Application Note

Designing a Deterministic Scalable Ethernet Backbone for Robotics

Shaunak Deshpande, Surbhi Kapoor, Ranga Rakesh Jonnalagadda, Pradeep HN, Nilabh Anand, Sriramakrishnan Govindarajan, Pratheesh Gangadhar TK

ABSTRACT

Modern humanoid robots face a critical communication bottleneck that can limit their performance, safety, and commercial viability. Communication infrastructure is no longer just a transport layer between sensors and controllers; it is increasingly becoming a part of the control system itself. These complex systems must coordinate multiple motor drives, multiple sensors (IMUs, cameras, tactile arrays), and controllers in tight synchronization for balance, locomotion, and manipulation. This application note presents an implementation of daisy-chained Ethernet solutions using AM261x-LP (LaunchPad), leveraging the CPSW (Common Platform SWitch) peripheral integrated into the device. The architecture makes the use of TSN standards IEEE 802.1AS (gPTP), IEEE 802.1Qbu/Qbr (frame preemption) IEEE 802.1 Qbv (Enhanced Traffic Scheduling, and CPSW-based hardware Inter-VLAN routing and cut-through switching to achieve deterministic communication across serially connected nodes. The implementation serves as a reference for engineers designing systems where centralized ethernet switching is impractical and a deterministic, low-latency communication is required.

Table of Contents

Trademarks

All trademarks are the property of their respective owners.

1 Introduction

1.1 The Challenge of Determinism in Robotics

Industrial robotics applications increasingly require distributed intelligence, in which sensors, actuators, and control logic are distributed across multiple points along a structure. In multi-axis robot arms, for example, each joint may contain position encoders, torque sensors, motor controllers, and safety monitors that must communicate with a central motion controller. Similarly, modular production lines consist of independent processing stations that coordinate through real-time message exchange. Traditional network topologies use star configurations with centralized Ethernet switches, but this approach becomes impractical when nodes are physically distributed along linear or articulated structures. Running individual cables from each node back to a central switch increases cable weight, cost, and mechanical complexity, particularly problematic for robot arms where cables must traverse moving joints and rotating axes. Daisy-chain topologies address these constraints by connecting nodes in series, where each device has two Ethernet ports: one receiving from the upstream device and one transmitting to the downstream device. This reduces cabling to single links between adjacent nodes and eliminates the need for centralized switches. However, daisy-chain configurations introduce other fundamental challenges such as cumulative latency. Motion control packets may require bounded latency and extremely low jitters, while telemetry or diagnostic traffic can tolerate some delays.

Each packet must traverse multiple hops to reach its destination, and processing delays accumulate at every intermediate node. For a motion control system operating at a 1-millisecond cycle time (1 kHz control loop), even small per-hop delays can become significant when packets must traverse four or five devices. So, the fundamental challenge is simultaneously satisfying three conflicting design requirements:

1. Deterministic sub-millisecond latency for real-time control loops operating at kilohertz frequencies
2. Gigabit-scale bandwidth for high-resolution perception data from multiple cameras and LiDAR sensors
3. Cost-effective scalability using affordable components without vendor lock-in or licensing fees
4. Handle Mixed criticality traffic efficiently.

Traditional fieldbus protocols (CAN, EtherCAT, PROFINET) offer determinism but suffer from limited bandwidth (1 Mbps for CAN), and proprietary stacks that create vendor lock-in. Existing communication solutions for robotics fall into the categories below. CAN suffers from both bandwidth constraints (1 Mbps) and high latency, typical round-trip times range from 2-10 milliseconds depending on bus load and node count, far exceeding the sub-millisecond requirements for dynamic balance control. EtherCAT improves performance with cycle times of 250-500 microseconds in master-slave configurations, but multi-hop forwarding through daisy-chained slaves adds 20-50 microseconds per node. The EtherCAT implementation on the AM26x MCUs is limited to 100M. PROFINET IRT achieves 1-2 millisecond cycle times but requires specialized hardware and licensing. More critically, these protocols rely on rigid topologies (transmitter-receiver, star) and proprietary stacks. The table below summarizes the existing solution vs the solution proposed as a part of this appnote.

Table 1-1. Existing Solutions vs Proposed Solution

Protocol	Cost Effective	Bandwidth	Determinism
CAN	Yes	No	No
PROFINET	No	Yes	Yes
EtherCAT	No	Yes	Yes
Standard Ethernet	Yes	Yes	No
Proposed Ethernet + TSN solution	Yes	Yes	Yes

1.2 Why standard ethernet fails for real-time communication

Standard Ethernet is a best-effort protocol. When multiple traffic streams compete for bandwidth, lower-priority packets may be delayed unpredictably while higher-priority traffic is transmitted. Three specific problems affect daisy-chain industrial networks:

1.2.1 Head-of-Line Blocking

When a high-priority packet arrives at a switch while a large, low-priority frame is already being transmitted, the high-priority packet must wait for the entire low-priority frame to complete, potentially hundreds of microseconds

for maximum-sized Ethernet frames at 100 Mbps. This delay, known as head-of-line blocking, occurs at every hop in a daisy chain, causing cumulative unpredictability.

1.2.2 Lack of Time Synchronization

Closed-loop control algorithms require time-synchronized sensor sampling across multiple nodes. Without a common time reference, coordinated motion becomes impossible.

1.2.3 No Traffic Scheduling

Standard Ethernet provides no mechanism to reserve bandwidth for periodic control traffic. Background traffic (diagnostics, logging, parameter updates) can interfere with time-critical messages, causing missed deadlines and control instability.

Standard Ethernet without TSN exhibits highly variable latency, does not guarantee determinism, reliability, showcases high jitter and packet drops. Even recent TSN implementations in automotive and industrial robotics focus on star topologies with centralized switches, running on industrial PCs rather than embedded microcontrollers, and rarely report performance for multi-hop daisy-chain configurations exceeding 3-5 nodes. This work addresses this multi-dimensional challenge by demonstrating how Time-Sensitive Networking (TSN) extensions to standard Ethernet, combined with optimized daisy-chain architectures and hybrid protocol stack designs implemented on cost-effective embedded microcontrollers, can simultaneously deliver deterministic latency, high bandwidth, and commercial scalability, reliability for next-generation Robotics use-cases.

1.3 Time-Sensitive Networking (TSN) based proposed solution

1.3.1 What is TSN?

Time-Sensitive Networking (TSN) is a collection of IEEE 802.1 standards that add determinism to standard Ethernet without requiring proprietary protocols or specialized hardware beyond what modern Ethernet switches can provide. The solution proposes an Ethernet-based communication backbone for the entire robotic system, engineered to deliver deterministic, reliable, and better performance to satisfy strict closed loop latency requirements. This backbone interconnects multiple system components operating with short cycle times and high bandwidth demands that exceed the capabilities of existing solutions. In practical terms, TSN allows Ethernet to evolve from a best-effort packet transport into a deterministic communication fabric. That matters because robotics systems need more than bandwidth. The important architectural takeaway is this, TSN strengthens Ethernet by making time, priority, and schedule first-class elements of the network design. That is exactly what robotics requires. A TSN-enabled network can align transmission to a common timebase, reserve transmission windows, isolate traffic classes, and reduce the uncertainty introduced by congestion and contention.

The table below summarizes the TSN standards supported on the TI AM261x device.

Table 1-2. TSN Support summary on AM261x

TSN standard	Description	Support in HW / SW	
		CPSW 3G	Free RTOS support
802.1Q	VLANs	Yes	Yes
802.1AS-2020	Time Synchronization (AS-Rev)	Yes	Yes
802.1Qbv	Time-Aware Shaper(TAS) / Enhancements Scheduled Traffic (EST)	Yes	Yes
802.1Qbu	Interspersed Express Traffic (IET): Frame Preemption	Yes	Yes
802.1Qav	Forward / Queuing of Time Sensitive Streams (FQTSS) also known as eAVB (Ethernet Audio Video Bridging)	Yes	Yes
802.1Qci	Per-Stream Filtering and Policing (PSFP)	No On/off gate support	Partial
802.1CB	Frame Replication and Elimination for Reliability (FRER)	No	n/a
802.1Qch	Cyclic Queuing and Forwarding	No	n/a
802.1Qcr	Asynchronous Traffic Shaping	No	n/a
802.1Qat	Stream Reservation Protocol (SRP)	No	n/a
802.1Qcc	Enhancements to SRP		

Table 1-2. TSN Support summary on AM261x (continued)

TSN standard	Description	Support in HW / SW	
		CPSW 3G	Free RTOS support
802.1BA	Audio Video Bridging (AVB) Systems	Yes	Yes
Cut-through switching	No approved IEEE standard	Yes	Yes
Inter/Intra VLAN routing	No approved IEEE standard	Yes	Yes

The appnote suggests integration of some of the above standards in the standard ethernet application to help introduce determinism, robustness and guaranteed performance required for robotics use-cases.

