Universitat de les Illes Balears

Presentación de Trabajo Final de Grado (TFG) para el
Grado de Ingeniería Electrónica Industrial y Automática

Palma, Octubre 2017

Implementation and Validation of the Fundamental Mechanisms of the Flexible Time-Triggered Communication Paradigm for Ethernet-based Highly-Reliable Systems

Sergi Arguimbau Guarinos

Tutor: Alberto Ballesteros Varela

Índice

- Introducción
- FTT (Flexible Time Triggered)
- Trabajo desarrollado
- Experimentos
- Conclusiones y trabajo futuro

Índice

- **Introducción**
- FTT (Flexible Time Triggered)
- Trabajo desarrollado
- Experimentos
- Conclusiones y trabajo futuro

Introducción

Ethernet

Ventajas:

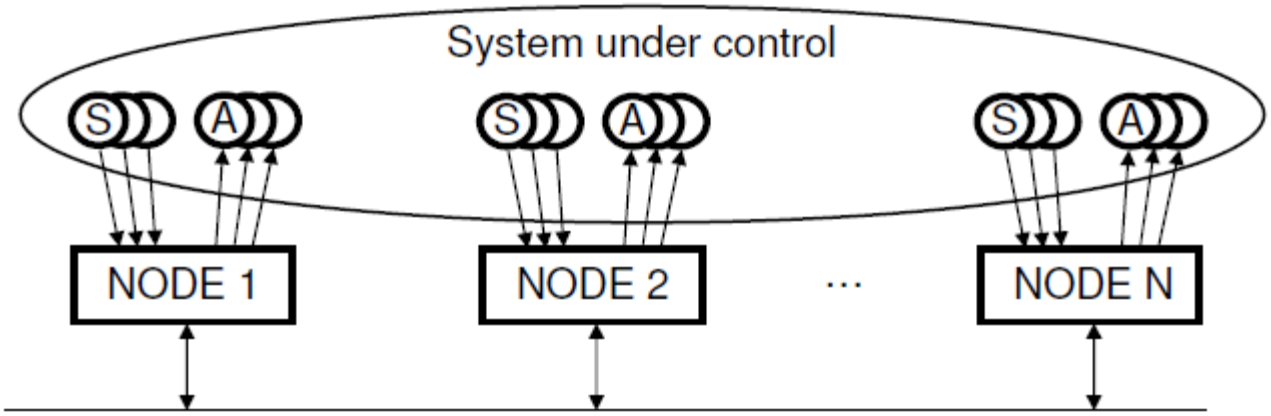
- ✓ **Alto ancho de banda: velocidad ↑↑**
- ✓ **Componentes baratos: precio ↓↓**
- ✓ **Uso muy extendido: compatibilidad ↑↑**

Introducción

Ethernet

Es interesante su uso en **sistemas industriales**

Ejemplo: Sistemas de Control Distribuidos



Introducción

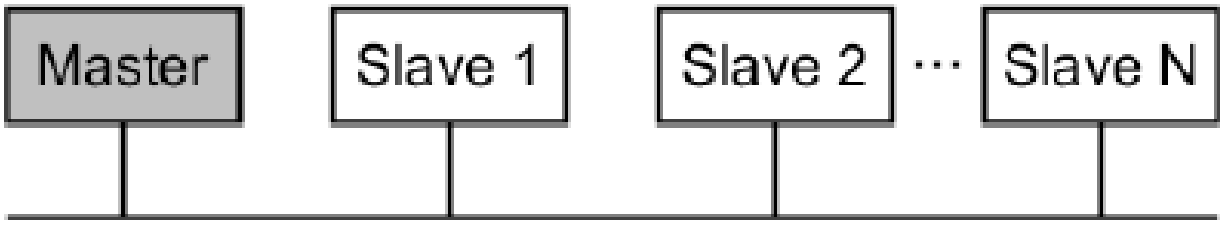
Sistemas Industriales

- Requisitos
  - Tiempo real: cumpliendo un deadline
  - Garantía de funcionamiento
    - Fiabilidad
  - Flexibilidad: necesaria en sistemas dinámicos

Introducción

Sistemas Industriales + Ethernet

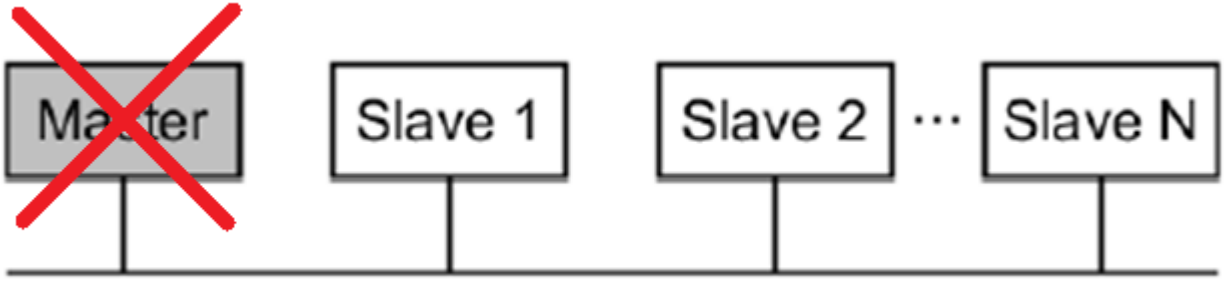
- **Ethernet** no cumple estos **requisitos**
- Una **solución** es utilizar el paradigma de comunicaciones **Flexible Time Triggered (FTT)**
  - ✓Tiempo real
  - ✓Flexibilidad
  - ❑Fiabilidad → falta garantía de funcionamiento



Introducción

Sistemas Industriales + Ethernet

- **Ethernet** no cumple estos **requisitos**
- Una **solución** es utilizar el paradigma de comunicaciones **Flexible Time Triggered (FTT)**
  - ✓Tiempo real
  - ✓Flexibilidad
  - ❑Fiabilidad → falta garantía de funcionamiento

