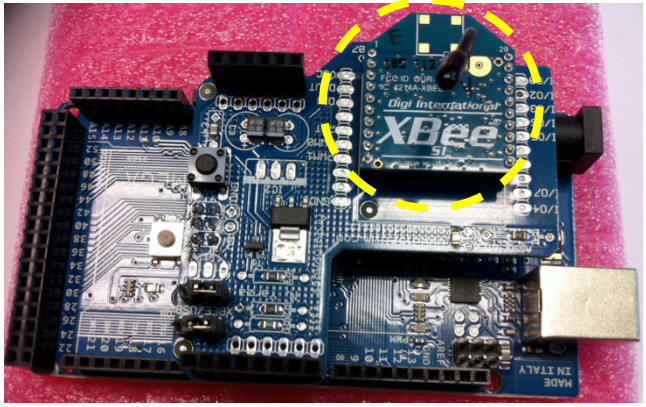
Communications entre capteurs et mise en réseau

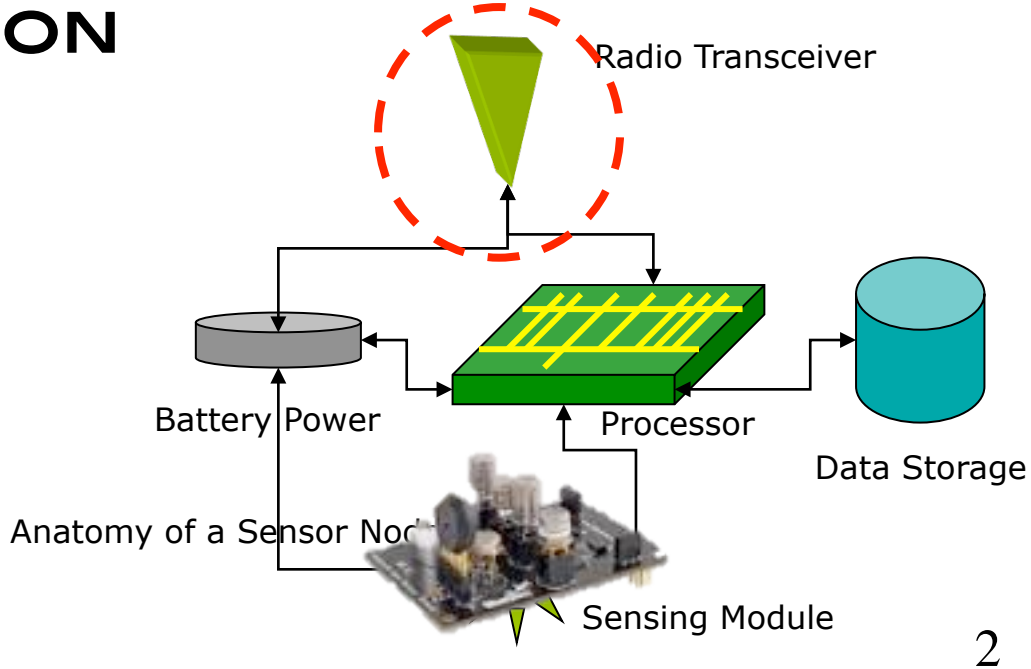
Programme Doctoral International de Modélisation des Systèmes Complexes

