# Power cases experimentations

Author
Julien GUILLEMOT

# Outline

1. **Introduction – aims of the experimentations**
2. **System architecture**
3. **Code implementation**
4. **Results and analysis**
5. **Conclusion and future work**

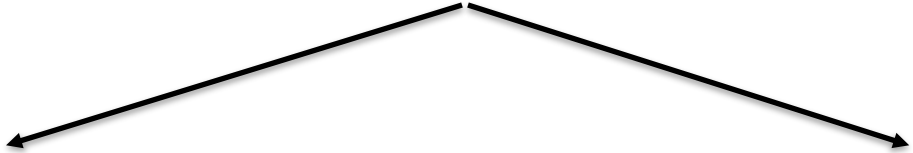
# Aims of the experimentations 1/2

1. **Latency Calculation** : The aim is to calculate the round-trip time (RTT) for packets of varying sizes as they are transmitted between different points in the network architecture.

**Packet Sizes:** The RTT will be tested with packets of the following sizes: 64 bytes, 128 bytes, 256 bytes, 512 bytes, 1024 bytes, and 1500 bytes. These sizes represent typical data packets used in network communications.

# Aims of the experimentations 2/2

**2. Power consumption analysis** : The second major aim is to measure and analyze the power consumption of the network setup (end and middle device)



**LoRa Modules**: Measure power consumption during transmission, reception, and idle states.

**Raspberry Pi Devices**: Measure overall power usage, including script processing and communication interfaces (Wi-Fi, Ethernet, LoRa).

