Network Anomaly Detection in Cars based on Time-Sensitive Ingress Control

Philipp Meyer, Timo Häckel, Franz Korf, and Thomas C. Schmidt
Dept. Computer Science, *Hamburg University of Applied Sciences*, Germany
{philipp.meyer, timo.haeckel, franz.korf, t.schmidt}@haw-hamburg.de

***Abstract*—Connected cars need robust protection against network attacks. Network anomaly detection and prevention on board will be particularly fast and reliable when situated on the lowest possible layer. Blocking traffic on a low layer, however, causes severe harm if triggered erroneously by falsely positive alarms. In this paper, we introduce and evaluate a concept for detecting anomalous traffic using the ingress control of Time-Sensitive Networking (TSN). We build on the idea that already defined TSN traffic descriptors for in-car network configurations are rigorous, and hence any observed violation should not be a false positive. Also, we use Software-Defined Networking (SDN) technologies to collect and evaluate ingress anomaly reports, to identify the generating flows, and to ban them from the network. We evaluate our concept by simulating a real-world zonal network topology of a future car. Our findings confirm that abnormally behaving individual flows can indeed be reliably segregated with zero false positives.**

***Index Terms*—Vehicular network security, Time-Sensitive Networking, 802.1Qci, Software-Defined Networking**

I. Introduction

Current vehicles build new features on sensors, actuators, and Electronic Control Units (ECUs). Modern premium cars use a combination of traditional bus systems such as Controller Area Network (CAN), and Ethernet technologies for newly selected links. Future in-car networks will transform to flat Ethernet topologies [1]. In those networks, communication of various domains and distinct requirements on timing share the same Ethernet infrastructure.

This technological transition to shared Ethernet backbones raises the challenge of meeting the diverse Quality-of-Service (QoS) requirements of in-car communication. Real-time Ethernet extensions can provide robust QoS guarantees. The IEEE umbrella standard Time-Sensitive Networking (TSN) is the leading candidate for deployment in the vehicular world.

Complementary security challenges arise from connecting driving vehicles to the global network, and rigid countermeasures are needed to protect in-car networks against attacks. The attack surface of a car spans multiple interfaces [2]. With the release of a new standard for cyber-security in road vehicles (ISO/SAE 21434), the industry must harden security protections for future cars, which promises highest robustness on the lowest available layer. Online capabilities and the tightened interconnection of internal domains with distinct security requirements increase the vulnerability of safety-critical functions and raise arms for multi-sided measures to secure future cars [3].

This work is funded by the German Federal Ministry of Education and Research (BMBF) within the SecVI project.

In this work, we show that technologies, which are already under design discussion for in-car networks, can jointly improve the security of in-car communication architectures. First, TSNs ingress control [4] is part of a TSN network design. It forces all inbound traffic to match regular patterns (e.g. timing, bandwidth, size) by dropping frames. Second, Software-Defined Networking (SDN) uses a global network view to steer and control traffic flows. The central SDN network controller enables state management and reconfiguration of forwarding devices. We combine these technologies to form a Network Anomaly Detection System (NADS) that uses the ingress control of TSN to detect abnormal communication behavior and SDN to collect this detections. We evaluate the anomaly detection performance of TSN ingress control in a case study.

The remainder of this work is structured as follows. Section II discusses background knowledge and related work. In Section III, we introduce our concept and its underlying mechanisms. Our evaluation in Section IV is based on a case study, which we simulate using a realistic topology. Finally, we conclude in Section V with an outlook on future challenges.

II. Background & Related Work

A. Network Anomaly Detection

Intrusion Detection Systems (IDSs) are used to identify attacks on a system. The observed system can be a host (H) or a network (N). There are two IDS flavors. (1) Signature detection systems, which can detect attack patterns saved in predefined attack signatures. (2) Anomaly Detection Systems (ADSs) use a predefined description of regular patterns and all irregular patterns detected susceptible to be attacks. Our approach is a Network Anomaly Detection System (NADS) because it inspects a network (N).

