TTEthernet Development and CFS Integration

DSG CFS / Open SW Multi-lateral TIM
NASA JSC, Houston, TX
11 – 13 Sept. 2017

Andrew Loveless (NASA/JSC)
andrew.loveless@nasa.gov

TTE Chip-IP

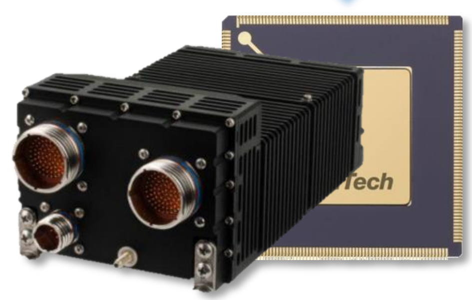
Gen. 2 (Phoenix IP) → Gen. 3 (Pegasus IP)

Lab → Space

- MAC layer only
- VL-based, no ports
- Proprietary loader
- ARINC 664-p7 like RC traffic

- UDP/IP layers in hardware
- Port-based with shared VLs
- ARINC 615A data loader
- ARINC 664-p7 RC traffic

- Includes TTTech and Honeywell IP
- Custom ASICs
- <u>High-Integrity end systems and switches</u>



- TTTech Chip-IP and ASICs
- Plastic package with ceramic planned (QML-V)
- <u>High-Integrity switches only</u>

TTE Chip-IP

Gen. 2 (Phoenix IP) → **Gen. 3 (Pegasus IP)**

Lab ↓ Space

Gen. 2 (Phoenix IP)

Lab

- MAC layer only
- VL-based, no ports
- Proprietary loader
- ARINC 664-p7 like RC traffic

Space

- Includes TTTech and Honeywell IP
- Custom ASICs
- <u>High-Integrity end systems and switches</u>

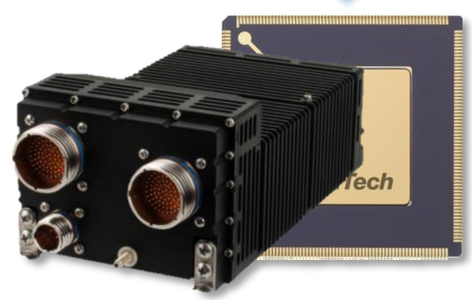
Gen. 3 (Pegasus IP)

Lab

We Are Using This

- UDP/IP layers in hardware
- Port-based with shared VLs
- ARINC 615A data loader
- ARINC 664-p7 RC traffic

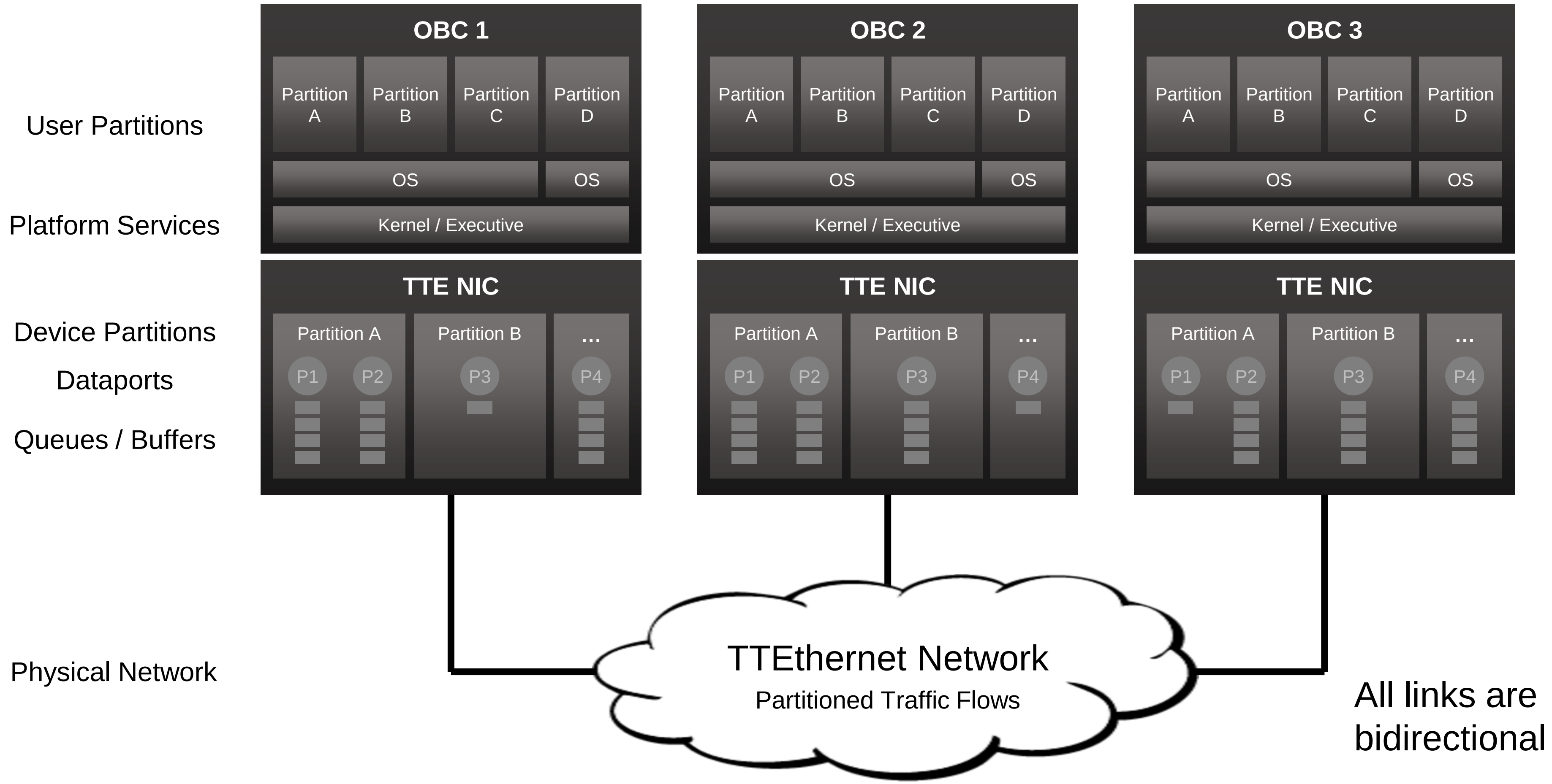
Space



- TTTech Chip-IP and ASICs
- Plastic package with ceramic planned (QML-V)
- <u>High-Integrity switches only</u>

