Bringing M2M to the web with Paho

Connecting Java Devices and online dashboards with MQTT

Your presenters

DC² square

Dominik Obermaier

Christian Götz

Agenda

- Introduction & Setup
- What is M2M, IoT and MQTT?
- Build a device simulator with Eclipse Paho in Java
- MQTT in the web browser
- Build a web dashboard with Paho JS
- Summary and outlook

What you will learn today

- M2M & MQTT
- Using MQTT in Java applications with Eclipse Paho
- Using MQTT in the browser with websockets
- Building Web Dashboards with real-time push capabilities with MQTT

Use case

- Device with sensors in your garden or balcony
- Frequent live updates on web dashboard
- The device should be controlled from a web dashboard
- Unreliable (cellular) network

Big Picture


HiveMQ
MQTT Broker
TCP
Websockets
Device Simulator
Web Dashboard

Required Software

- HiveMQ MQTT Broker
- Modern web browser
- Eclipse Kepler Java EE - Edition *

* Or any other Java Eclipse with WTP installed

Source Code

- *https://github.com/dc-square/mqtt-with-paho-eclipsecon2013*

Installation Eclipse

- Download Kepler for your OS

http://www.eclipse.org/downloads/packages/eclipse-ide-java-ee-developers/keplersr1

- Import the source code for the simulator & web dashboard
  - *File | Import | Existing Projects into workspace*

Installation HiveMQ

- Download our prepared archive (includes websockets configuration and MQTT Message Log plugin)
  - http://www.dc-square.de/hivemq-eclipsecon.zip
- **Windows**:
  - Double click on bin/run.bat in the extracted folder
- **Linux/BSD/MacOSX:**
  - Open Terminal and change to the extracted folder
  - *chmod 755 bin/run.sh*
  - *./bin/run.sh*

Verify HiveMQ Installation

2013-10-26 22:12:54,311 INFO - Activating $SYS topics with an interval of 60 seconds

2013-10-26 22:12:54,469 INFO - Starting on all interfaces and port **1883**

2013-10-26 22:12:54,482 INFO - Starting with **Websockets** support on all interfaces and port **8000**

2013-10-26 22:12:54,484 INFO - **Loaded Plugin HiveMQ MQTT Message Log Plugin** - v1.0.0

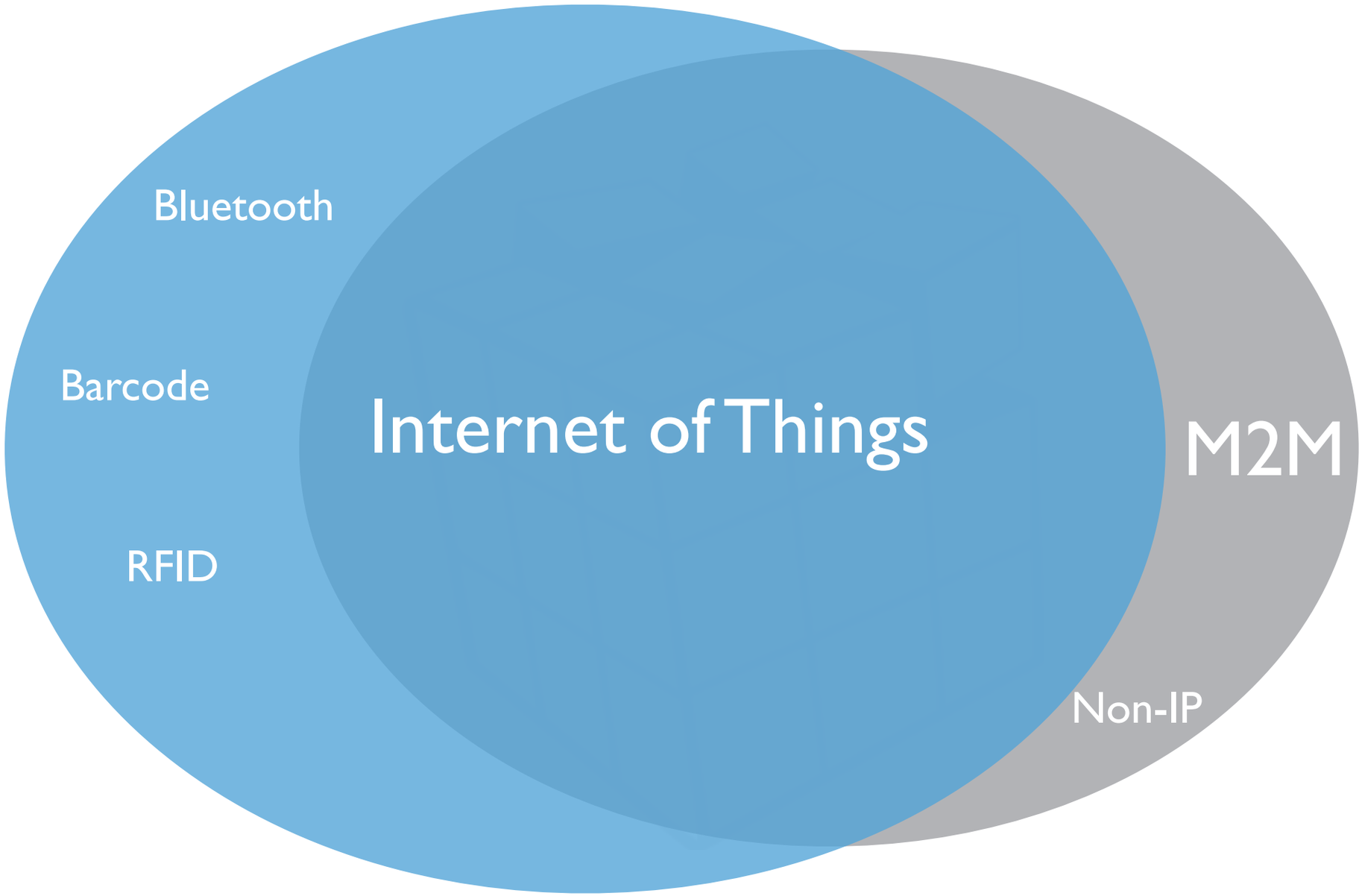
2013-10-26 22:12:54,488 INFO - **Started HiveMQ 1.4.1 in 3075ms**

