Application of Contiki on 6LoWPAN Mesh Wireless Sensor Network and Cloud Connectivity

Yukhe Lavinia

Overview

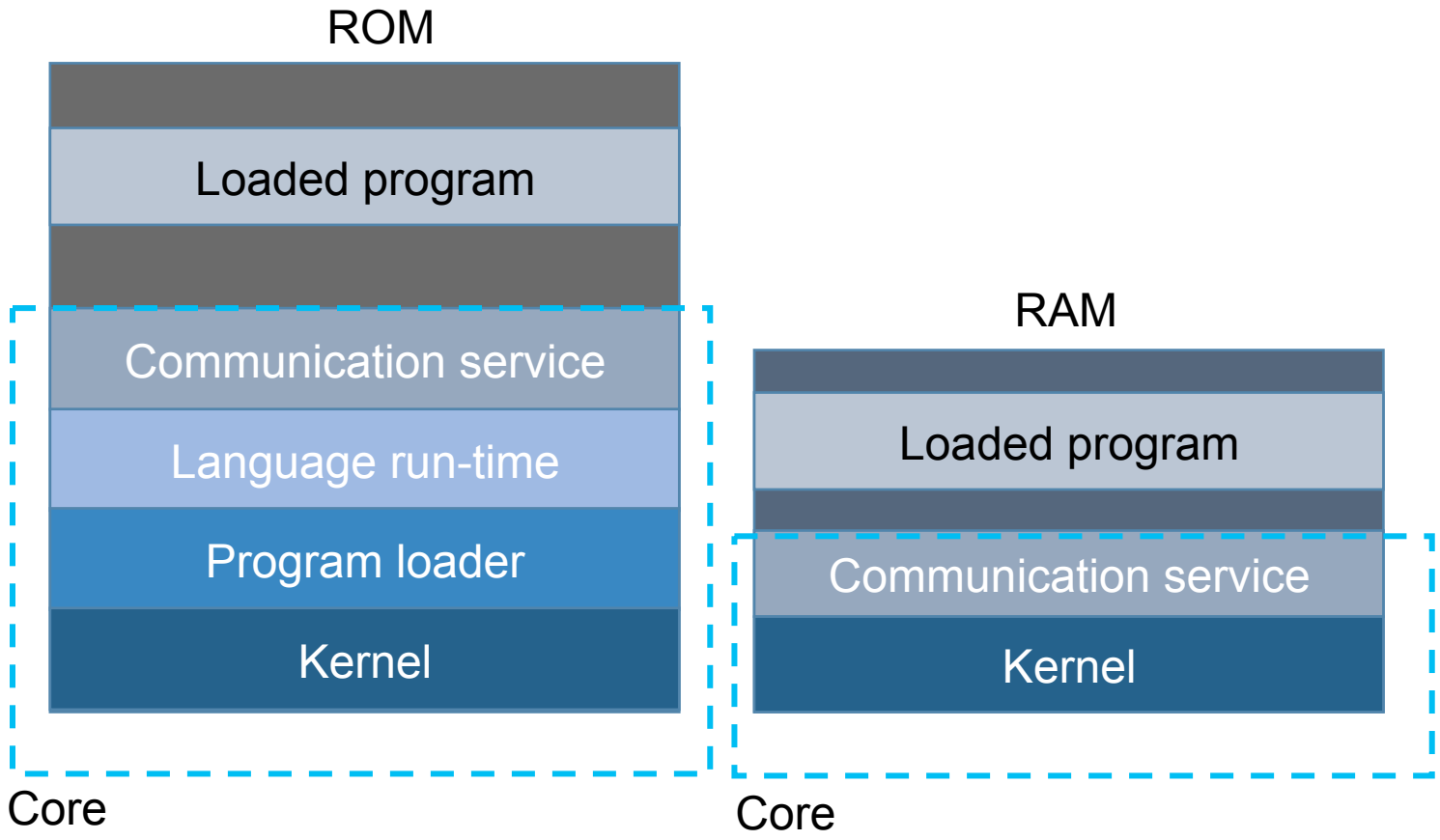
Introduction
Contiki
IPv6
6LoWPAN
RPL
Border Router
MQTT
System Architecture
Tools

Introduction

- Internet of Things (IoT)
- Billions of connected devices by 2050
- Lightweight OS
- Contiki as one of the most widely used

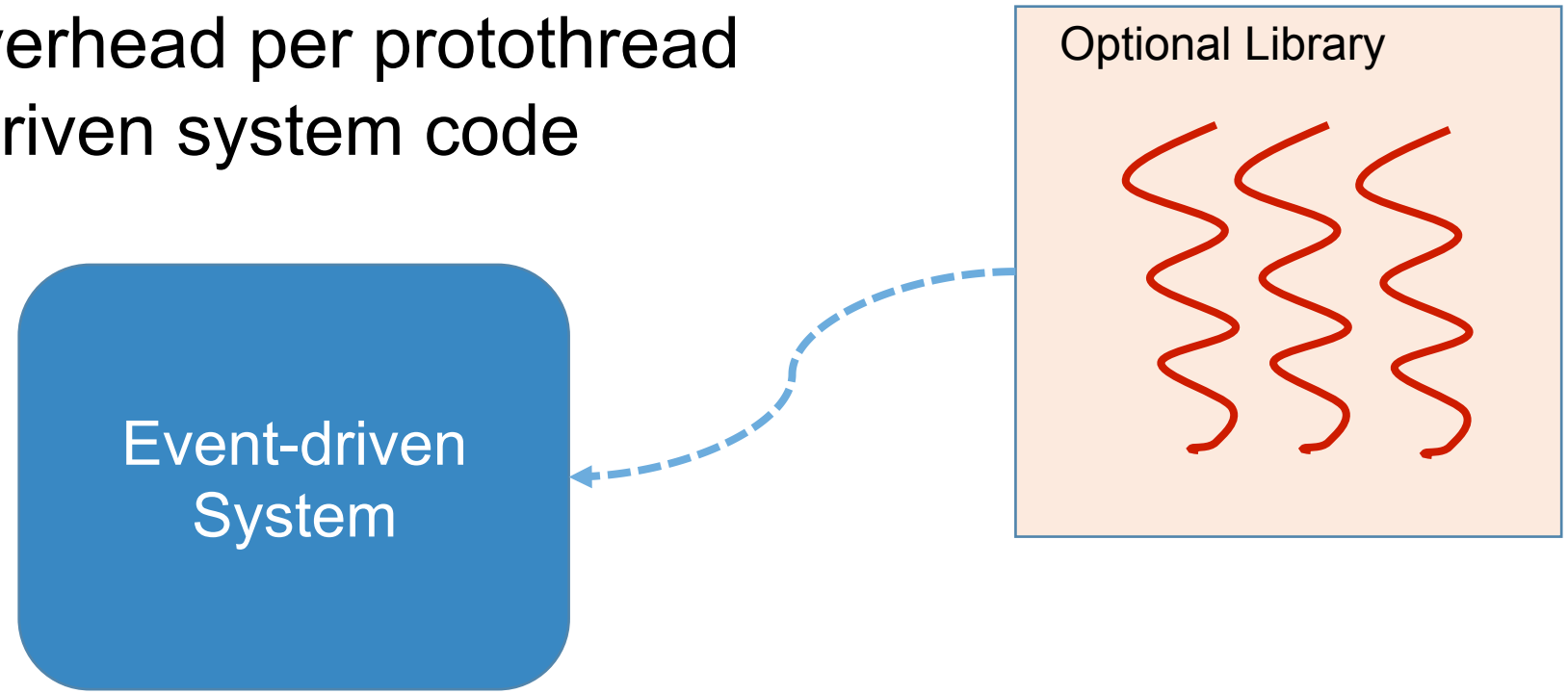
Contiki

- Consists of
  - Kernel
  - Basic libraries
  - Program loader
  - Processes
    - Application programs
    - Services
- Partitioned into two:
  - Core
  - Loaded program



