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

May 12, 2015

.

INTRODUCTION TO 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
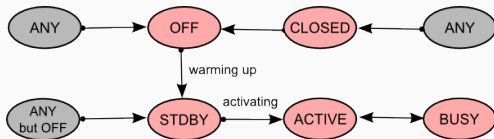- It's still in beta and is evolving towards IoT

- Mote Runner system provides:
  - A Virtual Machine for executing byte codes
  - An Operating System for:
    - organizing access to different devices
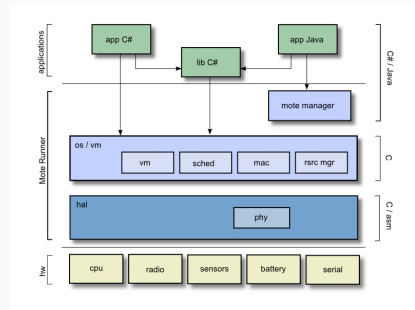   - scheduling the various activities
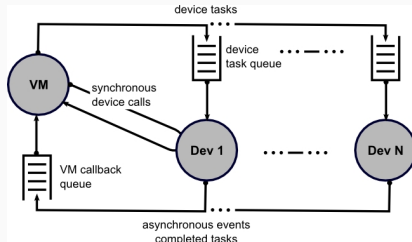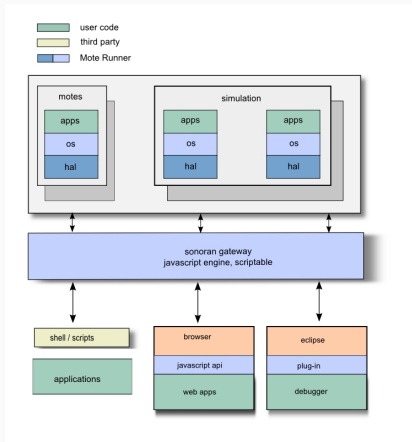
· The OS assumes that all devices have the following
  states

· The OS manage implicitely most of the state changes:
  · Makes sure that the device ramp up happens before the
    requested time
  · Keeps device in states with the lowest energy consumption
  · Application, however, can put devices into the states
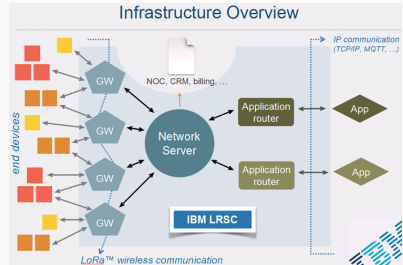   CLOSED, OFF and STDBY

- 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

- 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)

- 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

Infrastructure Overview

IP communication (TCP/IP, MQTT, …)

end devices

GW

GW

GW

GW

NOC, CRM, billing, …

Network Server

Application router

Application router

App

App

IBM LRSC

LoRa™ wireless communication

IBM

- The Long Range Signaling and Control (LRSC) system is a network infrastructure which relies on LoRa™, modulation technology developed by Semtech for wireless bidirectional communication over distances of up to 15 km in semi-rural environments and up to five km in dense urban environments.
- All communication is generally bi-directional, although uplink communication from end devices to the network server is strongly favored, and is based on LoRa.

The Mote Runner SDK ships with:

- LoRa Mac library providing an API for accessing a LRSC network.
- LIP shell interface to control the Mac from the Mote Runner shell MRSH.

The main constraint for our intial purpose depends on the fact that the end devices cannot communicate directly. Any message should be sent over the LRSC network.

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

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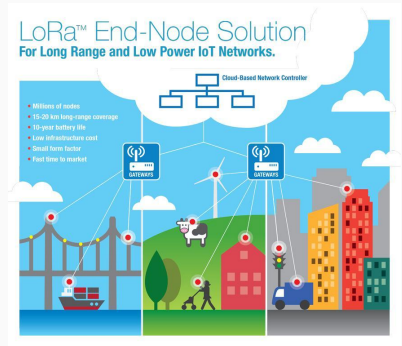
- LoRa™Technology
  - LoRaWAN pledeges to extend the radio range by 10x while using only one third of the power used by competing solutions
  - Star (of stars) topology
  - Gateways relay messages between end-devices and a central network server
  - Communication between end-devices and gateways is spread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

- …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/

- · 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

.

TESTING MOTE RUNNER

- 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)

- 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 reception mode)
    - transmit: begin to transmit a pdu

- 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

Figure 1: PDU header format
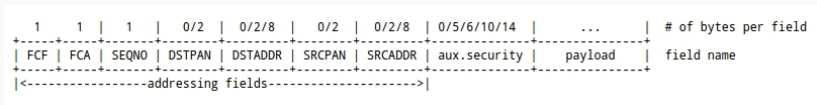

Figure 2: Frame Control Flags
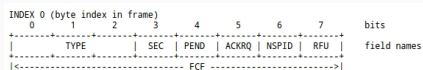

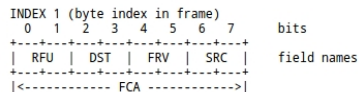Figure 3: Frame Control Address Flags

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

...