Introduction to MQTT

May 15th 2013

Webinar Presenters

- Dave Locke
  - locke@uk.ibm.com

Why isn't HTTP enough?

- The HTTP standard revolutionized how *people* consume data
  - A single simple model: Send a request, read the response
  - Available via any tablet, laptop, phone, PC etc.
- The Internet of Things has fundamentally different challenges
  - HTTP remains ideal for requesting data from a known source
  - We also need an event-oriented paradigm:
    - Emitting information *one to many*
    - Listening for events *whenever they happen*
    - Distributing minimal packets of data in *huge volumes*
    - *Pushing* information over *unreliable networks*

Additional comms challenges for Mobile and M2M apps

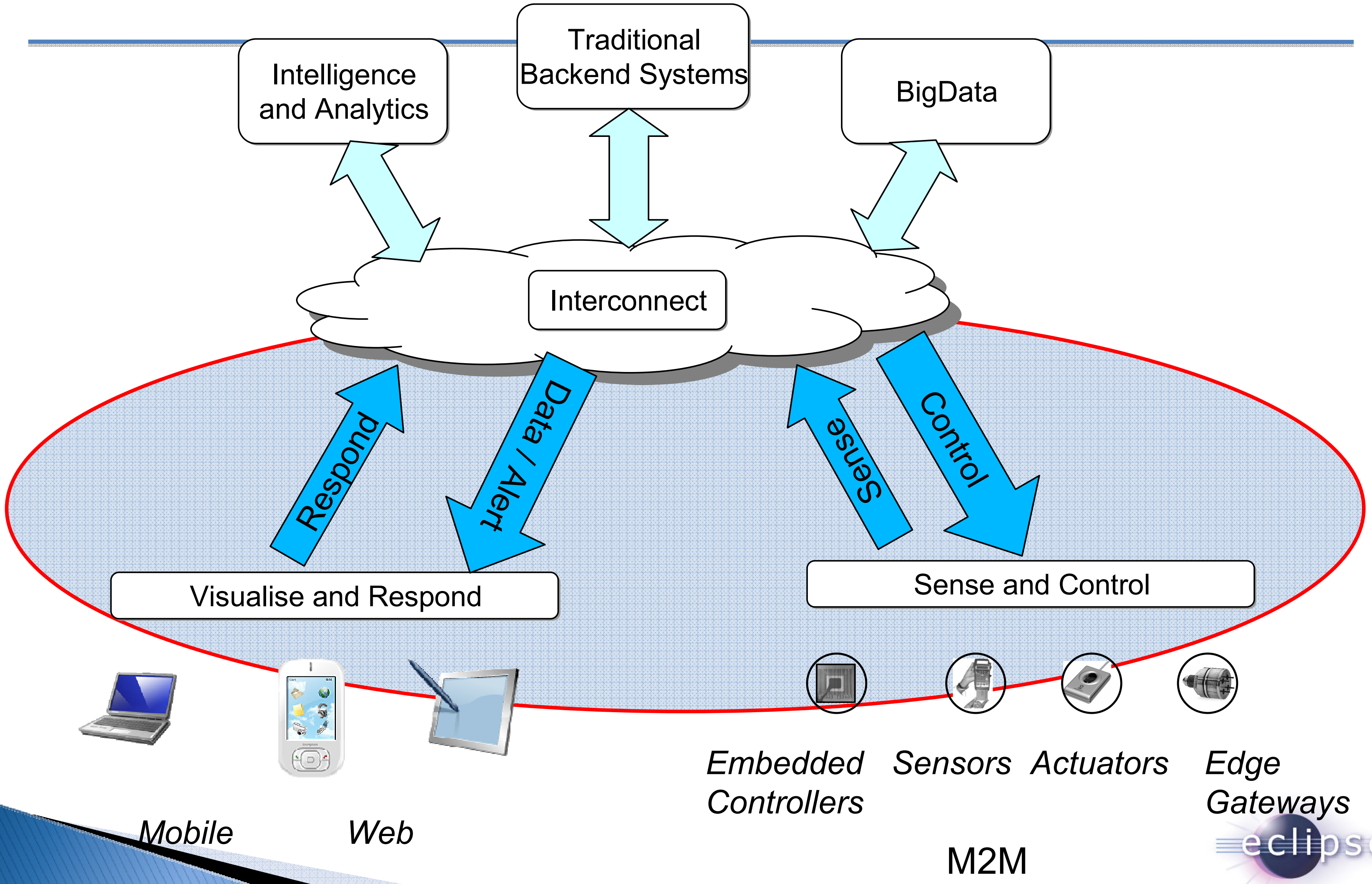
- Volume (cost) of data being transmitted (especially in M2M with limited data plans)
- Power consumption (battery powered devices)
- Responsiveness (near-real time delivery of information)
- Reliable delivery over fragile connections
- Security and privacy
- Scalability

