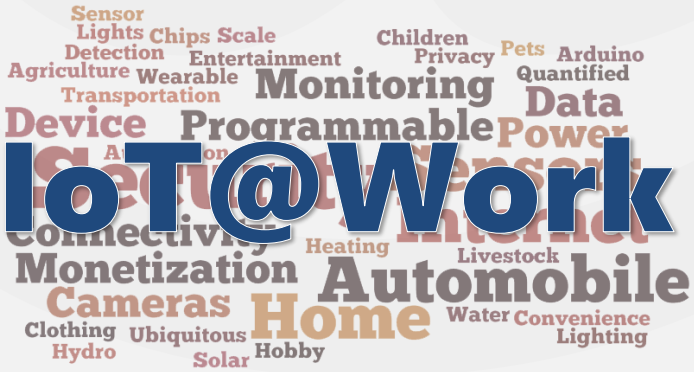
DomusDotNet
User Group romano per lo studio e la divulgazione delle tecnologie .NET

TinyCLR.it
La Community Italiana su .NET Micro Framework

IoT protocols landscape

Paolo Patierno
Microsoft MVP Windows Embedded
ppatierno@live.com
@ppatierno



Who am I ? Contacts

Software Engineer (Ditron S.r.l.)

Microsoft MVP Windows Embedded

"DotNetCampania" member

http://dotnetcampania.org/blogs/paolopat/default.aspx

"Embedded101" board of director member

http://www.embedded101.com/Blogs/PaoloPatierno.aspx

"TinyCLR.it" member

http://www.tinyclr.it

Linkedin

http://it.linkedin.com/in/paolopatierno

DomusDotNet
User Group romano per lo studio e la divulgazione delle tecnologie .NET

TinyCLR.it

Agenda

- IoT communication patterns
- IoT 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

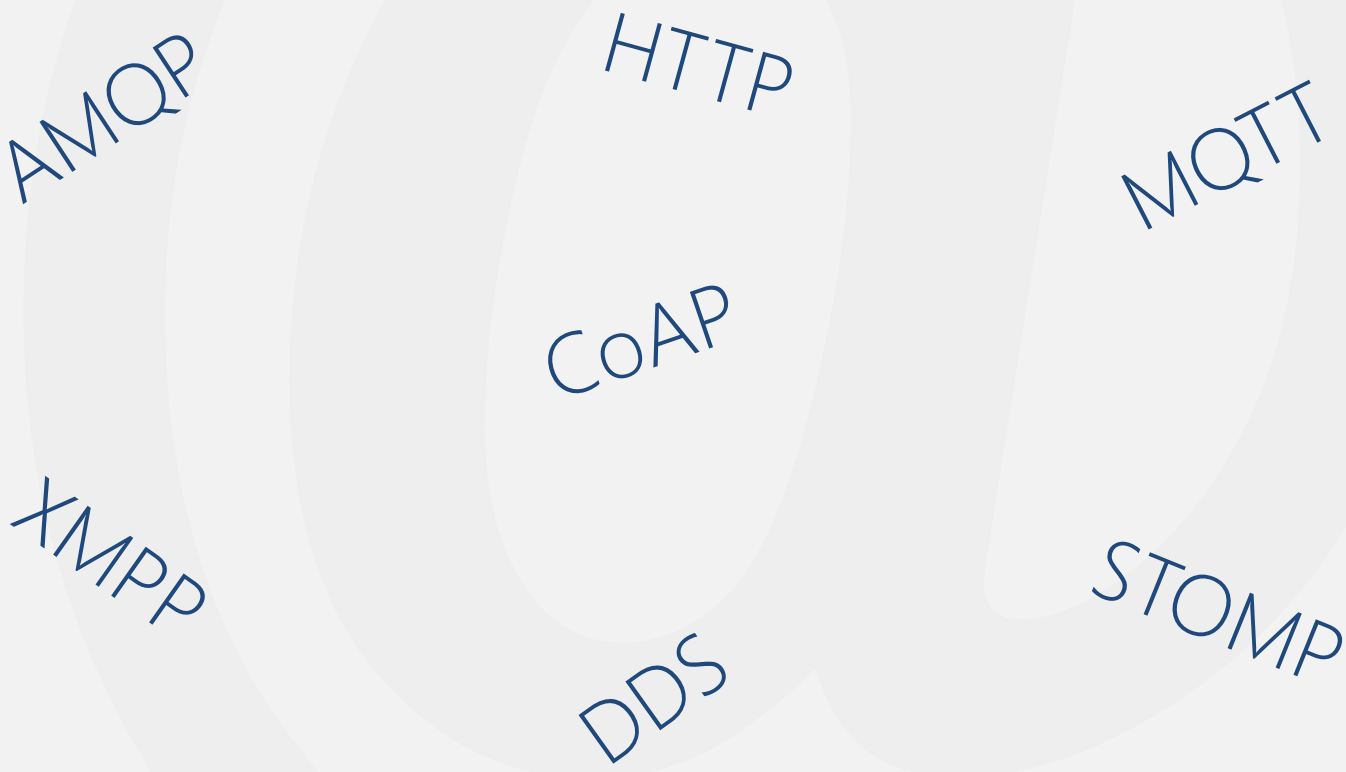
Commands

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

Notifications

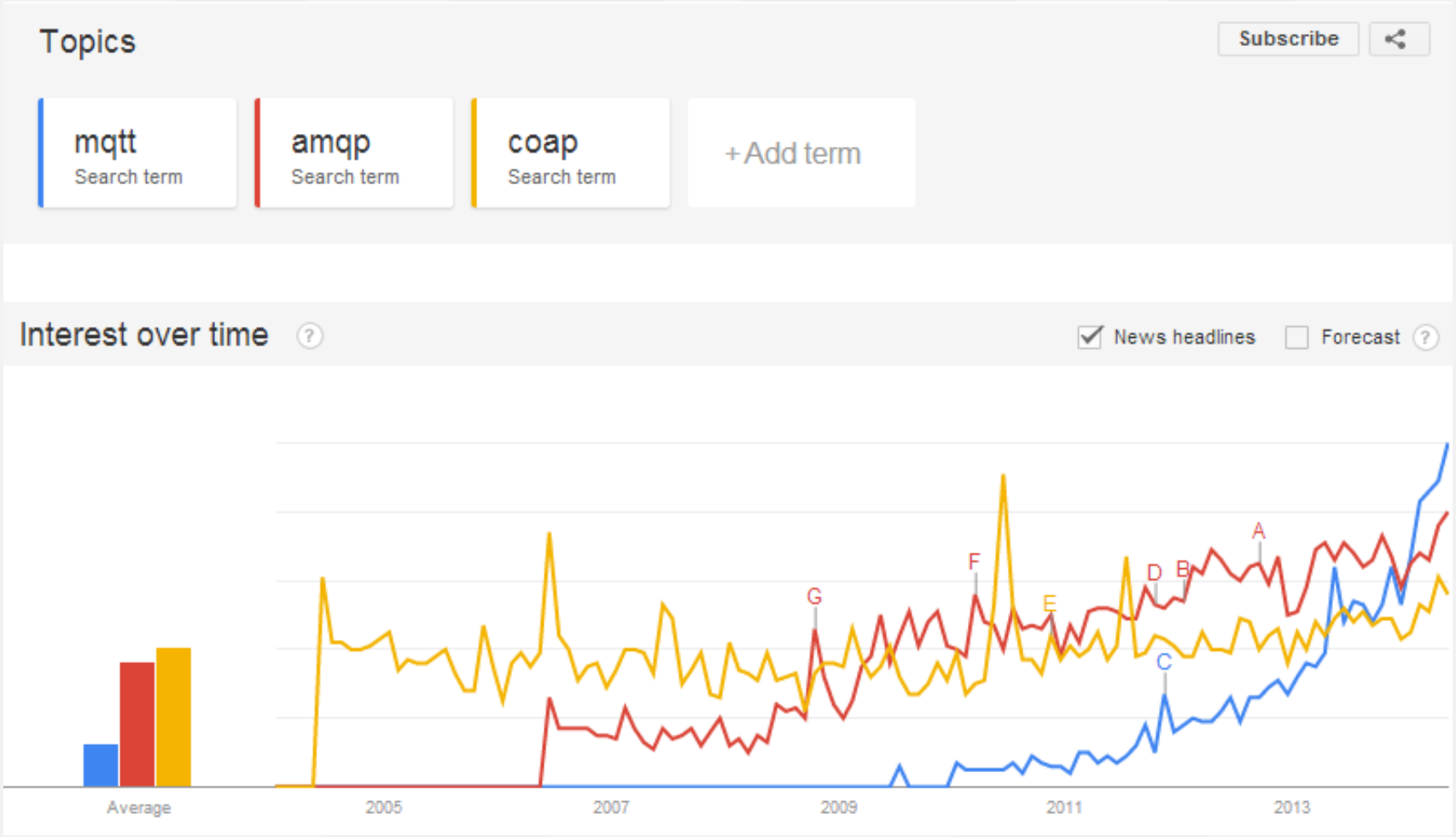
Information flows from other systems to a device or a group for conveying status changes in the world

