# COMP3310/6331 – #17

IoT and MQTT

*Dr Markus Buchhorn:* *markus.buchhorn@anu.edu.au*

# Applications choose their transport

- UDP-based applications:
  - Short messages
  - Simple request/response transactions
  - Light server touch
  - ARQ suffices

- TCP-based applications:
  - Larger content transfers
  - Longer, and more complex, sessions
  - Reliability matters
  - Packaging and presentation becomes important – TCP is a bytestream

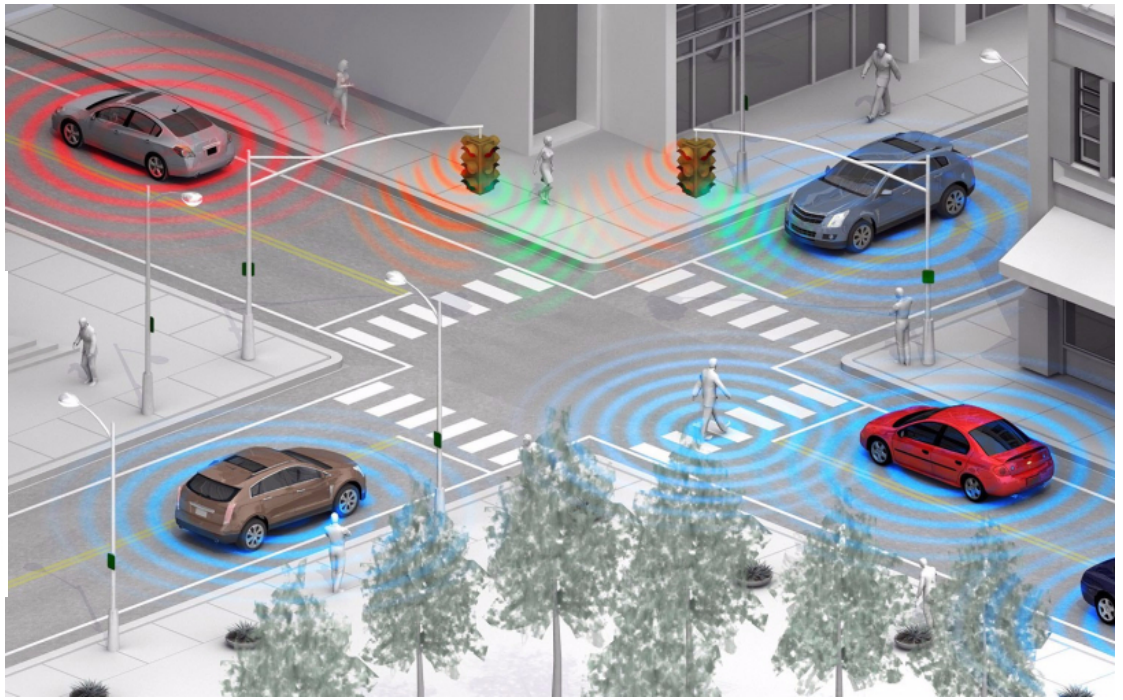
# Examples

- HTTP - TCP application – (more) reliable bytestreams
  - Exchange messages, command/responses, strong client/server relationship
- RTP - UDP applications – short messages, ARQ, low-delays
  - Send messages, hope they arrive, weak client/server relationship
- Useful when everything is a sizable computer, with good bandwidth
- What about IoT?

# What is IoT/IoE

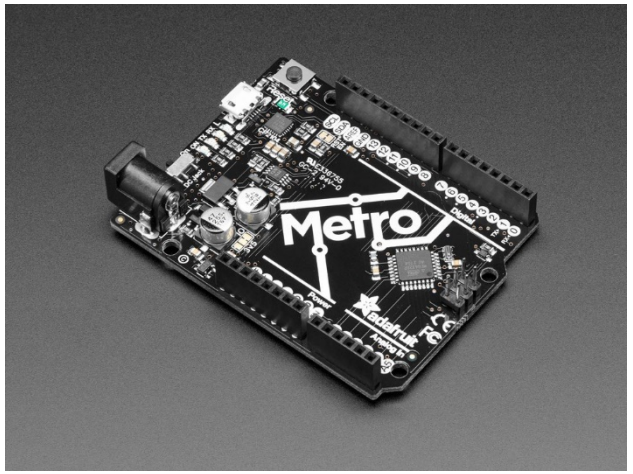
- Internet of Things, Internet of Everything
- Independent devices
  - acting on their own,
  - acting collectively
  - Sensors, controllers, …
  - Smart homes, smart cities
- From the large
  - Appliances
  - Vehicles