Prof. Congduc Pham
http://www.univ-pau.fr/~cpham
Université de Pau, France

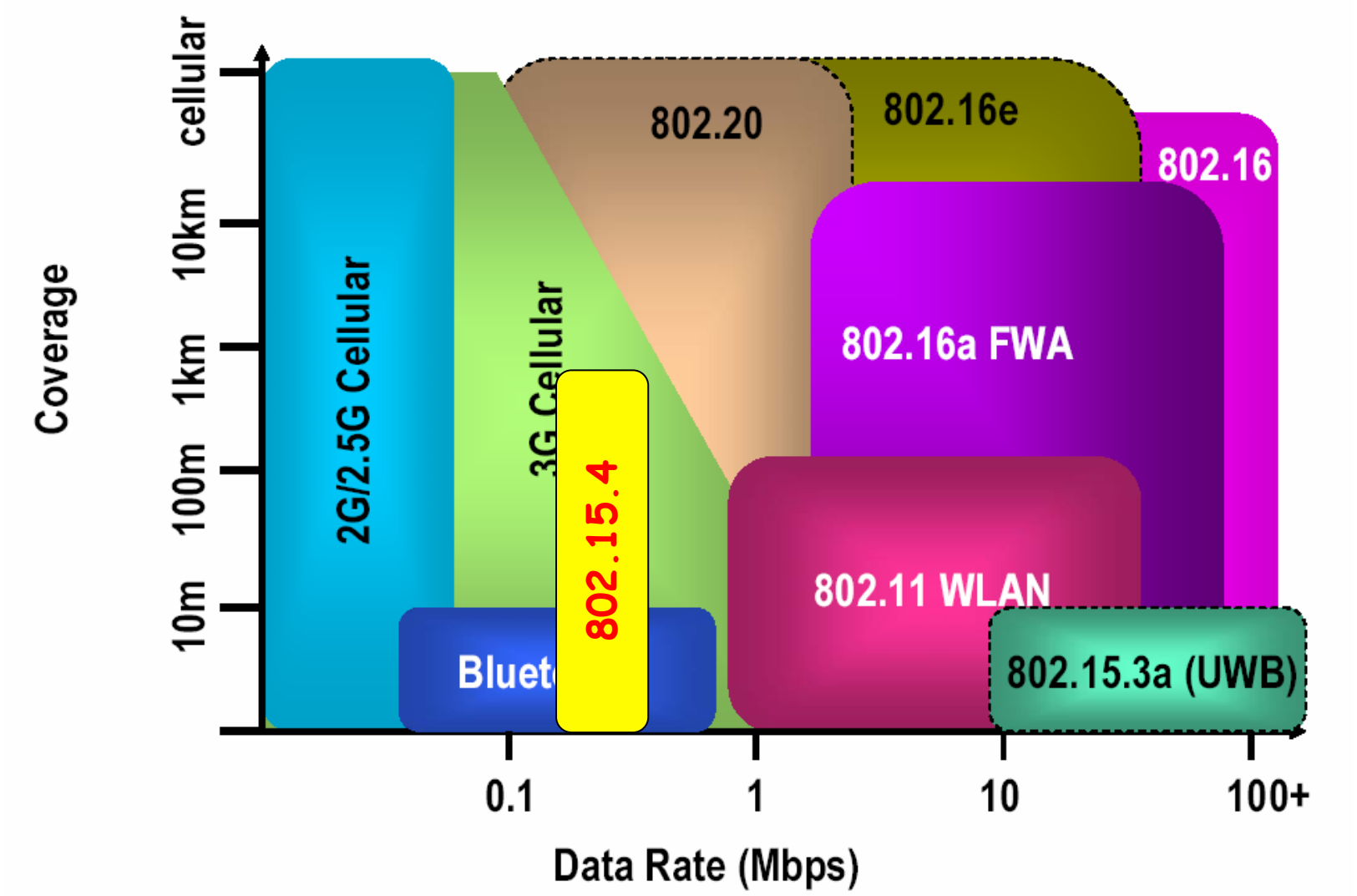
Wireless autonomous sensor

- In general: low cost, low power (the battery may not be replaceable), small size, prone to failure, possibly disposable
- Role: sensing, **data processing**, communication

