Anàlisi de la interconnexió de dispositius lògics programables mitjançant Ethernet

Marko Peshevski

TFM, MUESAEI

Q1 2016 – 2017

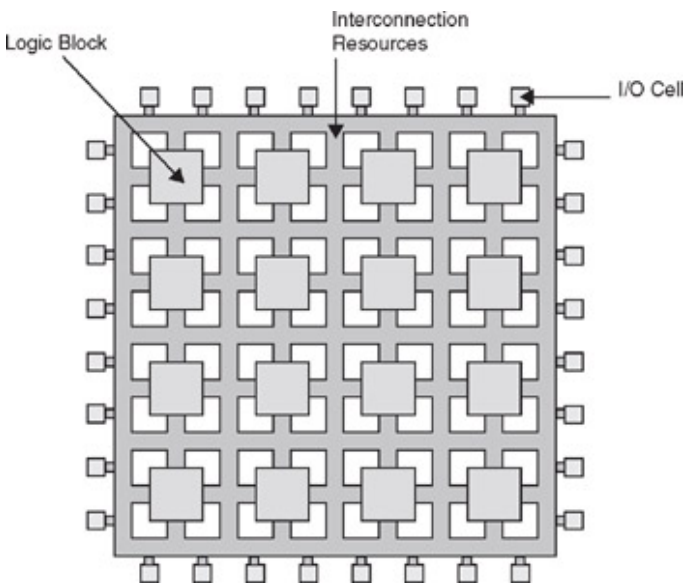
Taula de continguts

Objectius

- Conèixer Ethernet en profunditat
- Conèixer en profunditat alguns protocols de xarxes
- Desenvolupar un sistema incrustat amb Ethernet sobre una FPGA
- Intentar desenvolupar una pila de protocol bàsica

Estructura

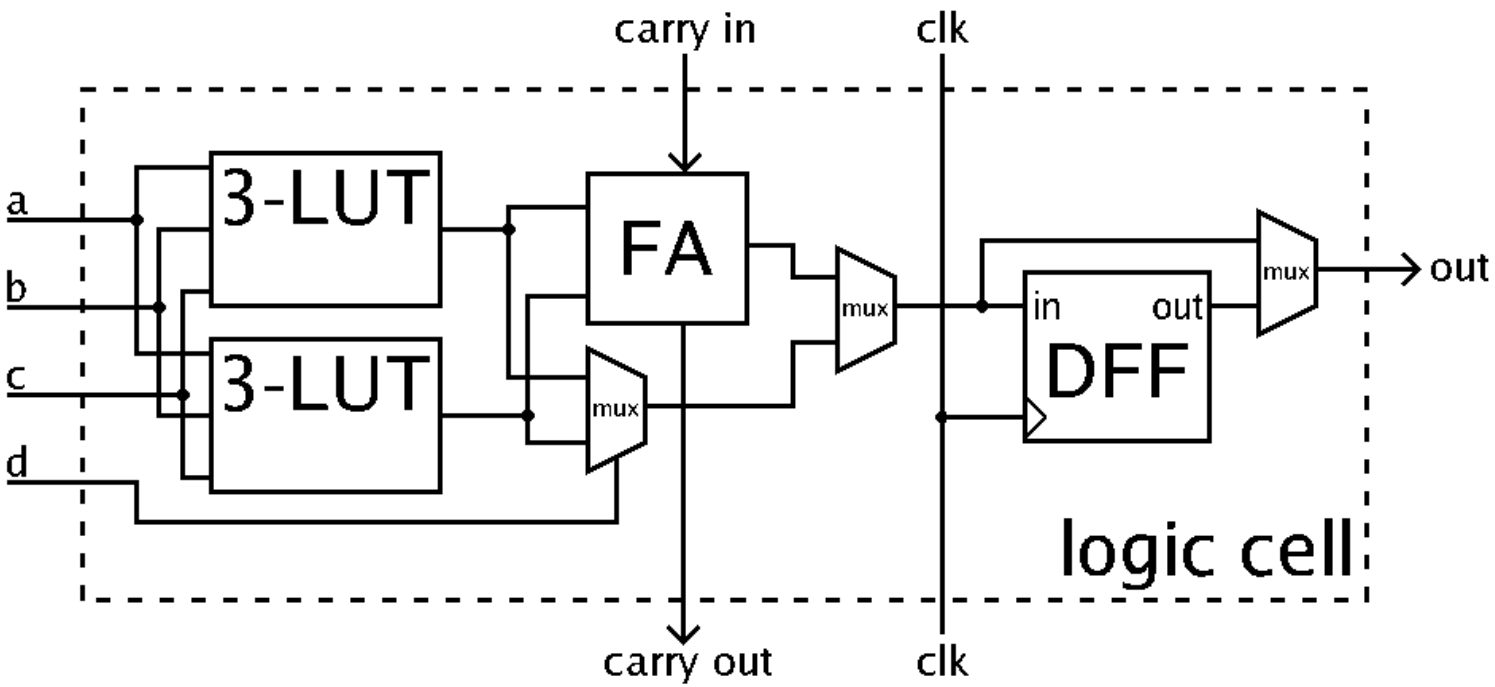
- Matriu de blocs lògics, interconnectats
- Recursos d'interconnexió
- Blocs d'entrada/sortida



Estructura

Blocs lògics

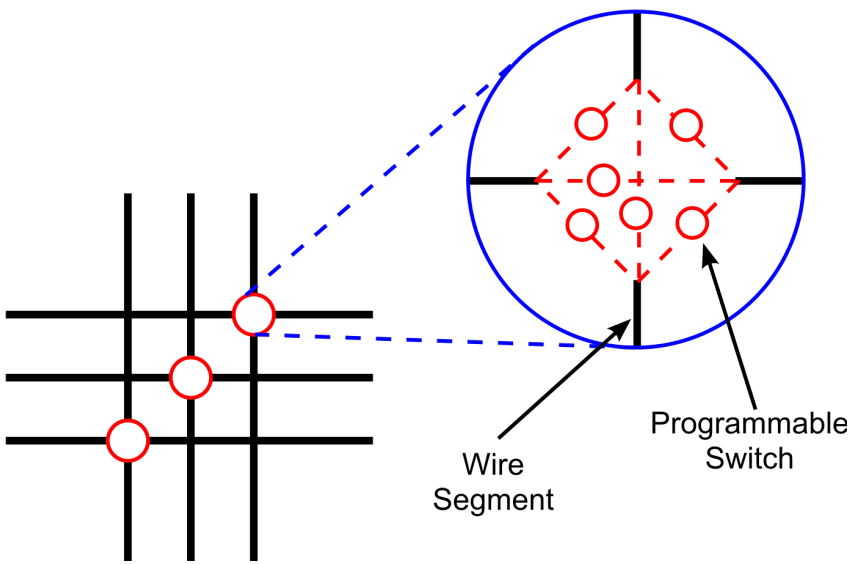
- Taules d'entrada (Look Up Table)
- Lògica operativa per càlculs
- Multiplexors
- Biestables síncrons



