REALIZATION OF AN 802.15.4-LIKE MAC LAYER 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

July 21, 2015

INTRODUCTION

- Object of this project is the exploration of Mote Runner, an IBM's infrastructure platform for WSN
- For a deep understanding of MR the focus of this works was the design and develop of a 802.15.4-like MAC layer
- Oscilloscope is an applications developed to test the MAC layer
- The application was tested on IRIS mote

INDEX

- Introduction to Mote Runner
- LoRaWAN
- Physical Layer in Mote Runner
- A MAC Layer in Mote Runner
- 6LoWPAN implementation in Mote Runner

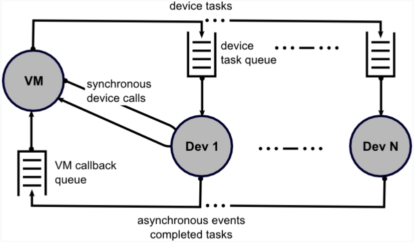
INTRODUCTION TO MOTE RUNNER

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

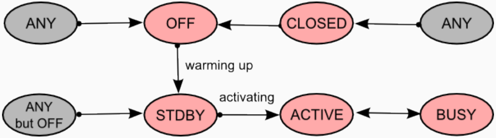
MOTE RUNNER OPERATING SYSTEM

- Mote Runner system provides:
  - A Virtual Machine for executing byte codes
  - An Operating System for:
    - organizing access to different devices
    - scheduling the various activities



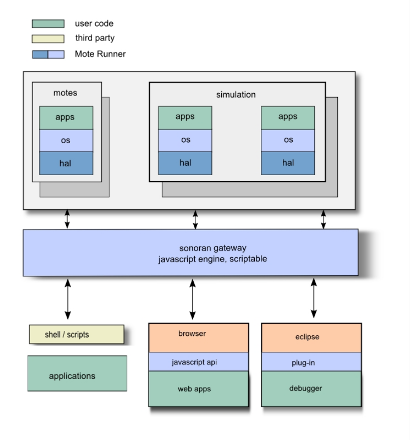
DEVICE MODEL

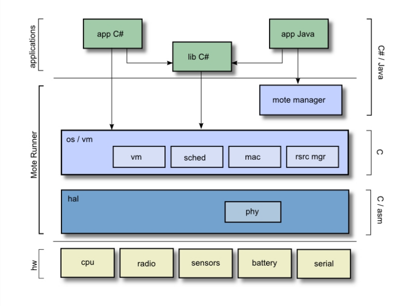
- The OS assumes that all devices have the following states



- 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

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

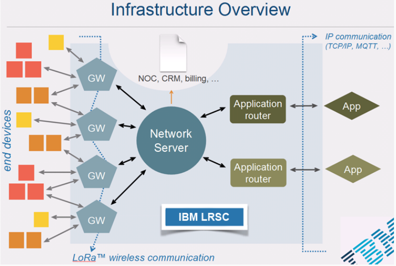
MOTE RUNNER - V.11, V.13 BETA

- 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

- 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



IBM LRSC

- 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.

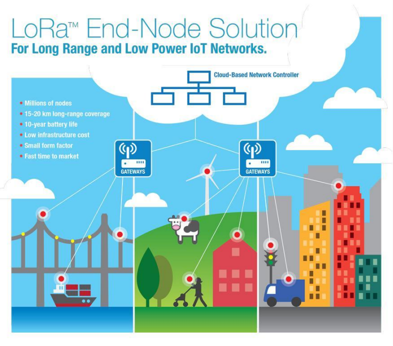
MOTE RUNNER - CONCLUSION

- For the purpose of this work:
  - 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
    - this MAC could be possibly used to build upper layer with WIDS
- This does not exclude a future integration with LoRa-LRSC

LORAWAN

LORA™

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



- LoRaWAN is a Low Power Wide Area Network (LPWAN) specification intended for wireless battery operated Things in regional, national or global network.
- LoRaWAN target key requirements of internet of things:
  - secure bi-directional communication
  - mobility
  - localization services

For time synchronization gateways periodically broadcast so-called beacons. Each beacon minimally contains:

- network identifier (NetID)
- current GPS time (Time)

