

M2M Communication over heterogeneous WSN infrastructure

Implementation of MQTT-SN

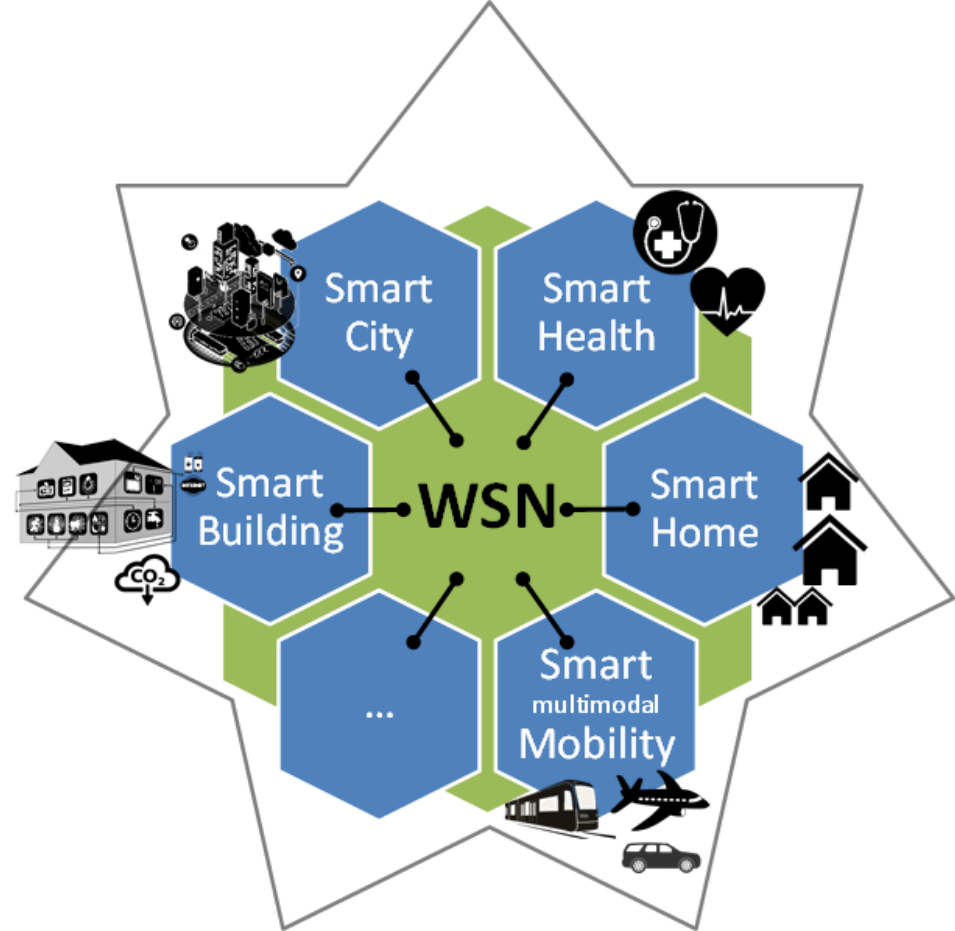
Kamaratakis Georgios - mtp214

A dark blue diagonal gradient bar that starts from the bottom left corner and extends towards the top right corner, covering the lower half of the slide.

Table of contents:

- WSNs
- Purpose
- Example
- MQTT
- MQTT-SN
- MQTT vs MQTT-SN
- Pros & Cons
- MQTT-SN Gateway
- Implementation
- Devices used for this project

