CAMES: Enabling Centralized Automotive Embedded Systems with Time-Sensitive Network

Xiangrui Yang, Chenglong Li, Ling Yang, Chuhao Han, Tao Li, Zhigang Sun*
{yangxiangrui11,lichenglong17,yangling20,hanchuhao16,taoli,sunzhigang}@nudt.edu.cn
National University of Defense Technology
Changsha, China

ACM Reference Format:
Xiangrui Yang, Chenglong Li, Ling Yang, Chuhao Han, Tao Li, Zhigang Sun. 2021. CAMES: Enabling Centralized Automotive Embedded Systems with Time-Sensitive Network. In *SIGCOMM '21 Poster and Demo Sessions (SIGCOMM '21 Demos and Posters), August 23–27, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3472716.3472870

1 INTRODUCTION

A tremendous investment in automotive embedded systems has been observed in recent years. And the trend continues to grow considering the emerging self-driving vehicles [2, 6]. Those vehicles are equipped with complex sub-systems like multi-view camera system, lane-keeping assistant system, pedestrian avoidance system, etc. The systems are parts of the embedded system in the vehicle which contains hundreds of endpoints (cameras, millimeter-wave radars, GPS, brake system, etc.) and form a sophisticated network [12].

The state-of-art network of automotive embedded systems tends to be deployed in a distributed manner, where various sub-systems are divided into domains [6, 12]. In each domain, a Domain Control Unit (DCU) is responsible for computing tasks. The DCU can be seen as a SoC which contains a CPU and sometimes a FPGA/GPU [5] for remote acceleration. The endpoints within/between each domain are connected via specialized network/bus like LIN, CAN or FlexRay, which provides deterministic network in the loops between sensors, DCUs and actuators.

However, the emerging trend of self-driving in automotive brings at least three new challenges that the current systems find hard to address. **1) high-bandwidth and deterministic interconnection**. Currently, 60-100 endpoints (sensors/actuators) are deployed per vehicle. But recent industry figures suggest the number to reach over 200 per vehicle [1]. In addition, inter-domain traffic grows rapidly as domains are increasingly integrated. The ever-scaling system desires lightweight and high-bandwidth network rather than sophisticated and low-bandwidth buses. Moreover, deterministic features should also be preserved for real-time and security-related applications. **2) high-performance computing for AI-powered inference.** real-time AI models (e.g., convolutional neural network) has been widely adopted for autopilot functionalities [2, 3]. However, low-end computing platforms [5] have difficulties deploying large scale AI models [2, 9]. For instance, Tesla requires >50 Tops neural network performance while maintaining the power under 40W per chip (cooling reasons). This requires high-end computing node being implemented. Considering the budget and cooling constraints in automotive systems, deploying those computing nodes in a distributed manner would be infeasible. **3) Over-the-air (OTA) updates for automotive software/firmware.** Due to the fast evolution of software/firmware on vehicles, a number of manufacturers tend to update software/firmware via OTA techniques. This requires DCUs to be networked with the cloud from time to time. The distributed positioning increases the complexity of the remote updates and may incur security issues [6].

To address the challenges above, we present CAMES, a TSN-enabled centralized architecture for automotive embedded systems. In the center of CAMES, the central computing platform takes the advantage of high-end FPGAs for remote acceleration. DCUs can be largely simplified as no power-hungry computing tasks running locally. Between DCUs and the centralized computing platform, TSN (time-sensitive network) [11] is implemented to provide high-bandwidth and deterministic L2 connection for both real-time and non-real-time traffic. On top of TSN, we proposed a lightweight transport layer protocol for data delivering between sensors/actuators and the central computing platform. Since software/firmware of DCUs/sensors/actuators are updated by the central computing platform instead of exposing directly to the remote, security and simplicity feature can be better maintained during OTA. This demo demonstrates the advantages and feasibility of such a design with two usecases which are typical in automotive scenarios.

2 DESIGN

In our design, the computing resources originally in different DCUs are integrated into the central computing platform consisting of a FPGA pool (for remote acceleration) and CPU array (for corresponding software). All heterogeneous computing resources are centrally pre-allocated and scheduled according to domain requirements.

