Internet of Things

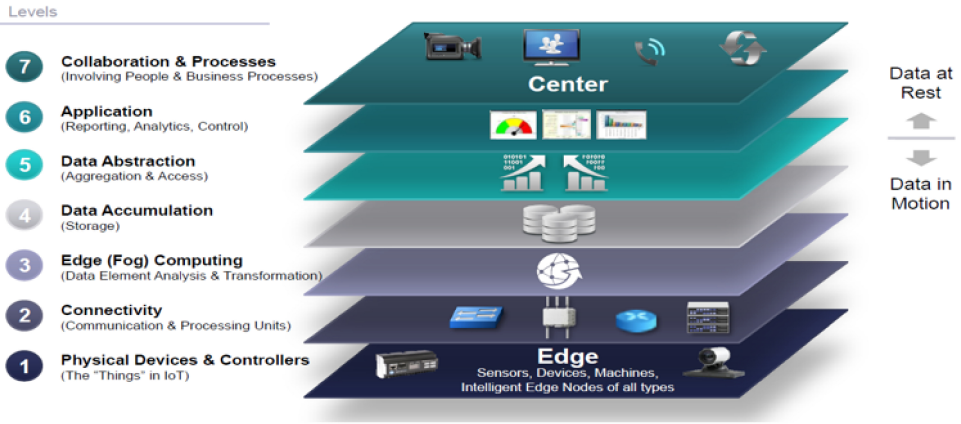
Protocols for Data Retrieval

Ioannis Chatzigiannakis

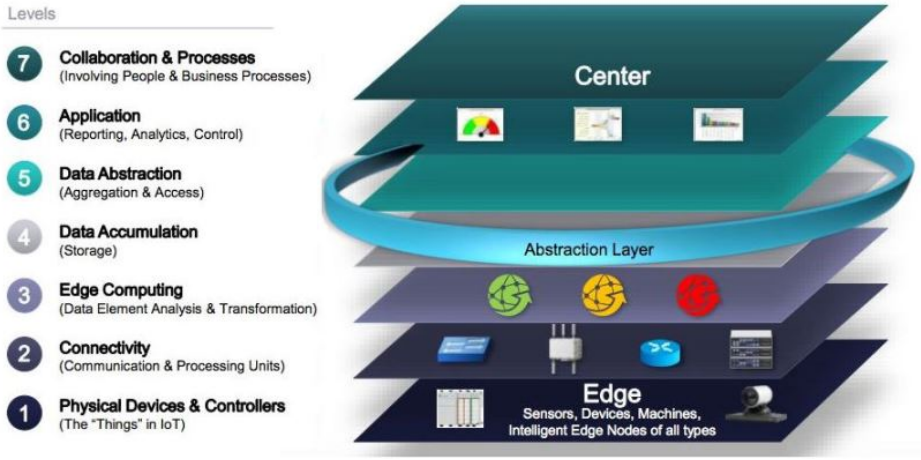
Sapienza University of Rome
Department of Computer, Control, and Management Engineering (DIAG)

Lecture 5:
IoT Protocols for Data Retrieval

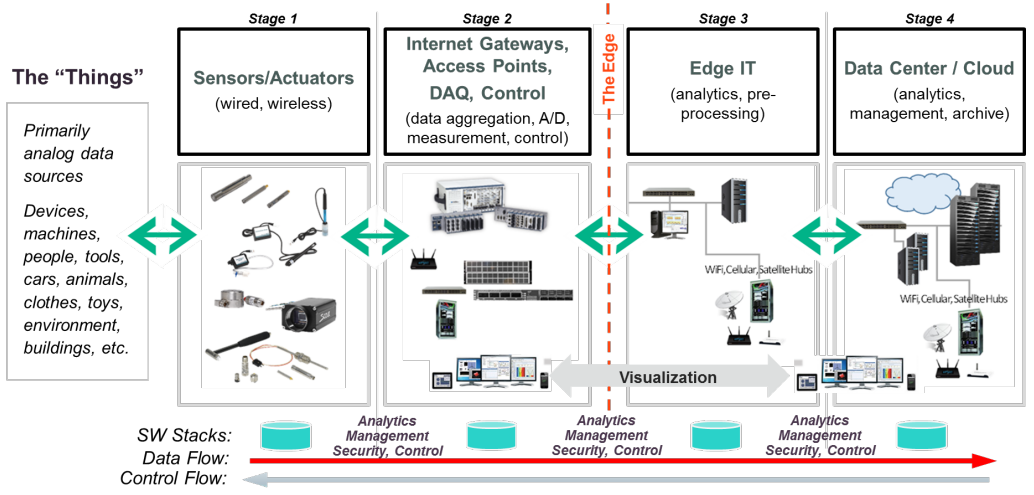
Main Architectural Levels



Main Architectural Levels



Main Components and Processing Stages



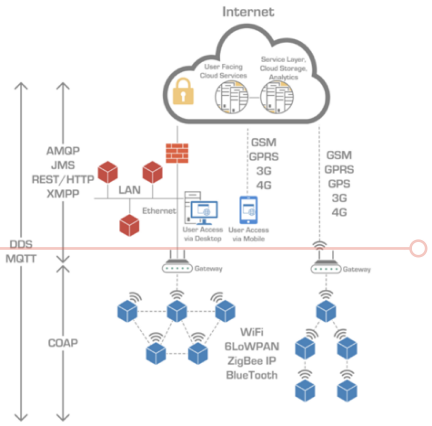
Components, Processing Stages and Protocols