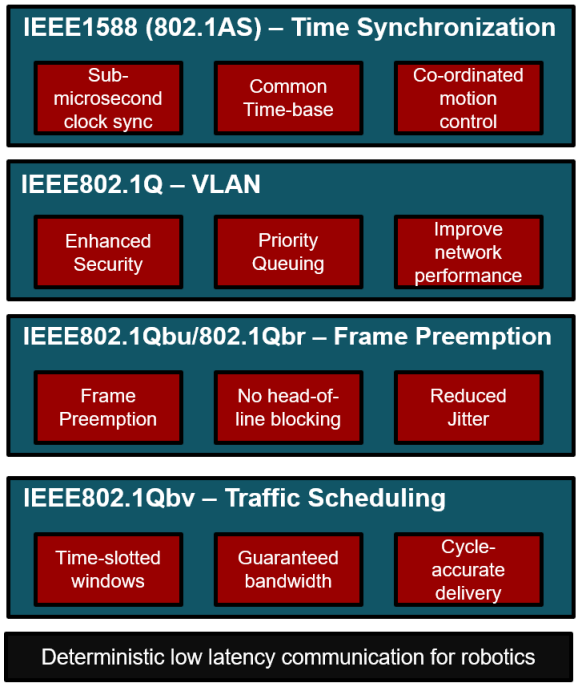

Figure 1-1. Proposed Standard Ethernet + TSN solution

1.3.2 IEEE 1588 (802.1AS gPTP - generalized Precision Time Protocol)

First step is to integrate IEEE802.1AS Precision Time Protocol. While PTP synchronization alone cannot guarantee deterministic frame transmission timing but provides the critical foundation for deterministic communication by establishing nanosecond-precision time synchronization across all network nodes. gPTP provides sub-microsecond time synchronization across all devices on an Ethernet network. Each device maintains a local clock synchronized to a grand-master clock by periodically exchanging timestamped messages. Hardware timestamping at the MAC layer eliminates software processing jitter, achieving accuracy better than 40 nanoseconds on capable hardware. The AM261x MCUs supports two-step PTP time synchronization via a combination of hardware and software blocks. Synchronized clocks enable coordinated sampling of sensors across multiple nodes, time-based traffic scheduling, and accurate latency measurement. In motion control, this allows a central controller to timestamp position commands and ensure all joints begin motion simultaneously.

PTP Critical Foundation Role:

- PTP provides synchronized time base required for coordinated network scheduling
- Without PTP synchronization scheduled transmission becomes ineffective and unpredictable
- Nanosecond timing accuracy enables precise coordination of traffic management decisions

1.3.3 IEEE 802.1Q (VLAN)

The next component to integrate is the IEEE 802.1Q (VLAN). VLAN creates virtual network segments using Priority Code Point fields in frame headers. Critical robotics control packets assigned priority 7 while background traffic uses lower priorities. Switch queues process high-priority frames before low-priority traffic to reduce control loop latency.

VLAN Protocol Limitations:

- Priority queuing cannot interrupt ongoing large frame transmissions
- High-priority frames still experience serialization delays from background traffic
- No time synchronization between network nodes causes timing inconsistencies

1.3.4 IEEE 802.1Qbu/Qbr (IET - Interspersing Express Traffic / Frame Preemption)

IET allows high-priority frames to preempt low-priority frames already in transmission. When a time-critical packet arrives at the MAC layer while a large background frame is being transmitted, the switch suspends the background frame, transmits the high-priority packet immediately, then resumes the background frame from where it was interrupted.

Preemption eliminates head-of-line blocking, the primary source of latency variation in daisy-chain networks. Without preemption, a 1500-byte frame at 100 Mbps takes 120 microseconds to transmit, during which time-critical packets must wait. With preemption, this delay is reduced to a few microseconds.

1.3.5 IEEE 802.1Qbv (EST - Enhancements for Scheduled Traffic)

EST provides time-aware scheduling in which the switch opens and closes transmission gates for different traffic classes according to a repeating schedule synchronized to the gPTP time domain. This creates guaranteed time windows for time-critical traffic, preventing interference from lower-priority streams.

EST enables cycle-based communication where all nodes know exactly when to expect control packets. Combined with gPTP, it creates deterministic end-to-end latency bounds even across multiple hops.

1.3.6 CPSW Specific hardware features

Along with the above IEEE TSN specs, the design also leverages hardware InterVLAN switching and cut-through switching from the CPSW sub-system. InterVLAN switching allows modification of some parameters of the ethernet packet in hardware, without the need to rely on software. The fields that can be modified are source and destination MAC address, source and destination IP address, ALE Time to live, VLAN Tags.

In this case, if the packet is at node it is addressed to, the application will swap the Source and Destination MAC address and loop it back, else it forwards the packet to the next node until it reaches the destined node.

Section 4. discusses the implementation and performance benchmarks in detail.

2 Sample Use Cases: Distributed Motion Control in Robotics

2.1 Representative scenario

Consider an industrial robot arm with each joint containing:

- Brushless DC motors with encoder (position feedback)
- Torque sensor (force feedback for compliant motion)
- Brake actuator (safety function)
- Local controller (joint-level motion interpolation)

A central motion controller generates trajectory commands at 1 kHz (1 millisecond cycle time) and requires position feedback from all joints to compute the next control output. Requirements dictate that if any joint fails to respond within the cycle deadline, all joints must enter a safe braking state. This needs a strong reliable and deterministic communication backbone, in this case implemented over standard ethernet.

2.2 Network Topology Requirements

2.2.1 Why Daisy-Chain?

The robot arm's mechanical structure makes daisy-chaining natural: joints are arranged in series from base to end-effector, and running individual cables from each joint back to the base requires cable routing through rotating and articulated sections. Each additional cable increases:

- Mechanical wear on moving parts
- Cable harness weight (reduces payload capacity)
- Installation and maintenance complexity

A daisy-chain topology mirrors the physical structure: the central controller connects to Joint 1, which connects to Joint 2, which connects to Joint 3, and so forth to the end-effector.

2.2.2 Real world applications of daisy chain ethernet solutions

The daisy-chained ethernet topology is especially advantageous in embedded systems with structured and linear deployment environments. Below are some relevant industrial and embedded use cases:

1. Modular Factory Equipment

In modular production lines, each manufacturing module (e.g., pick-and-place unit, welding station, inspection block) can be independently powered and networked. Daisy chaining these modules reduces wiring complexity and offers a scalable way to construct flexible manufacturing lines. Each module, powered by an AM26x-based controller, can communicate with adjacent modules and the central control unit without the need for a centralized ethernet switch. Real-time data such as sensor readings, control commands, and diagnostics are propagated through the chain.

2. Robotics: Articulated Arms and Mobile Swarms

For industrial or collaborative robots (co-bots), each joint or actuator stage may have its own embedded controller. Daisy-chaining enables efficient communication from the robot base (controller) to the end effector through intermediate joints. Latency-critical control signals can be routed deterministically using the internal CPSW switch in cut-through mode. In swarm robotics, multiple mobile platforms can be chained during recharging or docking, exchanging logs, sensor data, or mission parameters without requiring a wireless backbone.

3. Distributed Sensor Networks

In systems like structural health monitoring (e.g., bridges, tunnels, or oil rigs), sensor modules are deployed linearly over large distances. Daisy chaining allows a simple, rugged deployment where each sensor node samples local parameters and forwards them upstream. AM26x devices offer a compact, low-power processing unit for such edge analytics.

4. Building Automation

Linear or floor-wise network topology in large buildings (e.g., HVAC controls, access panels, lighting control) often aligns naturally with daisy-chained wiring. Each control unit communicates with its neighboring panels and central server, minimizing cabling and switch infrastructure.

5. EV Charging and Battery Management

In electric vehicle charging stations with multiple charge outlets, or in battery management systems where multiple modules need to exchange information (temperature, charge level, current flow), a linear ethernet connection avoids bus arbitration issues common in CAN or RS485 and provides deterministic latency with higher bandwidth.

2.3 Communication Requirements

Cycle Time: 1 millisecond (1 kHz control loop)

Traffic Types:

1. Time-Critical Control Traffic (Priority 7)

- Position commands: Central controller → Each joint (64-byte packets)
- Position feedback: Each joint → Central controller (64-byte packets)
- Latency requirement: < 150 microseconds round trip time, low jitter
- Determinism: Must not be delayed by background traffic

2. Background Traffic (Priority 0-3)

- Parameter updates (tuning gains, motion limits)
- Diagnostic data (temperature, current draw, error counters, logs etc)
- File transfers (new motion programs)
- Latency requirement: Best effort, can tolerate delays

Synchronization Requirement:

All joints must sample position and transmit feedback synchronized to within 1ms to avoid control artifacts from time-skewed measurements.

2.4 Test Implementation

The test system uses five AM261x LaunchPads in a daisy chain, representing:

- Spirent/TI AM64x processor: Central motion controller (or gateway to controller)
- TI AM261x LaunchPad Nodes 1 – 5: Joints in the robot arm
- The setup uses a 100M RGMII full-duplex link.

The external PC/ Spirent/ TI AM64x processor, acting as the central motion controller, generates packets to each node in the network every cycle. So, at the start of every 1ms cycle, the packets are generated for and sent to the 5th Node (farthest node), then to the 4th node and the last packet to the 1st Node (nearest node).

The reason for sending the first packet to the last node is to avoid network congestion between external PC and Node-1. Traffic in the network:

- 1. **High priority traffic (priority = 7): 64B** Unicast packet sent to every node at start of a 1ms cycle, (burst of 5 packets for 5 AM261x-LPs). As soon as the 5 packets are transmitted, background traffic starts.
  2. **Background traffic (priority = 0): 512B** background traffic is sent from packet generator to nodes for remaining time in the 1ms cycle (addressed to farthest node first, nearest node last).

