Implementing Real-Time System Using Intel Time-Sensitive Networking Capable Ethernet Controller on Linux Operating System

Table of Contents:

Ong Boon Leong (Intel Corporation)
Principal Software Engineer

Gunnar Lessmann (Phoenix Contact)
Master Specialist Profinet

Overview of IEEE TSN standards

The Internet has been around for decades and the range of applications that are powering it has grown tremendously from simple web pages with text and pictures, Internet Relay Chat to on-demand streaming of audio/video contents, Voice over Internet Protocol telephony and two-way interactive video calls. The need for networking bandwidth too has sky-rocketed from Fast Ethernet to Gigabit Ethernet (1G), 10G, 40G, 100G and beyond in the data centers backhaul. Ethernet technology was originally designed to provide best effort delivery for lightly loaded networks and has over time evolved into prioritizing media streaming traffic over other traffics. The dawn of Internet of Things and market movements such as Industry 4.0 (“Smart Factory”) and autonomous vehicles are driving Ethernet technology to provide data transfer in a reliable and timely manner. Such is known as Time-Sensitive Networking and its acronym “TSN” word has been a recent buzz word in Ethernet Technology domain.

TSN consists of multiple mechanisms specified in IEEE 802.1 covering: (a) time synchronization, (b) bounded and low packet delivery latency, (c) network resource management and (d) reliability. In layman’s terms, TSN enables networked applications to interact in a deterministic fashion. The adoption of TSN is critical as TSN enables converged network for deterministic OT and best-effort IT applications that can co-exist without interference.

Overview of Time Synchronization

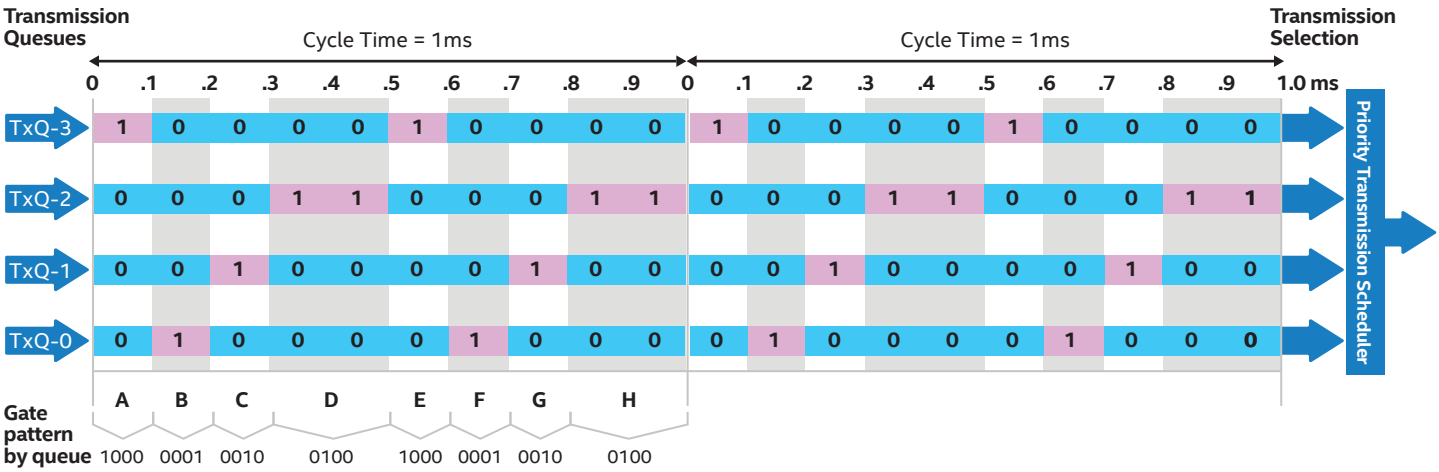
Foundational to real-time networked application is the capability to time synchronize the local clock that drives the real-time operations within these networked devices. IEEE Std. 802.1AS-2011, aka. generalized Precision Time Protocol (gPTP) enhances the accuracy of time synchronization between two networked nodes from millisecond (achievable by Network Time Protocol (NTP)) to microsecond or sub-microsecond. This is made possible as the PTP messages are time-stamped at the network layer instead of the software layer in the case of NTP. As the use of gPTP increases, more enhancements such as Fine Timing Measurement for IEEE 802.11 transport, one-step processing, faster grandmaster change over and reduction of Best Master Clock Algorithm (BMCA) converge time are added to IEEE Std. 802.1AS-2020. In addition, the support of multiple clock domains (wall and working clock) and redundant clock synchronization added to IEEE Std. 802.1AS-2020 are important enhancements for mission-critical applications.

