vms Software

# MQTT: A protocol for communicating with your things

Brett Cameron, March 2020

## Agenda

- Introduction and background
- MQTT overview
- Some real-world examples
- Summary
- Questions

# About me

**1992 - 2014**

- Digital Equipment Corporartion (DEC), Compaq, HP
  - Open source
  - Distributed systems
  - Integration
  - Modernization
  - Cloud

**2014 - present**

- VMS Software, Inc.
  - Chief Solutions Officer (open source strategy, professional services, …)

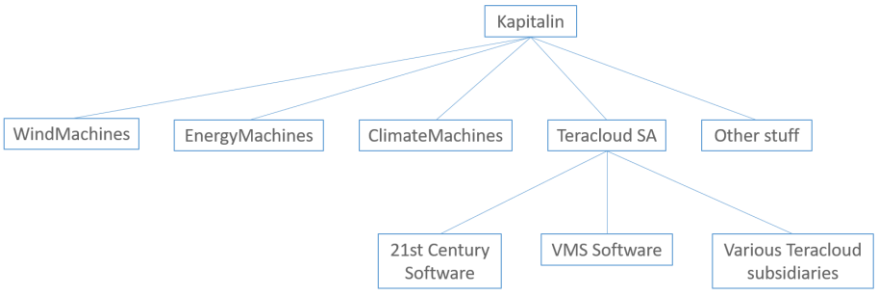


# VMS Software Inc. (VSI)

- Formed July 2014
  - Wholly owned by Teracloud SA (part of a larger group of companies… see next slide)
  - Currently headquartered in Bolton, MA
  - Second office in Malmö, Sweden
  - Some 130 staff across 8-countries (US, Sweden, Germany, France, Netherlands, Australia, Russia, ~~New Zealand~~)
- License agreement with HPE to develop, sell and support the OpenVMS operating system and layered products
- Sole developer of future versions of OpenVMS
- Perpetual Worldwide License
- Right to use the OpenVMS trademark
- Right to release OpenVMS on any hardware (including the mainstream Intel x86-64 platform)
- Agreement to transition HPE customers to VSI (driven by demand)
- http://www.vmssoftware.com

vms Software

# Corporate structure (simplified)




- Green energy solutions
  - Generation
  - Heating and cooling, air quality
  - Associated software (next-generation SCADA, simulations, digital twin, …)
  - Embedded systems
- Software
  - Legacy IBM assets
  - OpenVMS
  - Others in the pipeline

www.vmssoftware.com

# OpenVMS?

- A server operating system that runs on VAX, Alpha, and Intel Itanium processors... and soon on x86-64
- Originally developed by Digital Equipment Corporation (DEC) in 1977
- Proprietary
- Not at all UNIX/Linux-like
- Considered highly secure and stable
- Substantial and loyal installed base
- Used for numerous purposes across all sectors
  - Retail and manufacturing
  - Transportation control and monitoring
  - SCADA systems
  - Banking and financial services (some large stock exchanges still run OpenVMS)
  - Healthcare
  - Telco
  - Government
  - Military
  - Chip manufacturers
  - …

# OpenVMS - looking to the future

- OpenVMS was withering away and dying
- We are now moving forward

- Modernizing the operating system and layered products
- Embracing virtualization and cloud
  - VMware
  - KVM
  - …

- From being marooned on Alpha and Itanium islands we're now moving to the x86-64 mainland
  - And maybe elsewhere




# OpenVMS - looking to the future

- OpenVMS is unlikely to be adopted by too many new users for enterprise computing
- So we basically have two options...
  - Support existing users until they all do something else; or
  - Try to do something new and different with the operating system (and maybe give it a different name)

- We're running with the second option
  - Although obviously we will continue to look after our existing users!!

- Maybe look to combine assets across our business units
- Thinking...
  - Stripped-down version of the operating system
  - Black-box appliance
  - Embedded systems
  - Industry 4.0, IoT, ...
  - …

Think GRiSP with OpenVMS instead of RTEMS (with appropriate tuning OpenVMS approaches being a real-time operating system).