The broadcasting of beacons (in implicit mode) is done time-synchronously (BEACON_INTERVAL) by all gateways of a network with “no interference”

PHY	Preamble	BCNPayload

Size (bytes)	3	4	1/2	7	0/1	2
BCNPayload	NetID	Time	CRC	GwSpecific	RFU	CRC

All gateways send their beacon at exactly the same point in time:

- on first 15 bytes there are no visible on-air collisions
- wrt the optional part , device within the proximity of more than one gateway will still be able to decode the strongest beacon with high probability

Size (bytes)	1	6
GwSpecific	InfoDesc	Info

Size (bytes)	3	3
Info	Lat	Lng

Figure 1: Beacon optional part

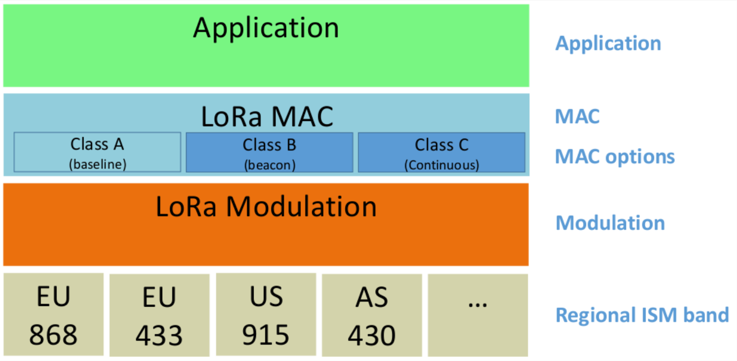
LORA MAC PAYLOAD FRAME

Octets	1	1..59	4
PHYPayload	MHDR	MACPayload	MIC

Bit#	7..5	4..2	1..0
MHDR	MType	RFU	Major

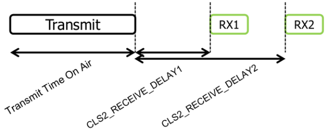
MType	Description
000	Join Request
001	Join Accept
010	Data Unconfirmed
011	Data Confirmed
100..110	RFU
111	Proprietary

LORAWAN CLASSES



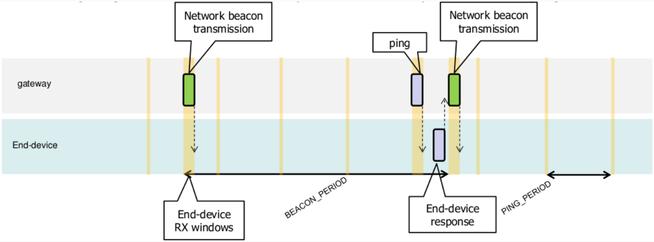
CLASS A END-DEVICES

- Its funtionalities are implemented by every end-device.
- Uplink: devices transmit following Aloha method.
- Downlink: after a transmission two tiny time windows are opened to allow reception
  - RX1 uses the same frequency channel as the uplink and a data rate depending on the one in the uplink;
  - RX2 uses a fixed configurable frequency and data rate;
  - Devices are active in rx only if a preamble is detected.



CLASS B END-DEVICES

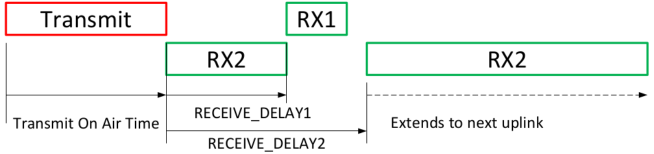
- Class B end-devices are optimized for mobile and fixed battery-powered end-devices.
- They add a synchronized reception window called “ping slot”
  - Synchronization requires beacons;
  - Devices selects randomly a ping slot at each beacon to avoid collisions.



CLASS B END-DEVICES

- All end-devices start and join the network as end-devices of Class A.
- The end-device application can then decide to switch to Class B.
- The end-device waits a beacon and selects a ping slot of 30 ms from the 4096 available in a beacon interval.
- When the mote is far from BS the duration is extended, if beacon is not received the device tries to mantain the synchronization for 2 hours after that it returns to class A

- This mode is used when there are no need for energy awareness and there’s no need to minimize reception time.
- Class C end-devices cannot implement Class B option.
- These devices will listen with RX2 windows parameters as often as possible.