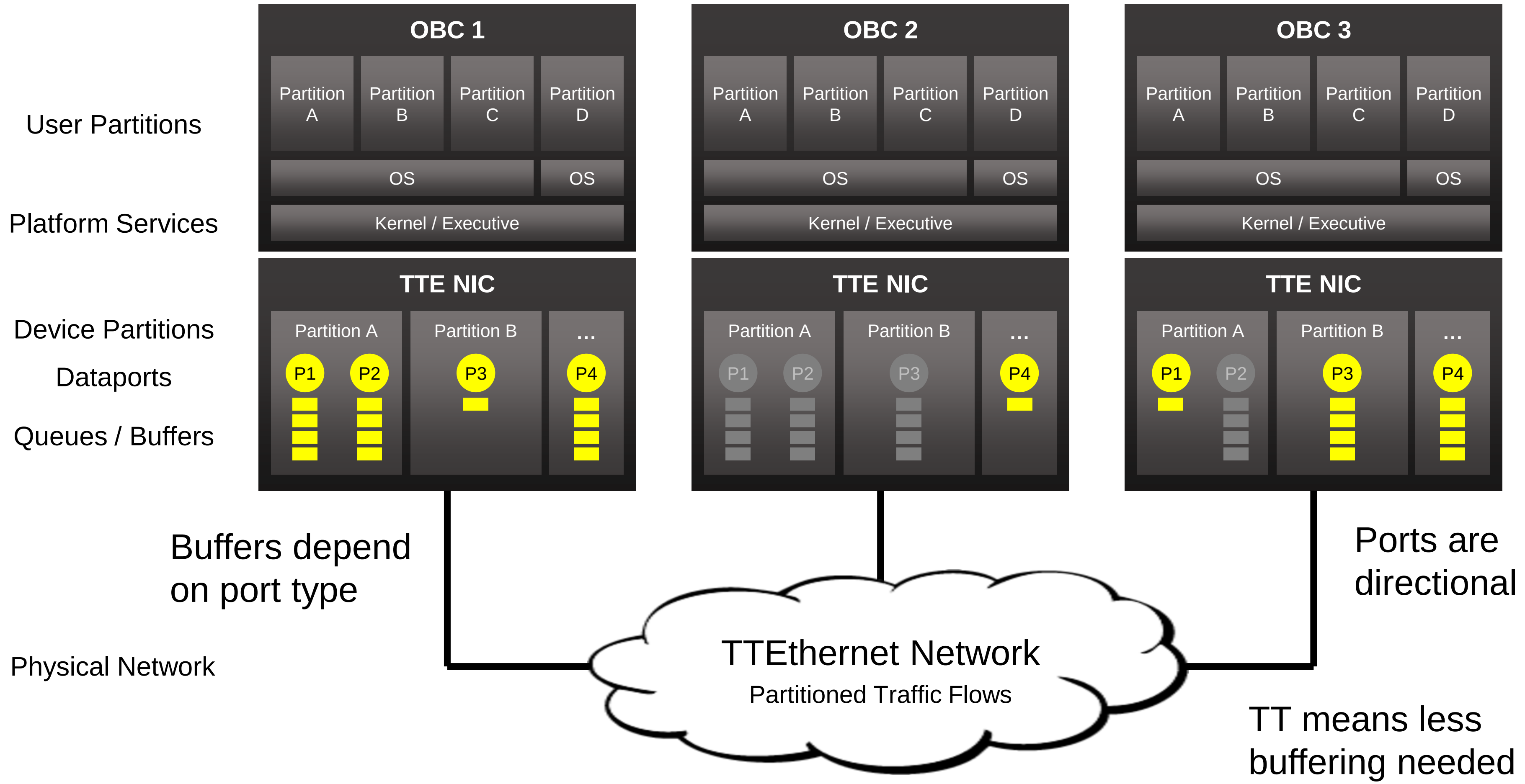
TTE Dataflow Concept

- A TTEthernet network is composed of end systems and switches.
- The end system contains the host and network controller (NIC).



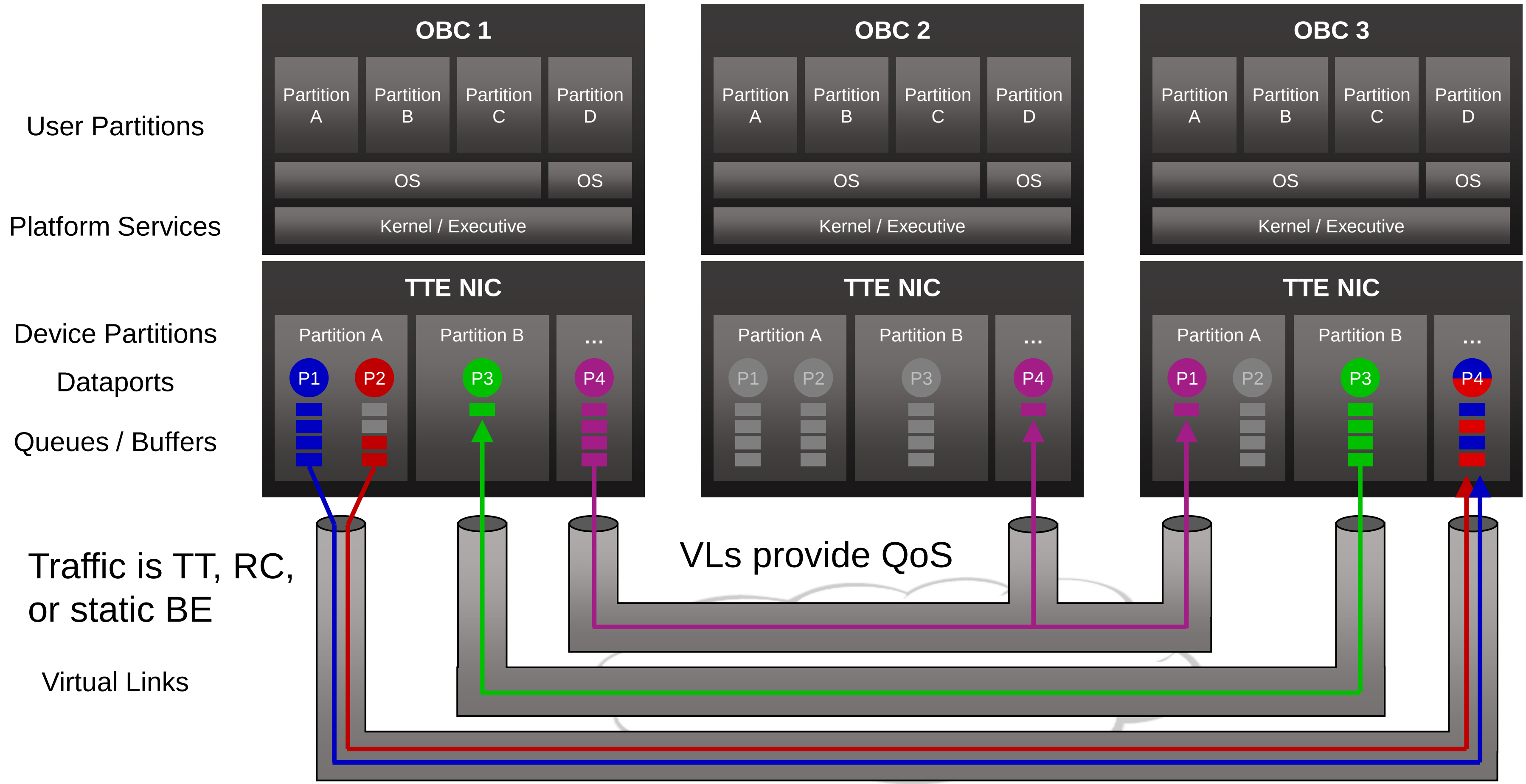
TTE Dataflow Concept (ports)

- TTE dataports represent the end points for all communication.
- Ports are Sampling or Queuing, and use UDP, IPv4, or MAC layers.



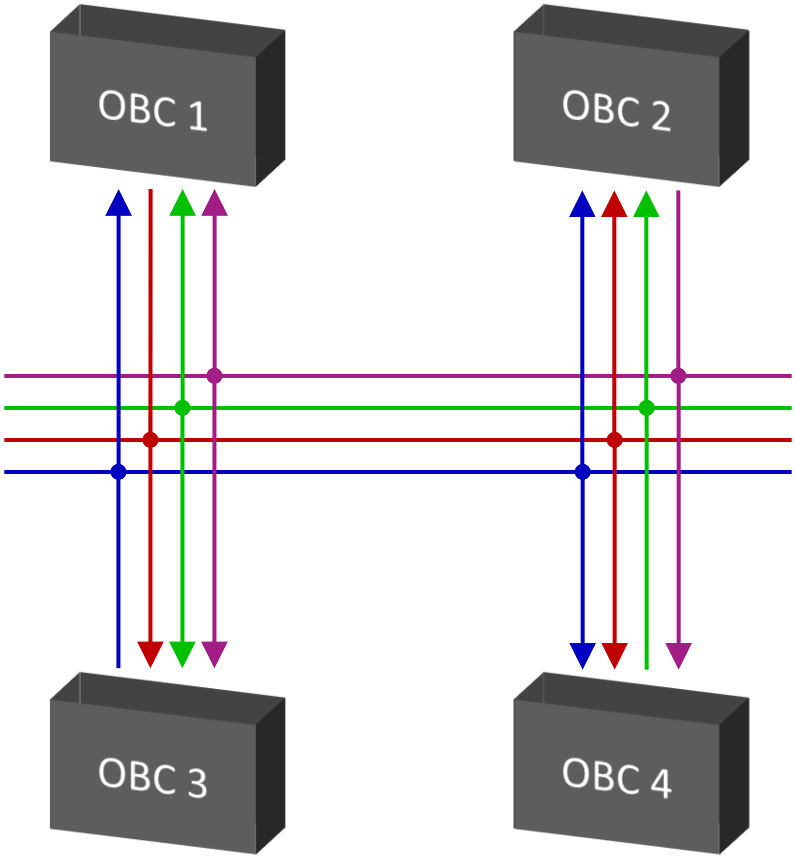
TTE Dataflow Concept (VLs)

- Ports are connected to one another by Virtual Links (VLs).
- Virtual Links are logically point-to-point – one sender, N receivers.

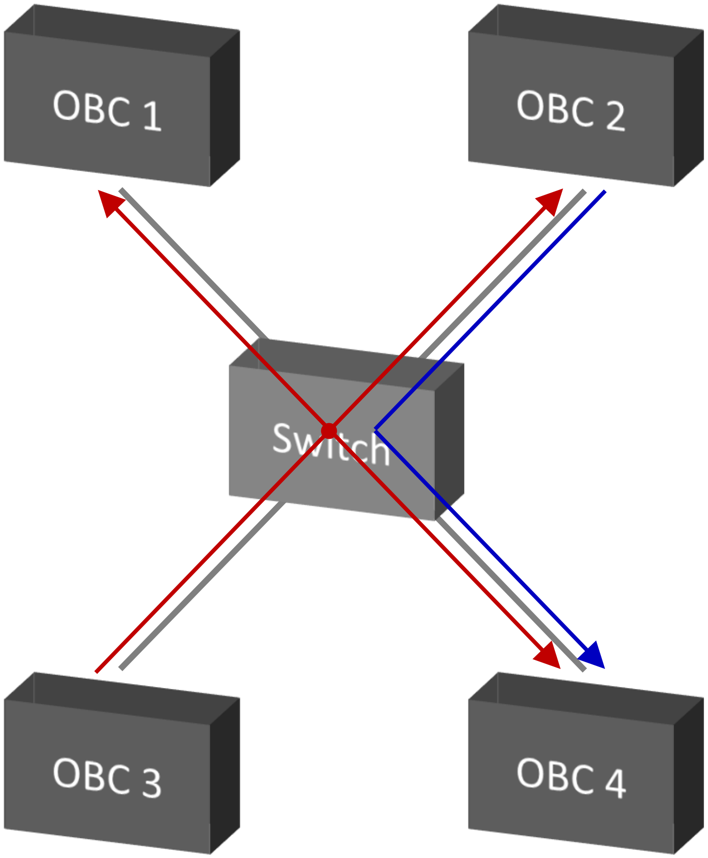


TTE Dataflow Concept (VLs)

- Traffic from different VLs is multiplexed over the same cabling.
- Logically the same as ARINC 429, except the “links” aren’t real.



