Internet of Things Technologies

Table of Contents

Introduction

The Internet of Things (IoT) is the interconnection of uniquely identifiable embedded computing devices within the existing Internet infrastructure. Typically, IoT is expected to offer advanced connectivity of devices, systems, and services that goes beyond machine-to-machine communications (M2M) and covers a variety of protocols, domains, and applications. The interconnection of these embedded devices (including smart objects), is expected to usher in automation in nearly all fields, while also enabling advanced applications like a Smart Grid.

Things, in the IoT, can refer to a wide variety of devices such as heart monitoring implants, biochip transponders on farm animals, automobiles with built-in sensors, or field operation devices that assist fire-fighters in search and rescue. Current market examples include smart thermostat systems and washer/dryers that utilize wifi for remote monitoring.

The embedded computing nature of many IoT devices means that low-cost computing platforms are likely to be used. In fact, to minimize the impact of such devices on the environment and energy consumption, low-power radios are likely to be used for connection to the Internet. Such low-power radios do not use WiFi, or well established Cellular Network technologies, and remain an actively developing research area. However, the IoT will not be composed only of embedded devices, since higher order computing devices will be needed to perform heavier duty tasks (routing, switching, data processing, etc.). Companies such as FreeWave Technologies have developed and manufactured low power wireless data radios (both embedded and standalone) for over 20 years to enable Machine-to-Machine applications for the industrial internet of things.

Besides the plethora of new application areas for Internet connected automation to expand into, IoT is also expected to generate large amounts of data from diverse locations that is aggregated and very high-velocity, thereby increasing the need to better index, store and process such data.

IoT Hardware devices

A lot of small, portable and relatevly cheep hardware devices have recently emerged, that allow you to create your own small Internet of Things solutions.

Just to give you an idea of how easy it is to get started and build a simple IoT application, here is a tutorial of how to create your own smart greenhouse.

This beeing said, the market offers a couple of nice hardware devices to get you started:

- **Arduino** is a tool for making computers that can sense and control more of the physical world than your desktop computer. It's an open-source physical computing platform based on a simple microcontroller board, and a development environment for writing software for the board. Arduino can be used to develop interactive objects, taking inputs from a variety of switches or sensors, and controlling a variety of lights, motors, and other physical outputs.

- **Mbed OS** is an operating system for IoT devices and is especially well-suited to run in energy constrained environments. TheOS includes the connectivity, security and device management functionalities required in every IoT device. This OS is designed for ARM® Cortex®-M-based MCUs
- **BeagleBone** is the low-cost, high-expansion focused BeagleBoard using a low cost Sitara AM335x Cortex A8 ARM processor from Texas Instruments. It is similar to the earlier BeagleBoards and can act as a USB or Ethernet connected expansion companion for your current BeagleBoard and BeagleBoard-xM or work stand-alone. The BeagleBone is small even by BeagleBoard standards and with the high-performance ARM capabilities you expect from a BeagleBoard, the BeagleBone brings full-featured Linux to places it has never gone before.
- **Raspberry Pi** is a low cost, **credit-card sized computer** that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. It is a capable little device that enables people of all ages to explore computing, and to learn how to program in languages like Scratch and Python. It's capable of doing everything you'd expect a desktop computer to do, from browsing the internet and playing high-definition video, to making spreadsheets, word-processing, and playing games. What's more, the Raspberry Pi has the ability to interact with the outside world, and has been used in a wide array of digital maker projects, from music machines and parent detectors to weather stations and tweeting birdhouses with infra-red cameras. We want to see the Raspberry Pi being used by kids all over the world to learn to program and understand how computers work.

IoT Standards and Frameworks

As most of the targeted devices have low computational power, we need special standards and protocols to communicate with these devices.