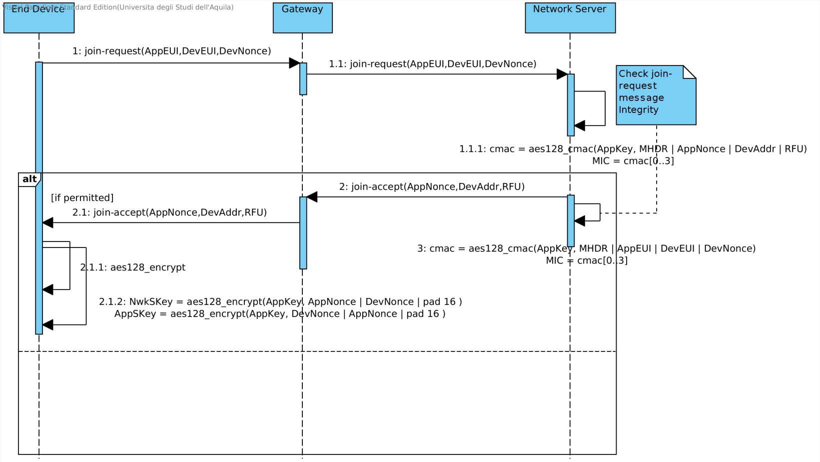
LORA SECURE COMMUNICATIONS - 1

In order to partecipate in a LoRa network an end device first has to be personalized and then activated.
Activation of an end device can be achieved in two ways:

- OTAA (over-the-air activation) when an end device is deployed or reset;
- APB (activation by personalization) one-step personalization and activation.

During activation the end device holds the following informations:

- **DevAddr:** device ID of 32 bits that uniquely identifies the end device.
- **AppEUI:** globally unique application ID that uniquely identifies the application provider of the end device.
- **NwkSKey:** device-specific network session key, ensures data integrity and os used to encrypt/decrypt MAC data messages payload
- **AppSKey:** device-specific application session key, used to encrypt and decrypt the payload field of application-specific data messages.


Visual Paradigm Standard Edition(Universita degli Studi dell'Aquila)
End Device
Gateway
Network Server
1: join-request(AppEUI,DevEUI,DevNonce)
1.1: join-request(AppEUI,DevEUI,DevNonce)
Check join-request message Integrity
1.1.1: cmac = aes128_cmac(AppKey, MHDR | AppNonce | DevAddr | RFU)
MIC = cmac[0..3]
alt
2: join-accept(AppNonce,DevAddr,RFU)
[if permitted]
2.1: join-accept(AppNonce,DevAddr,RFU)
3: cmac = aes128_cmac(AppKey, MHDR | AppEUI | DevEUI | DevNonce)
MIC = cmac[0..3]
2.1.1: aes128_encrypt
2.1.2: NwkSKey = aes128_encrypt(AppKey, AppNonce | DevNonce | pad 16)
AppSKey = aes128_encrypt(AppKey, DevNonce | AppNonce | pad 16)

LoRa network data rates are:

- Network controlled for fixed devices by means of using ADR bit in the PHY payload of data messages:
  - If is set, the network will control the data rate of the end device through the appropriate MAC commands
  - If is cleared, the network will not attempt to control the data rate of the end device independently of the received signal quality
- default for mobile end-devices.

Octets	4	1	2	0..15
FHDR	DevAddr	FCtrl	FCnt	FOpts

Bit#	7	6	5	4	3..0
FCtrl	ADR	ADRACKReq	ACK	FPending	FOptsLen

LoRa modes (868Mhz band):

Mode	BW	CR	SF	Sensitivity (dB)	Transmission time (ms) for a 100-byte packet sent	Transmission time (ms) for a 100-byte packet sent and ACK received	Comments
1	125	4/5	12	-134	4245	5781	max range, slow data rate
2	250	4/5	12	-131	2193	3287	-
3	125	4/5	10	-129	1208	2120	-
4	500	4/5	12	-128	1167	2040	-
5	250	4/5	10	-126	674	1457	-
6	500	4/5	11	-125,5	715	1499	-
7	250	4/5	9	-123	428	1145	-
8	500	4/5	9	-120	284	970	-
9	500	4/5	8	-117	220	890	-
10	500	4/5	7	-114	186	848	min range, fast data rate, minimum battery impact

Figure 2: SX1272 module modes

For what applications is LoRa a good option?