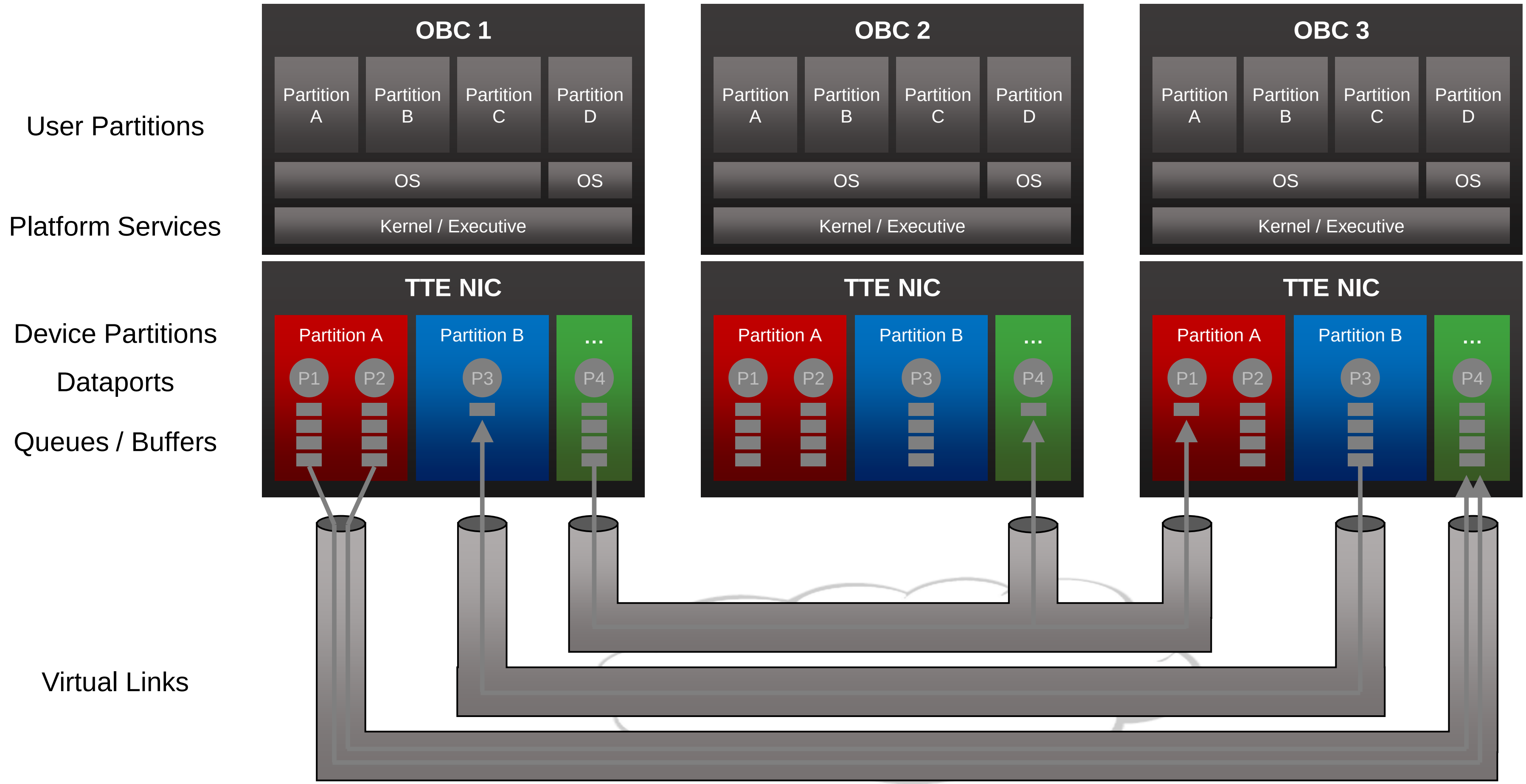
ARINC 429



TTE / ARINC 664

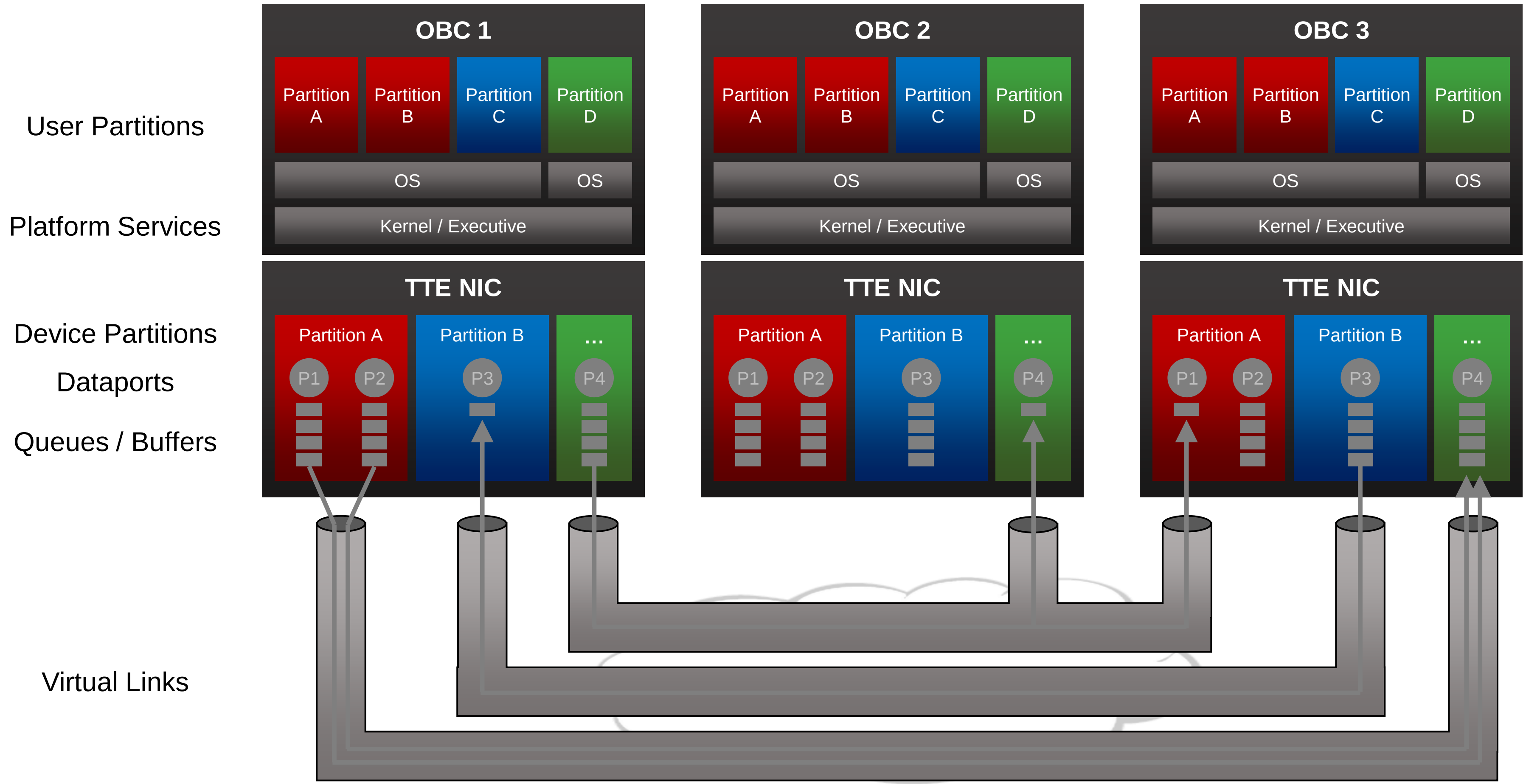
TTE Dataflow Concept (partitions)

- Ports are grouped in device partitions (8 per end system).
- Different partitions can be accessed simultaneously – no mutexes.