Some of these protocols and standards are:

1. **MQTT** is an OASIS standard that implements a publish-subscribe communication model. It has several QoS levels making it easy to find the perfect tradeoff between reliability and resources/bandwidth usage.
   The protocol runs over TCP/IP, or over other network protocols that provide ordered, lossless, bi-directional connections. Its features include:
   1. Use of the publish/subscribe message pattern which provides one-to-many message distribution and decoupling of applications.
   2. A messaging transport that is agnostic to the content of the payload.
   3. Three qualities of service for message delivery:
      1. "At most once", where messages are delivered according to the best efforts of the operating environment. Message loss can occur. This level could be used, for example, with ambient sensor data where it does not matter if an individual reading is lost as the next one will be published soon after.
      2. "At least once", where messages are assured to arrive but duplicates can occur.
      3. "Exactly once", where message are assured to arrive exactly once. This level could be used, for example, with billing systems where duplicate or lost messages could lead to incorrect charges being applied.
   4. A small transport overhead and protocol exchanges minimized to reduce network traffic.
   5. A mechanism to notify interested parties when an abnormal disconnection occurs.
2. **OMA Lightweight M2M** is another interesting standard from the Open Mobile Alliance that is getting lot of traction in the domain of Device Management. LwM2M is proposing a standard way to do things like: reboot a device, install a new software image (yes, similarly to what happens on your smartphone and that is based on an ancestor of LwM2M called OMA-DM), etc.
3. **CoAP** (Constrained Application Protocol), which is an IETF standard targetting very constrained environments in which it's still desirable to have the kind of features you would expect from HTTP. CoAP provides a request/response interaction model between application endpoints, supports built-in discovery of services and resources, and includes key concepts of the Web such as URIs and Internet media types.CoAP is designed to easily interface with HTTP for integration with the Web while meeting specialized requirements such as multicast support, very low overhead, and simplicity for constrained environments.

The standards presented above are also supported by the Eclipse community. They provide some sandbox servers and libraries for the protocols:

1. Eclipse Paho project libraries and Mosquitto server for the MQTT protocol
2. Eclipse Californium server and implementation for the CoAP protocol
3. Eclipse Wakaama and the proposed project Leshan are implementing LwM2M stacks in C and Java that you can use to manage your IoT devices

You can check out this "Getting Started" guide if you want to read more general things about these protocols, and their libraries.

Also, here is a list of open source sandbox servers provided by Eclipse. You can use them to test your IoT solutions, without creating and setting up the servers yourself.

Further on, I will present the MQTT and the CoAP protocols and their libraries, and I'll work through some examples to demonstrate their functionalities.

Some notes before going forward:

- I will not always get into a lot of details. I strongly suggest you read the specifications of the framewords for a better understanding. You can find them in the References section.

The MQTT Standard

Introduction, Main Concepts

Terminology

As mentioned before, the MQTT protocol implements a publish/subscribe communication model, and a client-server system architecture. This means that an MQTT system will be split between the following two main components:

1. **Client:** A program or device that uses MQTT. A Client always establishes the Network Connection to the Server. It can:
    1. Publish Application Messages that other Clients might be interested in.
    2. Subscribe to request Application Messages that it is interested in receiving.
    3. Unsubscribe to remove a request for Application Messages.
    4. Disconnect from the Server.
2. **Server:** A program or device that acts as an intermediary between Clients which publish Application Messages and Clients which have made Subscriptions. A Server
    1. Accepts Network Connections from Clients.
    2. Accepts Application Messages published by Clients.

3. Processes Subscribe and Unsubscribe requests from Clients.
4. Forwards Application Messages that match Client Subscriptions.

Also, the terminology corresponding to the publish/subscribe communication model and that specific to the MQTT protocol defines the following terms:

Subscription:
A Subscription comprises a Topic Filter and a maximum QoS. A Subscription is associated with a single Session. A Session can contain more than one Subscription. Each Subscription within a session has a different Topic Filter.

Topic Name:
The label attached to an Application Message which is matched against the Subscriptions known to the Server. The Server sends a copy of the Application Message to each Client that has a matching Subscription.

Topic Filter:
An expression contained in a Subscription, to indicate an interest in one or more topics. A Topic Filter can include wildcard characters.