MQTT in M2M & IoT

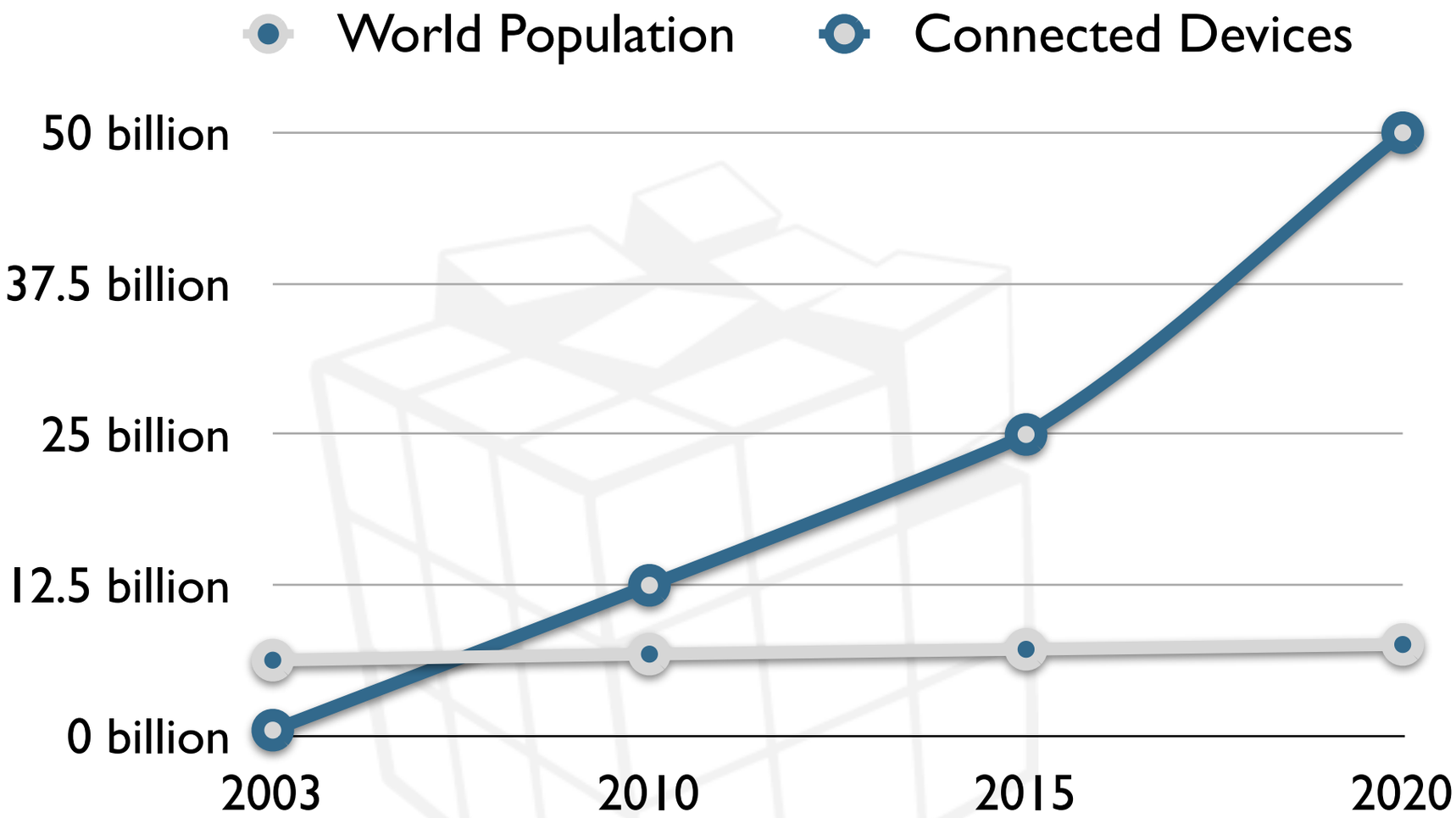
M2M

Technology that supports wired or wireless communication of devices

M2M & IoT



Why care about M2M/IoT?



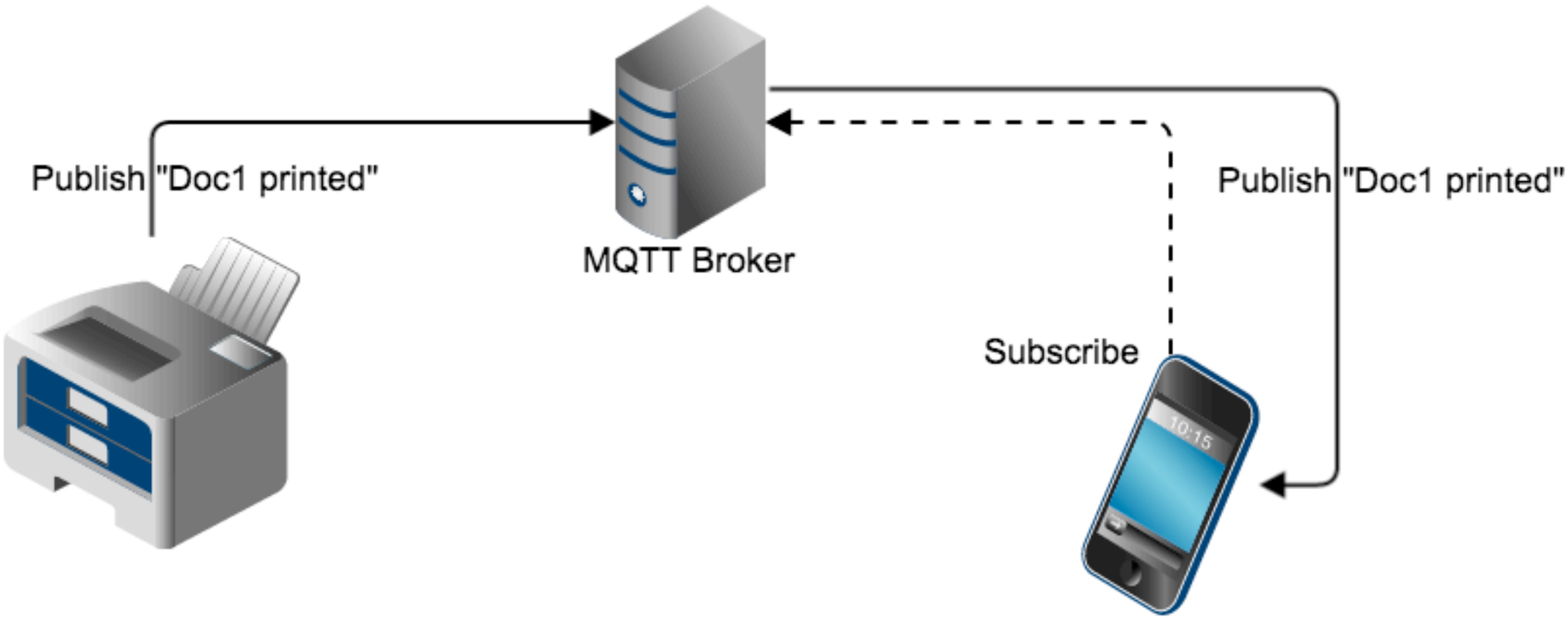
Devices per person:	0,08	1,8	3,4	6,5

Source: http://www.cisco.com/web/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf