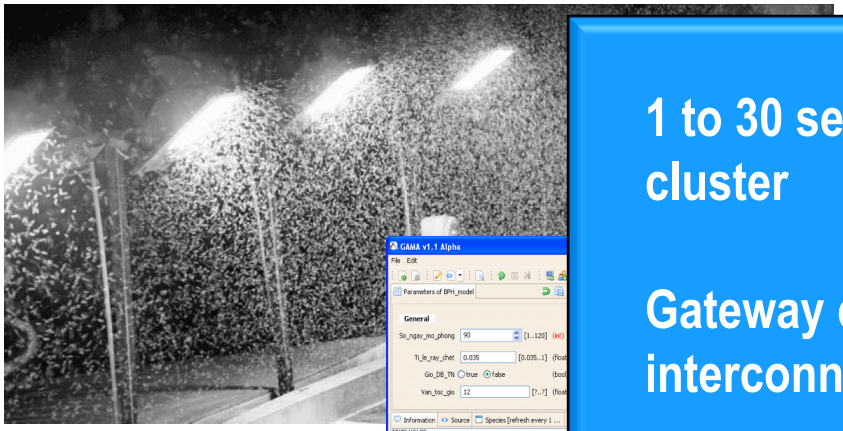


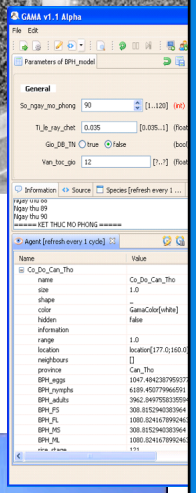


Summary of Wireless technologies

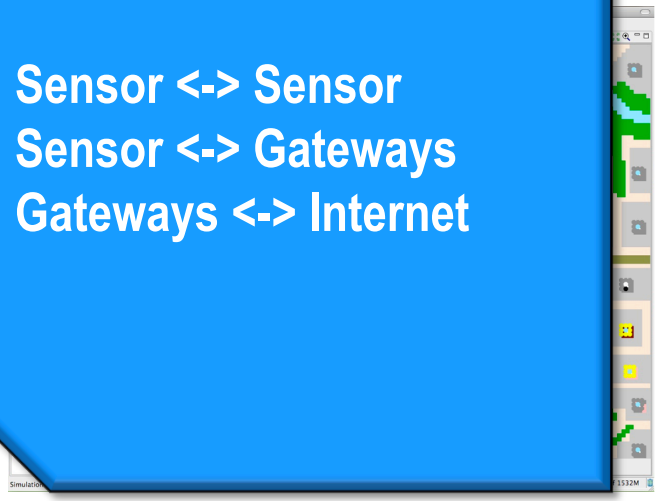


Application example

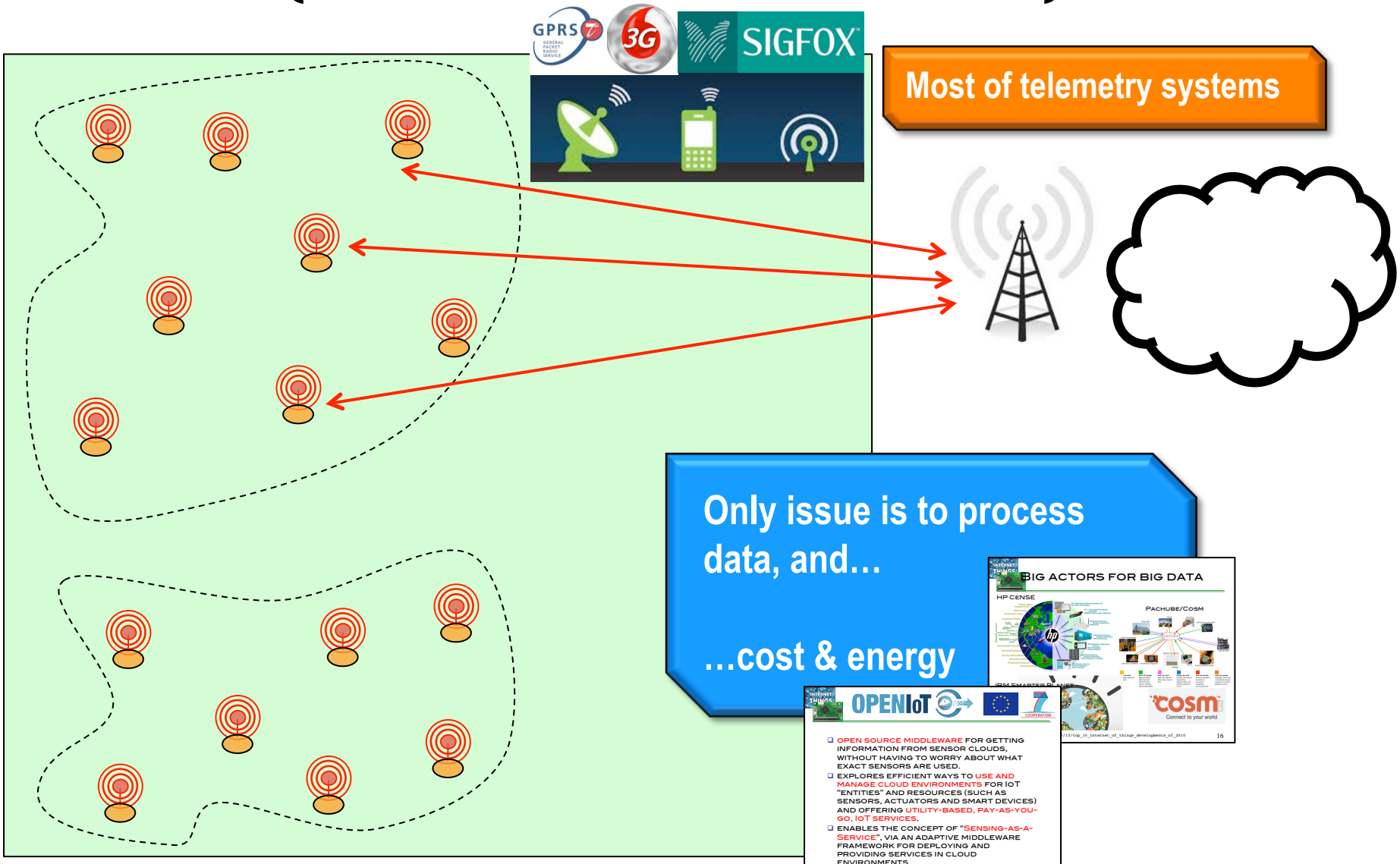




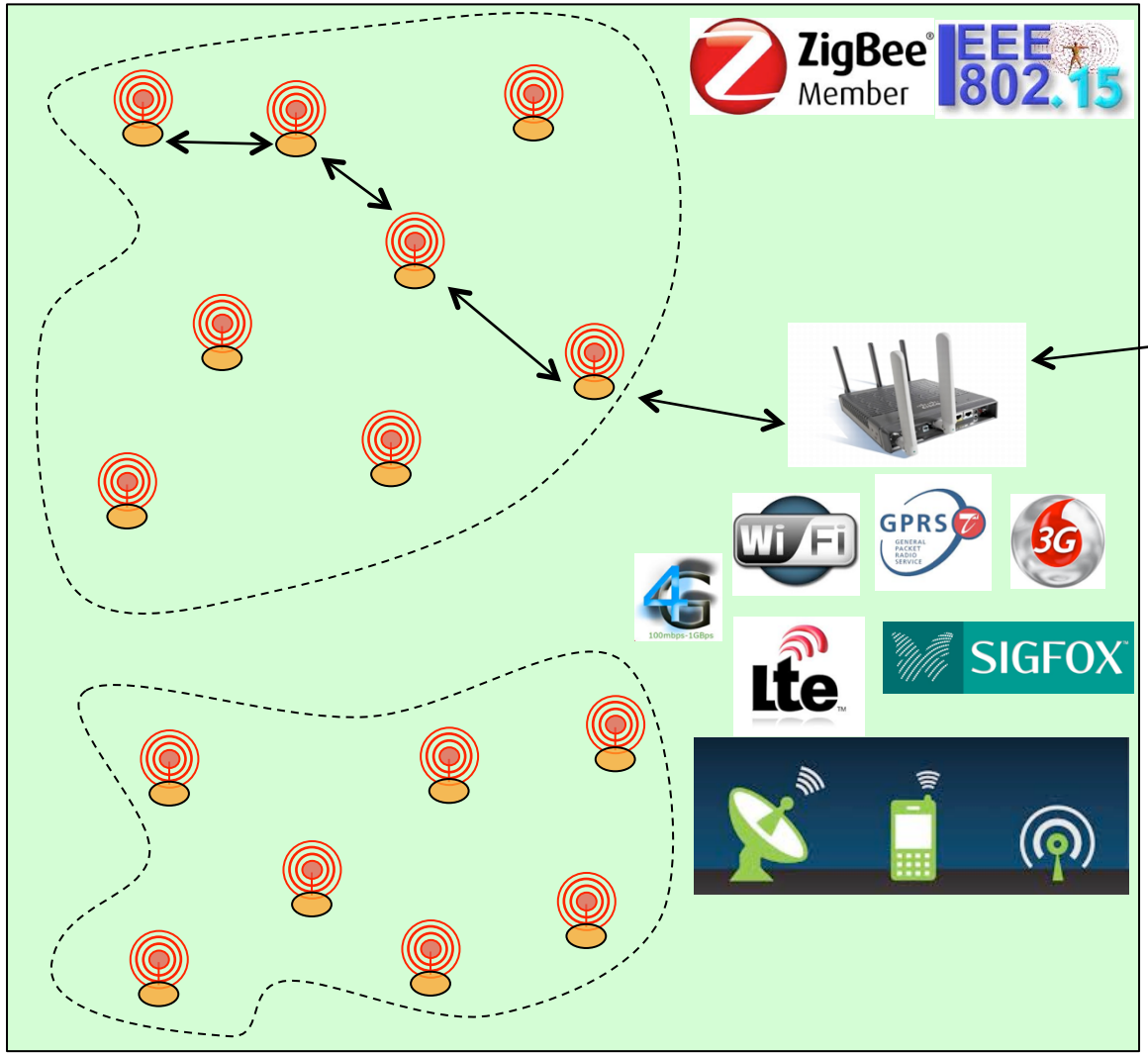
- 1 to 30 sensor nodes per cluster
- Gateway can interconnect clusters
- Communication needs:
  - Sensor <-> Sensor
  - Sensor <-> Gateways
  - Gateways <-> Internet

