CT106-3-M-BIA - Building IoT Applications

CT127-3-M-ODL - BIA - Building IoT Applications

Topic 6 – IoT Application Protocols

TOPIC LEARNING OUTCOMES

At the end of this topic, you should be able to:

- Understand the concept of IoT communication patterns and their significance in enabling effective communication between IoT devices.
- Gain a comprehensive understanding of the IoT application protocols landscape and their role in enabling seamless communication between IoT devices.
- Identify and describe the key characteristics and requirements of IoT application protocols, such as low power consumption, scalability, and security.
- Compare and contrast different IoT application protocols, including HTTP, CoAP, MQTT, and AMQP, in terms of their architecture, features, and suitability for specific IoT applications.

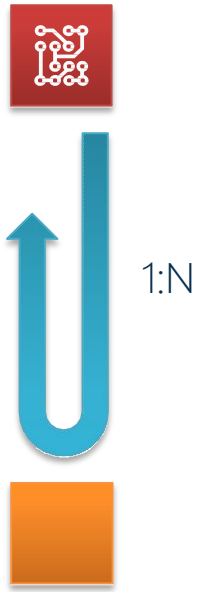
Contents & Structure

- IoT communication patterns
- IoT Application Protocols landscape
  - Introduction
  - Architecture
  - Features
- HTTP
- CoAP
- MQTT
- AMQP

IoT communication patterns

Telemetry

Information flows from device to other systems for conveying status changes in the device



Inquiries

Requests from devices looking to gather required information or asking to initiate activities

Commands

Commands from other systems to a device or a group of devices to perform specific activities

Notifications

Information flows from other systems to a device or a group for conveying status changes in the world

Why do we need IoT application protocols?

- Communication challenges faced in an IoT project:
  - Device-to-device communication on a local network
  - Device-to-device communication over the Internet
  - Device-to-server communication over the Internet
  - Server-to-server communication over the Internet for storing data
- The challenge is resolved by IoT application protocols like HTTP, MQTT, CoAP etc.

IoT protocols landscape

AMQP

HTTP

MQTT

CoAP

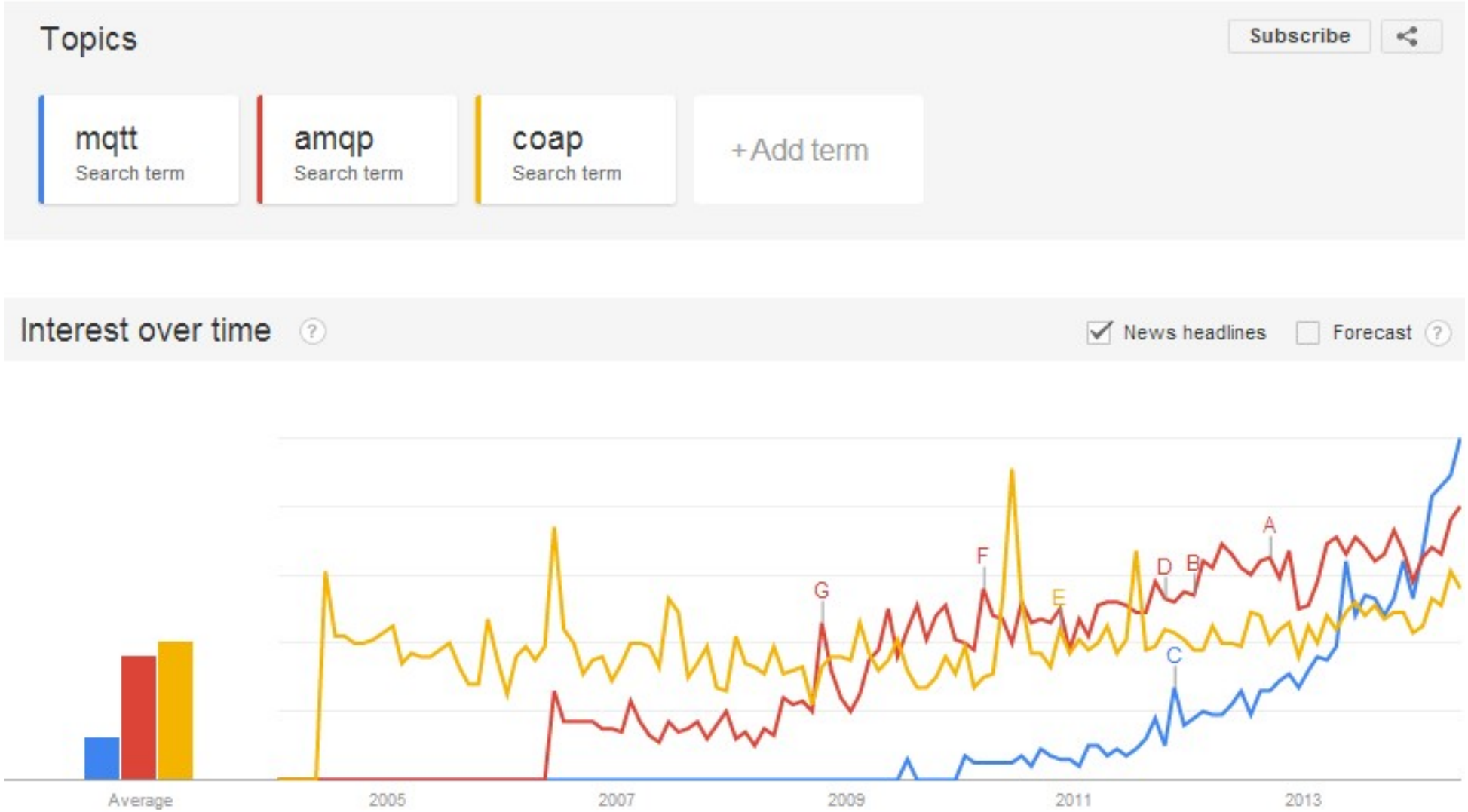
XMPP

DDS

STOMP

... we could continue

IoT application protocol trends



Standardization

HTTP	• IETF standard (RFC 2616 is HTTP/1.1)
CoAP	• IETF draft 18 (December 2013)
MQTT	• OASIS and ISO/IEC 20922 (MQTT 3.1.1)
AMQP	• OASIS and ISO 19464 standard (1.0)

HTTP (Hypertext Transfer Protocol)

HTTP is an application-layer protocol for transmitting hypermedia documents. In a normal web environment, HTTP is used by a web browser to retrieve web pages from a server with the GET method. In an IoT environment, a common use of HTTP is to allow devices to POST to a resource that represents the device state on the IoT service.

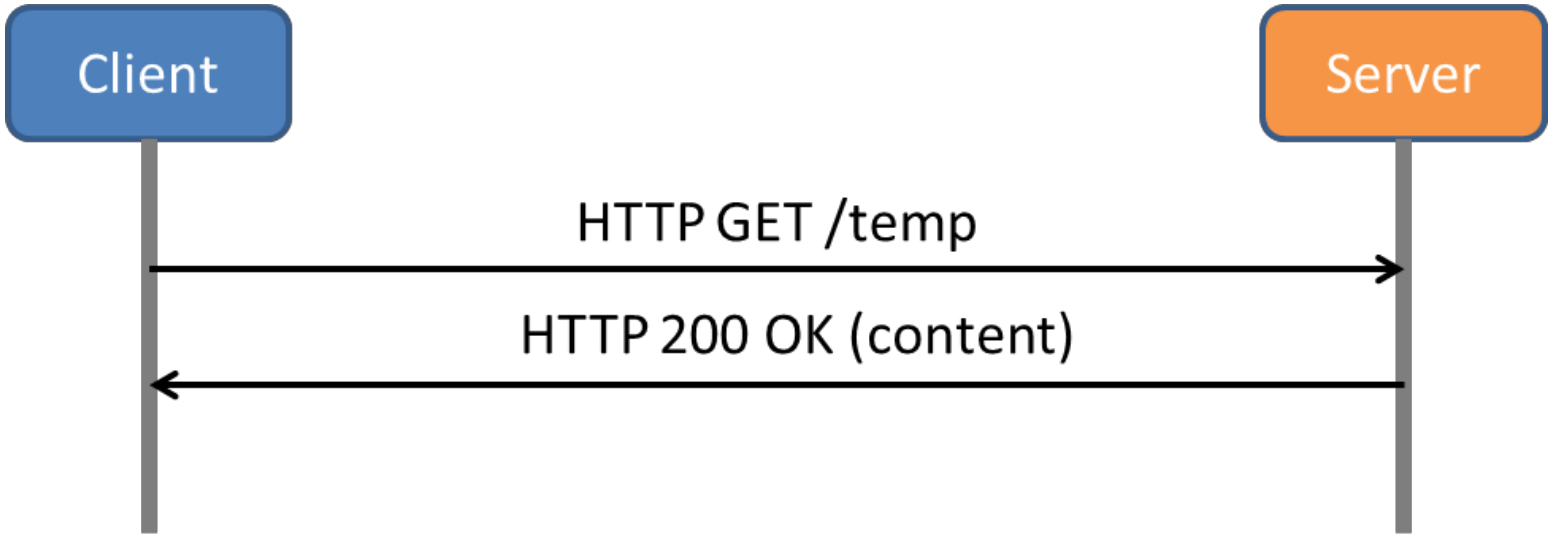
Feature Highlights

- HTTP is unidirectional
- utilize high system resources
- based on the request-response model
- MQTT message payloads are binary, whereas HTTP are textual hence binary data needs to be base64 encoded

HTTP: request/response

one to one

one to many with more requests



HTTP : push and weight

- push on client
  - (long) polling
  - WebSocket
  - forever frame
  - server sent events
- more heavyweight
  - ASCII, text-based headers, ...
  - client more complex (ASCII parser)
  - more bytes to pay on data transfer

HTTP : QoS and architecture

- no Quality of Service
- security
  - basic & digest authentication
  - over SSL/TLS
- no “messaging middleware”
- REST architecture
  - resources accessed by URIs
  - CRUD operations by HTTP methods

CoAP (Constrained Application Protocol)

CoAP is a specialized application protocol designed for constrained devices. It's designed to require low power, work across lossy networks, and can be used to connect devices to each other or other general nodes on the Internet. CoAP was designed to look like and be compatible with HTTP.

