IPoL, un protocol per la transmissió de dades mitjançant senyals lluminosos

Lluc Parés Olivares

Resum– Aquest treball pretén mostrar el procés de creació i implementació d'un protocol per transmetre dades mitjançant senyals lluminosos. S'ha realitzat per tal d'entendre els requisits que s'han de tenir en compte a l'hora d'utilitzar aquestes tecnologies i aprendre sobre la implementació en sistemes Unix a més baix nivell. El resultat doncs, han estat les especificacions del protocol juntament amb la seva caracterització, que demostren la factibilitat de transmetre informació mitjançant llum i donen un punt de partida pel desenvolupament i implementació en sistemes més adequats i fiables.

Paraules clau– Protocol, UNIX, interfície, comunicació, llum, xarxa, Raspberry Pi, LED

Abstract– This project aims to present the procedure to create and implement a protocol that sends data using light signals. It has been developed to understand the requirements that should be taken into account when using these technologies and to learn about low-level implementation in Unix systems. Its results are the protocol specs and characterisation, which ensure the feasibility of transmitting information through light and set a starting point for developing and implementing it in a more appropriate and reliable system.

Keywords– Protocol, UNIX, interface, communication, light, networking, Raspberry Pi, LED

1 Introducció

ACTUALMENT existeixen diverses tecnologies i protocols de comunicació que permeten la transmissió d'informació mitjançant diferents mètodes, des de variacions de potència en un cable, fins a canvis de freqüència en les ones electromagnètiques.

L'any 1977 es va desenvolupar la tecnologia coneguda com a *Light-Fidelity* (Li-Fi) [1] que permet la transmissió de dades mitjançant senyals lluminosos. Això presenta una gran millora respecte a altres tecnologies de comunicació sense cable, ja que fa ús del rang d'ones de llum visible, que és més ample que el d'altres tecnologies com Wi-Fi.

El principal avantatge però, és que ja existeix una infraestructura, és a dir, fonts d'il·luminació, que permet la utilització d'aquest tipus de protocols. Imaginem doncs, un futur pròxim on una bombeta qualsevol estigui transmetent informació a la vegada que fa la seva funció principal d'il·luminar.

A part de Li-Fi, existeixen altres protocols que també fan ús de senyals lluminosos, per exemple mitjançant làsers [2], però actualment Li-Fi és el més estès.

Pel que fa a la implementació en dispositius reals, els sistemes UNIX permeten la redirecció de datagrames IP cap a aplicacions en user space mitjançant la utilització de les interfícies *tun*.

El propòsit d'aquest projecte és el d'explorar en la creació d'un protocol que permeti la comunicació mitjançant senyals lluminosos i la implementació d'aquest en un sistema UNIX. Això s'aconseguirà utilitzant una font de llum i un sensor que sigui capaç de captar les variacions de lluminositat en l'ambient.

1.1 Objectius

L'objectiu principal d'aquest projecte és la creació i implementació d'un protocol de comunicació mitjançant senyals lluminosos. Per assolir-lo s'han determinat diferents metes, o subobjectius, i es llisten a continuació per ordre de prioritat:

- Definició del sistema de comunicació. Aquest subobjectiu pretén definir l'escenari, avaluant quins són els sistemes, components i arquitectura més adients pel projecte. També es pretén implementar l'escenari per poder transmetre bits/bytes. El resultat doncs, són tant

• E-mail de contacte: lluc.pares@outlook.com
• Menció realitzada: Tecnologies de la Informació
• Treball tutoritzat per: Sergi Robles Martínez (dEIC)
• Curs 2016/17

les especificacions de l'escenari, com les funcions per transmetre bytes mitjançant el sistema.

- Definició del protocol, incloent-hi els formats dels missatges, l'esquema de comunicació a utilitzar i tot el que comporta un protocol de la capa d'enllaç. També caldrà definir les condicions esperades pel seu correcte funcionament. El resultat és l'especificació estàndard del protocol, de l'estil de la de IEEE 802.3 [3]. D'ara endavant, es farà referència a aquest com a *IP over Light* (IPoL).

- Implementació en un sistema UNIX. Per complir aquest subobjectiu cal implementar IPoL en aplicacions en user space i instanciar una interfície de xarxa de tunneling, que permet interceptar datagrames IP i redirigir-los cap a les aplicacions en user space. El resultat ha de ser similar al que es va obtenir en la implementació del RFC1149 [4] feta per Bergen Linux User Group [5].

- Proves de rendiment. Consisteix en una avaluació del protocol en la qual es calcula el *throughput* i la taxa d'error entre d'altres. Aquestes proves ajuden a treure conclusions respecte del protocol i la seva implementació.

1.2 Metodologia

La metodologia escollida per la realització dels objectius és la Kanban, que consisteix a definir les tasques, o *user stories*, i un *workflow* per passos. En un taulell es mostren els diferents estats del *workflow* (New, Ready, Prototyping, In Progress, On Hold, Done) i les tasques es col·loquen en algun d'aquests estats.

S'ha escollit aquesta metodologia d'acord amb la simplicitat del projecte pel que fa als recursos, ja que ja hi ha prou amb una metodologia que permeti definir unes tasques amb les seves dependències i mostrar el seu progrés.

1.3 Estructura del document

El document està estructurat de la següent forma:

- Definició de l'escenari, on es mostra l'escenari utilitzat durant el desenvolupament del projecte per tal de realitzar la implementació, juntament amb el funcionament d'alguns dels components.

- Limitacions i problemes, on es tracten les diferents limitacions que sorgeixen per les decisions preses tant en l'apartat dels objectius com en la definició de l'escenari.

- Definició del protocol, on es mostren les especificacions del protocol resultant i es descriu el seu funcionament.

- Implementació, on es mostra com s'ha implementat IPoL en un sistema Unix en user space.

- Resultats, on s'avaluen les característiques del protocol mitjançant un seguit de proves que ajuden a entendre el comportament d'aquest.

- Conclusions, on es fa un recull del què s'ha vist durant el projecte i es mostren les conclusions.

- Possibles millores i treball futur, on es tracten les alternatives que es podrien haver pres i que podrien millorar el rendiment i fiabilitat d'IPoL.

2 Definició de l'escenari

L'arquitectura escollida és la punt a punt, ja que una amb més de dos sistemes, incrementaria el pressupost i la complexitat de la prova de concepte. Per tant, s'ha escollit una arquitectura punt a punt on només intervenen dos nodes, això facilita molt el desenvolupament i simplifica el protocol, ja que, per exemple, només es podran provocar col·lisions entre dos dispositius.

Pel que fa a la placa que implementa el protocol, s'ha escollit utilitzar dues Raspberries Pi [6], una Zero i una Zero W, ja que tot i oferir menor còmput que altres similars, l'ofereixen a un preu molt reduït i un consum energètic molt baix. Aquestes plaques treballen a una freqüència de 1GHz. Cal destacar que el protocol hauria de permetre a diferents plaques, que funcionin a diferents freqüències, comunicarse sense problemes amb el mínim d'ajusts possibles.

Per interactuar amb els components es presenten dues alternatives. La primera consisteix a utilitzar els pins GPIO de la Raspberry. La segona és utilitzar plaques Arduino per sobre la Raspberry, connectades mitjançant un cable sèrie. Això permetria implementar el protocol en temps real, mentre que utilitzar els GPIOs obliga a implementar-lo sobre software, és a dir, tenir dependències amb el comportament del Sistema Operatiu. S'ha escollit utilitzar els GPIOs per evitar l'increment de complexitat i cost que representaria fer servir Arduinos. De totes maneres, en l'apartat 8 s'explica l'avantatge que suposa la implementació del protocol en temps real.

El sensor escollit per detectar la lluminositat de l'ambient és un TSL2561 [7], que és més precís i sensible que fotodíodes estàndards i a més incorpora un convertidor analògic a digital (ADC) que converteix el corrent del fotodíode a digital. Aquesta mesura es pot convertir a la unitat de lluminositat més comuna, el Lux, segons unes fórmules que vénen donades en el *datasheet* del sensor. Es vol evitar el temps de conversió a Lux, per tant, s'utilitzaran les lectures digitals de l'ADC directament. Com no existeix nom per aquestes unitats es farà referència a elles com a *Brightness Digital Mesure* (BDM).

