COMP6733 IoT Design Studio

CoAP, MQTT, Cloud Services for the IoT

This lecture

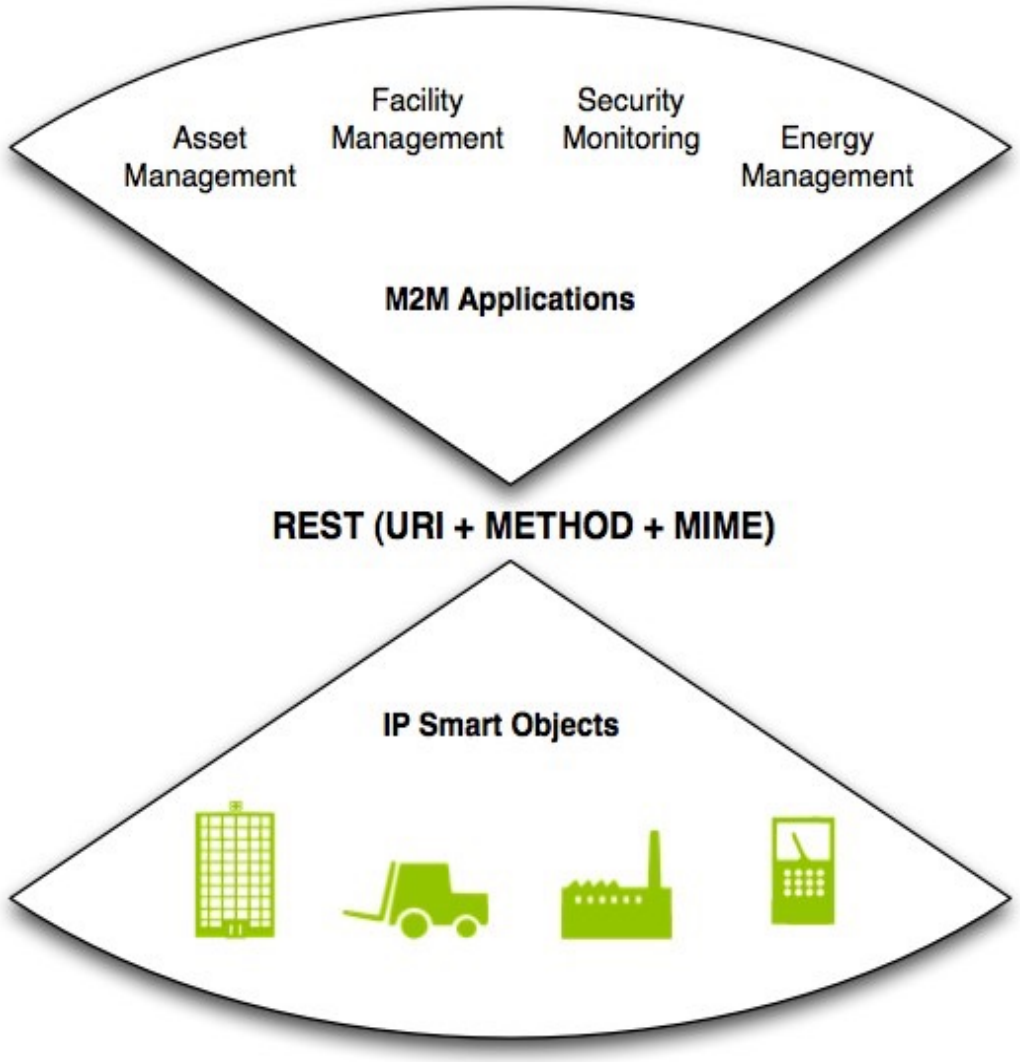
- The Constrained Application Protocol (CoAP)
- MQTT
- Cloud service for IoT

The Web of Things

Services

The Web

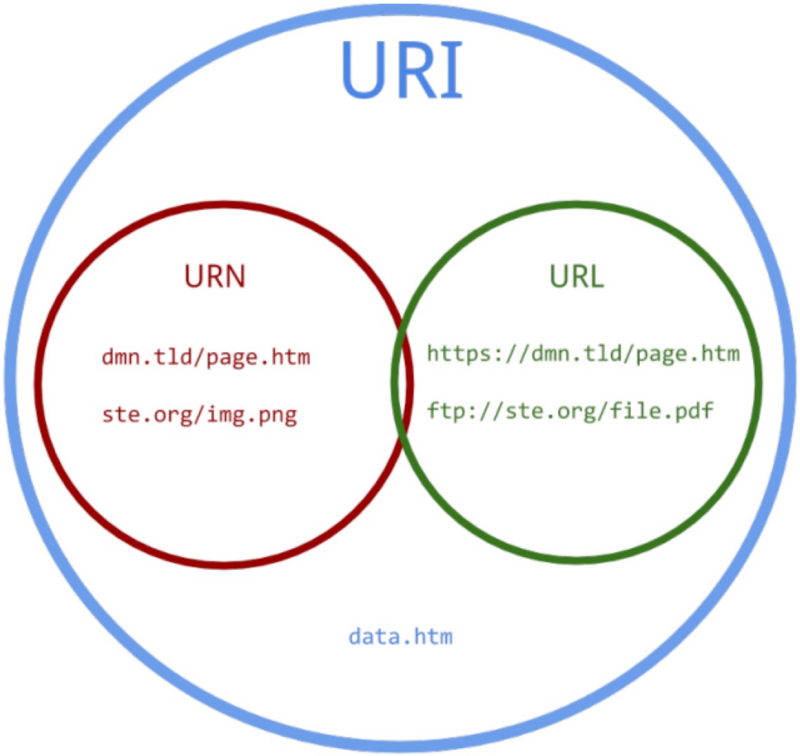
Things



Big Data

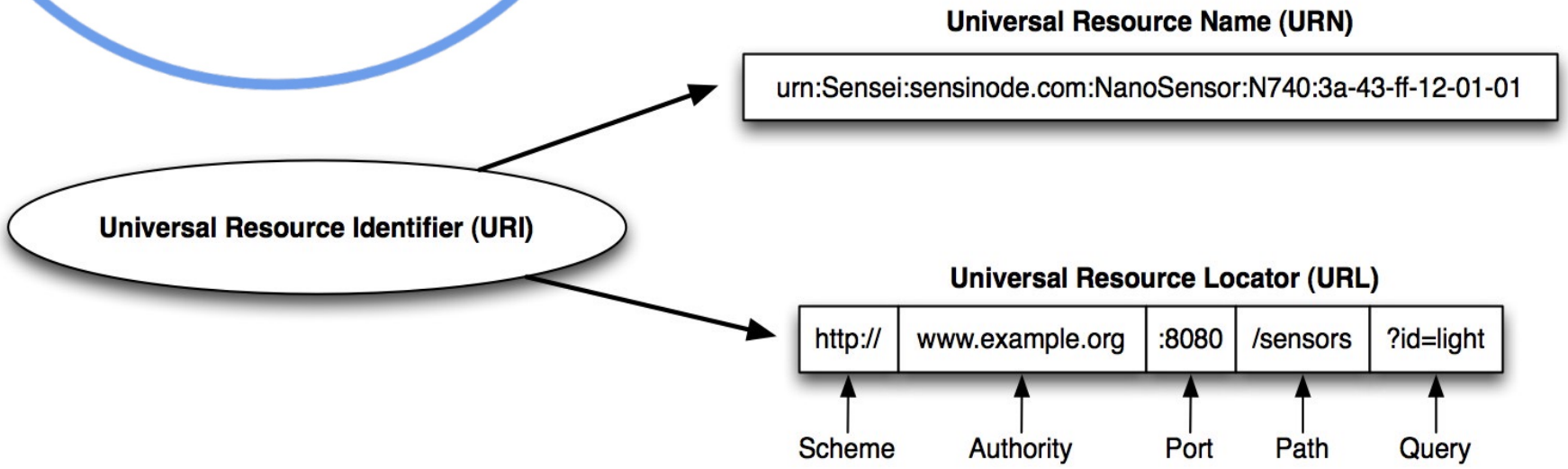
Little Data

Web Resource Identification

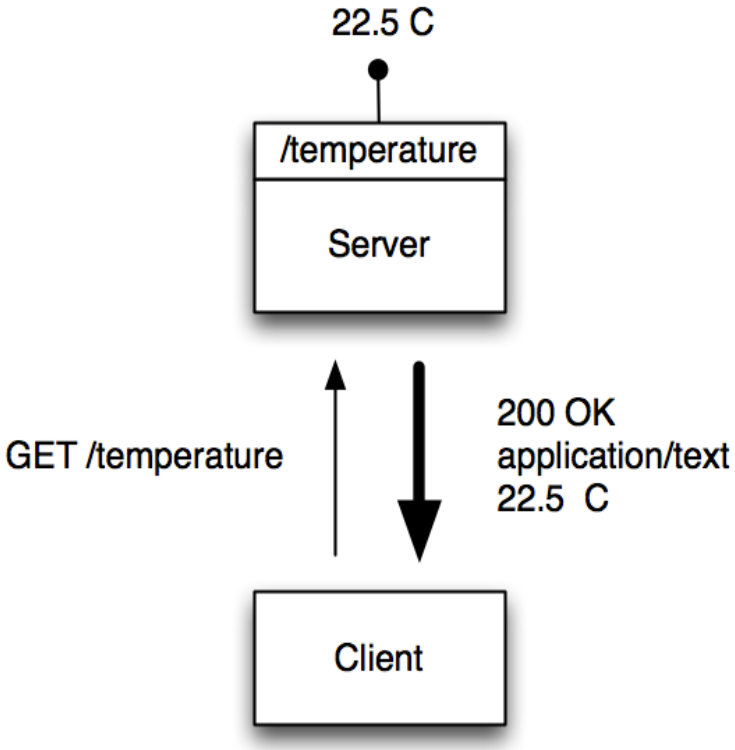


WIKIPEDIA CAPTURES THIS WELL with the following simplification:

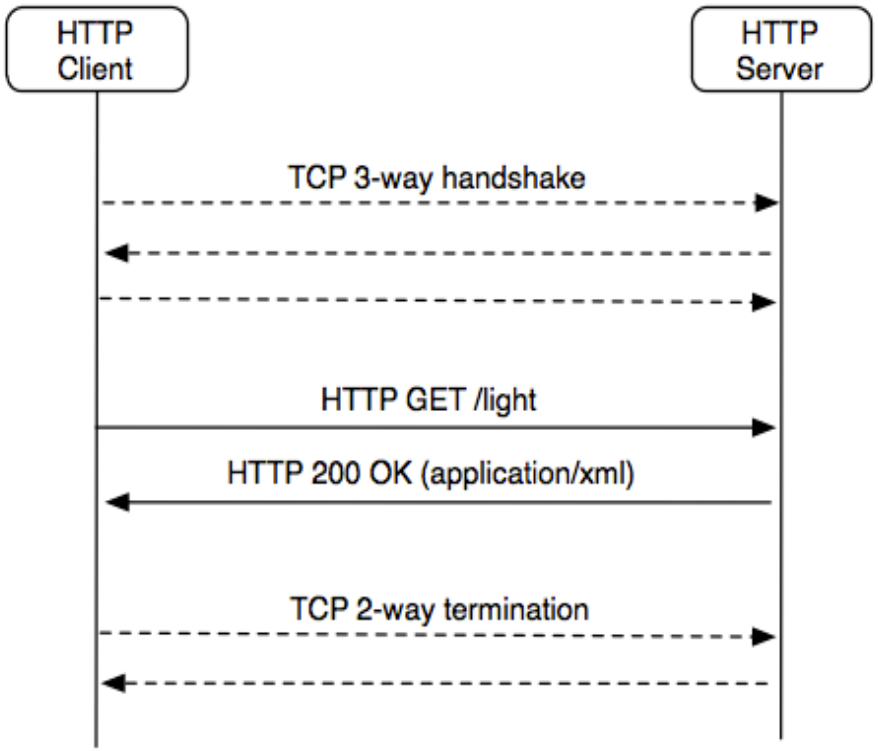
> One can classify URIs as locators (URLs), or as names (URNs), or as both. A Uniform Resource Name (URN) functions like a person's name, while a Uniform Resource Locator (URL) resembles that person's street address. In other words: the URN defines an item's identity, while the URL provides a method for finding it.