Feature Highlights

- Compatible with HTTP
- Low Power usage
- Used on constrained devices

CoAP : binary HTTP-like

HTTP-like based on UDP (no TCP)

- request/response
- packet order and retransmission into sw stack
- HTTP verbs, status codes, …
- “options” like HTTP headers but binary
- client more simple than HTTP

observer pattern available

- avoid HTTP (long) polling
- separate response/response back after a while

CoAP : QoS and architecture

Quality of Service

- «confirmable» and non «confirmable» messages

security

- DTLS (Datagram TLS)

resource discovery (CoRE link format)

CoAP node acts as a server

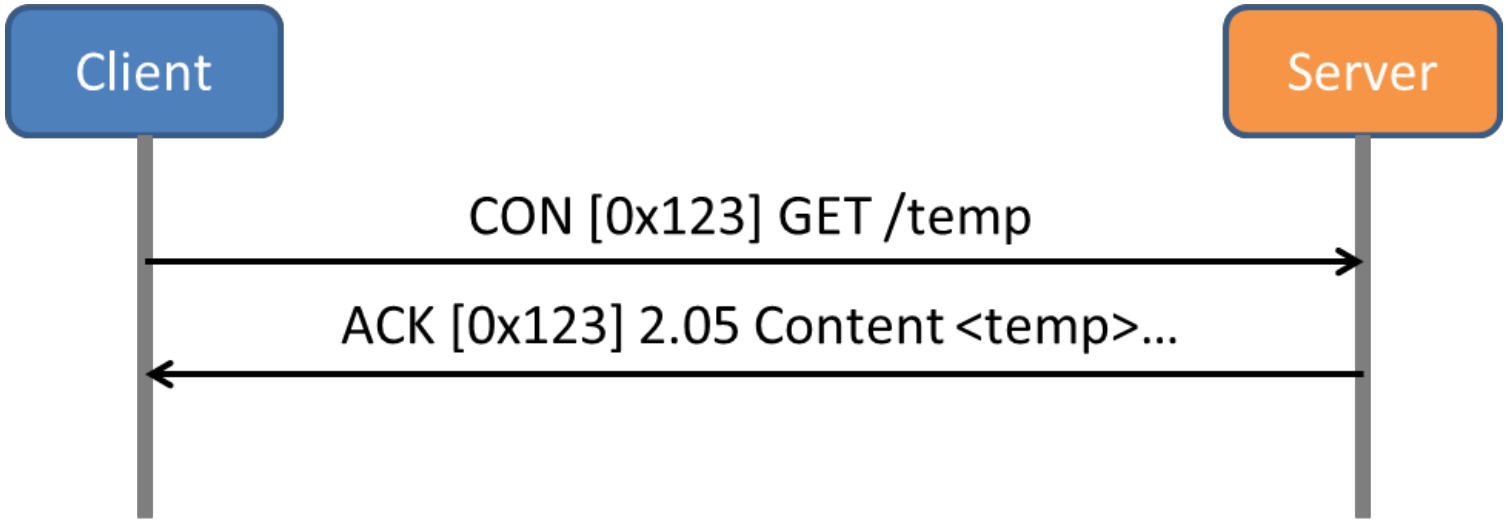
- addressing the problem (mobile roaming, NAT, …)

REST architecture

proxy CoAP – HTTP simple (with caching)