Pel que fa a la font lluminosa, s'utilitzarà un LED estàndard de color vermell. La Raspberry permet utilitzar *Pulse-Width Modulation* (PWM), per realitzar variacions en la intensitat del LED i així poder incloure més nivells en la transmissió. Tot i això, l'apartat 3.2.1 explica perquè finalment no s'ha realitzat un protocol multinivell. El LED anirà connectat a una resistència de 330 ohms, per limitar el corrent d'aquest a un valor segur.

En resum, cada un dels nodes consistirà en:

- Raspberry Pi Zero (W) amb el sistema operatiu Raspbian i la versió de Kernel 4.4.50+. Les aplicacions estan desenvolupades en Python 2.7.9 fent ús de les llibreries *smbus*, que implementa el protocol I2C, i *RPi.GPIO*, per interactuar amb els GPIOs.

- Un sensor TSL2561.
- LED vermell estàndard.
- Resistència de 330 ohms.

La figura 1 mostra les connexions entre els diferents components i la figura 2 mostra la implementació final de l'escenari amb els dos dispositius.

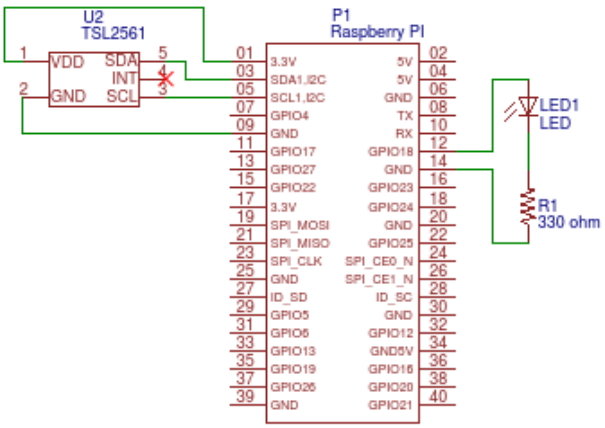

Fig. 1: Circuit electrònic d'un node. La Raspberry es connecta al TSL2561 mitjançant els pins de 3.3V, SDA, SCL i GND i al LED mitjançant el pin GPIO18.

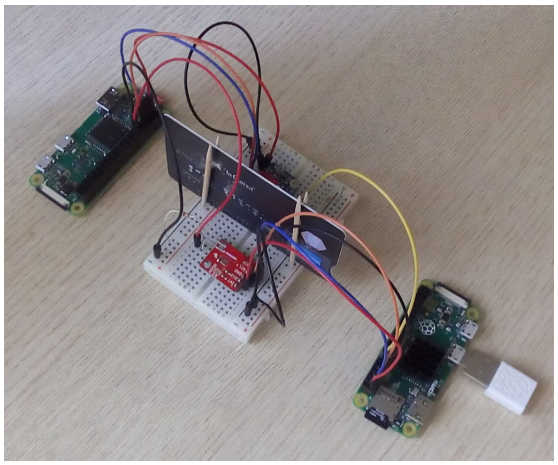

Fig. 2: Implementació de l'escenari. Dues Raspberries es connecten als sensors i leds mitjançant una protoboard i *jumper wires*. L'estructura opaca entremig separa els dos canals de comunicació.

2.1 Funcionament TSL2561

El TSL2561 es comporta de forma similar al d'una càmera fotogràfica. Disposa de dos paràmetres que cal configurar: el guany i el temps d'integració. El guany pot prendre els valors *x16*, si es vol alta sensitivitat (adequat per condicions amb baixa lluminositat), o *x1*, per baixa sensitivitat (millor per condicions amb alta lluminositat). El temps d'integració representa la quantitat de temps que el sensor dedica a fer una captura, per tant quan comença un cicle d'integració comença una captura, i quan es finalitza el cicle s'acaba la lectura i es guarda el valor llegit, en unitats derivades del *lux*, en un registre.

Per tal de fer les lectures de lluminositat, s'han de realitzar un seguit d'operacions, interactuant amb el sensor TSL2561 mitjançant el protocol I2C. El cicle de la lectura comença enviant una comanda al sensor per iniciar el cicle d'integració, a continuació es fa una pausa amb la durada del temps d'integració desitjat. Passat aquest temps s'envia una comanda per acabar el cicle i es fa una lectura al registre del sensor, que conté el valor llegit. L'algorisme per realitzar una captura és el següent:

```
startIntegrationCycle
wait integrationTime
stopIntegrationCycle
readValue
```

El temps d'integració determina aproximadament la freqüència a la qual funciona el protocol, per exemple, si el temps d'integració és de 25ms, la freqüència serà aproximadament de $\frac{1}{25ms} = 40Hz$.

2.2 Algoritmes bàsics de comunicació

Aquest apartat mostra de forma simplificada els cicles de lectura i escriptura que s'han de realitzar per tal d'escriure o llegir informació mitjançant el LED i el sensor.

2.2.1 Emissor

L'emissor itera sobre els bits a enviar i els transmet un a un. Per fer-ho, posa la intensitat del LED al 100%, si es tracta d'enviar un 1, o al 0%, si ha d'enviar un 0 i seguidament s'espera durant un període de temps d'integració + temps de deriva. L'apartat 3 explica què és el temps de deriva.

```
for i in range(len(toSend)):
  if toSend[i] == 0:
    GPIO.output(ledPin, GPIO.LOW)
  else:
    GPIO.output(ledPin, GPIO.HIGH)
  sleep(integrationTime + deriveTime)
```

2.2.2 Receptor

La lectura consisteix a iterar sobre el nombre de bits a llegir. Així doncs, es realitza un cicle de lectura com s'ha explicat en 2.1 i es recupera el nombre de BDMs capturats pel sensor. Per saber si s'està llegint un 1 o un 0 es fan servir dos llindars i el valor llegit anteriorment, pel motiu que s'explica en l'apartat 3.2.

```
for i in range(toRead):
   startIntegration()
   sleep(integrationTime)
   stopIntegration()
   newValue = getLuminosity() # in BDMs
   if newValue > upperThreshold:
      # A 1 was read
      oldValue = 1
   elif newValue > lowerThreshold:
      if oldValue == 0:
         # A 1 was read
         oldValue = 1
      else:
         # A 0 was read
```

```
        oldValue = 0
    else:
        # A 0 was read
        oldValue = 0
```

3 Limitacions i problemes

Les decisions preses en la definició de l'escenari a part de presentar alguns avantatges respecte a la simplicitat, el cost o el consum energètic, presenten diversos desavantatges que s'han hagut de tenir en compte a l'hora de dissenyar i implementar el protocol.

3.1 Temps de deriva

El primer a notar és que s'està implementant un protocol que hauria de ser en temps real dins un sistema que no ho és. El problema aquí és que els processos que controlen la transmissió de dades comparteixen temps del processador amb els altres processos del sistema. Per tant es presenten, entre d'altres, algunes dificultats per aconseguir que els dos nodes se sincronitzin correctament.

Anomenem temps de deriva a la diferència de temps que hi ha entre el processament que fa el receptor per fer una lectura i el que fa l'emissor per fer una escriptura. El temps que el sensor està fent la lectura és el temps que dura la instrucció *sleep(integrationTime)*. Les altres instruccions però, requereixen un temps d'execució, per tant les lectures no es fan cada *x temps d'integració*, sinó que es fan cada *x temps d'integració + temps d'execució de les instruccions*. La figura 9 de l'annex A.1 mostra de forma visual aquesta descripció.

Com el receptor ha de realitzar diferents càlculs, per cada iteració es dessincronitza lleugerament respecte a l'emissor. La solució és allargar les iteracions de l'emissor amb aquest temps de deriva.

La freqüència doncs, no només depèn del temps d'integració, sinó que el temps de deriva també influeix, ja que es faran escriptures i lectures amb una freqüència de $\frac{1}{temps\ d'integracio + temps\ de\ deriva}$ (on els temps es representen en segons i la freqüència en hertz) aproximadament.

3.2 Desincronització inicial

Aquest problema sorgeix pel funcionament del sensor, que per tal de fer una lectura requereix un temps d'integració, és a dir, un temps d'espera. Idealment esperem que el cicle de lectura del receptor comenci exactament a la mateixa vegada que el cicle d'escriptura de l'emissor. La figura 10 de l'annex A.1 mostra de forma visual aquest cas. A la realitat però, aquest comportament és improbable. Normalment l'emissor comença a transmetre la informació en qualsevol moment, en el pitjor dels casos començarà el seu cicle a la meitat del del receptor.

