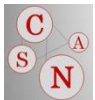




Διάλεξη

MQ Telemetry Transport (MQTT)

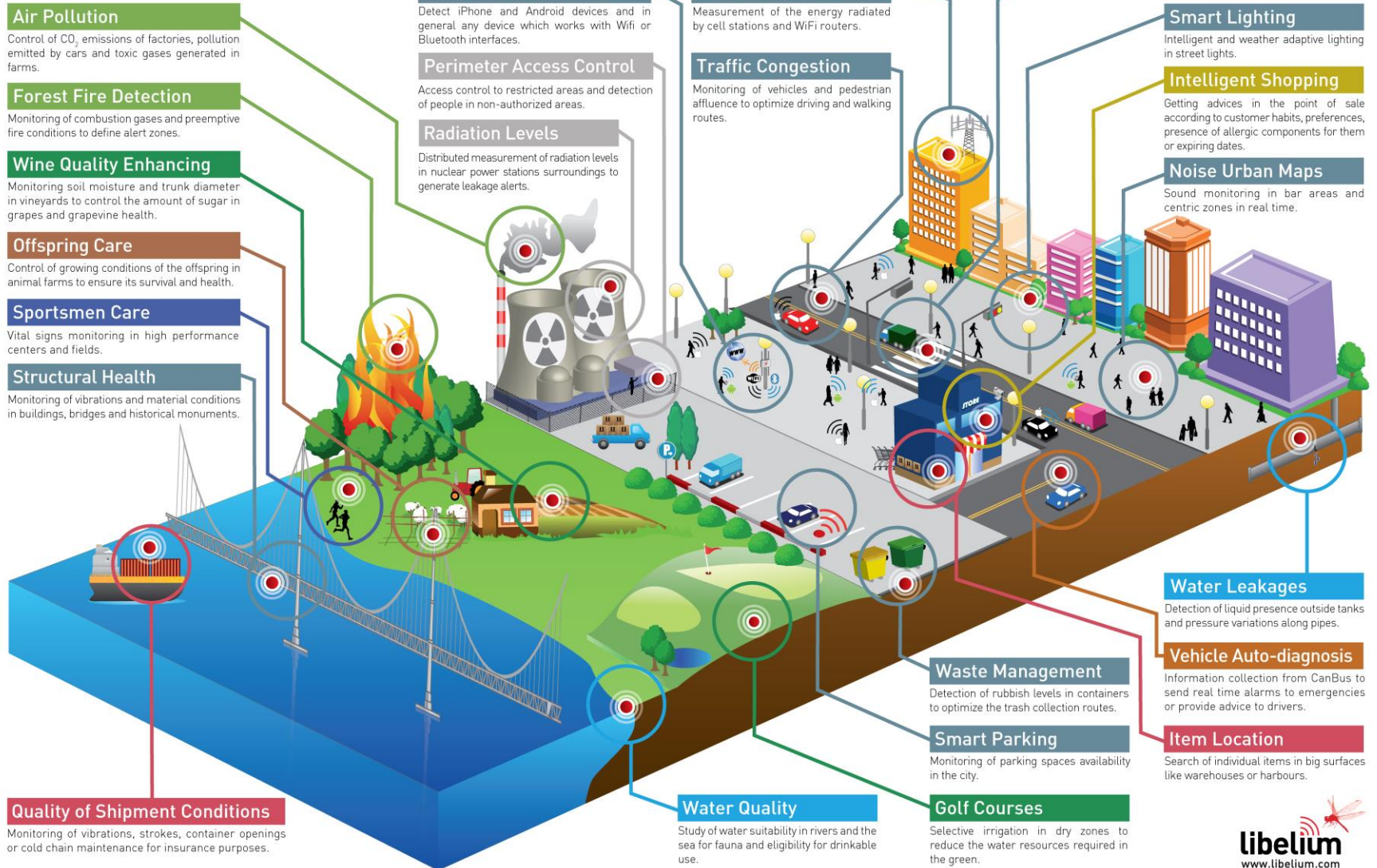
- Internet of Things
- Publish – Subscribe model
- MQTT



□ Internet of Things (IoT)

IoT applications

Libelium Smart World



IoT applications

□ Devices

- Programming Boards, Sensors, Actuators, **Smart Phones**

□ Programming languages

- **Java, Android**, Javascript, Python

□ Messaging

- **MQTT**, Constrained Application Protocol (CoAP)

□ Scaled scenarios

- Home automation, smart cities, industrial applications

Publish – subscribe model

Publish-subscribe

- Message pattern ή message queue paradigm ή message oriented middleware ή messaging protocol ή connectivity protocol
- Ασύγχρονη επικοινωνία
 - αποστολή δεδομένων σε πραγματικό χρόνο
- Μικρό μέγεθος μηνυμάτων
 - δεδομένα αισθητήρων
- Χαμηλή κατανάλωση μπαταρίας
 - smartphone, embedded boards

Publish-subscribe

□ Publish

- Δημοσίευση μηνυμάτων ενός συγκεκριμένου topic στον message broker

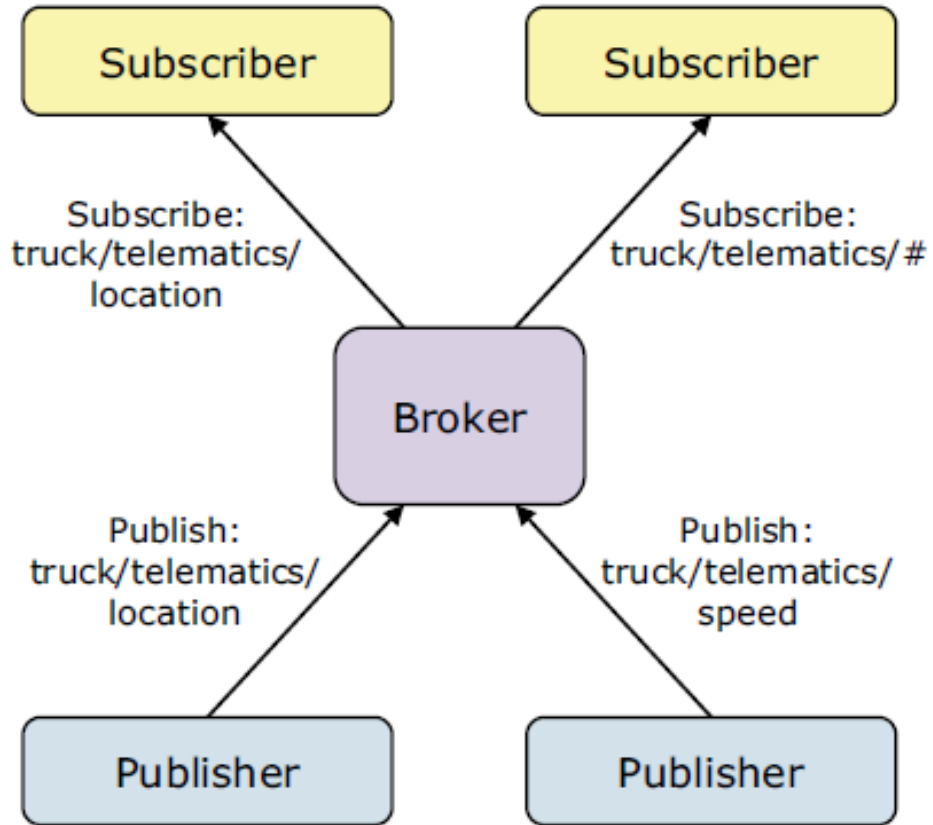
□ Subscribe

- Εγγραφή σε συγκεκριμένο topic στον message broker

□ Broker

- Πρόγραμμα διαμεσολαβητή για τη διαχείριση μηνυμάτων μεταξύ ετερογενών messaging πρωτοκόλλων.