# System architecture

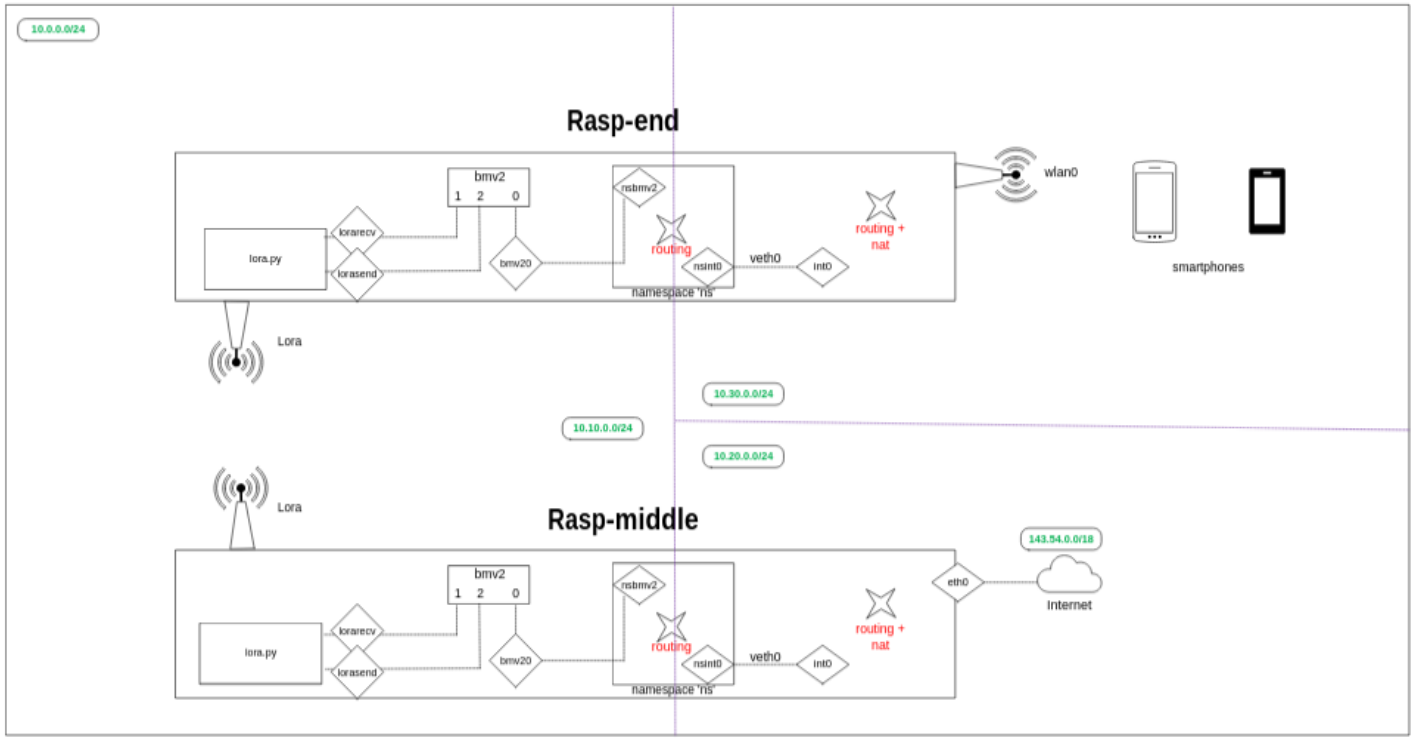



*Fig 1 : System architecture fo the raspberry pi*

# Code implementation

**Raspberry pi end**

latency.py
calcul_power.py

**Raspberry pi middle**

latency.py
calcul_power.py

- latency.py : This script is designed to send packets from the end device to the middle device using LoRa communication, manage retransmissions, and check for acknowledgment (ACK) responses.

- calcul_power.py : This script estimates the power consumption of a Raspberry Pi based on its CPU usage and frequency.

## Raspberry pi end (latency.py)

Utility functions

This script creates a packet, sends it to the middle device, waits for an acquittement, and then proceeds to the next packet.

Explanation of the functions :

- timestamp() : **Returns** the **time elapsed** since the program started
- get_packet_size() : **Prompts** the user to input the **packet size** within a specified range.
- get_repetitions() : **Prompts** the user to input the **number of repetitions**.
- get_ttl() : **Returns** the **TTL value** from the IP layer of the packet, if it exists.

## **Raspberry pi end (latency.py)**
Managing operations 1/2

Introduce the Handler class, which is central to managing packets, iterations, and transmission times.

- end_test() : **Ends** the current test **iteration** and resets for the next one if needed or finishes all iterations.
- calculate_average_time() : **Calculates** and **prints** the average **transmission time** for all iterations.
- reset_for_next_iteration() : **Resets the state** for the next test iteration.

## **Raspberry pi end (latency.py)**
Managing operations 2/2

- reinitialize_lora_module() : **Reinitializes** the **LoRa module** for the next iteration.
- run() : **Main loop** for sending packets, waiting for **ACKs**, and **managing** the test flow.
- on_rx_done() : **Handles** the event when a packet is **received**.
- on_tx_done() : **Handles** the event when a packet has been successfully **transmitted**.

# Raspberry pi middle (latency.py)

The same script as the Raspberry end, the only things that change are :

- The **utility functions** are removed (the functions that return the ttl, the packet size...)
- The middle script primarily **listens** for **incoming packets** from the end device. Once a packet is received, it logs the reception, processes it, and then sends an **ACK** back to the end device.
- The middle script's **on_rx_done** method is more complex because it deals with receiving the full packet from the end device. Once a **full packet** is received, it **calculates** the time taken **to receive** the packet and logs this information. After processing the packet, it sends an ACK back to the end device.

## Raspberry pi end and middle (calcul_power.py)

This script is developed to estimate the electric consummation of the Raspberrys Pi. Here´s the explanation of the different functions :

- get_cpu_frequency() : **return** the **CPU frequency** in MHz.
- estimate_current() : **estimate** current **based on** CPU usage and frequency.
- calculate_power() : **return** the **power consumption** in Watts.

When the script is launched, that return the CPU frequency, the current and the power consumption every 5 seconds.