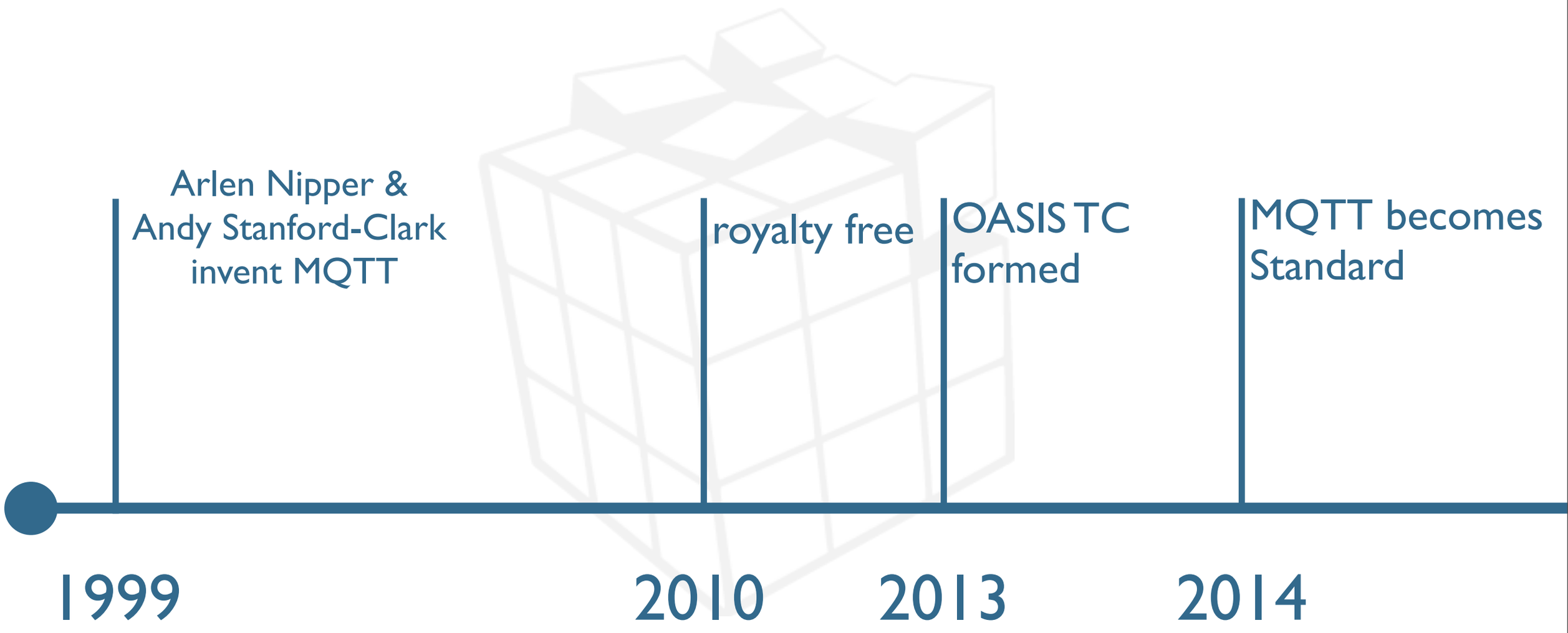
Disadvantages of HTTP

- Not optimized for mobile
- Only point-to-point communication
- (partly) too complex for simple data
- Too verbose
- No reliable exchange (without retry logic)
- No Push Capabilities

MQTT


Publish "Doc1 printed"
MQTT Broker
Publish "Doc1 printed"
Subscribe
10:15

History


Arlen Nipper &
Andy Stanford-Clark
invent MQTT
1999
royalty free
2010
OASIS TC
formed
2013
MQTT becomes
Standard
2014

Goals

- Simple
- Quality of Service
- Lightweight & bandwidth efficient
- Data-agnostic
- Continuous session aware

Features

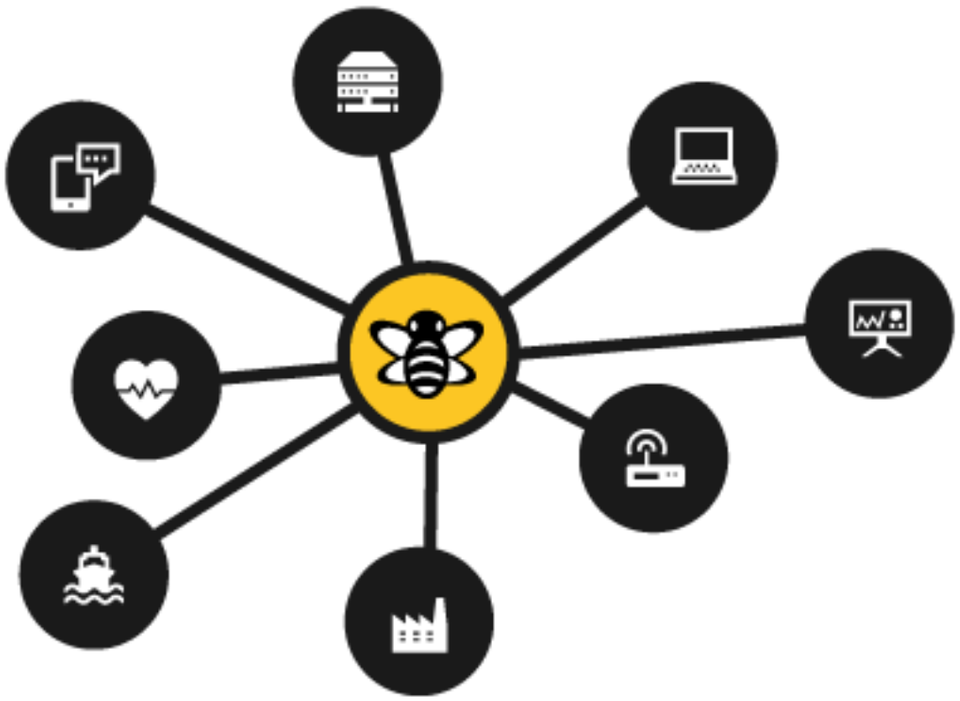
- Last Will and Testament
- Retained Messages
- Persistent Sessions
- Quality of Service
  - 0: At most once
  - 1: At least once
  - 2: Exactly once

MQTT Broker

- Different implementations are available
  http://mqtt.org/wiki/doku.php/brokers

- HiveMQ

  - High performance MQTT broker
  - Open Source Plugin System
  - Native Websockets Support
  - Cluster functionality
  - Embeddable

