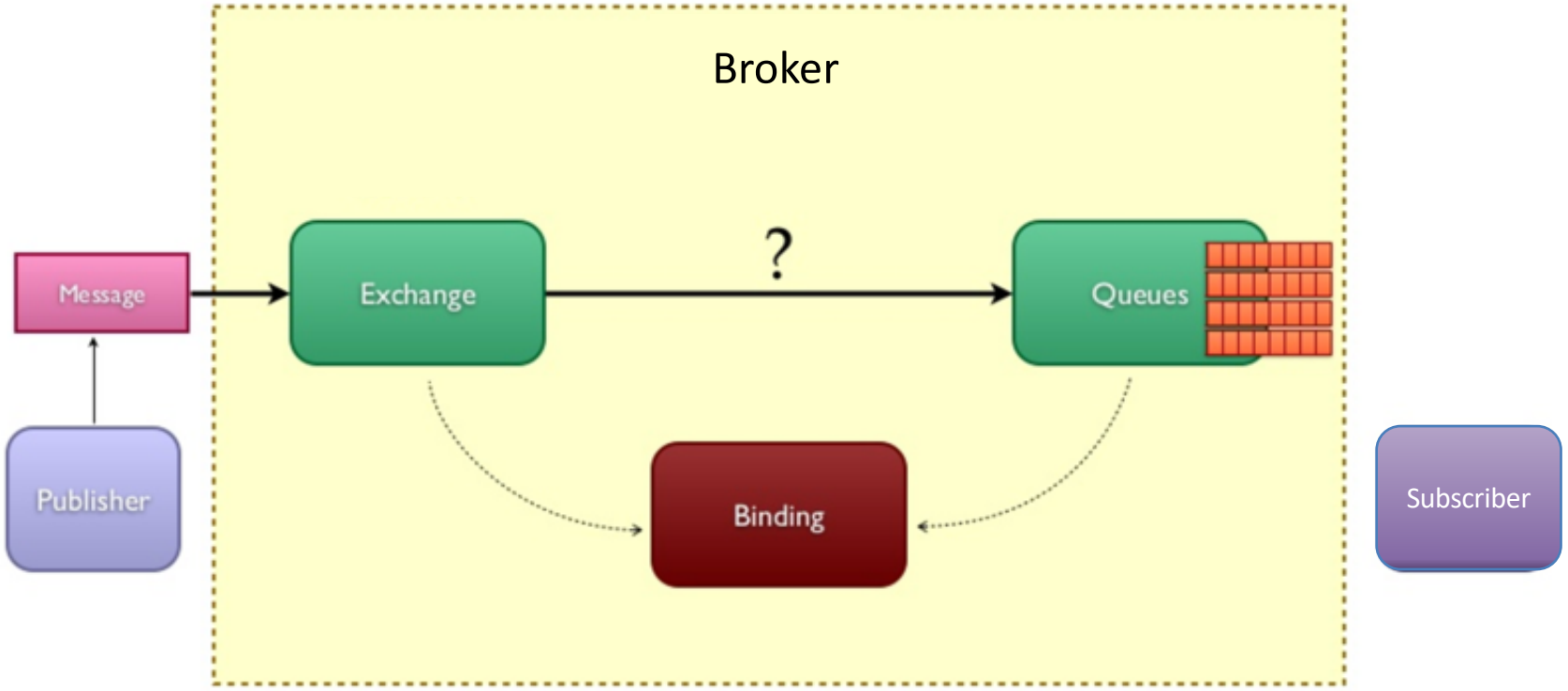
AMPQ – Advanced Message Queue Protocol

- AMQP communication system requires that:
  4. messages received by the exchange have to be matched to the queue via a process called “binding”.



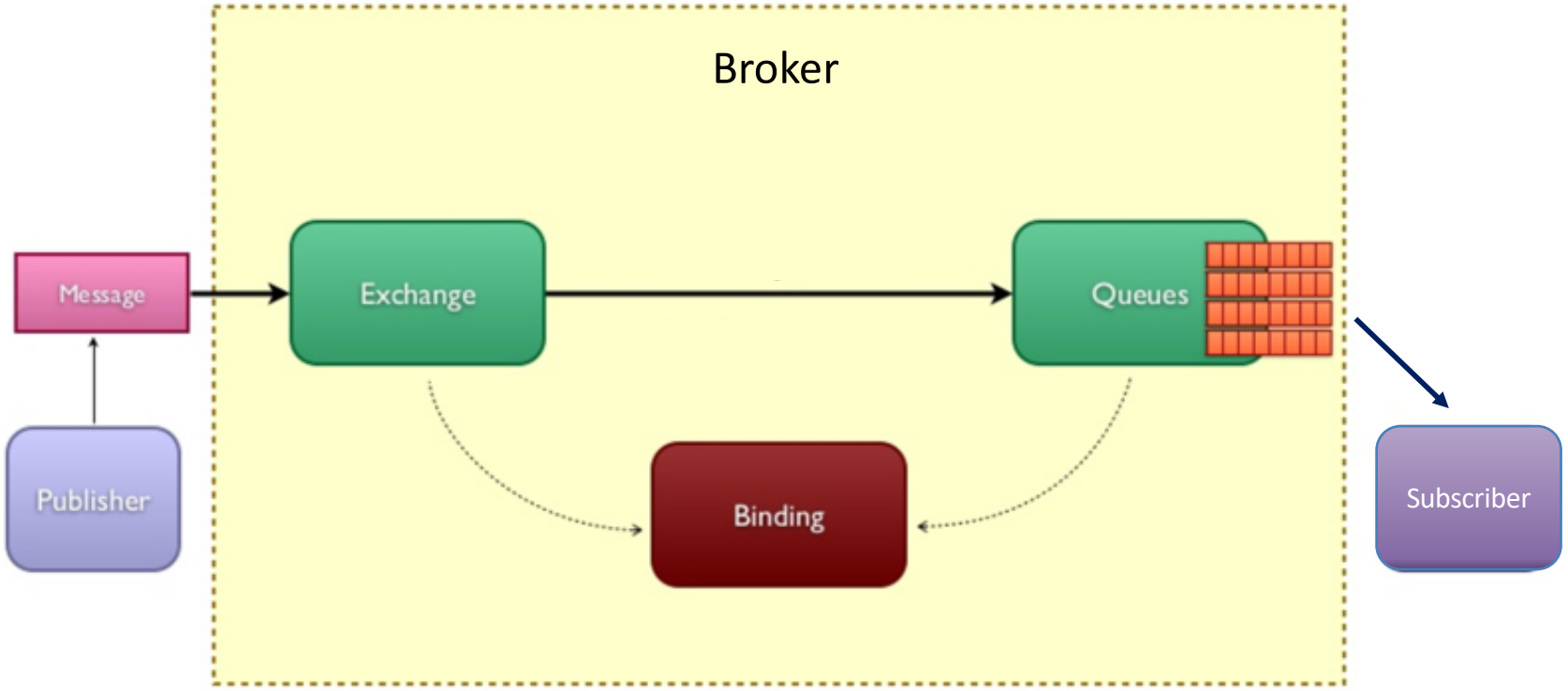
AMPQ – Advanced Message Queue Protocol

- AMQP communication system requires that:
  4. messages received by the exchange have to be matched to the queue via a process called “binding”.



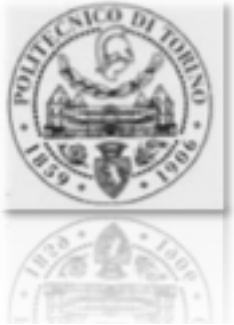
AMPQ – Advanced Message Queue Protocol

- AMQP communication system requires that:
  5. messages are sent to the subscriber



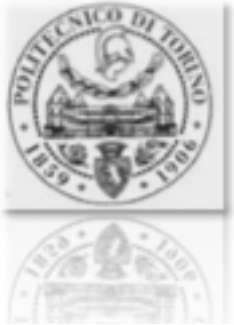
AMPQ – Advanced Message Queue Protocol

- AMQP is a binary protocol and normally requires **fixed header of 8-bytes** with **small message payloads** (depending on broker's settings).
- AMQP **uses TCP** as a default transport protocol and TLS/SSL for security.
- The communication between client and broker is a **connection-oriented**. Reliability is a core feature offering two levels of Quality of Service (**QoS**) for delivery of messages:
  - **Unsettle Format**, i.e. not reliable
  - **Settle Format**, i.e. reliable



Comparative Analysis (1)

Criteria	MQTT	CoAP	AMQP	HTTP
1. Year	1999	2010	2003	1997
2. Architecture	Client/Broker	Client/Server or Client/Broker	Client/Broker or Client/Server	Client/Server
3. Abstraction	Publish/Subscribe	Request/Response or Publish/Subscribe	Publish/Subscribe or Request/Response	Request/Response
4. Header Size	2 Byte	4 Byte	8 Byte	Undefined
5. Message Size	Small and Undefined (up to 256 MB maximum size)	Small and Undefined (normally small to fit in single IP datagram)	Negotiable and Undefined	Large and Undefined (depends on the web server or the programming technology)
6. Semantics/ Methods	Connect, Disconnect, Publish, Subscribe, Unsubscribe, Close	Get, Post, Put, Delete	Consume, Deliver, Publish, Get, Select, Ack, Delete, Nack, Recover, Reject, Open, Close	Get, Post, Head, Put, Patch, Options, Connect, Delete
7. Cache and Proxy Support	Partial	Yes	Yes	Yes



Comparative Analysis (2)

Criteria	MQTT	CoAP	AMQP	HTTP
8. Quality of Service (QoS)/ Reliability	QoS 0 - At most once (Fire-and-Forget), QoS 1 - At least once, QoS 2 - Exactly once	Confirmable Message (similar to At most once) or Non-confirmable Message (similar to At least once)	Settle Format (similar to At most once) or Unsettle Format (similar to At least once)	Limited (via Transport Protocol - TCP)
9. Standards	OASIS, Eclipse Foundations	IETF, Eclipse Foundation	OASIS, ISO/IEC	IETF and W3C
10. Transport Protocol	TCP (MQTT-SN can use UDP)	UDP, SCTP	TCP, SCTP	TCP
11. Security	TLS/SSL	DTLS, IPSec	TLS/SSL, IPSec, SASL	TLS/SSL
12. Default Port	1883/ 8883 (TLS/SSL)	5683 (UDP Port)/ 5684 (DLTS)	5671 (TLS/SSL), 5672	80/ 443 (TLS/SSL)
13. Encoding Format	Binary	Binary	Binary	Text
14. Licensing Model	Open Source	Open Source	Open Source	Free
15. Organisational Support	IBM, Facebook, Eurotech, Cisco, Red Hat, Software AG, Tibco, ITSO, M2Mi, Amazon Web Services (AWS), InduSoft, Fiorano	Large Web Community Support, Cisco, Contiki, Erika, IoTivity	Microsoft , JP Morgan, Bank of America, Barclays, Goldman Sachs, Credit Suisse	Global Web Protocol Standard

What is the best messaging protocol?

What is the best messaging protocol?

The selection of a standard and effective messaging protocol is a challenging and daunting task for any organisation because **it depends on the nature of the IoT system and its messaging requirements**

What is the best messaging protocol?

The selection of a standard and effective messaging protocol is a challenging and daunting task for any organisation because **it depends on the nature of the IoT system and its messaging requirements**

Messaging protocol is an ongoing dilemma for the IoT industry; consequently, it is important to understand the pros and cons of the widely accepted and emerging messaging protocols for IoT systems to determine their best-fit scenarios.

Analysis od messaging protocols or IoT Systems

HTTP, CoAP, MQTT and AMQP are **very extensive protocols and different from each other** because they have been evolved through different processes and needs.

