Messaging for the Internet of Things

Andreas Schreiber <andreas.schreiber@medando.de>

EuroPython 2013, Florence, July 3, 2013
Scientist, Head of department

Deutsches Zentrum für Luft- und Raumfahrt
German Aerospace Center

Founder, CEO

Python user since 1992

medando

cologne

PyHPC2013

PyCon. DE 2013
Why?

What is my motivation for messaging and the Internet of Things?

Quantified Self – tracking myself

• With sensors
• With smartphone apps
Steps
Activity
Blood Pressure
**Smartphone: Sleep, Coffee, Medication, Money**
Medando: **BloodPressureCompanion**
Medando: WeightCompanion

- Gewicht (kg)
  - 74,3
  - 74,4
  - 74,5
  - 74,6
  - 74,7
  - 74,8
  - 74,9
  - 75,0
  - 75,1
  - 75,2
  - 75,3
  - 75,4
  - 75,5
  - 75,6
  - 75,7

- Liste der Messwerte
  - Diese Woche
    - 10.02.13 15:16: 78,2 kg
      Waage: Withings
      Bauch: 95,0 cm  Hüfte: -
      BMI: 25,5  WHR: -  WHtR: 0,5
  - Letzte Woche
    - 05.02.13 15:16: 77,3 kg
    - 02.02.13 15:16: 75,7 kg
      Gesundes Mittagessen gehabt
  - Dieser Monat
    - 29.01.13 15:16: 76,3 kg
  - Letzter Monat
    - 03.01.13 15:16: 76,6 kg
  - Dieses Jahr

- Analyse
  - Verlauf

EuroPython 2013, Florence, 03.07.2013
Many Devices, Sensors, and Apps

Data Exchange
Internet of Things

Billions of devices, sensors, and chips

- Connected physical objects (or their virtual representation)
- Connected via the internet
- Uniquely identified
- They interact
The “Things” are

- Embedded controllers
- Sensors
- Actuators
Number of devices connected to the internet grow every day

50.000.000.000 “Things” by 2020
Communication

Internet of Things

Communication infrastructure
MQ Telemetry Transport

- Machine-to-machine (M2M) connectivity protocol
- Publish/subscribe messaging
- Expect unreliable networks with low bandwidth and high latency
- Expect clients with limited processing resources
- Provides Quality of Service, if network/environment allows
- Easy to implement
Broker

MQTT broker

Client

publish

Client

subscribe

Client

Client

MQTT broker

(optional)
bridge

topic/subtopic
MQTT Protocol

- One-to-many message distribution over TCP/IP
- Notifies if clients disconnect abnormally
- Message format
  - Fixed 2-byte header
  - Variable header for some message type
  - Payload (e.g., the topic or small pieces of data)
Topics

• Messages in MQTT are published on topics
• No need to configure, just publish on it
• Topics are hierarchical, with “/” as separator

my/home/temperature/kitchen
my/home/temperature/livingroom
my/server/temperature
### MQTT Implementations

#### Servers/Brokers
- IBM Websphere MQ
- RSMB
- Mosquitto
- Eclipse Paho
- MQTT.js
- Apache ActiveMQ
- RabittMQ
- HiveMQ

#### Libraries for
- C/C++
- Java
- Python
- Perl
- PHP
- Ruby
- ...

[http://mqtt.org/wiki/software](http://mqtt.org/wiki/software)
Open Source MQTT Broker

- [http://mosquitto.org](http://mosquitto.org)
- Implemented in C
- Source code on bitbucket
- Many binary packages
Starting a Broker

- Install it
  - `apt-get install mosquitto`

- Just start with config file
  - `mosquitto -c mosquitto.conf`
Mosquitto broker publishes status messages

$SYS/broker/messages/sent
$SYS/broker/subscriptions/count
$SYS/broker/uptime

...
Publicly available Mosquitto MQTT server/broker

**MQTT**

This is test.mosquitto.org. It hosts a publicly available Mosquitto MQTT server/broker. MQTT is a very lightweight protocol that uses a publish/subscribe model. This makes it suitable for "machine to machine" messaging such as with low power sensors or mobile devices.

For more information on MQTT, see [http://mosquitto.org/](http://mosquitto.org/) or the Mosquitto MQTT man page.

**The server**

The server listens on ports 1883, 8883 and 8884. Port 1883 is the standard unencrypted MQTT port and can be used with any MQTT client. Ports 8883 and 8884 use certificate based SSL/TLS encryption and require client support to connect. In both cases should use the certificate authority file [mosquitto.org](http://mosquitto.org/) to verify the server connection. Port 8883 allows unrestricted connections. Port 8884 requires clients to provide their own certificate to authenticate their connection. If you wish to obtain a client certificate, please get it touch.

You are free to use it for any application, but please do not abuse or rely upon it for anything of importance. You should also build your client to cope with the broker restarting.

If you have the mosquitto clients installed try:

- `mosquitto_sub -h test.mosquitto.org -t "#" -v`
Python client module

- Single file, pure Python implementation
- Publishing and receiving messages
- Callbacks
  - Connect
  - Disconnect
  - Publish
  - Message
  - Subscribe
import mosquitto

def on_message(mosq, obj, msg):
    print(msg.topic + ' ' + str(msg.payload))

mqtt_client = mosquitto.Mosquitto()
mqtt_client.on_message = on_message

mqtt_client.connect('test.mosquitto.org')
mqtt_client.subscribe('#', 0)  # all topics

return_code = 0
while return_code == 0:
    return_code = mqtt_client.loop()
import mosquitto

mqtt_client = mosquitto.Mosquitto()

mqtt_client.connect('test.mosquitto.org')

mqtt_client.publish('europython/demo', 'hello world', 1)
Tools for publishing and subscribing MQTT topics

- mqtt.io (Web)
- Eclipse Paho (Java library and Eclipse View)
- MQTT.app (Mac OS X)
- ...

