Putting MQTT in your toolkit

Sean Dague, Developer Advocate, IBM
July 19th 2018
Interaction Patterns

Client driven queries

Client \[\xrightarrow{\text{HTTP}}\] Server

Data

Server driven data distribution
Real Time Event Strategies

- **Client Polling**
  - Easy to implement
  - Really inefficient - 99+% noop

- **HTTP Long Poll**
  - Keep HTTP socket open, block on requests
  - Heavy server resource usage

- **Websockets**
  - Connected sockets in web browser

- **Webhooks**
  - Register callback URL
  - Need highly available "catcher" service
  - Only available for service owners

- **MQTT / AMQP**
  - Optimized pub/sub systems
  - Web client support requires bridging
Publish / Subscribe Design Pattern

- Common message bus
  - Everyone can publish to it
  - Messages directed to topics
  - Consumers subscribe to specific topics (possibly by wildcard)

- Good for many to many interactions
What is MQTT?

- MQTT is an open standard
  - Created in 1999 by IBM & Cirrus Link, OASIS standard since 2013

- MQTT is designed for “small footprint” and “limited network bandwidth”

- MQTT is the backbone of all major Public Cloud IoT services
  - Google IoT Core - [https://cloud.google.com/iot-core/](https://cloud.google.com/iot-core/)
  - Amazon IoT - [https://aws.amazon.com/iot-platform](https://aws.amazon.com/iot-platform)

- MQTT has many open source implementations
  - Mosquitto very popular broker
  - Clients bindings: A couple dozen languages
  - Even an Arduino library – PubSubClient

ESP8266
- Arduino with built in Wifi
- $9.95 at Adafruit
# MQTT Message Format

<table>
<thead>
<tr>
<th>QoS</th>
<th>Retain</th>
<th>Topic</th>
<th>Payload</th>
</tr>
</thead>
<tbody>
<tr>
<td>0,1,2</td>
<td>0,1</td>
<td>220 characters</td>
<td>Any content, up to 2 GB</td>
</tr>
</tbody>
</table>

- **QoS** - 0 best effort, 1 deliver at least once, 2 deliver exactly once
- **Retain** - content will be stored on the server, replayed on connect, defaults to not stored
- **Topic** - name for message, / are special
- **Payload** - anything, 2 GB payload limit

*Note: no metadata on packets (like time sent), must put it in payload manually*
What makes a good Topic Structure?

- Root with App Name
  - Assume multi tenancy

- Find any natural hierarchy of data

- Optimize for subscriber efficiency
  - Often need to rethink topic structure once you see usage patterns

- There is no formal schema system
  - Topic structure encoded into applications

Example:

{app name}/{location}/{device type}/{sensor type}

Allows the following subscriptions:

mysensors/portland/# - see everything in Portland

mysensors/+/weather/+ - see all weather sensors
MQTT Wills

- Event based system - messages only sent when event happens
- If nothing is sent, is the client healthy with no new data, or did it disappear?
- Clients can set a "Will" on client connect
  - a message stored in the server that will be sent if the socket connection to the client breaks
  - building block for fault tolerance
Example Applications
My Home Weather Station

- Bill of Materials
  - 10+ Oregon Scientific Weather Sensors
  - Raspberry Pi 3 – needs specific placement to see all sensors
  - RTL-SDR Dongle to decode 433Mhz signal out of air

- Long term (10 year) background hacking project

- Reports to Weather Underground via PWS API

- Collects temperature from nearly every room in the house
  - Extremely useful for tuning Forced Air HVAC system
In early 2017, discovered Home Assistant

How to integrate my weather station to this?
433 Mhz over the air

0x850022a9603814179
0xd54012a96038141600304060079

mqtt:
broker: 10.42.0.3
sensor 1:
platform: arwn

mosquitto

arwn/temperature/Freezer {"bat": "LOW", "sensor_id": "6a:03", "humid": 53.0, "temp": -10.8, "dewpoint": -23.2, "units": "F", "timestamp": 1527695510}
arwn/temperature/Arwen Room {"bat": "OK", "sensor_id": "ce:08", "humid": 54.0, "temp": 72.7, "dewpoint": 55.1, "units": "F", "timestamp": 1527695511}
arwn/wind {"bat": "OK", "sensor_id": "33:00", "timestamp": 1527695512, "units": "mph", "gust": 0.9, "speed": 2.5, "direction": 315.0}
Rain Gauge - Retain Topics

- Self emptying bucket
  - 3 increments added based on time between bucket dumps
- Reports Rain Total
  - (10ths of mm accumulator)
- Wunderground API
  - `rainin` - [rain inches over the past hour] -- the accumulated rainfall in the past 60 min
  - `dailyrainin` - [rain inches so far today in local time]
When Should I Charge My Car?

- Supports Time of Departure Charging
  - Be fully charged by a set time every day

- Time of Use metering at our home
  - Peak is 2 - 7pm Weekdays
  - Peak power costs 120%, off peak costs 89%

- What's the difference in power off peak?
  - What is generating the power?
  - What's the carbon intensity at different times?

