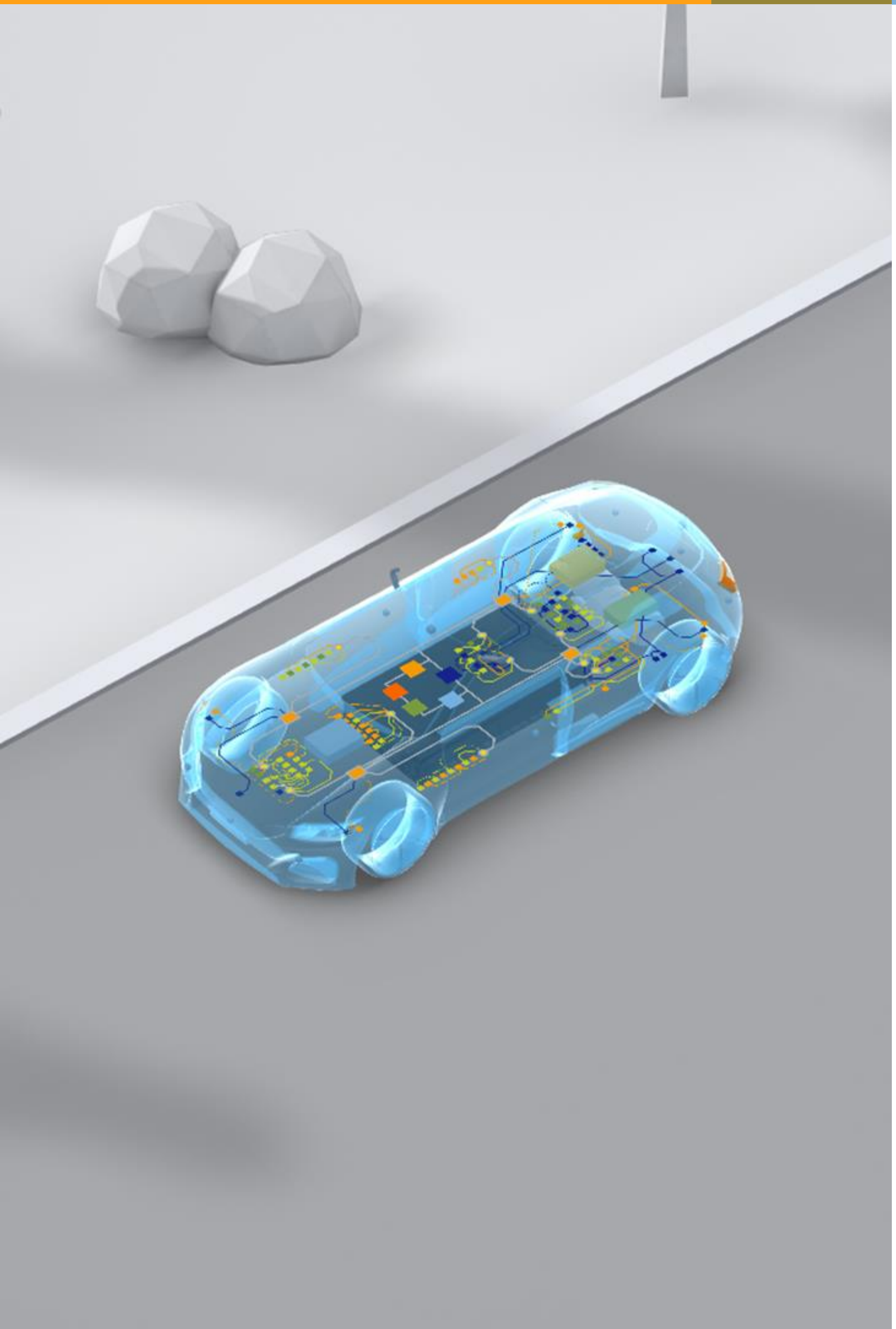
PFE/LLCE COMMUNICATIONS OFFLOAD ENGINES

Arthur Mackay, Systems & Applications Engineer
Chikita Nangia, Application Engineer

JUNE 2021

SECURE CONNECTIONS FOR A SMARTER WORLD

PUBLIC



Agenda

- Ethernet IP Evolution
- Packet Forwarding Engine (PFE) Overview
- PFE Features
- PFE Flow
- Example Use Cases
- PFE Software Stack
- Ethernet Security
- LLCE Background
- Introduction to LLCE, Low Latency Communication Engine
- LLCE Integration with AUTOSAR®
- LLCE Features
- LLCE Host Interface

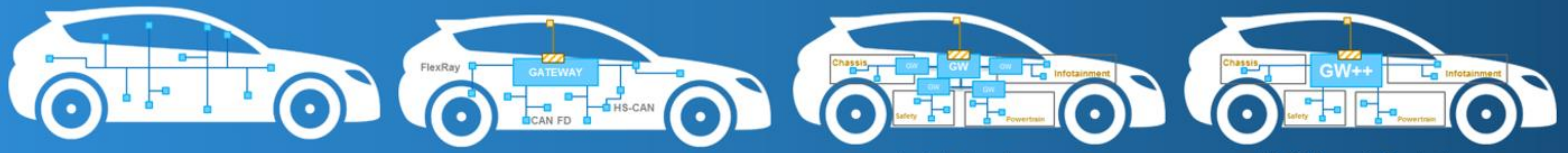
Ethernet IP Evolution

SECURE CONNECTIONS
FOR A SMARTER WORLD

PUBLIC

BANDWIDTH AND SECURITY TRANSFORM VEHICLE NETWORKS

Past	Today	Trending	Future



Past	Today	Trending	Future
Full network exposed to attacks (LOW protection level)	Gateway limits impact Unprotected sub-networks (MEDIUM protection level)	Isolated systems Domain GWs w/ controlled access (HIGH protection level)	Multi-Domain Controller Function clustering/virtualization (full self-driving car)
No Gateway	Central Gateway	Central GW + Domain Controllers	Central Compute / Gateway

PAST VEHICLE NETWORKS

- Dominated by classic CAN
- Limited bandwidth / scalability
- No security
- Need for gateway

ADDRESSING VEHICLE NETWORK TRENDS

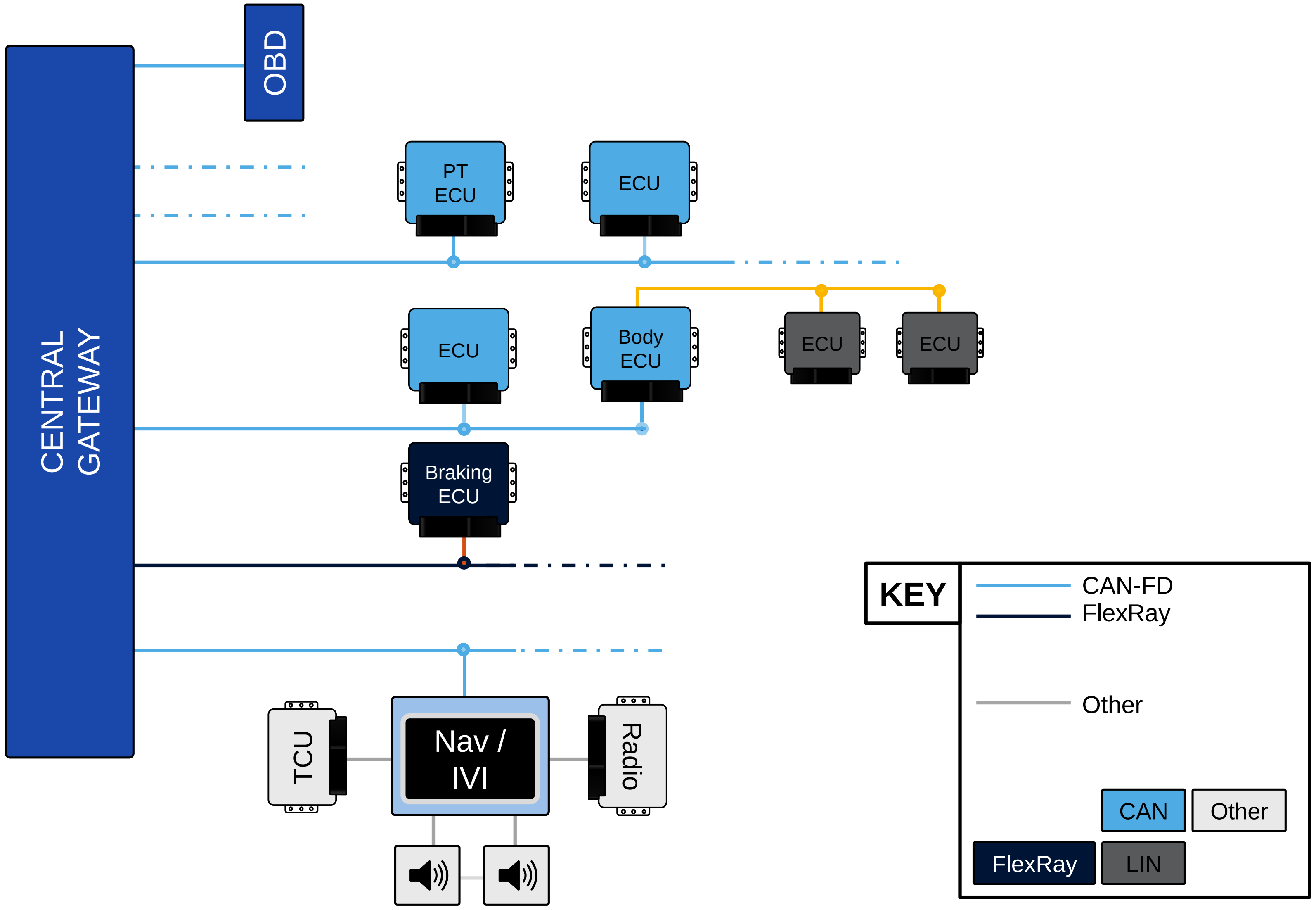
- ECU consolidation
- Managing multiple high-speed networks
- Ethernet connectivity (Gigabit Ethernet)
- Applications processing for new services
- Connected car and secure OTA updates
- Higher levels of security and safety

FUTURE VEHICLE NETWORKS

- Reduced ECUs and wiring complexity
- Evolving Ethernet (10-100-1000 Mbps+)
- Trusted OEM connectivity
- Service-Oriented architectures
- Higher security beyond EVITA Full
- Context-aware security
- ASIL B → ASIL D functional safety

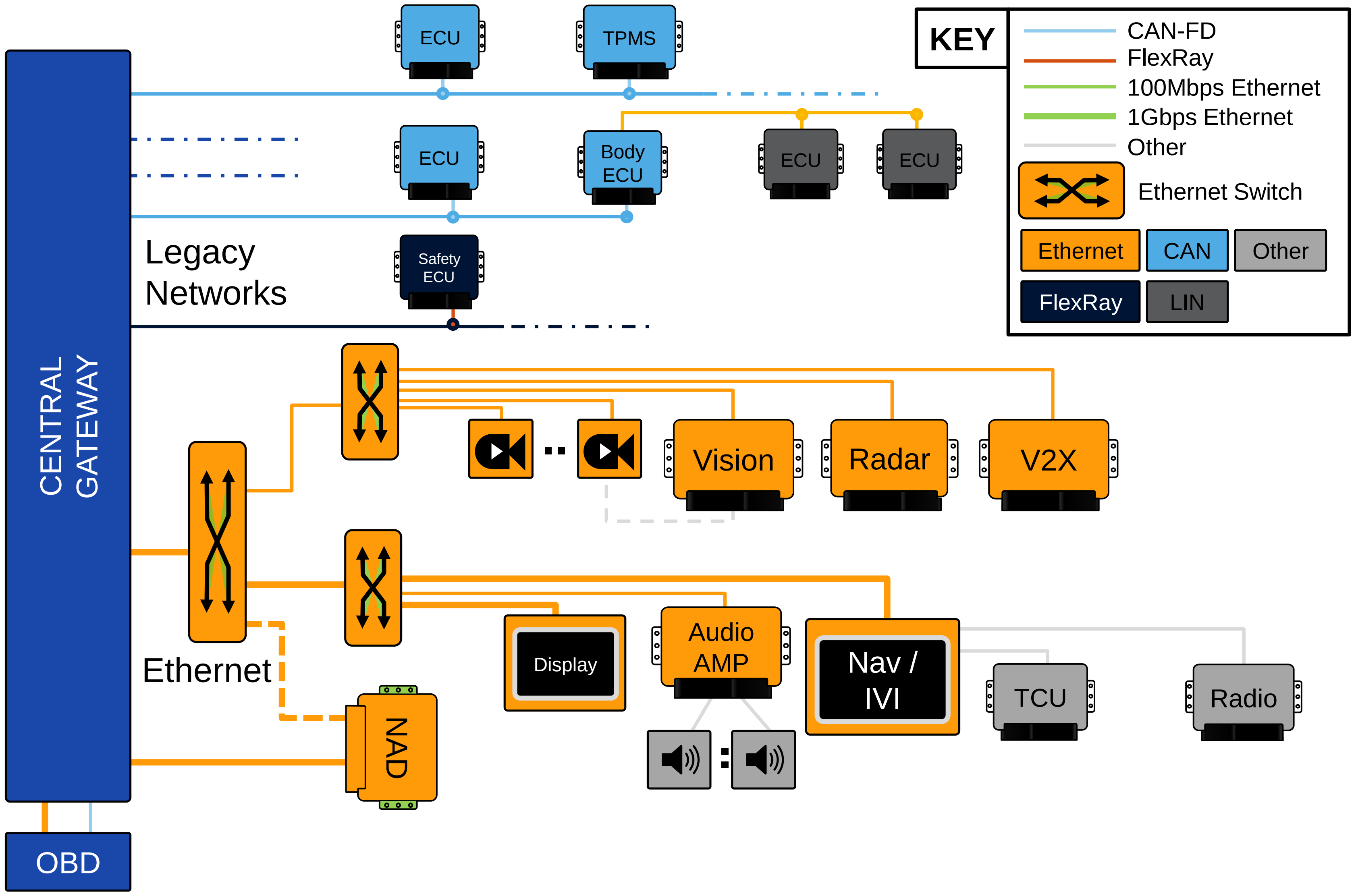
CAN CENTRAL GATEWAY ARCHITECTURE

- Legacy automotive networks
  - Typically 3-8 CAN networks
  - Typically 1-2 FlexRay networks
- Increased bandwidth
  - but, small compared to consumer / networking world
  - Proprietary protocols for higher bandwidth (e.g. MOST)
- Physical isolation
  - Functional domains
  - Safety / Non-safety
- Gateway role
  - Firewall internal traffic
  - Protocol translation



HYBRID ETHERNET ARCHITECTURE

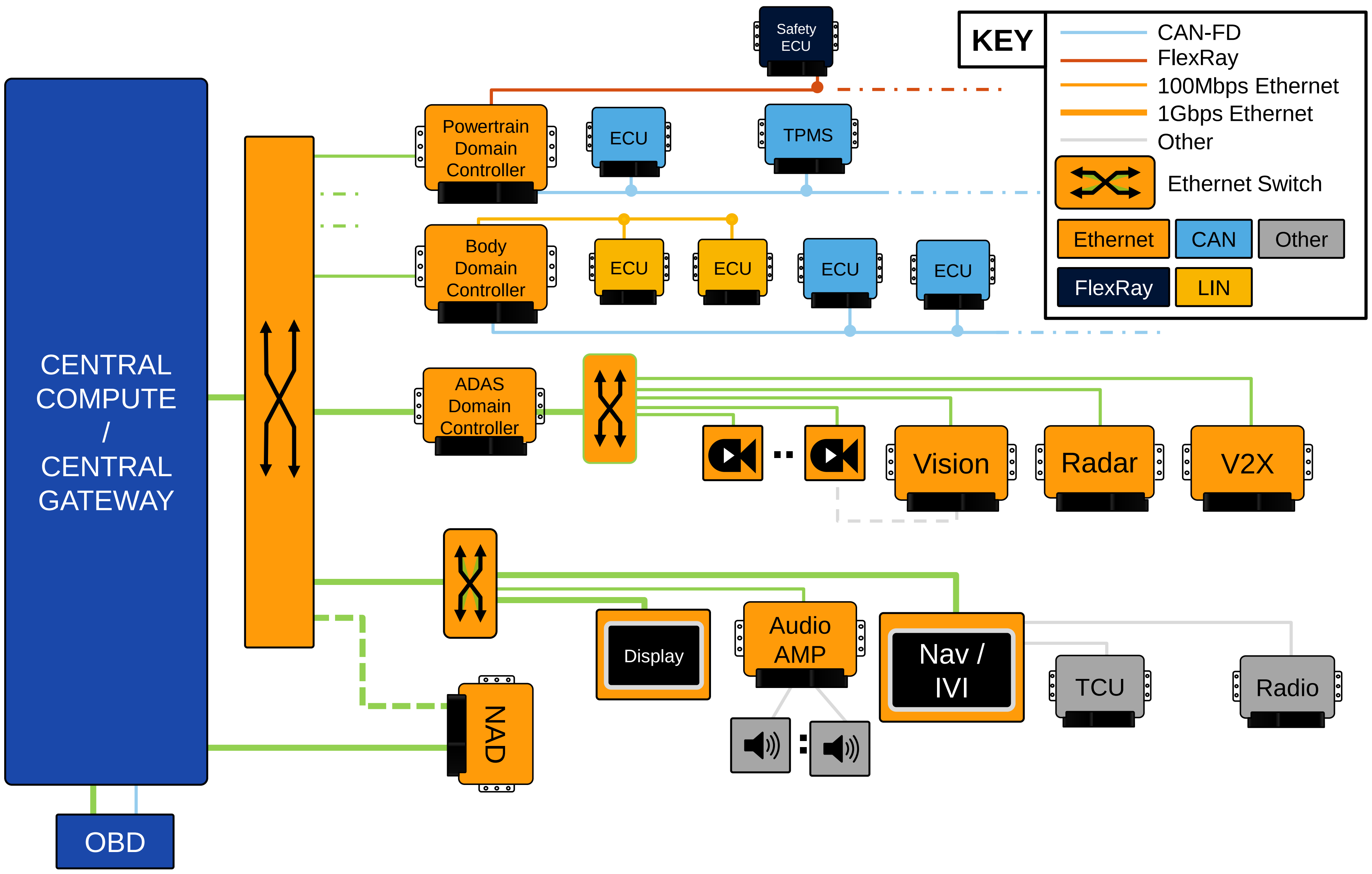
- Legacy + Ethernet networks
  - CAN, FlexRay & Ethernet
- High-bandwidth data
  - 100 Mbit →1 Gbit Ethernet
  - ADAS and infotainment drive higher data rates
  - Improved ECU program time in factory
- Gateway role
  - Firewall internal & external
  - Efficient protocol translation
  - ECU consolidation
  - New apps & services



ETHERNET BACKBONE WITH DOMAIN CONTROLLERS

- **Ethernet backbone with domain controllers**
  - ECU consolidation
  - Distributed gateway

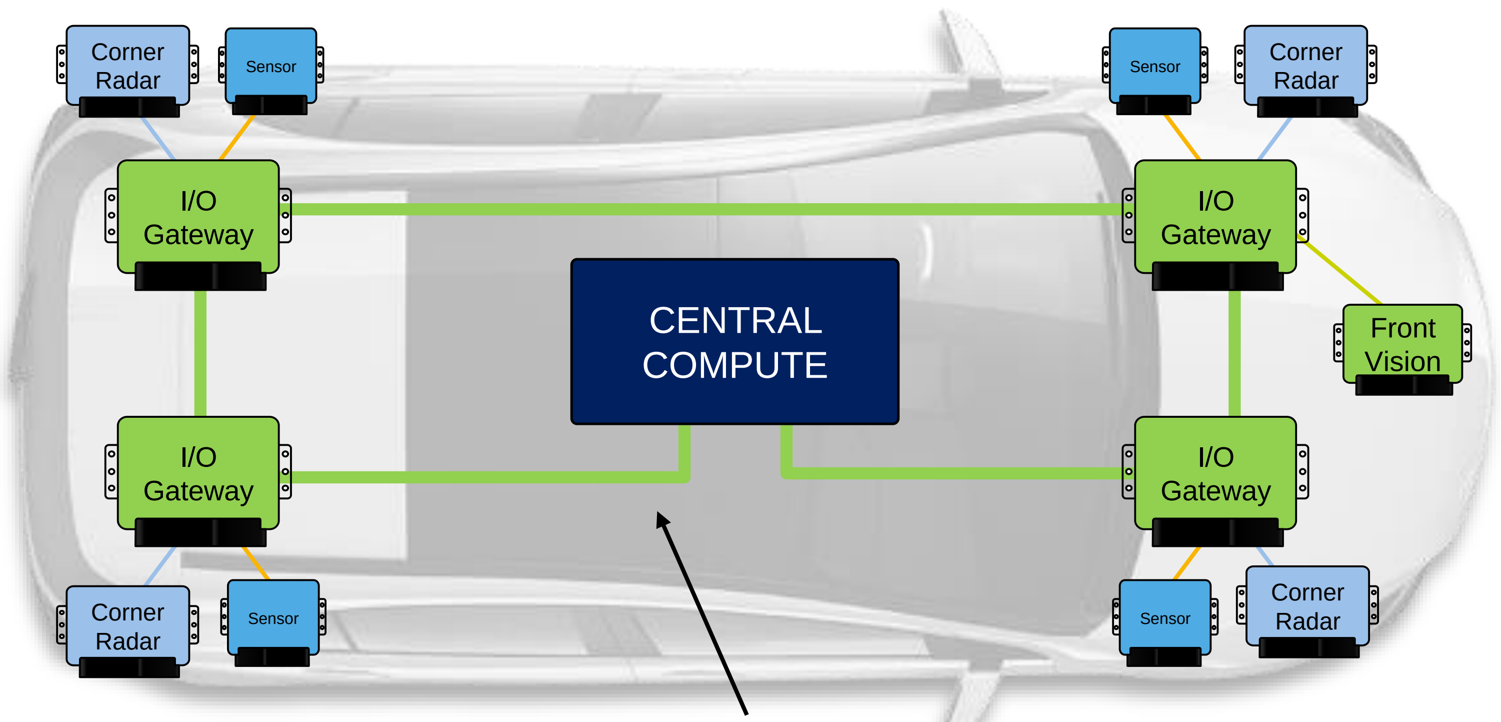
- Central compute
  - Strategy / decision making
  - Distributed vs centralized



ZONAL/CENTRAL COMPUTE ARCHITECTURE

- Central compute + I/O gateways
  - No functional domains
  - Strategy for vehicle fully owned by central compute
- I/O gateways connect Edge nodes to central compute
  - Distributed processing
  - Optimize network utilization
- Benefits:
  - Network architecture optimised to vehicle topology
  - Less wires (less weight, power, cost)

'Edge Nodes'
e.g. Radar ECU, Powertrain ECU



High speed Ethernet network (Mesh/Ring)
Optimised to vehicle topology for reduced wiring

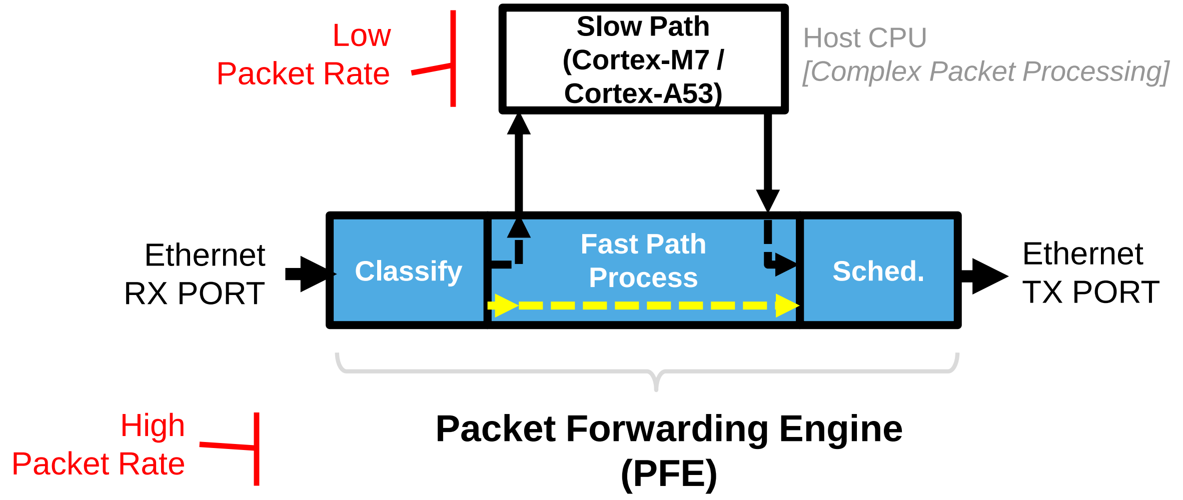
Packet Forwarding Engine (PFE) Overview

SECURE CONNECTIONS
FOR A SMARTER WORLD

ETHERNET PACKET FORWARDING ENGINE (PFE)

Ethernet bandwidth is growing

- 5G Cellular, more ADAS data, more ECUs to program
- Secure Gateway will firewall and route traffic
- Customer concerns: Higher performance needed and lack of Ethernet expertise. Value a SW-enabled, tested, efficient solution
- PFE: Packet Accelerator leveraged from NXP Layerscape products
  - Fully offloads Ethernet routing from host CPU