Overview of Bounded and Low Packet Delivery Latency

To prioritize one traffic streams over another, IEEE Std. 802.1Q-2005 adds the capability to mark traffic streams in the network using the Priority Code Point (PCP) field of the Virtual Local Area Network (VLAN) header located at the start of a network packet. As prioritized traffic flows through an inter-connected mesh of network bridges, congestion may still happen on heavy-loaded networks. Such a situation leads to unbounded packet delivery latency from one end-point to another and is not desirable for real-time networked applications.

In networking, an Ethernet frame must be fully transmitted together with its checksum before the next frame can be sent.

The worst-case packet delivery latency is seen if a short high priority network packet is blocked by a very long lowest priority network packet. To overcome such scenario, IEEE Std. 802.1Qbv-2015 adds the functionality to time-control (time-aware shaper) which traffic streams could be selected for transmission within an Ethernet MAC controller through the addition of programmable transmission gate for each transmission queue. The transmission gate operates in cyclic fashion (synchronized to gPTP working clock), opening and closing, according to a user-defined pattern known as gate control list. An example of gate control list and the availability of traffic transmission patterns (open = blue, close = red) is shown below in Figure 1.



Each time interval has a pattern saying which queue is open or closed during that interval:

Open/closed patterns
A. 1000 (0x8) for first 0.1ms
B. 0001 (0x1) for next 0.1ms
C. 0010 (0x2) for next 0.1ms
D. 0100 (0x4) for next 0.2ms
E. 1000 (0x8) for next 0.1ms
F. 0001 (0x8) for next 0.1ms
G. 0010 (0x2) for next 0.1ms
H. 0100 (0x4) for next 0.2ms

Each pattern + interval becomes an entry in the gate control list on board A and the switch, with intervals in nanoseconds:

Gate control list (GCL)
 # SetGates 0x8 100000
 # SetGates 0x1 100000
 # SetGates 0x2 100000
 # SetGates 0x4 200000
 # SetGates 0x8 100000
 # SetGates 0x1 100000
 # SetGates 0x2 100000
 # SetGates 0x4 200000

The traffic schedule defined by the GCL is repeated every 1ms. In each time interval, traffic for a queue is allowed through based on whether the gate for that queue is configured to be open or closed during the interval.

Open (1) Closed (0)

Figure-1: An Example of Gate Control List for IEEE 802.1Qbv Time-Aware Shaper

IEEE Std. 802.1Qbv-2015 Time Aware Shaper is beneficial in creating transmission windows solely for cyclic traffics and other transmission window slots for best-effort traffics shown in Figure 2. To avoid any delay in cyclic traffic transmission, guard bands are typically set to the maximum packet size (a.k.a. Maximum Transmission Unit (MTU)) allowed in the network to prevent the creeping of best-effort traffic into the transmission window belonging to cyclic traffic. However, doing such does come with the penalty of losing network bandwidth utilization due to the guard band.