IoT protocols landscape



... we could continue

IoT protocols trends



Standardization

HTTP

IETF standard (RFC 2616 is HTTP/1.1)

CoAP

IETF draft 18 (December 2013)

MQTT

soon (August 2014 ?) OASIS standard

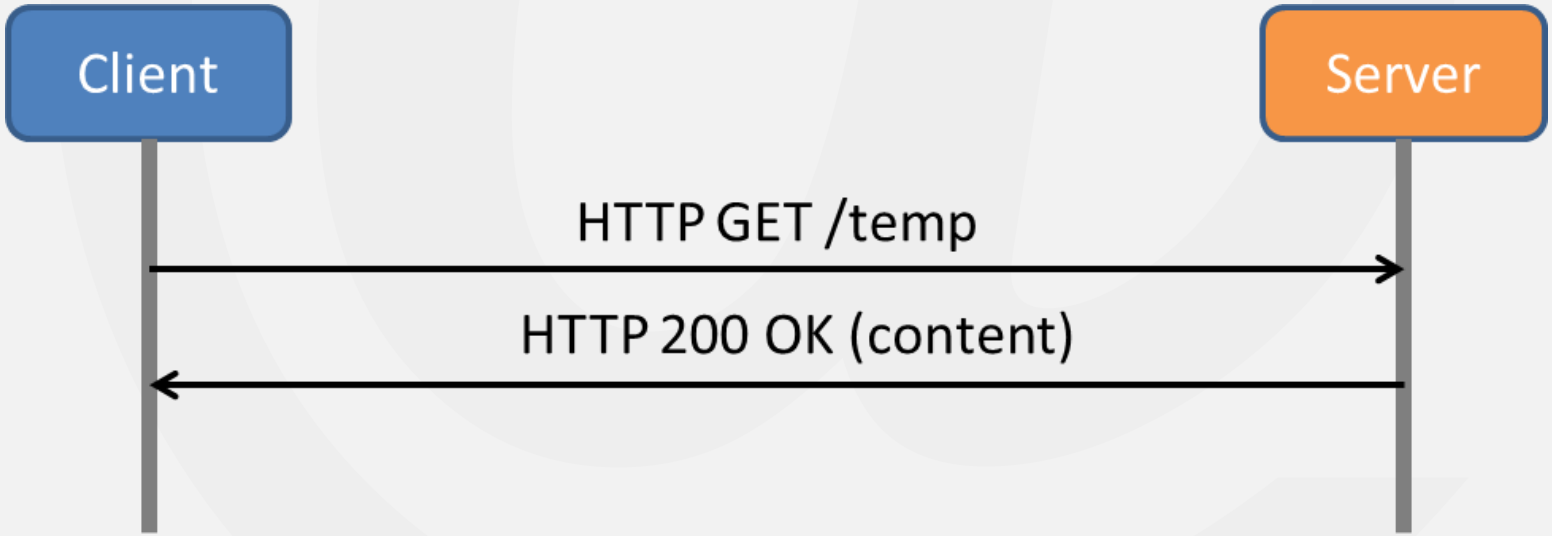
AMQP

OASIS and ISO 19464 standard (1.0)

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 access by URIs
  - CRUD operations by HTTP methods

CoAP : binary HTTP-like

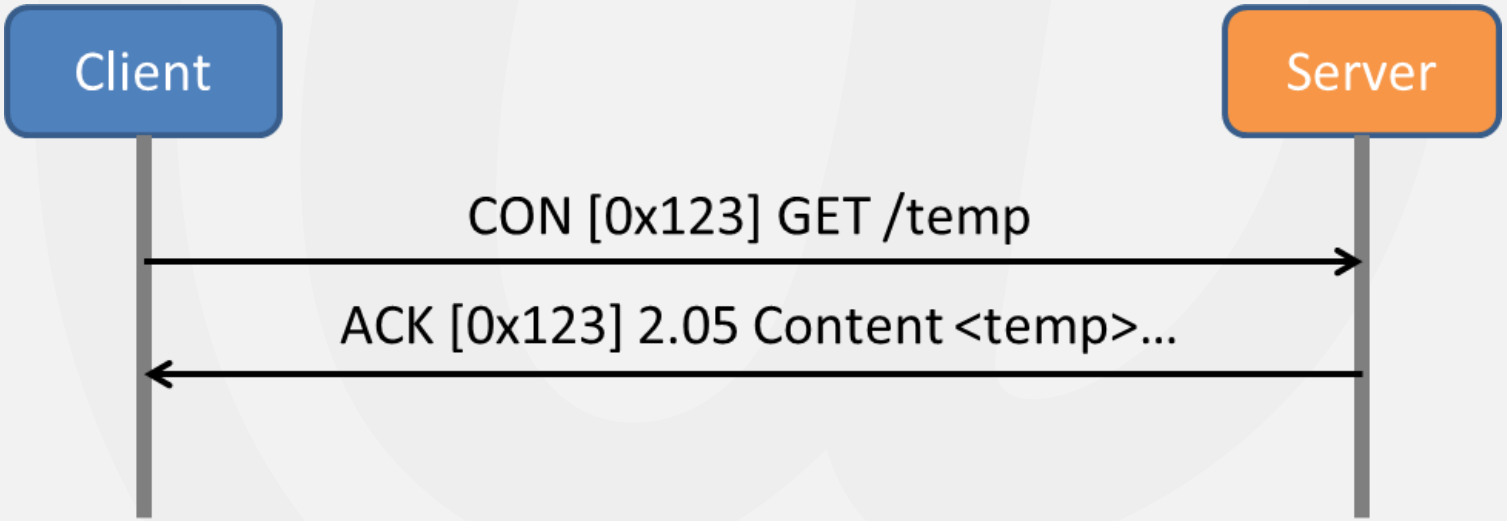
- HTTP-like based on UDP (no TCP)
  - request/response
  - packet order and retrasmission into sw stack
  - HTTP verbs, status codes, …
  - “options” like HTTP headers but binary
  - client more simple than HTTP
- observer pattern available
  - avoid HTTP (long) polling
  - separate response/response 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 server
  - addressing problem (mobile roaming, NAT, …)
- REST architecture
- proxy CoAP – HTTP simple (with caching)

