MQTT

A protocol for communicating with your things

David Brinnen and Brett Cameron

SESAM Seminar, March 2018

Abstract

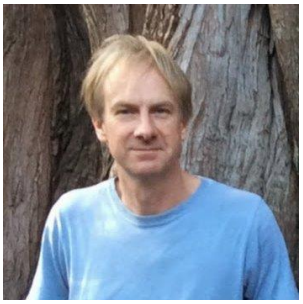
The Internet of Things refers to the ever-growing network of physical devices that have IP connectivity, allowing them to connect to the internet, and the communication that occurs between these devices and other internet-enabled devices and systems. In this talk, Brett and David will introduce the Internet of Things and will discuss some of the key technologies associated with the creation of Internet of Things solutions and services within the manufacturing domain. Particular attention will be given to MQTT, which is gaining acceptance as the preferred protocol for use by the Internet of Things applications. Currently available implementations of MQTT will be briefly reviewed and case studies illustrating the application of the protocol will be presented. Some examples of how MQTT might be used to implement secure, fault-tolerant, and scalable Internet of Things solutions will be discussed, and the application and use of MQTT in Next Generation SCADA systems that need to monitor and control devices at scale in a connected world will be will be discussed and demonstrated.

About David

Since completing his Masters studies in Embedded Software at the Swedish Royal Institute of Technology 2015 (KTH), David has been working as a software engineer across a range of projects, including the implementation of control logical and the commissioning of Energy Machines integrated energy systems and air handling units (https://www.energymachines.com/), and the development of a next-generation SADA platform (ControlMachines™) and simulation software to control, monitor, and model Energy Machines deployments. David has also recently joined VMS Software Inc., focusing primarily in the compiler space (Clang and LLVM) as part of the OpenVMS x86 port, but also working in various other areas ranging from consulting through to the design and development of remote monitoring and SCADA-like solutions. In his spare time David likes to investigate new technologies and develop his skills in software languages and using computers as tools. Besides computing David likes to travel and live the Spanish life with a Swedish touch, or vice versa. David was born in Sweden and currently lives in the Swedish city of Malmö.

About me

Brett Cameron works as Director of Applications and Open Source Services at VMS Software (http://www.vmssoftware.com), helping to define and implement the company's Open Source strategy for the OpenVMS operating system. Before joining VMS Software Brett worked as a senior architect with HP's Cloud and Enterprise Services groups. Brett comes from Christchurch, New Zealand and has worked in the software industry since starting work with Digital Equipment Corporation in 1992, and in that time he has gained experience in a wide range of software technologies, focusing primarily in areas such as distributed systems, transaction processing, and integration. Over the past decade Brett has spent considerable time travelling the world helping organisations to modernize their legacy application environments and to better leverage Open Source technologies. In more recent times, his involvement with various Open Source projects and his work in the cloud computing space has caused him to develop a liking for functional programming languages, and Erlang in particular, which he has ported to several exotic platforms. Brett holds a doctorate in chemical physics from the University of Canterbury, and maintains close links with the University, delivering guest lectures and acting as an advisor to the Computer Science department on course structure and content. In his spare time Brett enjoys listening to music and playing the guitar.



AGENDA

- **The Internet of Things and next generation SCADA**
- MQTT
- Control Machines (overview and demonstration)
- Summary
- Questions

Wikipedia tells us…

"The Internet of Things (IoT) is the network of physical objects or "things" embedded with electronics, software, sensors, and network connectivity, which enables these objects to collect and exchange data. The Internet of Things allows objects to be sensed and controlled remotely across existing network infrastructure, creating opportunities for more direct integration between the physical world and computer-based systems, and resulting in improved efficiency, accuracy and economic benefit…"

https://en.wikipedia.org/wiki/Internet_of_Things

Defining *Things*

- Technologies and solutions that enable integration of real-world data and services into current information networking technologies
  - *Technology that connects devices over wired or wireless networks*

- A system falls under the Internet of Things definition if it meets the following criteria:
  - It must connect
    - To the physical world around itself collecting information
    - To other things in order to interact with them effectively
    - To the internet or a network
    - …
  - It must compute
    - By processing the inputs it receives in some way and making them meaningful to other systems
  - It must communicate
    - With the network, with other things, with users if necessary, …

The Internet of Things is a concept in which anything from inanimate objects to people are assigned an IP address and provided with the ability to transfer data over a network.