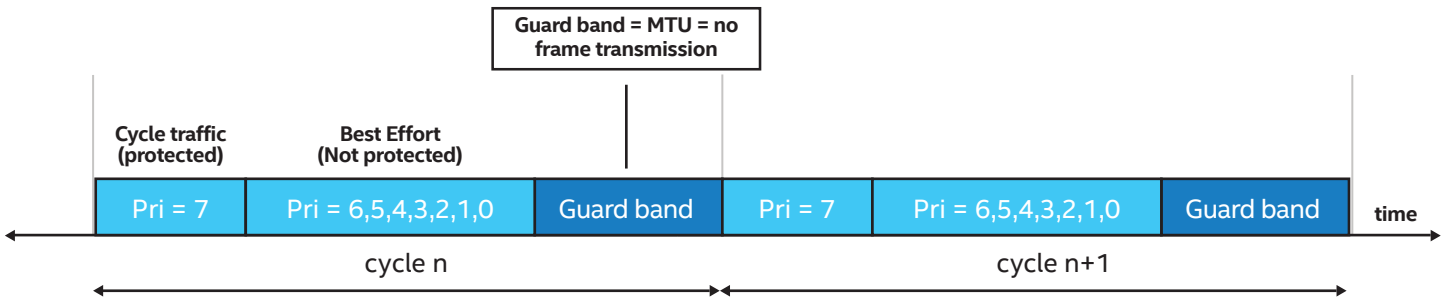


Figure-2: An Example of Gate Control List for IEEE 802.1Qbv Time-Aware Shaper

Bandwidth loss in guard band could be reduced by applying frame preemption technology defined by IEEE Std. 802.3br-2016 whereby the transmission of best effort traffic (aka preemptable frame) is paused momentarily to give way for high priority cyclic traffic (aka express frame). The transmission of the remaining packet of the preemptable frame is resumed immediately after the transmission of the express frame. To ensure preemptable packet fragments are not seen as corrupted Ethernet frames, IEEE Std. 802.3br-2016 adds new start frame delimiter (SMD) definitions (SMD-Sx and SMD-Cx) to the original Ethernet frame format as shown in Figure-3 below. In addition, each of the Ethernet fragments (of the preemptable frame) has its own check-sum (aka mCRC) for frame integrity purposes.

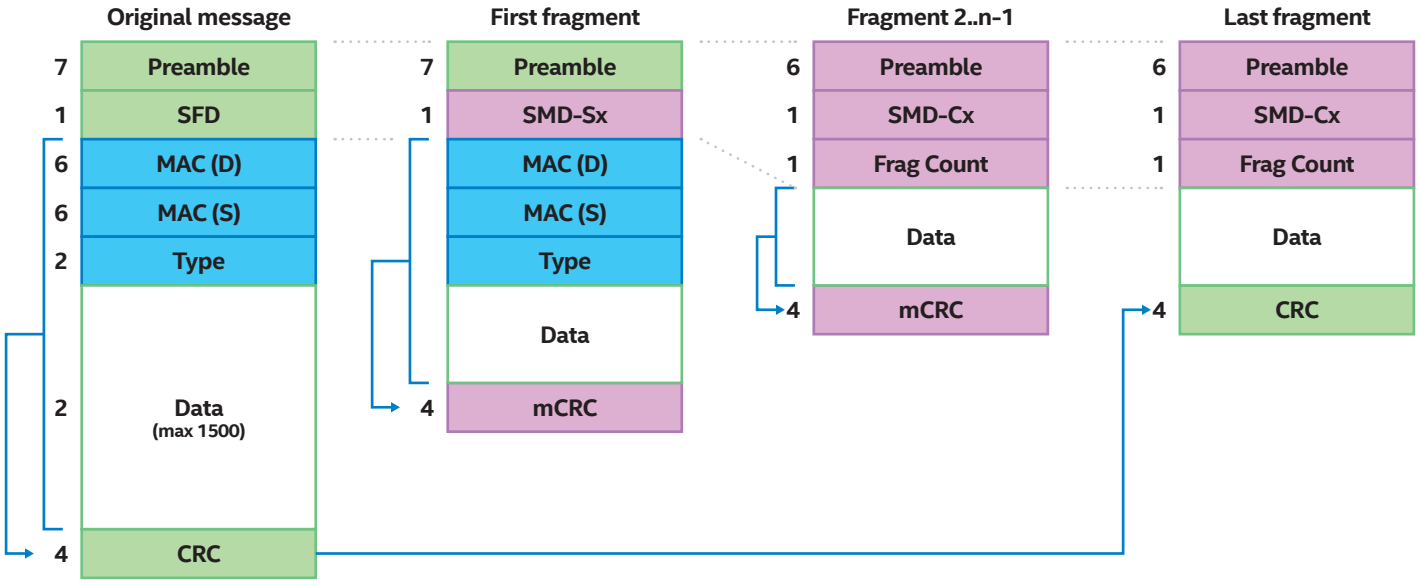


Figure-3: Preempted frame that is broken into numerous fragments under IEEE Std. 802.3br

IEEE Std. 802.1Qbu-2016 adds frame preemption fine-tuning control to the IEEE 802.1Qbv-2015 gate control list commands, i.e. hold or release the transmission of preemptable frames within a transmission window. In addition, the mapping of application traffic streams to either express frame or preemptable frame transmission queue is also added by IEEE Std. 802.1Qbu-2016. Figure 4 shows the effect of express frames (E1 to E5) that preempt best-effort preemptable frames into fragments that continue their transmission on the next cycle time. Clearly, there is no loss of network bandwidth during the guard band as the interval are filled with fragments from best-effort preemptable frame.