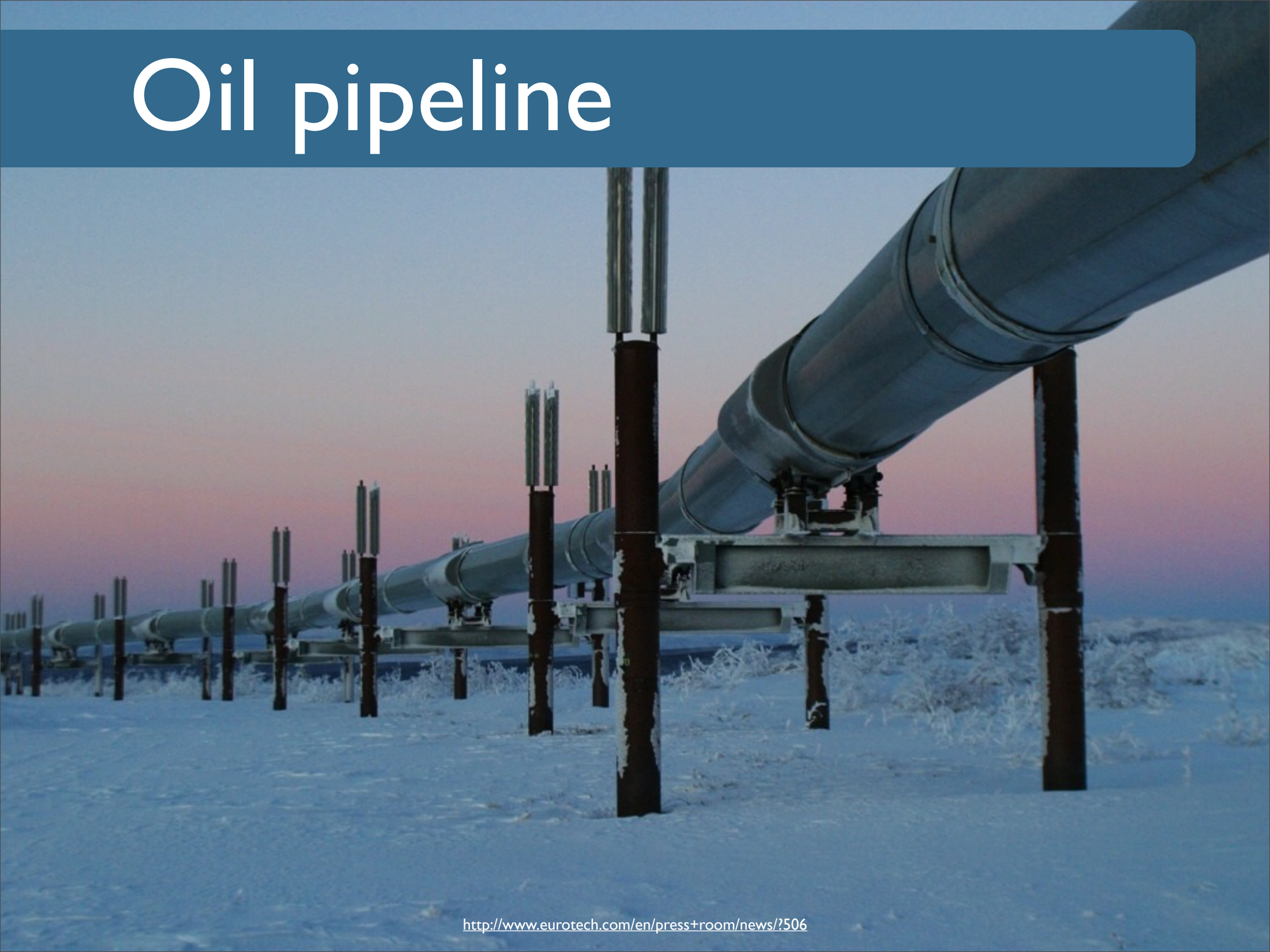


Prominent Examples

MQTT

IN ACTION

Oil pipeline



http://www.eurotech.com/en/press+room/news/?506

Facebook Messenger

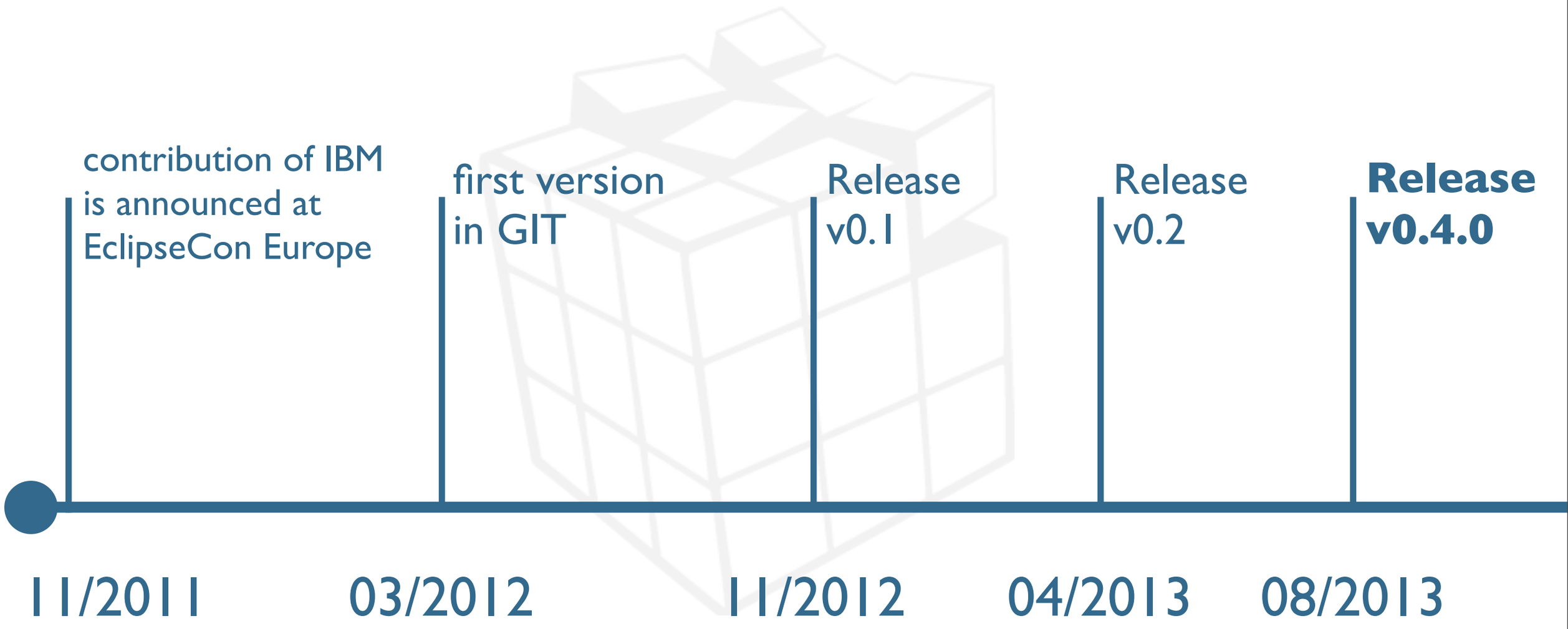
https://www.facebook.com/notes/facebook-engineering/building-facebook-messenger/10150259350998920


pahoo

Paho Project

- Protocol implementations for M2M & IoT
- Focus on MQTT
- C, C++, Java, JavaScript, Lua, Python
- Eclipse Incubator Project

