Designing a CAN-to-TSN Ethernet gateway

N. D. Zervas, A. Sousek, P. Vrbka, CAST Inc.

The CAN network is designed to serve local systems with a relatively small number of nodes and bitrate. Transferring CAN frames over Ethernet is an efficient way of connecting multiple CAN domains using proven and cost-effective technology. The set of Time Sensitive Networking (TSN) standards made possible very low latency, low jitter and reliable communication and enabled the use of Ethernet networks for real-time control applications. CAN and TSN Ethernet endpoints and networks are expected to co-exist and cooperate in the same systems in the near future. The development of such hybrid-protocol systems requires gateways enabling communication between the CAN and Ethernet domains. This paper describes the architecture of a CAN-to-TSN gateway providing bridging functionality between CAN/CAN-FD buses and a TSN Ethernet network.

Introduction

The set of Time Sensitive Networking (TSN) Ethernet standards [1] have been designed to offer higher bandwidth, yet reliable, deterministic-time and low-latency communication as required by the evolution of automotive and industrial automation networks. However, TSN is not expected to replace the Controller Area Network (CAN bus) [2] in the near future. The large number of proven devices with CAN interface and CAN-related infrastructure cannot be replaced in a short period of time. Also, it would be challenging for TSN networks to meet the low latency characteristics of Time Triggered CAN networks, or other characteristics of specialized CAN networks. Therefore, CAN bus and TSN are expected to co-exist in the foreseeable future of automotive and industrial networks.

CAN-to-TSN gateways are essential components for the realization of such networks. These gateways need to introduce very small latency in order to allow for real time control from the one network domain to the other (e.g. from TSN to CAN).

This paper presents the architecture of a low-latency CAN to TSN gateway designed using commercially available silicon IP cores. The following sections describe the gateway's hardware architecture, protocol translation methods, latency characteristics and use-cases.

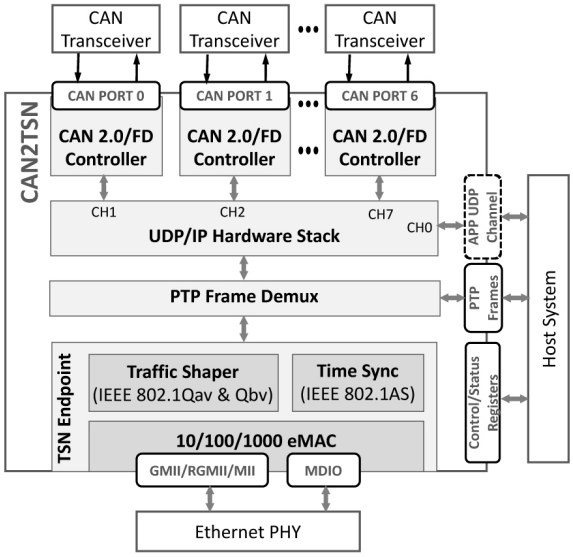

Figure 1: CAN-to-TSN Gateway

Hardware Architecture

The first wave of TSN Ethernet deployments in industrial and automotive systems use 100Mbps or 1Gbps, while CAN networks run up to 10Mbps in the best case (e.g. short distance communication between CAN-FD nodes). So, it makes sense for a CAN-to-TSN gateway to allow multiple CAN nodes to connect to Ethernet via a common port. The presented gateway is able to bridge up to seven CAN ports to a single Ethernet port.

Each CAN port is controlled by Fraunhofer IPMS' CAN controller IP core [3]. This CAN controller core supports both CAN 2.0 and CAN-FD protocols and allows filtering

incoming messages based on the CAN identifier, which is typically associated with the source of the message and its priority. The filtering functionality allows dedicating CAN ports to specific CAN message identifiers for traffic shaping purposes, as discussed later in this paper.
The Ethernet port is controlled by a Fraunhofer IPMS' TSN Endpoint controller core [4]. This TSN controller core integrates hardware stacks for timing synchronization (IEEE 802.1AS [11]) and TSN traffic shaping (IEEE 802.1Qav [10] and 802.1Qbv [9]), as well as a low-latency Ethernet MAC. Furthermore, the gateway uses CAST's UDPIP hardware stack [12] for encapsulating and decapsulating CAN frames into/from UDP/IP packets. A block diagram of CAN-to-TSN gateway is shown in Figure 1.