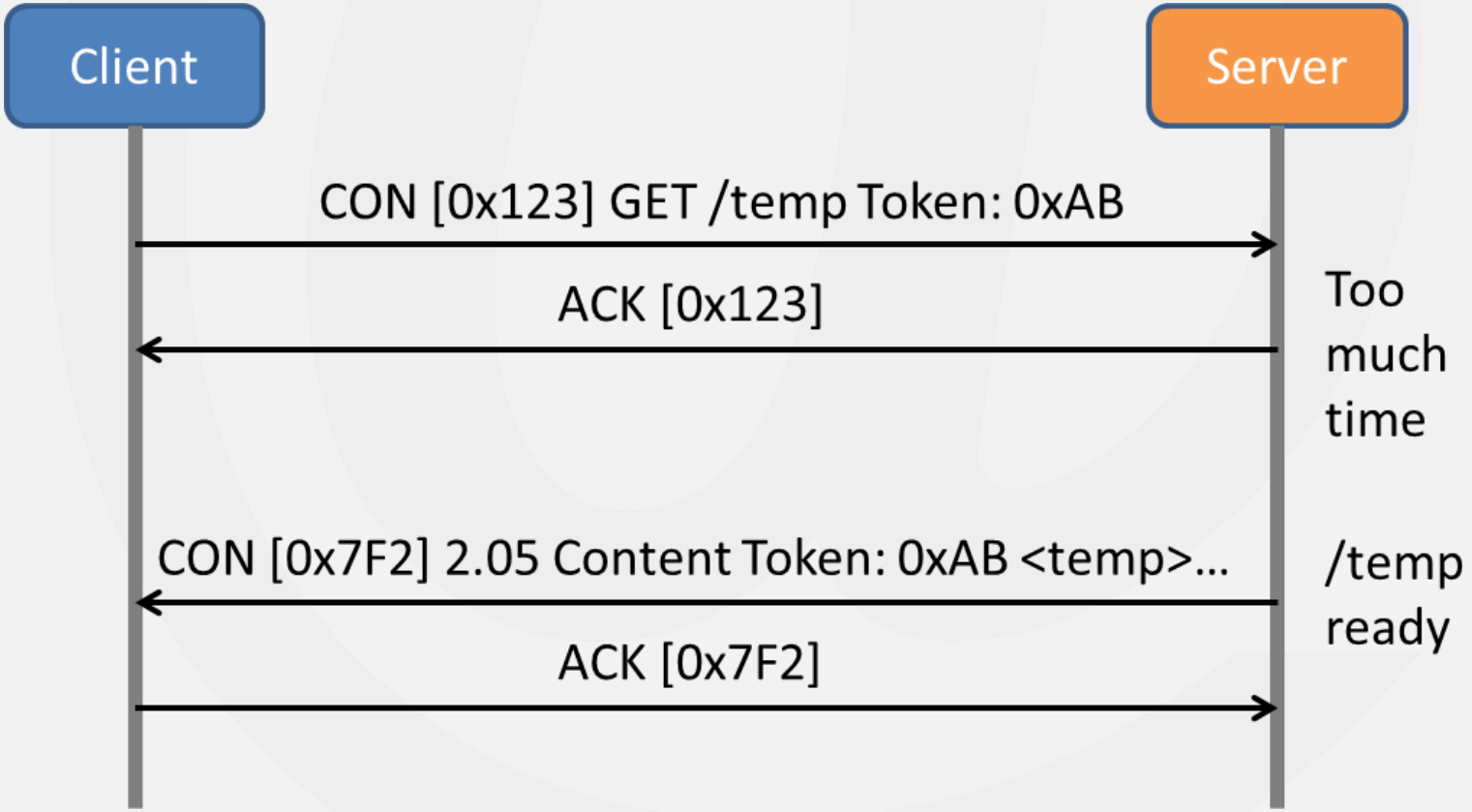
CoAP : communication patterns

confirmable request



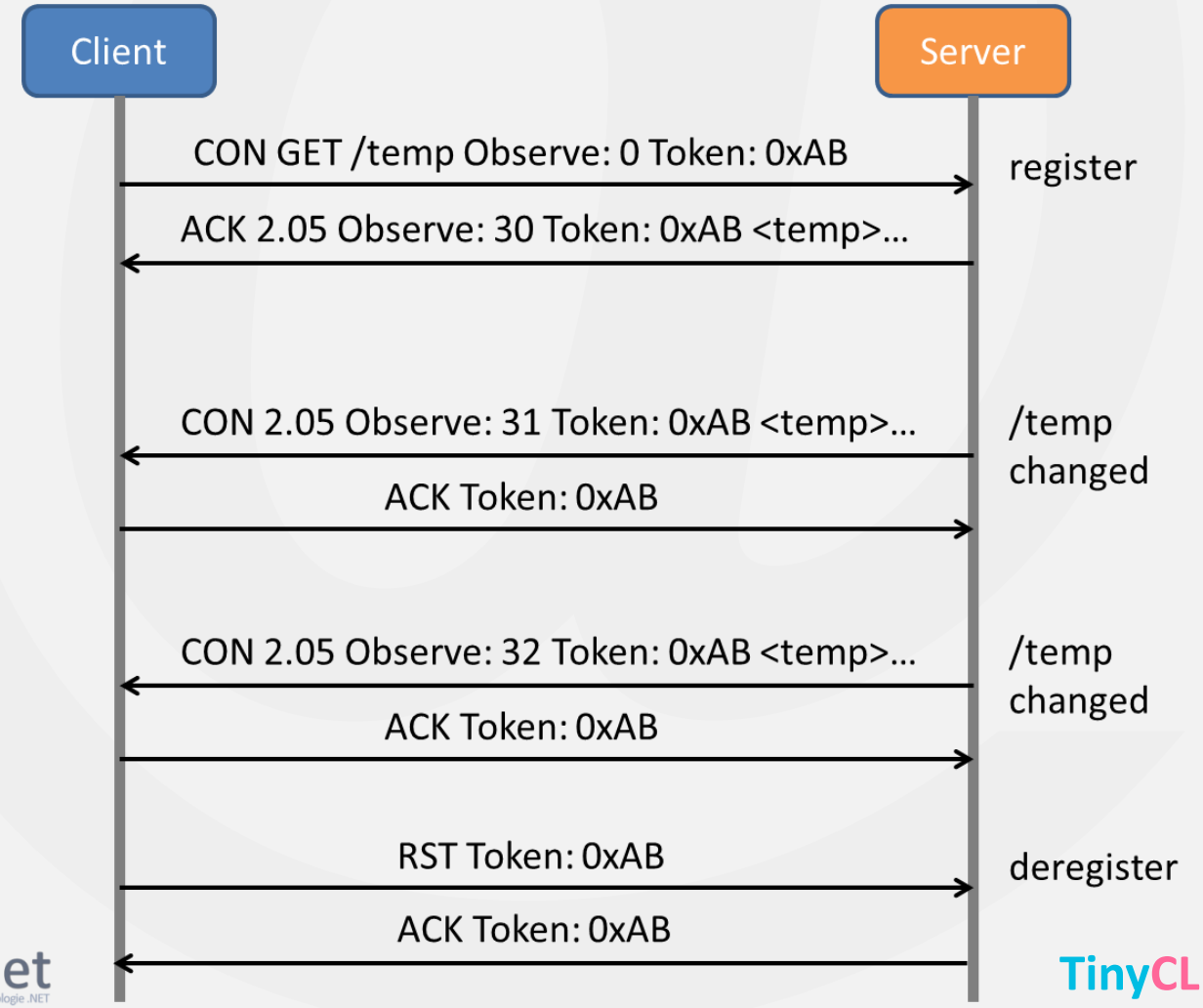
CoAP : communication patterns

separate response/response back after a while



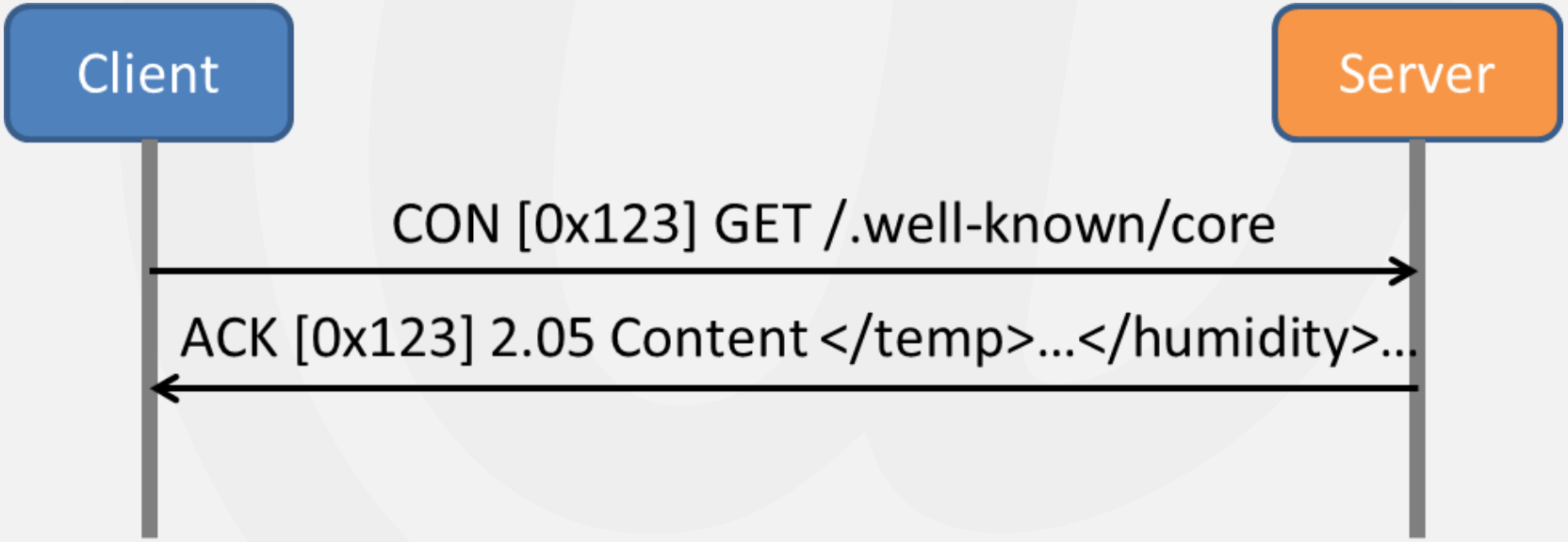
CoAP : communication patterns

observer



CoAP : communication patterns

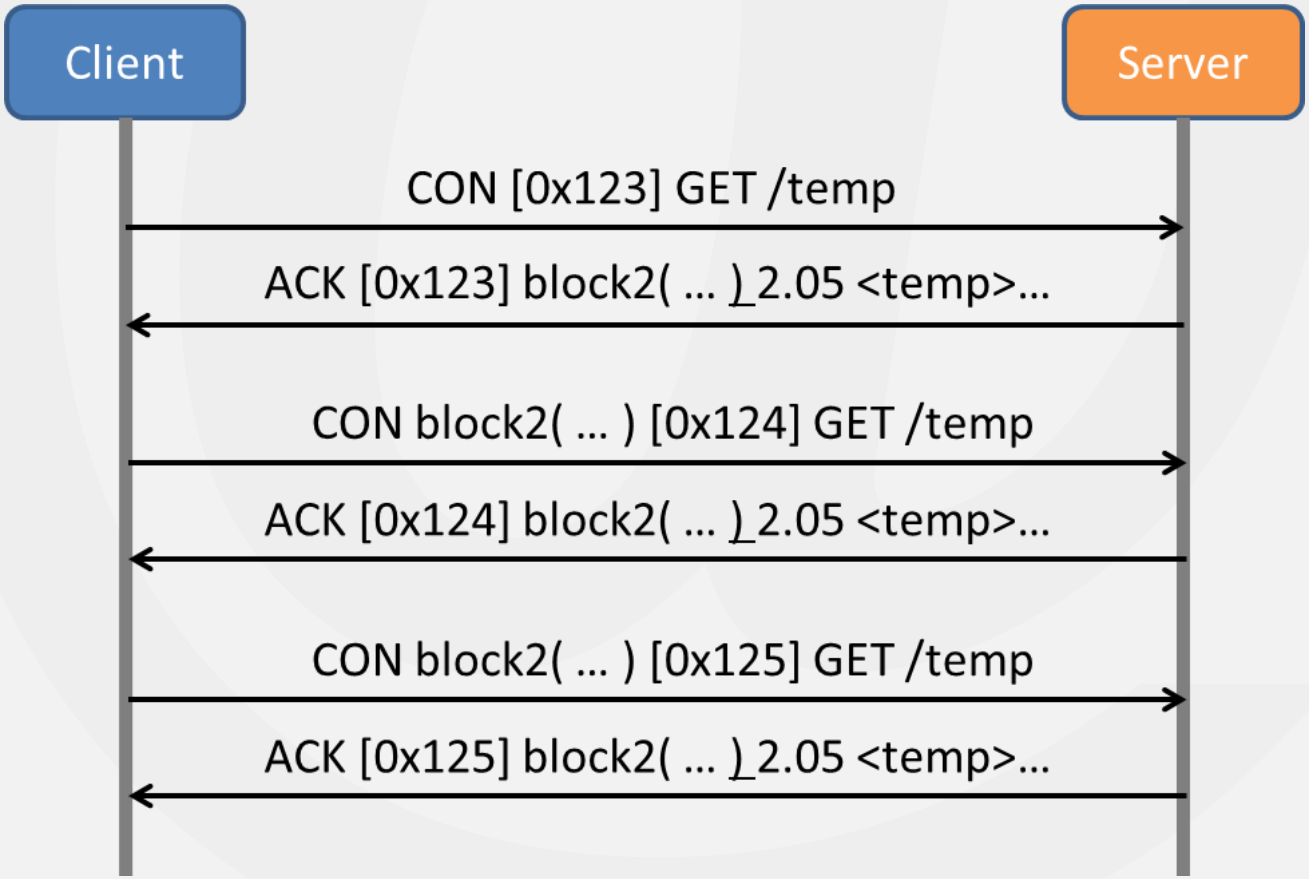
resource discovery



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

CoAP : communication patterns

block transfer



MQTT : introduction

MQTT …

- yesterday : Message Queue Telemetry Transport
  - created by IBM & Eurotech
- today : MQ Telemetry Transport … no queue