     Standard IPG gap (12B) between High priority and Background traffic.
  3. **AM261x generated traffic:** Every 1ms, each node sends a Unicast packet to packet generator. **(516B)**

Table 2-1. Packet priorities

Packet priority (VLAN PCP based)	Packet size	Packet source	Packet destination
P-0 (highest) – High priority control traffic	64B	Packet generator	AM261x Node
P-5 – Node generated traffic	512B	AM261x Node	Packet generator

Table 2-1. Packet priorities (continued)

Packet priority (VLAN PCP based)	Packet size	Packet source	Packet destination
P-7 (lowest) – Background traffic	64B or 512B	Packet generator	AM261x Node

To get a highly accurate timestamp, special timestamping hardware like ProfiShark or other hardware network taps can also be used. One thing to note here is that when Wireshark is used with Windows, the timestamps observed are not accurate to the scale of microseconds and the results seen might differ. The AM261x can support up to 8 different priorities based on application requirements.

The flow of packets into the daisy chain network is shown below:

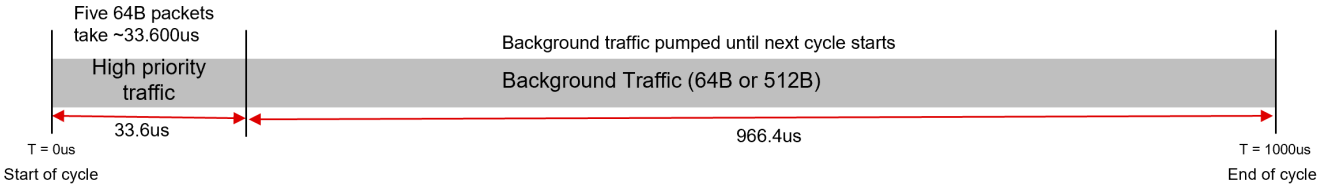

Figure 2-1. Ethernet traffic in a cycle

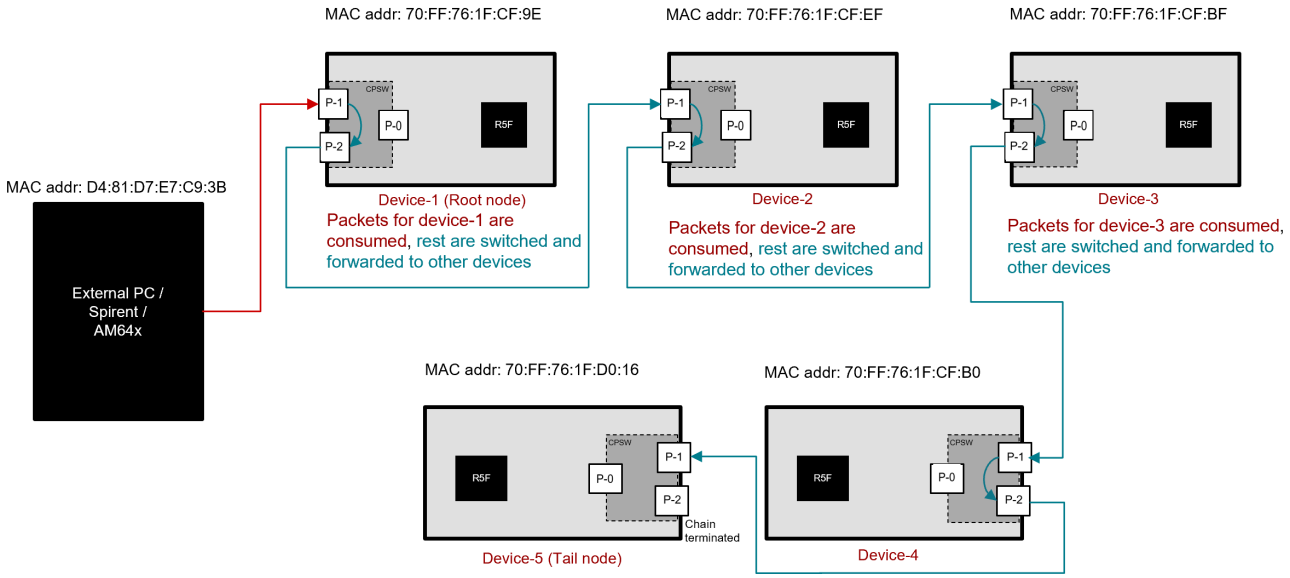

Figure 2-2. Sample packet propagation in the daisy chain network

The cycle starts with sending five high priority P0 frames to the Daisy Chain network, followed by low priority background traffic to the network until the end of the cycle.

The round-trip time is calculated for each node for the high priority traffic. This round-trip time includes the wire propagation time, CPSW processing and switching time for both, packet transmit from external packet generator to the daisy chain network and back to the packet generator. This setup helps in validating:

- Whether TSN can maintain sub-150 microsecond latency to the farthest node (Node 5)
- How latency scales with the number of hops
- Whether frame preemption effectively isolates high-priority traffic from background load
- Whether gPTP maintains sufficient synchronization accuracy.

While the actual robot arm can have more than five joints, the five-node test chain provides sufficient validation of the scaling behavior and establishes a reference for engineers to extrapolate performance for longer chains.

3 System Overview and Architecture

3.1 Hardware Architecture

The Texas Instruments AM261x is a microcontroller designed for industrial automation applications. Key features relevant to this implementation:

- Arm Cortex-R5F CPU:

  Running at 500 MHz with Real-time processing capabilities and sufficient compute for networking stack and application logic, 256KB TCM memory per core, and 1.5MB SRAM. This ensures the performance of the gPTP software stack is optimal to help achieve a high synchronization accuracy.
- CPSW3G Ethernet Switch:

  Three-port Gigabit Ethernet switch (one host port, two external ports), Hardware-accelerated Layer 2 switching and routing, TSN-capable switch supporting gPTP, IET (frame preemption), EST (time-aware scheduling), VLAN, Cut-through switching support at line rate, Address Lookup Engine (ALE) for MAC address filtering and VLAN management, Hardware Inter-VLAN routing with configurable routing rules, supports MII, RMII, RGMII interfaces with 10/100/1000M support.

This implementation uses a single ARM R5F Core, 100 Mbps RGMII interface over the two DP83869HM on-board LaunchPad PHYs.

3.1.1 AM261x LaunchPad

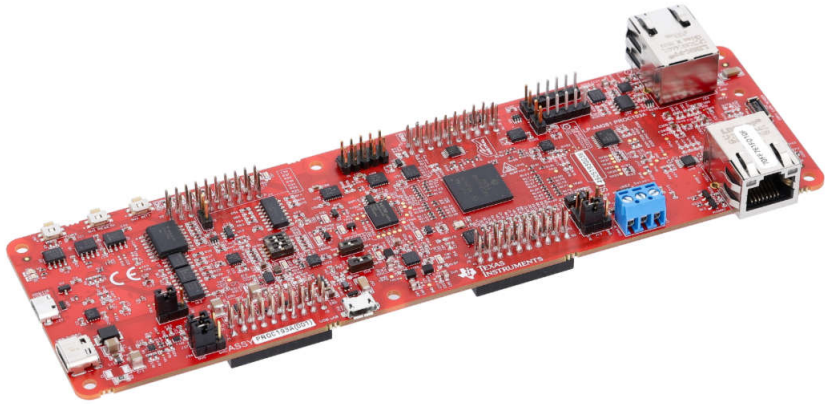

Figure 3-1. AM261x LaunchPad

The test hardware platform is the AM261x LaunchPad development board, Ethernet Configuration:

- Two external Ethernet ports (Port 1 and Port 2)
- DP83869HM Gigabit Ethernet PHY transceivers
- RJ45 connectors with link/activity LEDs
- RGMII interface between AM261x MAC and PHY

Power and Debug:

- USB-based JTAG debug interface via XDS110
- External 5V DC power supply

Board Modifications: None required for this implementation. The LaunchPad is used in its stock configuration.

Note: While this document talks about the AM261x, any other Micro-controller/Micro-processor from TI with CPSW interface can be used to implement a similar design.