La millor forma de representar-lo és mitjançant una línia temporal, ignorant el temps de deriva explicat prèviament. La figura 3 mostra aquest cas. Tal com es veu, l'emissor comença a escriure a la meitat del cicle de lectura, per tant, quan s'acaba la primera lectura, no és clar si s'ha rebut un 1 o un 0, ja que la meitat del temps s'ha estat amb el LED al 0% i l'altra meitat al 100%. El mateix passa amb els següents cicles.

La solució proposada a aquest problema és la utilització d'una variable que emmagatzema l'últim bit llegit i dos valors llindars en BDMs, un lleugerament inferior al valor que s'obté amb una lectura del 50% del temps, i un altre lleugerament superior. El cicle de lectura que mostra l'apartat 2.2.2 el que es fa és mirar si els BDMs captats són més que el llindar més petit. Si també és superior al llindar gran, està clar que s'ha llegit un 1. En cas contrari és quan s'entra en conflicte i es fa ús del bit anterior. Mai es donarà el cas que dos bits iguals consecutius, tinguin com a resultat un valor de lluminositat aproximat al que s'obté amb un 50%, per tant simplement s'ha d'agafar el contrari que l'anterior.

Les figures 11 i 12 de l'annex A.1 mostren aquest comportament.

3.2.1 Multinivell amb PWM

Aquest problema dificulta molt la utilització d'un protocol multinivell, ja que a l'hora de fer les lectures, si es rep un valor en BDMs que es correspon al 50%, no es podrà saber si s'ha rebut per aquest problema o perquè el LED ha transmès el bit a un altre nivell. És per aquest motiu que el IPoL només té dos nivells, un per l'1 (100% d'intensitat) i un altre pel 0 (0% d'intensitat).

4 Definició del protocol

Aquest apartat pretén definir el funcionament d'IPoL juntament amb els paràmetres amb els quals treballa i com seleccionar-los.

4.1 Condicions prèvies

Un protocol que utilitzi senyals lluminosos per la transferència de dades requereix que els dispositius estiguin corrent en un ambient controlat, en el qual no pot haver-hi canvis de lluminositat, ja que aquests canvis es podrien entendre com a variacions intencionades per part de l'emissor. És per això que els dispositius que implementin IPoL requereixen estar en, per exemple, una habitació tancada, amb una o més fonts d'il·luminació que no variïn de posició ni d'intensitat, això inclou qualsevol dispositiu mòbil que pugui apuntar directament cap a algun dels sensors.

4.1.1 Paràmetres

Un cop es compleixen les condicions especificades, cal determinar un seguit de valors, o paràmetres, que prendrà IPoL.

- Temps d'integració. Explicat en l'apartat 2.1. Aquest paràmetre ha de ser el mateix per tots dos dispositius.
- Llindars. Valors que marquen la diferència entre rebre un 0 o un 1, explicats en l'apartat 3.2. Varien entre dispositius, ja que la mínima variació en la localització de cada sensor fa que els valors de lluminositat llegits variïn.

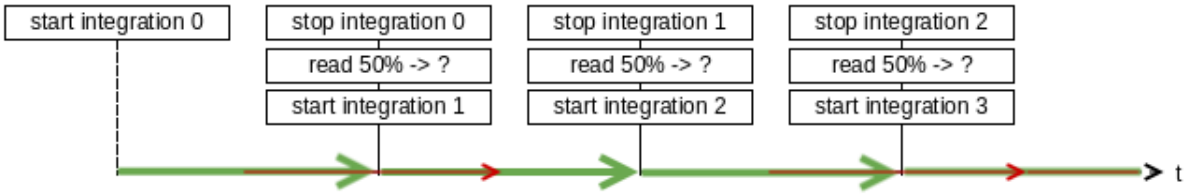

Fig. 3: Línia temporal que mostra el pitjor cas de desincronització entre el receptor i l'emissor. Les línies verdes representen els cicles de lectura que realitza el receptor, i les línies vermelles representen els cicles d'escriptura de l'emissor quan es transmet un 1, és a dir, els moments en els quals el LED està encès.

- Temps de deriva. Explicat en l'apartat 3.1. Aquest valor depèn de la velocitat de processament de l'altre dispositiu.

Cal destacar que la distància entre el sensor i el LED pot variar, sempre que es calculin els llindars per aquella distància, i la llum d'un LED no interfereixi en els dos sensors a la vegada, és a dir, que es presenti un escenari full-dúplex.

4.2 Especificacions

IPoL és un protocol que es troba en la capa Física/Enllaç de dades, per tant, és l'encarregat d'enviar els bits que li arribin de les capes superiors pel medi, en aquest cas, l'aire. La figura 4 mostra la seva localització en la pila TCP/IP.

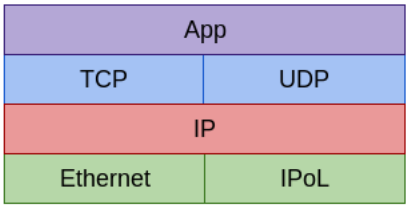

Fig. 4: Pila TCP/IP amb els protocols en Ethernet i IPoL per la capa d'enllaç de dades/física.

Es tracta d'un protocol que no requereix connexió, és a dir, es comporta com Ethernet, que envia els missatges pel medi i no es preocupa de si hi ha algú que els escolti o no. A més, és punt a punt, per tant, només dos dispositius es podran comunicar utilitzant el mateix enllaç. Pel que fa a la configuració de l'enllaç, IPoL és full-dúplex i està pensat de manera que per una de les connexions un dels dispositius faci de receptor i l'altre d'emissor, i en l'altra connexió sigui a la inversa. En cap cas pot haver-hi interferències entre totes dues connexions.

Els problemes de sincronització esmentats en la fase de desenvolupament anterior, fan que aplicar un mètode de modulació amb més de 2 nivells, és a dir, transmetre més d'un bit per cicle, sigui totalment inviable. Per tant, IPoL només suporta la lectura de dos nivells diferents, o 0 o 1.

Els paquets consisteixen d'un byte que representa la mida de les dades a enviar, un *checksum* per la detecció d'errors i finalment el *payload*.

Per començar la transmissió s'envia un bit d'inici, i abans d'enviar els paquets s'escriu un byte de preàmbul, per tal que el receptor vegi si està rebent la informació de forma correcta. Així doncs, la seqüència enviada és tal com es mostra en la taula 1.

Cal destacar, que per evitar problemes de sincronització, s'han afegit uns bits de resincronització, que s'envien intercalats amb les dades. És a dir, que realment el *payload* pot arribar a contenir més de 32 octets si es conten els bits de resincronització. L'apartat de sincronització ho explica en més detall.

La màxima unitat de transferència (MTU) és de 96 bytes, pel motiu que s'explica en l'apartat 6.3.

IPoL és poc fiable respecte a la seguretat que arribi un paquet al destinatari, aquesta tasca es deixa a protocols de més alt nivell, si escau.

4.3 Sincronització

A l'hora d'enviar i rebre els missatges, tant l'emissor com el receptor han d'estar sincronitzats per saber en quins intervals de temps s'està transferint informació. És per això, que primer l'emissor envia un bit d'inici. Aquest bit consisteix a encendre el LED al màxim durant un període igual al del temps d'integració + temps de deriva. Això indica al receptor que comença la transmissió.

A continuació s'envia la seqüència de sincronització 11010011, per tal que el receptor pugui comprovar que a part d'entendre que comença la transmissió, pot llegir un byte sense obtenir un resultat erroni. Així es proporciona sincronització a nivell de byte.

Es pot dir doncs, que abans d'enviar un missatge s'envia un preàmbul format per 9 bits. La seqüència 111010011 marca l'inici del paquet. Així s'evita que es facin lectures errònies ja des del principi, o que una pertorbació lluminosa en l'ambient resulti en el receptor començant a llegir, quan l'emissor no està enviant res.

Un dels problemes presentats en l'apartat 3 és el temps de deriva. Aquest es pot solucionar de forma bastant senzilla, afegint el temps de deriva al cicle de l'emissor, quan no hi ha gaires processos executant-se o les Raspberries funcionen a una freqüència similar. Quan comencen a treballar a freqüències diferents, però, l'escenari canvia i és necessari afegir una altra mesura, ja que el temps de deriva no sempre és constant.