  - donated to Eclipse Foundation and OASIS standard soon

Features …

- Lightweight
- Reliable
- Simple

MQTT : introduction

- Lightweight
  - smallest packet size 2 bytes (header)
  - reduced clients footprint
- Reliable
  - three QoS levels
    - at most once
    - at least once
    - exactly once
  - avoid packet loss on client disconnection

MQTT : introduction

Simple

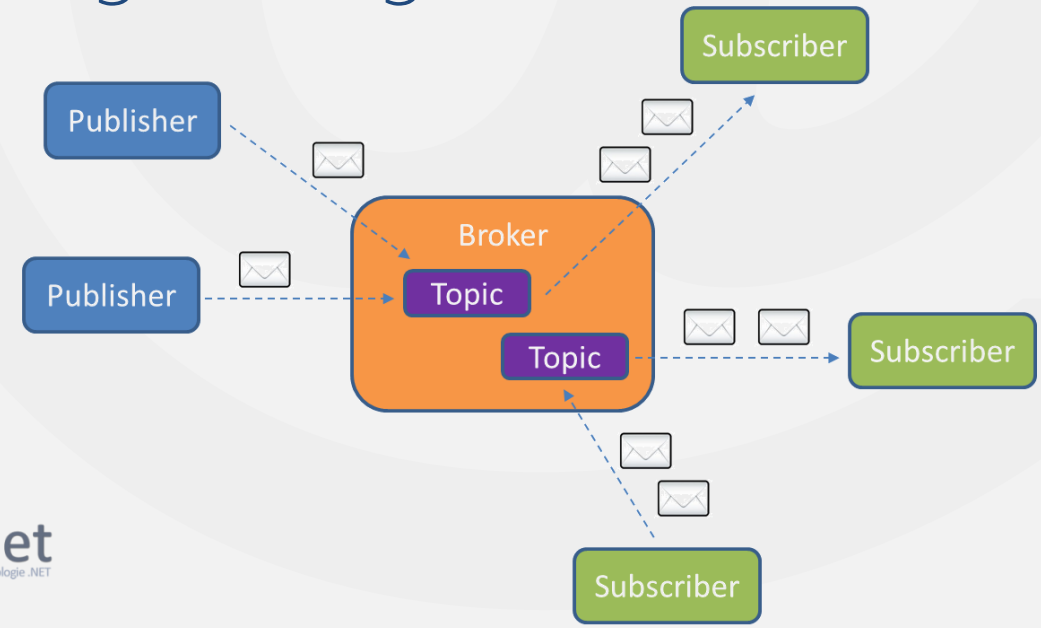
- TCP based : socket connection oriented
- Asynchronous : no wait for response
- Publish/Subscribe : decoupling producers and consumers
- Payload agnostic :
  - no data types
  - no metadata
  - any data format (text, binary, JSON, XML, BSON, ProtoBuf)