More things are being connected…

- Home/daily-life devices (home automation)
- Business and public infrastructure
- Healthcare
- …

- *Things connecting to things*

- *People connecting to things*

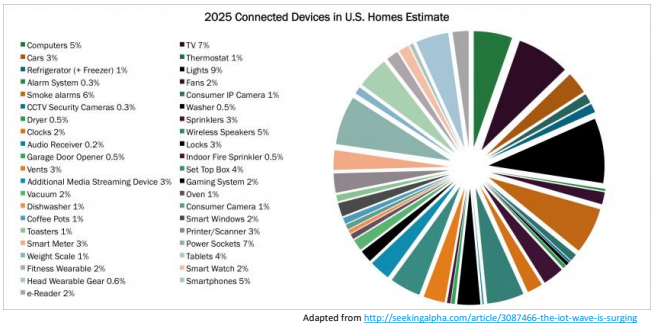
Adapted from http://personal.ee.surrey.ac.uk/Personal/P.Barnaghi/teaching/EEEM048/EEEM048_Lecture1_Introduction_Part_1.pdf

Sensor devices becoming widely available…

- Programmable devices
- Off-the-shelf gadgets and tools
- …
- Trillions of these things

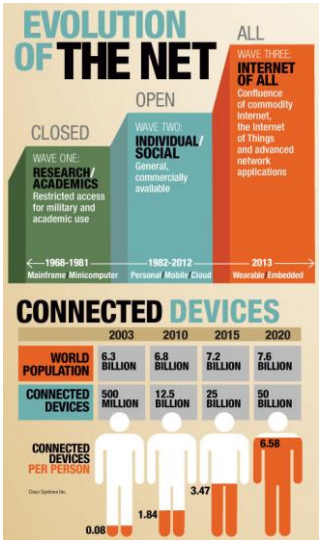
It's a happening thing

- While the Internet of Things may not be well known to the average consumer, it is very much upon us
  - Internet-connected devices such as speakers, fans, heat pumps, televisions, thermostats, lights, and refrigerators are currently available
  - Many more items will become connected within the next few years
  - In January 2015, Samsung stated that 90% of the goods it produces will be part of the Internet of Things by 2017, and 100% by 2020
  - IDC forecasts that the Internet of Things market will grow roughly 21% per year to $7.1 trillion by 2020
  - In US households alone, nearly four billion devices will be connected by 2020, and nearly seven billion by 2025



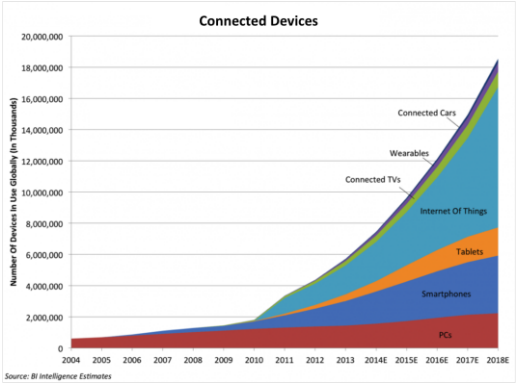
Adapted from http://seekingalpha.com/article/3087466-the-iot-wave-is-surging

Numbers of things connected to the internet



http://www.govtech.com/dc/articles/FutureStructure-The-New-Framework-for-Communities.html

- During 2008 (possibly earlier, depending on who you ask), the number of things connected to the internet exceeded the number of people on earth
- By 2020 it is predicted that there will be 50 billion internet-connected devices of some form or another



Market potential and opportunities

Cisco CEO Pegs Internet of Things as $19 Trillion Market:
http://www.bloomberg.com/news/2014-01-08/cisco-ceo-pegs-internet-of-things-as-19-trillion-market.html

- Emerging markets include (but are not at all limited to):
  - Consumers/retail
  - Home automation
  - Environmental monitoring
  - Energy management
  - Commercial
  - Healthcare and monitoring
  - Connected cities
  - Wearable's (see http://www.omsignal.com/ for example)
  - Industry
  - …

The applicability and usefulness of IoT is generally obvious, and it is not difficult to think of scenarios where IoT-related tools and technologies can be applied.

The Internet of Things and Industry 4.0

The effects of the Internet of Things on industry have been cited as the next industrial revolution. Each revolutionary period can be viewed as a natural evolution of the previous period.

	When	What
Industry 1.0	1784	Mechanical production powered by steam and water (industry)
Industry 2.0	1870	Assembly line and electrically powered production (mass production)
Industry 3.0	1969	Computers, automation, and distributed control (mass production plus computers)
Industry 4.0	2017	Cyber-physical systems and the internet (mass production plus computers plus the internet)

Industry 4.0

- The term "industry 4.0" originates from a German government initiative focused integrating the manufacturing industry with internet technologies
  - https://ec.europa.eu/digital-single-market/en/industrial-platforms-and-large-scale-pilots
  - https://ec.europa.eu/digital-single-market/en/public-private-partnerships
- The initiative aims to accelerate the integration of technologies such as big data, cloud, high-performance computing and the Internet of Things with advancements in areas like artificial intelligence, autonomous systems, 3D printing, ...

"The Digitising Industry strategy reinforces the role of Public Private Partnerships to focus on key technologies and their implementation through federated projects. More than €20 billion are planned to be invested in the 5 coming years in the context of the Digital Single Market."