Stage4: Data-Centers and Cloud

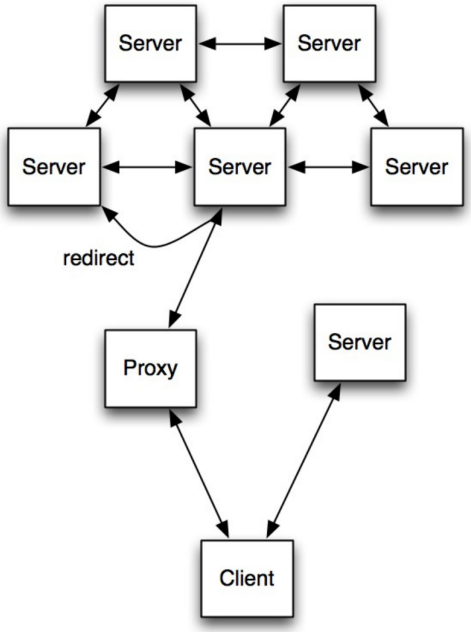
Stage3: Edge IT

Stage2: Internet GW & data acquisition/ aggregation /control systems

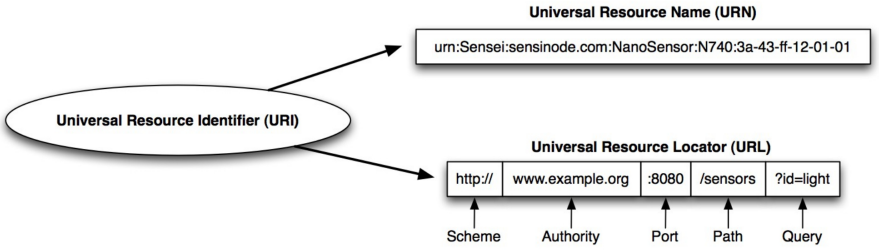
Stage1: sensors & Actuators



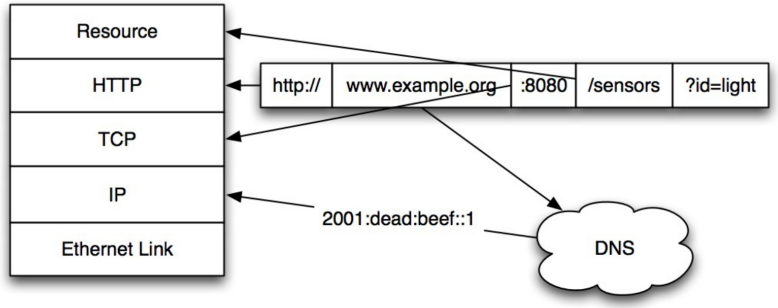
The Web and REST



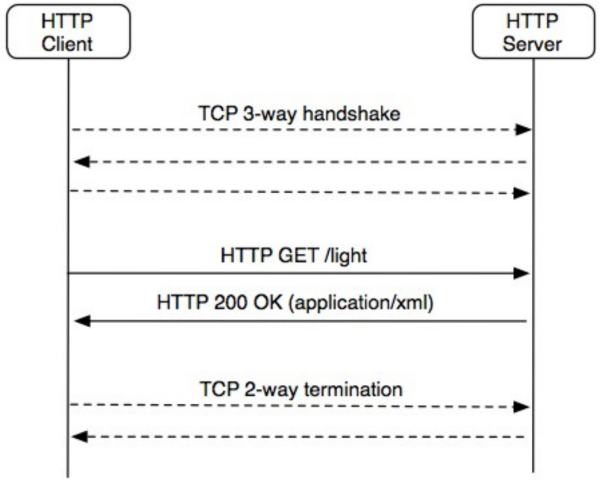
Web Naming



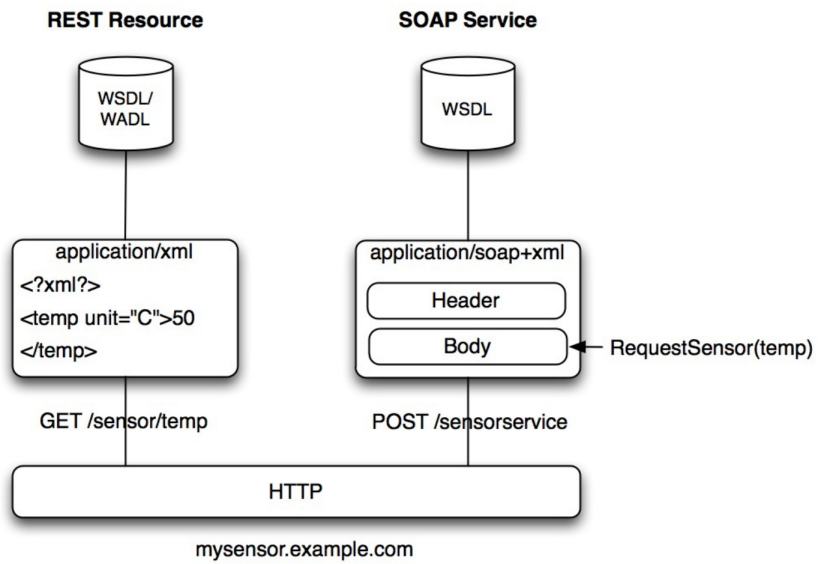
URL Resolution



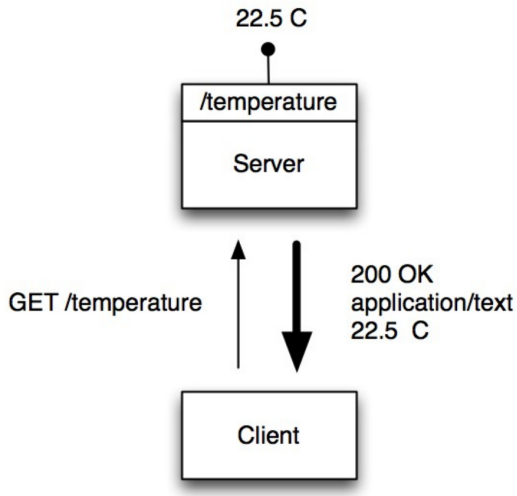
An HTTP Request



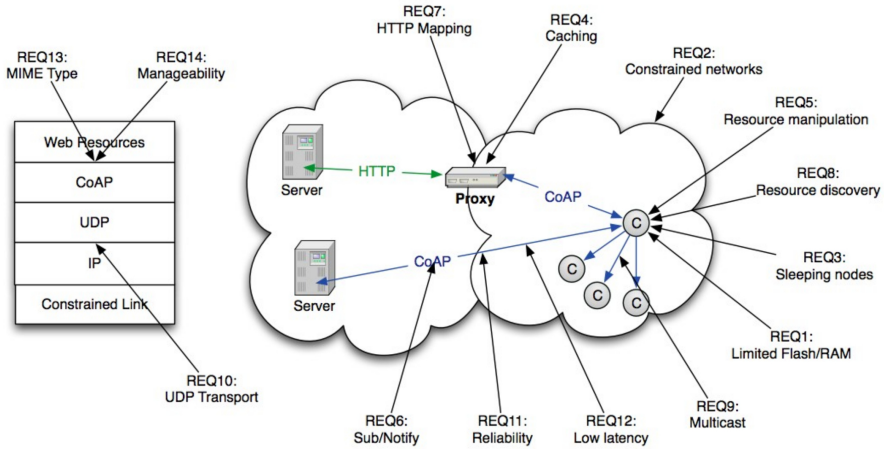
Web Paradigms



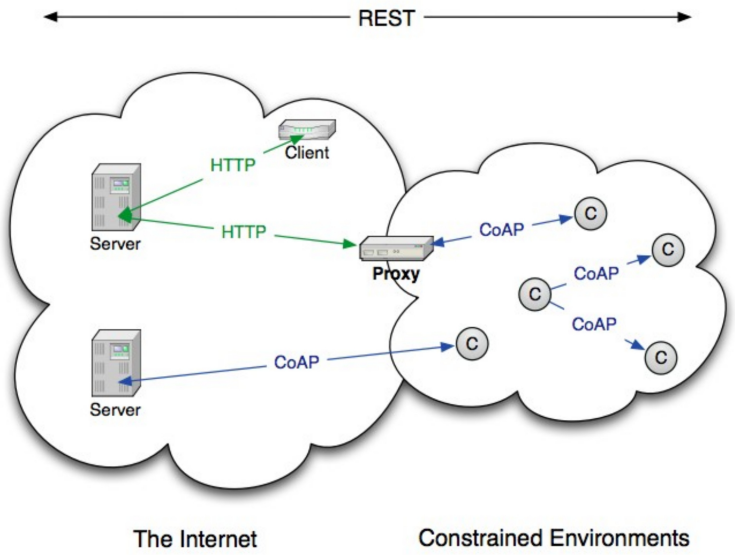
A REST Request



CoAP Design Requirements



The CoAP Architecture



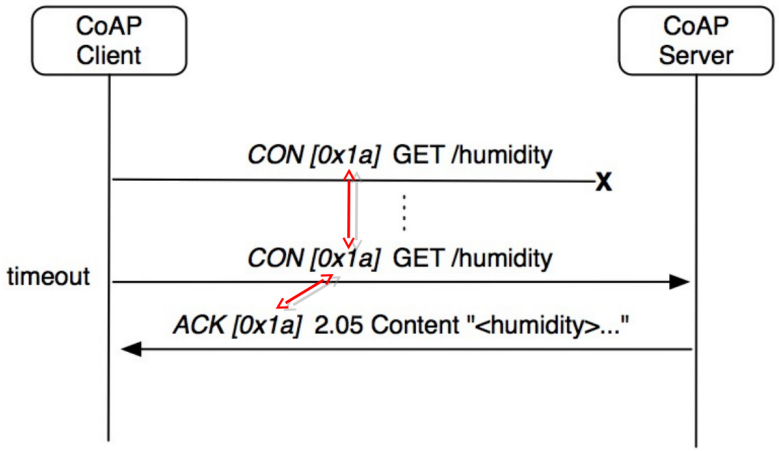
