Creating Resilient Deterministic Networks

Jeff Steinheider

Director Product Marketing Industrial Applications Processors

May 2019 | AMF-IND-T3529

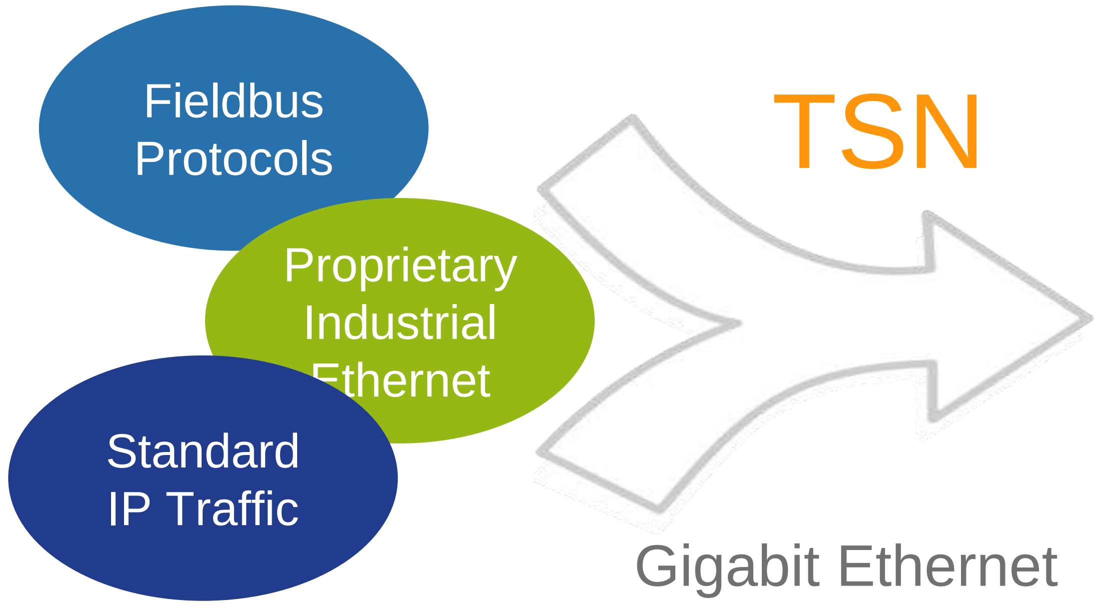
SECURE CONNECTIONS FOR A SMARTER WORLD

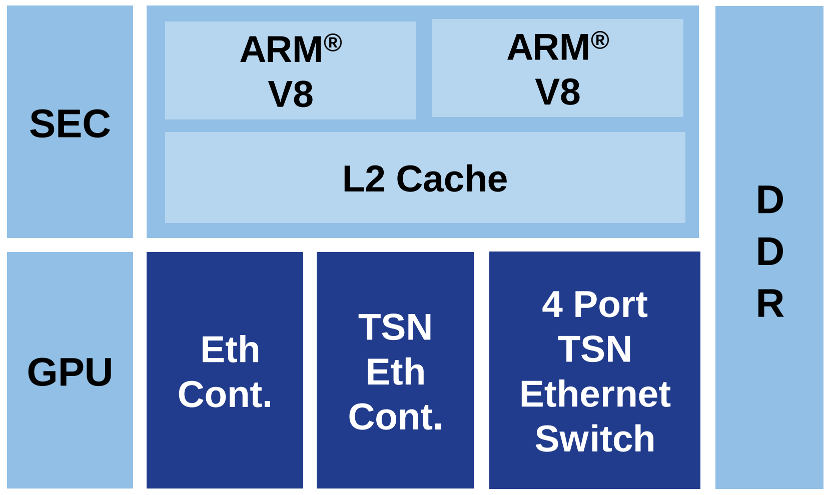
Agenda

- Time Sensitive Networking
- LS1028A
- Timing and Synchronization
- Time-Aware Shaper
- Frame Preemption
- Frame Replication and Elimination

Embedded Time-Sensitive Networking (TSN)

- Converge OT and IT traffic in a single network
- Determinist Ethernet at gigabit speeds
- Reduce network delays, improve robustness
- Embedded in Multi-processor SoCs

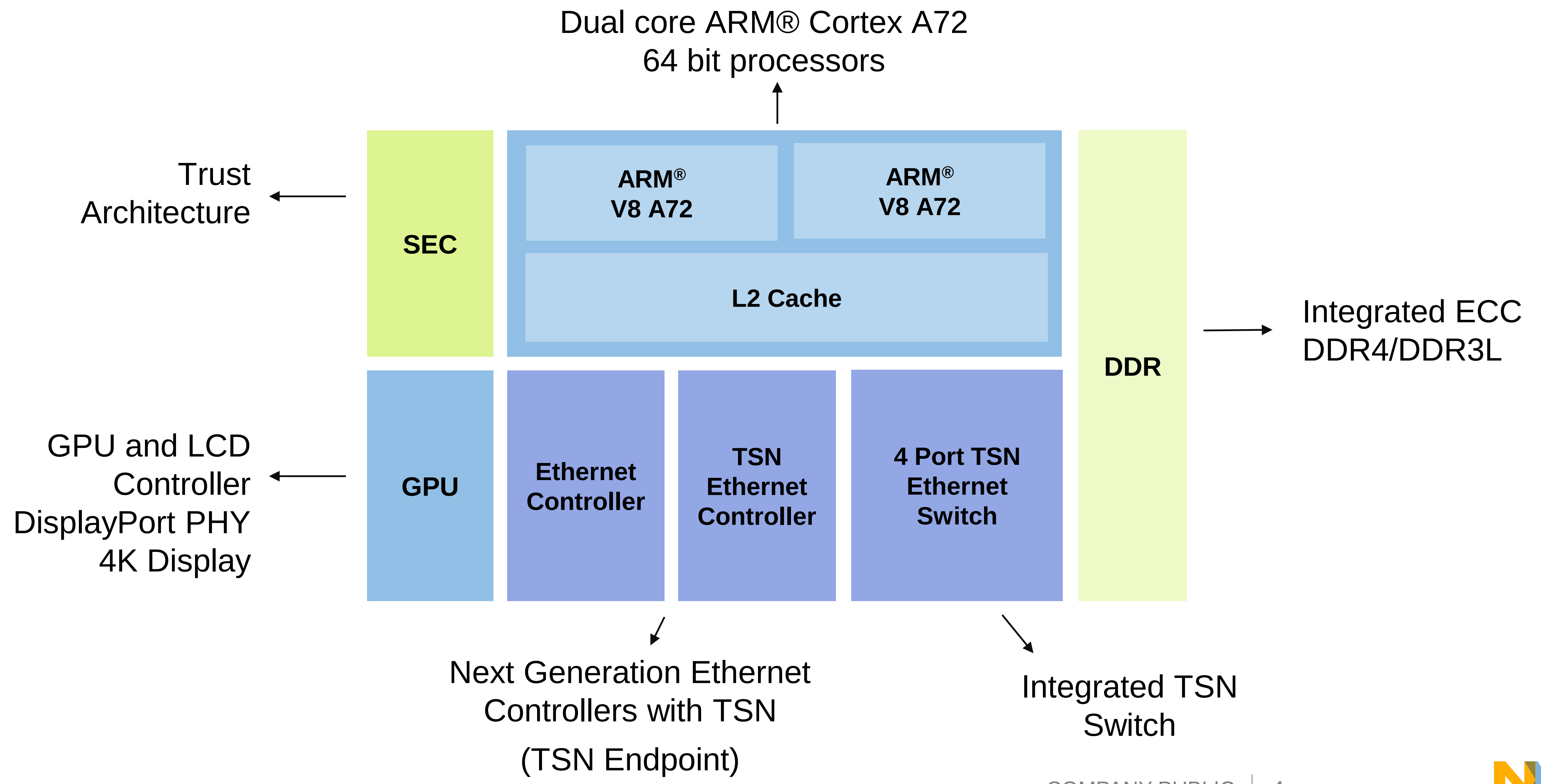




Key TSN Standards

Standard	Description
802.1Qbv	Scheduled Traffic
802.1Qav	Forwarding and Queuing Enhancements
802.1Qbu, 802.3BR	Frame Preemption
802.1CB	Frame Replication and Elimination for Reliability
802.1Qci	Per-stream filtering and policing
802.1AS	Timing and synchronization for Time-Sensitive Applications