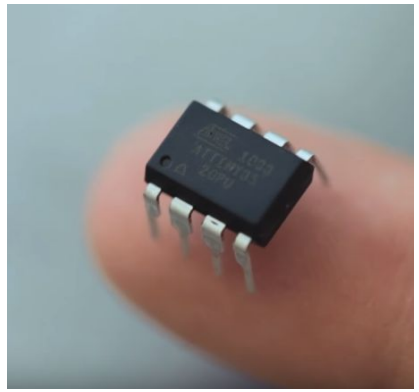


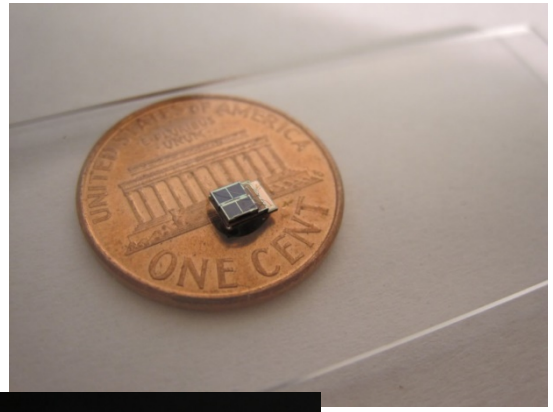


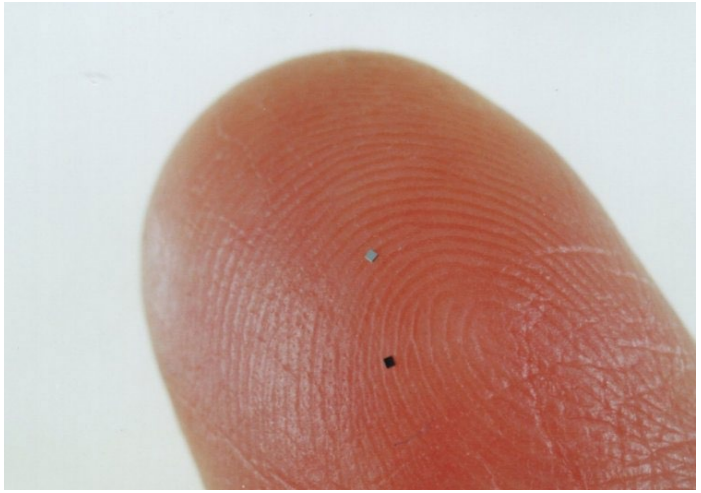
# What is IoT/IoE

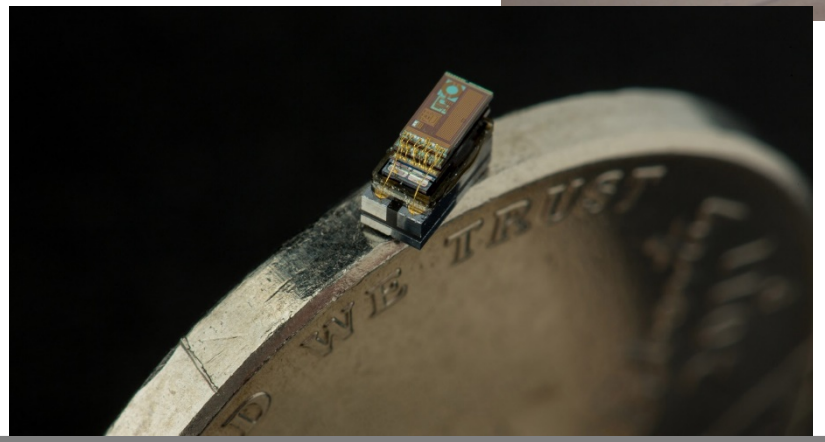
- To the small
  - Cover farms/battlefields, distribute across factories
  - Insert into bulk cargo, pipelines
  - Inject into people