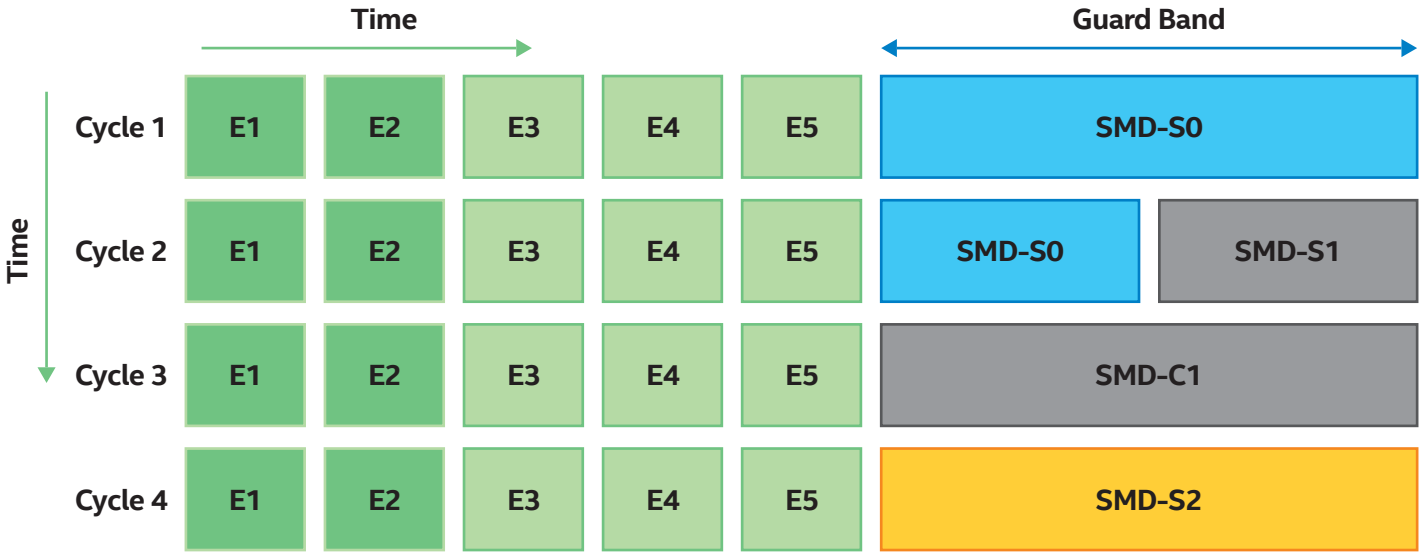


Figure-4: Preempted frame that is broken into numerous fragments under IEEE Std. 802.3br

Overview of Linux Preempt-RT and TSN software interface in Linux

Figure 5 shows a real-time application model where a control function has to be executed and completed at a specific time within the application cycle-time reliably. As application cycle time is aligned to network cycle time, the application is categorized as isochronous real-time application. The control function processes the input data that arrives at the start of the cycle time (within the transfer time with safety margin) and the output of the computation is transmitted at the start of the next network cycle. At the network level, cyclic real-time traffics are allocated network bandwidth slots in cyclic fashion using TSN technologies discussed above. Best effort traffics are exchanged after control function execution has started in the application processor and the exchange of best effort traffic must end before the start of the next network cycle.