Estructura

Recursos d'interconnexió

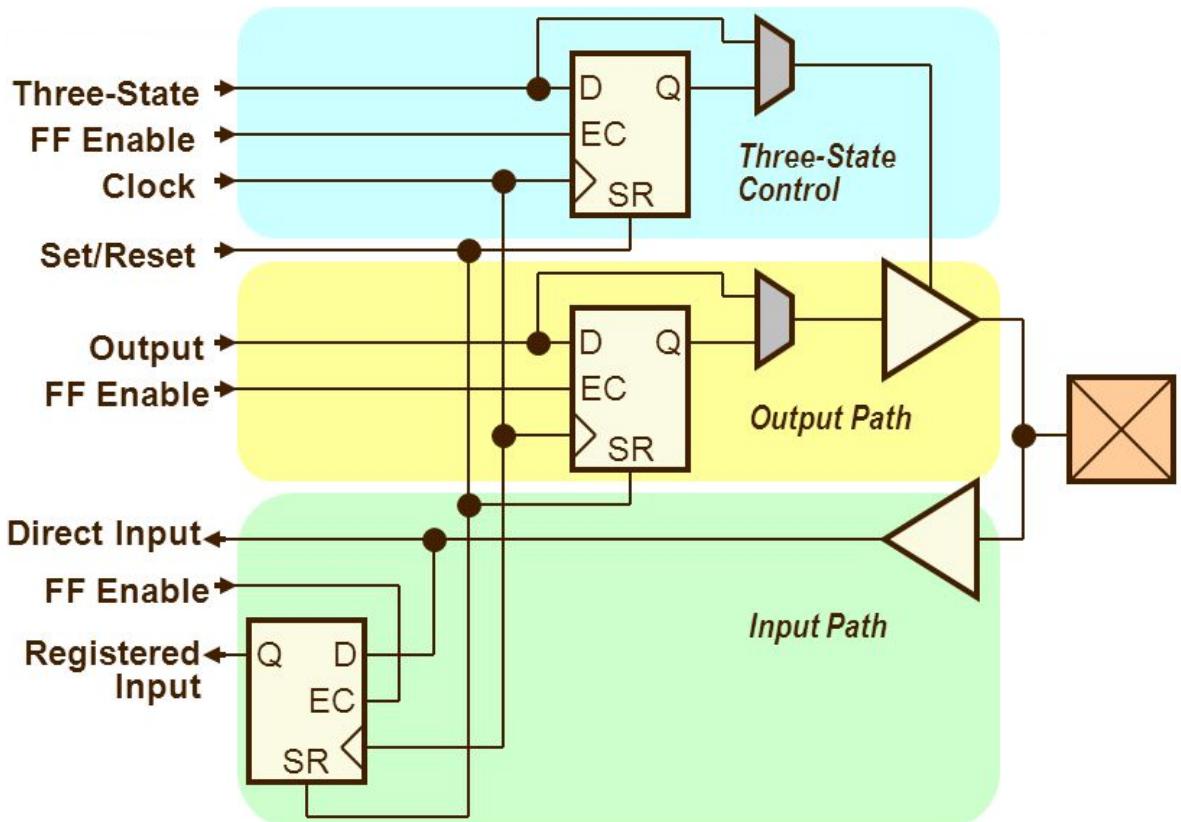
- Permeten encaminar les connexions entre blocs lògics i blocs d'entrada/sortida
- Es tria l'estat dels interruptors al programar la FPGA



Estructura

Blocs d'entrada/sortida

- Biestables síncrons
- Control tri-estat: entrada, sortida o alta impedància
- Connexió amb el món exterior



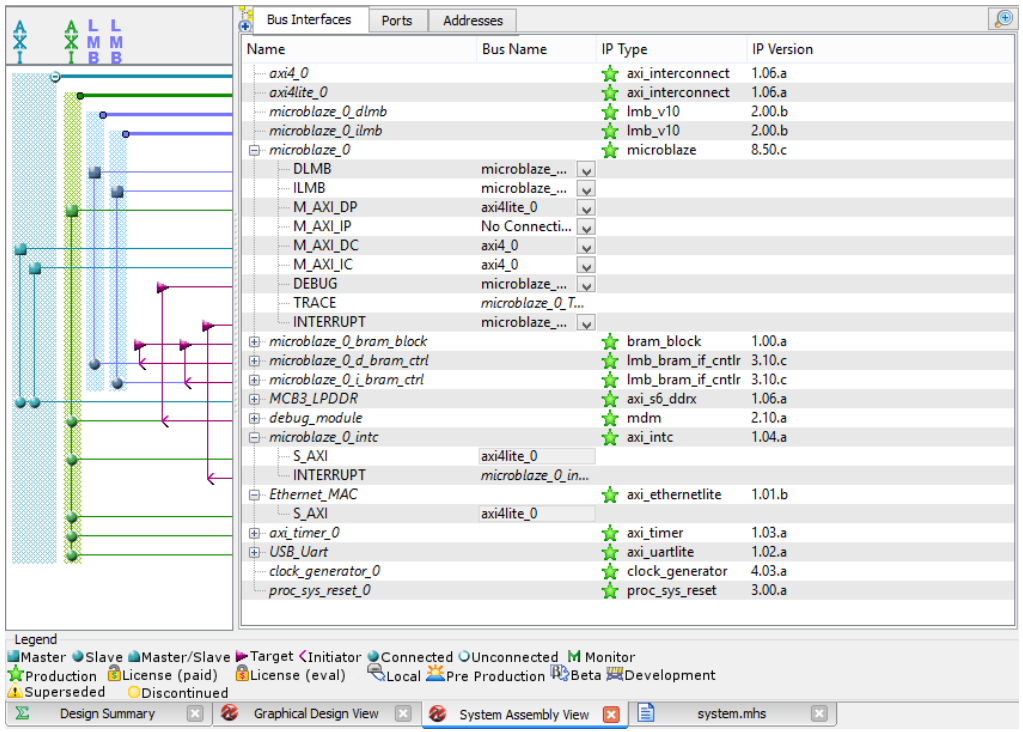
Nuclis de propietat intel·lectual

- Implementacions en llenguatge de descripció de hardware (principalment VHDL o Verilog)
- Generalment anomenats soft-cores
- Existeixen els hard-cores, incrustats sobre el silici de la FPGA
- Exemples variats: perifèric SPI, microcontrolador sencer (MicroBlaze de Xilinx), convertidor digital-analògic, ...

