Summary

Below is a summary of the most important links I came across during research. I'm still at a loss when it comes to "horizontal scaling" of MQTT brokers. I haven't been able to find a single "this is how it's done" example. With the exception of HiveMQ.

I'm still not sure how I will implement load-balance testing with mqtt clients. The best I could find is this tool called malaria. The second best being the taxi TRANSIT simulator (see link below).

TRANSIT: Flexible pipeline for IoT data with Bluemix and OpenWhisk

https://medium.com/openwhisk/transit-flexible-pipeline-for-iot-data-with-bluemix-and-openwhisk-4824cf20f1e0#.n5nl9z2cc

This is the best link I have found so far. It helped me understand Node-RED and Openwhisk a little better. It doesn't address the issue of scalability however, well at least I think it doesn't.

It's using an *"MQTT broker dedicated to your org"*. I've been spending considerable time looking at different brokers such as rabbitMQ, mosquitto, and mosca (node.js). Trying to figure out how I would deploy them as containers, but this solution avoids all that hassle by just using the Watson IoT platform and the Message Hub service (based on Kafka). Perhaps a cost evaluation/comparison of using this solution over containers should be drawn up? Containers could still be employed as the "OpenWhisk actions" just not for the load balancing of mqtt clients.

This solution uses object storage and apache spark however, which is a different direction than I'm going for. I think I'll be using Redis/Cloudant for storage in my solution, and not IBM Graph as I can't justify it's requirement. Also I don't have the time nor the intellect required to learn it.

How to Build an High AvailabilityMQTT Cluster for the Internet of Things

https://medium.com/@lelylan/how-to-build-an-high-availability-mqtt-cluster-for-the-internet-of-things-8011a06bd000#.qj1uofotv

This is the best link I could find on load balancing. It uses Redis database service to share between 2 separate mqtt servers and a HAProxy container for load balancing. It uses something called "nscale" however, so I'd have to come up with my own implementation for deploying using IBM DevOps services.

Testing RabbitMQ Clustering using Docker

http://www.levvel.io/blog-post/testing-rabbitmq-clustering-using-docker-part-1/

This is one tutorial I managed to complete over Christmas break. I have to admit I don't know what's going on here, I'd have to dig deeper if I do decide to use this implementation. I prefer the idea of using mosca (as above link) because it's using node.js.

This tutorial demonstrates high availability however, with built in test scripts, so it could be worth the effort of using it.

IOT: Kafka to MQTT bridge using Mosca

Here's an interesting link from the Irish Marine Institute. IBM Openwhisk has Kafka built into it's framework so there might be a way of using this. Although I don't think it will be used, I was just pleasantly surprised to see the Irish affiliation.

Simple Metrics Collector – Microservices Edition

https://developer.ibm.com/clouddataservices/2016/03/03/simple-metrics-collector-microservices-edition/

I wish I had found this link earlier. I'll be using it to wrap my head around "consumer microservices" and microservices in general. This example also demonstrates the use of multiple databases, depending on the topic.

RESTful API User Authentication with Node.js and AngularJS – Part 1/2: Server

RESTful API User Authentication with Node.js and AngularJS – Part 2/2: Frontend App

These 2 links will be very useful in trying prototype a working mobile application using the ionic framework. I won't touch this until I have a solid footing in everything else. It's just there as part of the "blue sky/happy path" development plan. My initial plan was to create a basic angular js front-end anyway, so if all I have in May is this tutorial working for the application, I'll be relatively happy as it demonstrates some for of end to end security.

More Ionic links (not interesting reads, but useful for me)

https://www.quora.com/Is-it-a-good-idea-to-replace-a-REST-API-server-with-a-MQTT-broker-since-MQTT-provides-2-way-communication-and-a-publish-subscribe-model

Ionic IOT (MQTT) Client using Eclipse Paho – Part 4

https://blog.codecentric.de/en/2014/09/home-pi-reloaded-home-automation-ionic-mqtt/

Message broker with REST endpoints

Below is an interesting read that discusses the combination of REST and MQTT, something that is frying my brain at the moment. For now I'm resolving to an architecture that is strictly MQTT for embedded devices (e.g cars). And REST http for the mobile app. Otherwise the big picture starts to look messy and confusing. Example question→ is the mobile app using MQTT and REST?