Layerscape LS1028A – Industry Ready



LS1028A Reference Design

Memory and Storage

- 4 GB 32-bit DDR4 1300 MT/S
- 8 GB eMMC
- 256 MB XSPI NOR
- 512 MB QSPI NAND

Ethernet

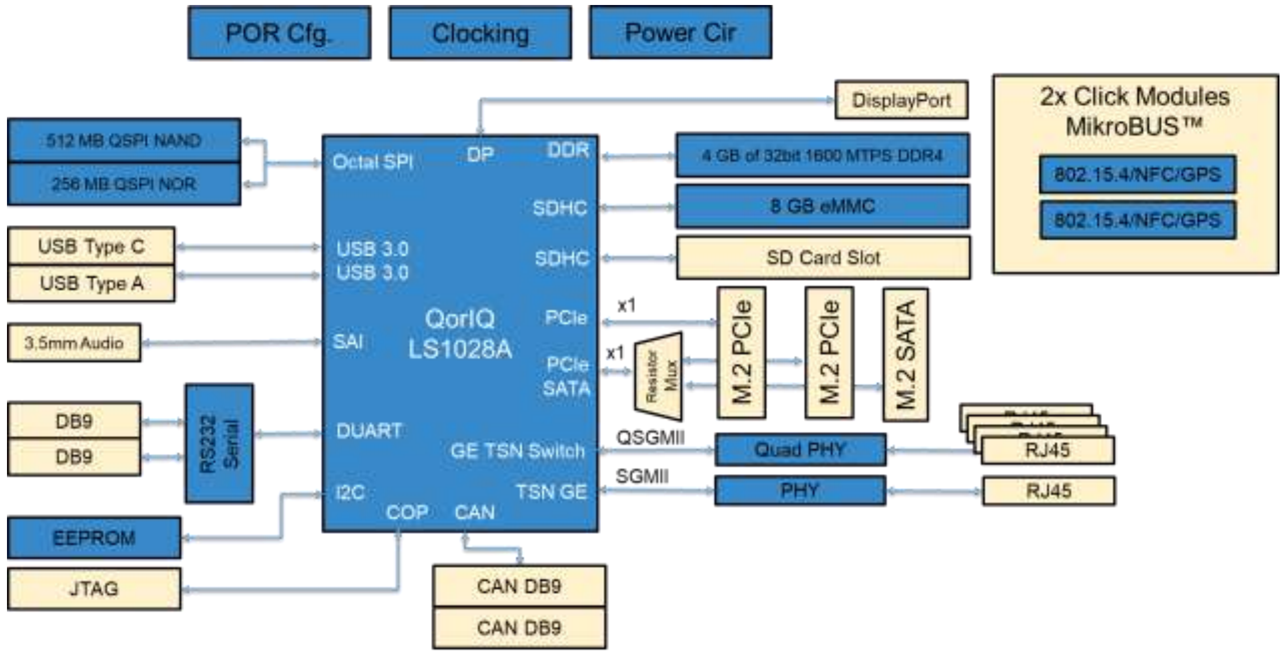
- 4 Switched Gb Ethernet with TSN
- 1 Gb Ethernet with TSN
- IEEE 1588 timing support

High Speed IO

- M.2 slot with PCIe Gen 3.0 x1
- M.2 slot with either PCIe Gen 3.0 x1 or SATA 3.0
- Type C USB 3.0
- Type A USB 3.0

Connectivity

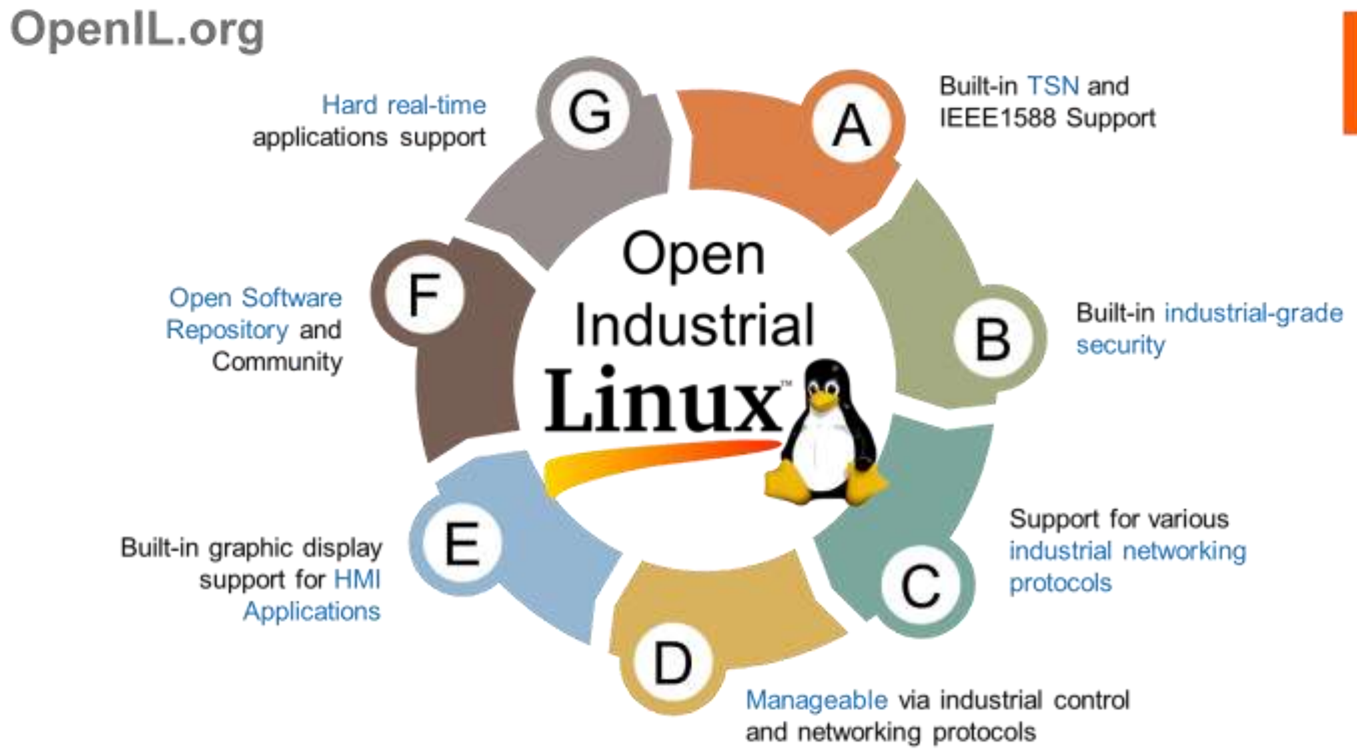
- mikroBUS™ socket for low speed wireless click boards™
- 2x CAN interfaces
- 2x RS232 Serial Ports
- SD Card slot
- Display Port



