UNIVERSITÀ DEL PIEMONTE ORIENTALE

Message Queuing Telemetry Transport

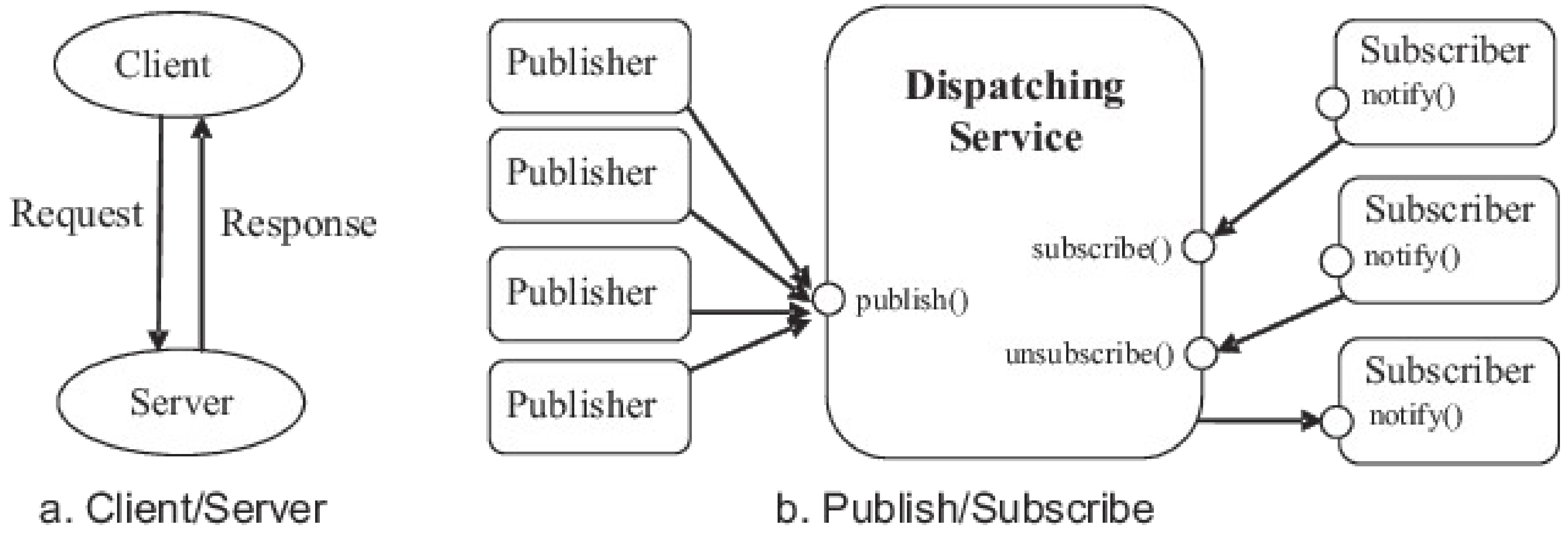
**Introduction to MQTT and its properties.
Example in Java with the Eclipse Paho library
and the Mosquitto broker.**