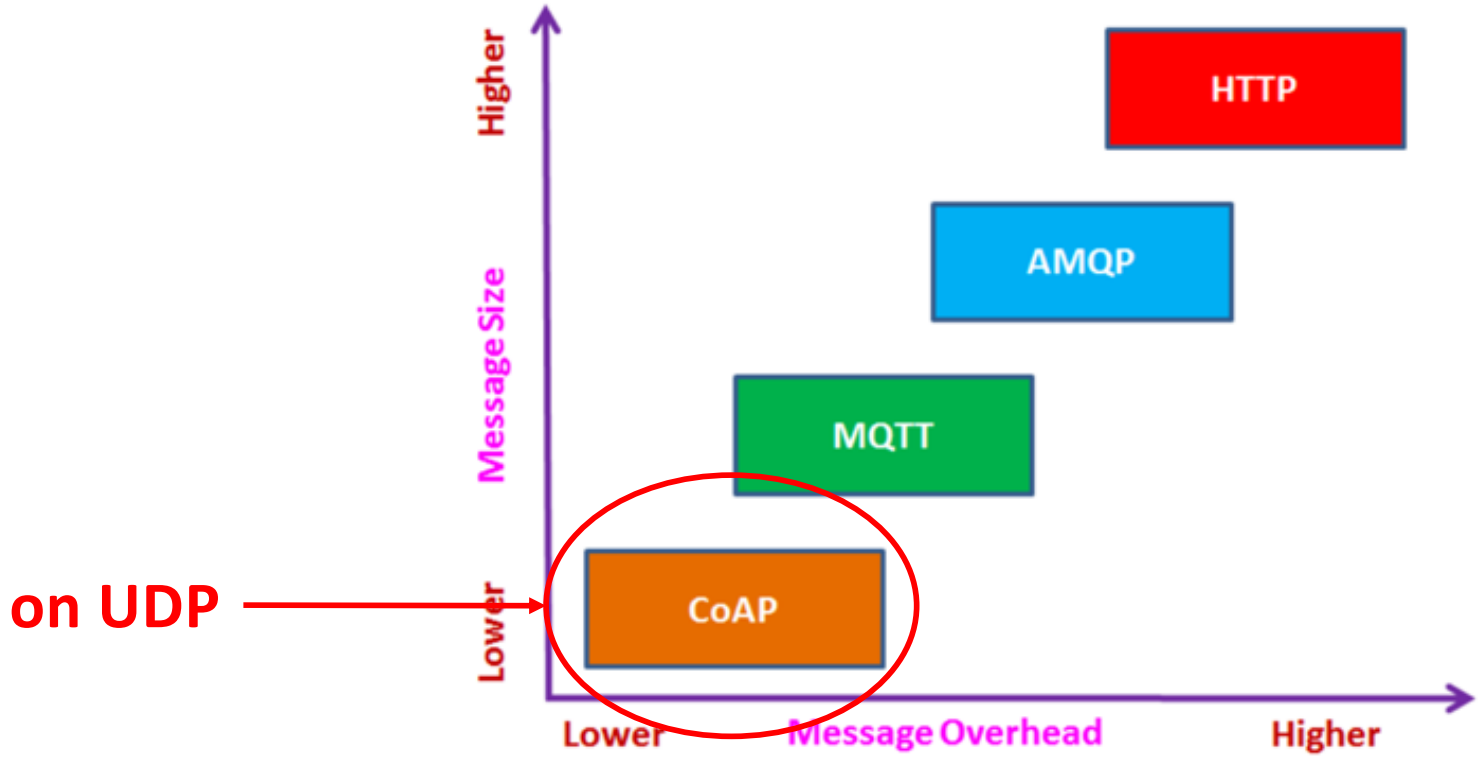
Their precise and relative comparisons depend on the types of IoT systems, devices, resources, applications, and specific conditions and requirements of the system.

Analysis od messaging protocols or IoT Systems

The evaluation is based on static components and some empirical evidence from the literature. Nonetheless, it does not consider the dynamic network conditions and overheads incurring in the retransmission of packets, which may also change comparison results.

Message Size vs. Message Overhead

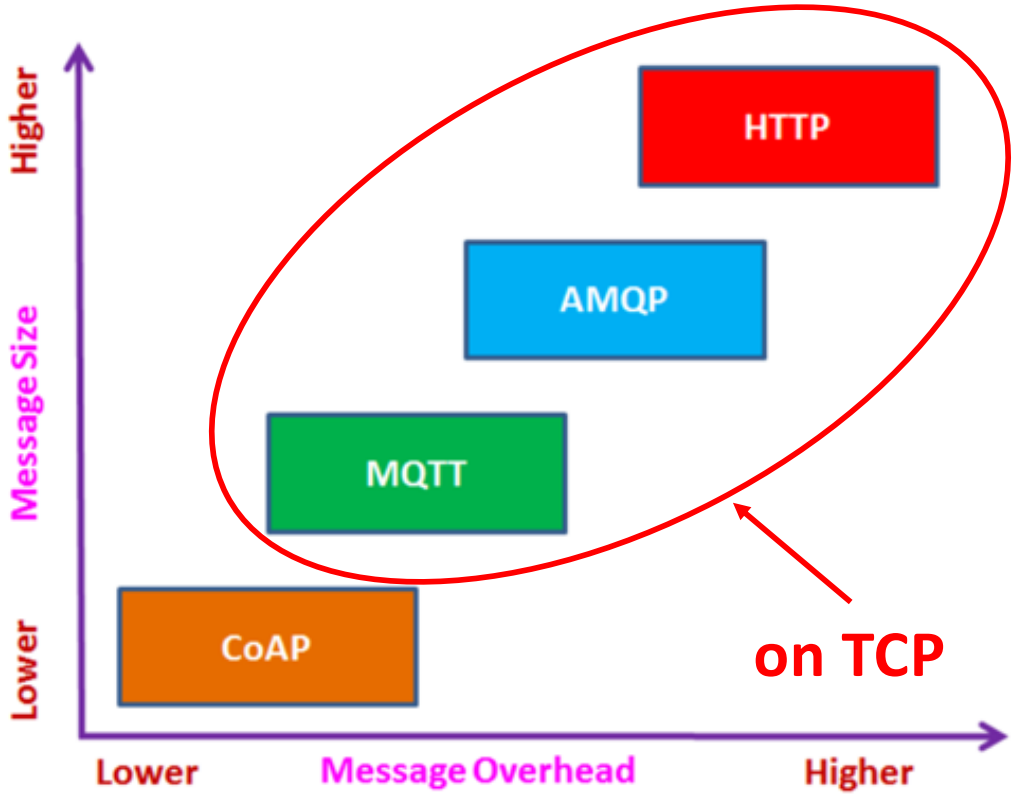
CoAP runs on UDP. It does not incur connection overheads as UDP works in fire and forget basis. This reduces the overall overhead considerably, and thus the whole message size.



Message Size vs. Message Overhead

Protocols running on TCP incurs on TCP connection overheads for connection establishment and closing.

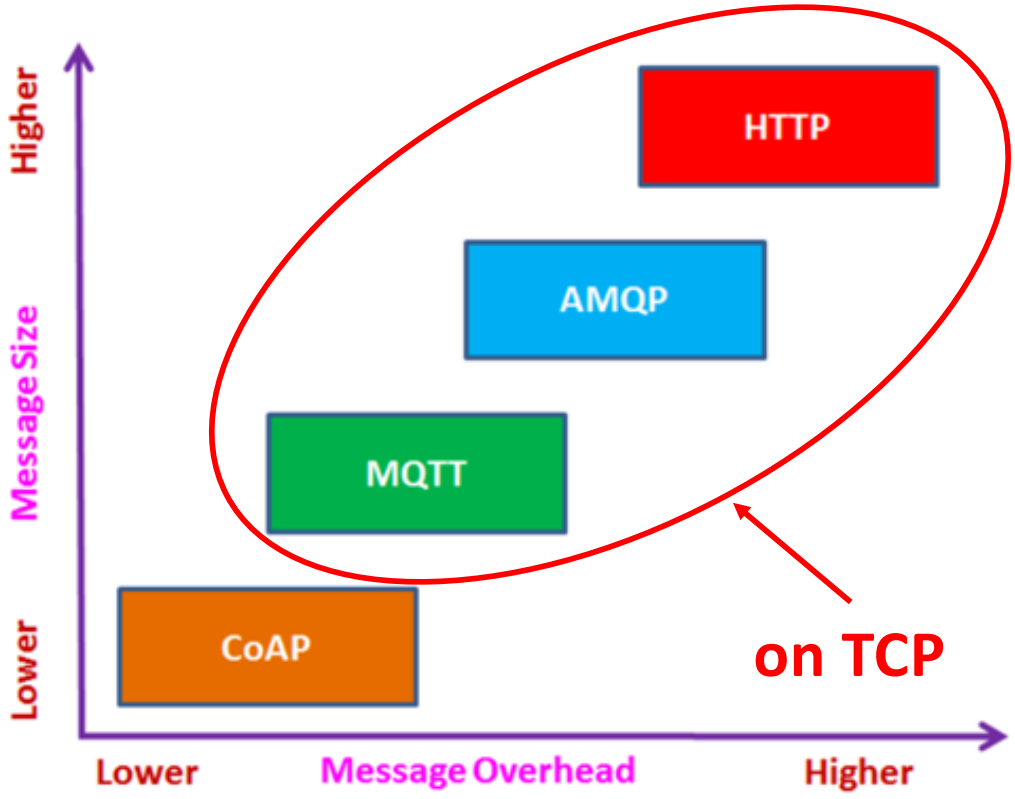
MQTT is lightweight and has the least header size of 2-byte per message but its requirement of TCP connection increases the overall overhead, and thus the whole message size.



Message Size vs. Message Overhead

Protocols running on TCP incurs on TCP connection overheads for connection establishment and closing.

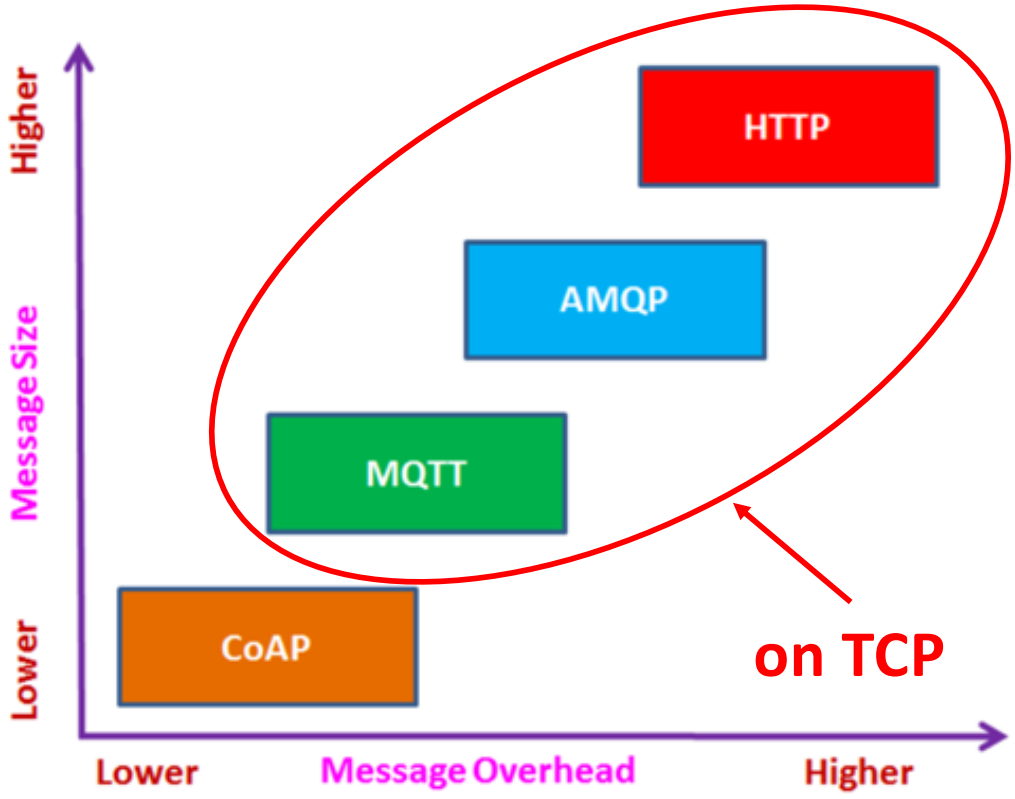
AMQP is a lightweight binary protocol; however, its support for security, reliability, provisioning and interoperability increases the overhead and message size.