3.1.2 CPSW Sub-System overview:

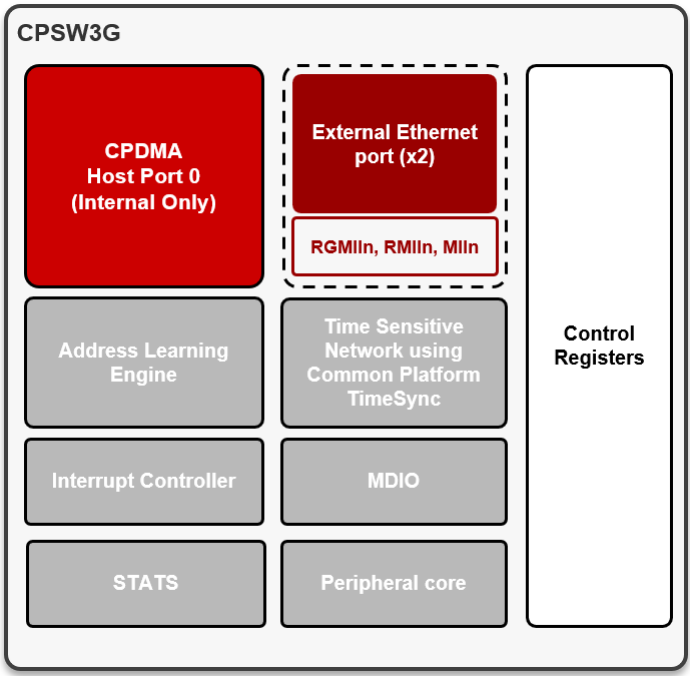

Figure 3-2. CPSW SubSystem Block diagram

CPSW or Common Platform Switch is a TI proprietary HW IP which enables the networking capabilities on the AM26x devices. CPSW subsystem provides IEEE 802.3 standard Ethernet gigabit speed packet communication for the device and can also be configured as an Ethernet switch. CPSW3G supports 10/100/1000 Ethernet ports with selectable MII, RMII, RGMII interfaces. The AM26x devices carry a 3-port Gigabit CPSW3G subsystem. The 3-port gigabit ethernet subsystem supports two external MAC ports and one internal CPPI port (communications port programming interface) or Host port.

Internal Port - One port (Port-0) is always reserved as the CPPI host port (CPDMA port) responsible for transfer of data between the CPU and the CPSW Peripheral core. Only the Host port can communicate with the CPDMA or R5F. Any incoming or outgoing data passes through the host port.

External Ports - The other (n-1) ports in CPSWNG are external MAC ports. E.g. CPSW3G has 2 MAC ports (port-1 and port-2) and one Host port (port-0). The external ports are MAC ports supporting Media Independent Interface (MII) like MII, Gigabit Media Independent Interface (GMII), Reduced Media Independent Interface (RMII), Reduced Gigabit Media Independent Interface (RGMII), Serial Gigabit Media Independent Interface (SGMII) and Quad Serial Gigabit Media Independent Interface (QSGMII).

The software configuration for the above sub-blocks is taken care of by the Syscfg auto-generated code and the Enet-LLD drivers. Based on the requirements of the application, these sub-blocks can be further configured in the application layer.

- CPDMA: In simple words, the CPDMA is responsible for transmission of data between the Host port and the R5F core. The CPDMA submodule is a packet DMA transfer controller. The Host CPPI Receive and Transmit interfaces can support line rate bandwidths on the ethernet ports. The Host Software communicates with network frames through what is called the CPDMA CPPI 3.0 and CBA 3.1 compliant packet DMA transfer controller host interface.
- ALE (Address Learning Engine): ALE is responsible for processing all the packets (based on the configurations of the ALE and incoming packet) and decide how/where the packet is forwarded. The ALE uses the incoming packet received port number, destination address, source address, length/type, and VLAN information to determine how the packet should be forwarded. The output of the ALE is a port-mask which is used to indicate the ports the packet should be forwarded to.
- Interrupt Controller: CPSW can fire interrupts from 6 different sources, which are used to handle the packets in the Enet-LLD driver (Software). Some examples include, THost (from Ethernet to host) non-paced level

interrupt, interrupt to update per-port statistics, Miscellaneous level interrupt used for events such as Link status change, ECC error conditions, configuration changes, ALE events etc.

- Stats module: CPSW contains an in-built stats module which can store the per-port statistics. This module is useful for evaluating the performance of the application as well as debugging. CPSW stats module captures important data such as Frame counts for good and bad frames, Frame drop counts for various cases such as ALE overrun, FIFO overrun, Fragmentation errors, portmask drops, over/undersized frames, CRC and alignment errors etc.
- CPTS (Common Platform TimeSync): CPTS sub-module enables time-sync capabilities on the device. it enables event timestamping for every packet received or transmitted from CPSW. The TimeSync events are pushed to the CPSW FIFO
- MDIO (Management Data Input Output): The Management interface module implements the 802.3 serial management interface to interrogate and control external Ethernet PHY using a two-wire bus. Some key functionalities of the MDIO module are:
  - MDIO acts as the bridge between CPSW and PHY
  - MDIO module allows the CPSW to configure and control various aspects of the PHY devices such as link speed, Duplexity, power management etc.
  - MDIO also is used to monitor the status of each PHY, detecting Link up/down events.
  - MDIO is also responsible for auto-negotiation between CPSW and PHY

Apart from the above, the CPSW is also capable of features such as Packet classification, VLAN support, Policing/Rate limiting which are relevant to this app-note. Certain applications require the functionality of classifying ethernet packets before processing them. Different packet classes might have different processing logic or need a quicker turn-around time. CPSW can perform packet classification based on the following parameters:

- Destination MAC Address
- VLAN ID
- Priority (based on VLAN IDs)
- Ether-Type
- IP header fields
- Combination of any of the above

VLAN Support

Virtual LAN (VLAN) support provides a mechanism for logical segmentation of network traffic within a daisy-chained topology. By assigning different VLAN IDs to traffic originating from or destined for specific nodes or functional blocks, developers can isolate control traffic from data streams, prioritize safety-critical messages, or enable multi-tenant communication over the same physical network. This is particularly useful in industrial automation or modular robotic systems, where different subsystems need to securely share a common Ethernet backbone without interfering with each other's data domains. VLAN tagging also helps in managing broadcast domains and reducing unnecessary traffic propagation across the chain, improving determinism and bandwidth utilization. The CPSW and Enet-LLD driver supports ethernet packets having VLAN Tags and even hardware classification based on VLAN Tags

CPSW also supports InterVLAN and IntraVLAN routing in Hardware. This means that based on the VLAN tag in the ethernet packets, CPSW can decide which VLAN the packet needs to be forwarded to.

Rate Limiting

Policers and rate limiting mechanisms enable fine control over Ethernet traffic, particularly important in daisy-chained topologies. Policers monitor the rate of incoming or outgoing packets and enforce defined bandwidth limits by dropping or remarking packets that exceed configured thresholds. This ensures that no single device or application overwhelms the shared network medium, thereby preserving fairness and preventing congestion across the chain. Rate limiting can be applied at the ingress or egress of each MAC port to cap the bandwidth used by specific traffic classes or flows. In daisy-chained deployments, this helps maintain predictable latency and ensures that low-priority or non-critical traffic does not starve time-sensitive control or synchronization packets. When used in conjunction with QoS and VLAN tagging, policers and rate limiting provide a robust traffic management framework that enhances the reliability and determinism of multi-node industrial Ethernet systems.

Each AM261x node is configured at boot to initialize its CPSW, set up port VLAN IDs, populate the ALE table with MAC-to-port mappings, and run the network application logic based on its chain position.

Read more about CPSW here.

3.2 Software architecture

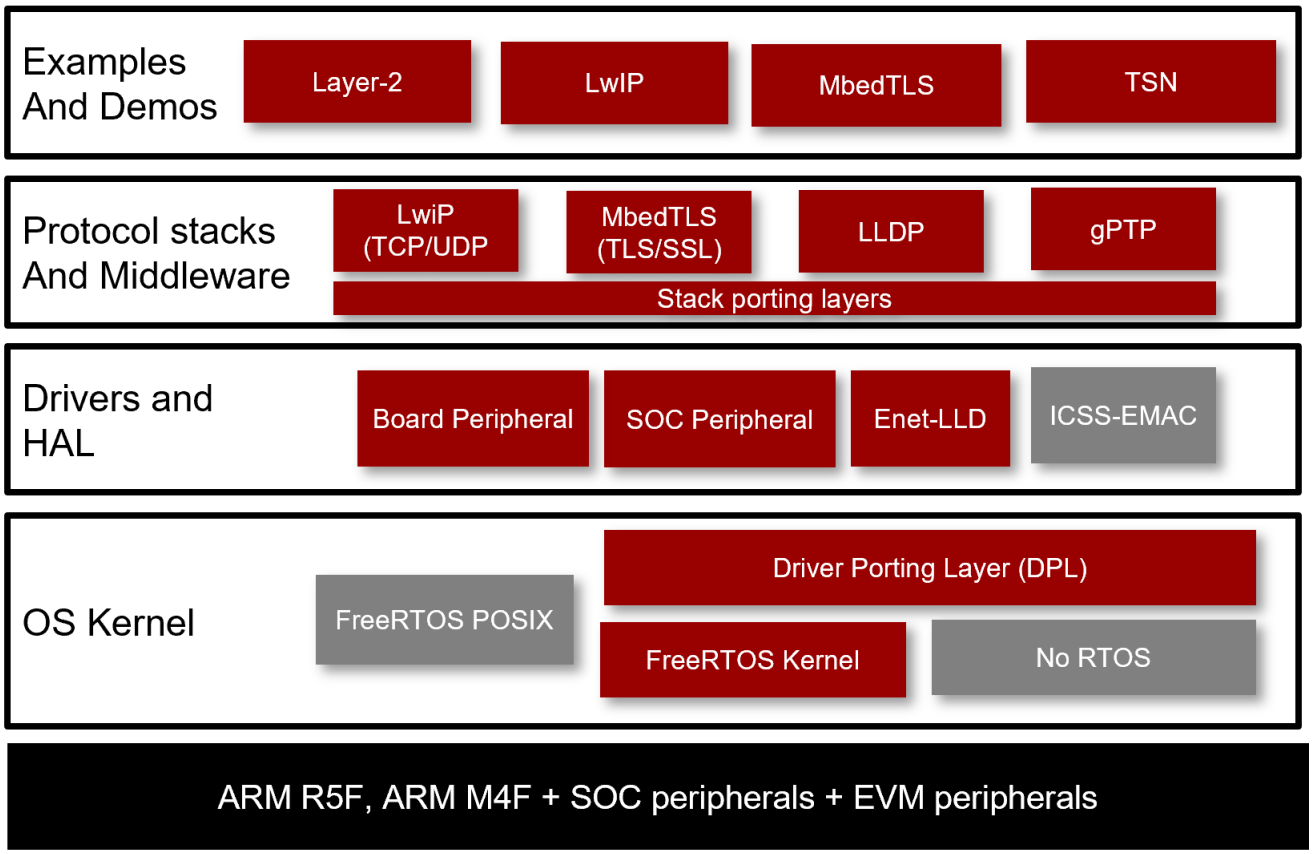

Figure 3-3. MCU_PLUS_SDK block diagram

For a general ethernet application running on the AM261x, the software is organized into the following layers:

