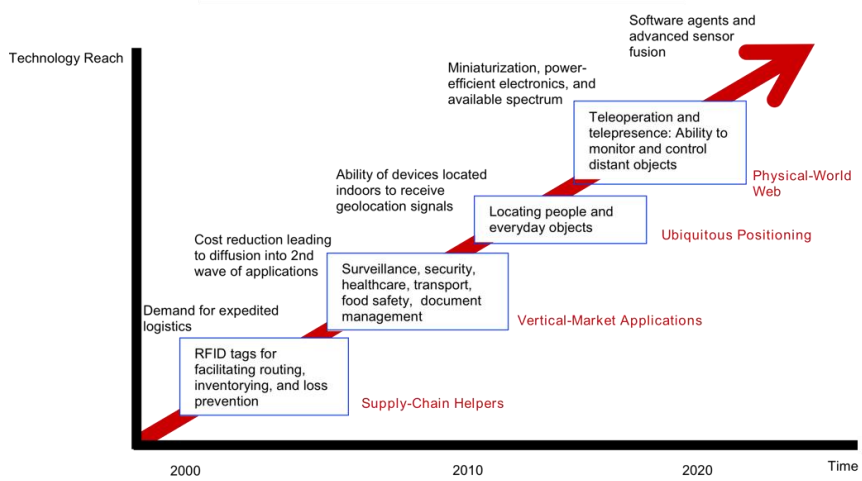
IoT technology roadmap



Source: SRI Consulting Business Intelligence

Challenges

- Efficiency and reliability
  - Power consumption and constrained devices
  - Scalability
  - Fault-tolerance and recoverability
  - Unreliable networks
  - Push messaging (polling does not scale)
  - …
- Security and privacy
  - A big area of concern
  - A number of interesting (and sometimes amusing) reports have recently appeared in the news
    - "*Experts hack smart LED light bulbs*": http://www.bbc.co.uk/news/technology-28208905
- Standards
  - Hardware
  - Communications protocols (more on this later)
  - …

With regard to security concerns, check out "*Yes, Your Refrigerator Is Trying To Kill You: Bad Actors and the Internet of Things*" (https://www.youtube.com/watch?v=Vd8dXzAL-W8), Beth Flanagan, OSCON 2014

Market potential and opportunities

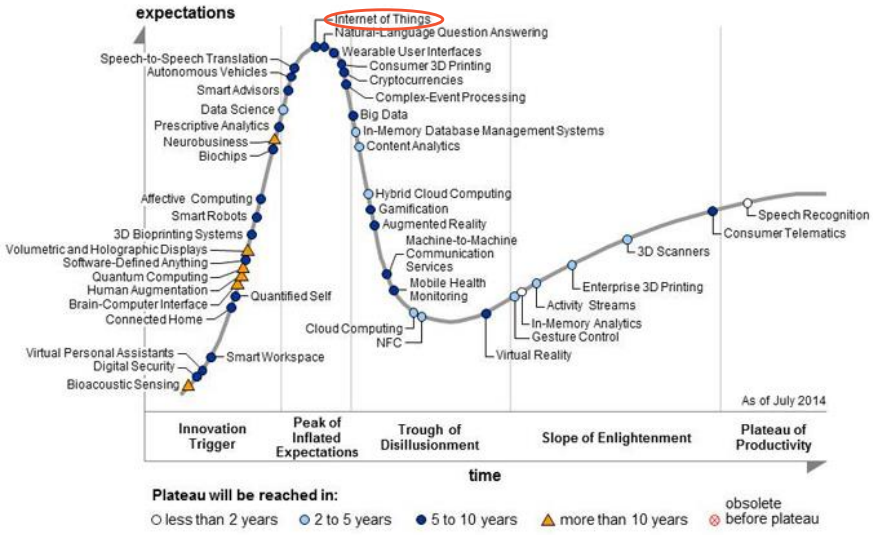
Cisco CEO Pegs Internet of Things as $19 Trillion Market:
http://www.bloomberg.com/news/2014-01-08/cisco-ceo-pegs-internet-of-things-as-19-trillion-market.html

- Emerging markets include (but are not at all limited to):
  - Consumers/retail
  - Home automation
  - Environmental monitoring
  - Commercial
  - Healthcare and monitoring
  - Cities/infrastructure (more on this shortly)
  - Wearable's
  - 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 could be applied.