A REST Request



An HTTP Request



See RFC2616 - Hypertext Transfer Protocol v1.1

CoAP

RFC 7252 Constrained Application Protocol

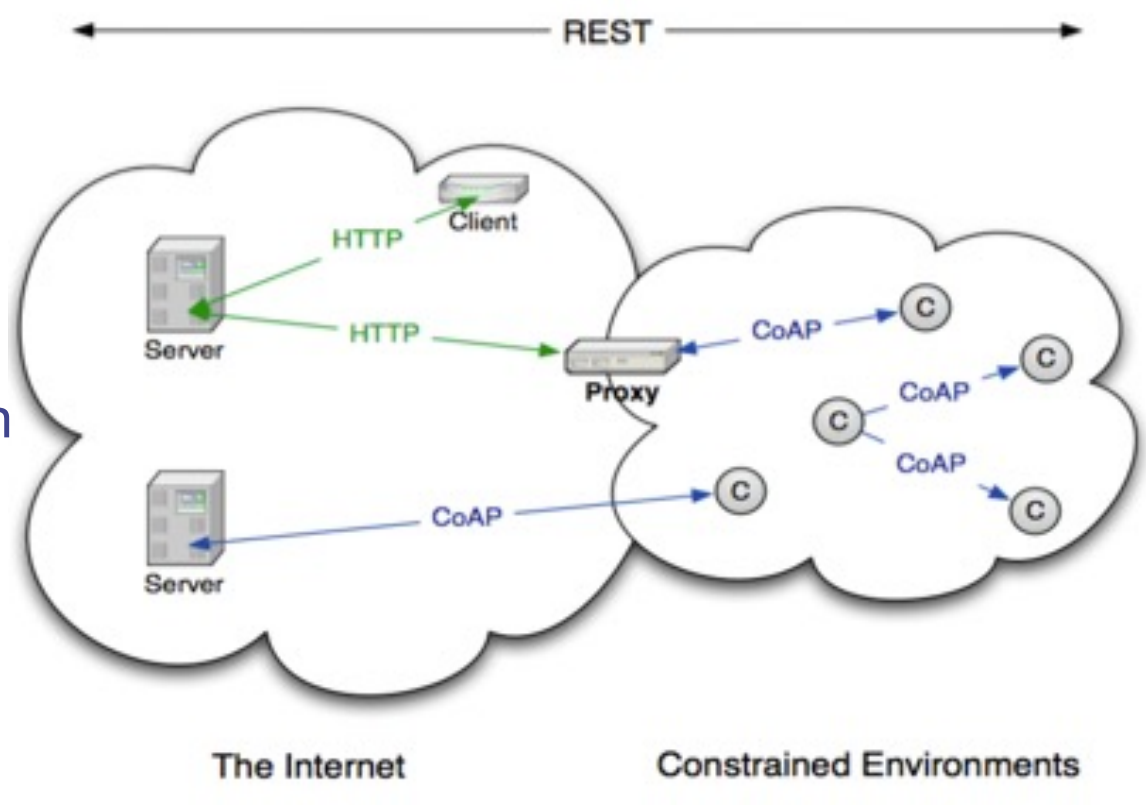
"The Constrained Application Protocol (CoAP) is a specialized web transfer protocol for use with constrained nodes and constrained networks in the **Internet of Things.**

The protocol is designed for machine-to-machine (M2M) applications such as smart energy and building automation."

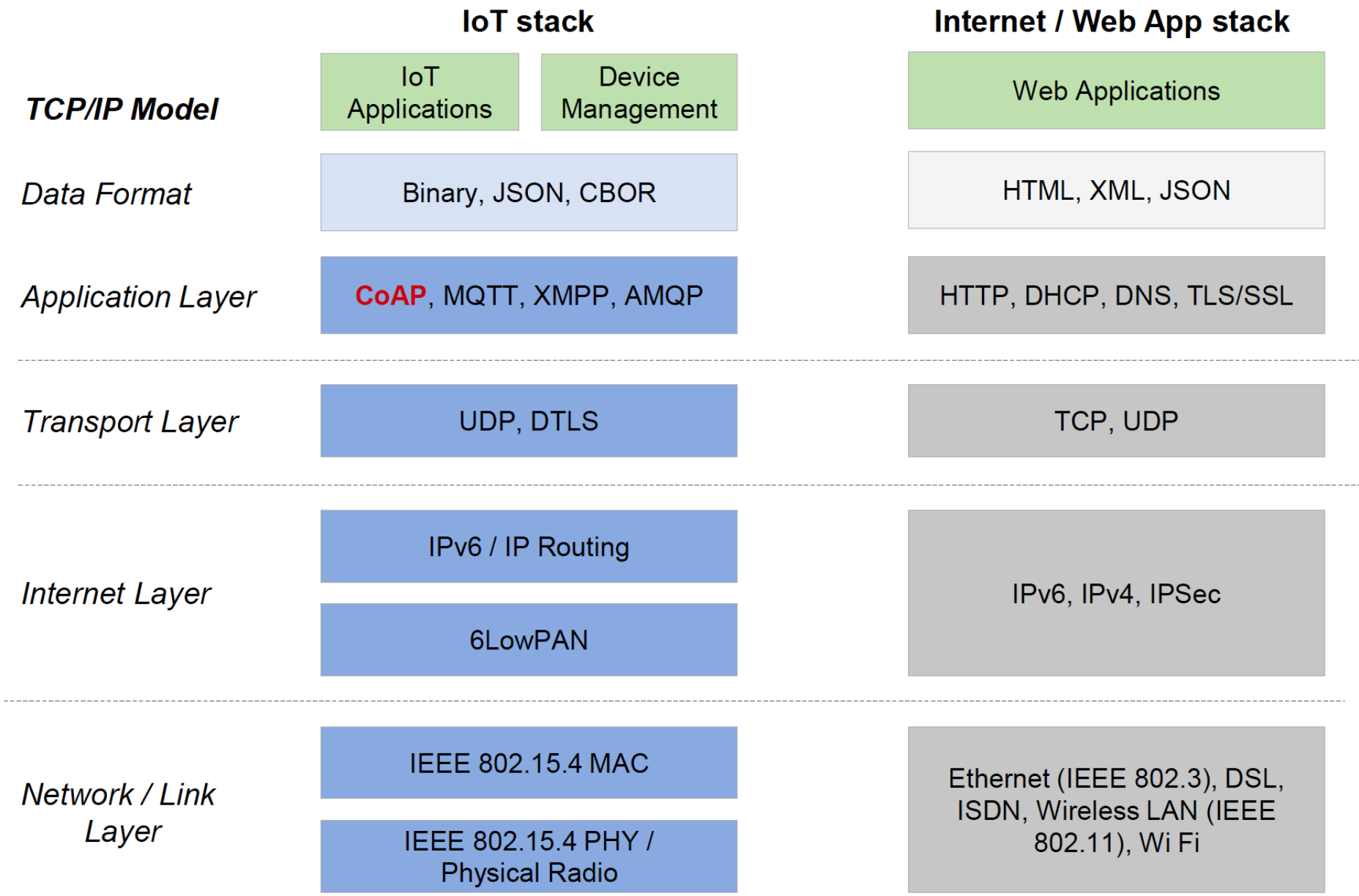
- CoAP is an application-layer protocol and follows the request-response pattern used by HTTP - CoAP has transparent mapping to HTTP
- CoAP uses familiar HTTP interactions like Methods (Get, Post, Put, Delete), Status Codes, URI, Content type/MIME
- Think of CoAP as HTTP REST for constrained environment
- Like HTTP, CoAP can carry different types of payloads. CoAP integrates with XML, JSON, CBOR, etc.

CoAP: The Web of Things Protocol

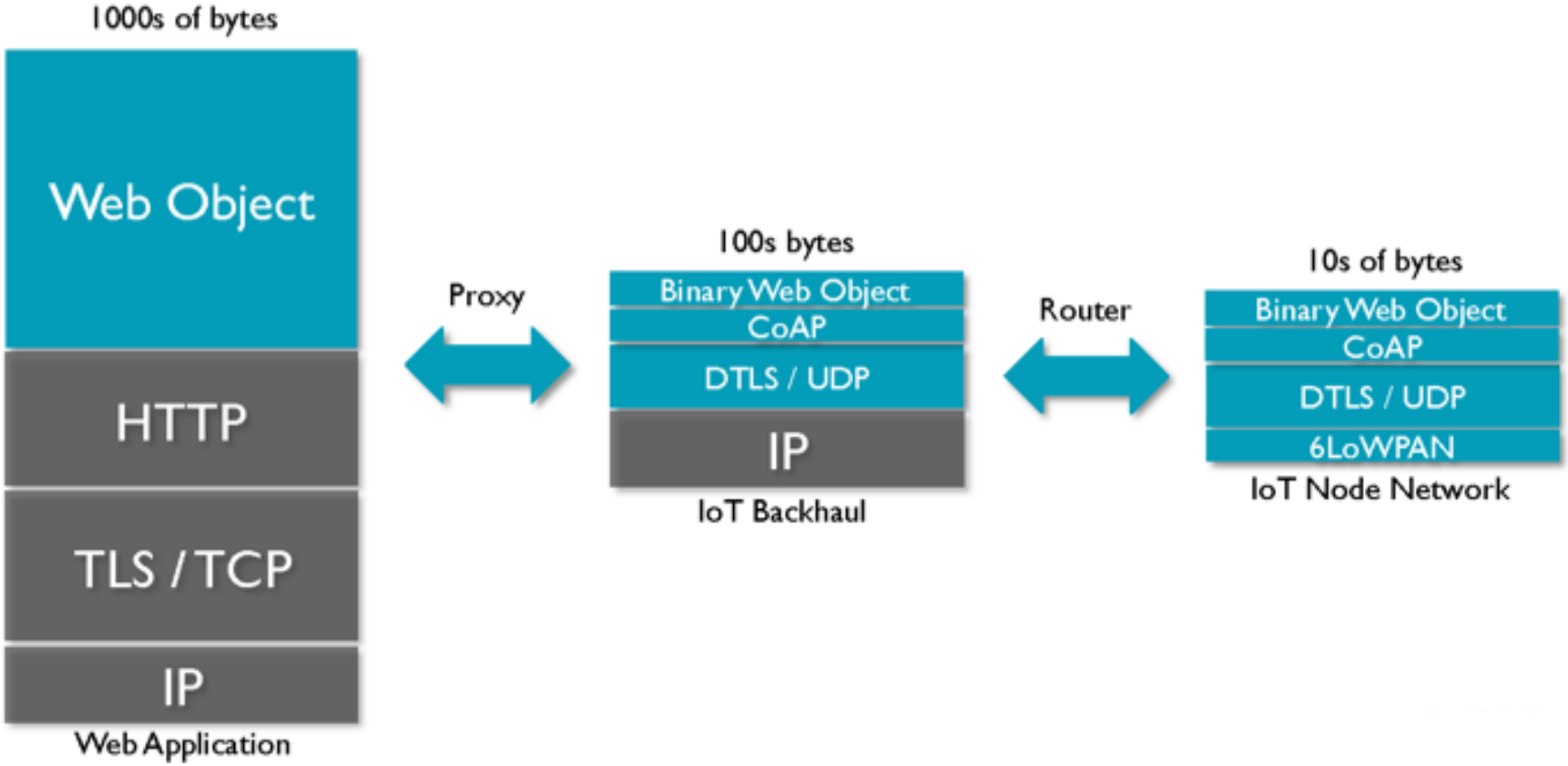
- Open IETF Standard (RFC 7272)
- Compact 4-byte header
- UDP, SMS, (TCP) Support
- Strong DTLS Security
- Asynchronous Subscription
- Built-in Discovery