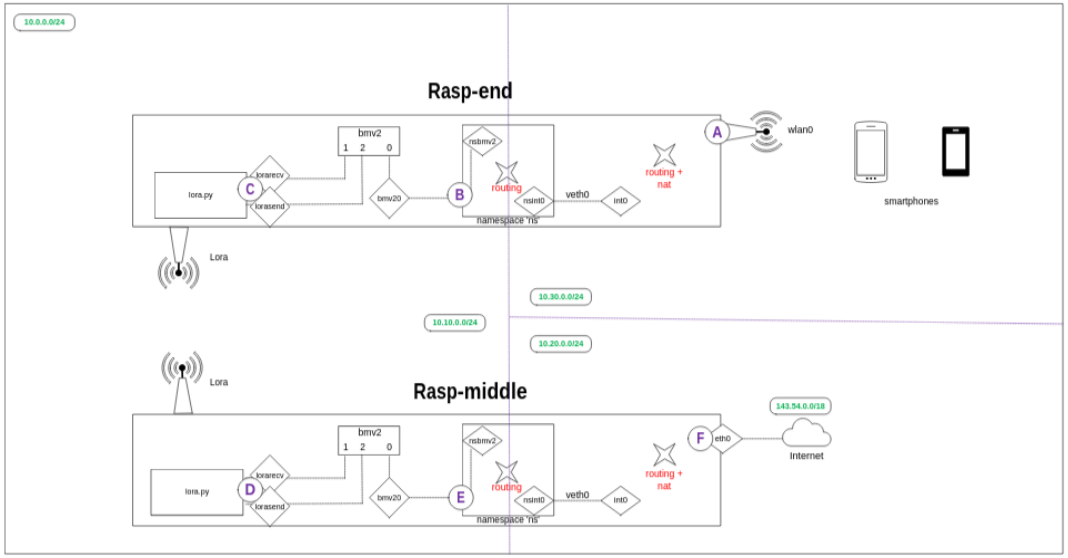
# How to launch the scripts to see what´s going on

| | |
|---|---|
| root@p4pi-end:~/pySX127x# ./latency.py -i int_in -o int_out -m end –f<br><br>***Ask to the user how many repetitions he want and the size of the packets.***<br>***Then send the packets divided with a maximum RTT of 127 to the raspberry middle.***<br>(-m for the mode and –f to send all the packet and not just a part of the packet) | root@p4pi-middle:~/pySX127x# ./latency.py -i int_in -o int_out -m middle -f<br><br>***Wait the packets from the end device and send the acquittement when it receive the whole packet.***<br><br>(-m for the mode and –f to send all the packet and not just a part of the packet) |
| root@p4pi-end:~/pySX127x# ./calcul_power.py<br><br>***Calcul the power of the end device.*** | root@p4pi-middle:~/pySX127x# ./calcul_power.py<br><br>***Calcul the power of the middle device.*** |

# All the paths the packets take

3 different measurements points do to :

1. **A ↔ F:** RTT between Wi-Fi on Rasp-end and Ethernet on Rasp-middle.
2. **B ↔ E:** RTT between internal virtual interfaces on each Raspberry Pi.
3. **C ↔ D:** RTT across the LoRa link only.



# 1. How to run the scripts from A to F

Raspberry end :
A : ./latency.py -i wlan0 –o wlan0 -m end –f
(-i for the in interface, -o for the out interface, -m for the mode and –f to send the entire packet)

Raspberry middle :
F : ./latency.py -i eth0 -o eth0 -m middle –f

# 2. How to run the scripts from B to E

Raspberry end :
B : ./latency.py -i nsbmv2 –o nsbmv2 -m end -f

Raspberry middle :
E : ./latency.py -i nsbmv2 -o nsbmv2 -m middle –f

# 3. How to run the scripts from C to D

Raspberry end :
C : ./latency.py -i lorarecv –o lorasend -m end -f

Raspberry middle :
D : ./latency.py -i lorarecv -o lorasend -m middle –f

# Results and analysis

Time depending on packet size

***Gnuplot*** is a command-line plotting software on linux that allows users to create a wide range of 2D and 3D plots. It's widely used for visualizing data in scientific, engineering, and mathematical contexts.

For example, used it for calculated the time take the packet to be transmitted depending on the packet size.
Here's a graph that calculates the time it takes for packets to be transmitted based on packet size from point A to F.