Message Size vs. Message Overhead

Protocols running on TCP incurs on TCP connection overheads for connection establishment and closing.

HTTP among all four **is the most verbose and heavyweight protocol**. It was originally designed for the Web and not for the IoT. It requires maximum overhead and message size among all.

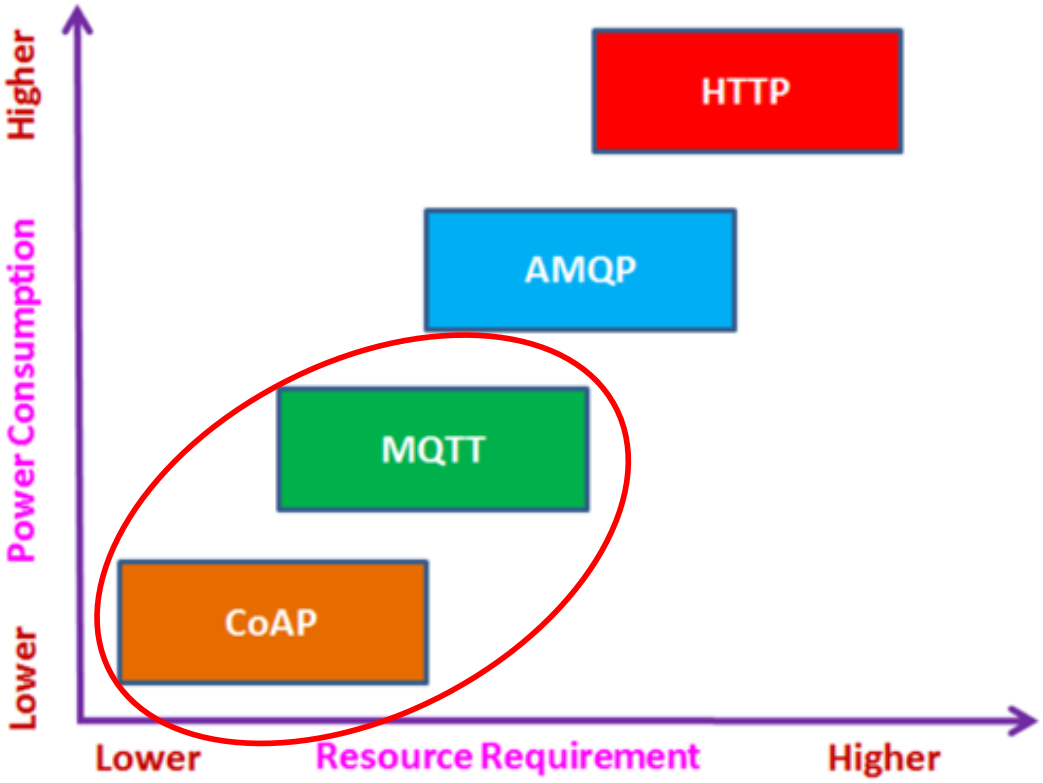


Power Consumption vs. Resource Requirement

The graph highlights the similar patterns as the first one.

Both **CoAP and MQTT** are designed **for low bandwidth** and **resource-constrained devices**.

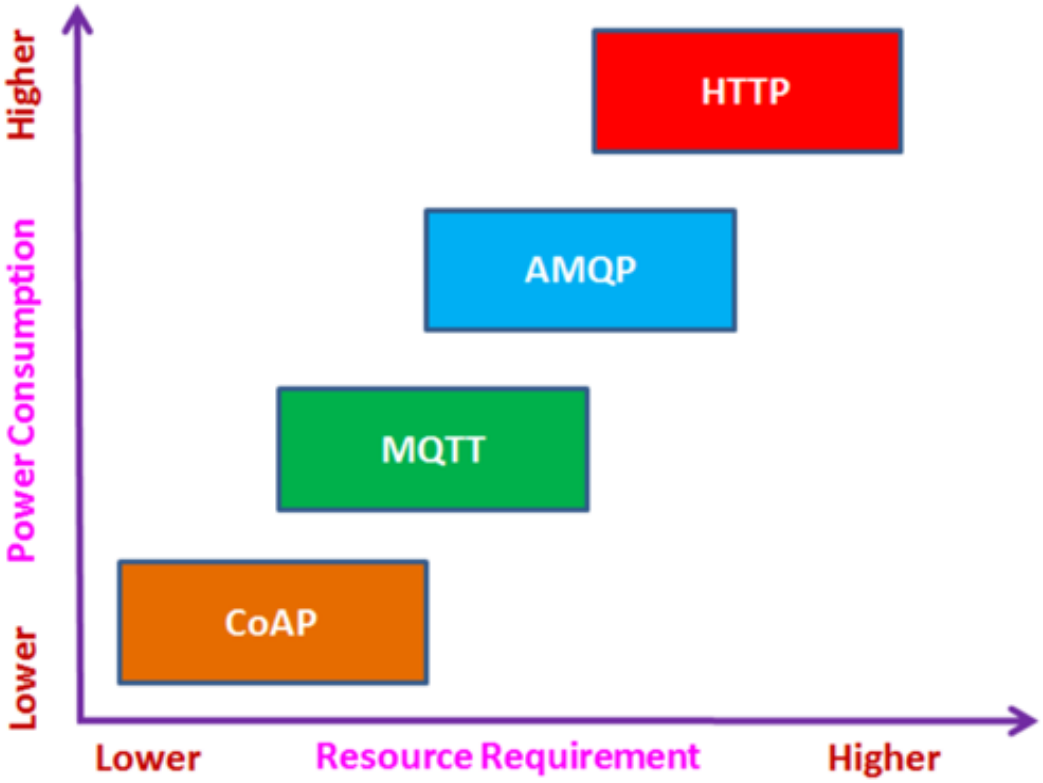
They can be used on an 8-bit controllers and 100s bytes of memory.



Power Consumption vs. Resource Requirement

The graph highlights the similar patterns as the first one.

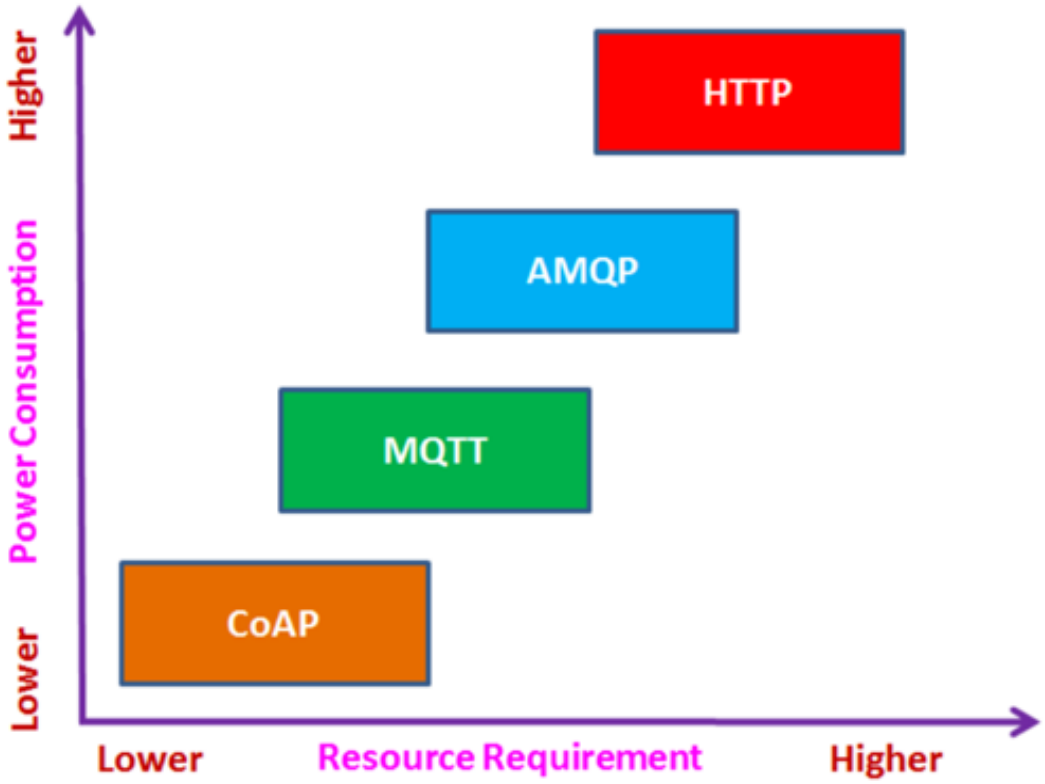
AMQP requires **slightly higher power and resources** due to performing other necessary operations for provisioning and reliability.



Power Consumption vs. Resource Requirement

The graph highlights the similar patterns as the first one.

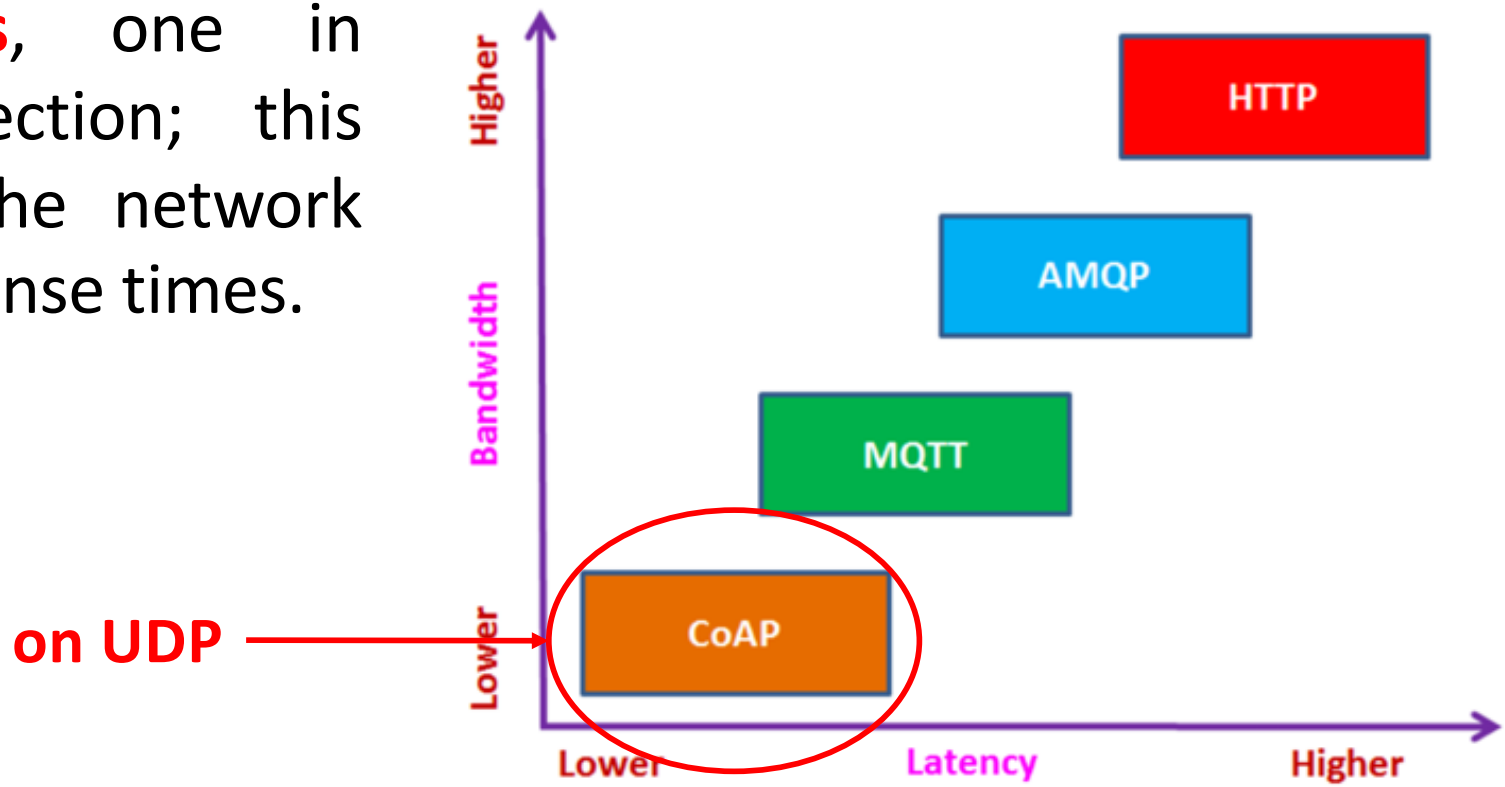
HTTP requires highest power and resources than the other protocols for the same operation.