IoT Protocol Stack



From Web Applications to IoT Nodes



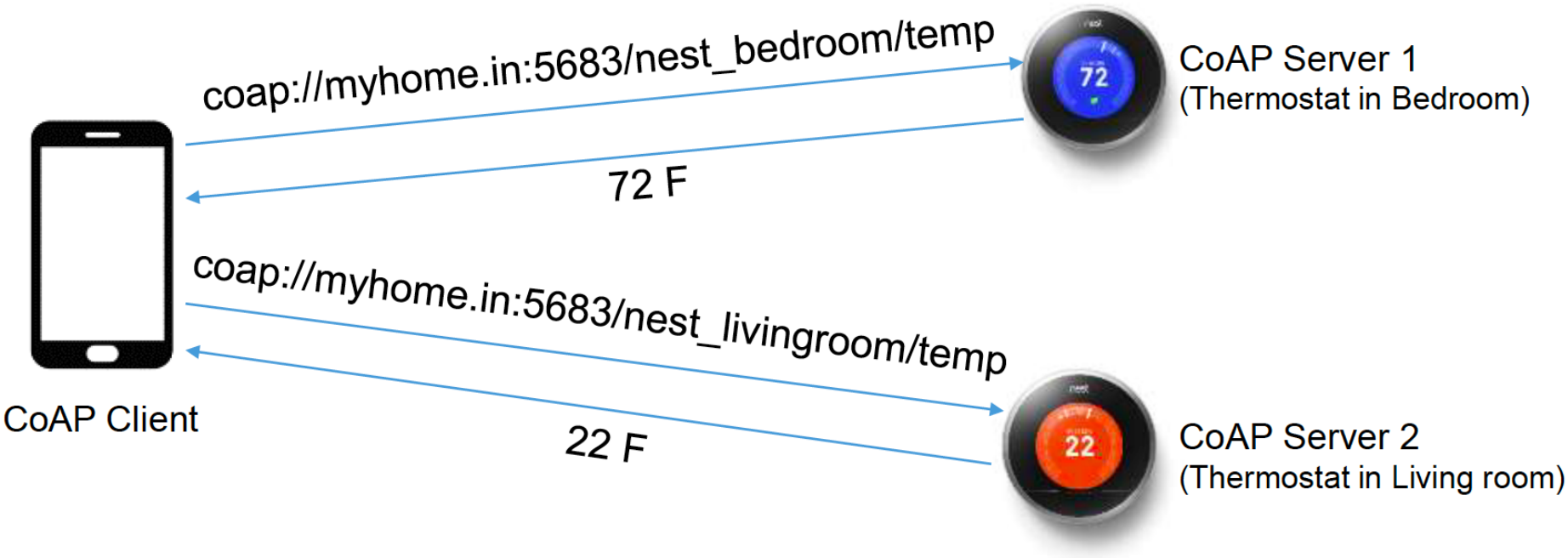
CoAP: Similarities with HTTP

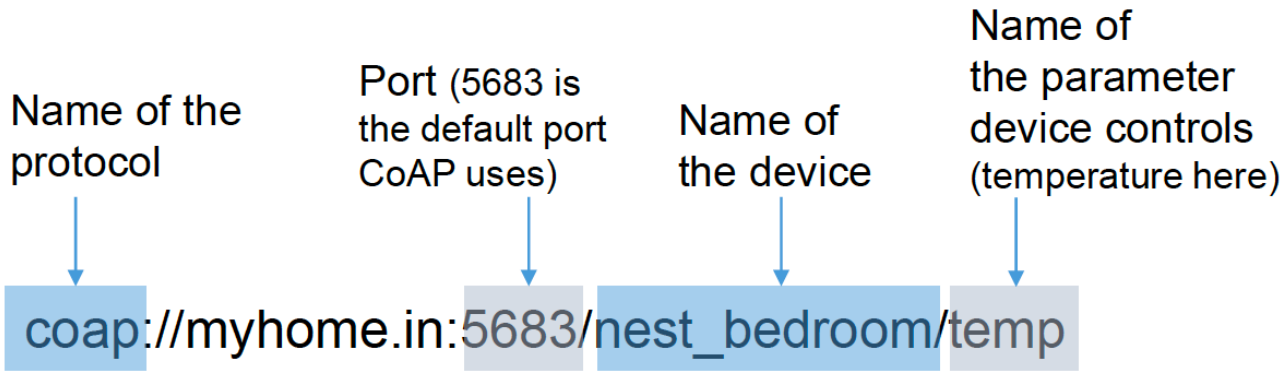
- Inspired from HTTP – similar to HTTP CoAP also uses a request response model.
- Can be transparently mapped to HTTP - However, CoAP also provides features that go beyond HTTP such as native push notifications and group communication.
- From a developer point of view, CoAP feels very much like HTTP. Obtaining a value from a sensor is not much different from obtaining a value from a Web API.
- **REST model for small devices** - It implements the REST architectural style
- **Made for billions of nodes with very less memory -** The Internet of Things will need billions of nodes, many of which will need to be inexpensive. CoAP has been designed to work on microcontrollers with as low as 10 KiB of RAM and 100 KiB of code space.
- Designed to use minimal resources, both on the device and on the network. Instead of a complex transport stack, it gets by with UDP on IP. A 4-byte fixed header and a compact encoding of options enables small messages that cause no or little fragmentation on the link layer.
- Like HTTP, CoAP can carry different types of payloads, and can identify which payload type is being used. CoAP integrates with XML, JSON, CBOR, or any data format of your choice.
- **Discovery integrated** - CoAP resource directory provides a way to discover the properties of the nodes (devices/things in IoT) on your network.

CoAP: Differences with HTTP

- CoAP runs over UDP and not TCP (HTTP typically uses TCP, though it can use UDP also)
- CoAP replaces the text headers used in HTTPU (HTTP Unicast) with more compact binary headers
- It reduces the number of options available in the header
- CoAP also reduces the set of methods that can be used; it allows
  - GET
  - POST
  - PUT, and
  - DELETE
- Method calls can be made using confirmable & nonconfirmable message services
  - When a confirmable message is received, receiver always returns an acknowledgement. The sender resends messages if an acknowledgement is not returned within a given time.
  - When a nonconfirmable message is received, receiver does not return an acknowledgement.
- No of response code has also been reduced (to make implementation simpler)
- CoAP also broke away from the Internet Media Type scheme used in HTTP and other protocols and replaced this with a reduced set of Content-Formats.

CoAP – Request Response





CoAP Message Types

CON / Confirmable message

A confirmable message requires a response, either a positive acknowledgement or a negative acknowledgement. In case acknowledgement is not received, retransmissions are made until all attempts are exhausted.

NON / Non-confirmable message

A non-confirmable request is used for unreliable transmission (like a request for a sensor measurement made in periodic basis. Even if one value is missed, there is not too much impact). Such a message is not generally acknowledged.

ACK / Acknowledgement

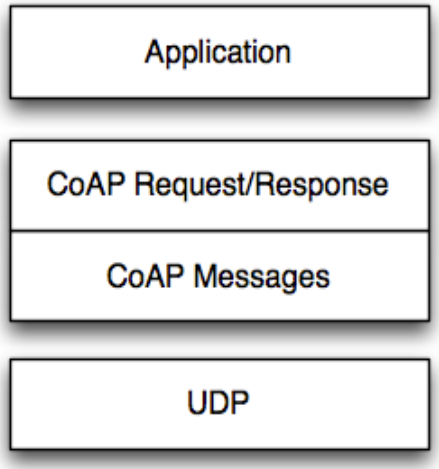
Sent to acknowledge a confirmable (CON) message.

RST / Reset

This represents a negative acknowledgement and means “Reset”. It generally indicates, some kind of failure (like unable to parse received data)

The Transaction Model

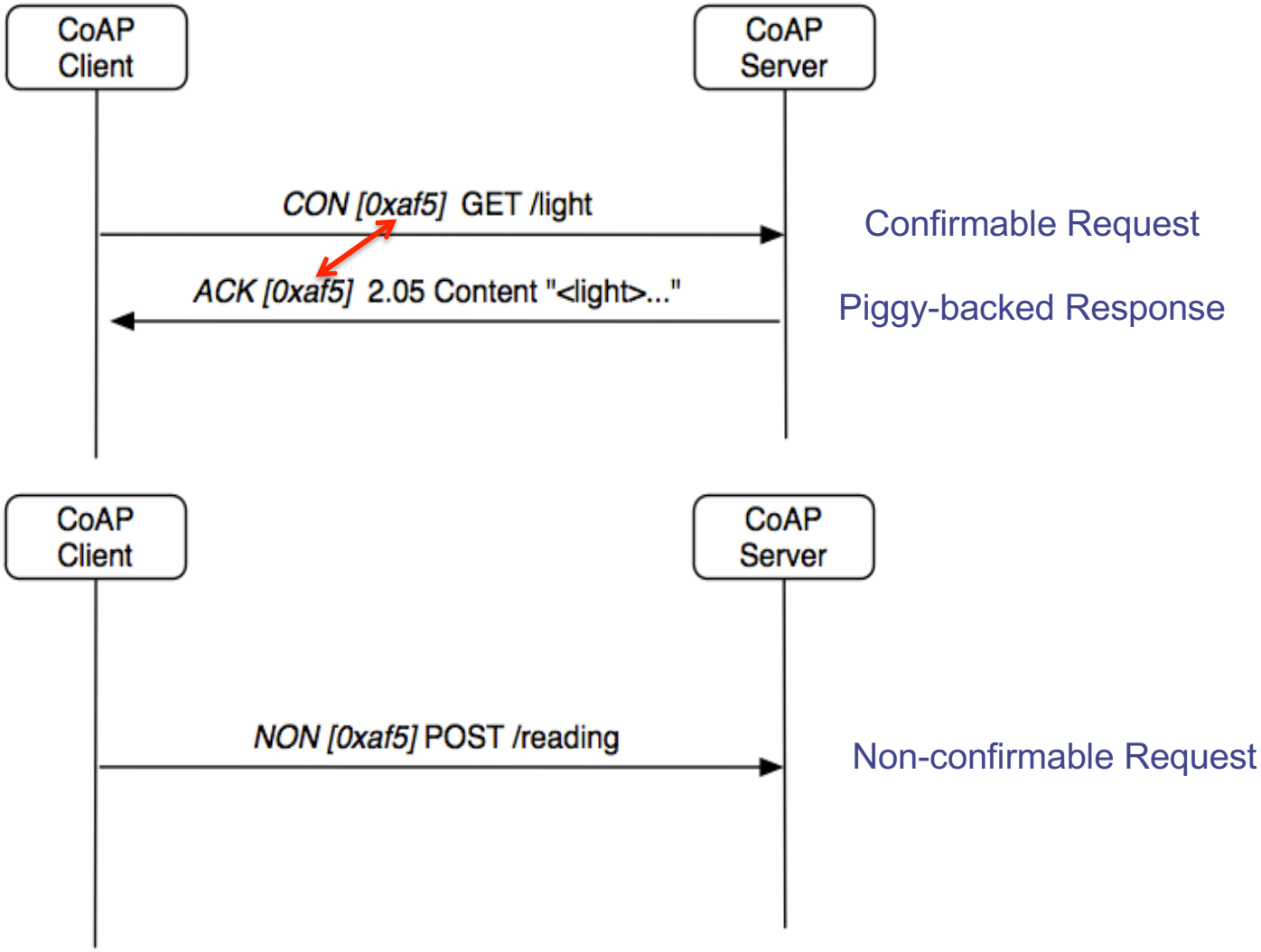
- Transport
  - UDP binding with DTLS security
  - CoAP over SMS or TCP possible



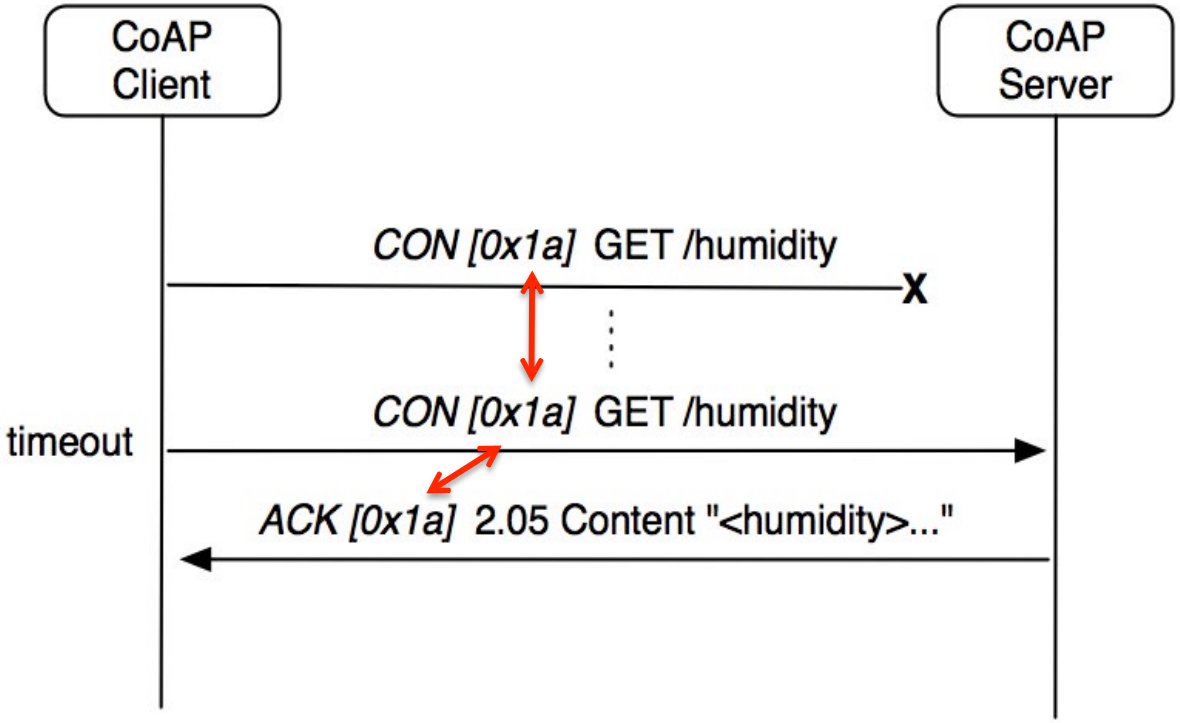
- Messaging
  - Simple message exchange between end-points
  - Confirmable or Non-confirmable Messages answered by Acknowledgment or Reset Message

- REST
  - Request/Response piggybacked on messages
  - Method, Response Code and Options (URI, content-type etc.)