- Επικαιροποίηση
- Μετασχηματισμός
- Δρομολόγηση



□ MQ Telemetry Transport

MQ Telemetry Transport (MQTT)

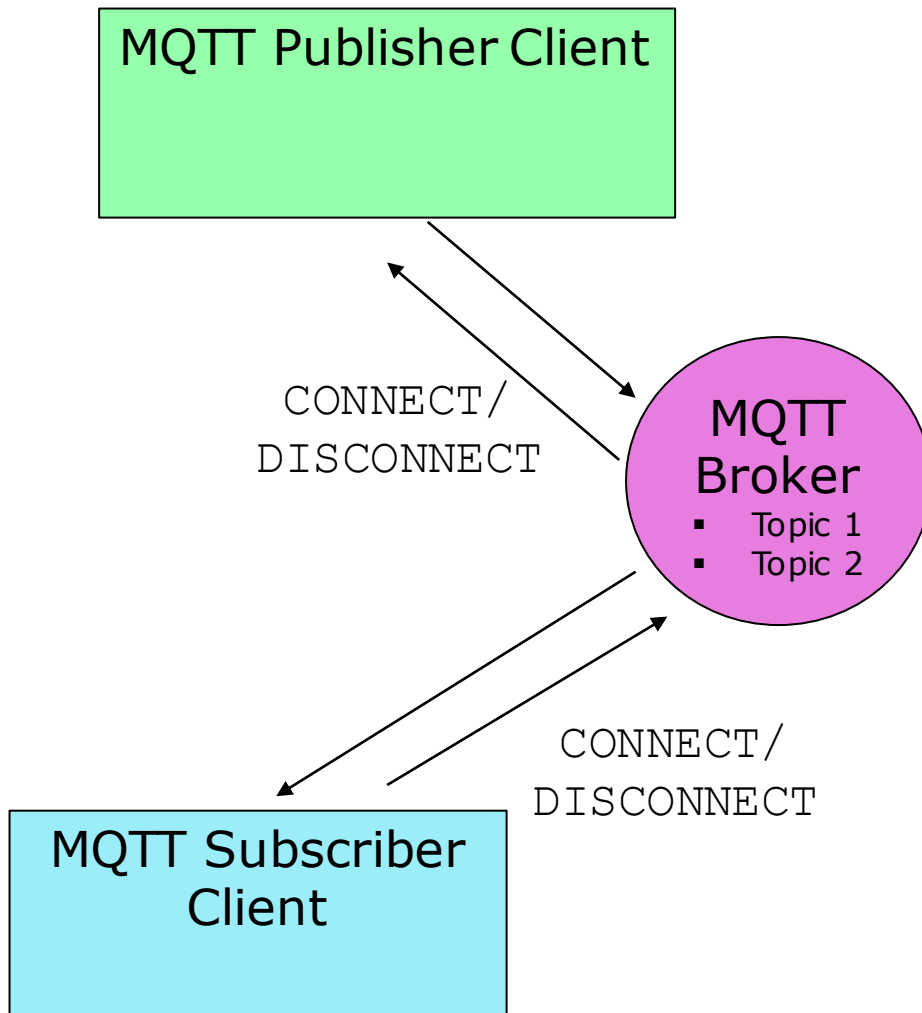
- ❑ Ξεκίνησε από τους Dr Andy Stanford-Clark της IBM και Arlen Nipper της Arcom (σημερινή Eurotech) το 1999
- ❑ Χρησιμοποιήθηκε από το Facebook Messenger το 2011
- ❑ Η έκδοση 3.1.1 έγινε δεκτή ως OASIS standard το 2014

MQ Telemetry Transport (MQTT)

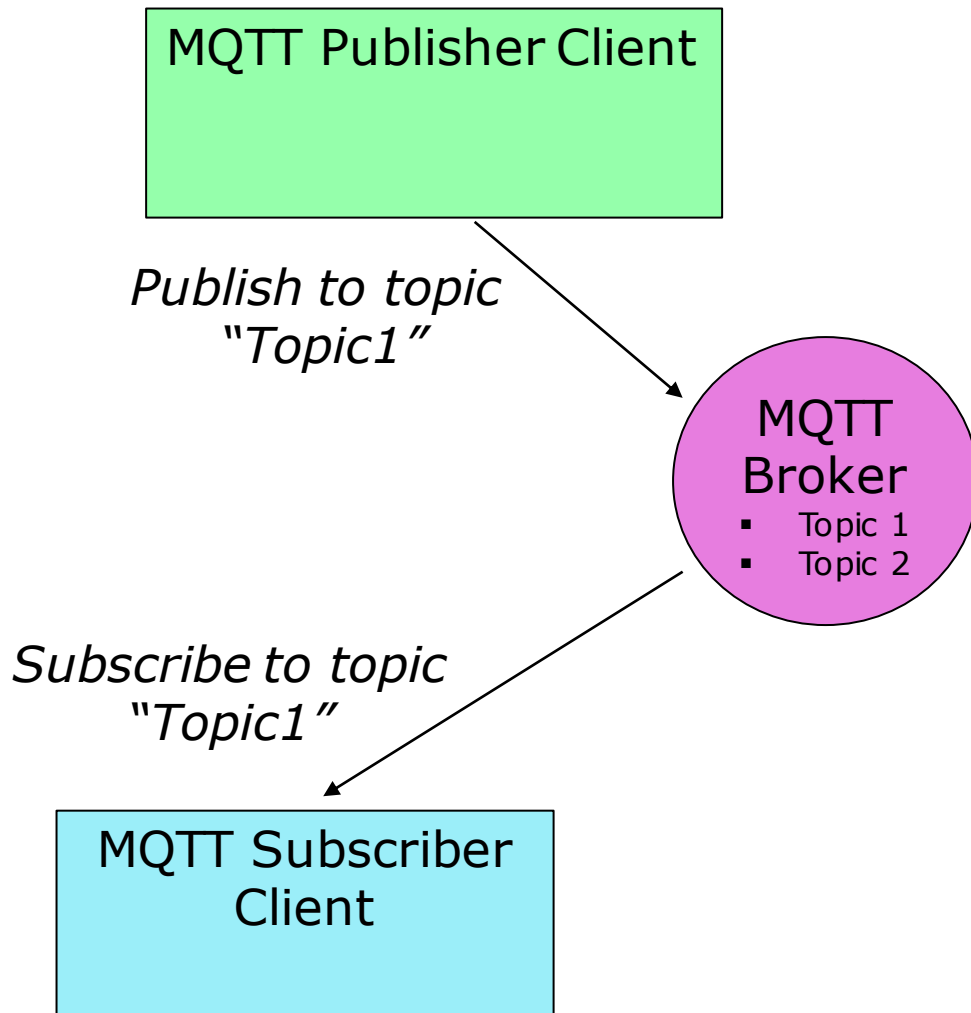
- MQTT αποτελεί ένα εξαιρετικά **απλό** και **ελαφρύ** πρωτόκολλο μεταφοράς μηνυμάτων (**messaging protocol**)
 - Το MQTT μήνυμα είναι όσο το δυνατόν μικρότερο.
 - Σταθερό header (2 bytes), η κατά απαίτηση push-style message διανομή των μηνυμάτων διατηρεί το network utilization χαμηλό.
- Η αρχιτεκτονική τύπου publish/subscribe έχει σχεδιαστεί για να είναι ανοικτή και εύκολη στην υλοποίηση
 - Χιλιάδες συσκευές χρήστες να συνδέονται σε έναν MQTT server (broker)
 - Δεν απαιτείται άδεια χρήσης από τις συσκευές/λειτουργικά συστήματα/πλατφόρμες κτλ
 - Οι εφαρμογές/συσκευές που στέλνουν δεδομένα δεν χρειάζεται να γνωρίζουν οτιδήποτε για τους λήπτες
- Είναι ιδανικό για περιορισμένα περιβάλλοντα:
 - Χαμηλόρυθμη σύνδεση, υψηλή καθυστέρηση, συσκευές με περιορισμένη επεξεργαστική ισχύ και μνήμη (μικρό μέγεθος βιβλιοθηκών)