- Leverage the EnergyMachines pseudo-edge/PLC device (or some variant thereof)...

# OpenVMS - looking to the future



- The aforementioned pseudo-edge/PLC device…

- 4x1.2GHz CPU, 1GB RAM, 8GB EMMC, 2xUSB, 2xEthernet
- 32pins, rated to 34V
  - All inputs
  - 8 Analog outputs (0-30V, 0-500mA)
  - 8 current loop sense
  - 8 contact close
  - 2, 3, 4 wire resistance reading
- Redundant CANFD bus
  - Noise resilient
  - Classic CAN backwards compatible
- Up to 64 stackable units (centipede on DIN rail)
- USB-C isolated 2 domain power
- …

# Agenda



- Introduction and background
- MQTT overview
- Some real-world examples
- Summary
- Questions

# MQTT overview

- MQ Telemetry Transport
- Lightweight broker-based pub/sub messaging protocol
  - Asynchronous push delivery
  - Simple (and small) set of verbs
- Designed for constrained devices
  - Low bandwidth
  - High latency
  - Unreliable networks
  - High-cost networks (satellites)


www.vmssoftware.com

# MQTT overview

- Built to minimize network bandwidth usage and device resource requirements
- Minimized on-the-wire format
  - Smallest possible packet size is 2 bytes
  - No application message headers
- Reduced complexity/footprint
  - Small client footprint (30KB C; 100KB Java)

# MQTT overview

- Offers various reliability levels of message delivery
  - At most once delivery
  - Assured delivery but may be duplicated
  - Once and once only delivery

- Functionality to handle loss of contact between client and server
  - “Last will and testament” to publish a message if the client goes offline
  - Stateful “roll-forward” semantics
  - Durable subscriptions

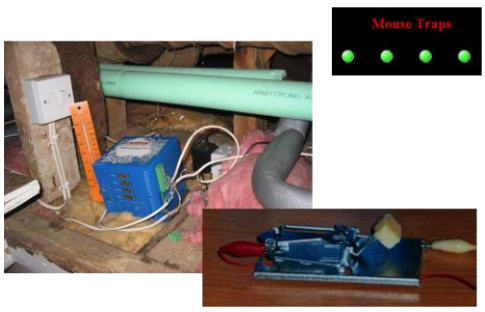
- MQTT v3.1.1 is OASIS standard (2014)
- Payload agnostic
- Initially developed by IBM and Eurotech for M2M communication for embedded devices



# 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.

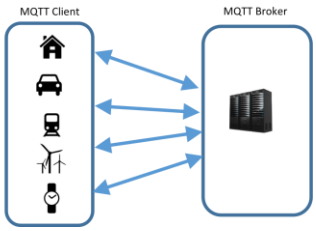
At IBM Andy became immersed in the technology for machine-to-machine communication in the late 1990s, when IBM were working with industry partners to mine sensor data from offshore oil rigs for preventive and predictive maintenance. One of those industry partners was Arlen Nipper, an American engineer 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.



# Design goals

- 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
- Loose coupling to support dynamic environments where high volumes of messages and events need to be made available to consumers in ways that may not have been originally anticipated
- Provide multiple deterministic message delivery qualities of service to reflect trade-offs between bandwidth, availability, and delivery guarantees
- Capable of supporting large numbers of devices (10,000+ clients)
- Simple for developers
- Publish the protocol for ease of adoption
- Industry agnostic

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



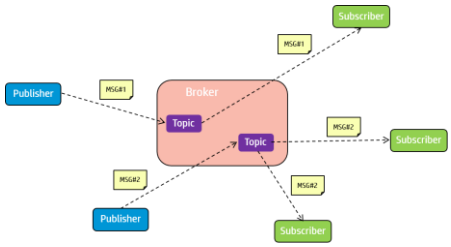

Adapted from http://wiki.eclipse.org/images/7/7c/MQTTIntroEclipseWebinarv1.pdf



# Topics and pub/sub patterns

- MQTT is a broker-based pub/sub messaging protocol
  - A single published message can be received via the broker by multiple consumers
  - Decoupling between producers and consumers
- A producer 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 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 pub/sub patterns