Bandwidth vs. Latency

The graph reveals the very similar patterns as the first two.

In CoAP, **a UDP transaction requires only two UDP datagrams**, one in each direction; this reduces the network load response times.

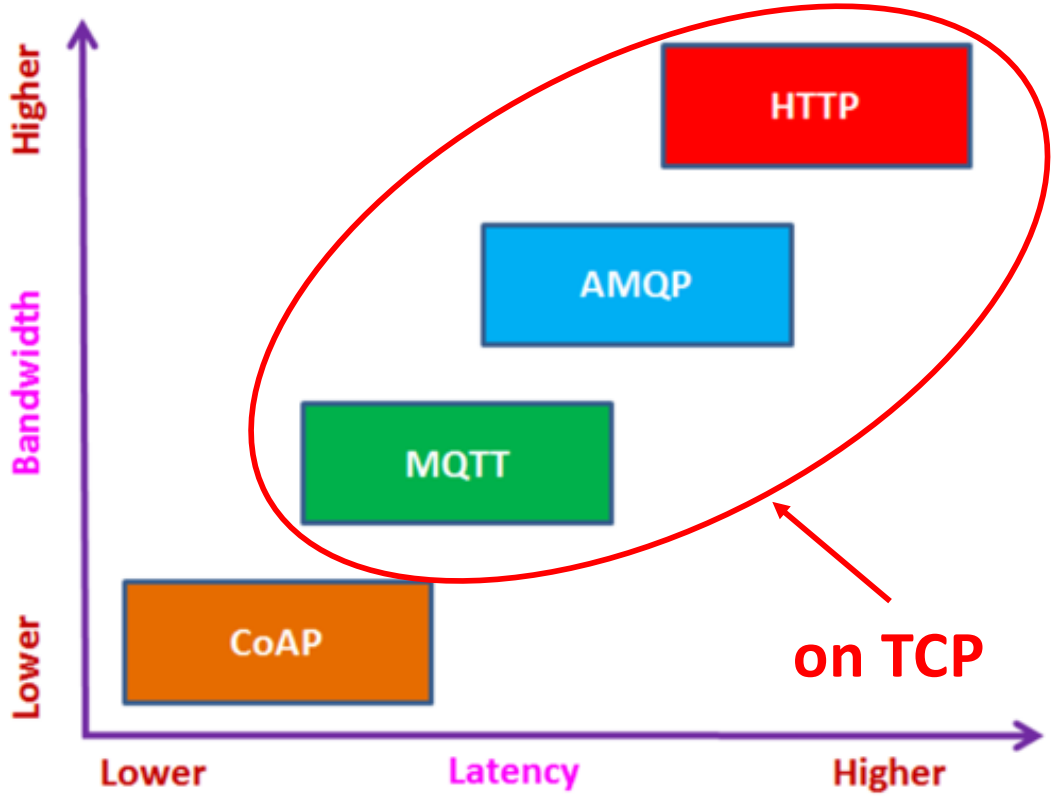


Bandwidth vs. Latency

The graph reveals the very similar patterns as the first two.

The use of TCP is a major factor in determining the latency

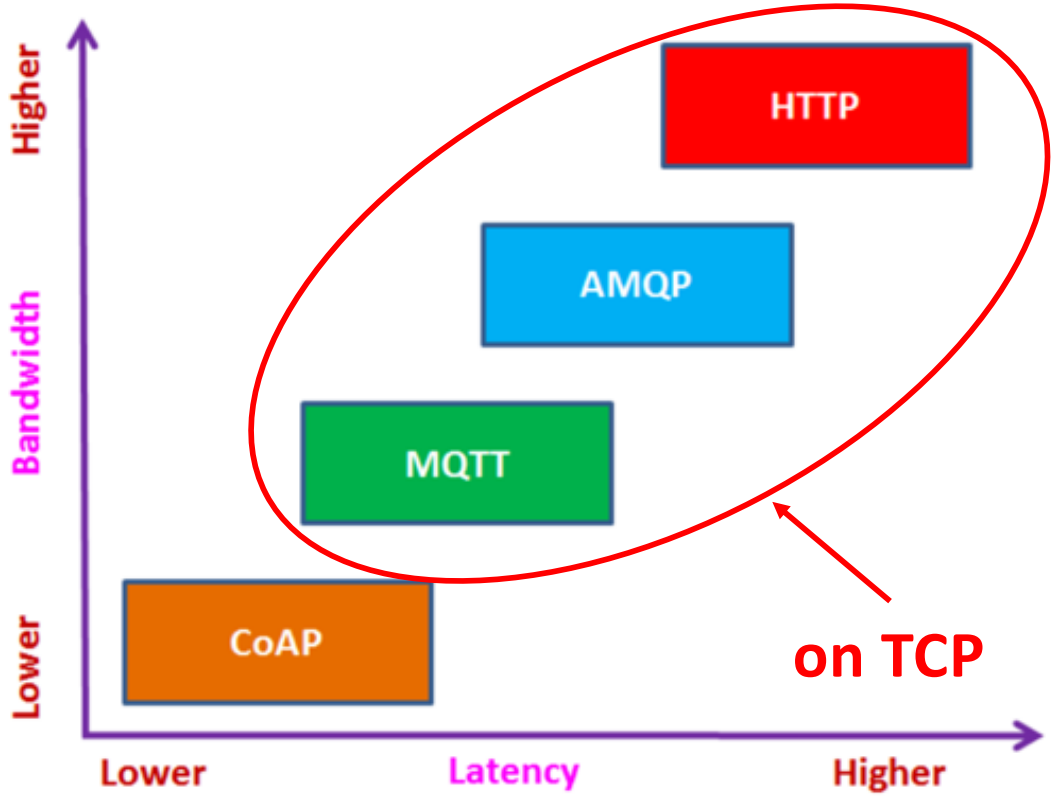
and bandwidth requirement.
TCP does not help in improving latency **because it does not fully utilize the whole available network bandwidth to avoid network congestion**.



Bandwidth vs. Latency

The graph reveals the very similar patterns as the first two.

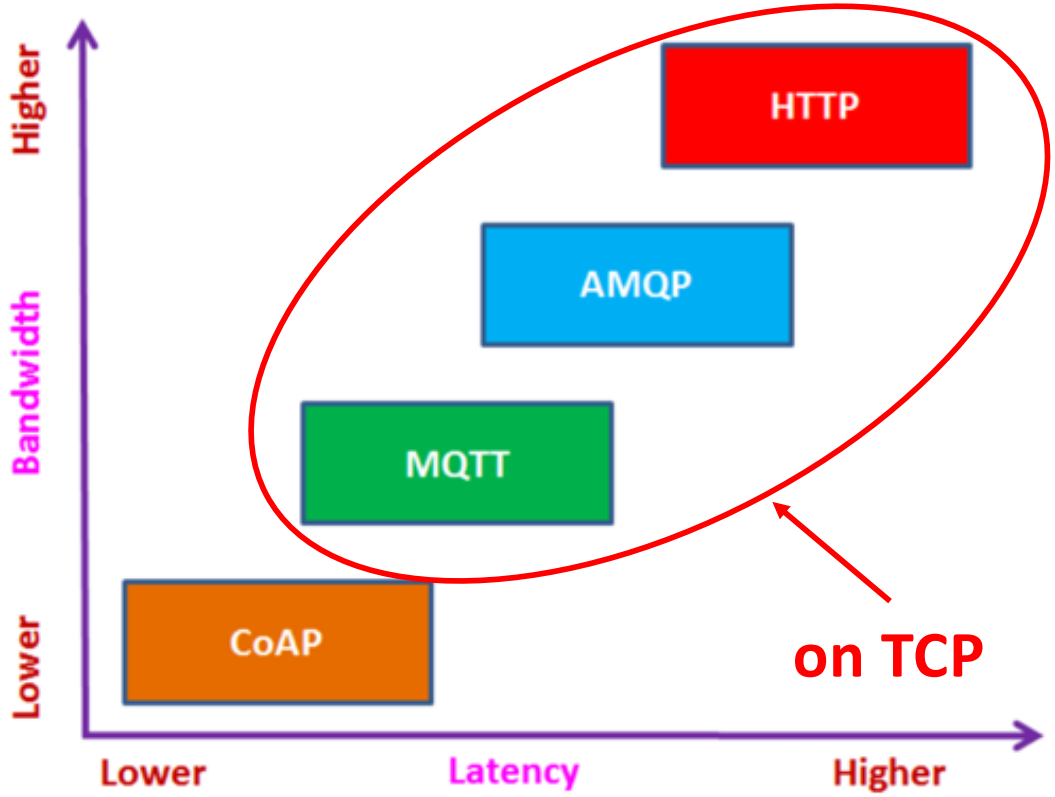
Various experimental studies found that MQTT consumes higher bandwidth than CoAP for transferring same payload under same network condition.



Bandwidth vs. Latency

The graph reveals the very similar patterns as the first two.

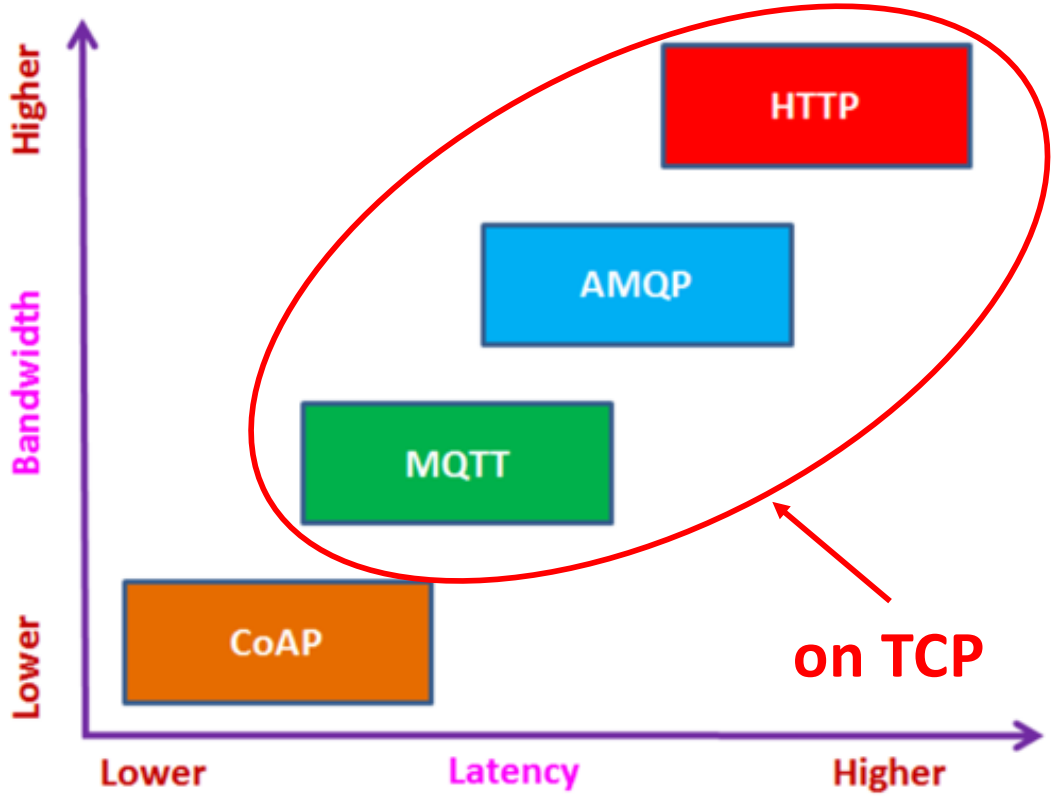
AMQP's extra services demand moderately higher bandwidth and latency



Bandwidth vs. Latency

The graph reveals the very similar patterns as the first two.

