



CME466 - Design of an Advanced Digital System

IoT Protocols

Instructor: Prof. Khan A Wahid
TA: Omid Yaghoobian
Winter 2023



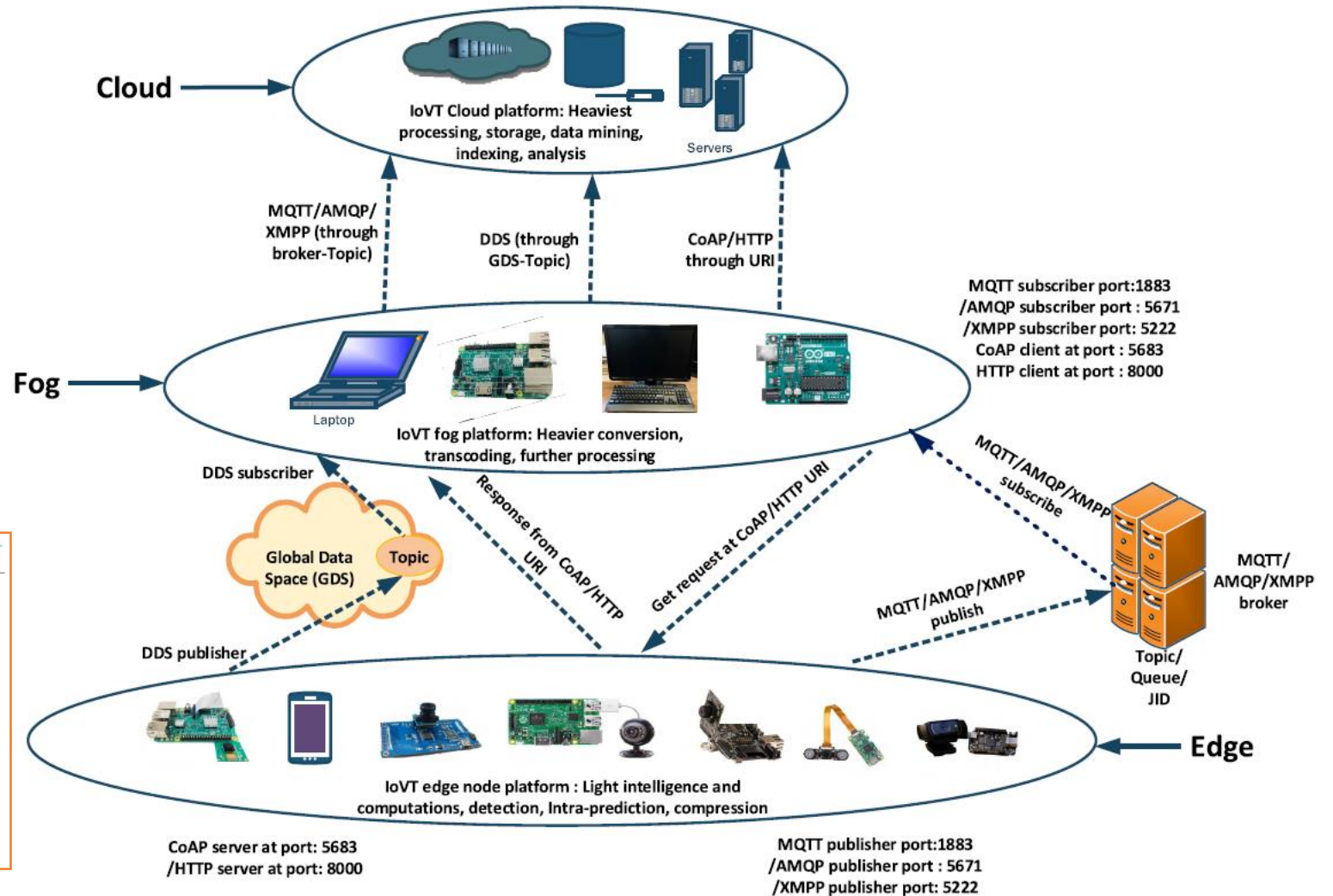
Copyright notice

- These slides are intended to be used in CME466 course only.
- The materials presented in this entire document are protected by copyright laws.
- You shall not reuse, publish, or distribute any of these slides without obtaining permission from the presenter and individual copyright owners.



General Protocol Requirements in IoT

- Requires a real-time, event-driven model
- Publishing information **one-to-many**
- Listening for events as they happen
- Sending **small packages of data** from small devices
- Reliably publishing data **over unreliable networks**



IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 24, 2019, accepted March 11, 2019, date of publication March 26, 2019, date of current version April 12, 2019.
Digital Object Identifier 10.1109/ACCESS.2019.2907525

Choice of Application Layer Protocols for Next Generation Video Surveillance Using Internet of Video Things

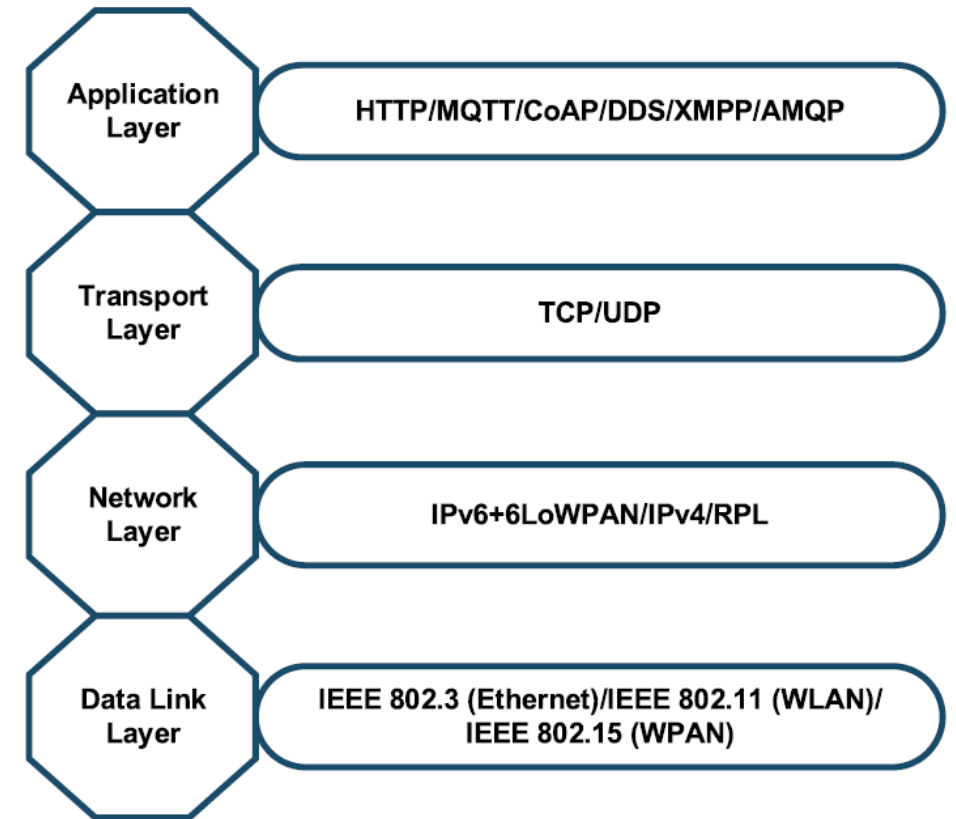
TANIN SULTANA¹ AND KHAN A. WAHID², (Senior Member, IEEE)
Department of Electrical and Computer Engineering, University of Saskatchewan, Saskatoon, SK S7N 5A9, Canada
Corresponding author: Tanin Sultana (tanin.sultana@usask.ca)