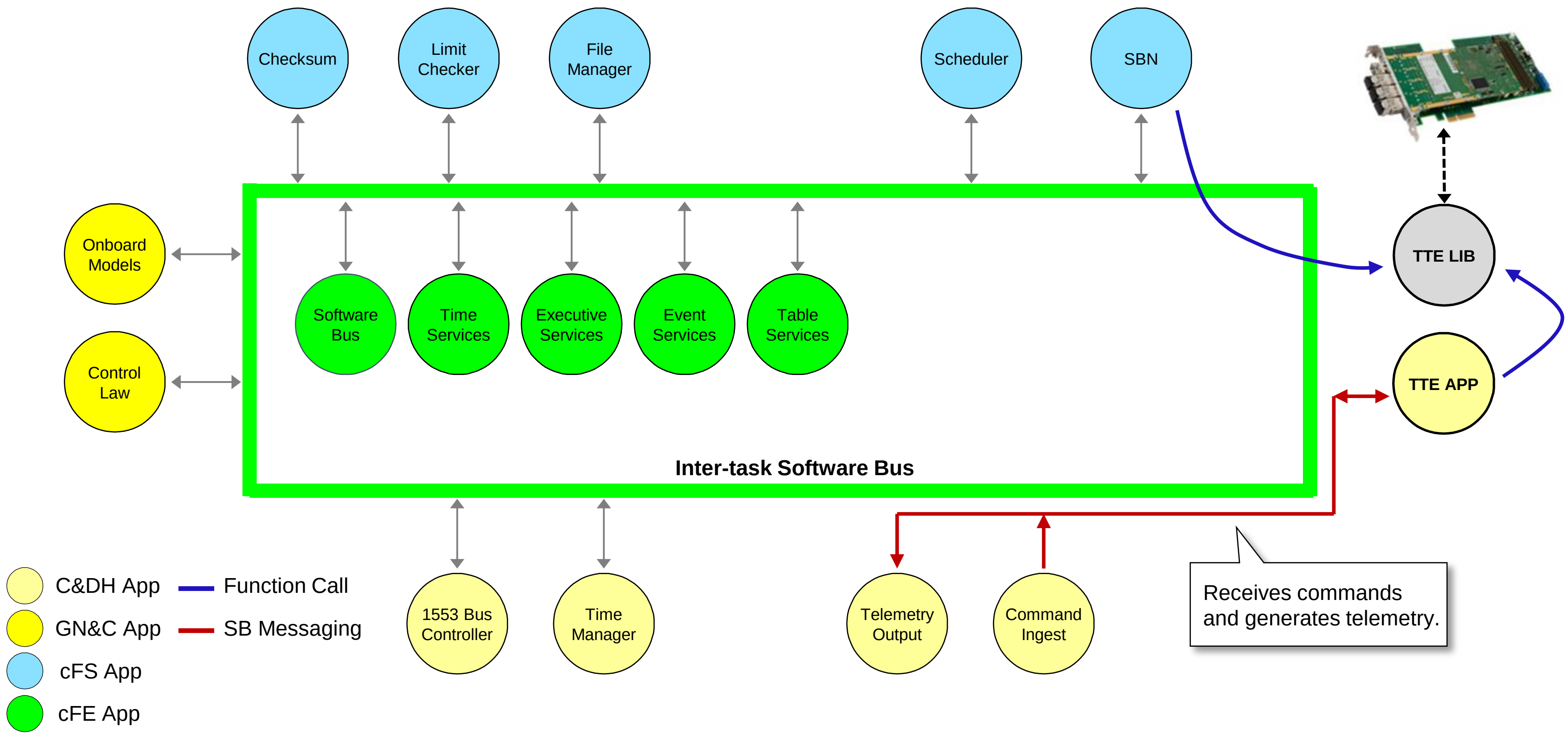
TTE Dataflow Concept (partitions)

- Device partitions can be mapped to the RTOS partitions.
- Thus partitioning is end to end – from app → network → app.



TTE CFS App

- The **TTE** application enables TTE communication in CFS.
- Includes pre-compiled drivers for supported targets and OSs.
- Provides API to claim ports and send or receive messages.



TTE CFS App (design)

Port Mapping Tables:
- Defines information needed to access local ports.
  - Types, direction, payload, etc.
- CFS apps claim ownership of ports via the API.

ES Config. Tables:
- Defines end system binary data resulting from scheduling process.

Apps

Libraries and API

Library Info:
- Sources: 22
- Headers: 26
- SLOC: 12,161
- OSs: Linux, VxWorks
  - Xenomai is planned.

TTE App
Main Task

Main App Info:
- Sources: 6
- Headers: 13 + 8
- SLOC: 6,521

IRQ Handler Task

Waits on semaphore

CI

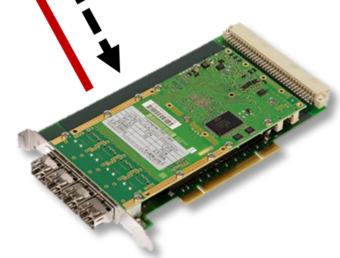
Prebuilt Drivers

Driver Info:
- OSs: Linux, VxWorks
  - Xenomai is planned.
- Arch.: x86, PowerPC

Commands:
- `TTE_SEND_HK_MID`
- `TTE_CMD_MID`
  - `TTE_NOOP_CC`
  - `TTE_RESET_CC`
  - `TTE_SEND_DIAG_CC`
- Commands for table config may be added in the future.

Telemetry:
- `TTE_HK_TLM_MID`
  - General state, status, counters, etc.
- `TTE_DIAG_TLM_MID`
  - Detailed port status information.

HK, TO



TTE Network Controller

SB Messages

Function Calls

Input / Output

Interrupt

TTE CFS App (design)

Port Mapping Tables:
- Defines information needed to access local ports.
  - Types, direction, payload, etc.
- CFS apps claim ownership of ports via the API.

ES Config. Tables:
- Defines end system binary data resulting from scheduling process.

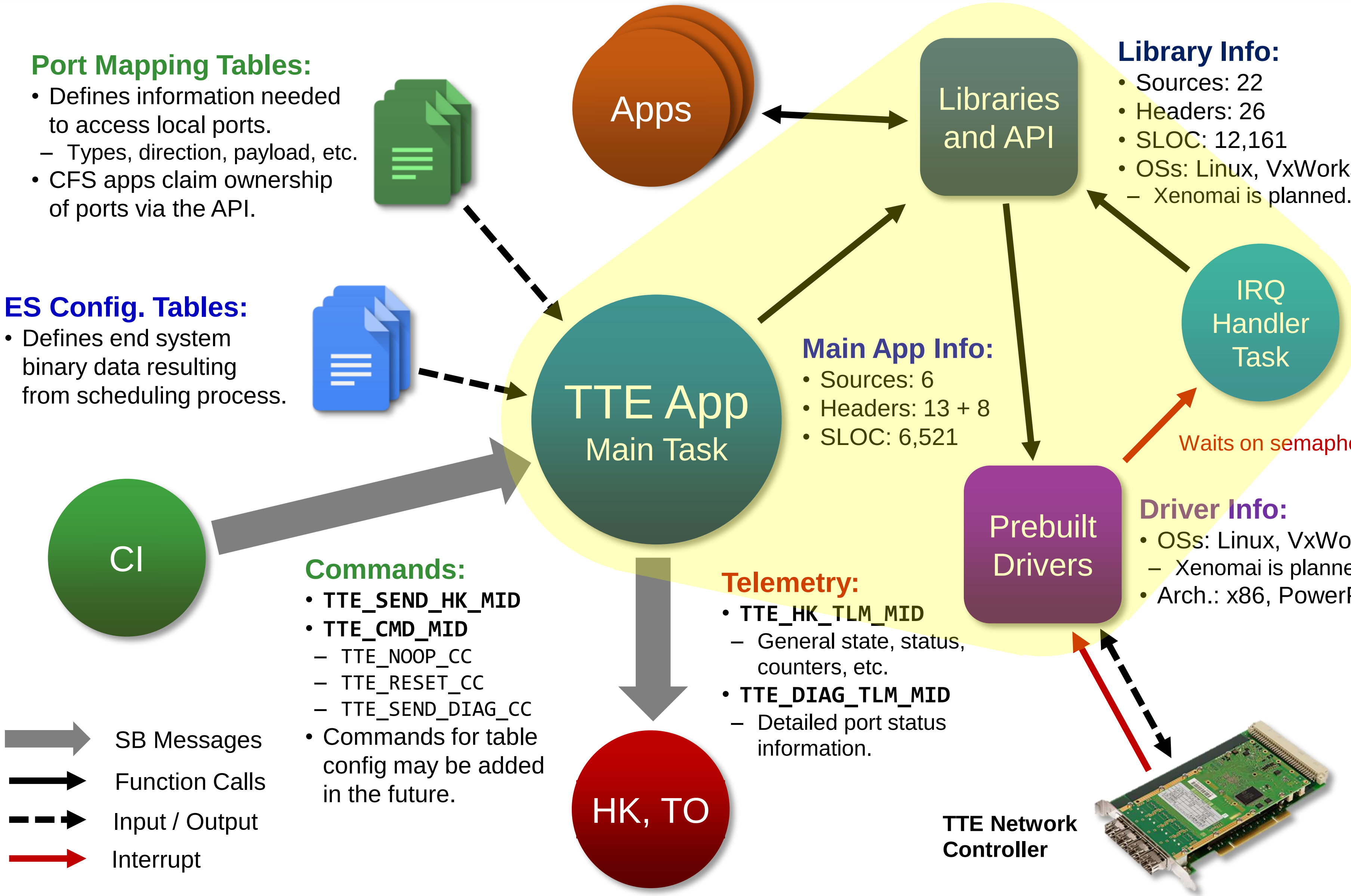
Library Info:
- Sources: 22
- Headers: 26
- SLOC: 12,161
- OSs: Linux, VxWorks
  - Xenomai is planned.