Value Propositions

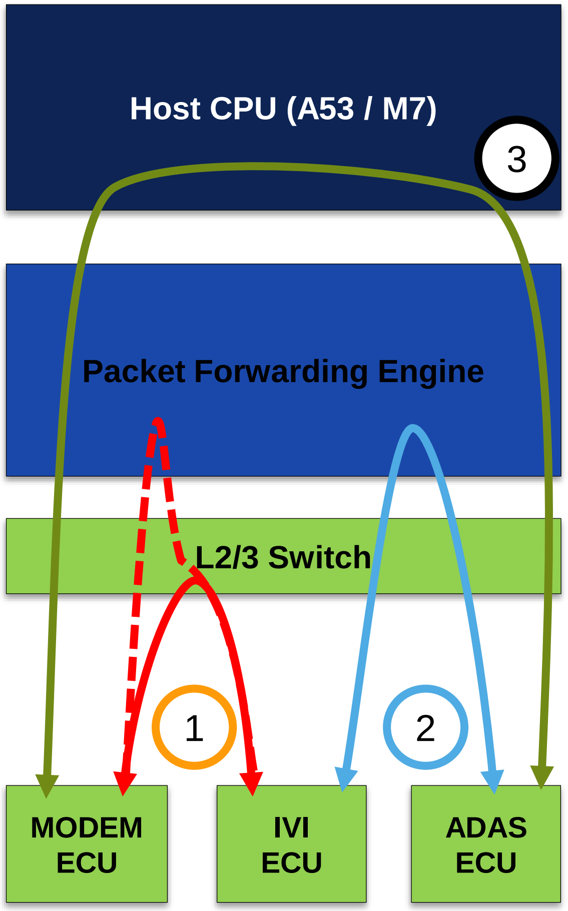
- **Proven solution (in NXP Digital Networking)**
- **Offloads host processor**
- **33% power dissipation of CPU-based alternative**

ETHERNET PROCESSING HIERARCHY

Deep Packet Inspection (e.g. IDPS)
Monitor payload of packets to detect unusual patterns

Flow Switching
208-byte header firewall (e.g. L2/3/4, DOIP, SOME/IP)
NAT Translation
IPSec Offload

L2/L3 Switching
Support for ports at line-rate



- Classic networking hierarchy
  - Multiple layers to handle different traffic types
  - Typ. depends on security requirements

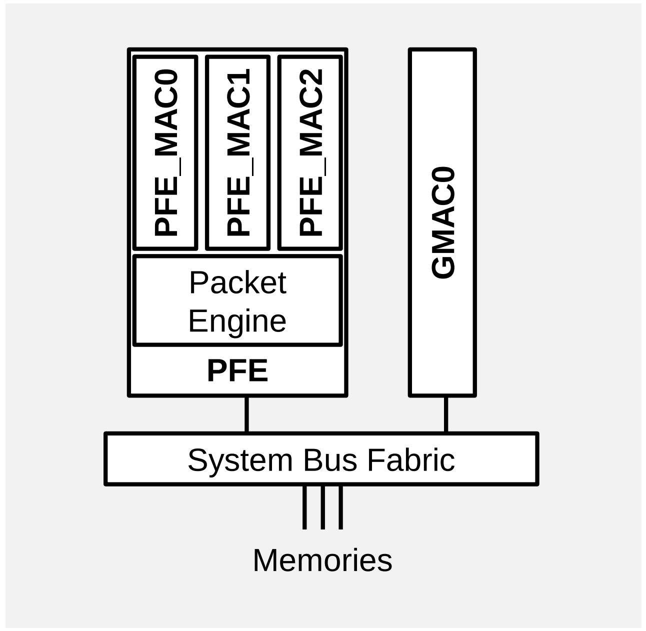
- Example flows:
  1. Modem-to-IVI: Both domains "untrusted", may allow direct connection with minimal firewalling (not impacting vehicle operation)
     - L2/3 Switch could support L2/3 switching
     - May need NAT translation, supported by PFE
  2. IVI-to-ADAS: Filter traffic based on specific SOME/IP service ID, may want to support authentication of traffic using IPSEC
     - SOME/IP filter & IPSec offload in PFE
  3. Modem-ADAS: May have valid paths, but want to monitor payload to detect if data is in 'unusual' state (i.e. rule based check, or machine learning check)
     - CPU flexibility & high performance required

S32G ETHERNET MUXING

- S32G supports 4X Ethernet MACs
  - 3X integrated with Packet Forwarding Engine (PFE)
  - 1X as standalone Ethernet Controller (GMAC)

- S32G 525FC-BGA supports:
  - All 4 MACs can be used in parallel
  - Max of 3X RGMII/RMII/MII* interfaces (mux options available for all MACs)
  - Max of 4X SGMII interfaces (PFE_MAC0/1/2, GMAC)

- PHY interface speed options
  - SGMII: 1Gbps and 100Mbps
    - PFE_MAC0 also supports 2.5Gbps
  - RGMII: 100Mbps and 1Gbps
  - MII/RMII: 10 and 100Mbps



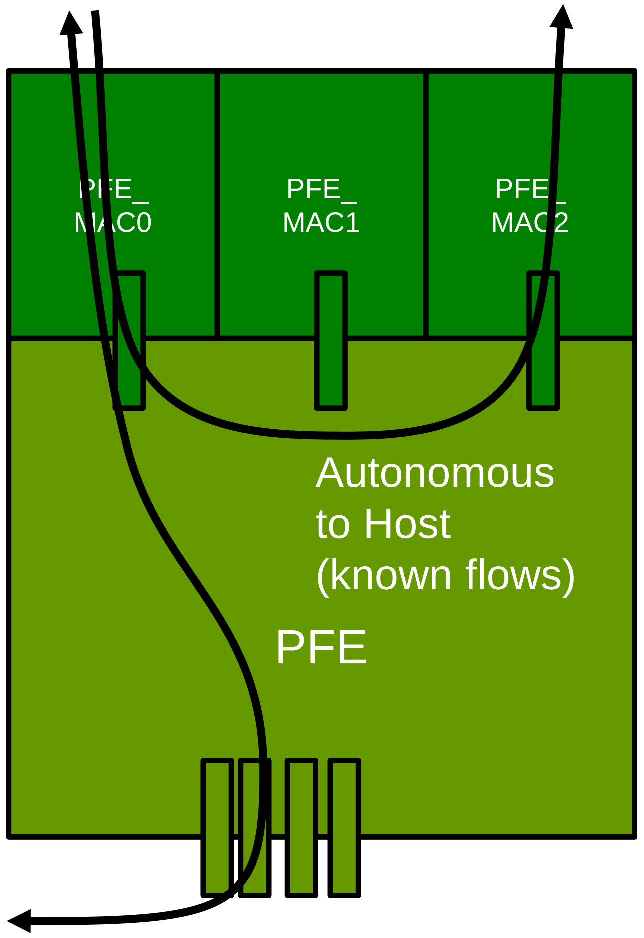
Each RGMII additionally supports RMII & MII
(*MII requires additional mux signals in LLCE)

I/O Supply	Nominal Voltage	Wakeup Support?	IO Count	Primary Function	Muxed Function	Muxed Function	Muxed Function	Muxed Function
VDD_IO_GMAC0	1.8V / 3.3V		14	GPIO	PFE_MAC1 (RGMII)	GMAC0 (RGMII)		
VDD_IO_GMAC1	1.8V / 3.3V	N	14	GPIO	PFE_MAC0 (RGMII)	PFE_MAC2 (RGMII)	LPSPI1	
							I2C2	
							I2C3	
							I2C1	
							I2C0	
VDD_IO_QSPI	1.8V	N	16	GPIO	QSPI (A-Side)			
VDD_IO_SDHC	1.8V / 3.3V	N	13	GPIO	USDHC (eMMC/SD/SDIO)	QSPI (B-Side)	I2C2	
							I2C3	
							I2C1	
							I2C0	
VDD_IO_USB	1.8V / 3.3V	N	14	GPIO	USB (ULPI)	PFE_MAC2 (RGMII)	DSPI2	FlexCAN2
								FlexCAN3
							LPSPI1	DSPI3
							FlexCAN1	
							FlexCAN0	LINFlex2

VDD_IO_PCIE0	1.8V	N	11	PCIE0 (X2) - mode #0	PCIE0 (X1) - mode #1	PCIE0 (X1) - mode #2	SGMII_GMAC0 - mode #3
					SGMII_GMAC0 - mode #1	SGMII_PFE_MAC2 - mode #2	SGMII_PFE_MAC2 - mode #3
VDD_IO_PCIE1	1.8V	N	11	PCIE1 (X2) - mode #0	PCIE1 (X1) - mode #1	PCIE1 (X1) - mode #2	SGMII_PFE_MAC0 - mode #3
					SGMII_PFE_MAC0 - mode #1	SGMII_PFE_MAC1 - mode #2	SGMII_PFE_MAC1 - mode #3

*Refer to S32G Reference Manual for latest IOMUX summary

PFE ARCHITECTURE



to/from Host: A53 or M7
(specific traffic only)

L2 Basics
- CRC, Basic HASHing
- L3/4 Checksum offload
- IEEE1588 / 802.1AS-Rev

PFE MACs

PFE Engine

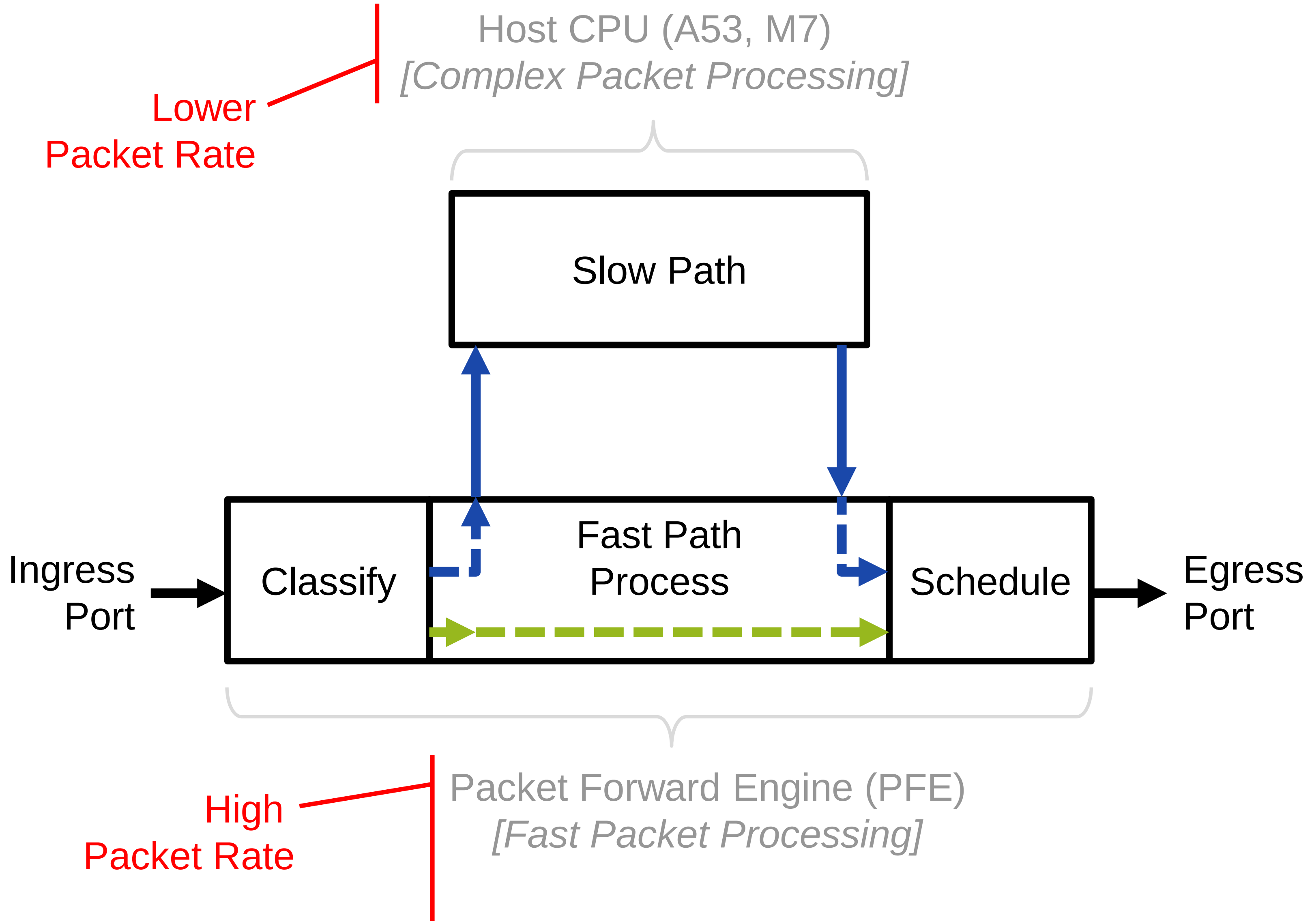
Classify (L2, L3, L4,…)
VLAN support
Ingress QoS
Modify (Headers: MAC, NAT, …)
Security Offload (e.g. IPSec)
Multi queue transmit
Multicast
Multi channel host interface

PFE Features

SECURE CONNECTIONS
FOR A SMARTER WORLD

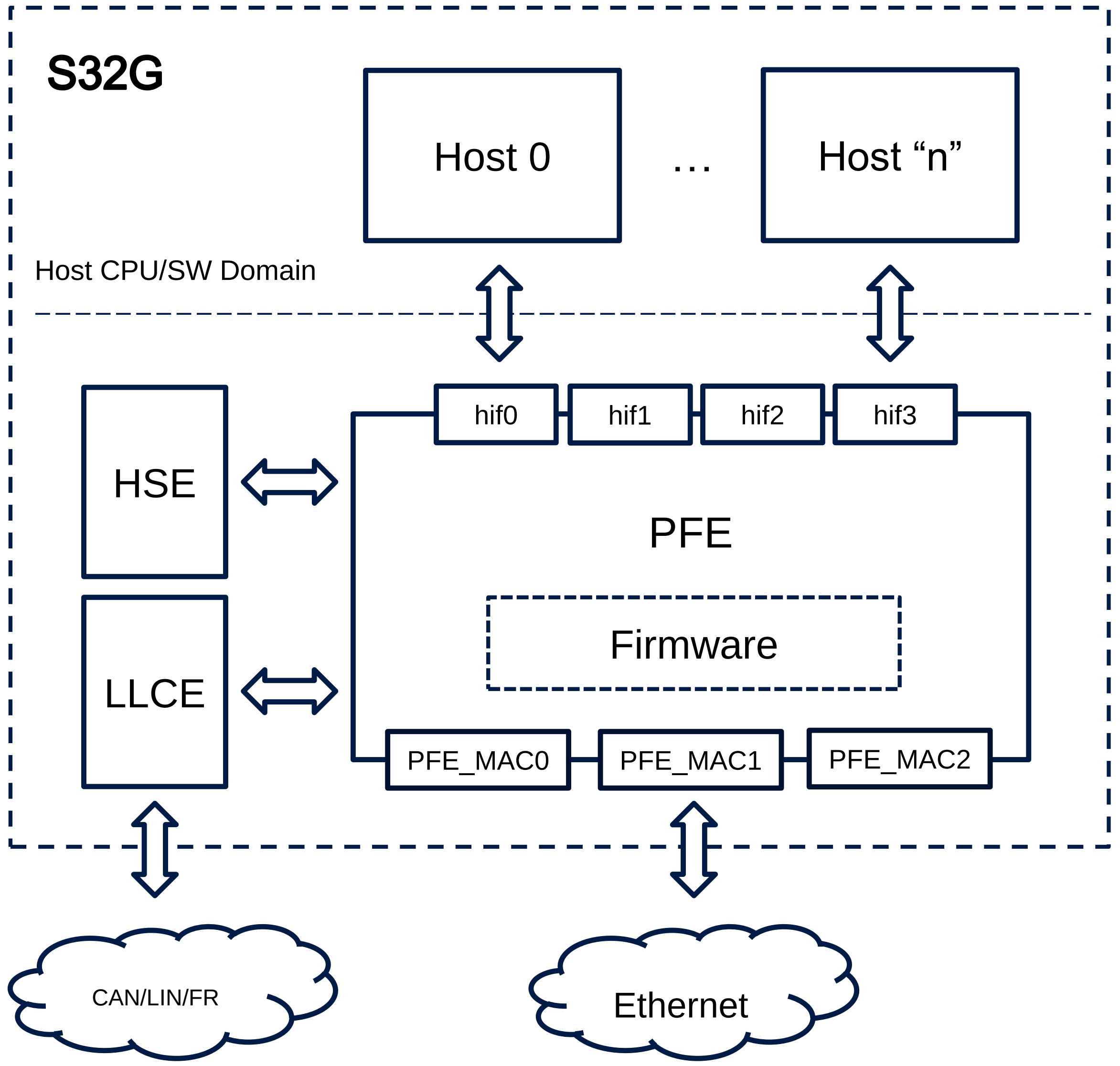
S32G ETHERNET IP ROUTING ENGINE

- Aim: Accelerate majority of routed packets, with minimal CPU load
  - Fast Path / Slow Path concept
  - Traffic heavily utilises Fast Path
- Slow Path
  - Host CPU
  - Control Path & Complex Packet Processing (e.g. DPI)
- Fast Path
  - Hardware acceleration
  - Heavily pipelined to maximise packet throughput



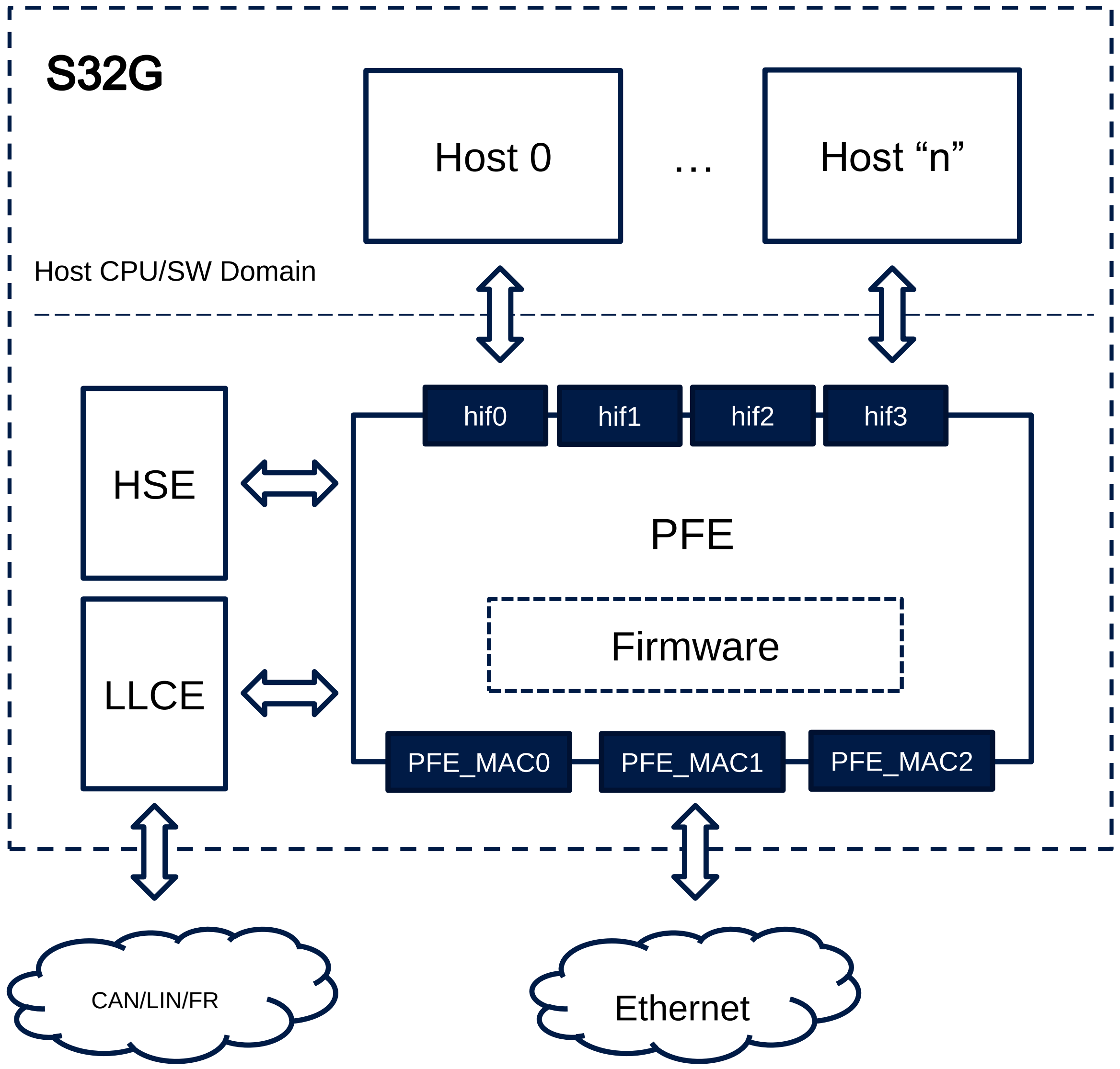
PFE OVERVIEW

- PFE is networking accelerator **executing NXP firmware**
- Access to host memory space and interaction with host CPUs (**IRQs**)
- Access to **security services** provided by HSE
- Available to LLCE for **bus bridging**



PFE INTERFACES

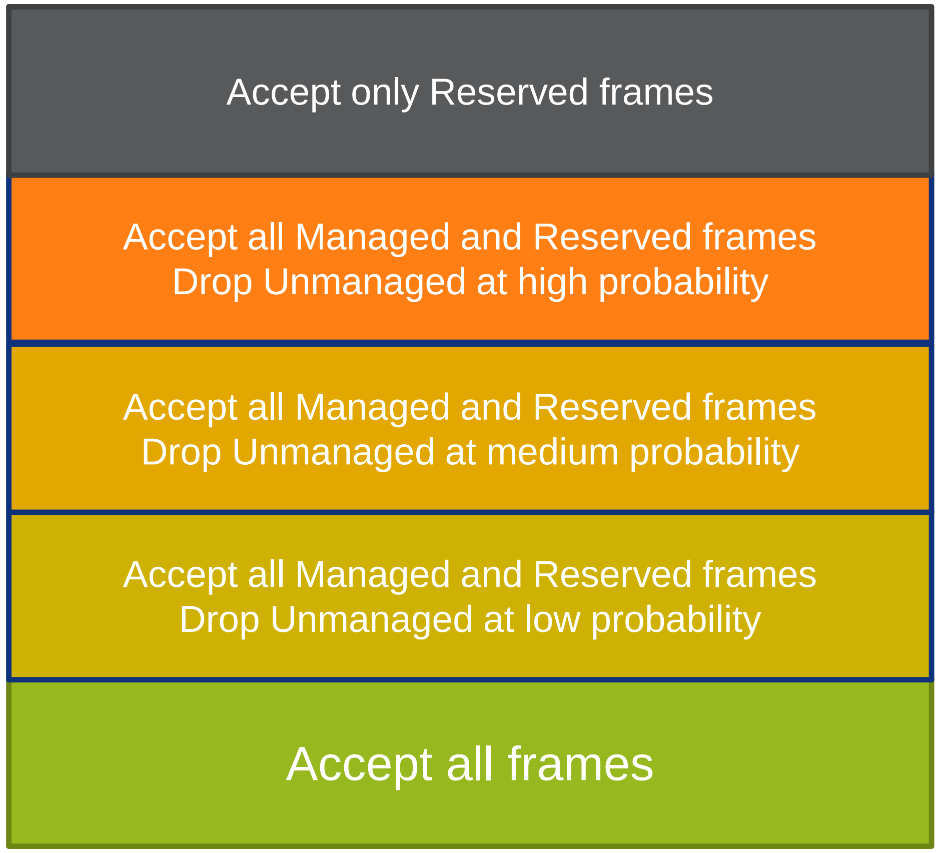
- Total of **7** available data interfaces
  - **4x host**
    - Independent data interfaces to host CPUs
    - **Register isolation** via XRDC
    - **Coherency** support (A53)
  - **3x PFE_MAC**
    - Data interface to external world
    - SGMII: 1Gbps, 100Mbps, PFE_MAC0 also 2.5Gbps
    - RGMII: 100Mbps + 1Gbps
    - MII/RMII: 10Mbps + 100Mbps
- Ability to classify, modify, prioritize, and distribute (**not only Ethernet**) traffic across all interfaces without host CPU intervention.



INGRESS QOS MANAGEMENT

Ingress QoS

- Congestion avoidance and malicious flows termination
- Ingress traffic prioritization
- Can be used to offload some IDPS-triggered actions (drops, rate limitation, …)
- **Credit-based shaper**
  - Allows limiting ingress data/packet rate per PFE interface
- **WRED**
  - Ability to prioritize ingress traffic using **table of rules**. In case of PFE congestion the WRED starts dropping less significant traffic with probability increasing as the congestion level gets more severe.


