REALIZATION OF A NETWORK STACK THAT SUPPORTS TAKS+WIDS ON WSN WITH MOTE RUNNER

DISIM - Università degli Studi dell'Aquila

Students:
Andrea Salini - 231413
Lorenzo Di Giuseppe - 227515
Matteo Gentile - 230997

Professors:
Fortunato Santucci
Luigi Pomante

April 15, 2015

INTRODUCTION

-

INDEX

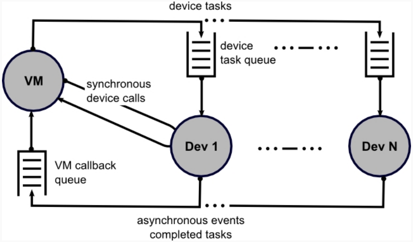
- Introduction to Mote Runner
- Testing Mote Runner

INTRODUCTION TO MOTE RUNNER

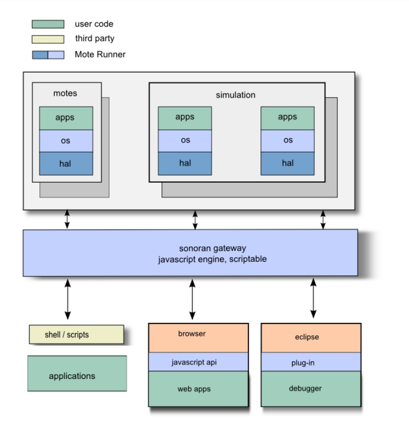
MOTE RUNNER

- An OS and a runtime and development environment for WSN
- Key features:
  - Support for RT constraints & energy awareness
  - Portability thanks to a VM that abstracts the HW
  - Event oriented programming paradigm
  - High level coding (Java - C#)
  - Debugging & simulation environments
- It’s still in beta and is evolving towards IoT

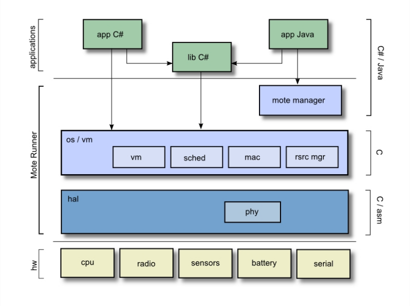
MOTE RUNNER


device tasks
device
task queue
VM
synchronous
device calls
Dev 1
Dev N
VM callback
queue
asynchronous events
completed tasks

MOTE RUNNER



user code
third party
Mote Runner
motes
apps
os
hal
simulation
apps
os
hal
apps
os
hal
sonoran gateway
javascript engine, scriptable
shell / scripts
applications
browser
javascript api
web apps
eclipse
plug-in
debugger



applications
app C#
lib C#
app Java
C# / Java
Mote Runner
mote manager
os / vm
vm
sched
mac
rsrc mgr
C
hal
phy
C / asm
hw
cpu
radio
sensors
battery
serial

MOTE RUNNER - V.11, V.13 BETA

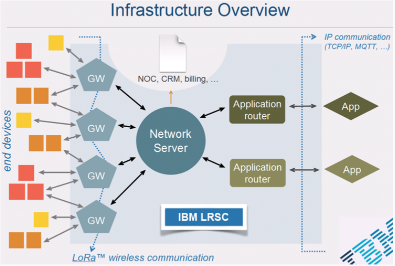
- They support IEEE 802.15.4
  - exposing a low radio level API that can be used to implement custom MAC layer
  - dropping messages with header structure not 802.15.4 compliant in the radio stack
- Offer Hopi
  - A multi-hop data gathering protocol
  - Used to collect data from motes setting automatically a tree network

MOTE RUNNER - V.17.1.8C (LATEST)

- Supports only two platforms: IMST & Blipper
- It's based on a different radio layer: LoRa™
- It offers a build-in MAC layer: LRSC - Low Range Signaling & Control
  - It supports only a network topology: the LRSC one
  - The offered API is poor since the radio is hidden in the firmware (not compatible with previous versions)

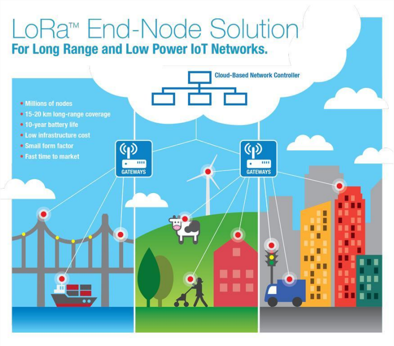
LRSC - ARCHITECTURE

- Gateways (GW) are connected to server on IP
- Motes comunicate with server in tunneling TCP/UDP over IP
- Motes comunicate with GW with LoRa single-hop