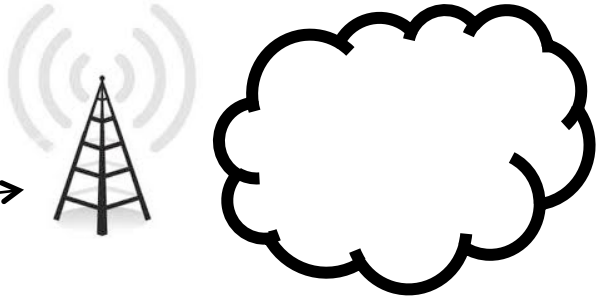


1-HOP COMMUNICATION (CELLULAR MODEL)



Multi-hop to gateways

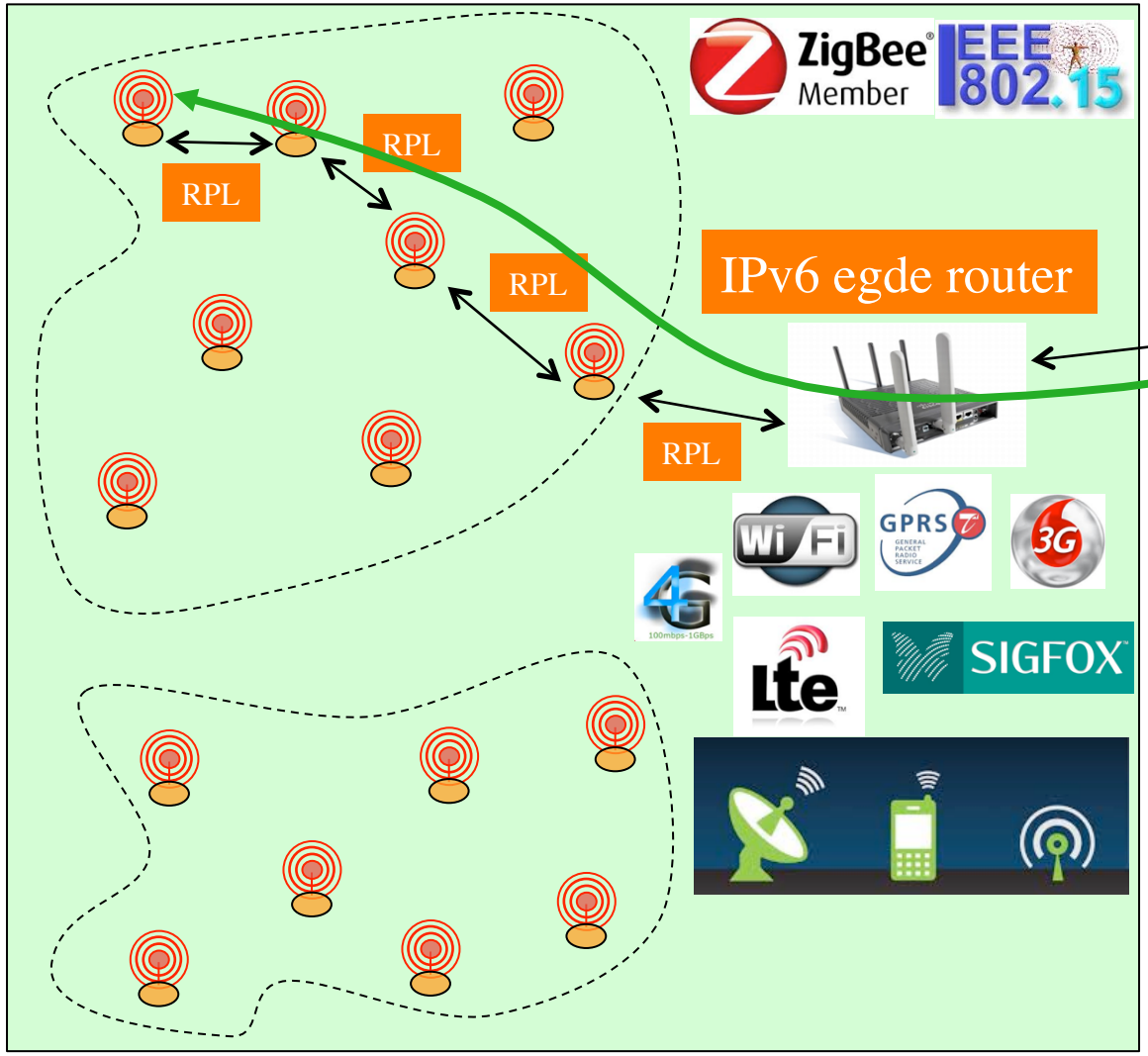


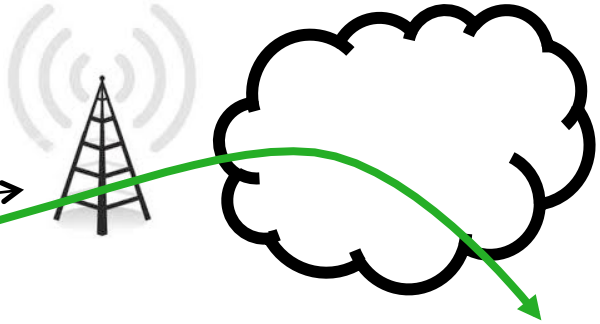


MAC & Routing issues, well-known to WSN community…

…but can we interoperate?