MQ Telemetry Transport (MQTT)

- Πολλαπλά επίπεδα ποιότητας υπηρεσίας (Quality of Service 3) δίνουν ευελιξία στη διαχείριση μηνυμάτων διαφορετικού τύπου
 - *most once, at least once, exactly once.*
- Απλό σύνολο εντολών
 - CONNECT, PUBLISH, SUBSCRIBE, UNSUBSCRIBE DISCONNECT
- Ενσωματωμένη υποστήριξη για όταν υπάρχει απώλεια σύνδεσης
 - Ο server ενημερώνεται όταν η σύνδεση διακόπτεται
 - Τα μηνύματα ξαναστέλνονται ή κρατούνται για παράδοση αργότερα



- **Broker** - ο ενδιάμεσος εξυπηρετητής (server) που διανέμει την πληροφορία στους πελάτες (client) που είναι συνδεδεμένοι σε αυτόν και ενδιαφερόμενοι για ένα θέμα (topic).
- **Client** - η συσκευή τύπου πελάτη που είναι συνδεδεμένη στον broker προκειμένου να αποστείλει ή λάβει πληροφορία.
- **Topic** - Το θέμα το οποίο ενδιαφέρει τους clients. Οι clients κάνουν publish, subscribe, ή και τα δύο, σε ένα topic.



- **Publisher** - Οι Clients που στέλνουν πληροφορία στον Broker προκειμένου να την διανείμει στους ενδιαφερόμενους για το topic clients.
 - **Subscriber** - Οι clients ενημερώνουν το broker για ποιο/α θέμα/τα ενδιαφέρονται. Όταν ένας client εγγράφεται σε ένα topic, όλα τα μηνύματα που στέλνονται στον broker στέλνονται στους subscribers που είναι εγγεγραμμένοι στο topic. Οι Clients μπορούν να κάνουν **unsubscribe** για να σταματήσουν να λαμβάνουν μηνύματα για το συγκεκριμένο topic.
-