The CoAP Protocol

- A very efficient RESTful protocol
- Ideal for constrained devices and networks
- Specialized for M2M applications
- Easy to proxy to/from HTTP
- Does not replace HTTP
- Is not a cut-down HTTP version
- Not just for resource-constrained networks

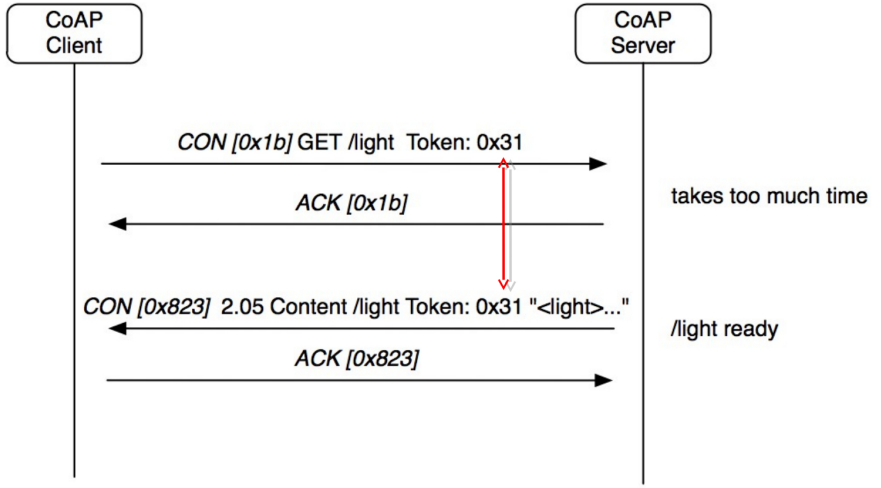
CoAP Features

- Embedded web transfer protocol (coap://)
- Asynchronous transaction model
- UDP binding with reliability and multicast support
- GET, POST, PUT, DELETE methods
- URI support
- Small, simple 4 byte header
- DTLS based PSK, RPK and Certificate security
- Subset of MIME types and HTTP response codes
- Built-in discovery
- Optional observation and block transfer