There are many ADS algorithms to describe and evaluate a regular state [5]. Because ADSs build on features to distinguish between normal and abnormal behavior, any unclear or ill-defined distinction or rare event may lead to false anomaly reports and degrade detection quality. One key challenge for ADSs is to keep false-positive rate as low as possible. This work builds on the IEEE 802.1Qci part of a time-sensitive network, which should clearly and correctly define the normal real-time traffic link-layer behavior from quality assured network design phases.

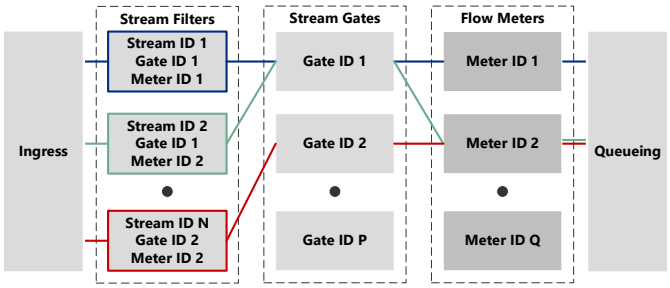

Fig. 1. IEEE 802.1Qci Per-Stream Filtering and Policing [4]

B. Per-Stream Filtering and Policing

TSN defines real-time Ethernet standard extension summarized in IEEE 802.1Q [4]. It subsumes different protocol mechanisms for traffic shaping and time synchronization. In particular, the 'IEEE 802.1Qci Per-Stream Filtering and Policing' (Qci) defines shaping of incoming traffic per port.

Figure 1 shows the structure of the filtering process in IEEE 802.1Qci, which applies to all ingress traffic. A frame must pass through three stages before queueing. The first stage identifies individual streams and maps them to the corresponding gates and meters. The second stage consists of the stream gates. A gate can be "OPEN" or "CLOSED", which changes on a predefined schedule based on a network-wide synchronized clock. If the gate is "CLOSED", the frame will be dropped. If the responsible gate is "OPEN", a flow meter inspects the frame. A flow meter applies an algorithm that determines whether a frame is allowed to pass. An example flow meter could control a minimum time between two frames. Such a flow meter would drop a frame if too little time has passed since the previous frame arrived.

C. Software-Defined Networking

SDN [6] introduced the paradigm of programmable switches, which since then has largely impacted data-center and campus networks. In recent years, use cases of SDN extended to other areas such as vehicular networks [10] and industrial plants [11]. In SDN, a central controller with global network knowledge decides per flow on forwarding rules. For a frame without specific forwarding rule, a default will be applied, which drops the frame or forwards it to the controller for further actions. We use the ability of the central controller to collect statistics of the forwarding devices, identify the abnormal flows, and execute possible countermeasures.

D. Related Work

Miller and Valasek [8] show that modern cars are vulnerable to attacks. In their work, they describe how they obtain unpermitted control of an unaltered passenger vehicle. They gained remote control over safety-critical functions like engine and brakes. The entry point is the cellular connection of the cars infotainment system. They use the entry point to send manipulative messages into the in-car infrastructure. A NADS could detect such messages.

Rajbahadur et al. [9] worked out a survey on anomaly detection for connected vehicles. They list numerous work in the spectrum of anomaly detection for current cars. Most of the works discussing the use of ADS inside the car are considering traditional CAN bus devices and infrastructure. Ethernet in-car infrastructures have very different communication patterns leading to different requirements on a NADS.

SDN is already in design discussion for in-car networks. Halba et al. [10] show how SDN controller applications can improve the robustness and safety of the in-car network with fast fail-over mechanisms. Gerhard et al. [11] use SDN to configure TSN networks in the context of industrial ethernet. In previous work [12], we analyzed the real-time capabilities of SDN and introduced an integrated Time-Sensitive Software-Defined Networking (TSSDN) switching methodology that combines TSN and SDN capabilities. In this work, we use SDN for collection of statistics and countermeasures.

III. Detecting Network Anomalies with TSN

In IEEE 802.1Qci each unique traffic flow is represented as a stream, to which Per-Stream Filtering and Policing applies. A rigorous Qci network configuration enforces valid traffic patterns on the link-layer. For example by restricting streams to a reserved bandwidth or validate the timing of cyclic signals. In this concept TSN "streams" and SDN "flows" describe identical network traffic.