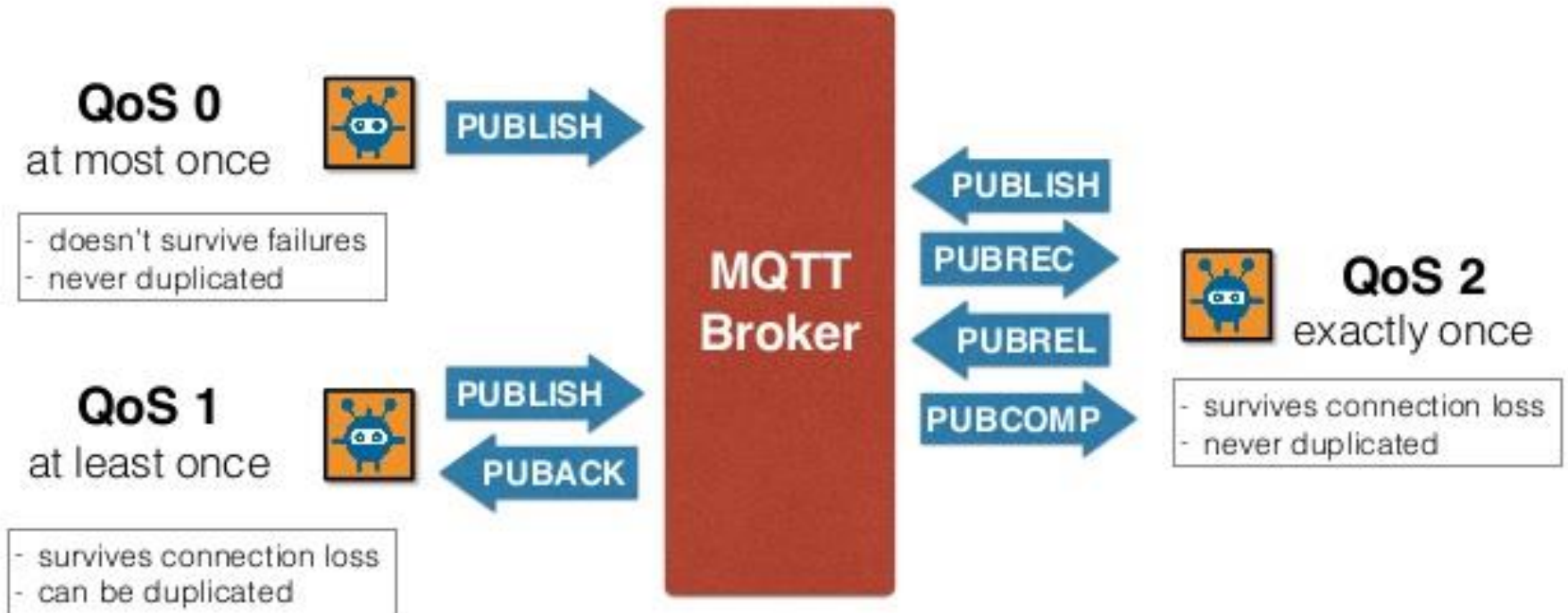
QoS - Ποιότητα υπηρεσίας

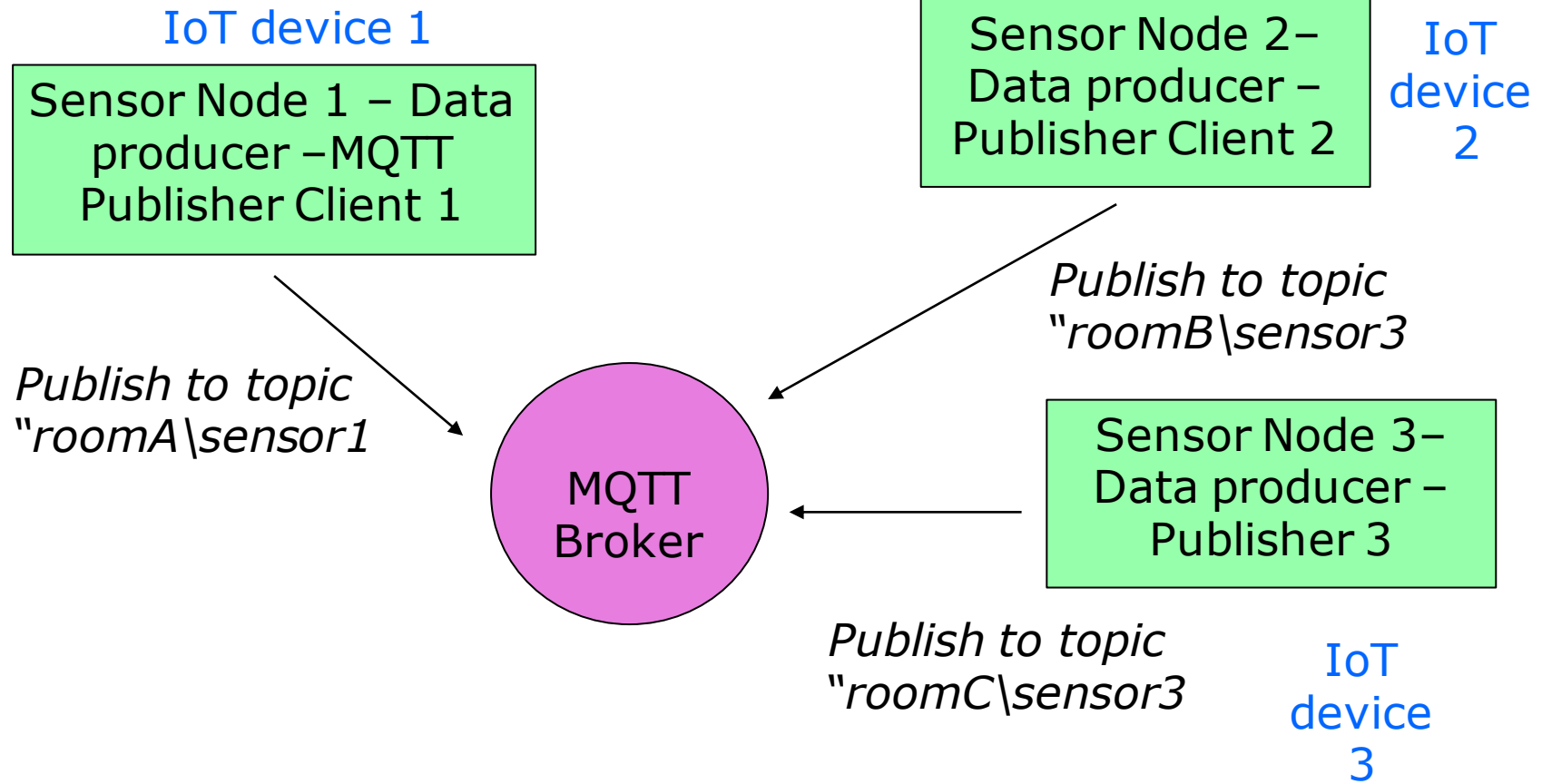
- Κάθε MQTT σύνδεση μπορεί να καθορίσει την ποιότητα υπηρεσίας στον broker με μια ακέραια τιμή που κυμαίνεται από 0-2. Το QoS δεν επηρεάζει τον χειρισμό των μεταδόσεων δεδομένων TCP, μόνο μεταξύ των πελατών MQTT.
 - Η τιμή 0 καθορίζει αποστολή το πολύ μία φορά ή μία και μόνο μία φορά χωρίς να απαιτείται επιβεβαίωση παράδοσης. Αυτό αναφέρεται συχνά ως «πυροβολήστε και ξεχάστε».
 - Η τιμή 1 καθορίζει τουλάχιστον μία φορά. Το μήνυμα αποστέλλεται πολλές φορές μέχρι να ληφθεί μια επιβεβαίωση, γνωστή αλλιώς ως επιβεβαιωμένη παράδοση. Μπορεί να υπάρξουν πολλαπλά αντίγραφα.
 - Η τιμή 2 καθορίζει ακριβώς μία φορά. Οι πελάτες αποστολέα και παραλήπτες χρησιμοποιούν χειραψία δύο επιπέδων για να εξασφαλίσουν ότι λαμβάνεται μόνο ένα αντίγραφο του μηνύματος, γνωστό ως εξασφαλισμένη παράδοση.
-

QoS

MQTT

Quality of Service for **reliable messaging**



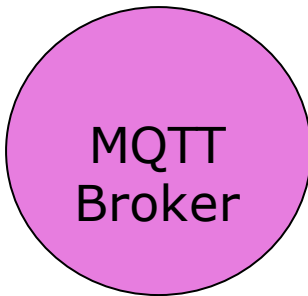


Sensor Node 1 – Data producer –MQTT Publisher Client 1

Sensor Node 2– Data producer – Publisher Client 2

Publish to topic "roomA\sensor1"

Publish to topic "roomB\sensor2"



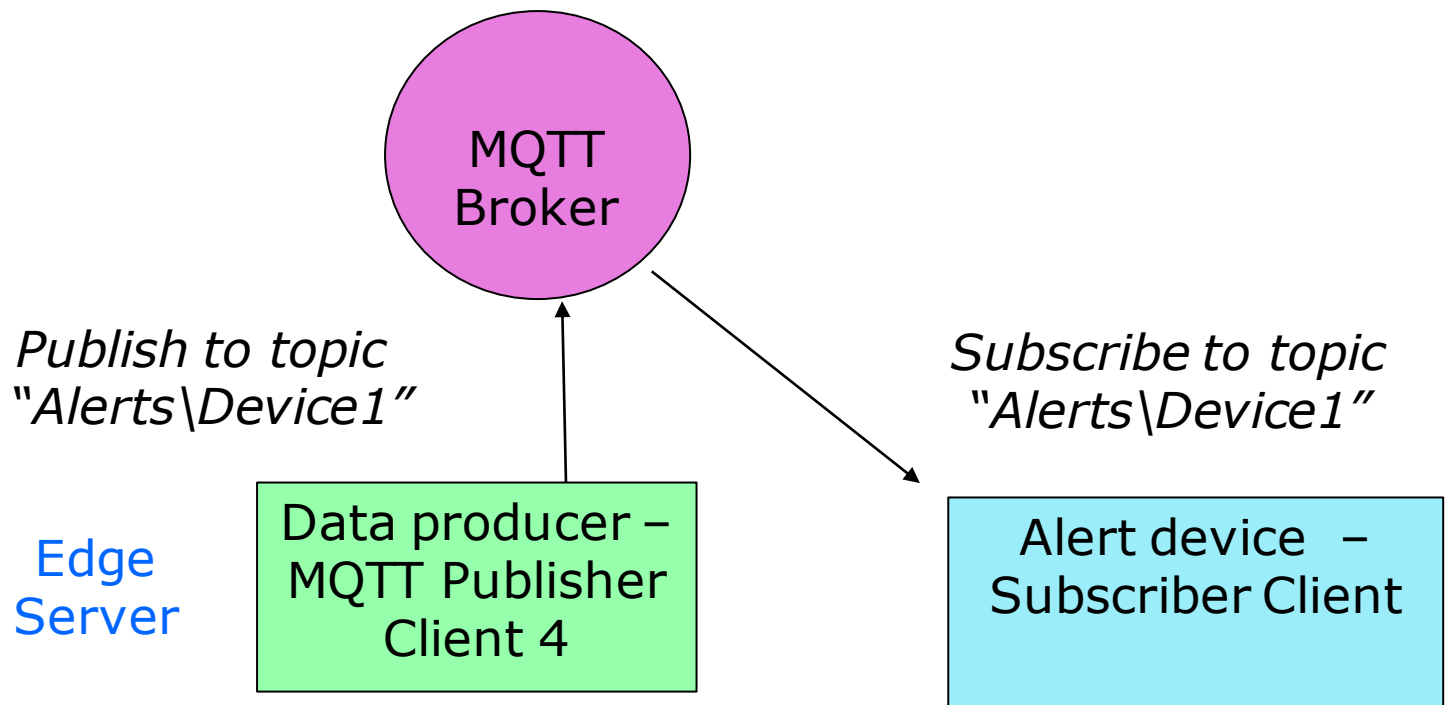
Sensor Node 3– Data producer – Publisher 3

