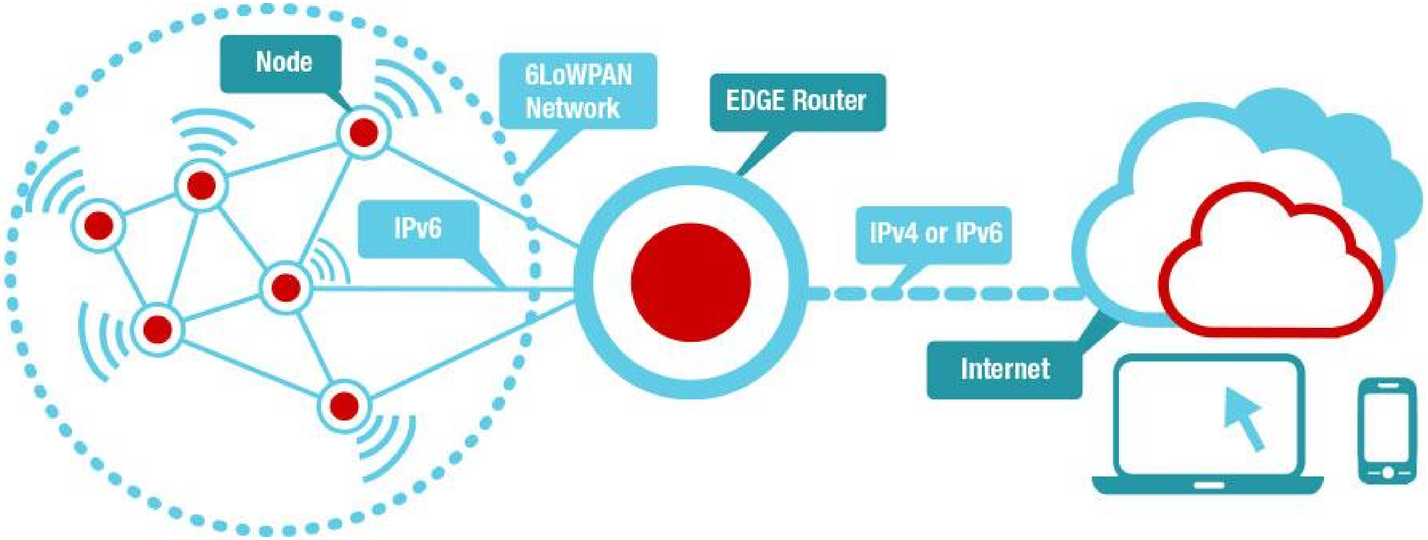
ICTP Workshop, day 3

Connecting to the external world

Workshop on Scientific Applications for the Internet of Things (IoT), March 2015 – Antonio Liñán Colina, Zolertia.