The safety-critical communication flows of in-car networks are well-known and a priori specified in network designs. This knowledge includes worst-case analyses, communication patterns, and timing details. Implementing Qci should include configurations based on the known network parameters (e.g., flow and burst sizes, maximum transmission unit, time slotting). A rigorous configuration can thus enforce the expected behavior of traffic flows in the network.

Thereby, *the Qci configuration serves as an implicit description of the boundaries between normal and malicious network behavior on the link-layer.* In this sense, a frame dropped by a Qci rule indicates an abnormal behavior and can be interpreted as a detector of an anomaly—a valuable input for a NADS.

Switches collect statistics of critical values for administrative inspection. Counting individual frame drops of flow meters and stream gates accumulates anomaly indicators, which are invisible to other systems in regular networks. These indicators remain free of false positives, provided the Qci network configuration is correct.

Switches can communicate their statistics to a central manager, e.g., by NETCONF. Such a manager may be a dedicated network monitor or an SDN controller. The latter can request periodic transmissions of statistics from all switches in the network using the OpenFlow protocol. The duration of this period introduces a delay until an anomaly is reliably discovered. Still, implicit pieces of information from Qci can reach a destination where advanced controller applications can be implemented without putting additional stress onto the switches or adding NADS devices to the network. Such a NADS controller application can gather this data and inspect it independently or in combination with other reports.

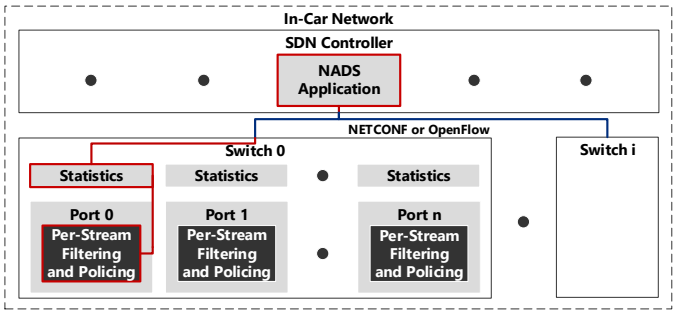

Fig. 2. Per-Stream Filtering and Policing combined with SDN to a NADS

Figure 2 visualizes the concept of combining Qci with SDN to form a NADS. Each port of the switch (at bottom) hosts an instance of Qci containing its individual configuration of regular ingress traffic, which also collects key statistics (e.g., number of dropped frames per gate/meter). The switch forwards these statistics upstream to the SDN controller. In this example, the Per-Stream Filtering and Policing in port 0 of switch 0 receives a frame that violates a meter configuration and gets dropped. This triggers a change of dropped frames statistics at its meter, which is communicated to the SDN controller. The NADS application of the controller inspects whether the reports of the switches implicitly indicate an anomaly in the vehicle network (e.g., a counter of a statistic increases). Whenever an anomaly is detected, the NADS controller application could escalate combined reports to an even higher instance (e.g., a cloud defense center [13]), or initiate countermeasures by reconfiguring the network flow tables and the TSN settings.

Flow meters that report anomalies from Qci instances can also be used for detection tasks only without a deletion of frames. By adding other statistics (e.g., number of anomalies), it is possible to use flow meters that never drop frames but still inspect the assigned traffic for abnormal behavior and report it. This way, attacks could be detected even though the meter is unable to mitigate. In this case, meters count and report statistics, which are examined by the NADS controller application with more complex algorithms. The SDN controller may then mitigate anomalies using any kind of network reconfiguration.

IV. Case Study

We now evaluate the NADS concept in a case study. This case study is based on a real in-car communication matrix in a zonal Ethernet topology. We also used this communication matrix in previous work [14] [15]. The network contains TSN forwarding and filtering on each port. Our simulation environment consist of OMNeT++ discrete event simulator[1], INET framework[2] and our public open-source simulation models[3] CoRE4INET, FiCo4OMNeT and SignalsAndGateways.