MQTT in a Nutshell

- MQTT == MQ Telemetry Transport
- In a nutshell

  “A light weight event and message oriented protocol allowing devices to asynchronously communicate efficiently across constrained networks to remote systems”

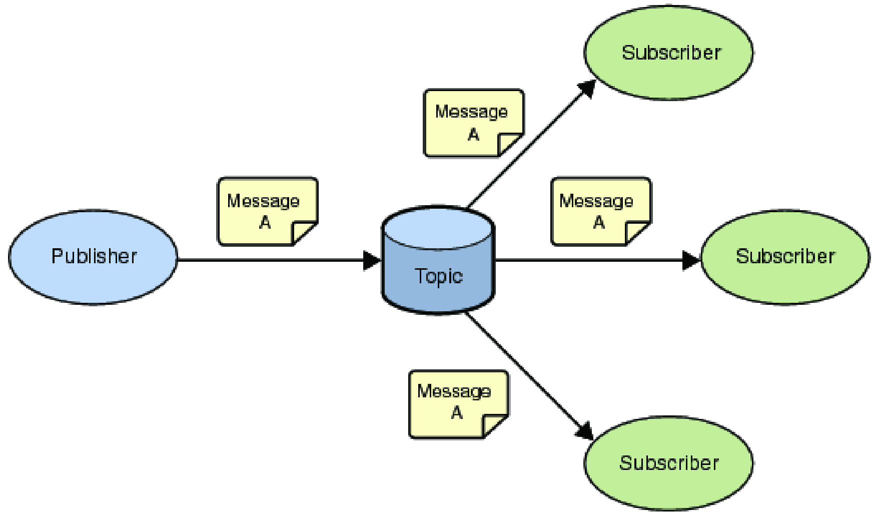
The Realm of MQTT



Background to MQTT / Original Design Goals

- To make it simple to connect the M2M (physical) world to the traditional IT world
- Expect and cater for frequent network disruption – built for *low bandwidth*, *high latency*, *unreliable*, *high cost* networks (cost per byte)
- Expect that client applications may have very limited resources available (8 bit controller, 256kb ram)
- Provide loose coupling to support dynamic system environments where high volumes of physical world messages and events need to be made available to enterprise servers and other consumers in ways that may not have been originally anticipated.
- Provide multiple deterministic message delivery qualities of service to reflect tradeoffs between bandwidth, availability, and delivery guarantees
- Capable of supporting large numbers of devices (10k MQTT clients)
- Simple for application developers and implementers of the protocol
- Publish the protocol for ease of adoption by device vendors and third-party client software enabling support for a proliferation of platforms, technologies and networks that are driven by very diverse equations of cost, technology and physical constraints.
- To be industry agnostic

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

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*
      - *<country>/<region>/<town>/<postcode>/<house>/alarmState*
    - *And subscribes for control commands:*
      - *<country>/<region>/<town>/<postcode>/<house>/thermostat/setTemp*

- 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
  - Cannot 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

Use on Constrained Networks

- 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. Webscokets
  - Tested on many networks – vsat, gprs, 2G….

- Provides multiple deterministic message delivery qualities of service
  - 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*

Constrained Device

- Designed for constrained devices:
  - Suited to applications / devices that may have limited resources available
    - 8 Bit controllers upwards
    - Battery
  - Multiple MQTT client implementations available in many form factors / languages
    - Tiny footprint MQTT client (and server) libraries e.g. a c client lib in 30Kb and a Java lib in 64Kb

Open and Easy to use

- The MQTT specification is open and royalty free for ease of adoption
  - http://www.ibm.com/developerworks/webserices/library/ws-mqtt/index.html

- Is industry agnostic
  - Move a payload / data / event for any form of data

- Many implementations from enterprise scale fully supported through open source and hobbyist
  - See http://mqtt.org for full details

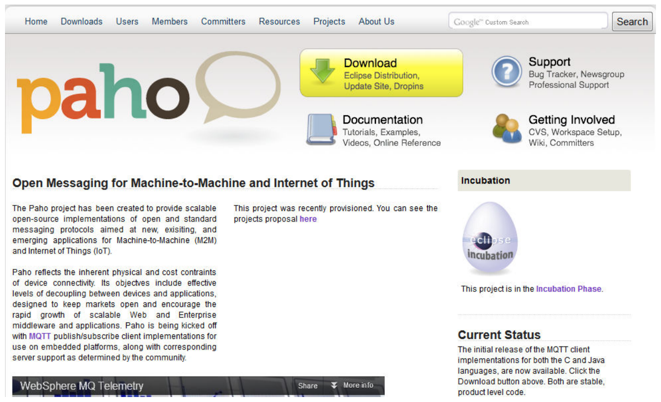
- API is simple to use via small set of well defined verbs