History - Paho Java



Part 1: Simulator

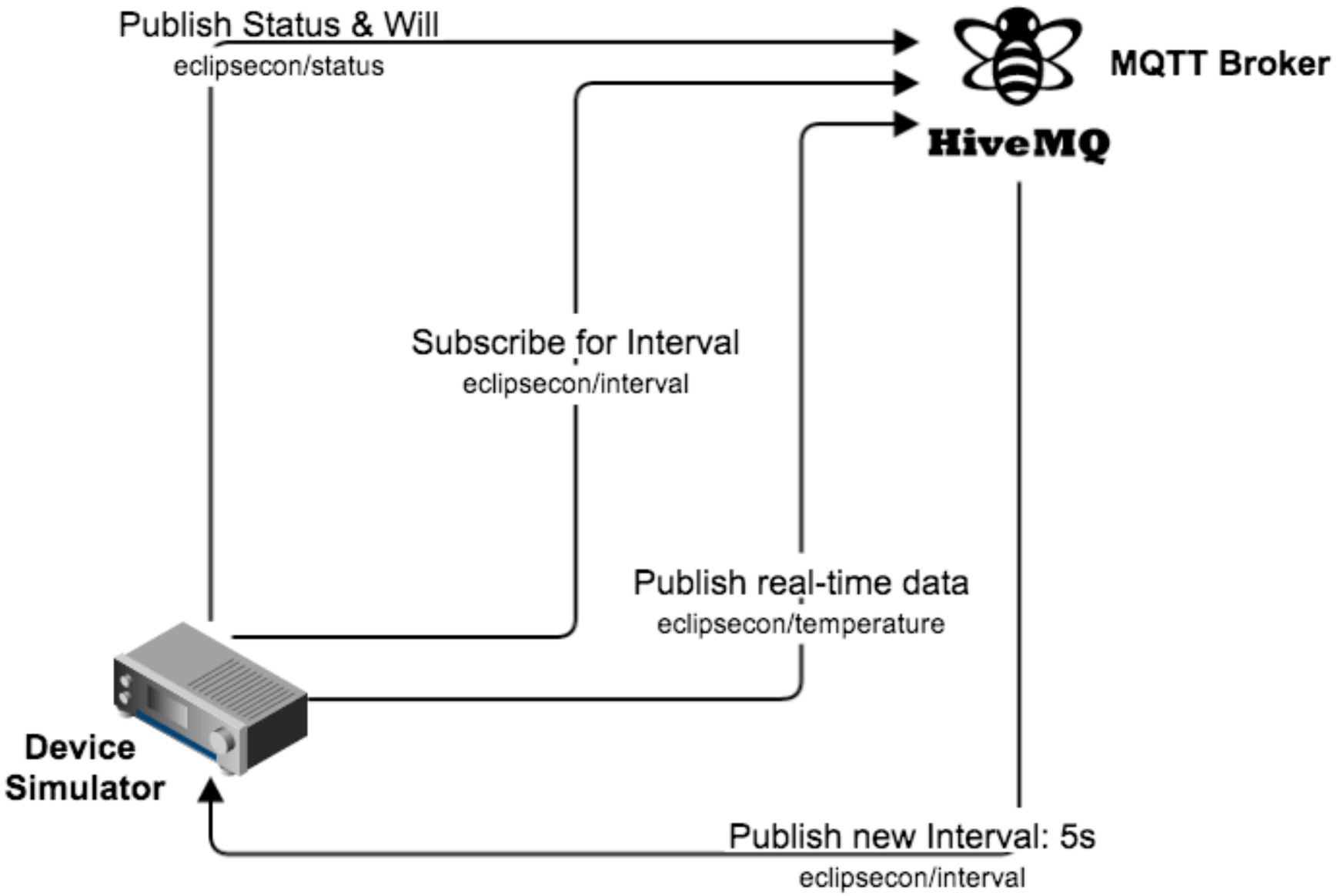
Focus


TCP
HiveMQ
MQTT Broker
Websockets
Device Simulator
Web Dashboard

Features

- Report Status of Device
- Send Data to MQTT Broker periodically
  - Temperature
  - Weather
  - Glaze Warning
- Change interval if requested by MQTT message

Under the Hood



Exercise 1

Paho API

```
MqttClient client = new MqttClient(BROKER_URL, CLIENT_ID);

client.connect();

client.publish(TOPIC, MESSAGE, QoS, true);


client.setCallback(new SubscribeCallback(this, TOPIC));

client.subscribe(TOPIC, QoS);
```

Connect

```
client = new MqttClient(BROKER_URL, CLIENT_ID, new MemoryPersistence());

client.connect();
```

- Broker URL *e.g. tcp://localhost*
- Unique ClientID
- Persistence Provider

Extended Connect

```
client = new MqttClient(...);

final MqttConnectOptions mqttConnectOptions = new MqttConnectOptions();

mqttConnectOptions.setWill(TOPIC, "message".getBytes(), 2, true);

client.connect(mqttConnectOptions);
```

- ConnectionOptions
  - setWill
    - Topic
    - Message
    - Quality of Service Retained Flag

Publish

```
client.publish(TOPIC, "message".getBytes(), 2, true);
```

- Topic
- Message
- QoS
- Retained Flag

Subscribe

```
client.setCallback(new SubscribeCallback());
client.subscribe(TOPIC, 2);
```

```
class SubscribeCallback implements MqttCallback {

    @Override
    public void connectionLost(Throwable cause) {
    }

    @Override
    public void messageArrived(String topic, MqttMessage message)
                                                        throws Exception {
       System.out.println(topic);
       System.out.println(new String(message.getPayload(), "UTF-8"));

      }

    @Override
    public void deliveryComplete(IMqttDeliveryToken token) {
    }
}
```

Do it Yourself !

- Connect to your local HiveMQ broker
  - Choose unique client id
  - Broker URL **tcp://localhost**
  - Use simple connect (without LWT)
  - Connect

Start at "S1_Start"

Add Status of Device

- Set Last Will and Testament
  - Topic: *eclipsecon/status*
  - Message: offline
- Publish message on *eclipsecon/status* after connect
  - Use retained message

Start at "S2_Publish_Status_Messages"

Publish Temperature

- Use Random Temperature Generator
- Query for new value every 5 sec
- Publish to *eclipsecon/temperature*

Start at "S3_Publish_Temperatur_Periodically"

Adjust Interval

- Subscribe to *eclipsecon/interval*
- Implement callback to change the interval on new message arrival
- Use interval in temperature publish routine

Start at "S4_Subscribe_to_Interval_Update"

Weather Status

- Use *RandomGenerator*
- Publish to *eclipsecon/weather*

Start at "S5_Add_Publish_Weather"