- The effects will be far ranging on both businesses and consumers
  - Industry 4.0 provides companies with a blueprint to modernize their operations and drive huge efficiency gains through their business and into their value chains

Industry 4.0

- The concept of this “digital transformation” of manufacturing has spread to include many industries
  - Logistics
  - Transportation
  - Oil and gas
  - Building management and maintenance
  - Utilities
  - Traffic management
  - Healthcare
  - ...
- And the concept is catching on in a big way across Europe...

https://solace.com/blog/use-cases/industry-4-0-driving-event-driven-architecture-forefront-modernization-europe

Evolution of SCADA

Supervisory Control And Data Acquisition

> *"A control system architecture that uses computers, networking, and graphical user interfaces for process monitoring and control, operating in conjunction with peripheral devices such as programmable logic controllers, sensors, and actuators to interface to plant and machinery"*

- An automation and control system using computers
  - Process control and real-time data gathering from remote locations for exercising this control on equipment and conditions ☺
  - Feedback loop
- Primary functionality being...
  - Monitoring
  - Configuration and Control
  - Alarms and events
  - Data acquisition, analysis, and presentation
  - Historical data
  - GUI/HMI

Evolution of SCADA

- Typically used to automate complex processes where human control is impractical
  - Manufacturing and process control
  - Building/facilities management
  - Electric power generation, transmission, and distribution
  - Water and sewage
  - Refining
  - Oil and gas reticulation
  - Traffic flow
  - Airports and shipping
  - …
- Facilities and operations are often distributed and widely geographically dispersed across multiple sites
  - Requires reliable communication
  - Redundant links

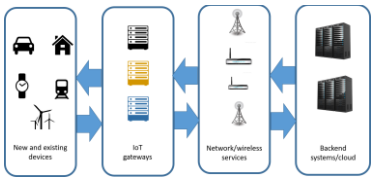
SCADA systems can be used not only for industrial process control but also for infrastructure and facilities-based processes, and systems can range in size from single-digit to many thousands of process loops.

Evolution of SCADA

- SCADA systems have evolved from being monolithic to networked and distributed

With the commercial availability of cloud computing, SCADA systems have increasingly adopted Internet of things technology. As a result, SCADA systems can now report and manage state in near real-time and use the horizontal scalability available in cloud environments to implement more complex and powerful systems than were previously possible. Such solutions leverage a new generation of software technologies including massively scalable NoSQL databases and Open Standards-based communication and messaging protocols.

- The Internet of Things is a logical and natural evolution of SCADA
  - SCADA has evolved beyond individual process control to bring unified control to entire systems of (distributed) processes
  - Concepts like real-time data visualization and machine-to-machine communication were born in SCADA
- *… we'll come back to this later!*



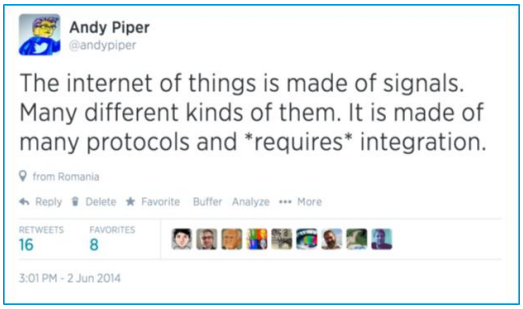
Adapted from http://wiki.eclipse.org/images/7/7c/MQTTIntroEclipseWebinary1.pdf

AGENDA

- The Internet of Things and next generation SCADA
- **MQTT**
- Control Machines (overview and demonstration)
- Summary
- Questions

MQTT (Message Queuing Telemetry Transport)

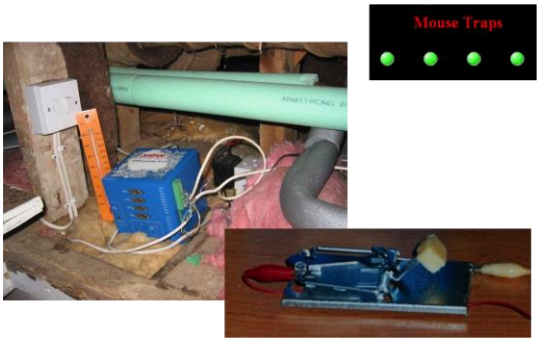
- MQTT as an Internet of Things protocol
  - Overview
- MQTT features
  - Topics and publish/subscribe patterns
  - Quality of service levels
  - Retained messages
  - Persistent sessions
  - Last will and testament
  - Heartbeats/keep-alive messages
  - Security
- Broker and client API implementations
- MQTT-SN
- The Internet of Things protocol wars



A little bit of history

MQTT's origins go back nearly two decades. Its co-inventor, Andy Stanford-Clark (IBM Distinguished Engineer) has long been a passionate home-automation tinkerer. His laboratory has been his house, a 16th-century cottage on the Isle of Wight, and his electronic gadgets range from temperature and energy monitors to an automated mousetrap. His home automation projects required machine-to-machine data communication, and for this purpose Andy wrote his own code.