Accept only Reserved frames
Accept all Managed and Reserved frames Drop Unmanaged at high probability
Accept all Managed and Reserved frames Drop Unmanaged at medium probability
Accept all Managed and Reserved frames Drop Unmanaged at low probability
Accept all frames

EGRESS SCHEDULING/SHAPING

- Traffic management unit is passed packets after classification/modification
- It computes next packet to be transmitted
- Each Ethernet port has:
  - 8 queues
  - Schedulers: PQ, WRR, RR
  - Credit based shapers
- Flexible configuration of shapers/schedulers by host software
- Shaping at queue or port level
- Supports multicast of traffic

PFE FOR IDPS

- PFE offers support to aid DoS attack prevention
  - Capable of recognizing several upper layer protocol fields and taking additional actions in order to support prevention of DoS attacks
  - Actions for the packet filters can be:
    - Discard the packet
    - Forward the packet normally
    - Send the packet to the host CPU for processing
    - Send the packet to the host CPU and also forward it normally

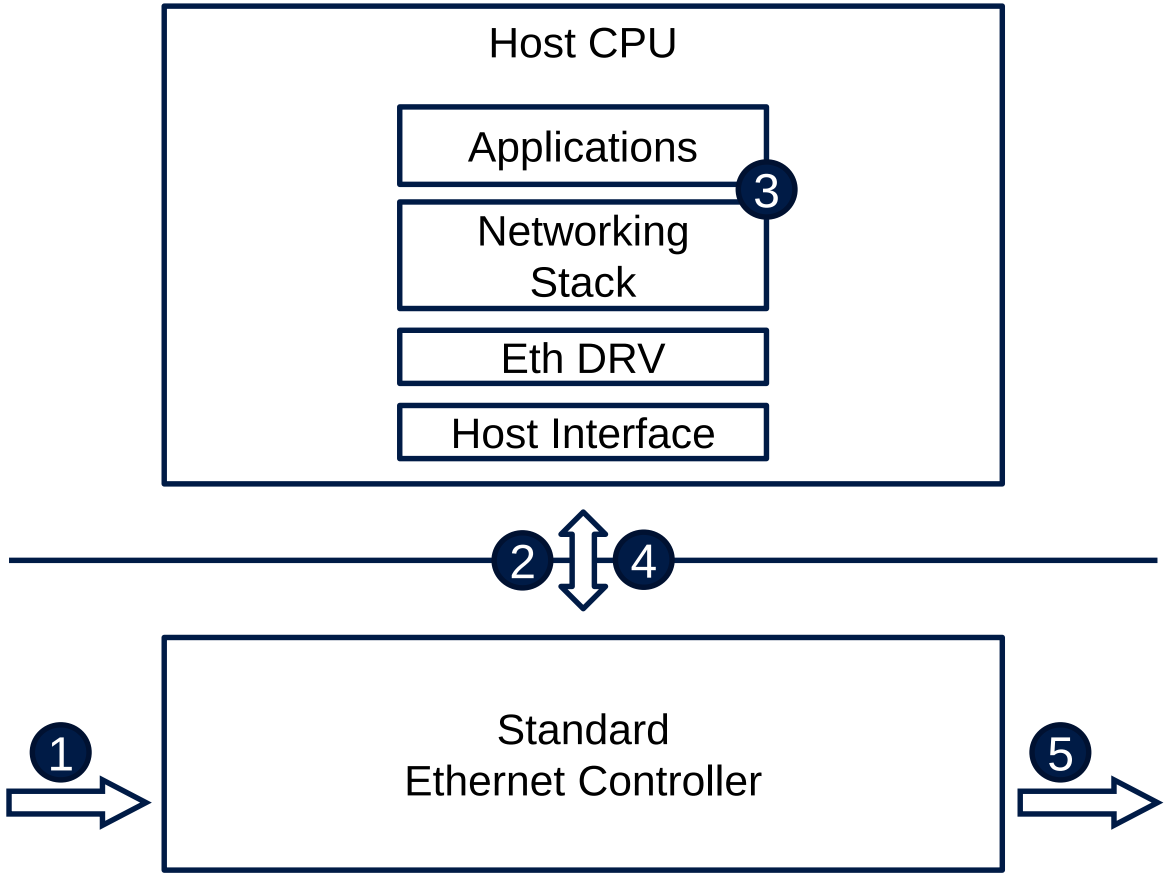
- PFE firmware extensions are also possible to add additional monitors including counter based violations, e.g. detect >100 SYN messages per second
  - For more details on PFE firmware extension requests please contact your NXP representative

PFE Flow

SECURE CONNECTIONS
FOR A SMARTER WORLD

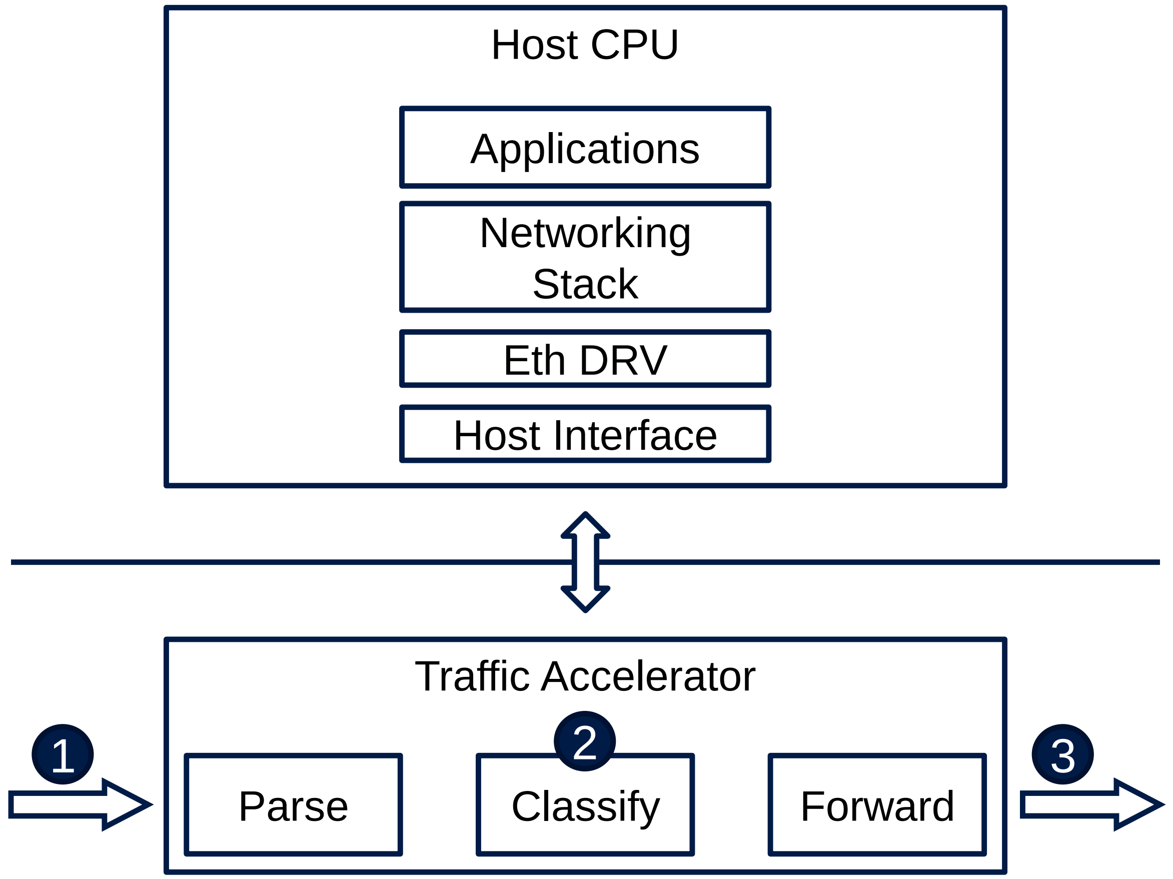
STANDARD APPROACH (SLOW-PATH)

1. Traffic is received
2. Traffic is sent to host CPU for processing
3. Host runs some classification algorithm and decides that given flow shall be forwarded to another node
4. Traffic is modified and sent back to the Ethernet controller to be forwarded
5. Traffic is forwarded



ACCELERATED APPROACH (FAST-PATH)

1. Traffic is received
2. PFE runs local classification algorithm and decides what to do with given flow
3. Known traffic is processed and forwarded directly via target interface without host CPU intervention



Example Use Cases

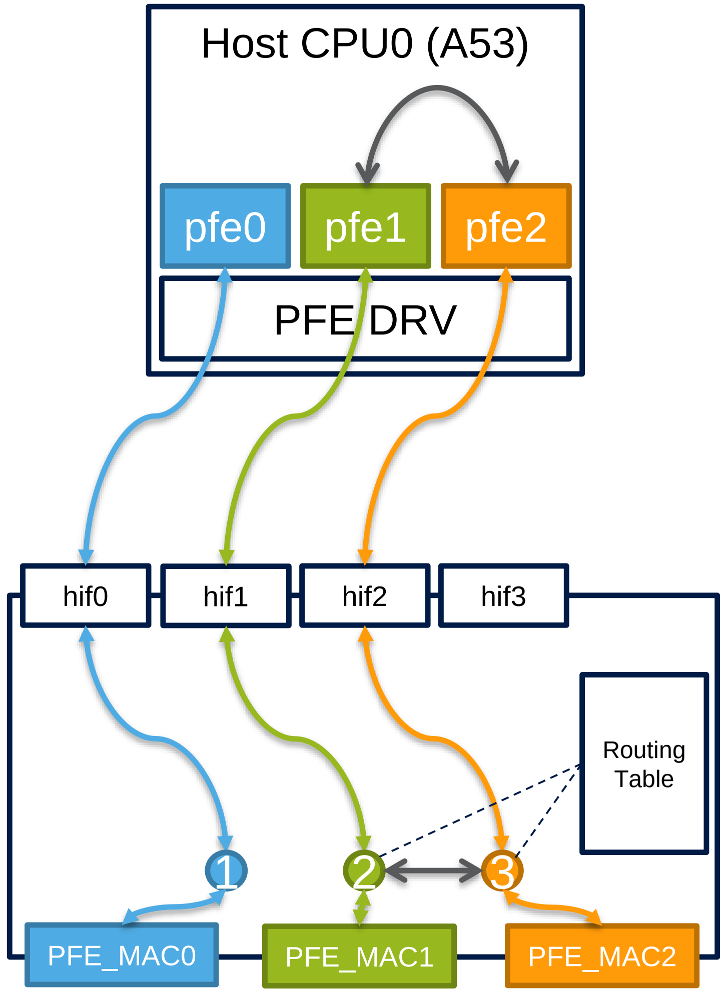
SECURE CONNECTIONS
FOR A SMARTER WORLD

IPV4 / IPV6 ROUTER

Host application can change operation mode of a physical interface to IP Router and specify routing rules into **routing table** using FCI API.

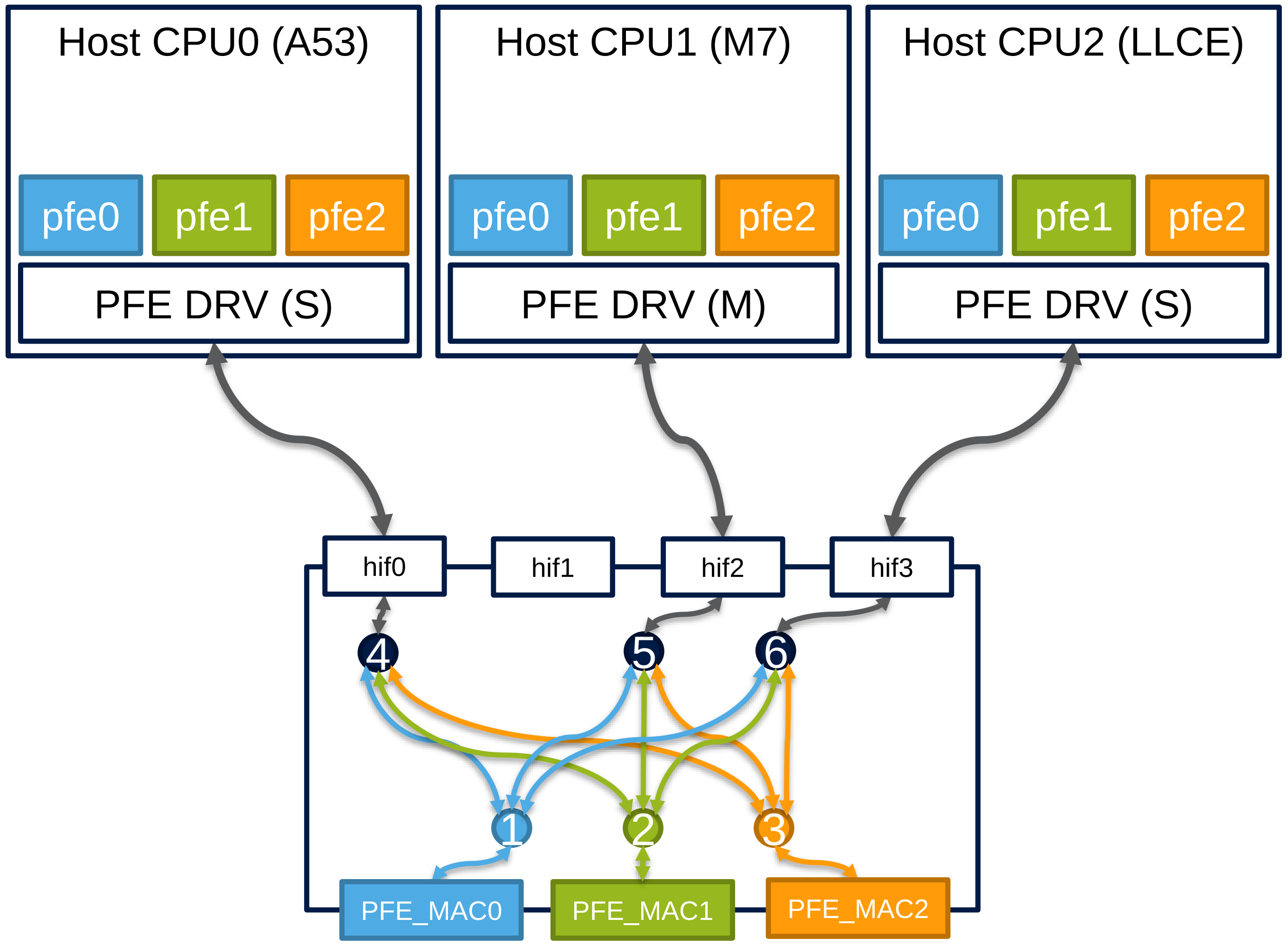
Example:

- Fast-path routing between PFE_MAC1 and PFE_MAC2 and slow-path routing within host CPU
- Once configured, the routing points ❷ and ❸ forward traffic received via PFE_MAC1 or PFE_MAC2, which matches a routing table entry, to desired destination interface(s) (PFE_MAC2 or PFE_MAC1) while the rest of traffic is sent to default interface (CPU0) where the host software can perform custom, slow-path routing



DATAPATH ENDPOINT

- Three hosts can access PFE_MACs via respective logical interface (pfe0-2)
- CPU1 is running Main driver while CPU0 and CPU2 are running secondary drivers
- ➊, ➋, and ➌ are configurable routing points where PFE decides which interface the traffic received from PFE_MACn shall be forwarded to
- ❹, ❺, ❻ are configurable routing points to dispatch traffic received via HIF to respective PFE_MAC

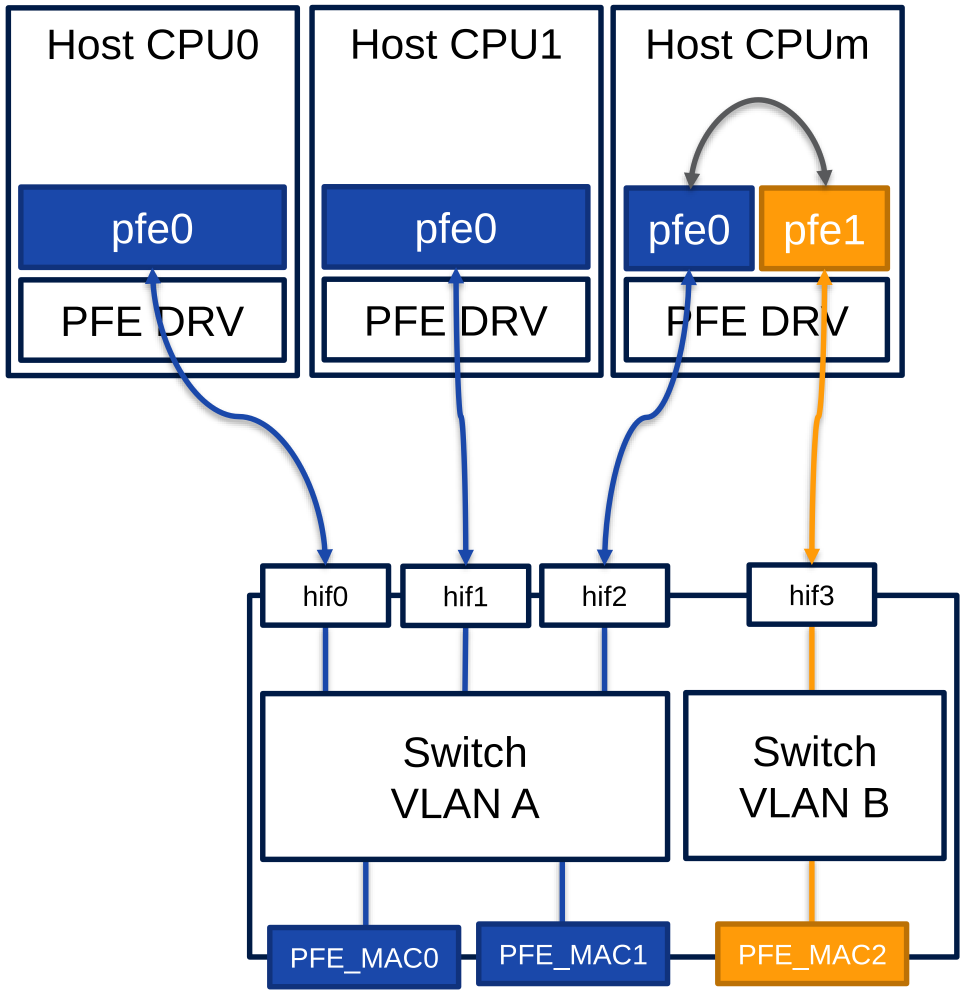


VLAN-AWARE L2 BRIDGE

- Ethernet switch
- Interfaces can be assigned to a bridge domain using VLAN
- Traffic is being routed using MAC addresses
- MAC addresses can be learned or statically assigned

Example:

- 2 bridge domains A and B
- A: CPU0, CPU1, CPUm, PFE_MAC0, PFE_MAC1
- B: CPUm, PFE_MAC2, CPUn
- Traffic in domain A is never visible to domain B but CPUm can act as a cross-domain router and forward traffic between domains (if security rules allow that…)

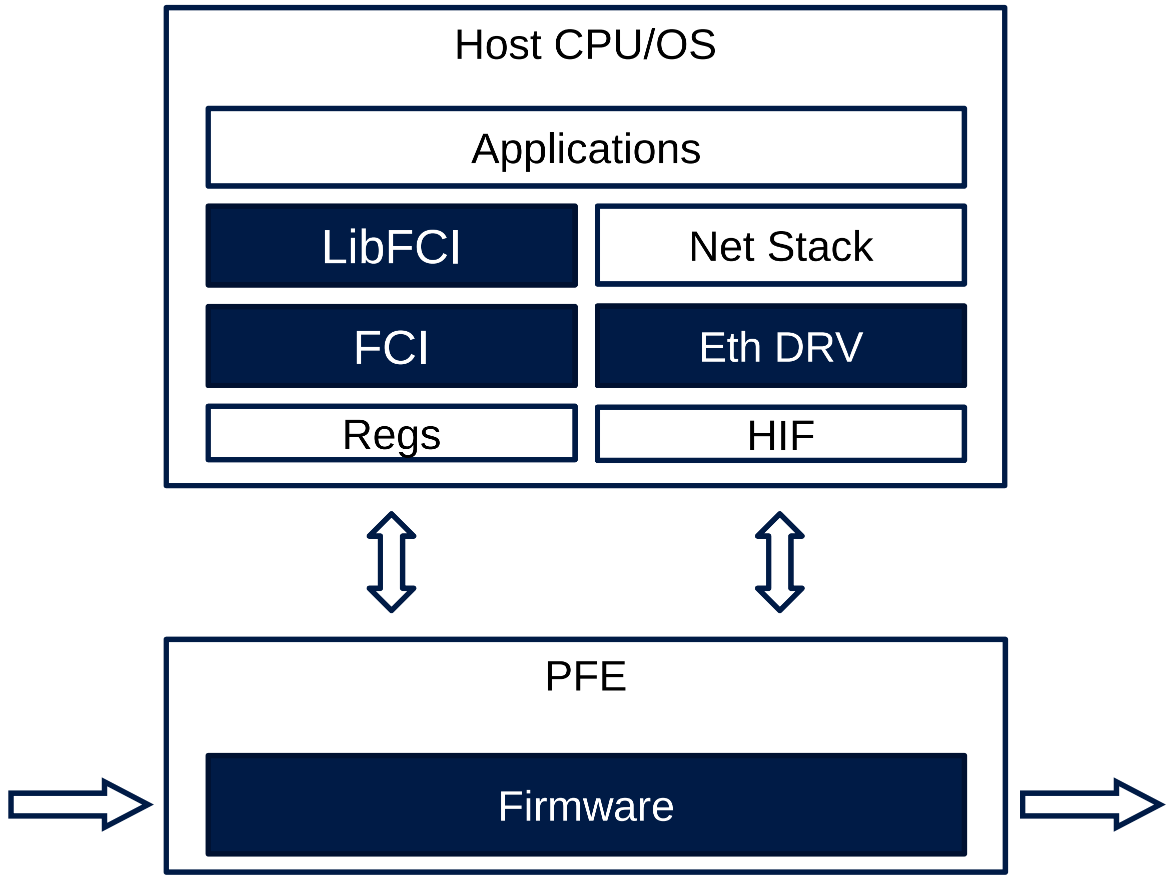


PFE Software Stack

SECURE CONNECTIONS
FOR A SMARTER WORLD

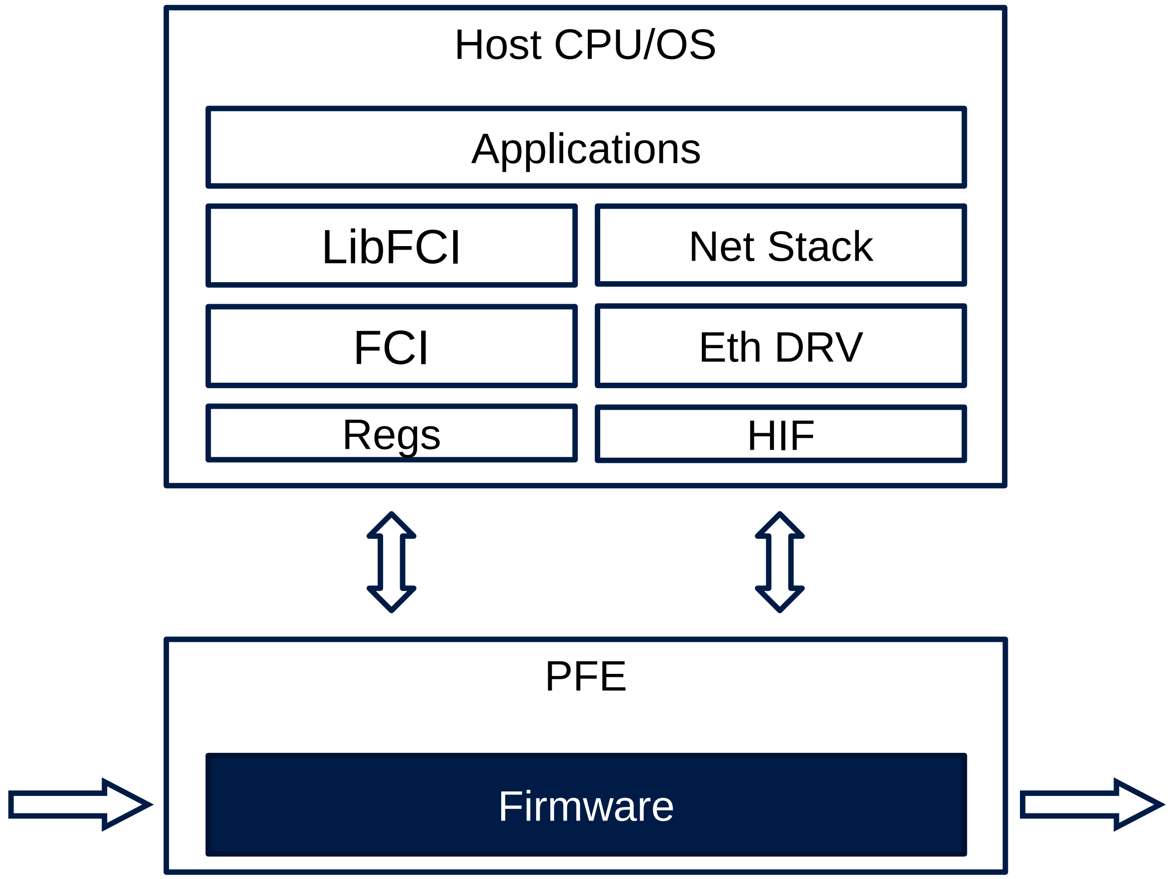
S32G NETWORKING STACK

- PFE Firmware
- Ethernet Driver
- FCI (Fast Control Interface)
  - Endpoint
  - Library