WSNs



- Bluetooth / BLE
- LoRa
- Zigbee / 6LowPan
- WiFi

Purpose

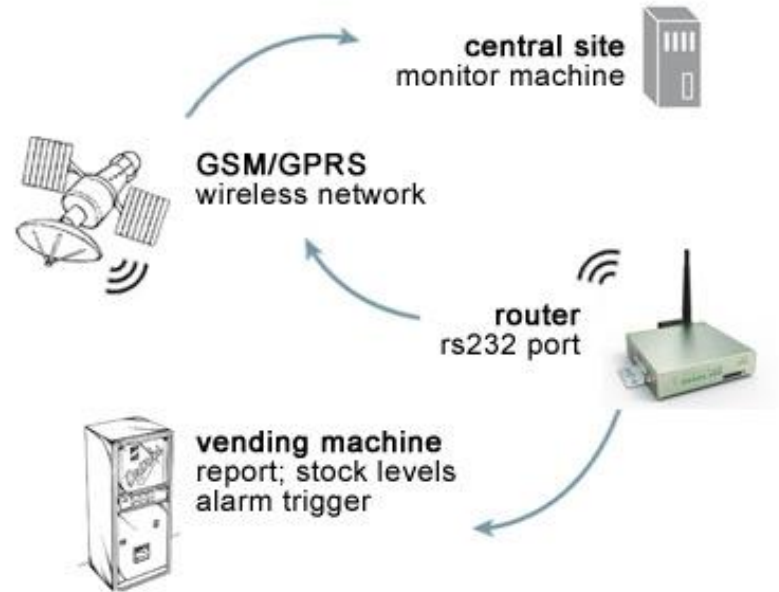


Connect to **any** device despite the **wireless technology**

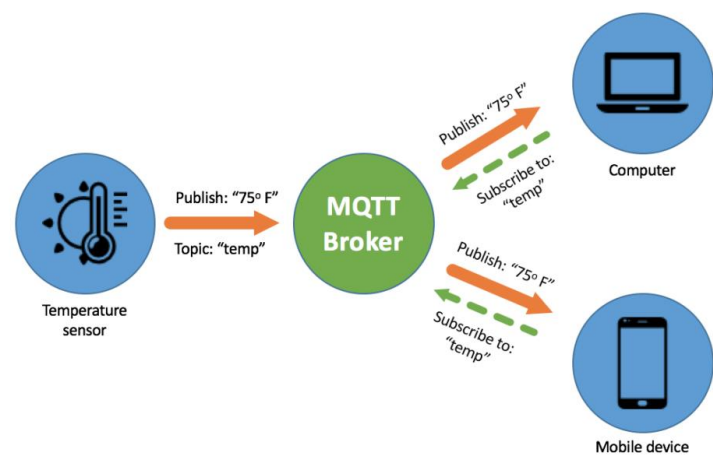
Example

Zigbee device ↔ **Bluetooth** device ↔ **WiFi** device

Or



MQTT



- **MQTT Broker**
- **MQTT Client**
- **TOPIC**
 - Subscribe
 - Publish
- **QoS**
 - QoS 0 - At most Once
 - QoS 1 - At least Once
 - QoS 2 - Exactly Once
- **Runs over TCP** protocol like HTTP

MQTT-SN

- It is similar to MQTT
- It is even more Lightweight
- It is designed with scalability in mind
- Taking account of power consumption of IoT devices
 - There is a new offline keep-alive procedure for the support of sleeping clients.
- Designed to be used between different communication networks
 - It is designed that the messages that MQTT-SN produces are small enough to be contained.
- **Architecture:**
 - MQTT-SN Client
 - MQTT-SN Gateway
 - MQTT-SN Forwarder
- **QoS**
 - QoS 0 - At most Once
 - QoS 1 - At least Once
 - QoS 2 - Exactly Once

Creator : Ian Craggs (IBM)

MQTT-SN

4 MQTT-SN Architecture

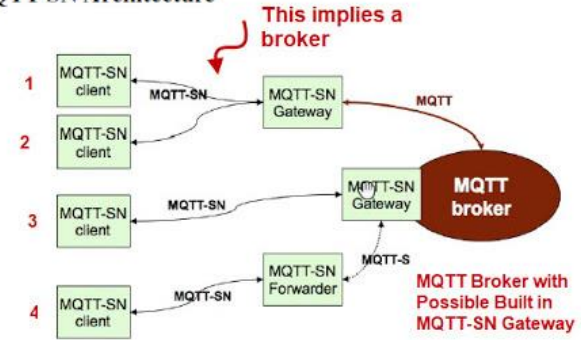
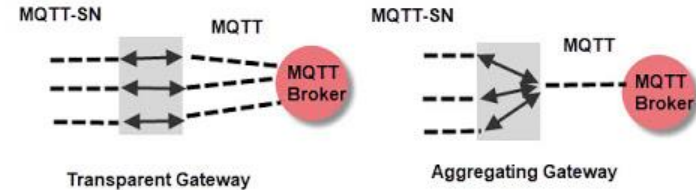


Figure 1: MQTT-SN Architecture

Is The connection between client 1 and client 2 MQTT-SN? or MQTT-SN>MQTT>MQTT-SN ?

MQTT-SN GateWay Types



Notes: With a transparent gateway each MQTT-SN connection has a MQTT connection to the broker.