This work was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC), and in part by the Canada First Research Excellence Fund (CFREF).

Communication architecture

Application layer protocol

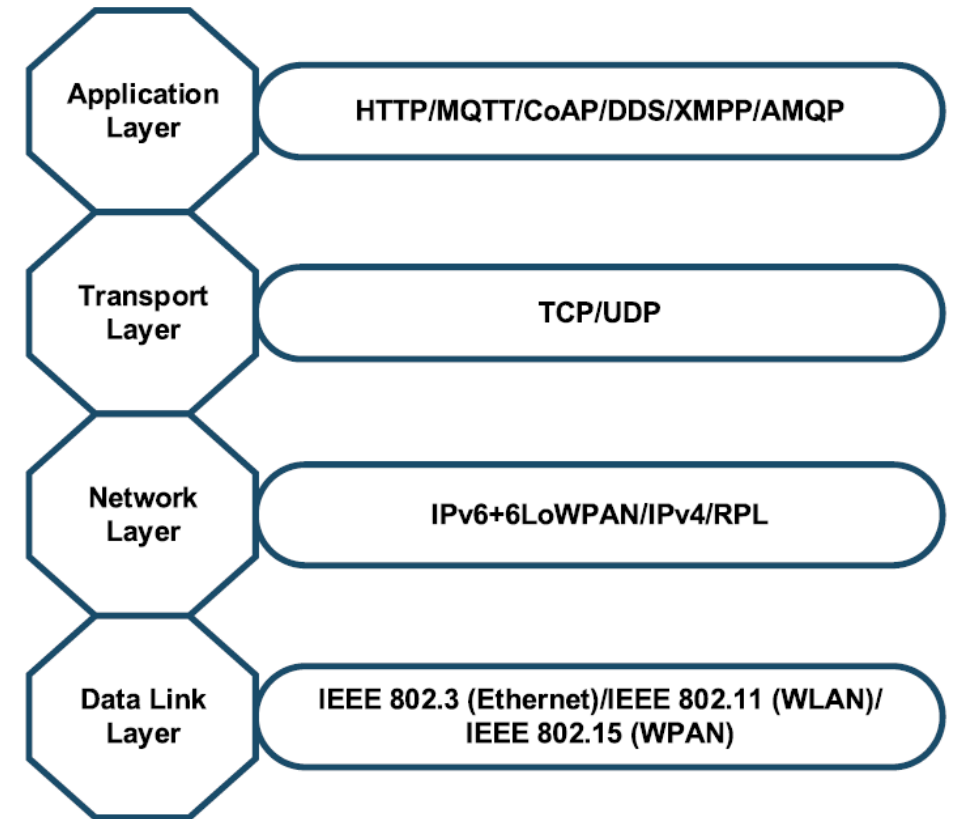
- Protocol is like a language. Not matter what language you speak, it is the same message transferred (using transport layer)
- The application layer provides services and ensures effective communication between application programs in a network
- Based on the application developers' perspective, the design alternatives for developing IoT-based real-time applications are communication protocols, message encoding format, and the Web platform



Communication architecture

Transport layer

- It breaks the message/data into segments so that it can be transported
- The protocols of this layer provide end-to-end communication services for applications.
- Transmission Control Protocol (TCP), User Datagram Protocol (UDP)
 - TCP is a connection-oriented protocol (handshake)
 - UDP is a connectionless protocol (lossy)
- TCP is comparatively slower than UDP, but more reliable than UDP
- UDP is faster, simpler, and efficient protocol, however, retransmission of lost data packets is only possible with TCP.
- UDP is beneficial in time-sensitive communications