A. Topology & Scenarios

As shown in figure 3 the network topology is split into nine zones (3xFront, 3xCenter, 3xRear) and contains four

[1] omnetpp.org [2] inet.omnetpp.org [3] sim.core-rg.de

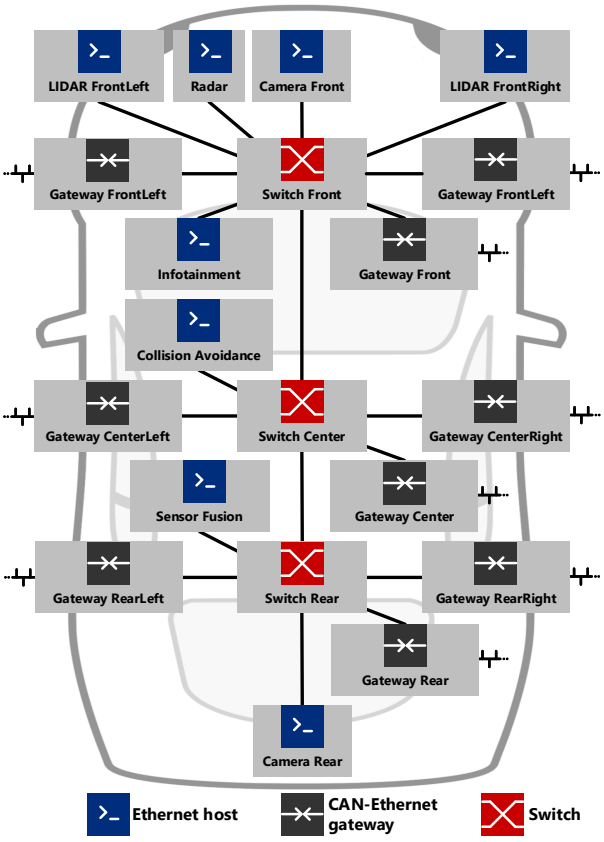

Fig. 3. Network topology of the simulation-based case study

types of devices (CAN hosts, Ethernet hosts, switches, CAN-Ethernet gateways). CAN hosts are connected via busses to the closest gateway depending on their physical placement. Each gateway represents a zone of the topology. The CAN traffic is generated from an anonymous communication matrix of a real production car. CAN messages sent between zones are transported via an Ethernet backbone. In addition to the gateways, Ethernet hosts such as cameras, LIDARs, a radar, and ECUs are using this backbone for communication. The backbone consists of three switches interconnected via $100\,\mathrm{Mbit/s}$ Ethernet links.

There are three traffic classes in this network:

- **Safety critical (pcp 7):** Synchronous frames with the highest priority are sent from "Radar" and "Sensor Fusion" to "Collision Avoidance". A static Time Division Multiple Access (TDMA) schedule in each network device implements this synchronous traffic by configuration of Time-Aware Shaping (TAS) and its gate control lists.
- **Data streams (pcp 5):** Cameras and LIDARs are streaming data with priority 5 to the "Sensor Fusion" ECU. For this traffic a certain bandwidth is reserved along its paths and a Credit Based Shaping (CBS) is shaping egress.
- **CAN tunneling (pcp 0,1,3,6):** CAN messages that are exchanged between devices located in different zones are send via Ethernet. There are a total of 416 different CAN IDs generated by the CAN hosts of which 201 are transported over the backbone network. They are sent with four different priorities mapped to their criticality.