Per tant, el protocol compta amb un altre mètode de sincronització, l'ús d'un bit de resincronització. Cada X bytes (variable segons les característiques de l'entorn), l'emissor s'espera dues vegades el temps d'integració i envia un bit de resincronització, per evitar que hi hagi un desfasament entre cicles, de l'emissor i el receptor, de més del 50% del temps d'integració. El receptor va iterant i cada X bytes s'espera a llegir el bit de resincronització. El següent codi mostra un cicle d'escriptura en el qual es realitza la tasca de resincronització.

TAULA 1: SEQÜÈNCIA D'ENVIAMENT D'UN MISSATGE

Bit inici	Preambul	Mida dades	Checksum	Dades
1 bit	1 octet	1 octet	1 octet	0-96 octets

```
for byte in bytearray:
   if resync == RESYNC_NUM_BYTES:
      GPIO.output(LED_PIN, GPIO.LOW)
      resync = 0
      sleep(INTEGRATION_TIME*2)
      GPIO.output(LED_PIN, GPIO.HIGH)
      sleep(INEGRATION_TIME_WITH_OFFSET)
   resync += 1
   ipolSendByte(byte)
```

Això permet que tots dos dispositius es tornin a sincronitzar a mig enviament de dades i mai arribi a haver-hi un temps de dessincronització superior a la meitat del temps d'integració.

4.4 Detecció d'errors

Tal com mostra l'apartat de resultats IPoL és un protocol molt insegur, ja que presenta una taxa d'error elevada. És per això que els missatges enviats contenen un byte de *Frame Check Sequence*, és a dir, un *checksum* per tal de poder detectar els errors. Aquest *checksum* es correspon al complement a dos, del byte menys significatiu, del sumatori de tots els bytes. El seu càlcul es realitza de la següent forma:

$$checksum = (\sim (sum(byteArray)\&0xFF)+1)\&0xFF$$

Cal destacar però, que el protocol només permet la detecció d'errors, i no la correcció. Per tant, qualsevol missatge que rebi el receptor i el *checksum* no es correspongui amb el rebut en la capçalera del missatge, es descarta.

4.5 Detecció de col·lisions

IPoL és un protocol full dúplex, on no es presenten col·lisions, ja que en un canal un dels dispositius fa d'emissor i l'altre de receptor, i en l'altre canal es fa a la inversa. Per aconseguir aquest escenari full dúplex s'ha de col·locar una superfície opaca entre els dos canals, tal com es veu en la figura 2.

5 Implementació

El protocol s'ha implementat en user space en dispositius amb la versió 4.4.50+ del Kernel de Linux i la versió 2.7.9 de Python. A més, s'han utilitzat eines específiques de sistemes encastats, per facilitar el desenvolupament. Tot i això, no s'han utilitzat eines per compilació creuada, ja que només existeix un fitxer que requereixi compilar-se.

5.1 Interfície tun/tap

Les interfícies tun/tap [8] són una implementació d'un driver de xarxa només en software. Permeten que els usuaris puguin interactuar amb paquets de xarxa en user space, per tal de fer el que es vulgui amb ells. Aquestes interfícies funcionen exactament igual que les altres, però amb la peculiaritat que redirigeixen el tràfic cap a les aplicacions (en user space) que estan enganxades a elles.

La diferència entre tun i tap, és que la primera intercepta els datagrames IP, mentre que la segona intercepta les trames ethernet. És per aquest motiu que la utilització de la interfície tun és l'opció més adient, en comparació amb la de la creació d'un driver de xarxa o la utilització de tap, per tal d'implementar IPoL en user space.

Per fer-la servir, simplement s'ha d'iniciar el mòdul de kernel, instanciar-la dins l'aplicació i posar-la en estat UP. A més, permet posar-la en mode punt a punt, com es faria amb qualsevol altra interfície. A continuació es mostra com s'instància:

```
ip link set tun0 mtu 96 up
ip addr add 10.0.0.1 dev tun0 peer 10.0.0.2
```

La figura 5 mostra l'execució de la comanda $ip\ a\ s$ per mostrar la interfície un cop s'ha instanciat.

5.2 Disseny

S'ha desenvolupat un seguit de fitxers per implementar el protocol i que s'executen en user space. Aquests són els següents:

- tun.c: inicia la interfície tun i s'encarrega de tota la comunicació amb ella. Per tant, és el fitxer que fa read/write al descriptor del fitxer que apunta a *tun0*.
- ipol.py: implementa el protocol, i conté mètodes a mode llibreria tant per rebre com per enviar informació mitjançant aquest. Per tant, interactua amb el sensor i amb el LED mitjançant llibreries de GPIO i I2C.
- sender.py: conté una cua que emmagatzema els datagrames rebuts, un cop es rep algun, el thread encarregat d'enviar-lo fa les crides necessàries a ipol.py.
- receiver.py: espera a rebre informació mitjançant el sensor, fent les crides necessàries a ipol.py. Quan rep un nou paquet, el posa en una cua, i un altre thread s'encarrega de transmetre els datagrames a tun.c.

La figura 6 mostra els elements que interactuen en el procés de comunicació.

Al fer un *ping*, el programa tun.c intercepta el datagrama i l'envia cap a sender.py perquè ho enviï mitjançant el LED. Aquest fa la crida a la funció *ipolSend()* que apareix dins el paquet *ipol.py*.

A l'altra banda, el receptor ha cridat al mètode *ipolRecv()* que estarà fent *polling* al sensor, per esperar rebre el bit d'inici i començar la lectura d'un missatge. Quan el llegeix el transfereix cap a *tun.c* que s'encarrega de transmetre'l

```
pi@rpil:~ $ ip a s dev tun0
3: tun0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 96 qdisc pfifo_fast state UNKNOWN group default q
len 500
    link/none
    inet 10.0.0.1 peer 10.0.0.2/32 scope global tun0
       valid_lft forever preferred_lft forever
```

Fig. 5: Resultat de la comanda *ip a s dev tun0*, on es mostra l'adreça ip del dispositiu, la de l'altre node, l'MTU i les característiques de l'interfície (punt a punt, no-arp, up).

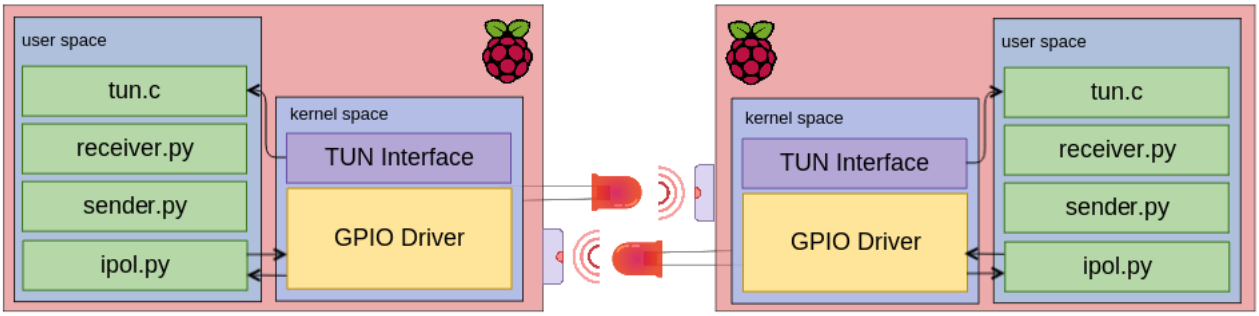

Fig. 6: Estructura del projecte, mostrant els diferents dispositius i els fitxers que executen.

a la interfície *tun0*. El programa que implementa el ping genera la resposta i l'envia de la mateixa forma que l'altre node.

L'annex A.2 mostra un diagrama de seqüència amb els passos descrits.

6 Resultats

S'han realitzat un seguit de proves per tal de caracteritzar el protocol per tenir una visió global de les possibles aplicacions d'aquest i ajudar a decidir valors com el seu MTU.

Els paràmetres utilitzats influeixen en els resultats, és per això que s'ha escollit fixar-los. El temps d'integració és de 150ms i s'envien 16 bytes abans de realitzar la resincronització. Tots els altres s'han ajustat segons aquests mitjançant uns scripts. Els llindars s'han calculat mitjançant el mètode de prova i error segons la lluminositat en l'ambient. Pel primer node, que captava 76 BDMs en l'ambient han estat de 425, 430, mentre que pel segon node, que en captava 81 han estat 356, 361. El temps de deriva resultant és de 2ms.