1. **Board and Peripherals Initialization:** This is majorly handled by the syscfg auto-generated code. The example.syscfg provides customers with a GUI interface to configure peripherals. The syscfg then auto-generates code to handle pinmux, clocking and handle initialization for the application.
2. **CPSW Configuration Layer (Enet LLD):** The Enet-LLD code initializes CPSW3G, ensuring the PHY and MAC is configured and up properly. This involves setting up
   - Port control registers
   - ALE entries
   - Port states (forwarding/blocked)
   - Bringing up the external MAC ports with appropriate PHY drivers and link monitoring.
   - Configures static forwarding rules to avoid dynamic learning in the ALE.
   - Buffer Management and DMA Configuration
   - Setting up TX/RX descriptors for the host port.
   - Setting up interrupts to handle run-time events such as link change
3. **Operating System:** FreeRTOS or NoRTOS (BareMetal) layer. This layer provides features such as CPU Startup code, Interrupt handling, Memory Protection Unit APIs, Cache, Semaphore, Timer and Queue related APIs. In case of FreeRTOS, this layer additionally provides OS features such as Task scheduling etc. The Enet-LLD driver is agnostic of this layer, but the middleware stack configuration depends on the OS being used. This application will use the FreeRTOS based application.

4. **Middleware stack (LwIP, MbedTLS, TSN etc)**: In our case, this layer houses the TSN GPTP Stack. Additionally, the application can also use LwIP/MbedTLS stack.
5. **Application interface:** The application handles the task creation for Rx and Tx of ethernet packets, handling packet allocation, freeing, and buffer recycling with zero-copy strategies where possible (LwIP stack). ALE is programmed in no-learning mode to ensure deterministic switching behavior. The application handles the creation of CPSW classifiers for packet matching based on certain criteria. The application takes care of initializing the stacks and the interface layers. The IET, EST and InterVLAN configurations are present at this layer, but the actual hardware configuration happens at the enet-lld layer.

The diagram below depicts what the application discussed in this appnote looks like. The complete application implemented as a part of this appnote is available here: https://github.com/TexasInstruments/AM26x-add-on-SDK/tree/main/MCU_PLUS_SDK/reference_applications/AM261x_TSN_daisychain_app. The application is tested with v11.01.00.19 version of AM261x MCU_PLUS_SDK over the AM261x LaunchPad.

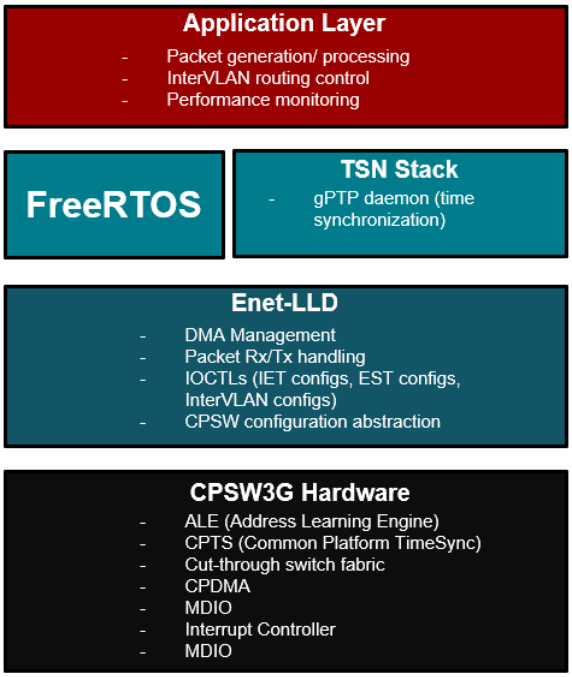

Figure 3-4. Application block diagram

4 Sample Implementation

This section discusses the implementation with incremental improvements made to reduce the round-trip time.

By default, the CPSW has cut-through switching enabled, which means that the packet forwarding will not be blocked until the entire packet has arrived, thus making the system faster.

The below figure shows the cut-through switching latencies (in us) for CPSW at 100M and 1G link.

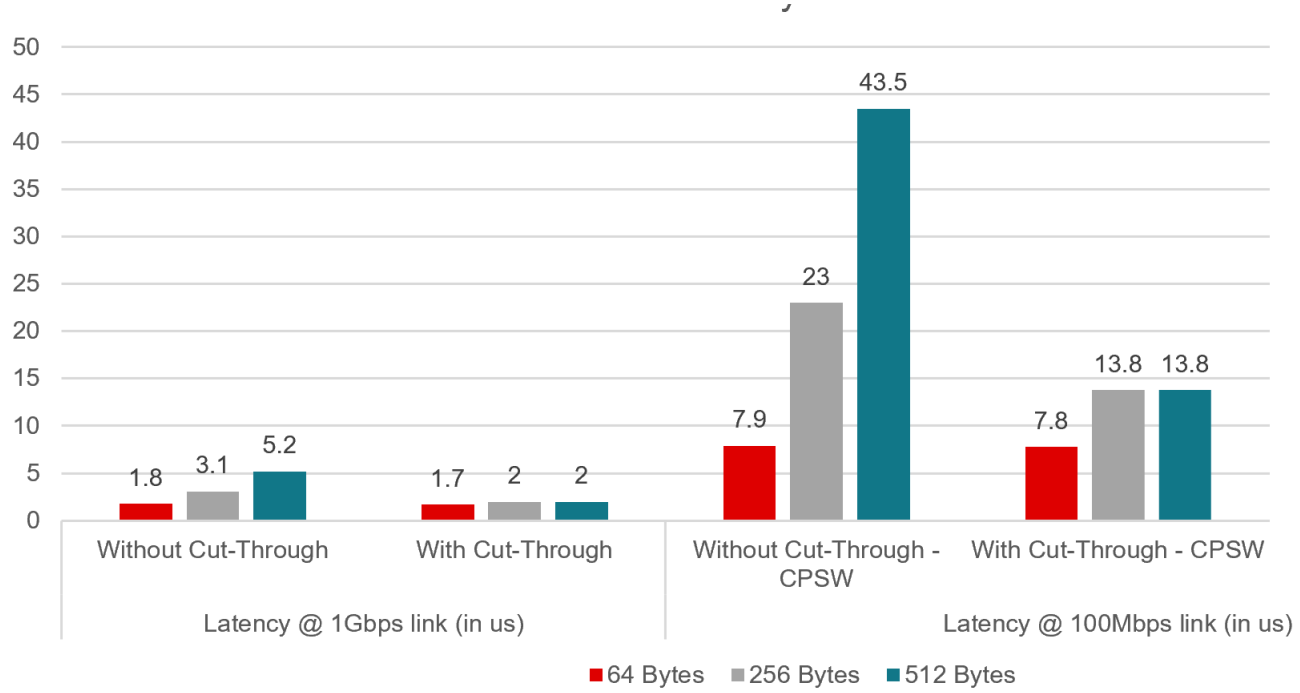

Figure 4-1. AM261x CPSW switch latencies

4.1 Standard Ethernet + CPSW InterVLAN routing

This is the baseline test, where the implementation uses hardware Inter-VLAN routing in the CPSW3G switch.

4.1.1 What is Inter-VLAN Routing

The CPSW is capable of InterVLAN routing where the Address Lookup Engine (ALE) determines an InterVLAN egress opcode for each packet to be VLAN routed via a classifier/policer configured for the route. The ALE classifier/policer can use the ingress packet destination address, source address, VLAN, IPDA, and/or IPSA to determine if a packet is to be VLAN routed or not. If a packet is to be routed, then the InterVLAN opcode is used on Ethernet packet egress. The routing includes the capability to modify some of the packet parameters such as VLAD ID, IP address and MAC address.

4.1.2 How This Implementation leverages Inter-VLAN Routing:

In this daisy-chain system, all traffic uses a single VLAN (VLAN 255). The Inter-VLAN routing feature is not used for actual VLAN-to-VLAN forwarding, but rather here it is used to swap source/destination MAC address swapping capability, which enables efficient packet loopback. The incoming packet Destination MAC address is checked, if it does not match the current nodes MAC address, the packet is simply switched and forwarded to the next node via MAC Port-2. If the MAC address matches, the source and destination address is swapped and the packet is looped back.