- Can we make data available in real time?
  - Others might want to do things with this data
Data Graphs and Fuel Mix Chart

Real-Time Fuel Mix as of 02/12/2018 2:00pm EST

Energy generated within New York State

- Dual Fuel: 13.63%
- Natural Gas: 29.83%
- Nuclear: 35.47%
- Hydro: 22.15%
- Wind: 6.84%
- Other Renewables: 1.54%
- Renewables: 4.679

Output (MW)
Total: 15.270
Renewables: 4.679
## Real-Time Fuel Mix

<table>
<thead>
<tr>
<th>CSV Files</th>
<th>Last Updated</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>02-12-2018</td>
<td>02/12/18 14:10 EST</td>
<td></td>
</tr>
<tr>
<td>02-11-2018</td>
<td>02/12/18 00:05 EST</td>
<td></td>
</tr>
<tr>
<td>02-10-2018</td>
<td>02/11/18 00:05 EST</td>
<td></td>
</tr>
<tr>
<td>02-09-2018</td>
<td>02/10/18 00:05 EST</td>
<td></td>
</tr>
<tr>
<td>02-08-2018</td>
<td>02/09/18 00:05 EST</td>
<td></td>
</tr>
<tr>
<td>02-07-2018</td>
<td>02/08/18 00:05 EST</td>
<td></td>
</tr>
<tr>
<td>02-06-2018</td>
<td>02/07/18 00:05 EST</td>
<td></td>
</tr>
<tr>
<td>02-05-2018</td>
<td>02/06/18 00:05 EST</td>
<td></td>
</tr>
<tr>
<td>02-04-2018</td>
<td>02/05/18 00:05 EST</td>
<td></td>
</tr>
<tr>
<td>02-03-2018</td>
<td>02/04/18 00:05 EST</td>
<td></td>
</tr>
</tbody>
</table>

## Archived Files (zip format)

<table>
<thead>
<tr>
<th>CSV Files</th>
<th>Last Updated</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>02-2018</td>
<td>02/12/18 14:10 EST</td>
<td></td>
</tr>
<tr>
<td>01-2018</td>
<td>02/01/18 00:05 EST</td>
<td></td>
</tr>
<tr>
<td>12-2017</td>
<td>01/01/18 00:05 EST</td>
<td></td>
</tr>
<tr>
<td>11-2017</td>
<td>12/01/17 00:05 EST</td>
<td></td>
</tr>
<tr>
<td>10-2017</td>
<td>11/01/17 00:05 EDT</td>
<td></td>
</tr>
<tr>
<td>09-2017</td>
<td>10/01/17 00:05 EDT</td>
<td></td>
</tr>
<tr>
<td>08-2017</td>
<td>09/01/17 00:05 EDT</td>
<td></td>
</tr>
<tr>
<td>07-2017</td>
<td>08/01/17 00:05 EDT</td>
<td></td>
</tr>
<tr>
<td>06-2017</td>
<td>07/01/17 00:05 EDT</td>
<td></td>
</tr>
<tr>
<td>05-2017</td>
<td>06/01/17 00:05 EDT</td>
<td></td>
</tr>
<tr>
<td>04-2017</td>
<td>05/01/17 00:05 EDT</td>
<td></td>
</tr>
<tr>
<td>03-2017</td>
<td>04/01/17 00:05 EDT</td>
<td></td>
</tr>
<tr>
<td>02-2017</td>
<td>03/15/17 11:23 EDT</td>
<td></td>
</tr>
<tr>
<td>01-2017</td>
<td>03/15/17 11:05 EDT</td>
<td></td>
</tr>
<tr>
<td>12-2016</td>
<td>03/15/17 11:24 EDT</td>
<td></td>
</tr>
<tr>
<td>11-2016</td>
<td>03/15/17 10:36 EDT</td>
<td></td>
</tr>
<tr>
<td>10-2016</td>
<td>11/01/16 00:05 EDT</td>
<td></td>
</tr>
<tr>
<td>09-2016</td>
<td>10/01/16 00:05 EDT</td>
<td></td>
</tr>
<tr>
<td>08-2016</td>
<td>09/01/16 00:05 EDT</td>
<td></td>
</tr>
<tr>
<td>07-2016</td>
<td>08/01/16 00:05 EDT</td>
<td></td>
</tr>
<tr>
<td>06-2016</td>
<td>07/01/16 00:05 EDT</td>
<td></td>
</tr>
<tr>
<td>05-2016</td>
<td>06/01/16 00:05 EDT</td>
<td></td>
</tr>
<tr>
<td>04-2016</td>
<td>05/01/16 00:05 EDT</td>
<td></td>
</tr>
<tr>
<td>03-2016</td>
<td>04/01/16 00:05 EDT</td>
<td></td>
</tr>
<tr>
<td>02-2016</td>
<td>03/01/16 00:05 EDT</td>
<td></td>
</tr>
<tr>
<td>01-2016</td>
<td>02/01/16 00:05 EDT</td>
<td></td>
</tr>
</tbody>
</table>
ny-power microservices architecture

