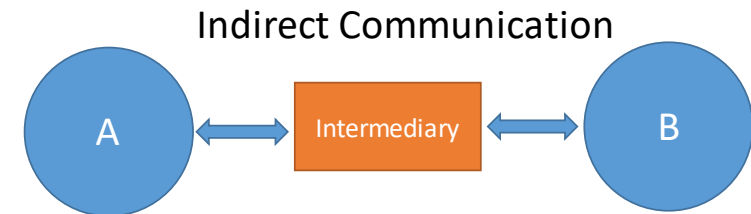
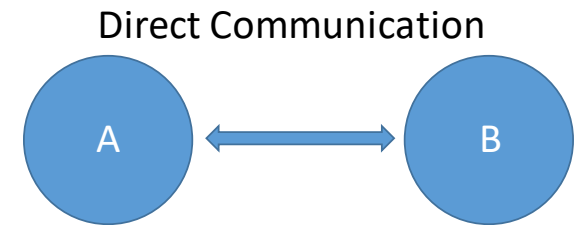


Indirect Messaging

Publish Subscribe

Using the “Middleman”

- Communication between processes using an intermediary
 - Sender → “The middle-man” → Receiver
 - No direct coupling
- Up to now, only considered Direct Coupling
 - Introduces a degree of rigidity
- Consider...
 - What happens if a device fails during communication in Direct Coupling?
 - What if you'd like to add
- Two important properties of intermediary in communication
 - **Space uncoupling** (devices don't need to "know" about each other)
 - **Time uncoupling** (devices don't need to be "available" at the same time)