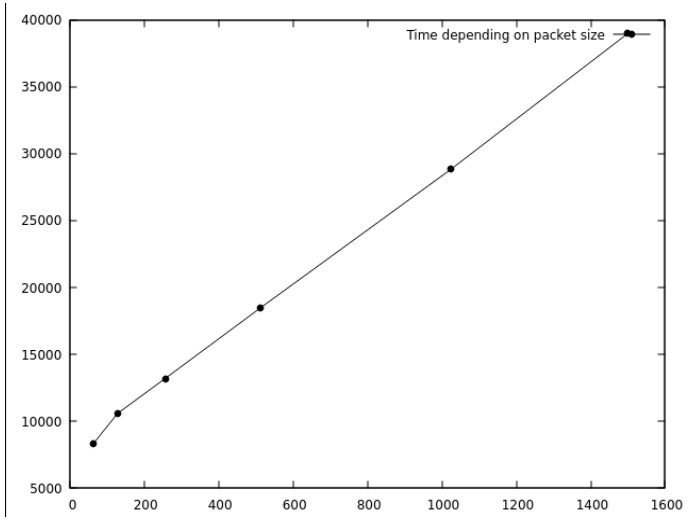



*Fig 2 : Time depending on the packet size  A<->F*

***plot "data_30_repetitions.dat" title "Time depending on packet size" with linespoints linetype 7 linecolor rgb "black"***

# Results and analysis

## Energie consumption 1/3

With the 2 calcul_power.py scripts, we can know the energy consumption of the 2 raspberry pi.

While these 2 programs are launched, we try to launch the 2 other programs which allow the packets to be transmitted between the 2 raspberry pi.

With these firsts tests, we can see that the middle device is consuming a lot waiting the packet of the end device, see on the next slide

# Results and analysis

Energie consumption 2/3

## Raspberry end

When the scripts latency.py isn´t running :

```
Raspberry end : CPU -> Usage: 1.3% & Frequency: 600.0 GHz -> Estimated Power: 1501.28 Watts
Raspberry end : CPU -> Usage: 1.0% & Frequency: 600.0 GHz -> Estimated Power: 1501.28 Watts
Raspberry end : CPU -> Usage: 1.3% & Frequency: 600.0 GHz -> Estimated Power: 1501.28 Watts
Raspberry end : CPU -> Usage: 0.5% & Frequency: 600.0 GHz -> Estimated Power: 1501.26 Watts
Raspberry end : CPU -> Usage: 0.7% & Frequency: 600.0 GHz -> Estimated Power: 1501.27 Watts
Raspberry end : CPU -> Usage: 0.3% & Frequency: 600.0 GHz -> Estimated Power: 1501.26 Watts
```

The script is launched, waiting for the user´s informations :

```
root@p4pi-end:~/pySX127x# ./latency.py -m end -f
Please enter the number for repetitions: 5
Please choose the size of the packet in b (between 64 and 1500):
```

Terminal

```
Raspberry end : CPU -> Usage: 25.6% & Frequency: 1500.0 GHz -> Estimated Power: 3751.89 Watts
Raspberry end : CPU -> Usage: 8.8% & Frequency: 1500.0 GHz -> Estimated Power: 3751.47 Watts
Raspberry end : CPU -> Usage: 0.3% & Frequency: 1500.0 GHz -> Estimated Power: 3751.26 Watts
Raspberry end : CPU -> Usage: 0.3% & Frequency: 600.0 GHz -> Estimated Power: 1501.26 Watts
```

## Raspberry middle

When the scripts latency.py isn´t running :

```
Raspberry middle : CPU -> Usage: 0.3% & Frequency: 600.0 GHz -> Estimated Power: 1501.26 Watts
Raspberry middle : CPU -> Usage: 0.5% & Frequency: 600.0 GHz -> Estimated Power: 1501.26 Watts
Raspberry middle : CPU -> Usage: 0.5% & Frequency: 600.0 GHz -> Estimated Power: 1501.26 Watts
Raspberry middle : CPU -> Usage: 0.8% & Frequency: 600.0 GHz -> Estimated Power: 1501.27 Watts
Raspberry middle : CPU -> Usage: 0.5% & Frequency: 600.0 GHz -> Estimated Power: 1501.26 Watts
Raspberry middle : CPU -> Usage: 0.3% & Frequency: 600.0 GHz -> Estimated Power: 1501.26 Watts
```