Packet Flow Example:

Test PC sends a packet to Node 3

- The packet travels via Node 1 and Node 2 to Node 3.
- CPSW3G ALE (Address Lookup Engine) performs lookup:

1. Match destination MAC = Node 3 MAC
2. Match VLAN = 255
3. ALE routing rule configured: "For packets to my MAC on VLAN 255, perform SA/DA swap and send to Port 1

- Outgoing packet from Node 3, Port 1, The packet propagates back through Node 2, Node 1, and arrives at Test PC.

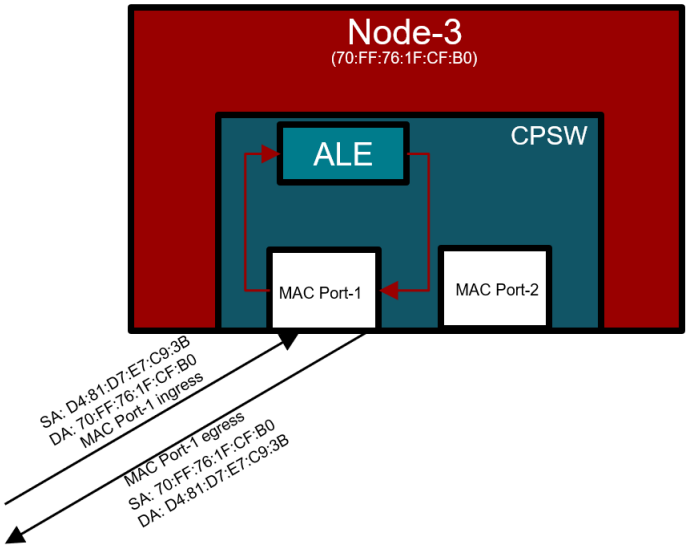

Figure 4-2. Packet loopback using InterVLAN routing

4.1.3 Test-1 Benchmarks

The baseline test numbers obtained with standard ethernet are discussed in Table 4. For each node, the round-trip latency refers to the time taken for a packet to travel from the packet generator to the node in the daisy chain network and be looped back to the packet generator (including the switching delays, wire delays, PHY delays, CPSW processing time).

This test does not involve any background traffic. The CPSW for this test is not configured to involve GPTP, EST or IET.

Table 4-1. Baseline results

Node	Round trip time (us) – (No background traffic)
Node-1 (nearest)	**Min:** 8.42 **Max:** 528.81 **Avg:** 511.96
Node-2	**Min:** 23.60 **Max:** 803.47 **Avg:** 656.64
Node-3	**Min:** 37.776 **Max:** 816.82 **Avg:** 671.89
Node-4	**Min:** 53.88 **Max:** 830.17 **Avg:** 686.91
Node-5 (farthest)	**Min:** 69.128 **Max:** 844.96 **Avg:** 702.20

Key observations:

- The round-trip latencies are ideal when the traffic injection just begins (the minimum numbers). The "minimum" is the best-case number for the daisy chain topology that can be achieved with 100M RGMII full-duplex link.
- Burst of 5 packets sent from farthest node to nearest node in order every 1ms. No CPSW packet drops observed when tested with a periodic burst of 550,000 packets (12B IPG). The above numbers are an average of 10 test iterations.
- After a while, when the network is congested with packets, the average and max times go out of bounds, making this implementation not ideal for our use-case.
- The difference between Node-2 and Node-1 latencies in ideal case is (23.60us – 8.42us = 15.18us). For a node "n" (n >= 1), the best-case latency can be formulated as:

Latency (n) = Node processing latency + 2*(n-1) * (Cut-Thru latency) + wire propagation time

- Using cut-through latency at 100M as 7.6us, wire propagation time as 5ns/m for CAT5 Ethernet cable, average latency per node is 8.4244 us.

Approximately,

Latency (n) = 8.4224us + 2*(n-1)*(7.6) + wire propagation time

- The average round-trip delays in Test-1 are very high because of no determinism or scheduling or synchronization in the network.

4.2 Integrating gPTP Time Synchronization (IEEE802.1AS)

Distributed deterministic systems require a shared time-base to operate in synchronization. Without synchronization, each node operates using an independent local clock domain, making coordinated scheduling and synchronized actuation extremely difficult. GPTP establishes a common network time reference across all nodes in the chain.

4.2.1 What is PTP time synchronization?

The Precision Time Protocol (PTP), defined by the IEEE 1588 standard, is a networking protocol used to synchronize clocks throughout a computer network with sub-microsecond accuracy.

The protocol operates using master-slave architecture. A high-precision reference clock, known as the Grandmaster, transmits timestamped messages across the network to subordinate devices. By measuring the precise departure and arrival times of these packets, the receiving nodes can mathematically calculate and remove network path delays. This reliance on hardware-level timestamping minimizes software processing lag, making PTP far more accurate than traditional software-based methods like the Network Time Protocol (NTP). Consequently, it is a vital technology for time-critical industries, including mobile telecommunications, electrical power grids, automated manufacturing, and high-frequency financial trading.

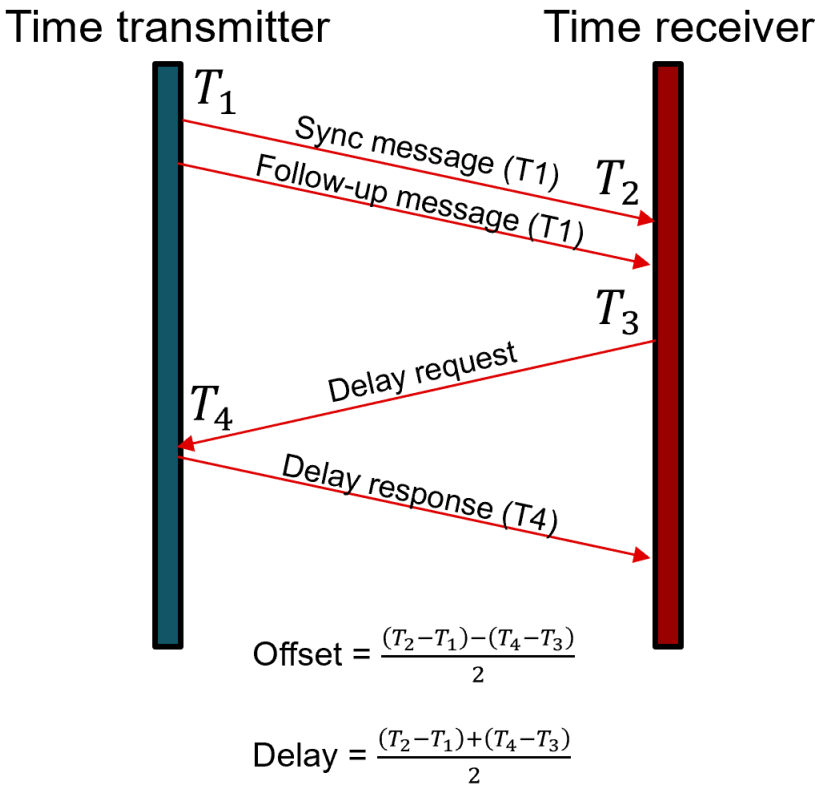

Figure 4-3. Two-step PTP time synchronization

The TI AM261x devices provide a 2-Step PTP mechanism, where the Time Transmitter (PTP Master) shares a SYNC message at timestamp T1, then sends a FOLLOW UP message with T1. The SYNC message arrives at receiver at T2. Then Controller sends a DELAY REQ at timestamp T3. Controller receives this at T4.

Using these four timestamps available at the PTP receiver, the delays and offset are calculated and the offset is used to adjust the PPM clocks via the CPTS block of CPSW.

Between 2 nodes, the benchmarks show that the TimeSync jitter is less than 40ns. So, at any point in time after sync, the clocks would not be at an offset more than 40ns, for two peer TI AM261x device.

4.2.2 How this implementation uses GPTP time synchronization

While no improvements are observed in round-trip latency with implementation of this, the implementation uses GPTP Time synchronization to synchronize the clocks for all the nodes in the daisy chain network. This provides a synchronized time base required for coordinated network scheduling. Without PTP synchronization, scheduled transmission becomes ineffective and unpredictable. Nanosecond timing accuracy enables precise coordination of traffic management decisions.

4.2.3 Test-2 Benchmarks

Enabling Time Synchronization does not help with reducing round trip latency, so the results are same as section 4.1.3.