Using IP protocols





UDP CoAP

End-to-end IPv6 connectivity. UDP or CoAP can be used to transport sensor data in an HTTP-like fashion

IEEE 802.15.4

Caractéristiques Radio dans les réseaux de capteurs

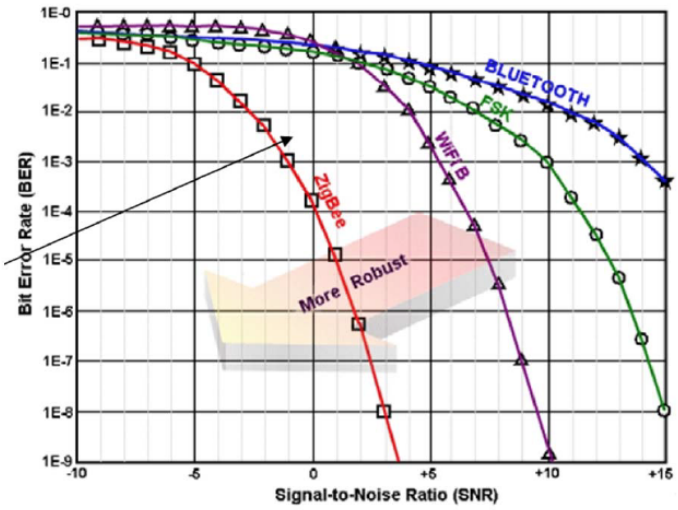
- Norme ZigBee (IEEE 802.15.4 PHY)

La norme IEEE802.15.4a, adaptées aux réseaux de capteurs, au contrôle industriel et aux dispositifs médicaux (CMI)

IEEE802.15.4 (couches 1 et 2):

- Three bands, 27 channels specified
  - 2.4 GHz: 16 channels, 250 kbps
  - 868.3 MHz : 1 channel, 20 kbps
  - 902-928 MHz: 10 channels, 40 kbps

Protocole	Zigbee	Bluetooth	Wi-Fi
IEEE	802.15.4	802.15.1	802.11a/b/g
Besoins mémoire	4-32 Kb	250 Kb +	1 Mb +
Autonomie avec pile	Années	Jours	Heures
Nombre de nœuds	65 000+	7	32
Vitesse de transfert	250 Kb/s	1 Mb/s	11-54 et + Mb/s
Portée	100 m	10-100 m	300 m