# Measuring, monitoring, detecting, reacting, …

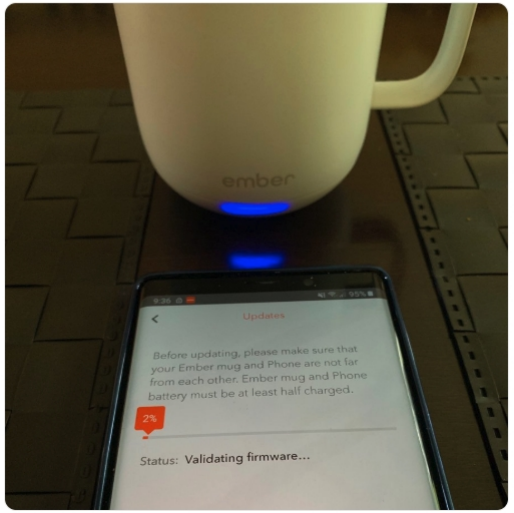
- *Focus on Machine-to-machine (M2M)*
- Engines, industrial equipment, predict failures
  - Temperatures, pressures, fluid levels
  - Noise – and what kind, compare to normal
- Weather, microclimates
- Ecological
  - Plant health, water quality, chemical/biological agents
- Traffic flows
- Presence of (bad) people in a space – face, gait, …
- Presence of (bad) cells in people
- Military applications…

# IoT, IoE, IoS…

- Just because you can, doesn't mean you should…

Unlocking your €100,000 car is now easier than ever


Updating the firmware of a mug. What a world we live in! /cc @internetofshit




What the world needs is a white guy who's obsessed with using citizen owned cameras to report on crime!

**Ring doorbell alarm company wants to cut crime in your neighborhood**

A year since Amazon bought Ring, CEO Jamie Siminoff says he won't quit until he sees a major drop in crime from more sales of his video doorbell products.

usatoday.com

# Why is IoT different?

1. **Scale**: Number of devices
2. **Power**: Ever smaller devices, doing smart/expensive things
3. **Networking**: Low power, remote locations, widely distributed
4. **Timeliness**: May need quick commands, responses
5. **Reliability**: Challenging – small devices

# What's needed?

1. **Scale**
   - No limits on number of devices (addresses) **and** relationships (connections)
     - *$N^2$ relationships, all storing state? Edges and routers*
   - Limit messages to avoid swamping networks
     - *1 billion devices at just 1 byte/sec…*

2. **Power**
   - Focus on minimal power needs – solar, batteries(!), RF
     - *Reduce transmission power*
     - *Turn off transmitters, …*
   - Do smart things elsewhere
     - *CPUs = power drain*

# What's needed?

## 3. Networking

- Limit transmission needs
  - *Reduce bandwidth/distance/# targets*
    - *Application and Protocol design, compression, heartbeats, …*
    - *Which devices/reports are crucial?*
  - *Take advantage of neighbourly assistance*
    - *Ad hoc mesh networks – needs better protocols, routing, transmission technologies, …*
    - *Trade-off: staying awake just to help neighbours?*

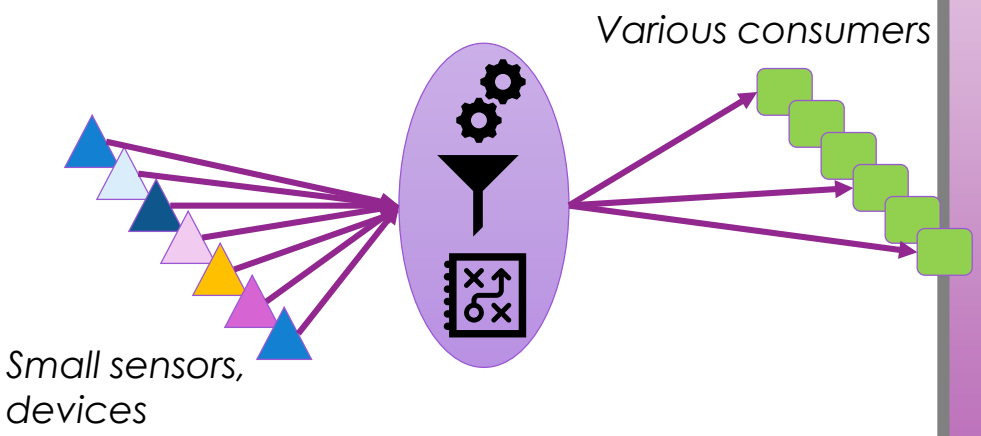
# What's needed?