Fig. 1 gives the high-level abstraction of the proposed architecture. It includes three typical functional domains: **Infotainment**, **Telematics** and **Monitor**, all of which are connected with **Central Computing Platform** using TSN fabric. A functional domain may require both FPGA modules and the corresponding software to complete a specific task. Inside the central computing platform, an independent crossbar guarantees real-time and deterministically connections between FPGA pool and CPU array. The choice of TSN rather than other bus protocols (e.g. PCIe) avoids the competition

*Zhigang Sun is the corresponding author.

SIGCOMM '21 Demos and Posters, August 23–27, 2021, Virtual Event, USA

ACM ISBN 978-1-4503-8629-6/21/08…$15.00
https://doi.org/10.1145/3472716.3472870

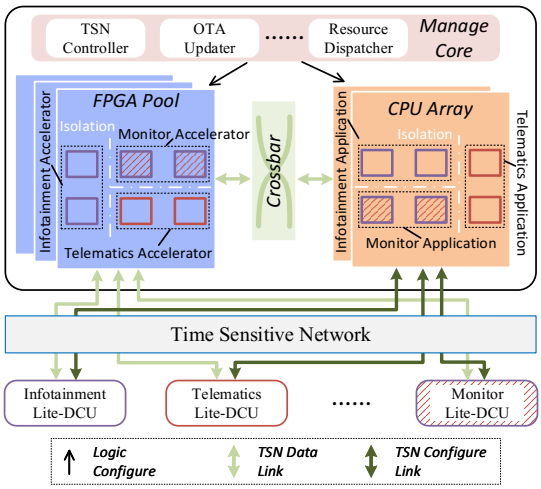

Figure 1: CAMES Architecture.

for shared communication resources and ensures better scalability. Furthermore, a designated core (i.e. Manage Core) is assigned for management tasks. Specifically, **Manage Core** is responsible for 1) computing resource dispatch. 2) TSN fabric configuration and scheduling. 3) OTA updates of all the DCUs. Below, we discuss in detail how different domains utilize centralized computing resources.

2.1 Remote Acceleration

Opposed from distributed architecture, the FPGA resources are shared among different domains. Thus, each domain of the automotive embedded system holds different slice of the resource pool (FPGA pool) to ensure isolation. While isolating different software process on the CPU is straightforward, we leverage an efficient virtualization method on the FPGA which improves resource utilization while keep different computing modules isolated both from functional and performance perspective.

Specifically, we incorporate the FAST framework [13] (for network processing) and Coyote [10] (for application acceleration) to make the best use of computing resources as well as isolating each task. FAST ensures there is no task preemption, performance disturbance, or shared exclusive FPGA resources among different domains. On the other hand, Coyote [10] is utilized for sharing FPGA external resources (DDR, HBM, etc.), which has been proven high-performance based on the idea of space partitioning and overlays.

2.2 Communication over TSN

An efficient fabric that delivers high-bandwidth and deterministic data communication is fundamental. In this regard, We adopt OpenTSN [4] in this demo. OpenTSN implemented a centralized version of TSN, which ensures the traffic of different priority can simultaneously processed while meeting their individual timing requirements. However, TSN only defines the data format of Layer 2, the communication between the functional domains and the central platform requires an efficient and lightweight transport layer protocol.

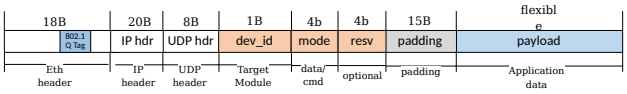

Figure 2: Format of the proposed lightweight transport layer protocol.

L4 Communication Protocol. In order to keep the design simple, we proposed a UDP-based transport layer protocol on top of TSN and IP (Fig. 2), which defines standards of register configuration, device control and data transmission. With the help of the proposed protocol, adding new hardware/software components becomes much more flexible within each domain.

Different from data centers [7, 8], the traffic in automotive embedded system can be divided into real-time and non-real-time flows. For real-time flows, the packets are controlled & shaped before injected into the network in order to be co-scheduled by the TSN controller. On the other hand, there is no such mechanism for non-real-time flows (i.e. best-effort). Different from FPGA-FPGA transport layer protocols in data centers.

Lite-DCU. We simplify the original DCU by removing redundant local computing resources. Lite-DCU only preserves data preprocessing abilities such as data encapsulation, data filtering, TSN-based traffic transmission & receiving between the sensor, the actuator and the central computing platform.