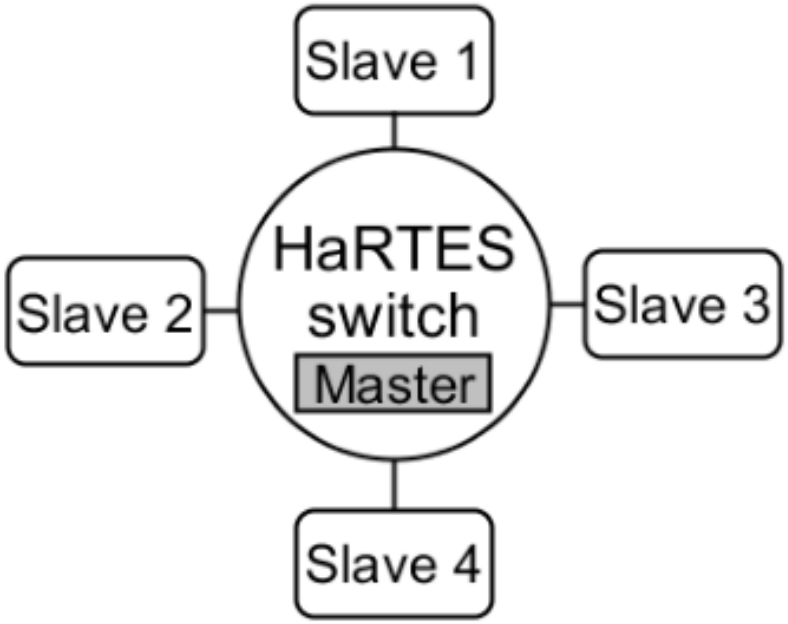


Introducción

Sistemas Industriales + Ethernet

Se propone el **proyecto DFT4FTT** en la UIB

- ✓Fiabilidad a través de **tolerancia a fallos** (FT)

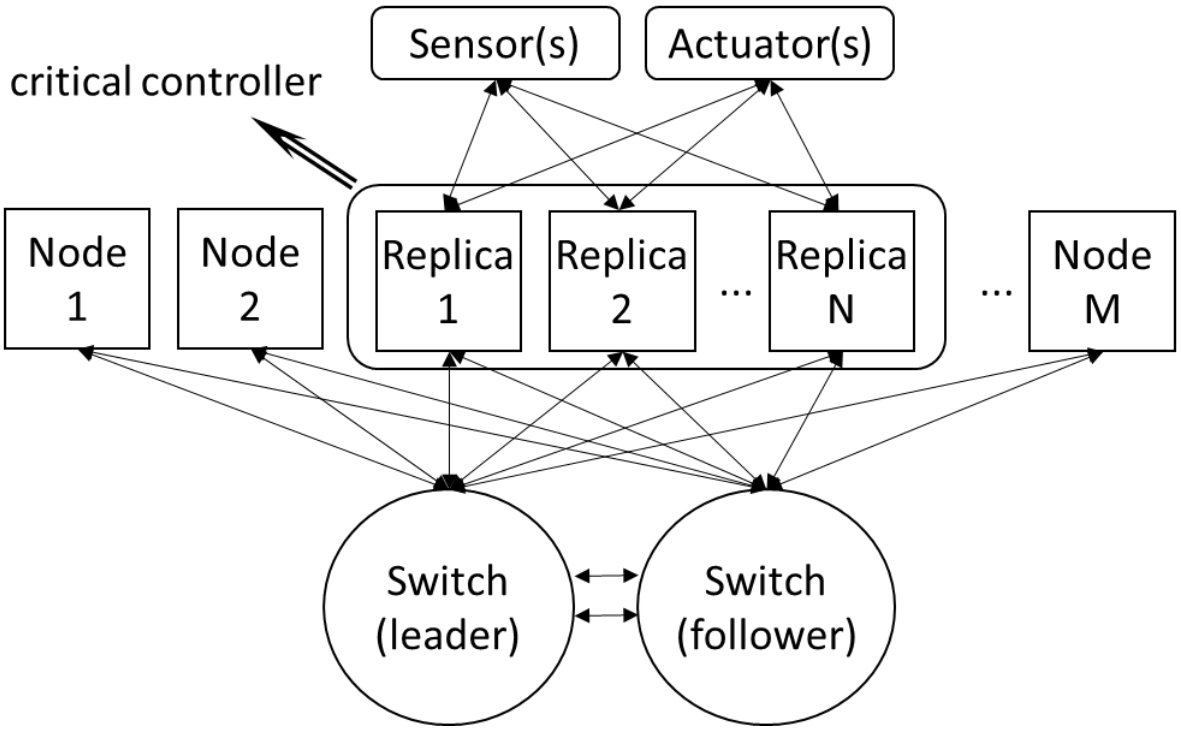


Introducción

Sistemas Industriales + Ethernet

Se propone el **proyecto DFT4FTT** en la UIB

- ✓Fiabilidad a través de **tolerancia a fallos** (FT)



Introducción

Sistemas Industriales + Ethernet

Implementar un **prototipo DFT4FTT**
a partir de **FTT** tiene varios **problemas**

- FTT no fue diseñado tendiendo en cuenta la FT
- Los mecanismos de FT no son ortogonales
- Contiene mecanismos prescindibles

Objetivo

Reimplementar FTT sobre Ethernet

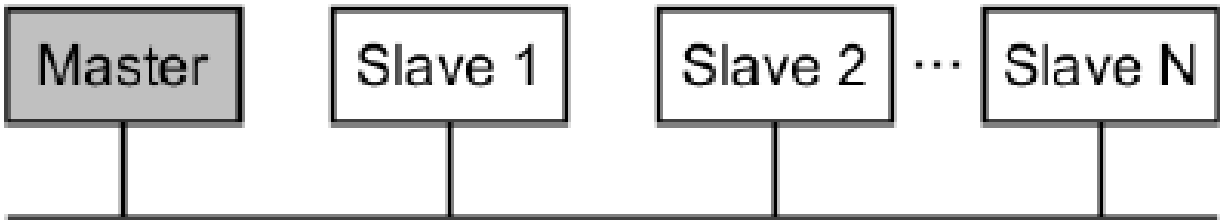
- Pensando en el **futuro de DFT4FTT**
  - Partiendo de un **nuevo diseño**
  - **Fácilmente extensible** con mecanismos de tolerancia a fallos
- Implementar la lógica a alto nivel
  - Master
  - Esclavos