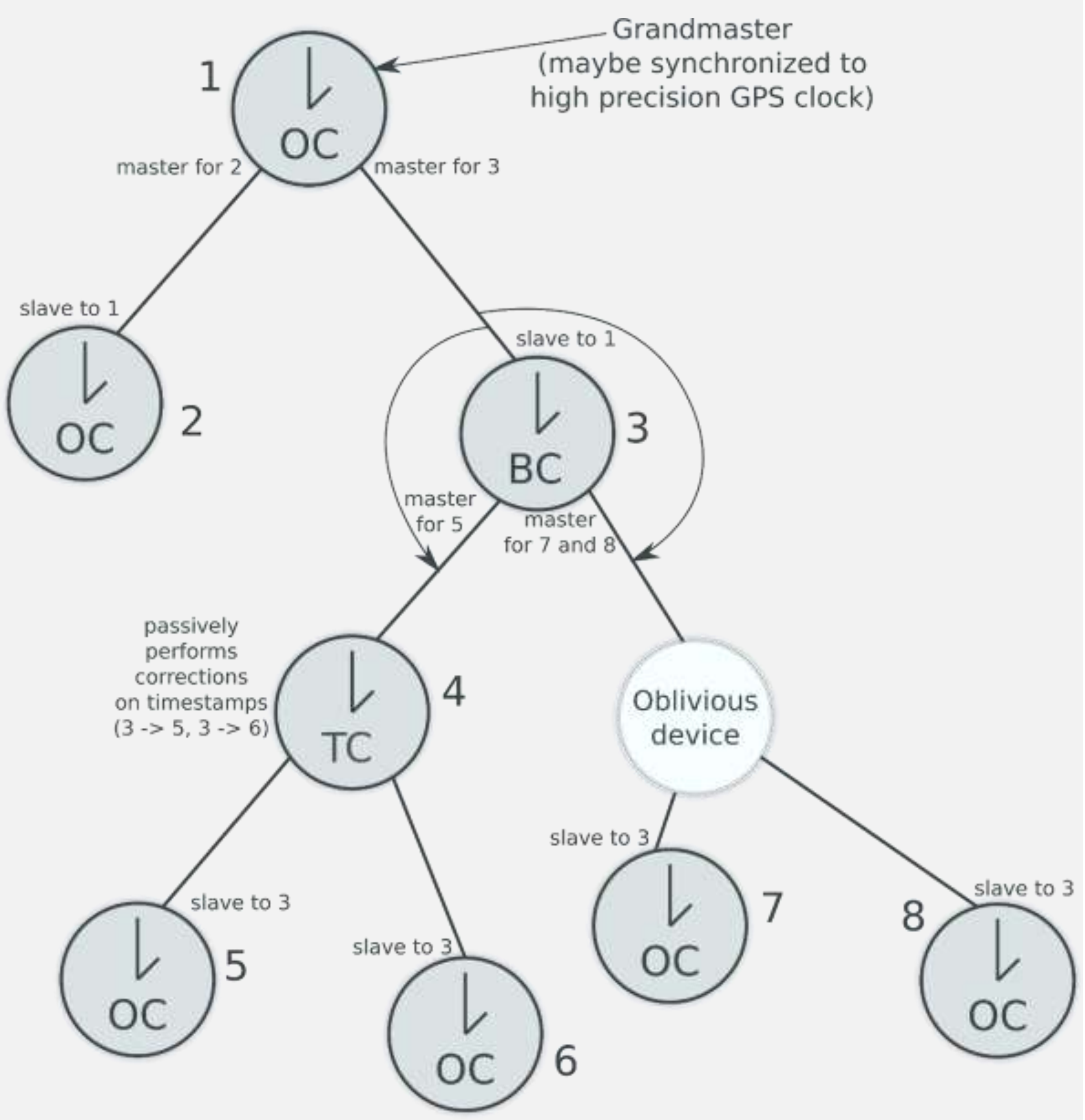
LS1028ARDB-PA Shipping Now! $895

OpenIL

Open Industrial Linux


OpenIL.org
Open
Industrial
Linux
G
Hard real-time applications support
A
Built-in TSN and IEEE1588 Support
B
Built-in industrial-grade security
C
Support for various industrial networking protocols
D
Manageable via industrial control and networking protocols
E
Built-in graphic display support for HMI Applications
F
Open Software Repository and Community

- Deterministic Computing
  Xenomai Linux
  Bare Metal Framework
- Open Source support for TSN
- Open source support for timing synchronization (1588v2, PTP)

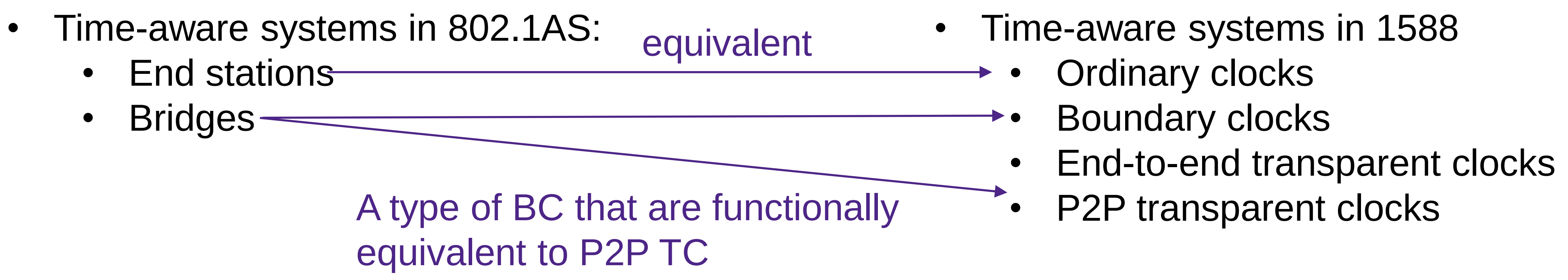


Precision Time Protocol (IEEE 1588/802.1AS)

- Clock distribution tree in an L2 network
- Point to point: if one link partner is master, the other(s) are slave
- Slave synchronizes its PTP Hardware Clock (PHC) to the master
- Ordinary Clock (OC), Boundary Clock (BC), Transparent Clock (TC)
- The basis of all other TSN protocols
- Has a quite broad specification - there also exist tighter “profiles” of it

Differences between PTP (1588-2008) and gPTP (802.1AS-2011)

- gPTP only defines L2 (Ethernet) as viable transport (1588 also permits L4 and others)



- Time-aware systems in 802.1AS:
  - End stations
  - Bridges

- Time-aware systems in 1588
  - Ordinary clocks
  - Boundary clocks
  - End-to-end transparent clocks
  - P2P transparent clocks

- Non-time-aware (oblivious) systems are **NOT** allowed in a gPTP network
- gPTP **requires** 2-step timestamps (1588 also allows 1-step timestamps)
- gPTP **requires** peer delay mechanism (1588 also allows end-to-end mechanism)
- All devices in gPTP network are **syntonized** (clocks running at the same rate)
- In steady state, there is only **a single active grandmaster** in a time-aware network. That is, there is only a single gPTP domain, whereas IEEE 1588 allows multiple overlapping timing domains.