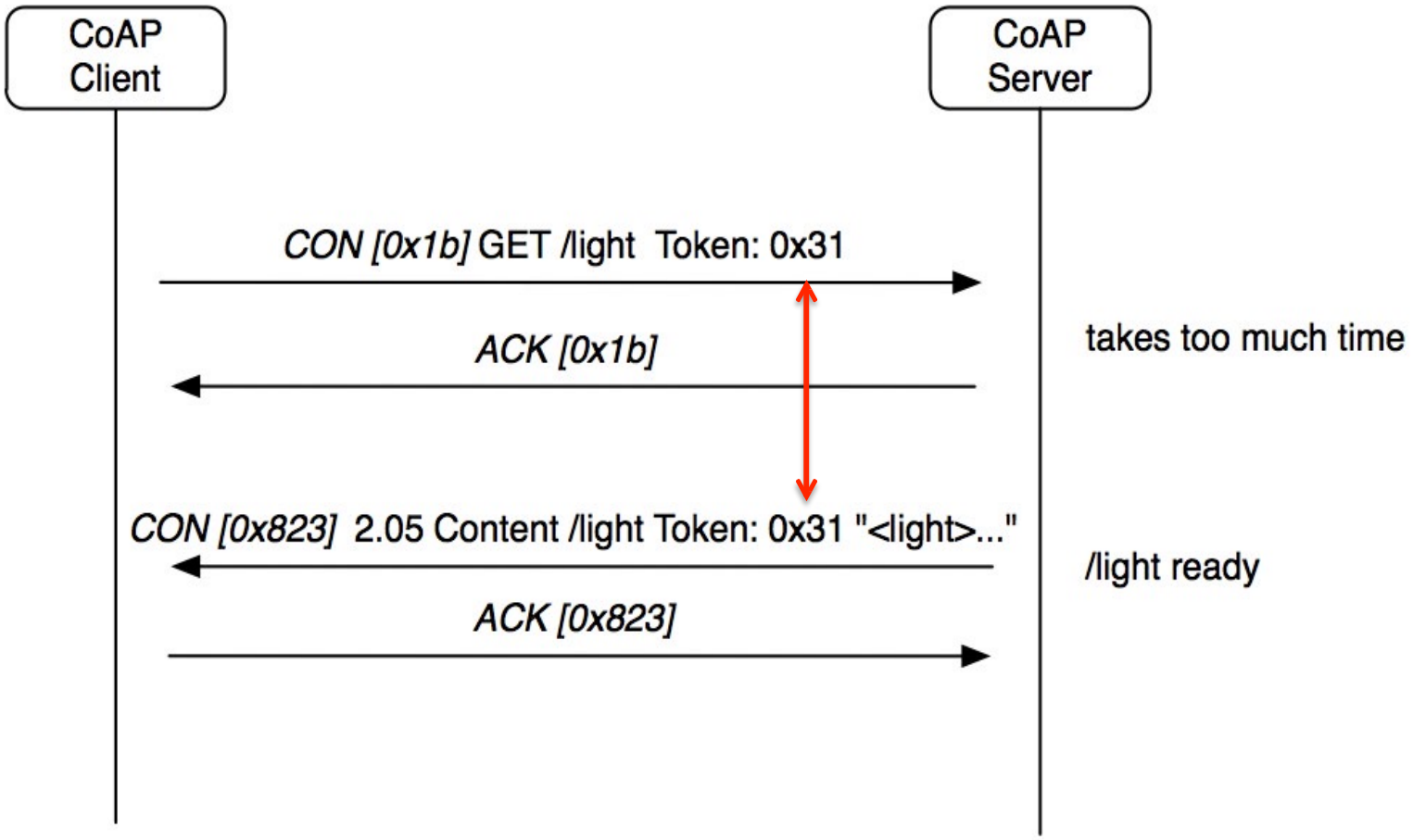
Request Examples



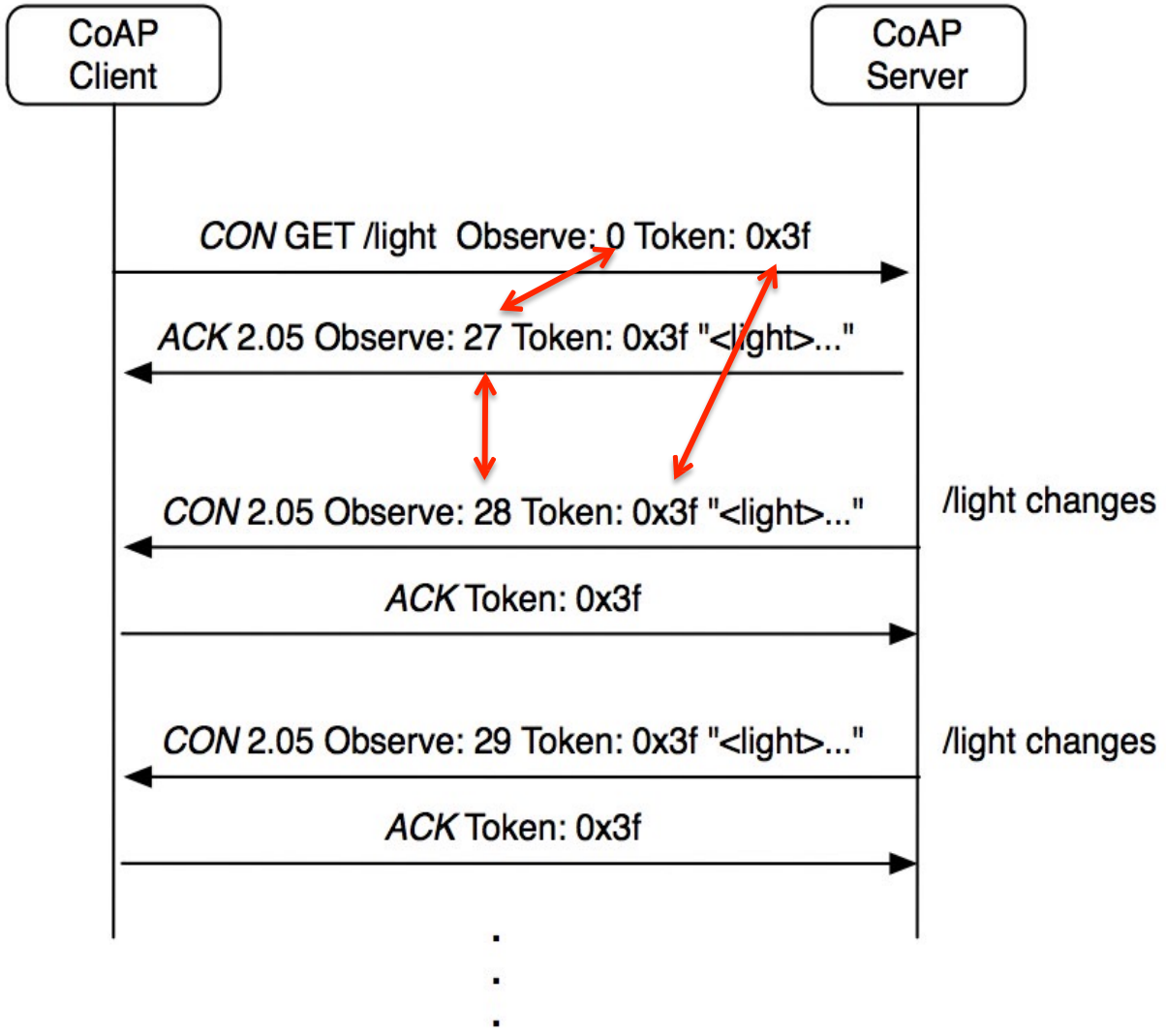
Dealing with Packet Loss



Separate Response

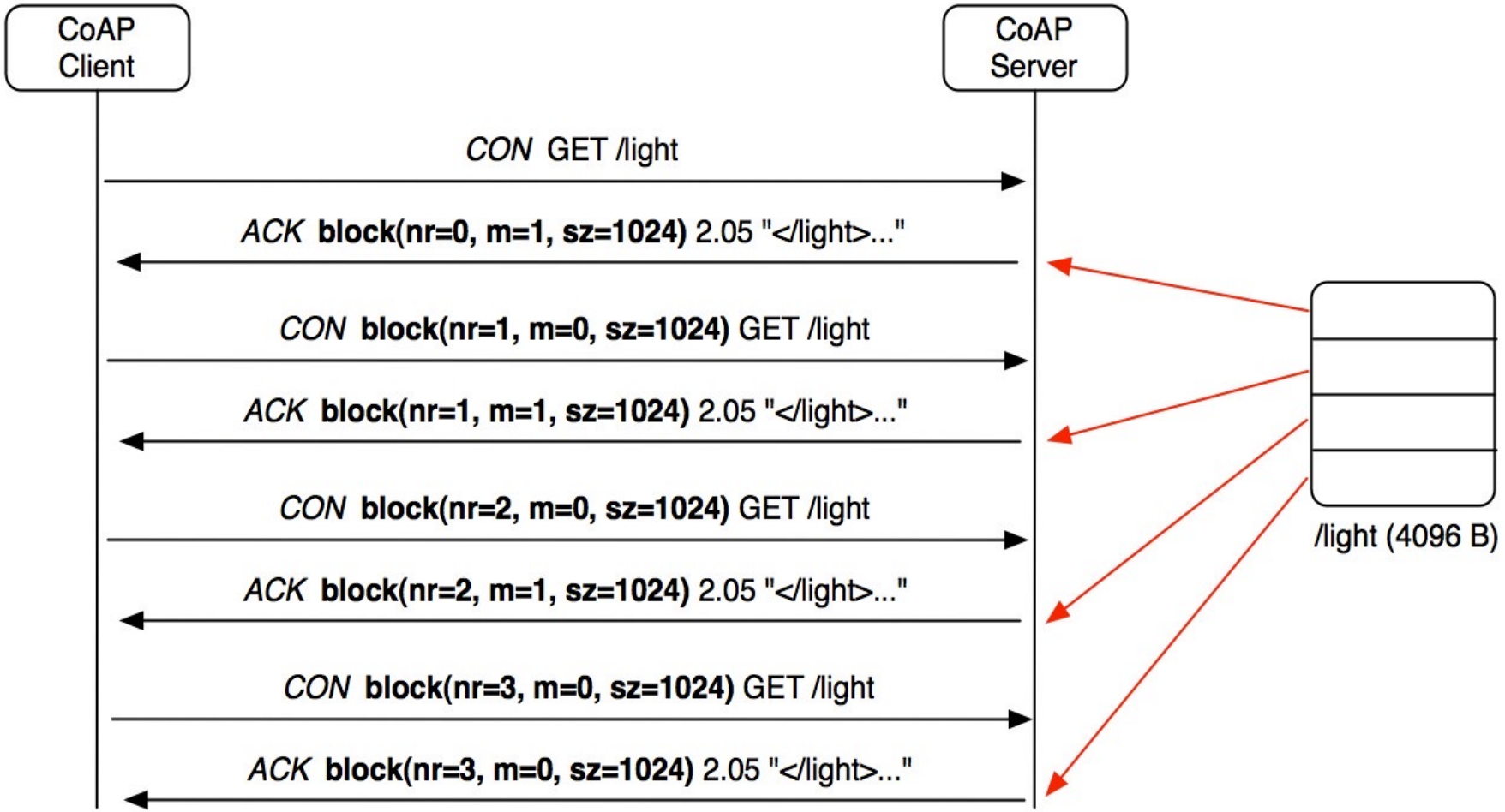


Observation



See draft-ietf-core-observe

Block transfer

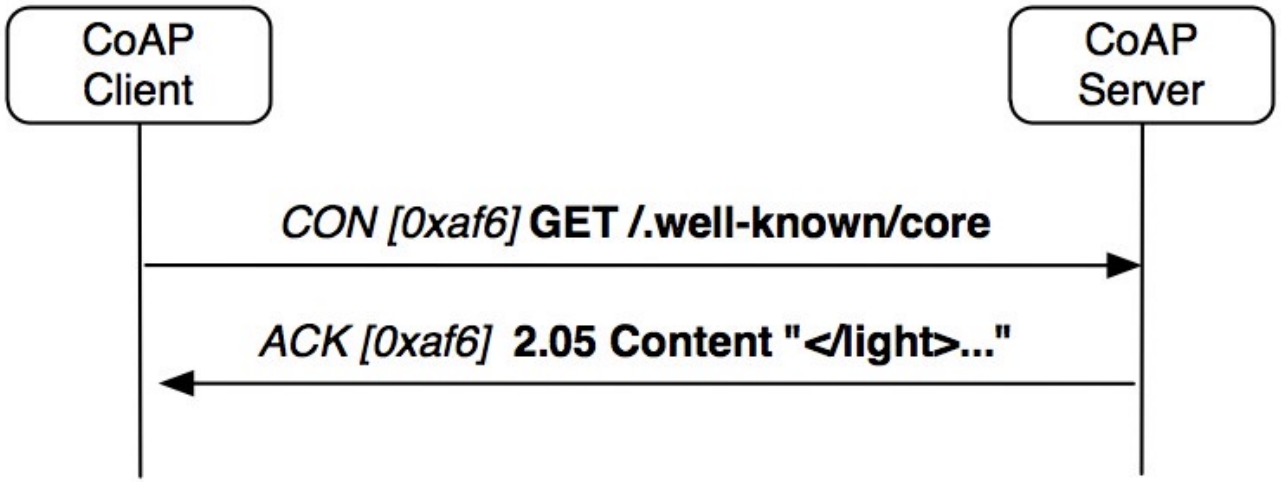


See draft-ietf-core-block

Constrained RESTful Environments (CoRE) Link Format

- RFC6690 is aimed at Resource Discovery for M2M
  - Defines a link serialization suitable for M2M
  - Defines a well-known resource where links are stored
  - Enables query string parameters for filtered GETs
  - Can be used with unicast or multicast (CoAP)
- Resource Discovery with RFC6690
  - Discovering the links hosted by CoAP (or HTTP) servers
  - GET /.well-known/core?optional_query_string
  - Returns a link-header style format
  - URL, relation, type, interface, content-type etc.