Glaze Warning

- Use *RandomGenerator*
- Publish to *eclipsecon/glaze*

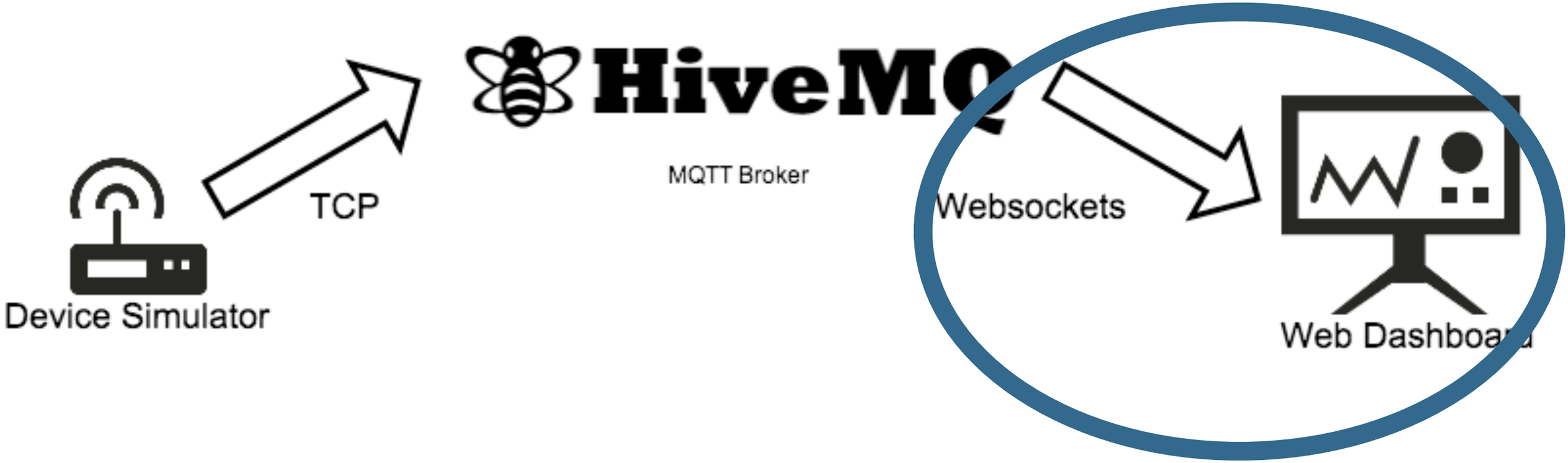
Start at "S6_Add_Publish_Glazed_Warning"

Simulator completed!

complete simulator located in S7_Complete_Solution

Part II: Web Dashboard

Focus


HiveMQ
MQTT Broker
TCP
Device Simulator
Websockets
Web Dashboard

Websockets

- Full duplex communication over a single TCP connection
- Implemented in most modern web browsers
- Websockets over TLS (secure websockets) possible
- Allows sub protocols

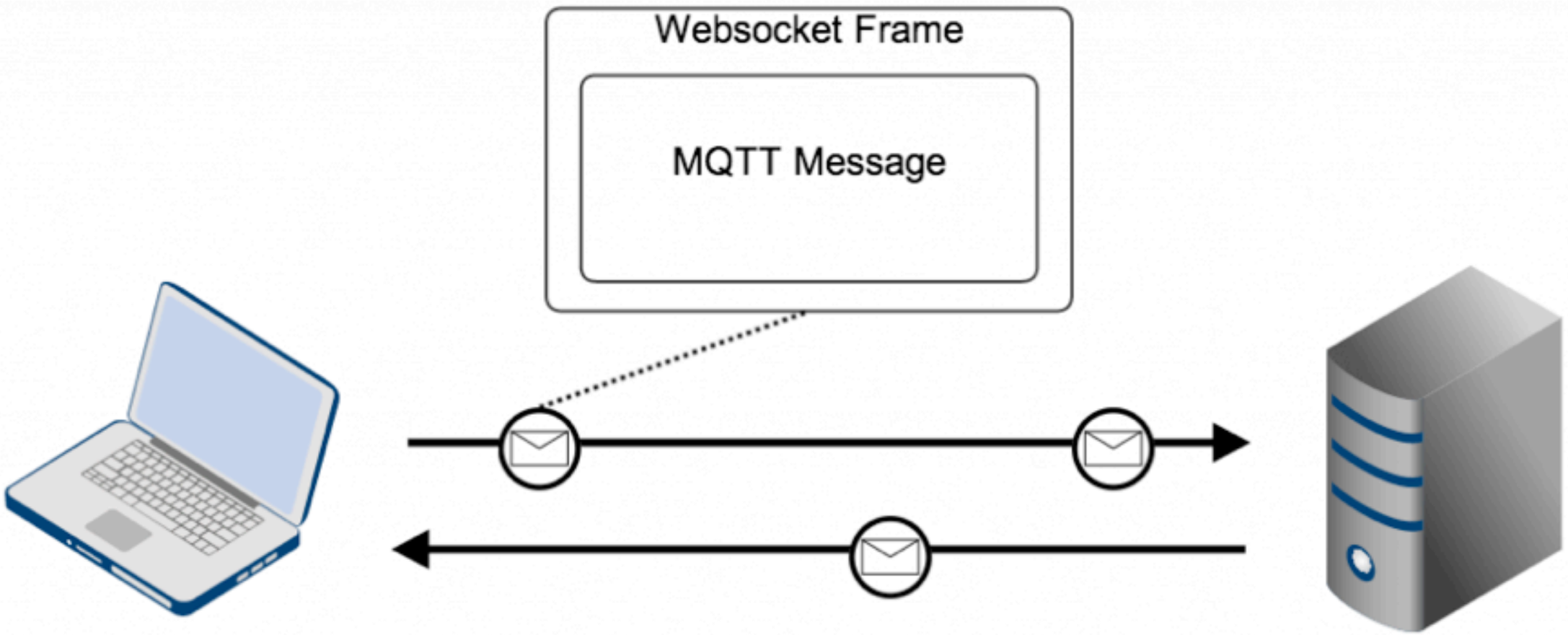
Advantages Websockets

- Real Push to the browser
- No Request / Response (like classic HTTP)
- Server can actively work with the connection
- Minimum overhead

MQTT over Websockets

- Every browser can be a MQTT “device”
- Queuing of messages with QoS > 0
- Retained messages
- LWT
- Persistent Sessions

How does it work?



Javascript Basics I

- Dynamically typed
- Runs in the browser *
- Interpreted language
- Multi-paradigmal
- Powerful function concept

* Nowadays Javascript can also run on the server with node.js.

Javascript Basics II

```
var myObject = {
      sayHi : function() {
            console.log('hi');
      },
      myName : 'EclipseCon'
};
myObject.sayHi();
```

Javascript Basics III

```
var greet = function(person, greeting) {
    var text = greeting + ', ' + person;
    return text;
};

console.log(greet('EclipseCon','Hi'));
```

Paho.JS

- MQTT Javascript implementation
- Designed for usage in browsers
- Contributed by IBM in 2013
- Very well structured and documented source

The Dashboard

Weather Dashboard

Hello, EclipseCon!