NY ISO CSVs

<table>
<thead>
<tr>
<th>Time Stamp</th>
<th>TZ</th>
<th>Fuel</th>
<th>Gen MW</th>
</tr>
</thead>
<tbody>
<tr>
<td>05/09/2018 00:05:00</td>
<td>EDT</td>
<td>Dual Fuel</td>
<td>1400</td>
</tr>
<tr>
<td>05/09/2018 00:05:00</td>
<td>EDT</td>
<td>Natural Gas</td>
<td>2144</td>
</tr>
<tr>
<td>05/09/2018 00:05:00</td>
<td>EDT</td>
<td>Nuclear</td>
<td>4114</td>
</tr>
</tbody>
</table>

ny-power-pump

MQTT publish

ny-power-status/fuel-mix/updated ("ts": "05/09/2018 00:05:00")
ny-power/upstream/fuel-mix/Nuclear ("units": "MW", "value": 4114, "ts": "05/09/2018 00:05:00")
ny-power/upstream/fuel-mix/Dual Fuel ("units": "MW", "value": 1400, "ts": "05/09/2018 00:05:00")
ny-power/upstream/fuel-mix/Natural Gas ("units": "MW", "value": 2144, "ts": "05/09/2018 00:05:00")

ny-power-influx

ny-power-archive

ny-power/web

ny-power/archive/co2/24h ("units": "g / kWh", "values": [162.698, 163.928, 161.587 ... ], "ts": [...])

ny-power/computed/co2 ("units": "g / kWh", "value": 135.088, "ts": "05/09/2018 00:05:00")

ny-power-mqtt (access with "mosquitto_sub -h 169.60.78.157 -t ny-power/# -v")

http://ny-power.org
ny-power topics

{app name}/{source}/{details}/{more details}

Allows the following subscriptions:

ny-power/upstream/fuel-mix/{fuel type}
ny-power/computed/co2
ny-power/archive/co2/24h
```javascript
var client = new Paho.MQTT.Client("mqtt.ny-power.org", Number("80"), "client-" + Math.random());

// set callback handlers
client.onMessageArrived = onMessageArrived;

// connect the client
client.reconnect = true;
client.connect({onSuccess: onConnect});

// called when the client connects
function onConnect() {
    // Once a connection has been made, make a subscription and send a message.
    console.log("onConnect");
    client.subscribe("ny-power/computed/co2");
    client.subscribe("ny-power/archive/co2/24h");
    client.subscribe("ny-power/application/webui");
    client.subscribe("ny-power/upstream/fuel-mix/#");
}

// called when a message arrives
function onMessageArrived(message) {
    console.log("onMessageArrived:" + message.destinationName + message.payloadString);
    if (message.destinationName == "ny-power/computed/co2") {
        var data = JSON.parse(message.payloadString);
        $("#co2-per-kwh").html(Math.round(data.value));
        $("#co2-units").html(data.units);
        $("#co2-updated").html(data.ts);
    }
    if (message.destinationName.startsWith("ny-power/upstream/fuel-mix")) {
        fuel_mix_graph(message);
    }
    if (message.destinationName == "ny-power/archive/co2/24h") {
        var data = JSON.parse(message.payloadString);
        var plot = [
            {x: data.ts, y: data.values, type: 'scatter'}
        ];
        Plotly.newPlot('co2_graph', plot);
    }
    if (message.destinationName == "ny-power/application/webui") {
        if (message.payloadString == "reload") {
            location.reload();
        }
    }
}
```
CO2 calculated from 2016 totals (MW & Emissions Per fuel source)

Provided as MQTT stream

Answer: complete charging by 5am before load / NG starts ramping up

http://github.com/IBM/ny-power
- Helm Kube application
- Core logic in python
- 5 pods
Put MQTT in your toolkit

- In a world awash in data, efficient event streams are critical
- Open event streams can be a public good
- MQTT dominates IoT, but also useful in other domains
- The Pub / Sub programming makes you think of problems in new ways

Thank You!
Twitter: @sdague
Email: sean.dague@ibm.com / sean@dague.net
Blog: https://dague.net - software engineering, open source projects, climate & energy
Get the code:
- arwn: https://dague.net/arwn
- ny-power project: https://dague.net/ny-power
Call for Code inspires developers to solve pressing global problems with sustainable software solutions, delivering on their vast potential to do good. Bringing together NGOs, academic institutions, enterprises, and startup developers to compete build effective disaster mitigation solutions, with a focus on health and well-being. The American Red Cross, and the United Nations Human Rights Office combine for the Call for Code Award to elevate the profile of developers.

Award winners will receive long-term support through the Linux Foundation, financial prizes, the opportunity to present their solution to leading VCs, and will deploy their solution through IBM’s Corporate Service Corps. Developers will jump-start their project with dedicated IBM Code Patterns, combined with optional enterprise technology to build projects over the course of three months. Judged by the world’s most renowned technologists, the grand prize will be presented in October at an Award Event.