Nuclis de propietat intel·lectual

Disseny de hardware

- Cada fabricant ofereix les seves eines per generar hardware
- En el cas de Xilinx, la eina és el Xilinx Platform Studio

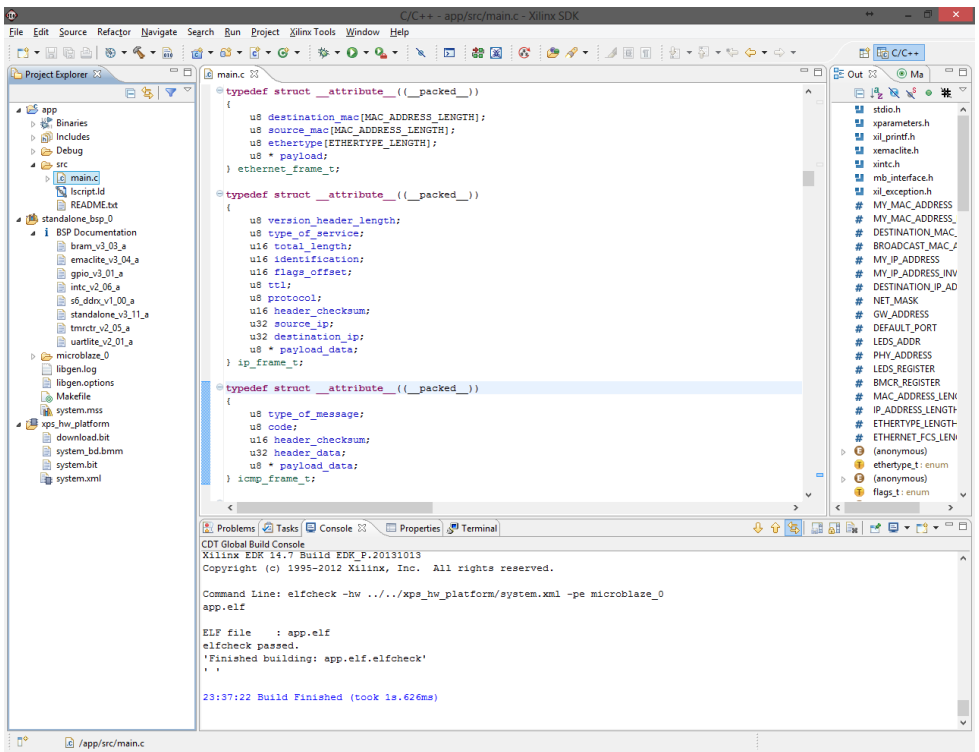


- Un cop creat el hardware, si hi ha un microcontrolador o microprocessador s'ha de programar

Nuclis de propietat intel·lectual

Programació de software

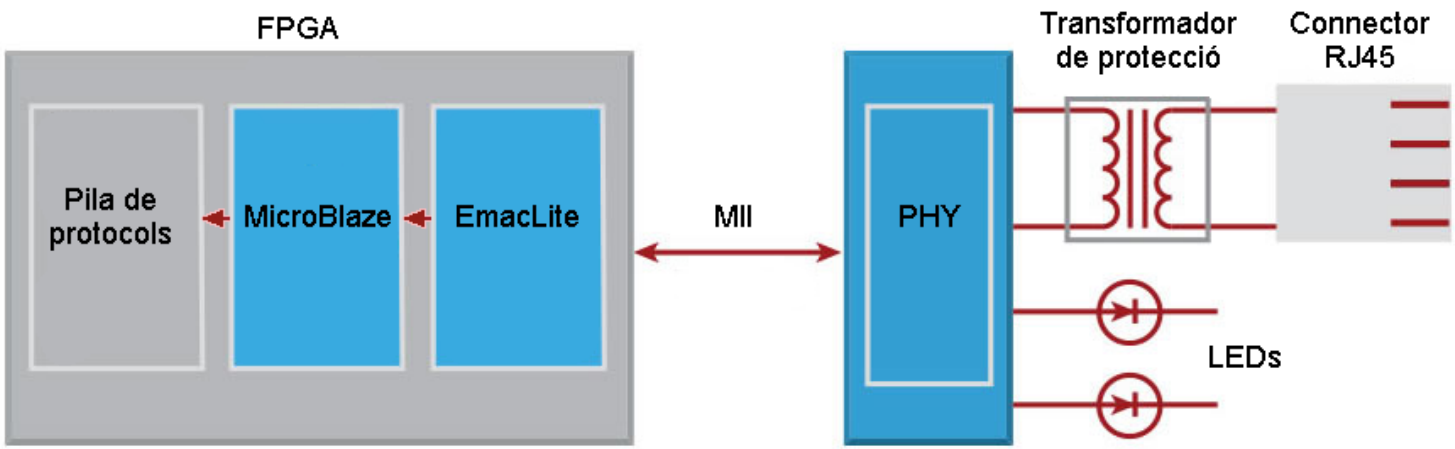
- En el cas de Xilinx, és el Xilinx Software Development Kit
- És un entorn de desenvolupament complet, inclou eines per programar i depurar, llibreries, ...
- Encara que admet llenguatge d'assemblador, la majoria es fa en C/C++



Ethernet

Funcionament a la placa

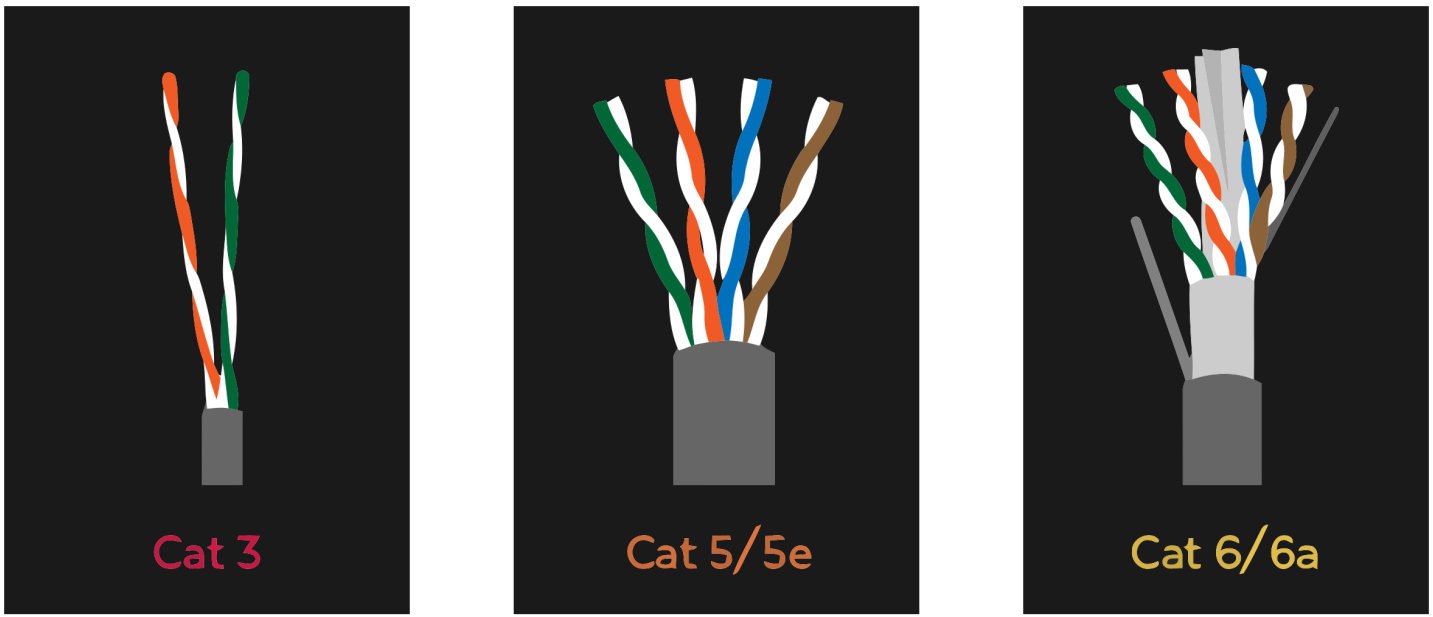
- En general, tots els dispositius que implementen Ethernet funcionen de forma similar