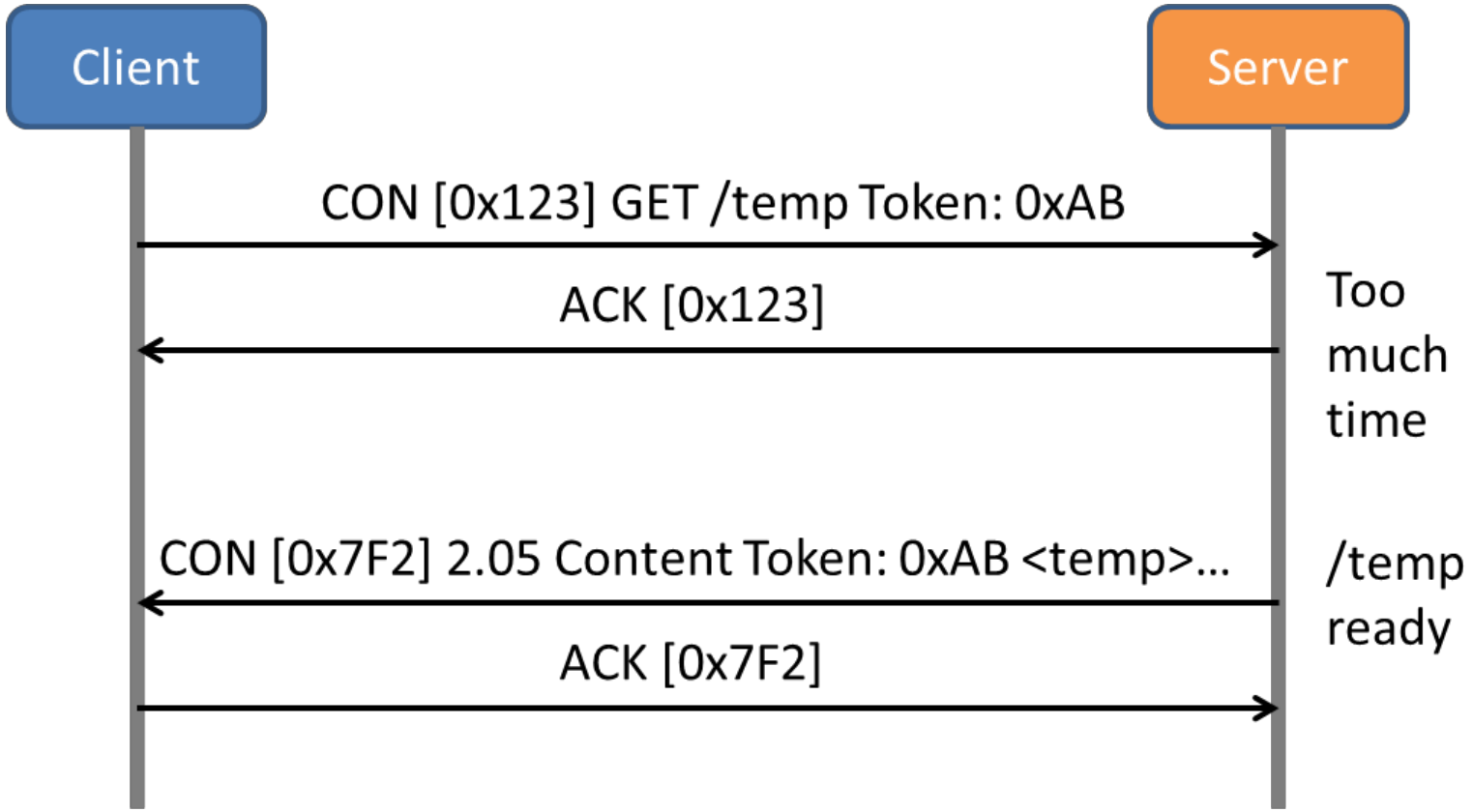
CoAP : communication patterns

confirmable request



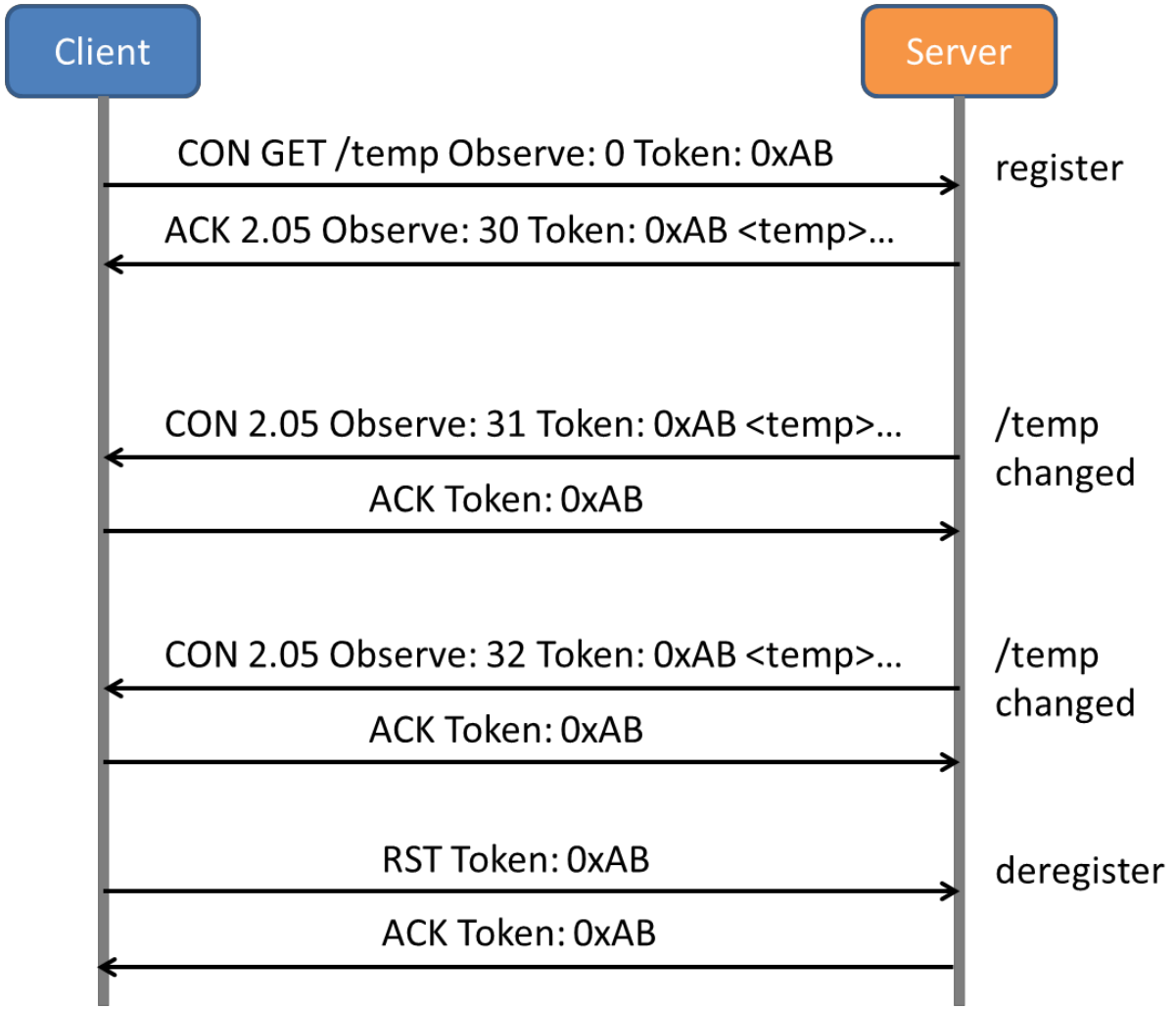
CoAP : communication patterns

separate response/response back after a while



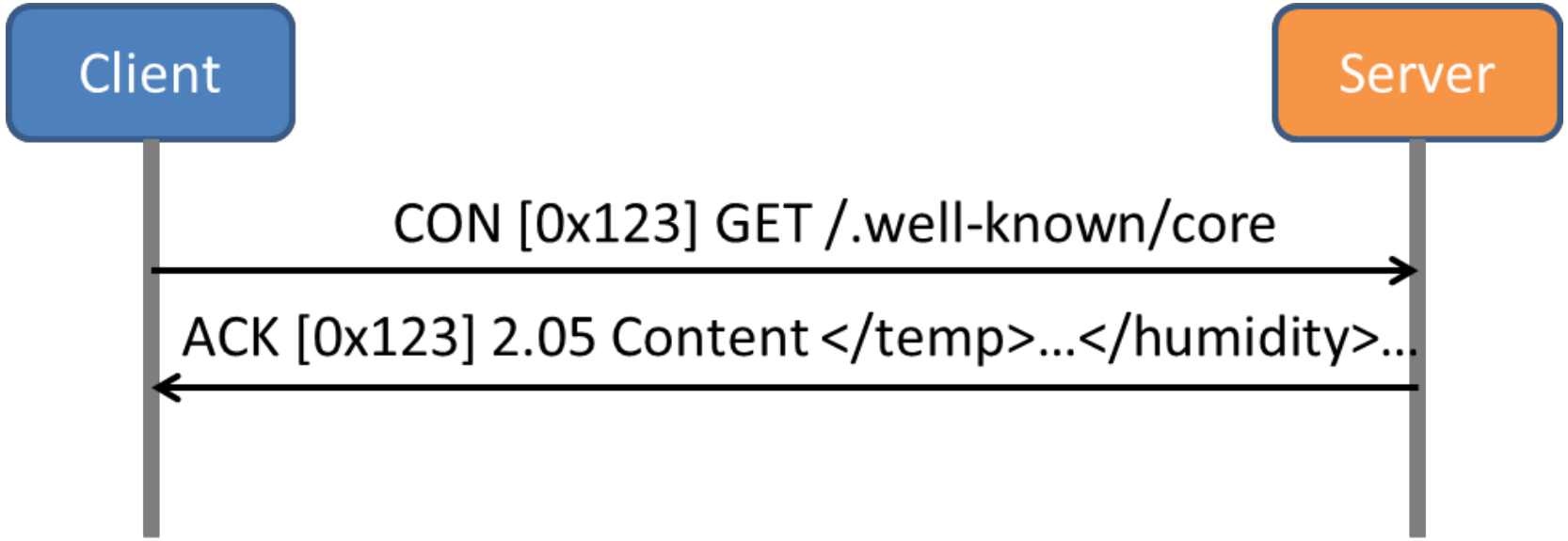
CoAP : communication patterns

observer



CoAP : communication patterns

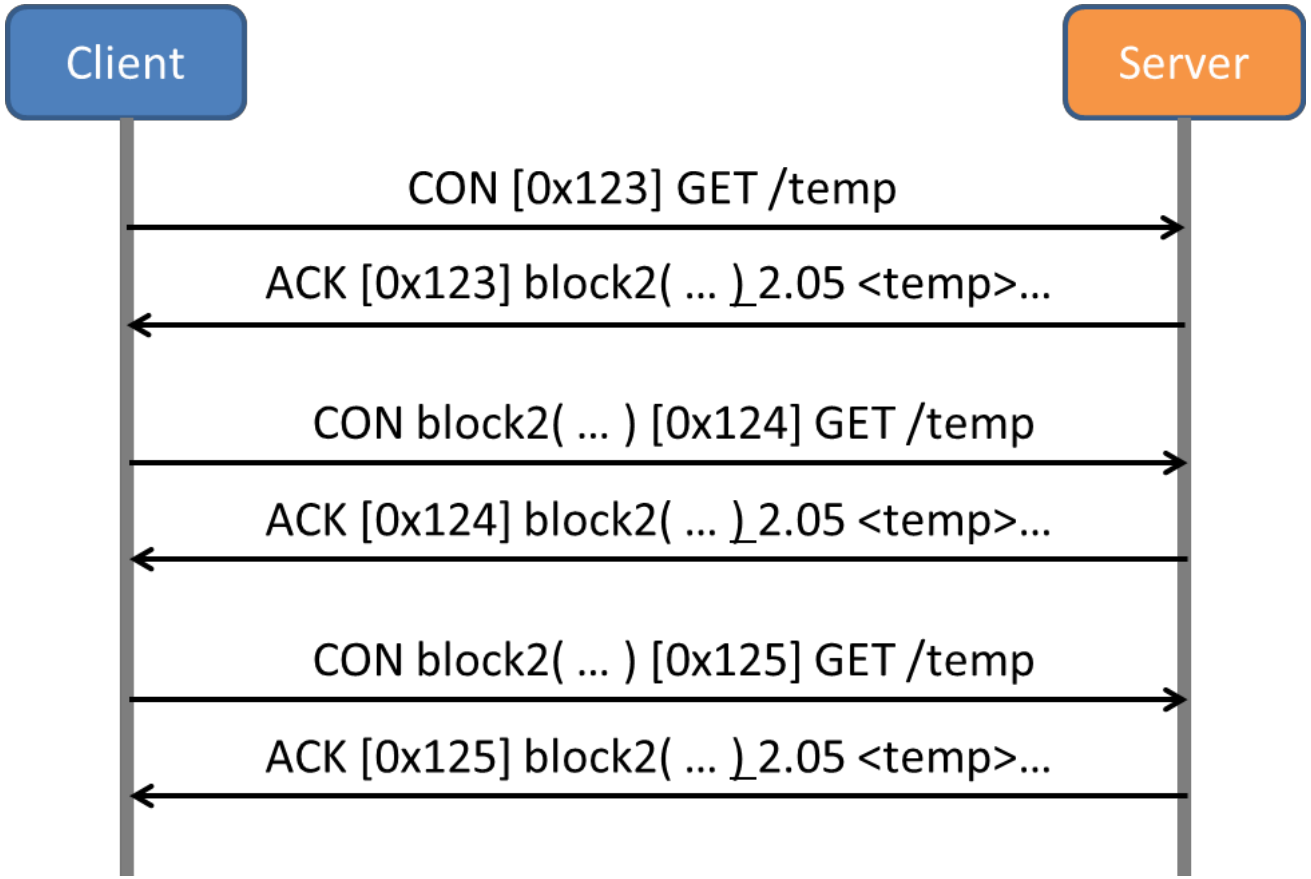
resource discovery



Example : </temp>;ct=50;title=Temperature</humidity>;ct=50;title=Humidity

CoAP : communication patterns

block transfer



MQTT (MQ Telemetry Transport)

MQTT is designed to be lightweight and is ideal for connection scenarios where IoT devices may have limited bandwidth or other constraints requiring remote devices with small code footprints.