With an aggregating gateway the MQTT-SN connections share a single MQTT

MQTT-SN

Messages:

- ADVERTISE
- SEARCHGW
- GWINFO
- CONNECT
- CONNACK
- REGISTER
- REGACK
- PUBLISH
- PUBACK
- PUBREC, PUBREL, and PUBCOMP
- SUBSCRIBE
- SUBACK
- UNSUBSCRIBE
- UNSUBACK
- PINGREQ
- PINGRESP
- DISCONNECT

MQTT vs MQTT-SN

MQTT:

Runs on TCP:

- Client
- Broker

Simple Implementation

MQTT-SN:

Runs on UDP:

- Client
- Gateway
- Broker
- Forwarder

Smaller message payload

Lower energy consumption

Keep-alive connections (for sleeping devices)

PROs & CONs of MQTT-SN

PROs:

1. MQTT-SN uses **topic ID** instead of topic name. First client sends a registration request with topic name and topic ID (2 octets) to a broker. After the registration is accepted, client uses topic ID to refer the topic name. This saves media bandwidth and device memory - it is quite expensive to keep and send topic name e.g: home/livingroom/socket2/meter in memory for each publish message.
1. MQTT-SN **does not require TCP/IP** stack. It can be used over a serial link (preferred way), where with simple link protocol (to distinguish different devices on the line) overhead is really small. Alternatively it can be used over UDP, which is **less hungry than TCP**.

CONs:

1. You need some sort of **gateway**, which is nothing else than a TCP or UDP stack moved to a different device. This can also be a simple device (e.g.: Arduino Uno) just serving multiple MQTT-SN devices without doing other job.
2. MQTT-SN is **not well supported**.

MQTT-SN

Inside a packet

Message Header (2 or 4 octets)	Message Variable Part (n octets)	Length (1 or 3 octets)	MsgType (1 octet)
-----------------------------------	-------------------------------------	---------------------------	----------------------

Table 1: General Message Format

Table 2: Message Header

MsgType Field Value	MsgType	MsgType Field Value	MsgType
0x00	ADVERTISE	0x01	SEARCHGW
0x02	GWINFO	0x03	reserved
0x04	CONNECT	0x05	CONNACK
0x06	WILLTOPICREQ	0x07	WILLTOPIC
0x08	WILLMSGREQ	0x09	WILLMSG
0x0A	REGISTER	0x0B	REGACK
0x0C	PUBLISH	0x0D	PUBACK
0x0E	PUBCOMP	0x0F	PUBREC
0x10	PUBREL	0x11	reserved
0x12	SUBSCRIBE	0x13	SUBACK
0x14	UNSUBSCRIBE	0x15	UNSUBACK
0x16	PINGREQ	0x17	PINGRESP
0x18	DISCONNECT	0x19	reserved
0x1A	WILLTOPICUPD	0x1B	WILLTOPICRESP
0x1C	WILLMSGUPD	0x1D	WILLMSGRESP
0x1E-0xFD	reserved	0xFE	Encapsulated message
0xFF	reserved		

Table 3: Values of the MsgType field

DUP (bit 7)	QoS (6,5)	Retain (4)	Will (3)	CleanSession (2)	TopicIdType (1,0)
----------------	--------------	---------------	-------------	---------------------	----------------------

Table 4: Flags field

Length (octet 0)	MsgType (1)	Flags (2)	TopicId (3-4)	MsgId (5-6)	Data (7:n)
---------------------	----------------	--------------	------------------	----------------	---------------

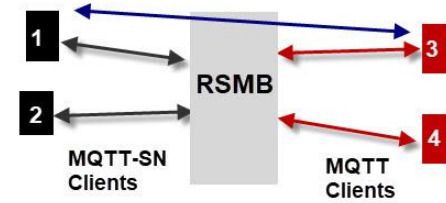
Table 16: PUBLISH Message

Bits are numbered by MSB (most-significant bit first)