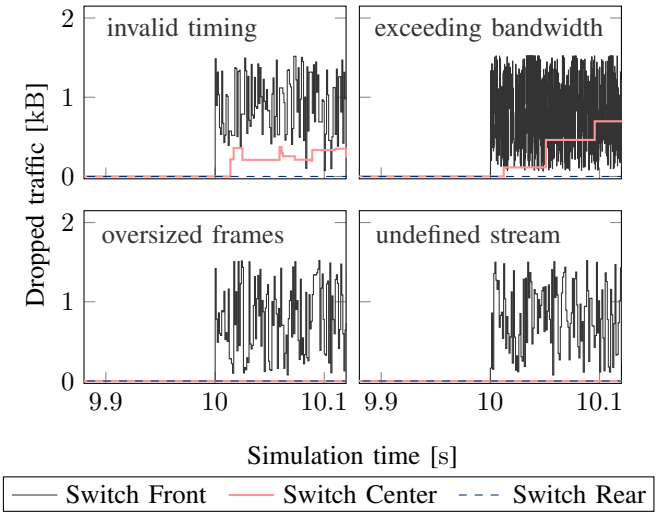

Fig. 4. Traffic dropped in the switches by different ingress control types before ($< 10\,\mathrm{s}$) and during attack ($\geq 10\,\mathrm{s}$)

Each port contains a Qci configuration for all incoming traffic classes. Any configuration uses known parameters of the network design to check its valid behavior. We use four different types of ingress control in this case study:

- **Timing:** This type uses the static TDMA schedule of the *safety critical* traffic. It is used to configure the gate control list that controls the state of the stream gates.
- **Bandwidth:** For the *data streams* a metering is configured to enforce a maximum incoming bandwidth [16].
- **Frame size:** Because a *CAN tunneling* frame contains always one CAN message the maximum size of these Ethernet frames is $64\,\mathrm{B}$. For the ingress control of this traffic a meter drops all frames greater than $64\,\mathrm{B}$.
- **Undefined stream/flow:** Ingress control drops traffic without a matching stream filter.

This case study inspects the simulation of four attack scenarios targeting the four ingress control types. In all scenarios the attack starts at $10\,\mathrm{s}$ simulation time and the size of injected attack frames is uniformly distributed. In each scenario a different device connected to "Switch Front" becomes the source of the attack:

- **Radar:** This attack uses the stream of the *safety critical* traffic to communicate with "Collision Avoidance".
- **Camera Front:** The second attack is executed by "cam_Front" using its *data stream*.
- **Gateway FrontLeft:** The third attack uses the *CAN tunnneling* stream between gateways.
- **Infotainment:** This attack does not use an established stream. It tries to get in contact with any devices.

B. Detection

The detection performance of the NADS concept depends on the Qci configuration. In this case study, the ingress control always drops frames violating the Qci configuration. Therefore the indicator for anomaly detection is dropping of frames.

Figure 4 presents a simulation period for each of the four ingress control type scenarios. They show the number

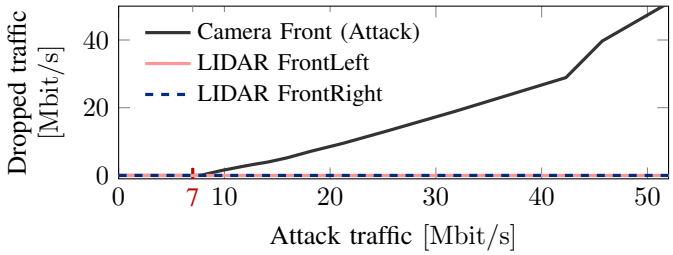

Fig. 5. Dropped traffic in "Switch Front" per data stream in relation to the attack stream traffic size produced by "Camera Front" (Valid: $7\,\mathrm{Mbit/s}$)

of dropped bytes per switch in a simulation time window between $9.9\,\mathrm{s}$ and $10.1\,\mathrm{s}$. In all four scenarios the ingress control of "Switch Front" starts to drop frames when the attack starts at $10\,\mathrm{s}$. In case of the timing and bandwidth ingress control types "Switch Center" detects some frames that pass the checks of the first switch because of ingress control limitations. Because dropping of frames is also mitigating the attack impact "Switch Rear" detects no invalid behavior. When this information is used e.g. to increment dropped bytes counters, the NADS controller application can detect abnormal behavior of individual flows. This demonstrates the detection of invalid behavior of individual streams. Further, there is no dropping before the attack starts. Which shows, with suitable configuration there are no false positive detections.

In case of timing, bandwidth and frame size ingress control, some attack frames are reaching the target host. The attack frames that are passing the ingress control show the limitations of the detection. Attack patterns that comply to ingress control rules are not detected.