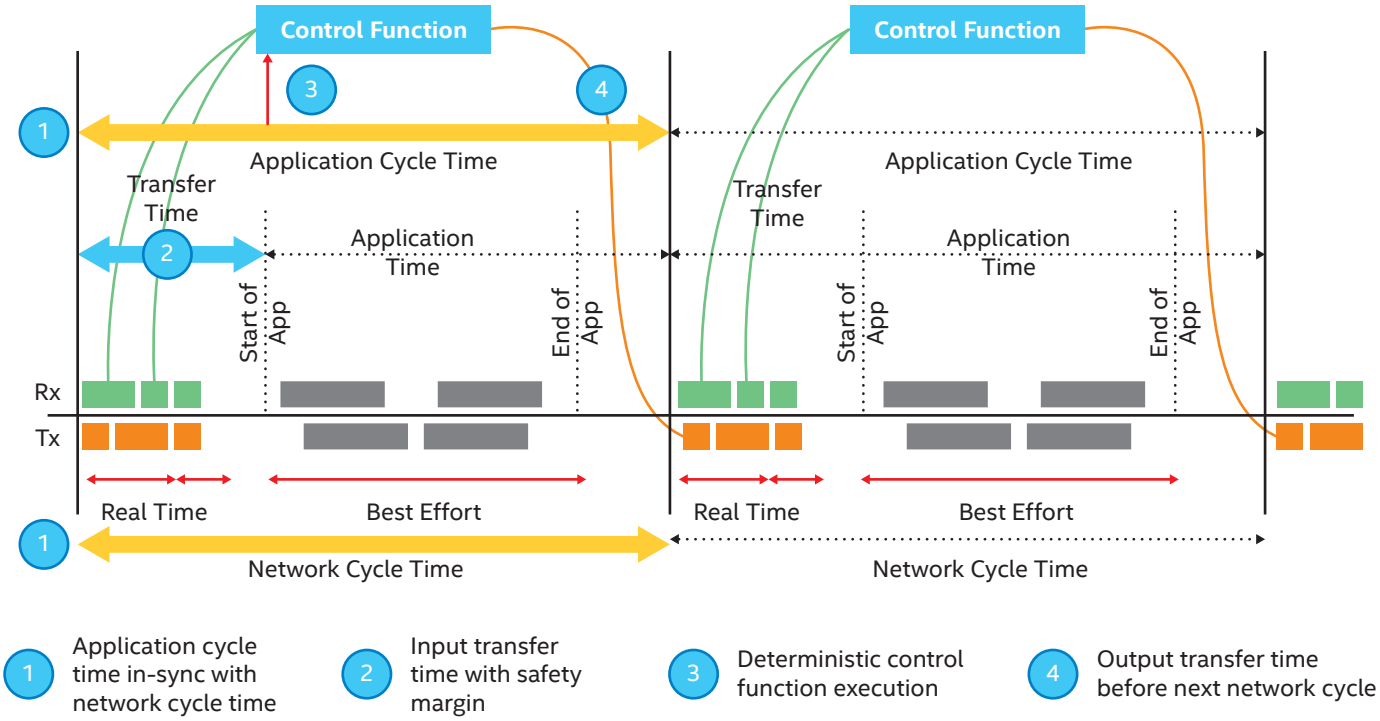


Figure 5: Isochronous Real-Time Control Loop Application

Preempt real-time (PREEMPT_RT) patches that transform Linux from general-purpose operating system (OS) to real-time OS has been around for decades. In recent years, significant advancement has been made in merging these patches as part of the Linux kernel project making the adoption of Linux-based real-time systems easier than before. With Linux preempt-RT support, process scheduling has become bounded and low in latency, often in lower 2-digit micro-seconds range, enabling real-time applications to be supported on Linux kernel. Low and bounded process scheduling determinism is important in fulfilling real-time application processing as shown in Figure-5. In an actual real-time system, numerous real-time applications/processes are running concurrently with independent real-time traffic streams flowing through the same Ethernet controller. Therefore, in a Linux-based system running on multiple processing CPU cores, it is important to assign dedicated CPU cores for real-time workloads. Figure 6 shows an example of two CPU cores that are solely used for real-time processing.

The Linux PTP project offers two important utilities, namely ptp4l and phc2sys. To synchronize network time between two stations with bounded two digit nano-seconds accuracy, ptp4l utility calculates the path delay, rate ratio and correction time using hardware time-stamping of PTP messages that are exchanged between them. The phc2sys utility is used to synchronize OS system clock to network clock and could achieve similar two digit nano-seconds accuracy if the underlying network controller is capable of cross time-stamping its PTP hardware clock and the CPU local clock. Traffic shapers such as Time Aware Shaper and Frame Preemption are hardware capabilities inside Ethernet controllers used to ensure network traffic entering the TSN network in a time-coordinated and deterministics manner. Since 2017, intel has been working closely with Linux community to drive standardized application programming interface (API) in Linux mainline for configuring TSN-related traffic shapers through Linux traffic control/queue discipline (QDisc) interfaces as listed below:

- CBS QDisc for IEEE 802.1Qav Credit Based Shaper.
- TAPRIO QDisc for IEEE 802.1Qbv Time Aware Shaper.
- ETF QDisc for ensuring transmit packets from multiple streams are reordered in correct chronological order.

