An SDN Architecture for Automotive Ethernets

Marco Haeberle*, Florian Heimgaertner*, Hans Loehr†, Naresh Nayak†,
Dennis Grewe†, Sebastian Schildt†, and Michael Menth*
* University of Tuebingen, Chair of Communication Networks, Tuebingen, Germany
Email: {marco.haeberle,florian.heimgaertner,menth}@uni-tuebingen.de,
† Robert Bosch GmbH, Corporate Sector Research and Advance Engineering, Renningen, Germany
Email: {hans.loehr,naresh.nayak,dennis.grewe,sebastian.schildt}@de.bosch.com

I. Introduction

The electrical/electronic (E/E) architecture of cars has evolved from a single CAN bus to a domain-based model with domain-specific buses and centralized gateways. With increasing bandwidth demand resulting from the integration of camera and multimedia applications, Ethernet becomes a relevant network technology for in-vehicle networks [1]. Automotive Ethernet is based on 100Base-T1 over unshielded twisted single pair (UTSP) cables. Today, automotive Ethernet networks are still statically deployed and configured during the manufacturing process like traditional bus systems. In order to decouple hardware- and software-development cycles [2] dynamic in-vehicle networks are needed. We propose an SDN [3] architecture for automotive Ethernet networks that can be reconfigured after delivery to the customer or even during operation. For Ethernet-based real-time communication, 802.1 Time Sensitive Networking (TSN) [4] can be used.

This paper is structured as follows. Section II describes use cases for flexible in-vehicle networks. Section III presents a novel SDN architecture for automotive Ethernets. Section IV explains mechanisms for device and application discovery. We conclude our work in Section V.

II. Use Cases

A use case that could particularly benefit from reconfigurable in-vehicle networks are trailers connected to cars or trucks. Today, trailers are connected to the car electrically using one of several standardised connectors using 5 to 22 pins. The connectors support only few fixed functions, e.g., tail lamps, stop lamps, turn signals or in rare cases electric brakes. More sophisticated applications are not supported by traditional trailer connectors. An automotive SDN architecture, however, brings the ability to connect networked components in the trailer to the car's network. One of the most obvious applications in this use case is connecting park distance control (PDC) sensors or a rear view camera located in the trailer to the car's infotainment system.

Aside from feature use-cases like connecting trailers to cars, an automotive SDN architecture can also change how connected components are developed and serviced during their life cycle. Traditionally, the feature set of a car and the functionality of it's components does not change after the car is produced. If the software running on a component needs to be patched, the car needs to be brought to a repair shop with specialized diagnostic hardware. Adding new features with updates is usually not possible at all as the car's E/E architecture is static and cannot be changed. In recent years, software in cars has gotten more and more complex and the state of the art of complex systems like driver-assistance changes rapidly. This makes the ability to patch existing systems and add new features to it via over-the-air (OTA) updates desirable. An automotive SDN architecture gives manufacturers more flexibility to develop updates and new features for cars as it enables them to change the car's data network. One example is an update of a collision avoidance system. Originally, the collision avoidance monitors traffic in front of the car using a camera. The manufacturer then wants to update the system and add a feature that monitors traffic behind the car while reversing by checking the PDC sensors. Using a traditional E/E architecture, this feature cannot be added as the collision avoidance system has no access to the PDC sensors. Using an SDN architecture, the network can be re-configured to clone the packets coming from the PDC sensors and send the copies to the collision avoidance system.

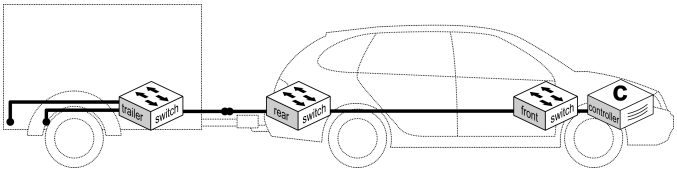

Fig. 1. Trailer connected to the in-vehicle network.

III. Architecture

Figure 2 shows an overview of the proposed network architecture. We use the topology introduced in [5] as a basis.

The in-car data plane connects all components and applications to each other and to the management system. Traffic on the data plane is divided into hard real-time traffic, soft real-time traffic, best-effort traffic and network configuration traffic classes. Hard real-time traffic originates at safety-critical components, e.g., the brake system, and must be transmitted with bounded low latency. Soft real-time traffic is associated with systems that are less critical and can operate in a degraded state if the deadline is not met, e.g., light and rain sensors. A rate limiter prevents faulty components from

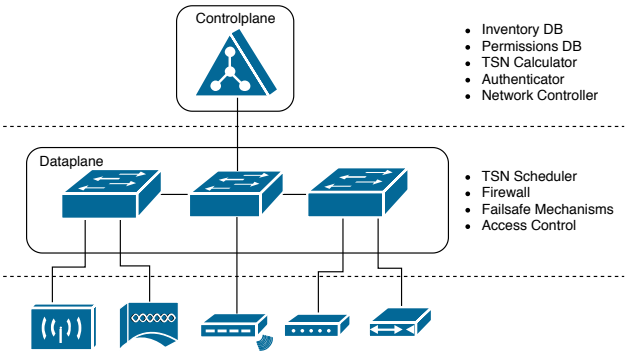

Fig. 2. Architecture overview.

flooding the network. Additionally, traffic of different classes and possibly within a class are isolated from each other to prevent faulty systems from impacting other systems and to prevent malicious attacks from intruders.