See http://mqtt.org/wiki/software
MQTT.app (OS X)
Mosquitto on Android

- The Python module works with python-for-android
- Easy to use in Kivy clients
Xively – Public Cloud for the Internet of Things

test.mosquitto.org details

Feed ID: 43810
API Endpoint: https://api.xively.com/v2.feeds/43810

Channels: Last updated a few seconds ago

Graph: Last updated 14 minutes ago

6 hours averaged datapoints

2496
MQTT Usage Examples

• Home automation with Raspberry Pi
• Android Push Notification
Getting sensor data with sensors connected via 1-Wire

- **1-Wire**: Single line bus system, low-speed
- Sensors for temperature, voltage, light, humidity, ...
- Connected via 1-Wire-USB adapter
Temperature Sensor

http://www.iButtonLink.com
Temperature Sensor

http://www.iButtonLink.com
Mosquito works nicely on Raspberry Pi

- Just install
  - `apt-get install mosquitto`
- You can start the broker or clients
Getting Temperature

Getting measurements from 1-Wire devices on Linux

- Two solutions that work with Python
  - OWFS: One Wire File System (http://owfs.org)
  - DigiTemp and DigitemPy (http://www.digitemp.com)
import time
import os
import mosquitto

file_name = os.path.join('/', 'mnt', '1wire', '10.67C6697351FF', 'temperature')

mqtt_client = mosquitto.Mosquitto('home-temperature')
mqtt_client.connect('test.mosquitto.org')

while 1:
    file_object = open(file_name, 'r')
    temperature = '%s\00C' % file_object.read()
    mqtt_client.publish('home/demo/temperature', temperature, 1)
    mqtt_client.loop()
    time.sleep(5)
    file_object.close()
Getting data from Quantified Self gadgets to Android

- The Gadget sends data to “somewhere” in the Cloud
- Withings, Fitbit, and Nike provide APIs to access the data
- Register for callbacks to get notifications
- We use a Django app that registers as callback listener and send MQTT messages on updates
- MQTT Java client on Android receives notifications
MQTT Push Notification Architecture

Android phone
- register phone
- send messages

Django App
- register callback
- publishes notification
- receives callbacks

MQTT broker

Gadget Vendor (API)
- register callback
- measures

Gadget
• Implementation includes OAuth stuff
• Most complex part was the Java code on Android (error handling etc.)
• Deployment on Amazon Web Services
def callback(request):
    """ Callback function for Withings notifications. """

    ... # request parameter handling

devices = RegisteredWithingsUser.objects.filter(user_id=user_id)

mqtt_client = MosquittoHandler(len(devices))

for device in devices:
    device_id = device.device_id
    mqtt_topic = 'medando/weightcompanion/weights/%s/%s' %
                 (user_id, device_id)
    payload = simplejson.dumps({'startdate': startdate, 'enddate': enddate})
    mqtt_client.publish(mqtt_topic, payload, 2, True)

mqtt_client.wait()
### MQTT Messages

<table>
<thead>
<tr>
<th>URL</th>
<th>Startdate</th>
<th>Enddate</th>
</tr>
</thead>
<tbody>
<tr>
<td>medando/weightcompanion/weights/1883073/34bae8cbe8dd92f3</td>
<td>1371856646</td>
<td>1371856647</td>
</tr>
<tr>
<td>medando/weightcompanion/weights/1791607/898efc38ac5d4211</td>
<td>1372742400</td>
<td>1372742401</td>
</tr>
<tr>
<td>medando/weightcompanion/weights/1527601/2ebcf034b8585668</td>
<td>1368851117</td>
<td>1368851118</td>
</tr>
<tr>
<td>medando/weightcompanion/weights/16121/f2a8ca66fd067954</td>
<td>1372750563</td>
<td>1372750564</td>
</tr>
<tr>
<td>medando/weightcompanion/weights/449599/4d701e076912648f</td>
<td>1372751111</td>
<td>1372751112</td>
</tr>
<tr>
<td>medando/weightcompanion/weights/642578/b3335681163a389</td>
<td>1370585275</td>
<td>1370585276</td>
</tr>
<tr>
<td>medando/weightcompanion/weights/2019258/33b1d416aeae9ef</td>
<td>1371377131</td>
<td>1371377132</td>
</tr>
<tr>
<td>medando/weightcompanion/weights/2019258/61bdf242b37d8a29</td>
<td>1371377131</td>
<td>1371377132</td>
</tr>
</tbody>
</table>
Blutdruck vs. Gewicht

Korrelation: Blutdruck gegen Gewicht

Systolisch - Diastolisch — Regression - Systolisch — Regression - Diastolisch
Status Page

The slide features a status page with graphs and data for Medando messaging service. The graphs show:
- Total Clients
- Active Clients
- Received Messages
- Sent Messages

Below the graphs, there are sections for Information, Developers, Status, and Kontakt.

Information:
- MQTT
- Mosquito

Developers:
- Django Admin
- CODM

Status:

Kontakt:
- Medando Postfach 13 02 85
- 07135
- Tel: 0173-3123103
- Email: info@medando.de
Conclusions

- There are other message broker
- There are other push notification services
- MQTT is very lightweight
- Mosquitto is easy to use from Python
Questions?

Andreas.Schreiber@medando.de

@MedandoEN | @onyame