Ethernet

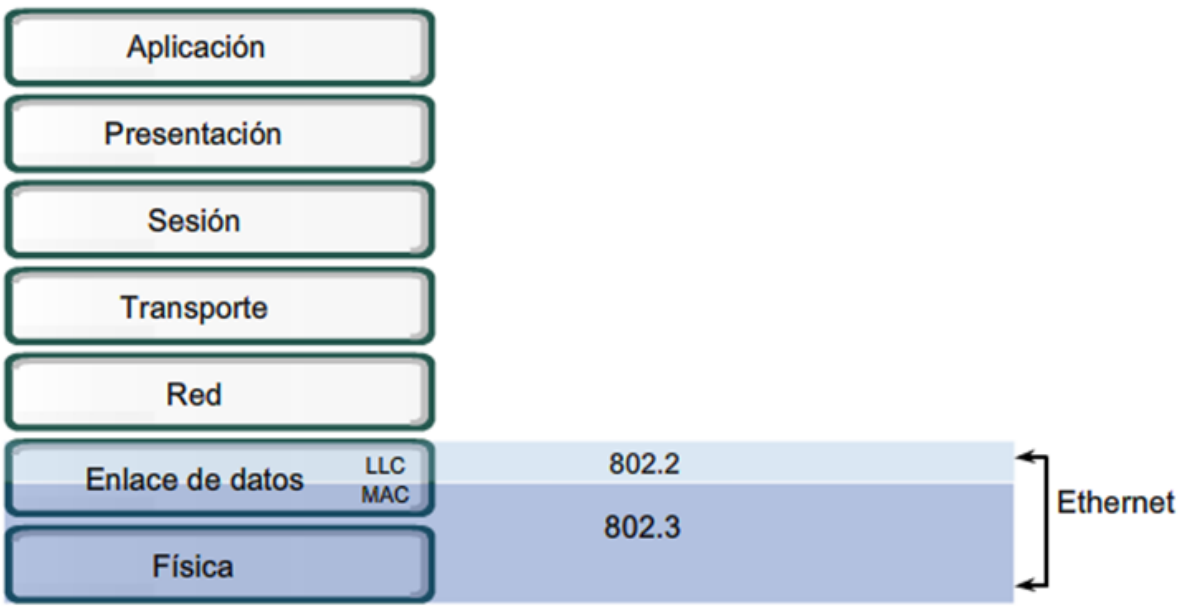
Cablejat

- Parells diferencials trenats. Generalment al menys un de recepció (RX±) i un de transmissió (TX±)
- Depenent de la velocitat de transmissió requerida, es necessita cablejat diferent



Pila de protocol per capes

- Generalment les comunicacions en xarxes es regeixen pel model OSI (Open Systems Interconnection) de ISO
- Aquest model organitza els protocols per capes: física, de xarxa, d'aplicació, ...
- Cada capa rep dades de la inferior i subministra dades a la superior

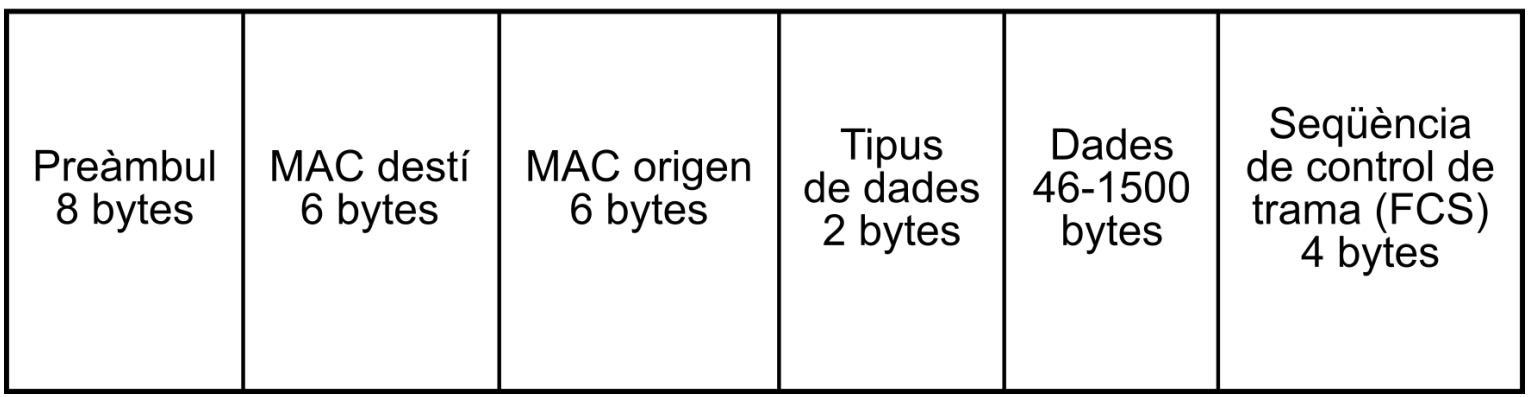


Protocols

Ethernet

- Aquest és un protocol senzill, on s'han d'enviar poques dades a més de les que es volen transportar (26 bytes + dades)
- El preàmbul no és informació útil com a tal, serveix per sincronitzar