- An MQTT topic forms a sort of namespace
  - UTF8 string
  - 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)

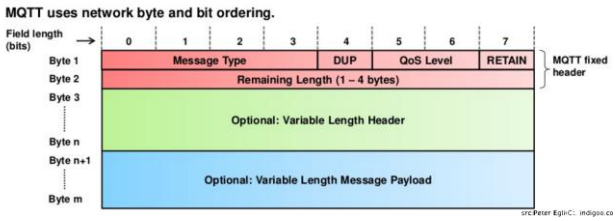
For the topic of "a/b/c/d", the following subscriptions would match:

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

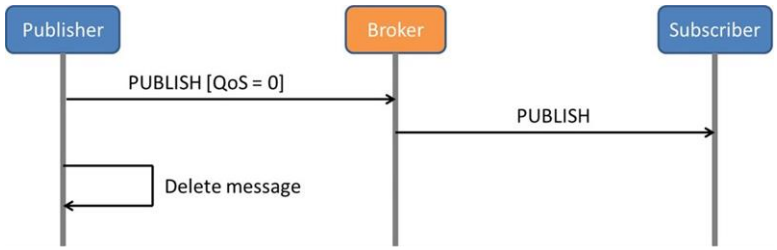


# Message format

- Very lean message format
  - Tries to save and reuse bytes as much as possible
- 2 byte fixed-length header
- Optional message-specific variable length header
- Optional message payload
  - Length encoded
  - Up to 256MB
- Fixed header indicates the API call, length of payload, and Quality of Service
- Contents of variable header depends on API call
  - Message ID
  - Topic name
  - Client identifier
  - …



# Quality of Service – QoS 0

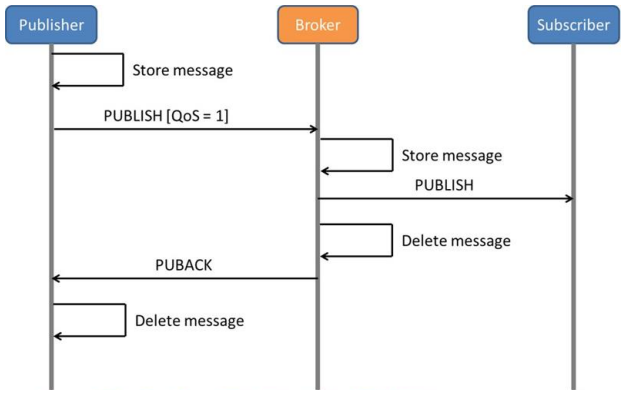



http://www.embedded101.com/Portals/0/images/DNNArticle/Windows-Live-Writer/3.7_E248/Fig3.81.jpg

- At most once delivery
- Minimal guarantee of message delivery
- Receiver doesn't acknowledge receipt of message
- No retry semantics
- Sender doesn't store message for redelivery (non-persistent)
- Same guarantee as underlying TCP protocol
- The message arrives either once or not at all
- Has the least overhead



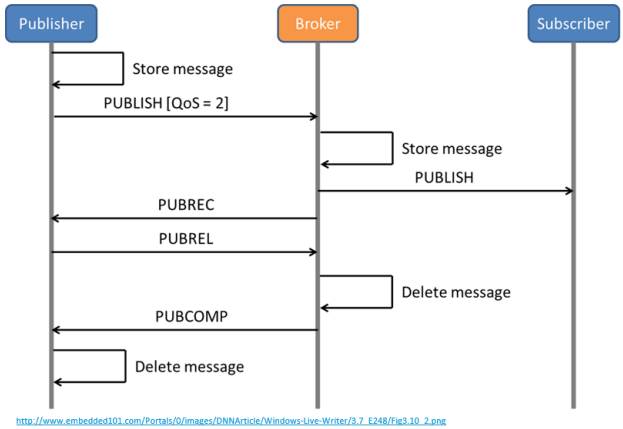
# Quality of Service – QoS 1




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

- Guarantees that a message will be delivered at least once
- Duplicate messages may be received