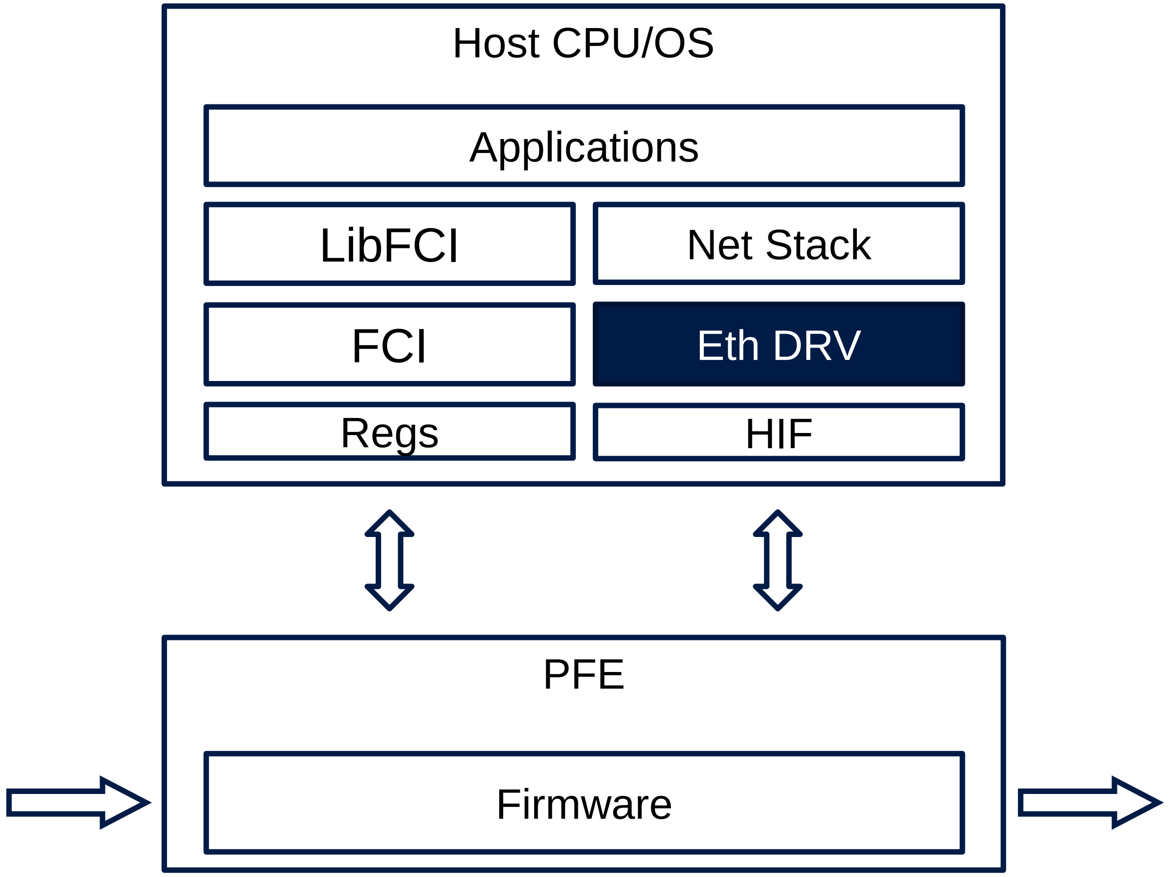
S32G NETWORKING STACK : FIRMWARE

- Runs on processing engines within the PFE
- Implements soft PFE features
- Main FW for packet classification, routing, modification, and distribution
- Auxiliary FW for IPsec offload and other utility tasks
- Delivered in form of binary files
- Firmware is under full control of NXP and can be modified via software services



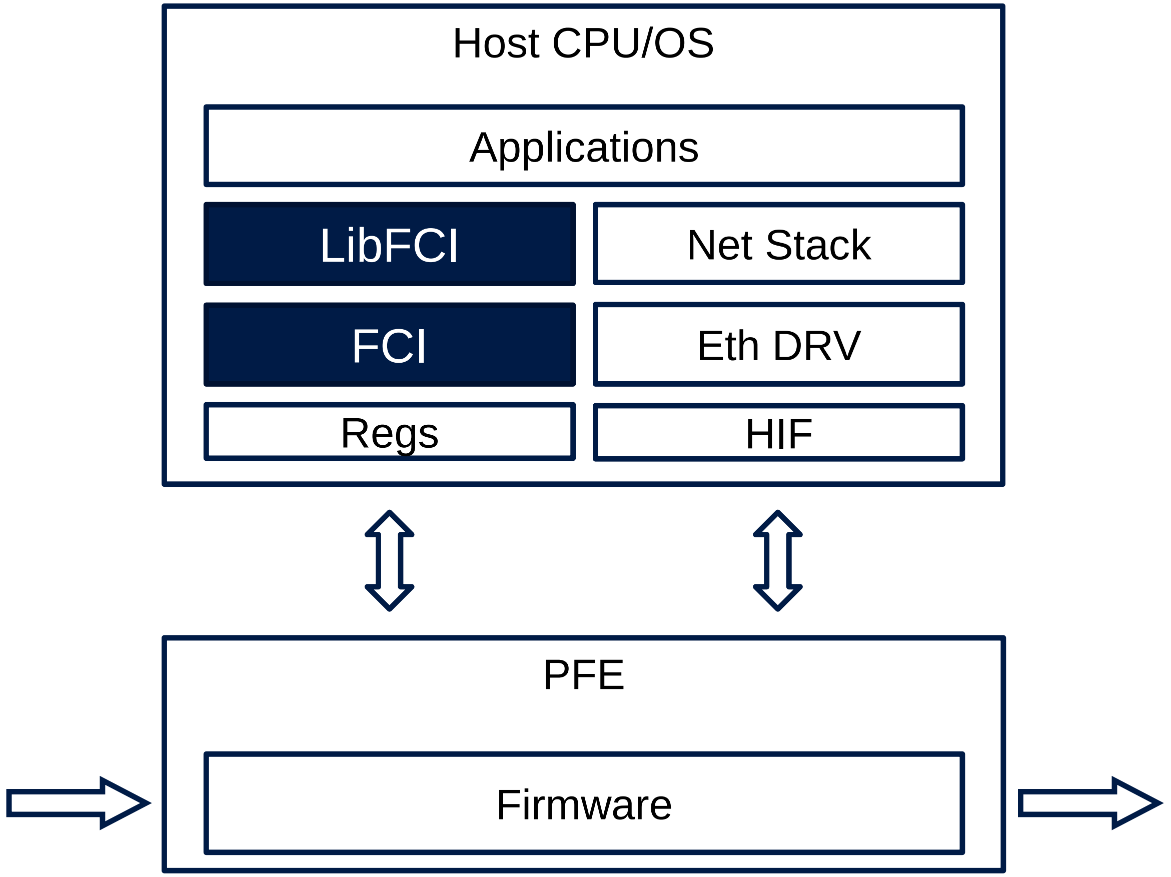
S32G NETWORKING STACK : ETHERNET DRIVER

- Provides data-path functionality, connection with networking stack, and exposes logical interfaces to the host system:
  - **Generic** low-level PFE platform driver
    - HW bring-up, firmware upload
    - Monitoring and management
    - HIF driver
  - **OS-specific** part to support portability
    - Connection to networking stack
    - OS Abstraction Layer
    - OS-specific features
- Supports multi-instance deployment over multiple different CPUs/VMs

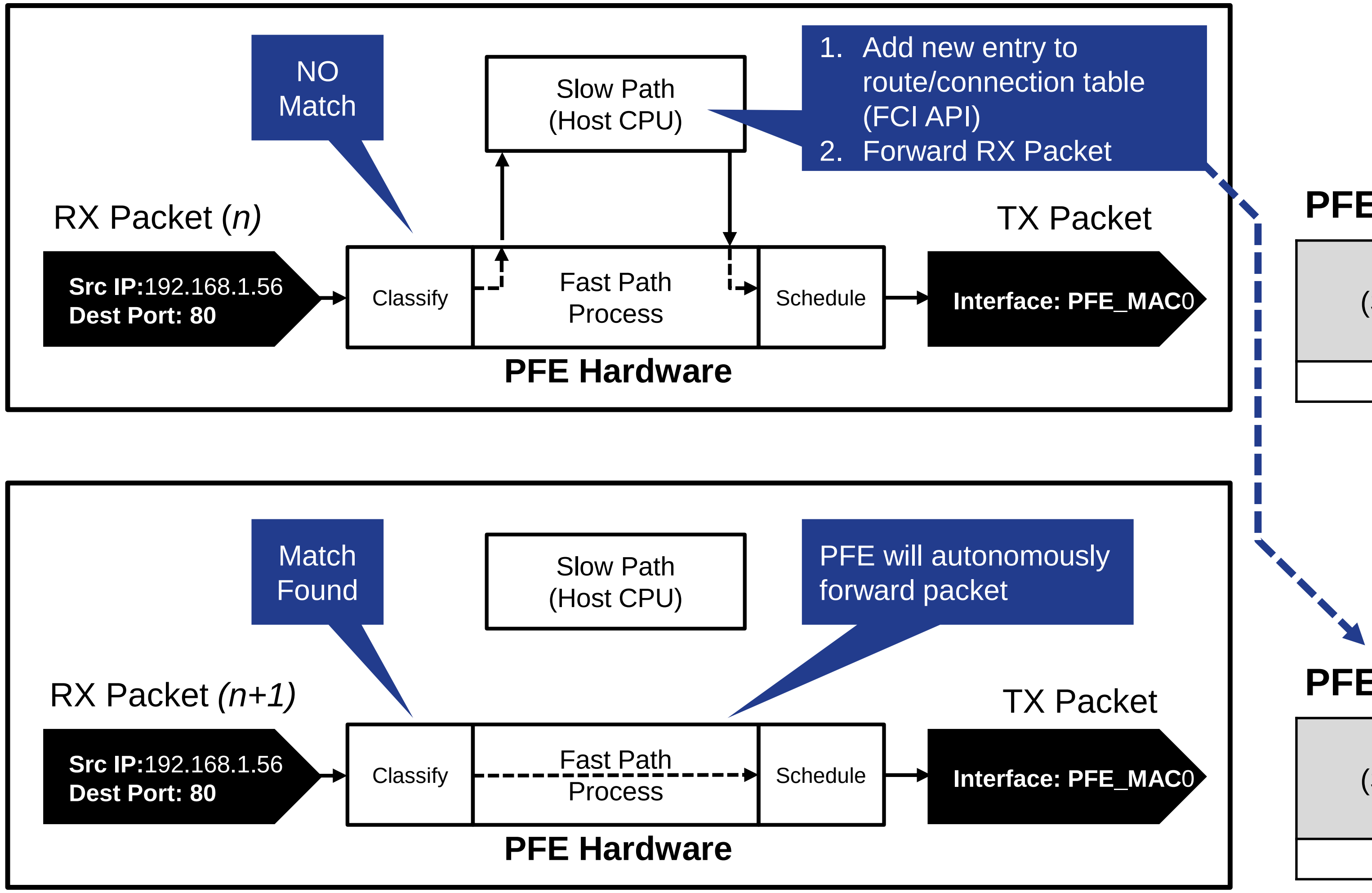


S32G NETWORKING STACK : FCI - FAST CONTROL INTERFACE

- FCI together with LibFCI provides **control interface** between the HW/FW and user's applications
- **IOCTL-like API** provided by FCI library
- Allows top-level applications to **configure the PFE** as well as receive events in opposite direction
- Examples:
  - Assign interfaces to L2 bridge domain, manage static entries, manage VLANs, control learning
  - Manage and monitor L3 routing table entries
  - Configure traffic-to-HIF distribution
  - Configure physical/logical interface properties
  - Configure classification algorithms
  - Configure ingress/egress QoS
  - Get statistics
  - …
- Detailed description of FCI API with usage examples can be found within the ***FCI API Reference*** document delivered within driver release packages



EXAMPLE: FAST PATH IN ACTION



PFE Conntrack table

MATCH Criteria (Src IP Addr/Port, Dest IP Addr/Port, Protocol)	Route ID
EMPTY	EMPTY

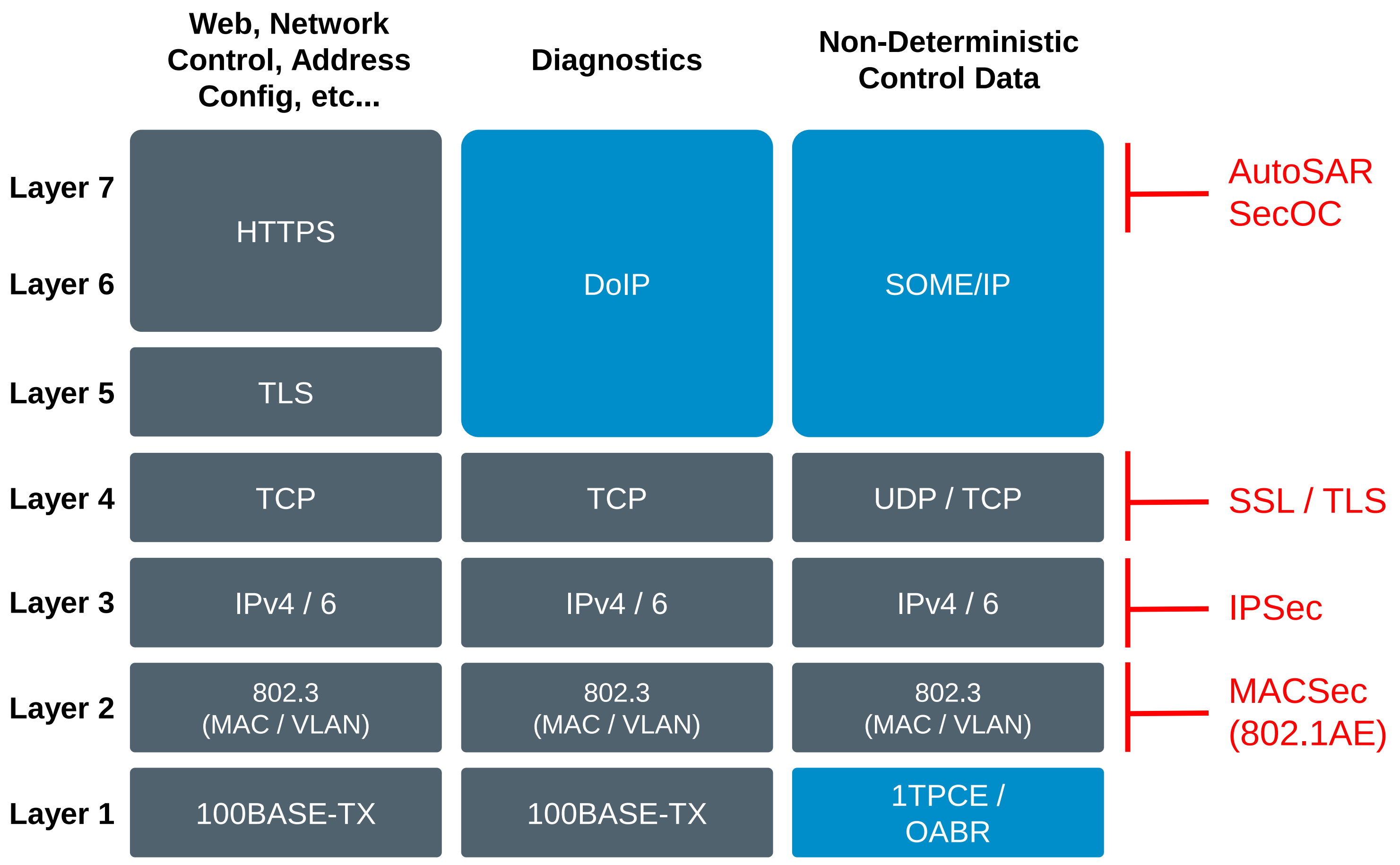
PFE Conntrack table

MATCH Criteria (Src IP Addr/Port, Dest IP Addr/Port, Protocol)	Route ID
SADDR: 192.168.1.56, DPORT: 80	999

Ethernet Security

SECURE CONNECTIONS
FOR A SMARTER WORLD

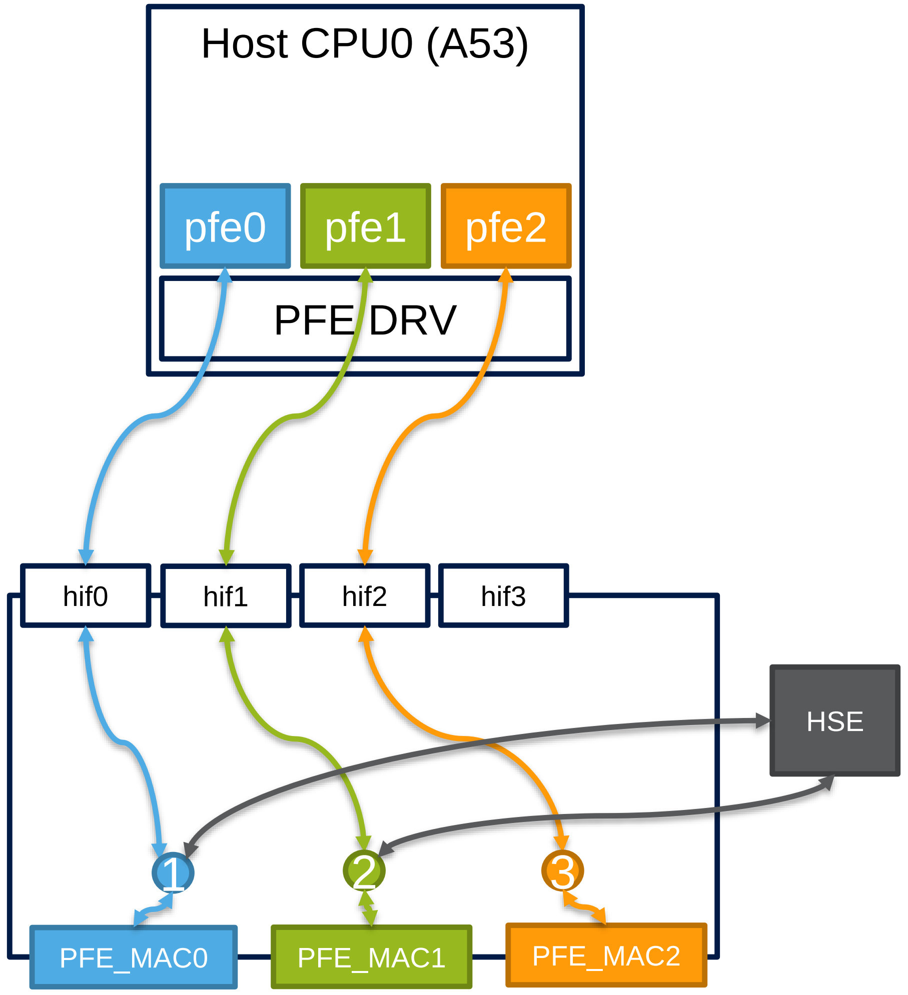
AUTHENTICATION / ENCRYPTION OVER ETHERNET



- Several possibilities for security over Ethernet
- External traffic
  - e.g. to Internet/OEM
  - Internet stds: TLS, IPSec VPN
- Internal traffic
  - Required: Auth + Integrity
  - Optional: Encryption
  - No industry consensus on which layer to protect
  - Balance cost vs protection
- PFE supports IPSec offload to reduce host core utilization
  - Interfaces directly to HSE
  - PFE can pre-filter other secure protocol streams to a specific HIF channel to be handled by a specific host core

Example: IPsec ESP tunnel between two interfaces

- PFE_MAC0 is “non-secure” and PFE_MAC1 “secure” interface
- **NonSecure-to-Secure Port:** Traffic received via PFE_MAC0 is classified by routing point ➊ and sent to HSE for encapsulation/encryption if is liable to IPsec processing. Then it is forwarded via PFE_MAC1 as IPsec traffic. Rest of packets are sent to host for normal processing.
- **Secure-to-NonSecure Port:** In opposite direction the routing point ➋ selects traffic liable to IPsec processing and sends it to HSE for decryption/decapsulation. Plain packets are then forwarded via PFE_MAC0.