MQTT

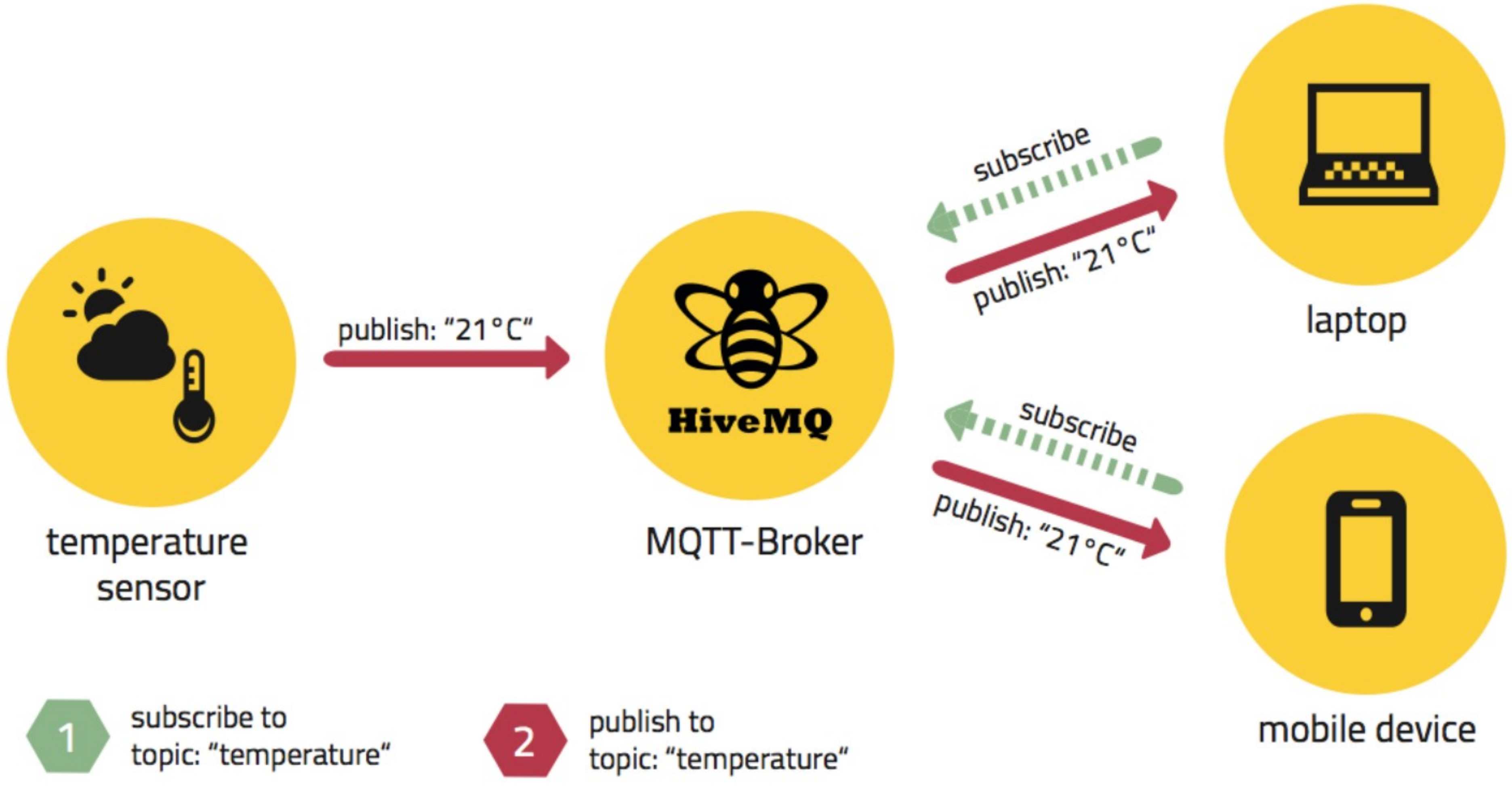
- "A machine-to-machine (M2M) connectivity protocol, extremely simple and lightweight **publish/subscribe** messaging protocol, designed for constrained devices and low-bandwidth, high-latency or unreliable networks."
  - well suitable for the IoT
- Message Queuing Telemetry Transport
  - http://mqtt.org
- Invented in 1999 by Andy Stanford-Clark (IBM) and Arlen Nipper (now Cirrus Link)
- Version 5 is an OASIS standard

Communication models

- Request/response model
  - es. HTTP
- Publish/subscribe model
  - es. MQTT



MQTT architecture



MQTT Publisher/Subscriber

- It implements a brokered publisher/subscriber pattern by relying on TCP
- Broker
  - the central communication point
  - it is in charge of dispatching all messages between the senders (publishers) and the rightful receivers (subscribers)
- Publisher
  - the sender of one or more messages
  - each message has a *topic*
- Subscriber
  - the receiver of one or more messages
  - each subscriber listens to one or more *topics*