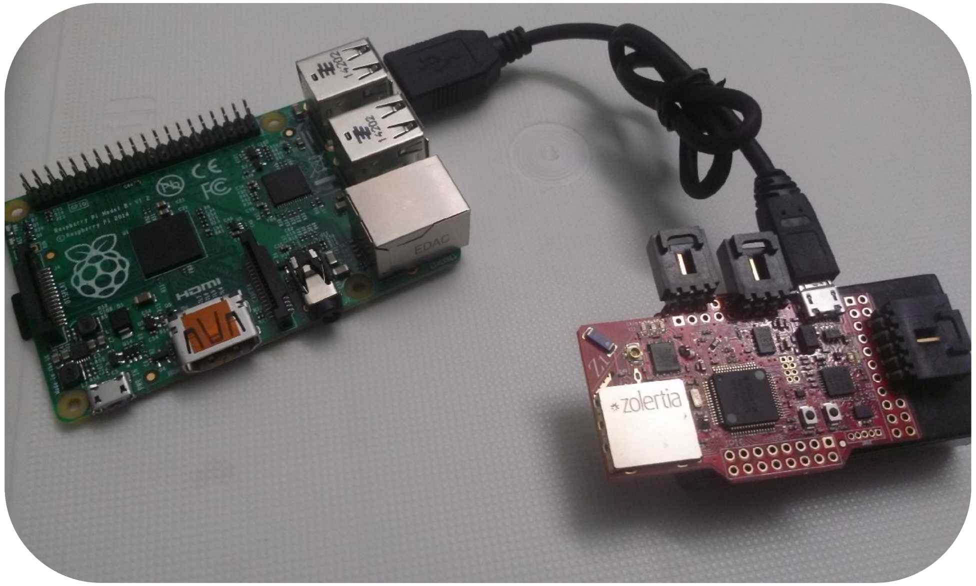
The Border Router

The border router or edge router is typically a device sitting at the edge of our network, which allow us to talk to outside networks using its built-in network interfaces



How the Border Router Works?

In Contiki it uses a serial-based interface called SLIP, it allows to connect a given mote to a host, and assign an IPv6 prefix to set the network global IPv6 addresses.



Non-sleepy Border Router

Normally is preferable to configure the border router as a non-sleeping device using the project-conf.h file, add:

```
#undef NETSTACK_CONF_RDC
#define NETSTACK_CONF_RDC nullrdc_driver
```

Set up the Border Router

```
cd examples/ipv6/rpl-border-router/
make TARGET=z1 savetarget
make clean
make border-router.upload && make connect-router PREFIX=aaaa::1/64
```

The Global IPv6 address of the Border Router and the other Nodes will be created using the given prefix plus the Z1 MAC address.
You need to provide a /64 prefix!

Border Router running - 1

```
********SLIP started on ``/dev/ttyUSB0''
opened tun device ``/dev/tun0''
ifconfig tun0 inet `hostname` up
ifconfig tun0 add aaaa::1/64
ifconfig tun0 add fe80::0:0:0:1/64
ifconfig tun0

tun0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          inet addr:127.0.1.1  P-t-P:127.0.1.1  Mask:255.255.255.255
          inet6 addr: fe80::1/64 Scope:Link
          inet6 addr: aaaa::1/64 Scope:Global
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:500
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

*** Address:aaaa::1 => aaaa:0000:0000:0000
Got configuration message of type P
Setting prefix aaaa::
Server IPv6 addresses:
 aaaa::c30c:0:0:100
 fe80::c30c:0:0:100
```

Border Router running - 2

```
ifconfig

tun0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          inet addr:127.0.1.1  P-t-P:127.0.1.1  Mask:255.255.255.255
          inet6 addr: fe80::1/64 Scope:Link
          inet6 addr: aaaa::1/64 Scope:Global
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:500
          RX bytes:0 (0.0 B)  TX bytes:152 (152.0 B)
```

Border Router running - 3

```
ping6 aaaa::1
PING aaaa::1(aaaa::1) 56 data bytes
64 bytes from aaaa::1: icmp_seq=1 ttl=64 time=0.074 ms
64 bytes from aaaa::1: icmp_seq=2 ttl=64 time=0.060 ms
64 bytes from aaaa::1: icmp_seq=3 ttl=64 time=0.037 ms

ping6 aaaa::c30c:0:0:100
PING aaaa::c30c:0:0:100(aaaa::c30c:0:0:100) 56 data bytes
64 bytes from aaaa::c30c:0:0:100: icmp_seq=1 ttl=64 time=932 ms
64 bytes from aaaa::c30c:0:0:100: icmp_seq=2 ttl=64 time=21.8 ms
64 bytes from aaaa::c30c:0:0:100: icmp_seq=3 ttl=64 time=21.8 ms
```