- solar or mains-powered nodes transmitting every 10 or 15 minutes in networks with low or medium number of nodes
- very wide networks, with long-range links (Up to 22km, Sensitivity -134dBm)

LORA CONCLUSIONS 2

For what applications is NOT LoRa a good option?

- projects which require high data-rate and/or very frequent transmissions (e.g., each 10 seconds)
- including receipt of ACK message, mode 10 (the fastest), takes twice the time of XBee (<200ms)
- due to low data-rates OTA re-programming is not easily achieved (3G, GPRS may be better choices)

PHYSICAL LAYER IN MOTE RUNNER

RADIO INTERFACE

- MR v.13 offers a Radio interface IEEE 802.15.4 compliant
  - It is a generic class in the IBM saguaro system that permits to use the radio device
  - It offers an 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

RADIO INTERFACE

- In addition Radio:
  - Handles transmission and reception notifying to higher level by delegation, registering functions that will handle the events (tx and rx)
  - Manages acks notifying states of failure or success to callbacks
  - Permits to set parameters as PAN identifier, short address, radio channel

transmission & reception

- These operations require much attention:
  - Radio permits to transmit every type of pdu, but it's possible to receive only packets with 802.15.4 well formed headers
  - Receiving in promiscuous mode allows to receive every kind of packet, but this exposes to interferences
- Each mote holds 3 addresses:
  - a 16-bit PAN identifier
  - a 64-bit extended address that uniquely identifies a mote (EUI-64)
  - 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 3: 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 4: 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 5: Frame Control Address Flags

TX/RX AND REAL TIME CONSTRAINTS

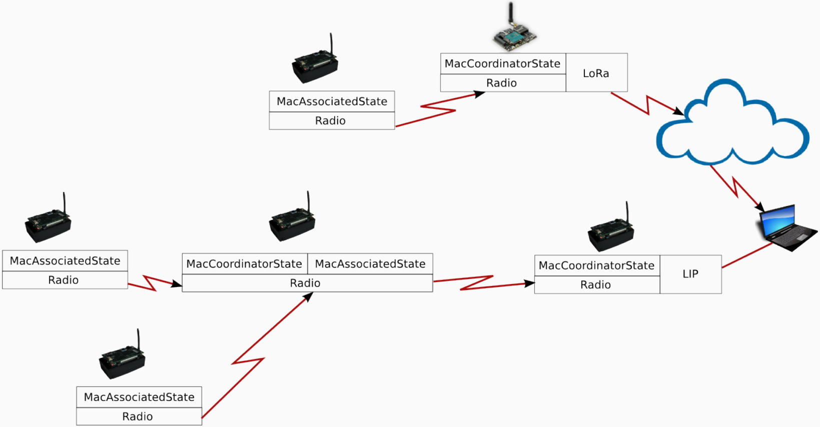
- It's possible to operate in many different ways with regards to real time constraints:
  - ASAP, EXACT, TIMED, RX4EVER indicate when the operation should begin and/or end given two time instant
  - MR manages autonomously all warm up and ramp up to make the device ready given one of these modes
  - At the end the device turns off and an event is raised to be managed with delegation
  - If the device cannot be ready or cannot complete a task within the specified time an error occurs

A MAC LAYER IN MOTE RUNNER

DESIGN STRATEGY

- Beacon enabled, A-TDM with CSMA/CA
- Mac class behaviours:
  - Coordinator -> Sends beacons and handles request and data frames from motes
  - Unassociated -> Tries to associate with a Coordinator
  - Associated -> Sends data from upper layer and receives data from Coordinator
- Focus on flexibility:
  - State changes have to be ruled by Mac class through events
  - Mac should handle more than one state -> Mac - entities
    - e.g.: a single mote acting as Coordinator and Associated