Open and Easy to use

- Open royalty free specification

- Open Source
  - Eclipse.org Paho project
  - IBM Contributed MQTT clients

- MQTT Implementations
  - Hobbyist to Enterprise
  - Open Source to Commercial
  - 16+ MQTT severs
  - 40+ MQTT clients
  - See http://mqtt.org



Open Messaging for Machine-to-Machine and Internet of Things

The Paho project has been created to provide scalable open-source implementations of open and standard messaging protocols aimed at new, exisiting, and emerging applications for Machine-to-Machine (M2M) and Internet of Things (IoT).

Paho reflects the inherent physical and cost contraints of device connectivity. Its objectves include effective levels of decoupling between devices and applications, designed to keep markets open and encourage the rapid growth of scalable Web and Enterprise middleware and applications. Paho is being kicked off with MQTT publish/subscribe client implementations for use on embedded platforms, along with corresponding server support as determined by the community.

This project was recently provisioned. You can see the projects proposal here

Incubation

This project is in the Incubation Phase.

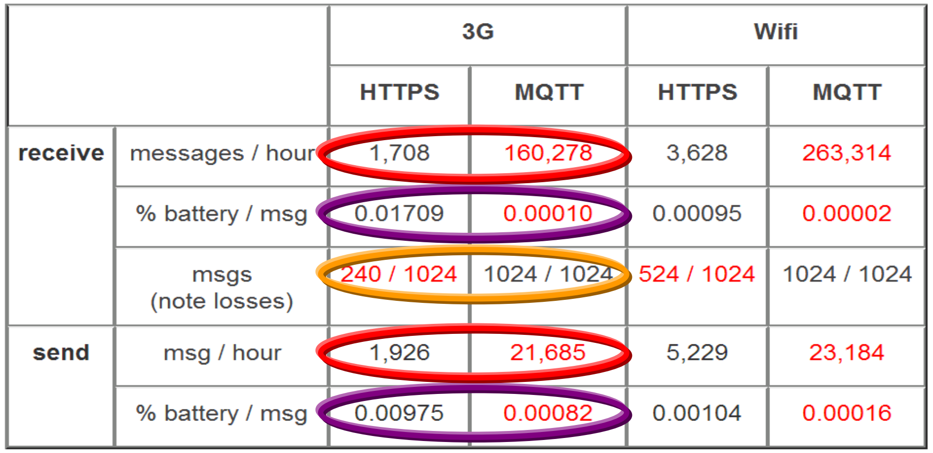
Current Status

The initial release of the MQTT client implementations for both the C and Java languages, are now available. Click the Download button above. Both are stable, product level code.

- Standard
- MQTT is being standardised at OASIS

Benefits of MQTT verses 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
- Efficient use of network
  - For an M2M project the number of bytes with MQTT was **137130 bytes per device per month** with HTTP the number of bytes was **801000 bytes per device per month**
- Reliable delivery over fragile network
  - MQTT will deliver message to QOS even **across connection breaks**
- Decoupling and publish subscribe – **one to many delivery**