Main App Info:
- Sources: 6
- Headers: 13 + 8
- SLOC: 6,521

Driver Info:
- OSs: Linux, VxWorks
  - Xenomai is planned.
- Arch.: x86, PowerPC

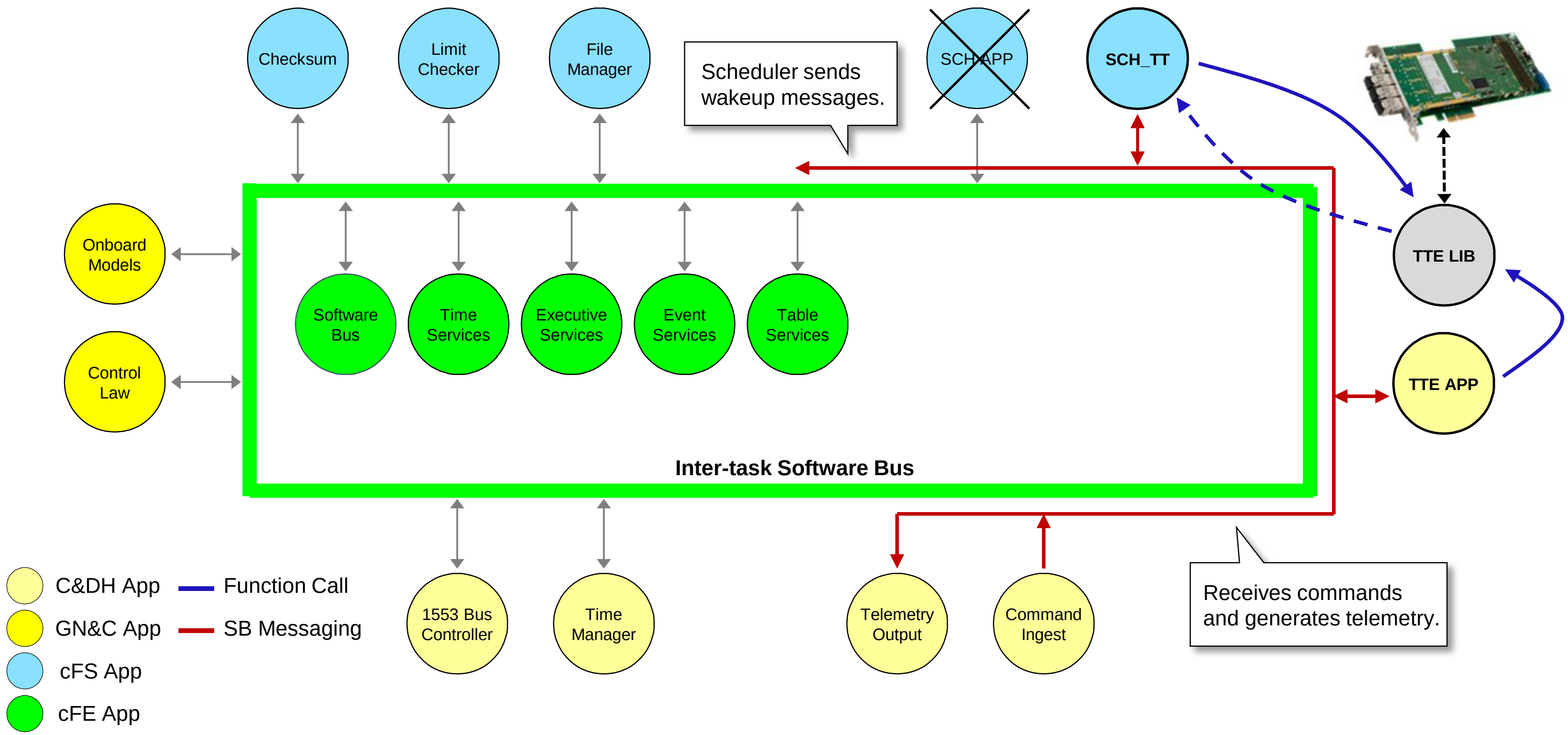
Commands:
- **`TTE_SEND_HK_MID`**
- **`TTE_CMD_MID`**
  - `TTE_NOOP_CC`
  - `TTE_RESET_CC`
  - `TTE_SEND_DIAG_CC`
- Commands for table config may be added in the future.

Telemetry:
- **`TTE_HK_TLM_MID`**
  - General state, status, counters, etc.
- **`TTE_DIAG_TLM_MID`**
  - Detailed port status information.