Trama Ethernet



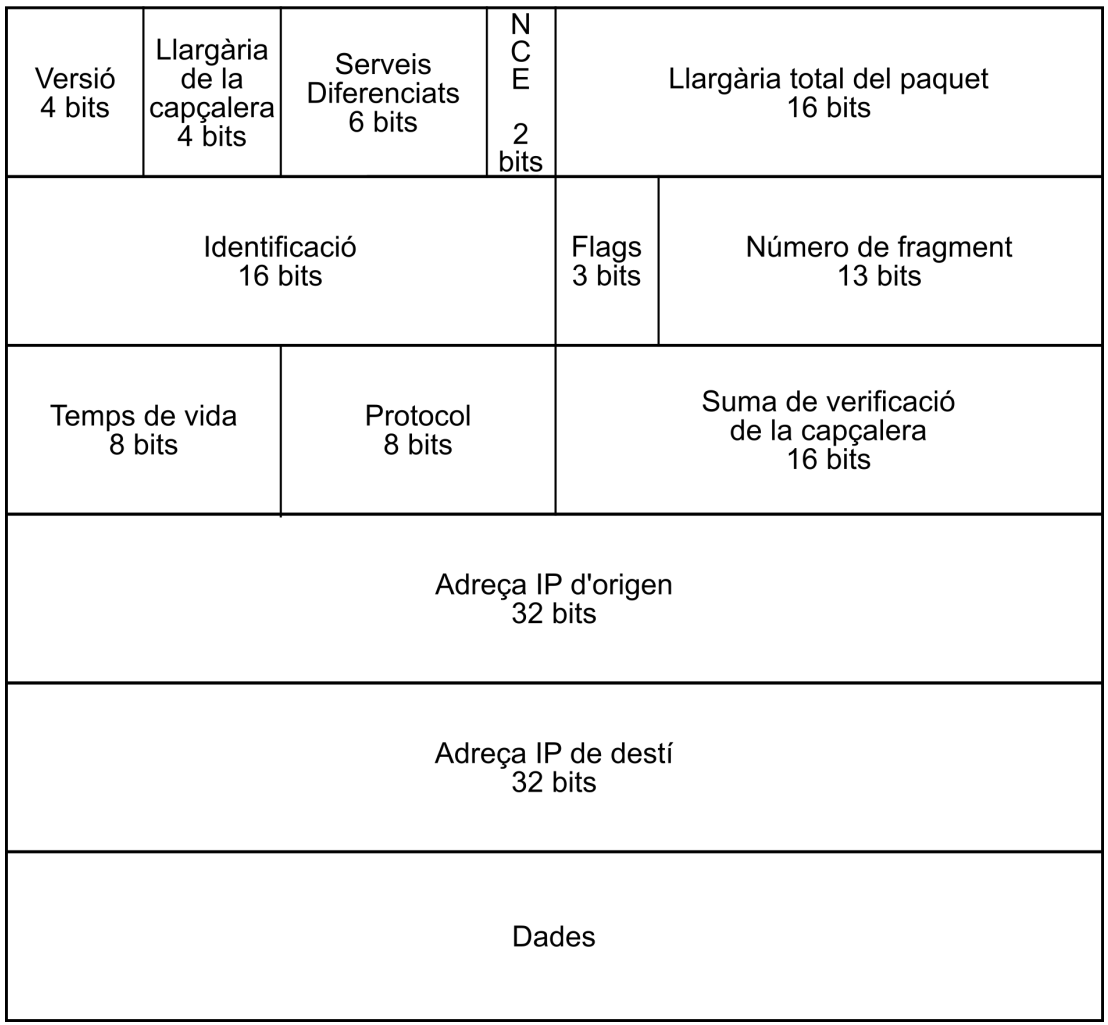
Protocols

IPv4

- Internet Protocol version 4
- Protocol bàsic per dirigir-se als equips d'una xarxa, sigui local o global
- Imprescindible a la xarxa de xarxes
- Relativament complicat perquè ha de contemplar gran varietat de casos (fragmentació, diferents subprotocols, . . .)

Protocols

IPv4



Protocols

ARP

- Address Resolution Protocol
- Protocol bàsic per identificar els equips dins d'una xarxa local
- Complement ideal entre IPv4 i Ethernet per xarxes locals

Protocols

ARP

<table>
<tr><td colspan="2">Tipus de medi
16 bits</td></tr>
<tr><td colspan="2">Identificació
16 bits</td></tr>
<tr><td>Llargària
adreça
física
8 bits</td><td>Llargària
adreça
lògica
8 bits</td></tr>
<tr><td colspan="2">Operació que es realitza
16 bits</td></tr>
<tr><td colspan="2">Adreça física d'origen
48 bits (6 bytes)</td></tr>
<tr><td colspan="2">Adreça lògica d'origen
32 bits (4 bytes)</td></tr>
<tr><td colspan="2">Adreça física de destí
48 bits (6 bytes)</td></tr>
<tr><td colspan="2">Adreça lògica de destí
32 bits (4 bytes)</td></tr>
</table>

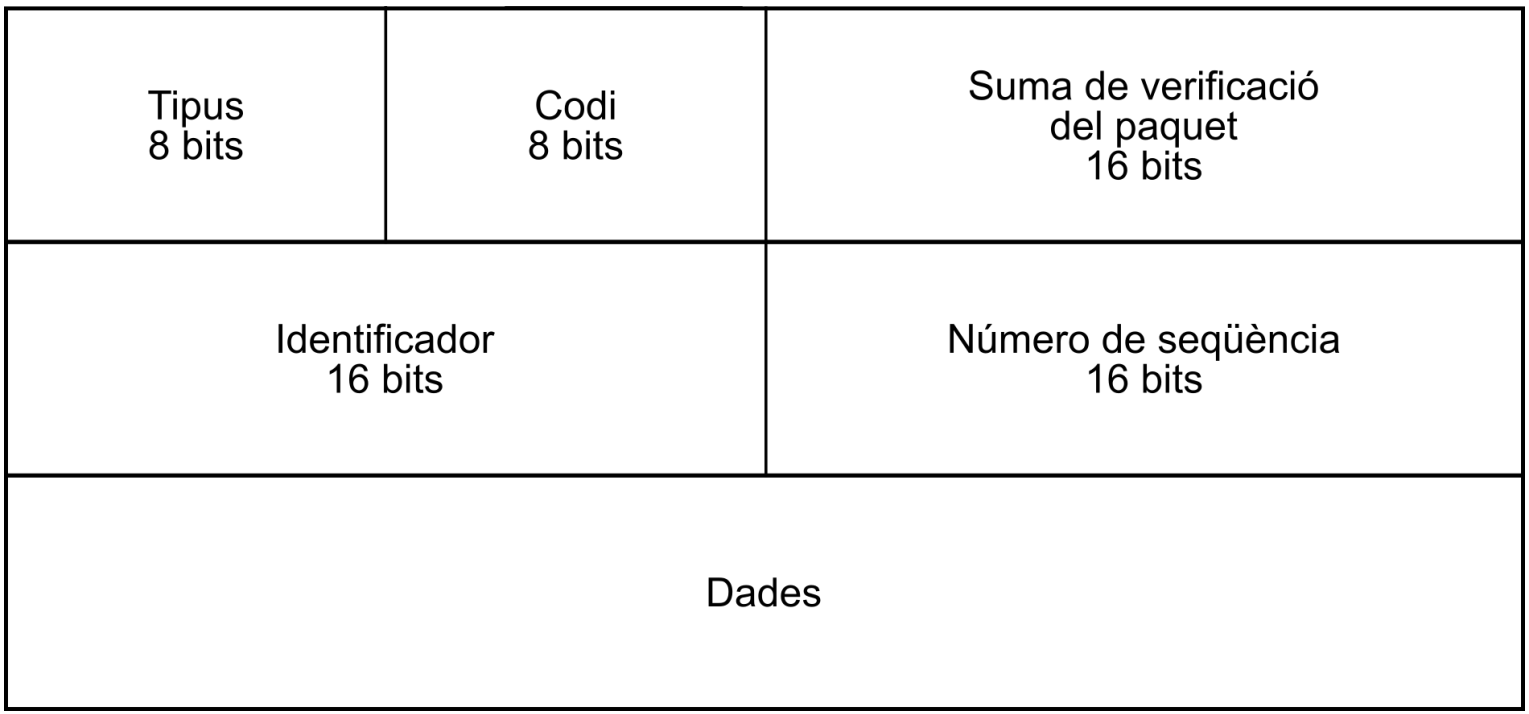
Protocols

ICMP

- Internet Control Message Protocol
- Protocol que serveix per regular el tràfic, generalment, a la xarxa de xarxes
- Serveix per notificar de pèrdua de paquets, comprovar l'estat de la xarxa
- Es pot utilitzar en xarxes locals per provar la capacitat de la xarxa

Protocols

ICMP



Implementació pràctica

LwIP

- LwIP és una de les implementacions de TCP/IP més utilitzades
- Pila específicament dissenyada per aplicacions incrustades
- Microcontroladors amb poca memòria de codi (ocupa al voltant de 40 kilobytes per si sola)
- No necessita de molta memòria RAM, sempre en funció de les demandes de la xarxa i l'aplicació
- Pila de protocols molt capaç, suporta la majoria de protocols utilitzats a Internet