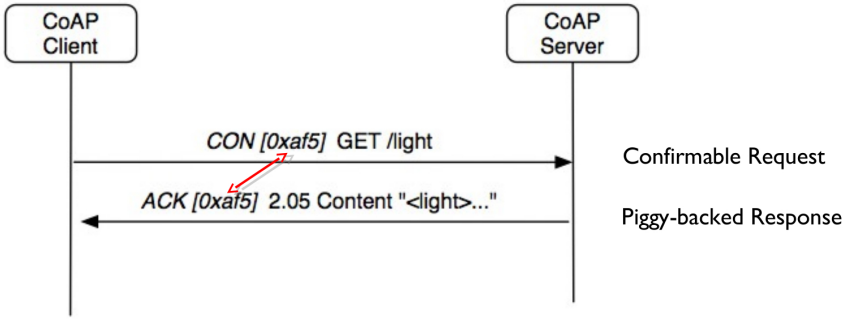
Request Example



Separating Response and Acknowledgement



Dealing with Packet Loss



A Hands-on Example using libCoap

- libCoap: C-Implementation of CoAP `https://libcoap.net/`
- Simple command-line server included for testing:

```
coap-server -A 127.0.0.1 -p 13001
```

- Simple command-line client included for testing:

```
coap-client coap://127.0.0.1:13001/.well-known/core
</>;title="General_Info";ct=0,</time>;if="clock";rt="ticks";title="Internal_Clock"
```

Various implementations available on GitHub

- CoAPthon: a python library to the CoAP protocol aligned with the RFC – `https://github.com/Tanganelli/CoAPthon`
- Simple client in python:

```
from coapthon.client.helperclient import HelperClient

host = "127.0.0.1"
port = 13001
path ="/"

client = HelperClient(server=(host, port))
response = client.get(path)
print(response.pretty_print())
client.stop()
```

- Interroperable with libCoap

```
Source: ('127.0.0.1', 13001)
Type: ACK
MID: 11490
Code: CONTENT
Token: rK
Content-Type: 0
Max-Age: 196607
Payload:
This is a test server made with libcoap (see https://libcoap.net)
```

CoAP server resource in Python + CoAPthon

```
from coapthon.server.coap import CoAP
from coapthon.resources.resource import Resource

class BasicResource(Resource):
    def __init__(self, name="BasicResource", coap_server=None):
        super(BasicResource, self).__init__(name, coap_server, visible=True,
                                            observable=True, allow_children=True)
        self.payload = "Basic_Resource"

    def render_GET(self, request):
        return self

    def render_PUT(self, request):
        self.payload = request.payload
        return self

    def render_POST(self, request):
        res = BasicResource()
        res.location_query = request.uri_query
        res.payload = request.payload
        return res

    def render_DELETE(self, request):
        return True
```

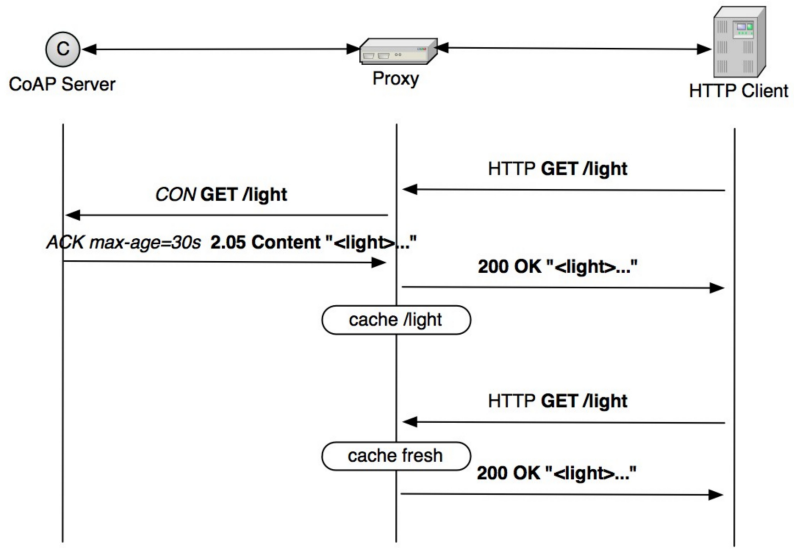
CoAP server in Python + CoAPthon

```
from coapthon.server.coap import CoAP
from coapthon.resources.resource import Resource

class CoAPServer(CoAP):
    def __init__(self, host, port):
        CoAP.__init__(self, (host, port))
        self.add_resource('basic/', BasicResource())

def main():
    server = CoAPServer("0.0.0.0", 5683)
    try:
        server.listen(10)
    except KeyboardInterrupt:
        print("Server_Shutdown")
        server.close()
        print("Exiting...");

if __name__ == '__main__':
    main()
```