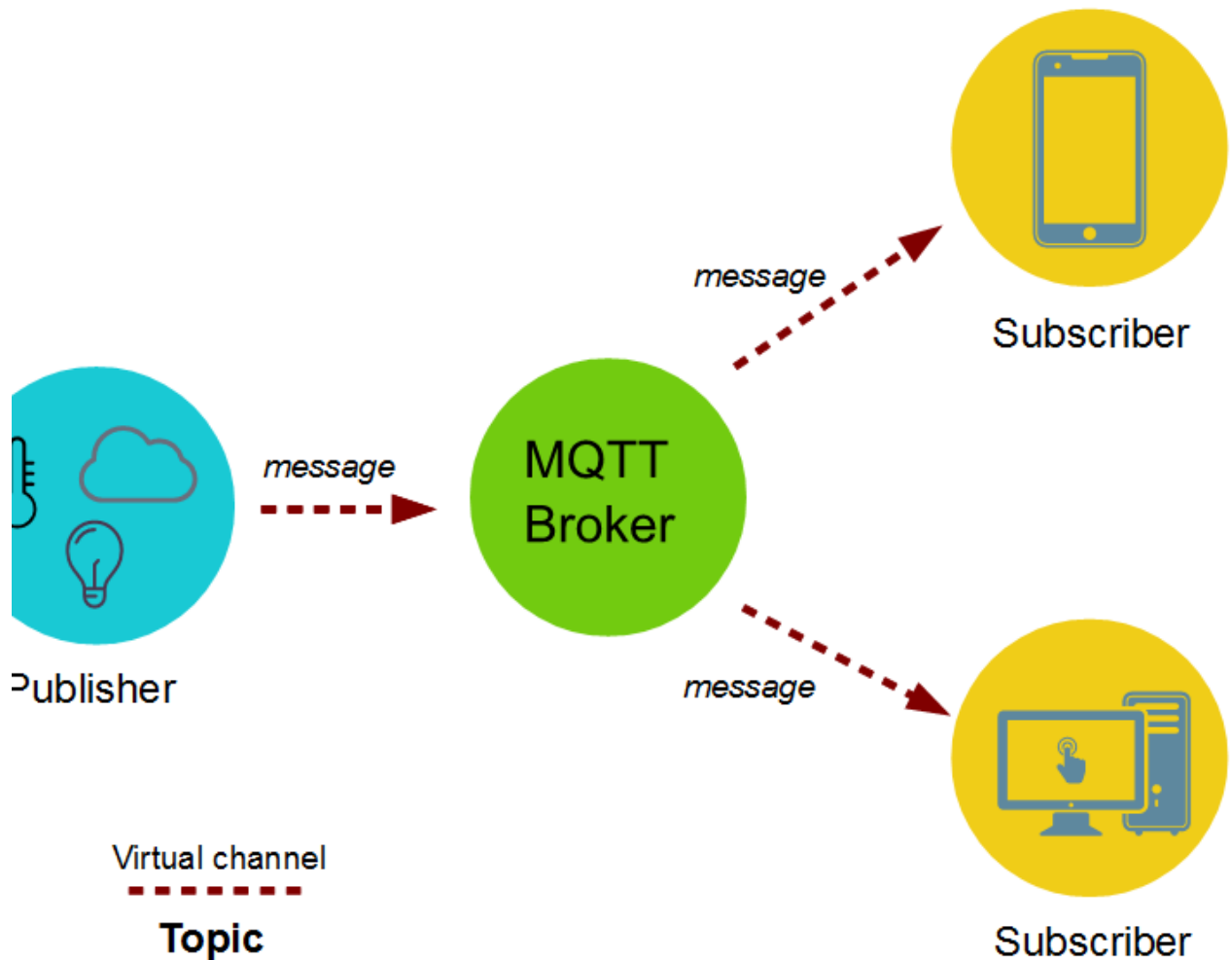


MQTT

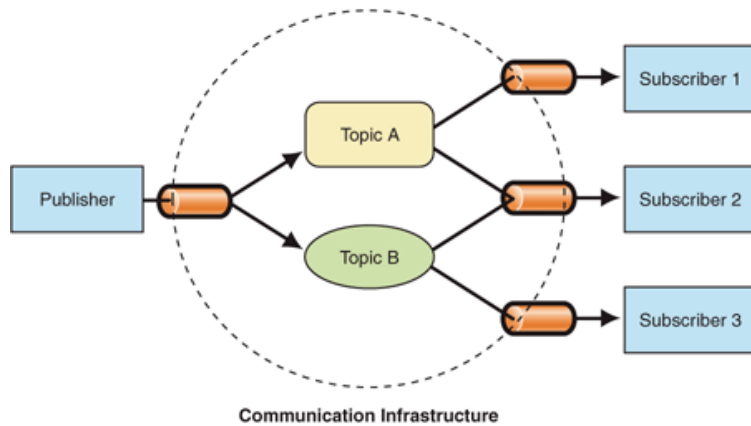
- MQ Telemetry Transport (MQTT)
- Telemetry
 - Remote measurements
- Created by IBM
 - from message queueing (MQ) architecture used by IBM for service oriented networks.
 - Telemetry data goes from devices to a server or broker.
 - Uses a **publish/subscribe** mechanism.
- Lightweight both in bandwidth and code footprint

MQTT – publish subscribe

- **Topics/Subscriptions:** Messages are published to topics.
 - Clients can subscribe to a topic or a set of related topics
- **Publish/Subscribe:** Clients can subscribe to topics or publish to topics.



Publish Subscribe Process



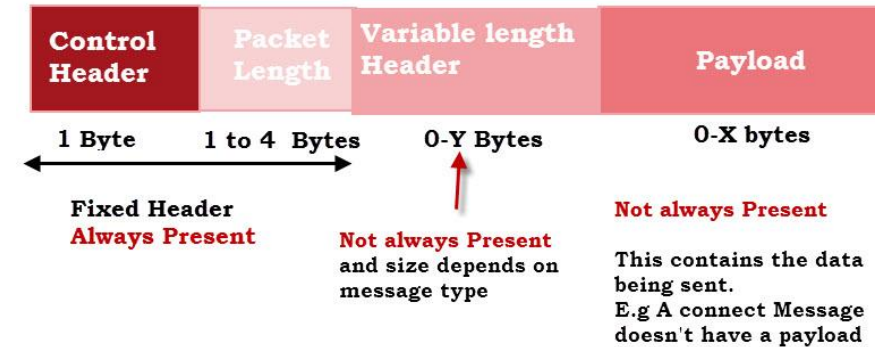
- A message is published once by a publisher.
- Many things can receive the message.
- The messaging service, or “broker”, provides decoupling between the producer and consumer(s)
- A producer sends (publishes) a message (publication) on a topic (subject)
- A consumer subscribes (makes a subscription) 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

Publish-Subscribe Characteristics

- A published messages may be retained
 - A publisher can mark a message 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
- A Subscription can be durable or non-durable
 - Durable: messages forwarded to subscriber immediately, If subscriber not connected, message is stored and forwarded when connected
 - Non-Durable: subscription only active when subscriber is connected to the server / broker

MQTT Characteristics

- MQTT protocol compresses to small number of bytes
 - Smallest packet size 2 bytes
 - Supports always-connected and sometimes connected
 - Provides Session awareness
 - “Last will and testament” enable applications to know when a client goes offline abnormally
 - Typically utilises TCP-based networks.