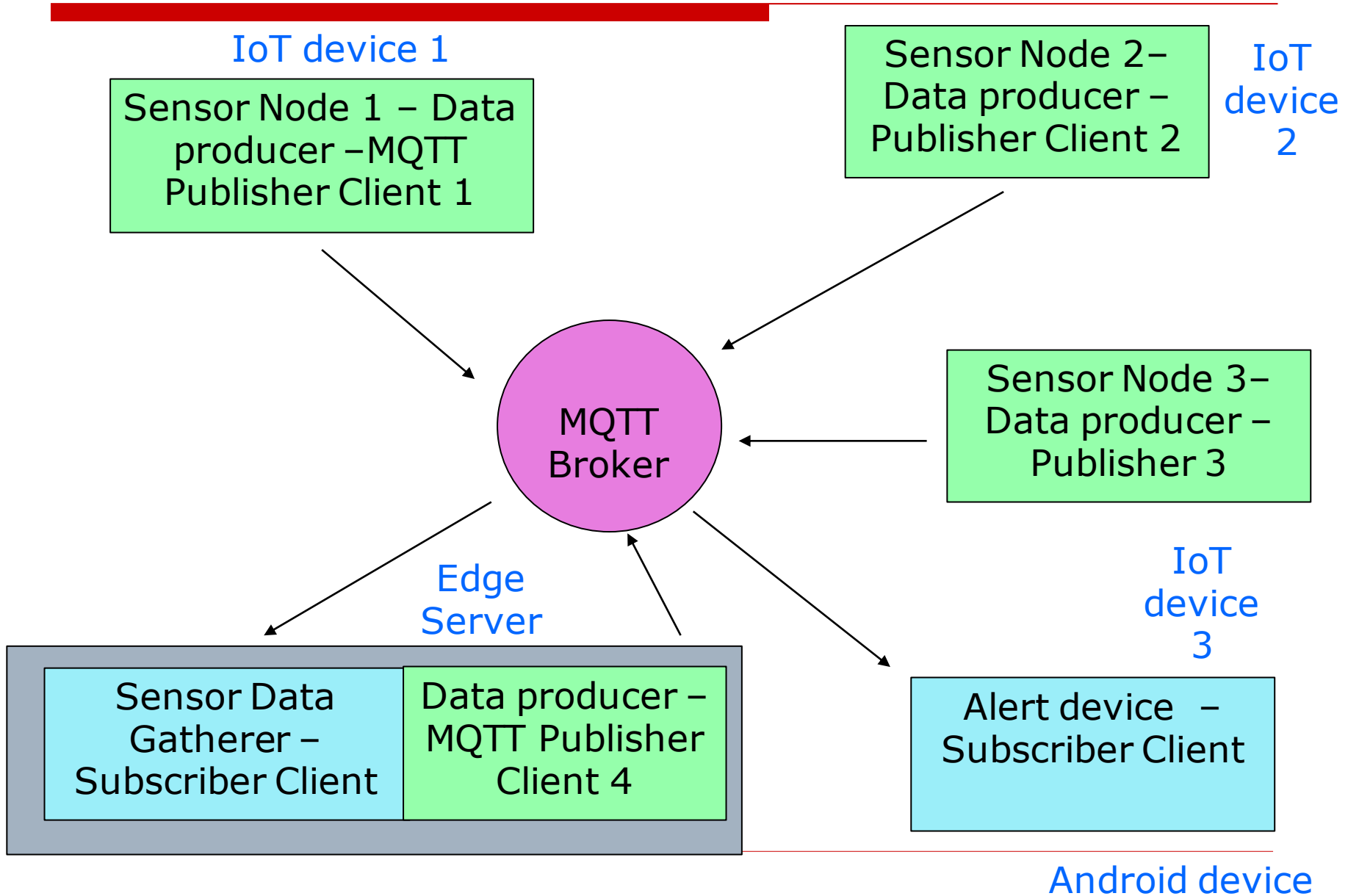
Publish to topic "roomC\sensor3"

Subscribe to topics "roomA\sensor1", "roomB\sensor2", "roomC\sensor3"

Sensor Data Gatherer – Subscriber Client

Edge Server





Εργαλεία

□ **IoT – Android εφαρμογή**

- Android SDK
- Java
- Eclipse Paho Android Client (Publisher)