Address and Priority Translation

Any gateway needs to implement two basic functions: a) translate addressing information from one protocol to the other, and b) convey data from one protocol to the other. To the best of author's knowledge, there is no standard specification nor recommendation for mapping addresses and payloads from Ethernet to CAN and vice versa. Therefore, a new, non-standardized method was used.
The widely-used Internet Protocol (IP) [6] was the obvious choice for the Ethernet side, as it allows easy and cost-effective integration. The User Datagram Protocol (UDP) [4] was chosen over the Transmission Control Protocol (TCP) [5], as low latency is of paramount importance for the gateway. It is worth noting that the unreliable UDP can become absolutely reliable in TSN networks as time-aware shaping guarantees packets delivery for some or all traffic classes.
The CAN protocol is designed such that every node can receive every message. So, there are no explicit source and destination address fields in CAN frames, but rather an 11- or 29-bit identifier that typically encodes the type of data, the source of data, and the priority of a message.
On the other hand, IP networks use the IP address identify a network node, and the UDP or TCP port numbers identify the source and destination points within a network node. Priority is encoded in Priority Code Point (PCP) field in the VLAN tag of the Ethernet frame, and the priority can be further mapped to a traffic class for traffic shaping purposes [7].
Obviously, a direct, one-to-one mapping of the CAN message identifier to addresses and priorities of IP traffic is not possible. However, a CAN-to-TSN gateway should implement such mapping so that addressing different nodes in either the CAN or the IP network is feasible and priority can be signaled when moving from one protocol to the other.
The gateway described herein, associates each CAN port to a specific destination and source UDP port; the gateway itself is assigned a specific IP address. This way, messages from a CAN port are forwarded to the destination IP address and UDP port, and are marked as coming from the source UDP port which are associated with the specific CAN port (Figure 2). The destination IP address can be chosen to be a Unicast, Multicast or Broadcast. The destination IP address, source and destination IP addresses are independently set at run time for each CAN port.

Figure 2: CAN-to-Ethernet Routing

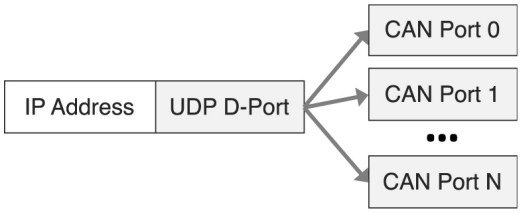

Figure 3: Ethernet-to-CAN Routing

Furthermore, each of the seven CAN ports is associated with one of the seven traffic classes. This allows mapping of any CAN identifier to any traffic class, by properly setting the acceptance filers of the CAN controller attached to each CAN port.
In the opposite direction the gateway will only accept packets addressed to the local IP address, and associated them with a CAN port based on the destination UDP port (Figure 3). The gateway may also be programmed to accept broadcast and multicast messages.

CAN Frames Encapsulation

The CAN-to-TSN Gateway encapsulates one or more CAN Messages from a CAN port as payload of a UDP frame, as shown in Figure 4, and expects incoming UDP frames to follow the same format. Table 1 provides short descriptions of the fields of the encapsulated CAN frame.