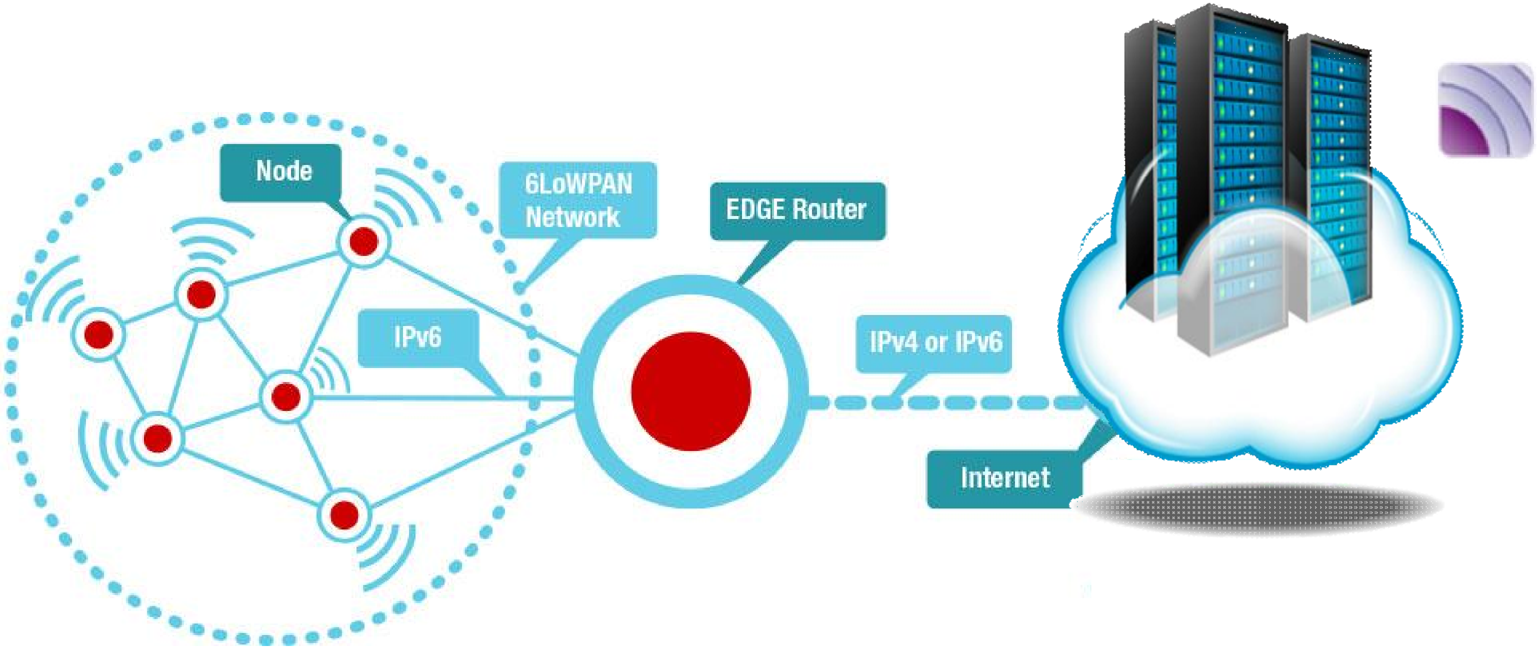
Border Router running - 4

```
http://[aaaa::c30c:0:0:100]/

Neighbors

fe80::c30c:0:0:1
fe80::c30c:0:0:12d7
fe80::c30c:0:0:12d1
fe80::c30c:0:0:12f2
fe80::c30c:0:0:12e5
fe80::c30c:0:0:12d4
fe80::c30c:0:0:12c1
fe80::c30c:0:0:12d3

Routes

aaaa::c30c:0:0:12f2/128 (via fe80::c30c:0:0:12f2) 50s
aaaa::c30c:0:0:12d1/128 (via fe80::c30c:0:0:12d1) 50s
```

Pro-tip: sniff the Border Router using Wireshark

Run Wireshark and sniff the tun0 interface

Hands-on: MQTT mesh network



Workshop on Scientific Applications for the Internet of Things (IoT), March 2015 – Antonio Liñán Colina, Zolertia.