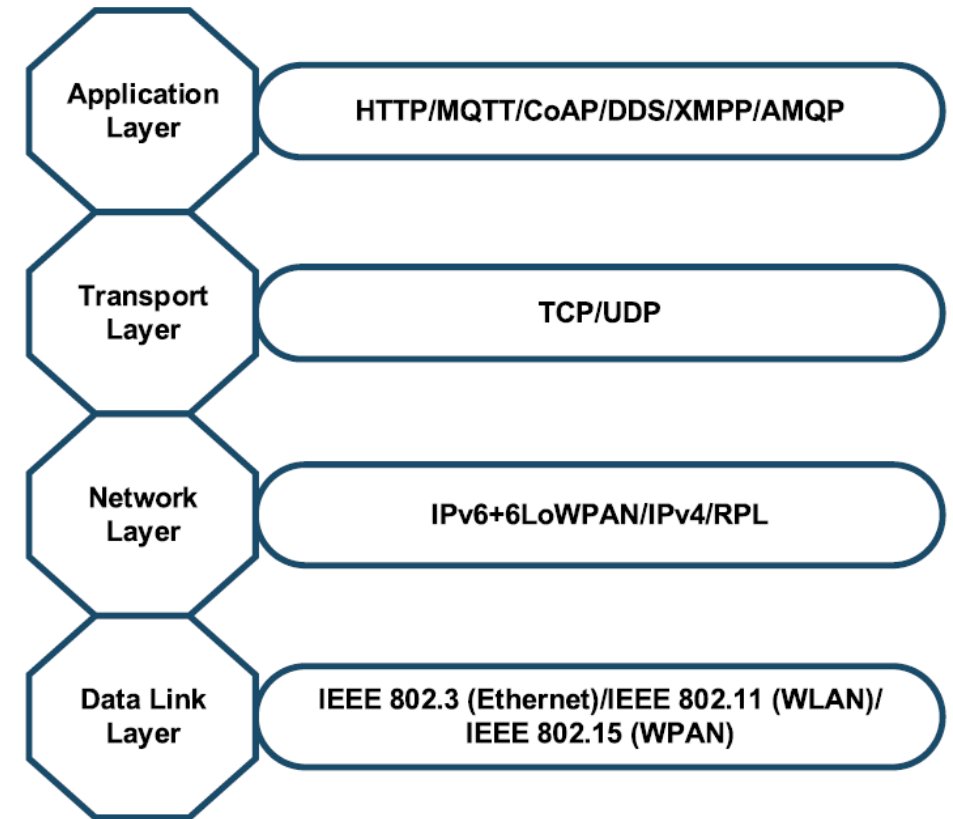
Communication architecture

Network layer/ Internet

- IPv6 Routing Protocol for Low-power and Lossy Networks (RPL) is another network layer convention which provides the facility for devices to perform point-to-point or multipoint communication
- 6LoWPAN provides the necessary adaption between IPv6 and IEEE 802.15.4

Data link layer/ Physical

- Physical devices, like IoT visual sensors or cameras, will work. IEEE physical layer standards,
- IEEE 802.3 (Ethernet),
- IEEE 802.11 (Wireless Local Area Network/WLAN),
- IEEE 802.15.4 (Wireless Personal Area Network/WPAN).
- A popular communication protocol in this layer for standard IoT-based systems is IEEE 802.15.4



IoT Protocols

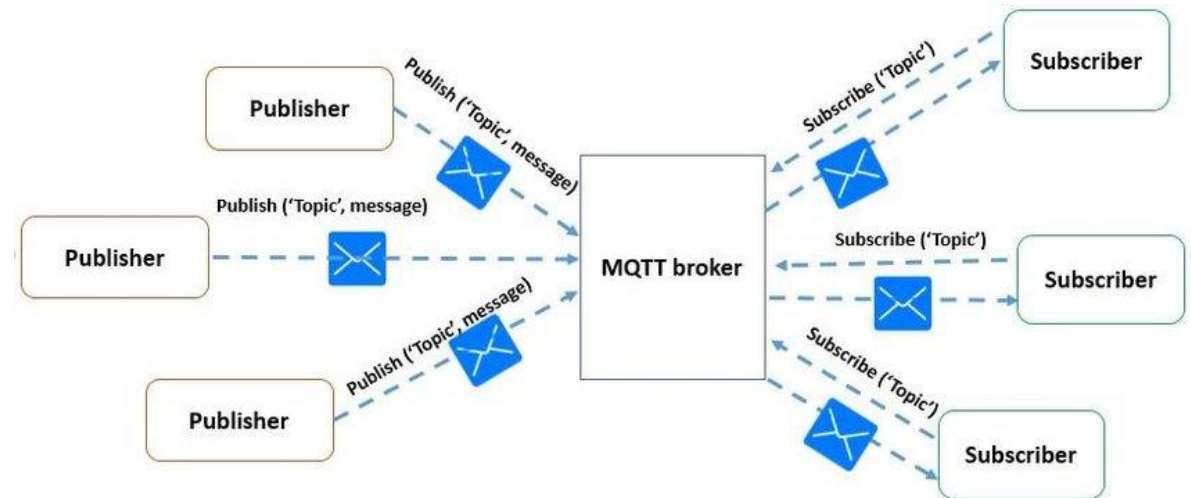
- **Adafruit Video: All the Internet of Things - Episode Two: Protocols**
 - <https://learn.adafruit.com/allthethings-protocols/introduction>

HTTP (HYPER TEXT TRANSFER PROTOCOL)

- HTTP is a globally accepted web messaging standard which offers several features such as persistent connections, request pipelining, and chunked transfer encoding
- HTTP supports request/response RESTful Web architecture (GET, PUT, POST, DELETE)
- HTTP is:
 - asynchronous protocol, this means the client waits for the server to send the data
 - a text-based protocol that does not define the size of header and message payloads
 - a one way street, **only the client makes the request**
 - one to one protocol, one client request at a time
 - **data heavy, filled with headers and rules**
 - uses the Universal Resource Identifier (URI) instead of topics (like MQTT)

MQTT (MESSAGE QUEUING TELEMETRY TRANSPORT)

- MQTT, designed in 1999, is a lightweight machine to machine (M2M) communication protocol that supports the **publish/subscribe architecture** with minimal bandwidth requirements, power consumption, and message data overhead
- An MQTT client **publishes messages to a broker through an address known as Topic**. Another client can receive the messages by subscribing to that Topic. Clients can subscribe to multiple topics.
- MQTT publish/subscribe protocol provides a scalable and reliable way to connect devices over the Internet. Today, MQTT is used by many companies to connect millions of devices to the Internet
- <https://mosquitto.org/>
- www.mqtt.org/



MQTT (MESSAGE QUEUING TELEMETRY TRANSPORT)

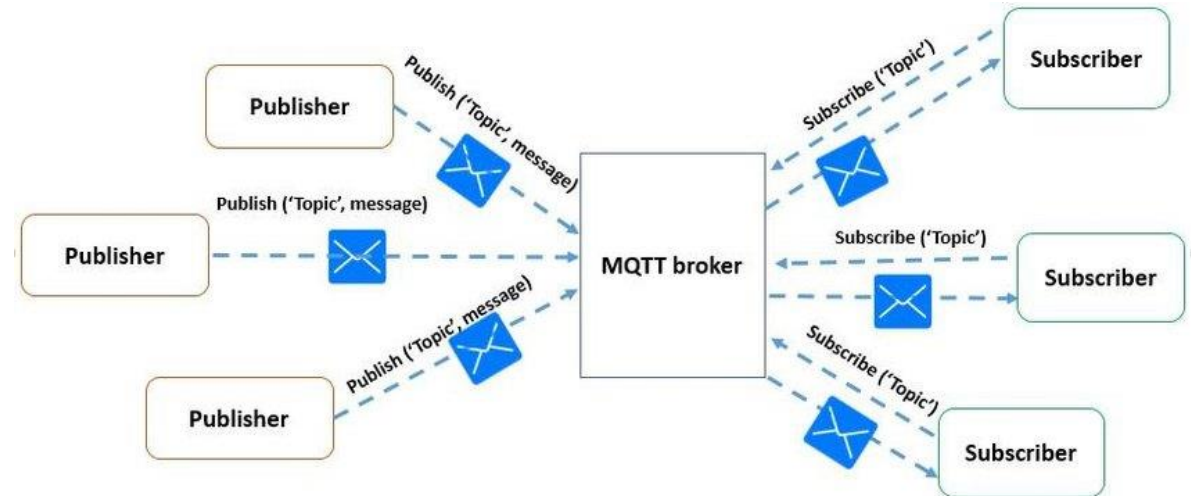
MQTT Clients

- MQTT clients publish a message to an MQTT broker and other MQTT clients subscribe to messages they want to receive
- Implementation of MQTT clients typically requires a minimal footprint, so it is well suited for deployment on small constrained devices and are very efficient in their bandwidth requirements