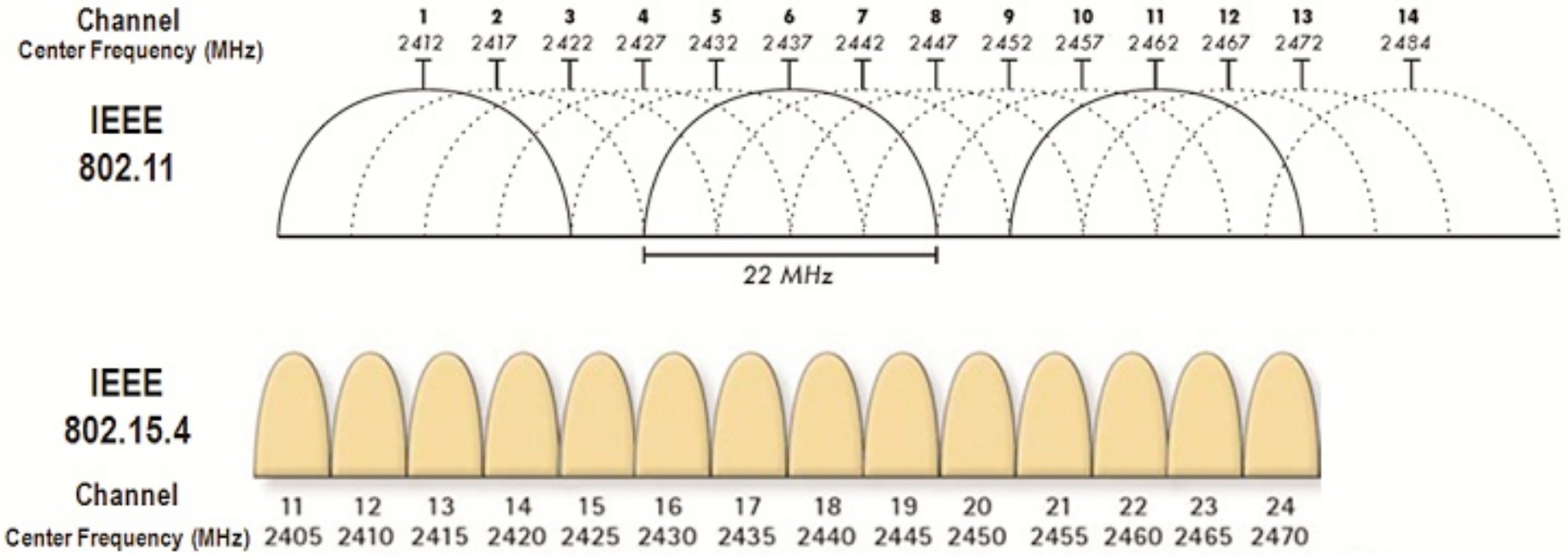


- Comparaison entre les normes ZigBee, Bluetooth et Wifi

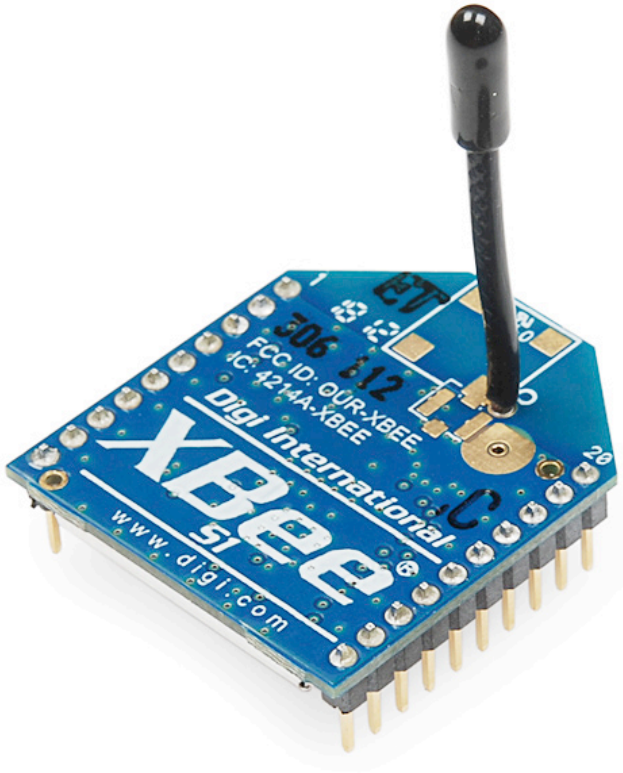
5

K.C. Yao, CommRadio dans les RxCapteurs

Spectrum band



XBee radio module from Digi



Implements IEEE 802.15.4 standard

64-bit hardware MAC address

0x0013A200409C0343

MAC FRAME FORMAT

Max size = 127 bytes

Max=102 bytes

Octets: 2	1	0/2	0/2/8	0/2	0/2/8	0/5/6/10/14	variable	2
Frame Control	Sequence Number	Destination PAN	Destination Address	Source PAN	Source Address	Auxiliary Security Header	Frame Payload	FCS
CC61	58	3332	0013A200 40922078	3332	0013A200 4086D834		HELLO	2B32
MHR		Addressing fields					MAC Payload	MFR