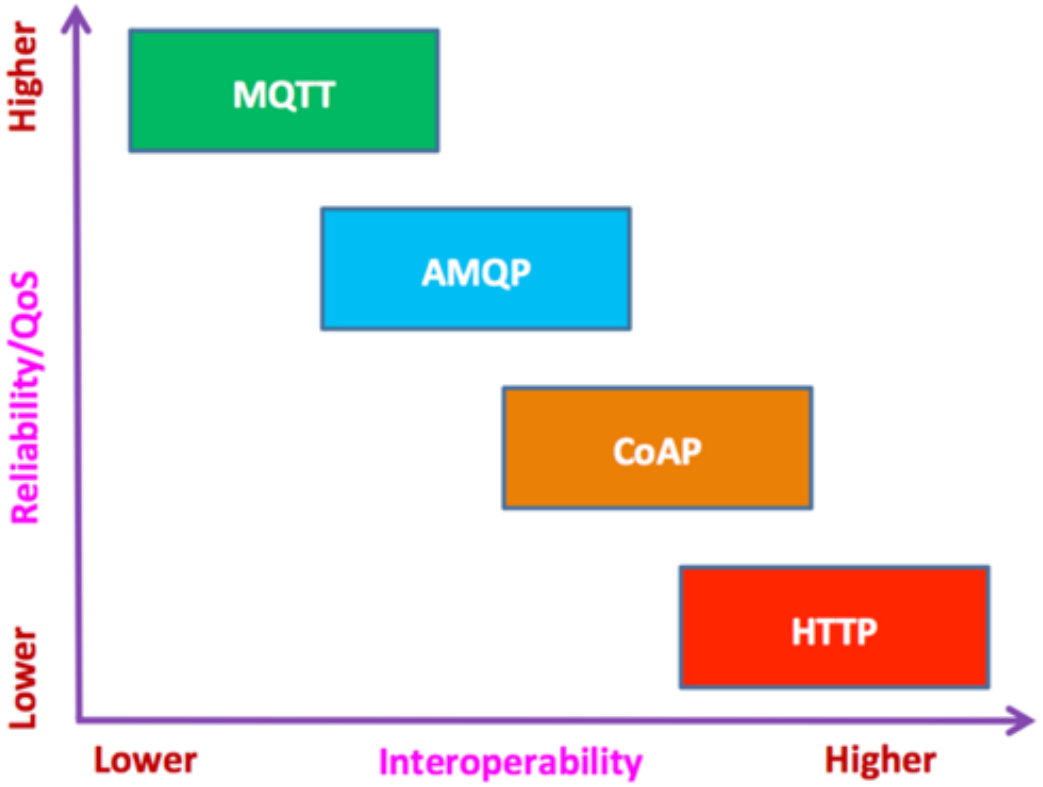
HTTP involves largest bandwidth and latency than any other protocols



Reliability/QoS vs. Interoperability

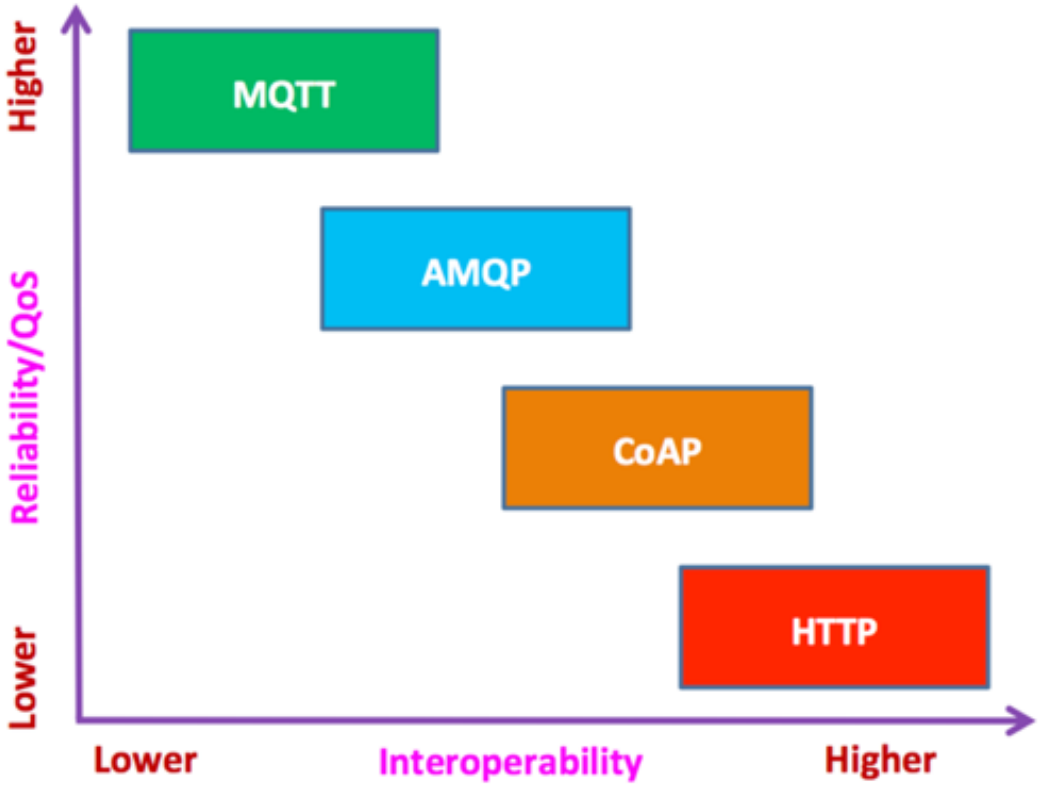
Reliability: MQTT offers the highest level of quality of services with least interoperability. MQTT defines three QoS levels:

QoS-0 at most once (only TCP guarantee)
QoS-1 at least once (MQTT guarantee with confirmation)
QoS-2 exactly once (MQTT guarantee with handshake).



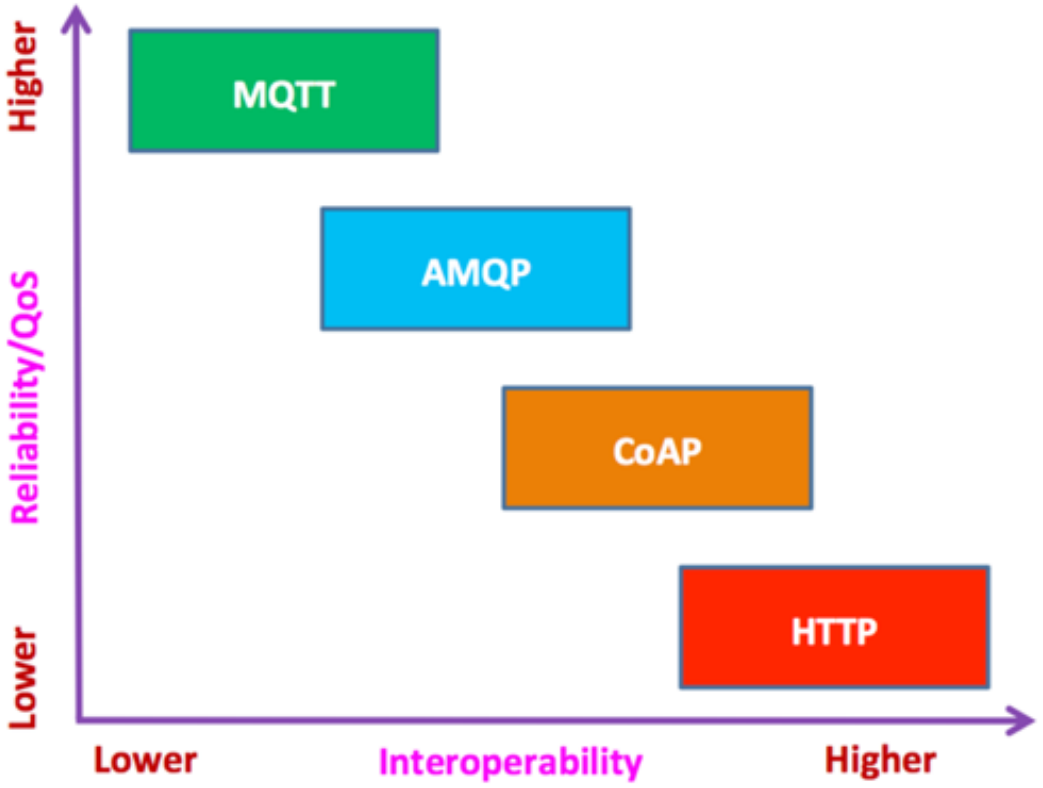
Reliability/QoS vs. Interoperability

Interoperability: MQTT only supports the publish/subscribe pattern of communication, which barely covers all use cases within the IoT.



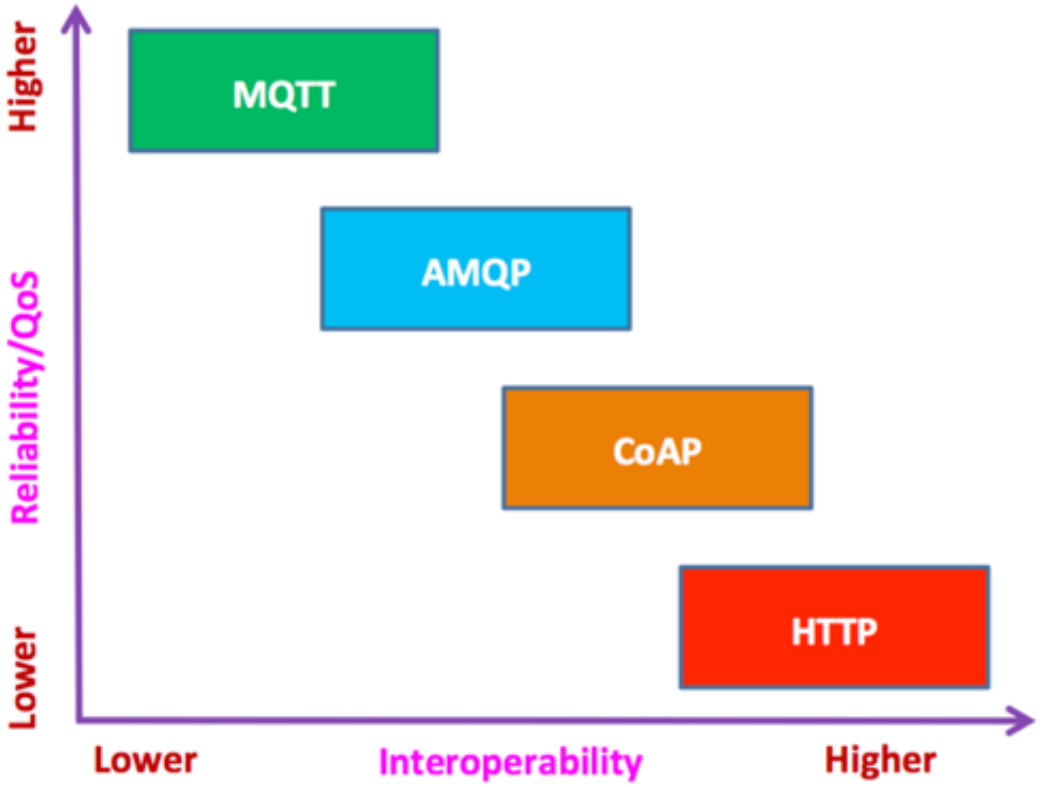
Reliability/QoS vs. Interoperability

Reliability: AMQP defines two QoS levels: **Settle Format** (similar to MQTT QoS-0) and **Unsettle Format** (similar to MQTT QoS-1).