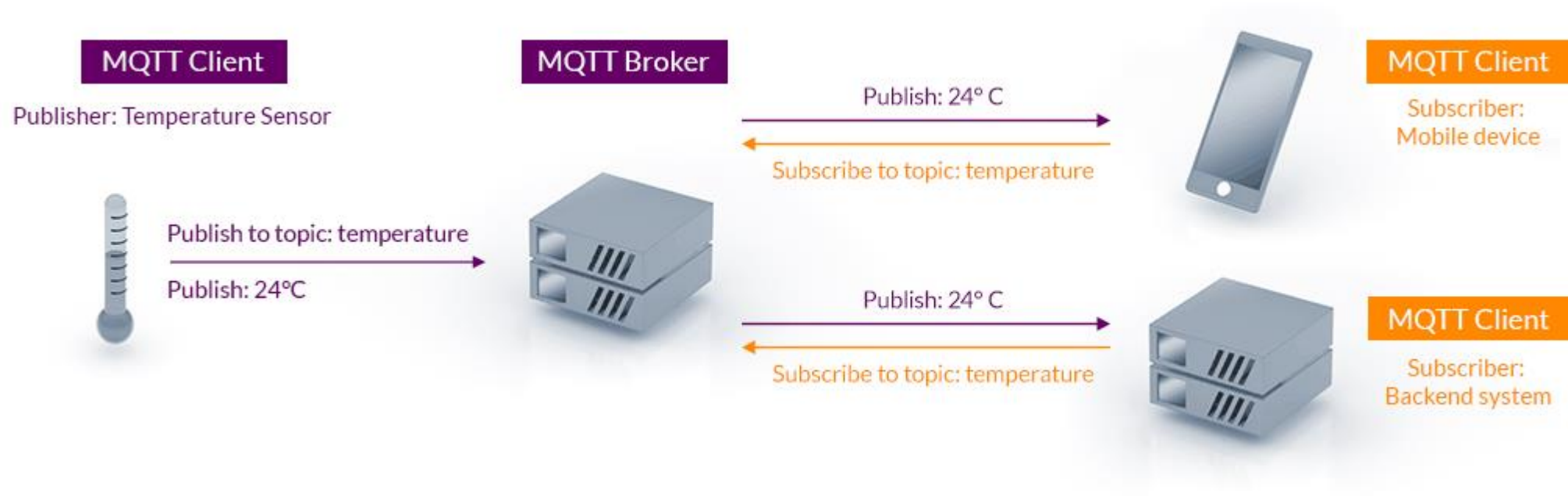


MQTT Broker

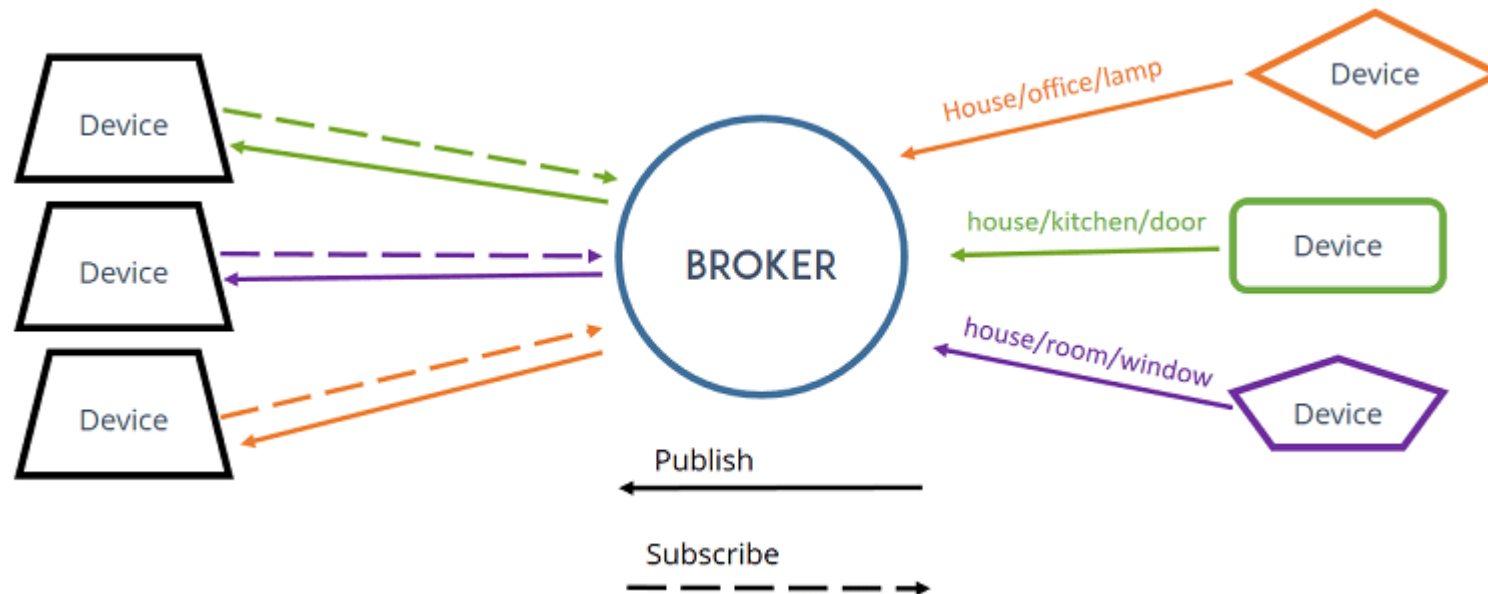
- MQTT brokers receive published messages and dispatch the message to the subscribing MQTT clients
- An MQTT message contains a message topic that MQTT clients subscribe to and MQTT brokers use these subscription lists for determining the MQTT clients to receive the message



MQTT: Publisher/ Subscriber example



MQTT: Broker tasks



Broker tasks:

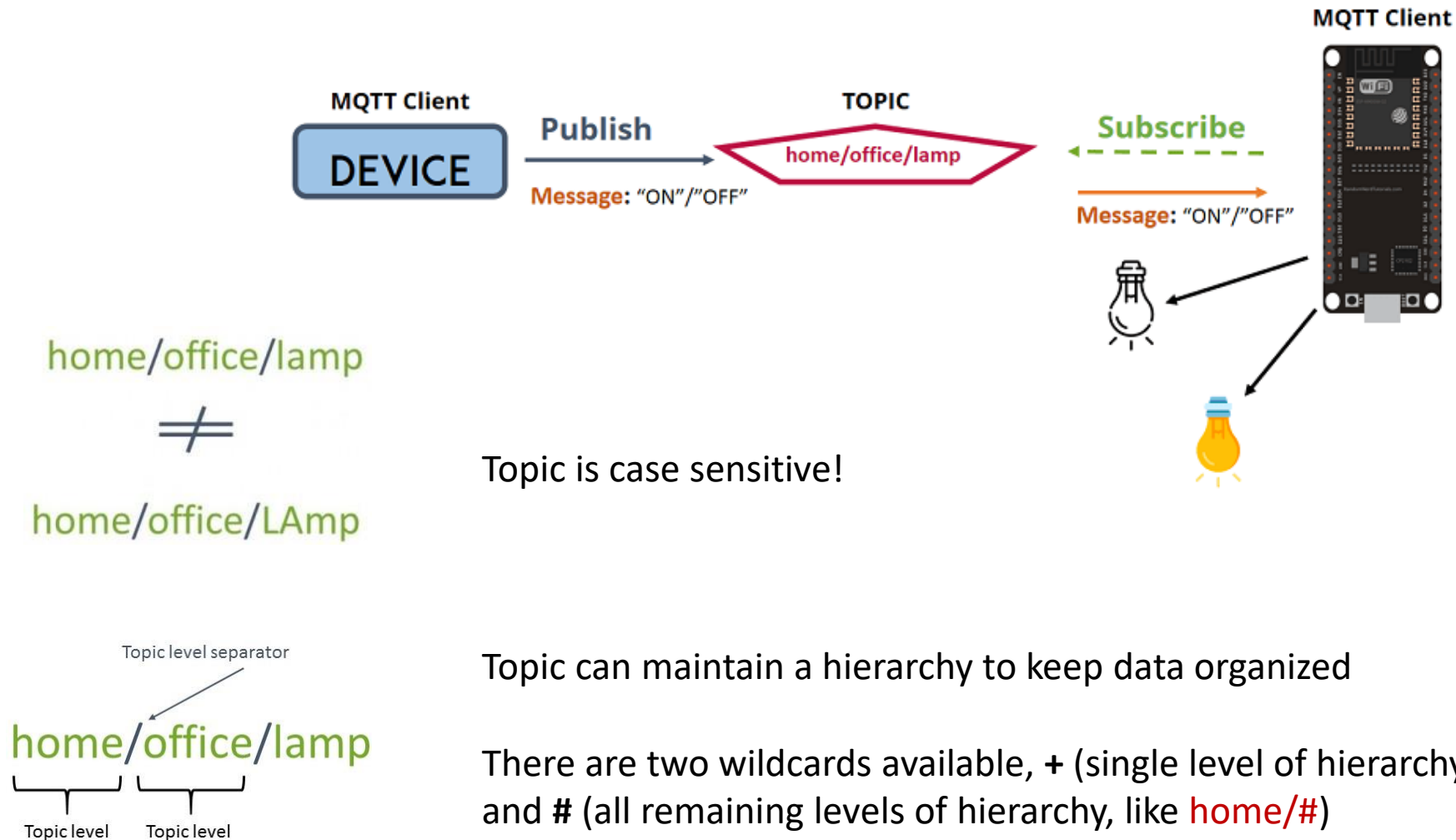
Receiving: It receive all messages

Filtering: The filtering activity of the broker makes it possible to control which client/ subscriber receives which message

Decide: According to the topics it decides, which message belong to which device

Publishing: Send (publish) the messages to subscribed clients

MQTT: TOPIC and Message



COAP (CONSTRAINED APPLICATION PROTOCOL)

- CoAP which is another **extremely lightweight** M2M protocol was standardized in 2014.
- It provides a request-reply interaction model like REST (Representational State Transfer) to constrained devices and environments.
- CoAP uses mostly UDP
- **It supports both request/response and resource/observe architectures.** Like HTTP, CoAP uses URI.
- The server sends data through the URI, and the client receives data from a specific resource indicated by that URI [14]. Its payload size should be small, and the maximum size of the payload depends on the web server or the programming technology.
- CoAP uses confirmable (CON) and non-confirmable (NON) messages to provide two different levels of QoS. The receiver acknowledges CON messages with an ACK packet, unlike non-confirmable messages that do not need ACK.

<https://radiocrafts.com/technologies/coap-constrained-application-protocol/>

AMQP (ADVANCED MESSAGE QUEUING PROTOCOL)

- AMQP, developed in 2003, is a lightweight M2M protocol designed for reliability, security, provisioning, and interoperability.
- This protocol supports both request/response and publish/ subscribe architectures.
- In AMQP (version 0.9.1) clients publish messages to a broker. The broker stores the messages in queues as long as the subscribers to those queues have not received the messages.
- Depending on the exchange type, there are four possible ways of routing messages between publishers and consumers: directly, in fanout form, by topic, or based on headers.
- <https://www.amqp.org/>
- <https://www.cloudamqp.com>
- <https://www.rabbitmq.com/>



DDS (DATA DISTRIBUTION SERVICE)

- DDS uses a data-centric publish-subscribe (DCPS) model for real-time M2M communications and was developed by Object Management Group (OMG) in 2004.
- Applications use Data-Writer of the given data type to publish data objects to a topic over Publishers component. Data-Reader of the given data type receives data objects over the Subscriber component.
- The publishers discover the subscribers dynamically, using matching topics and data types. In contrast to other publish-subscribe protocols like MQTT or AMQP, DDS relies upon broker less architecture and uses multicasting.
- It supports 23 QoS policies providing various communication criteria like reliability, priority, urgency, and security.
- DDS species the use of multicast UDP within LAN (Local Area Network) and TCP transport for communication over WAN (Wide Area Network).
- The two chief stakeholders in the DDS market proposed the two main implementations of DDS. One is OpenSplice by PrismTech, and the other is Connex DDS by Real-Time Innovation (RTI).
- <https://www.dds-foundation.org/>



XMPP (EXTENSIBLE MESSAGING AND PRESENCE PROTOCOL)