Feature Highlights

- Lightweight protocol (great for constrained networks)
- Supports Publish / Subscribe messaging
- Low power usage
- Minimized data packet size
- OASIS standard protocol

MQTT : introduction

MQTT ...

yesterday : Message Queue Telemetry Transport

created by IBM & Eurotech

today : MQ Telemetry Transport ... no queue

donated to Eclipse Foundation and OASIS standard

Features ...

Lightweight

Reliable Simple

MQTT : introduction

Lightweight

- smallest packet size of 2 bytes (header)
- reduced clients footprint

Reliable

- three QoS levels
  - 0 at most once
  - 1 at least once
  - 2 exactly once
- avoid packet loss on client disconnection

MQTT : introduction

Simple

TCP based : socket connection oriented

Asynchronous : no wait for response

Publish/Subscribe : decoupling producers and consumers

Payload agnostic :

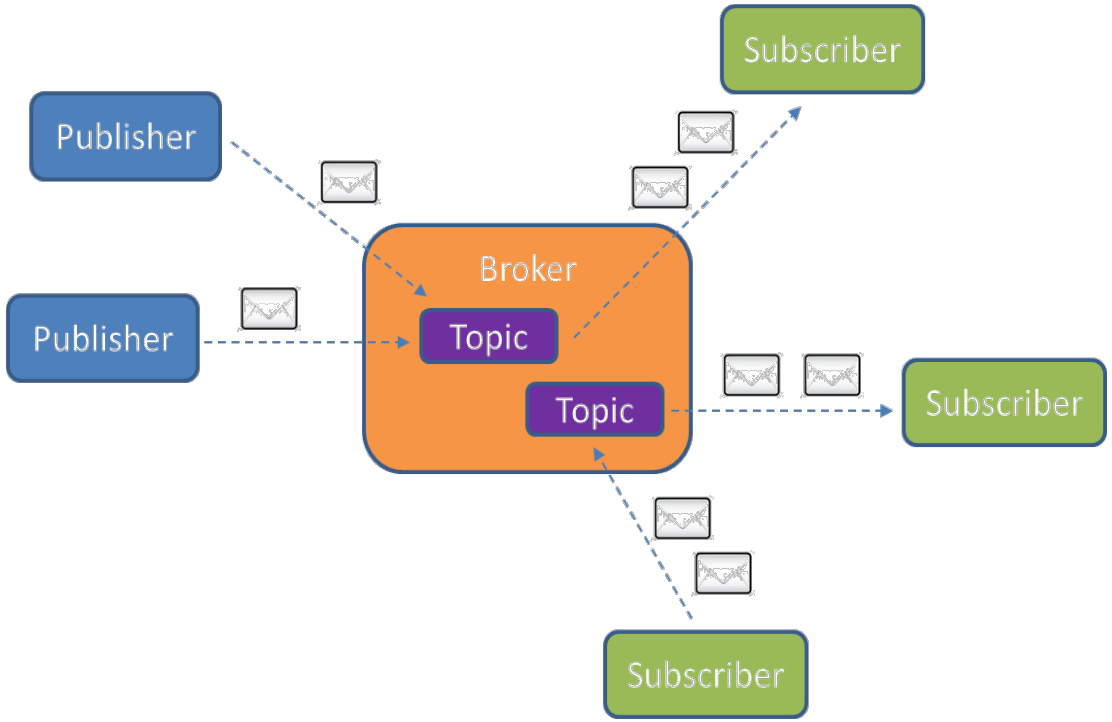
- no data types no metadata
- any data format (text, binary, JSON, XML, BSON, ProtoBuf)

MQTT : publish/subscribe

Broker and connected Clients

- broker receives subscriptions from clients on topics
- broker receives messages and forwards them
- clients subscribe/publish on topics

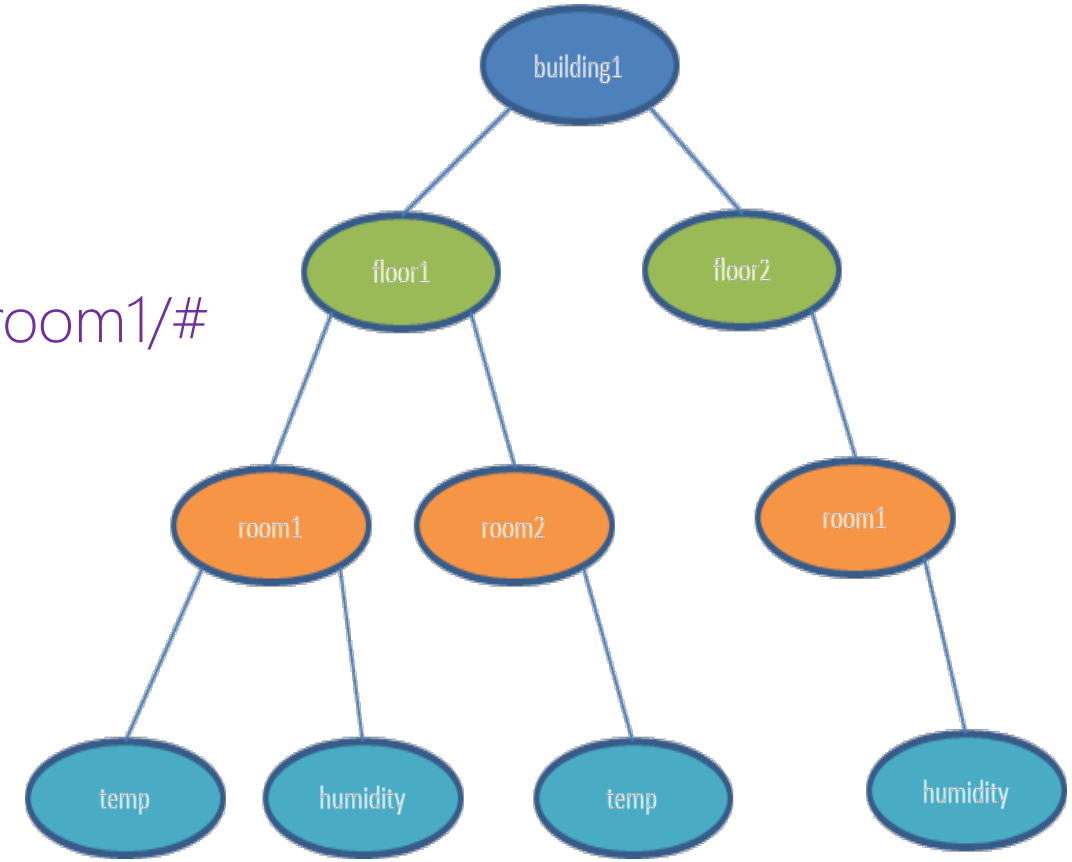
Brokers bridge configuration



MQTT : topics

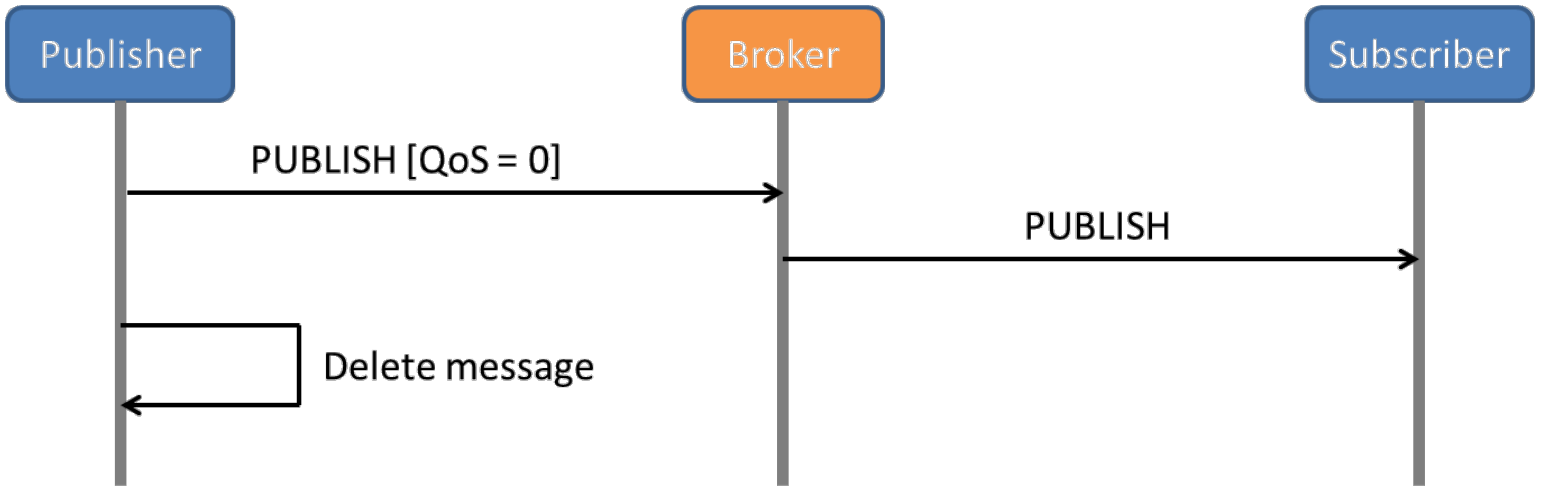
Topics for publish and subscribe

- hierarchical
- wildcards (# and +)
  - ex. building1/+/room1, building1/floor1/room1/#