Les proves han consistit a enviar 15 paquets, mitjançant la comanda *ping -c 1 -s X -w 500 10.0.0.X*, per cada una de les mides 32, 64, 96, 128 i 160 bytes anotant quants d'ells arribaven correctament al destinatari, els temps que han tardat a transmetre's i les temperatures dels dispositius en cada moment.

6.1 Taxa d'error

La taula 2 mostra els resultats obtinguts segons les proves realitzades i la figura 14 de l'annex A.3 mostra la seva gràfica.

La taxa d'error augmenta segons la quantitat de bytes. Com ja s'intuïa, la probabilitat que un bit es llegeixi de forma incorrecta augmenta segons el pas del temps. Per tant, enviar més bytes augmenta la possibilitat que algun arribi malament.

Pels valors baixos es manté bastant estable però a partir dels 64 bytes comença a augmentar d'una forma lineal, de

TAULA 2: TAXA D'ERROR SEGONS LA MIDA DEL PAQUET

Mida paquet (bytes)	Taxa d'error (%)
32	33.3
64	33.3
96	40.0
128	60.0
160	80.0

20% en 20% per cada 32 bytes. Possiblement aquest comportament es dóna perquè les Raspberries a partir d'un cert temps processen a diferents freqüències, de forma que arriba a influenciar en la transmissió.

6.2 Temps de transmissió

Per aquesta prova s'han agafat els temps de transmissió per cada un dels paquets, tant si han arribat correctament com si no, de les proves anteriors. No s'inclouen els paquets de 160 bytes, ja que presenten una taxa d'error massa gran i per això s'ha preferit treballar amb els paquets de fins a 128 bytes.

La velocitat de transmissió hauria de ser un valor constant al nombre de bytes a enviar, per tant s'hauria de presentar una funció lineal. La taula 3 mostra els resultats obtinguts.

Es compleix la hipòtesi i el temps de transmissió varia de forma totalment lineal segons la quantitat de bytes enviats, tal com es mostra a la figura 15 de l'annex A.3.

Les velocitats aconseguides són d'aproximadament $\frac{96bytes}{123segons} = 0.78bytes/segon$, per tant, no s'arriba a transmetre ni tan sols un byte per segon, tot i això, no s'ha definit cap requisit per la velocitat de transmissió de IPoL.

6.3 MTU

Les proves de l'apartat 6.1, on es calcula la taxa d'error per les diferents mides de paquets possibles i l'RTT obtingut

TAULA 3: TEMPS DE TRANSMISSIÓ PER MIDA DE PAQUET

Mida paquet (bytes)	Temps (segons)
32	43
64	83
96	123
128	163

dels paquets, ens permeten fer una estimació de quin és el MTU adequat per utilitzar TCP sobre IPoL.

Per realitzar l'estimació s'ha adaptat la fòrmula de Mathis [9] de la següent forma:

$$BW = \frac{MSS}{RTT}(1 - \frac{Taxa\ d'error}{100})$$

que indica el màxim rati de transmissió que es pot arribar a aconseguir en TCP amb les característiques del protocol i/o xarxa que es vol avaluar, tenint en compte la taxa d'error.

La taula 4 mostra les característiques obtingudes durant les proves i el límit teòric de TCP calculat amb la fórmula anterior, marcant el valor *C* com 1 per simplicitat. Així doncs, l'MTU ideal en aquest cas és el de 96 bytes, ja que presenta una ràtio possible més elevada. Cal destacar però, que els càlculs s'han simplificat per reduir la complexitat. Per exemple, s'ignoren les esperes en la cua de sortida.

Veiem doncs, que utilitzar TCP per sobre d'IPoL, amb aquestes característiques, proporciona una velocitat de màxim 1.09 bits/s.

6.4 Temperatura

A continuació s'avalua si la temperatura de les CPUs de les Raspberries, quan s'envien i reben els paquets, influeixen a l'hora d'obtenir els missatges correctament, ja que sabem que la freqüència de processament en aquests dispositius varia segons la temperatura a la qual es troben. Només s'han tingut en compte els paquets de 64 bytes.

La figura 7 mostra la gràfica de temperatura dels dispositius en el moment d'enviar els paquets. A primera vista sembla que no influeix en el resultat, és a dir, que un paquet arribi correctament no depèn de la temperatura a la qual es troba la CPU. Tot i això, sabem que la freqüència de processament d'una Raspberry varia segons la temperatura. Per tant, s'ha intentat comprovar el mateix mirant la diferència de temperatura entre ambdós dispositius.

La figura 8 mostra la gràfica de diferència de temperatures per cada paquet, és a dir, el resultat d'aplicar $TemperaturaEmissor - TemperaturaReceptor$ en cada una de les transmissions. Per tant, els valors positius en l'eix Y indiquen que l'emissor era el que estava a una temperatura major i els negatius que ho estava el receptor.

La gràfica mencionada aporta informació essencial per entendre el comportament del protocol. Per començar, mostra que en la majoria dels casos, 11 de 15, el receptor té la temperatura més alta, possiblement perquè ha de realitzar més operacions que l'emissor. Mentre un ha d'interactuar mitjançant I2C i realitzar operacions sobre els resultats d'aquestes, l'altre només ha d'enviar un senyal al LED per indicar-li que s'encengui o s'apagui.

Una altra cosa a veure, i la més important, és que la possibilitat que un paquet arribi correctament quan l'emissor està a una temperatura més alta, és del 100% i considerablement major a què arribi correctament quan el receptor està més calent (45.5%). Possiblement perquè al escalfarse més al receptor li costa sincronitzar-se correctament amb l'emissor.

6.5 Resum

Els resultats mostren que el protocol té una taxa de fiabilitat del 60% quan s'envien dades amb el seu MTU al màxim. Per paquets amb mides més petites, aquesta taxa augmenta fins al 77.7%.

Pel que fa a la velocitat de transmissió, com és un protocol bit a bit, el temps augmenta com més gran és el paquet, d'una forma lineal, tal com es preveia. S'ha assolit una velocitat del voltant de 0.78 bytes/segon.

Aquestes proves han permès determinar que 96 bytes és el MTU més adequat per IPoL amb els paràmetres definits i que la ràtio de transmissió més elevada possible per TCP és de 1.09 bits per segon, a causa de la sobrecàrrega que impliquen els 40 bytes de capçaleres.

Finalment, l'anàlisi realitzat respecte a les temperatures, mostra que la possibilitat que un paquet arribi correctament quan l'emissor està a major temperatura que el receptor és més gran que si fos a la inversa. Aquests resultats remarquen els problemes de sincronització que presenta el protocol, i que són més perceptibles quan el receptor treballa a una freqüència més baixa a la de l'emissor.

En la definició de l'escenari s'ha dit que el protocol hauria de poder funcionar entre dispositius que treballin a diferents freqüències, tot i això, aquests resultats mostren la dificultat d'assolir-ho, ja que no hi ha prou en conèixer el temps de deriva. Aquest és un dels motius pels quals se solen fer servir sistemes a temps real per la transmissió de dades.

7 Conclusions

Durant el transcurs d'aquest projecte, s'ha pogut observar el desenvolupament d'un protocol de xarxa, per transmetre datagrames IP mitjançant senyals lluminosos, i la seva implementació en sistemes amb poca capacitat computacional com les Raspberries. S'han vist les dificultats que suposa implementar-lo en user space i la importància que pren la sincronització en aquestes aplicacions. A continuació es mostren els subobjectius plantejats a l'inici del projecte i com s'han resolt.

- Definició del sistema de comunicació. Aquest objectiu s'ha pogut complir definint l'escenari de desenvolupament i fent les primeres proves de caracterització del comportament dels components.

- Definició del protocol. A partir dels avenços anteriors, s'ha pogut definir un protocol de comunicació que utilitzes senyals lluminosos i l'aire com a medi de transmissió. Això ha resultat en un seguit d'especificacions que permeten implementar-lo en diferents escenaris, sempre que es compleixin les condicions esmentades.

TAULA 4: LÍMIT TEÒRIC DE TCP CALCULAT A PARTIR DE LA FÓRMULA DE MATHIS I LES CARACTERÍSTIQUES D'IPOL PELS DIFERENTS MTUS, AGAFANT C COM A 1.