- XMPP is an Instant Messaging (IM) standard used for real-time messaging with XML (eXtensible Markup Language) streaming technology at its core.
- It connects a client to an XMPP server using a stream of XML messages called stanzas, and uses TCP as the default transport protocol and TLS/SSL for security.
- XMPP serves various purposes including publish/subscribe messaging, multi-user chatting, and sensor data exchange.
- <https://xmpp.org/software/servers.html>



Comparison among application layer protocols

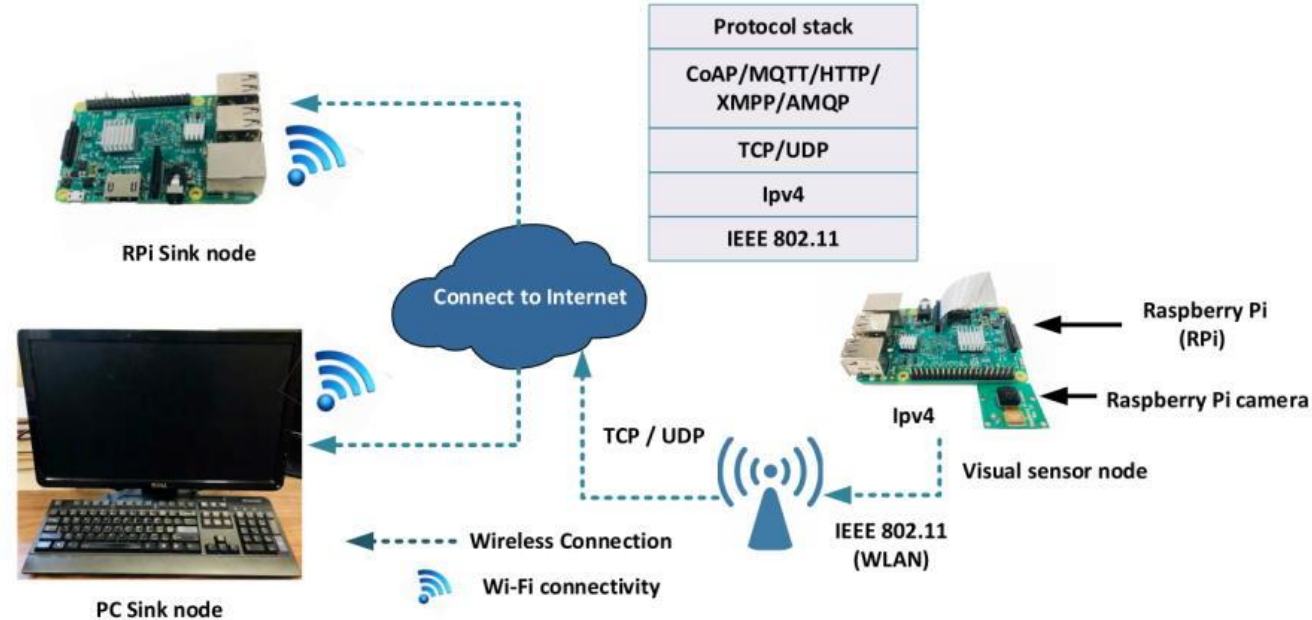
Parameter	MQTT ^a	HTTP ^b	CoAP ^c	AMQP ^d	DDS ^e	XMPP ^f
Header size	2 Byte	No limit/ Undefined	4 Byte	8 Byte	Typically, large overhead, varies with situation and configuration	Very large with no limit (Varies with data size)
Maximum payload per packet	256 MB	Defined by web server/ programming technology	64 KB (for block-wise UDP transfer) 1280 Bytes (for IPv6 to avoid IP fragmentation) 60-80 Bytes (to avoid adaptation layer fragmentation for 6LoWPAN)	Negotiable and undefined (depends on the broker/ server and programming technology)	Large data transmission is supported, it can be limited by transport	10000 Bytes (Max. stanza size depends on the server configuration)
Encoding format	Binary/Text	Binary/Text/file	Binary/Text	Binary/Text	Efficient Binary	Text (XML) based and EXI (Efficient binary)
Transport protocol	TCP	TCP	UDP	TCP	UDP, TCP	TCP
Quality of service (QoS)	3 QoS: QoS 0 – at most once QoS 1 – at least once QoS 2 – at exactly once	Limited	CON – must be acknowledged NON – acknowledgement is not required	Settle – at most once Unsettle – at least once	23 QoS	Till no QoS is supported
Security	TLS/SSL	TLS/SSL	DTLS	TLS/SSL	TLS/SSL/DTLS	TLS/SSL
Resource locator	Topic name	URI	URI	Exchange-queue	Topic	JID (Jabber Identifier)
Default port no.	1883	80/8000	5683	5672	-	5222
RESTful methods	Absent	present	Present	Absent	Absent	Absent

a. <http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html>
b. <https://tools.ietf.org/html/rfc2616>

c. <https://tools.ietf.org/html/rfc7252>
d. <https://www.rabbitmq.com/resources/specs/amqp0-9-1.pdf>

e. <https://www.omg.org/spec/DDS/1.4/>
f. <https://tools.ietf.org/html/rfc6120>

Performance Analysis: Experimental Setup



- According to the generic architecture earlier, several test video surveillance applications were set up to transmit real-time images and video through the IoVT network
- In this case, an IoVT source node was used to capture images and video and send them to the IoVT sink nodes, utilizing the application layer protocols
- The source node will act either as an edge or a fog, while the sink node serves either as the fog or the cloud, depending on the role of the source node.

Important parameters

LATENCY

- Network latency is defined as the **amount of time it takes for a packet to reach a destination device from its source device**
- Latency can be measured in several ways, including one way and round trip
- Synchronizing the clocks at both ends and then subtracting the packet transmission time from the packet arrival time at its destination gives the latency in one-way measurement

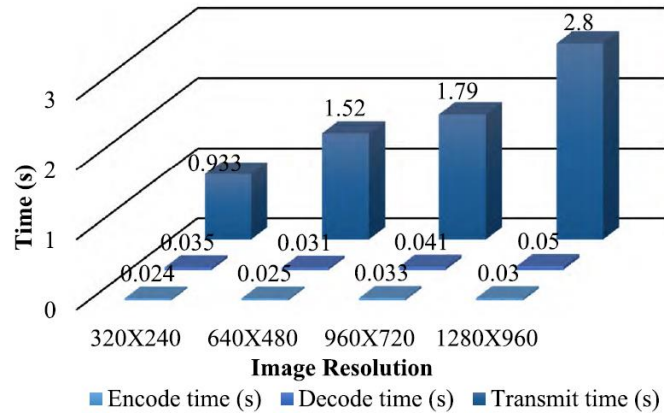
THROUGHPUT

- Throughput is **the quantity of data being sent or received** per unit time
- The throughput was calculated using only the payload-containing packets per second (packets/s)