The QDisc configuration is set through user-space traffic control (tc) utility. Tunable parameters related to Ethernet controllers such as number of DMA channels and RX/TX interrupt coalesce controls are configured using a user-space utility called ethtool. Both tc and ethtool utilities encapsulate and pass configuration parameters into Linux kernel via AF_NETLINK socket (not shown in Figure-6 for simplicity reason).

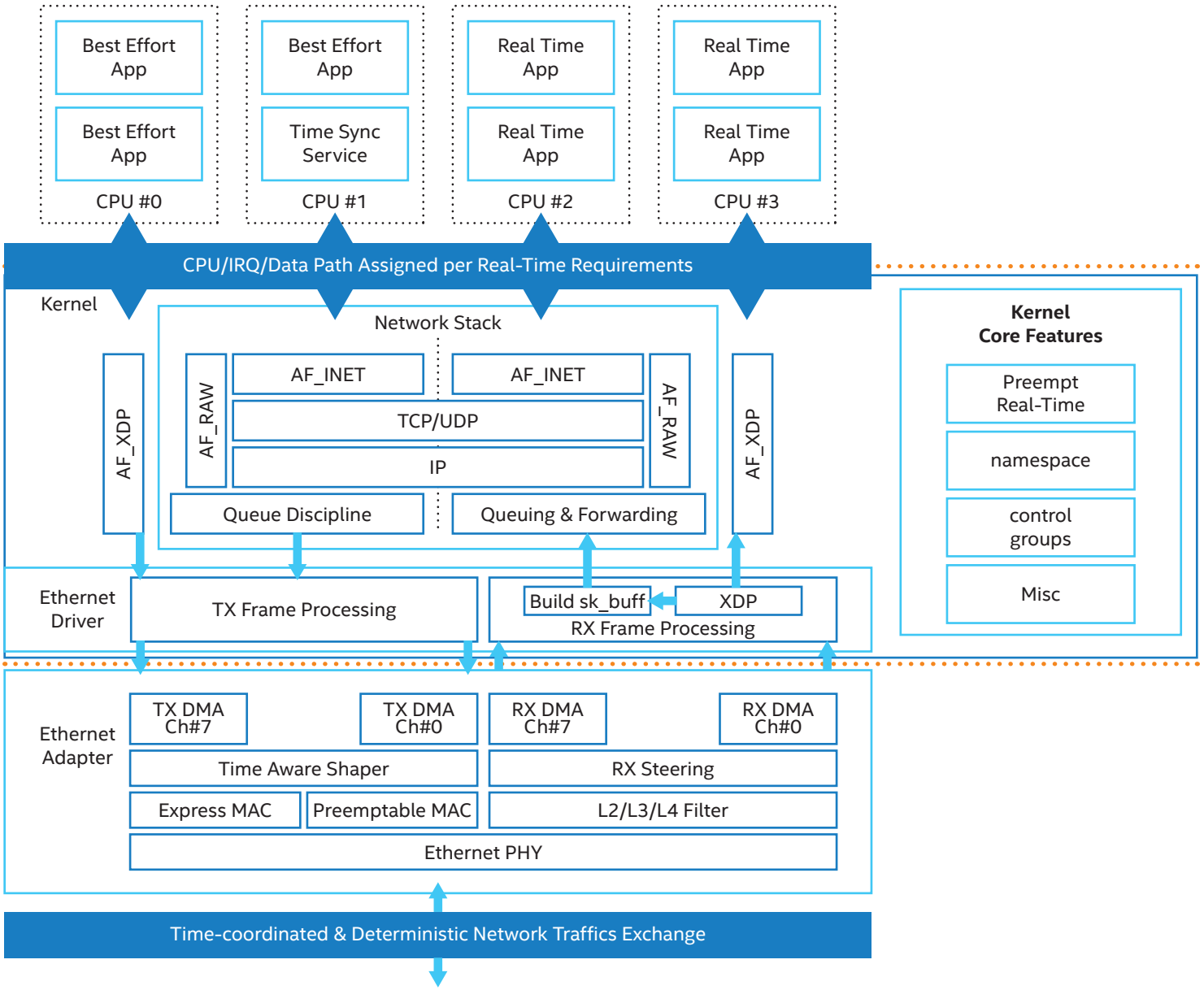


Figure 6: Mixed Criticality Applications in Linux preempt-RT based Kernel Serviced by Time-Sensitive Networking Capable, Multi-queue Ethernet Network Controller.