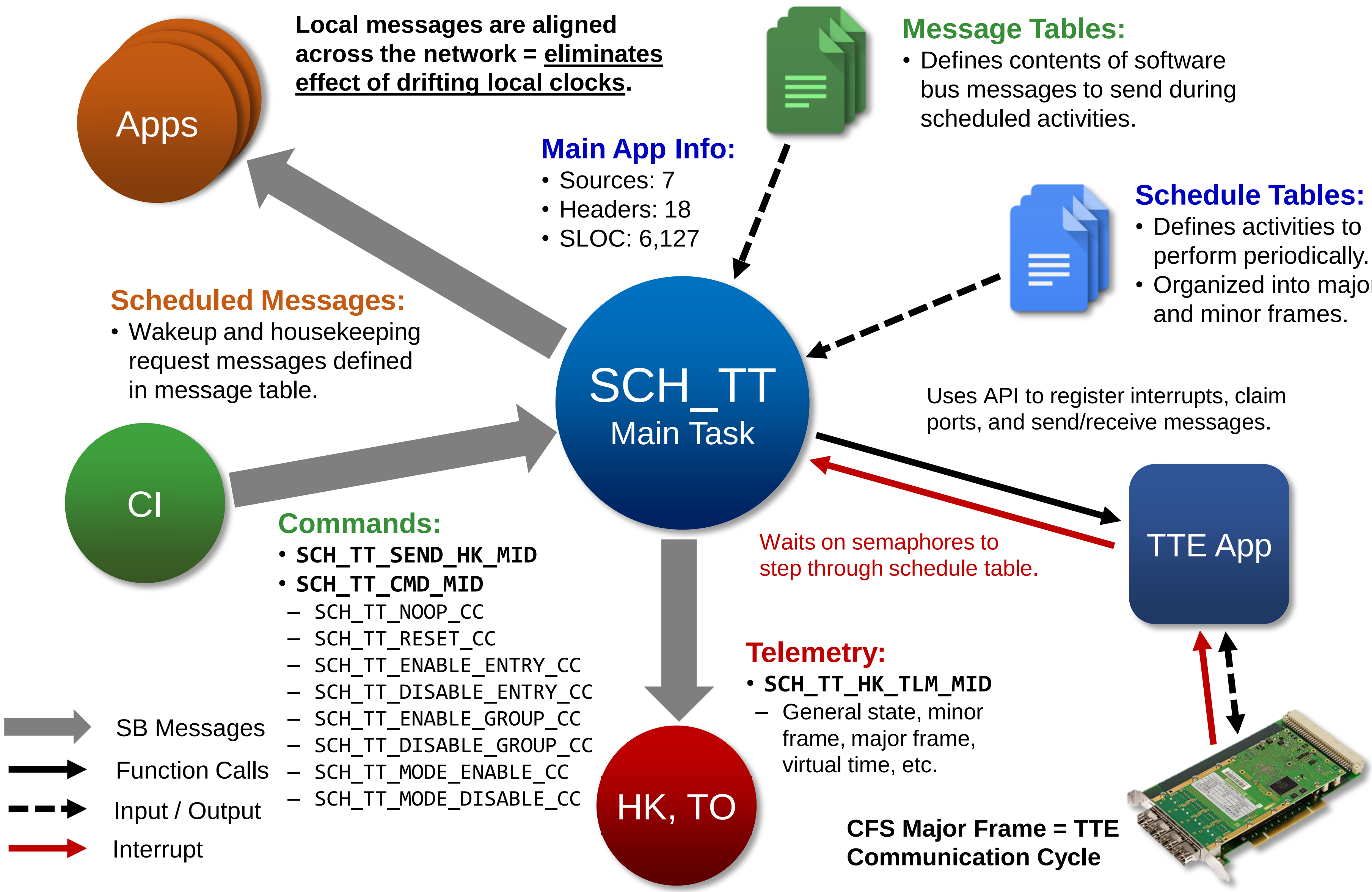


SCH_TT CFS App

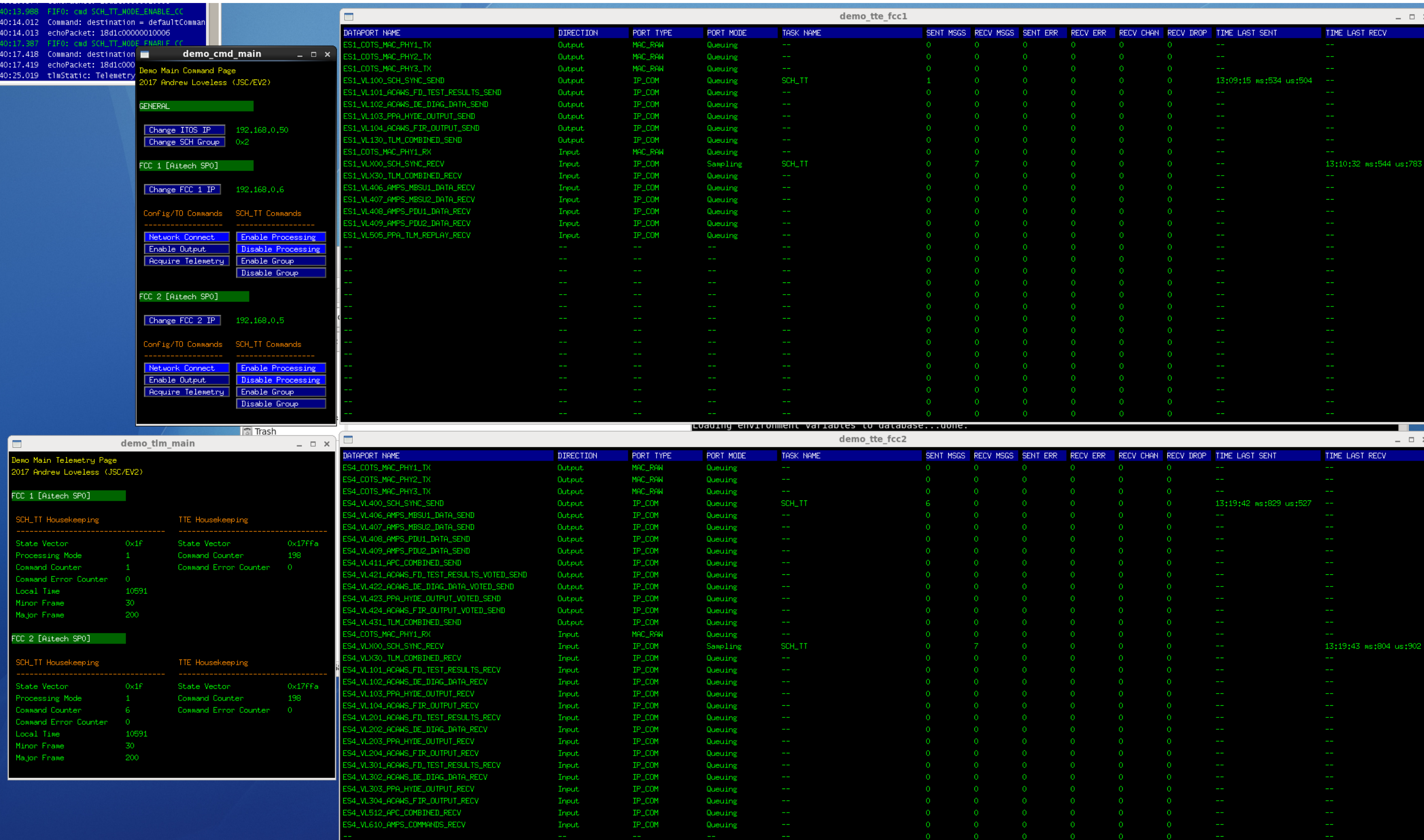
- **SCH_TT** is a drop-in replacement for the standard **SCH** scheduler.
- Drives the FSW execution according to scheduled interrupts.
- Aligns the task scheduling with the TT message scheduling.



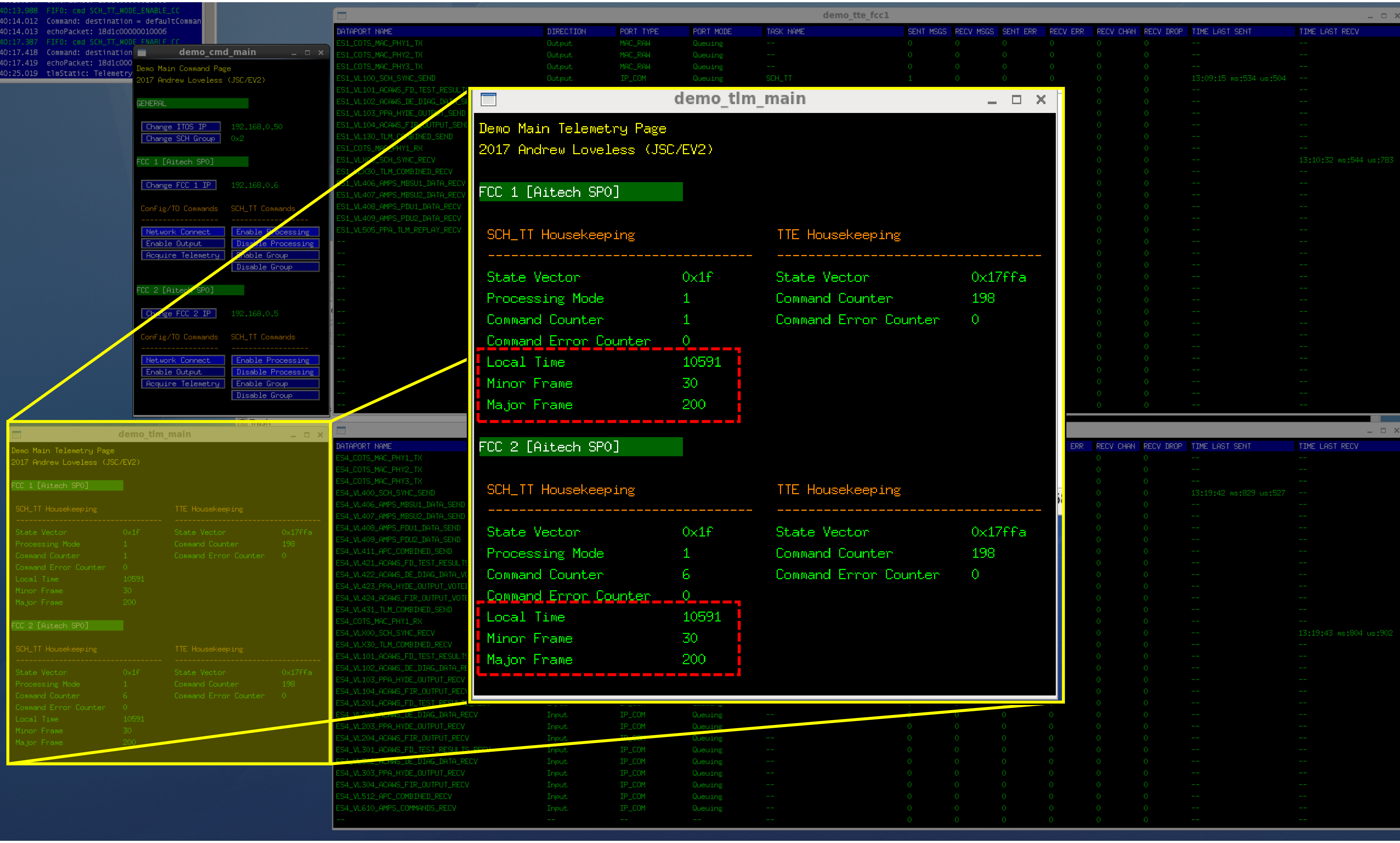
SCH_TT CFS App (design)