CoRE Resource Discovery

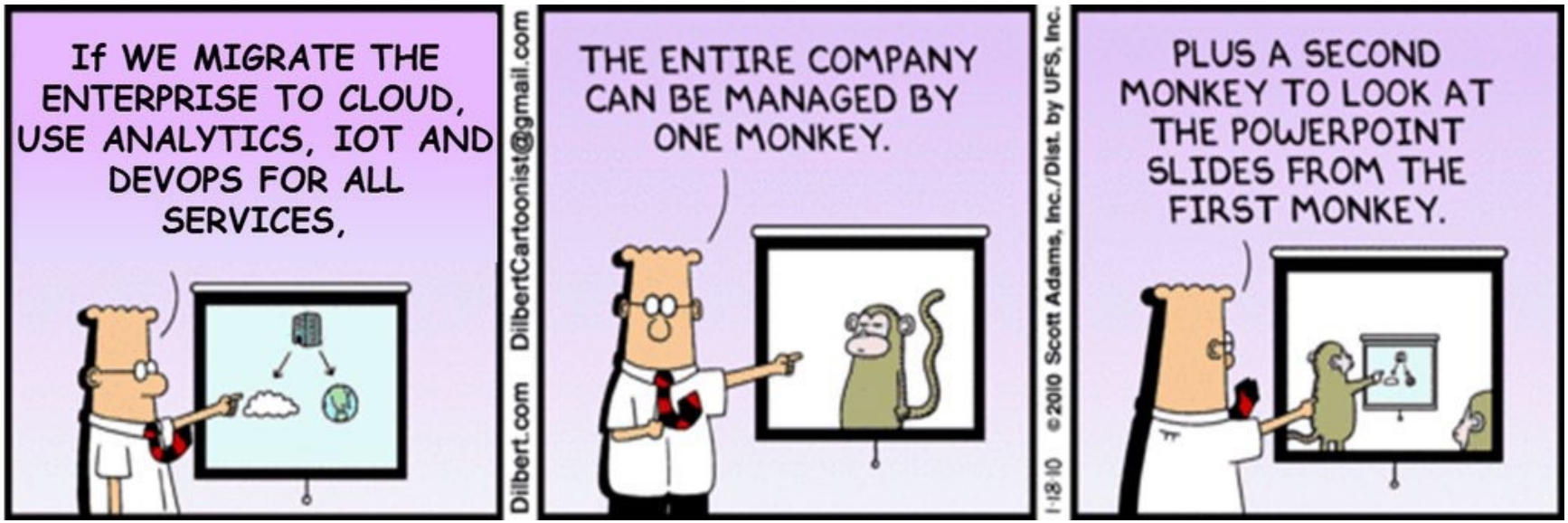


Getting Started with CoAP

- There are many open-source implementations available
  - mbed includes CoAP support
  - Java CoAP Library Californium
  - C CoAP Library Erbium
  - libCoAP C Library
  - jCoAP Java Library
  - OpenCoAP C Library
  - TinyOS and Contiki include CoAP support
  - More on http://coap.technology/impls.html
- CoAP is already part of many commercial products/systems
  - Thread group
  - ARM Sensinode NanoService
  - RTX 4100 WiFi Module
- Firefox has a CoAP plugin called Copper
- Wireshark has CoAP dissector support

This lecture

- The Constrained Application Protocol (CoAP)
- MQTT
- Cloud service for IoT


IF WE MIGRATE THE ENTERPRISE TO CLOUD, USE ANALYTICS, IOT AND DEVOPS FOR ALL SERVICES,
THE ENTIRE COMPANY CAN BE MANAGED BY ONE MONKEY.
PLUS A SECOND MONKEY TO LOOK AT THE POWERPOINT SLIDES FROM THE FIRST MONKEY.
DilbertCartoonist@gmail.com
Dilbert.com
1-18-10 ©2010 Scott Adams, Inc./Dist. by UFS, Inc.

MQTT: Message Queue Telemetry Transport

Overview

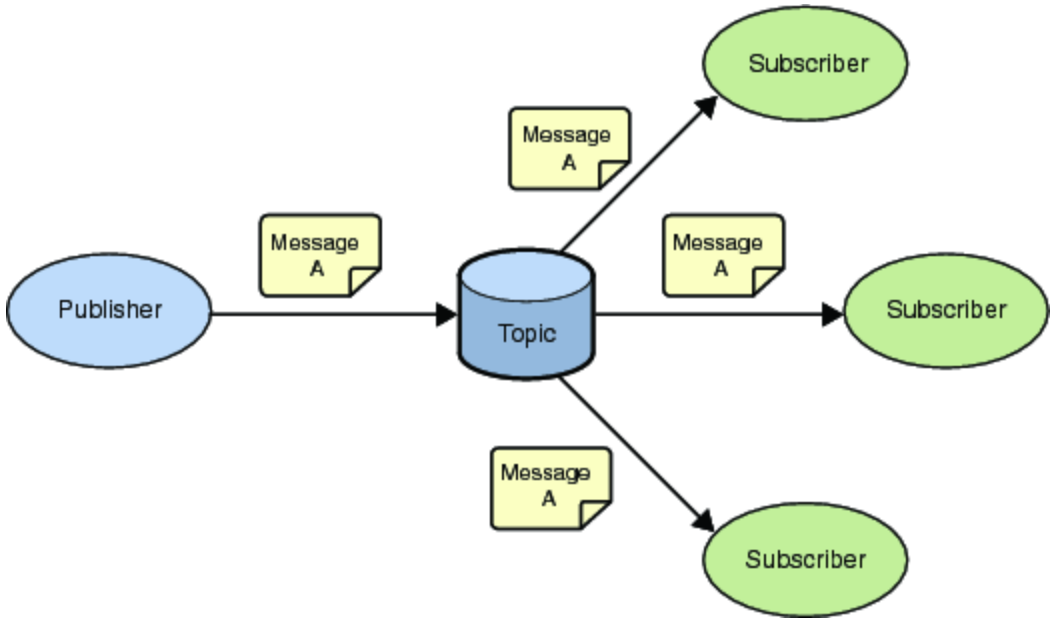
- MQTT was co-invented by IBM and Arcom Systems over 13 years ago.
- Lightweight messaging protocol for M2M communication
- Telemetry = Tele-metering = remote measurements
- MQ originated from “message queueing” architecture used by IBM for service oriented networks. There is **no** queueing in MQTT !!
- Telemetry data flows from devices to a server or broker. Uses a publish-subscribe mechanism
- Lightweight = Low network bandwidth and small code footprint
- Spec: http://www.ibm.com/developerworks/webserices/library/ws-mqtt/index.html

Abstract from the MQTT spec web site: The MQ Telemetry Transport (MQTT) protocol is a lightweight publish/subscribe protocol flowing over TCP/IP for remote sensors and control devices through low bandwidth, unreliable or intermittent communications. This protocol specification has not been standardized. It is made available here under a royalty free license.

Overview (contd.)

- Facebook messenger uses MQTT to minimize battery usage.
- Several other applications in medical, environmental
- applications
- Many open source implementations of clients and brokers are available
  - Really small message broker (RSMB): C
  - Mosquitto (https://mosquitto.org)
  - Micro broker: Java based for mobile devices, notebooks

MQTT - Publish Subscribe Messaging aka One to Many



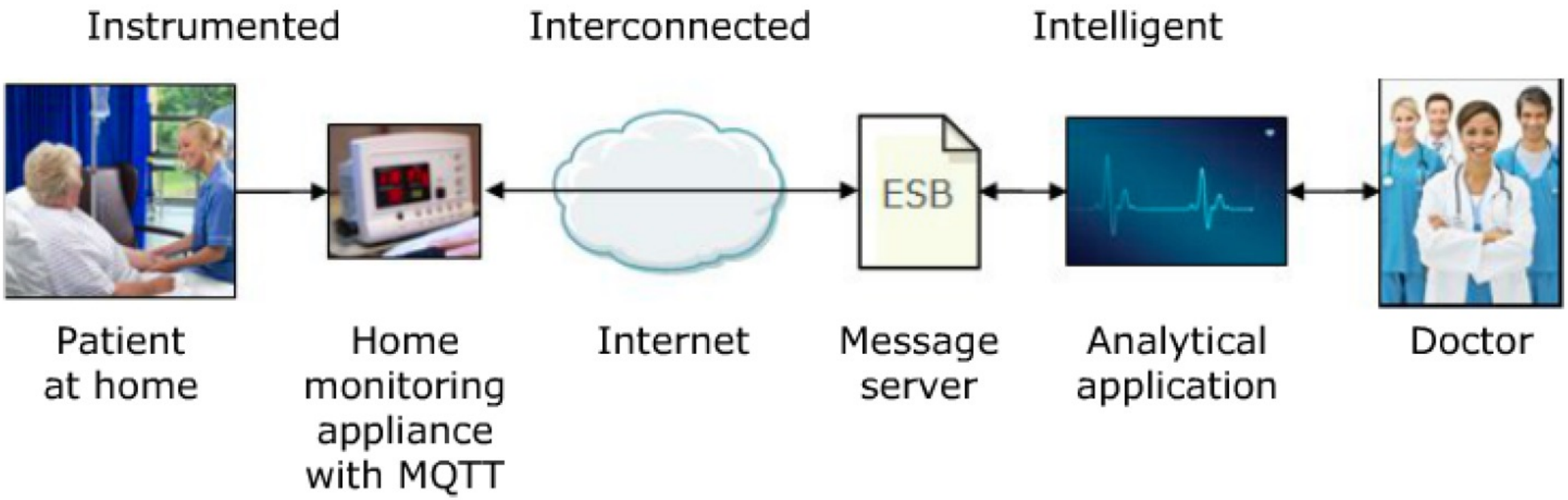
A Publish Subscribe messaging protocol allowing a message to be published once and multiple consumers (applications / devices) to receive the message providing decoupling between the producer and consumer(s)

A producer sends (publishes) a message (publication) on a topic (subject)
A consumer subscribes (makes a subscription) for messages on a topic (subject)

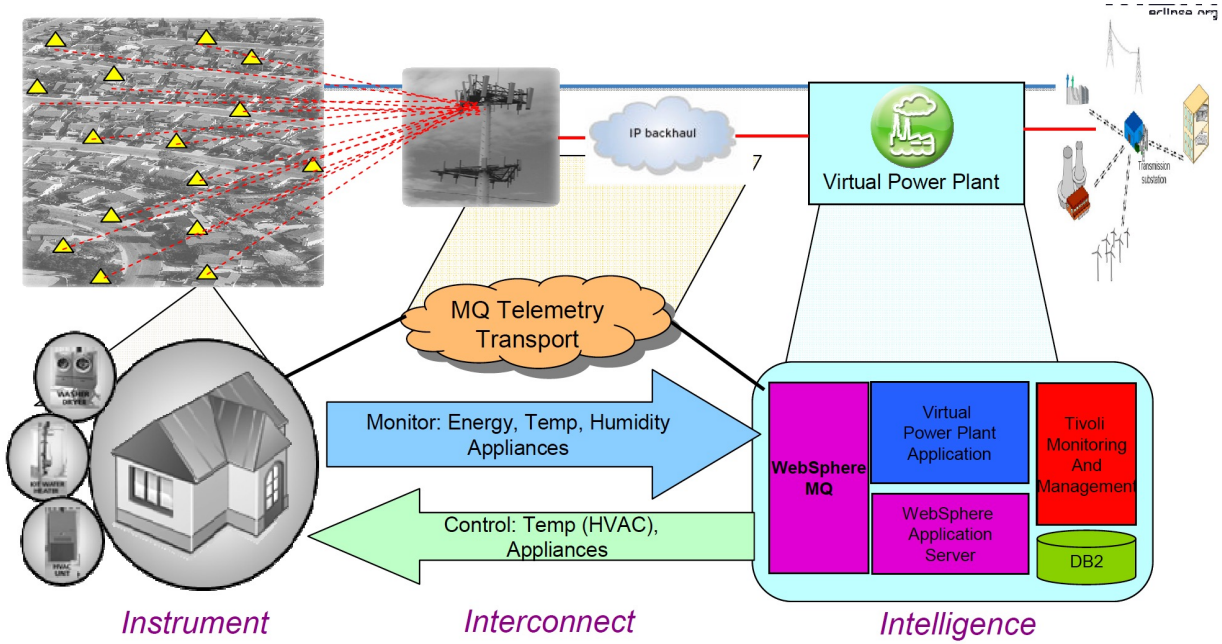
A message server / broker matches publications to subscriptions
•If no matches the message is discarded
•If one or more matches the message is delivered to each matching subscriber/consumer

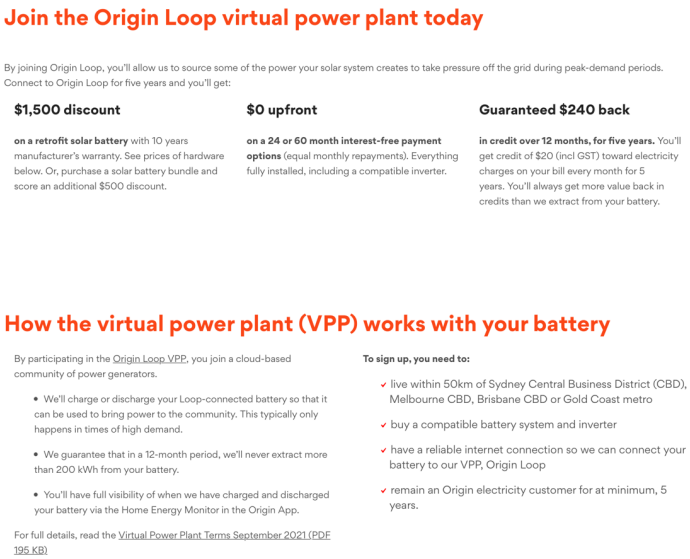
Example: Health Monitoring Application

- Home pacemaker monitoring solution
  - Sensors on patient
  - Collected by a monitoring equipment in home (broker) using MQTT
  - Subscribed by a computer in the hospital
  - Alerts the doctor if anything is out-of-order



Example: Virtual Power Plant





Join the Origin Loop virtual power plant today

By joining Origin Loop, you'll allow us to source some of the power your solar system creates to take pressure off the grid during peak-demand periods. Connect to Origin Loop for five years and you'll get:

$1,500 discount

on a retrofit solar battery with 10 years manufacturer's warranty. See prices of hardware below. Or, purchase a solar battery bundle and score an additional $500 discount.