LORA^{TM}

- LoRa™Alliance
  - Target: IoT, machine-to-machine (M2M), smart city, and industrial applications
  - Intiated to standardize Low Power Wide Area Networks (LPWAN)



LORA™

- LoRa™Technology
  - LoRaWAN pledeges to extend the radio range by 10x while using only one third of the power used by competing solutions
  - Star (of stars?) topology
  - Gateways relay messages between end-devices and a central network server
  - Communication between end-devices and gateways is spread out on different frequency channels and data rates.
  - Data rates: 0.3 - 50 kbps

LORA™

- …and more
  - adaptive data rate (ADR)
  - secure communication (on network and application layers and end-point device key)
  - three classes of end-point devices.
  - More info on http://lora-alliance.org/

MOTE RUNNER - CONCLUSION

- For the purpose of this work (TAKS & WIDS):
  - MR allows dynamic reprogramming of motes with a control server using WLIP
  - v.17.1.8c is not suitable
    - LoRa is available only for a limited number of platforms (until now!)
    - LRSC doesn't permit to customize the MAC behaviour
    - The radio is not exposed
  - v.11, v.13 are better choices:
    - radio interface could be used to implement an 802.15.4 MAC with TAKS support
    - this MAC could be used to build upper layer with WIDS
- This does not exclude a future integration with LoRa-LRSC

MOTE RUNNER

•

TESTING MOTE RUNNER

PROBLEMS

- MR v.13 offers:
  - Radio interface IEEE 802.15.4 compliant
  - Hopi
  - A simulation environment IRIS friendly
  - Many nice features (Debugger, Logger and so on)

PROGRAMMING THE RADIO

- com.ibm.saguaro.system.Radio
  - This is a generic class in the IBM saguaro system to use the device radio
  - It offers a low level API with the following functionality:
    - open: opens the radio, once opened no other assembly can use it
    - close: releases the radio so that others can use it
    - setter and getters for channel and network parameters (addresses, panid...)
    - startReceive: listens the channel (in one of the many receiption mode)
    - transmit: begin to transmit a pdu

TRANSMISSION & RECEPTION

- These operations require much attention:
    - The radio permits to transmit every type of pdu, but it's possible to receive only packets with 802.15.4 well formed headers
    - It's also possible to receive in promiscuous mode to sniff for every packet, but this exposes to interferences
- Each mote mantains 3 addresses:
    - a 16-bit PAN identifier
    - a 64-bit extended address that uniquely identifies a mote
    - a 16-bit short address that's application and protocol specific

```
   1     1  |   1    |   0/2   |  0/2/8   |   0/2  |  0/2/8   | 0/5/6/10/14 |      ...      | # of bytes per field
+-----+-----+-------+---------+---------+--------+---------+-------------+---------------+
| FCF | FCA | SEQNO | DSTPAN  | DSTADDR | SRCPAN | SRCADDR | aux.security |    payload    |  field name
+-----+-----+-------+---------+---------+--------+---------+-------------+---------------+
|<-----------------addressing fields--------------------->|
```

Figure 1: PDU header format

```
INDEX 0 (byte index in frame)
    0       1       2       3       4       5       6       7        bits
+-------+-------+-------+-------+-------+-------+-------+-------+
|          TYPE          |  SEC  | PEND  | ACKRQ | NSPID |  RFU  |   field names
+-------+-------+-------+-------+-------+-------+-------+-------+
|<------------------------------ FCF ----------------------->|
```

Figure 2: Frame Control Flags

```
INDEX 1 (byte index in frame)
  0   1   2   3   4   5   6   7      bits
+---+---+---+---+---+---+---+---+
|  RFU  |  DST  |  FRV  |  SRC  |    field names
+---+---+---+---+---+---+---+---+
|<------------ FCA ------------>|
```

Figure 3: Frame Control Address Flags

TX/RX REAL AND REAL TIME CONSTRAINTS

- It's possible to operate in many different ways with regards to real time constraints:
  - It's possible to receive/transmit ASAP (As Soon As Possible) or EXACTLY at the specified time or …
  - Rx/Tx require a start operation time and an end one
  - MR manages autonomously all warm up and ramp up to make the device ready at the specified time
  - The device turn off at the end and an event is raised to be managed with delegation
  - If the device cannot be ready at the specified time or an error occurs an error reports this status

A MAC LAYER IN MOTE RUNNER

COMING SOON

...