Contiki: Protothread

A hybrid of event-driven system and multi-threads
Implemented as optional library
2 KB of memory overhead per protothread
Only 2/3 of event-driven system code



Contiki: Events vs Protothread

```
state: {ON, WAITING, OFF}

radio_wake_eventhandler:
  if (state = ON)
    if (expired(timer))
      timer ← t_sleep
      if (not communication_complete())
        state ← WAITING
        wait_timer ← t_wait_max
      else
        radio_off()
        state ← OFF
  elseif (state = WAITING)
    if (communication_complete() or
        expired(wait_timer))
      state ← OFF
      radio_off()
  elseif (state = OFF)
    if (expired(timer))
      radio_on()
      state ← ON
      timer ← t_awake
```

Events, in pseudocode

```
radio_wake_protothread:
  PT_BEGIN
  while (true)
    radio_on()
    timer ← t_awake
    PT_WAIT_UNTIL(expired(timer))
    timer ← t_sleep
    if (not communication_complete())
      wait_timer ← t_wait_max
      PT_WAIT_UNTIL(communication_complete() or
                    expired(wait_timer))
    radio_off()
    PT_WAIT_UNTIL(expired(timer))
  PT_END
```

Protothreads, in pseudocode

Contiki: Threads vs Protothread

Multiple stacks	One stack
Implicit blocking	Explicit blocking
Preemption	No preemption
Automatic variables saved	No automatic variables saved

Source: Dunkels et al. (2005)

IPv6

- Next generation of IPv4
- 128 bits
- 2^{128} distinct addresses
- Example: 2001:0ea4:0000:0000:0289:0000:14a7:37bd

IPv6: Rules

- Suppress leading zeros
  - Before: 2001:0ea4:0000:0000:0289:0000:14a7:37bd
  - After: 2001:ea4:0000:0000:289:0000:14a7:37bd
- Replace two or more contiguous zero blocks with double colon
  - Before: 2001:ea4:0000:0000:289:0000:14a7:37bd
  - After: 2001:ea4::289:0000:14a7:37bd
- Replace a single zero block with zero
  - Before: 2001:ea4::289:0000:14a7:37bd
  - After: 2001:ea4::289:0:14a7:37bd
- Lowercase a, b, c, d, e, f in hexadecimal

IPv6: IPv4 Representation

- Example: 192.168.0.11
  i. x:x:x:x:x:x:192.168.0.11
  ii. 0:0:0:0:0:0:192.168.0.11
  iii. ::192.168.0.11
  iv. ::c0a8:b (equivalent to 192.168.0.11 in hexadecimal)
- Note: we will use the hexadecimal equivalent of the IPv4 (representation iv) in implementation

IPv6: Special Address

0000 0000 …… 0000	ffff	IPv4 address
80 bits	16 bits	32 bits

Combining ::ffff/96 prefix with the IPv6 representation iv of IPv4 in the previous slide:

0000 0000 …… 0000	ffff	c0a8:b
80 bits	16 bits	32 bits

IPv6: Neighbor Discovery (ND)

Router discovery

Prefix discovery

Parameter discovery

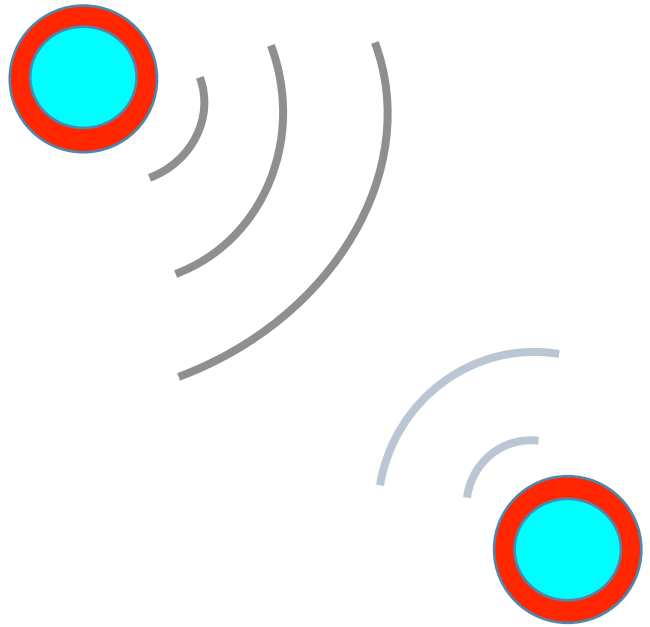
Address autoconfiguration

Address resolution

Next-hop determination

Neighbor unreachability detection

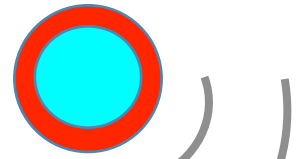
Duplicate address detection



IPv6: ND ICMPv6

Router Solicitation

to discover other IPv6 routers

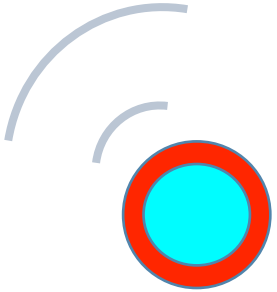


Router Advertisement

to respond to Router Solicitation

Neighbor Solicitation

to discover link layer address of IPv6 neighbor router



Neighbor Advertisement

to respond to Neighbor Solicitation

Redirection

to inform packet originator that there is a better first hop

IPv6: uIP

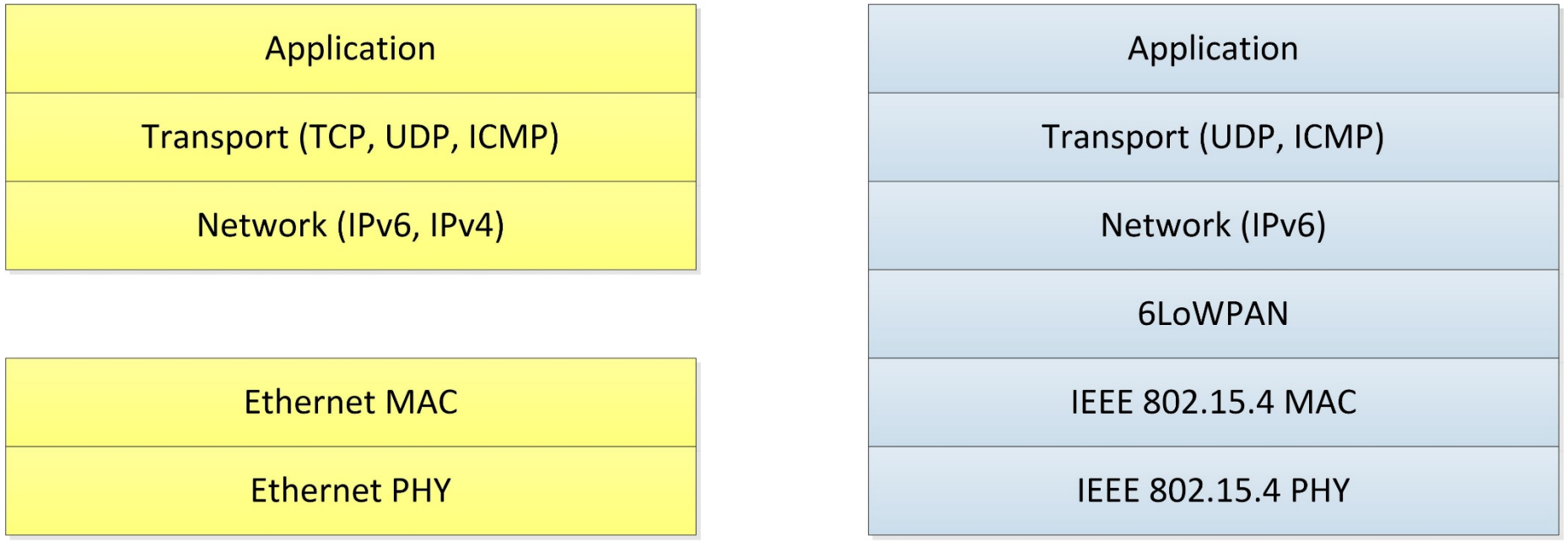
- Read: micro IP
- In IPv6: uIPv6
- IP network stack for low power devices
- Support UDP, TCP, ICMP, HTTP

uIPv6:

- less than 2 KB of RAM
- ~11.5 KB of ROM
- Doesn’t keep copies of IP packets sent/received

6LoWPAN

IPv6 over Low power Wireless Personal Area Network
Adaptation layer between network and link layers



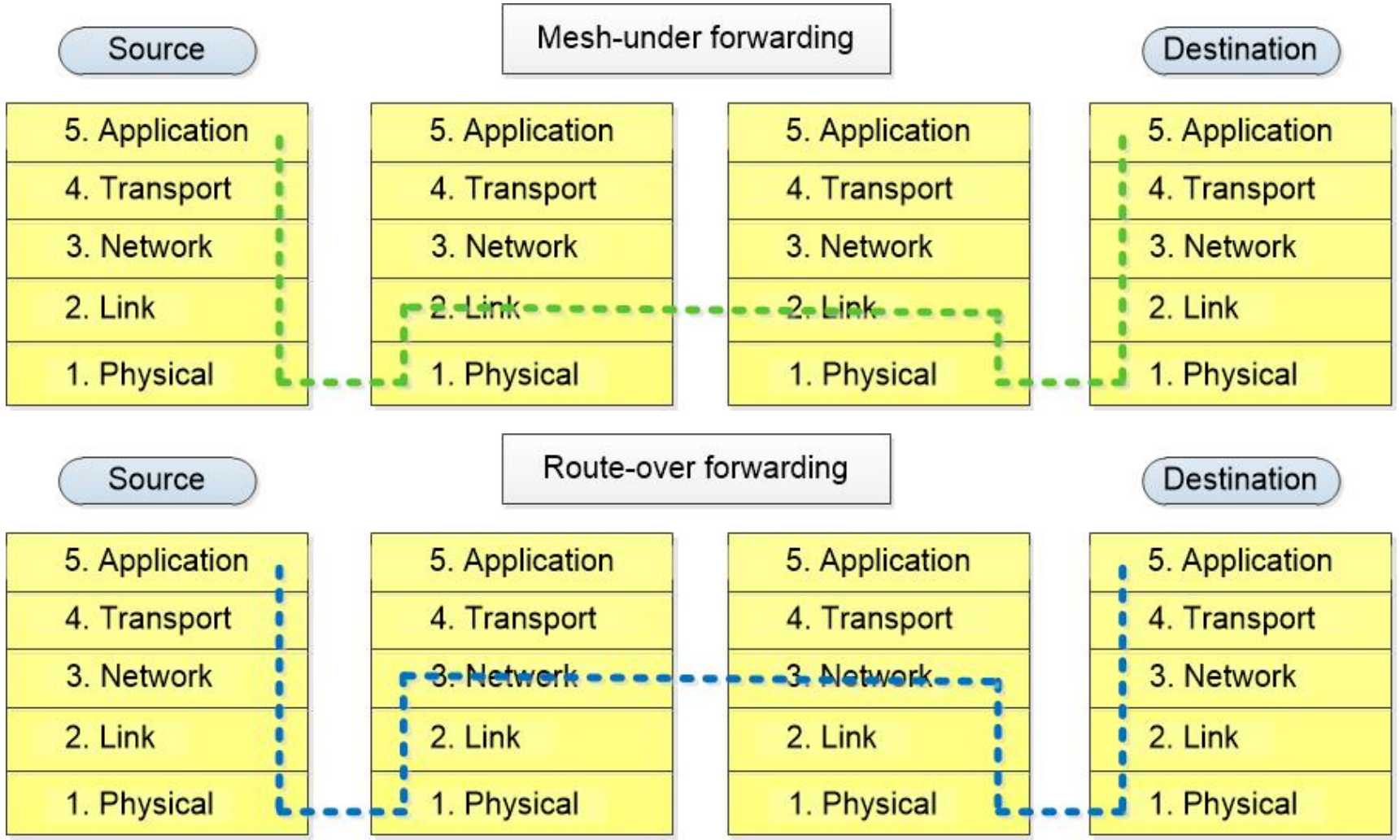
6LoWPAN

- IPv6 minimum MTU: 1280 bytes (layer 3)
- IEEE 802.15.4 link MTU: 127 bytes (layer 2)
- Need adaptation
- 3 services
- Packet fragmentation and reassembly
- Network and link forwarding
- Header compression

6LoWPAN: Fragmentation & Reassembly

- Fit bigger packet (1280 bytes) into smaller frame (127 bytes)
- Additional headers to track fragments
- Fragmentation sequence depends on routing type