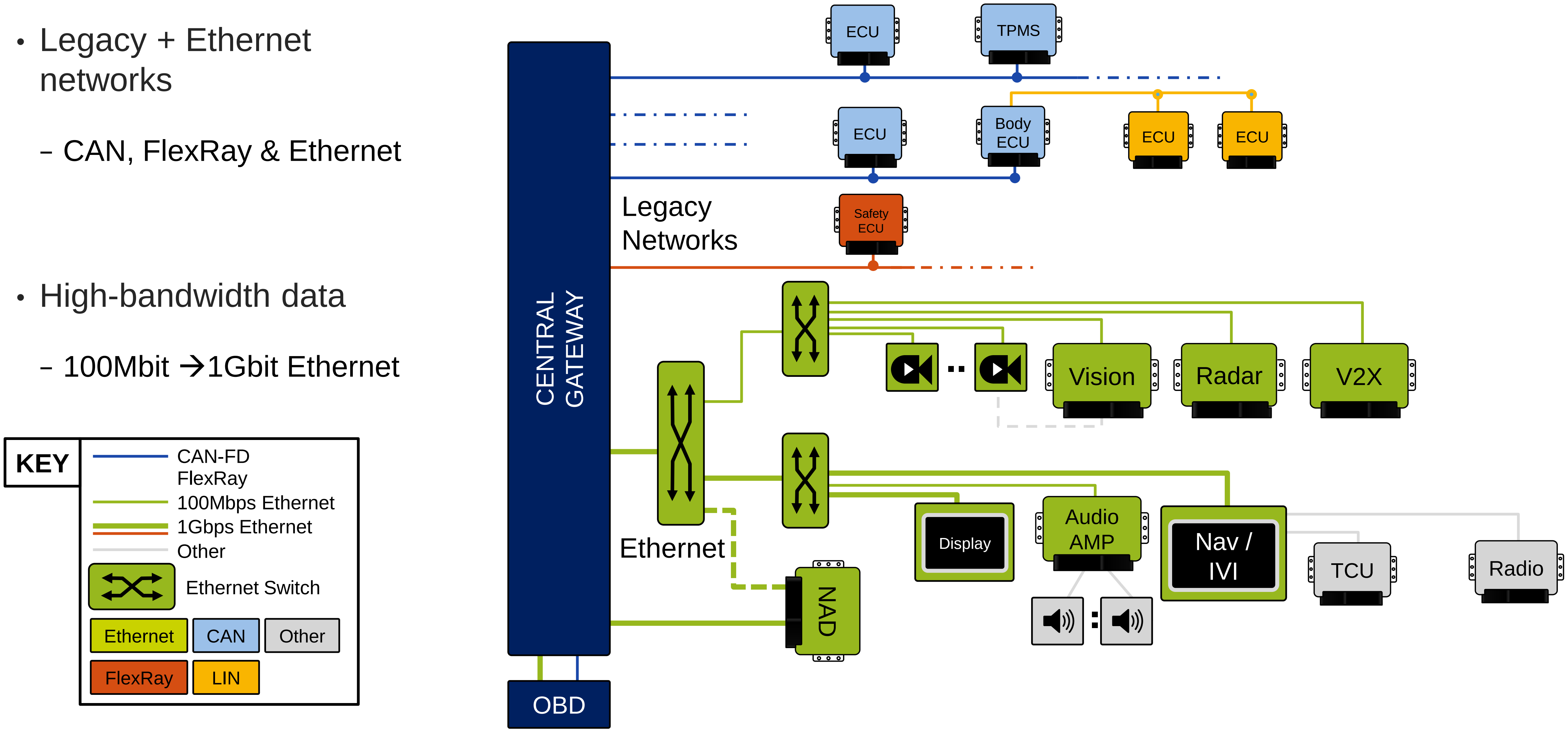
LLCE Background

SECURE CONNECTIONS
FOR A SMARTER WORLD

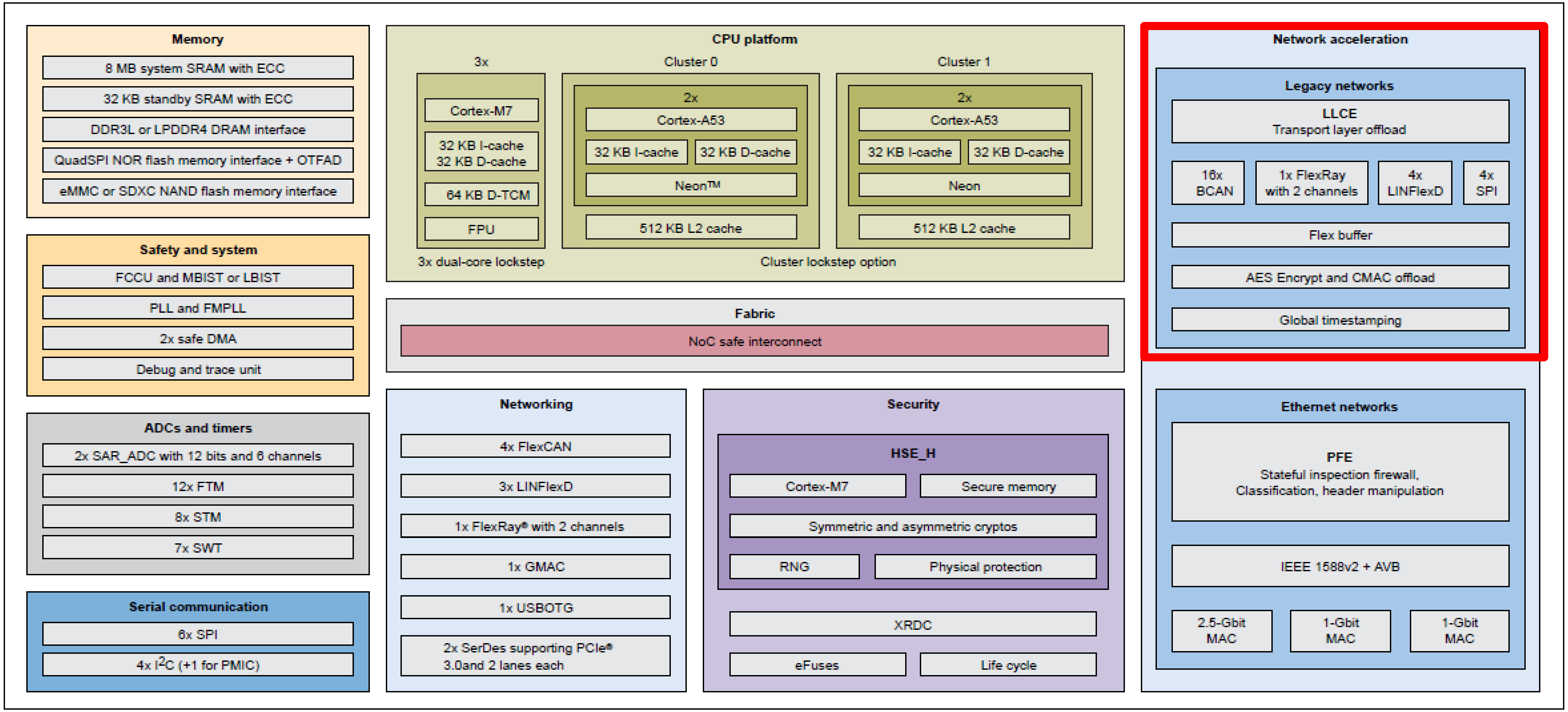
GATEWAY ARCHITECTURE

- Legacy + Ethernet networks
  - CAN, FlexRay & Ethernet

- High-bandwidth data
  - 100Mbit →1Gbit Ethernet



S32G274A: VEHICLE NETWORK PROCESSOR



Introduction To Low Latency Communication Engine

SECURE CONNECTIONS
FOR A SMARTER WORLD

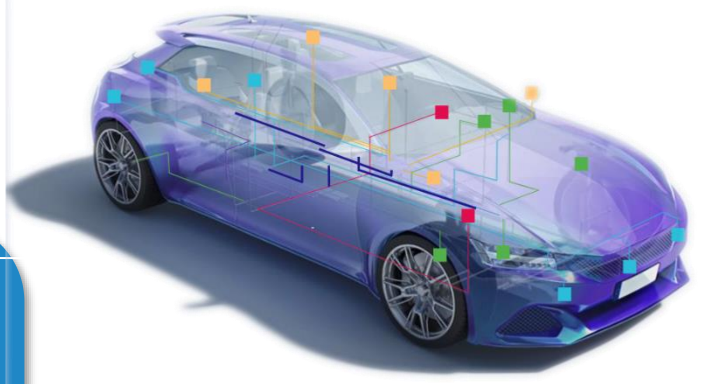
LLCE - LOW LATENCY COMMUNICATION ENGINE

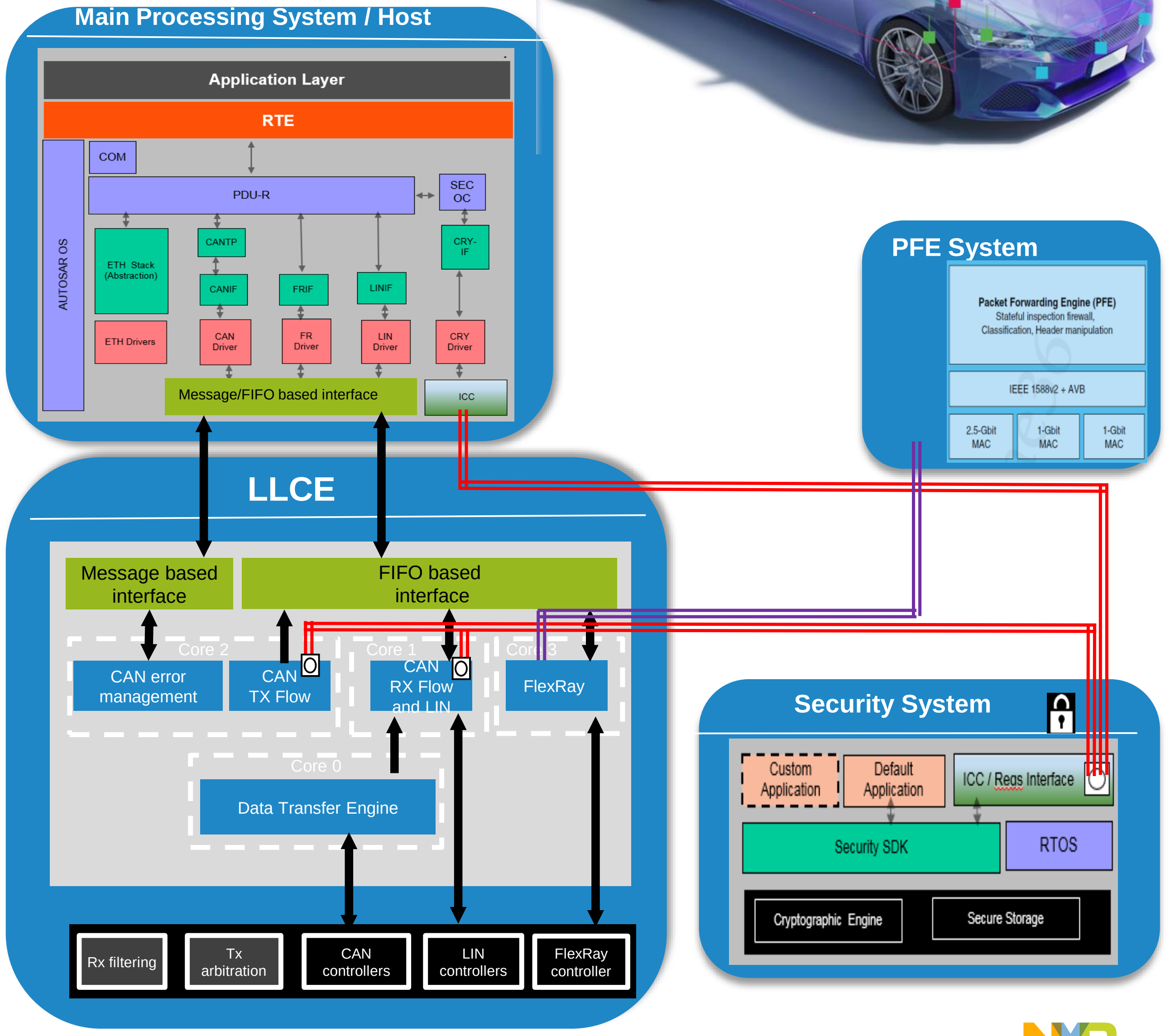
LLCE is a communication engine, a combination of cores, memory, hardware acceleration IP blocks and firmware to provide acceleration for standard automotive communication interfaces.

Fully programmable engine with firmware supporting:

- Offload of host CPU for all communication interface related tasks
  - Reduced interrupt load on the host core
  - Advanced software filtering
- Direct data transfer to/from HSE for security related tasks
- Flexible control and data interface exposed to the Host cores
- Efficient support of security over network protocols and global time synchronisation
- Hardware acceleration for filtering and prioritization of messages

Integrated into AUTOSAR® Communication stack via AUTOSAR MCAL drivers (CAN, FR, LIN)





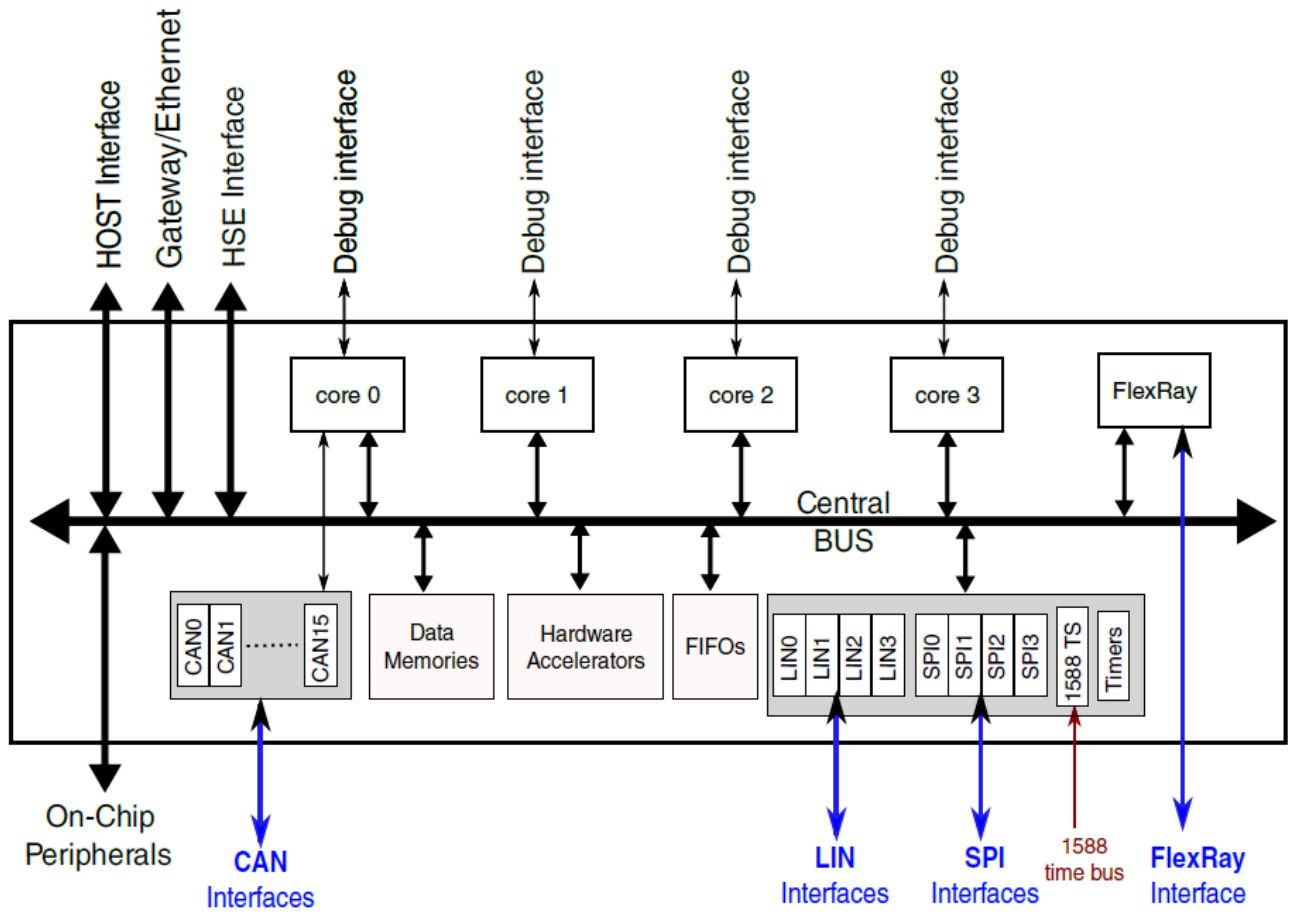
LOW LATENCY COMMUNICATIONS ENGINE HW ARCHITECTURE

LLCE features

- 4x Arm® Cortex® M0+ cores
  - Each with dedicated instruction/data RAM
- 16x BCAN (CAN 2.0 and CAN-FD)
- 4x LIN
- 1x FlexRay
- 4x SPI hardware interfaces, can be enabled by LLCE firmware
- Global time base
- Shared memory 2x 160kB
- FIFOs to manage pointers for message buffers
- Comms HW accelerators (RX-LUT, TX-LUT)
- Watchdogs, CRC, Core2core, Semaphore

LLCE connectivity

- Host cores (M7 and A53)
- Ethernet
- HSE (Security)



KEY ADVANTAGE OF LLCE VS TRADITIONAL APPROACH

Flexibility to Customer Application

- Flexible host interface (F/W based)
  - Can be customised to specific customer use case

Host CPU Offload Possibilities

- Function offload from main core
  - Advanced filtering (Bitwise, range, ..)
    - E.g. Removes Host Look up (100+ cycles)
  - Low latency frame gateway functions
    - E.g. Mirroring functions, Protocol conversion
- Reduced interrupt loading on the host core

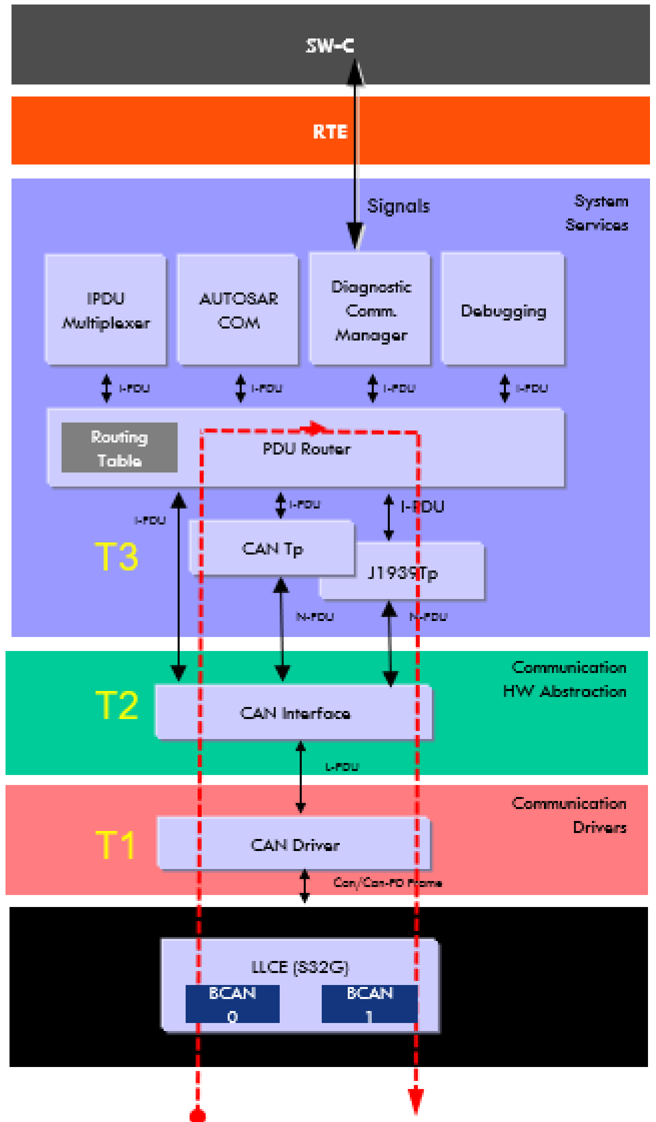
Host Interface

- Standard interface for CAN, FlexRay, LIN
- Back to back messages supported
- Reduction in host buffer management
  - Fire and forget interface
- Dynamic “MB”
  - Only limited by data RAM size

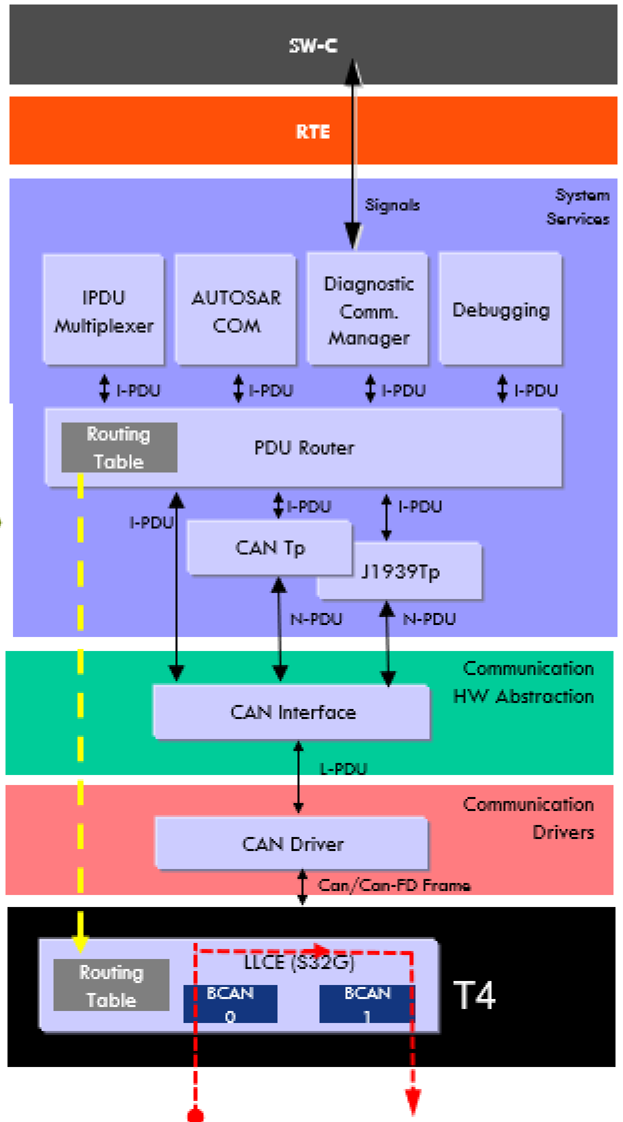
Enhanced Feature Support

- Security offload
- Global timestamping

HOST OFFLOADING BY LLCE INTERNAL ROUTING



Routing using host



Routing without host

Routing latency (using host) :
2xT1 + 2xT2 + 2xT3

Routing latency (without host) : T4

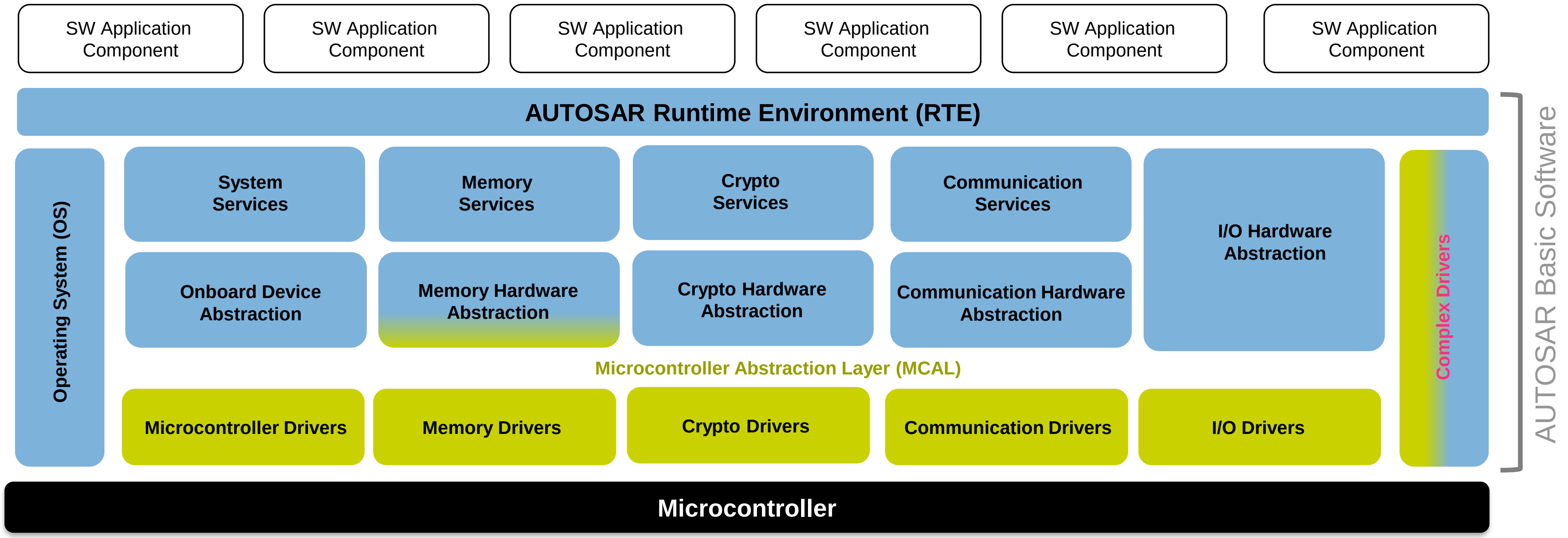
Routing Latency reduces by using LLCE internal routing as it saves time in high level SW stacks, interrupts overheads, etc.

LLCE Integration with AUTOSAR®

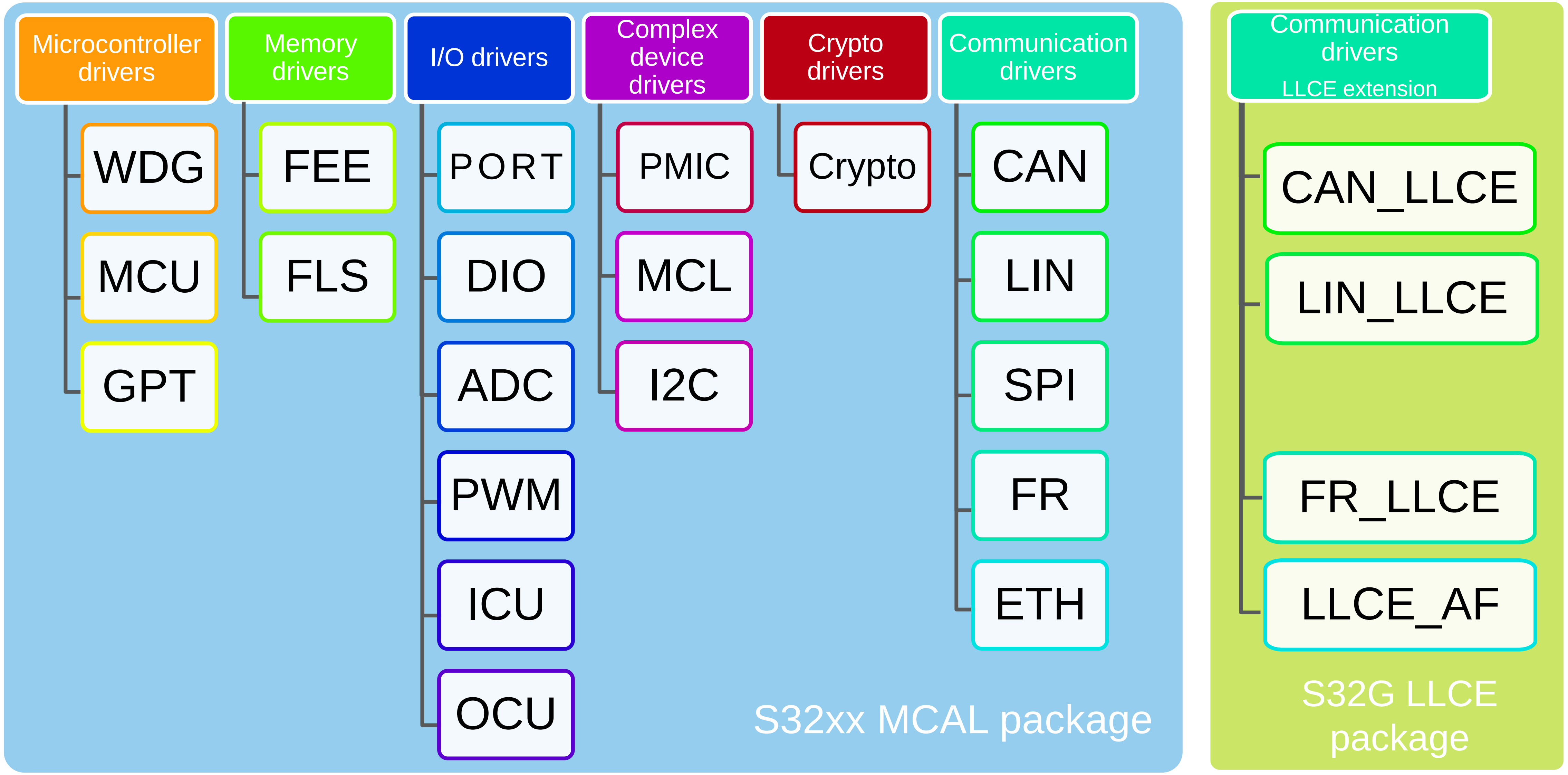
SECURE CONNECTIONS
FOR A SMARTER WORLD

NXP BASIC AUTOSAR® SOFTWARE

- NXP Standard Products – MCAL (source code), OS (source code) and Configuration Tool (executable) for MCAL and OS
- Partner Products (Elektrobit, Vector, KPIT, etc.) – The rest of AUTOSAR basic software as needed & Integration Services (NXP IP + Partner IP + Customer IP)
- Complex Drivers – custom software offered by NXP Consulting & Professional Engineering Services