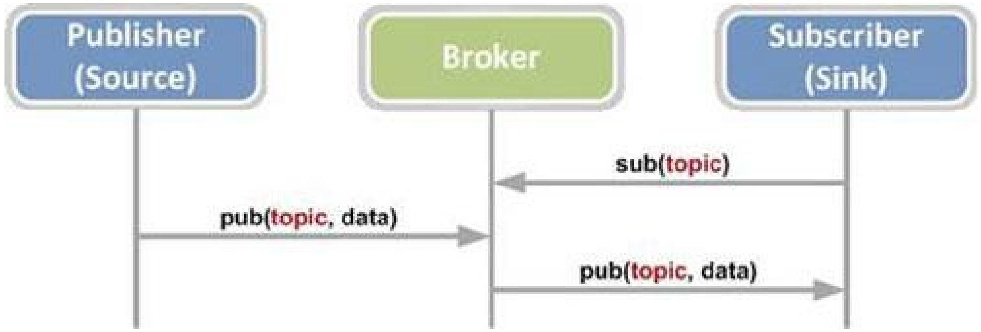
MQTT?

MQTT is a machine-to-machine (M2M)/"Internet of Things" protocol.

It was designed as an extremely lightweight publish/subscribe messaging transport.

It is useful for connections with remote locations where a small code footprint is required and/or network bandwidth is at a premium.

Runs on top of TCP/IP, connection-oriented.



Hands-on: rules

I will be the host and the Border Router, only one allowed!

No other radio activity or application running at the same time! At least one that sends data on channel 26 with PANID 0xABCD.

Teams can set up a Sniffer + Wireshark (IoT book, page 63: 7.3.2. SenSniff IEEE 802.15.4 wireless sniffer).

Other teams are to program a Z1 mote as a MQTT-enabled device, using the SHT25 temperature and humidity sensor.

Hands-on: get the MQTT example

From the ICTP workshop repository:
https://github.com/alignan/contiki/tree/ictp_2015

Clean instructions to get a fresh copy and create a work branch:

```
cd $HOME
git clone https://github.com/alignan/contiki contiki-ictp-mqtt
cd contiki-ictp-mqtt
git checkout –b ictp_2015 origin/ictp_2015
git checkout –b work
```

Hands-on: compile and upload!

```
cd examples/z1/mqtt-demo
make TARGET=z1 savetarget
```

Check if the mote is connected and claimed by the Instant Contiki VM

```
make z1-motelist
```

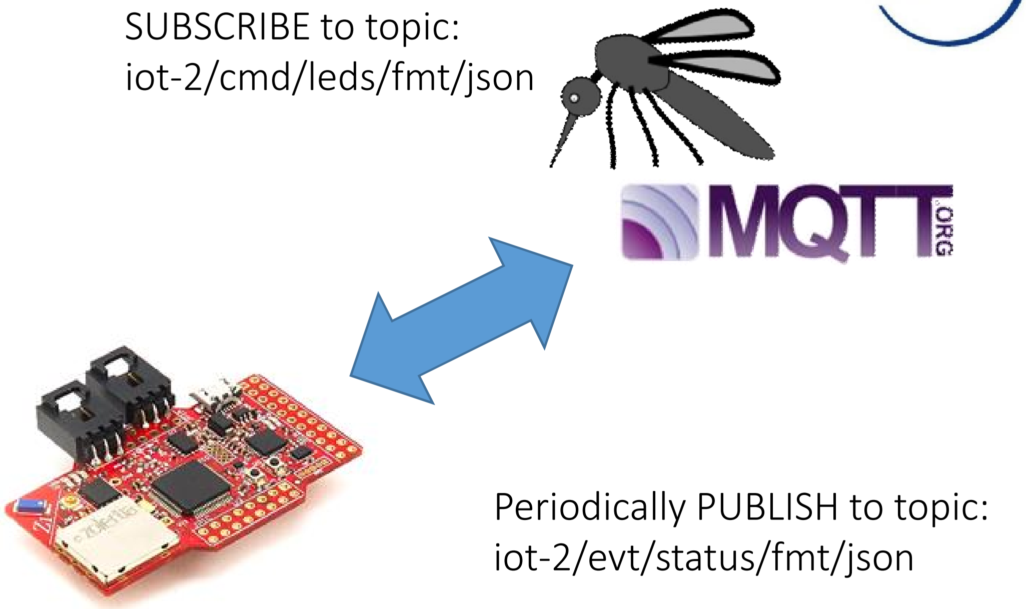
In the project-conf.h file, change BOARD_STRING with your name (short!)