□ **Android – Android εφαρμογή**

- Android SDK
 - Java
 - Eclipse Paho Android Client (Subscriber)
-

Εργαλεία

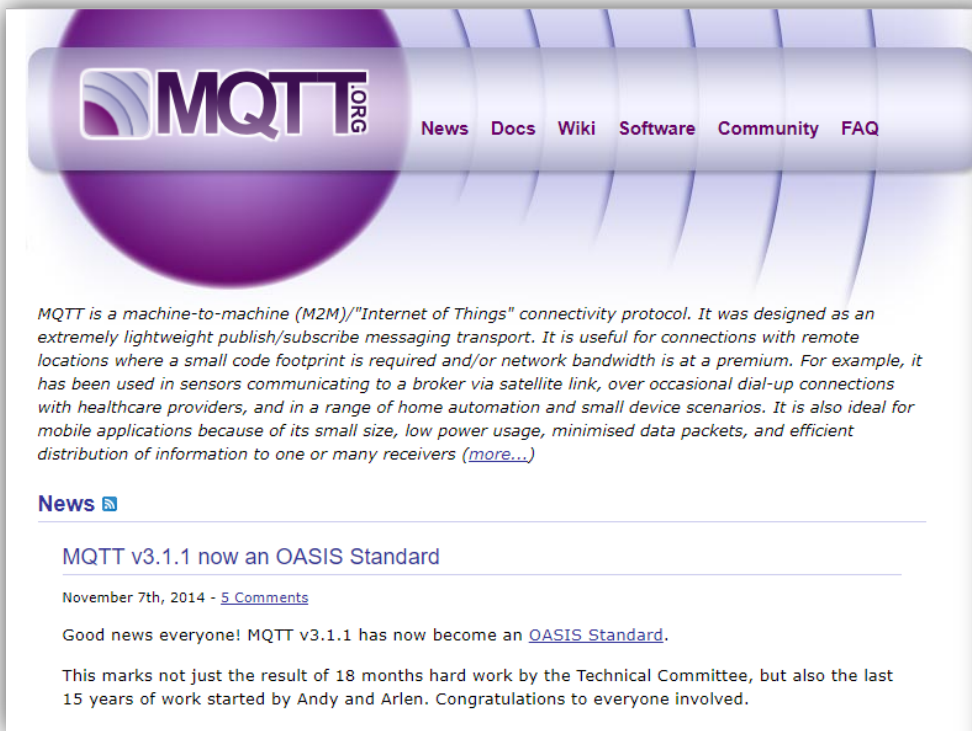
□ **Broker**

- Mosquitto broker

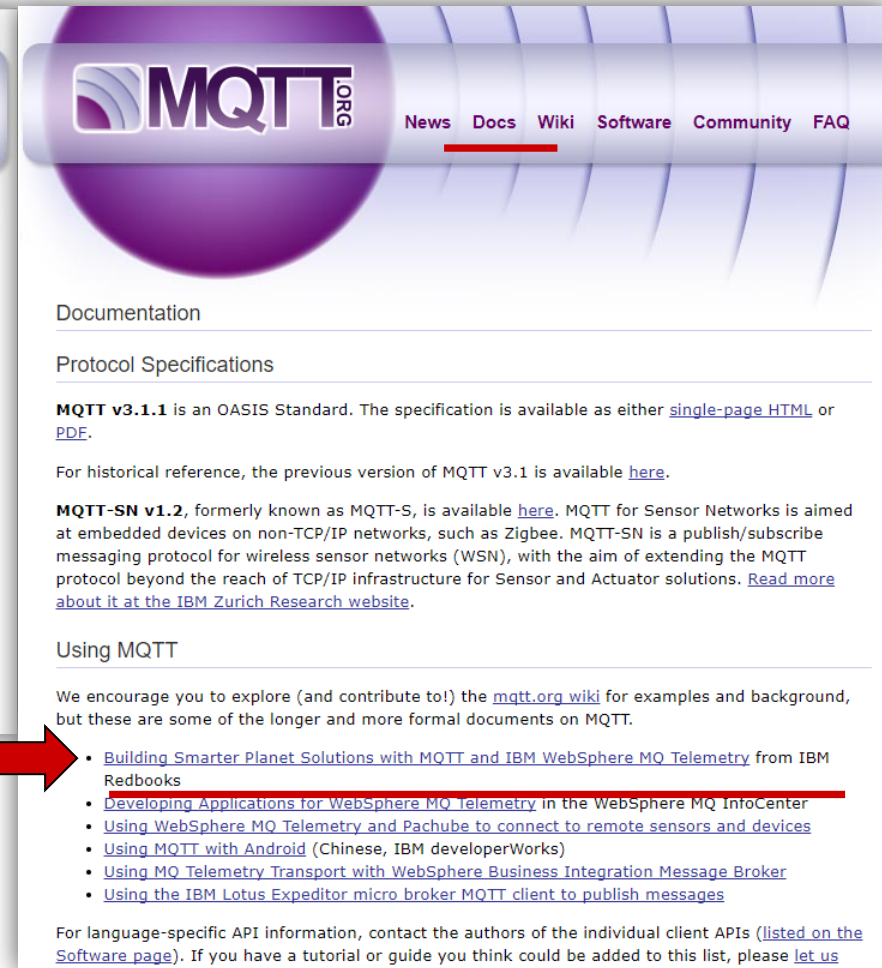
□ **Edge server**

- Java IDE
 - Java
 - Eclipse Paho Java client
-

<http://mqtt.org/>



The screenshot shows the top navigation bar of the MQTT.ORG website. The logo is on the left, followed by links for News, Docs, Wiki, Software, Community, and FAQ. Below the navigation bar is a paragraph of introductory text about MQTT. Further down, there is a 'News' section with a sub-header 'MQTT v3.1.1 now an OASIS Standard' and a date 'November 7th, 2014 - 5 Comments'. The main text of the news item states: 'Good news everyone! MQTT v3.1.1 has now become an OASIS Standard. This marks not just the result of 18 months hard work by the Technical Committee, but also the last 15 years of work started by Andy and Arlen. Congratulations to everyone involved.'



The screenshot shows the 'Documentation' and 'Using MQTT' sections of the MQTT.ORG website. The 'Documentation' section includes 'Protocol Specifications' and a paragraph about MQTT v3.1.1 being an OASIS Standard. The 'Using MQTT' section includes a paragraph encouraging exploration of the MQTT.ORG wiki and a list of links to various documents and guides. A red arrow points from the news section of the left screenshot to the 'Using MQTT' section of this screenshot.

Documentation

Protocol Specifications

MQTT v3.1.1 is an OASIS Standard. The specification is available as either [single-page HTML](#) or [PDF](#).

For historical reference, the previous version of MQTT v3.1 is available [here](#).

MQTT-SN v1.2, formerly known as MQTT-S, is available [here](#). MQTT for Sensor Networks is aimed at embedded devices on non-TCP/IP networks, such as Zigbee. MQTT-SN is a publish/subscribe messaging protocol for wireless sensor networks (WSN), with the aim of extending the MQTT protocol beyond the reach of TCP/IP infrastructure for Sensor and Actuator solutions. [Read more about it at the IBM Zurich Research website](#).

Using MQTT

We encourage you to explore (and contribute to!) the [mqtt.org/wiki](#) for examples and background, but these are some of the longer and more formal documents on MQTT.

- [Building Smarter Planet Solutions with MQTT and IBM WebSphere MQ Telemetry](#) from IBM Redbooks
- [Developing Applications for WebSphere MQ Telemetry](#) in the WebSphere MQ InfoCenter
- [Using WebSphere MQ Telemetry and Pachube to connect to remote sensors and devices](#)
- [Using MQTT with Android](#) (Chinese, IBM developerWorks)
- [Using MQ Telemetry Transport with WebSphere Business Integration Message Broker](#)
- [Using the IBM Lotus Expeditor micro broker MQTT client to publish messages](#)

For language-specific API information, contact the authors of the individual client APIs ([listed on the Software page](#)). If you have a tutorial or guide you think could be added to this list, please [let us](#)

1ο & 2ο κεφάλαια

Mosquitto

MQTT – Mosquitto (mosquitto.org)

- ❑ Το Mosquitto Broker εκτελείται (αρχικές ρυθμίσεις) με την παρακάτω εντολή:

```
{path}/{to}/mosquitto -c {path}/{to}/mosquitto.conf
```

- ❑ Mosquitto Publisher
- ❑ Mosquitto Subscriber
- ❑ Χρήσιμοι σύνδεσμοι

- <https://mosquitto.org/download/>
- <https://mosquitto.org/documentation/>
- <https://sivatechworld.wordpress.com/2015/06/11/step-by-step-installing-and-configuring-mosquitto-with-windows-7/>
- <http://www.steves-internet-guide.com/install-mosquitto-broker/>
- <http://test.mosquitto.org/>

Mosquitto Test @ windows

- Στο 1^ο παράθυρο:

- `C:\Program Files\mosquitto>mosquitto -p 1883`

```
C:\Program Files\mosquitto>mosquitto -p 1883
```

- Σε 2^ο παράθυρο

- `Netstat -an -p tcp`

```
TCP        127.0.0.1:1883          0.0.0.0:0              LISTENING
```

- Στο ίδιο παράθυρο

- `mosquitto_sub -t rooms/room1/sensors/temp`

- Σε 3^ο παράθυρο:

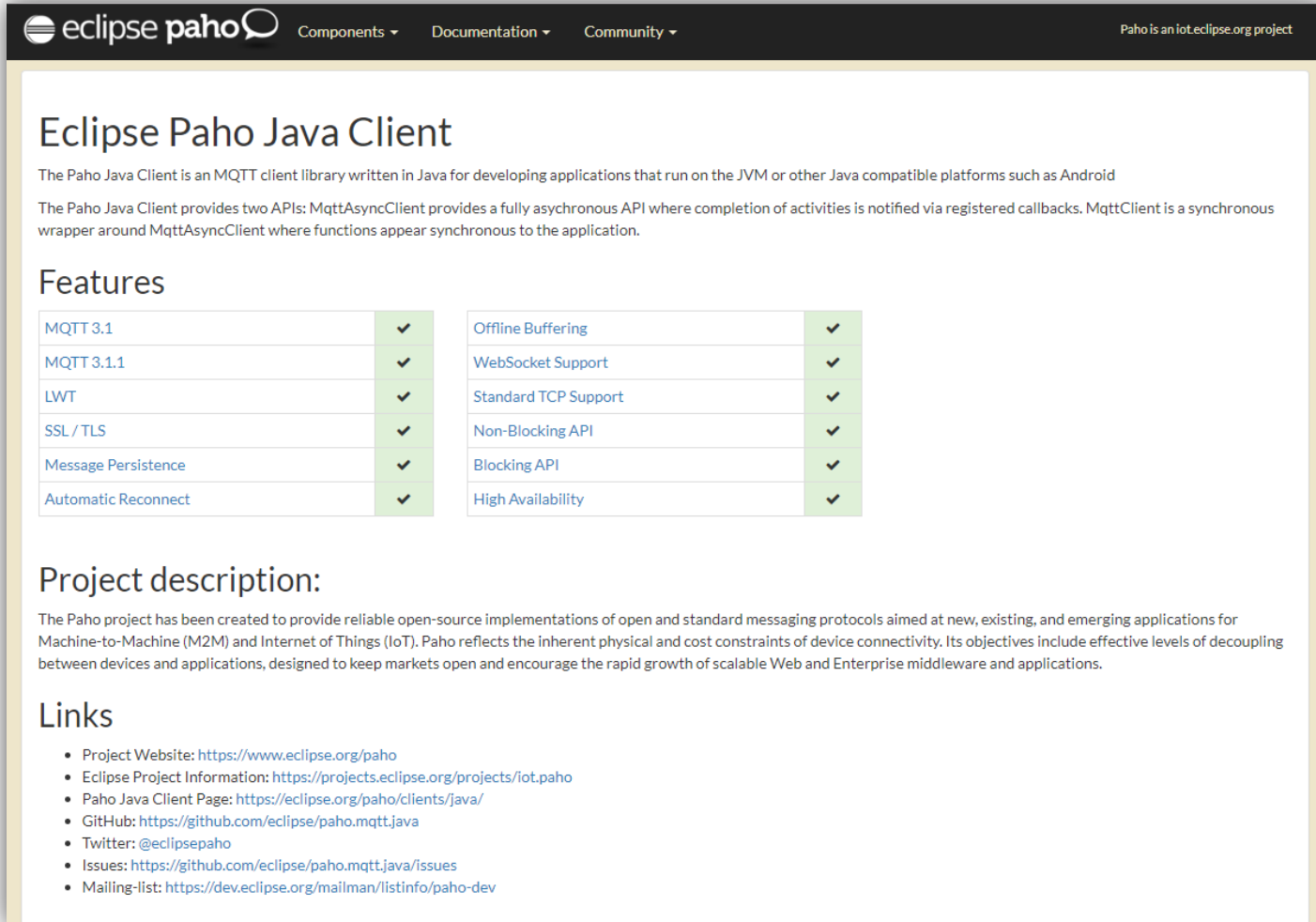
- `C:\Program Files\mosquitto>mosquitto_pub -t rooms/room1/sensors/temp -m "1 2 3 4"`