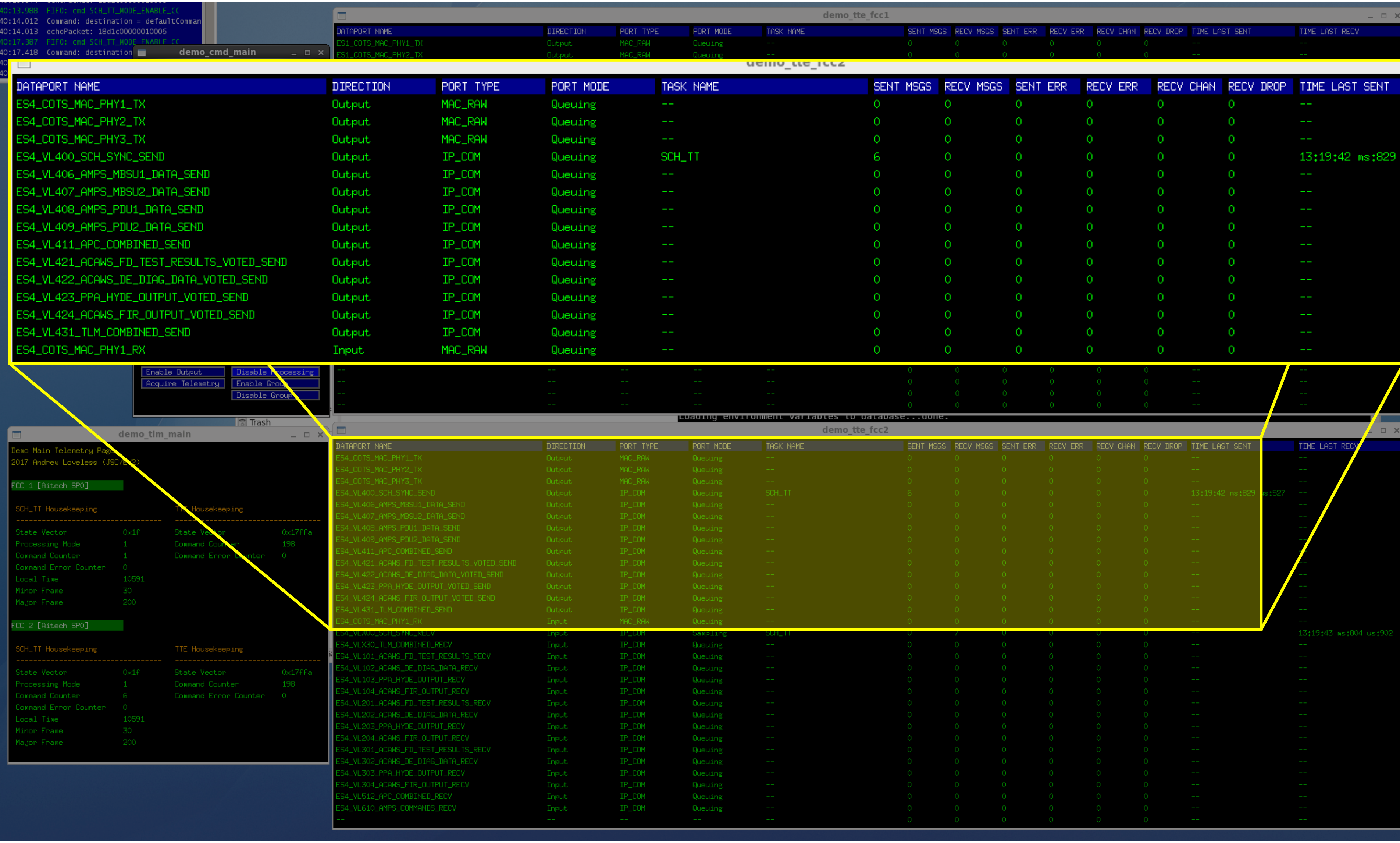
Telemetry Example



Telemetry Example (cont.)

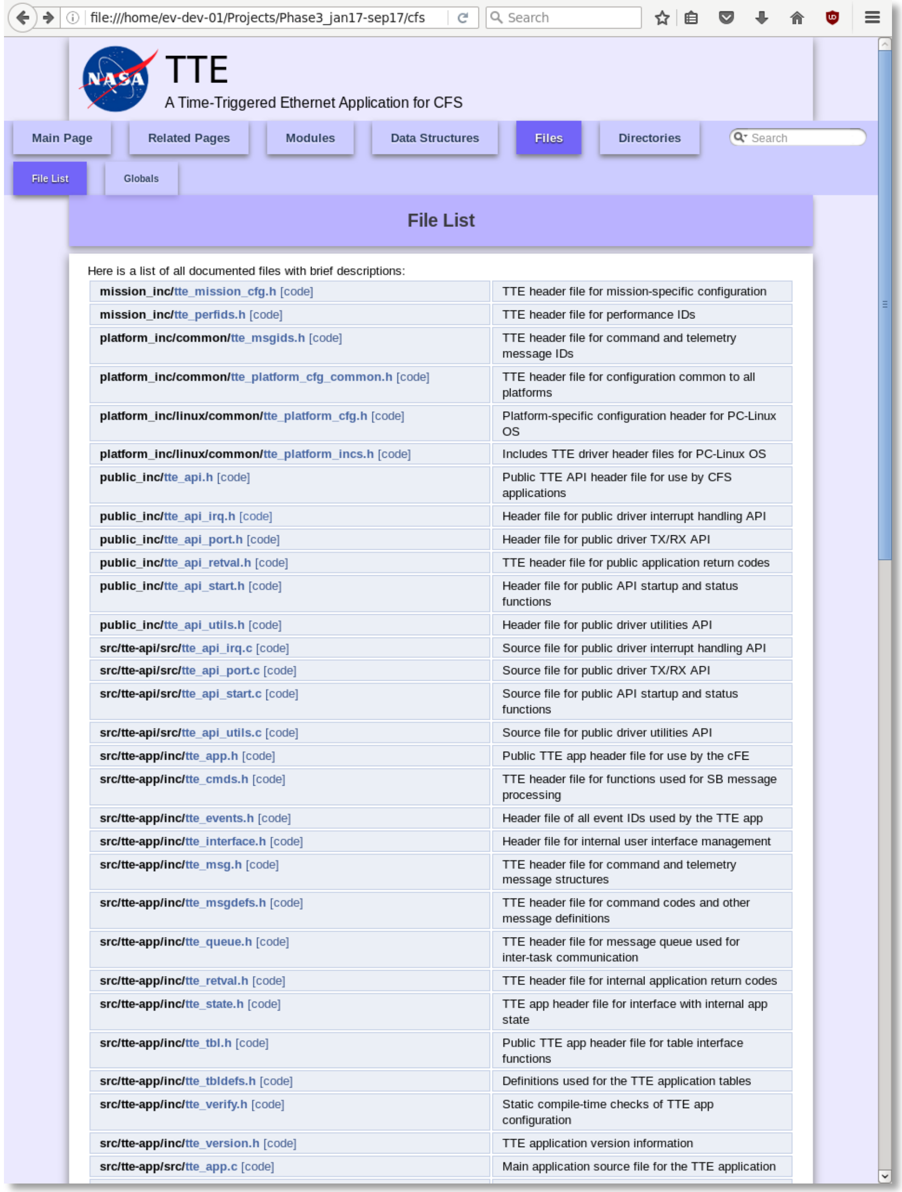


Telemetry Example (cont.)



App Availability

- Both planned to be open source, but are not yet.
- Both are fully documented.
  - Doxygen used for all sources.
- So far only tested on non-partitioned operating systems.
  - Linux, RTLinux, VxWorks.
- Currently only supports PowerPC and x86 targets.



mission_inc/tte_mission_cfg.h [code]	TTE header file for mission-specific configuration
mission_inc/tte_perfids.h [code]	TTE header file for performance IDs
platform_inc/common/tte_msgids.h [code]	TTE header file for command and telemetry message IDs
platform_inc/common/tte_platform_cfg_common.h [code]	TTE header file for configuration common to all platforms
platform_inc/linux/common/tte_platform_cfg.h [code]	Platform-specific configuration header for PC-Linux OS
platform_inc/linux/common/tte_platform_incs.h [code]	Includes TTE driver header files for PC-Linux OS
public_inc/tte_api.h [code]	Public TTE API header file for use by CFS applications
public_inc/tte_api_irq.h [code]	Header file for public driver interrupt handling API
public_inc/tte_api_port.h [code]	Header file for public driver TX/RX API
public_inc/tte_api_retval.h [code]	TTE header file for public application return codes
public_inc/tte_api_start.h [code]	Header file for public API startup and status functions
public_inc/tte_api_utils.h [code]	Header file for public driver utilities API
src/tte-api/src/tte_api_irq.c [code]	Source file for public driver interrupt handling API
src/tte-api/src/tte_api_port.c [code]	Source file for public driver TX/RX API
src/tte-api/src/tte_api_start.c [code]	Source file for public API startup and status functions
src/tte-api/src/tte_api_utils.c [code]	Source file for public driver utilities API
src/tte-app/inc/tte_app.h [code]	Public TTE app header file for use by the cFE
src/tte-app/inc/tte_cmds.h [code]	TTE header file for functions used for SB message processing
src/tte-app/inc/tte_events.h [code]	Header file of all event IDs used by the TTE app
src/tte-app/inc/tte_interface.h [code]	Header file for internal user interface management
src/tte-app/inc/tte_msg.h [code]	TTE header file for command and telemetry message structures
src/tte-app/inc/tte_msgdefs.h [code]	TTE header file for command codes and other message definitions
src/tte-app/inc/tte_queue.h [code]	TTE header file for message queue used for inter-task communication
src/tte-app/inc/tte_retval.h [code]	TTE header file for internal application return codes
src/tte-app/inc/tte_state.h [code]	TTE app header file for interface with internal app state
src/tte-app/inc/tte_tbl.h [code]	Public TTE app header file for table interface functions
src/tte-app/inc/tte_tbldefs.h [code]	Definitions used for the TTE application tables
src/tte-app/inc/tte_verify.h [code]	Static compile-time checks of TTE app configuration
src/tte-app/inc/tte_version.h [code]	TTE application version information
src/tte-app/src/tte_app.c [code]	Main application source file for the TTE application