Índice

- Introducción
- **FTT (Flexible Time Triggered)**
- Trabajo desarrollado
- Experimentos
- Conclusiones y trabajo futuro

FTT (Flexible Time Triggered)

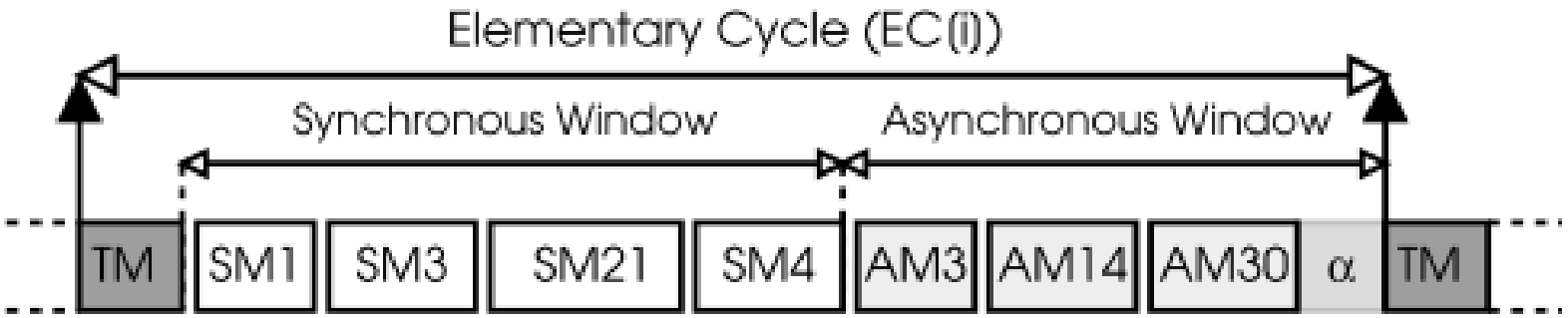
- Arquitectura master/multi-esclavo



- **Master** controla las comunicaciones
- **Esclavos** siguen las instrucciones del master para enviar los mensajes de datos

FTT (Flexible Time Triggered)

- Elementary Cycle (EC)

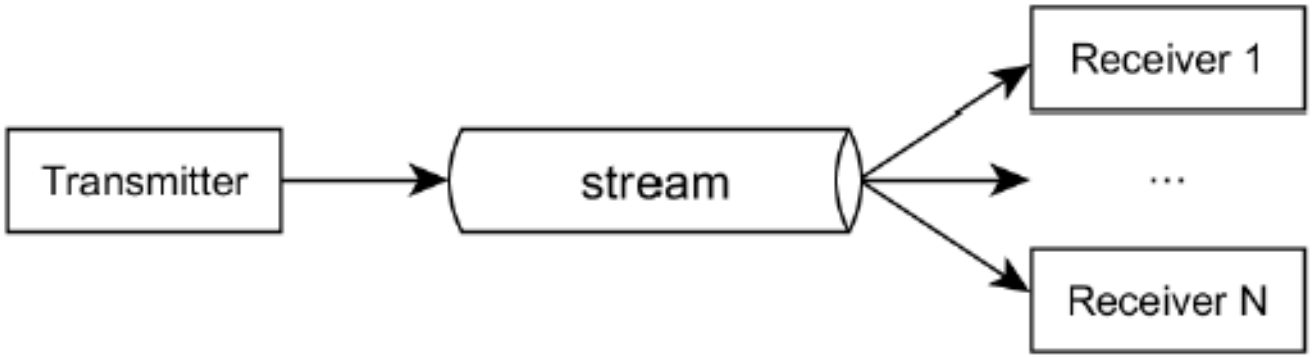


- TM → Master
- SW
- AW

Slaves

FTT (Flexible Time Triggered)

- Streams: canales virtuales de comunicación



- Transmisor = Publisher (1)
- Receptores = Subscribers (N)

FTT (Flexible Time Triggered)

- Cada stream tiene ciertos parámetros:
  - Stream ID
  - Period [ECs]
  - Offset [ECs]
  - Publisher [node ID]
  - Subscribers [node ID]
  - Data size [bytes]

- **Stream Database:** contiene el listado mensajes que se van a enviar en los EC

FTT (Flexible Time Triggered)

- Número de secuencia en cada EC

EC 0	EC 1	EC 2	EC 3	EC 4	EC 5	EC 6	EC 7	…

- Condición del master scheduler:

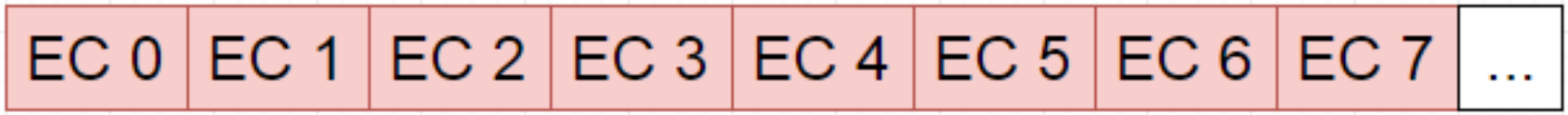
$$num\_EC \bmod T_i - O_i == 0$$

T_i: period of stream i

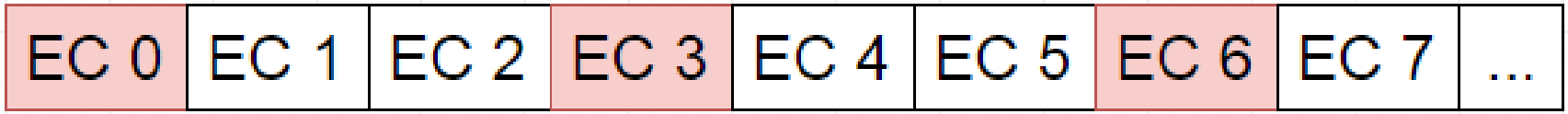
O_i: offset of stream i

FTT (Flexible Time Triggered)