4. **Timeliness**
   - Design accordingly
     - *Exceptions vs Regular Reports*
     - *Transmitter vs Receiver requirements*
     - *Short messages, prioritised messages*

5. **Reliability**
   - Add only where needed
   - Make it lightweight
     - *ARQ (push/pull) – or delegate it*

# Design: PubSub: Publication/Subscription

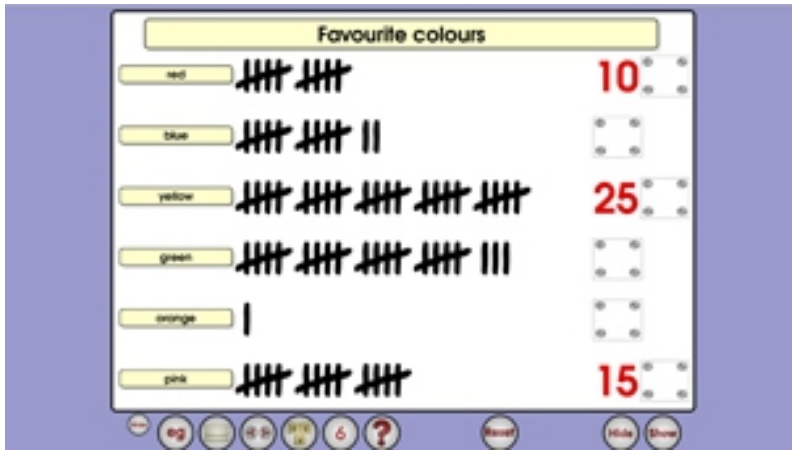
- Separate the '**announcement**' of data/state from '**consumption**'
  - Announcements: really easy
  - Consumption: as flexible as needed
    - Ask the server: *what do you have, …*
  - Allow for any type/number of consumers to subscribe
  - Allow for any type/number of sources to publish
  - Avoid 'connections'
- Needs a broker (or server)
  - Lightweight, fast, flexible, open, …
    - i.e. not a webserver