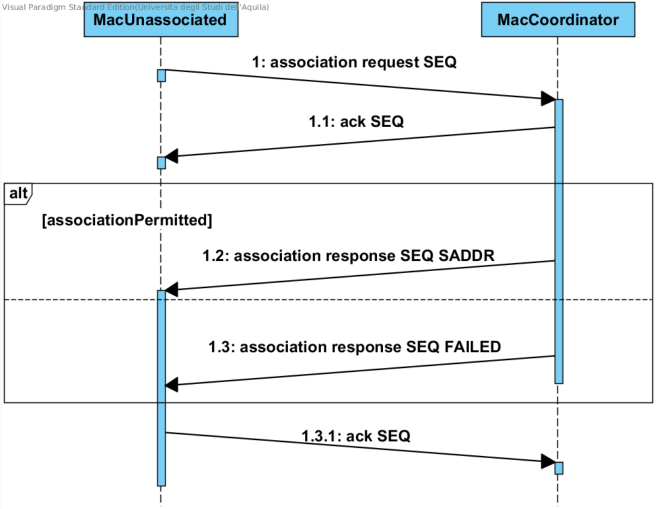
THE NETWORK'S CONCEPT



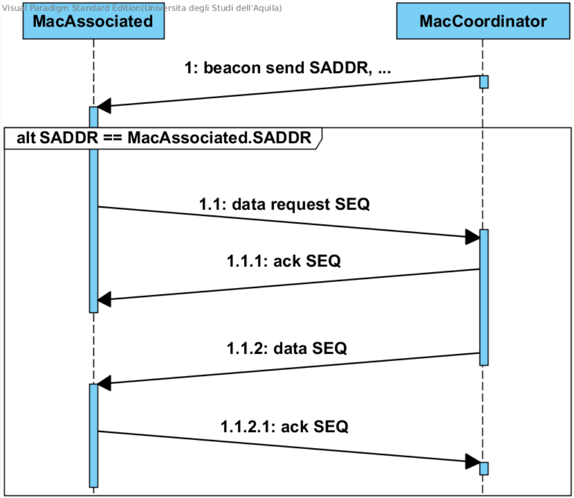
ABOUT THE NETWORK

- Motes are subdivided into PANs
  - Every PAN has a PAN ID
  - Every mote has a unique short address (SADDR) inside the PAN
- To obtain the SADDR the mote must associate with the PAN coordinator
- To grant communication between motes synchronization is crucial
  - Beacon + Superframe
- The adopted procedures follow 802.15.4 standard

ASSOCIATION

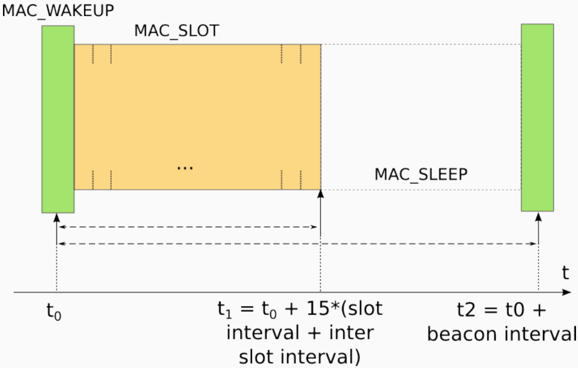


DATA INDIRECT



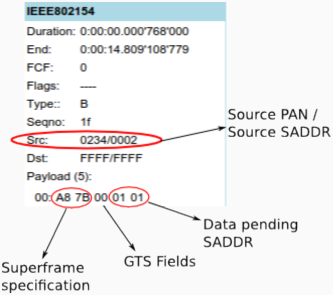
TIMING WITH BEACON

- Grants synchronization between mote and coordinator
- Realized with a timer and scheduled events