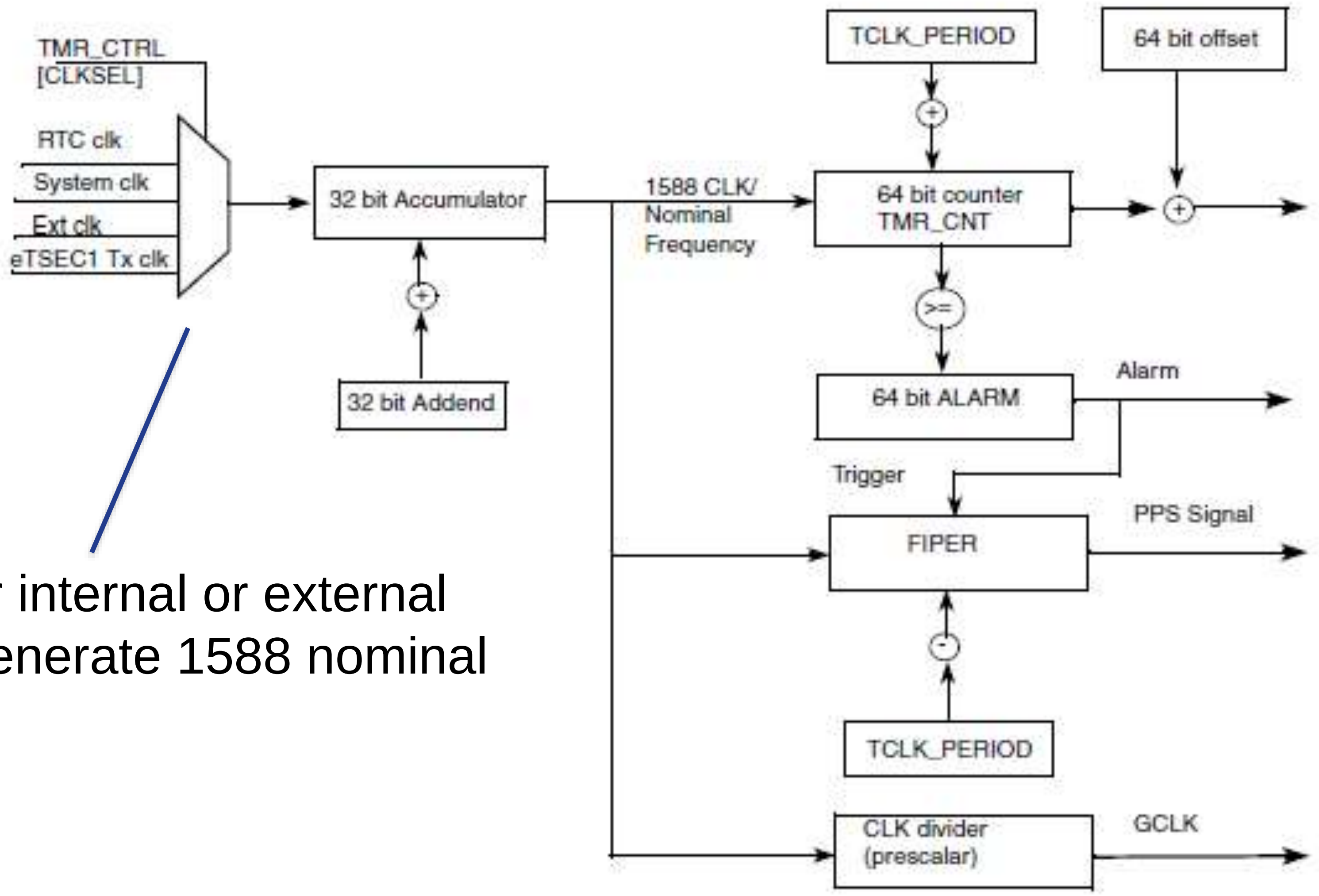
Differences between 802.1AS-2011 and 802.1AS-Rev

- Time-aware ~~**bridges**~~ **relays** (name change)
- gPTP **requires** 2-step timestamps (1588 also allows 1-step timestamps) **with an optional 1-step processing mode**
- ~~In steady state, there is only **a single active grandmaster** in a time-aware network. That is, there is only a single gPTP domain, whereas IEEE 1588 allows multiple overlapping timing domains.~~

ENETC - Precision Time Protocol (IEEE 1588/802.1AS)

- The driver supports primitives for the following operations in a manner compliant to the Linux kernel API for PTP:
  - Steering the PTP Hardware Clock: clock stepping and frequency adjustments
  - Timestamping PTP frames on reception and transmission
- Compatibility with the 1588 protocol and its various profiles is handled by the application stack (linuxptp)
- There is also ancillary circuitry which facilitates the use of LS1028 as a grandmaster device
  - PPS input may be used for high-precision synchronization to a GPS module

1588/PTP Clock Circuits Available in Layerscape SoCs



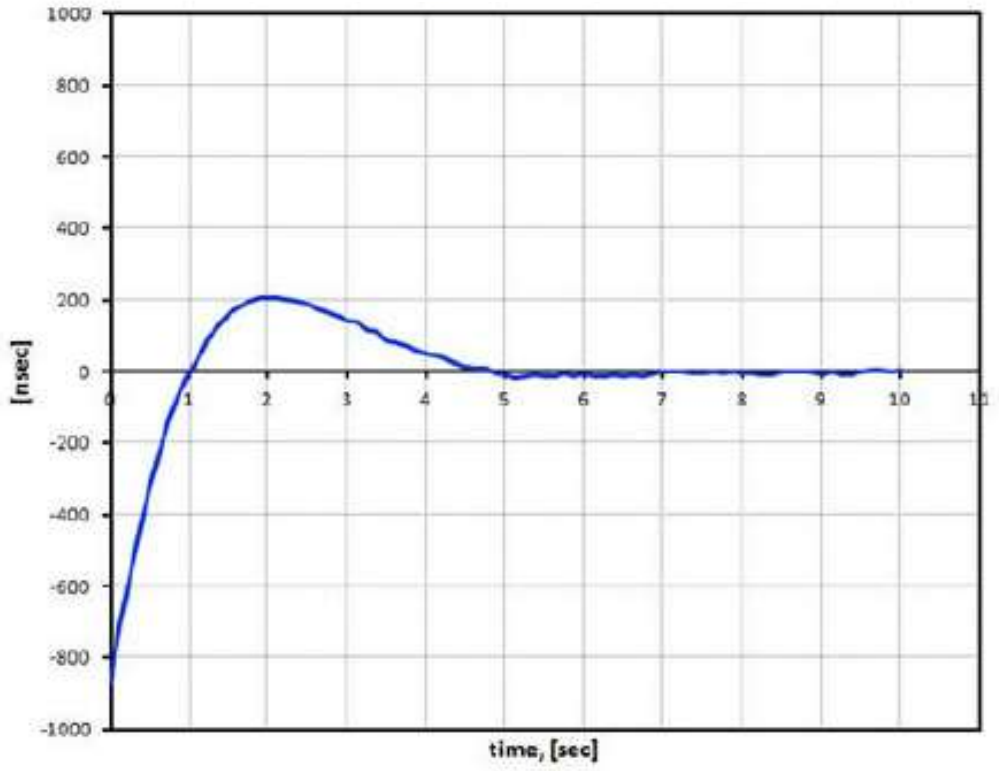
Timestamping value used for Tx and Rx packets