S32G MCAL AND LLCE DRIVERS

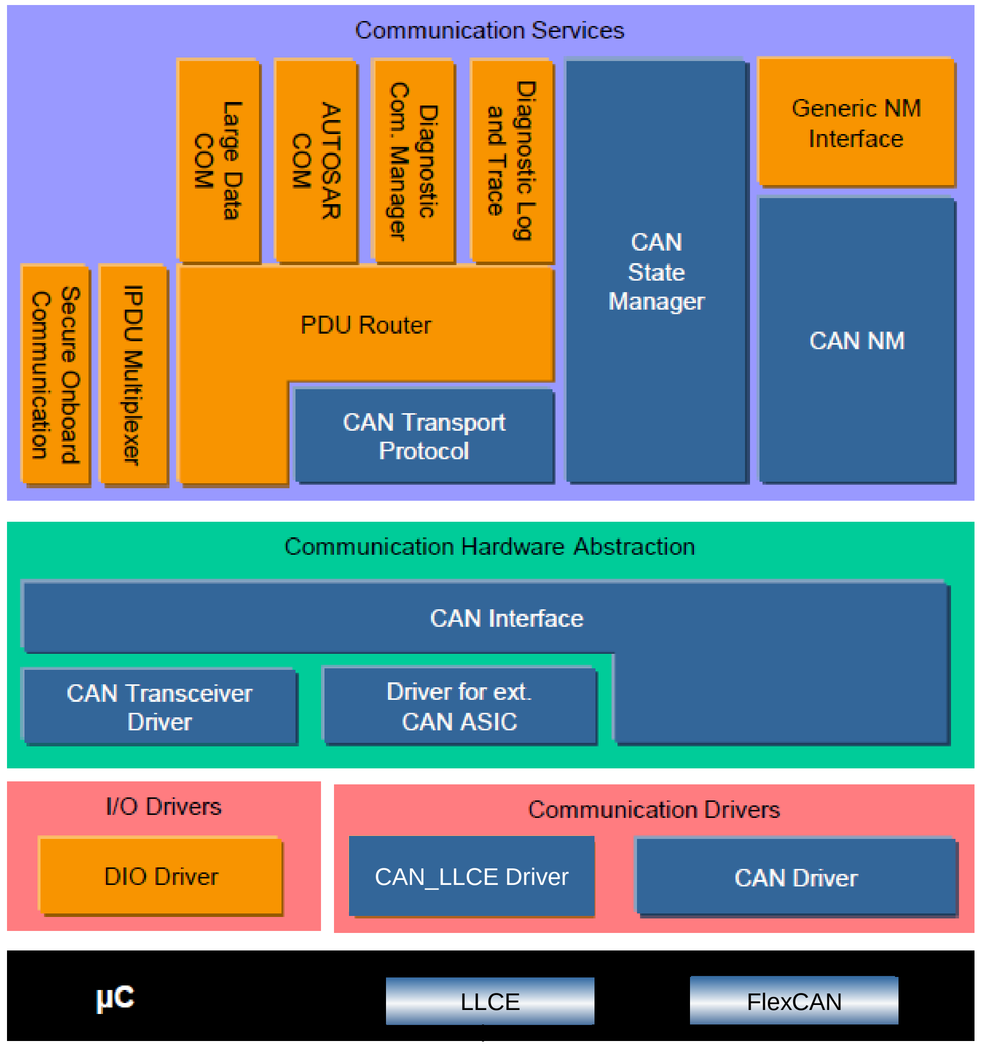


INTEGRATING LLCE WITH FLEXCAN

- AUTOSAR® specification supports multiple communication drivers
- CAN_LLCE and FlexCAN drivers can run in parallel

- Changes are needed in CAN interface
- Partners can easily integrate LLCE CAN/LIN/FR standard functionalities in AUTOSAR Stack

- Usage of advanced features of LLCE requires moving features from PDU Router and other components to LLCE firmware



LLCE Features

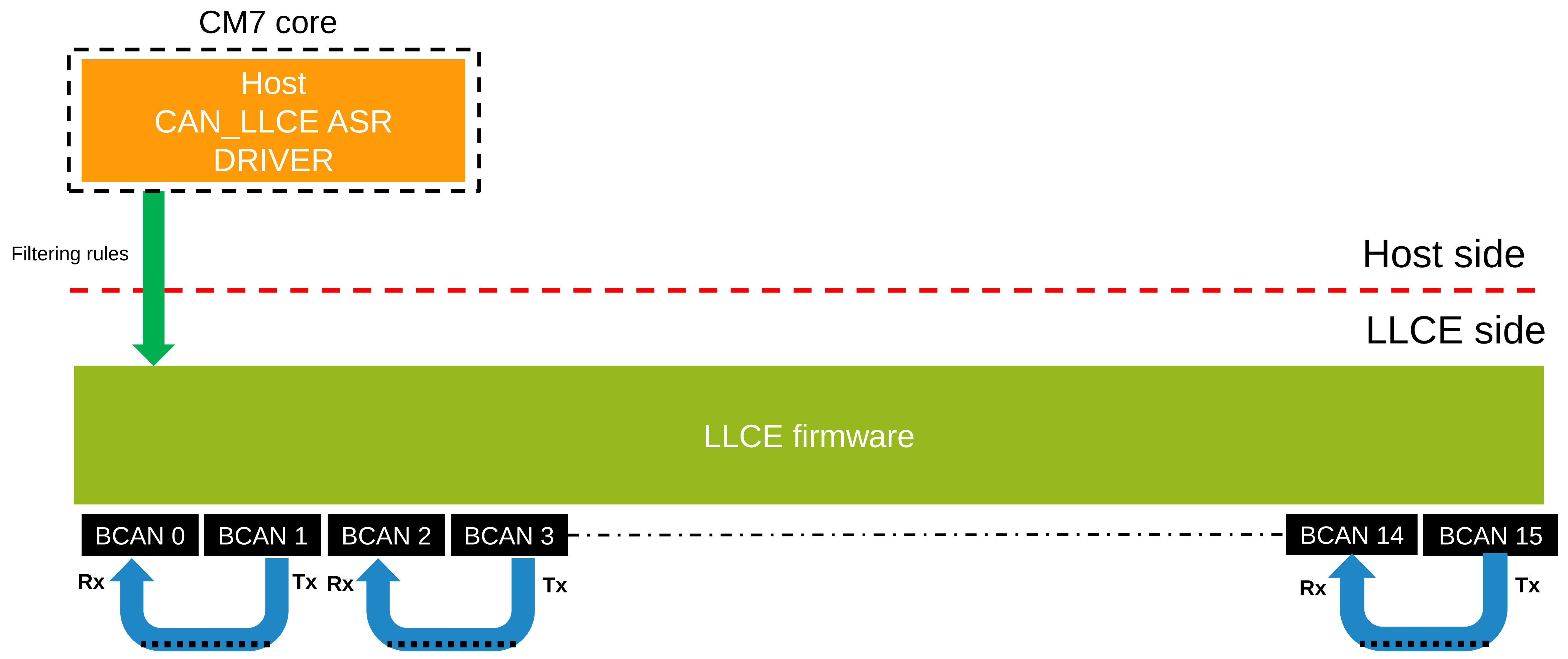
SECURE CONNECTIONS
FOR A SMARTER WORLD

FEATURES IN LLCE

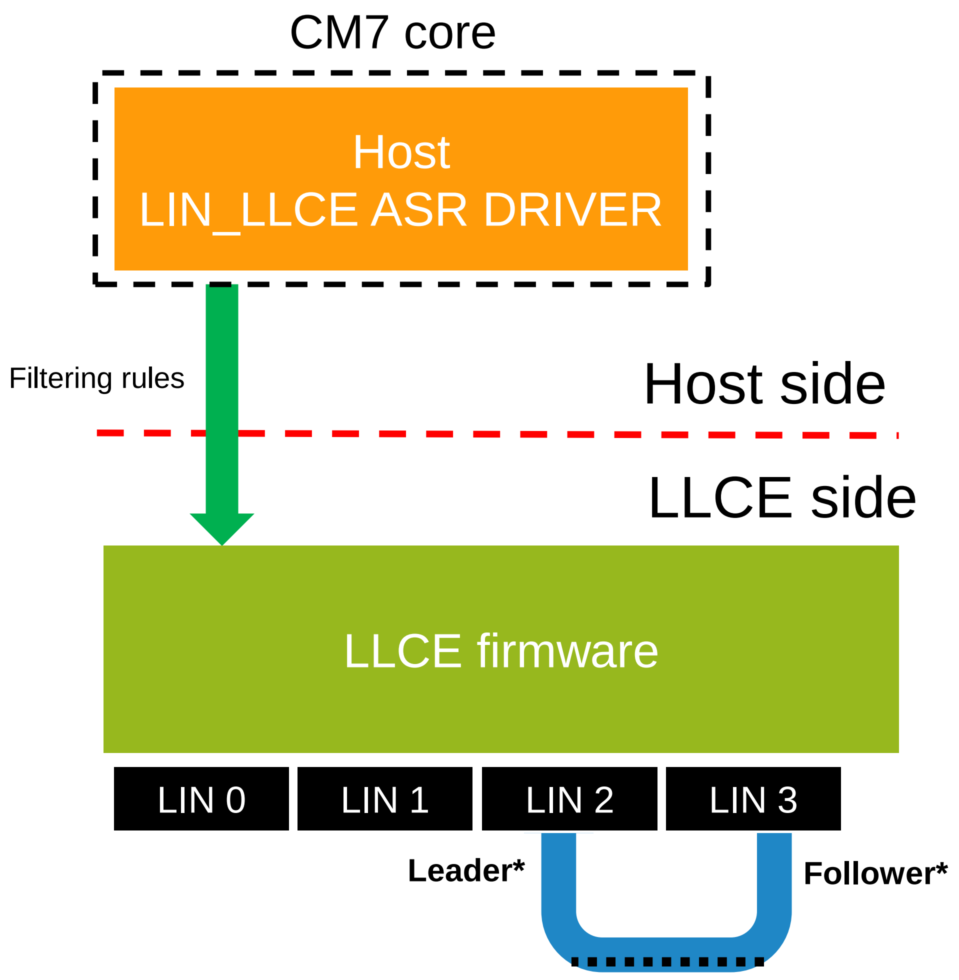
- CAN LOOPBACK
  - Simple sending and receiving frames from one channel to the other connected through wire
- LIN LOOPBACK
  - Creates a leader-follower* communication between 2 nodes
- CAN Multihost
  - Deliver received frames to multiple hosts
- CAN to CAN routing
  - Receive a frame from a BCAN and send it to one or multiple configured BCAN
  - Receive a frame from a BCAN, change the ID and send it to one or multiple configured BCAN
  - Convert a received standard CAN frame to CAN FD frame
  - Convert a CAN FD frame to CAN frame
- CAN to Ethernet routing
  - Packing selected CAN frames into an IEEE1722 AVTP protocol and sending over PFE on the Ethernet network
- Ethernet to CAN routing
  - Sending an IEEE1722 compliant frame to the specified RGMII port
  - Valid CAN frames contained in the Ethernet frame will be unpacked and sent to the respective channels

* Leader/follower, as the default NXP substitution, will be used in lieu of the recommendation of a standards organization.

LLCE CAN LOOPBACK

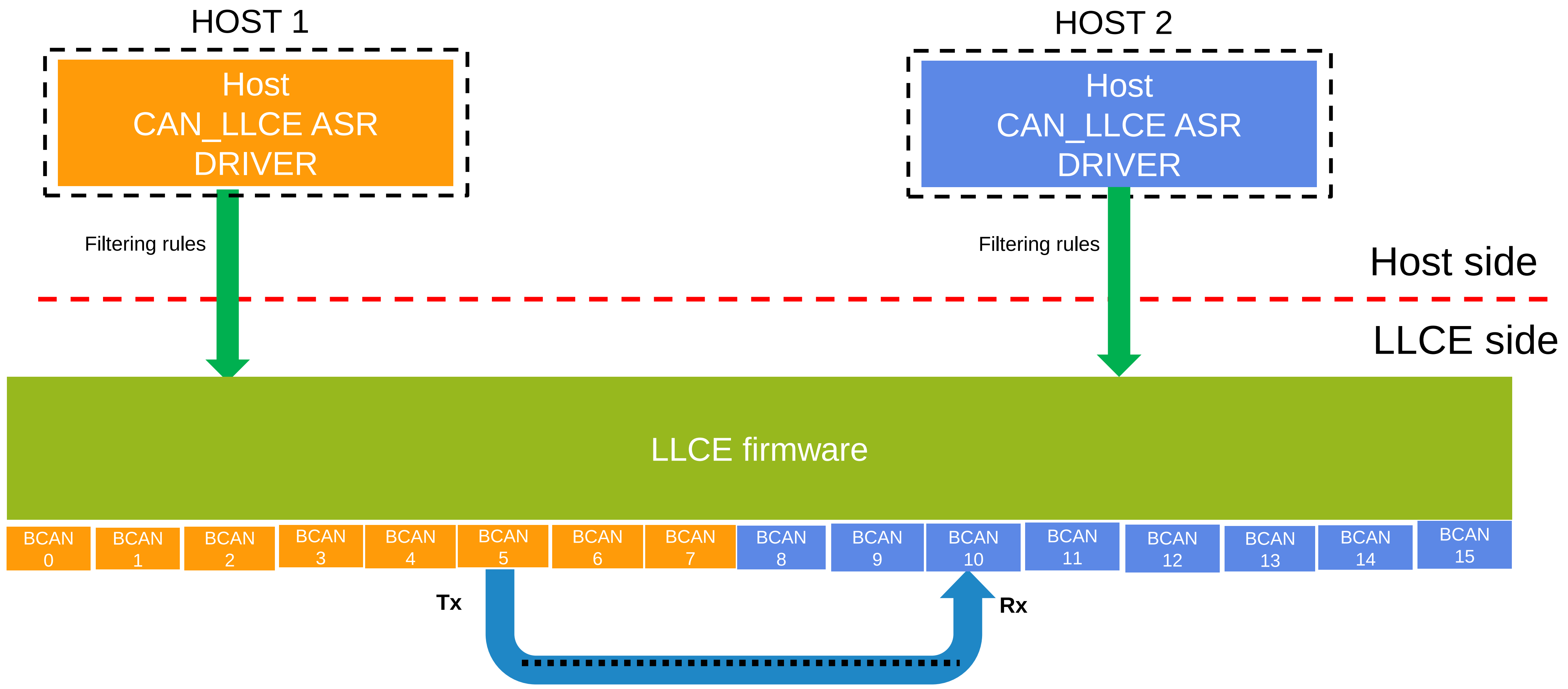


LLCE LIN LOOPBACK

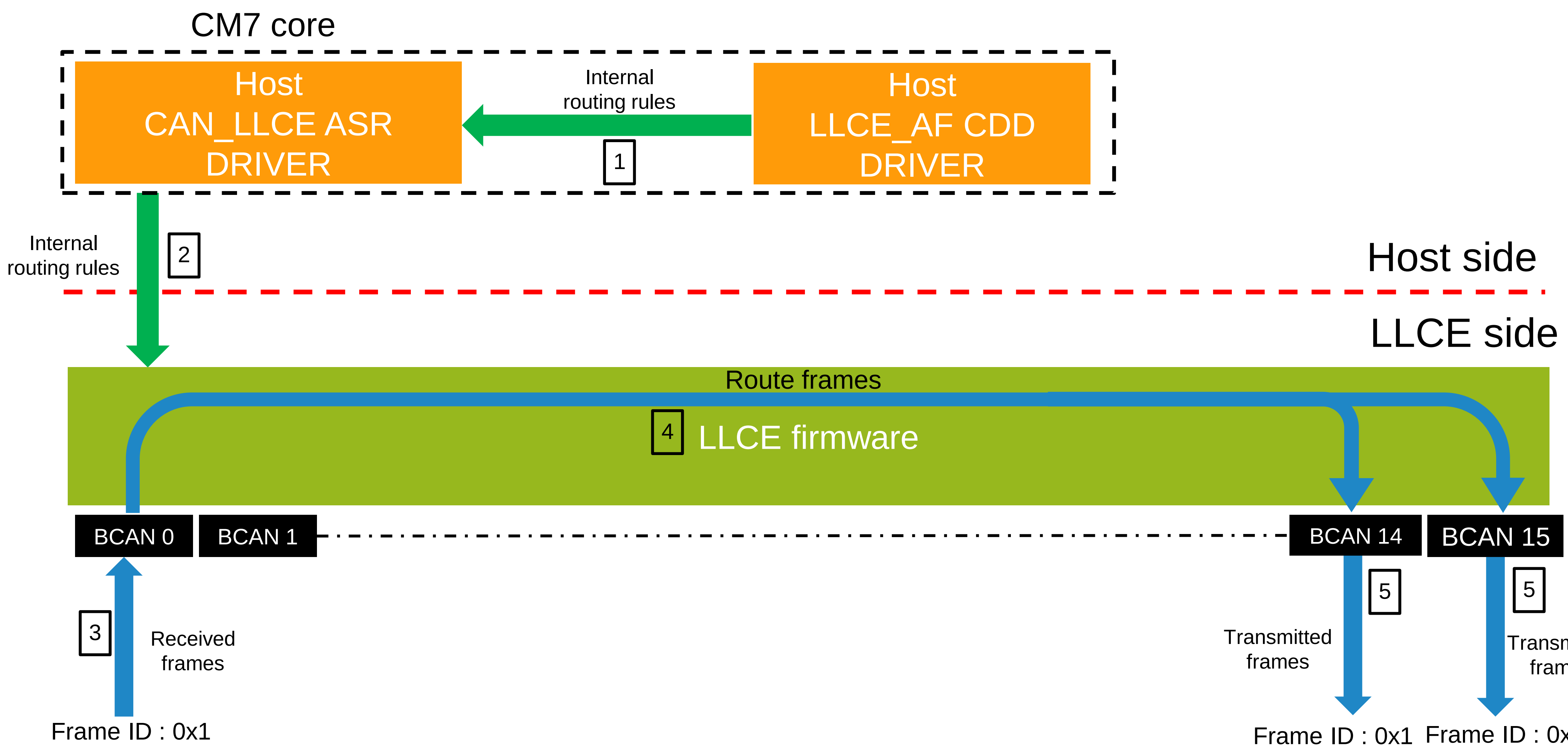


* Leader/follower, as the default NXP substitution, will be used in lieu of the recommendation of a standards organization.