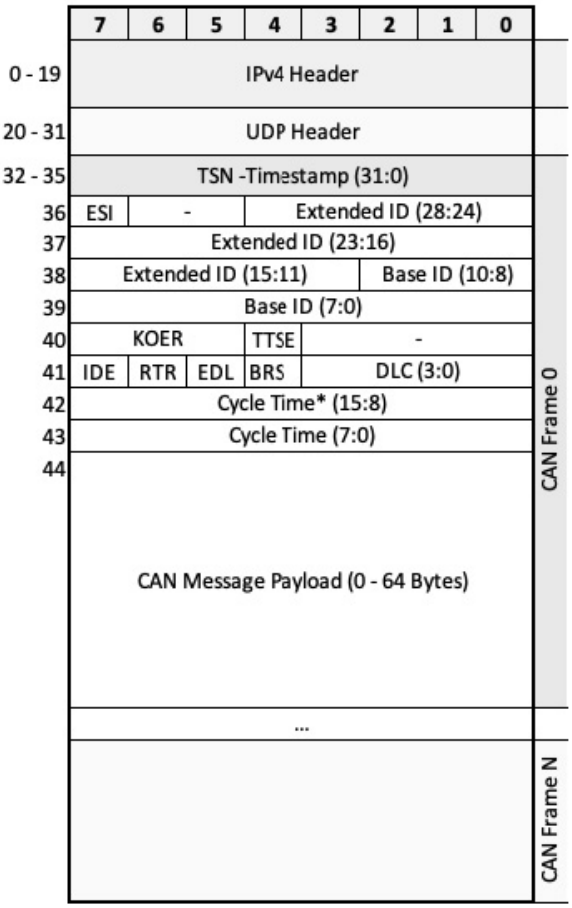

Figure 4: CAN Frame Encapsulation

With about 60 bytes, the Ethernet/UDP/IP framing represents a large overhead when transferring single CAN frame. To improve the Ethernet bandwidth utilization, the gateway allows encapsulating multiple CAN messages into a single UDP/IP packet. However, this grouping of CAN messages introduces additional latency in the CAN to TSN direction, which may not be always acceptable. The gateway allows users to trade latency for bandwidth utilization per CAN port by using the following two parameters:

- UDP payload size limit – limits number of CAN frames in the single UDP packet
- Transmit timeout limit – limits time till the frame is sent to the network.

Table 1: CAN frame fields

Field	Description
IDE	Identifier Extension 0 – Standard Format: ID(10:0) 1 – Extended Format: ID(28:0)
RTR	Remote Transmission Request 0 – data frame 1 – remote frame
EDL	Extended Data Length 0 – CAN 2.0 frame (up to 8 bytes payload) 1 – CAN FD frame (up to 64 bytes payload)
BRS	Bit Rate Switch 0 – nominal / slow bit rate for the complete frame 1 – switch to data / fast bit rate for the data payload and the CRC
ESI	Error State Indicator This is a status bit applicable only for received frames. 0 – CAN node is error active 1 – CAN node is error passive
TTSE	Transmit Time-Stamp (TTS) Enable 0 – no acquisition of TTS stamp for this frame 1 – TTS update enabled

Latency characterization

Low latency communication has been one of main goals in the design of the CAN to TSN gateway. To this end a custom hardware architecture, using the smallest amount of buffering possible has been implemented. The end result is that ingress (Ethernet to CAN) and egress (CAN to Ethernet) latency is less than 30 µs, excluding any delays related to TSN traffic shaping. This is the time between the instance a CAN frame has been received on a CAN port to the instance the same frame has been transmitted to the Ethernet port, when CAN message grouping is turned off. With message grouping turn on, the latency is increased by the time takes to receive all CAN messages that will be encapsulated as payload in the same UDP packet. However, this additional latency is under the designer control, via the programmable limits of the message grouping mechanism (I,e, UDP payload size limit and transmit timeout limit), and can be even eliminated if needed.

Latencies in the order of 30µs or even 60 µs (which is what it would take for a round trip from CAN to Ethernet and back) are within the limits even for real-time, closed-loop control systems that typically require reaction times in the order few ms, or few tens of ms.