Reliability/QoS vs. Interoperability

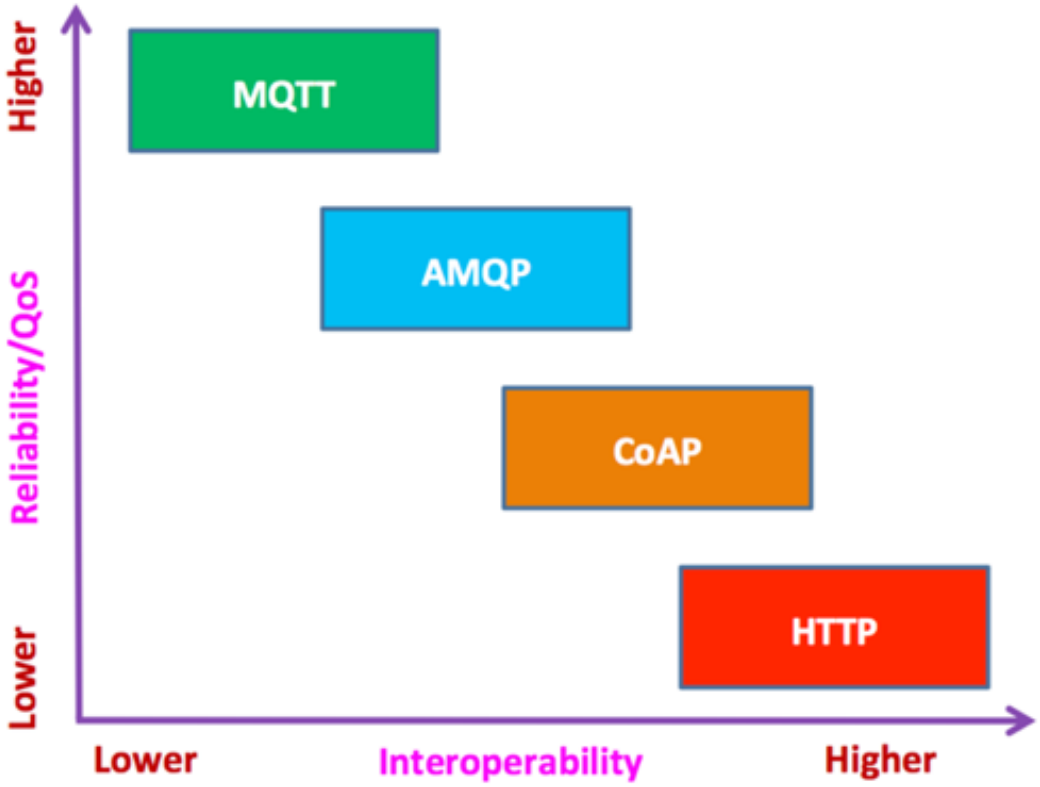
Interoperability: AMQP provides features to use serialization formats such as Protocol Buffers, MessagePack, Thrift, and JSON to serialize structured data in order to publish it as the message payload.



Reliability/QoS vs. Interoperability

Reliability: CoAP compensates for the unreliability of UDP by defining a retransmission mechanism and providing resource discovery mechanism with resource description.

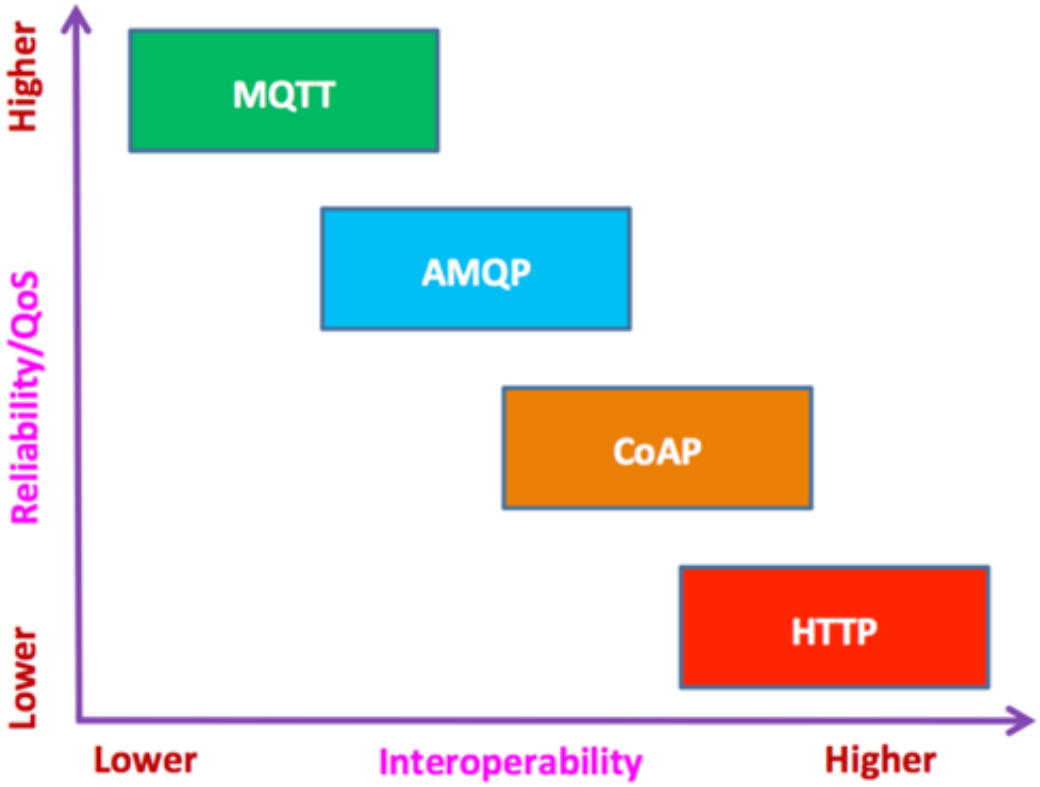
CoAP provides features for **non-confirmable** messages (similar to MQTT QoS-0) and **confirmable** messages (similar to MQTT QoS-1)



Reliability/QoS vs. Interoperability

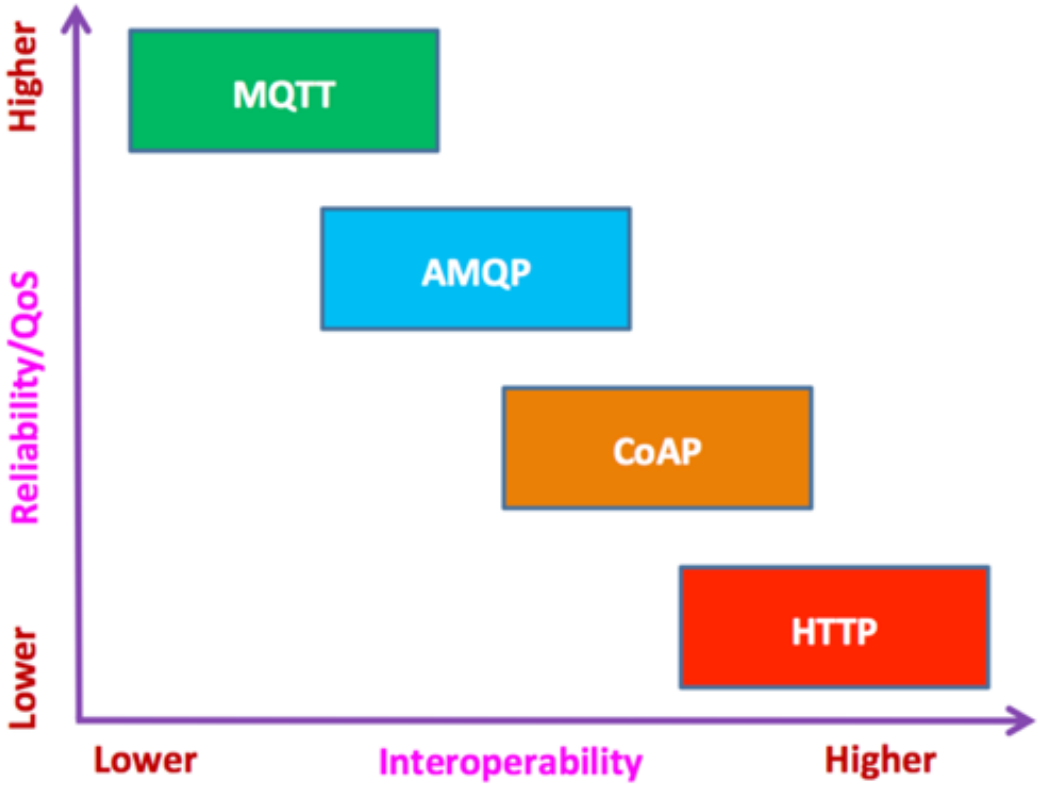
Interoperability: CoAP is a part of the Web architecture and best suited for devices that support UDP, however, making it **limited to a few special kinds of IoT devices**.

HTTP-based RESTful clients and servers are the most interoperable because all that is needed to support message exchanges, is an HTTP stack.



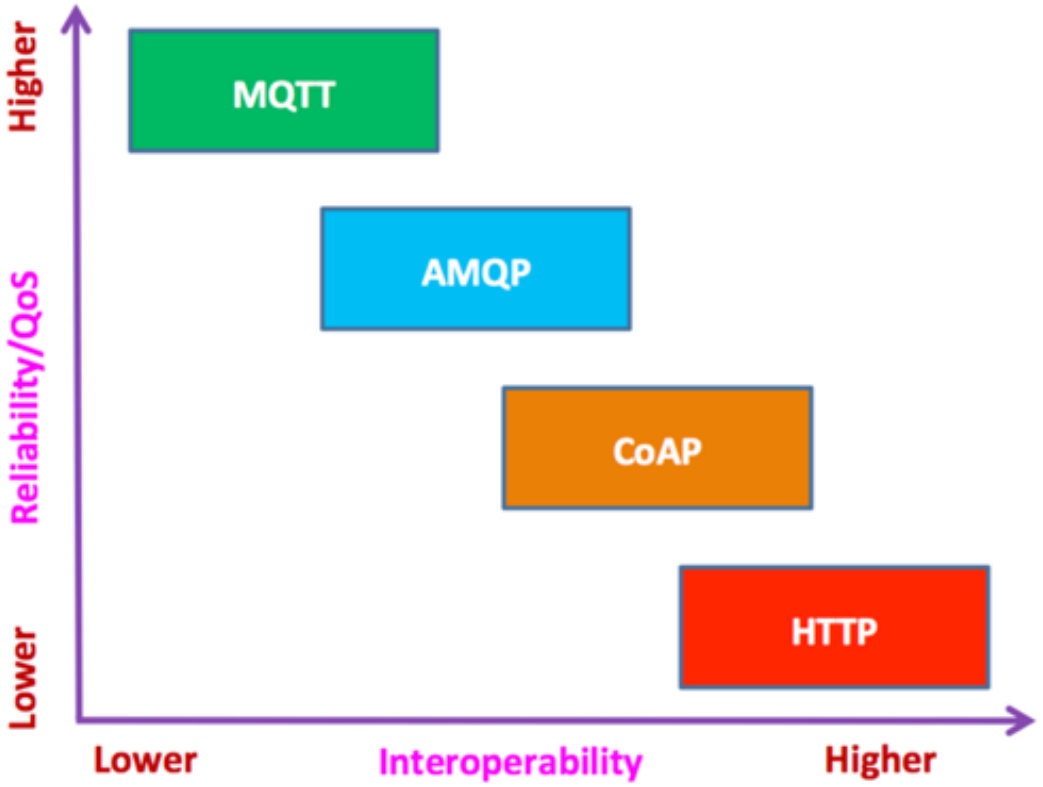
Reliability/QoS vs. Interoperability

HTTP was designed for **greatest interoperability** on the Web and **did not include reliability** as a core feature.