$0 upfront

on a 24 or 60 month interest-free payment options (equal monthly repayments). Everything fully installed, including a compatible inverter.

Guaranteed $240 back

in credit over 12 months, for five years. You'll get credit of $20 (incl GST) toward electricity charges on your bill every month for 5 years. You'll always get more value back in credits than we extract from your battery.

How the virtual power plant (VPP) works with your battery

By participating in the Origin Loop VPP, you join a cloud-based community of power generators.

- We'll charge or discharge your Loop-connected battery so that it can be used to bring power to the community. This typically only happens in times of high demand.
- We guarantee that in a 12-month period, we'll never extract more than 200 kWh from your battery.
- You'll have full visibility of when we have charged and discharged your battery via the Home Energy Monitor in the Origin App.

For full details, read the Virtual Power Plant Terms September 2021 (PDF 195 KB)

To sign up, you need to:

- live within 50km of Sydney Central Business District (CBD), Melbourne CBD, Brisbane CBD or Gold Coast metro
- buy a compatible battery system and inverter
- have a reliable internet connection so we can connect your battery to our VPP, Origin Loop
- remain an Origin electricity customer for at minimum, 5 years.

Smart homes enabled with smart devices (HVAC, appliances,…)
Devices connect to a home gateway box via 802.15.4 or ….
Home gateway monitors devices collecting data that impacts energy usage
Home gateway publishes energy data to Virtual Power Plant (VPP) every 5 mins over 3G/4G
VPP analyses real time data from all homes and other sources. When required it sheds load:
Publishing control commands to the home gateway of multiple homes
Home gateway controls smart device(s) when instructed by VPP

More examples

POS

Kiosks

Slot Machines

Automotive/
Telematics

Environment &
Traffic Monitoring

Fire & Gas
Testing

Chemical
Detection

Asset tracking /
management

SCADA

Medical

Field Force
Automation

Home
Automation

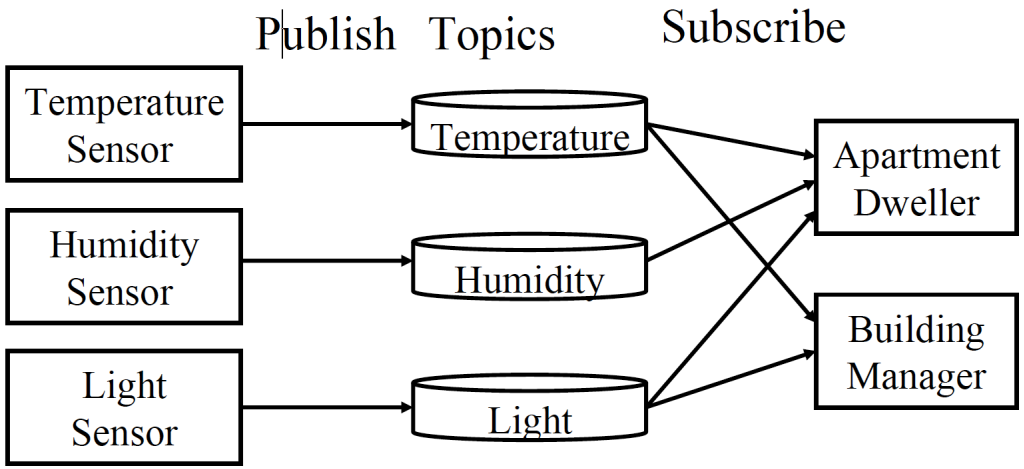
eclipse

Railway

MQTT - Publish Subscribe Messaging aka One to Many

- A topic forms the namespace
  - Is hierarchical, with each “sub topic” separated by a /

- An example topic space
  - A house publishes information about itself on:
    - <country>/<region>/<town>/<postcode>/<house>/energyConsumption
    - <country>/<region>/<town>/<postcode>/<house>/solarEnergy
    - <country>/<region>/<town>/<postcode>/<house>/alarmState
  - And subscribes for control commands:
    - <country>/<region>/<town>/<postcode>/<house>/thermostat/setTemp



MQTT - Publish Subscribe Messaging aka One to Many

- A subscriber can subscribe to an absolute topic or can use wildcards:
  - Single-level wildcards “+” can appear anywhere in the topic string
  - Multi-level wildcards “#” must appear at the end of the string
- Wildcards must be next to a separator
  - Can't use wildcards when publishing
- For example
  - UK/Hants/Hursley/SO212JN/1/energyConsumption
    - Energy consumption for 1 house in Hursley
  - UK/Hants/Hursley/+/+/energyConsumption
    - Energy consumption for all houses in Hursley
  - UK/Hants/Hursley/SO212JN/#
    - Details of energy consumption, solar and alarm for all houses in SO212JN

MQTT - Publish Subscribe Messaging aka One to Many

- A subscription can be durable or non durable
  - Durable:
    - Once a subscription is in place a broker will forward matching messages to the subscriber:
      - Immediately if the subscriber is connected
      - If the subscriber is not connected messages are stored on the server/broker until the next time the subscriber connects
  - Non-durable: The subscription lifetime is the same as the time the subscriber is connected to the server/broker
- A publication may be retained
  - A publisher can mark a publication as retained
  - The broker/server remembers the last known good message of a retained topic
  - The broker/server gives the last known good message to new subscribers
    - i.e. the new subscriber does not have to wait for a publisher to publish a message in order to receive its first message

Designed for Constrained Device

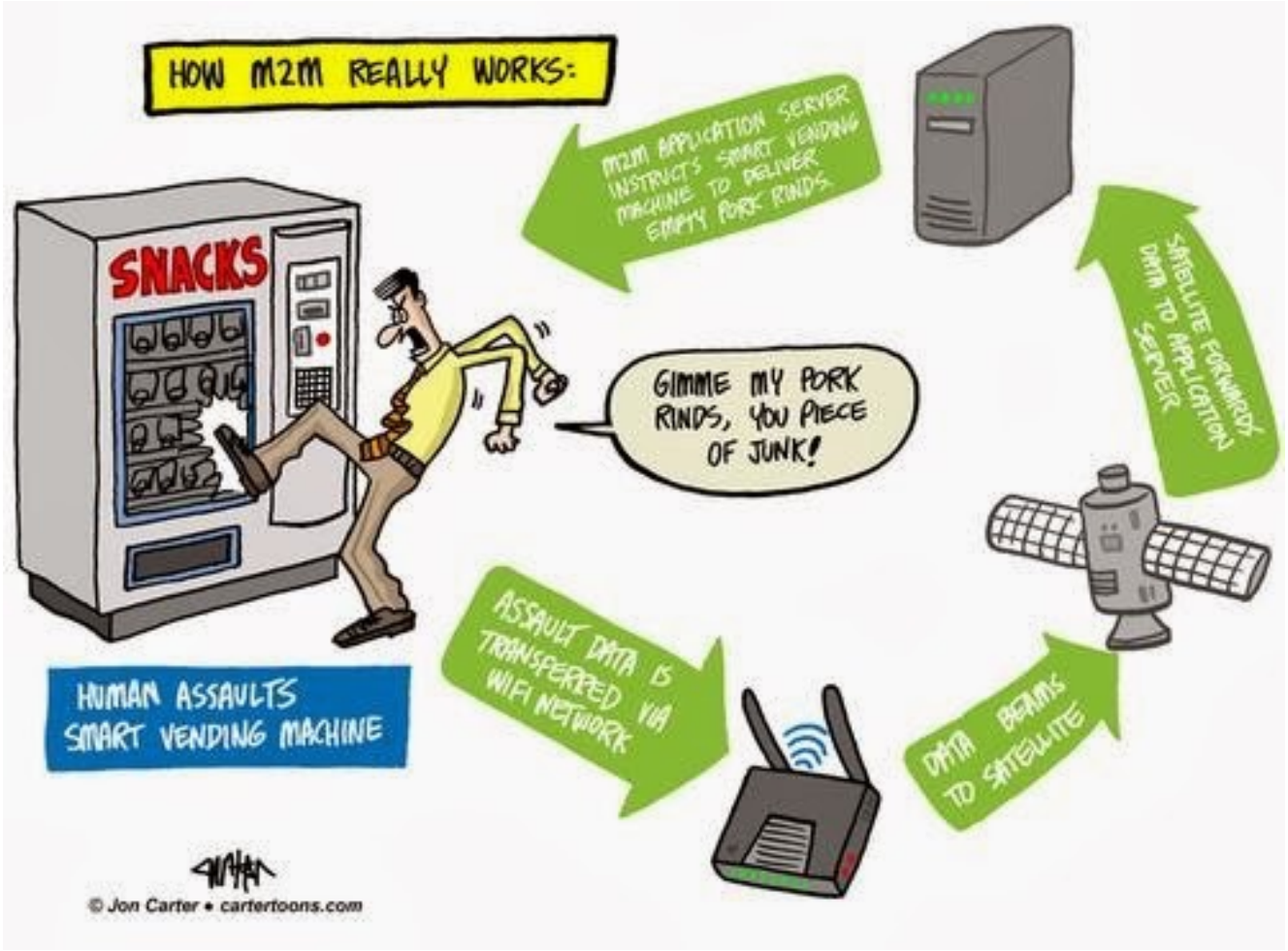
- Suited to applications / devices that may have limited resources available
  - 8 Bit controllers upwards
  - Battery operated
- Multiple MQTT client implementations available in many form factors / languages
  - Tiny footprint MQTT client (and server) libraries e.g., a C client library of 30Kb and a Java library of 64Kb
- Clients available in .NET, Perl, Python, Ruby….
- Works with range of embedded devices including Arduino, Mbed, Netdunio, Mbed,…
- Open source: http://www.eclipse.org/paho/

Designed for Constrained Networks

- Protocol compressed into bit-wise headers and variable length fields
- Smallest possible packet size is 2 bytes
- Asynchronous bidirectional "**push**" delivery of messages to applications **(no polling)**
  - Client to server and server to client
- Supports always-connected and sometimes-connected models
- Provides Session awareness
  - Configurable keep alive providing granular session awareness
  - "Last will and testament" enable applications to know when a client goes offline abnormally
- Typically utilises TCP based networks e.g. Websockets
- Tested on many networks – vsat, gprs, 2G….
- Provides multiple deterministic message delivery QoS
  - 0 – message delivered at most once.
  - 1 – message will be delivered but may be duplicated
  - 2 – once and once only delivery
  - *QOS maintained over fragile network even if connection breaks*

MQTT vs HTTP

	MQTT	HTTP
Design	Data centric	Document centric
Pattern	Publish/Subscribe	Request /Response
Complexity	Simple	More Complex
Message Size	Small. Binary with 2B header	Large. ASCII
Service Levels	Three	One
Libraries	30kB C and 100 kB Java	Large
Data Distribution	1 to zero, one, or n	1 to 1 only


HOW M2M REALLY WORKS:
M2M APPLICATION SERVER INSTRUCTS SMART VENDING MACHINE TO DELIVER EMPTY PORK RINDS.
SATELLITE FORWARDS DATA TO APPLICATION SERVER
SNACKS
GIMME MY PORK RINDS, YOU PIECE OF JUNK!
HUMAN ASSAULTS SMART VENDING MACHINE
ASSAULT DATA IS TRANSFERRED VIA WIFI NETWORK
DATA BEAMS TO SATELLITE
© Jon Carter • cartertoons.com

This lecture

- The Constrained Application Protocol (CoAP)
- MQTT
- Cloud service for IoT

Do you use the Cloud?