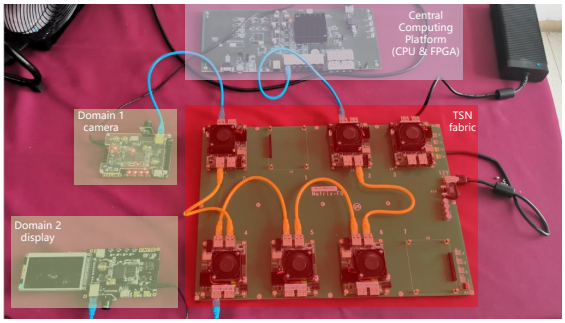

Figure 3: Experiment setup.

3 DEMO SETUP

Fig. 3 gives the implementation of the demo. The prototype simulates a road obstacle detection containing three domains: monitoring, information display and body control. The monitoring domain collects the original data from the camera. The central computing platform leverages the computing power on the FPGA to execute real-time detection. The information display domain displays the detected result. Moreover, the braking command is issued to the body control domain, which is defined as the time-sensitive traffic. We demonstrate that the TSN fabric guarantees the co-existence of time-sensitive traffic and other traffic in our prototype.

ACKNOWLEDGMENTS

This work was funded by the National Natural Science Foundation of China (91938301) and the Open Project of Zhejiang Lab (2020LE0AB01).

REFERENCES

[1] [n.d.]. Automotive Sensors and Electronics Expo 2017. http://www.automotivesensors2017.com.

[2] [n.d.]. Compute and Redundancy Solution for the Full Self-Driving Computer. https://old.hotchips.org/hc31/HC31_2.3_Tesla_Hotchips_ppt_Final_0817.pdf.

[3] [n.d.]. Hot Chips 31 Live Blogs: Tesla Solution for Full Self Driving. https://www.anandtech.com/show/14766/hot-chips-31-live-blogs-tesla-solution-for-full-self-driving.

[4] [n.d.]. OpenTSN 3.0 Opensource Project. https://github.com/fast-codesign/OpenTSN3.0.

[5] [n.d.]. Xilinx Automotive Grade Zynq-7000 SoCs. https://www.xilinx.com/products/silicon-devices/soc/xa-zynq-7000.html.

[6] Stefan Brunner, Jurgen Roder, Markus Kucera, and Thomas Waas. 2017. Automotive E/E-architecture enhancements by usage of ethernet TSN. In *2017 13th Workshop on Intelligent Solutions in Embedded Systems (WISES)*. IEEE, 9–13.

[7] Adrian M Caulfield, Eric S Chung, Andrew Putnam, Hari Angepat, Jeremy Fowers, Michael Haselman, Stephen Heil, Matt Humphrey, Puneet Kaur, Joo-Young Kim, et al. 2016. A cloud-scale acceleration architecture. In *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 1–13.

[8] D. Chiou. 2017. The microsoft catapult project. In *2017 IEEE International Symposium on Workload Characterization (IISWC)*. IEEE Computer Society, Los Alamitos, CA, USA, 124–124. https://doi.org/10.1109/IISWC.2017.8167769

[9] Eric Chung, Jeremy Fowers, Kalin Ovtcharov, Michael Papamichael, Adrian Caulfield, Todd Massengill, Ming Liu, Daniel Lo, Shlomi Alkalay, Michael Haselman, et al. 2018. Serving dnns in real time at datacenter scale with project brainwave. *iEEE Micro* 38, 2 (2018), 8–20.

[10] Dario Korolija, Timothy Roscoe, and Gustavo Alonso. 2020. Do {OS} abstractions make sense on FPGAs?. In *14th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 20)*. 991–1010.

[11] Wei Quan, Wenwen Fu, Jinli Yan, and Zhigang Sun. 2020. OpenTSN: an open-source project for time-sensitive networking system development. *CCF Transactions on Networking* 3, 1 (2020), 51–65.

[12] Soheil Samii and Helge Zinner. 2018. Level 5 by layer 2: Time-sensitive networking for autonomous vehicles. *IEEE Communications Standards Magazine* 2, 2 (2018), 62–68.

[13] Xiangrui Yang, Zhigang Sun, Junnan Li, Jinli Yan, Tao Li, Wei Quan, Donglai Xu, and Gianni Antichi. 2019. FAST: Enabling fast software/hardware prototype for network experimentation. In *2019 IEEE/ACM 27th International Symposium on Quality of Service (IWQoS)*. IEEE, 1–10.