At IBM Andy became immersed in the technology for machine-to-machine communication in the late 1990s, when IBM were working with industry partners to mine sensor data from offshore oil rigs for preventive and predictive maintenance. One of those industry partners was Arlen Nipper, an American engineer and expert in embedded systems for oil field equipment. Together they wrote the initial version of MQTT in 1998, based on Andy's original home automation code.



MQTT

- An Open Standard; ISO/IEC 20922:2016
  - Invented by Andy Stanford-Clark of IBM, and Arlen Nipper of Cirrus Link Solutions
  - Donated to Eclipse "Paho" M2M project
    - http://www.eclipse.org/paho/
- A lightweight publish/subscribe messaging protocol
  - TCP-based (typically)
  - Asynchronous bi-directional push (no polling)
  - Simple, small number of verbs
  - Protocol compressed into bit-wise headers and variable length fields
    - Smallest packet size 2 bytes
  - Small client footprint
  - Reliability levels, session awareness
  - Data-centric, payload-agnostic
    - Separates data (payload) from metadata (topic)
- Facilitates the transfer of telemetry-style data to a server or message broker from pervasive devices over high-latency or otherwise constrained networks
  - Sensors and actuators
  - Mobile phones
  - Embedded systems on vehicles
  - Laptops and other computing devises

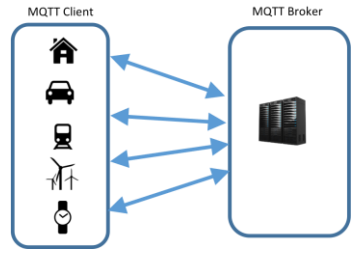
MQTT is a machine-to-machine (M2M)/"Internet of Things" connectivity protocol. It was designed as an extremely lightweight publish/subscribe messaging transport. It is useful for connections with remote locations where a small code footprint is required and/or network bandwidth is at a premium. For example, it has been used in sensors communicating to a broker via satellite link, over occasional dial-up connections with healthcare providers, and in a range of home automation and small device scenarios. It is also ideal for mobile applications because of its small size, low power usage, minimized data packets, and efficient distribution of information to one or many receivers...

http://www.mqtt.org

Original MQTT 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
- 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 (10,000+ clients)
- Simple for application developers
- Publish the protocol for ease of adoption
- To be industry agnostic

- Simple pub/sub protocol
- Small footprint
- Minimal on-the-wire format and payload

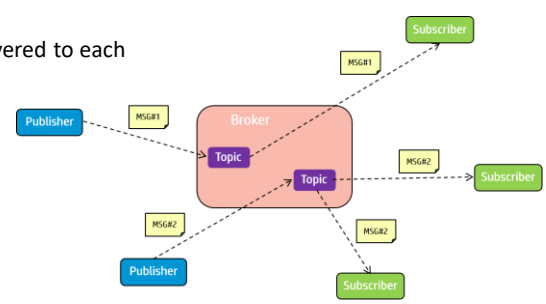


Adapted from http://wiki.eclipse.org/images/7/7c/MQTTIntroEclipseWebinarv1.pdf

Topics and publish/subscribe patterns

- MQTT is a broker-based publish/subscribe messaging protocol
  - A single published message can be received via the broker by multiple consumers
  - Decoupling between producers and consumers
- A producer sends (publishes) a message on a topic
- A consumer subscribes for messages on a topic (or multiple topics)
- The broker matches publications to subscriptions
  - If no match is found for a message then it is discarded
  - If one or more matches are found the message is delivered to each matching subscriber

The MQTT protocol is based on the principle of publishing messages and subscribing to topics (pub/sub). Multiple clients connect to a broker and subscribe to topics that they are interested in. Other clients connect to the broker and publish messages to topics. Many clients may subscribe to the same topics and clients can subscribe to multiple topics. Subscriptions may be to an explicit topic, in which case only messages to that topic will be received, or they may include wildcards.



Topics and publish/subscribe patterns

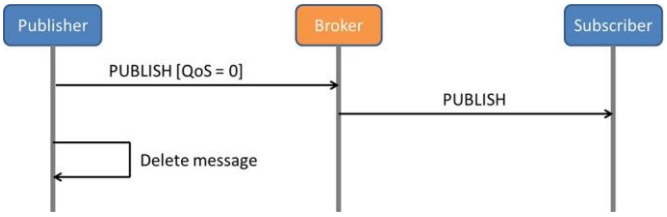
- An MQTT topic forms a sort of namespace
  - Topics are treated as a hierarchy, using a slash (/) as a separator
  - This allows sensible arrangement of common themes to be created, much in the same way as a file system
- There is no need to configure a topic
  - Publishing on it is sufficient
- A subscriber can subscribe to an absolute topic or 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
  - Producers cannot use wildcards (it would make no sense)