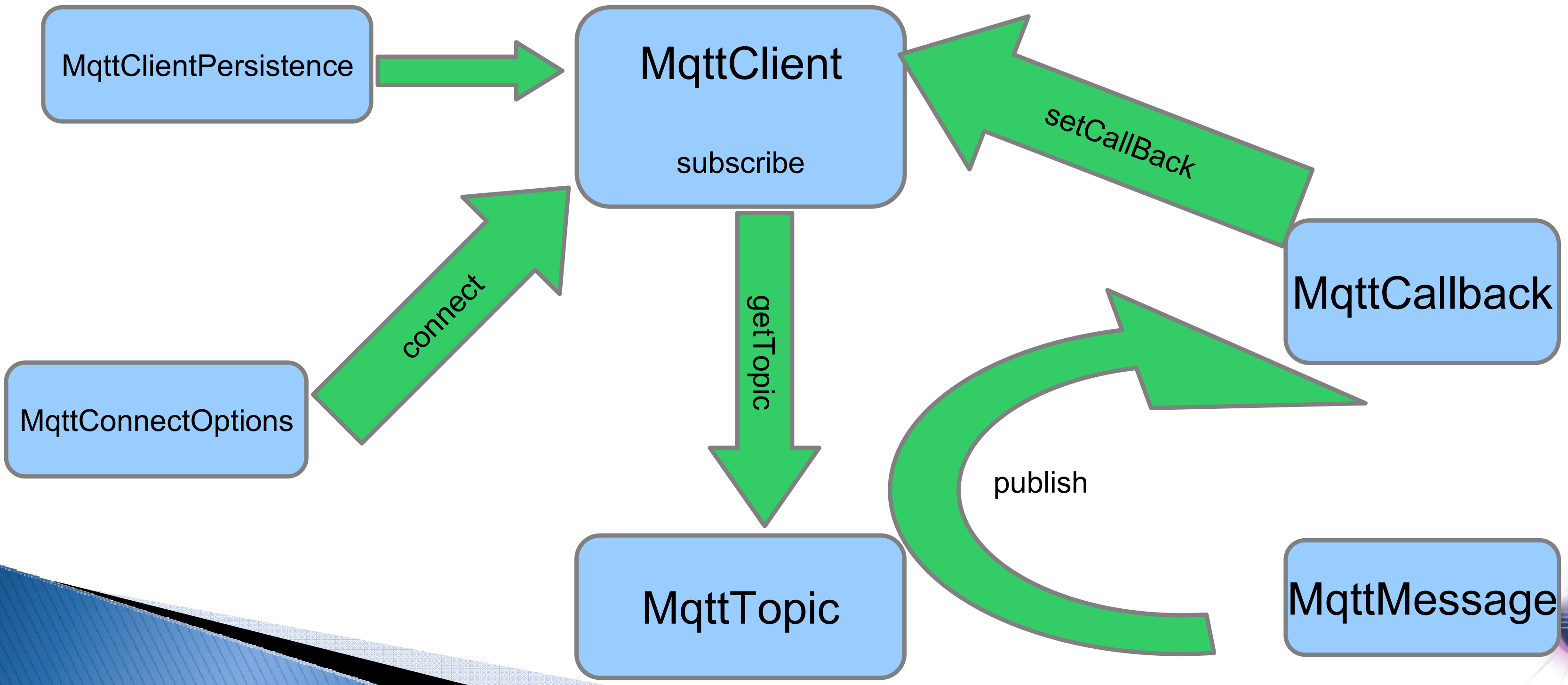


		3G		Wifi	
		HTTPS	MQTT	HTTPS	MQTT
receive	messages / hour	1,708	160,278	3,628	263,314
	% battery / msg	0.01709	0.00010	0.00095	0.00002
	msgs (note losses)	240 / 1024	1024 / 1024	524 / 1024	1024 / 1024
send	msg / hour	1,926	21,685	5,229	23,184
	% battery / msg	0.00975	0.00082	0.00104	0.00016

Source: http://stephendnicholas.com/archives/1217

Programming with Paho Java API

- MqttClient object is the starting point
- Messages and notifications received via callbacks
- Publishes made asynchronously
- Local persistence store used to achieve high QoS levels



Connect

First step is to create an MqttClient object

- Specify optional persistence store, URI and ClientID
- ClientID must be unique for the broker it is connecting to

```
MqttClientPersistence persistence = new MqttDefaultFilePersistence("/tmp");
MqttClient client = new MqttClient("tcp://localhost:1883", "MQTTSub",persistence);
```

Then specify connection options, and connect!

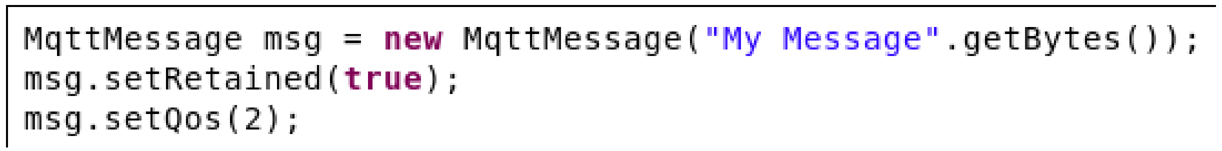
- Keep alive of 480 seconds
- A retained publication Will message with QoS of 1

```
MqttConnectOptions opts = new MqttConnectOptions();
opts.setKeepAliveInterval(480);
opts.setWill(client.getTopic("WillTopic"), "Something bad happened".getBytes(), 1, true);
client.connect(opts);
```

Create a message and Publish

Create a message

- Message properties allow it to be set as a retained publication and what the QoS needs to be
- Message payload is always a byte array



```
MqttMessage msg = new MqttMessage("My Message".getBytes());
msg.setRetained(true);
msg.setQos(2);
```