MQTT properties

- The publisher and subscribers don't know about the existence of one another
- The clients don't know each other, but only the message broker, which filters all incoming messages and distributes them to the correct subscribers
- Therefore, we have:
  - space decoupling, publisher and subscriber do not need to know each other (e.g., by IP address and port)
  - time decoupling, publisher and subscriber do not need to be connected at the same time
  - synchronization decoupling, operations on both components are not halted during publishing or receiving messages

MQTT topics

- The routing information for the broker
- A simple UTF-8 string that can have more hierarchy levels – separated by a slash
- E.g., a sample topic for sending temperature data of a living room could be **home/living-room/temperature**
- The subscriber can subscribe to the exact topic or use a wild card

MQTT topics wildcards

- +
  - A single level wild card that only allows arbitrary values for one hierarchy
- #
  - a multilevel wild card that allows to subscribe to all the underlying hierarchy levels
- Examples:
  - home/+/temperature
  - home/#

MQTT QoS

- Each MQTT message is published with one of three Quality of Service (QoS) level
- Each QoS level is associated with different guarantees with regards to the reliability of the message delivery
- Both client and broker may provide additional persistence and redelivery mechanisms to increase reliability in case of network failures, restarts of the application, and other unforseen circumstances

MQTT QoS

QoS Level	Description
0	At most once delivery. The sender tries with best effort to send the message and relies on the reliability of TCP. No retransmission takes place.
1	At least once delivery. The receiver will get the message at least once. If the receiver does not acknowledge the message or the acknowledge gets lost on the way, it will be resent until the sender gets an acknowledgement. Duplicate messages can occur.
2	Exactly once delivery. The protocol makes sure that the message will arrive exactly once at the receiver by using a four step handshake. This increases communication overhead but is the best option when neither loss nor duplication of messages are acceptable.

Other options

- Last Will and Testament (LWT)
  - A LWT message can be specified by a publisher when connecting to a MQTT broker.
  - If that client doesn't disconnect gracefully, the broker sends out the LWT message on behalf of the client when connection loss is detected.
- Retained Messages
  - Each message may be sent as a retained message, i.e., its last known good value. It persists at the MQTT broker for the specified topic.
  - Every time a new client subscribes to that specific topic, it will instantly receive the last retained message on that topic.

Other options

- Clean/Durable Session
  - On connection, a client may set the "clean session" flag
  - If clean session is true, then all subscriptions will be removed for the client when it disconnects
  - If clean session is set to false, then the connection is treated as durable: when the client disconnects, any subscriptions it has will remain and any subsequent QoS 1 or 2 messages will be stored until it connects again in the future (with the same client id)