MQTT : publish/subscribe

Broker and connected Clients

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

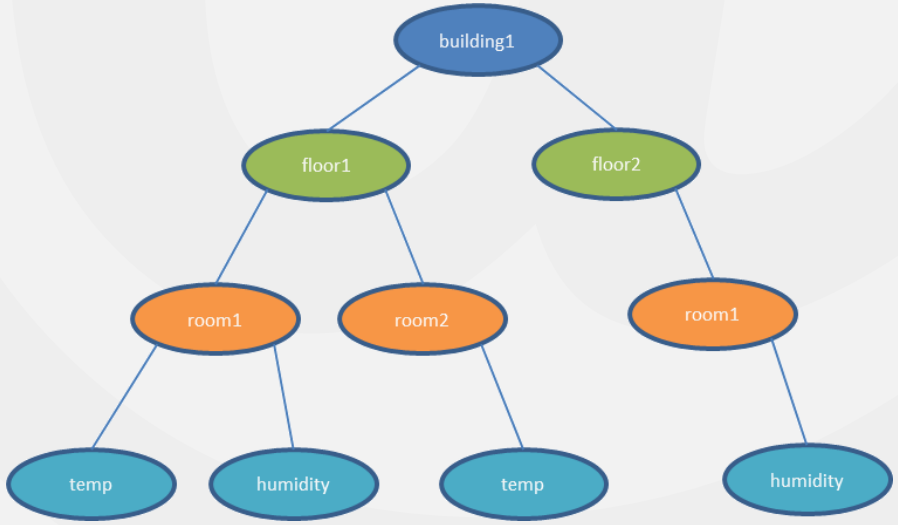
Brokers bridge configuration



MQTT : topics

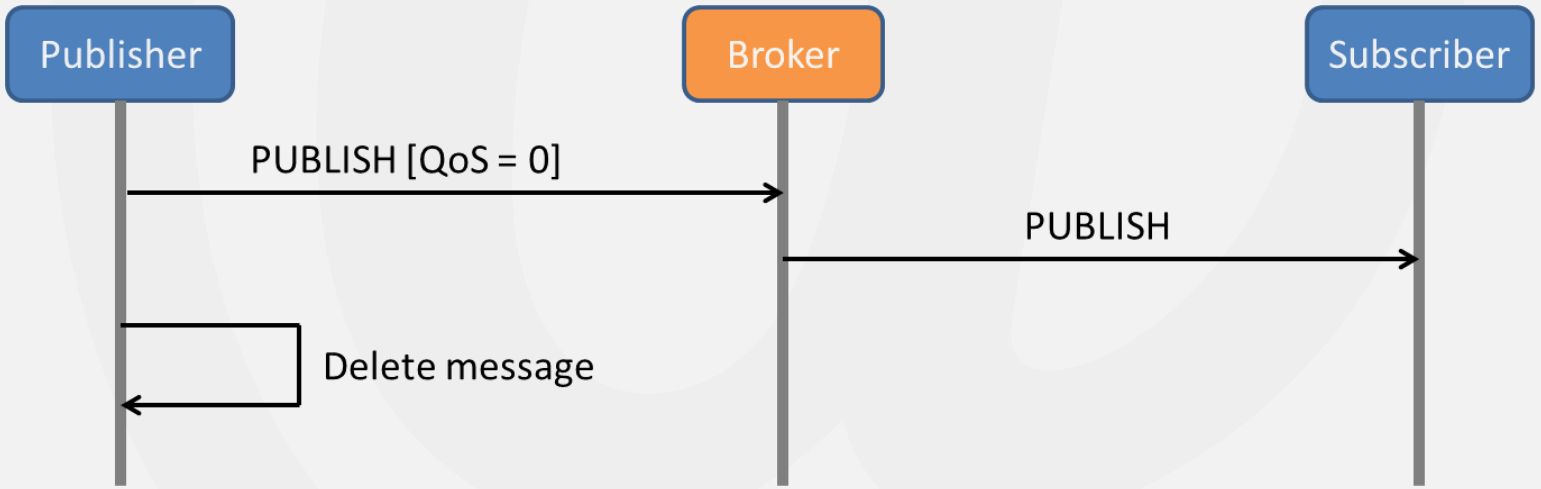
Topics for publish and subscribe

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



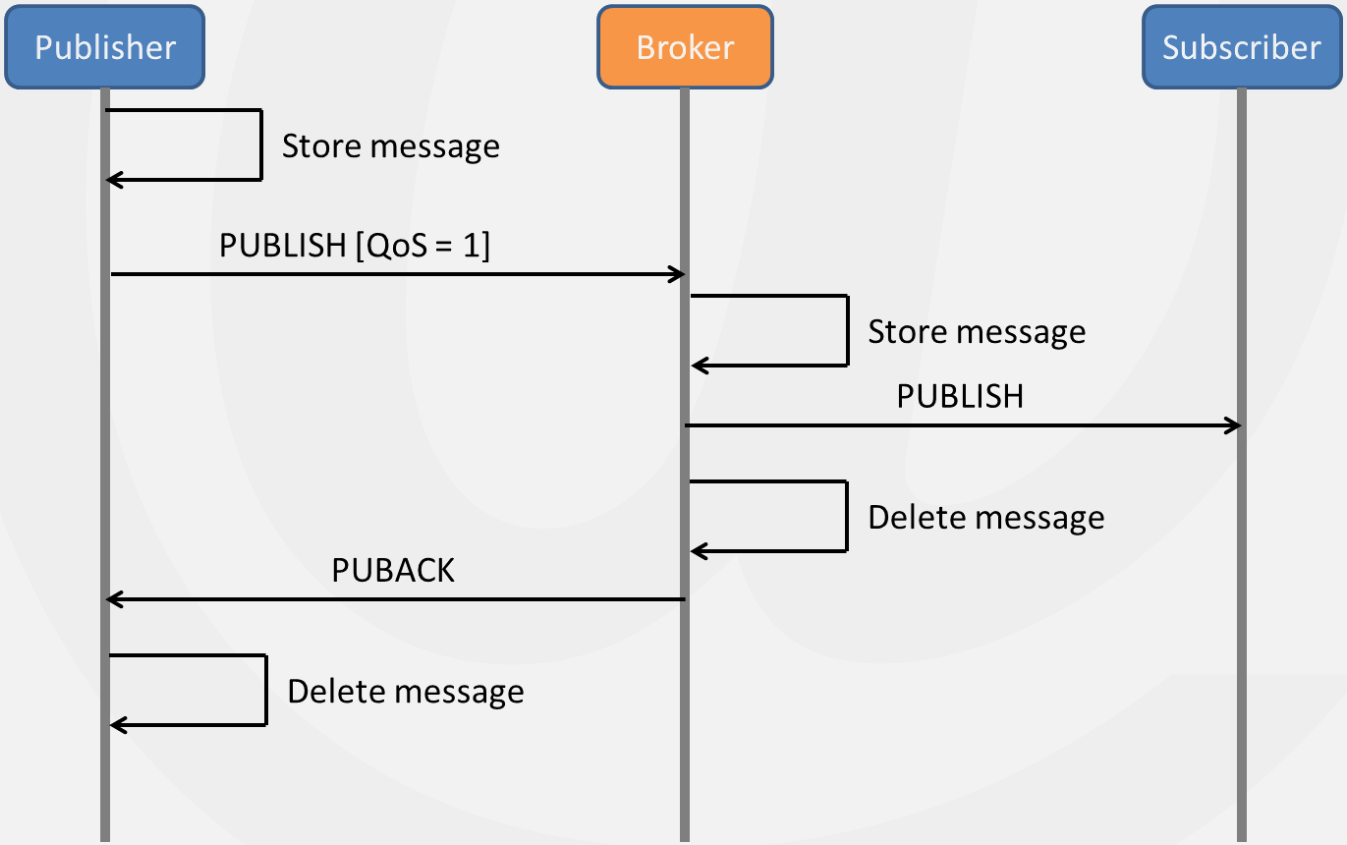
MQTT : Quality of Service

QoS 0 : At most once (fire and forget)



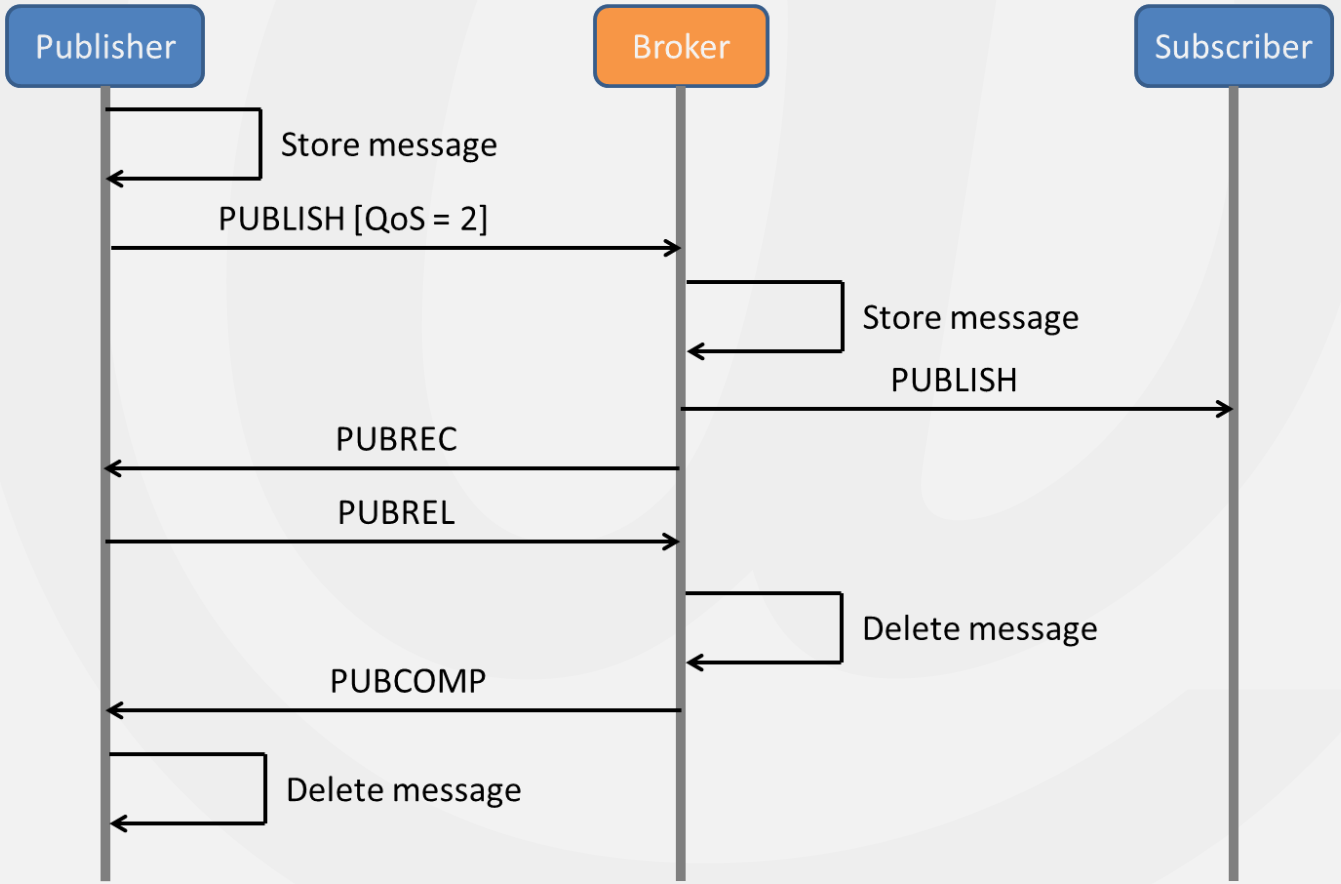
MQTT : Quality of Service

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 connection

encrypt payload (MQTT is payload agnostic)