- Στο 2^ο εμφανίζεται 1 2 3 4

```
1 2 3 4
```

BIBΛΙΟΘΗΚΕΣ ΓΙΑ JAVA & ANDROID

Eclipse Paho Java client



The screenshot shows the Eclipse Paho Java Client project page. The header includes the Eclipse Paho logo, navigation links for Components, Documentation, and Community, and the text "Paho is an [iot.eclipse.org](https://www.eclipse.org/iot) project". The main content area features the title "Eclipse Paho Java Client", a brief description of the MQTT client library, and a list of features. The features are presented in two columns, each with a green checkmark indicating support. Below the features is a "Project description" section and a "Links" section with a list of project-related URLs.

Eclipse Paho Java Client

The Paho Java Client is an MQTT client library written in Java for developing applications that run on the JVM or other Java compatible platforms such as Android

The Paho Java Client provides two APIs: `MqttAsyncClient` provides a fully asynchronous API where completion of activities is notified via registered callbacks. `MqttClient` is a synchronous wrapper around `MqttAsyncClient` where functions appear synchronous to the application.

Features

MQTT 3.1	✓	Offline Buffering	✓
MQTT 3.1.1	✓	WebSocket Support	✓
LWT	✓	Standard TCP Support	✓
SSL / TLS	✓	Non-Blocking API	✓
Message Persistence	✓	Blocking API	✓
Automatic Reconnect	✓	High Availability	✓

Project description:

The Paho project has been created to provide reliable open-source implementations of open and standard messaging protocols aimed at new, existing, and emerging applications for Machine-to-Machine (M2M) and Internet of Things (IoT). Paho reflects the inherent physical and cost constraints of device connectivity. Its objectives include effective levels of decoupling between devices and applications, designed to keep markets open and encourage the rapid growth of scalable Web and Enterprise middleware and applications.

Links

- Project Website: <https://www.eclipse.org/paho>
- Eclipse Project Information: <https://projects.eclipse.org/projects/iot.paho>
- Paho Java Client Page: <https://eclipse.org/paho/clients/java/>
- GitHub: <https://github.com/eclipse/paho.mqtt.java>
- Twitter: @eclipsepaho
- Issues: <https://github.com/eclipse/paho.mqtt.java/issues>
- Mailing-list: <https://dev.eclipse.org/mailman/listinfo/paho-dev>

□ <https://eclipse.org/paho/clients/java/>

Eclipse Paho Java client



Components ▾

Documentation ▾

Community ▾

Paho is an [eclipse.org](https://www.eclipse.org) project

Using the Paho Java Client

Downloading

Eclipse hosts a Nexus repository for those who want to use Maven to manage their dependencies. The released libraries are also available in the Maven Central repository.

Add the repository definition and the dependency definition shown below to your pom.xml.

Replace `%REPOURL%` with either <https://repo.eclipse.org/content/repositories/paho-releases/> for the official releases, or <https://repo.eclipse.org/content/repositories/paho-snapshots/> for the nightly snapshots. Replace `%VERSION%` with the level required. The latest release version is `1.0.2` and the current snapshot version is `1.0.3`.

```
<project ...>
<repositories>
  <repository>
    <id>Eclipse Paho Repo</id>
    <url>%REPOURL%</url>
  </repository>
</repositories>
...
<dependencies>
  <dependency>
    <groupId>org.eclipse.paho</groupId>
    <artifactId>org.eclipse.paho.client.mqttv3</artifactId>
    <version>%VERSION%</version>
  </dependency>
</dependencies>
</project>
```

If you find that there is functionality missing or bugs in the release version, you may want to try using the snapshot version to see if this helps before raising a feature request or an issue.

Building from source

There are two active branches on the Paho Java git repository, `master` which is used to produce stable releases, and `develop` where active development is carried out. By default cloning the git repository will download the `master` branch, to build from `develop` make sure you switch to the remote branch: `git checkout -b develop remotes/origin/develop`

To then build the library run the following maven command: `mvn package -DskipTests`

This will build the client library without running the tests. The jars for the library, source and javadoc can be found in the `org.eclipse.paho.client.mqttv3/target` directory.

Documentation

Reference documentation is online at: <http://www.eclipse.org/paho/files/javadoc/index.html>

Log and Debug in the Java Client: <https://wiki.eclipse.org/Paho/LogandDebugintheJavaclient>

```
import org.eclipse.paho.client.mqttv3.MqttConnectOptions;
import org.eclipse.paho.client.mqttv3.MqttException;
import org.eclipse.paho.client.mqttv3.MqttMessage;
import org.eclipse.paho.client.mqttv3.persist.MemoryPersistence;
```

```
public class MqttPublishSample {
    public static void main(String[] args) {
        String topic = "MQTT Examples";
        String content = "Message from MqttPublishSample";
        int qos = 2;
        String broker = "tcp://iot.eclipse.org:1883";
        String clientId = "JavaSample";
        MemoryPersistence persistence = new MemoryPersistence();

        try {
            MqttClient sampleClient = new MqttClient(broker, clientId, persistence);
            MqttConnectOptions connOpts = new MqttConnectOptions();
            connOpts.setCleanSession(true);
            System.out.println("Connecting to broker: "+broker);
            sampleClient.connect(connOpts);
            System.out.println("Connected");

            System.out.println("Publishing message: "+content);
            MqttMessage message = new MqttMessage(content.getBytes());
            message.setQos(qos);
            sampleClient.publish(topic, message);
            System.out.println("Message published");

            sampleClient.disconnect();
            System.out.println("Disconnected");
            System.exit(0);
        } catch (MqttException me) {
            System.out.println("reason "+me.getReasonCode());
            System.out.println("msg "+me.getMessage());
            System.out.println("loc "+me.getLocalizedMessage());
            System.out.println("cause "+me.getCause());
            System.out.println("excep "+me);
            me.printStackTrace();
        }
    }
}
```

JAVA Publisher

JAVA subscriber

```
package com.anap.second;

import com.sun.jmx.snmp.Timestamp;
import org.eclipse.paho.client.mqttv3.*;
import org.eclipse.paho.client.mqttv3.persist.MemoryPersistence;

public class Main implements MqttCallback{

    public static void main(String[] args) {

        String topic    = "MQTT Example";
        int qos         = 2;
        String broker   = "tcp://localhost:1883";
        String clientId = "JavaSampleSubscriber";
        MemoryPersistence persistence = new MemoryPersistence();

        try {

            //Connect client to MQTT Broker
            MqttClient sampleClient = new MqttClient(broker, clientId, persistence);
            MqttConnectOptions connOpts = new MqttConnectOptions();
            connOpts.setCleanSession(true);

            //Set callback
            Main main = new Main();
            sampleClient.setCallback(main);

            System.out.println("Connecting to broker: "+broker);
            sampleClient.connect(connOpts);
            System.out.println("Connected");

            //Subscribe to a topic
            System.out.println("Subscribing to topic \""+topic+"\" qos "+qos);
            sampleClient.subscribe(topic, qos);

        } catch(MqttException me) {

            System.out.println("reason " + me.getReasonCode());
            System.out.println("msg " + me.getMessage());
            System.out.println("loc " + me.getLocalizedMessage());
            System.out.println("cause " + me.getCause());
            System.out.println("excep " + me);
            me.printStackTrace();

        }

    }

}
```