MTU (bytes)	MSS (bytes)	RTT (ms)	Taxa d'error (%)	Límit teòric TCP (bit/s)
64	44	166000	33.3	0.77
96	76	246000	40.0	1.09
128	108	326000	60.0	0.86

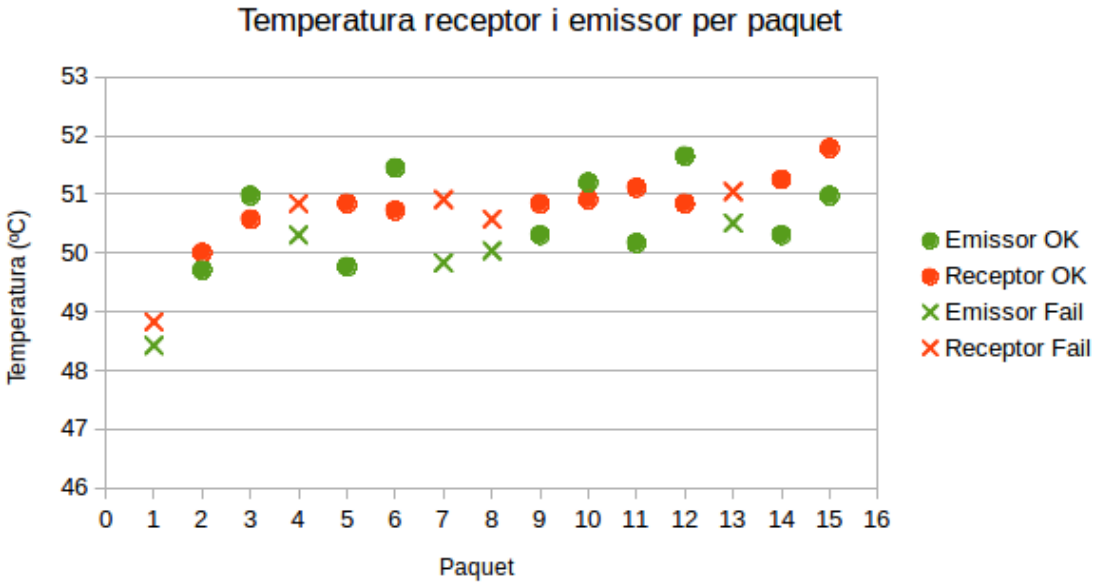

Fig. 7: Temperatures mitjanes de les CPUs de l'emissor i receptor en els moments que es feia la transmissió per cada un dels paquets de 64 bytes. L'eix X indica el número de paquet i l'eix Y la temperatura mitjana de l'emissor i el receptor. La del receptor es mostra en verd i la de l'emissor en tronja, en tots dos casos, les rodones indiquen que el paquet s'ha transmes correctament i les creus que la transmissió ha fallat.

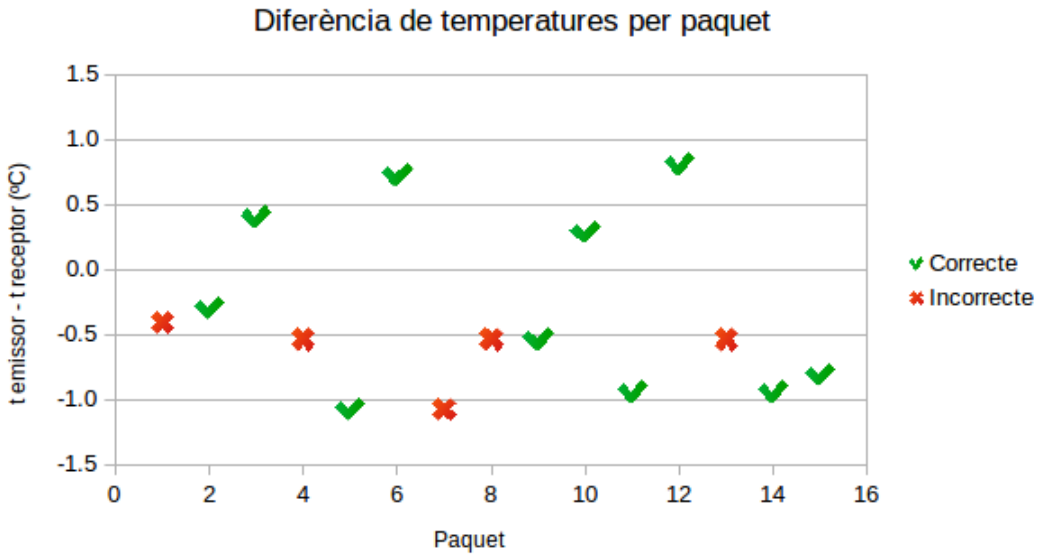

Fig. 8: Diferència de temperatura mitjana que hi ha entre l'emissor i receptor en el moment d'enviar cada un dels paquets de 64 bytes. L'eix X representa el paquet i l'eix Y la temperatura, amb els valors positius indicant que l'emissor la tenia més elevada i els negatius indicant que el receptor tenia major temperatura. Els tics indiquen que el paquet s'ha transmès correctament i les creus que la transmissió ha fallat.

- Implementació del protocol en el sistema. S'ha utilitzat les interfícies de tunneling per interceptar els datagrames IP i s'han redirigit a la nostra aplicació, on s'ha implementat IPoL, per transmetre'ls. El resultat ha estat un seguit de fitxers que permeten a qualsevol persona, amb un sistema com l'utilitzat, transmetre dades entre dispositius utilitzant el protocol definit anteriorment.

- Proves de rendiment. Tal com es veu en l'apartat 6, aquest objectiu ha permès analitzar el funcionament del protocol per determinar les seves característiques i entendre millor el comportament de la implementació.

El compliment d'aquests, ha permès donar per tancat l'objectiu principal del projecte, creació i implementació d'un protocol de comunicació mitjançant senyals lluminosos. Així doncs, qualsevol persona que reprodueixi els passos seguits durant el transcurs del treball hauria de poder utilitzar IPoL en els seus dispositius.

Aquest treball mostra la creació d'una prova de concepte, que un cop millorada i posada en un entorn de producció donaria pas a algunes utilitats que ara mateix no es podrien suportar.

Una de les utilitats que podria tenir IPoL, és la de proporcionar transmissió de dades en llocs on es vulgui mantenir el camp electromagnètic i no es puguin posar cables per ser un edifici històric o per causes estètiques. També pot ser una alternativa si es vol controlar la difusió dels paquets, ja que els senyals no traspassen estructures opaques, i per tant, no es poden capturar des d'altres localitzacions, com passa amb el Wi-Fi.

Algunes aplicacions que tindria doncs, es poden trobar en l'àmbit del IOT, ja que els components requereixen transmissió de dades, i poden estar en gairebé qualsevol lloc. Per tant, es podria oferir com a alternativa en aquests casos. Una altra aplicació seria d'utilitzar-lo en entorns industrials sensibles a interferències electromagnètiques.

8 Possibles millores i treball futur

Donades totes les limitacions esmentades durant el treball, han anat sorgint alternatives que probablement ofereixen unes millors prestacions al protocol. Aquest apartat pretén explicar les més rellevants.

- Implementació en un sistema en temps real. El problema principal que ha presentat el projecte, és treballar sobre un entorn no controlat, en aquest cas les Raspberries, ja que presenten comportaments inesperats.

  Utilitzar un sistema en temps real, permet reduir els problemes de sincronització, ja que les instruccions s'executen en el moment que es criden i no passen per el *scheduler* de Linux.

  Per tant, alguns dels errors que es donaven a causa d'una instrucció tardant més del previst en executar-se, es solucionarien. Un mètode d'implementació seria el que fan altres protocols com Ethernet, que utilitzen targetes de xarxa connectades a l'ordinador mitjançant un port PCI. Així doncs, un driver de xarxa obtindria els paquets IP i els redirigiria a la targeta de xarxa que implementa IPoL.

- Utilització d'interrupcions, per evitar haver d'estar realitzant peticions al sensor per llegir els valors. El TSL2561 permet indicar un llindar, i quan una lectura de lluminositat el supera, es llança una interrupció. Així doncs, es reduiria el temps de processament del receptor, ja que no hauria d'estar fent *polling*.

- Reducció de la llargada del bit d'inici. Tal com s'ha comentat en l'apartat de Sincronització, abans de començar una transmissió s'envia un bit d'inici. El cicle d'aquest bit és igual de llarg que els altres cicles. Si es reduís, es podria aconseguir que els dispositius estiguessin més ben sincronitzats, ja que com més gran és el cicle per escriure un bit la desincronització pot ser major.

