IoT: Cloud Services

Device to Cloud (D2C)

How do IoT communication with Cloud

- Core properties of an IoT messaging functionality are the reliability and durability of messages.
    - These properties enable resilience
        - intermittent connectivity on the device side
        - load spikes in event processing on the cloud side.
- IoT Cloud app must exposes the messages/events built-in endpoint for your back-end services to read the device-to-cloud received messages.

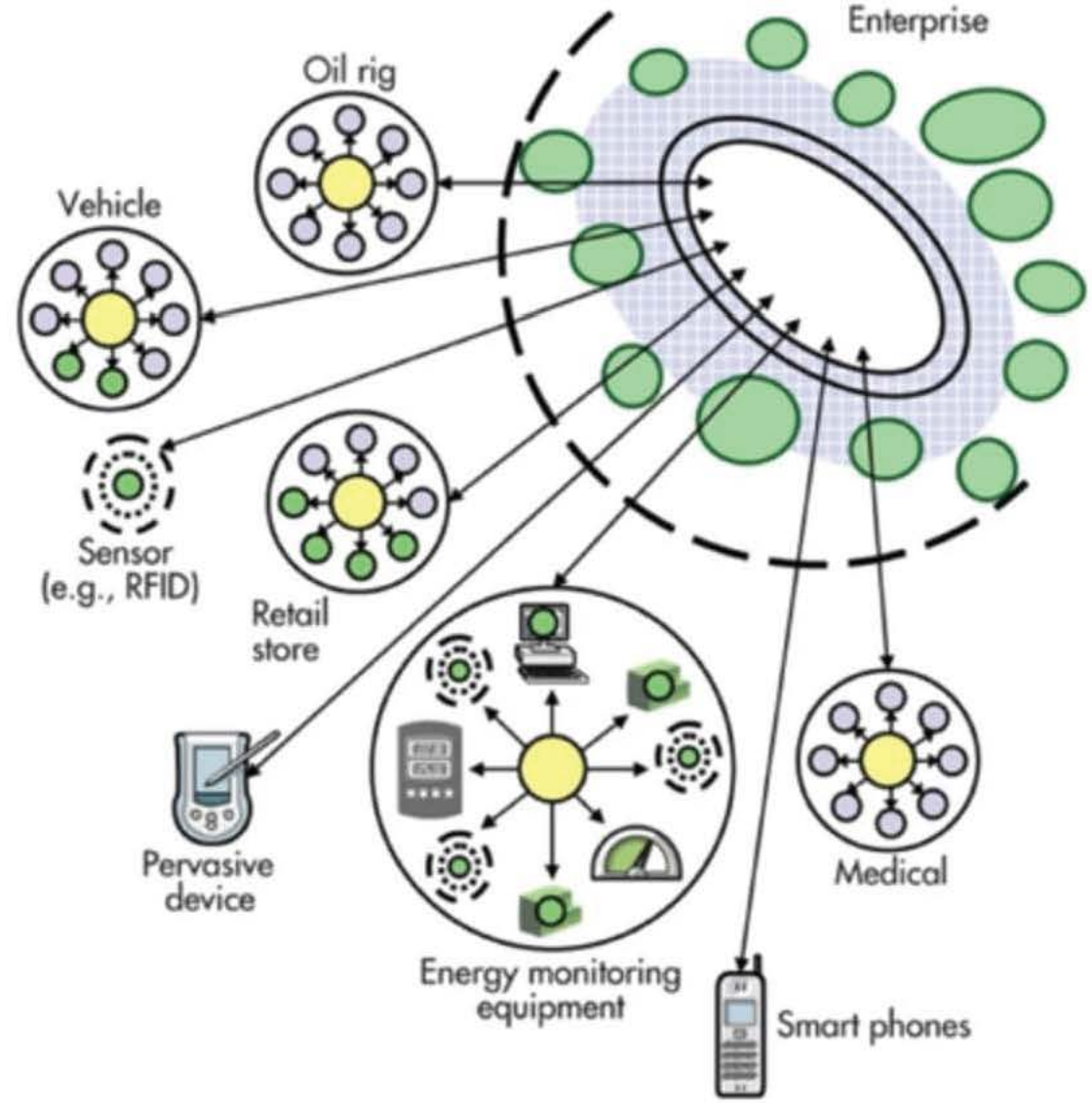
Device facing protocols

- MQTT
- AMPQ
- XMPP
- HTTP

MQTT

- MQ Telemetry Transport (MQTT) from IBM is the bad boy of small device messaging.
    - It uses TCP, is lightweight, and has features beneficial to IoT devices.
    - It is mature and there are a lot of client and server implementations, making it easier to develop upon.