MQTT : basic and advanced features

Keep-Alive message

- PINGREQ/PINGRESP message
- 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

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

Clean session

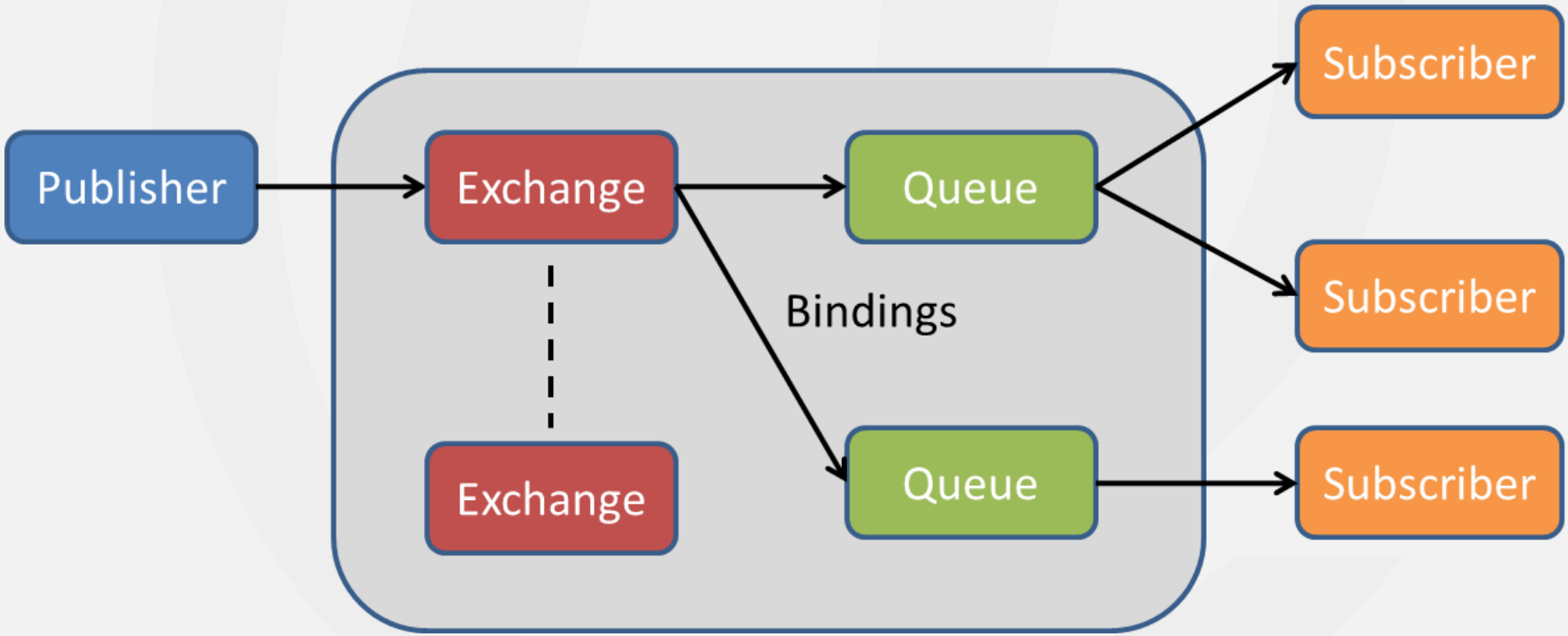
- on client disconnection, all subscriptions are kept
- no need to re-subscribe on re-connection client receives all messages published when offline

AMQP (0.9.1)

- 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 (0.9.1) : broker architecture

AMQP Model



AMQP (0.9.1) : 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 queue (pub/sub) and not use a routing key