Subscriber in JAVA

```
/**
 * @see MqttCallback#connectionLost(Throwable)
 */
public void connectionLost(Throwable cause) {
    // This method is called when the connection to the server is lost.
    System.out.println("Connection lost!" + cause);
    System.exit(1);
}

/**
 * @see MqttCallback#deliveryComplete(IMqttDeliveryToken)
 */
public void deliveryComplete(IMqttDeliveryToken token) {
    //Called when delivery for a message has been completed, and all acknowledgments have been received
}

/**
 * @see MqttCallback#messageArrived(String, MqttMessage)
 */
public void messageArrived(String topic, MqttMessage message) throws MqttException {
    //This method is called when a message arrives from the server.

    String time = new Timestamp(System.currentTimeMillis()).toString();
    System.out.println("Time:\t" +time +
        " Topic:\t" + topic +
        " Message:\t" + new String(message.getPayload()) +
        " QoS:\t" + message.getQos());
}
}
```

Eclipse Paho Android Client

The screenshot shows the Eclipse Paho Android Service project page. The header includes the Eclipse Paho logo and navigation links for Components, Documentation, and Community. The main heading is "Eclipse Paho Android Service". Below it, a paragraph describes the service as an MQTT client library for Android. A second paragraph provides instructions on how to get started, mentioning Android Studio and the Android SDK. The "Features" section contains two tables listing various capabilities with checkmarks or an 'x' indicating status. The "Project description" section explains the project's goals for M2M and IoT. The "Links" section provides a list of URLs for the project website, information, client page, GitHub, Twitter, issues, and mailing list.

Eclipse Paho Components ▾ Documentation ▾ Community ▾ Paho is an eclipse.org project

Eclipse Paho Android Service

The Paho Android Service is an MQTT client library written in Java for developing applications on Android.

To get started, download [Android Studio](#). You will also need to download the [Android SDK](#). Currently you will need the SDK for 19,21 and 22, This will hopefully be simplified soon.

Features

MQTT 3.1	✓	Offline Buffering	✓
MQTT 3.1.1	✓	WebSocket Support	✓
LWT	✓	Standard TCP Support	✓
SSL / TLS	✓	Non-Blocking API	✓
Message Persistence	✓	Blocking API	✗
Automatic Reconnect	✓	High Availability	✓

Project description

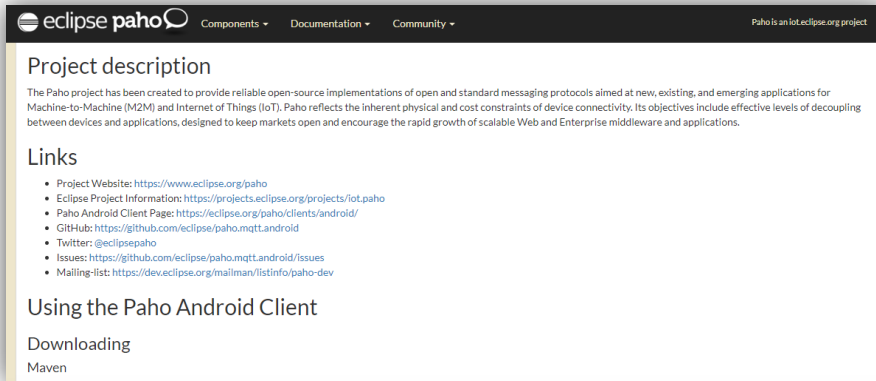
The Paho project has been created to provide reliable open-source implementations of open and standard messaging protocols aimed at new, existing, and emerging applications for Machine-to-Machine (M2M) and Internet of Things (IoT). Paho reflects the inherent physical and cost constraints of device connectivity. Its objectives include effective levels of decoupling between devices and applications, designed to keep markets open and encourage the rapid growth of scalable Web and Enterprise middleware and applications.

Links

- Project Website: <https://www.eclipse.org/paho>
- Eclipse Project Information: <https://projects.eclipse.org/projects/iot.paho>
- Paho Android Client Page: <https://eclipse.org/paho/clients/android/>
- GitHub: <https://github.com/eclipse/paho.mqtt.android>
- Twitter: [@eclipsepaho](https://twitter.com/eclipsepaho)
- Issues: <https://github.com/eclipse/paho.mqtt.android/issues>
- Mailing-List: <https://dev.eclipse.org/mailman/listinfo/paho-dev>

□ <https://eclipse.org/paho/clients/android/>

Eclipse Paho Android Client



The screenshot shows the Eclipse Paho project description page. The header includes the Eclipse Paho logo and navigation links for Components, Documentation, and Community. The main content area is titled "Project description" and contains a paragraph about the project's purpose. Below this is a "Links" section with a list of external resources. The "Using the Paho Android Client" section is partially visible, showing a "Downloading" subsection with "Maven" listed.

Gradle

If you are using Android Studio and / or Gradle to manage your application dependencies and build then you can use the same repository to get the Paho Android Service. Add the Eclipse Maven repository to your `build.gradle` file and then add the Paho dependency to the `dependencies` section

```
repositories {
    maven {
        url "https://repo.eclipse.org/content/repositories/paho-snapshots/"
    }
}

dependencies {
    compile 'org.eclipse.paho:org.eclipse.paho.client.mqttv3:1.0.2'
    compile 'org.eclipse.paho:org.eclipse.paho.android.service:1.0.2'
}
```

Note: currently you have to include the `org.eclipse.paho:org.eclipse.paho.client.mqttv3` dependency as well. We are attempting to get the build to produce an Android `AAR` file that contains both the Android service as well as it's dependencies, however this is still experimental. If you wish to try it, remove the `org.eclipse.paho:org.eclipse.paho.client.mqttv3` dependency and append `@aar` to the end of the Android Service dependency. E.g. `org.eclipse.paho:org.eclipse.paho.android.service:1.0.2@aar`

If you find that there is functionality missing or bugs in the release version, you may want to try using the snapshot version to see if this helps before raising a feature request or an issue.

Building from source

- Open a terminal and navigate to this directory (`org.eclipse.paho.android.service`)
- Run the command `./gradlew clean assemble exportJar` or on Windows: `gradlew.bat clean assemble exportJar`

Running the sample app:

- Open the this current directory in Android Studio (`org.eclipse.paho.android.service`).
- In the toolbar along the top, there should be a dropdown menu. Make sure that it contains 'org.eclipse.paho.android.sample' then click the Green 'Run' Triangle. It should now build and launch an Virtual Android Device to run the App. If you have an Android device with developer mode turned on plugged in, you will have the opportunity to run it directly on that.
- If you have any problems, check out the Android Developer Documentation for help: <https://developer.android.com>

Χρήσιμοι σύνδεσμοι

□ MQTT

- <http://mqtt.org/>

□ Eclipse Paho Java client

- <https://eclipse.org/paho/clients/java/>
- <https://www.eclipse.org/paho/files/javadoc/org/eclipse/paho/client/mqttv3/package-summary.html>

□ Eclipse Paho Android Client

- <https://eclipse.org/paho/clients/android/>

□ Mosquitto Broker

- <https://mosquitto.org>
-

Χρήσιμοι σύνδεσμοι

- [Introduction to MQTT - learn.sparkfun.com](http://learn.sparkfun.com)
 - [MQTT 101 Tutorial: Introduction and Hands-on using Eclipse Mosquitto - Atadiat](#)
-