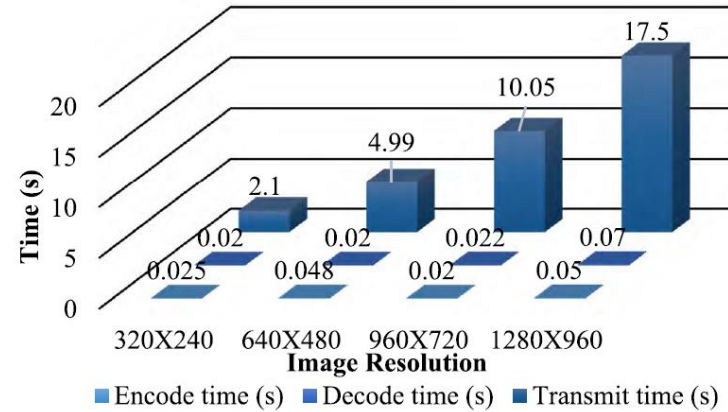
PACKET LOSS

- Packet loss is the number of packets lost per 100 packets sent from a source
- Wireshark was used to capture the packet information

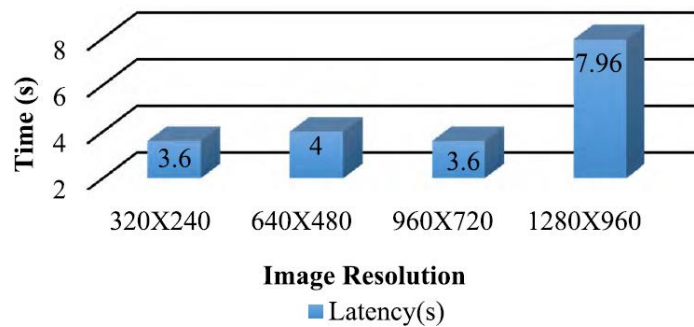
Latency comparison



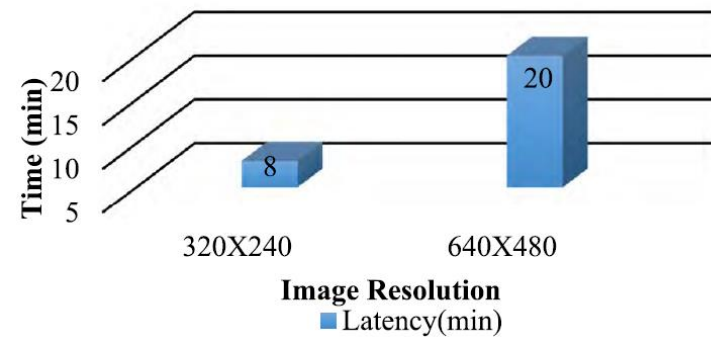
MQTT



AMQP



HTTP

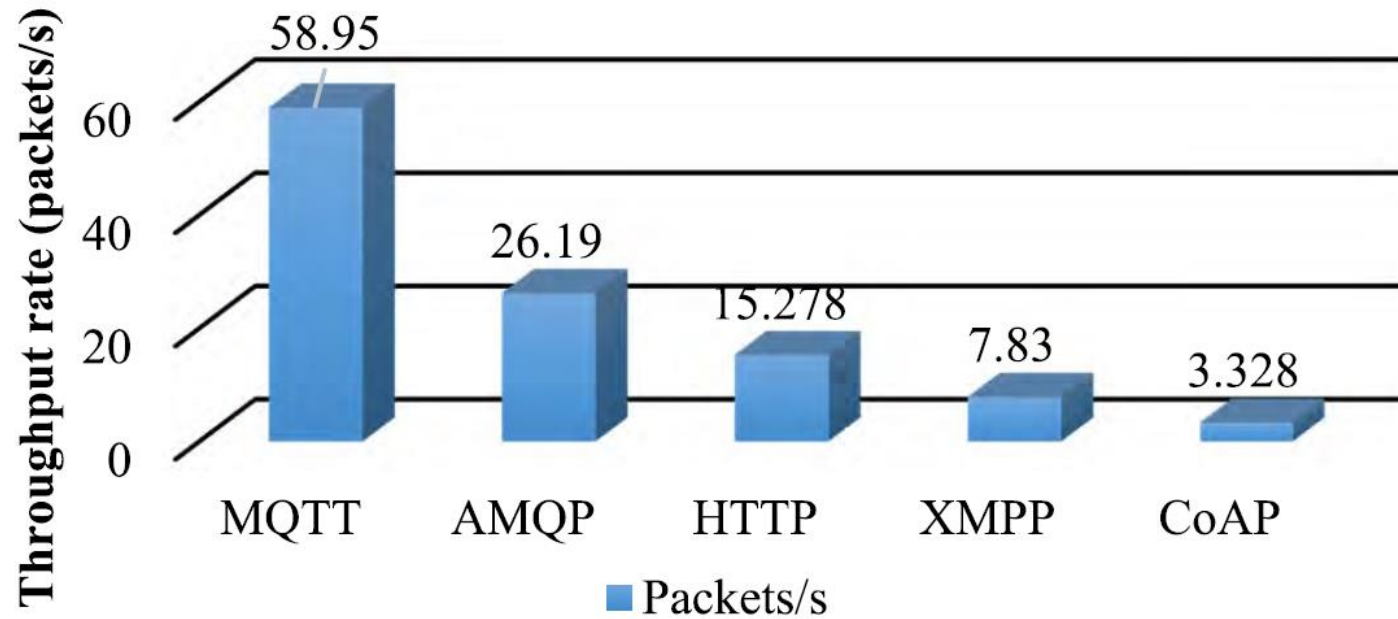


CoAP

Comparison of throughput and latency per packet

Protocols	MQTT	AMQP	HTTP	XMPP	CoAP
Packet size (Bytes)	1514	1514	1514	1514	118
Max. Payload size (Bytes)	1448	1448	1448	1448	64
Total packets	55	56	55	59	1225
Packet processing time (s)	.0009	.0005	.0014	0.0019	.0005
Packet transmit time (s)	.017	.038	.063	0.125	.2986
Packet decoding time (s)	.0005	.00093	.0009	.0009	.0009
Throughput (packets/s)	58.95	26.19	15.278	7.83	3.328
Throughput (Bytes/s)	85432	37924	22123	11338	213