BEACON

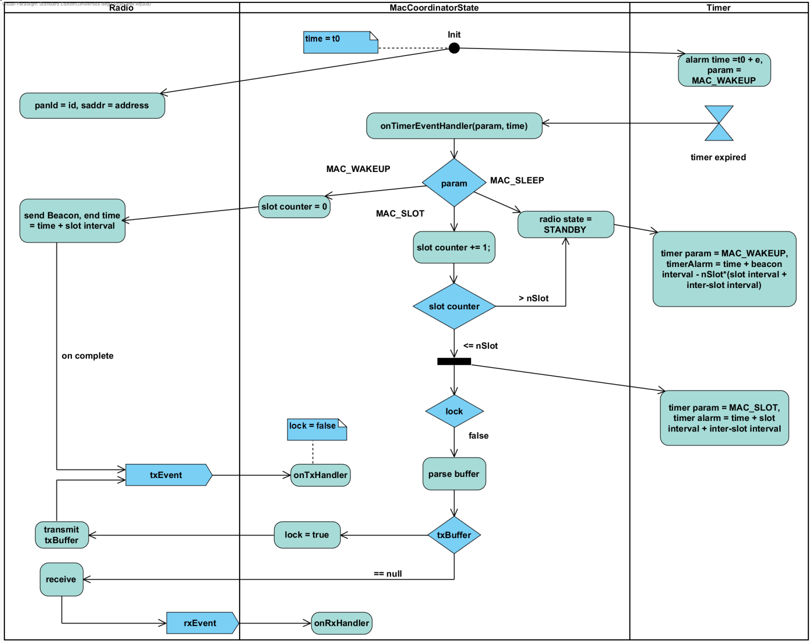
- Superframe Specification:
  - Beacon Order -> BO
  - Superframe Order -> SO
  - Association permitted



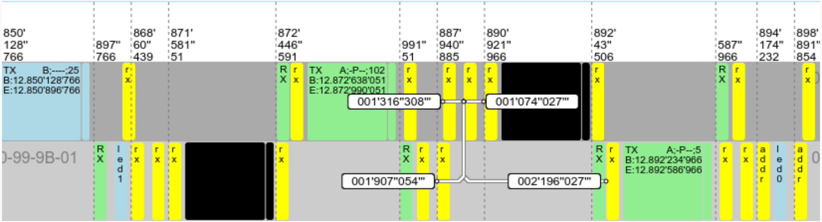
$$\text{Beacon Interval} = \frac{60sym \cdot n.Slot \cdot 2^{BO}}{20\text{kbps}}$$

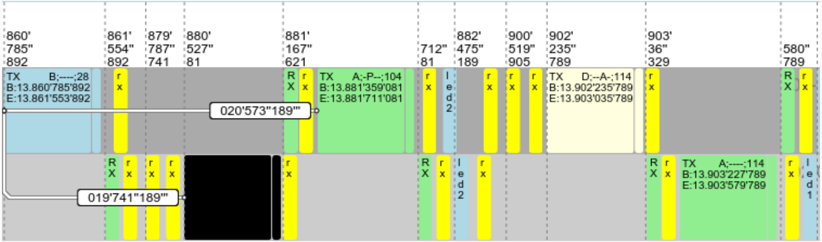
$$\text{Superframe Duration} = \frac{60sym \cdot n.Slot \cdot 2^{SO}}{20\text{kbps}}$$

MAC COORDINATOR BEHAVIOUR

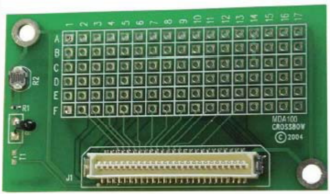


The node associates with coordinator, then responds to beacon pending list and gets data.





- Periodically reads values of TEMPERATURE and LIGHT
- Read interval and type can be setted by master
- Readings are sent through MAC once associated to master
- Readings done by MDA100 board



MASTER OSCILLOSCOPE

- It creates a PAN with the MAC layer
- It listens LIP for commands that sends to associated motes
- MAC layer sends readings to Master Oscilloscope that are redirected through LIP
- A JavaScript Socket running on Sonoran process displays the readings


live
DEMO

6LoWPAN implementation in Mote Runner

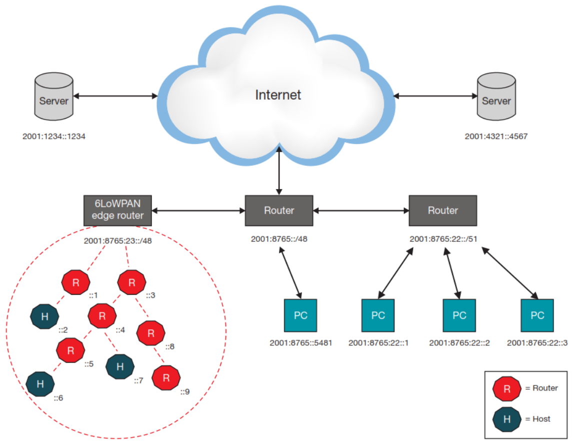
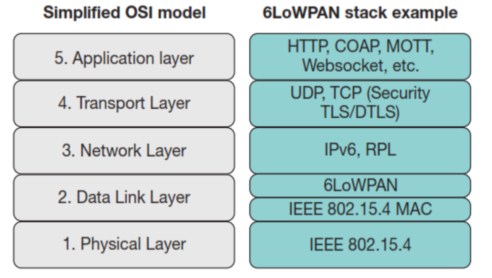