The *timing* ingress control allows incoming frames in static time windows using a globally synchronized time. When a frame is of suitable size or send with suitable timing it can pass the ingress control of the first switch. Only frames that have the size and send time of the original flow are not detected by any switch and reach the target host.

Bandwidth ingress control limits the incoming bandwidth. When an attack is not exceeding the allowed bandwidth it can pass the ingress control and is not detected. Figure 5 shows an evaluation of such a bandwidth meter. The original data stream send by "Camera Front" has a bandwidth of ca. $7\,\mathrm{Mbit/s}$. Dropping of frames is related to the attack bandwidth and how much it exceeds the original stream. Frames of other streams are not dropped in any case. When the bandwidth of an attack is lower than the original stream an attack is not detected.

The frame size ingress control does not drop frames with an allowed size (in this case $64\,\mathrm{B}$). If the attack consists only of such frames it is not detected

The results show that a link-layer anomaly detection is possible with Qci. With suitable configuration the established NADS performs with zero false positive detections. Yet, there are false negatives that could not be detected. To reduce the number of false negatives, new ingress control types can be combined or implemented. For example, a combination of frame size and timing check would detect more invalid behavior.

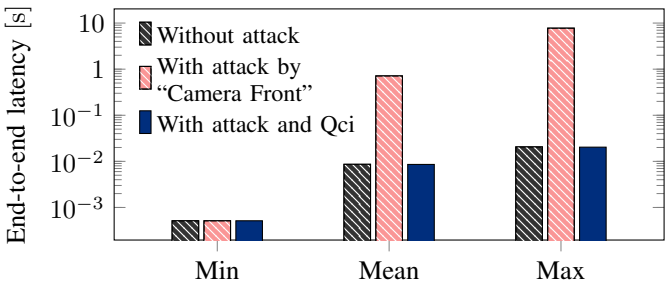

Fig. 6. Minimum, mean and maximum end-to-end latency of the **data stream** from "LIDAR FrontLeft" to "Sensor Fusion" in three different cases

C. Mitigation

The NADS concept comes with advantages for mitigation.

One advantage is the dropping of frames by Qci. By dropping frames of the compromised stream, the impact of an attack on other streams on the same path is reduced.

Figure 6 shows the minimum, mean and maximum end-to-end latency of the data stream send by "LIDAR FrontLeft" to "Sensor Fusion" in three different cases. In the first case there is no invalid traffic in the car network. In the second case "Camera Front" starts a DoS attack but Qci is disabled on all ports. In the last case Qci is enabled. Because the queues in our simulations are configured with an infinit size, no frames are dropped during the forwarding process. For this reason, the data stream does not lose any frames due to buffer overflow. The result is, that a DoS on a concurrent stream without Qci results in a maximum end-to-end latency of over 7 s after 20 s simulation. The same attack in a simulation with Qci enabled mitigates the impact of the DoS on this data stream.

Another mitigation advantage is that a SDN controller is able to reconfigure, disable and enable flows to protect the attack target. Further, a controller can also reconfigure the TSN forwarding and ingress control. Discarding unknown streams also protects the SDN controller from DoS attacks. The programming option introduces by the combination of SDN and TSN preserves the ability to inspecting this traffic. E.g., on detection a controller can configure Qci to let every 10th frame pass and install a flow rule to forward this undefined traffic for further inspection. With the information held by the NADS controller application other complex inspections can be used as base for reconfiguration of the in-car network. An example of this could be a fail-save networking mode.

V. Conclusion & Outlook

Security for in-car networks is a critical challenge for future connected automobiles. Many constrained controllers with predictable communication patterns in cars will largely profit from protective functions in the network core, which are most efficiently implemented on the lowest possible layer.

We devised and analysed a link-layer scheme for sanitizing in-car networks based on the Qci metering of TSN ingress control. We could show that our approach can detect significant classes of network anomalies including DDoS without generating falsely positive alarms. This early work is promising in particular because protective measures are solely built on preconfigured network designs that are common in cars. We also proposed a companion SDN controller that can reconfigure the in-car network under attack, or report incidents to a cloud infrastructure.