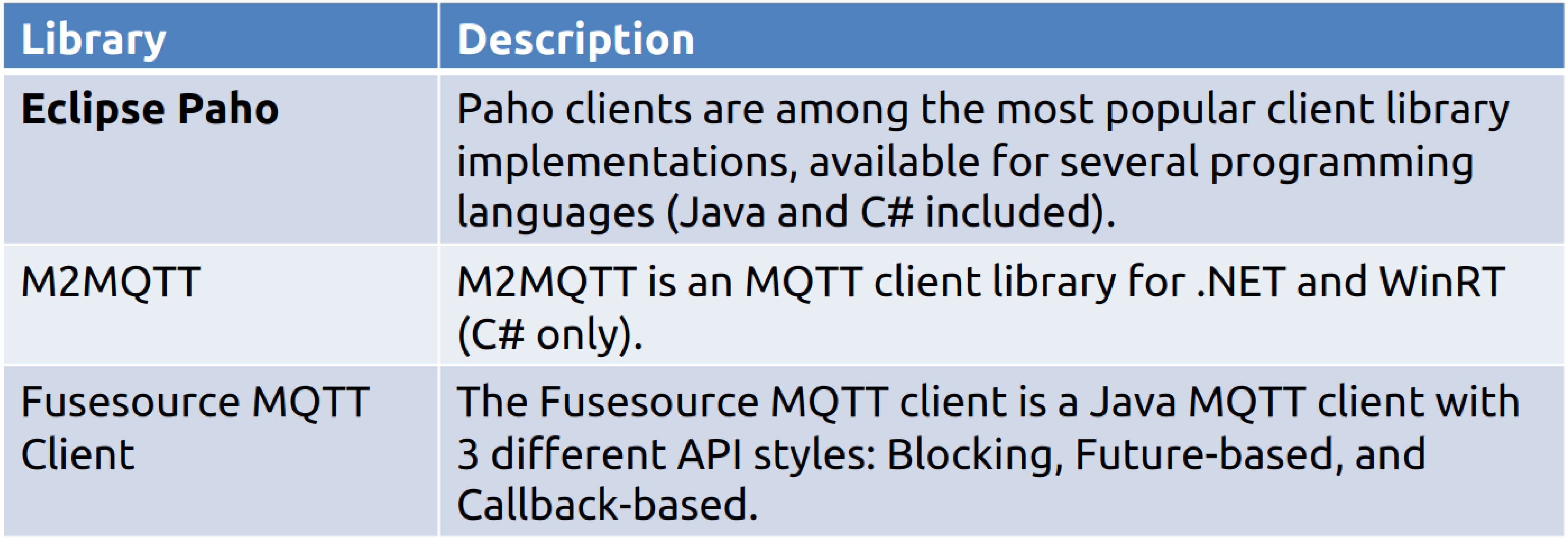
MQTT brokers

Broker	Description
Eclipse mosquitto	An open source MQTT broker written in C. It fully supports MQTT 3.1 and MQTT 3.1.1 and is very lightweight. Due to its small size, this broker can be used on constrained devices.
HiveMQ	A scalable, high-performance MQTT broker suitable for mission critical deployments. It fully supports MQTT 3.1 and MQTT 3.1.1 and has features like websockets, clustering, and an open-source plugin system for Java developers.
IBM WebSphereMQ	A commercial message- oriented middleware by IBM that fully supports MQTT.

Mosquitto installation

- Binary available for multiple OS, often directly in the system packages manager
  - https://mosquitto.org/download/
- Test brokers available at
  - http://test.mosquitto.org
- UniUPO broker
  - smartcity-challenge.edu-al.unipmn.it (IP: 193.206.52.98) porte standard di mqtt
  - user pissir, passwd pissir2020

MQTT Client Library (Java and C#)



Library	Description
Eclipse Paho	Paho clients are among the most popular client library implementations, available for several programming languages (Java and C# included).
M2MQTT	M2MQTT is an MQTT client library for .NET and WinRT (C# only).
Fusesource MQTT Client	The Fusesource MQTT client is a Java MQTT client with 3 different API styles: Blocking, Future-based, and Callback-based.

Eclipse Paho

- Open source implementation of the MQTT messaging protocol
- https://www.eclipse.org/paho/
- Available for several programming languages
- Latest version of the Java library:
  - 1.2.1

Eclipse Paho Installation

- Using gradle

```
compile
"org.eclipse.paho:org.eclipse.paho.client.mqttv3:1.2.5"
```

- or Maven

```
<dependency>
    <groupId>org.eclipse.paho<groupId>
    <artifactId>org.eclipse.paho.client.mqttv3</artifactId>
    <version>1.2.5</version>
</dependency>
```

MQTT Client Tools

Tool	Description
MQTT.fx	A JavaFX application with a clean interface: http://mqttfx.jensd.de. For Windows, macOS, and Linux.
mqtt-spy	An open source desktop and command line utility intended to help you with monitoring activity on MQTT topics: https://github.com/eclipse/paho.mqtt-spy. For Windows, macOS, and Linux.
MQTTLens	A Google Chrome extension, which connects to a MQTT broker and is able to subscribe and publish to MQTT topics: https://chrome.google.com/webstore/detail/mqttlens/hemojaaeigabkbcookmlgmdigohjobjm?hl=en

UNIVERSITÀ DEL PIEMONTE ORIENTALE

References

- MQTT
  - http://mqtt.org
- MQTT wiki
  - https://github.com/mqtt/mqtt.github.io/wiki
- Mosquitto man page
  - https://mosquitto.org/man/mqtt-7.html
- MQTT 101
  - https://www.hivemq.com/blog/how-to-get-started-with-mqtt
- MQTT Essentials
  - https://www.hivemq.com/mqtt-essentials/

License