Figure 6: IPv6 network with a 6LoWPAN mesh network

6LOWPAN STACK



MRV6: AN IMPLEMENTATION OF 6LOWPAN IN MR

- TDMA and beacon based multi-hop network which allows for an IPv6 based communication between motes
- It is not a built-in MR component, but it is fully implemented in C#
- Datagram packets exchanged adheres to a subset of the 6LoWPAN specifications
- The edge mote decides upon:
  - Association requests
  - Assigns communication schedules between wireless nodes
  - Determines the routes in the network

MRV6: LIMITATIONS

- Only the transmission of UDP packets within the 6LoWPAN network is supported
- Exists only a proprietary broadcast operation to reach all motes in the network
- Is not suited for low latency application
- Does not support packet segmentation, reassembly and flow control
- Has been deployed in 900MHz or 2.4GHz frequency ranges and uses a single channel in the 2.4GHz band yet

SCHEDULING

- The network tree is only know to the edge
- The communication slots between parent and children are globally assigned by the edge and do never overlap

SF edge	SF mote 1	SF mote 2	…	SF max mote

SUPERFRAME

- At the beginning of their communication period parent motes send out beacon messages
- Other then a fixed exclusive slot, parent offers a shared slot (e.g. association requests and responses, broadcast messages)
- Beacon, shared and fixed slots form the superframe whose timings are assigned by the edge

Beacon slot	Shared slot	Child slot 0	...	Child slot n	Gap

ASSOCIATION

- The joining mote evaluates information in the beacons (e.g. number of children or hops to the edge)
- Mote sends and association request in the shared slot of the potential parent
- Parent forward the request to the edge with the EUI-64 of the joining mote
- When the edge accepts the mote:
    - It allocates short address and superframe timings for the mote
    - The parent forwards the response to the new child in its shared slot and adds it to its list

SHORT COMPARISON

Our Mac-Like	MRv6
Contention based	Scheduled
Trasmitt only when requested	TDMA based
Association managed by pan-coordinator	Association managed by the edge
...	...

CONCLUSION

FINAL CONSIDERATIONS ABOUT MOTE RUNNER

- Pro:
  - Good simulation environment
  - It allows to develop mote applications in high-level object-oriented languages
  - Good assumptions and expectations with respect to LoRa
- Con:
  - It's still in beta
  - Low support (docs, API, ...)
  - Not open source (till now)

MAC STATUS

- Only three states are supported, there are others (orphan,..)
- Superframe parameters can't be dinamically changed
- In the associated state motes are not energy aware
- Some funtionality have not been implemented (e.g. disassociation, channels scan,...)
- Transmission from PAN-C to devices seems to require EUI64, but SADDRs are more suited


Diolch
Kiitos
Sheun
umesc
Kasih
Mamnoon
Todah
Dekuju/Dekujeme
Shnorhakalutiun
Te°ekkür
Hvala
Shokriya
Dziękuję
Shokran
Spaas
Mul
Ači
Gamsahapnida
Ngiyabonga
Dank
Takk
Cam
Gra
or
Dhanyavaad
Xie
Dhanyavaadalu
Waad
Kop
Salamat
Merci
al
Dankie
Go
Grazie
Faleminderit
Dakujem
Daw
Dhanyavad
Khopjai
Kruthagnathalu
krap
Thank You
kun
Arigatou
Tack
Shukriya
Or
Dhonnobaad
ederim
Asante
Grazzi
raibh
Gracias
Nandree
Blagodariya
Gomapsupnida
Euxaristo
Kun
Hain
Dhan
Fyrir
Terima
Enkosi
Danke
dank
daa