If OK, then program:

```
make clean && make mqtt-demo.upload && make z1-reset && make login
```

Hands-on: awww yisssss!

```
Rime started with address 193.12.0.0.0.0.0.158
MAC c1:0c:00:00:00:00:00:9e Ref ID: 158
Contiki-2.6-2076-g2192083 started. Node id is set to 158.
CSMA nullrdc, channel check rate 128 Hz, radio channel 26
Tentative link-local IPv6 address fe80:0000:0000:0000:c30c:0000:0000:009e
Starting 'MQTT Demo'
MQTT Demo Process
MQTT - Registered successfully
MQTT - Sending CONNECT message...
MQTT - (send_out_buffer) Space used in buffer: 70
MQTT - Got TCP_DATA_SENT
MQTT - Got TCP_DATA_SENT
MQTT - Got TCP_DATA_SENT
MQTT - Got CONNACK
MQTT - Done sending CONNECT
MQTT - Call to mqtt_subscribe...
MQTT - Accepted!
MQTT - Got mqtt_do_subscribe_mqtt_event!
MQTT - Sending subscribe message! topic iot-2/cmd/+/fmt/json topic_length 20
MQTT - Got SUBACK
MQTT - Done in send_subscribe!
```



SUBSCRIBE to topic:
iot-2/cmd/leds/fmt/json

Periodically PUBLISH to topic:
iot-2/evt/status/fmt/json

Hands-on: relevant files

apps/mqtt
examples/z1/mqtt-demo
examples/cc2538dk/mqtt-demo/README.md

MQTT broker: install and stuff (Ubuntu)

```
sudo apt-add-repository ppa:mosquitto-dev/mosquitto-ppa
sudo apt-get update
sudo apt-get install mosquitto mosquitto-clients
```

To start:

```
sudo service mosquitto start
```

Mosquitto
An Open Source MQTT v3.1 Broker

To stop:

```
sudo service mosquitto stop
```

As default binds to the local address and port 1883, in our example to [aaaa::1]:1883

MQTT broker at host: subscribe to event

```
mosquitto_sub -d -t iot-2/evt/status/fmt/json

Client mosqsub/21129-vm sending CONNECT
Client mosqsub/21129-vm received CONNACK
Client mosqsub/21129-vm sending SUBSCRIBE (Mid: 1, Topic: iot-2/evt/status/fmt/json, QoS: 0)
Client mosqsub/21129-vm received SUBACK
Subscribed (mid: 1): 0

Client mosqsub/21129-vm received PUBLISH (d0, q0, r0, m0, 'iot-2/evt/status/fmt/json', ... (149
bytes))
{"d":{"myName":"Antonio","Seq #":20,"Uptime (sec)":319,"Def
Route":"fe80::c30c:0:0:baba","RSSI (dBm)":-64,"SHT25 Temp (mC)":2265,"Humidity (RH)":6798}}
```

MQTT broker at host: publish (LED ON)

Turn the RED LED On!

```
mosquitto_pub –d -h aaaa::1 -m "1" -t iot-2/cmd/leds/fmt/json
```

```
Client mosqpub/24958-vm sending CONNECT
Client mosqpub/24958-vm received CONNACK
Client mosqpub/24958-vm sending PUBLISH (d0, q0, r0, m1, 'iot-2/cmd/leds/fmt/json', ... (1 bytes))
Client mosqpub/24958-vm sending DISCONNECT
```

And turn the RED LED Off!

```
mosquitto_pub –d -h aaaa::1 -m “0" -t iot-2/cmd/leds/fmt/json
```

Proposed exercises

Browse through the mqtt-demo.c file, search for the “publish(void)” function and see what data is being sent.

Have you noticed how the Green LED toggles faster after the boot, then slower, and finally only toggles every 15 seconds? Why is that? (Hint: check for the STATUS_LED, or just go through the README, but that’s not fun).

Use the sniffer and wireshark to see what is being transmitted, what protocols are we using?

In which part of the mqtt-demo are we creating the connection?

When we receive an update from a topic we are subscribed to, where is this handled? (Hint: check for the available mqtt events at “mqtt_event” function).