To send a message

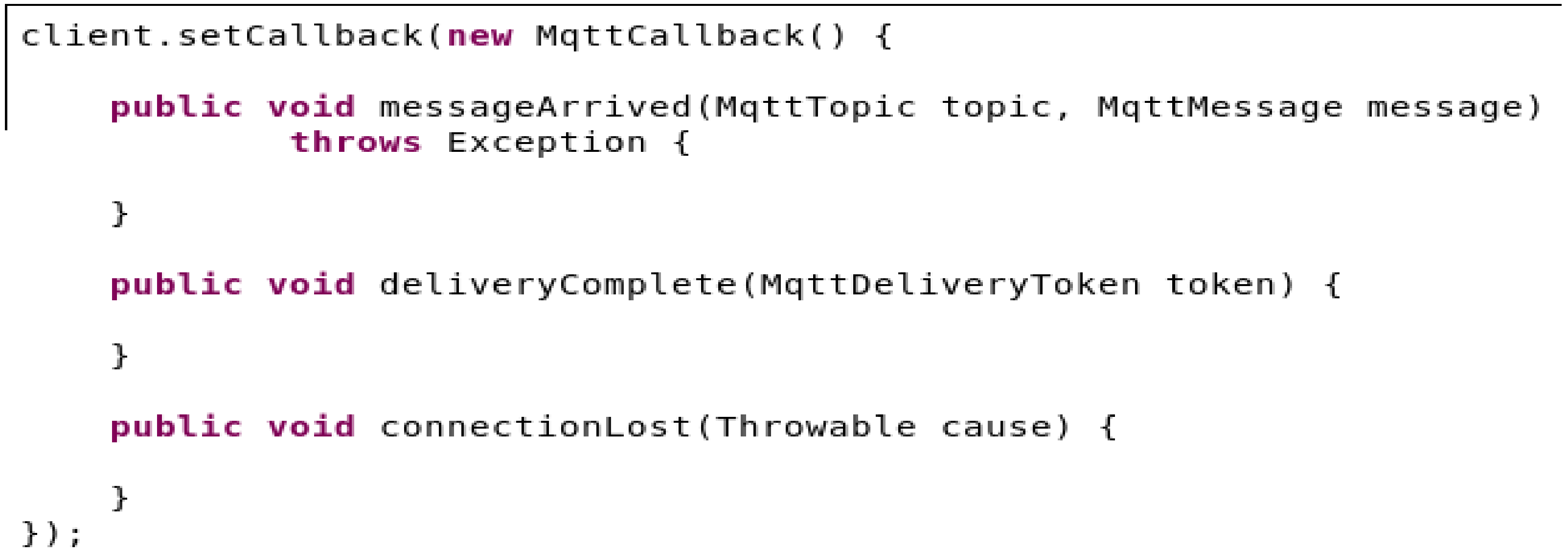
- Get a topic object and publish
- The returned DeliveryToken is used to determine when delivery is complete

```
MqttTopic topic = client.getTopic("Fruit/Grape/Red");
MqttDeliveryToken token = topic.publish(msg);
```

Setup callback

Messages are delivered via a callback mechanism

– This is also used to indicate when the connection is broken and a publish has completed



```
client.setCallback(new MqttCallback() {

    public void messageArrived(MqttTopic topic, MqttMessage message)
            throws Exception {

    }

    public void deliveryComplete(MqttDeliveryToken token) {

    }

    public void connectionLost(Throwable cause) {

    }
});
```

Subscribe for messages

After connecting, subscribe by providing the topic string:

– Messages will then be delivered to the callback

```
client.subscribe("Fruit/#");
```

Can subscribe to multiple topics at the same time, and provide QoS levels:

```
client.subscribe(new String[]{"MyTopic","Fruit/#"},new int[]{2,1});
```

To stop receiving messages, unsubscribe.

```
client.unsubscribe(new String[]{"MyTopic","Fruit/#"});
```

Java API : Further notes

To resume a previous session, set the clean session option to false

- Requires the application to use the same client identifier
- When false subscriptions are durable
- When true previous state and subscriptions are cleaned up at start and / or end of session and subscriptions are non-durable

```
opts.setCleanSession(false);
```