Local Storage

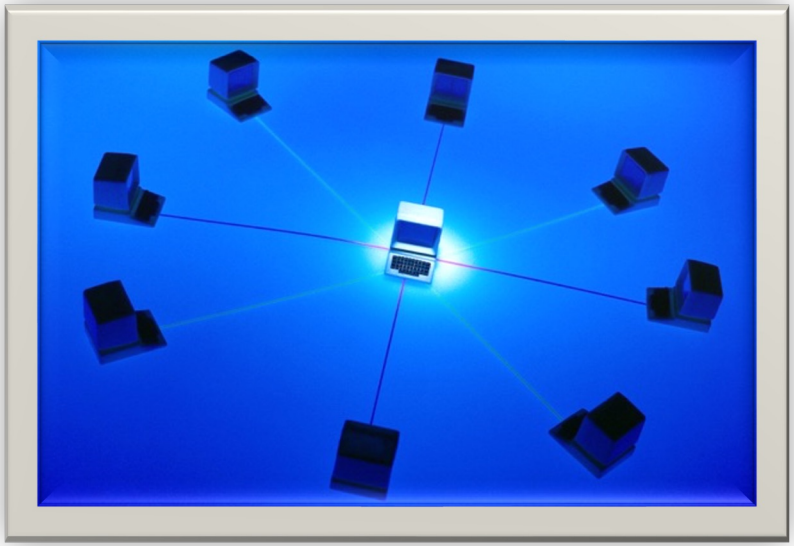
- Content is stored on THAT computer
- To use content must return to THAT computer
- Cannot access this content from another device or computer

Programs

- Purchase programs
- Load to the computer
- Each computer would need the program loaded and stored on the internal drive

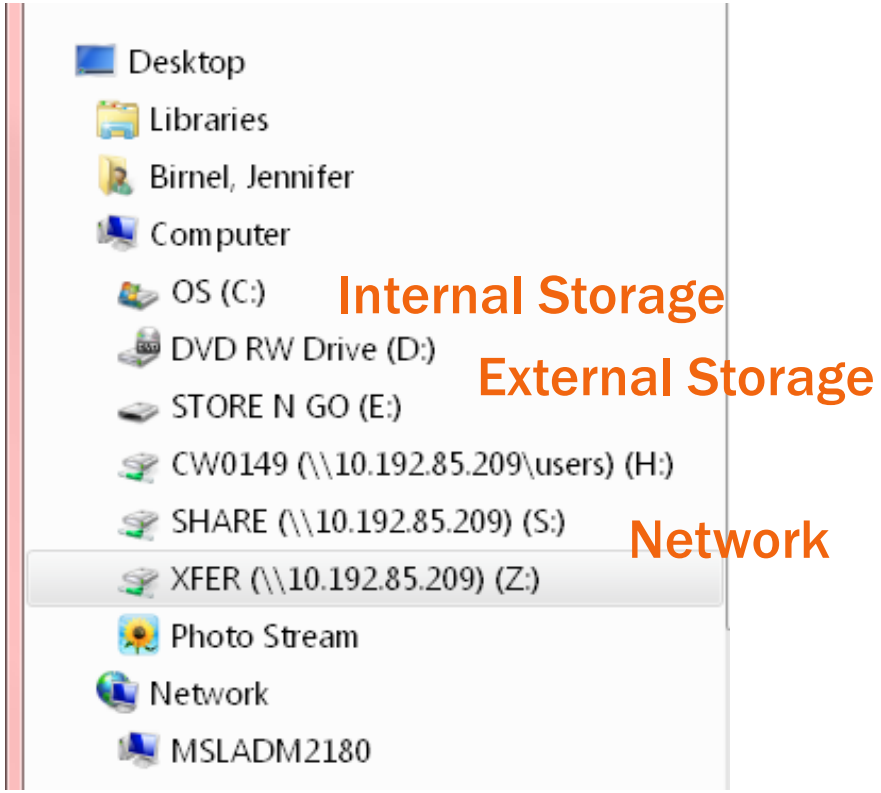
Networked Storage

- Multiple work stations talk to one unit that stores information and data.
- Data is not saved to the C: drive, but to a network drive
- Can retrieve the data stored to the network from any of the connected workstations.

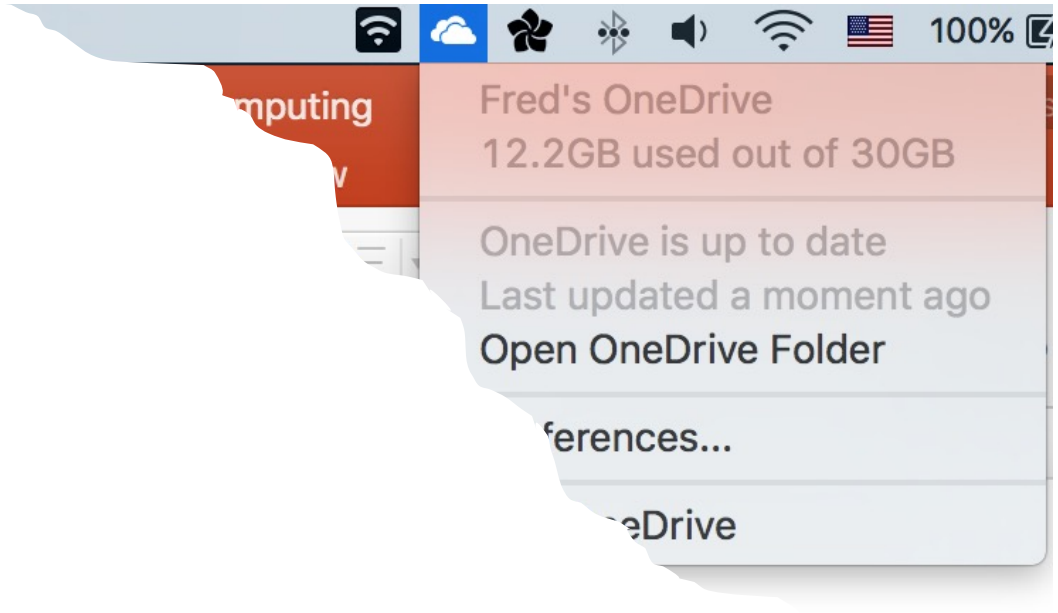


Saving documents

- When you do a “save as” on your computer, you choose where to save the material.



Cloud Storage



OneDrive

amazon web services EC2

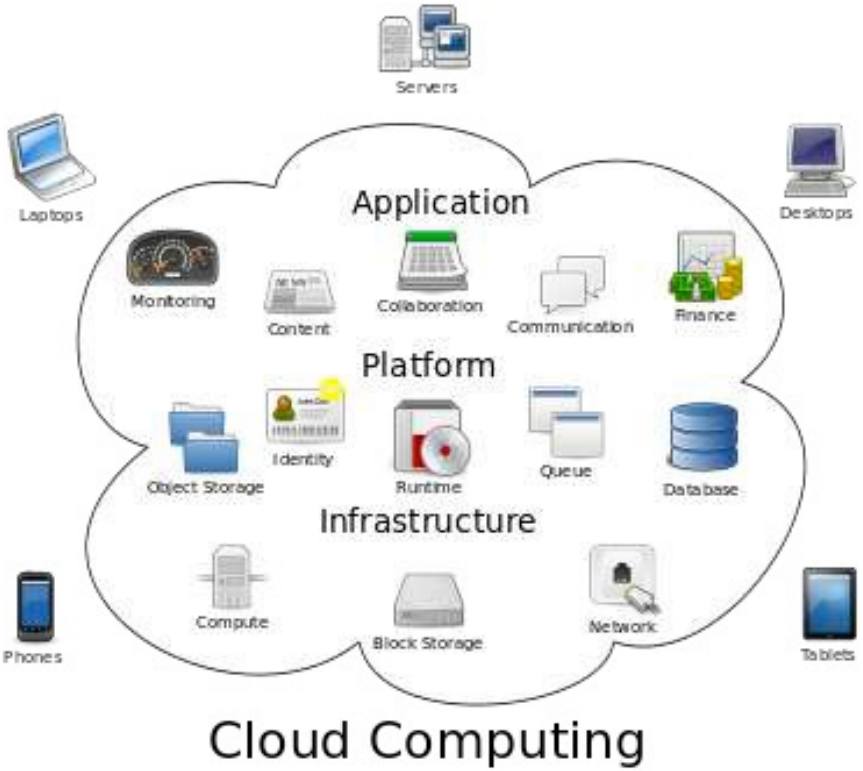
Dropbox

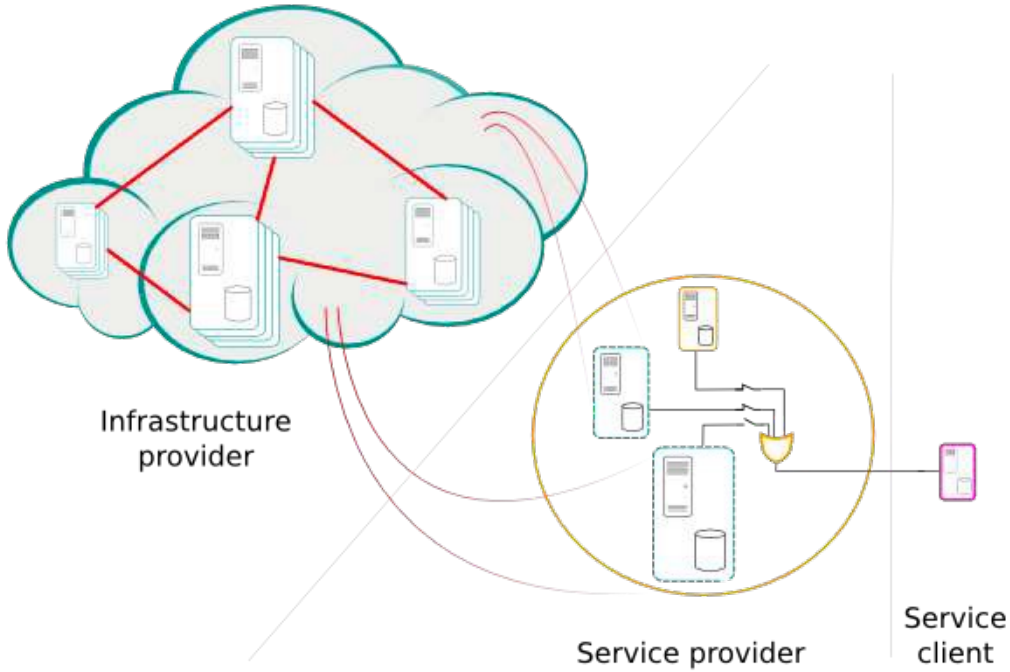
box

- Create an Account – User name and password
- Content lives with the account in the cloud
- Log onto any computer with Wi-Fi to find your content
- Synced to local storage