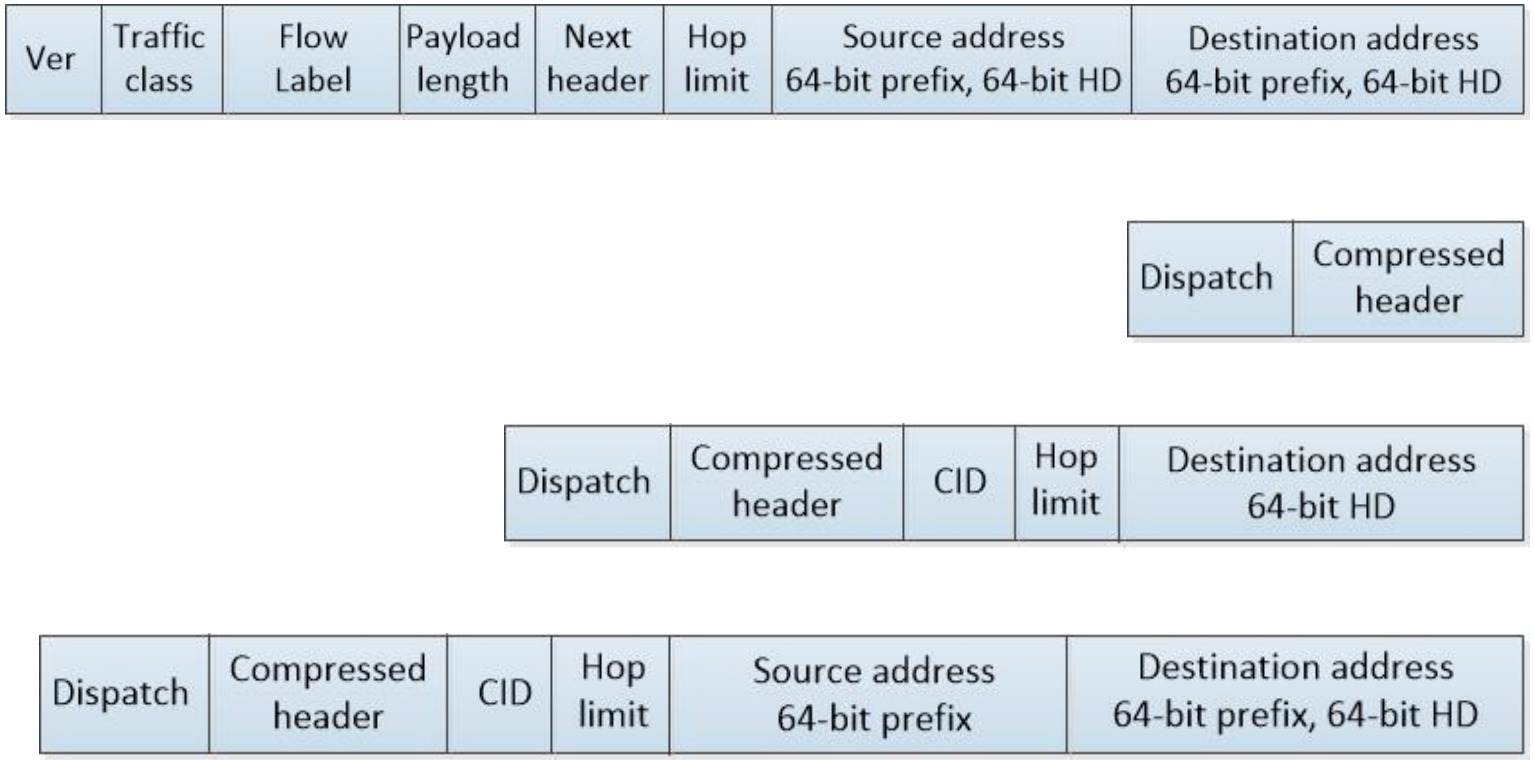
6LoWPAN: Routing Types



Reassembly at final destination

Reassembly at each hop

6LoWPAN: Header Compression



6LoWPAN: Header Formats

Mesh addressing header

Used when there is more than one hop

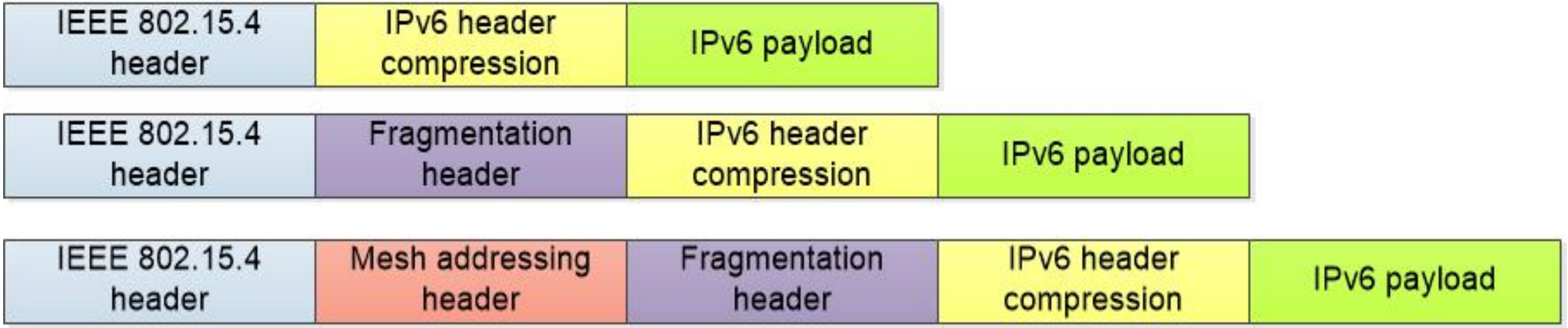
3 fields: hop limit, source and destination addresses

Fragmentation header

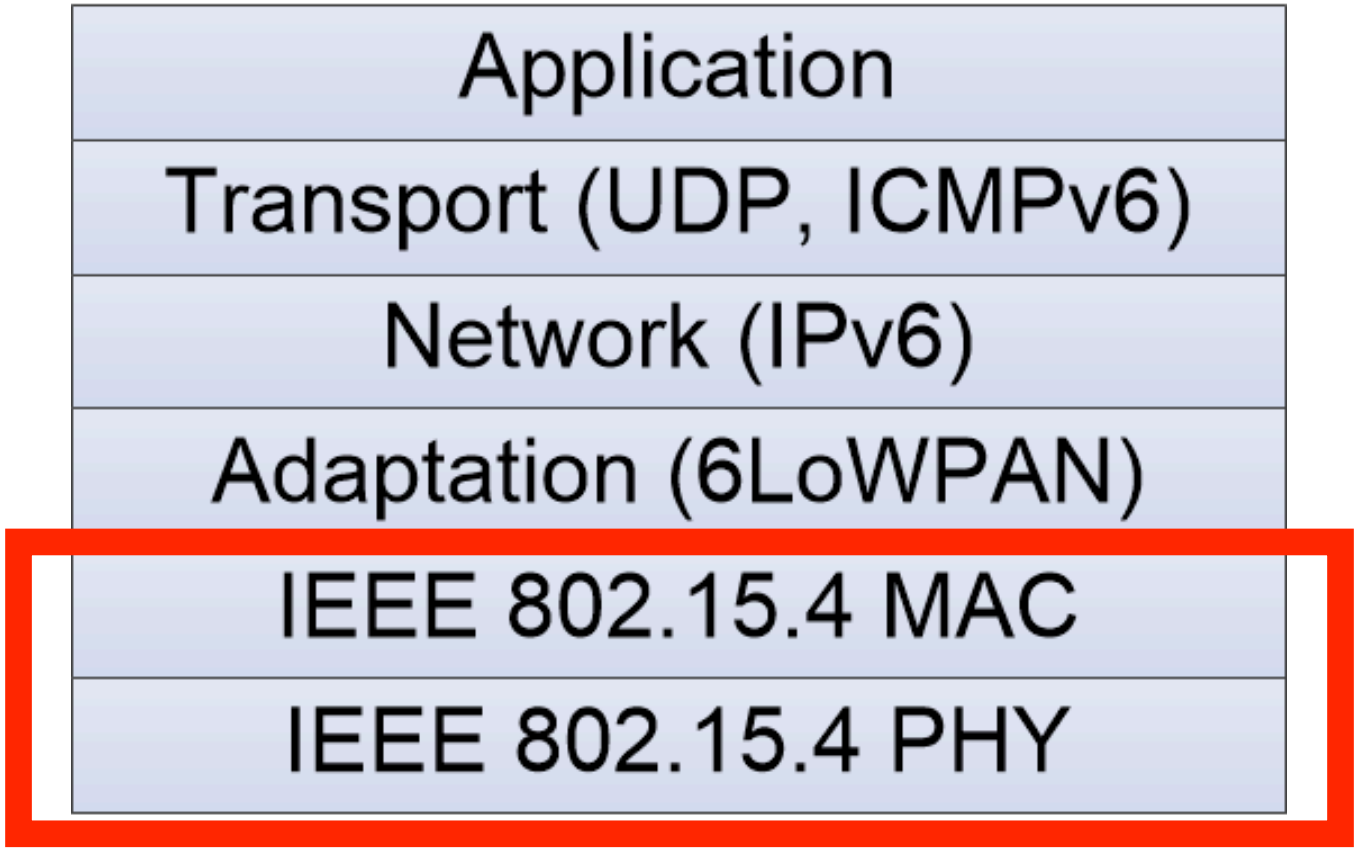
When need to fit bigger packet into smaller frame