To cleanly disconnect

```
client.disconnect();
```

MQTT technology

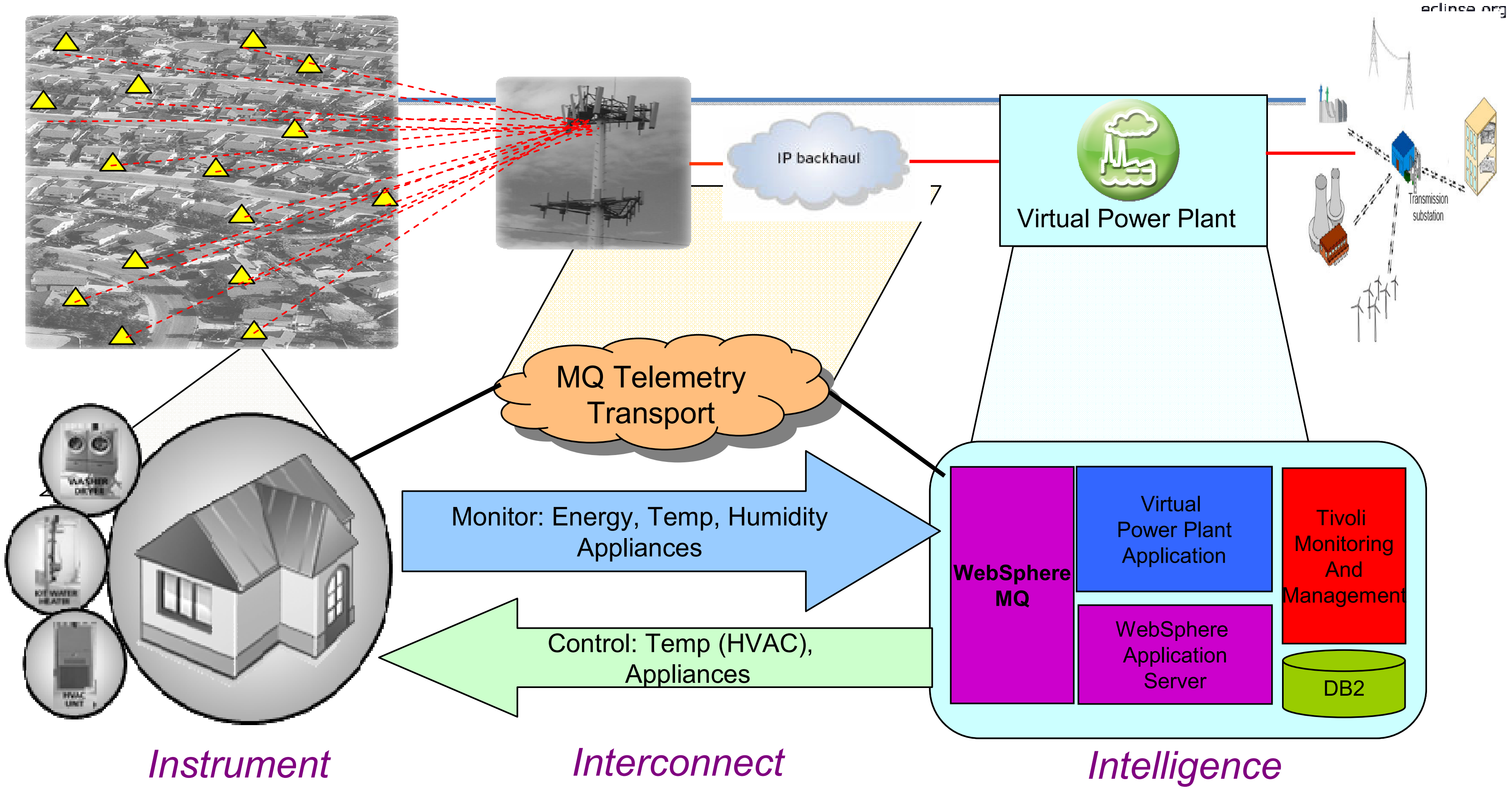
- In Paho
  - Java, JavaScript C and Lua client libraries
  - Utilities
- Others client libraries outside Eclipse
  - Python, Perl, Ruby...
  - See http://mqtt.org/software for a list
- Server implementations
  - Also http://mqtt.org/software

Facebook Messenger

Lucy Zhang, a software engineer at Facebook, has written about their new Facebook Messenger app:

*"One of the **problems** we experienced was **long latency** when sending a message. The method we were using to send was reliable but **slow**, and there were **limitations** on how much we could improve it. With **just a few weeks** until launch, we ended up building a new mechanism that maintains a persistent connection to our servers. To do this without **killing battery life**, we used a protocol called **MQTT** that we had experimented with in Beluga. MQTT is specifically designed for applications like sending telemetry data to and from space probes, so it is **designed to use bandwidth and batteries sparingly**. By maintaining an MQTT connection and routing messages through our chat pipeline, we were able to often achieve **phone-to-phone delivery in the hundreds of milliseconds, rather than multiple seconds**."*