For the topic of "a/b/c/d", the following subscriptions would match:

```
a/b/c/d
+/b/c/d
a/+/c/d
a/+/+/d
+/+/+/+
#
a/#
a/b/#
a/b/c/#
+/b/c/#
```


Quality of service

QoS 0: At most once delivery (fire and forget)

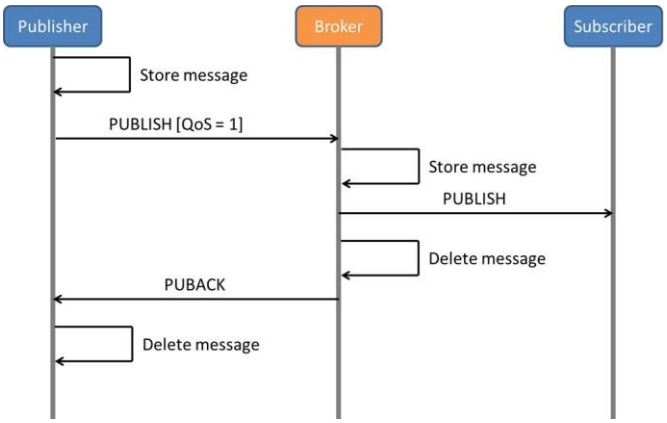


http://www.embedded101.com/Portals/0/images/DNNArticle/Windows-Live-Writer/3.7_E248/Fig3.81.jpg

- In this case MQTT does not add anything over and above what is provided by TCP
- Essentially equivalent to TCP best-efforts; no guarantees
- Messages may not be received and parties will generally be unaware of any drops

Quality of service

QoS 1: At least once delivery

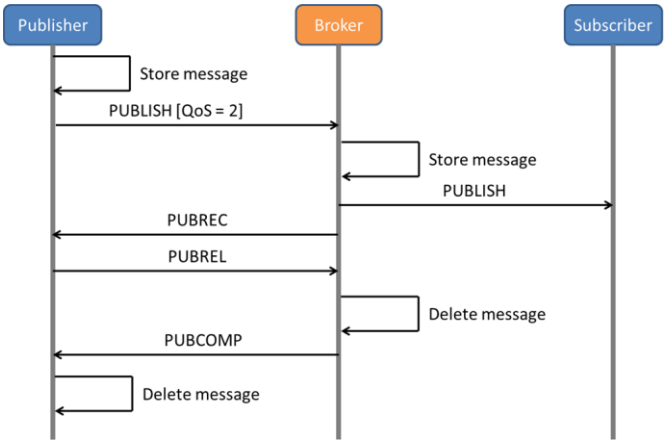


http://www.embedded101.com/Portals/0/images/DNNArticle/Windows-Live-Writer/3.7_E248/Fig3.9_2.jpg

- This level guarantees that messages are delivered to subscribers (or fail on publish)
- However duplicate messages may be received

Quality of service

QoS 2: Exactly once delivery



http://www.embedded101.com/Portals/0/images/DNNArticle/Windows-Live-Writer/3.7_E248/Fig3.10_2.png

- Subscribers will receive messages exactly once (no duplicates)
- Maximum bandwidth and computational overhead
- Broker needs to store messages locally

Subscription durability

Subscriptions 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 broker until the subscriber next connects
- Non-durable:
  - The subscription lifetime is the same as the time the subscriber is connected to the broker

On connection, a client can set the "clean session" flag to either true or false. If clean session is set to false, then the connection is treated as durable. This means that when the client disconnects, any subscriptions it has will remain and any subsequent Quality of Service 1 or 2 messages will be stored by the broker until it connects again in the future. If clean session is true, then all subscriptions will be removed for the client when it disconnects.

Retained messages

Published messages can be retained by the broker

- A publisher can mark a message as *retained*
- The broker remembers (retains) the last known good message for a topic
- The broker gives the last known good message to new subscribers
  - New subscribers do not have to wait for a publisher to publish a message in order to receive their first message
  - Useful for events that happen infrequently

Last will and testament

Clients can specify a *last will and testament*

- Broker publishes the last will and testament message on behalf of the client upon detecting the "death" of the client connection
- Useful for reporting problems
- Real push on device “death”
- Enables applications to know when a client goes offline abnormally
- Typically used for monitoring the connection status of a device

Security

- Security is a common (and often major) area of concern with respect to Internet of Things protocols
- Protocol layer
  - Username/password
  - Those feeling really paranoid could also encrypt message payloads
- Transport layer (TCP)
  - MQTT supports use of SSL/TLS
  - Client certificate authentication
- Broker
  - Publish/subscribe permissions (ACL's)
  - Integration to other systems
    - Databases
    - APIs
    - ...

Note that MQTT brokers such as Mosquito support authentication and authorization plugins, allowing developers to customize access to resources to meet specific requirements.