3 fields: datagram size, datagram tag, datagram offset

6LoWPAN: How Headers Stacked

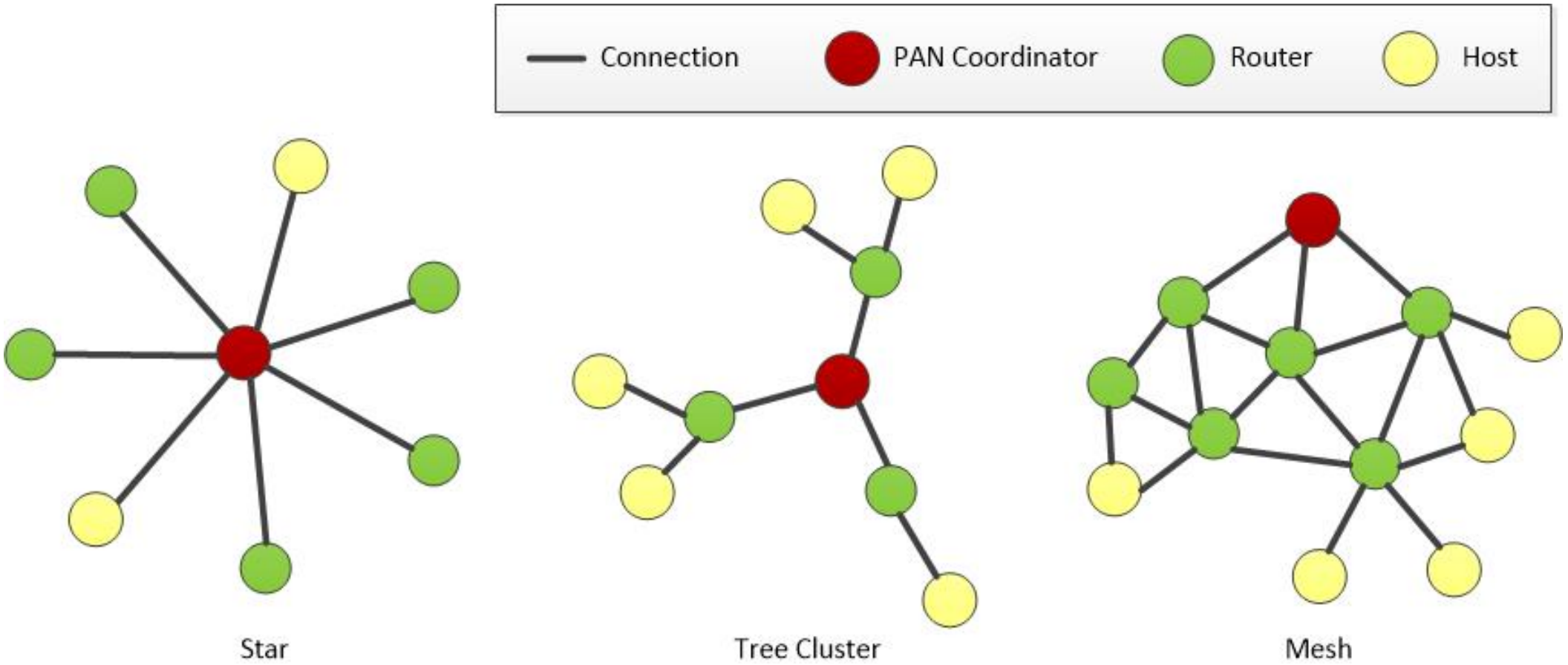


6LoWPAN: IEEE 802.15.4



6LoWPAN stack

6LoWPAN: IEEE 802.15.4 MAC Topology



6LoWPAN: IEEE 802.15.4 PHY

- Convert bits to signal
- Over-the-air transmission
- Allows both 2.4 GHz and Sub-1 GHz bandwidth

RPL

- Protocol for Low-power and Lossy Networks (LLNs)
- LLNs:
  - resource-constraint
  - lossy, unstable links
  - low packet delivery rates
- Our WSN fits the LLNs description

RPL: DODAG

- Destination-Oriented Directed Acyclic Graph
- Singe-root DAG
- No outgoing edges
- Border router is a DODAG root
- The mechanism behind 6LBR sensor node tree
- Need DIO (DODAG Information Object) to construct

RPL: DODAG Construction Algorithm

- Configuration and selection of DODAG roots
- Nodes advertising their presence, routing cost, DODAG affiliation, and other relevant metrics through link-local multicast DIO messages
- Nodes joining a DODAG after they listen to the received DIO messages
- Nodes populating the routing table with one or more DODAG parents as their next hop alongside other alternative routes

RPL: Role

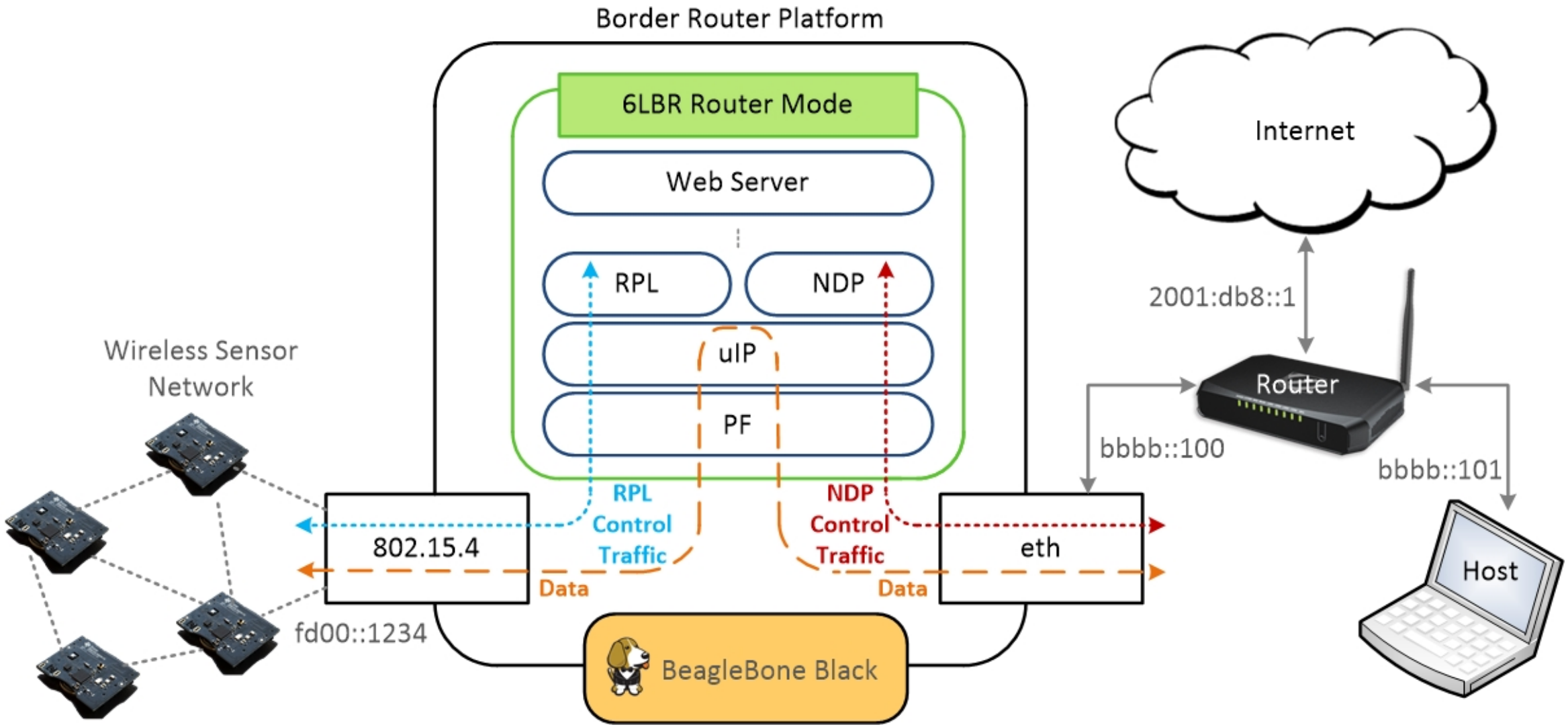
- RPL controls traffic on the WSN side of the border router
- NDP controls traffic on the eth side of the border router