19

Gartner Hype Cycle (August 2014)



20

Smart Cities and the iCity project

"The iCity project aims at making a step forward in fostering the co-creation of services of public interest by third parties pushing for their space as services providers in Smart Cities' urban spaces. This will be done by bootstrapping the involvement of open innovation ecosystems such as Living Labs in bringing the users together in the whole co-creation process using a self-centred methodology.

iCity involve into the co-creation of services of public interest different types of stakeholders: private and third sector as new services developers, and citizens and companies as beneficiaries and final users of the new services.

The project responds to the growing demand from social stakeholders to provide services of public interest based upon the exploitation of available public information, digital assets and infrastructure. In doing so, the project encompasses the concept of Open Data with a novel approach of Open Infrastructures where the available municipal ICT networks already deployed in urban spaces will be made available and accessible to open innovation ecosystems (specially SMEs) with the objective of maximizing the number of deployed services of public interest.

The services to be finally deployed will be designed by interested user-driven open innovation ecosystems according to selected priority areas for cities and will be developed on the top of a shared technological platform that the iCity project will integrate in the 4 participant cities: Barcelona, Bologna, Genova and London. The iCity platform will enable services deployment using on due time appropriate municipal infrastructures in each city and providing a software development kit (SDK). Transferring will be assured by making the platform be ready to be integrated in any other interested city willing to shift into a Smart City and adopt the co-creation approach in services provision and also by making available services of public interest to the citizenship on an iCity Open Apps Store."

(http://www.icityproject.eu/sites/default/files/iCity_factsheet.pdf)

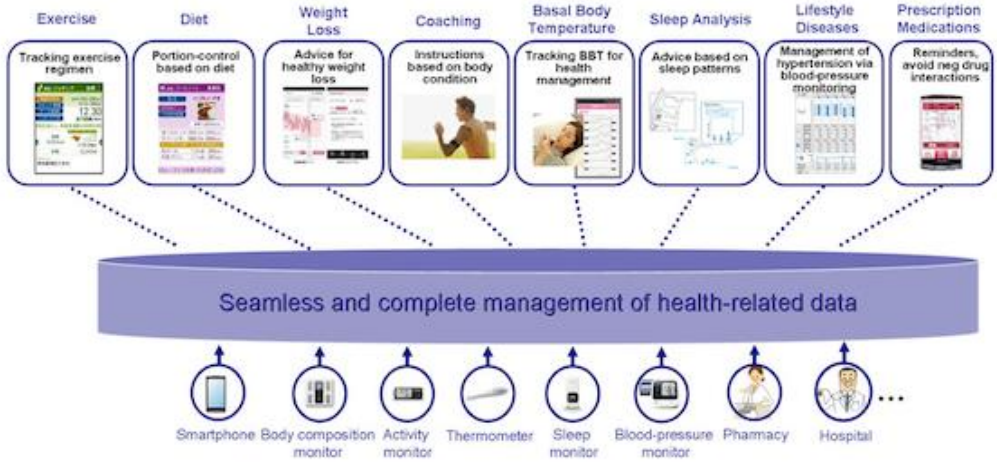
For more information about the iCity project, see http://www.icityproject.eu/