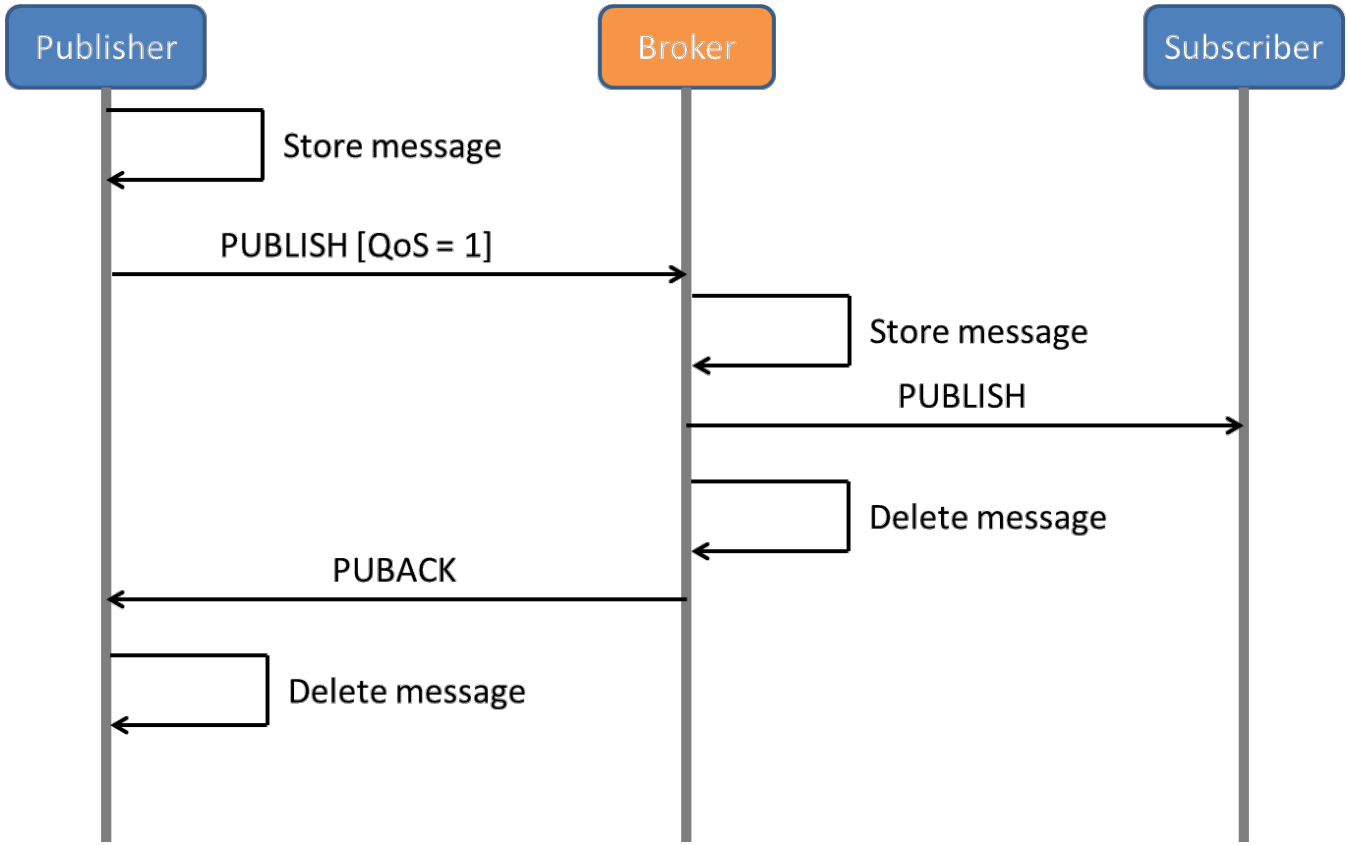
MQTT : Quality of Service

QoS 0 : At most once (fire and forget)



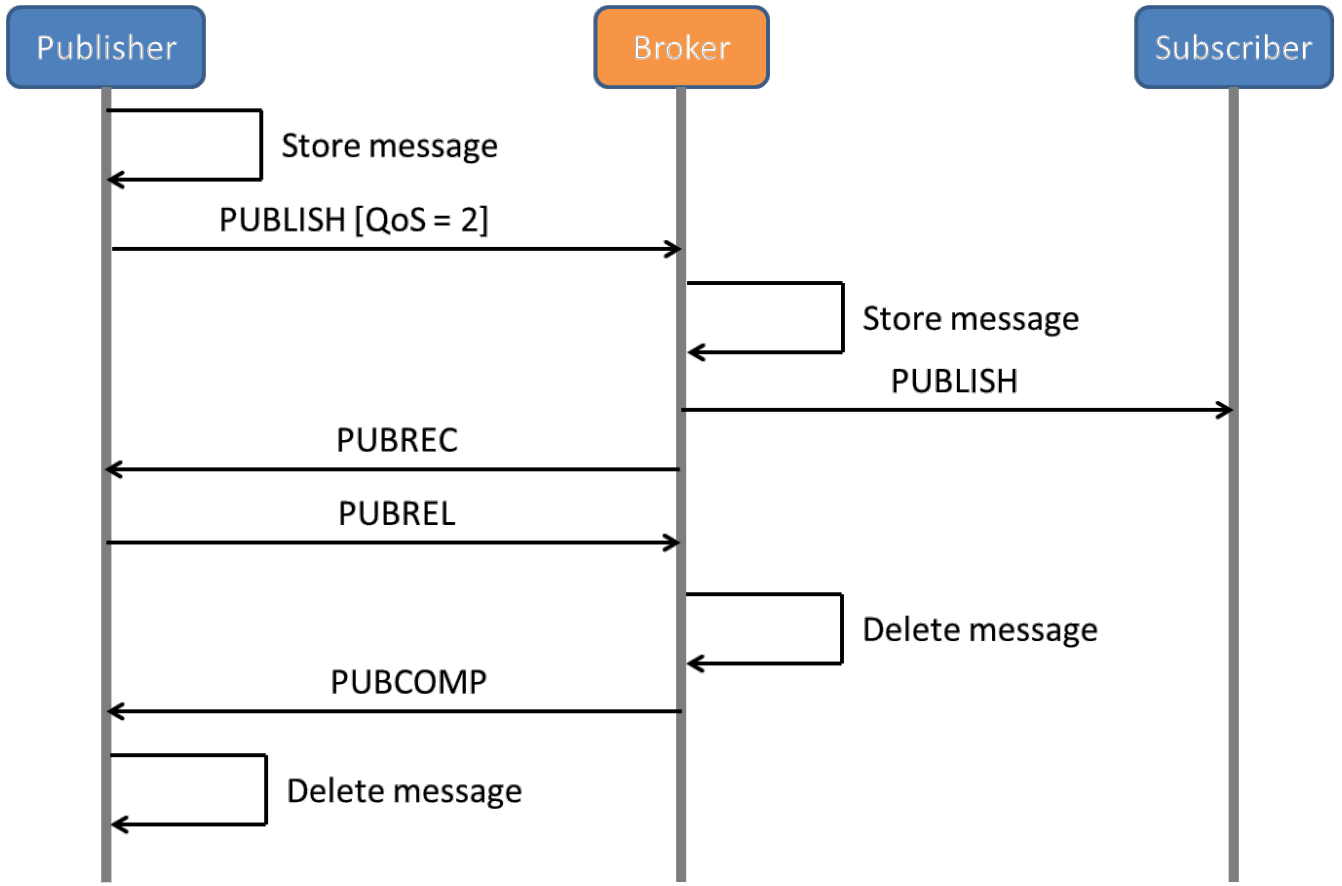
MQTT : Quality of Service

QoS 1 : At least once



MQTT : Quality of Service

QoS 2 : Exactly once



MQTT : Security

- common big problem for all IoT protocols
- MQTT is over TCP … use SSL/TLS for security
- username/password on the connection
- encrypt payload (MQTT is payload agnostic)

MQTT : basic and advanced features

Keep-Alive message

- PINGREQ/PINGRESP message
- The broker can detect client disconnection

Will & Testament

- will message with QoS and topic on connection
- broker sends on unexpected client disconnection

MQTT : basic and advanced features

Retain message

- the published message is kept on the broker
- a new subscriber on topic receives the «last known» good message

Clean session

- on client disconnection, all subscriptions are kept
- no need to re-subscribe on the re-connection
- the client receives all messages published when offline

AMQP (Advanced Message Queue Protocol)

AMQP is a binary protocol designed to support a wide array of messaging applications and communications patterns. It's not specifically built for IoT solutions, but it works very well for message communications which include many IoT scenarios.