Generate phase aligned periodic pulses

Use either internal or external clock to generate 1588 nominal frequency

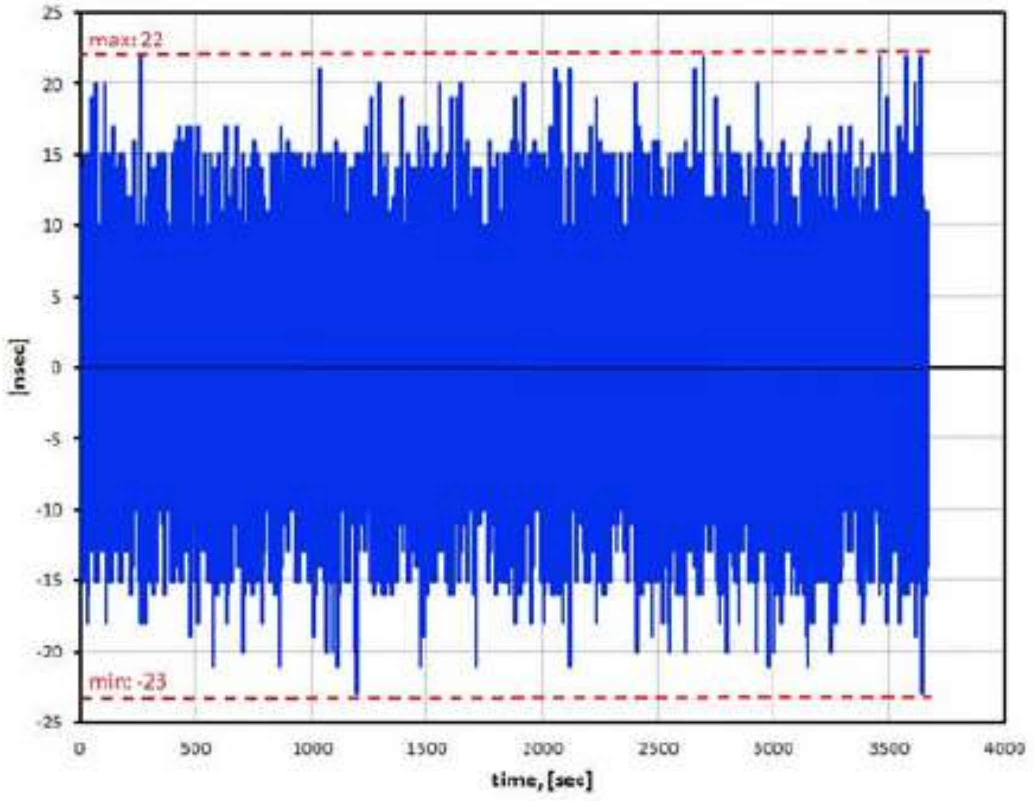
gPTP Performance

Offset from Master, Startup



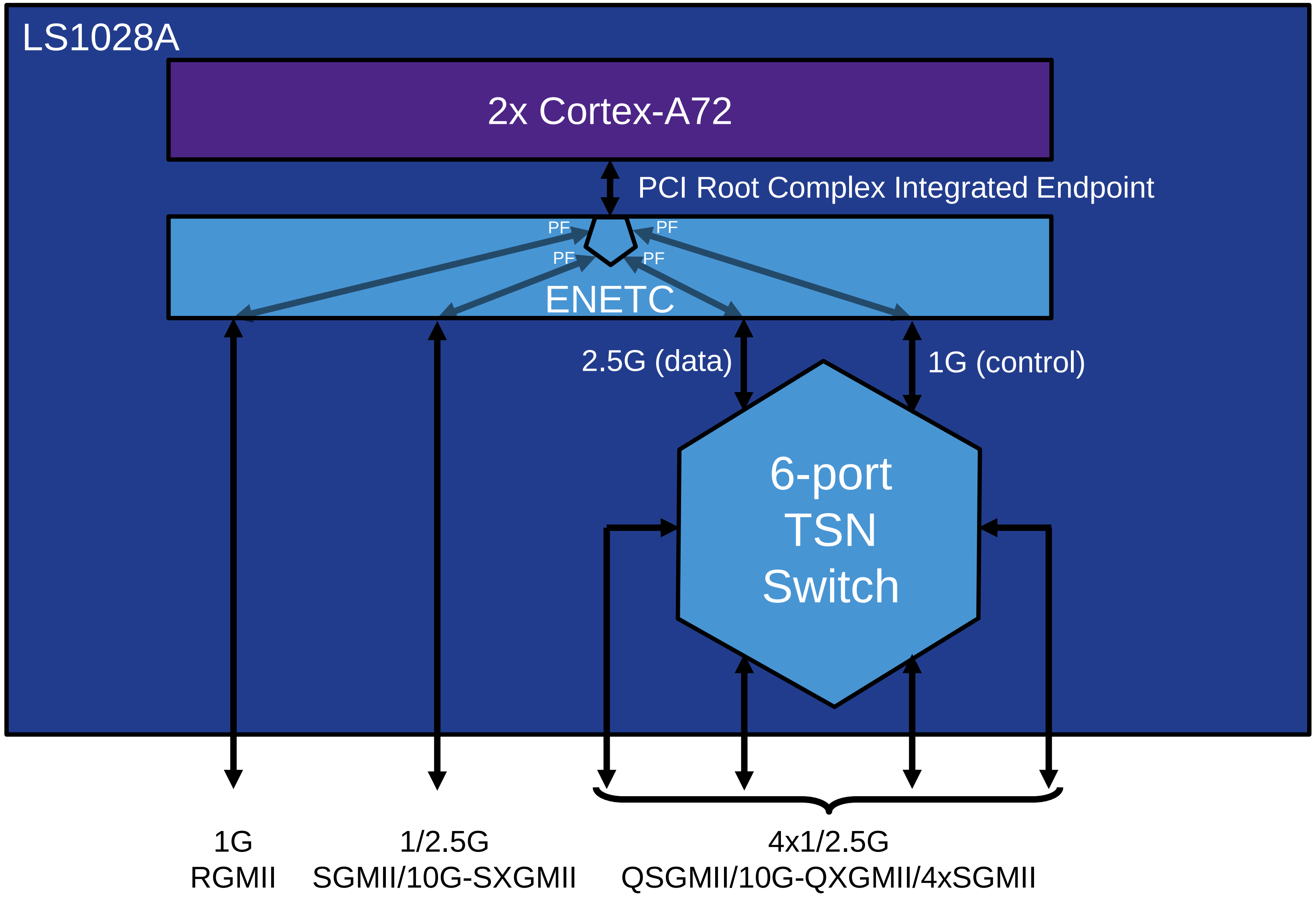
Timing settles within 5 seconds

Offset from Master, Stable State



Accuracy within ±23 nsec

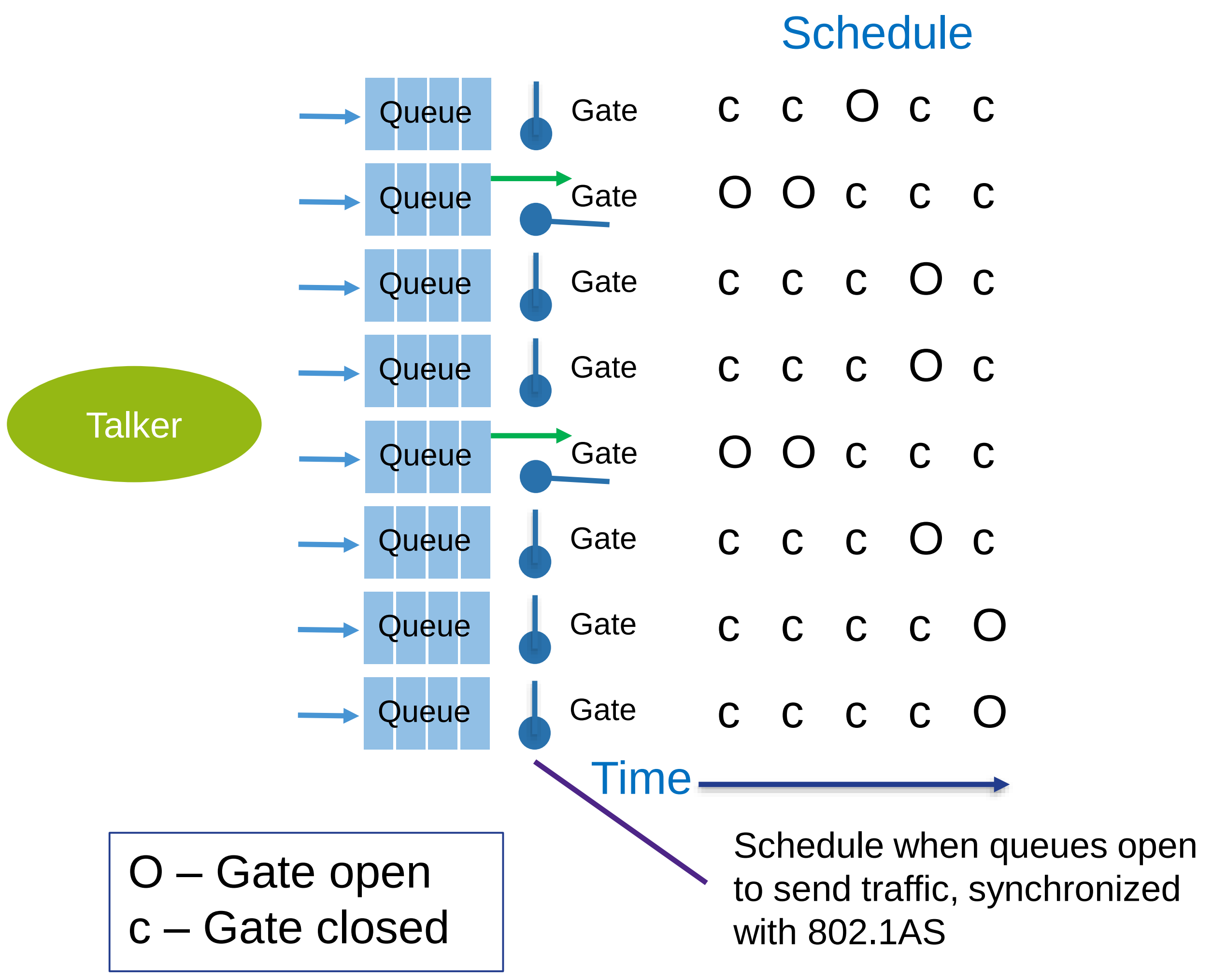
LS1028A Ethernet Solution



TSN Tool

- The tsntool program is a command-line interface over the libtsn. The latter facilitates netlink communication with a net device driver which implements a struct tsn_ops.
- Tsntool has an interactive and a non-interactive mode (prompt vs shell scripting)
  - For interfacing with other applications such as NETCONF server, it is recommended to call libtsn directly.
  - Libtsn API: https://github.com/openil/tsntool/blob/master/include/tsn/genl_tsn.h
- The following can be done with libtsn:
  - TSN capability discovery
  - Per-port TSN configuration using standard IEEE terminology
- For complete documentation of the tool, a separate PDF is available

802.1.Qbv – Time Aware Shaping



- Different priority traffic allocated for each queue
- Queue gate schedule synchronized to global time
- 8 Queues available

ENETC - Transmit queue selection

- The ENETC driver probes the hardware capability of each ENETC port in order to determine the number of available transmit and receive rings.
- For the eno0 interface, where 16 rings are available, the driver reserves only 8 for use, and 8 for the Virtual Station Interface (VSI).
- To view on which transmit ring a frame is steered to, the ethtool counters may be consulted:

```
$ ethtool -S eno0 | grep 'Tx ring'
Tx ring 0 frames: 0
Tx ring 1 frames: 6
Tx ring 2 frames: 28
Tx ring 3 frames: 1
Tx ring 4 frames: 0
Tx ring 5 frames: 1
Tx ring 6 frames: 70
Tx ring 7 frames: 2
```

ENETC - Transmit queue selection

- By default all transmit rings are of equal priority (0).
- The TSN features use the hardware priority, not the Tx ring number.
- In order to assign a priority to each transmit ring, one may install a root qdisc with a configurable priority mapping, such as tc-mqprio:

```
$ tc qdisc del dev eno0 root
$ tc qdisc add dev eno0 root handle 1: mqprio num_tc 8 map 0 1
2 3 4 5 6 7 hw 1
```

- Above is a one-to-one mapping between Tx ring index and hardware priority.
  - The Tx ring is selected based on skb (Linux packet) priority (configured by applications via the SO_PRIORITY socket ioctl API).
  - Any other priority mappings are possible.

ENETC - Transmit queue selection