In future work, we will further evaluate this concept. Based on a large collection of real-world attack traces, we want to derive a systematic assessment and classification of countermeasures based on our proposed method. We also want to test its feasibility in our real car prototype equipped with Ethernet TSN ingress control and SDN controller.

References

[1] S. Brunner, J. Roder, M. Kucera, and T. Waas, "Automotive E/E-Architecture Enhancements by Usage of Ethernet TSN," in *2017 13th Workshop on Intelligent Solutions in Embedded Systems (WISES)*. IEEE, 2017, pp. 9–13.

[2] S. Checkoway, D. Mccoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, and T. Kohno, "Comprehensive Experimental Analyses of Automotive Attack Surfaces," *USENIX Security*, 2011.

[3] P. Mundhenk, *Security for Automotive Electrical/Electronic (E/E) Architectures*. Göttingen: Cuvillier, Aug. 2017.

[4] Institute of Electrical and Electronics Engineers, "IEEE Standard for Local and Metropolitan Area Network–Bridges and Bridged Networks," IEEE, Standard, Jul. 2018.

[5] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Network Anomaly Detection: Methods, Systems and Tools," *IEEE Communications Surveys Tutorials*, vol. 16, no. 1, pp. 303–336, First 2014.

[6] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling Innovation in Campus Networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[7] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-Defined Networking: A Comprehensive Survey," *Proc. of the IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015.

[8] C. Miller and C. Valasek, "Remote Exploitation of an Unaltered Passenger Vehicle," *Black Hat USA*, vol. 2015, p. 91, 2015.

[9] G. K. Rajbahadur, A. J. Malton, A. Walenstein, and A. E. Hassan, "A Survey of Anomaly Detection for Connected Vehicle Cybersecurity and Safety," in *2018 IEEE Intelligent Vehicles Symp. (IV)*. IEEE, Jun. 2018.

[10] K. Halba, C. Mahmoudi, and E. Griffor, "Robust Safety for Autonomous Vehicles through Reconfigurable Networking," in *Proc. of the 2nd Int. Workshop on Safe Control of Autonomous Vehicles*, ser. Electronic Proc. in Theoretical Computer Science, vol. 269. Open Publishing Association, 2018, pp. 48–58.

[11] T. Gerhard, T. Kobzan, I. Blöcher, and M. Hendel, "Software-defined Flow Reservation: Configuring IEEE 802.1Q Time-Sensitive Networks by the Use of Software-Defined Networking," in *2019 24th IEEE Int. Conference on Emerging Technologies and Factory Automation (ETFA)*, Sep. 2019, pp. 216–223.

[12] T. Häckel, P. Meyer, F. Korf, and T. C. Schmidt, "Software-Defined Networks Supporting Time-Sensitive In-Vehicular Communication," in *2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring)*. Piscataway, NJ, USA: IEEE Press, Apr. 2019, pp. 1–5.

[13] F. Langer, F. Schüppel, and L. Stahlbock, "Establishing an Automotive Cyber Defense Center," in *17th escar Europe : Embedded Security in Cars*, 2019.

[14] P. Meyer, F. Korf, T. Steinbach, and T. C. Schmidt, "Simulation of Mixed Critical In-vehicular Networks," in *Recent Advances in Network Simulation*. Springer, 2019, pp. 317–345.

[15] M. Cakir, T. Häckel, S. Reider, P. Meyer, F. Korf, and T. C. Schmidt, "A QoS Aware Approach to Service-Oriented Communication in Future Automotive Networks," in *2019 IEEE Vehicular Networking Conference (VNC)*, Dec. 2019.

[16] P. Meyer, T. Häckel, F. Korf, and T. C. Schmidt, "DoS Protection through Credit Based Metering - Simulation-Based Evaluation for Time-Sensitive Networking in Cars," in *Proc. of the 6th Int. OMNeT++ Community Summit 2019*, ser. EPiC Ser. in Comp., M. Zongo, A. Virdis, V. Vesely, Z. Vatandas, A. Udugama, K. Kuladinithi, M. Kirsche, and A. Förster, Eds., vol. 66. EasyChair, Dec. 2019, pp. 52–59.