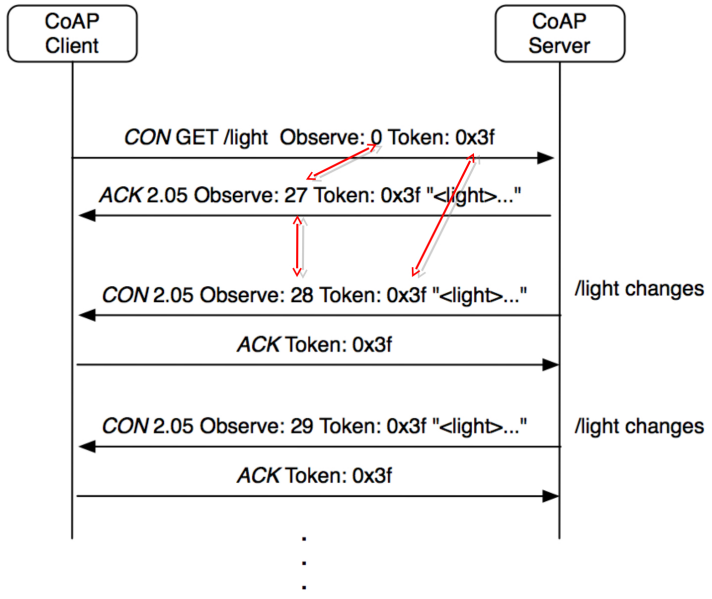
Interroperable with libCoap command-line client:

```
coap-client coap://127.0.0.1:5683/basic
Basic Resource
```