Notable broker implementations

- Mosquitto
  - http://mosquitto.org
  - C, small, fast standalone binary, MQTT only, fully standards compliant (more on Mosquitto later)
- RabbitMQ MQTT plugin
  - http://rabbitmq.com
  - Erlang, enterprise-grade MQTT plugin for RabbitMQ; currently not 100% compliant with the MQTT standard
- VerneMQ
  - https://verne.mq/
  - A massively scalable Open Source Erlang-based MQTT broker
- HiveMQ
  - http://hivemq.com
  - Standalone Java MQTT broker, not Open Source, free for personal use
- Mosca
  - https://github.com/mcollina/mosca
  - A Node.js-based MQTT 3.1 compliant broker, which can be used standalone or embedded in another Node.js application
- eMQTT (EMQ)
  - https://github.com/emqtt
  - Another highly scalable Erlang-based broker
- ActiveMQ
  - http://activemq.apache.org/
  - Supports MQTT and several other protocols
- And a few others
  - See http://mqtt.org/wiki/doku.php/brokers)

MQTT client implementations

Paho:

- http://www.eclipse.org/paho/
- Open Source
  - Active community
- Reference implementation
- API's available for many languages
  - C/C++, Go, Java, JavaScript/Node.js (MQTT over WebSockets), Lua, Python, ...

"The Paho project provides scalable open-source client implementations of open and standard messaging protocols aimed at new, existing, and emerging applications for Machine-to-Machine (M2M) and Internet of Things (IoT)"

pāho (*verb*) to broadcast, make widely known, announce, disseminate, transmit (from the Maori dictionary)

Numerous others:

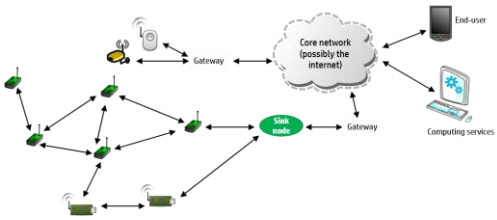
- See http://github.com/mqtt/mqtt.github.io/wiki/libraries

MQTT as a service

- Xively
  - https://xively.com/dev/docs/api/communicating/mqtts/
- Litmus Automation
  - http://litmusautomation.com/
- Octoblu
  - https://octoblu.github.io/
- Eurotech Everyware Device Cloud
  - http://www.eurotech.com/en/products/software+services/everyware+device+cloud
- Carriots
  - https://blog.carriots.com/mqtt-support-and-new-electric-imp-tutorial/
- AWS IoT
  - https://aws.amazon.com/iot/CloudMQTT
- CloudMQTT
  - https://www.cloudmqtt.com/
- ... and various others

MQTT-SN

- Even though MQTT is designed to be lightweight, it has two drawbacks for very constrained devices
  1. Every MQTT client must support TCP and will typically hold a connection open to the broker at all times; for some environments where packet loss is high or computing resources are scarce, this is a problem
  2. MQTT topic names are often long strings, which make them impractical for low-rate wireless personal area networks (IEEE 802.15.4)
- These matters are addressed by the MQTT-SN protocol (MQTT for Sensor Networks), which defines a UDP mapping of MQTT and adds broker support for indexing topic names
  - Clients (devices) communicate via a gateway to a MQTT broker on the TCP/IP network
  - Gateway translates messages between MQTT-SN and MQTT



- Very low power consumption
- Very low bandwidth requirements
- Very small client footprint
- QoS reliable messaging in a potentially unreliable environment
- Used in things ranging from US and UK military's sensor fabrics to home automation
- See http://mqtt.org/2013/12/mqtt-for-sensor-networks-mqtt-sn

MQTT – real-world usage

Facebook Messenger

- Facebook chat application; more than 850 million users (probably closer to 1 billion)
- MQTT provided various advantages
  - Better battery life for mobile devices
  - Lower latencies
  - Reduced bandwidth requirements

"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."

Lucy Zhang, software engineer at Facebook, August 2011https://www.facebook.com/notes/facebook-engineering/building-facebook-messenger/10150259350998920

MQTT – real-world usage

Smart homes/home automation

- MQTT as the central message bus
- Easy to integrate with other technologies
- Remote monitoring and control
- Many Open Source Smart Home frameworks support MQTT

- Numerous home automation frameworks available…
  - Google Nest
  - Apple HomeKiT
  - OpenHAB
  - Eclipse SmartHome
  - Smarthings Hub
  - AllJoyn
  - …

MQTT – real-world usage

Pipeline monitoring

- ConocoPhillips pipeline
- 30,000+ devices
- 17,000+ km of pipeline
- Remote monitoring and control via MQTT on satellite links (bandwidth very expensive)
- Enables real-time conditional maintenance

About 25% of manufacturing companies are already using Internet of Things devices in one way or another to increase production and reduce costs through machine-to-machine (M2M) communication, and this figure is anticipated to grow to over 80% by 2025.