The script is launched, waiting for the packets :

```
root@p4pi-middle:~/pySX127x# ./latency.py -m middle -f
Mode <- SLEEP
Mode <- FSK_STDBY
Mode <- SLEEP
Mode <- FSK_STDBY
Mode <- SLEEP
Mode <- RXCONT
```

Terminal

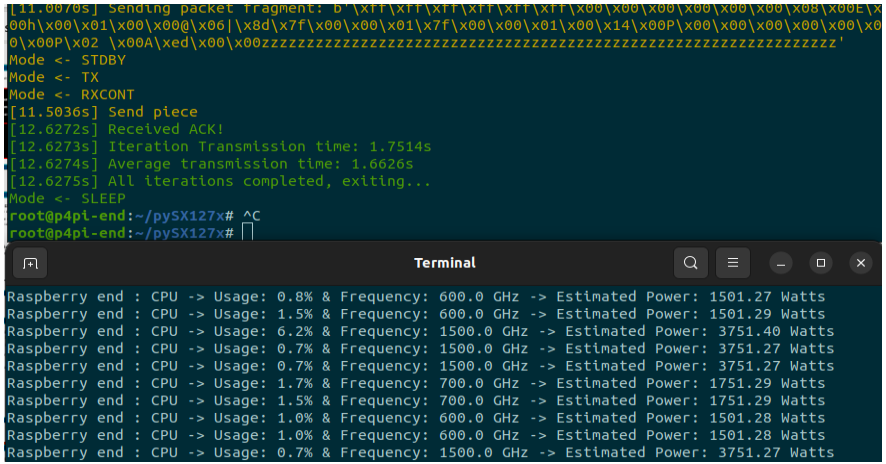
```
Raspberry middle : CPU -> Usage: 25.1% & Frequency: 1500.0 GHz -> Estimated Power: 3751.88 Watts
Raspberry middle : CPU -> Usage: 25.4% & Frequency: 1500.0 GHz -> Estimated Power: 3751.88 Watts
Raspberry middle : CPU -> Usage: 3.5% & Frequency: 1500.0 GHz -> Estimated Power: 3751.34 Watts
```

# Results and analysis

Energie consumption 3/3

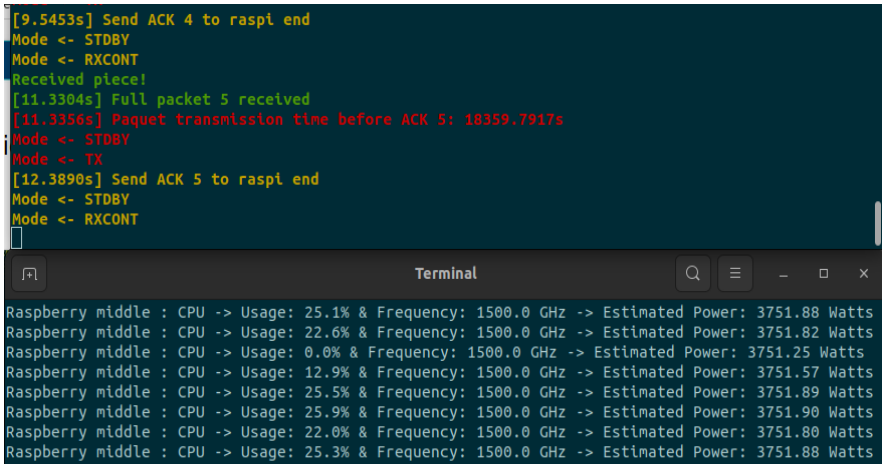
## Raspberry end

The packets are in transit :




## Raspberry middle

The packets are in transit :



# Conclusion and future work

- There are some ways to decrease the time for each repetitions, but if we remove these sleep() between the transmission of the splitting packets, we lose some packets.

- The time for the 3 tests A <-> F, B <-> E and C <-> D are the same for the moment, I have to figure out

- Investigate why it´s not working with 200B packets

# Resources

- Github : https://github.com/ComputerNetworks-UFRGS/Programmable_Low_End_Networks/tree/main
  - power_cases folder with the README for these experimentations I done these last 2 weeks
  - Documentation folder I done few weeks ago about one way we can better transmit the packets