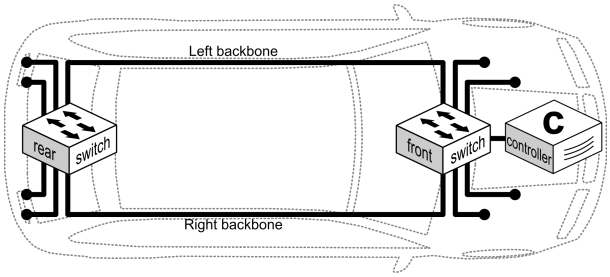

Fig. 3. Car network with in-band controller.

Figure 3 shows a possible network topology. During normal operation, the two links between the back and the front of the car are bonded together using link aggregation. Different scheduling variants can be used depending on the requirements with regard to throughput and safety. Possible scheduling variants may be round-robin scheduling for creating one logical link, distributing traffic on both links on a per-flow or per-component basis or using 1+1 protection for redundancy only. In case of a link failure, all traffic is redistributed to the functioning link. If maximum throughput on the single link is not sufficient, best-effort traffic can be dropped to guarantee a safe operation of the car. Furthermore, it is possible to reduce sensor rates to the minimal safe rate in order to reduce total traffic.

The data plane is configured by the network controller via in-band signalling. The network controller is connected directly to one of the switches. The network controller features a northbound interface that can be used by components and applications to trigger certain re-configurations of the network. Access to the northbound interface is regulated by access control lists (ACLs) and different permission levels. Configuration of the network is done using information from an inventory of components (e.g., sensors, actuators, and infotainment systems) and applications.

IV. Device and Application Discovery

The ability to add new components to the car's network, e.g., a trailer as described in section II, requires a mechanism for automated detection of new components and subsequent re-configuration of the network. At the same time, the network must not be compromised by malicious devices that are connected to the network.

To prevent attacks from unidentified devices, all traffic except for traffic on a dedicated and rate-limited discovery channel is dropped by the switch on ports that have not been in use so far. A device that gets connected to the network communicates its identification and requirements to the network by sending a signed manifest to the controller via a broadcast message on the discovery channel. The switches forward all packets sent on this channel to the network controller. The controller then checks the manifest's integrity and trustworthiness by checking if the manifest's signature is valid and originates from a trusted manufacturer. In case the device is deemed trustworthy, the controller then re-configures the network according to the requirements in the manifest.

In addition to devices that are added to the car's network, new applications may be installed on devices. These applications might have requirements to the network as well. Discovery of applications is performed similarly to the discovery of devices by sending a signed manifest to the controller. However, the manifest is not sent over a discovery channel by the application itself, but by the device running the application via the northbound interface.

V. Conclusion

Traditional in-vehicle networks are based on low bandwidth technologies like CAN. They are statically deployed and configured in the manufacturing process depending on the vehicle configuration. In this work we presented use cases that particularly benefits from configurable in-vehicle networks and proposed an SDN based architecture for automotive Ethernets.

Acknowledgment

The research leading to this results was funded by Robert Bosch GmbH. The authors alone are responsible for the content of this paper.

References

[1] L. L. Bello, "The Case for Ethernet in Automotive Communications," *SIGBED Rev.*, vol. 8, no. 4, pp. 7–15, Dec. 2011.

[2] S. Apostu, O. Burkacky, J. Deichmann, and G. Doll, "Automotive Software and Electrical/Electronic Architecture: Implications for OEMs," McKinsey & Co, Tech. Rep., Apr. 2020.

[3] W. Braun and M. Menth, "Software-Defined Networking Using OpenFlow: Protocols, Applications and Architectural Design Choices," *Future Internet*, vol. 6, no. 2, pp. 302–336, 2014.

[4] J. L. Messenger, "Time-Sensitive Networking: An Introduction," *IEEE Communications Standards Magazine*, vol. 2, no. 2, pp. 29–33, 2018.

[5] D. Pannell, L. Chen, J. Dorr, W. Lo, M. Potts, H. Zinner, and A. Zu, "Use Cases - IEEE P802.1DG V0.4," Jul. 2019.