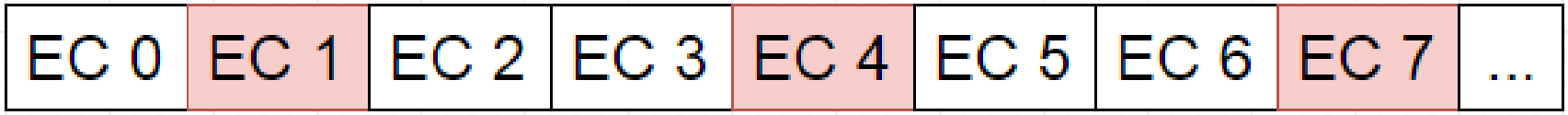
- Ejemplo periodicidad de un mensaje síncrono:
  - Period = 1 EC



  - Period = 3 ECs, Offset = 0 ECs



  - Period = 3 ECs, Offset = 1 ECs



Índice

- Introducción
- FTT (Flexible Time Triggered)
- **Trabajo desarrollado**
- Experimentos
- Conclusiones y trabajo futuro

Trabajo Desarrollado - Intro

Tecnología/herramientas utilizadas

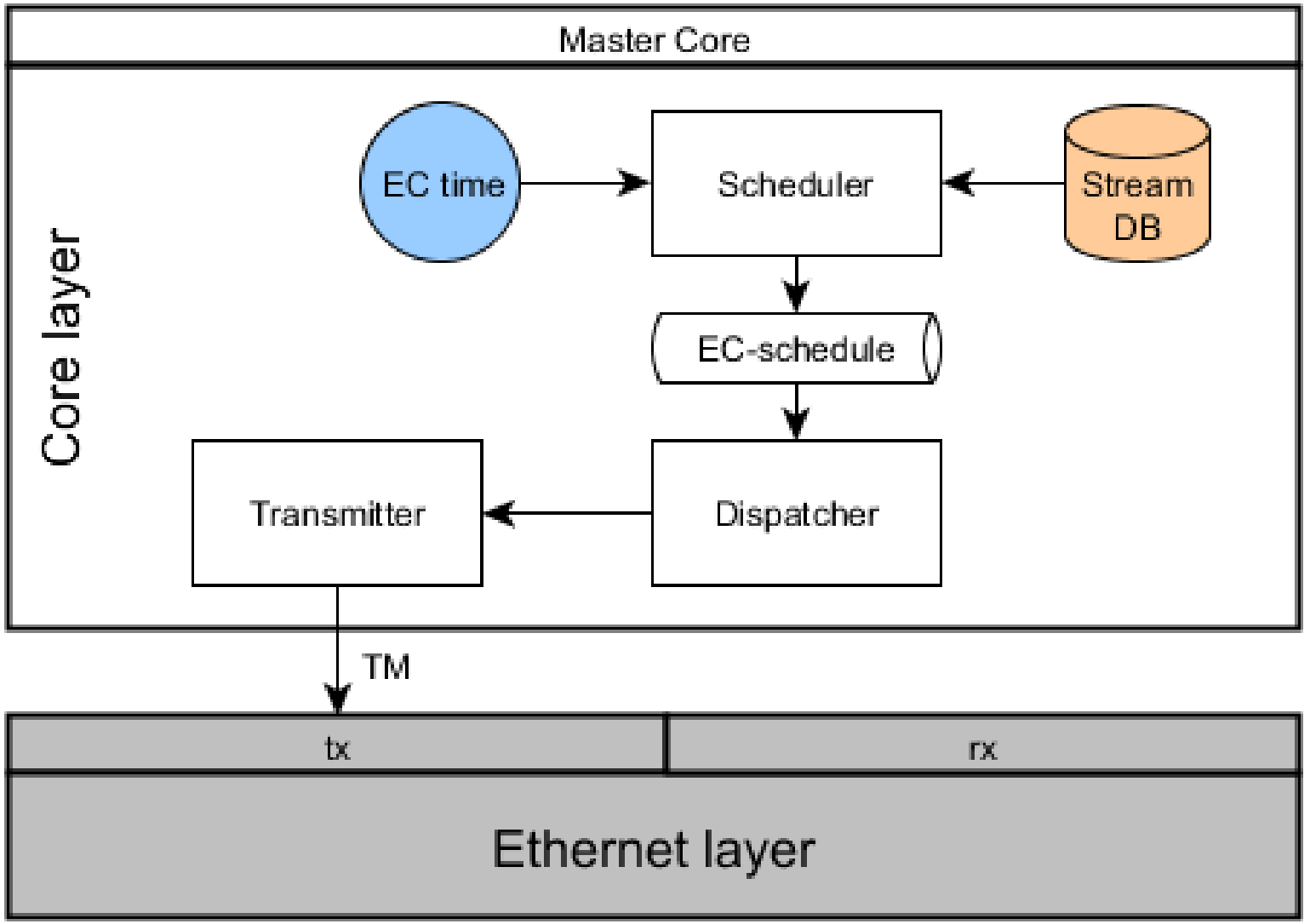
- Linux OS 

- Lenguaje C 

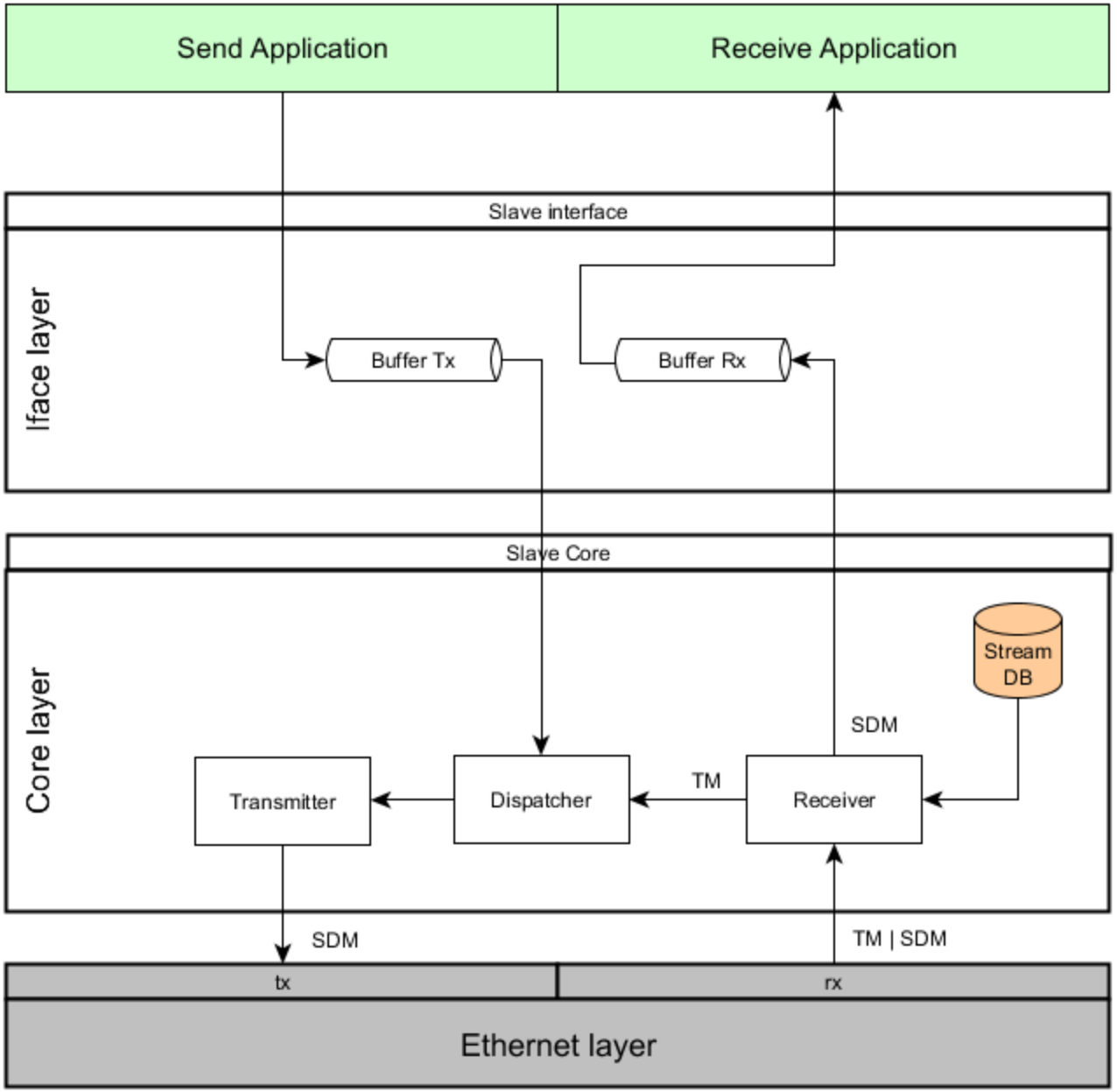
- CMake 

- Wireshark 

Trabajo Desarrollado - Master

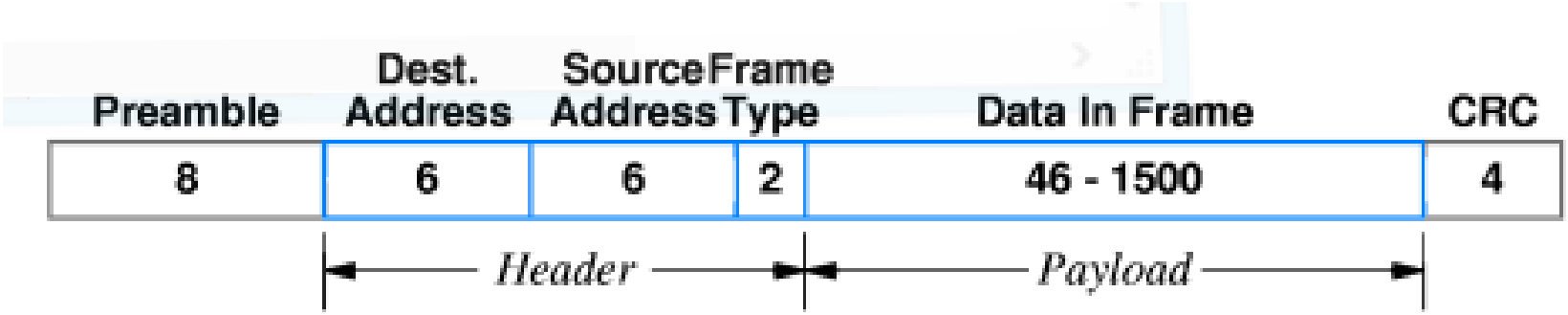


Trabajo Desarrollado - Esclavo



Trabajo Desarrollado - Mensajes

Mensaje Ethernet



- Header → MAC destino, MAC origen, TipoEth
- Payload → Mensaje FTT

LONGITUD MAX ETHERNET = 1514 bytes

Trabajo Desarrollado - Mensajes

Mensajes FTT

Trigger Message (TM)

FTT type [value = 0]	EC num [value = 0 - $4.29 \cdot 10^9$]	EC duration (ms) [value = 1 - 65535]	nsm [value = 0 - 255]	scheduling [length = nsm*2 bytes]
1 byte	4 bytes	2 bytes	1 byte	0 - 510 bytes

- Tipo FTT = 0
- Número de secuencia EC
- Duración del EC (en ms)
- Número de mensajes síncronos
- Scheduling: organizado en bloques de 2 bytes
  - stream ID
  - publisher node ID

Trabajo Desarrollado - Mensajes

Mensajes FTT

Synchronous Data Message (SDM)

FTT type [value = 1]	Stream ID	Data
1 byte	1 byte	0 - 1498 bytes

- Tipo FTT = 1
- Stream ID
- Datos

Índice

- Introducción
- FTT (Flexible Time Triggered)
- Trabajo desarrollado
- **Experimentos**
- Conclusiones y trabajo futuro