Border Router

- Allows 6LoWPAN network communicates with IP-based networks
- 6LBR
- NAT64

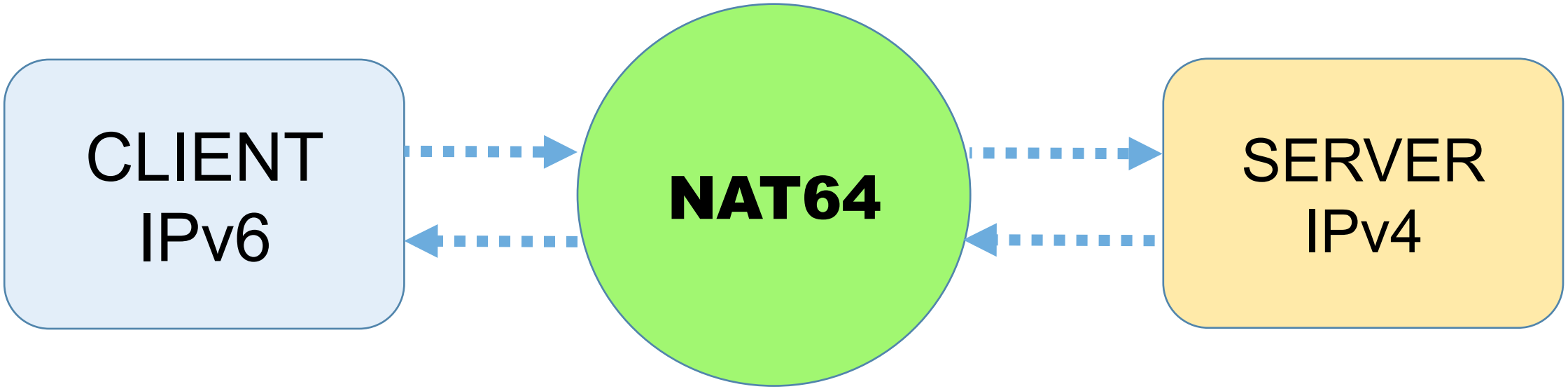
Border Router: 6LBR

- 6LoWPAN Border Router
- Contiki-based
- Handles two kinds of traffic:
  1. RPL (Routing Protocol for Low-power and Lossy Networks)
  2. NDP (Neighbor Discovery Protocol)

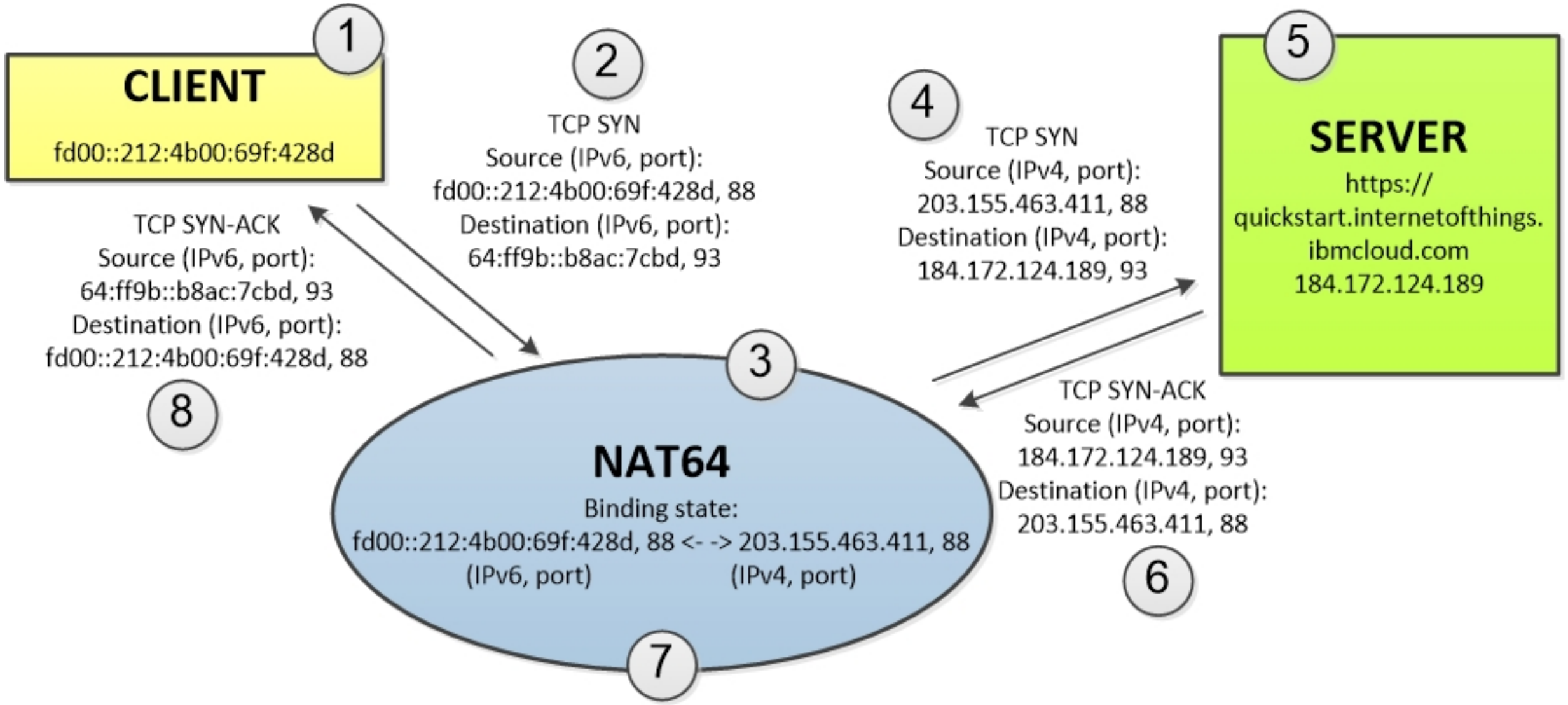

Border Router Platform
6LBR Router Mode
Web Server
RPL
NDP
uIP
PF
Internet
2001:db8::1
Router
bbbb::100
bbbb::101
Wireless Sensor Network
802.15.4
eth
RPL Control Traffic
NDP Control Traffic
Data
Data
fd00::1234
BeagleBone Black
Host