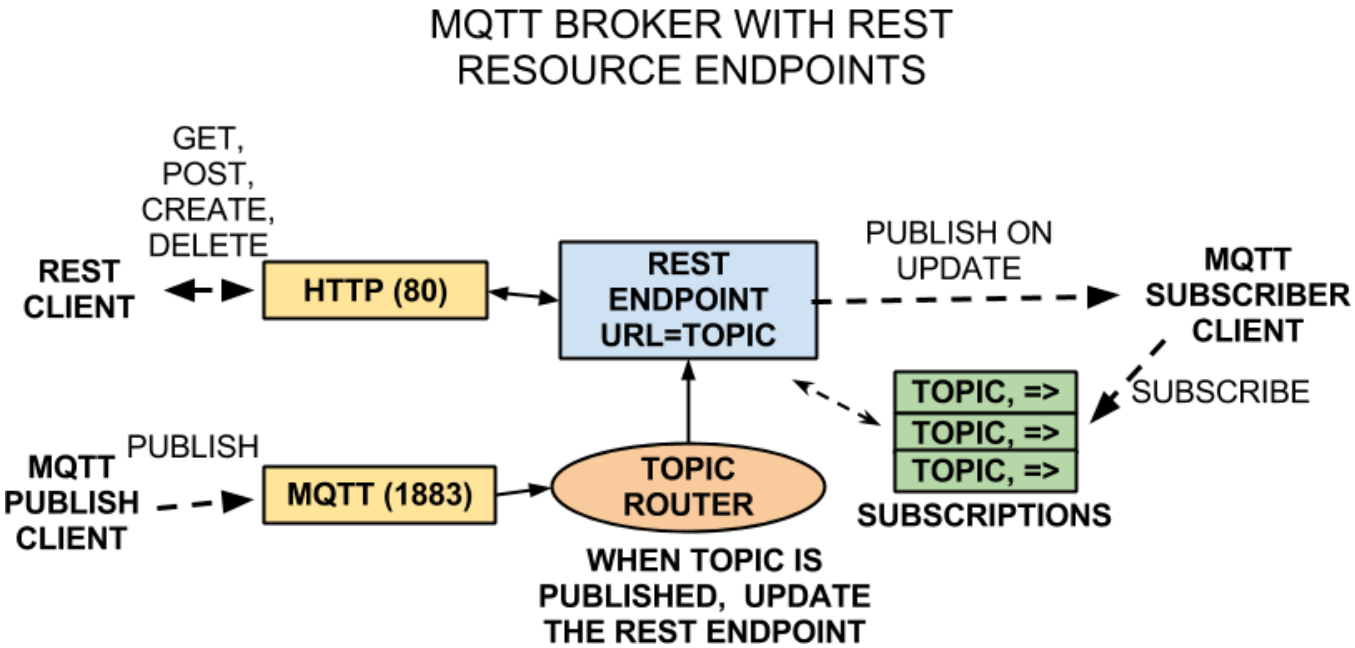
Message broker with REST endpoints

Another set of abstractions is to enable the configuration and of existing message systems such as MQTT using REST endpoints. There are two interfaces, a message interface and a REST interface, to shared stored resources.

There are two message operations supported, to enable clients or brokers to publish updates to REST endpoints through a message interface, and to publish updates of REST endpoints to brokers or clients subscribing through a message interface. There is also a subscribe operation using the message interface. Incoming messages update resources, and resource updates send outgoing messages.

The convention is to use URIs as topics that allow the system to construct URLs for the REST resource endpoints. The payload consists of the resource representation, e.g. JSON. This handles incoming and outgoing publish operations and acts as a message broker with REST access to topics and, if there is storage, topic history storage.

REST Message Broker allows message clients to publish to REST endpoints and subscribe to REST endpoints. An application can monitor an MQTT endpoint by observing the REST resource it publishes to, and an application can publish to an MQTT client by updating to the REST endpoint the MQTT endpoint is subscribed to. For example, sensors could connect with MQTT and publish updates for application software to read, and application software could update a REST endpoint, resulting in the system publishing the update to a sensor that is subscribed to the URL as topic.



The above example shows how an MQTT broker endpoint can be added to a REST server to enable REST clients and MQTT clients to share data by mapping URLs to topics. When either a REST client updates a resource an MQTT client publishes to a topic, the corresponding resource update is published to any subscribers.

Questions and answers

Below is a set of questions and answers I did out in a short space of time. Hopefully it will bring you up to speed as to where I’m at. This proved to be a very productive process as it highlighted what areas I need to buckle down on. Apologies for lighting and writing.

Q A

Q is it a nodejs backend for an Ionic (Angular) mobile application?

Ans → yes

Q ↳ will all communication [redacted] that takes place between backend & front end use MQTT? (for APP UI)

Ans → maybe? I know I want to use REST anyway

- Q ↳ will all other MQTT clients be notified when another one publishes?

Answer → I want the mobile clients to be updated once every 1 minute - 5 mins

Q updated on the data as published?

Option 1: Yes, the client streams the published data, once every few minutes

- Q what do you mean by stream?

(Rendering)
The client does the hard work of interpreting each GeoJson object as it comes, and updates the UI after every... Say 30 objects

Q How would account for error?

↳ Compare clients "total users" tally against DB
↳ Triggers microservice to do black magic if wrong

Q wont this option1 drain battery? Ans: yes, yes it would.

Questions

what is peak load?
↳ 10K msgs /per sec

what size is average msg?
↳ Compressed
↳ un Compresed

what is Mosca?
↳ an effort to implement the MQTT protocol using node.js

Why Redis in Mosca?
used as a pub/sub service in order to help mosca provide MQTT

All MQTT brokers share the same Redis store solution → where sharing happens

Q 11

Q How will the backend differ for an embedded device? will they be using REST ontop of MQTT???!

Ans I think the device should post/subscribe to a topic like "car-iot" or something.

The "mobile-iot" topic would handle the mobile user, granting him/her (access?) to the REST API.

Q So the only a mobile user can gain access to the map UI app is if they successfully transmit their location through MQTT?

Q what happens to the MQTT protocol?

Q your going to need it again in 15 mins when the client publishes their location again?

Ans I'm not sure how REST works with MQTT

Q why are you insisting on using REST?

Ans In order to decouple front and backend, to allow for IOS and Android development.

Q why not use REST for allcommunication?

Ans The mobile application might as well use the MQTT framework that will be put in place for other IoT devices. anyway.

what monitoring / feedback mechanisms will you implement?

ELK stack for debugging after the fact. (crash)

how will you roll out updates?

Amalgam8 microservices framework for A/B testing, canary deployment.

how will Amalgam8 work with Openwhisk?

Where does Node-Red come into this picture?
how will Node-Red be used for load balancing?

what are the costs for this application?

how big is a single payload from MQTT client?

How exactly does a mobile user get updated view of map?
time based trigger? how long? minimum? max?

what functions will open whisk be performing exactly?
what triggers?
why use an openwhisk microservice over Amalgam8 ms?