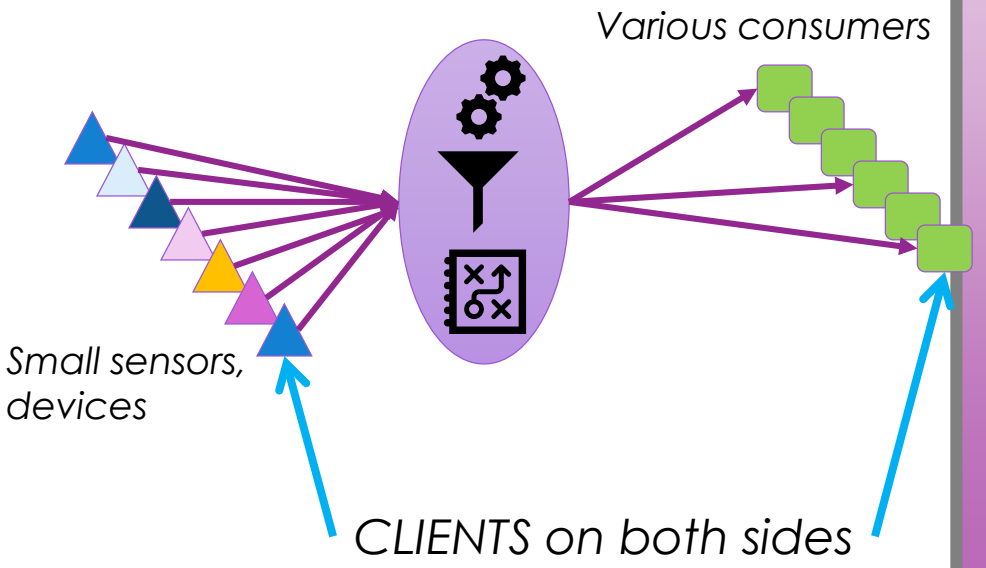
# MQTT

- Was *Message-Queueing Telemetry Transport* (1990s, IBM)
  - No longer queues (sort of) and less Telemetry-specific
- Runs over TCP (v3.1)
  - and (v5, May'18) UDP, and ZigBee and … as MQTT-SN
    - More scalable, more flexible, more lightweight, better error-reporting
- A "database" of key/value pairs
  - That deletes data as fast as it can.
- Standardised by OASIS, not IETF
  - Organisation for the Advancement of Structured Information Standards
  - Global industry association
  - Lots of business-related standards, markup languages, XACML/SAML, PKI, BPN, …

# MQTT for information sharing




- Shared whiteboard for information exchange
  - Publish *key=value* by writing
    - No arithmetic operations
  - Subscribe for reading
- HTTP: monitoring by asking, often, *for any given X*
- MQTT uses messages
  - As information
  - As 'triggers' (by listening clients)
  - So server/broker **pushes** messages
- Concept of 'topics'
  - Build your own database structure, on the fly!



# PubSub - **Pub**

- You (the sensor) **publish** a **message** to an MQTT broker
  - any (value) type (number, string, file, JSON, …)
    - Can include your own keywords, userId, timestamps, …
  - to a specific (key) "*topic/subtopic/sub-sub-topic/…*" – as you want

- "*Sensors/Paddock-A/Moisture/Sensor-1*" = 93%
- "*Sensors/Paddock-B/Temperature/Sensor-3*" = 28C

- "*PizzaPreferences/HouseMate/Malcolm*" = "*vegetarian, but no olives*"

# PubSub - **Sub**

- You (the consumer) **subscribe** to a particular **topic**
  - *No guarantee it exists! It may exist later…*
- or to a filtered-set (wildcard) of topics – (using **#** and **+**)
  - All temperature readings
  - All sensors from a given area

- *Sensors/Paddock-A/Moisture/Sensor-1* *(sensors/location/type/ID)*

- *Sensors/**#*** *(all sensors, anywhere)*
- *Sensors/**+**/Moisture/**+*** *(all moisture sensors, anywhere)*
- *Sensors/Paddock-A/**#*** *(all sensors, in Paddock-A)*

# Magic Topic(s)

- **$SYS/#**

- Holds very useful system information about the broker itself
  - Only broker publishes here
- Load, clients, bandwidth, storage, etc.

- Unfortunately:
  - Fields are not standardised (across brokers)
  - Resolution can be low (implementation-specific – 60sec?)

- Unwise to use MQTT to monitor health of your MQTT server?

# MQTT packets

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *Message Type* | | | | *Dup* | *QoS* | | *Rtn* | *Remaining Length* | | | | | | | | *Variable (optional) header(s)* | | | | | | | | | | | | | | | |
| ***Variable (optional) Payload(s)*** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

- Runs over TCP (up to v3.x), can also run over UDP and others (v5 onwards)
- Very bit-oriented
  - Very condensed messages, no plain English
  - Minimise load on publisher and network

# MQTT Messages

- 16 Messages types
  - Connect and Disconnect (and Ack)
    - Establish a channel and server state, and (you) identify yourself (lightweight security)
  - Ping request, and response
    - Server level, not ICMP
  - **Publish**, and Subscribe, Unsubscribe
    - Publish actually used both source->server and server->subscriber
  - Publish-Ack/Received/Released/Complete (various QoS guarantees)
  - Subscribe-Ack, Unsubscribe-Ack

# Main MQTT rule: minimalism

- Server wants to maintain the minimal possible amount of state
  - Subscribers: “Temporarily unreachable” or “no-longer interested”?
- Server does **not** ‘queue’ messages
  - Once messages are pushed to all subscribers, they are deleted (*)
  - Published messages for topics with no subscribers are deleted (*)