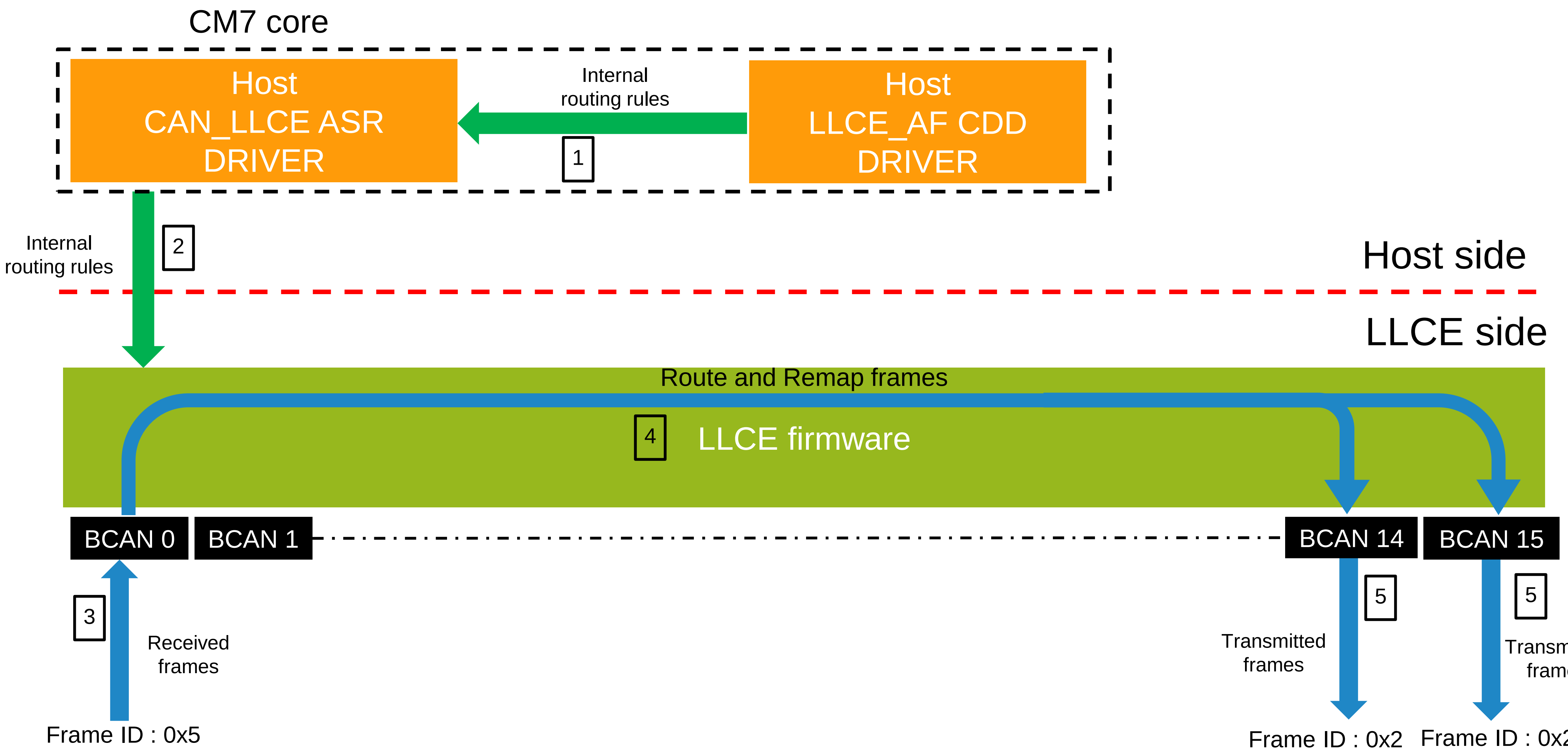
LLCE CAN MULTIHOST



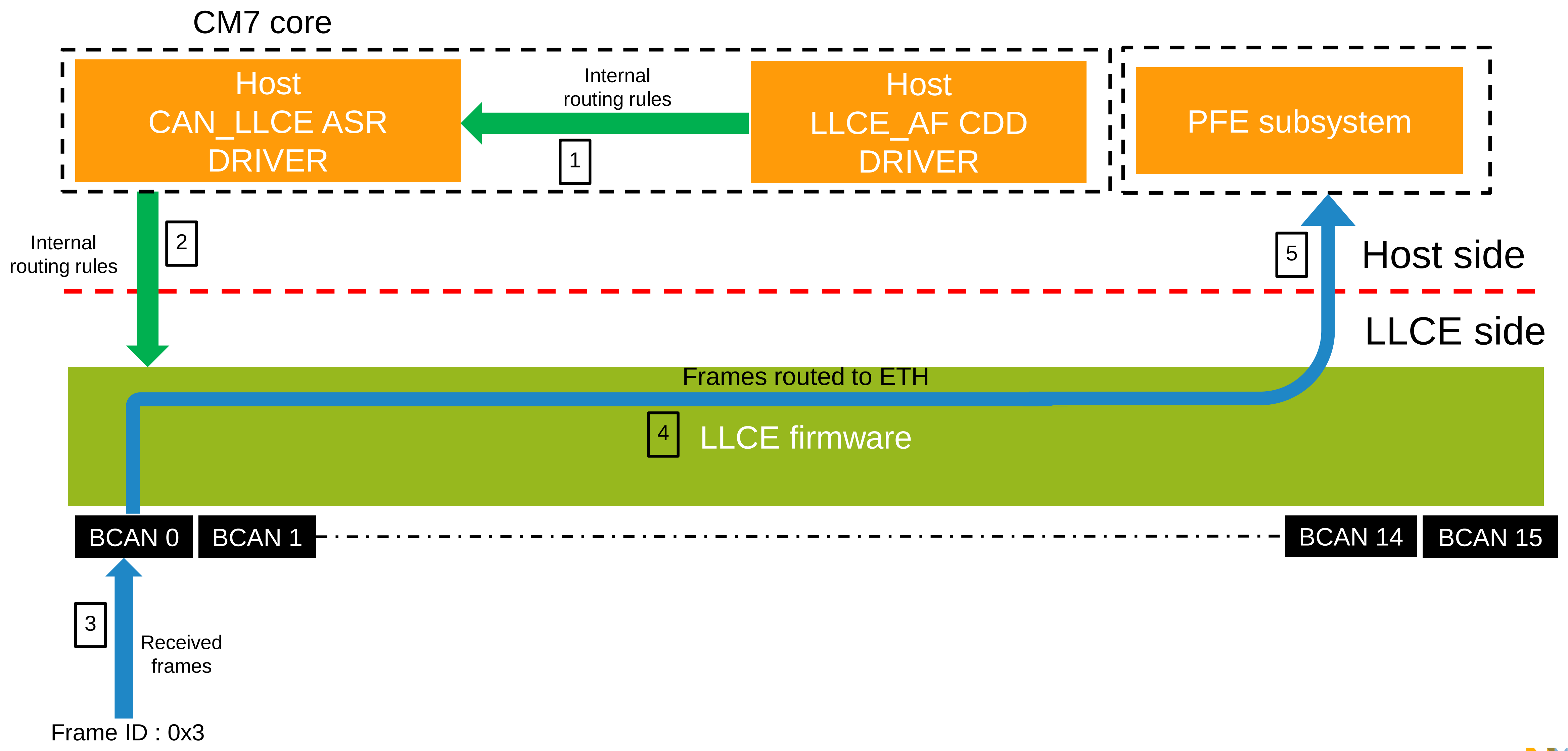
CAN-CAN ROUTING W/O ID REMAPPING



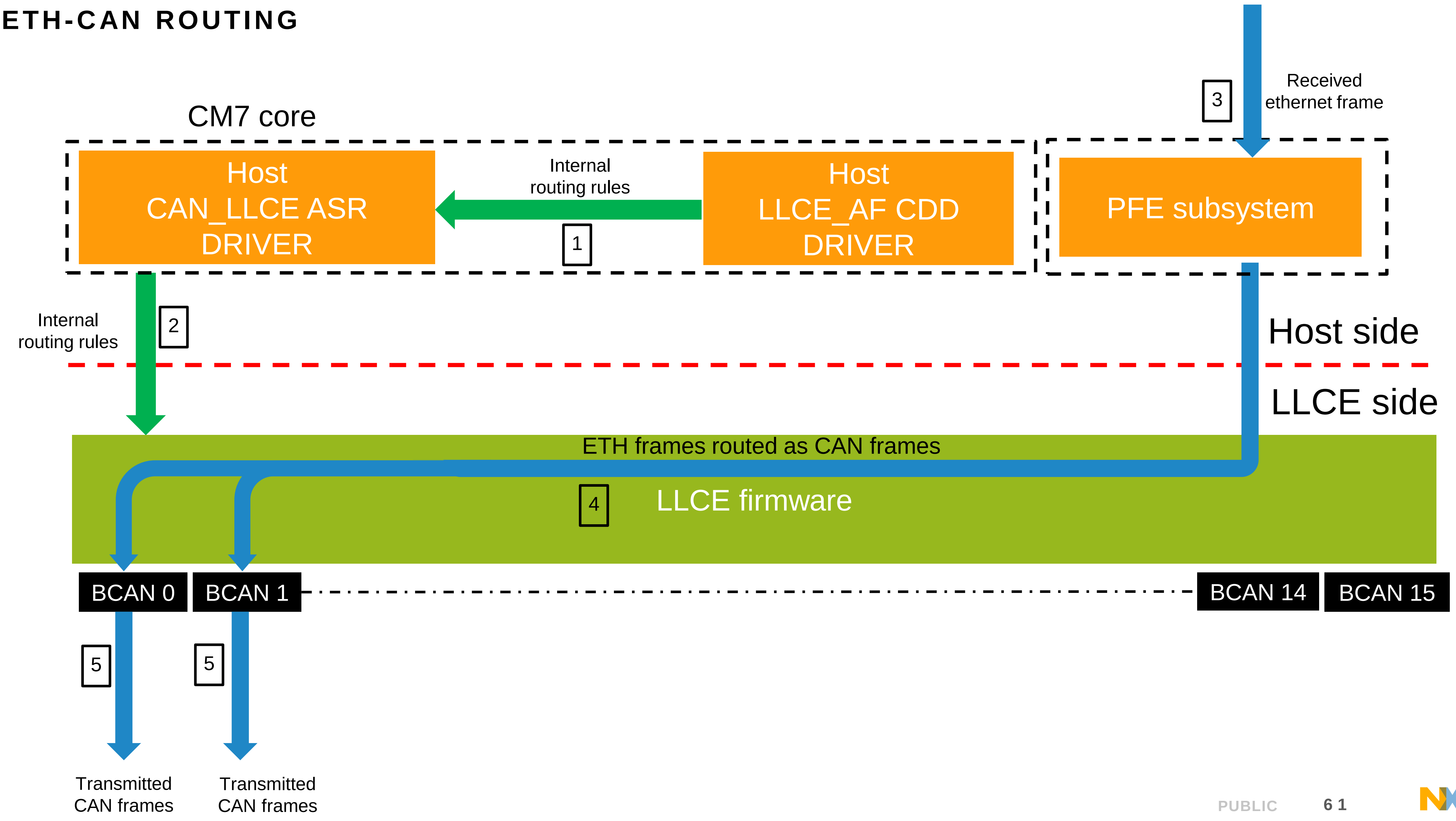
CAN-CAN ROUTING WITH ID REMAPPING



CAN-ETH ROUTING



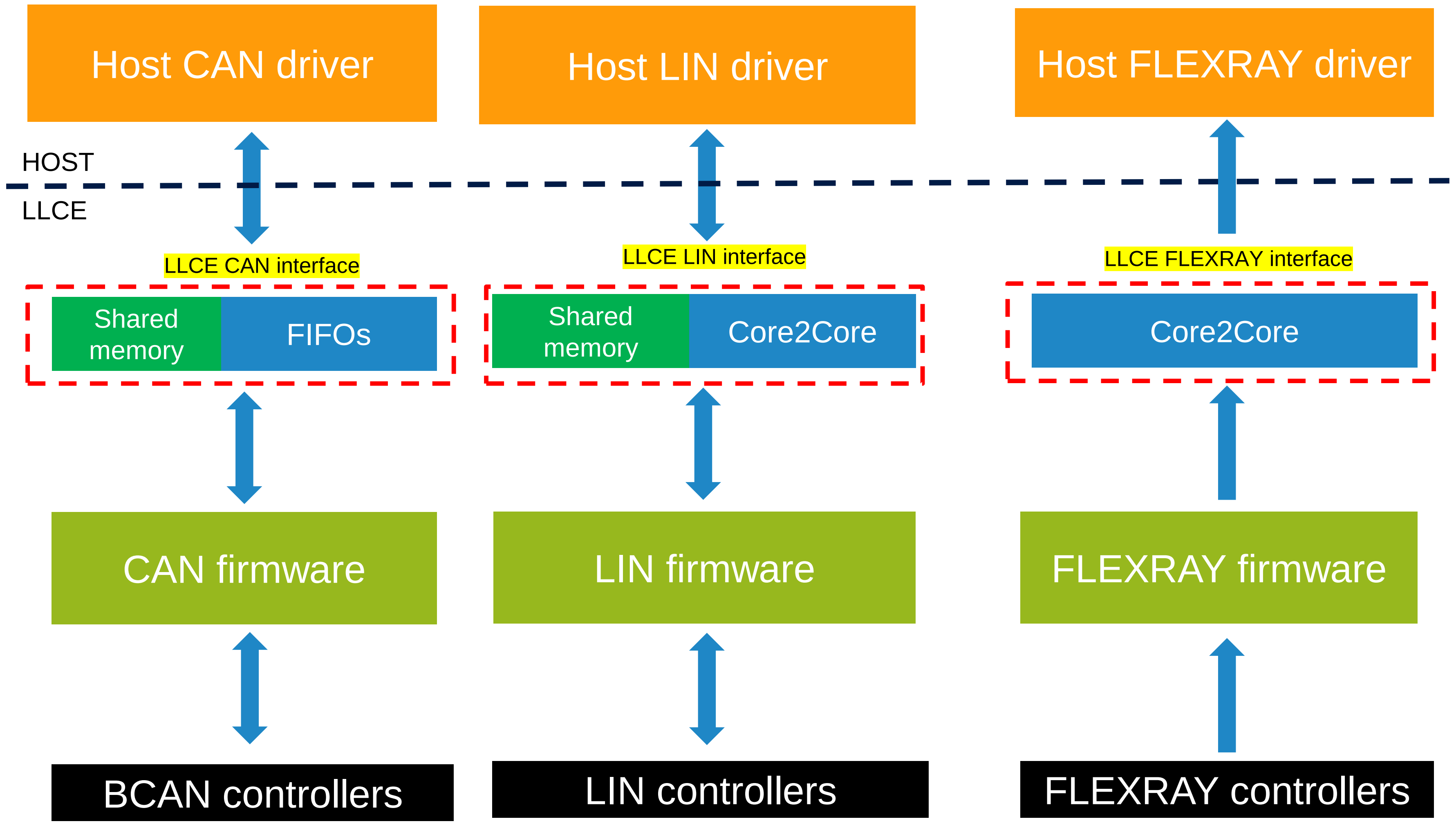
ETH-CAN ROUTING



LLCE Host Interface

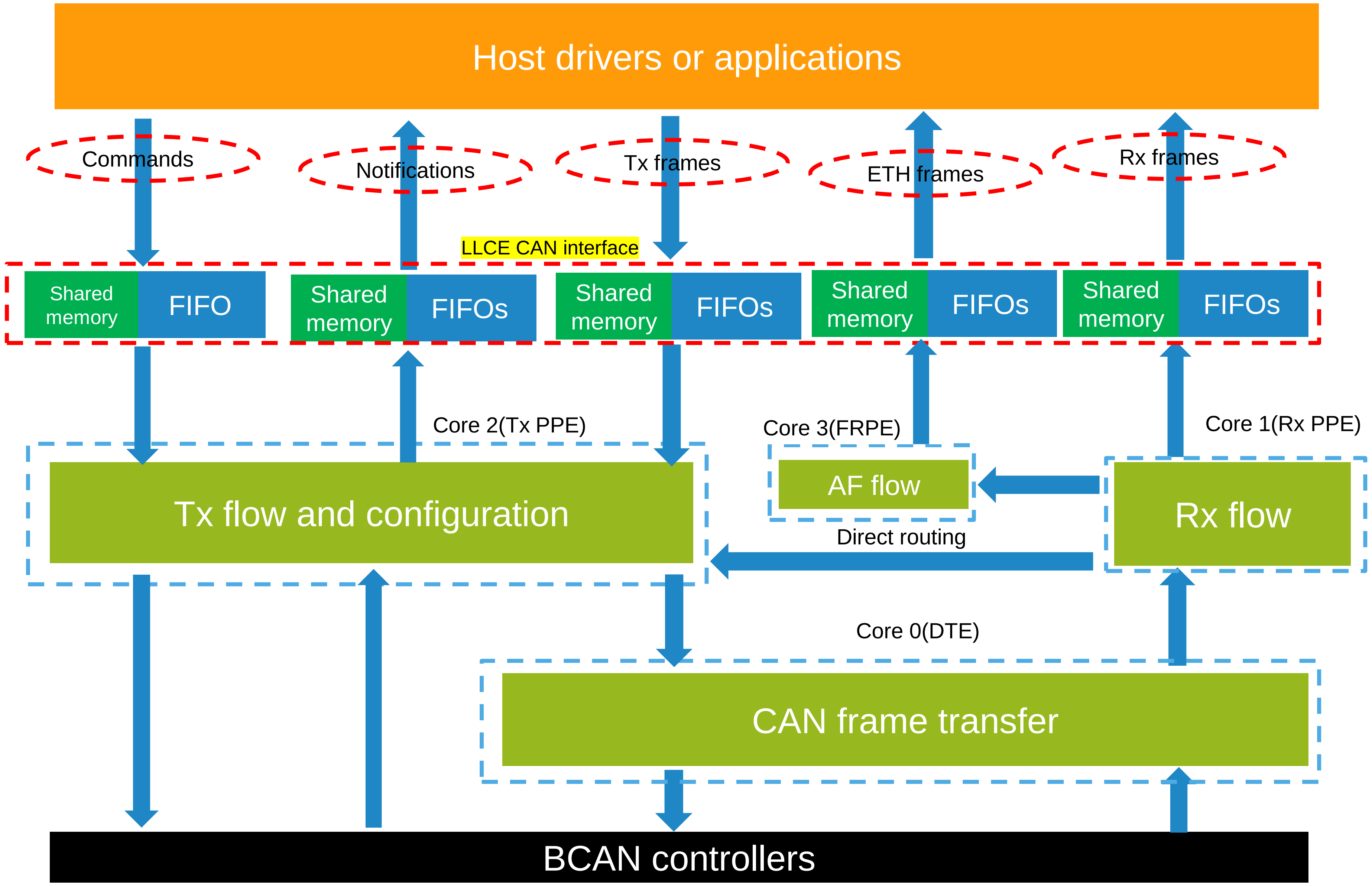
SECURE CONNECTIONS
FOR A SMARTER WORLD

LLCE FIRMWARE ARCHITECTURE - HIGH LEVEL VIEW



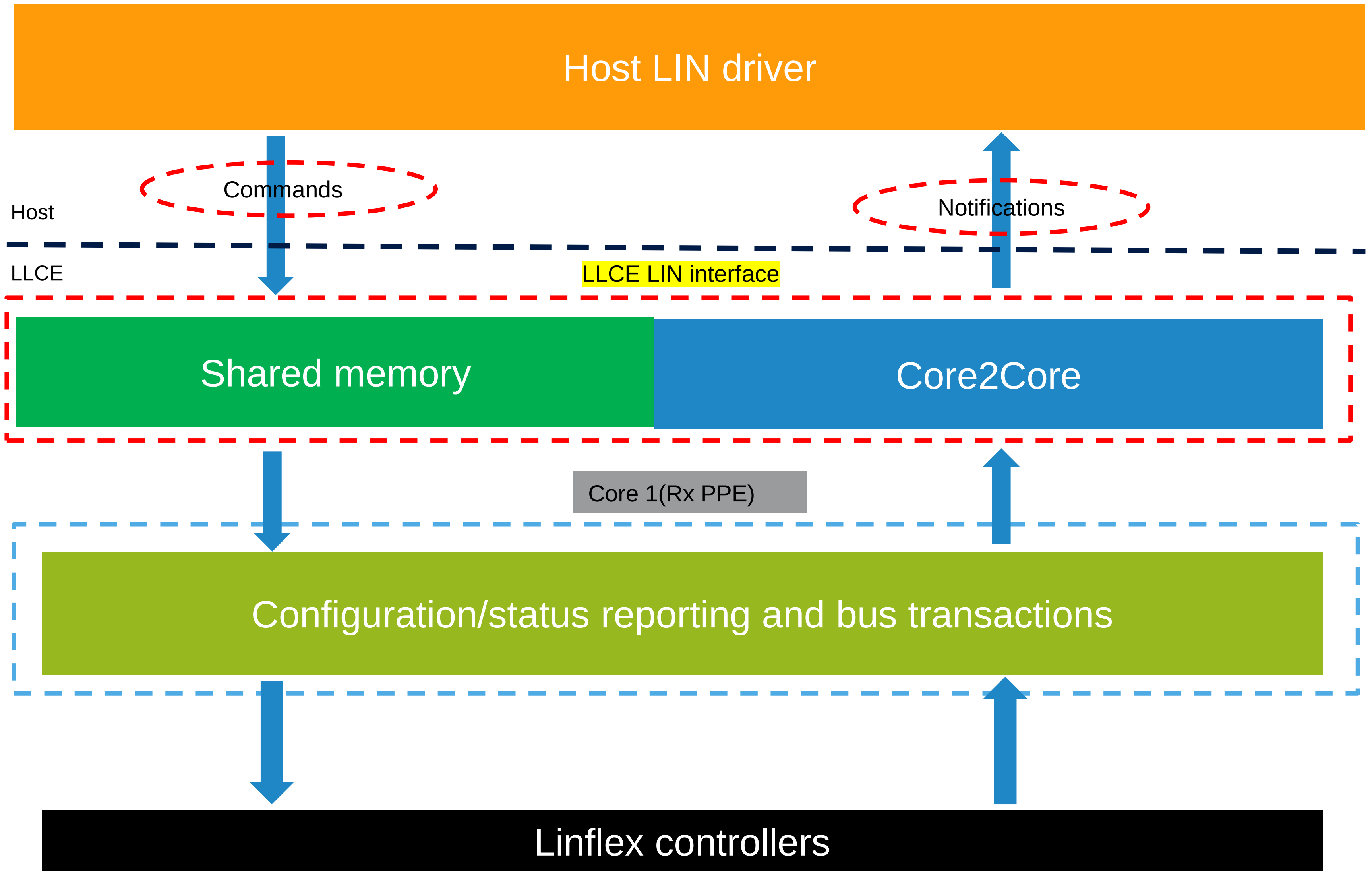
- Host side applications interacts with LLCE firmware by using 3 different/independent interfaces for each type of buses: CAN, LIN and FlexRay
- The host interface for each bus is composed from independent HW elements
- All the source files servicing each bus behavior are compiled together and the execution is distributed between multiple internal cores

LLCE CAN FIRMWARE ARCHITECTURE



- LLCE CAN firmware is distributed and runs on all 4 internal cores
- Interactions between host applications and CAN firmware is done by using multiple custom interfaces composed from different shared memory areas and HW FIFOs
- Hardware FIFOs are used also as inter-core communication mechanism inside LLCE
- DTE(Data Transfer Engine) core run fully in polling mode in order to get all frames from all BCANs

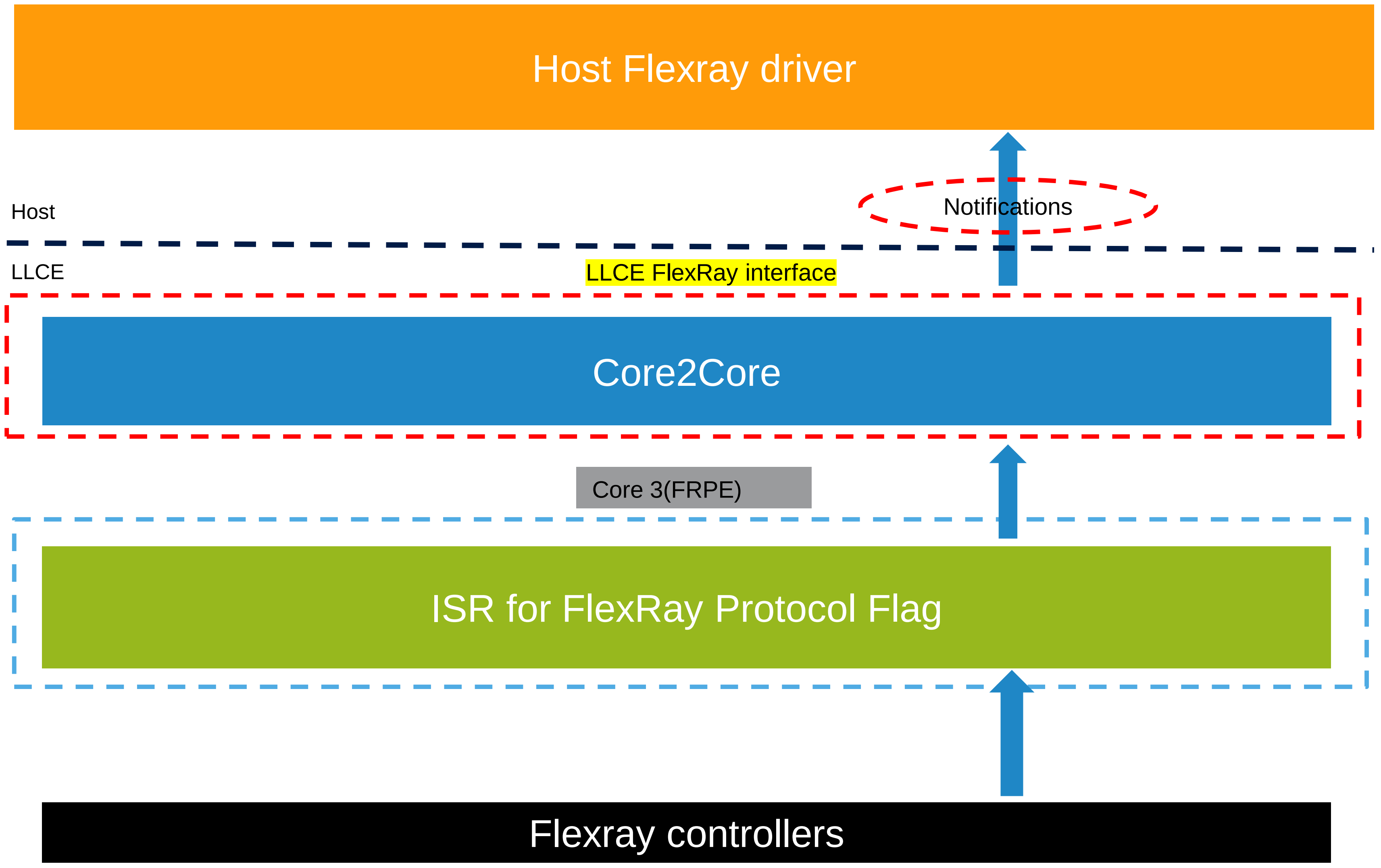
LLCE LIN FIRMWARE ARCHITECTURE



- LLCE LIN firmware is running fully on the Rx PPE core
- LIN firmware enables LIN node to act either as leader or follower* on the bus
- LIN firmware reacts only by responding to the host commands
- Host driver writes into shared memory the command parameters and notify LIN firmware by raising a flag inside Core2Core module

* Leader/follower, as the default NXP substitution, will be used in lieu of the recommendation of a standards organization.

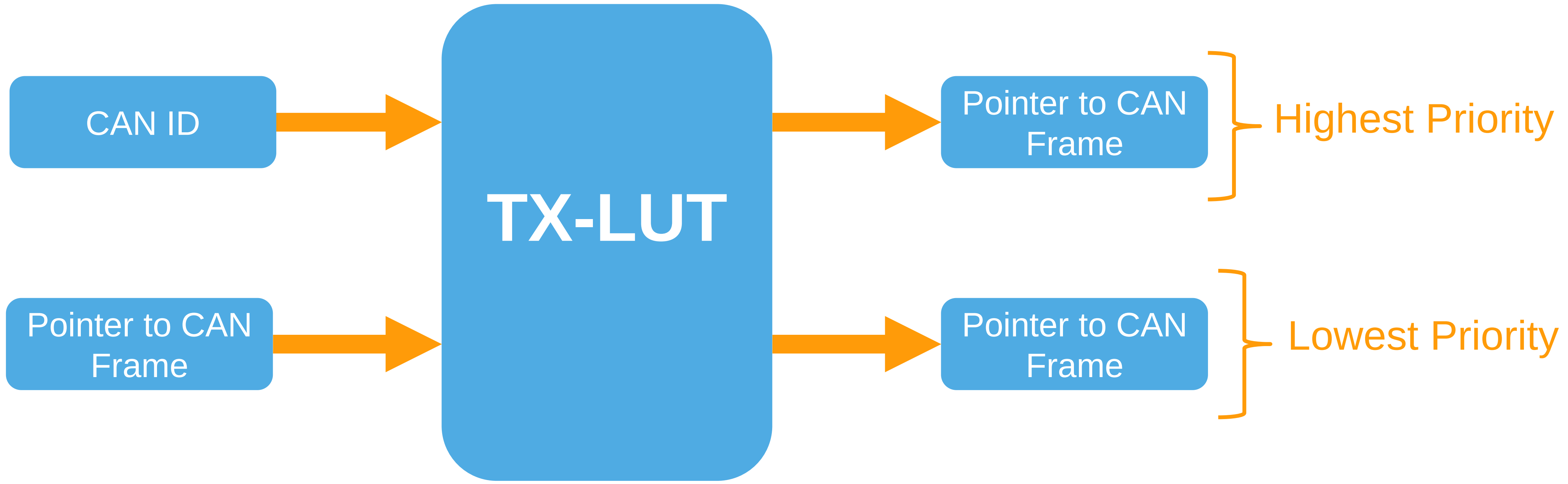
LLCE FLEXRAY FIRMWARE ARCHITECTURE



- Currently the FlexRay controller included into LLCE is managed directly by the host using AUTOSAR® FlexRay driver
- FlexRay related firmware just forward a FlexRay controller interrupt to host driver via Core2Core module in order to workaround specific hardware restrictions
- FlexRay related firmware is runs on the FRPE core

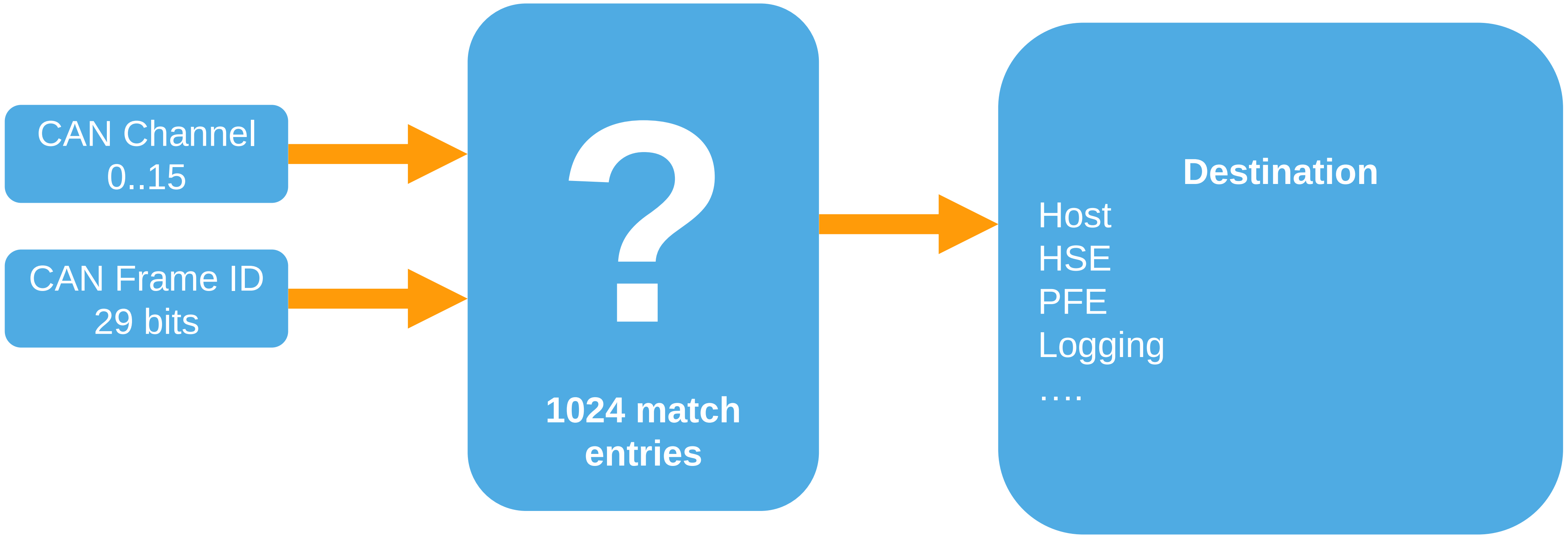
HW ACCELERATOR: TRANSMIT LOOKUP TABLE (TX LUT)

- LLCE has a dedicated hardware module to make decisions based on CAN ID for Transmitting CAN channel
- One TX-LUT per CAN channel => 16x TX-LUT
- Searches for highest and lowest priority message