Using the gateway

The CAN to TSN gateway can be used to configure, monitor and control the operation of CAN endpoint devices organized in networks or operating in a standalone basis. Recall, that each CAN port of the gateway is associated with a specific traffic class for traffic shaping purposes. This port to traffic class assignment can be changed at run time, and gives network designers the following options:

1. Connect all CAN ports of the gateway to the same CAN network, and configure each port to accept messages with a different subset of CAN identifiers, as shown in the example of Figure 5(a). This way CAN message identifiers are directly mapped to traffic classes for TSN traffic shaping.
2. Connect each CAN port of the gateway to a different CAN endpoint, as shown in the example of Figure 5(b). This way, TSN traffic classes are associated with endpoints.
3. Connect some CAN ports to the same CAN network, and some others to a different network or endpoint devices, as shown in the example of Figure 5(c). This enables endpoint and CAN networks, and CAN message identifiers within the CAN networks to be associated with different traffic shaping classes.

The highly flexible way of associating TSN traffic classes to CAN endpoints, CAN networks and CAN message Identifiers, is then exploited by the TSN traffic shapers of the CAN to TSN gateway, to implement reliable, low larceny and low jitter communication with the CAN nodes over Ethernet.

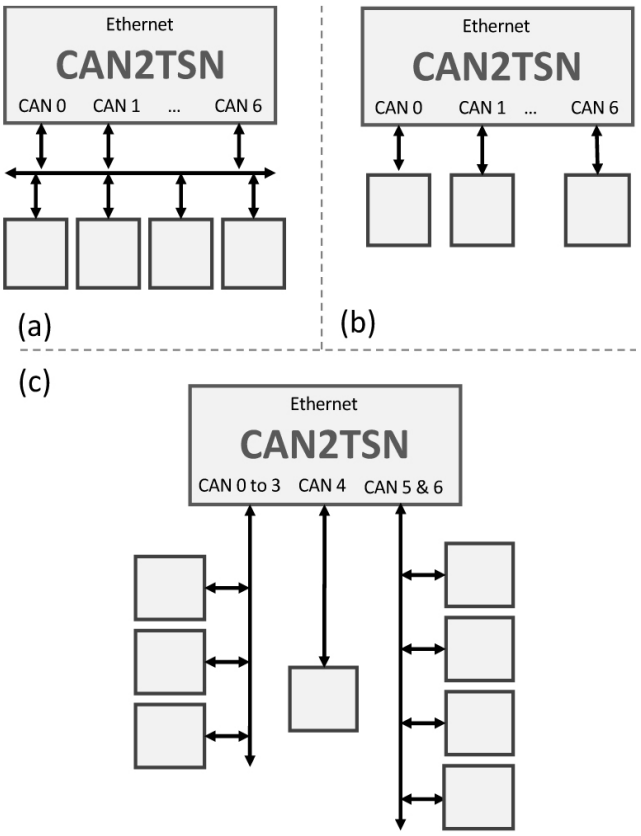

Figure 5: CAN ports usage examples

On the Ethernet side, the gateway can be configured to act as a simple end-point or switched end-point. A switched end point is suitable for implementing daisy-chained networks, such as the ring example of Figure 6, while a simple end-point is suitable for implementing a star network, as the example of Figure 7.