# Quality of Service – QoS 2




http://www.embedded101.com/Portals/0/images/DNNArticle/Windows-Live-Writer/3.7_E24B/Fig3.10_2.png

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



# Quality of Service

- QoS level is defined when creating a connection to the broker
- Would typically use QoS-0 when...
  - Have a stable connection
  - If occasional packet drop is acceptable
- Would typically use QoS-1 when...
  - Need to get every message and applications can handle duplicates
- Would typically use QoS-2 when...
  - It is critical to applications to receive all messages exactly once

# 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

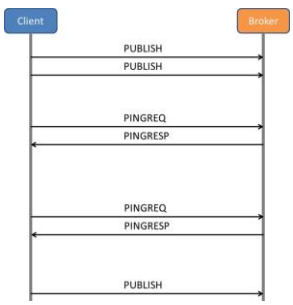
- Broker publishes the last will and testament message on behalf of the client upon detecting death of the client connection
- Useful for reporting problems
- Real push on device “death”
- Enables applications to know when a client goes offline abnormally
- Typically used for monitoring the connection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




# MQTT keepalive (heartbeats)

- Protocol includes support for client and broker to detect failed connections
  - At connection time, a keepalive value can be specified
  - Heartbeat interval

- If client does not send a ping request to the broker within specified interval the broker assumes the connection has failed
- If client does not receive a ping response within the specified interval after sending the ping request the client assumes the connection has failed
- Maximum keepalive interval is 18 hours
- A value of 0 disables heartbeats




http://www.embedded101.com/Develop-M2M-IoT-Devices-Ebook/DevelopM2MIoTDevicesContent/ArticleId/224/3-7-Message-Flow

# 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

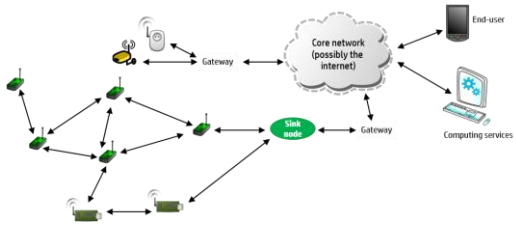
- Broker
  - Publish/subscribe permissions (ACL's)
  - Integration to other systems
  - ...

Many MQTT brokers support authentication and authorization plugins, allowing developers to customize access to resources to meet specific requirements.



# MQTT-SN

- MQTT for Sensor Networks
- Variation aimed at embedded devices on non-TCP/IP networks
- Clients communicate via a gateway to a MQTT broker on an 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

# Notable broker implementations

- Mosquitto
  - http://mosquitto.org
  - C, small, fast standalone binary, MQTT only, fully standards compliant
- RabbitMQ MQTT plugin
  - http://rabbitmq.com
  - Erlang, enterprise-grade MQTT plugin for RabbitMQ; 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






# Notable broker implementations

- Mosca
  - https://github.com/mcollina/mosca
  - A Node.js-based MQTT 3.1 compliant broker; 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
  - APIs 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)…"*

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



# MQTT as a service

- Xively
  - https://xively.com/
  - Acquired by Google
  - https://www.thefastmode.com/mobile-network-operators-m-a/12114-google-to-acquire-xively-iot-platform-for-50-million
- Litmus Automation
  - http://litmusautomation.com/
- Octoblu
  - https://octoblu.github.io/
  - Acquired by Citrix
  - Now mostly open source