Network and FSW Scheduling

Share with Partners

Undefined data exchange format

Network Scheduling

Knowledge Needed:

1. Physical properties of devices / platforms
2. Physical interconnect between platforms
3. Message flows and timing properties
4. Interrupt types and timing requirements
5. Port to device partition mapping

TTE Network Description (XML):

1. Network properties – e.g. sync, links, planes
2. Device properties – e.g. dataports, partitions
3. Virtual Links – e.g. periodicity, payload sizes

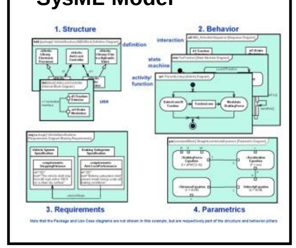
TTE Toolchain (TTTech):

- Converts network description to device-specific configurations
- Converts device configurations to .hex and .bin image files

TTE Toolchain (NASA):

- Converts .hex images to CFS end system config tables
- Converts device configs and network description to CFS port mapping tables.

SysML Model



Flight Software Scheduling

Knowledge Needed:

1. Task assignment to platforms and partitions
2. Timing requirements for task execution
3. Message structure and packet formats

Project Database

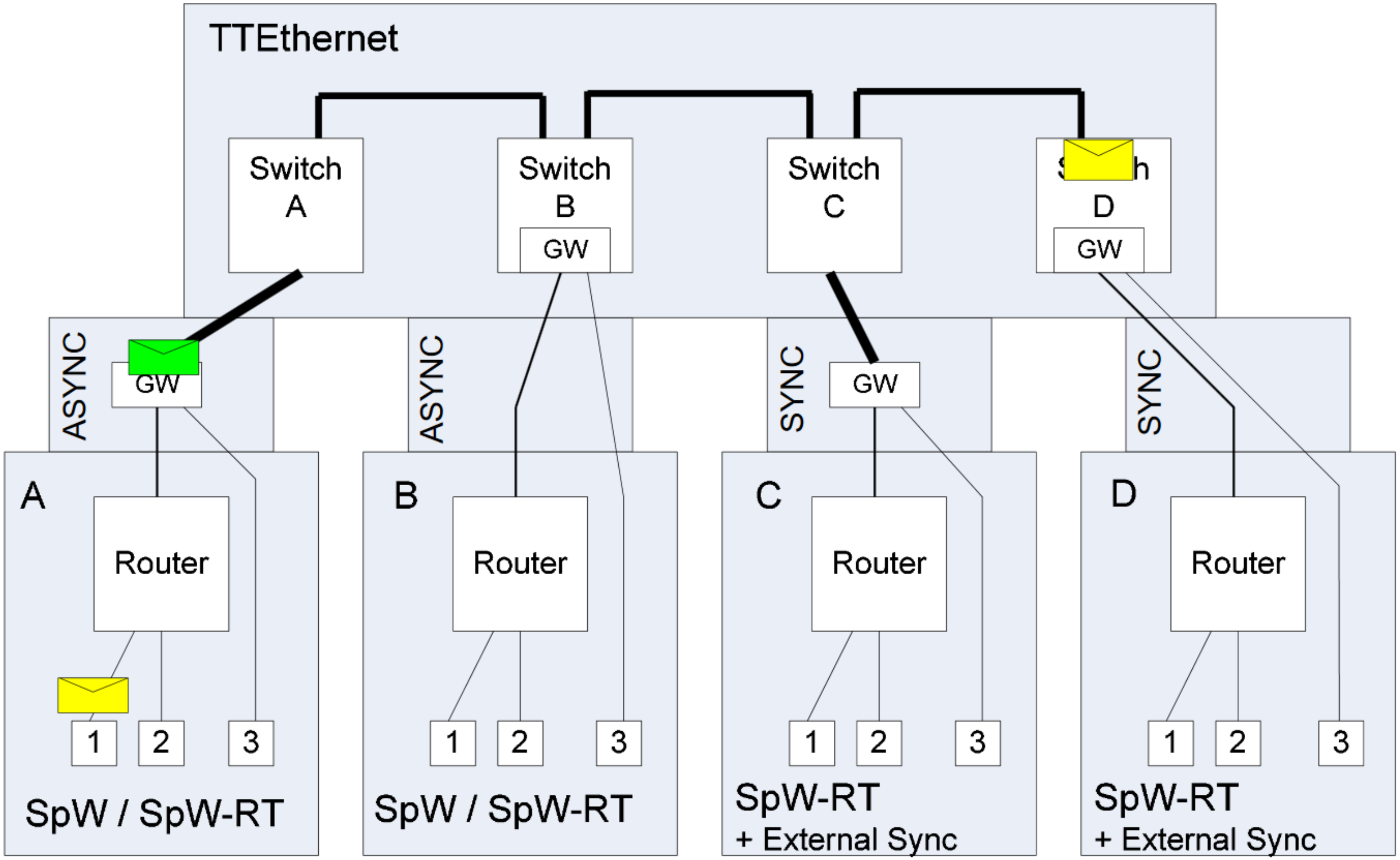
CCDD

Command & Data Dictionary

CFS Tables for SCH_TT App

CFS Tables for TTE App

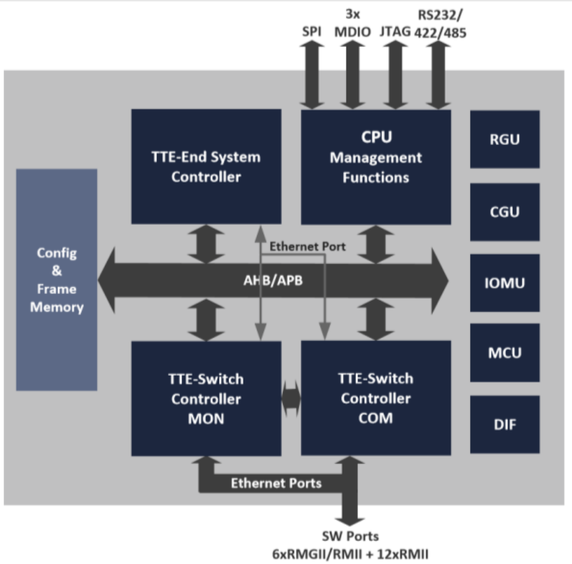
Integrating with Other Buses

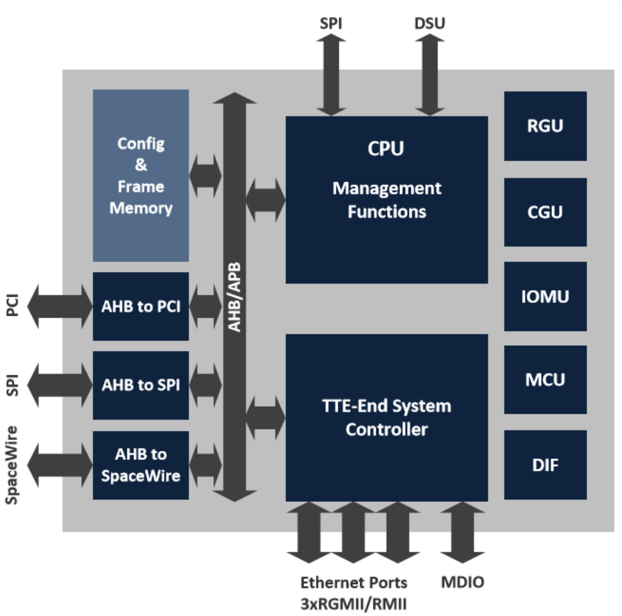


Steiner, Wilfried, “Towards Autonomous Time-Triggered Ethernet”, *Workshop on Spacecraft Flight Software*, Pasadena, CA, 2010.

TTE Hardware Status

- TTTech developing space ASIC for switches and end systems.
  - Testing plastic ASIC on the boards now.
  - Ceramic samples scheduled for Feb. 2018.
  - Fully qualified ceramic (QML-V) planned for Q3/2018.
- SEPHY targeting rad-hard PHY prototype for Dec. 2017.
  - Will perform network-level testing.
  - Electrical testing will have already been done.
- TI is in final development of DP83561R rad-tolerant PHY in ceramic.
  - First samples available in Q1/2018.
- Focus now is on 100/1000Base-T.
  - Some plans for Base-CX down the line.





Questions?