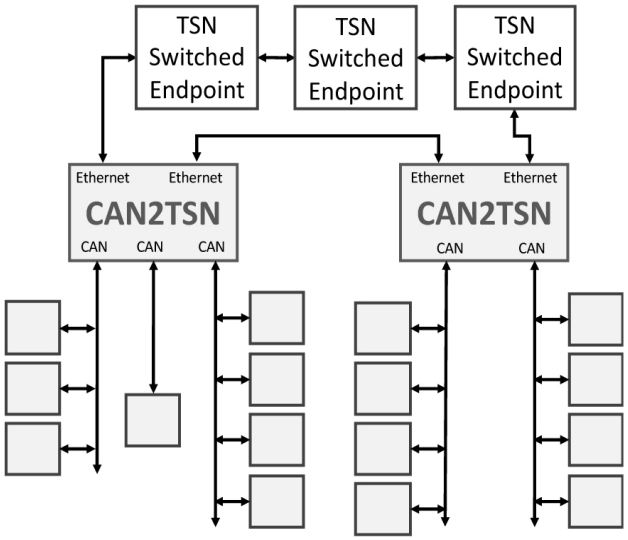

Figure 6: CAN to TSN Gateway Usage Example on an Ethernet Ring

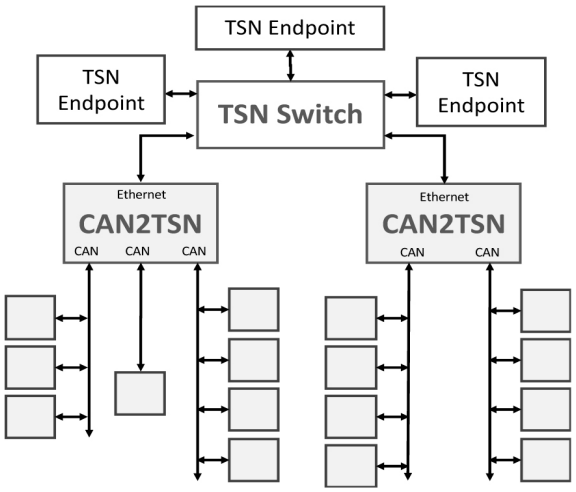

Figure 7: CAN to TSN Gateway Usage Example on an Ethernet Star

Conclusions

The architecture of a CAN-to-TSN gateway has been presented. The gateway uses proprietary UDP encapsulation of CAN messages, and a custom hardware implementation of all packet-processing functions. It introduces a latency smaller than 30us in both directions, and allows users to develop different traffic shaping scenarios by associating CAN identifiers to TSN traffic classes and allowing full control over TSN traffic shaping parameters. Furthermore, the gateway can operate in TSN network using either start or ring topologies. Therefore, this CAN to TSN gateway can be used to for the integration of legacy CAN devices in new TSN Ethernet networks, or in the control of local CAN networks over a TSN Ethernet backbone.

Nikos D. Zervas
CAST Inc.
11 Stonewall Court
US-07677 Woodcliff Lake, NJ
www.cast-inc.com

Antonin Sousek
CAST Inc.
Sumavska 15
CZ-60200 Brno
www.cast-inc.com

Pavel Vrbka
CAST Inc.
11 Stonewall Court
US-07677 Woodcliff Lake, NJ
www.cast-inc.com

References

[1] https://1.ieee802.org/tsn
[2] ISO 11898-1.2015, Road vehicles — Controller area network (CAN) — Part 1: Data link layer and physical signaling, December 2015.
[3] http://www.cast-inc.com/ip-cores/interfaces/can-ctrl/index.html
[4] http://www.cast-inc.com/ip-cores/interfaces/automotive/tsn-ep/index.html
[5] IETF RFC 768, User Datagram Protocol, 28 August 1980
[6] IETF RFC 793, Transmission Control Protocol, September 1981
[7] IETF RFC 791. Internet Protocol, September 1981
[8] IEEE Std 802.1Q-2014: IEEE Standard for Local and metropolitan area networks--Bridges and Bridged Networks”
[9] IEEE Std 802.1Qbv-2015: IEEE Standard for Local and Metropolitan Area Networks — Bridges and Bridged Networks — Amendment 25: Enhancements for Scheduled Traffic.
[10] IEEE Std 802.1Qav-2009: IEEE Standard for Local and Metropolitan Area Networks — Virtual Bridged Local Area Networks – Amendment 12: Forwarding and Queueing Enhancements for Time-Sensitive Streams, which specifies the Credit Based Shaper. (It is part of IEEE Std 802.1Q-2018.)
[11] IEEE Std 802.1AS-2011: Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks
[12] http://www.cast-inc.com/ip-cores/interfaces/udpip/index.html