- Utilització d'ones infraroges. El TSL2561 quan realitza un cicle d'integració, captura dos valors, un per la llum visible i un altre per les infraroges. Per tant, utilitzant un emissor d'infrarojos, es podrien fer dues lectures diferents en el mateix cicle, cosa que augmentaria la capacitat de transmissió.

Referències

[1] pureLiFi™, "The home of LiFi - pureLiFi™" [Online]. Disponible a: http://purelifi.com/, 2014. [Consultat: Feb. 02, 2017].

[2] A. Agrawal, G. Kumar, M.i Narayan Singh, "Data Transmission Using Laser Light," Galgotias College of Engineering and Technology, Greater Noida, India.

[3] "802.3-2012 - IEEE Standard for Ethernet," Dec. 28, 2012.

[4] [RFC1149] Waitzman, D., "Standard for the transmission of IP datagrams on avian carriers", RFC 1149, DOI 10.17487/RFC1149, Abr. 1, 1990.

[5] BLUG, "RFC-1149," [Online]. Disponible a: http://blug.linux.no/rfc1149/, Abr. 28, 2001 [Consultat: Feb. 19, 2017].

[6] Raspberry Pi. "Raspberry Pi - Teach, Learn, and Make with Raspberry Pi," [Online]. Disponible a: https://www.raspberrypi.org/ [Consultat: Feb. 28, 2017].

[7] Light-to-digital converter. TSL2561. Texas Advanced Optoelectronic Solutions. Nov. 2009.

[8] Waldner. "Tun/Tap interface tutorial" [Online]. Disponible a: http://backreference.org/2010/03/26/tuntap-interface-tutorial/, Mar. 26, 2010 [Consultat: Abr. 24, 2017].

[9] M. Mathis, J. Smeke, J. Mahdavi, "The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm", Pittsburgh Supercomputing Center, 1997.

Apèndix

A.1 Problemes de sincronització

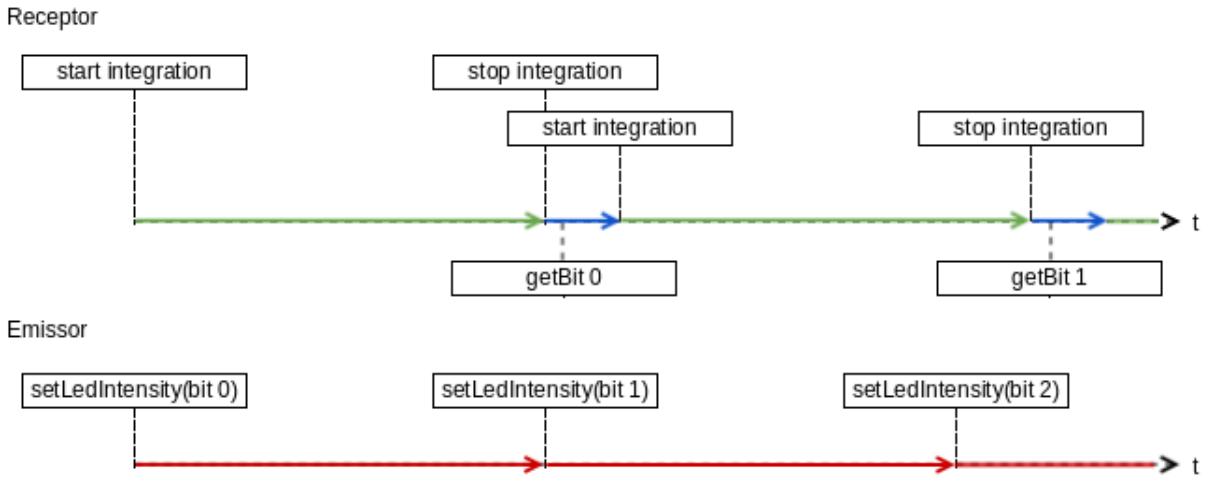

Fig. 9: Representació del temps de deriva en una línia temporal. Les fletxes verdes i vermelles representen el temps que duren els *sleeps* (igual al temps d'integració) en el receptor i l'emissor respectivament, mentre que les fletxes blaves representen el temps de deriva, que es correspon al temps que el receptor tarda a recuperar el valor del sensor i processar-lo.

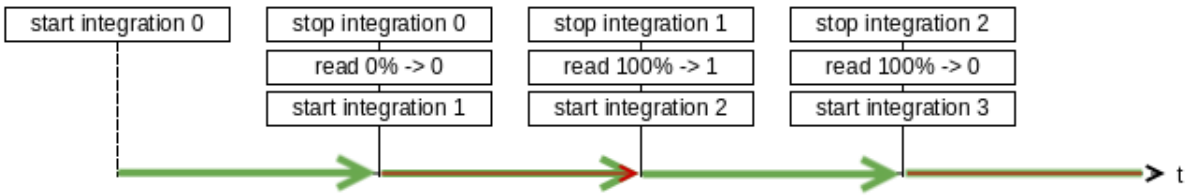

Fig. 10: Cas ideal de sincronització entre el receptor i l'emissor. Les línies verdes representen els cicles de lectura que realitza el receptor, i les línies vermelles representen els cicles d'escriptura de l'emissor quan es transmet un 1, és a dir, els moments en els quals el LED està encès.

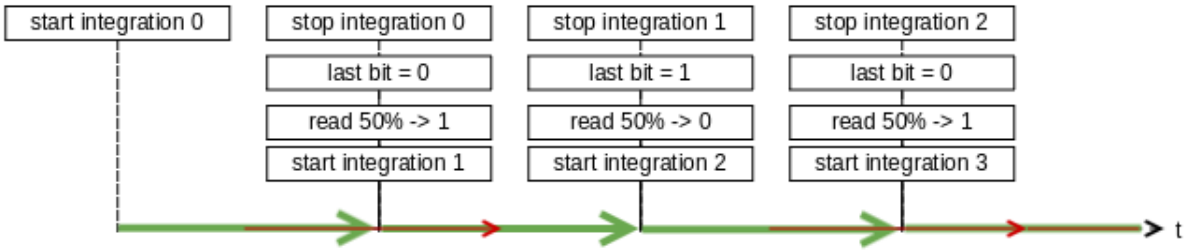

Fig. 11: Línia temporal que mostra la solució al pitjor cas de desincronització entre el receptor i l'emissor. Les línies verdes representen els cicles de lectura que realitza el receptor, i les línies vermelles representen els cicles d'escriptura de l'emissor quan es transmet un 1, és a dir, els moments en els quals el LED està encès. Es comprova si el bit anterior és 0 o 1 per tal de prendre la decisió quan es fan lectures d'aproximadament el 50% de BDMs.

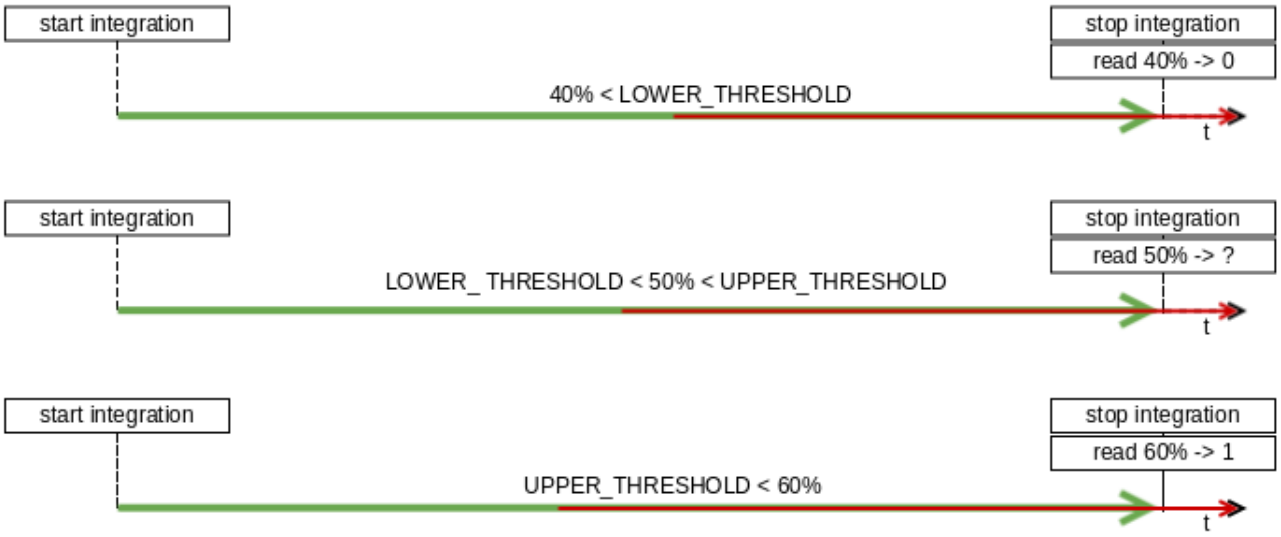

Fig. 12: Línies temporals que mostren tres cicles de lectura on s'han fet lectures del 40, 50 i 60% en BDMs. Les línies verdes mostren el temps que el receptor està fent una lectura i les vermelles el temps que l'emissor està escrivint un 1, és a dir, quan el led està encès. Mostra els tres casos de decisió possibles, un que es troba per sota del llindar inferior, per tant llegeix 0, un que es troba per sobre del superior, per tant llegeix 1 i un que es troba entre els dos llindars, per tant es fa ús del bit anterior per prendre la decisió.