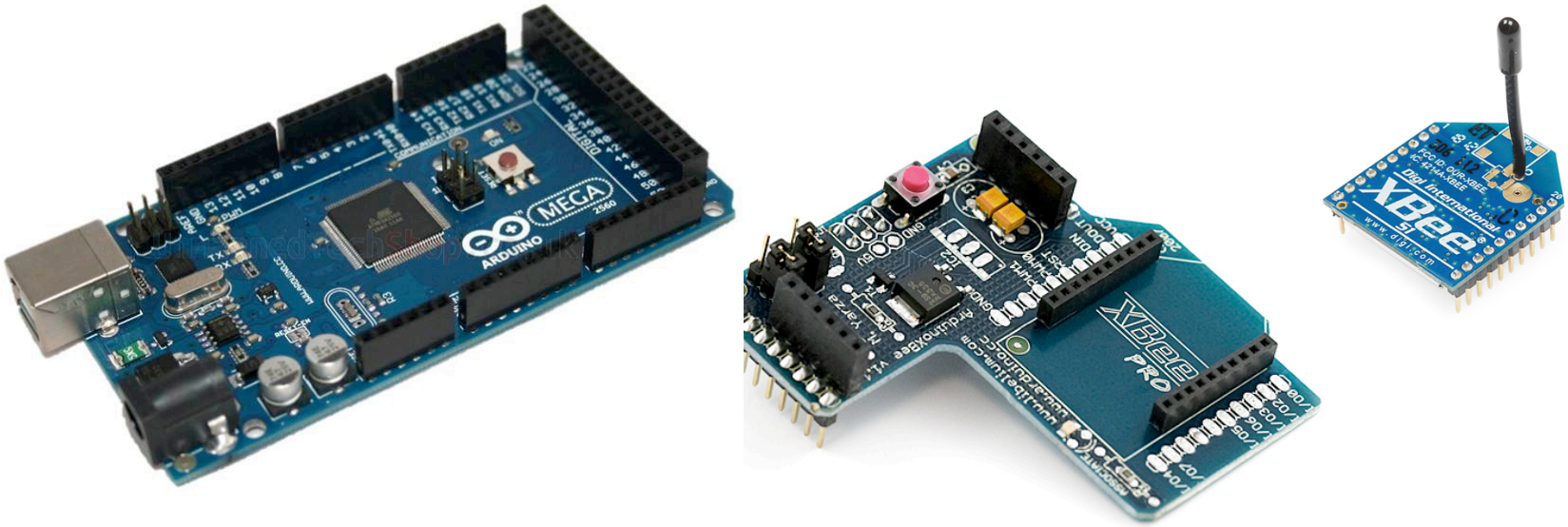
HELLO

64-bit 0x0013A2004086D834
16-bit 0x0010
CHANNEL 0x0C
PANID 0x3332

Can broadcast if sent to 0x000000000000FFFF

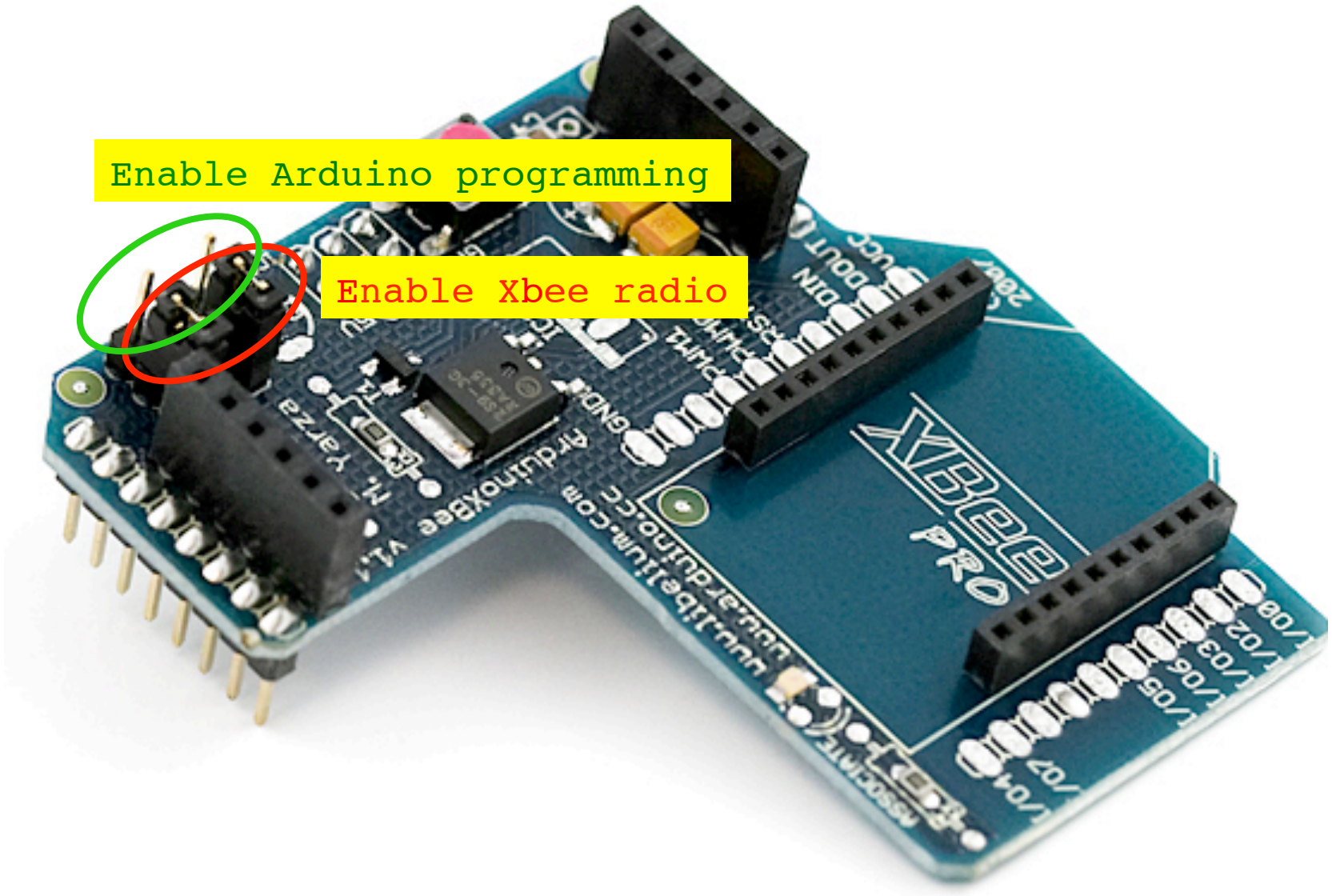
64-bit 0x0013A20040922078
16-bit 0x0020
CHANNEL 0x0C
PANID 0x3332

Arduino and XBee



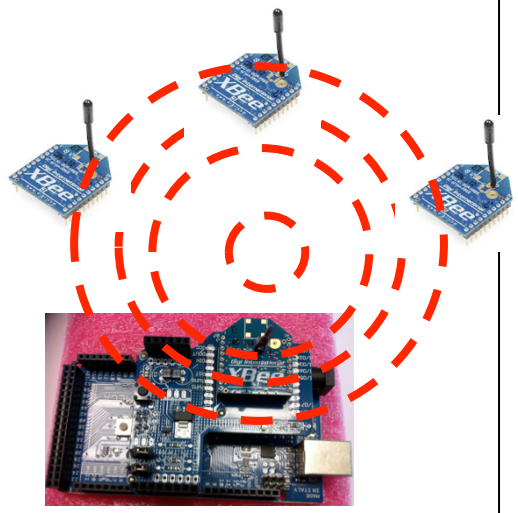
- Use software communication library to use XBee radio module : xbee-arduino
- Installation
  - Create a « libraries » folder in your Arduino sketch
  - Copy the Xbee library folder in sketch/libraries
- Source code
  - https://code.google.com/p/xbee-arduino/