- If applications do not use the SO_PRIORITY API, it is possible to attach tc filters on the root qdisc and edit their skb priority such that they exit the system on the correct transmit ring.

```
# https://www.tldp.org/HOWTO/Adv-Routing-HOWTO/lartc.qdisc.filters.html

$ tc qdisc add dev eno0 clsact

# ICMP is protocol 1 according to /etc/protocols, and 0xff is the matching mask
$ tc filter add dev eno0 egress prio 1 u32 match ip protocol 1 0xff action skbedit priority 5

# This matches iperf3 traffic sent to default port 5201
$ tc filter add dev eno0 egress prio 1 u32 match ip dport 5201 0xffff action skbedit priority 2

# This matches L4 PTP
$ tc filter add dev eno0 egress prio 1 u32 match ip dst 224.0.1.129/32 action skbedit priority 7
```

ENETC - Time Aware Scheduling (IEEE 802.1Qbv)

- The time gating standard requires ENETC to keep two sets of configurations (for no-downtime switching):
  - Operational (OPER): these parameters describe what is currently active in hardware (current and immutable)
  - Administrative (ADMIN): these parameters represent what software is programming during a reconfiguration (future and mutable)
- Upon reconfiguration, the hardware must wait until the currently running schedule is complete, and only then integrate the ADMIN parameters into OPER and start the new cycle according to OPER parameters.

ENETC - Time Aware Scheduling (IEEE 802.1Qbv)

- 802.1Qbv describes the existence of a cyclic schedule run by the egress port, known as the *Gate Control List* (GCL).
- The port executes the schedule starting from the *Base time* (a PTP time in the past or future).
- The beginning of each new schedule occurs at the Base time plus an integer multiple of the cycle length.
- The cycle length is usually the sum of the durations of all Gate Control List entries, or there can be an additional cycle length extension past that sum at the end of each cycle.
- Each Gate Control List entry has an action to perform on the egress gates (*open/close*), and a *delta time* until the next action in the list should be executed.

ENETC - Time Aware Scheduling (IEEE 802.1Qbv)

- A typical Qbv configuration is applied on an ENETC port using the following workflow:

```
$ cat > qbv0.txt << EOF
t0 00000001 1280000
t1 00000010 1280000
EOF
$ tsntool qbvset --device eno0 --entryfile qbv0.txt --enable
```

  - Create a GCL with two entries (t0 and t1). These names do not bear any significance.
  - The t0 GCL entry opens gate 0 and closes gates 1, 2, 3, 4, 5, 6, 7. It is active for 1.280.000 nanoseconds.
  - Then the t1 GCL gets invoked, which closes gate 0 and opens gate 1. The other gates are still closed. The active time for t1 is the same, then t0 is executed again by the ENETC port.