Reliability/QoS vs. Interoperability

One of the biggest benefits of using TCP as a transport protocol by MQTT, AMQP and HTTP is the guaranteed delivery of a packet.



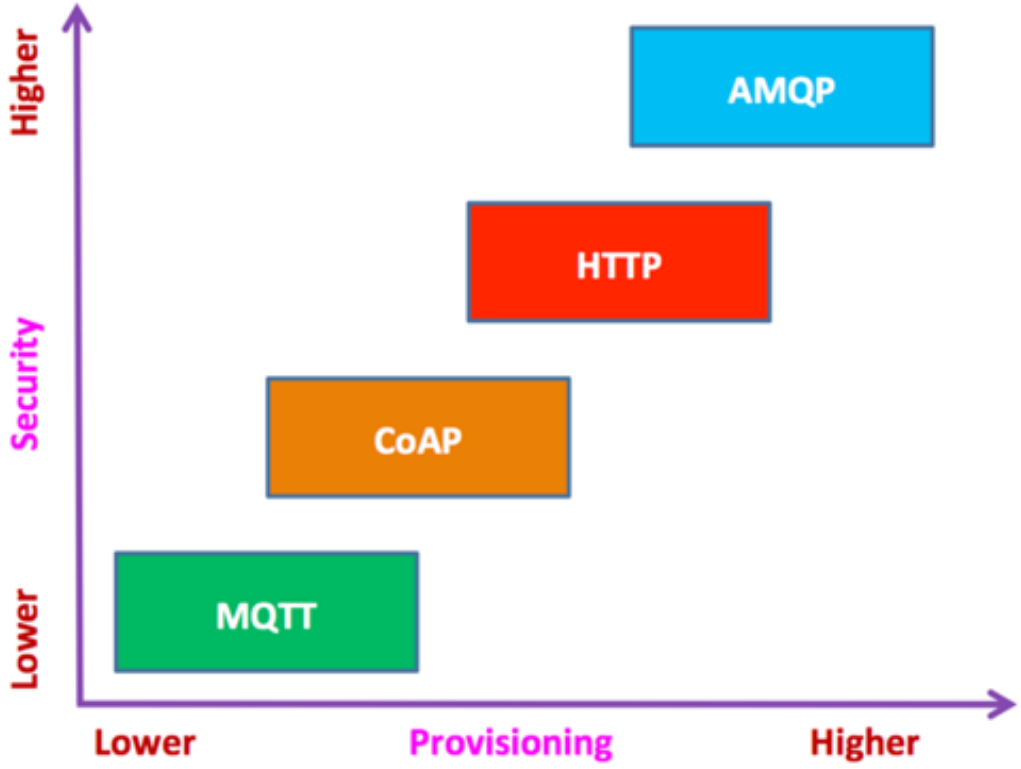
Security vs. Provisioning

MQTT is barely a messaging protocol and **supports the lowest level of security**. Except TLS/SSL, MQTT has minimal authentication features and only rely on simple username and password.

MQTT does not offer any extra services even message labelling.

Messages can be used for any purpose.

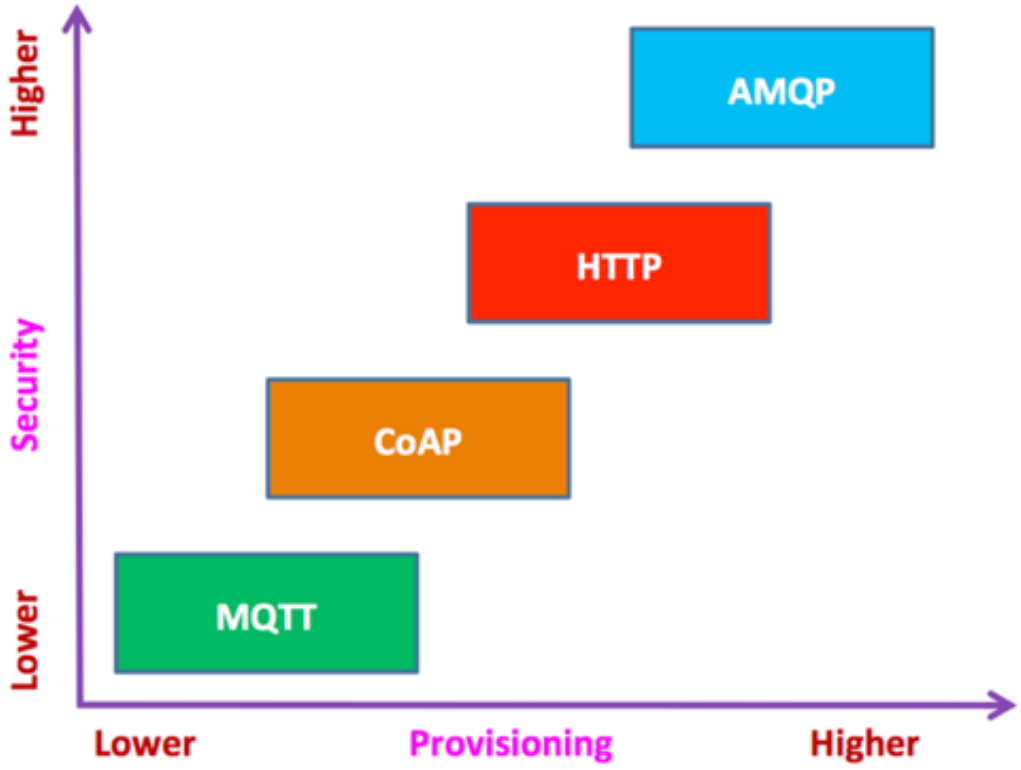
All clients must know the message formats up-front to allow communication.



Security vs. Provisioning

CoAP uses two methods DTLS and IPsec for authentication, integrity and encryption.

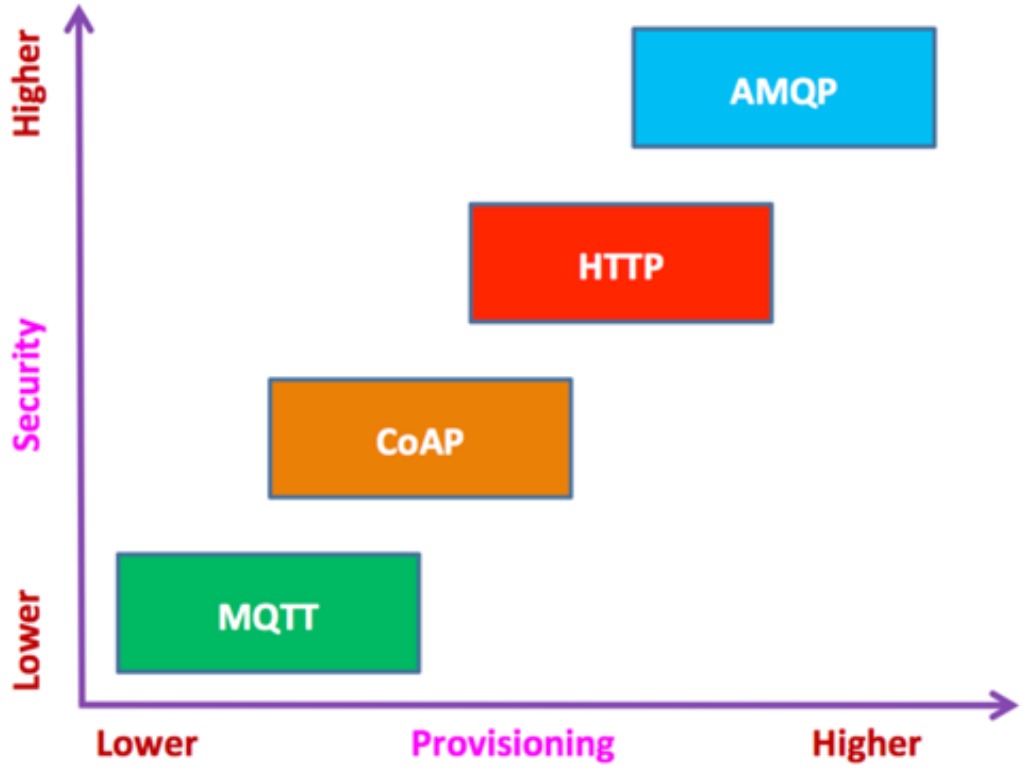
In CoAP, there are several extensions for enhanced services depending on the requirements of the IoT system such as support for multicast group communications, resource discovery and block-wise transfers.



Security vs. Provisioning

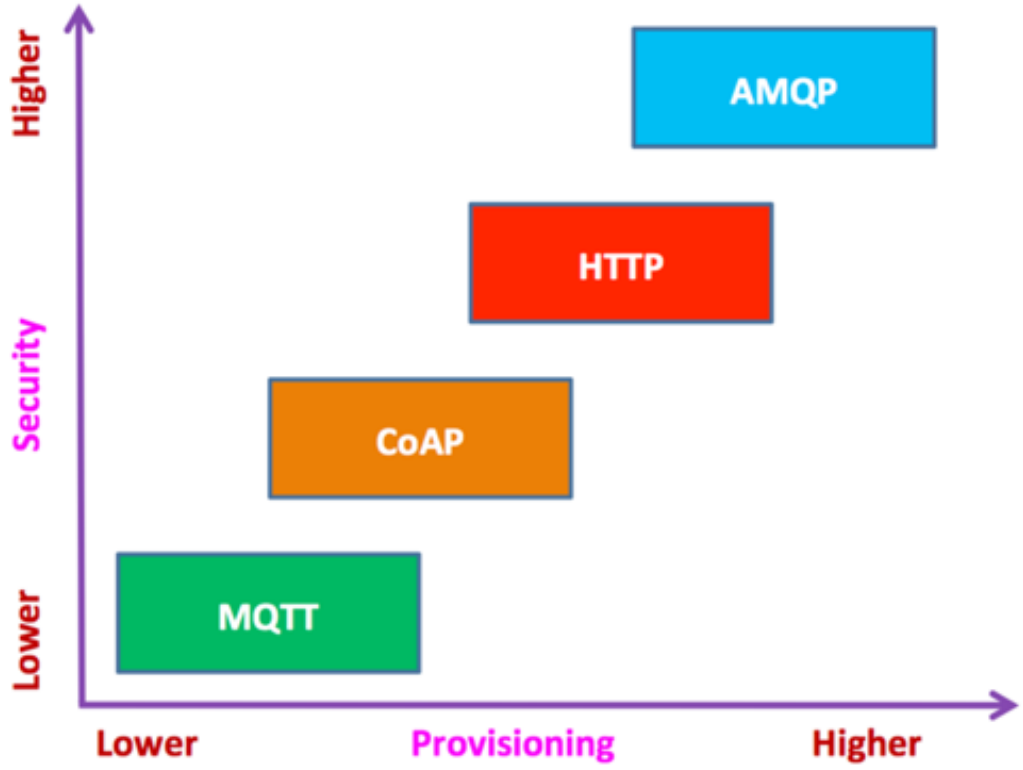
HTTP facilitates two authentication approaches: **HTTP Basic** and **HTTP Digest**. HTTP basic authentication uses unencrypted Base64-encoding username and password to authenticate a service client over TLS/SSL.

HTTP offers several services such as multiplexing and concurrency, stream dependencies and prioritization, header compression.