MQTT – real-world usage

Deutsche Bahn – train status information

- Deutsche Bahn wanted to provide displays that informed passengers at smaller stations about train status
- They additionally wanted the devices to be able to report back to the control center about display device and message status
- An MQTT-based solution facilitated reliable messaging and a bi-directional messaging capability
- MQTT-based solution reduced messaging costs by 60 percent over the old SMS-based system
- Bi-directional messaging facilitates remote monitoring and rebooting of displays, significantly reducing maintenance costs

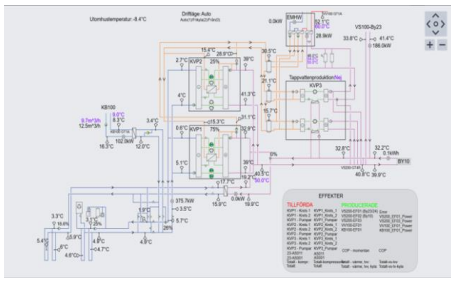
Deutsche Bahn cuts messaging costs by 60 percent

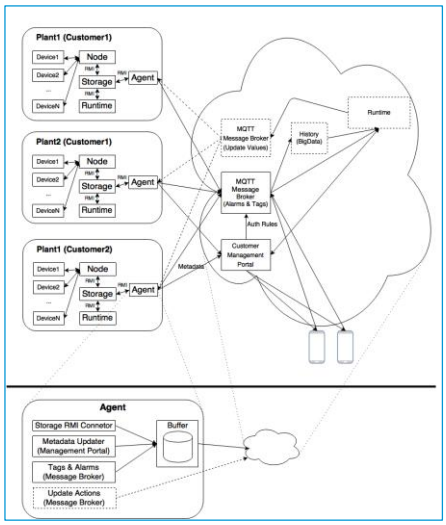
IBM WebSphere MQ Telemetry updates passengers on train status more cost-efficiently than SMS

MQTT – real-world usage

Control Machines

- Monitor and control Energy Machines deployments (see https://www.energymachines.com/)
- Next generation SCADA system
- More about this later ☺





Other places where MQTT is used

- Point of sale kiosks
- Slot machines
- Automotive/telematics
- RFID
- Environment and traffic monitoring
- Chemical detection
- Asset tracking and management
- SCADA (Supervisory Control and Data Acquisition)
- Medical/healthcare
  - St Jude Medical - 100,000 heart pacemakers monitored via MQTT
- Railways

"*Volvo launches telematics as standard across U.S. lineup*", 1 September 2014, http://analysis.telematicsupdate.com/fleet-and-asset-management/weekly-brief-volvo-launches-telematics-standard-across-us-lineup

Alternatives to MQTT

CoAP:

- Constrained Application Protocol
- Appears to be getting pushed by Cisco
  - http://blogs.cisco.com/ioe/beyond-mqtt-a-cisco-view-on-iot-protocols/
- HTTP REST-like (request/response), but based on UDP instead of TCP
- HTTP-like verbs and status codes
- Transparent mapping to HTTP
- Constructs called "options" are analogous to HTTP headers
  - Although binary a opposed to textual
- Provides mechanisms to avoid HTTP-like long polling
- Quality of service with confirmable messages
- Resource discovery

AMQP:

- Advanced Message Queuing Protocol
  - See my 2014 Boot Camp talk "*The Polyglot Rabbit; Adventures on OpenVMS with RabbitMQ, Erlang, and multi-protocol messaging*"
- http://www.amqp.org
- Considerable overlap with MQTT
- Can accommodate most MQTT use cases, but incurs more overhead
- More enterprise application oriented

HTTP:

- See subsequent slides…

Ultimately, the choice of protocol depends on the particular scenario under consideration, and indeed complex systems may require the use of multiple protocols.

HTTP versus MQTT

People seem obsessed with trying to use HTTP for everything. HTTP was basically built for request-response situations ("I want a document loaded into my web browser now, please"). Over the years HTTP has been bent and massaged into doing things that it was never intended or designed for. New features have been added, but ultimately the fundamental limitations imposed by the original design remain. HTTP is good at doing what it was designed to do (and indeed it has changed our world); however it is not so good at doing things that it was not intended to do! From an IoT perspective, HTTP is verbose and inefficient, and does not scale to the proportions demanded by the IoT; it is not fit for purpose.