In Linux systems, networked IT applications send and receive network packets through socket interfaces such as AF_INET, AF_INET6 and AF_PACKET that offer feature-rich transport and network layer processing inside the kernel. Traffic that flows through the network stack is subject to a traffic throttling mechanism called traffic control/queue discipline subsystem that presents between network stack and network driver as shown in Figure-6. In gist, Linux network stack enables sharing of network link for multiple applications at the same time and by nature does not focus on achieving real-timeness in the packet processing.

In recent years, Intel has been closely working with the Linux community to advance a new type of high-performance packet processing framework that adds a hook in network driver known as Express Data Path (XDP). The hook allows an early packet parsing program (using eBPF technology) on arriving packets before the packets are passed to the network stack. The verdict of the packet processing is of one of the following options:-

- XDP_PASS: Pass the packet to Linux network stack. This involves building a new socket buffer (sk_buff) to store the received packet.
- XDP_DROP: Drop the packet
- XDP_TX: Re-transmit the packet on the same network driver
- XDP_REDIRECT: Redirect the packet to another CPU, network driver or socket.

The ability to by-pass network stack and redirect packets directly to application socket reduces time spent inside Linux kernel and is very much desirable for both high throughput and real-time applications. This class of socket is known as AF_XDP (Express Data Path) socket interface with Zero-Copy (XDP ZC) mode. Under zero-copy mode, application-level frame buffers are pinned to a specific RX DMA channel of the underlying network controller for receiving incoming packet directly without involving extra data copy when the packet moves from kernel space to user space.

In Linux, network packets are processed in batch fashion through a packet processing technique called NAPI (New API) to reduce context switching due to packet arrival interruption. Interruption from the network controller (often per DMA channel) is disabled when the 1st packet arrives and all subsequent packets from the same RX queue are processed in kernel process (polling mode). At the end of batch packet processing, the interrupt from the network controller is

reenabled. Inside the NAPI framework, there are techniques used to keep the polling mode as long as possible (busy polling with limit control) and improve polling process scheduling timeliness by running the NAPI polling process as kernel thread instead of soft interrupt context. To tie up loose ends, these techniques are important in our discussion here because the redirection of packets to AF_XDP socket is done by network driver during the NAPI polling process.

To summarize, as an industry we have made really good stride in standardizing API to configure TSN-related traffic shapers, inclusion of high performance XDP framework, extension of AF_XDP socket with Zero-Copy capability and the fine-tuning of NAPI polling to keep packet processing as deterministic as possible. These are important capabilities much sought after by real-time developers for implementing Linux-based real-time system.

Overview of Product offering of Intel TSN controllers

Intel offers TSN-capable Ethernet controller in two forms, namely discrete PCIe based Ethernet Adapter card or integrated Ethernet controller on selected processors as shown in below table.

	Capabilities	Remarks
Intel® Ethernet Controller I225-IT/LM [1][2]	• 4 RX and 4 TX DMA Channels • PCIe 3.1 • 10/100/1000/2500-Mbps • IEEE 802.1AS • IEEE 802.1Qav • IEEE 802.1Qbv • IEEE 802.1Qbu • IEEE 802.3br • Precision Time Measurement (PTM) for cross time-stamping between network clock and CPU clock (subject to CPU PTM capability) • Other Ethernet functionalities, e.g. EEE, Jumbo frame, L2/L3/L4 filtering and VLAN support	
Integrated TSN MAC Controller	• 10/100/1000/2500-Mbps • IEEE 802.1AS • IEEE 802.1Qav • IEEE 802.1Qbv • IEEE 802.1Qbu • IEEE 802.3br • Cross time-stamping between network clock and CPU clock • Other Ethernet functionalities, e.g. EEE, Jumbo frame, L2/L3/L4 filtering and VLAN support • Paired with 3rd Ethernet PHY through RGMII or SGMII	Available in below processor: • Intel Atom® x6000 Processor, formerly Elkhart Lake [3] has three integrated controllers each with 8 RX and 8 TX DMA channels. • 11th Gen Intel® Core™ Processor, formerly Tiger Lake UP3 [4] has one integrated controller with 6 RX and 4 TX DMA channels. • 11th Gen Intel® Core™ Processor, formerly Tiger Lake H [5] has two integrated controllers each with 6 RX and 4 TX DMA channels. • 12th Gen Intel® Core™ Processor, formerly Alder Lake S [6] has two integrated controllers each with 6 RX and 4 TX DMA channels.