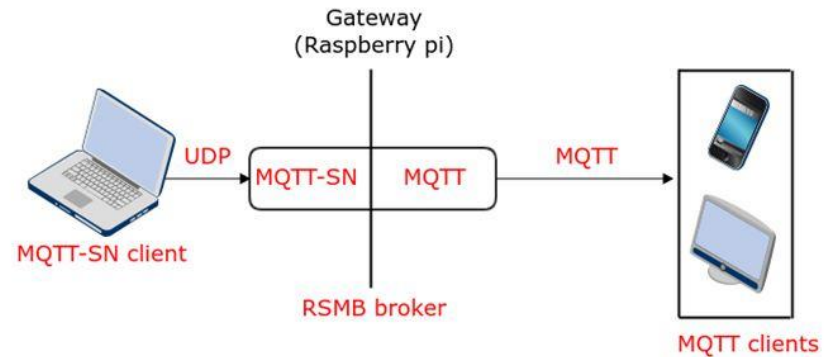
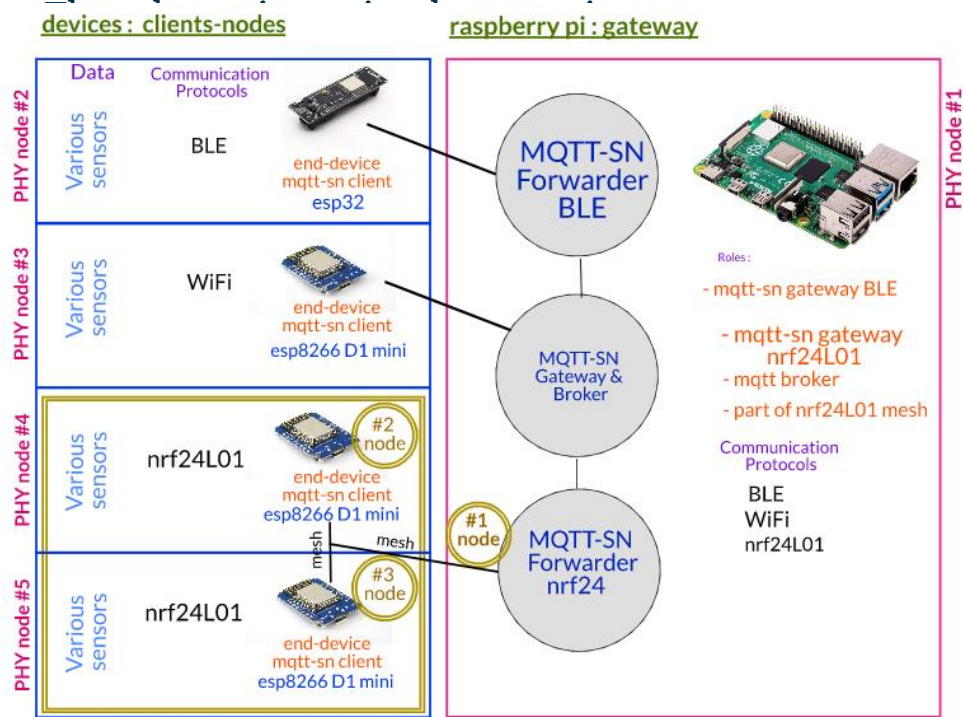
RSMB (Really Small Message Broker):

- Acts as an **aggregating gateway**
- Acts as a **broker**
- **Transforms messages** from **MQTT-SN** to **MQTT** and vice versa
- Has two udp ports
- Creator : Ian Craggs (IBM)
- History : Was the basis for Mosquitto MQTT broker
- Alternatives : **Project Eclipse Paho**

Really Small Message Broker (RSMB) Functionality Diagram



Client 1 to client 2 then RSMB=MQTT-SN Broker
Client 3 to client 4 then RSMB=MQTT Broker
Client 1 to client 3 then RSMB=MQTT-SN -MQTT Gateway



Schema

end-devices

Communication

1. BLE

MQTT-SN packets
through BLE connection

2. WiFi

MQTT packets

BLE <-> UDP

Raspberry
python file acting as
a forwarder

Raspberry Pi

RSMB

MQTT -SN
Gateway

MQTT
BROKER

Implementation

[MQTT-SN over WiFi communication]

This is the regular procedure of MQTT

We use the **mqtt library** and we connect to the WiFi

We make configurations for connecting to the
Broker that is connected to MQTT-SN gateway.

Start subscribing/publishing

Implementation

[MQTT-SN over BLE communication]

Regular Bluetooth:

- RX
- TX

BLE (Bluetooth Low Energy):

- Service UUID 1
 - Characteristic UUID 1
 - Characteristic UUID 2
 - Characteristic UUID 3
- Service UUID 2
 -

In my case:

We adapted the UART scheme on the BLE protocol:

- Service UUID (ble-uart-esp32)
 - Characteristic UUID for (RX)
 - Characteristic UUID for (TX)

Implementation

[MQTT-SN over BLE communication]

- We included the libraries for BLE and for MQTT-SN
- We implemented UART schema through BLE and now we can send data as it would be Serial
- MQTT-SN library on esp32 has handlers for sending & receiving the packets.
- By modifying those handlers we passed the buffers from the BLE characteristics to the MQTT-SN library to parse the packets.

ESP32 Scope

Implementation

[MQTT-SN over BLE communication]

- We installed the libraries for BLE in Python (BluePy) & UDP (Datagram Socket) library
- We read from Rx Characteristic of the RPi and we pass the buffer as it is to the MQTT-SN gateway through UDP connection
- Similar goes for when the gateway sends data through the Tx Characteristic

RPi Scope

Example of using MQTT-SN library

Boriz / MQTT-SN-Arduino / Arduino / MqttsnClient / MqttsnClient.ino

<https://github.com/boriz/MQTT-SN-Arduino>

Devices

Esp32 Ble WiFi Battery



Esp8266 WiFi Wemos D1 Mini



Questions ?