A.2 Implementació proposada

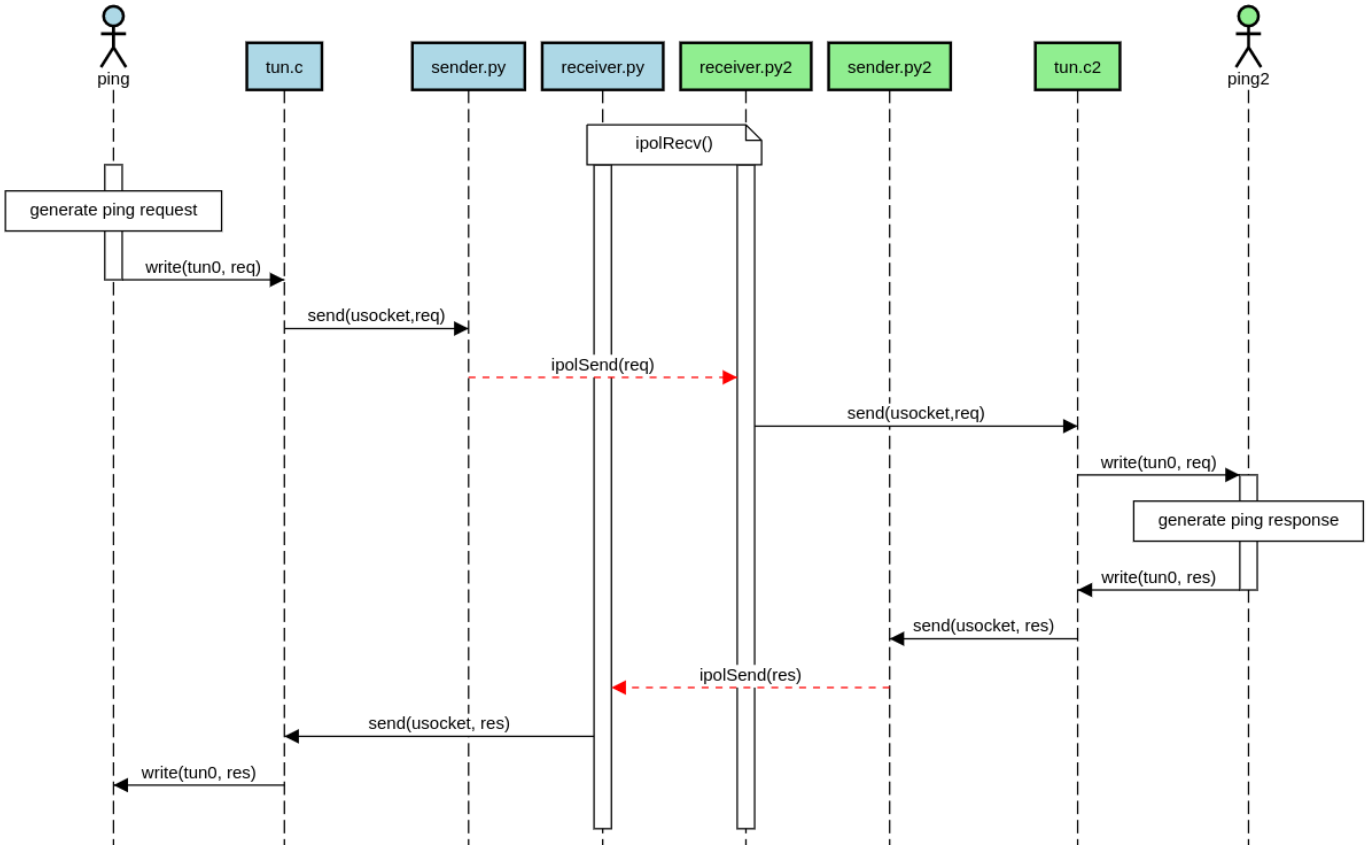

Fig. 13: Diagrama de seqüència de l'execució d'un ping d'un dels nodes a l'altre. Els elements blaus indiquen que pertanyen al node 1, mentre que els elements verds pertanyen al 2.

A.3 Rendiment

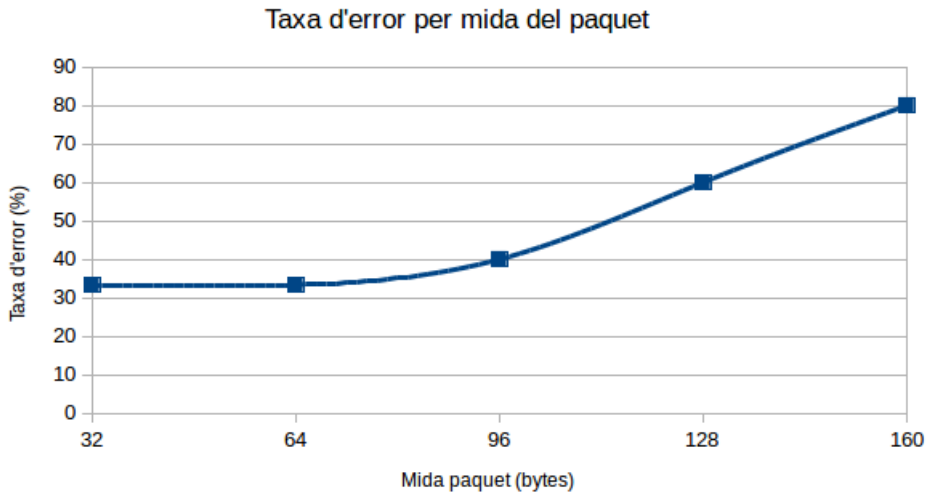

Fig. 14: Taxa d'error calculada a partir de l'enviament de 15 paquets per cada una de les mides.

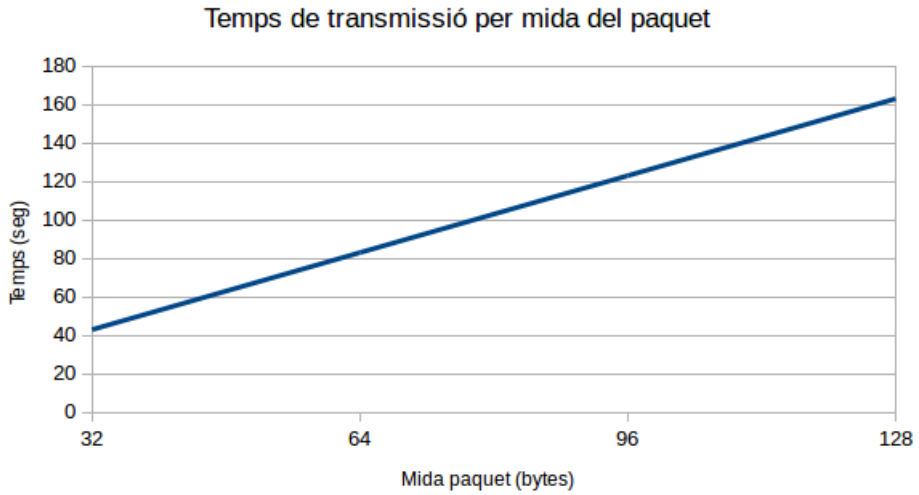

Fig. 15: Temps de transmissió segons la mida de paquet.