Protocol	Delivery of messages	Efficient use of network resources	Reliable delivery over fragile networks	Decoupling of producer and consumer operations
HTTP	• Push from client to server • Inherently synchronous request-response • Clients must poll the server to check for new data	• Heavy • Hundreds of bytes of overhead per message	• No QoS • Polling and retry mechanisms often come into play	• HTTP is inherently synchronous request-response • One-to-one • Any decoupling must be implemented separately behind the web server
MQTT	• Low-latency push from client to server and server to client • No polling	• Efficient use of network resources • Very minimal overhead per message	• Messages are delivered according to the specified QoS even across network breaks (handled by the protocol)	• Fully decoupled publish/subscribe • One-to-many delivery

HTTP versus MQTT

		3G		WiFi	
		HTTPS	MQTT	HTTPS	MQTT
Receive	Messages per hour	1708	160278	3628	263314
	% battery per message	0.01709	0.00010	0.00095	0.00002
	Messages received	240 / 1024	1024 / 1024	524 / 1024	1024 / 1024
Send	Messages per hour	1926	21685	5229	23184
	% battery per message	0.00975	0.00082	0.00104	0.00016

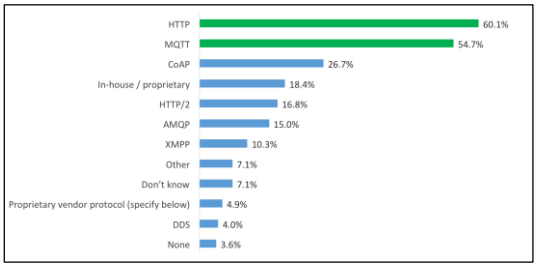
See "*Power Profiling: HTTPS Long Polling vs. MQTT with SSL, on Android*", http://stephendnicholas.com/archives/1217. Similar results have been obtained by others, clearly illustrating the efficiency of MQTT over HTTP for IoT use-cases.

Comparing MQTT and CoAP

- MQTT is a many-to-many communication protocol for passing messages between multiple clients through a central broker
  - Publish/subscribe
  - Decouples producer and consumer by letting clients publish and having the broker decide where to route and copy messages
  - A communications bus for live data
- CoAP is primarily a point-to-point protocol for transferring state information between client and server
  - RESTful model
  - Not purely event based
- MQTT clients make a long-lived outgoing TCP connection to a broker while CoAP clients and servers both send and receive UDP packets
  - Different ramifications from a NAT perspective
- MQTT provides no support for labeling messages with types or other metadata to help clients understand the content they're receiving
  - MQTT clients must know up-front about the message formats they will be receiving
  - CoAP provides support for content negotiation and discovery

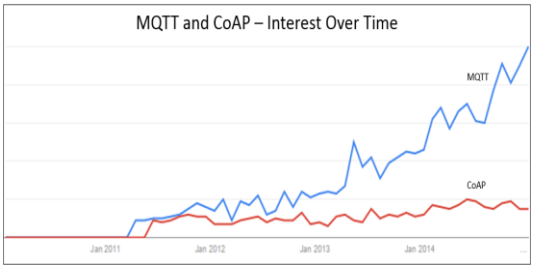
Most good messaging protocols share many features in common; however some are better suited to a particular set of use cases than others. MQTT and CoAP are both useful as Internet of Things protocols, but have fundamental differences that need to be taken into careful consideration when designing solutions. It's not a one-protocol-fits-all scenario.

MQTT adoption



IoT Developer Survey 2017 - Copyright Eclipse Foundation, Inc. (https://www.slideshare.net/JanSkerrett/iot-developer-survey-2017)

- HTTP remains the most prevalent messaging protocol but MQTT is catching up
- It will be interesting to see what happens with HTTP/2



AGENDA

- The Internet of Things and next generation SCADA
- MQTT
- **Control Machines (overview and demonstration)**
- Summary
- Questions

ControlMachines™

- A next generation SCADA system
- Monitor and control Energy Machines and Climate Machines deployments (see https://www.energymachines.com/)
  - Visualization
  - Real-time monitoring and altering
    - Analytics
    - Trends
    - Reporting
    - …
  - Control/fine-tuning

ControlMachines is complemented by a powerful simulation package that can be used to help guide, model, and tune physical design, perform what-if scenarios, and so on. More about this later (if time permits).

- Provides a single view across multiple machine deployments (highly scalable)
- Generic
  - Can be used with any system having a Modbus TCP/IP interface
  - Planning to support additional protocols
- Leverages MQTT for data collection

And now over to David…

AGENDA

- The Internet of Things and next generation SCADA
- MQTT
- Control Machines (overview and demonstration)
- **Summary**
- Questions

Summary

- The Internet of Things and Industry 4.0 are not just cute names; they are big business and are something that will change our world

- MQTT is an ideal protocol for the Internet of Things
  - Lightweight
  - Efficient
  - Fault tolerant (unreliable networks)
  - Scalable
  - Secure
  - Standard
  - Numerous interoperable client and broker implementations

Summary

- ControlMachines™ illustrates the use of MQTT in a next generation SCADA environment
- But protocols such as MQTT are just part of the picture
- There is a convergence of technologies that now make it possible to implement massively scalable Internet of Things solutions
  - Cloud computing
  - Big Data and analytics
  - Advances in AI
  - Advances in distributed computing
  - ...
- We live in interesting times... again

http://en.wikipedia.org/wiki/Internet_of_Things

AGENDA

- The Internet of Things and next generation SCADA
- MQTT
- Control Machines (overview and demonstration)
- Summary
- **Questions**

Questions