- MQTT works on a pub/sub architecture. A client subscribes to a channel on a server, and when a server receives new information for that channel, it pushes it out to that device.



MQTT (cnt'd)

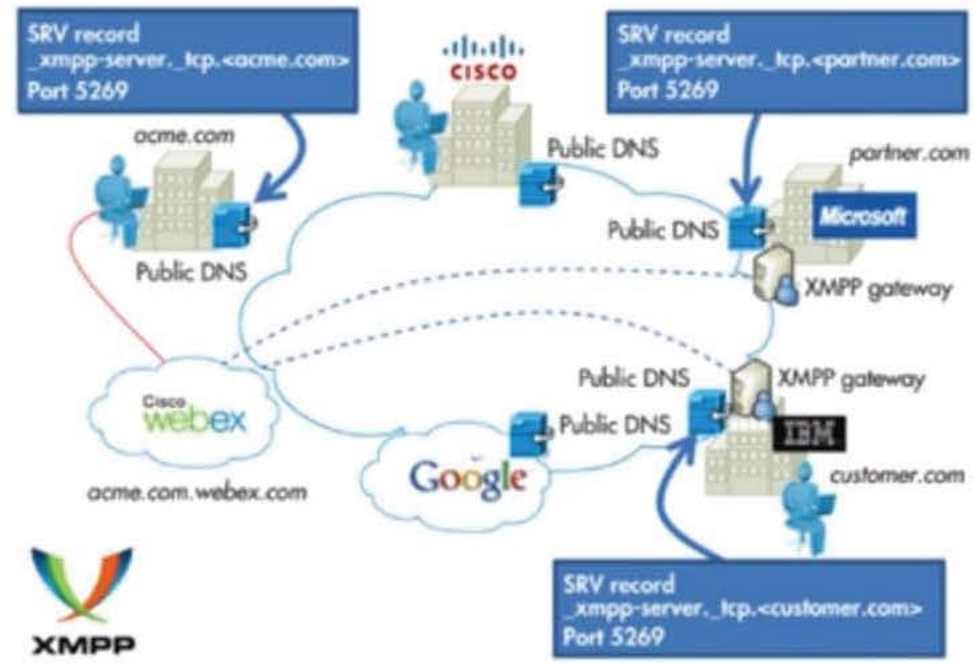
- Another great feature of MQTT is that you can set a priority or Quality of Service (QoS):
  - 0: The client/server will deliver the message once, with no confirmation required.
  - 1: The client/server will deliver the message at least once, confirmation required.
  - 2: The client/server will deliver the message exactly once by using a handshake process.
- Each level uses more bandwidth but will give you varying assurance of deliverability.
- With MQTT your application must have a library to talk to the MQTT server and handle publish/subscribe methods.

XMPP

- XMPP: a protocol best for connecting devices to people, a special case of the D2C pattern, since people are connected to the servers.

- XMPP was originally called “Jabber.” It was developed for instant messaging (IM) to connect people to other people via text messages.

- XMPP stands for Extensible Messaging and Presence Protocol. Again, the name belies the targeted use: presence, meaning people are intimately involved.