Feature Highlights

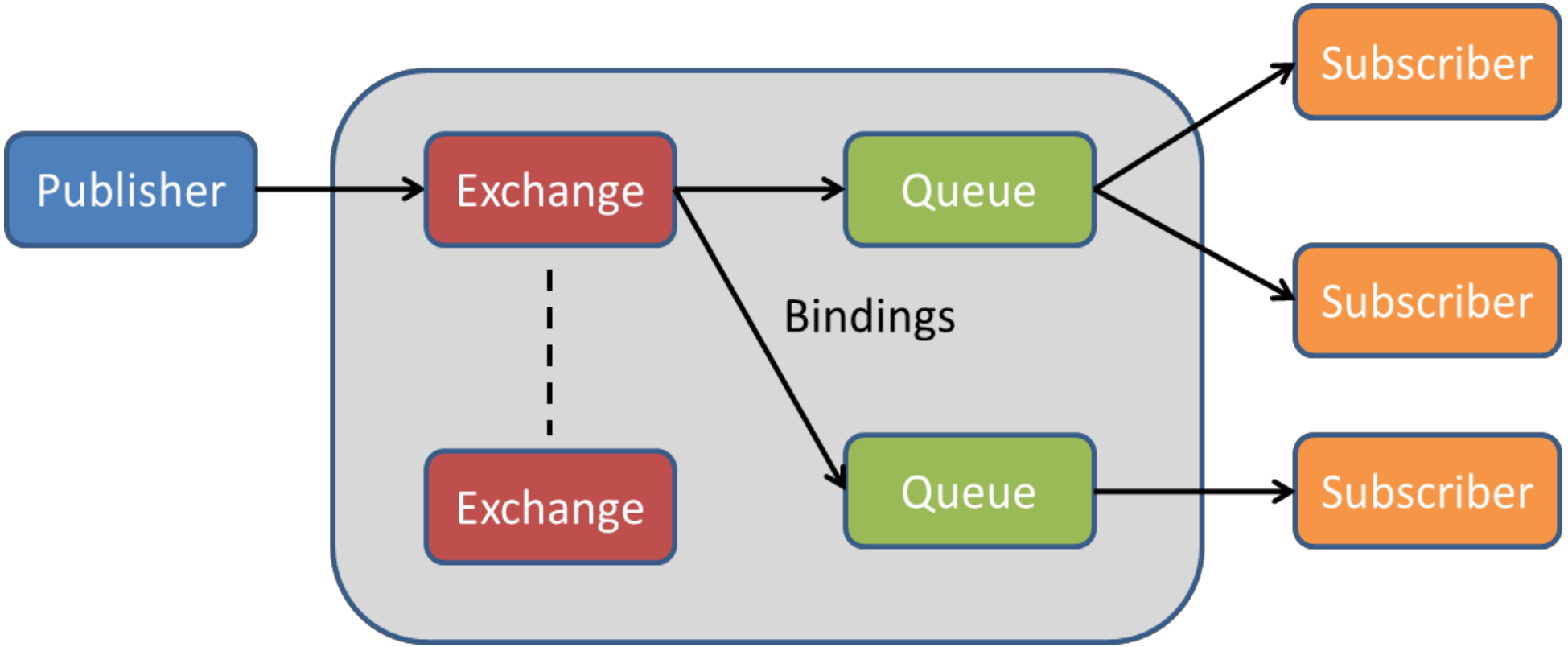
- Binary application layer protocol
- Can be used for Point-to-Point and Publish / Subscribe messaging
- Broad compatibility with messaging scenarios
- Supports end-to-end encryption of messaging

AMQP Introduction

- architecture
  - AMQP Model (broker definition)
  - wire protocol
- exchange
  - receive messages and apply routing
- binding
  - define rules to bind exchange to queue
- queue
  - store messages

AMQP : broker architecture

AMQP Model



AMQP : exchanges

default exchange (without a name)

- routing messages to a queue (routing key = name queue)

direct exchange

- routing message to a queue based on routing key (not necessary queue name, routing key = bind key)

fanout exchange

- routing message to more queues (pub/sub) and not using a routing key

AMQP : exchanges

topic exchange

routing message to a queue based on the routing key like a topic (routing key match pattern)

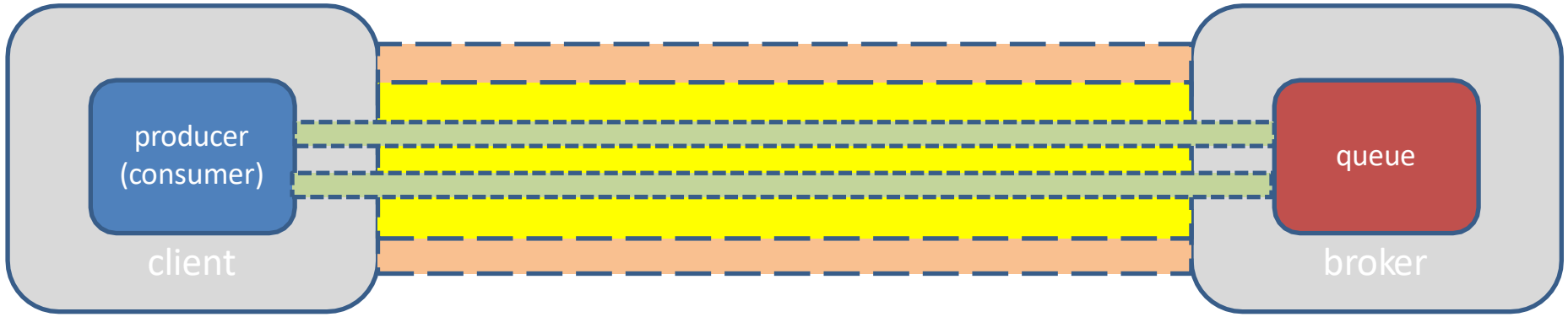
header exchange

routing message to queue based on header filters

AMQP (1.0) : containers and nodes

- architecture
  - wire protocol and data types system
- container
  - client contains producer and/or consumer
  - broker contains queue
- node
  - producer sends messages
  - consumer receives messages
  - queue store and/or forward messages