Experimentos - General

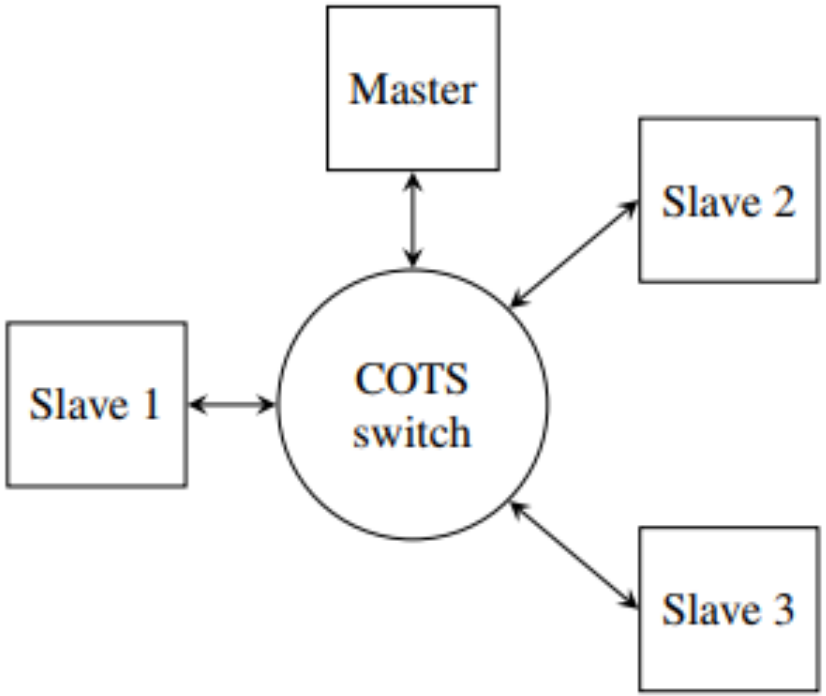
Variables de un experimento cualquiera con este proyecto

1. Contenido de la stream Database
2. Número de esclavos conectados
3. Aplicaciones ejecutadas en los esclavos

Experimentos - Avanzado

Características del experimento

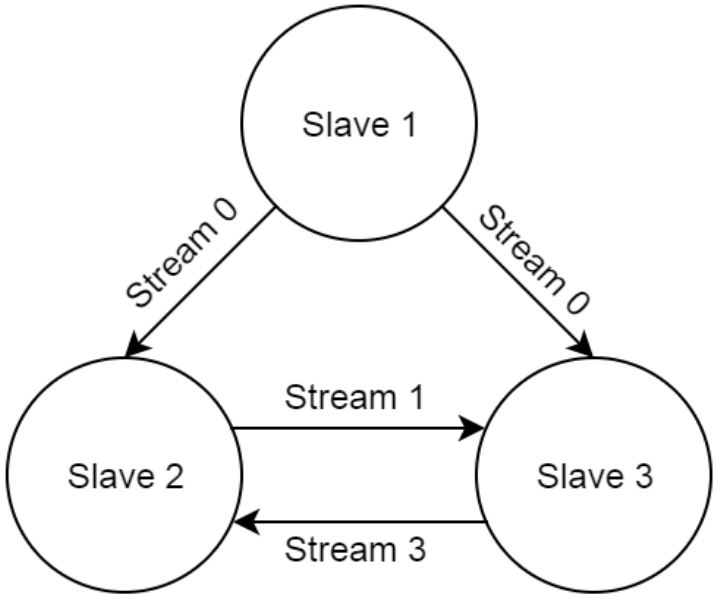
- Prototipo hardware
  - 1 master
  - 3 esclavos
  - Switch comercial



Experimentos - Avanzado

Características del experimento

- Configuración en la **stream Database** variada para tener diferentes tipos de comunicación
  - Stream 0: Multicast
  - Stream 1 y 3: Comunicación bidireccional
  - Stream 2: No existe

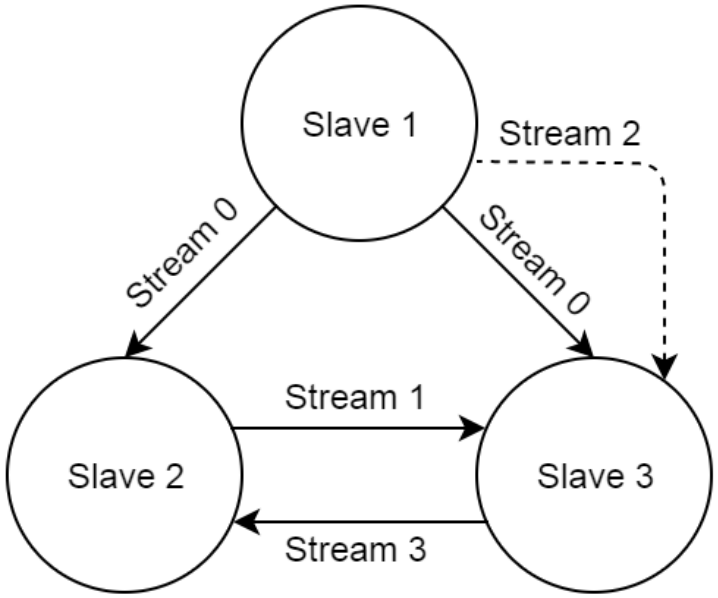


a) main

Experimentos - Avanzado

Características del experimento

- Configuración en la **stream Database** variada para tener diferentes tipos de comunicación
  - Stream 0: Multicast
  - Stream 1 y 3: Comunicación bidireccional
  - Stream 2: Unicast