Implementació pràctica

LwIP

```
int main(void) {
 /* neteja la consola UART i imprimeix missatge */
 xil_printf("%c[2J",27);
 xil_printf("------ TFM - Marko Peshevski - versio LwIP ------\r\n");
 inicialitza_temporitzador();
 inicialitza_interrupcions();
 inicialitza_lwip();
 xil_printf("Arrenca aplicacio que respon a eco... ");
 arrenca_app();
 xil_printf("Aplicacio en marxa\r\n");
 /* activa les interrupcions a nivell de processador */
 Xil_ExceptionEnable();
 while (1) {
  /* rep dades de la interficie de xarxa (driver ethernet MAC) */
  xemacif_input(&interficie_xarxa);
  /* consulta temporitzadors i executa tasques periodiques */
  if (temporitzador_tcp_rapid) {
   tcp_fasttmr();
   temporitzador_tcp_rapid = 0;
  }
  if (temporitzador_tcp_lent) {
   tcp_slowtmr();
   temporitzador_tcp_lent = 0;
  }
 }
 return 0;
}
```

Implementació pràctica

Sense LwIP

- Implementació feta per l'autor
- Pila de protocols molt reduïda: ARP, i ICMP sobre IPv4 (sense fragmentació)
- Intent d'optimitzar el màxim possible tant la memòria ocupada pel codi com la velocitat d'execució
- Bàsicament capaç de respondre a peticions d'eco i poc més

Implementació pràctica

Sense LwIP

```
int main(void) {
 /* neteja la consola UART i imprimeix missatge */
 xil_printf("%c[2J",27);
 xil_printf("------ TFM - Marko Peshevski - versio NO-LwIP ------\r\n");
 inicialitza_interrupcions();
 inicialitza_emaclite();
 /* activa les interrupcions a nivell de processador */
 Xil_ExceptionEnable();
 while (1) {
  if (sys.paquet_rebut) {
   sys.paquet_rebut = FALSE; /* neteja el flag */
   /* inverteix les direccions MAC, respon d'on ha vingut el paquet */
   memcpy(trama_ethernet->mac_desti, trama_ethernet->mac_origen, LLARGARIA_MAC);
   memcpy(trama_ethernet->mac_origen, direccio_mac, LLARGARIA_MAC);
   /* mira si el paquet es ARP. necessita girar els bytes */
   if(INVERTEIX_BYTES_16(trama_ethernet->ethertype) == ARP) {
    /* codi a la seguent diapositiva */
   }
   /* mira si es un paquet IPv4. necessita girar els bytes */
   else if(INVERTEIX_BYTES_16(trama_ethernet->ethertype) == IPv4) {
    /* codi a la seguent diapositiva */
   }
  }
  if(sys.paquet_enviat){
   sys.paquet_enviat = FALSE; /* neteja el flag */
  }
 }
 return 0;
}
```

Implementació pràctica

Sense LwIP

```
/* simplement es un cast que interpreta les dades de memoria per facilitar */
paquet_arp_t * paquet_arp = (paquet_arp_t *) &trama_ethernet->dades;
/* mira si el paquet anava dirigit per la nostra IP */
if(paquet_arp->ip_desti == direccio_ip) {
 if(paquet_arp->operacio == ARP_REQUEST_LINUX) {
  paquet_arp->operacio = ARP_REPLY_LINUX;
 }
 else if(paquet_arp->operacio == ARP_REQUEST_WINDOWS) {
  paquet_arp->operacio = ARP_REPLY_WINDOWS;
 }
 /* inverteix adreces IP i MAC del paquet */
 paquet_arp->ip_desti = paquet_arp->ip_origen;
 paquet_arp->ip_origen = direccio_ip;
 memcpy(paquet_arp->mac_desti, paquet_arp->mac_origen, LLARGARIA_MAC);
 memcpy(paquet_arp->mac_origen, direccio_mac, LLARGARIA_MAC);
 /* envia la resposta */
 XEmacLite_Send(&emaclite, buffer, sys.llargaria_paquet_rebut - LLARGARIA_FCS);
}
```

Implementació pràctica

Sense LwIP

```
/* simplement es un cast que interpreta les dades de memoria per facilitar */
paquet_ip_t * paquet_ip = (paquet_ip_t *) &trama_ethernet->dades;
/* mira si es un paquet de tipus ICMP */
if(paquet_ip->protocol == ICMP) {
 /* simplement es un cast que interpreta les dades de memoria per facilitar */
 paquet_icmp_t * paquet_icmp = (paquet_icmp_t *) &paquet_ip->dades;
 /* mira si es una peticio d'eco */
 if(paquet_icmp->tipus_de_missatge == ECHO_REQUEST) {
  /* canvia la suma de verificacio */
  paquet_icmp->suma_verificacio += ECHO_REQUEST; /* -ECHO_REPLY */
  /* modifica el tipus de missatge */
  paquet_icmp->tipus_de_missatge = ECHO_REPLY;
  /* inverteix les adreces del paquet IPv4 */
  paquet_ip->ip_desti = paquet_ip->ip_origen;
  paquet_ip->ip_origen = direccio_ip;
  /* envia la resposta */
  XEmacLite_Send(&emaclite, buffer, sys.llargaria_paquet_rebut - LLARGARIA_FCS);
 }
}
```