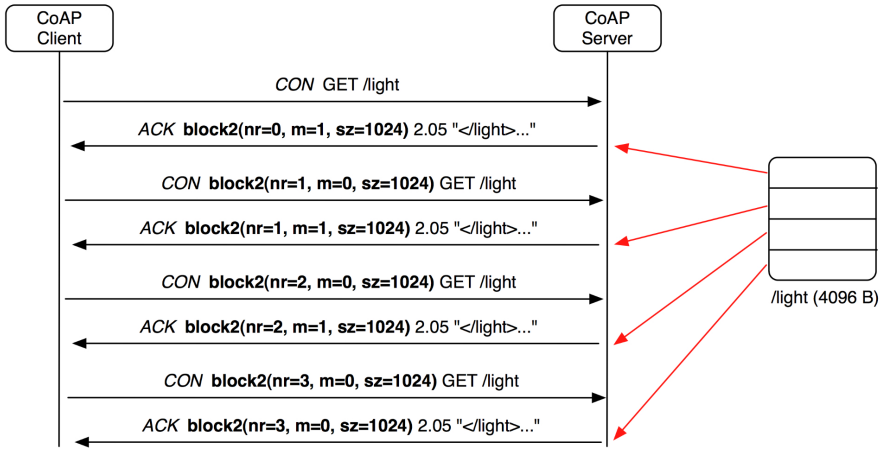
Proxying and caching



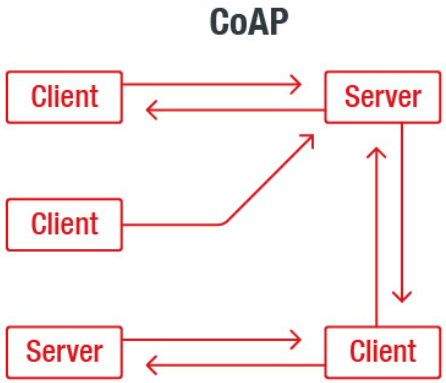
Observation



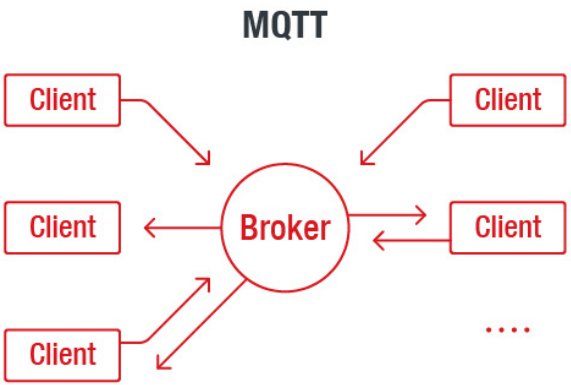
Block transfer

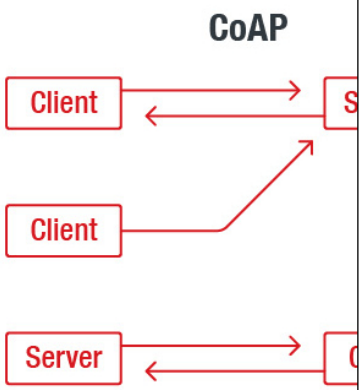


One-to-One

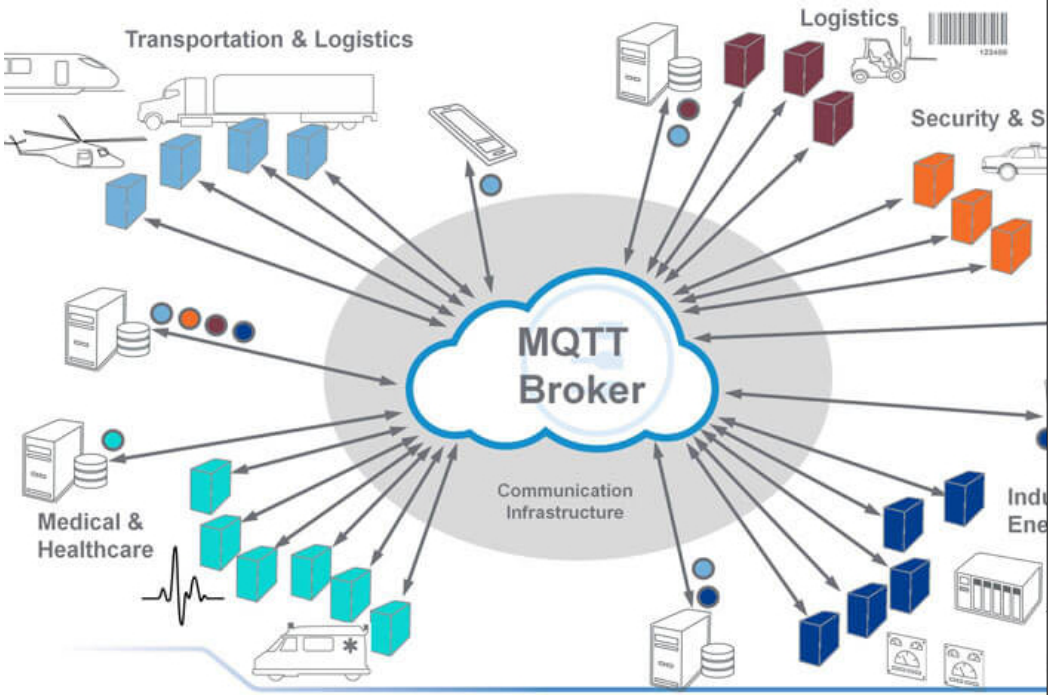


One-to-One vs Many-to-Many

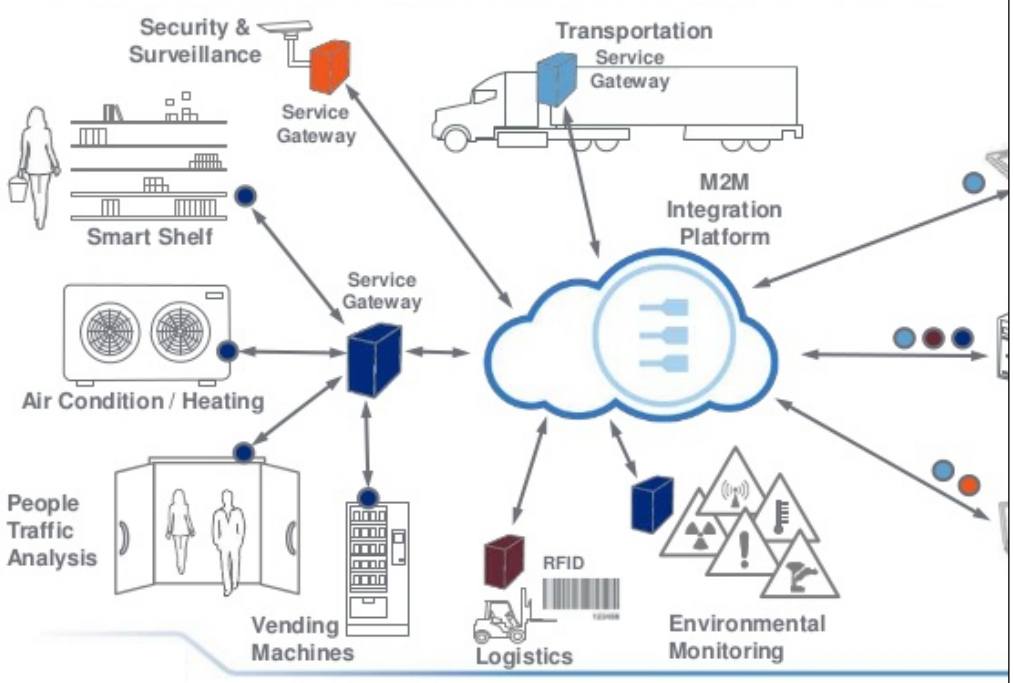




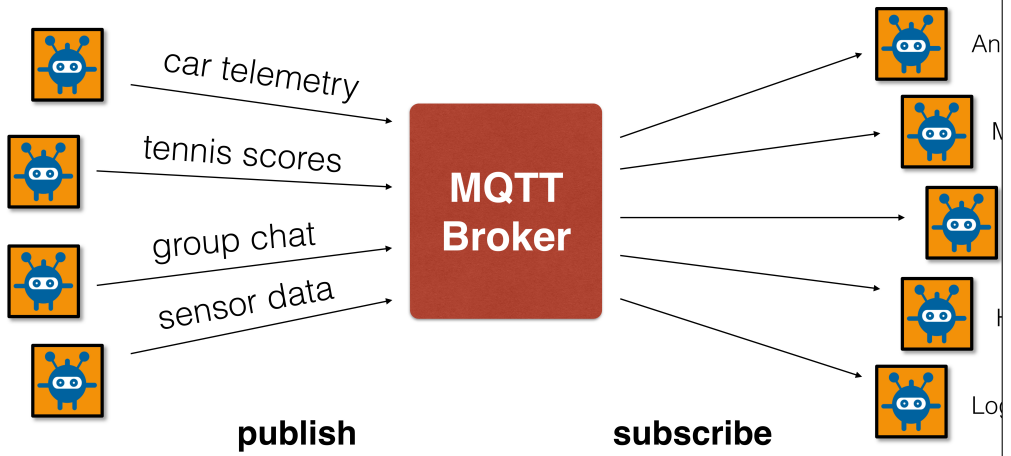
Decoupling Produces and Consumers