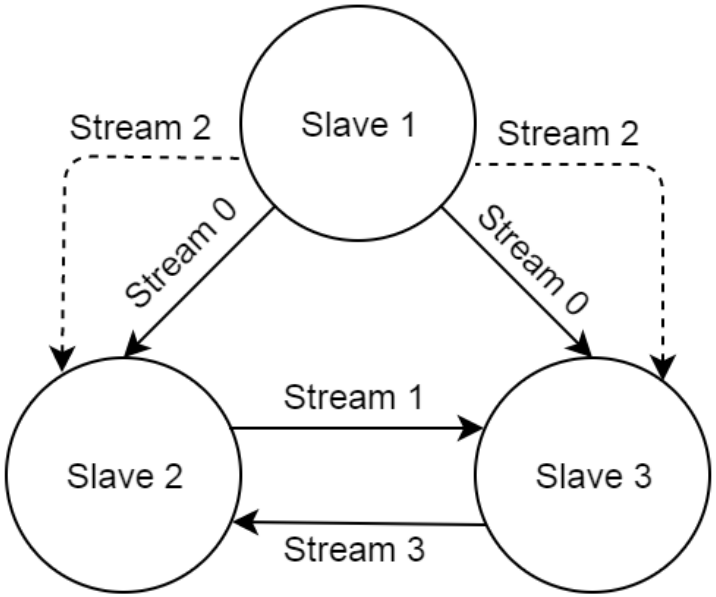


b) EC 3

Experimentos - Avanzado

Características del experimento

- Configuración en la **stream Database** variada para tener diferentes tipos de comunicación
  - Stream 0: Multicast
  - Stream 1 y 3: Comunicación bidireccional
  - Stream 2: Multicast

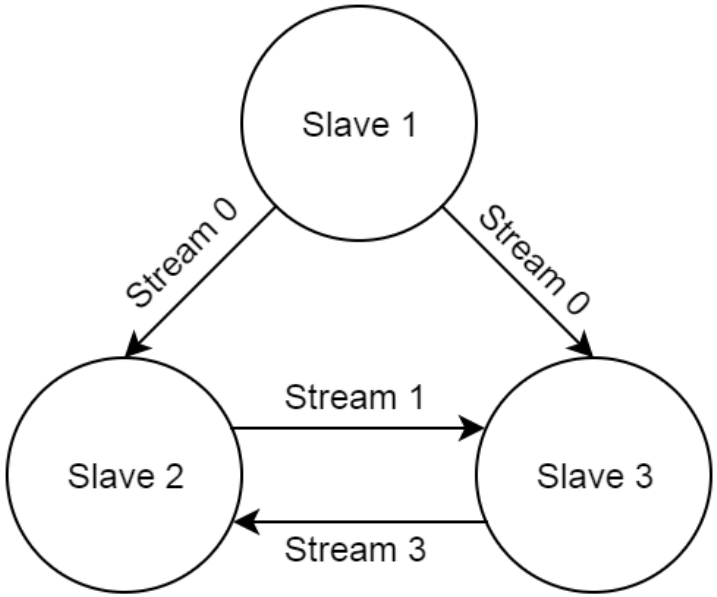


c) EC 4, 5, 6

Experimentos - Avanzado

Características del experimento

- Configuración en la **stream Database** variada para tener diferentes tipos de comunicación
  - Stream 0: Multicast
  - Stream 1 y 3: Comunicación bidireccional
  - Stream 2: No existe



a) main

Experimentos - Avanzado

Características del experimento

- **Stream 0 (rojo)**
  - Periodo = 3 ECs y Offset = 0 *(entre EC 0 y EC 9)*
  - Periodo = 5 ECs y Offset = 0 *(a partir del EC 10)*
- **Stream 2 (azul)**
  - Periodo = 1 EC y Offset = 0 *(activo entre EC 3 y EC 6)*
- **Stream 1 (verde)**
  - Periodo = 2 ECs y Offset = 1 ECs
- **Stream 3 (amarillo)**
  - Periodo = 2 ECs y Offset = 0 ECs

Experimentos - Avanzado

Resultados con Wireshark (1)

	No.	Time	Source	Destination	Protocol	Length	Info
EC 0	1	0.000000	Private_01:01:01	Broadcast	0x8ff0	26	Ethernet II
	2	0.000150	Private_01:02:03	Broadcast	0x8ff0	60	Ethernet II
	3	0.000002	Private_01:02:01	Broadcast	0x8ff0	60	Ethernet II
EC 1	4	0.999958	Private_01:01:01	Broadcast	0x8ff0	24	Ethernet II
	5	0.000133	Private_01:02:02	Broadcast	0x8ff0	60	Ethernet II
EC 2	6	0.999846	Private_01:01:01	Broadcast	0x8ff0	24	Ethernet II
	7	0.000214	Private_01:02:03	Broadcast	0x8ff0	60	Ethernet II
EC 3	8	0.999806	Private_01:01:01	Broadcast	0x8ff0	28	Ethernet II
	9	0.000130	Private_01:02:01	Broadcast	0x8ff0	60	Ethernet II
	10	0.000002	Private_01:02:01	Broadcast	0x8ff0	60	Ethernet II
	11	0.000115	Private_01:02:02	Broadcast	0x8ff0	60	Ethernet II
EC 4	12	0.999752	Private_01:01:01	Broadcast	0x8ff0	26	Ethernet II
	13	0.000125	Private_01:02:01	Broadcast	0x8ff0	60	Ethernet II
	14	0.000002	Private_01:02:03	Broadcast	0x8ff0	60	Ethernet II
EC 5	15	0.999879	Private_01:01:01	Broadcast	0x8ff0	26	Ethernet II
	16	0.000125	Private_01:02:01	Broadcast	0x8ff0	60	Ethernet II
	17	0.000002	Private_01:02:02	Broadcast	0x8ff0	60	Ethernet II
EC 6	18	0.999860	Private_01:01:01	Broadcast	0x8ff0	28	Ethernet II
	19	0.000128	Private_01:02:03	Broadcast	0x8ff0	60	Ethernet II
	20	0.000002	Private_01:02:01	Broadcast	0x8ff0	60	Ethernet II
	21	0.000002	Private_01:02:01	Broadcast	0x8ff0	60	Ethernet II
EC 7	22	0.999877	Private_01:01:01	Broadcast	0x8ff0	24	Ethernet II
	23	0.000126	Private_01:02:02	Broadcast	0x8ff0	60	Ethernet II
EC 8	24	0.999875	Private_01:01:01	Broadcast	0x8ff0	24	Ethernet II
	25	0.000147	Private_01:02:03	Broadcast	0x8ff0	60	Ethernet II
EC 9	26	0.999856	Private_01:01:01	Broadcast	0x8ff0	26	Ethernet II
	27	0.000131	Private_01:02:01	Broadcast	0x8ff0	60	Ethernet II
	28	0.000002	Private_01:02:02	Broadcast	0x8ff0	60	Ethernet II
EC 10	29	0.999851	Private_01:01:01	Broadcast	0x8ff0	26	Ethernet II
	30	0.000131	Private_01:02:03	Broadcast	0x8ff0	60	Ethernet II
	31	0.000009	Private_01:02:01	Broadcast	0x8ff0	60	Ethernet II
EC 11	32	0.999872	Private_01:01:01	Broadcast	0x8ff0	24	Ethernet II
	33	0.000129	Private_01:02:02	Broadcast	0x8ff0	60	Ethernet II
EC 12	34	0.999872	Private_01:01:01	Broadcast	0x8ff0	24	Ethernet II
	35	0.000129	Private_01:02:03	Broadcast	0x8ff0	60	Ethernet II
EC 13	36	0.999855	Private_01:01:01	Broadcast	0x8ff0	24	Ethernet II
	37	0.000140	Private_01:02:02	Broadcast	0x8ff0	60	Ethernet II