Resultats experimentals

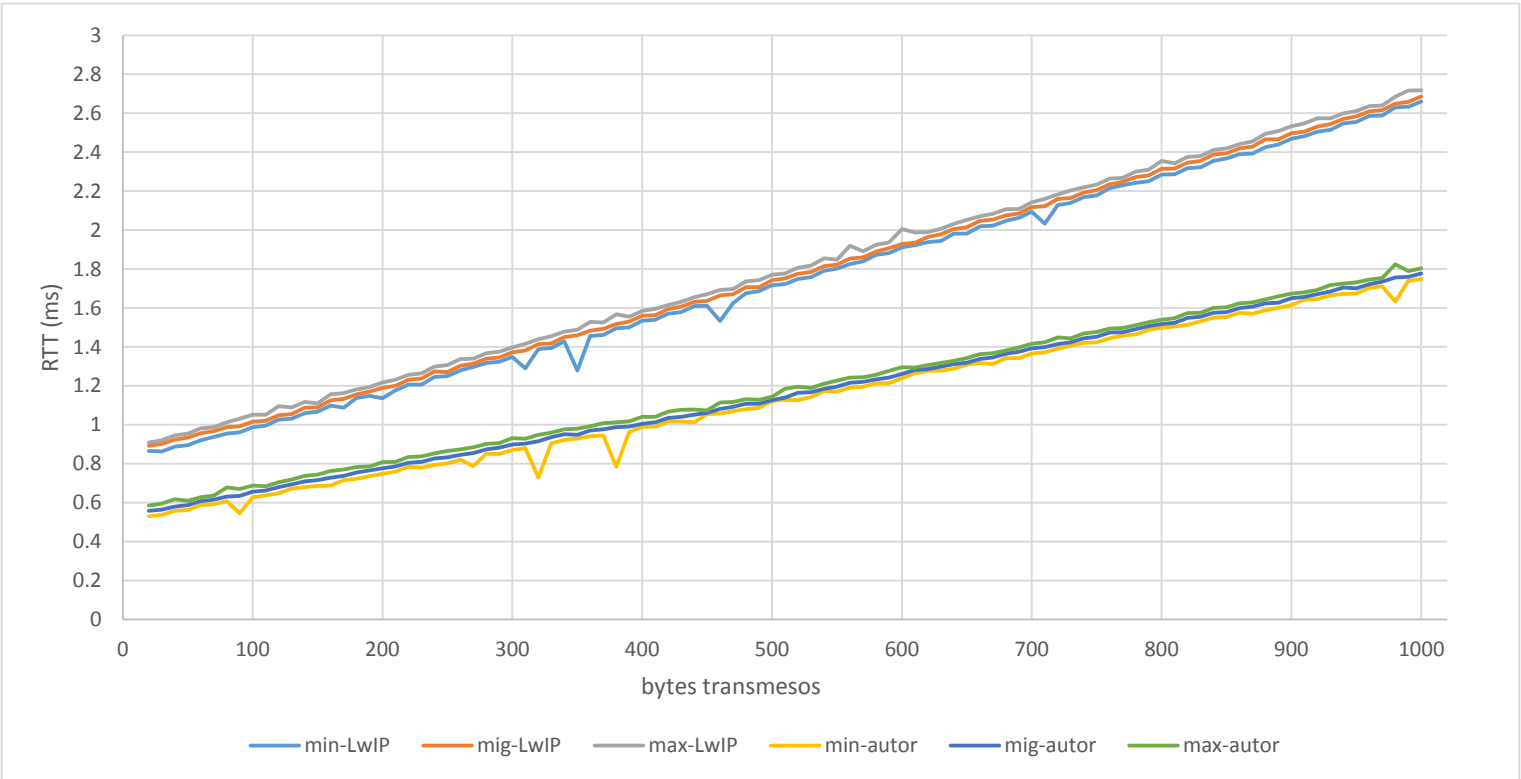
Mètode d'estudi

- S'executen gran nombre de peticions d'eco
- Llargària de les dades del paquet ICMP variable
- Automatitzat amb script de bash

```
#!/bin/bash
for i in `seq 20 10 1000`;
do
 echo $i
 sudo ping 192.168.1.200 -c 100 -s $i -q -i 0.01 -w 0.01 >> "arxiu.txt"
 sleep 0.5
done
```