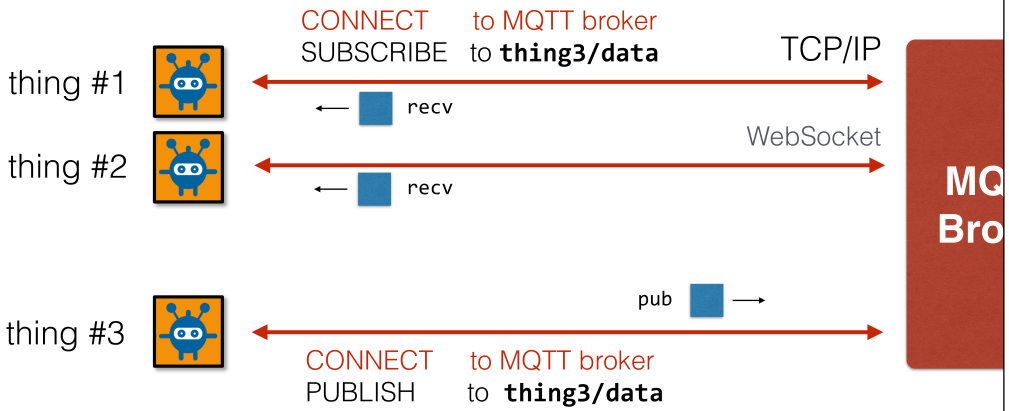
Application Examples



Publish - Subscribe Paradigm

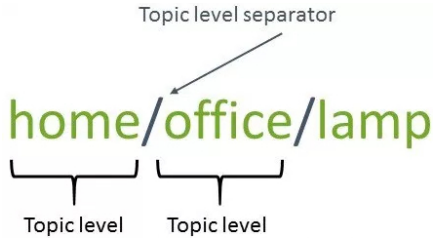


Bi-directional, asynchronous "push" communication

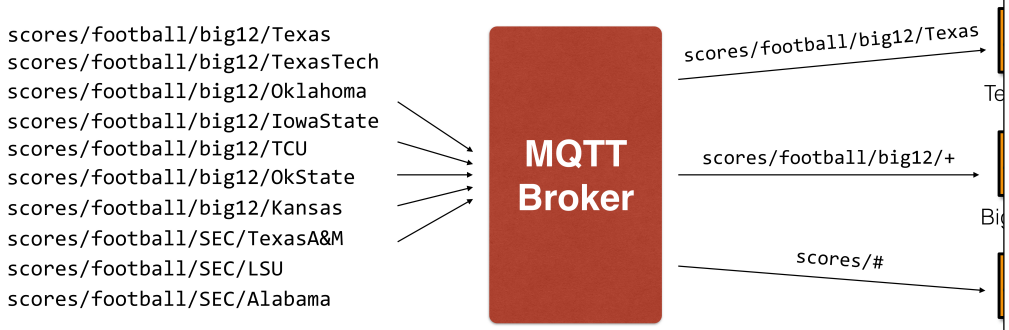


Topic-based communication

- Topics register interest for incoming messages.
- Specify where to publish messages.
- Topics are represented with strings separated by a forward slash.
- Each forward slash indicates a topic level.