ENETC - Time Aware Scheduling (IEEE 802.1Qbv)

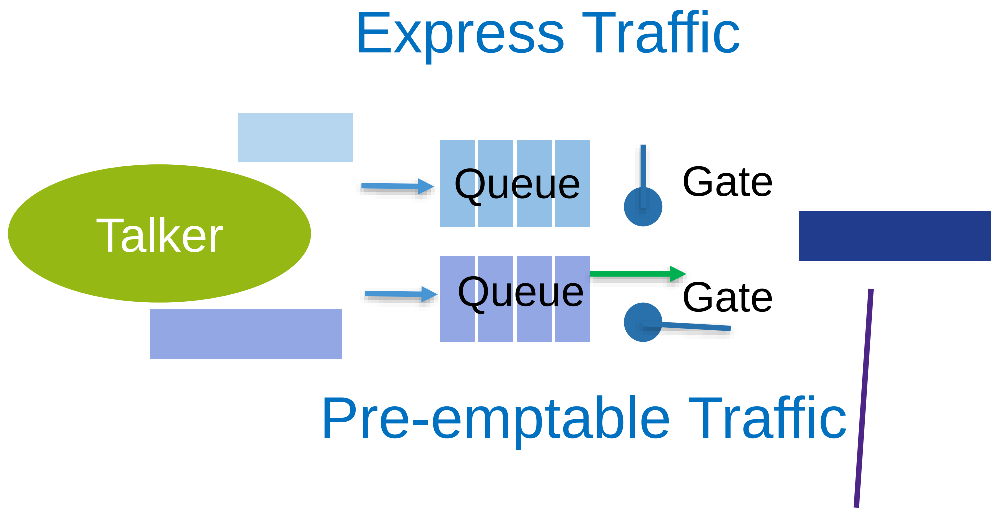
- Special consideration must be applied when choosing an active time for GCL entries.
- The Qbv scheduler has 2 properties when deciding that it can enqueue a frame for transmission:
  - If it cannot determine that there is enough time (based on frame size and link speed) to transmit this frame, it will not transmit it.
  - When cycling through the GCL entries, it will only look at a single entry in advance for the calculation of time reservations, and therefore will not fuse more than two GCL entries together.

ENETC - Time Aware Scheduling (IEEE 802.1Qbv)

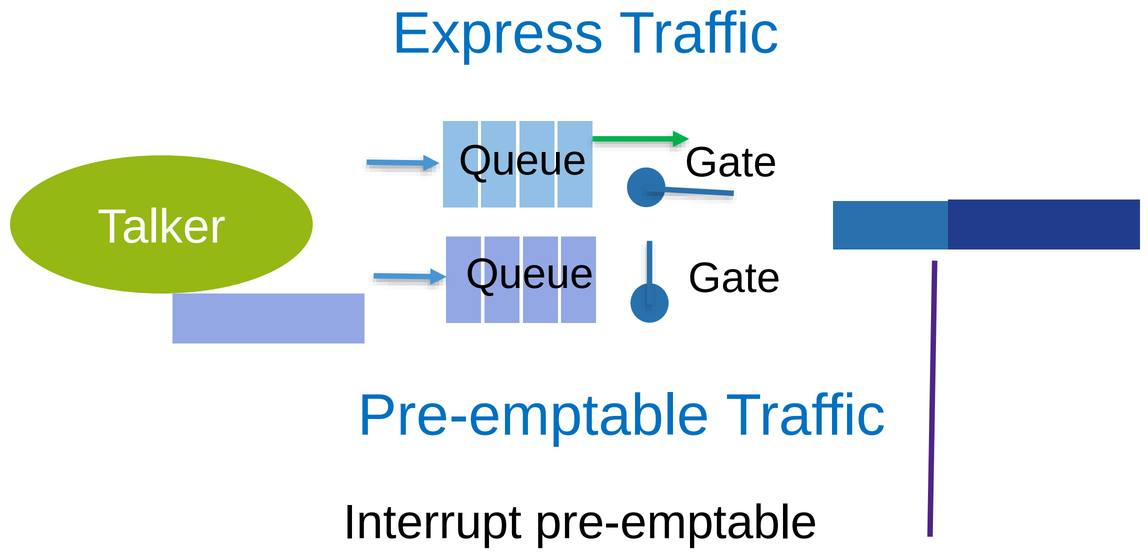
- Therefore frames that are larger than the minimum active time of a GCL should never be transmitted on an ENETC port where Qbv is active. If this happens, two expected outcomes are possible:
  1. The frame is dropped (and visible in the ethtool “Tx window drop” statistics counter).
  2. The frame is not dropped, and due to head-of-line blocking, the entire ENETC transmission ring is blocked waiting for the frame to be transmitted.
- The ENETC behavior for oversized frames for Qbv is configurable through the TG_DROP_DISABLE register.
- The current default is (1).

802.1Qbu – Frame Pre-emption

- Ensure zero delay for express traffic
- Efficient use of bandwidth for pre-emptable traffic
- Used with TAS, or stand-alone



Start to transmit frame of pre-emptable traffic



Interrupt pre-emptable frame with express frame. Will transmit remaining pre-emptable frame once express frame complete

ENETC - Frame Preemption (IEEE 802.1Qbu)

- Frame preemption may be enabled on ENETC as follows:

```
$ tsntool qbuset --device eno0 --preemptable $((1<<0 | 1<<3))
```

- This moves the Tx rings 0 and 3 to the preemptable MAC (all others will be dequeued by the express MAC).
- The MAC merge layer will then preempt a frame from ring 0 or 3, if a frame from a higher traffic class becomes available for sending.
- For debugging purposes, one may can check whether frame preemption is enabled by dumping bit 16 (Merge Enable) of MAC_MERGE_MMCSR:

```
$ tsntool regtool 0 0x11f00
```

ENETC - Frame Preemption (IEEE 802.1Qbu)

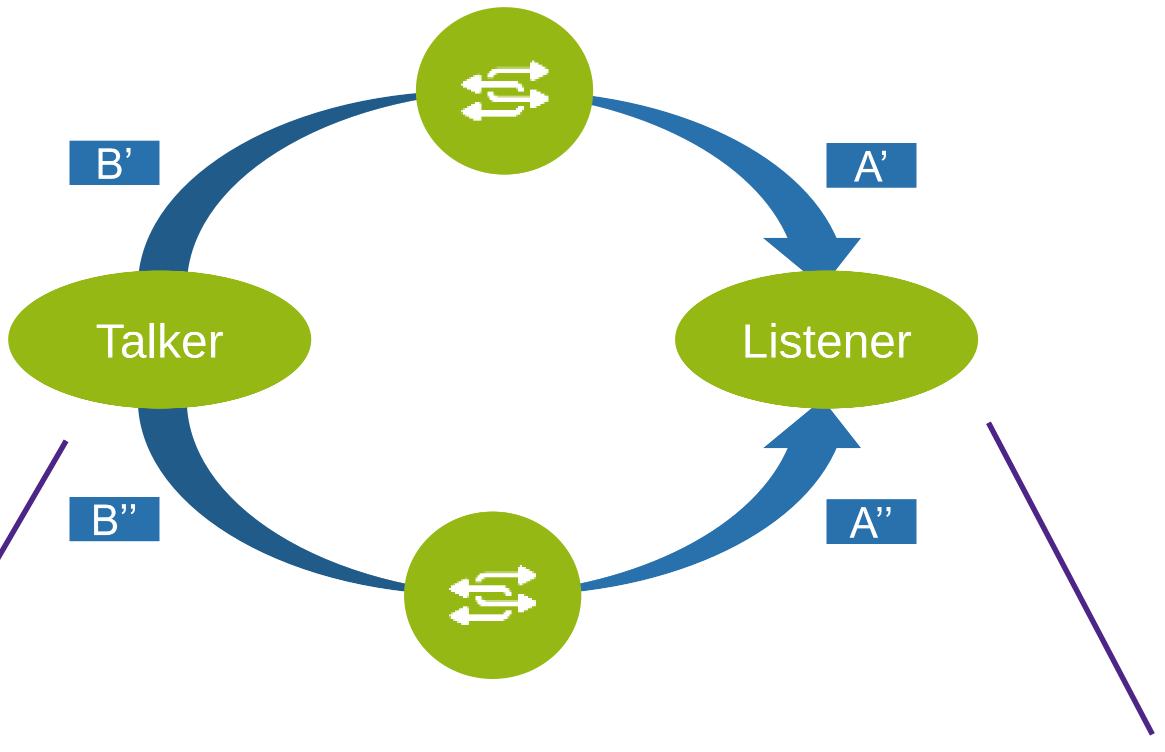
- In order for preemption to occur, the mqprio qdisc needs to be installed on the port (otherwise all Tx rings have equal priorities)
  - The same mprio 1:1 socket priority <-> Tx ring configuration from QoS classification can be used.
- For testing preemption, one can use the pktgen kernel infrastructure to start two kernel threads that inject frames into Tx rings 3 and 4 (a kernel with CONFIG_NET_PKTGEN=y is needed):

```
$ samples/pktgen/pktgen_twoqueue.sh -i eno0 -q 3 -n 0
```