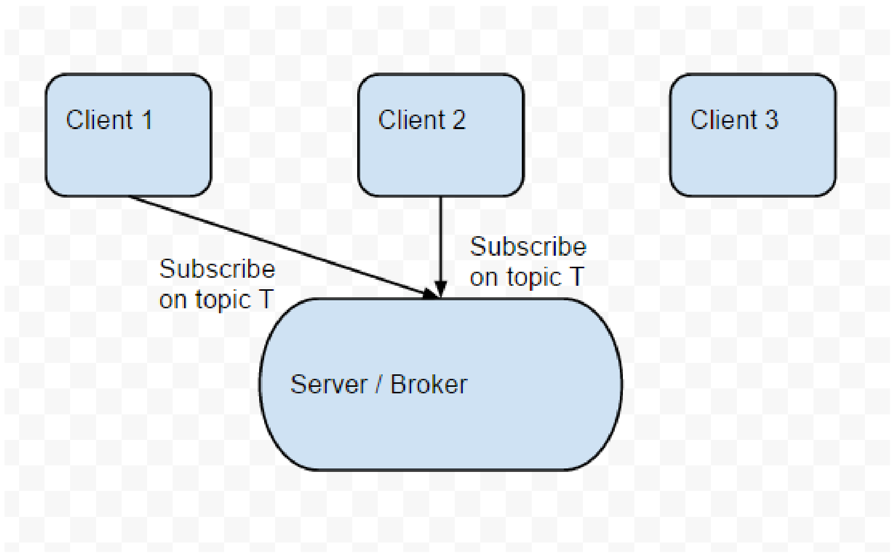
Session:
A stateful interaction between a Client and a Server**.** Some Sessions last only as long as the Network Connection, others can span multiple consecutive Network Connections between a Client and a Server.

MQTT Control Packet:
A packet of information that is sent across the Network Connection. The MQTT specification defines fourteen different types of Control Packet, one of which (the PUBLISH packet) is used to convey Application Messages.

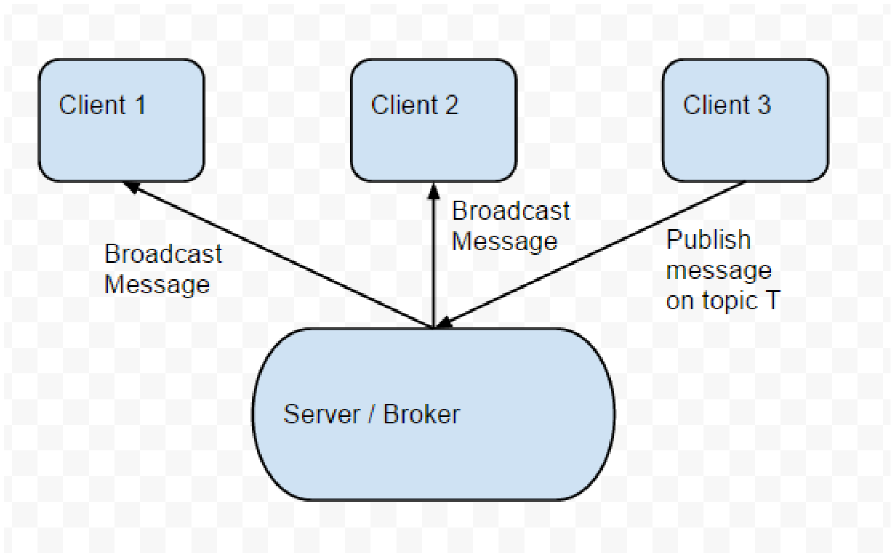
How it works

As mentioned above, MQTT implements a client/server architecture. More specifically it builds upon a broker architectural pattern. Here, the server will act as the broker, and broadcasts messeges published to a certain topic to all the subscribers of that topic.

For example, let's say we have three clients that want to communicate on topic T:



First, Client 1 and Client 2 will subscribe to topic T on the server, then the third client will publish a message on that topic. At this point, the server will broadcast the message to all subscribers of topic T, that being Client 1 and Client 2:



The server will match topics in an hierarchical mode. For example, consider the topics *foo/bar/topic* and *foo/bar/foobar/topic*. You can subscribe to a whole group of topics by using wildcards:

- + will match a single entry name
  e.g. *foo/+/topic* will match the first topic *foo/bar/topic,* but not the second *foo/bar/foobar/topic*
- # will match any number of entries
  e.g. *foo/#* will match both of the topics. You can use *foo/#* to subscribe to the whole *foo* group.

Note that using wildcards will only work for subscriptions. When publishing, you'll have to use a specific topic.

References

- 5 Things to get started with IoT – Eclipse
- MQTT Specification.
- COAP Specification
- Arduino Introduction
- Mbed OS
- BeagleBone
- Raspberry Pi
- verry nice presentation for MQTT and CoAP