This is a template for creating your MQTT Dashboard for the Eclipsecon 2013 Session

"Bringing M2M to the web with Paho: Connecting Java Devices and online dashboards with MQTT"

Glaze Warning!

Take care while driving, it could be icy today!

Temperature

Weather

Control Center

Client is connected!

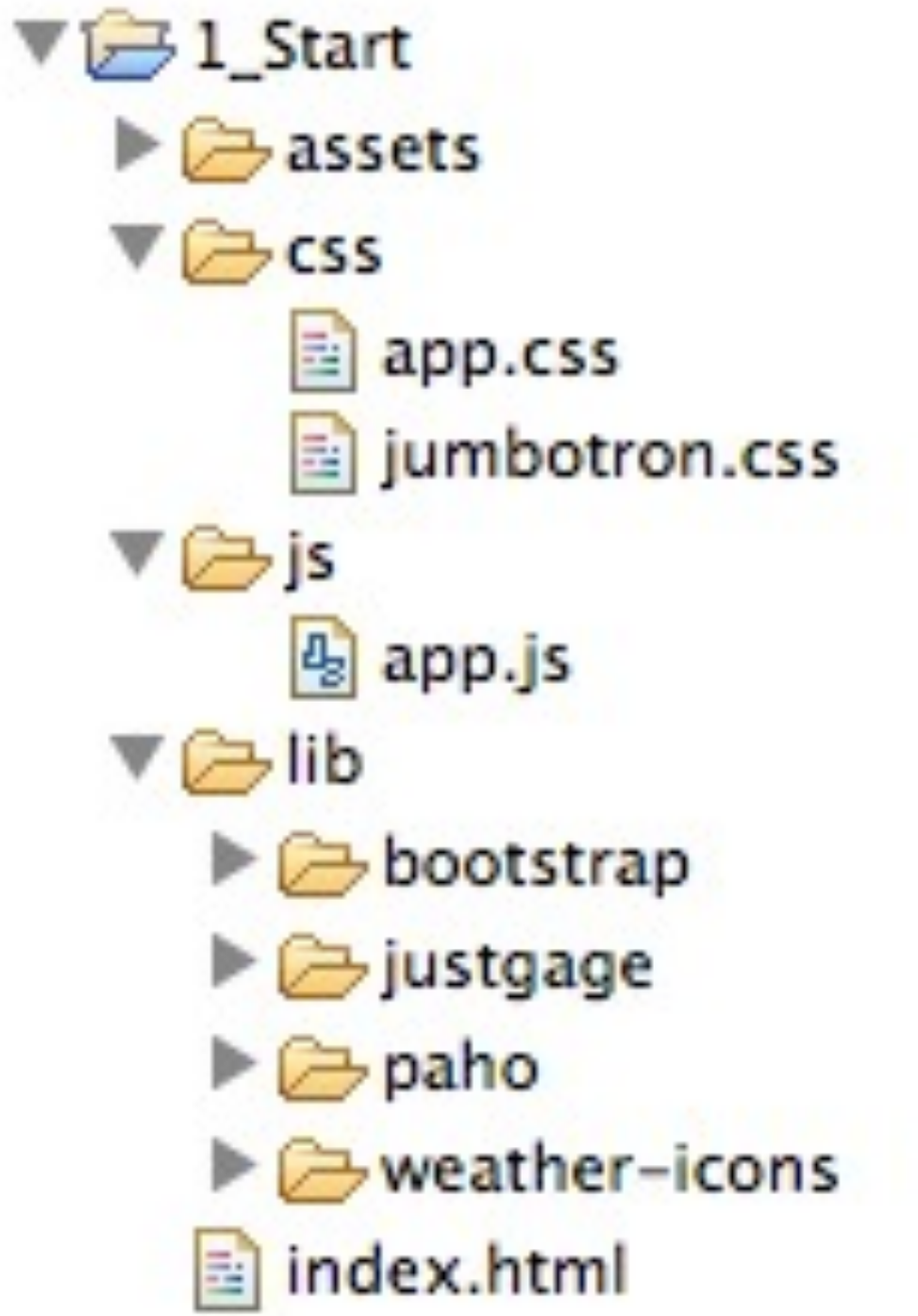
Reconfigure the simulator update interval

seconds | Update Interval

Technologies & Libraries

- HTML
- Javascript
- CSS
- JQuery
- Eclipse Paho.js
- Twitter Bootstrap
- JustGage
- WeatherIcons

Project Structure

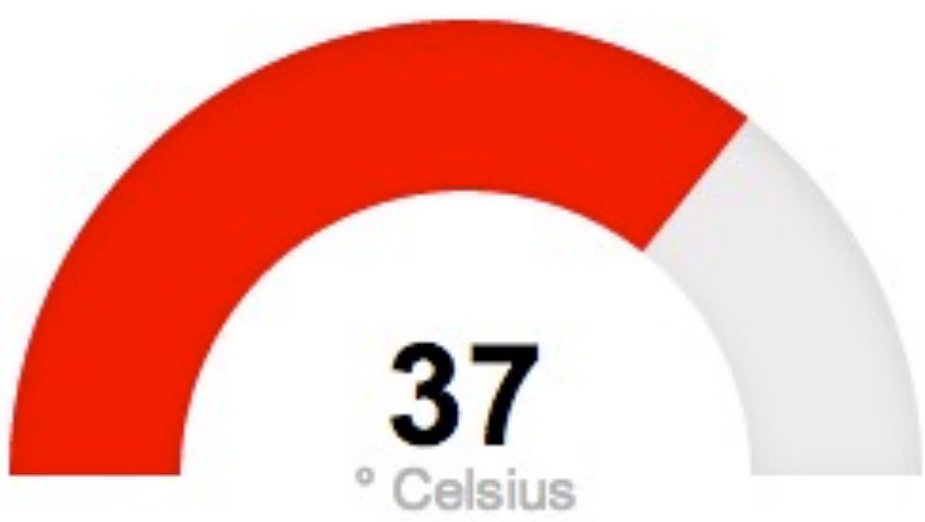


Part 1 - Do it yourself!