- [1] https://ark.intel.com/content/www/us/en/ark/products/184681/intel-ethernet-controller-i225it.html
- [2] https://ark.intel.com/content/www/us/en/ark/products/184675/intel-ethernet-controller-i225lm.html
- [3] https://www.intel.com/content/www/us/en/products/platforms/details/elkhart-lake.html
- [4] https://www.intel.com/content/www/us/en/products/platforms/details/tiger-lake-up3.html
- [5] https://www.intel.com/content/www/us/en/products/platforms/details/tiger-lake-h.html
- [6] https://www.intel.com/content/www/us/en/products/platforms/details/alder-lake-s.html

Benefits of adopting Linux-based Solution for Real-Time System

Linux kernel being a corner-stone of successful open-source software project enjoys a lot of innovative design and implementation from a vibrant developers community. With Linux preempt Real-Time functionality capable of achieving low and bounded 2-digit microseconds process scheduling determinism, a wide variety of real-time appliances can be implemented by product makers on Linux-based system. We have seen in chapter 2, there is standardized API for TSN-related traffic shaper configuration too and high performant low latency packet processing capabilities are already in the Linux kernel today. The benefit of adopting Linux-based solution to implement real-time system are listed below:

- Easier product maintenance and faster security update due to source code availability

- Modern and future-proof approach where new kernel capabilities (e.g. eBPF) can be integrated into product.
- No hardware vendor lock-in as Linux framework often abstracts out architecture and hardware IP differences.
- Legacy workloads consolidation (e.g. Windows-based HMI) into the same box through hypervisor-based solution such as KVM-based hypervisor.

Case: Phoenix Contact

Phoenix Contact with its PLCnext technology is a manufacturer of industrial automation appliances that uses Real Time Linux with PROFINET and OPC UA based industrial Ethernet communication. With the publication of PROFINET specification v2.4 that integrates TSN functionality, validating technology readiness of TSN-capable network adapters and their Linux network drivers from supplier such Intel Corporation is critical to Phoenix Contact to ensure smooth adoption of new technology on existing real-time use-cases and “custom of the shelf” hardware.

Being ingredient provider, Intel enables its TSN product through Linux-based building blocks ranging from network driver to configuration utilities (ethtool and tc) to TSN/AF_XDP-based reference application showcasing how real-time applications could be developed and perform for Intel’s silicon as shown in Figure 6. As it is important to ensure Intel’s TSN hardware capability and the software interface is right for actual industrial automation equipment makers like Phoenix Contact, both Intel and Phoenix Contact have been collaborating to ensure technological requirements are well-understood and enhancements are iteratively contributed back to Linux community over the course of TSN technology validation by Phoenix Contact.

> “A major advantage of TSN and Real Time Linux is the use of Ethernet in time critical applications, without special fieldbus systems, Ethernet drivers and real time operating-systems. This enables us to offer competitive and open automation products with "Custom of the shelf" technology. It has been shown that a high implementation quality and performance of the integrated TSN hardware and software is necessary. Through the cooperation with Intel, this quality could be proven by means of PROFINET over TSN. Our Linux-based PLCnext controllers will therefore implement TSN on Intel platforms without special hardware extensions like e.g., additional FPGAs.
>
> We expect, that the combination of Real Time Linux and integrated Ethernet TSN interfaces is the best fit for todays and future automation solutions.”,
>
> explained Gunnar Lessmann, PROFINET Master Specialist, Phoenix Contact.

Conclusion

Time-Sensitive Networking standards are driven by IEEE community and undergo iterative enhancements over years as companies start to adopt and contribute learning back to the community. To accelerate adoption of TSN technologies, a matured open source software implementation where developers could adopt directly and further improve is critical. It is of such belief, intel invested in working with Linux open source community to add standardized interfaces in the area of TSN-related traffic shaper configuration, optimize fast packet processing capability called Express Data Path (XDP) and enhance time synchronization accuracy through hardware-based time-stamping capability. In recent years, Linux community has made big progress in integrating Linux preempt real-time support into Linux mainline and together with the support of TSN and XDP capabilities in the kernel, development industrial automation product based on Linux mainline interface is becoming a reality.

0822/OBL/In2/PDF