21

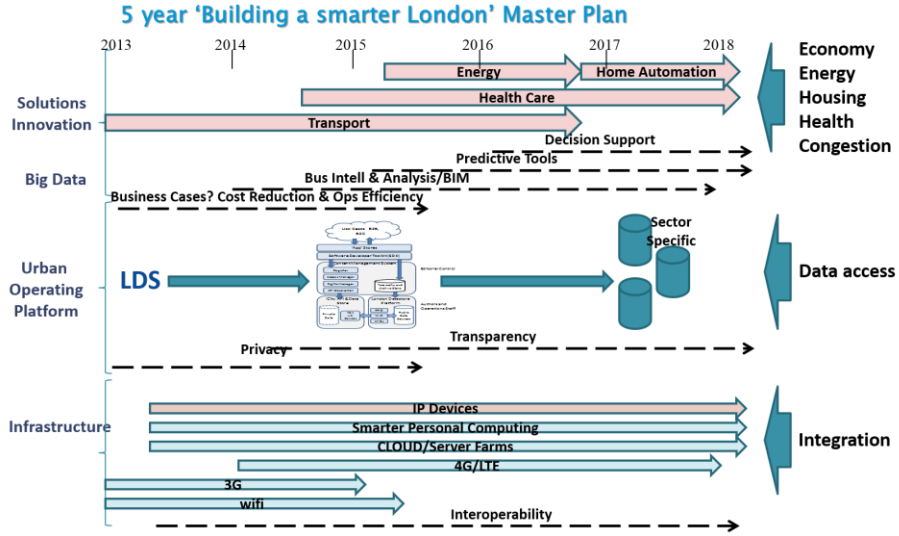
Smart Cities and the iCity project

22

Smart Cities and the iCity project



Smart Cities and the iCity project

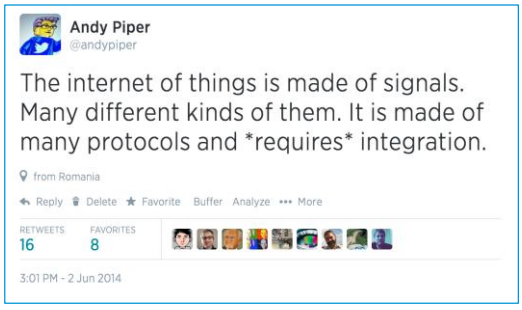


AGENDA

- Introduction and IoT overview
- **MQTT**
- MQTT, IoT, and OpenVMS
- Summary
- Questions and discussion

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



MQTT

- An Open Standard
  - 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 (the Paho-C client on OpenVMS is ~260K bytes)
  - Reliability levels, session awareness
  - Data-centric, payload-agnostic
    - Separates data (payload) from metadata (topic)
- Facilitate 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

27

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

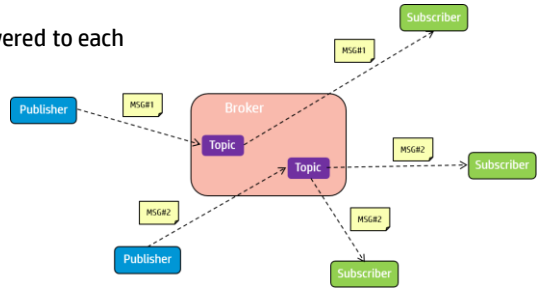
Adapted from http://wiki.eclipse.org/images/7/7c/MQTTIntroEclipseWebinarv1.pdf

28

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

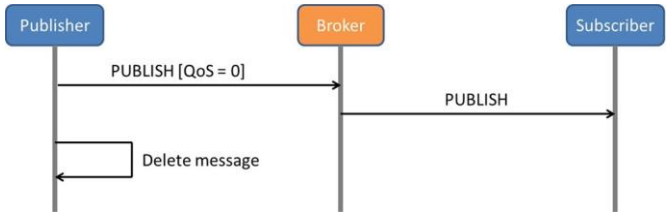
- A subscriber can subscribe to an absolute topic or use wildcards
  - Single-level wildcards "+" can appear anywhere in the topic string
  - Multi-level wildcards "#" must appear at the end of the string
  - Wildcards must be next to a separator
  - Producers cannot use wildcards (it would make no sense)

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)

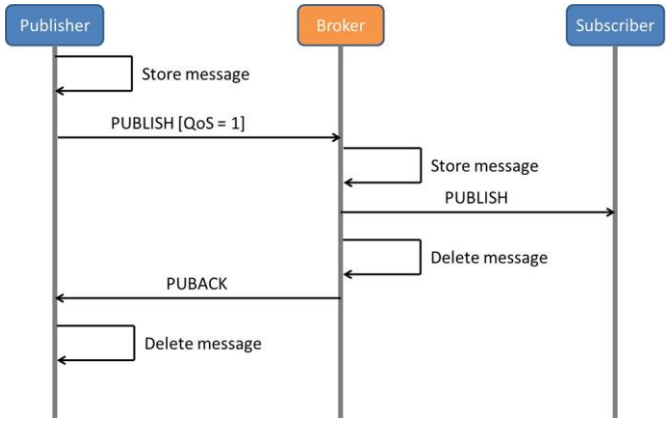


http://www.embedded101.com/Portals/0/images/DNNArticle/Windows-Live-Writer/3.7_E248/Fig3.81.jpg

- 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

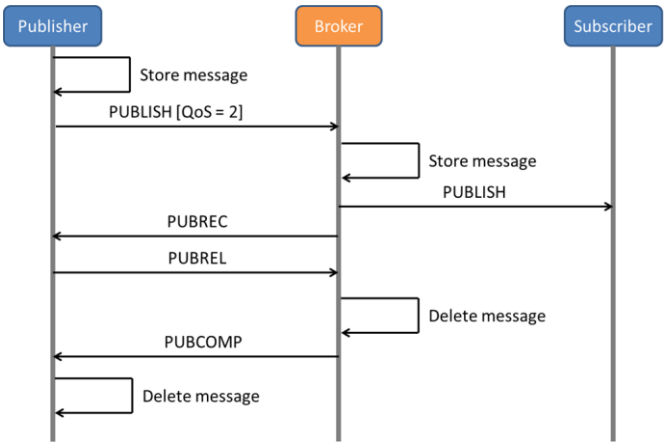


http://www.embedded101.com/Portals/0/images/DNNArticle/Windows-Live-Writer/3.7_E248/Fig3.9_2.jpg

- 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 IoT 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.