4.3 Integrating VLAN (IEEE802.1Q)

4.3.1 What is VLAN?

VLAN helps create virtual network segments using Priority Code Point fields in frame headers. This helps us create 8 different priorities of packets. The IEEE 802.1Q standard is the networking specification that defines how Virtual Local Area Networks operate on an Ethernet infrastructure. The protocol operates by inserting a 4-byte identification tag into the header of an Ethernet frame. This tag contains a unique VLAN ID that allows a single network switch to separate physical traffic into isolated virtual networks. This logical segregation helps designers enhance security and performance and reduce broadcast congestion. When data travels between

switches across a shared trunk link, the switches read these tags to deliver packets exclusively to the intended virtual network at the intended priority by co-working with Frame preemption logic

4.3.2 How this implementation leverages VLAN

This design uses different packet priorities, as discussed in section 2.4. All the packets use VLAN Tag 255, but this can be modified as per application requirements. CPSW also supports filtering packets based on VLAN tags and hardware modification of VLAN tags. The application leverages the VLAN Packet prioritization along with frame preemption logic in hardware to significantly improve the round-trip latencies

4.3.3 Test-3 benchmarks

Table 4-2. Benchmarks with IEEE802.1Q

Node	Round trip time (us) (background traffic 512B)	Average Improvement v/s Test-1
Node-1 (nearest)	**Min:** 13.83 **Max:** 67.00 **Avg:** 44.89	**91.23%**
Node-2	**Min:** 27.18 **Max:** 194.79 **Avg:** 125.71	**80.86%**
Node-3	**Min:** 40.77 **Max:** 194.76 **Avg:** 190.02	**71.72%**
Node-4	**Min:** 61.27 **Max:** 399.11 **Avg:** 251.20	**63.43%**
Node-5 (farthest)	**Min:** 74.63 **Max:** 453.00 **Avg:** 303.64	**58.76%**

4.4 Integrating IET Frame Preemption (IEEE802.1Qbu/Qbr)

4.4.1 What is IET (Interspersed Express Traffic)?

The IEEE 802.1Qbu and IEEE 802.3br standards work in tandem to define the Interspersing Express Traffic (IET) frame preemption mechanism. This technology reduces latency for time-critical data in Time-Sensitive Networking (TSN) applications. When a high-priority express packet arrives at a network switch while a low-priority, best-effort packet is already being sent, the switch temporarily pauses the transmission of the low-priority frame. The urgent traffic is immediately sent across the wire. Once the time-critical data has completely cleared the queue, the switch resumes the transmission of the fragmented lower-priority frame right from where it left off. This hardware-level optimization prevents massive data frames from blocking time-sensitive control communication. It eliminates the need for large, inefficient guard bands and ensures deterministic performance in automated environments like industrial robotics and automotive networks.

4.4.2 How this implementation leverages IET

When the CPSW switch is already transmitting a lower priority packet and a higher priority packet arrives in the CPSW egress FIFO, the transmission of the lower priority packet will be preempted and the higher priority packet will be transmitted, ensuring timely delivery of critical information in the network. In this application, P0 is marked as express traffic and all 7 other priorities are marked as preemptable traffic. The IET can completely be configured from the application's example.syscfg file, including the prioritization of traffic.

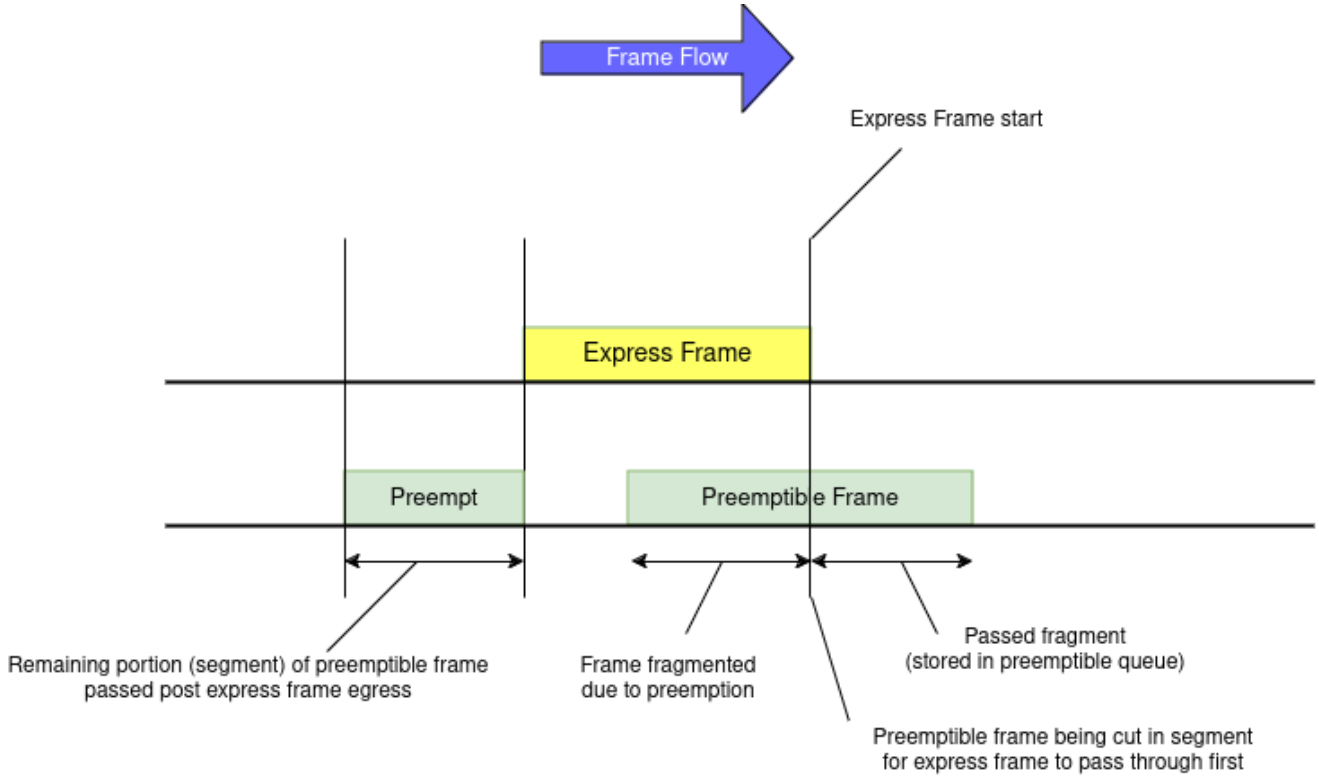

Figure 4-4. IET Frame preemption

4.4.3 Test-4 Benchmarks

After Integration of IEEE802.1Q, IEEE802.1AS, IEEE802.1Qbu/Qbr, a significant improvement in the overall round-trip latencies is observed.

Table 4-3. Benchmarks with IEEE802.1AS, IEEE802.1Q, IEEE802.1Qbu/Qbr

Node	Round trip time (us) (background traffic 512B)	Average Improvement v/s Test-1
Node-1 (nearest)	**Min:** 8.44 **Max:** 51.50 **Avg:** 29.74	**94.19%**
Node-2	**Min:** 23.60 **Max:** 75.82 **Avg:** 49.82	**92.41%**
Node-3	**Min:** 40.05 **Max:** 96.08 **Avg:** 69.10	**89.72%**
Node-4	**Min:** 57.13 **Max:** 115.87 **Avg:** 85.86	**87.50%**
Node-5 (farthest)	**Min:** 70.67 **Max:** 135.90 **Avg:** 101.95	**85.48%**

4.5 Integrating EST scheduling (IEEE802.1Qbv)

4.5.1 What is EST?

The IEEE 802.1Qbv standard, known as Enhancements for Scheduled Traffic (EST), introduces a time-division multiplexing mechanism to Ethernet networks via Time-Aware Shapers (TAS).

The protocol operates by implementing a time-controlled gate mechanism at the egress transmission queues of a network switch. A central schedule dictates exactly when these queue gates open or close, dividing network traffic into distinct, repetitive time slots. By blocking standard best-effort traffic during specific intervals, the standard creates a clear, contention-free window across the wire dedicated solely to high-priority, time-critical frames. This deterministic routing eliminates queuing delays and packet jitter for urgent data. Consequently, it guarantees precise, predictable delivery times in highly coordinated environments like aerospace communications and industrial automation.

The diagram below depicts how the gates mask the packet priorities and ensure only the configured priority packets can be transmitted/received in a particular slice of time in a cycle.

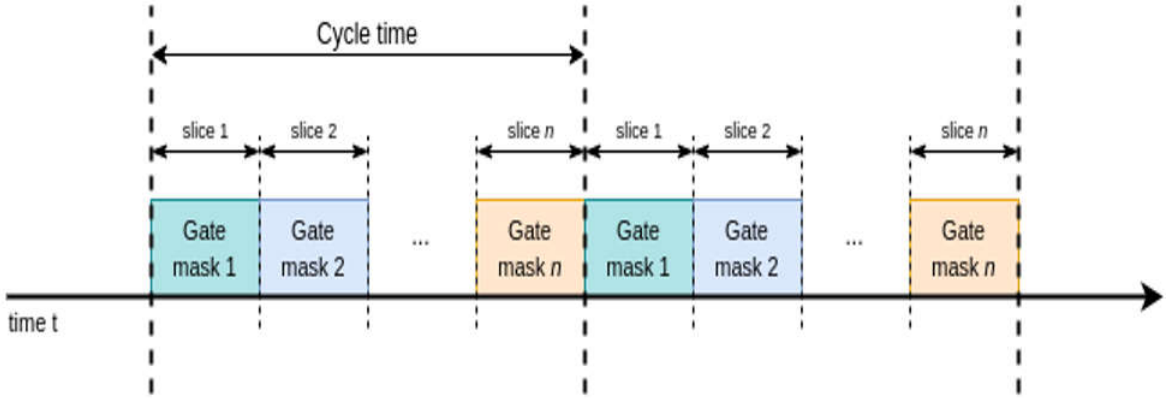

Figure 4-5. Enhanced Traffic Scheduling

This part of the application has not yet been implemented and tested. This is scoped in as future work, but the software and hardware support to implement this already exists:

- Refer to the out-of-box SDK example for EST gate scheduling: https://software-dl.ti.com/mcu-plus-sdk/esd/AM261X/latest/exports/docs/api_guide_am261x/EXAMPLES_ENET_CPSW_EST.html

The above example demonstrates configuration of the CPSW to schedule traffic of different priorities.