Border Router: NAT64

Translation mechanism



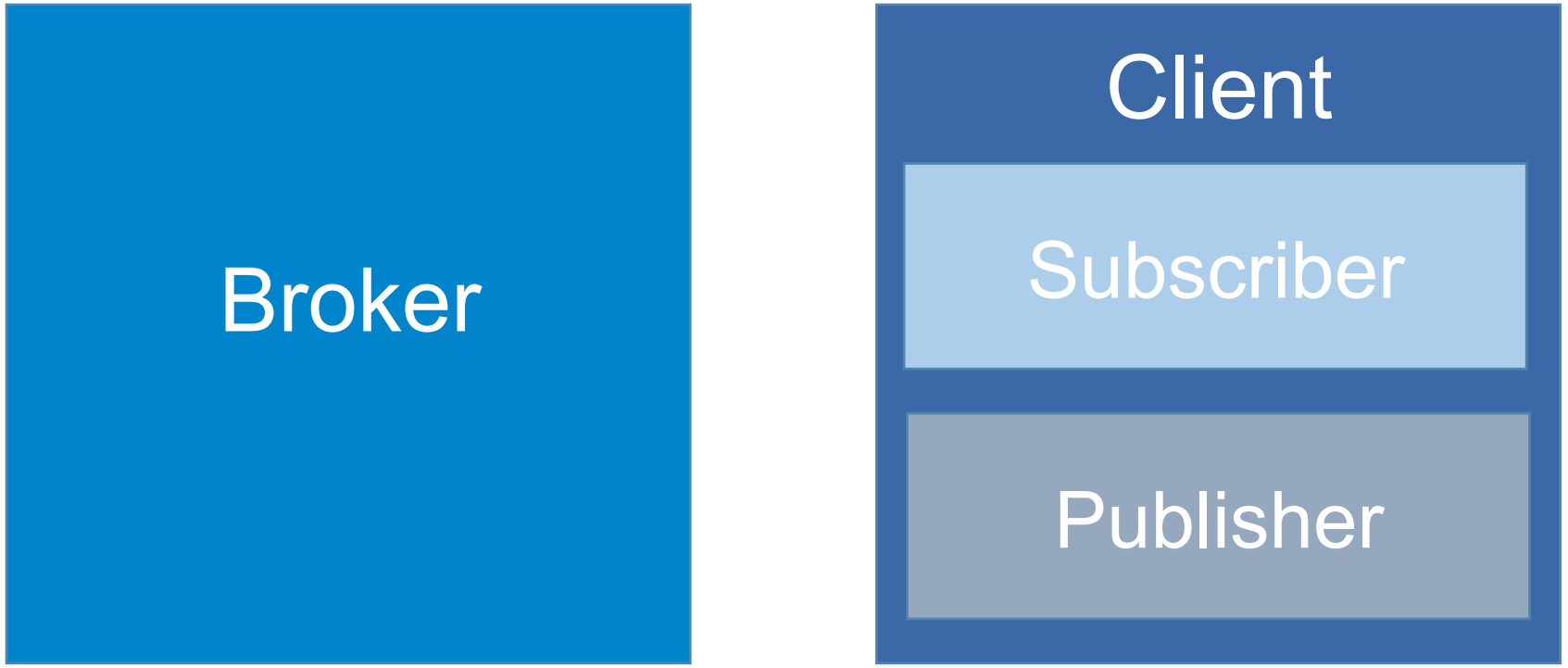
Stateful


1
CLIENT
fd00::212:4b00:69f:428d
2
TCP SYN
Source (IPv6, port):
fd00::212:4b00:69f:428d, 88
Destination (IPv6, port):
64:ff9b::b8ac:7cbd, 93
3
NAT64
Binding state:
fd00::212:4b00:69f:428d, 88 <- -> 203.155.463.411, 88
(IPv6, port) (IPv4, port)
4
TCP SYN
Source (IPv4, port):
203.155.463.411, 88
Destination (IPv4, port):
184.172.124.189, 93
5
SERVER
https://
quickstart.internetofthings.
ibmcloud.com
184.172.124.189
6
TCP SYN-ACK
Source (IPv4, port):
184.172.124.189, 93
Destination (IPv4, port):
203.155.463.411, 88
7
8
TCP SYN-ACK
Source (IPv6, port):
64:ff9b::b8ac:7cbd, 93
Destination (IPv6, port):
fd00::212:4b00:69f:428d, 88

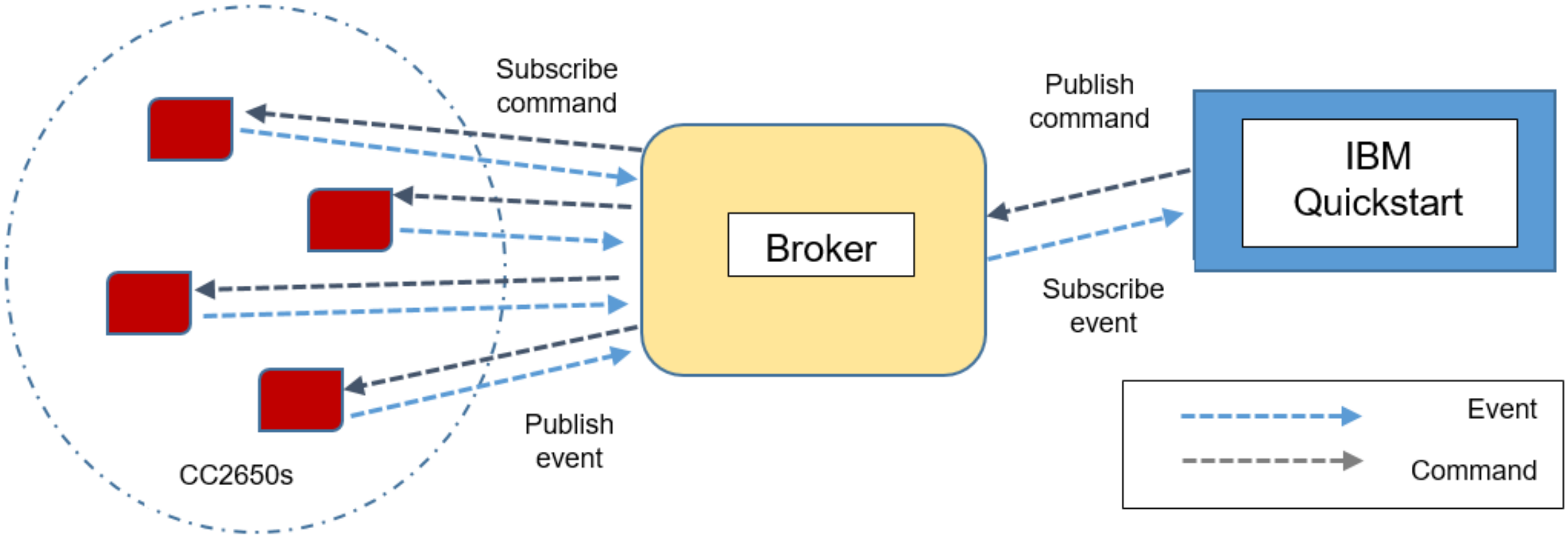
MQTT

- Message Queue Telemetry Transport
- Publish/subscribe messaging protocol
- Default protocol to communicate with IBM Quickstart

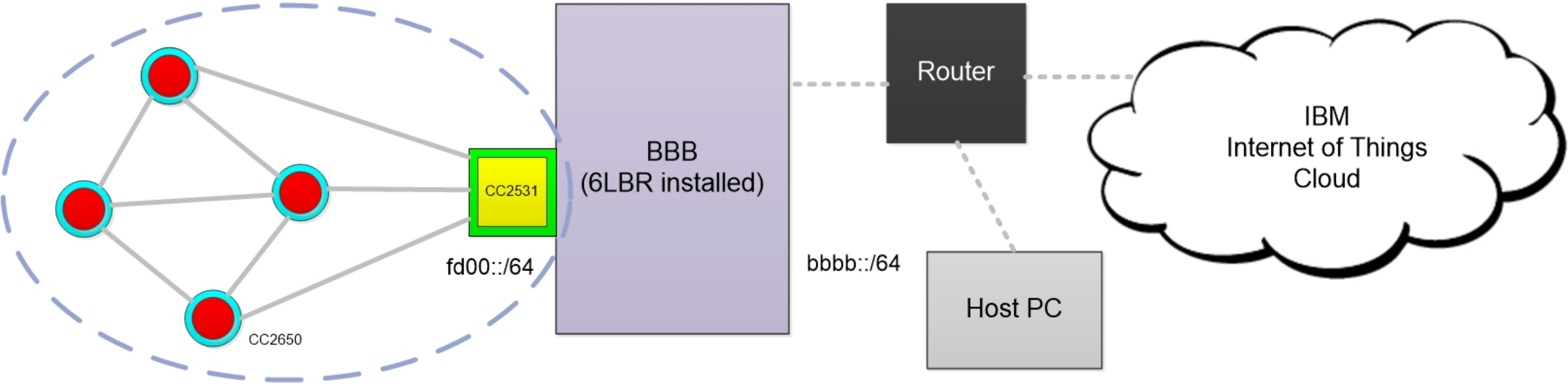
MQTT: Entities



MQTT

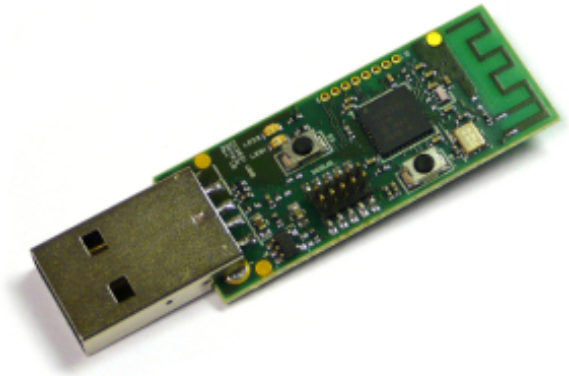

Subscribe
command
Publish
command
IBM
Quickstart
Broker
Subscribe
event
Publish
event
CC2650s
Event
Command