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-quality, MQTT plugin to AMQP 0.9.1 broker, currently not 100% compliant with MQTT standard
- HiveMQ
  - http://hivemq.com
  - Standalone Java MQTT broker, not Open Source, free for personal use
- Mosca
  - https://github.com/mcollina/mosca
  - A Node.js MQTT 3.1 compliant broker, which can be used standalone or embedded in another Node.js application
- ActiveMQ
  - http://activemq.apache.org/
  - Supports MQTT and several other protocols
- … and a few others (see http://mqtt.org/wiki/doku.php/brokers)

RabbitMQ
Messaging that just works

HiveMQ

ActiveMQ

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)

Others:

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

39

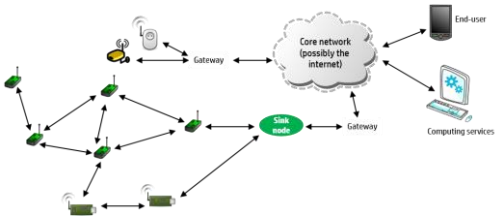
MQTT as a service

- Xively
  - https://xively.com/dev/docs/api/communicating/mqtts/
- Litmus Automation
  - http://litmusautomation.com/
- Meshblu Gateblu
  - http://skynet.im/, https://github.com/octoblu/gateblu
- Eurotech Everyware Device Cloud
  - http://www.eurotech.com/en/products/software+services/everyware+device+cloud
- 2lemetry ThingFabric
  - http://2lemetry.com/iot-platform/
- Carriots
  - https://blog.carriots.com/mqtt-support-and-new-electric-imp-tutorial/
- … and various others

40

MQTT-SN

MQTT for Sensor Networks



- MQTT for wireless sensor networks (WSN)
- A lightweight MQTT protocol implementation
- Datagram (UDP) instead of sockets
  - Very low power consumption
  - Very low bandwidth requirements
  - Very small client footprint
  - QoS reliable messaging in a potentially unreliable environment
- Clients are WSN nodes that communicate via a gateway to a broker on the TCP/IP network
  - Gateway translates messages between MQTT-SN and MQTT
- 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

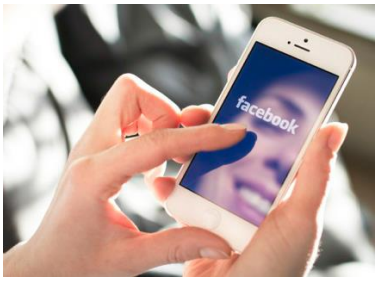
MQTT – real-world usage

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



MQTT – real-world usage

Smart homes

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

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

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
  - St Jude Medical - 100,000 heart pacemakers monitored via MQTT
- Railways

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-like (request/response) but based on UDP instead of TCP
- HTTP-like verbs and status codes
- 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 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

47

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.

48

AGENDA

- Introduction and IoT overview
- MQTT
- **MQTT, IoT, and OpenVMS**
- Summary
- Questions and discussion

MQTT, IoT, and OpenVMS

- The Mosquitto broker and Paho-C library have been ported to OpenVMS
  - OpenVMS 8.4 only at this time
  - Paho-C library requires latest CRTL
- Currently developing wrapper API for Paho-C that is more amenable to use by other OpenVMS 3GL's such as COBOL, FORTRAN, and Pascal
  - Click here to see a simple COBOL producer and here to see a simple FORTRAN consumer
- We also have RabbitMQ and the RabbitMQ MQTT plugin working on OpenVMS
  - See my talk "*The Polyglot Rabbit; Adventures on OpenVMS with RabbitMQ, Erlang, and multi-protocol messaging*"
- OpenVMS seems well-suited to the IoT world
  - High availability
  - Clustering
  - Security
  - MQTT and OpenVMS could be an ideal combination for many IoT-related solutions

AGENDA

- Introduction and IoT overview
- MQTT
- MQTT, IoT, and OpenVMS
- **Summary**
- Questions and discussion

Summary

- The Internet of Things is not just a cute name; it is big business and something that will change our world
- MQTT and other related technologies are integral to the Internet of Things
- OpenVMS has attributes that make it ideally suited to use by Internet of Things solutions

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

AGENDA

- Introduction and IoT overview
- MQTT
- MQTT, IoT, and OpenVMS
- Summary
- **Questions and discussion**