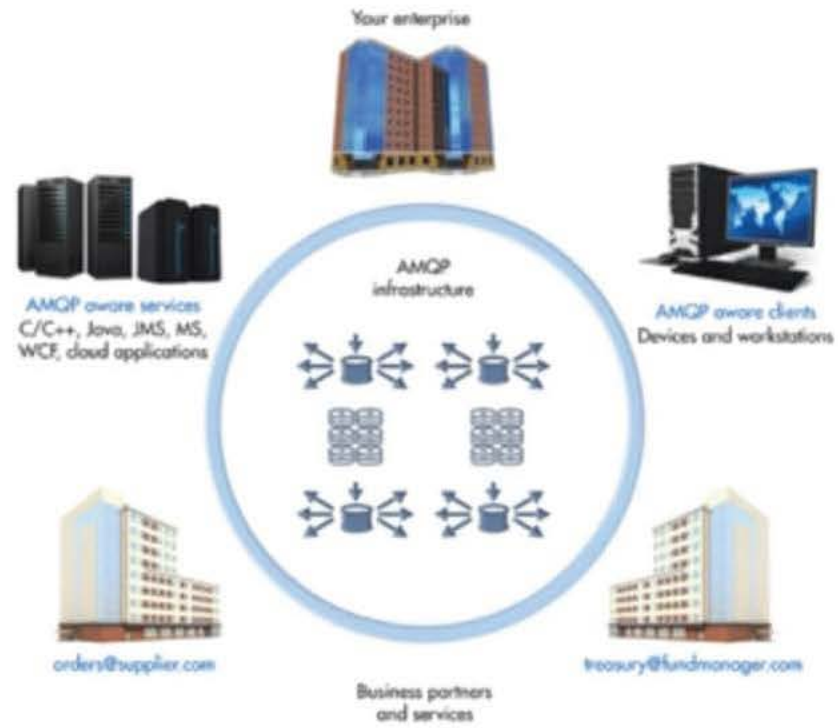


XMPP (cnt'd)

- XMPP uses the XML text format as its native type, making person-to-person communications natural.
    - Like MQTT, it runs over TCP, or perhaps over HTTP on top of TCP.
    - Its key strength is a name@domain.com addressing scheme that helps connect the needles in the huge Internet haystack.
- IoT world:
    - XMPP offers an easy way to address a device. This is especially handy if that data is going between distant, mostly unrelated points, just like the person-to-person case.
    - It's not designed to be fast. In fact, most implementations use polling, or checking for updates only on demand.
    - A protocol called BOSH (Bidirectional streams over Synchronous HTTP) lets severs push messages. But "real time" to XMPP is on human scales, measured in seconds.

AMPQ

- Advanced Message Queuing Protocol (AMQP) is sometimes considered an IoT protocol.
- It sends transactional messages between cloud services.
- As a message-centric middleware that arose from the banking industry, it can process thousands of reliable queued transactions.



AMPQ (cnt'd)

- AMQP is focused on not losing messages.
  - Communications from the publishers to exchanges and from queues to subscribers use TCP, which provides strictly reliable point-to-point connection.
  - Further, endpoints must acknowledge acceptance of each message.
    - The standard also describes an optional transaction mode with a formal multiphase commit sequence.
    - True to its origins in the banking industry, AMQP middleware focuses on tracking all messages and ensuring each is delivered as intended, regardless of failures or reboots.
- AMQP is mostly used in business messaging. It usually defines "devices" as mobile handsets communicating with back-office data centers. In the IoT context, AMQP is most appropriate for the control plane or server-based analysis functions.

IoT: Cloud Services

Device to Cloud (D2C) - part #2

WebSockets

- Now that browser websockets are standardized, they make a very compelling option for reading and writing data to IoT devices.

- Websockets are the best way to achieve full push/pull communications to a web server, which is not possible over basic HTTP protocol.

- Websockets are great if you have a full web client! However, with many IoT devices, it is a lot of overhead which might not even be an option.

- Another downside of using Websockets is that you would need to come up with your own protocol for the transmission of data. This isn't too difficult (for example, using JSON, XML) but does create a non-standard protocol.

Protocol over WebSocket

- A frequent use of Websockets is to actually use MQTT/XMPP/AMPQ over Websockets.
- WebSocket defines how can you raise an HTTP connection into a bidirectional channel,
    - the problem WebSocket want to resolve is HTTP is unidirectional and it dominated the web.
- In a higher layer, MQTT/XMPP/AMPQ defined how several endpoint exchange data when they can talk with the same broker.
- Metaphor:
    - MQTT works like FedEx, you sign addresses, it deliver your express.
    - WebSocket works like road if TCP is the roadbed, it enables you transport your express from starting point to the endpoint.
    - FedEx walks on the road, MQTT works on WebSocket, TCP, SCTP also.
    - So if we build a real-time IoT system, the probably architecture is we use MQTT over TCP in the connected devices, and we use MQTT over WebSocket in the web pages.