A Virtual Power Plant with MQ Telemetry

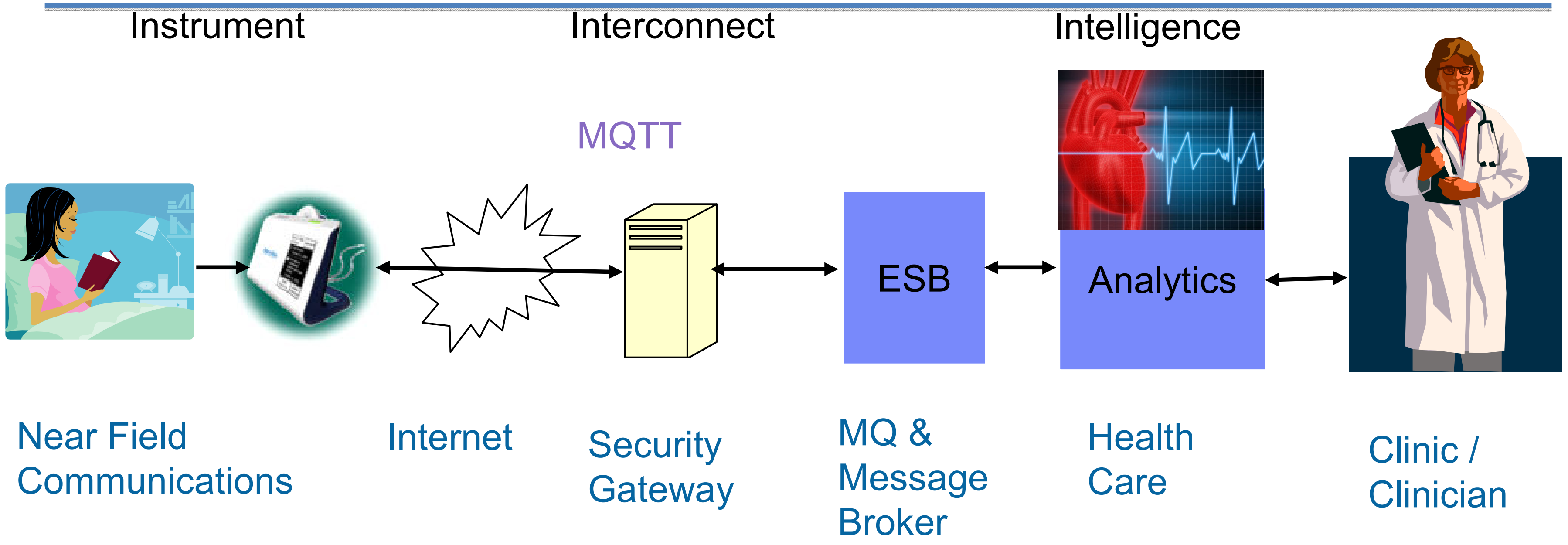


IP backhaul
Virtual Power Plant
Transmission substation
MQ Telemetry Transport
WASHER DRYER
HOT WATER HEATER
HVAC UNIT
Monitor: Energy, Temp, Humidity Appliances
Control: Temp (HVAC), Appliances
WebSphere MQ
Virtual Power Plant Application
WebSphere Application Server
Tivoli Monitoring And Management
DB2
Instrument
Interconnect
Intelligence

An Intelligent Utility Network

- Smart homes enabled with smart devices (HVAC, appliances…)
- Devices connect to a home gateway box via Zigbee or Homeplug
- Home gateway monitors devices collecting data that impacts energy usage
- Home gateway publishes energy data to Virtual Power Plant (VPP) every 5 mins over a mobile network
- 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