AMQP (1.0) : transport



container

node

connection

session

link

container

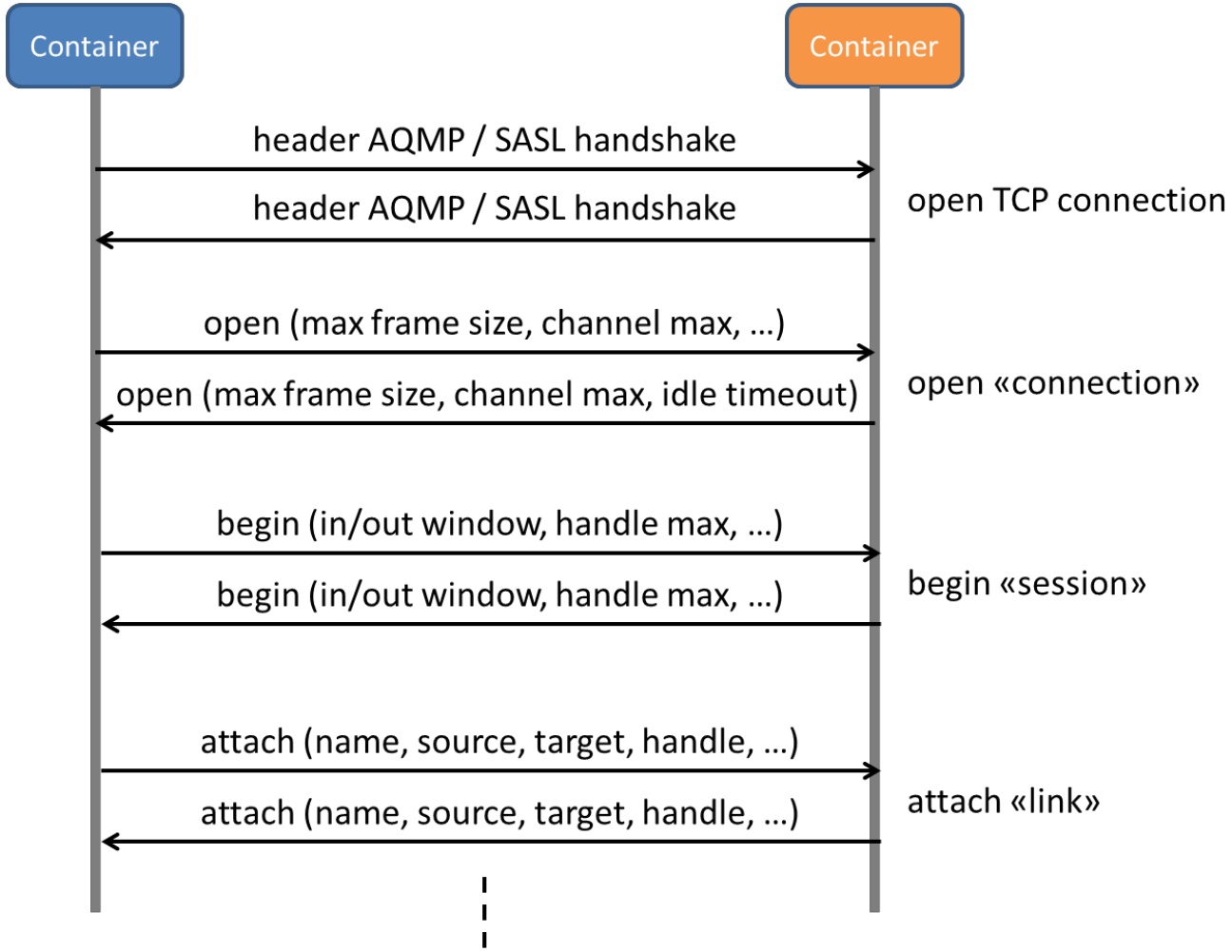
node

multiplexing frames on sessions and links

transport independent

AMQP (1.0) : communication

open connection/session/link

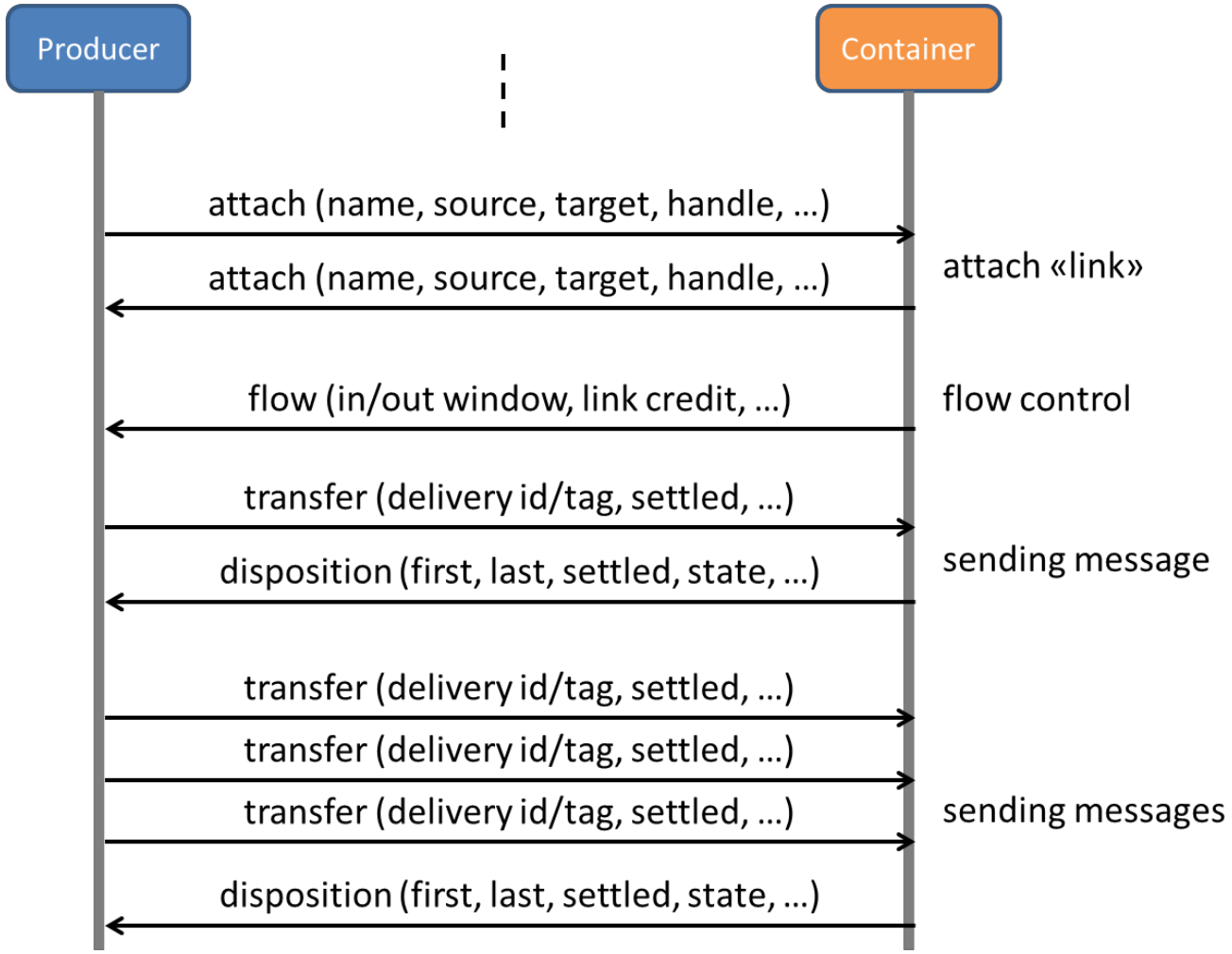


channel ⇔ session

handle ⇔ link

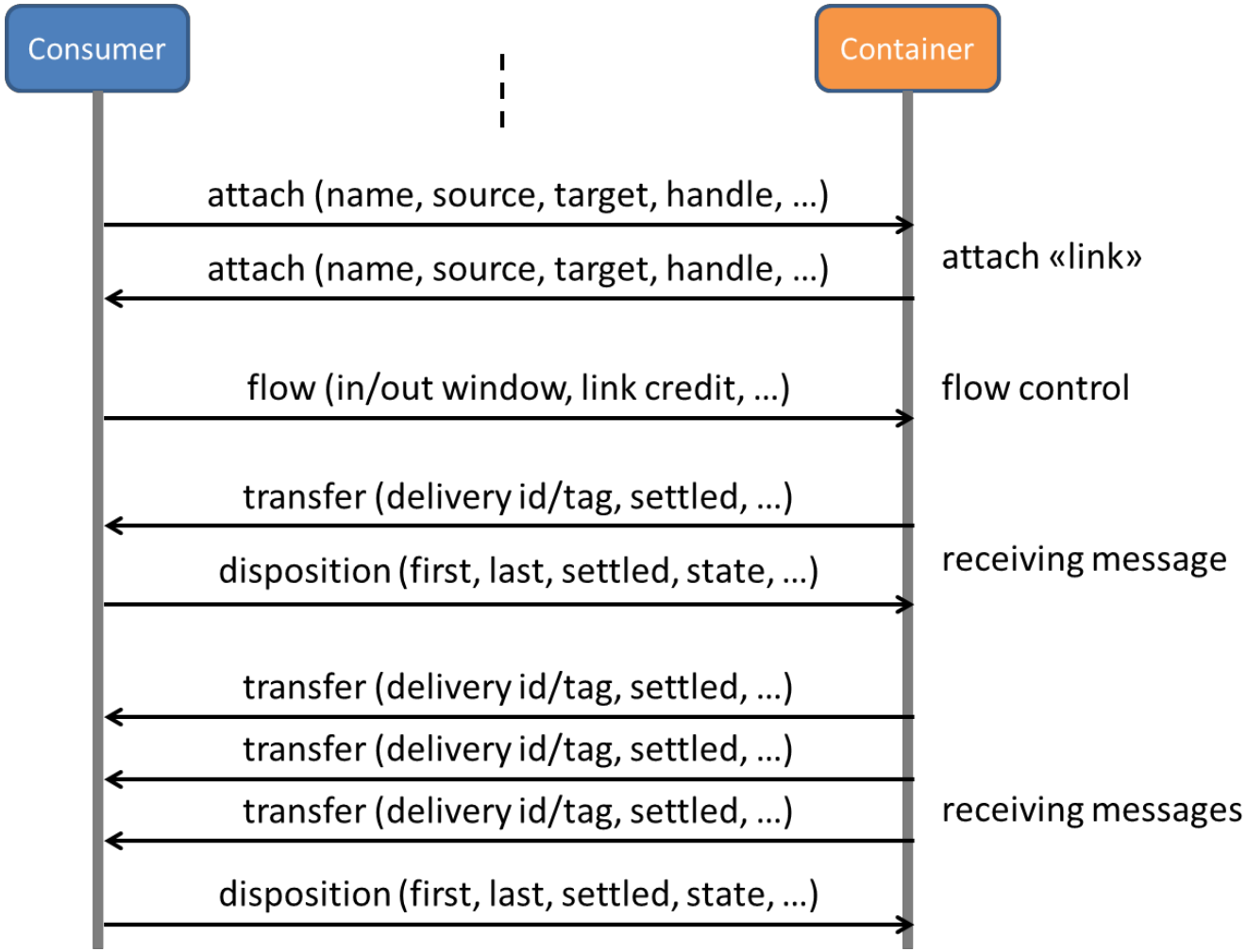
AMQP (1.0) : communication

send messages (ex. producer to queue)