IoT protocols

- The IoT needs many protocols.
- The ones outlined here differ markedly. Perhaps it's easiest to categorize them along a few key dimensions:
  - QoS
  - Addressing
  - Application

QoS of the transport protocol

- Quality of Service (QoS) control is a much better metric than the overloaded “real-time” term.
- QoS control refers to the flexibility of data delivery.
- A system with complex QoS control may be harder to understand and program, but it can build much more demanding applications.
- Example:
  - Consider the reliability QoS. Most protocols run on top of TCP, which delivers strict, simple reliability.
  - Every byte put into the pipe must be delivered to the other end, even if it takes many retries.
  - This is simple and handles many common cases, but it doesn’t allow timing control. TCP’s single-lane traffic backs up if there’s a slow consumer.

Discoverability

- Finding the data needle in the huge IoT haystack is a fundamental challenge.
- XMPP shines here for “single item” discovery.
  - Its “user@domain” addressing leverages the Internet’s well-established conventions.
  - XMPP doesn’t easily handle large data sets connected to one server.
- With its collection-to-a-server design, MQTT handles that case well. If you can connect to the server, you’re on the network.
- AMQP queues act similarly to Cloud services, but for C2C microservices.

Intended Applications

- Inter-device data use is a fundamentally different use case from device data collection.

- Example: For example, turning on your light switch (best for XMPP) is worlds apart from generating that power, monitoring the transmission lines (MQTT), or analyzing the power usage back at the data center (AMQP).

IoT: Cloud Services

D2C and C2D

Considerations

- Consider the following points when you choose your protocol for device-side communications:
    - HTTP does not have an efficient way to implement server push.
    - As such, when you are using HTTP, devices poll the cloud for cloud-to-device messages. This approach is inefficient for both the device and the Cloud Edge.
    - Under current HTTP guidelines, each device should poll for messages every 25 minutes or more.
    - On the other hand, MQTT and AMQP support server push when receiving cloud-to-device messages.
        - They enable immediate pushes of messages from IoT Hub to the device. If delivery latency is a concern, MQTT or AMQP are the best protocols to use.

Field Gateways

- In an IoT solution, a field gateway sits between your devices and your IoT Hub endpoints.
- It is typically located close to your devices.
- The IoT devices communicate directly with the field gateway by using a protocol supported by the devices.
- The field gateway connects to an IoT Cloud endpoint using a protocol that is supported by the Cloud Edge.
- A field gateway can be highly specialized hardware or a low-power computer running software that accomplishes the end-to-end scenario for which the gateway is intended.
- Other benefits: ability to multiplex the communication from multiple devices onto the same Cloud connection.

MQTT with Field Gateways

- When using MQTT and HTTP, you cannot connect multiple devices (each with its own per-device credentials) using the same TLS connection.

- Thus, for Field gateway scenarios, these protocols are suboptimal because they require one TLS connection between the field gateway and Cloud Edge for each device connected to the field gateway.

Low Resource Devices

- The MQTT and HTTP libraries have a smaller footprint than the AMQP libraries.

- As such, if the device has limited resources (for example, less than 1 MB RAM), these protocols might be the only protocol implementation available.

Network Traversal

- The standard AMQP protocol uses port 5671, while MQTT listens on port 8883, which could cause problems in networks that are closed to non-HTTP protocols.

- MQTT over WebSockets, AMQP over WebSockets, and HTTP are available to be used in this scenario.

Payload Size

- MQTT and AMQP are binary protocols, which result in more compact payloads than HTTP.

Cloud to Device Communication

- **Direct methods** for communications that require immediate confirmation of the result.
    - Direct methods are often used for interactive control of devices such as turning on a fan.
- **Twin's** desired properties for long-running commands intended to put the device into a certain desired state.
    - For example, set the telemetry send interval to 30 minutes.
- **Cloud-to-device messages** for one-way notifications to the device app.

Examples

	direct methods	Twin's desired properties	cloud to device messages
Scenario	Commands that require immediate confirmation, such as turning on a fan.	Long-running commands intended to put the device into a certain desired state. For example, set the telemetry send interval to 30 minutes.	One-way notifications to the device app.
Data flow	Two-way. The device app can respond to the method right away. The solution back end receives the outcome contextually to the request.	One-way. The device app receives a notification with the property change.	One-way. The device app receives the message
Durability	Disconnected devices are not contacted. The solution back end is notified that the device is not connected.	Property values are preserved in the device twin. Device will read it at next reconnection.	Messages can be retained by Cloud provider