- Add a Gauge for the temperature
- Add a placeholder weather icon
- Add a control panel section
  - Client online/offline status display
  - Simulator interval reconfiguration form

Start at Eclipse Project "W1_Start"

Part I - Gauge



Part I - Gauge

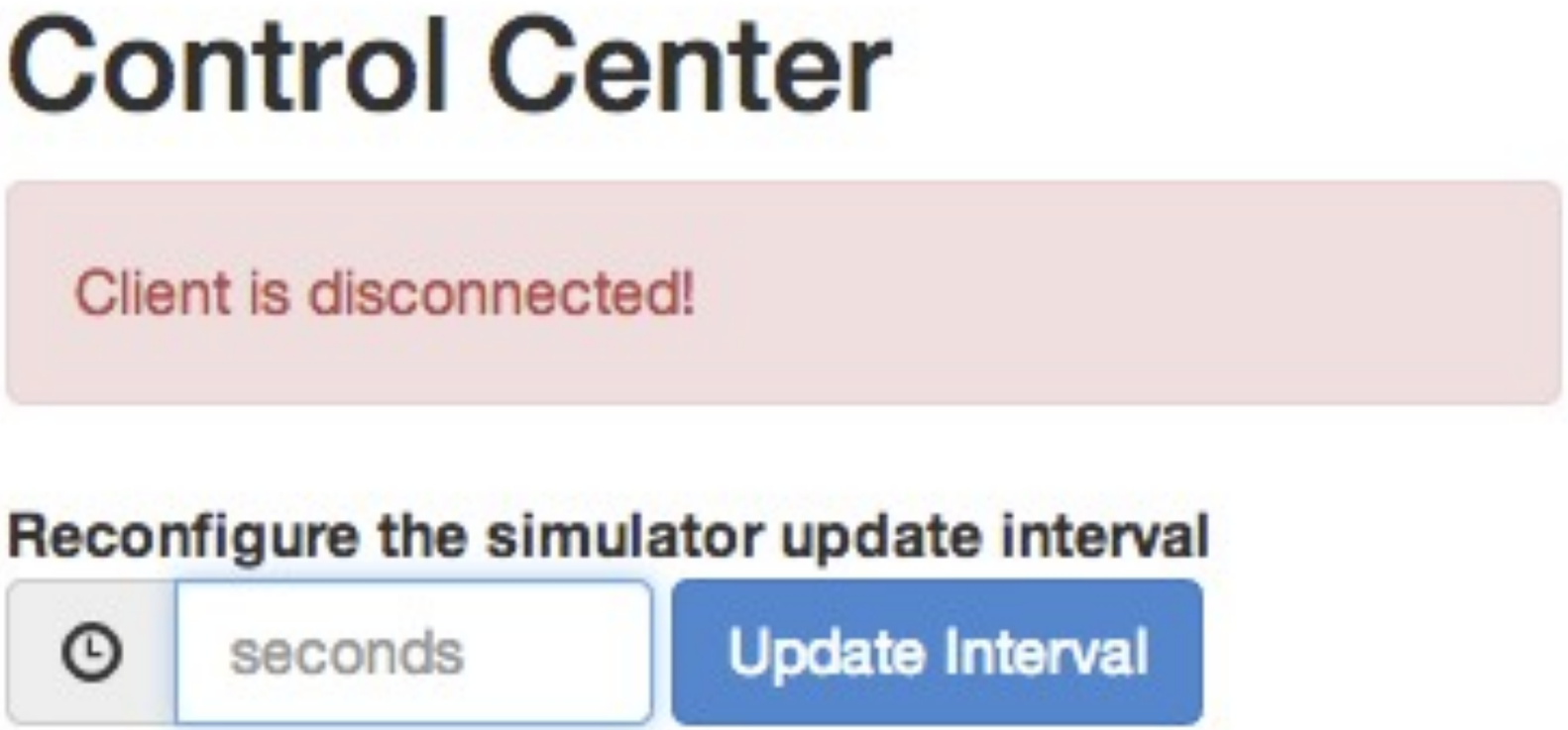
```
<div id="gauge"
class="gauge"></div>
```

Part 1 - Weather Icon

Part I - Weather Icon

```
<i id="wethericon"
class="wi-cloud-refresh
weather-icon"></i>
```

Part I - Control Center



Part I - Control Center

```
<div id="client_connected"
class="alert alert-success hide">
</div>
<div id="client_disconnected"
class="alert alert-danger hide">
</div>
```

Part I - Control Center II

```
<form class="form-inline" role="form">
      <div class="form-group">
            <div class="input-group" style="width: 150px">
                  <span class="input-group-addon">
                        <span class="glyphicon glyphicon-time"></span>
                  </span>
                  <input id="intervalinput" type="text"
                              class="form-control" placeholder="seconds">
            </div>
      </div>
      <div class="form-group">
         <button type="submit" class="btn btn-primary" onclick=
                        "publishInterval($('#intervalinput').val())">
            Update Interval
         </button>
      </div>
</form>
```

Part II - Do it yourself!

- Add the Paho.js library
- Implement a basic subscriber
  - *Connect to local broker*
  - *Subscribe to eclipsecon/status*
  - *Alert on message arrival topic + payload*
  - *Alert on connection failure*

Start at Eclipse Project "W2_Widgets"

Part II - Paho.js API I

Create a MQTT Client Object

```
var client = new Messaging.Client(
                    "hostname", 8000, “clientId");
```

Connect to the MQTT broker

```
client.connect(options);
```

Part II - Paho.js API II

Subscribe to a topic

```
client.subscribe("#",options);
```

Get message contents

```
client.onMessageArrived = function (message) {
        var topic = message.destinationName;
        var message = message.payloadString;
    };
```

Part III - Do it yourself!

- Subscribe to *eclipsecon/temperature*
- Write a function to refresh the gauge
- Refresh the gauge when a temperature message arrives

Start at Eclipse Project "W3_Using Paho"

Part III - Gauge Refresh

Refresh the Gauge

```
gauge.refresh(20.3);
```

Convert String to Float

```
parseFloat("20.3");
```

Part IV - Do it yourself!

- Subscribe to *eclipsecon/weather*
- Write a function to set the weather icon
- Possible Payloads: *STORM*, *SNOW*, *RAIN*, *CLOUD*, *SUN*

Start at Eclipse Project "W4_Temperature_Widget"

Part V - Do it yourself!

- Subscribe to *eclipsecon/status*
- Write a function to set connected / disconnected state in the UI
- Write a function to publish to *eclipsecon/interval*
- Call this publish function when clicking the "Update Interval" button.

Start at Eclipse Project "W5_Weather_Widget"

Part V - Paho.js API

Publish to a topic

```
var message = new Messaging.Message(interval);
    message.destinationName = "topic";
message.qos = 2;

client.send(message);
```

Part V - Show and Hide Elements

Show Element

```
$("#element_id").html('My text!').
removeClass("hide").hide().fadeIn("slow");
```

Hide Element

```
$("#element_id").addClass("hide").hide();
```

Part VI - Do it yourself!

- Subscribe to *eclipsecon/glaze*
- Add glaze alert to UI if there is a glaze warning
- Remove glaze alert from UI if there is no glaze alert

Start at Eclipse Project "W6_Control_Center_Widget"

Dashboard completed!

complete dashboard located in W7_Glaze_Warning

Outlook:

Real world applications

Limitations of the current implementation

- At this point there is no real data backend
- Only live data
- No historical data can be shown easily (e.g. charts)
- No authentication & authorization
- Secure Transport (TLS)

Potential Improvements

- Data persistence HiveMQ plugins
  - SQL
  - NoSQL
  - ESB
- Authentication and Authorization HiveMQ plugins
  - Integrate into existing applications & infrastructure

Q & A

Thanks!