System Architecture

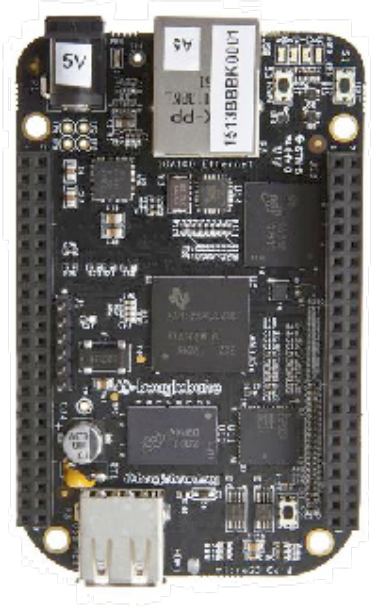


Tools

CC2650 SensorTag 2.0
Sensor devices



CC2531 USB Dongle
SLIP radio 2.4 GHz



BeagleBone Black
Edge router

IBM Quickstart

Tools: TI CC2650 SensorTag 2.0

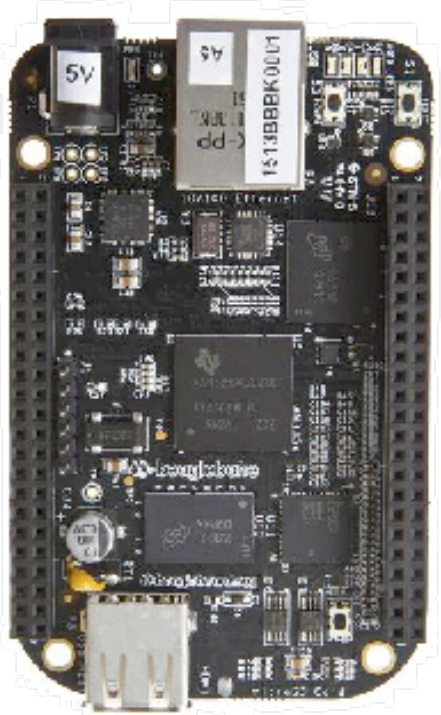
Ambient temperature
Light
Humidity
Magnetometer
Accelerometer (X, Y, Z)
Gyroscope (X, Y, Z)
Pressure
Object temperature

Tools: TI CC2531 USB Dongle

RF Transceiver
2.4 GHz
IEEE 802.15.4
USB-enabled SLIP

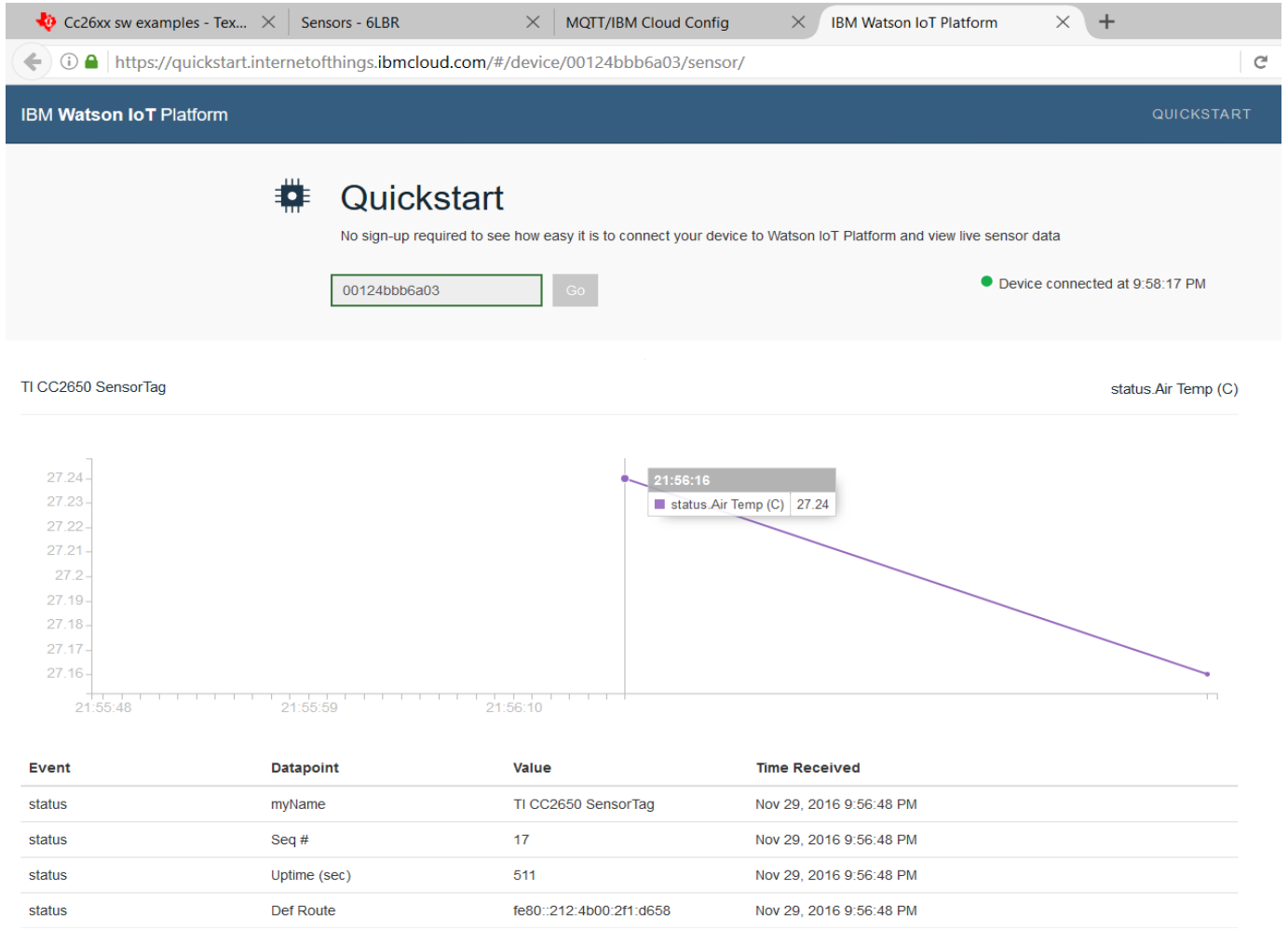
Tools: BeagleBone Black

1 GHz A8 PROCESSOR
AM3358 ARM® CORTEX™
512 DDR3 RAM
4GB 8-bit eMMC flash storage



IBM Quickstart

Data visualization
No sign up needed
IBM Bluemix account needed for further data processing



Conclusion

- Contiki application on 6LoWPAN mesh WSN
- Cloud connectivity via border router
- RPL and NDP controls data traffic in the border router
- Future works: data processing and analysis

References

- A. Dunkels, B. Gronvall and T. Voigt, "Contiki - A Lightweight and Flexible Operating System for Tiny Networked Sensors," in Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks, 2004.
- A. Dunkels, O. Schmidt and T. Voigt, "Using Protothreads for Sensor Node Programming," in Proceedings of the REAL WSN 2005 Workshop on Real World Wireless Sensor Networks, 2005.