# MQTT as a service

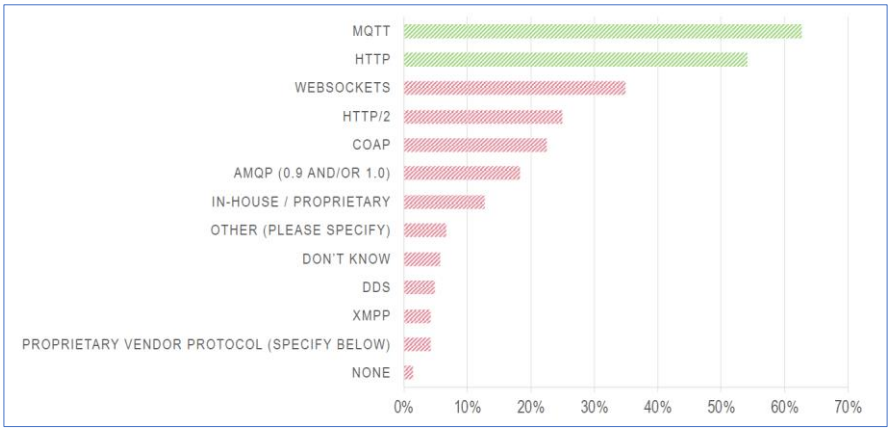
- Eurotech Everyware Device Cloud
  - http://www.eurotech.com/en/products/software+services/everyware+device+cloud
- Carriots
  - https://blog.carriots.com/mqtt-support-and-new-electric-imp-tutorial/
- AWS IoT
  - https://aws.amazon.com/iot/
- CloudMQTT
  - https://www.cloudmqtt.com/
- Microsoft Azure
  - https://azure.microsoft.com/en-us/free/iot/
- ... and various others

carriots

Everyware Cloud



# MQTT adoption




https://iot.eclipse.org/resources/iot-developer-survey/iot-developer-survey-2019.pdf

MQTT surpassed HTTP for the first time in 2018 as the preferred messaging protocol for IoT solutions.

# Alternatives to MQTT



- CoAP (Constrained Application Protocol)
  - Cisco
  - HTTP REST-like (request/response), but based on UDP instead of TCP
  - HTTP-like verbs and status codes
  - Transparent mapping to HTTP
  - Quality of service with confirmable messages
  - Resource discovery
- AMQP
  - Advanced Message Queuing Protocol
  - http://www.amqp.org
  - Considerable overlap with MQTT
  - Can accommodate most MQTT use cases, but incurs more overhead
  - More enterprise application oriented
- HTTP, OPC UA, WebSockets, …

# Agenda



- Introduction and background
- MQTT overview
- Some real-world examples
- Summary
- Questions

# 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 2011 https://www.facebook.com/notes/facebook-engineering/building-facebook-messenger/10150259350998920

www.vmssoftware.com

# 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
  - …

# 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.



# Train status information

- Deutsche Bahn wanted to provide displays that informed passengers at smaller stations about train status
- Additionally wanted the devices to be able to report back to the control centre about display device and message status

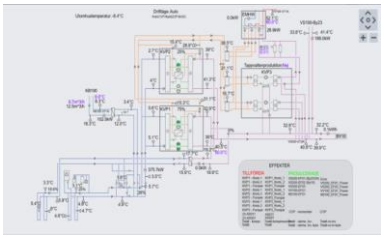
- An MQTT-based solution facilitated reliable messaging and a bi-directional messaging capability
- MQTT-based solution reduced messaging costs by 60 percent over the old SMS-based system
- Bi-directional messaging facilitates remote monitoring and rebooting of displays, significantly reducing maintenance costs

**Deutsche Bahn cuts messaging costs by 60 percent**

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

# Control Machines

- 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



- Provides a single view across multiple machine deployments (highly scalable)
- Generic

- Complemented by a powerful simulation package
  - Can be used to help guide, model, and tune physical design, perform what-if scenarios, ...



# 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
- Medical/healthcare
- 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

# Agenda

- Introduction and background
- MQTT overview
- Some real-world examples
- 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

- 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, containers, virtualization, …
  - Big Data and analytics
  - Advances in AI and machine learning
  - Advances in distributed computing
  - …


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



# Agenda

- Introduction and background
- MQTT overview
- Some real-world examples
- Summary
- Questions

# *Questions?*