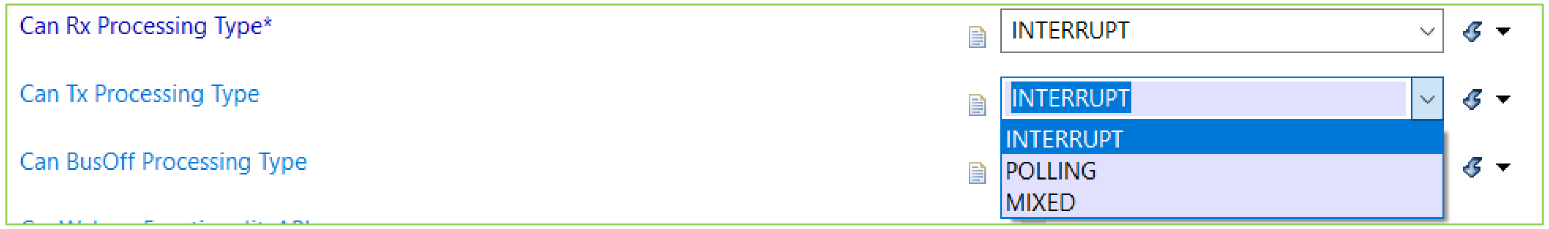
HW ACCELERATOR: RECEIVE LOOKUP ACCELERATOR (RX-LUT)

- LLCE has a dedicated hardware module to make decisions based on receiving CAN channel and CAN ID
- One RX-LUT for all 16 CAN channels
- LLCE supports 1024 filter entries:
  - 512 'exact match' type filter
  - 512 'masked or range' type filter



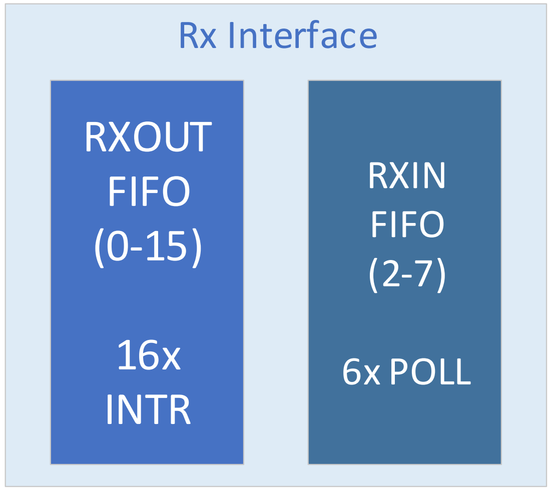
HOST – LLCE FIFO

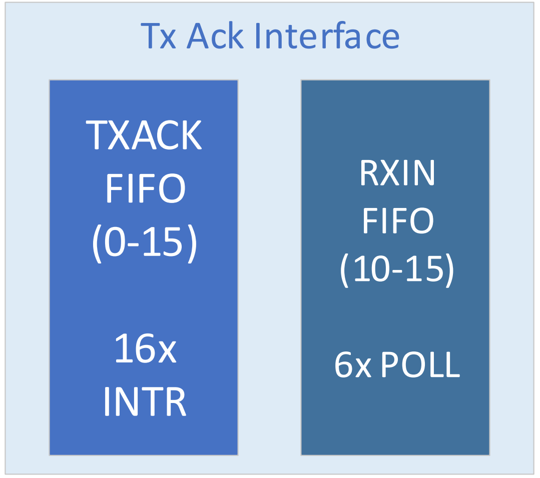
- Host interacts with LLCE using below HW FIFOs
  - RX-OUT FIFO
  - RX-IN FIFO
  - BLR-OUT FIFO
  - BLR-IN FIFO
  - TX-ACK FIFO
- LLCE provides 16 Tx, 22 Rx and 22 Tx Ack interfaces - 16 interrupts and 6 polling classes
- Interfaces from 0-15 are configured to work in interrupt mode and 16-21 in polling mode
- Tx Ack and Rx interface can be configured in tresos in interrupt, polling or mixed mode



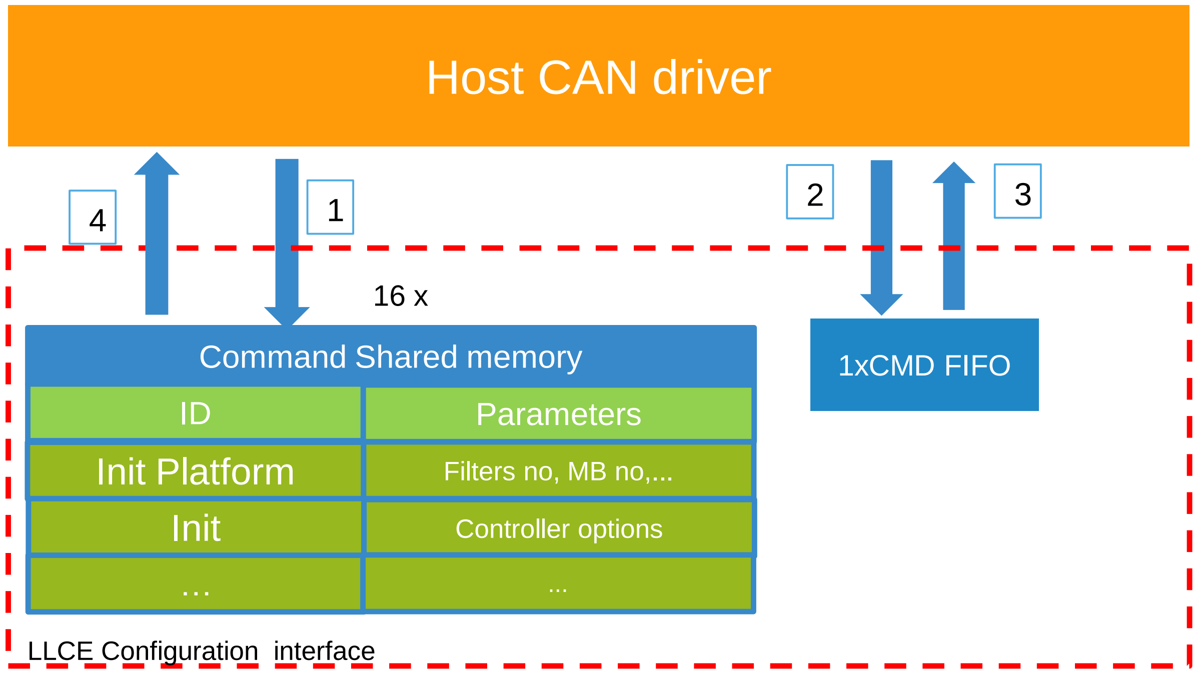
Can Rx Processing Type*	INTERRUPT
Can Tx Processing Type	INTERRUPT
Can BusOff Processing Type	

Selecting Processing Type as mixed, gives the flexibility to configure each HW object independently in interrupt or polling mode.





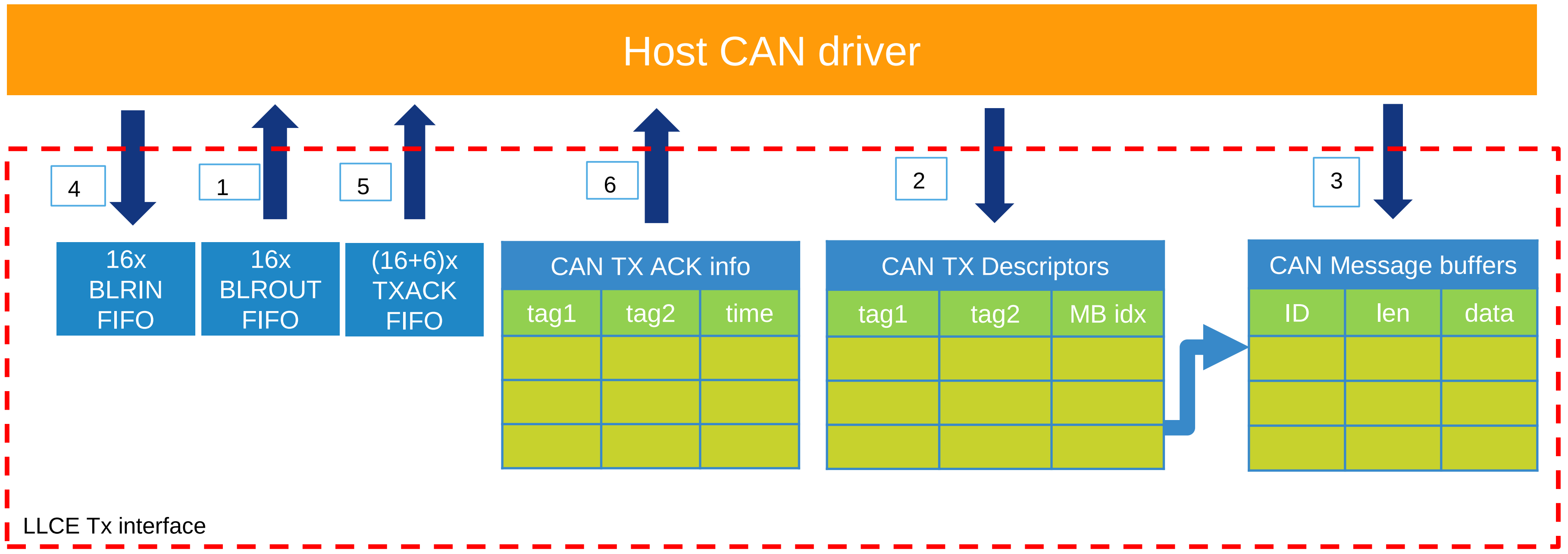
LLCE CAN INTERFACE - CONFIGURATION FLOWS



ID	Parameters
Init Platform	Filters no, MB no,...
Init	Controller options
…	...

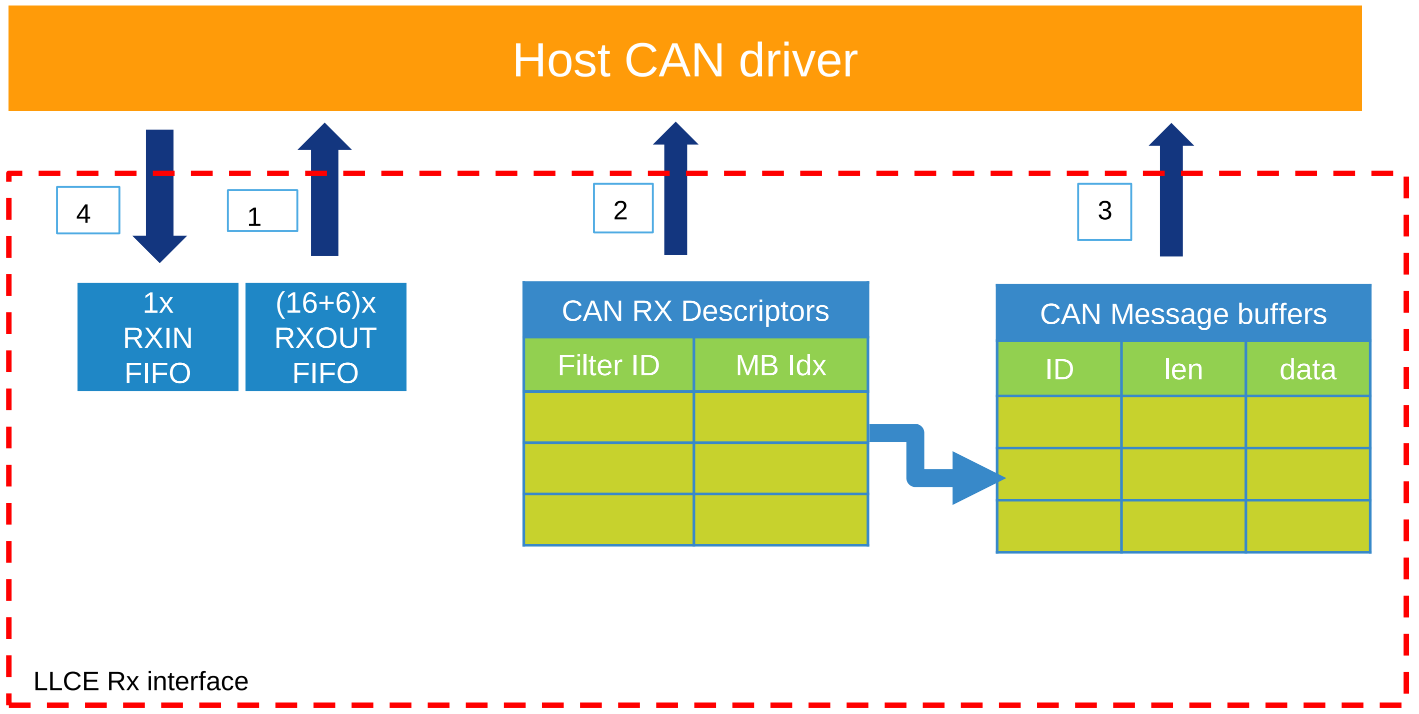
1. Write command ID and command parameters into the shared memory
2. Write the channel ID into the CMD FIFO
3. Check in polling way if the NOT_EMPTY interrupt flag of the CMD FIFO is cleared by the firmware in order to detect the completion of the command processing by the LLCE firmware
4. Read the command completion status from the same shared memory location

LLCE CAN INTERFACE - TRANSMISSION FLOW



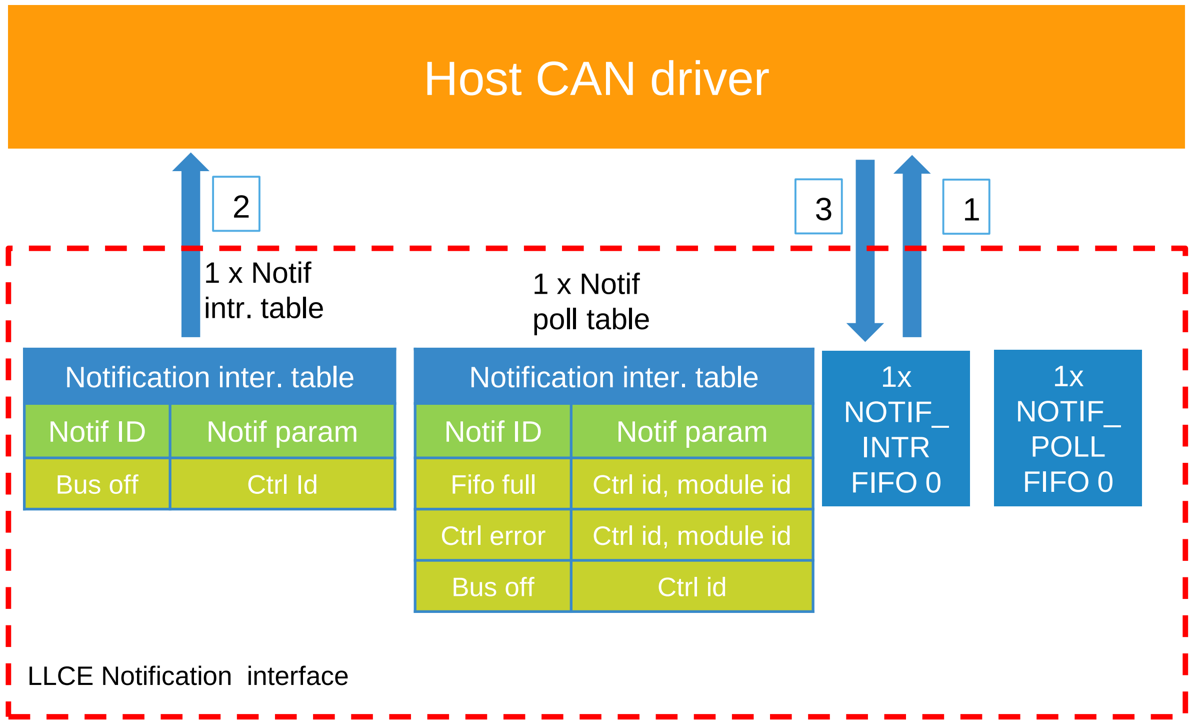
1. Read an index to an empty transmit descriptor from channel's BLROUT FIFO. This operation can be done in poling or in interrupt mode
2. Write the fields of transmission descriptor and extract the index to the message buffer table
3. Write the CAN frame content into the message buffer
4. Write the index of the transmission descriptor into the channel's BLRIN FIFO in order to trigger the transmission inside LLCE
5. Read an index into acknowledge information table from the channel's TXACK FIFO. This operation can be done either in polling or interrupt mode
6. Read the content of the referred entry from the acknowledge information table and process it

LLCE CAN INTERFACE - RECEPTION FLOW



1. Read an index to a full receive descriptor from channel's RXOUT FIFO. This operation can be done in poling or in interrupt mode.
2. Read reception descriptor content and extract from it the index into the CAN message buffer table.
3. Read the content of the full reception message buffer and process it.
4. Write the index of the receive descriptor to the RXIN FIFO in order to send it back to the LLCE to be reused.

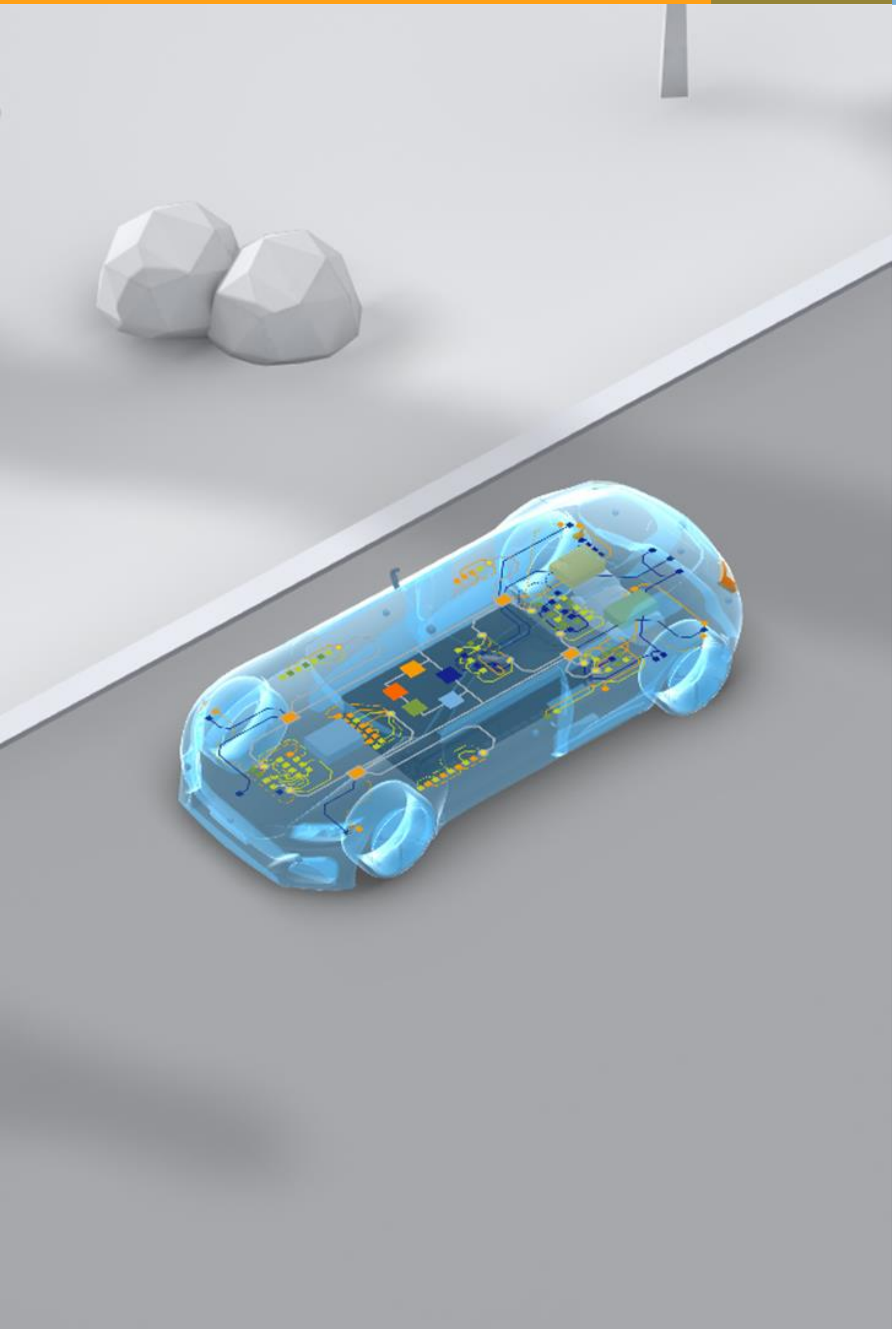
LLCE CAN INTERFACE - NOTIFICATION FLOW: INTERRUPT MODE



1. For notifications serviced by interrupt read an entry/index if available from the NOTIF_INTR FIFO.
2. Use the previous read index in order to access the notification parameters from the corresponding table and process the notification event.
3. Clear the interrupt status flag NOT_EMPTY of the NOTIF_INTR FIFO.

Notification examples:

LLCE_ERROR_FIFO_FULL
LLCE_ERROR_FIFO_EMPTY
LLCE_ERROR_MB_NOTAVAILABLE
LLCE_ERROR_CAN_PROTOCOL
LLCE_ERROR_HOH_NOTAVAILABLE
LLCE_ERROR_TXLUT_FULL
LLCE_ERROR_CMD_PROCESSING
LLCE_ERROR_HARDWARE_COMMAND
LLCE_ERROR_HARDWARE_RXLUT
LLCE_ERROR_HARDWARE_TXLUT
LLCE_ERROR_HARDWARE_BCAN
LLCE_ERROR_HARDWARE_BUSOFF
LLCE_ERROR_CTRL_NOT_READY
LLCE_ERROR_BUSOFF
LLCE_ERROR_FIFO_LOG_FULL
LLCE_ERROR_CAN2CAN



We recommend you follow these classes:

- Introducing the S32G GoldVIP (Vehicle Integration Platform)
- NXP Reference Design Board (RDB) for Vehicle Networking
- Applying TSN Tools for Quality of Service and Reliability
- Integrating TSN and Middleware Protocols

TECHNOLOGY SHOWROOM

JOURNEYS BY DESIRED ENGAGEMENT

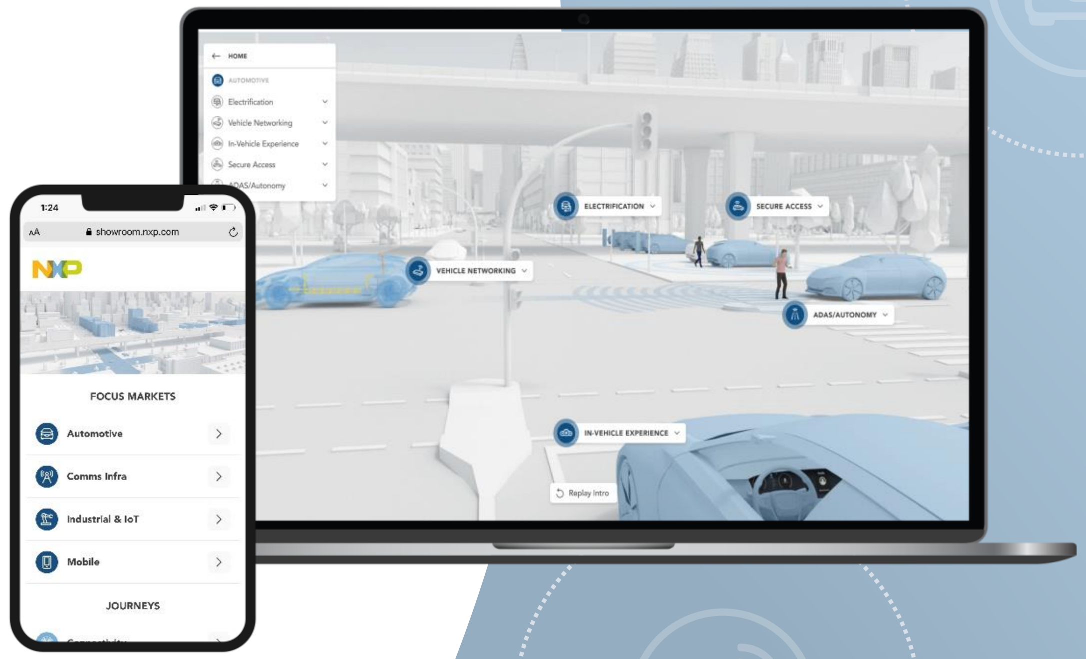
Self-guided tour
Live-streaming at set times
Guided tours

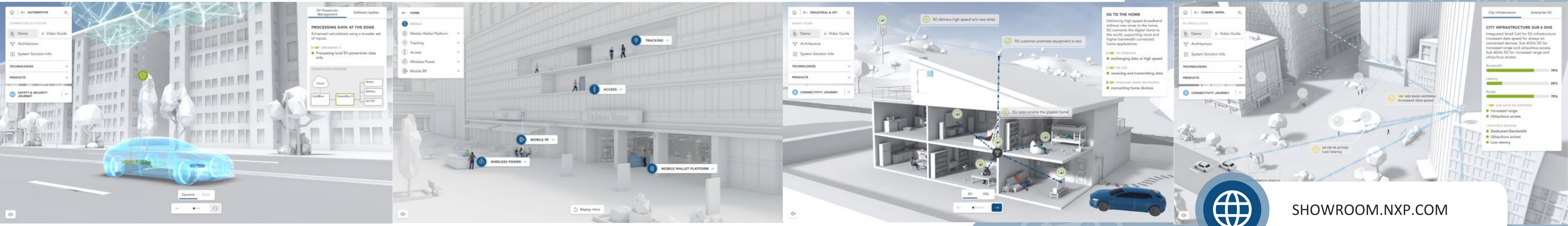
40+ VIRTUAL DEMOS

Focus on system solutions
Set up along NXP verticals

JOURNEYS BY DESIRED FOCUS

Edge & AI/ML
Safety & Security
Connectivity
Analog





SHOWROOM.NXP.COM

SECURE CONNECTIONS
FOR A SMARTER WORLD

SHOWROOM.NXP.COM