- Note that only the base queue, 3, is specified in the command. From time to time, the kernel thread for queue 4 will preempt the one for queue 3 (3 is preemptable, 4 isn’t).
- To see how many frames were preempted, one can check an ENETC counter:

```
$ tsntool regtool 0 0x11f18
```

802.1CB – Frame Replication and Elimination for Reliability



- TSN hardware performs replication/elimination
- Zero time failover if 1 path fails
- No need for upper level retry mechanisms
- Simpler code base with reliability

Talker replicates Ethernet frames and sends over multiple paths to Listener

- Listener provides first Ethernet frame that arrives to application
- Listener removes duplicates

```
0         1         2         3         4         5         6
 +--------------+--------------+--------------+
 |  EtherType   |  Reserved    |   Sequence   |
 |    0xF1C1    |              |    number    |
 +--------------+--------------+--------------+
```

Frame Replication and Elimination for Reliability (IEEE 802.1CB)

- The IEEE standard for Seamless Redundancy provides increased reliability (reduced packet loss rates) by duplicating frames across multiple paths in a network, and annotating them with a 6-byte redundancy tag (RED_TAG).
- When the RED_TAG exists in a frame, it is always after found after VLAN tags, if those are present.

```
0       1       2       3       4
 +-------------+-------------+
 |    0x2345   | 16 bit seq  |
 +-------------+-------------+
```

Frame Replication and Elimination for Reliability (IEEE 802.1CB)

- Note that by default, the switch processes EtherTypes of 0x2345 as redundancy tags.
  - This is configurable through the RED_TAG_ETYPE_VAL register.
- It is also only 4 bytes in size. Therefore the tags understood by the switch look as seen on the left.

IEEE 802.1CB

- The LS1028A switch supports the following stream transformations:
  - Sequence generation and splitting: Inserting redundancy tags with sequence numbers into frames, and replicating them, when used as a relay system close to the traffic source.
  - Sequence recovery: Parsing sequence numbers from redundancy tags, removing the redundancy tags and eliminating the duplicates when used as a relay system close to the traffic destination.

IEEE 802.1CB - Stream Identification

- Clause 6 of 802.1CB defines 4 stream identification functions:

Stream identification function	Examines	Overwrites
Null	DMAC, VLAN ID	None
Source MAC and VLAN	SMAC, VLAN ID	None
Active Destination MAC and VLAN	DMAC, VLAN ID	DMAC, VLAN ID, VLAN PCP
IP	DMAC, VLAN ID, IP SRC, IP DST, DSCP, IP Next Proto, Source Port, Destination Port	None

- The LS1028A switch supports the Null Stream Identification.
  - This means that based on the MAC table (FDB rules), the switch may perform a lookup and associate {DMAC, VLAN} pairs with one of up to 128 Seamless Stream IDs (SSID).

IEEE 802.1CB - Modes of operation

- The following flows are understood by the LS1028A switch for 802.1CB:

1. *PKT NON_SEQ -> unknown SSID -> SEND*:
   - Frames do not contain a RED_TAG and therefore are switched as regular traffic
2. *PKT SEQ -> unknown SSID -> SEND:*
   - The switch parses and finds the RED_TAG but does not find a matching Seamless Stream ID configured based on the {DMAC, VLAN} pair found in this frame. Therefore these redundant frames are simply transiting this switch and they are forwarded following the normal L2 rules.
3. *PKT NON_SEQ -> known SSID -> SEQ -> SPLIT -> SEND:*
   - The switch receives regular traffic which matches a Seamless Stream ID. So it pushes the configured RED_TAG, increments the sequence number for this stream, replicates the frame on the configured egress ports, and sends them out.

IEEE 802.1CB - Modes of operation

4. *PKT SEQ -> known SSID -> SPLIT -> SEND:*
   - The switch receives traffic that already contains a RED_TAG. It can further split the stream into multiple redundant paths. This is for situations such as “Ladder redundancy” which are covered by the 802.1CB standard.
5. *PKT SEQ -> known SSID -> DISC (POP RED_TAG) -> SEND:*
   - In this mode, the switch performs sequence recovery. It compares the sequence number of received frames with its internal sliding window for this Seamless Stream ID. If the sequence number is too far back in the window’s history, the frame is discarded. Otherwise, the switch marks this frame as received within its current sliding window, optionally advancing it, and then it sends the frame. The redundancy tag is stripped in this mode.
6. *PKT SEQ -> known SSID -> DISC -> SEND:*
   - Same mode as 5, except that the redundancy tag is not removed from frames on egress.

```
Board 1:

 +---------------------------------------------------------------------------------+
 |                                                                                 |
 | +------------+   +------------+  +------------+  +------------+  +------------+ |
 | |            |   | To Board 3 |  | To Board 2 |  |            |  |            | |
 | |            | +-|    SW0     |  |    SW1     |  |            |  |            | |
 | |            | | |            |  |            |  |            |  |            | |
 +-+------------+-|-+------------+--+------------+--+------------+--+------------+-+
        MAC0      |      SW0             SW1             SW2              SW3
                  |                       |
 Board 2:         |                       |
                  |                       |
 +----------------|-----------------------+----------------------------------------+
 |                |                       |                                        |
 | +------------+ | +------------+  +------------+  +------------+  +------------+ |
 | |            | | |            |  | To Board 1 |  | To Board 3 |  |            | |
 | |            | | |            |  |    SW1     |  |    SW2     |  |            | |
 | |            | | |            |  |            |  |            |  |            | |
 +-+------------+-|-+------------+--+------------+--+------------+--+------------+-+
        MAC0      |      SW0             SW1             SW2              SW3
                  |                                       |
                  |                                       |
 Board 3:         |                                       |
                  |                                       |
 +----------------|---------------------------------------+------------------------+
 |                |                                       |                        |
 | +------------+ | +------------+  +------------+  +------------+  +------------+ |
 | |            | | | To Board 1 |  |            |  | To Board 2 |  |            | |
 | |            | +-|    SW0     |  |            |  |    SW2     |  |            | |
 | |            |   |            |  |            |  |            |  |            | |
 +-+------------+---+------------+--+------------+--+------------+--+------------+-+
        MAC0             SW0             SW1             SW2              SW3
```

IEEE 802.1CB - Usage example

- Send a ping stream from Board 2 to Board 1 through two paths:
  - Directly through B2.SWP4 -> B2.SWP1 -> B1.SWP1 -> B1.SWP4
  - Through Board 3: B2.SWP4 -> B2.SWP2 -> B3.SWP2 -> B3.SWP0 -> B1.SWP0 -> B1.SWP4

TSN Benefits

- Use Time-Aware Shaper and Frame Preemption to provide determinism
- Use Frame Replication and Elimination to create resilient networks
- LS1028A and Open Industrial Linux enable Time-Sensitive Networking

SECURE CONNECTIONS
FOR A SMARTER WORLD