XBee shield



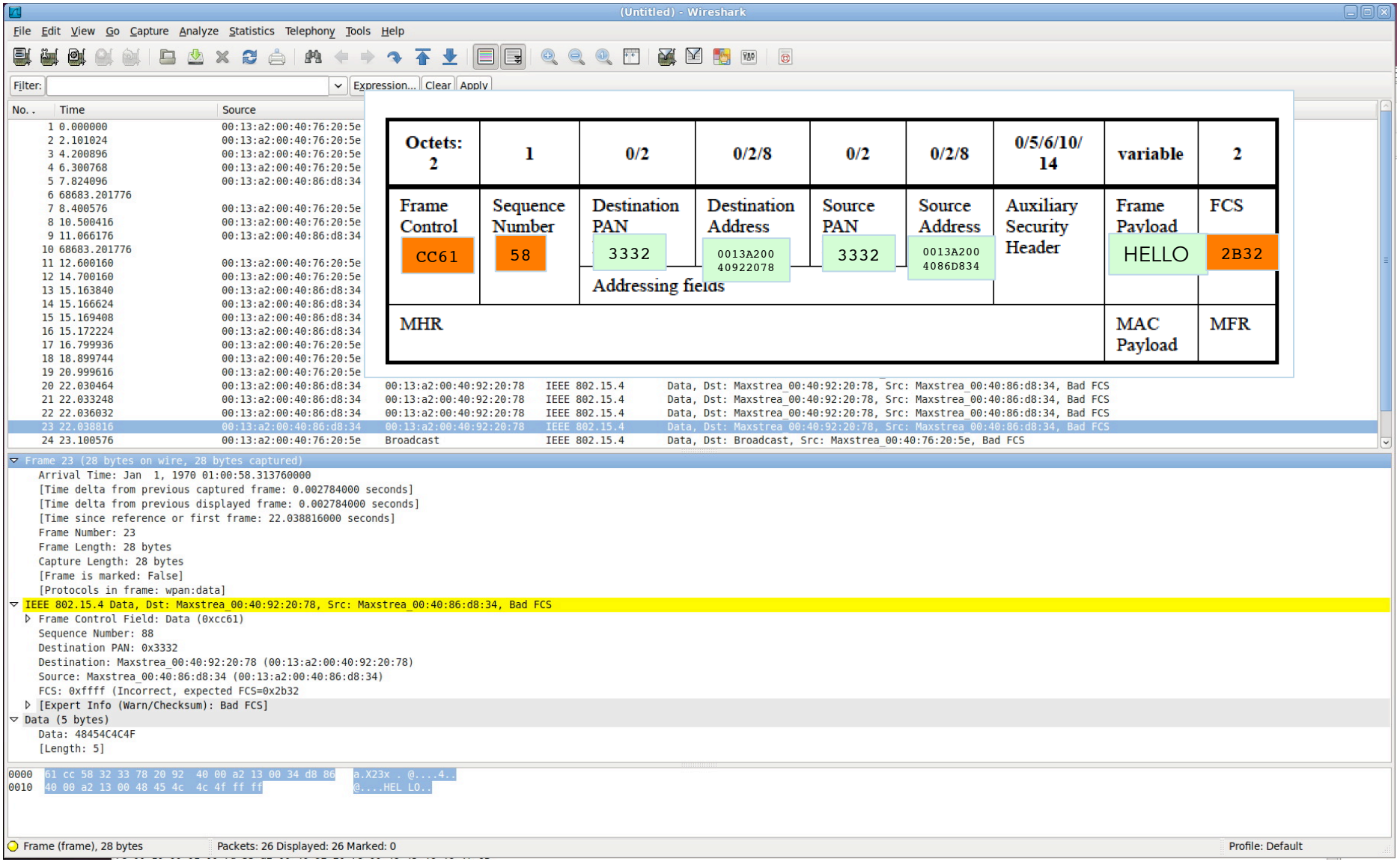
First step in communication: transmit

```
#include <XBee.h>
#include <SoftwareSerial.h>

XBee xbee = XBee();

uint8_t payload[80];

char data[6]="hello";

void setup() {

  xbee.begin(38400);

  Serial.begin(38400);
  Serial.println("Arduino. Will send packets.");
}

void loop() {

  delay(5000);

  // 64-bit addressing
  XBeeAddress64 addr64 = XBeeAddress64(0x00000000, 0x0000FFFF);

  for (int i; i<sizeof(data); i++)
    payload[i]=(uint8_t)data[i]);

  // in this way, we know the exact size of the payload
  Tx64Request tx = Tx64Request(addr64, (uint8_t*)data, sizeof(data));

  // Send your request
  xbee.send(tx);

  TxStatusResponse txStatus = TxStatusResponse();
}
```



Use broadcast, send to 0x000000000000FFFF

Radio sniffer



Next step in communication: receive

```
#include <XBee.h>
#include <SoftwareSerial.h>

XBee xbee = XBee();

uint8_t* payload;
uint8_t payload_len;

Rx64Response rx64 = Rx64Response();

void setup() {
  xbee.begin(38400);
  Serial.begin(38400);
  Serial.println("Arduino. Will receive packets.");
}

void loop() {
  // read incoming pkt
  xbee.readPacket();

  if (xbee.getResponse().isAvailable()) {

    // is it a response to the previously sent packet?
    if (xbee.getResponse().getApiId() == TX_STATUS_RESPONSE) {
    }

    if (xbee.getResponse().getApiId() == RX_64_RESPONSE) {

      xbee.getResponse().getRx64Response(rx64);

      payload = rx64.getData();
      payload_len=rx64.getDataLength();

      for (int i=0; i<payload_len; i++)
        Serial.print((char)payload[i]);

      Serial.println(" ");
    }
  }
}
```

Transmitting binary data

```
#include <XBee.h>
#include <SoftwareSerial.h>

XBee xbee = XBee();

// allocate two bytes for to hold a 10-bit analog reading
uint8_t payload[] = { 0, 0 };

int pin5 = 0;

void setup() {

  xbee.begin(38400);

  Serial.begin(38400);
  Serial.println("Arduino. Will send binary data.");
}

void loop() {

  delay(5000);

  // break down 10-bit reading into two bytes and place in payload
  pin5 = analogRead(5);
  Serial.println(pin5);
  payload[0] = pin5 >> 8 & 0xff;
  payload[1] = pin5 & 0xff;

  // 64-bit addressing: This is the SH + SL address of remote XBee
  XBeeAddress64 addr64 = XBeeAddress64(0x00000000, 0x0000FFFF);
  // unless you have MY on the receiving radio set to FFFF, this will be received as a RX16 packet

  // in this way, we know the exact size of the payload
  Tx64Request tx = Tx64Request(addr64, payload, sizeof(payload));

  // Send your request
  xbee.send(tx);

  TxStatusResponse txStatus = TxStatusResponse();
}
```

963_{10} on 10 bits

1111000011_2

0000001111000011_2

00000000_2 00000011_2

11111111_2

payload[0] 00000011_2

0x03

0000001111000011_2

11111111_2

payload[1] 11000011_2

0xC3

Receive binary data

```
#include <XBee.h>
#include <SoftwareSerial.h>

XBee xbee = XBee();

uint8_t* payload;
uint8_t payload_len;

Rx64Response rx64 = Rx64Response();

void setup() {
  xbee.begin(38400);
  Serial.begin(38400);
  Serial.println("Arduino. Will receive packets.");
}

void loop() {
  // read incoming pkt
  xbee.readPacket();

  if (xbee.getResponse().isAvailable()) {

    // is it a response to the previously sent packet?
    if (xbee.getResponse().getApiId() == TX_STATUS_RESPONSE) {
    }

    if (xbee.getResponse().getApiId() == RX_64_RESPONSE) {

      xbee.getResponse().getRx64Response(rx64);

      payload = rx64.getData();
      payload_len=rx64.getDataLength();

      for (int i=0; i<payload_len; i++)
        Serial.print((char)payload[i]);

      Serial.println(" ");
    }
  }
}
```

00000011_2	11000011_2
0x03	0xC3
payload[0]	payload[1]

```
int value;

value = payload[0] << 8  & 0xffff;
value += payload[1];

Serial.print(value);
```