MQTT Standard Packet Structure

MQTT Characteristics

- Three quality of service levels:
 - 0 = At most once (Best effort, No Ack),
 - 1 = At least once (Acked, retransmitted if ack not received),
 - 2 = Exactly once [Request to send (Publish), Clear-to-send(Pubrec), message (Pubrel), ack (Pubcomp)]
- Retained Messages
 - Server keeps messages even after sending it to all subscribers. New subscribers get the retained messages

MQTT is Open Source

- Lots of implementations:
 - Mosquitto
 - Micro broker
 - Really small message broker (RSMB): C
 - Cloud broker services



MQTT vs HTTP

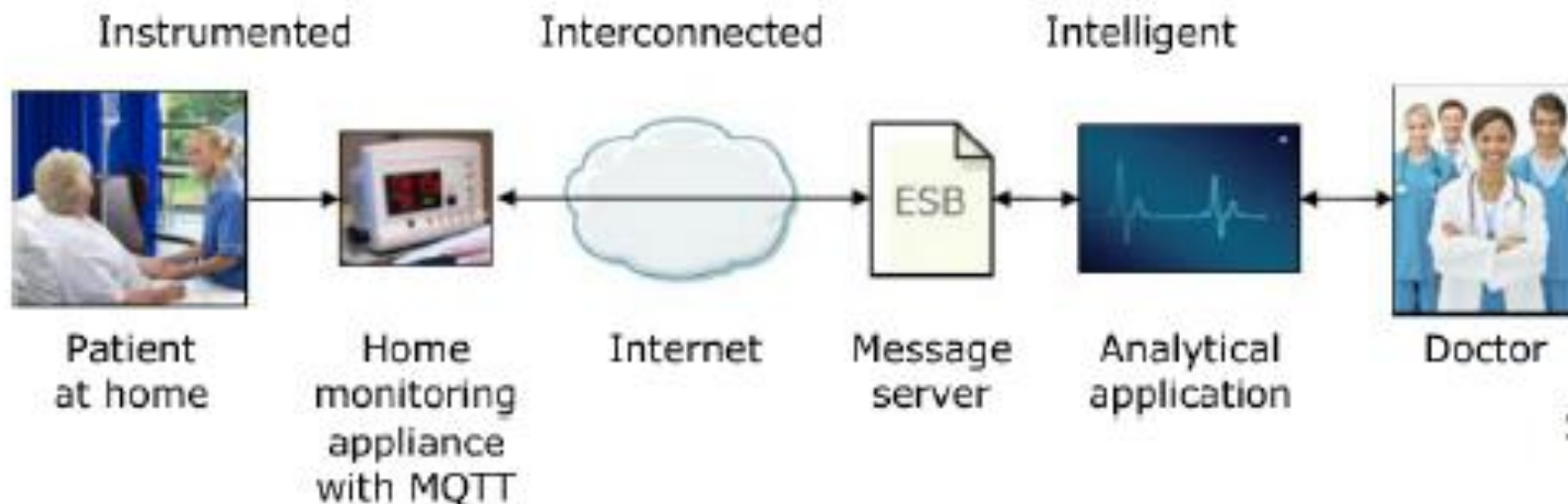
- Push delivery of messages / data / events
 - MQTT – low latency push delivery of messages from client to server and **server to client**. Helps bring an event oriented architecture to the web
 - HTTP – push from client to server but poll from server to client
- Reliable delivery over fragile network
 - MQTT will deliver message to QOS even **across connection breaks**
 - Decoupling and publish subscribe – **one to many delivery**

| Characteristics | | 3G | | WIFI | |
|------------------|--------------------------------------------|--------------|-------------|--------------|-------------|
| | | <i>HTTPS</i> | <i>MQTT</i> | <i>HTTPS</i> | <i>MQTT</i> |
| Receive Messages | Messages / Hour | 1,708 | 160,278 | 3,628 | 263,314 |
| | Percent Battery / Hour | 18.43% | 16.13% | 3.45% | 4.23% |
| | Percent Battery / Message | 0.01709 | 0.00010 | 0.00095 | 0.00002 |
| | Messages Received (Note the losses) | 240 / 1024 | 1024 / 1024 | 524 / 1024 | 1024 / 1024 |
| Send Messages | Messages / Hour | 1,926 | 21,685 | 5,229 | 23,184 |
| | Percent Battery / Hour | 18.79% | 17.80% | 5.44% | 3.66% |
| | Percent Battery / Message | 0.00975 | 0.00082 | 0.00104 | 0.00016 |

*sending and receiving 1024 messages of 1 byte each.(source: <https://www.ibm.com/developerworks>)

Application Example

- Home care monitoring solution
 - Home and patient instrumented with sensors.
 - E.g. door motion, blood pressure, pacemaker/defib.
 - Collected by monitoring service (broker) using MQTT
 - Subscribed by a health care service in the hospital
 - Alerts relations/health care profs. if anything is out-of-order



Source: Lampkin 2012