Home Pace Maker Monitoring Solution



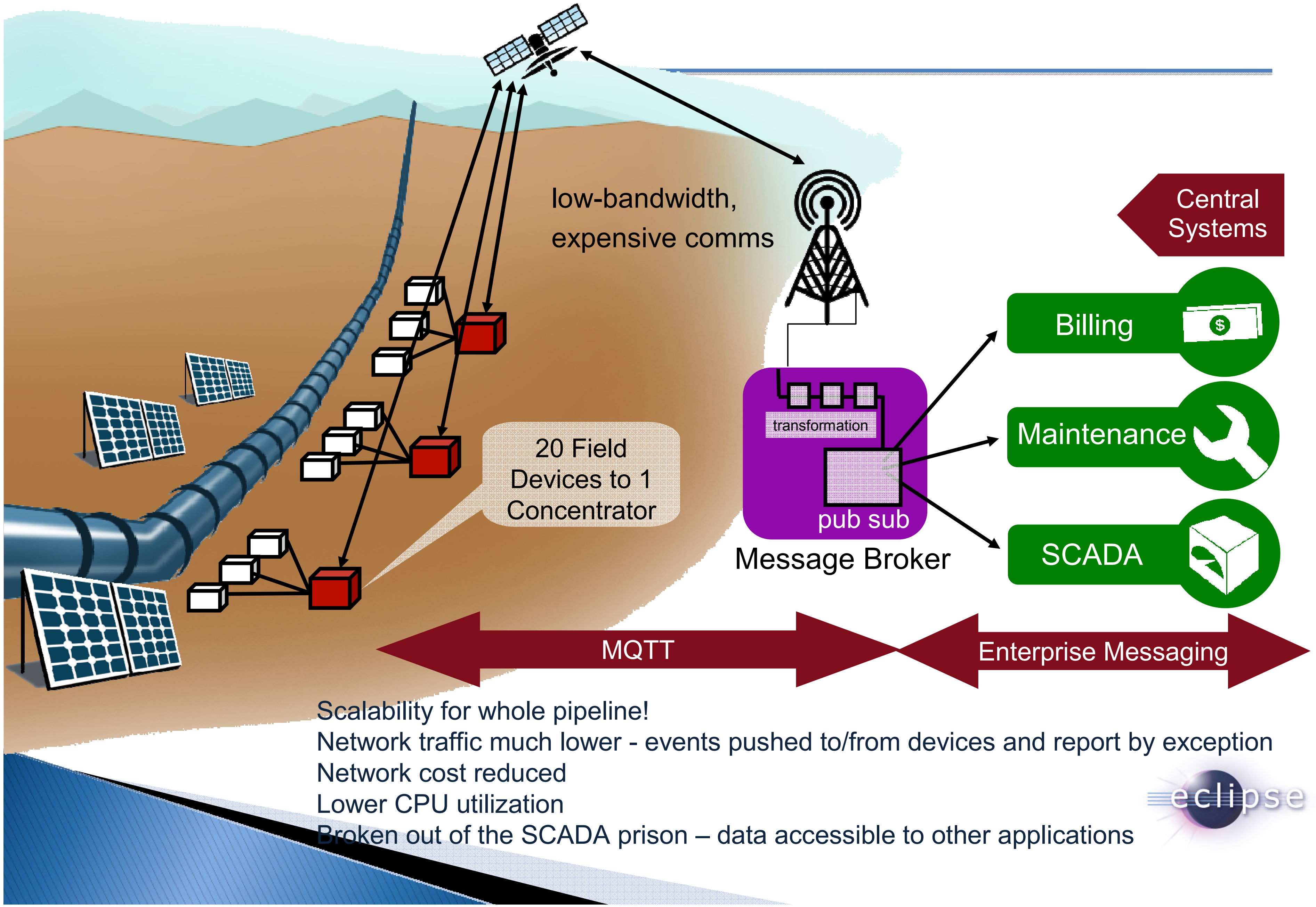
Monitor large numbers of patient’s with pace makers in their home

- An MQTT client is embedded in the home monitoring appliance
- Collects diagnostics when patient is in close proximity and periodically dials health care provider
- Immediately dials if abnormality detected
- Runs over dial up and mobile networks which may be “old”, very slow and fragile
- MQ/MB receives diagnostics and hands off to analytics application
- Analytics applications look for abnormalities and notifies clinician

Benefits

- Saves time, effort & money for the patient: there is no longer a need to visit the clinic on a fixed interval
- Peace of mind: problems are detected early preventing potentially life threatening incidents
- Clinicians time is better utilised: only patients with potential problems are seen

Enterprise to physical world solution with MQTT



Scalability for whole pipeline!
Network traffic much lower - events pushed to/from devices and report by exception
Network cost reduced
Lower CPU utilization
Broken out of the SCADA prison – data accessible to other applications

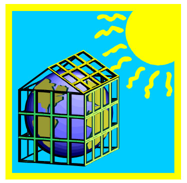
Some areas where MQTT has been used:

POS

Kiosks

Slot Machines

Automotive/
Telematics



Environment &
Traffic Monitoring

Fire & Gas
Testing

Chemical
Detection

Asset tracking /
management

SCADA

Medical

Field Force
Automation

Home
Automation

Railway

Further reading

- All things MQTT
  - http://mqtt.org
- MQTT Specification
  - http://www.ibm.com/developerworks/webservices/library/ws-mqtt/index.html
- Eclipse Paho
  - http://www.eclipse.org/paho/
- Eclipse M2M
  - http://wiki.eclipse.org/Machine-to-Machine
- MQTT: the Smarter Planet Protocol
  - http://andypiper.co.uk/2010/08/05/mqtt-the-smarter-planet-protocol/