Security vs. Provisioning

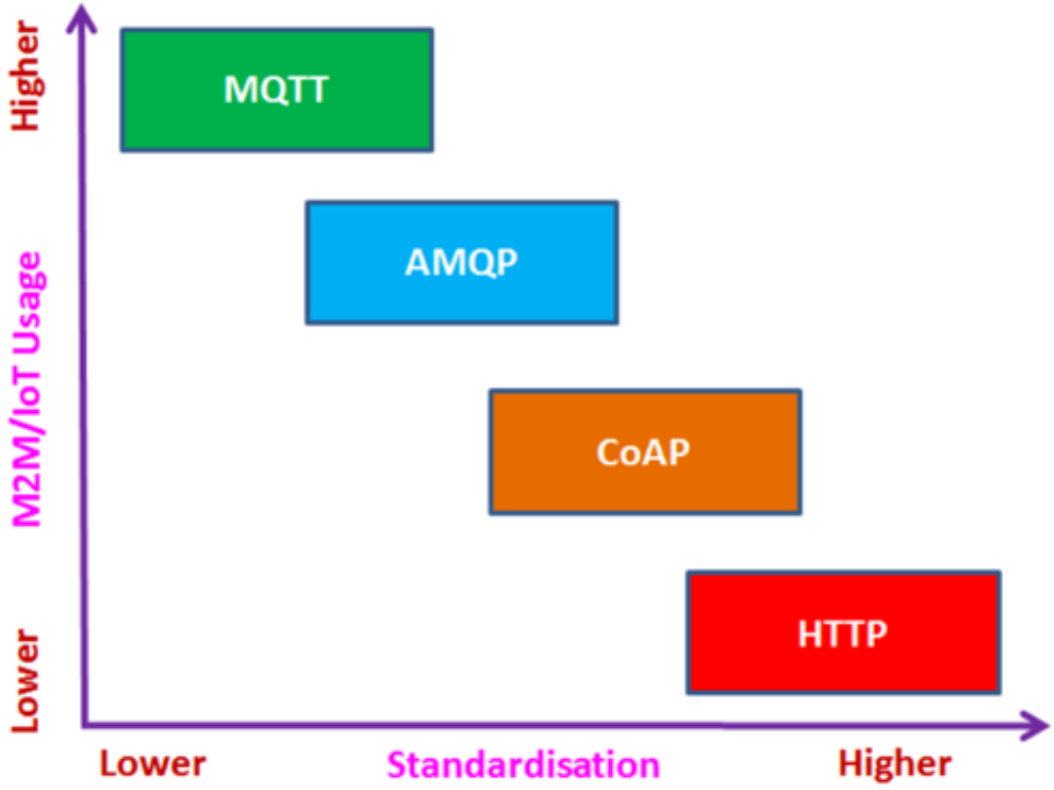
AMQP has the **highest level of support for security** with different approaches. For enhancing the security of IoT systems across multiple clouds, these messaging protocols can be combined with identity and access management protocols.



M2M/IoT Usage vs. Standardization

MQTT has been **employed by the large number of organisations** such as IBM, Facebook, Cisco, Red Hat, Amazon Web Services.

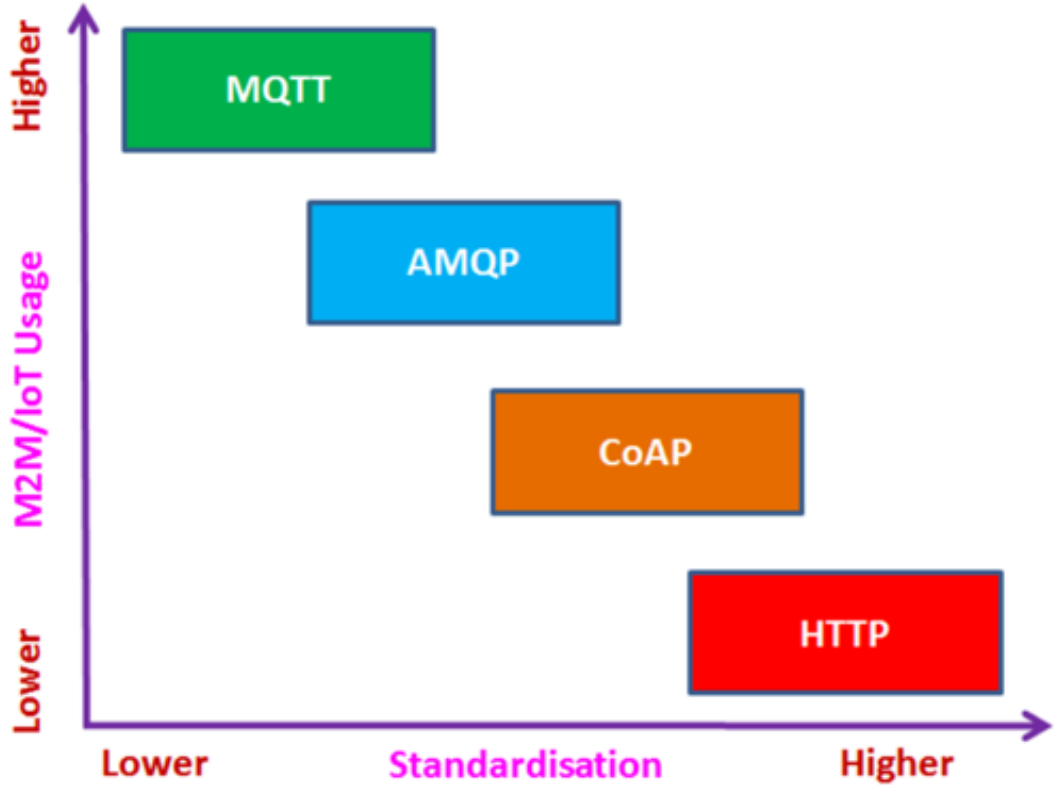
MQTT is emerging as a **de facto protocol** for the IoT and hosted by OASIS open standards consortium and Eclipse Foundation.



M2M/IoT Usage vs. Standardization

AMQP is the **most successful IoT protocol** that has been employed in the worlds biggest projects such as Oceanography's monitoring of the Mid-Atlantic Ridge, NASA's Nebula Cloud Computing and Indias Aadhar Project.

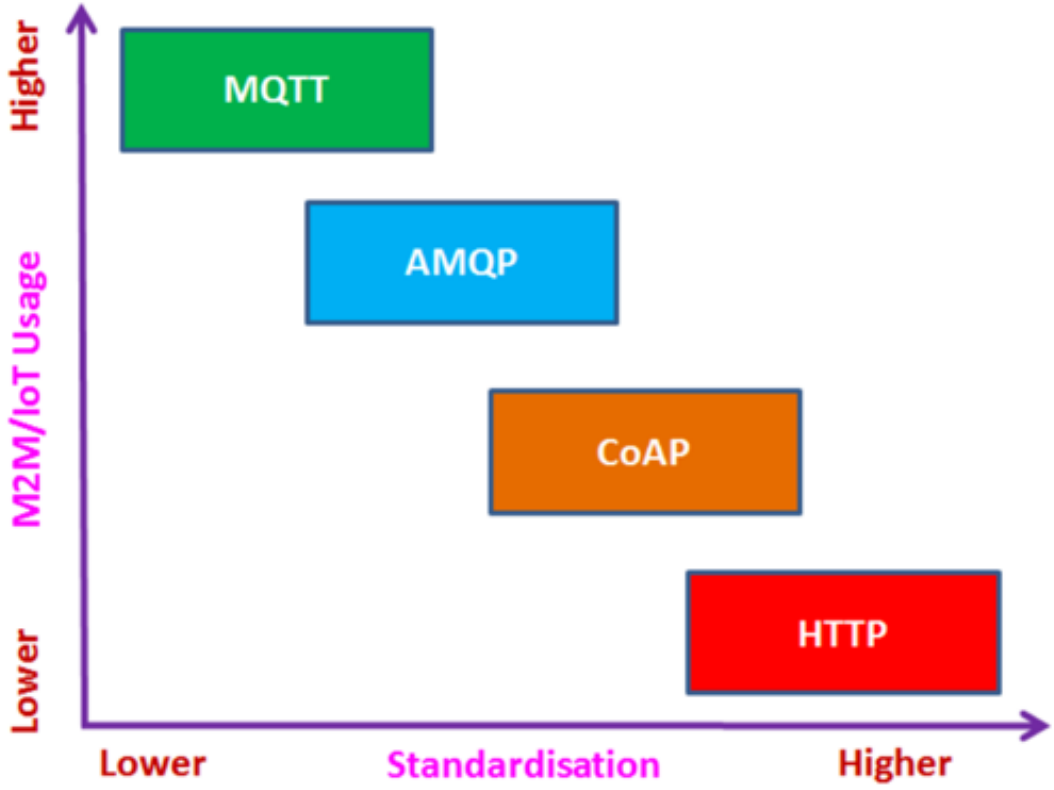
AMQP is an OASIS adopted international standard ISO/IEC 19464:2014.



M2M/IoT Usage vs. Standardization

CoAP has been swiftly gaining momentum and supported by many large companies such as Cisco, Contiki, Erika and IoTivity.

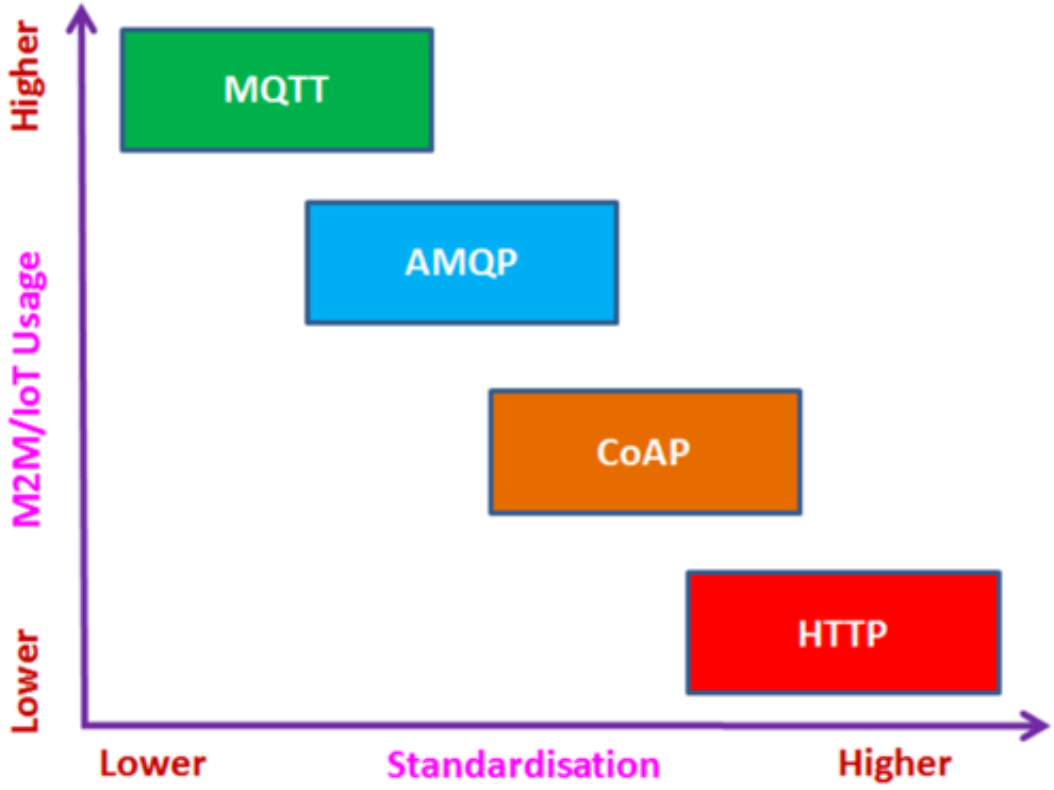
CoAP is an IETF standard specially designed to integrate the IoT and Web and supported by Eclipse Foundation.



M2M/IoT Usage vs. Standardization

HTTP is an IETF and W3C standard and already **established as a global standard for the Web**. However, it is mostly not suitable and used in the IoT industry.

The usage of HTTP in the IoT is limited due to its heavyweight size and slow performance.



M2M/IoT Usage vs. Standardization

A distributed IoT system consists of different components from resource-constrained devices (i.e. IoT devices) **up to more performant devices** (e.g. servers and cloud systems)

An IoT system can exploit more protocols, according to the original system's requirements

References

- https://www.broadbandsearch.net/blog/who-invented-the-internet-full-history
- https://www.iottechtrends.com/history-of-iot/
- Naik, Nitin. "Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP." *2017 IEEE international systems engineering symposium (ISSE)*. IEEE, 2017.
- Wee Keat Chin, “Distributed messaging with AMQP”