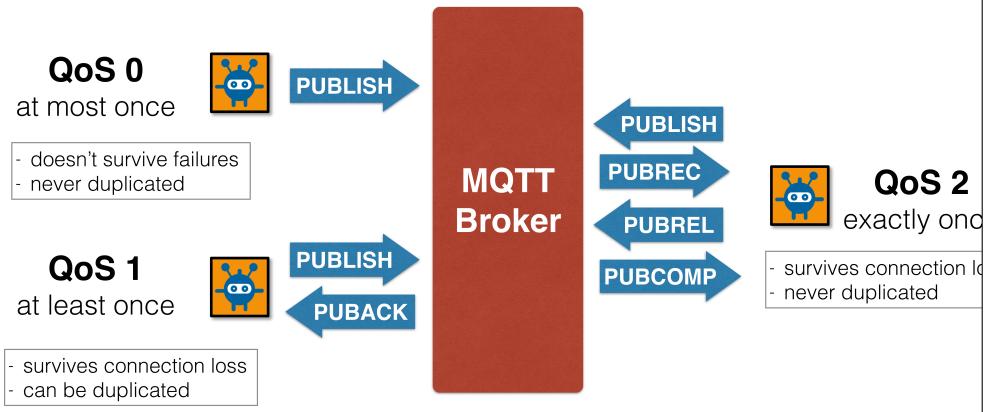


Multi-level Subscriptions

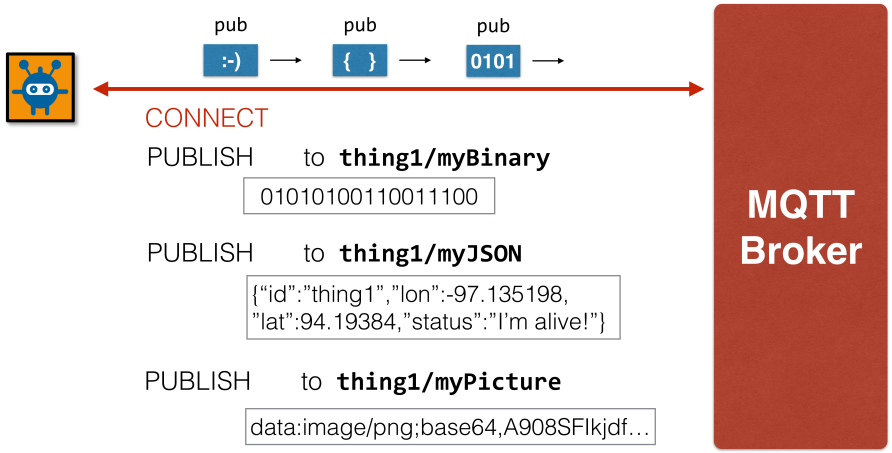


single level wildcard: **+** multi-level wildcard: **#**

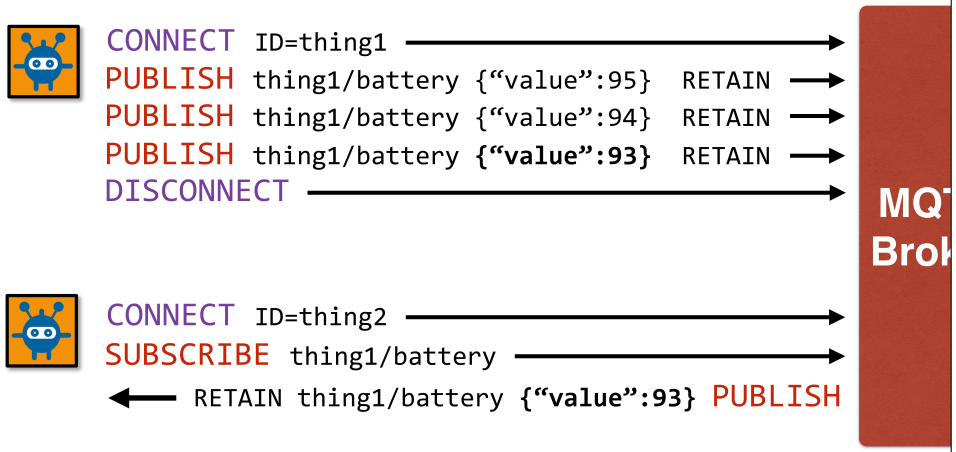
Quality of Service for reliable messaging



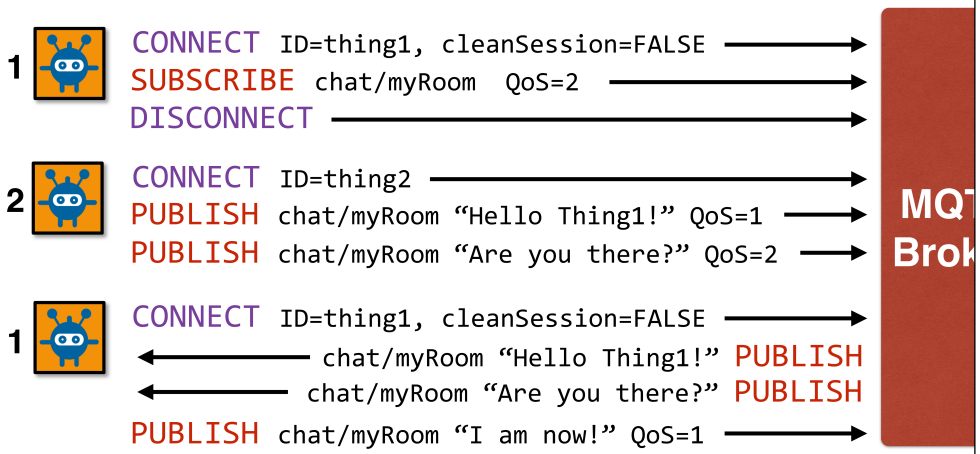
Agnostic payload for flexible delivery



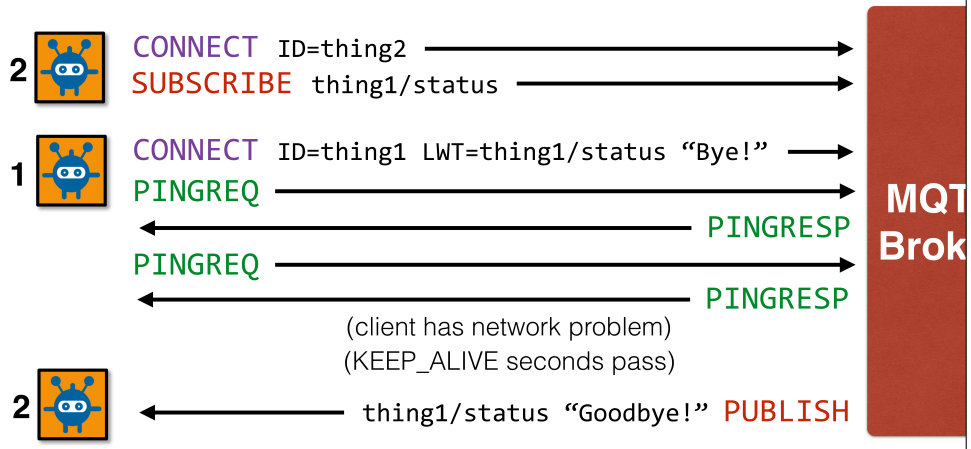
Retained messages for last value caching



Session state control



Last will and testament for presence



Setting up a MQTT Broker

- Several MQTT brokers are available to install locally
  - Eclipse Mosquitto: an open source implementation
    `https://mosquitto.org/`
- Many cloud-based MQTT brokers available
  - CloudMQTT: a hosted broker
    `https://www.cloudmqtt.com/`
  - HiveMQ – both local or clour-based
    `https://www.hivemq.com/`
- Adopted by all major cloud-providers: AWS, Azure, Google ...

Various implementations available on GitHub

- paho.mqtt: a python library to the MQTT protocol by Eclipse
  `https://pypi.org/project/paho-mqtt/`
- Simple client in python:

```
import paho.mqtt.client as mqtt

def on_connect(client, userdata, flags, rc):
    print("Connected with result code "+str(rc))

def on_message(client, userdata, msg):
    print(msg.topic+" "+str(msg.payload))

client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message

client.connect("localhost", 1883, 60)
client.subscribe("$SYS/#")

client.loop_forever()
```

- Interroperable with mosquitto / ...

```
Connected with result code 0
$SYS/broker/version b'mosquitto version 1.6.8'
$SYS/broker/uptime b'3399 seconds'
```