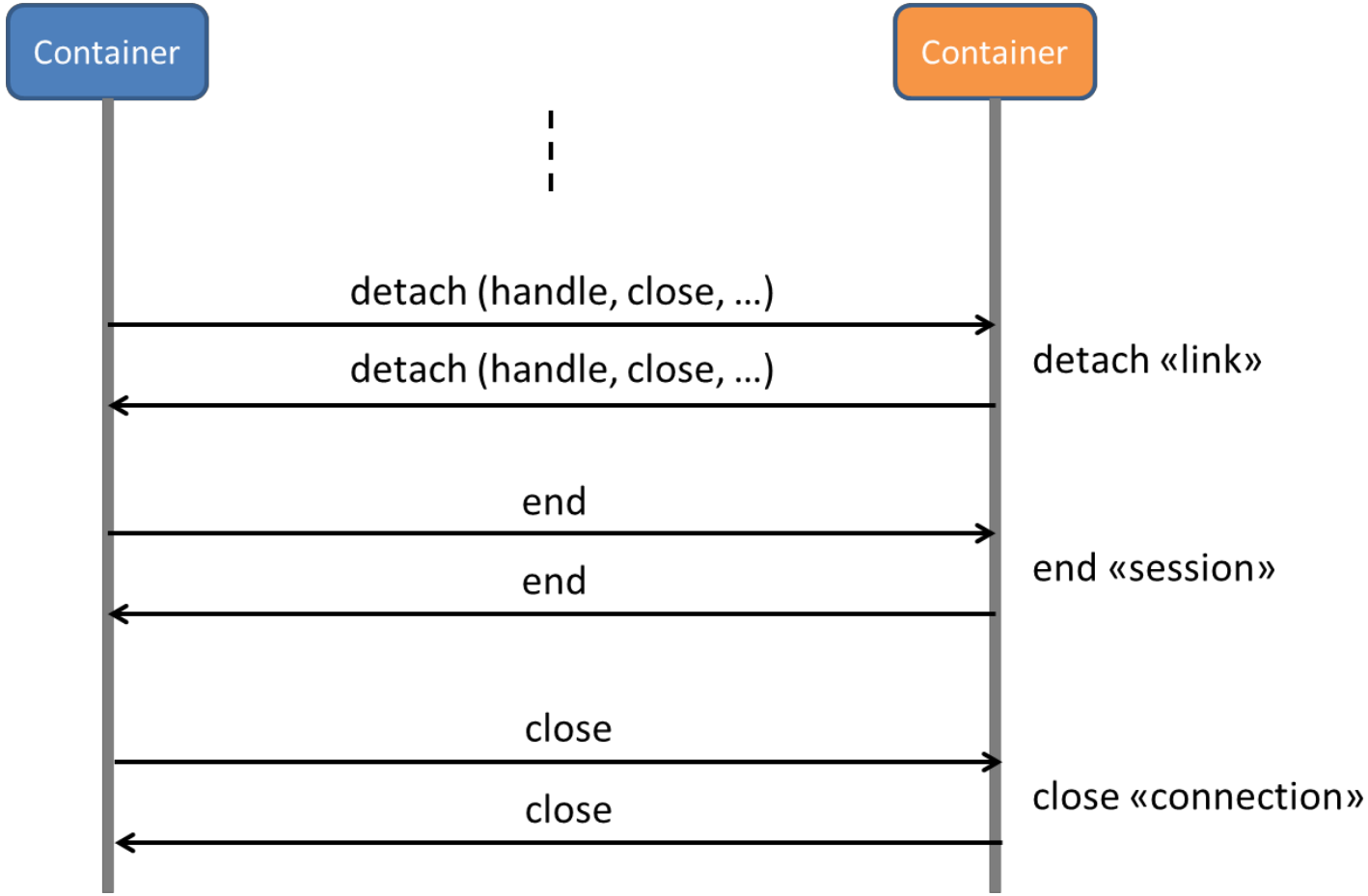
AMQP (1.0) : communication

receive messages (ex. consumer from queue)



AMQP (1.0) : communication

close link/session/connection



AMQP : messages

message

- header
  - system properties (ex. correlationId, replyTo, TTL, ...)
  - custom/user properties
- body (opaque)

message metadata

most times body empty, all values as properties

filter on properties

properties can be changed «on fly»

AMQP : main features

- messaging middleware
  - asynchronous : producer and consumer decoupled
  - poll mode credit based to receive
- more messaging patterns
  - load balancing on a queue (more consumers)
  - pub/sub on queue
  - messages redirection to queues based on filters
  - request/response w/ «correlation id» and «replyTo»
- session and transactional message transfer

AMQP : advantages

- efficient
  - binary connection-oriented
  - "flow control" credit-based
  - packet size 60 bytes
- reliable
  - Quality of Service (best effort, at least once, exactly once)
- security
  - SSL/TLS
  - SASL (Simple Authentication and Security Layer)

Conclusions

devices

- consider how much they are constrained

network

- how much it is reliable

messages rate

- how many messages per second and QoS

process data

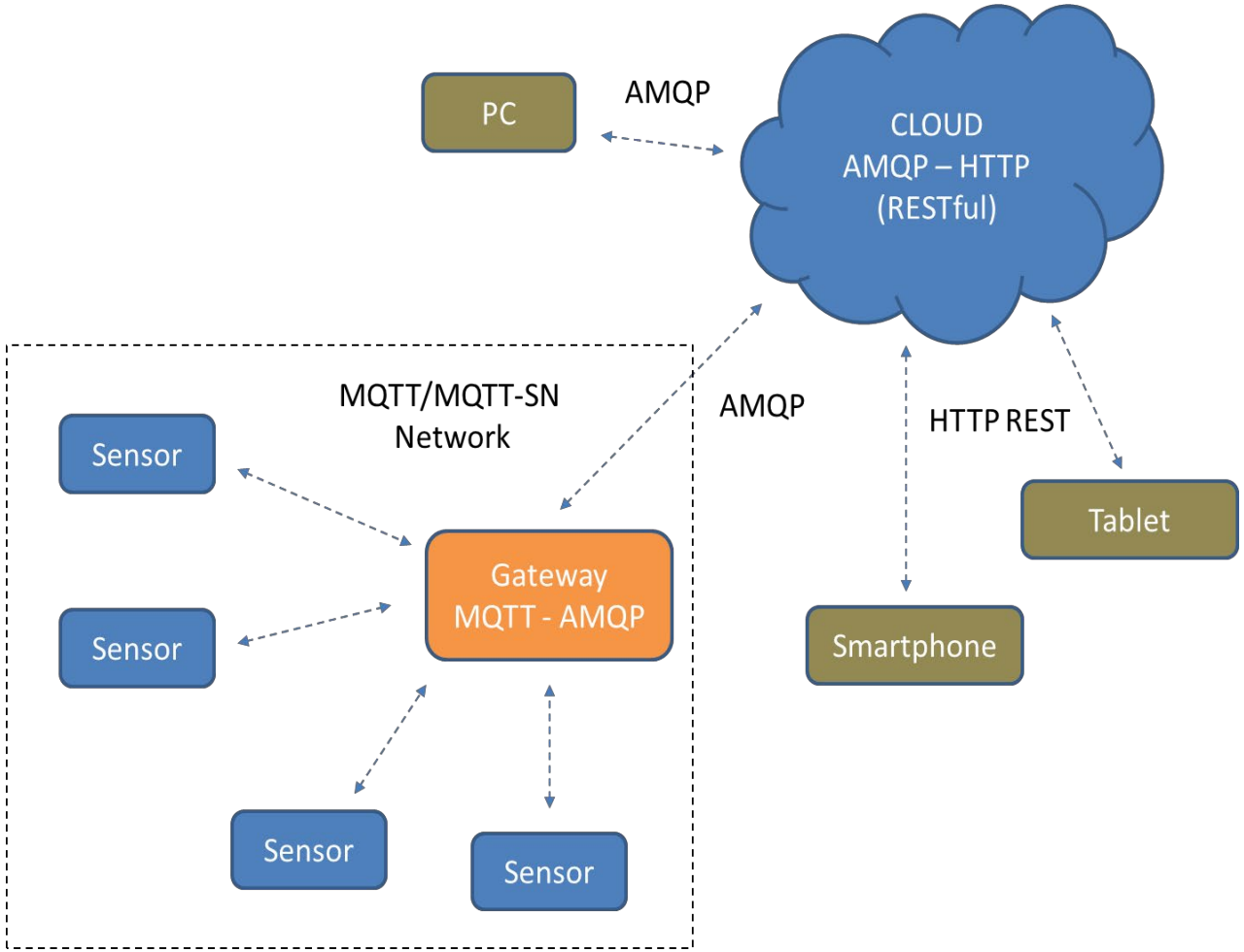
- needs of the system to process data

Conclusions

protocol choice depends on the scenario

some protocols have more features than other

a complex system can use more protocols



Resources

IoT/M2M

- Embedded101 free ebook : http://bit.ly/m2miotbook
- Subscribe! Blog : http://channel9.msdn.com/Blogs/Subscribe
- IBM redbook : http://www.redbooks.ibm.com/abstracts/sg248054.html
- IoT with Azure Service Bus : http://channel9.msdn.com/Events/Build/2014/3-635
- Windows and Internet of Things : http://channel9.msdn.com/Events/Build/2014/2-511

MQTT

- Official web site : http://mqtt.org
- M2Mqtt project : http://www.m2mqtt.net
- Mosquitto : http://mosquitto.org
- HiveMQ : http://www.hivemq.com
- Eclipse IoT : http://iot.eclipse.org
- MQTT An implementer's perspective : http://bit.ly/1koMZLF
- MQTT Another implementer's perspective : http://bit.ly/1rHDnAN

Resources

CoAP

- Official draft : https://datatracker.ietf.org/doc/draft-ietf-core-coap
- CoRE Link format : http://tools.ietf.org/html/rfc6690#section-3.1
- CoAPSharp : http://www.coapsharp.com
- Copper : https://github.com/mkovatsc/Copper

AMQP

- Official web site : http://www.amqp.org
- Microsoft Azure Service Bus : http://azure.microsoft.com/en-US/services/messaging/
- AMQP.Net Lite : https://amqpnetlite.codeplex.com/
- Qpid project : http://qpid.apache.org/
- RabbitMQ : http://www.rabbitmq.com
- ActiveMQ : http://activemq.apache.org/

Summary / Recap of Main Points

During this lesson, the following topics were covered:

- IoT communication patterns and their importance in facilitating efficient communication among IoT devices.
- IoT application protocols landscape, including the role of protocols in ensuring seamless communication between IoT devices.
- Key characteristics and requirements of IoT application protocols, such as low power consumption, scalability, and security.
- Compare and contrast popular IoT application protocols, such as HTTP, CoAP, MQTT, and AMQP, in terms of their architecture, features, and suitability for specific IoT use cases.

Review Questions

- How do IoT application protocols contribute to seamless communication between IoT devices in heterogeneous environments?
- Compare the architecture of HTTP, CoAP, MQTT, and AMQP in terms of their messaging paradigms and network models.
- Describe the differences between the publish/subscribe model used by MQTT and the request/response model used by HTTP.
- Discuss the role of security features in IoT application protocols and the potential risks of insecure protocols.
- Provide examples of IoT applications that benefit from the use of HTTP, CoAP, MQTT, or AMQP, and explain the rationale behind the selection.

What To Expect Next Week

In Class

- Topic 7 - IoT Network Access and Physical Layer Protocols

Preparation for Class

- Understanding the Network Access and Physical Layer