- Ensure the test-setup starts injecting the packets at the same moment as Time Synchronization is completed between all nodes. This will ensure that the gate open and close matches for all nodes across all the cycles.
- The results that can be expected after integration of EST traffic scheduling would be slightly better than the results discussed in Section 4.3.3.

5 Conclusion

This application note detailed the implementation of a daisy-chained Ethernet topology using AM26x devices, providing a comprehensive guide to building, configuring, and benchmarking such systems. It explored different IEEE TSN standards and how integration of each affects the round-trip latency in a simulated latency bounded environment created with AM261x LaunchPad with CPSW sub-system. The proposed design helps achieve over 90% improvement over the standard ethernet, ensuring performance, bandwidth, determinism, and no proprietary vendor lock-ins or need for specialized hardware/software.

Table 5-1. Benchmark Summary

Nodes	Standard framework (us)	+ GPTP + VLAN (us)	+ GPTP + VLAN + IET (us)	+ GPTP + VLAN + IET + EST (us)	Overall improvement (%)
Node-1 (nearest)	**Min:** 8.42 **Max:** 528.81 **Avg:** 511.96	**Min:** 13.83 **Max:** 67.00 **Avg:** 44.89	**Min:** 8.44 **Max:** 51.50 **Avg:** 29.74	*Future scope*	**94.19%**
Node-2	**Min:** 23.60 **Max:** 803.47 **Avg:** 656.64	**Min:** 27.18 **Max:** 194.79 **Avg:** 125.71	**Min:** 23.60 **Max:** 75.82 **Avg:** 49.82	*Future scope*	**92.41%**
Node-3	**Min:** 37.776 **Max:** 816.82 **Avg:** 671.89	**Min:** 40.77 **Max:** 194.76 **Avg:** 190.02	**Min:** 40.05 **Max:** 96.08 **Avg:** 69.10	*Future scope*	**89.72%**
Node-4	**Min:** 53.88 **Max:** 830.17 **Avg:** 686.91	**Min:** 61.27 **Max:** 399.11 **Avg:** 251.20	**Min:** 57.13 **Max:** 115.87 **Avg:** 85.86	*Future scope*	**87.50%**
Node-5 (farthest)	**Min:** 69.128 **Max:** 844.96 **Avg:** 702.20	**Min:** 74.63 **Max:** 453.00 **Avg:** 303.64	**Min:** 70.67 **Max:** 135.90 **Avg:** 101.95	*Future scope*	**85.48%**

- Predictable Latency Scaling: Hardware-accelerated Inter-VLAN routing in the CPSW3G switch achieves consistent 7.6 microsecond one-way latency per hop with cut-through switching. This linear scaling enables accurate latency budgeting for chains of 10-15 nodes within typical motion control requirements (< 200 µs one-way).
- Frame Preemption Eliminates Head-of-Line Blocking: IET (IEEE 802.1Qbu) reduces worst-case latency by ~85% under heavy background traffic load. Without preemption, 512-byte frames cause up to 473 microseconds worst-case round-trip latency to the fifth node. With preemption, this drops to ~136 microseconds, a reduction of 337 microseconds. This makes mixed-criticality traffic (time-critical control + best-effort diagnostics) viable on the same physical network.
- Sub-40 Nanosecond Time Synchronization: Hardware-timestamped gPTP (IEEE 802.1AS) achieves sub-40 nanosecond synchronization accuracy between all nodes, enabling coordinated sensor sampling and time-based traffic scheduling. This accuracy is three orders of magnitude better than software-based NTP and sufficient for even the most demanding motion control applications.
- Zero CPU Overhead for Transit Packets: The CPSW3G switch performs all packet forwarding, SA/DA swapping, and preemption entirely in hardware. CPU utilization remains under 12% even during heavy traffic, leaving 88% of processing capacity available for application logic, sensor fusion, and control algorithms.
- The performance can further be improved by using a small guard band towards the end of each cycle (for example ~100us of time where no new packets are transmitted and the packets already existing in the network get time to finish propagation, avoiding any new congestion),
- The Time Synchronization accuracy can further be improved by using PHYs that support hardware timestamping. Once the PHY timestamps and gets the T1, T2, T3, T4 values, the GPTP software stack will fetch these values from the PHY boundary instead of timestamping it when the packet enters the CPSW subsystem, making the offset calculations more accurate.

6 Challenges and Debug considerations

This section will address common debugging strategies, limitations of the architecture, and best practices to ensure system robustness and reliability.

6.1 Network Topology Verification

- **Link Integrity Check**: Ensure each node is physically connected and that the Ethernet PHYs are correctly linked. This can be validated using LEDs or software methods such as observing the PHY-Alive and MAC Port link up logs. In daisy-chained topologies, where each node's external MAC port is connected to the next node, ensure all upstream and downstream connections are properly established and remain stable.
- **Switch Configuration Verification**: Ensure the Address Lookup Engine (ALE) entries in CPSW3G are correctly populated for static forwarding of traffic. A misconfiguration of forwarding rules could lead to frame loss or improper routing. Ensure the application's example.syscfg file has all the MAC Port and Host port configurations done properly.
- **Loop Prevention**: Daisy-chained Ethernet networks can be susceptible to network loops if multiple paths are inadvertently established between nodes. This could lead to broadcast storms and network congestion. Ensure only the correct paths are active and implement software checks for loop detection if needed.

6.2 Traffic Flow Analysis

- **Packet Capture**: Use network analyzers (e.g., Wireshark) to capture packets at each node's external MAC ports to verify proper forwarding and correct payloads.
- **Traffic Patterns**: Observe the traffic patterns between nodes, ensuring that the forwarding is working as intended. It's important to look for dropped or misrouted packets, as well as verify that all application-specific protocols (e.g., UDP, raw Ethernet) are properly handled. If you encounter drops in packets at AM26x side, follow verify the drops and its root-cause using the CPSW stats. For more information, refer: AM26x Academy - Ethernet Debug Guide

6.3 Host Port Traffic Monitoring

- **Interrupts and Event Monitoring**: Disable Interrupt pacing (batch processing) for packets, Reduce the stats interrupt priority, Dis

6.4 PHY Link Management

- The AM261x-LP does not have any on-board PHYs. Make sure the PHY being used is configured correctly. If using boot-strapped settings, make sure the application is configured accordingly to work with Strapped PHY settings. If integrating a Non-TI PHY, follow the PHY Driver integration guide to setup the PHY correctly: https://software-dl.ti.com/mcu-plus-sdk/esd/AM261X/latest/exports/docs/api_guide_am261x/enetphy_guide_top.html
- PHY-related issues such as improper link negotiation, mismatch in duplex settings, or incorrect speed can cause instability. Verify the link status, auto-negotiation processes, and settings such as speed (10/100/1000 Mbps) and duplex (half/full). Refer: https://software-dl.ti.com/mcu-plus-sdk/esd/AM261X/latest/exports/docs/api_guide_am261x/enetphy_link_config_top.html for more details

6.5 Packets not forwarded to next node

Ensure that the ALE table has an entry for the configured VLAN tag with the correct membership mask. The port membership mask decides how the packet will further be transmitted

6.6 Error Handling and Retries

The applications used in the above tests did not implement any re-try mechanisms, but, In a multi-hop daisy-chained network, packet retransmission strategies should be employed if a node detects transmission failures (e.g., via timeout or checksum failure).

6.7 High latency or Jitter for high priority packets

Check the IET stats in the CPSW statistics for MAC Ports. Ensure the packets are getting fragmented and stitched back and IET is configured correctly. Use the CCS gel scripts for CPSW non-zero stats or manually add the expression (CSL_Xge_cpswRegs *) 0x52820000 in the CCS expressions window and monitor the stats.

6.8 gPTP not synchronizing

Check link status: Both ports must show link UP, then check port state. Check clock offset, If stuck at large offset (>100 µs), may indicate timestamping issue. Verify gPTP multicast is forwarded, Check ALE table for 01:80:C2:00:00:0E entry.

7 References

- Application link
- AM26x academy
- Getting started with Networking on AM26x devices
- Enet-LLD driver in MCU+SDK
- Enet-LLD API guide
- PHY integration guide
- PHY Link guide
- LwIP Guide
- MCU+SDK Ethernet one-pager
- IEEE802.1AS-20205
- IEEE802.1Qbu-2016
- IEEE802.1Qbv-2015
- IEEE802.1Q

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you fully indemnify TI and its representatives against any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale, TI’s General Quality Guidelines, or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products. Unless TI explicitly designates a product as custom or customer-specified, TI products are standard, catalog, general purpose devices.

TI objects to and rejects any additional or different terms you may propose.

Last updated 10/2025