AMQP (0.9.1) : exchanges

topic exchange

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

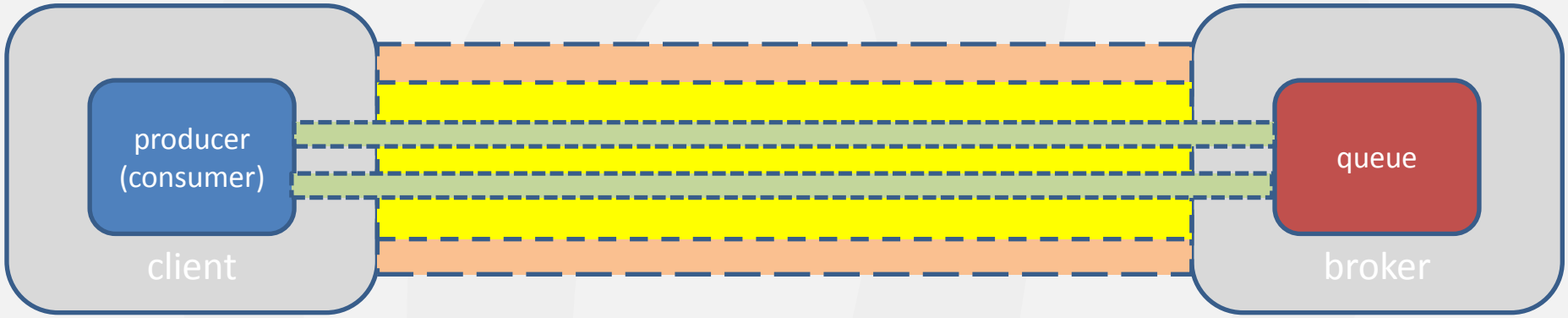
header exchange

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

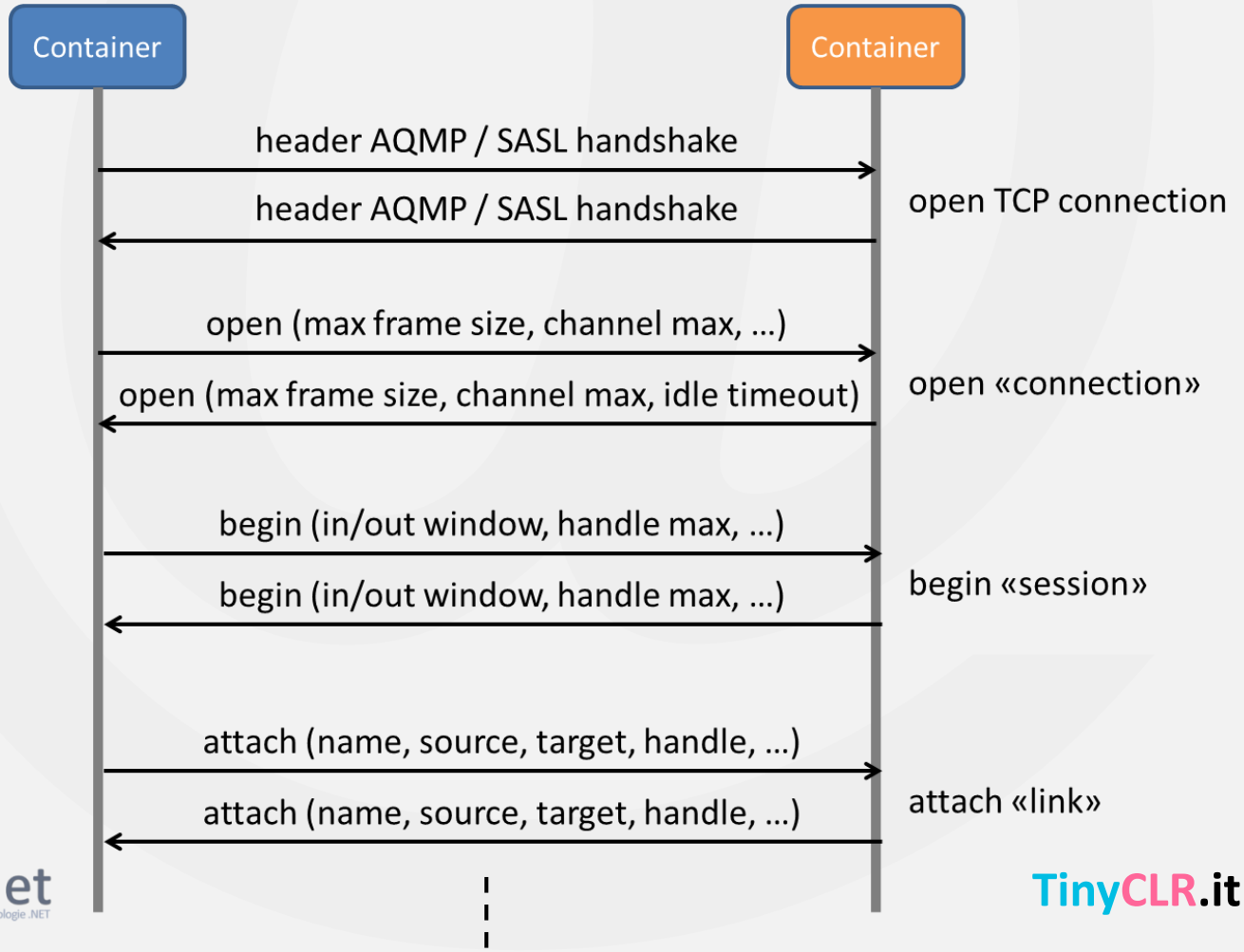
connection
session
link

container
node

multiplexing frames on sessions and links
transport indipendent

AMQP (1.0) : communication

open connection/session/link



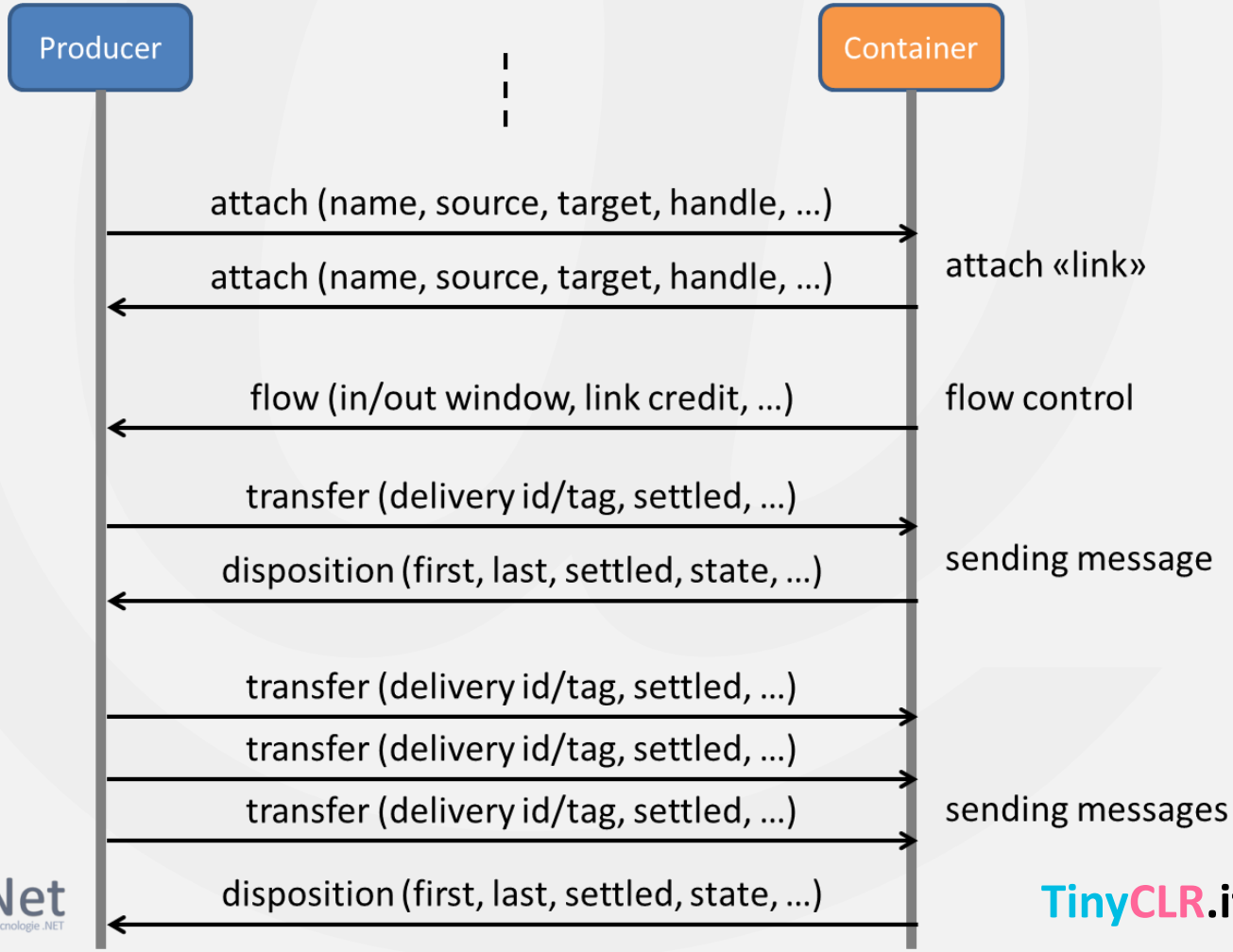
channel ⇔ session
handle ⇔ link

DomusDotNet
User Group romano per lo studio e la divulgazione delle tecnologie .NET

TinyCLR.it

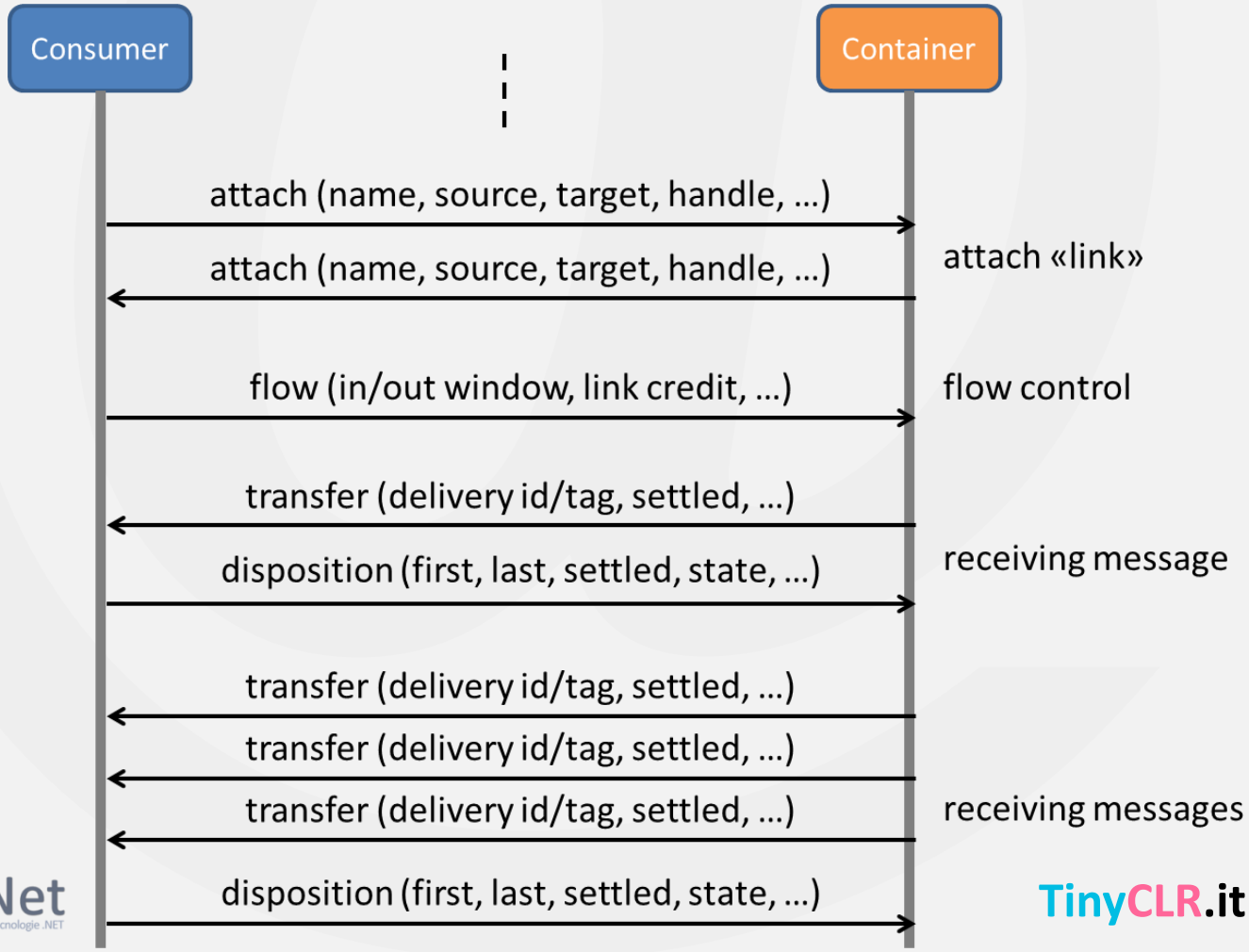
AMQP (1.0) : communication

send messages (ex. producer to queue)