*(*) mostly…*

- Give the server every chance to clean up its database

# QoS

- "Quality of Service"
- What guarantees can you give me about this service?
- *Is it timely, reliable, accurate, trustworthy, …*

- The Internet can have QoS features
  - MPLS, DiffServ, RSVP, CoS, …

- MQTT has QoS at the application level
  - Because some subscribers need to be **sure**
  - Because subscribers can join at **any time**
  - *Is that power off, is that gate open, how full is the tank now, …*

# MQTT Quality of Service (QoS)

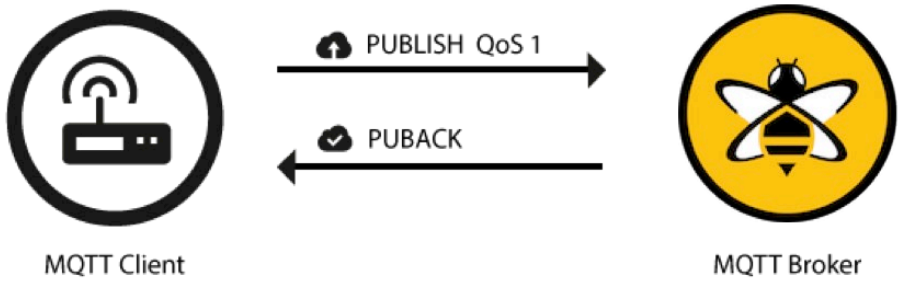
- Three levels (0, 1, 2)
  - Each with more load, storage, bandwidth, energy implications

- Level 0: (default)
  - **"Fire-and-Forget"**
  - Client/Server pushes a message out, then deletes it.


*From hiveMQ.com*

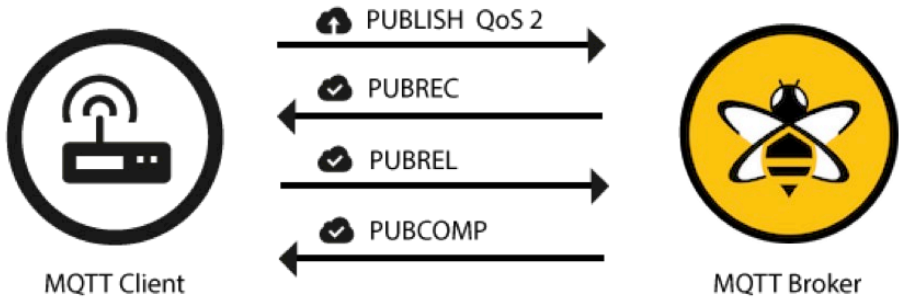
# MQTT Quality of Service (QoS)

- Level 1:
  - **"At least once"**
  - Guaranteed delivery, requires confirmation, but could transmit duplicates



# MQTT Quality of Service (QoS)

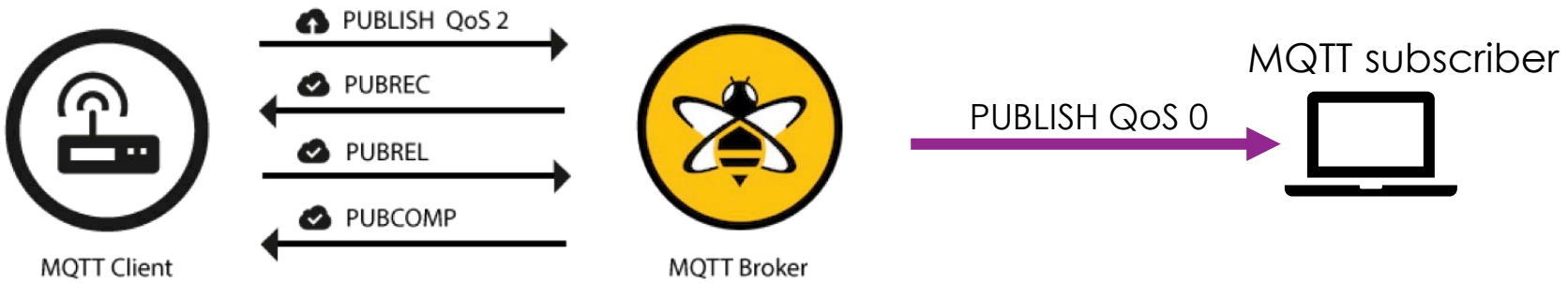
- Level 2:
  - **"Exactly once"**
  - Guaranteed delivery, 4-step handshake, with no duplicates
    - Lots of energy, time, storage, …