More than Storage

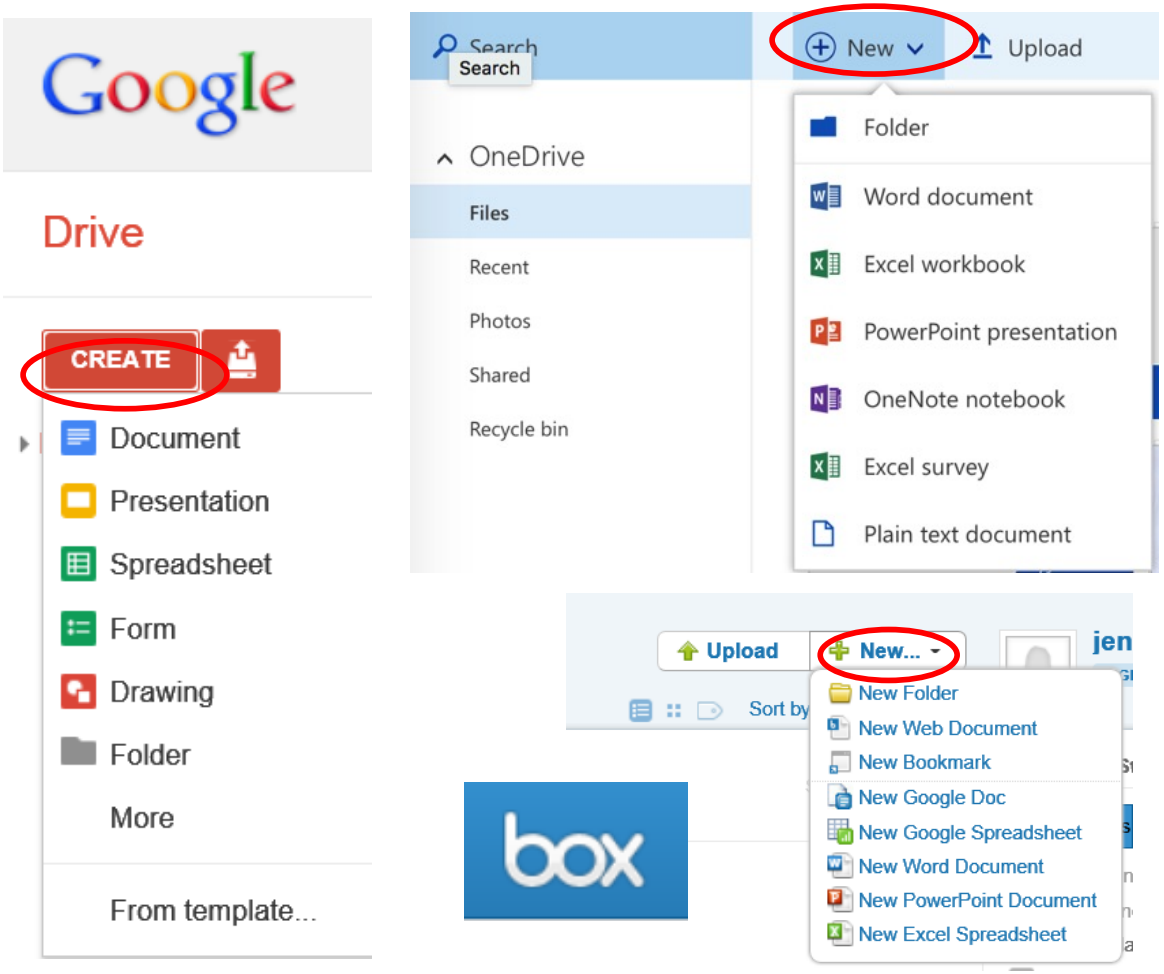
Cloud Computing





Document Creation

- Google Docs
- OneDrive
- Box



MORE THAN JUST SOFTWARE

It’s Not Just About APIs

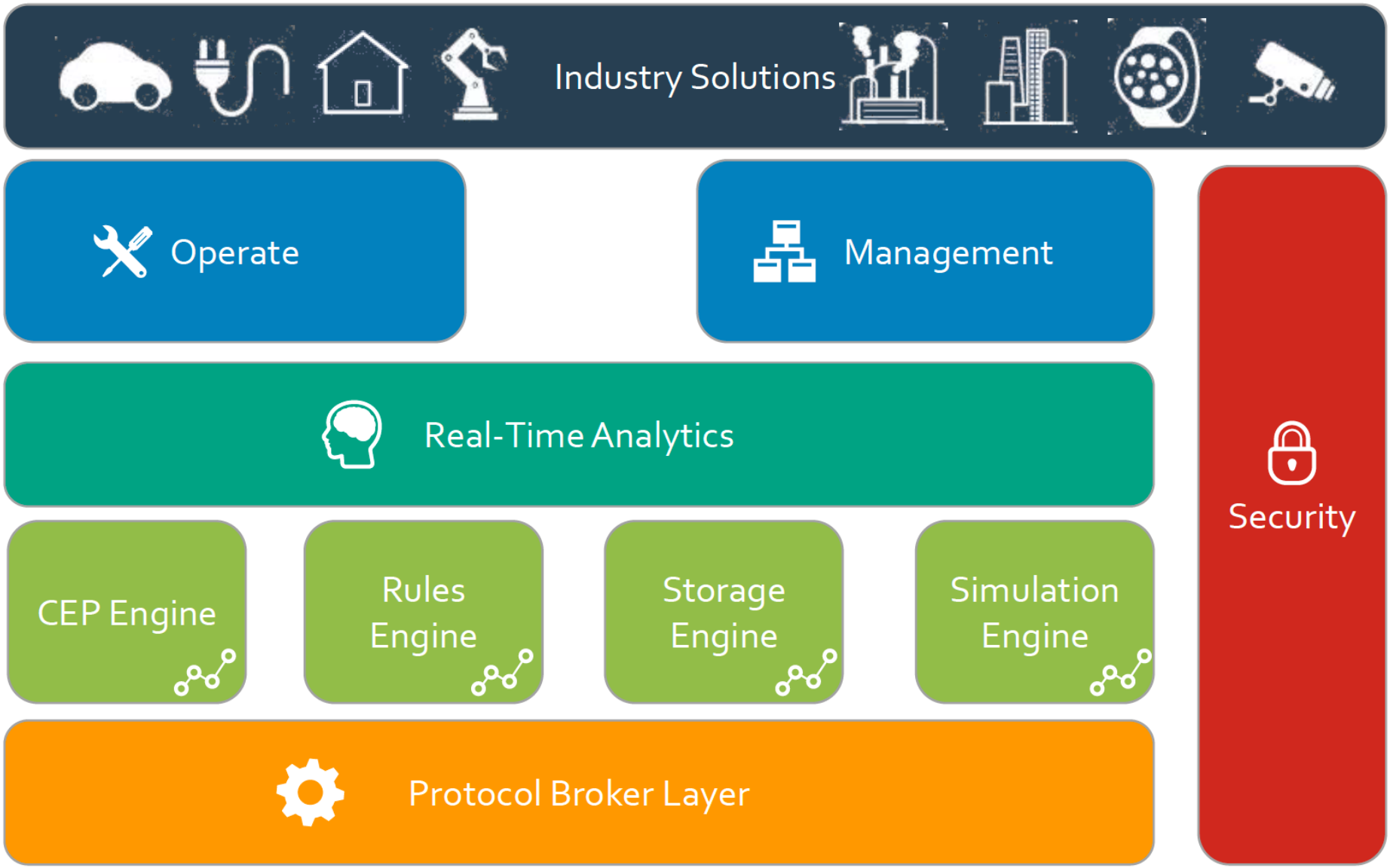
- Complex event processing
- Time series storage
- Data visualization platform
- Security and data privacy

Platform as a Service (PaaS)

PaaS provides all of the facilities required to support the complete life cycle of building and delivering web applications and services entirely from the Internet

- Typically applications must be developed with a particular platform in mind
- Multi tenant environments
- Highly scalable multi tier architecture

A Conceptual Framework



Protocol Brokering

- Goal: Receive data from smart devices using standard IoT protocols
- Relevant Technologies: 802.15.4, 6LoWPAN, MQTT, CoAP, LPWAN….
- Benefit: Provide a uniform layer for devices to plug into the platform

Event Processing

- Goal: Process queries against real time data streams
- Relevant Technologies: Complex event processing platforms (e.g. Azure Stream Analytics, SAP ESP, …)
- Benefit: Model queries based on conditions against the data produced by IoT devices

Storage Engine

- Goal: Store the data produced by IoT devices and the CEP engine for further processing
- Relevant Technologies: Time series databases
- Benefit: Store time-stamped events produced by IoT devices

Databases

SQL

- Stored in **tables** - relational model with rows and columns
- **Fixed** schema, which is easier for structured data, support **join** operations
- Scaling is vertical – need bigger servers for larger data
- MySQL, SQLite, IBM DB2, PostgresSQL, Oracle, MS SQL, etc.

NoSQL

- Stored as document, graph, key-value pairs
- **Scaling** – better for massively parallel processing, lends better to numerous commodity servers, virtual machines or cloud instances
- MongoDB, DynamoDB, Couchbase, Riak, Memcached, Redis, CouchDB, Hazelcast, Apache Cassandra and HBase

Simulation Engine

- Goal: Replay events already processed by the CEP engine
- Relevant Technologies: Complex event processing platforms (e.g. Azure Stream Analytics, SAP ESP, …)
- Benefit: Recreate and test scenarios based on historical data processed by the platform

Rules Engine

- Goal: Model if-then condition against the data processed by the platform
- Relevant Technologies: Dynamic languages: JavaScript, Ruby, Python
- Benefit: Dynamically detect Boolean conditions about the data produced by IoT devices

Real Time Analytics

- Goal: Model analytics and reports based on the data produced by IoT devices
- Relevant Technologies: AWS IoT Analytics etc.
- Data visualization technologies: D3JS, Kibana, QuickSight etc.
- Benefit: Obtain real time intelligence against the data processed by the IoT platform

Security

- Goal: Encrypt and sign data produced by IoT devices
- Relevant Technologies: Traditional data security libraries
- Benefit: Protect data exchanges between smart devices and corporate systems

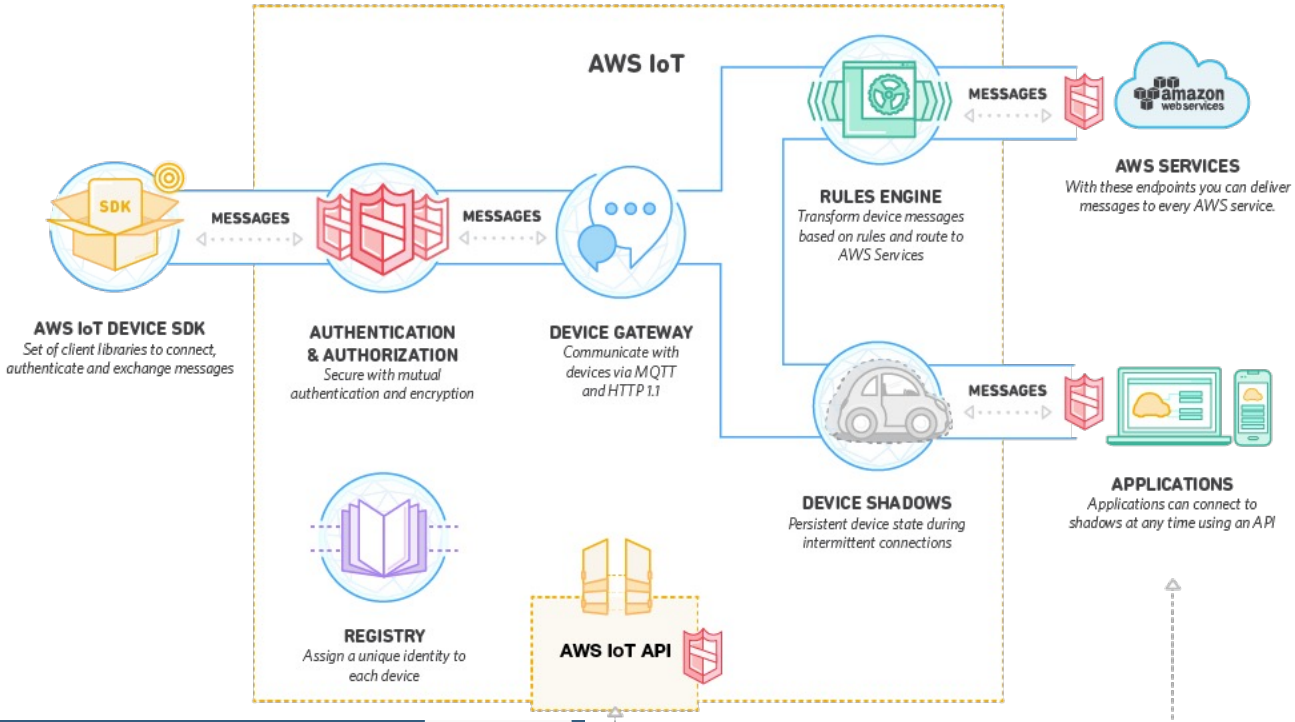
Benefits of IoT PaaS

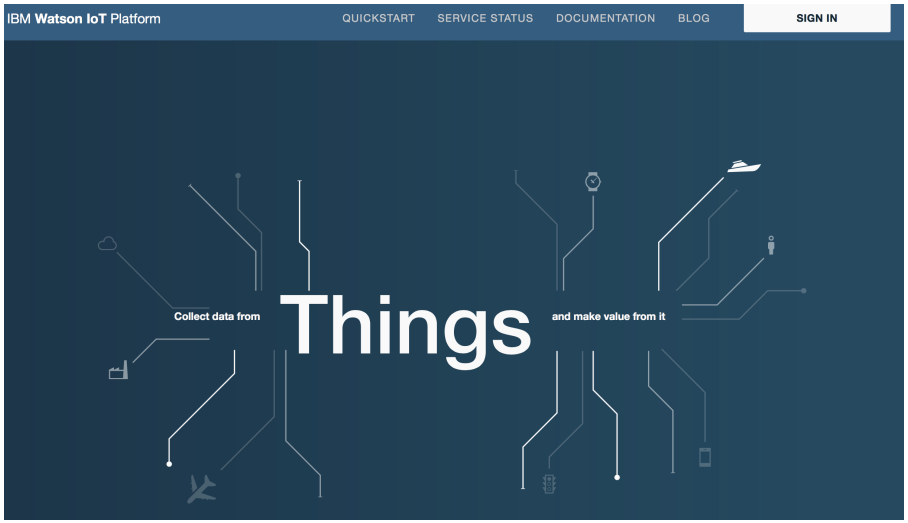
- Provides a common infrastructure to obtain value from industrial topologies
- Enable uniform communication, security, analytics and management layers for heterogenous IoT topologies
- Provide simpler and more agile models for building industrial IoT solutions

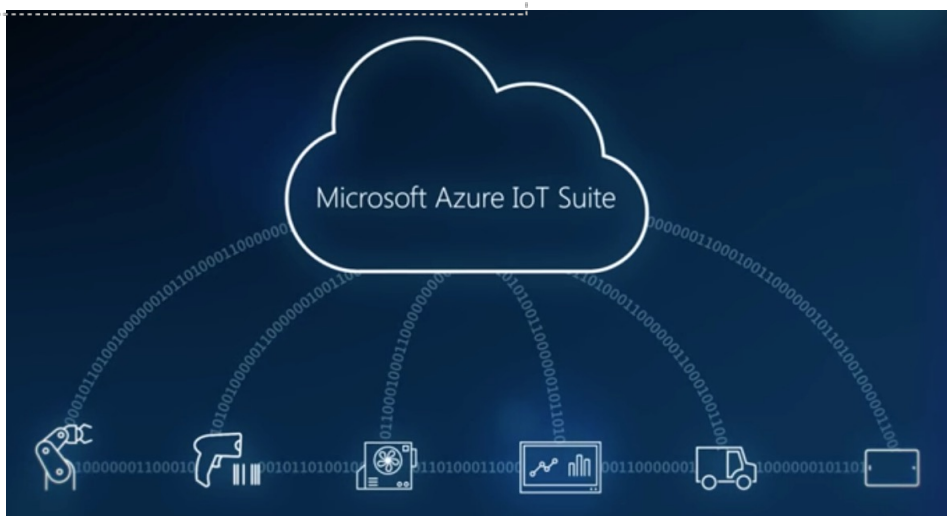
PaaS Examples

Sensing as a Service (S2aaS)

Cloud for IoT?







This lecture

- The Constrained Application Protocol (CoAP)
- MQTT
- Cloud service for IoT

Further readings

- Chapter 8, Internet of Things: a Hands-on Approach
- IETF RFC 7252 Constrained Application Protocol
- MQTT 3.1.1 specification