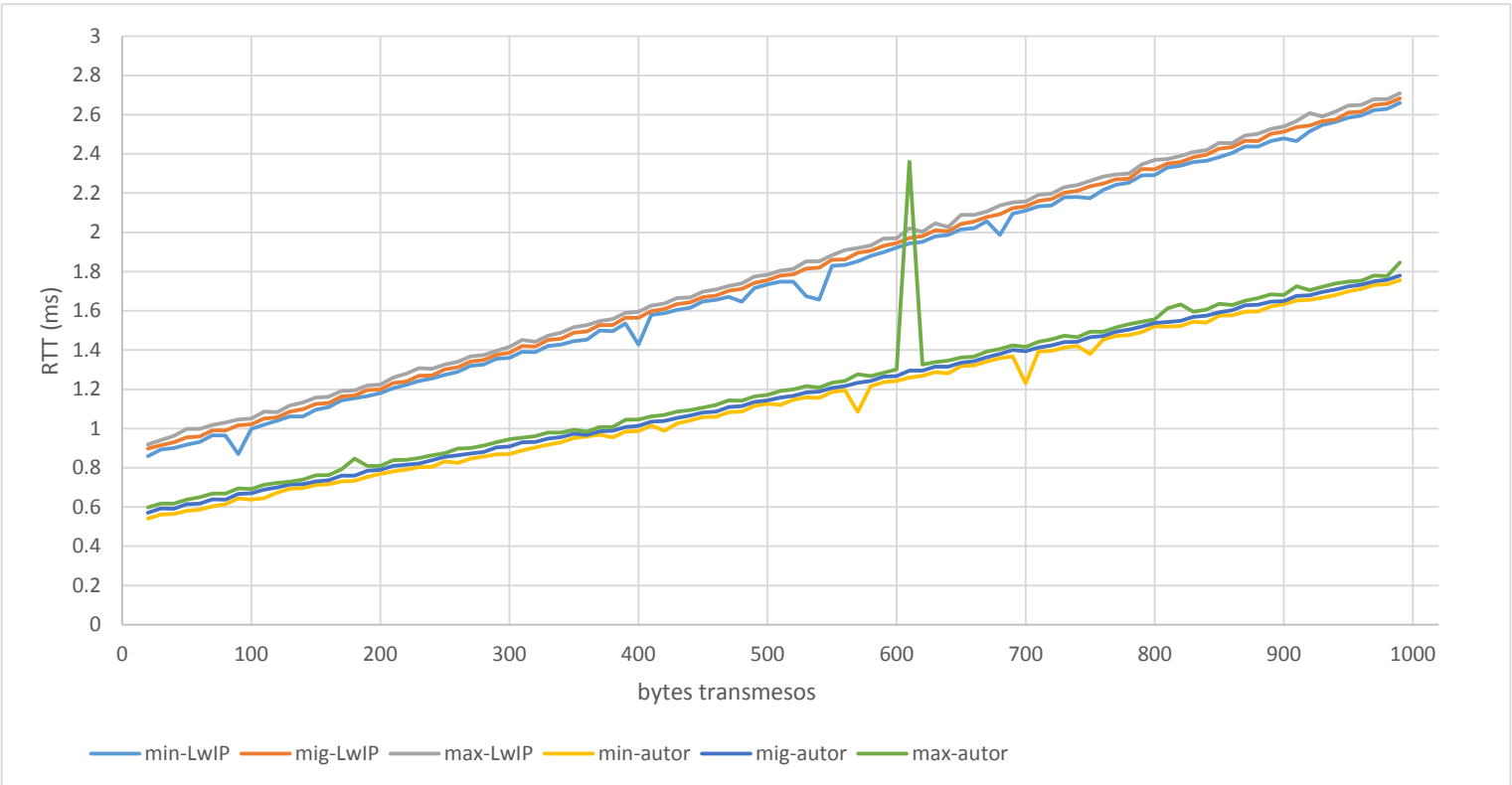
Resultats experimentals

Implementacions sobre cablejats diferents - Cat3



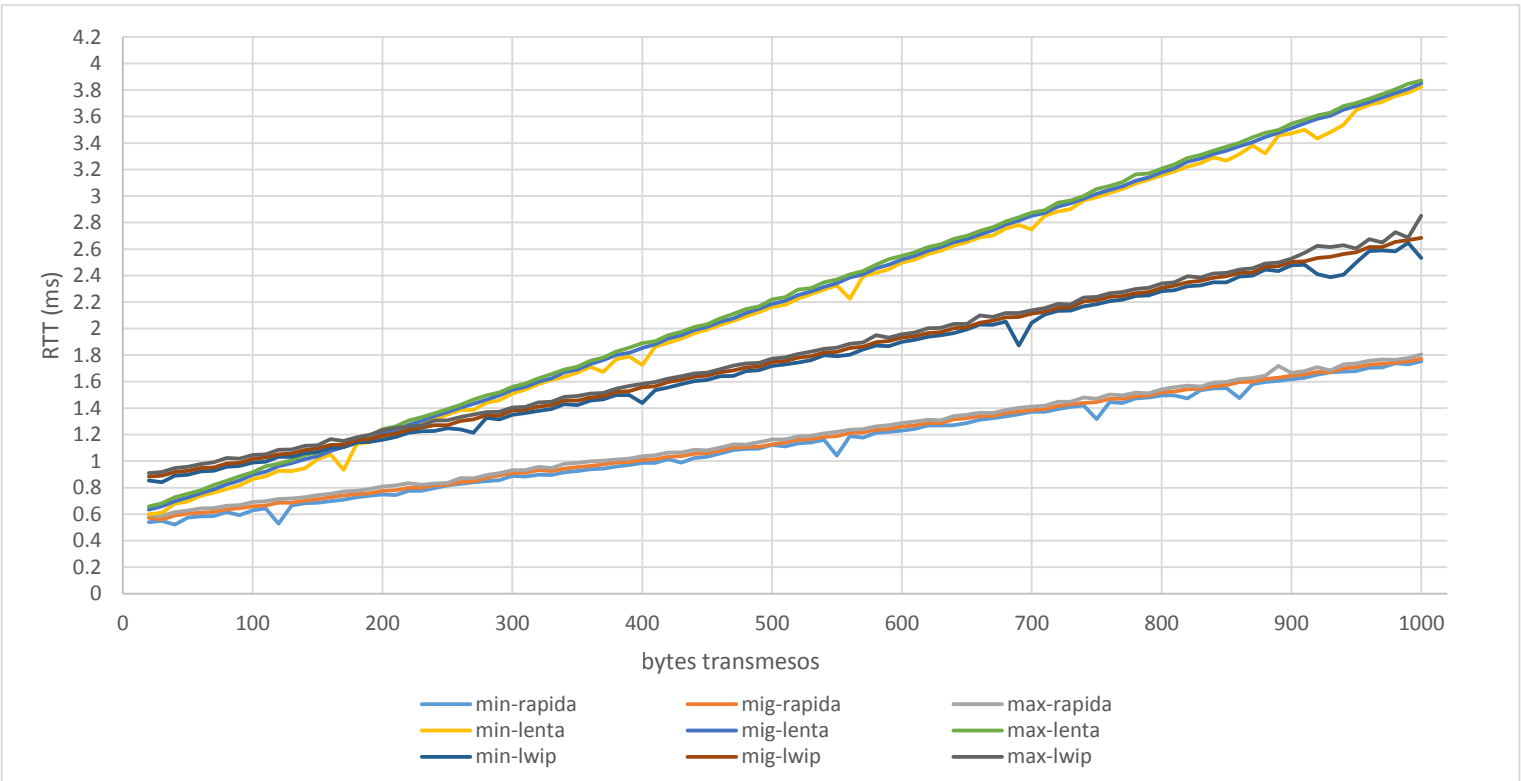
Resultats experimentals

Implementacions sobre cablejats diferents - Cat6



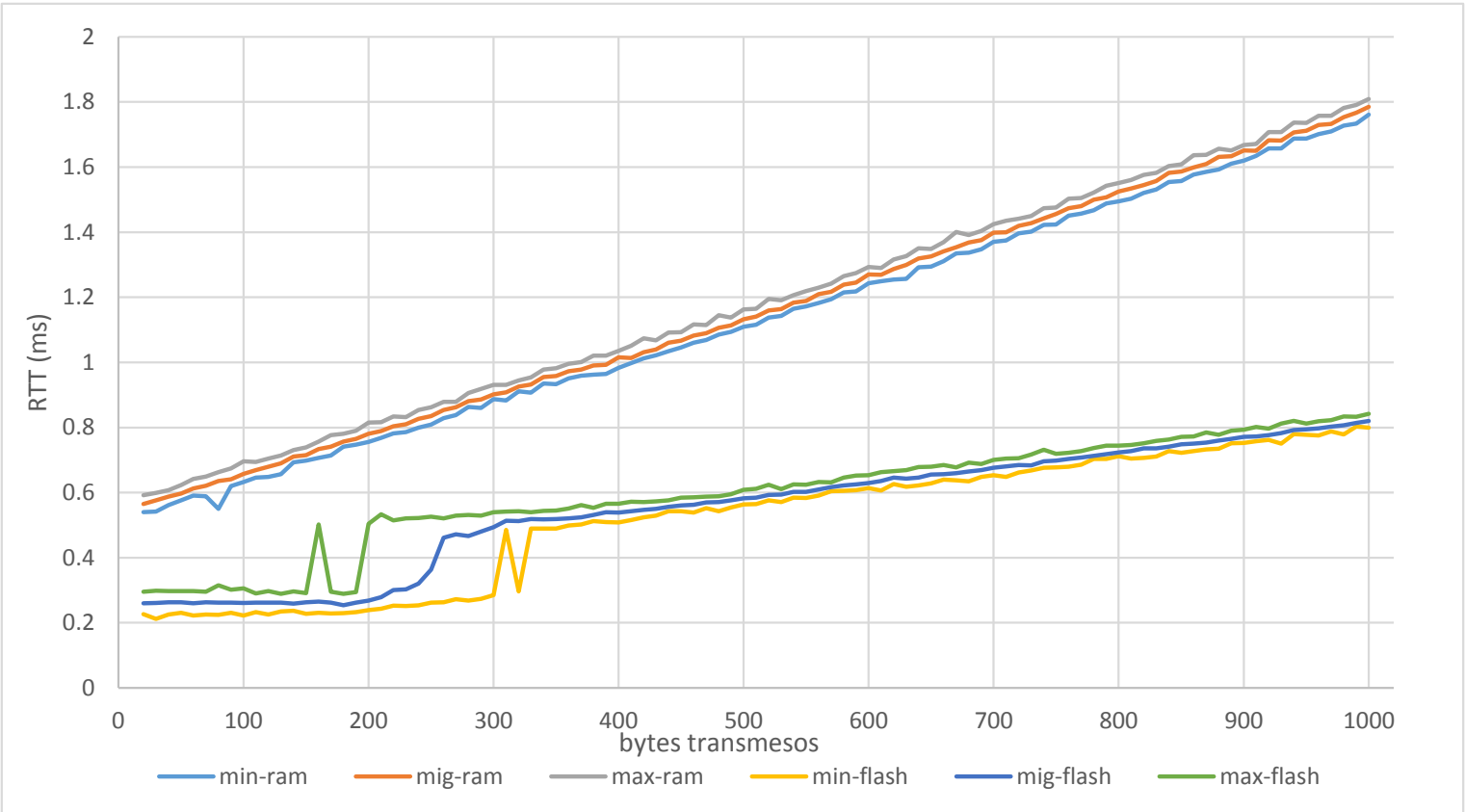
Resultats experimentals

Sumes de verificació



Resultats experimentals

Memòria d'execució



Conclusions

Més informació

Més informació i codi font a:
`https://github.com/markopesevski/TFM`

Gràcies per la vostra atenció

Dubtes? Comentaris? Preguntes?