# QoS is where?

- MQTT QoS is part of the SUBSCRIBE/PUBLISH set up
  - Which is used source->server **and** server->subscriber

- Can have reliable publishing turn to unreliable delivery
  - *And vice versa?*




- *And this is for every single subscriber*

# MQTT – "Last Known Good"

- Sensors that rarely report,
  - E.g. state changes: door open/close
  - E.g. very remote, low-power, "expensive" sensors
- Need to give new subscribers something to start from
  - Could be waiting a long time…
- "**Retain**" flag
  - Retained by server
  - Even across reboots
  - Sent on first subscription request by client

# MQTT – "Last Will and Testament"

- Sources can publish a default message for any topic(s) (e.g. client/status)
  - A retained message, but not sent immediately
- If server e.g.
  - Does not receive a published message after <*KeepAlive*> period, or
  - Sees (TCP) connection dropped without (MQTT) disconnect, then
  - Assumes connection is lost, source has failed, …
- Can then inform subscribers (new and old)
  - E.g. "Service temporarily/permanently down/redirected; contact x@y.com"
- QoS considerations. KeepAlive considerations.

# MQTT – "Clean Session"?

- Flagged on connection
- **Clean**: Pretend I'm brand new

- **Not clean**: Ask server to remember you – aka **Persistent** session
  - In case you drop off

- Pain for the server:
  - Store all your subscribed topics
  - Store all messages (with QoS 1 and 2)
  - Push in burst when reconnected – give me everything I've missed

# MQTT in a smart home

- Attach sensors
  - Brightness, temperature, humidity, movement, voice, locks, …
  - **Each publishes to a state topic**
    - Sensors/Temperature/Lounge = 18
    - Sensors/LightLevel/Lounge = 10%
- Allows you to monitor the environment
  - Lots of charts over time

*SuperHouse.TV*

# MQTT in a smart home

- Attach controllable devices
  - Lights, heaters, coolers, curtains, locks, AV system, …
  - **Each device subscribes to a command/state topic that you write to**
    - Lights/Lounge/Light-27 = Off [On, 10%, 50%]
    - Heater/Lounge = Off [On]

- Attach controllers
  - Physical switches, web-client, app, Alexa/Google/Siri, … (at the same time!)
  - **Each publishes to a command/state topic**
    - Switches/Lounge/Switch-19 = On [Off]
    - Thermostat/Lounge = 22

# MQTT in a smart home

- *Note: **controllers** are not directly publishing to the **controlled***
  - Gives you way more flexibility
  - Able to modify behaviours all in software, no hardware changes needed

- Connect topics by a **Rules Engine**
  - Given X (is published), do Y (publish something) *[e.g. NodeRed, IFTTT.com]*

- Overseen by a **State Machine**
  - Store state, Note changes, Combine rules, Create scenes *[e.g. OpenHAB]*
  - Bring in extra information (time of day, weather forecast, …)

# MQTT Security?

- **It has some!** (if enabled…)
- Username/password
- Client identifier (64kB!)
- Role-based Access Control
- X509 Certificates
- Payload signatures, integrity checking
- Encrypted connections
- …

# IoT security?

- **Ha!**
- Firmware v1.0 – 5 years later?
- Standard admin login?
- Standard access URLs? (google-able)
- Cloud gateways?

- Web-cameras, baby monitors, powerboards, …
- Take over for
  - local attack (home and home network)
  - external attack (Denial-of-Service/flooding)

# Do try this at home!

- Several Open Source implementations
  - Extremely lightweight
  - Run on RaspberryPI, old PCs, …
- Several Hosted/Cloud implementations
  - At varying levels of cost
  - Some VMs, Docker containers
- Various Public servers
  - You can read the temperature at over 10,000 sites around the world.
- Gateways to other services
  - IFTTT.com