Throughput rate comparison



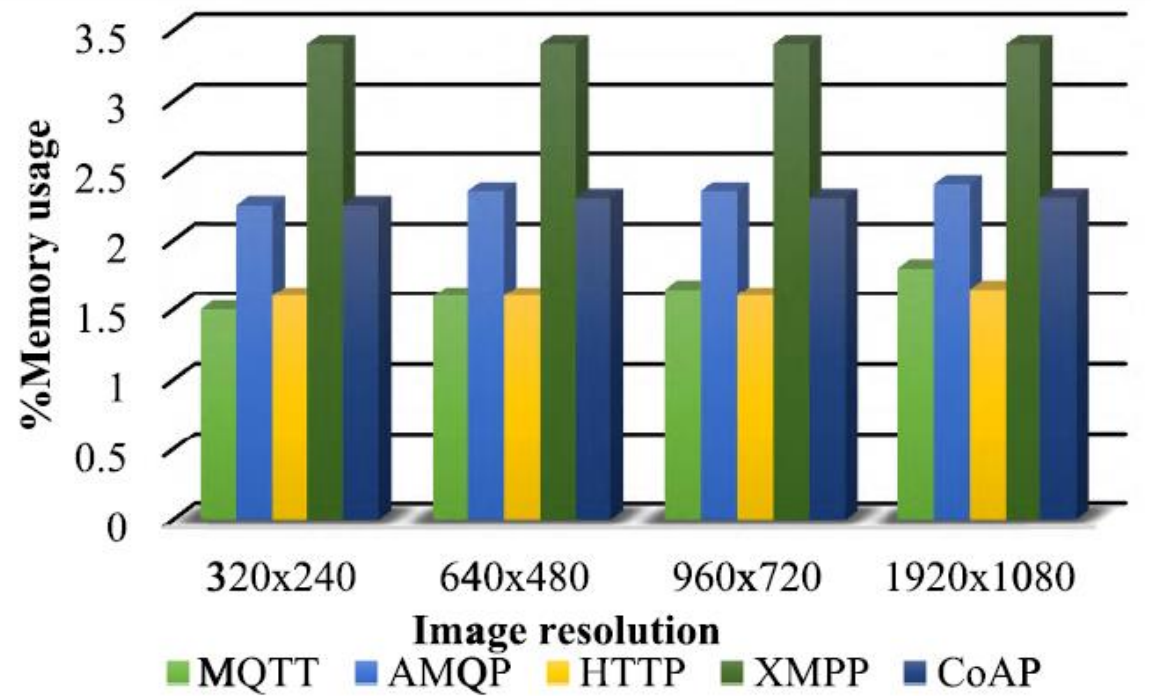
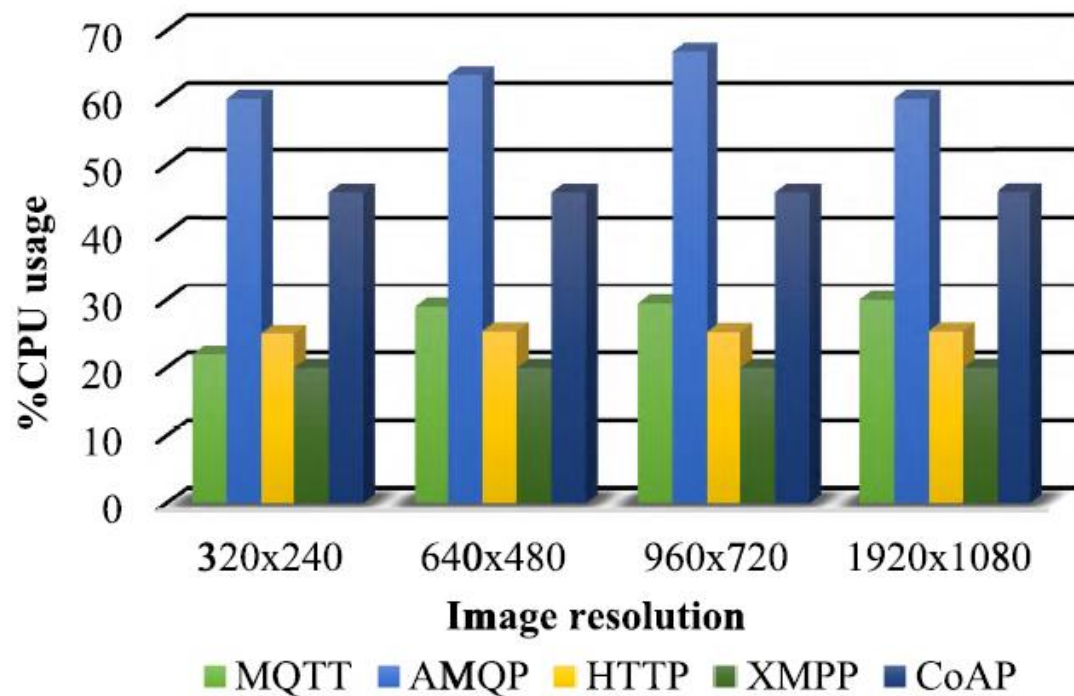
Energy consumption

Protocols	MQTT	AMQP	HTTP	XMPP (In-band)	CoAP
Max Drawn ^a Current (A)	0.19	0.22	0.24	0.26	0.24
Power ^b (W)	0.95	1.1	1.2	1.3	1.2
Total latency(s)	0.992	2.145	3.6	7.53	480
Energy ^b (J=W*s)	0.942	2.3595	4.32	9.789	576
Energy per Byte (μJ)	12	30	55	124	7343
Energy per packet (mJ)	16.15	41.8	75.6	162.5	358.32


a. RPi-B Normal operating condition Current is 0.27 A at 5V.

b. Power (W) = Voltage (V) × Current (A), Energy (J) = Power (W) × Latency (s)

CPU and Memory usage



Performance summary

Performance:	Highest					Lowest
Latency	CoAP	XMPP	HTTP	AMQP	MQTT	
Throughput	MQTT	AMQP	HTTP	XMPP	CoAP	
% protocol Overhead	CoAP	XMPP	AMQP	HTTP	MQTT	
%CPU usage	AMQP	CoAP	MQTT	HTTP	XMPP	
%Memory usage	XMPP	AMQP	CoAP	MQTT/ HTTP	MQTT/ HTTP	
BW (bits/s) consumption	HTTP	XMPP	AMQP	MQTT	CoAP	
Energy consumption (J)	CoAP	XMPP	HTTP	AMQP	MQTT	