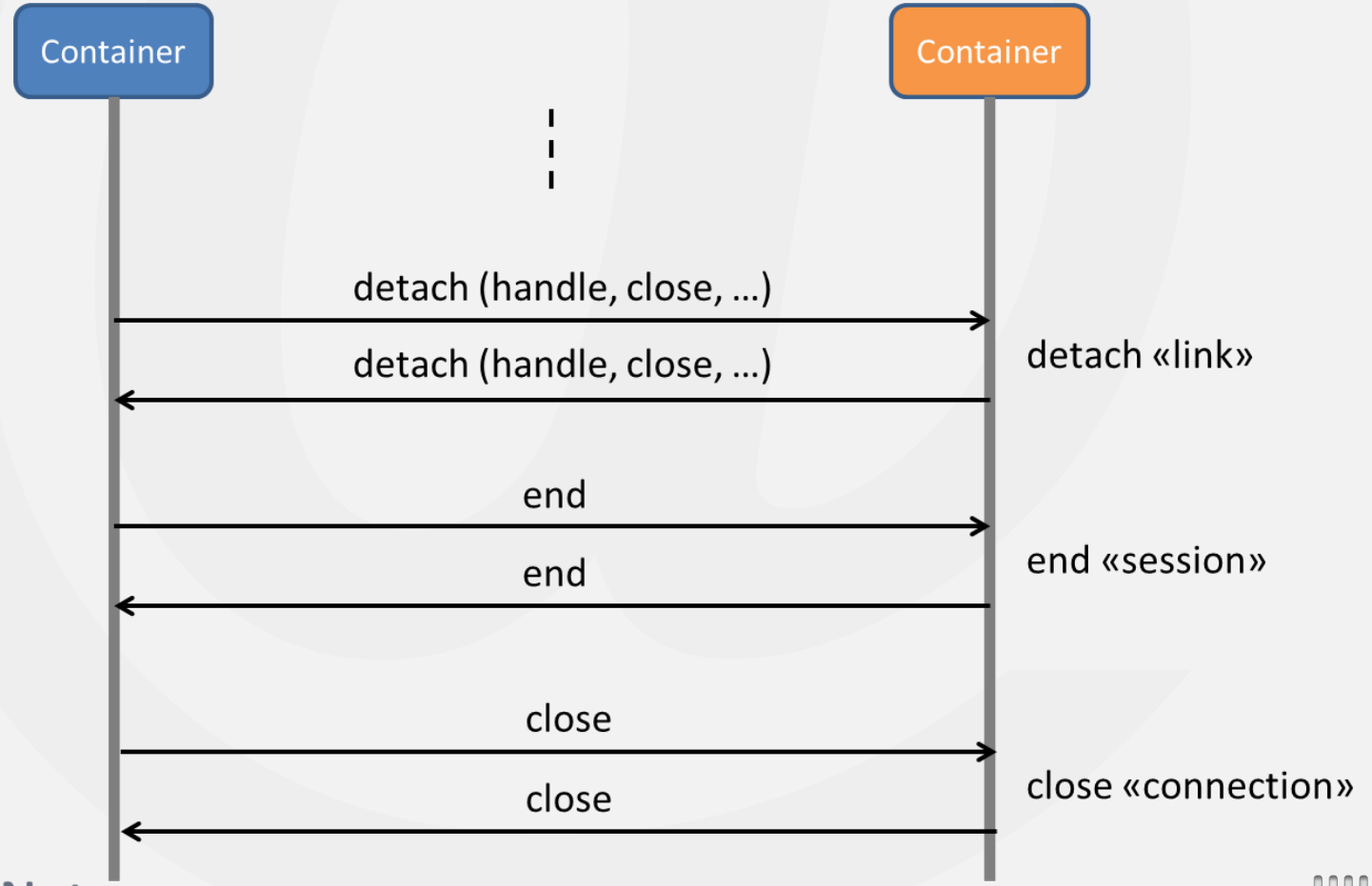
AMQP (1.0) : communication

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 : produce 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

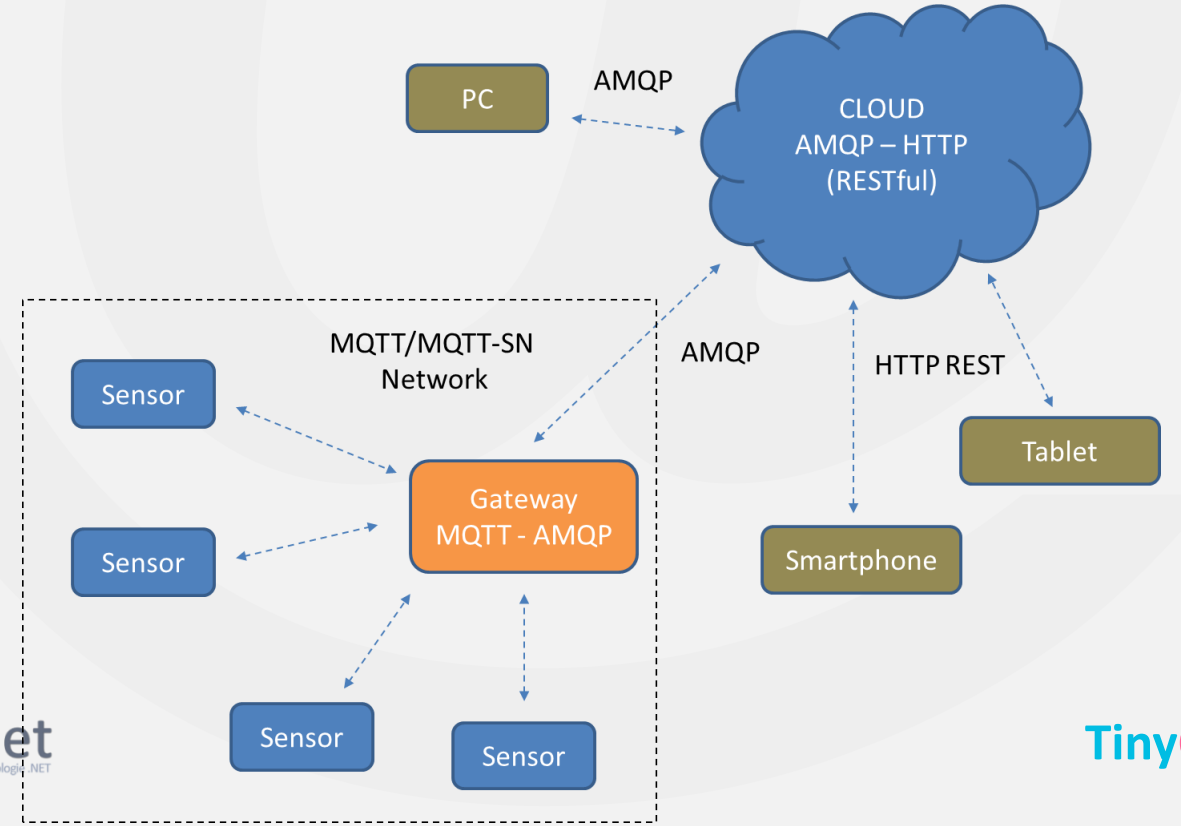
- devices
  - consider how much they are constrained
- network
  - how much it is reliable
- messages rate
  - how many messages per second and QoS
- process data
  - needs of the system to process data (metadata ?)

Conclusions

protocol choice depends on 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/

Grazie agli sponsor

Microsoft

telerik

deliver more than expected

pluralsight

hardcore developer training

O'REILLY®

intel

Software

DomusDotNet

User Group romano per lo studio e la divulgazione delle tecnologie .NET

TinyCLR.it

Per voi sono solo 10 minuti persi, per noi è utilissimo per poter crescere e migliorare!

https://it.surveymonkey.com/s/6QLNMMV

DomusDotNet
User Group romano per lo studio e la divulgazione delle tecnologie .NET

TinyCLR.it