Experimentos - Avanzado

Resultados con Wireshark (2)

	No.	Time	Source	Destination	Protocol	Length	Info
	37	0.000140	Private_01:02:02	Broadcast	0x8ff0	60	Ethernet II
EC 14	38	0.999853	Private_01:01:01	Broadcast	0x8ff0	24	Ethernet II
	39	0.000129	Private_01:02:03	Broadcast	0x8ff0	60	Ethernet II
EC 15	40	0.999929	Private_01:01:01	Broadcast	0x8ff0	26	Ethernet II
	41	0.000128	Private_01:02:02	Broadcast	0x8ff0	60	Ethernet II
	42	0.000002	Private_01:02:01	Broadcast	0x8ff0	60	Ethernet II
EC 16	43	0.999835	Private_01:01:01	Broadcast	0x8ff0	24	Ethernet II
	44	0.000130	Private_01:02:03	Broadcast	0x8ff0	60	Ethernet II
EC 17	45	0.999870	Private_01:01:01	Broadcast	0x8ff0	24	Ethernet II
	46	0.000129	Private_01:02:02	Broadcast	0x8ff0	60	Ethernet II
EC 18	47	0.999848	Private_01:01:01	Broadcast	0x8ff0	24	Ethernet II
	48	0.000122	Private_01:02:03	Broadcast	0x8ff0	60	Ethernet II
EC 19	49	0.999901	Private_01:01:01	Broadcast	0x8ff0	24	Ethernet II
	50	0.000126	Private_01:02:02	Broadcast	0x8ff0	60	Ethernet II
EC 20	51	0.999871	Private_01:01:01	Broadcast	0x8ff0	26	Ethernet II
	52	0.000125	Private_01:02:03	Broadcast	0x8ff0	60	Ethernet II
	53	0.000003	Private_01:02:01	Broadcast	0x8ff0	60	Ethernet II
EC 21	54	0.999875	Private_01:01:01	Broadcast	0x8ff0	24	Ethernet II
	55	0.000132	Private_01:02:02	Broadcast	0x8ff0	60	Ethernet II
EC 22	56	0.999845	Private_01:01:01	Broadcast	0x8ff0	24	Ethernet II
	57	0.000121	Private_01:02:03	Broadcast	0x8ff0	60	Ethernet II
EC 23	58	0.999886	Private_01:01:01	Broadcast	0x8ff0	24	Ethernet II
	59	0.000141	Private_01:02:02	Broadcast	0x8ff0	60	Ethernet II
EC 24	60	0.999875	Private_01:01:01	Broadcast	0x8ff0	24	Ethernet II
	61	0.000128	Private_01:02:03	Broadcast	0x8ff0	60	Ethernet II
EC 25	62	0.999875	Private_01:01:01	Broadcast	0x8ff0	26	Ethernet II
	63	0.000131	Private_01:02:02	Broadcast	0x8ff0	60	Ethernet II
	64	0.000002	Private_01:02:01	Broadcast	0x8ff0	60	Ethernet II
EC 26	65	0.999841	Private_01:01:01	Broadcast	0x8ff0	24	Ethernet II
	66	0.000123	Private_01:02:03	Broadcast	0x8ff0	60	Ethernet II
EC 27	67	0.999900	Private_01:01:01	Broadcast	0x8ff0	24	Ethernet II
	68	0.000125	Private_01:02:02	Broadcast	0x8ff0	60	Ethernet II
EC 28	69	0.999868	Private_01:01:01	Broadcast	0x8ff0	24	Ethernet II
	70	0.000125	Private_01:02:03	Broadcast	0x8ff0	60	Ethernet II
EC 29	71	0.999883	Private_01:01:01	Broadcast	0x8ff0	24	Ethernet II
	72	0.000130	Private_01:02:02	Broadcast	0x8ff0	60	Ethernet II

Índice

- Introducción
- FTT (Flexible Time Triggered)
- Trabajo desarrollado
- Experimentos
- **Conclusiones y trabajo futuro**

Conclusiones

- El resultado es un prototipo funcional con un master y varios esclavos, cada uno ejecutando una aplicación destinada a transmitir o recibir mensajes
- Hemos probado el experimento en un entorno real y el resultado es el esperado

Se ha conseguido implementar un master y esclavo FTT que funciona correctamente

Trabajo Futuro

- **Ampliar** las funcionalidades de la **interfaz aplicación-esclavo**
- **Implementar** la transmission de los **Asynchronous Data Messages** (ADM)s
- **Implementar** la **gestión dinámica** de los **requisitos** de comunicación
- **Implementar** mecanismos de **tolerancia a fallos**

Finalmente, este proyecto será utilizado como punto de partida para la implementación el prototipo de la arquitectura de DFT4FTT

Universitat de les Illes Balears

Presentación de Trabajo Final de Grado (TFG) para el
Grado de Ingeniería Electrónica Industrial y Automática

Palma, Octubre 2017

Implementation and Validation of the Fundamental Mechanisms of the Flexible Time-Triggered Communication Paradigm for Ethernet-based Highly-Reliable Systems

Sergi Arguimbau Guarinos

Tutor: Alberto Ballesteros Varela