INVITED: Towards Fail-Operational Ethernet Based In-Vehicle Networks

Mischa Möstl
IDA - Institute of Computer and Network Engieering
Technische Universität Braunschweig
moestl@ida.ing.tu-bs.de

Daniel Thiele
IDA - Institute of Computer and Network Engieering
Technische Universität Braunschweig
thiele@ida.ing.tu-bs.de

Rolf Ernst
IDA - Institute of Computer and Network Engieering
Technische Universität Braunschweig
ernst@ida.ing.tu-bs.de

ABSTRACT

In the future, vehicles are expected to act more and more autonomously. The transition towards highly automated and autonomous driving will push the safety requirements for in-vehicle networks. Such networks must support isolation between mixed-critical traffic (e.g. critical control and non-critical infotainment) and must be fail-operational. This paper will present new concepts and mechanisms to achieve these goals in Ethernet-based networks. It will cover advanced topics such as software defined networking (SDN) to implement isolation, fault recovery, and controlled degradation, e.g. to maintain (degraded) operation until the driver takes over or to reach a safe stop.

1. INTRODUCTION

Ever increasing bandwidth requirements from infotainment applications with best-effort delivery on the one hand and quickly growing sensor traffic with tight network latency requirements push traditional bus-based networks over their limits. Especially with the advent of highly automated driving high resolution and redundant image sensors for robust environment perception require bounded latencies at high volume data transmissions. A further trend is an increasingly complex low latency between different car domains ranging from legacy to highly interactive driving functions and control loops. The idea is to transition to switched Ethernet due to its high grade of standardization and bandwidths that grow with technology, i.e. from 100 Mbit/s to 1 Gbit/s. Furthermore Ethernet promises open network capabilities with IP as a standard protocol on top of approved technologies. Shared technology cost is a key driver as it avoids domain specific and expensive technologies as, e.g. a next generation FlexRay technology.

However the traffic in automotive systems is often time and safety critical, since reliable service delivery depends on sufficiently early delivery of data, i.e. before a certain deadline. Nevertheless, some traffic streams are more critical than others or have no safety-requirements at all, i.e. are best effort streams where deadline misses are undesirable but not critical or have no deadline constraint. This paper is organized as follows, in Section 2 we elaborate on possible source of interference in switched Ethernet network that endanger the safety of a design. Subsequently the safety implications on switched Ethernet in fail-operational designs are considered in Section 3. Section 4 concludes

DAC '16, June 05 - 09, 2016, Austin, TX, USA

ISBN 978-1-4503-4236-0/16/06…$15.00

DOI: http://dx.doi.org/10.1145/2897937.2905021

2. THE INTERFERENCE CHALLENGE

Systems with more than one criticality are commonly referred to as mixed-critical systems. In mixed-critical as well as safety-critical systems in general, freedom of interference between the criticalities must be established in order to be able to argue the safety of the system. In switched Ethernet networks the isolation property must be investigated at multiple points, starting from the protocol stacks that perform packaging and are responsible for message routing down to the network hardware, i.e. switches and links.

2.1 Switch Considerations

For sufficient isolation between traffic streams the switch matters to a great deal if traffic with real-time constraints is sent over the network. One challenge for real-time Ethernet designs is that freedom of implementation for the switches exists between different vendors as long as the switch is in accordance with the respective set of IEEE 802.1 standards. To the greatest possible extent, we focus our assessment of switch characteristics on generic or where not possible on typical Ethernet switch structures.

Ethernet switches receive frames on individual inputs referred to as ingress stage and emit frames from the egress stage. In the egress stage different queues can be present to perform link arbitration, i.e. according to IEEE 802.1Q priorities. Furthermore this queues again can be governed by different kinds of schedulers, depending on the switch and its configuration. Conceptually ingress and egress are connected via the switch fabric, realizing the forwarding of frames from ingress to egress. This process is conducted by a switch control logic by means of a forwarding table, mapping frames to egress stages based on their destination addresses. However, for the implementation of the fabric various possibilities exist, i.e. where buffers are located or necessary and whether arbitration of the fabric is necessary. In cases where arbitration of the fabric is necessary, this has to happen with a strict upper bound and clear separation of frame priorities for real-time capability of a switch. In this paper we focus on conceptually typical store-and-forward implementations of switches as depicted in Figure 1. For real-time capability, the ingress and egress stages must be able to store/read

frames to/from a global shared frame buffer contention free or with bounded overhead even if all switch ports are busy, i.e. sending and receiving. Since frames are stored in the global shared memory, this implies that along the datapath only pointers to frames in this memory are passed rather than the whole frame itself.

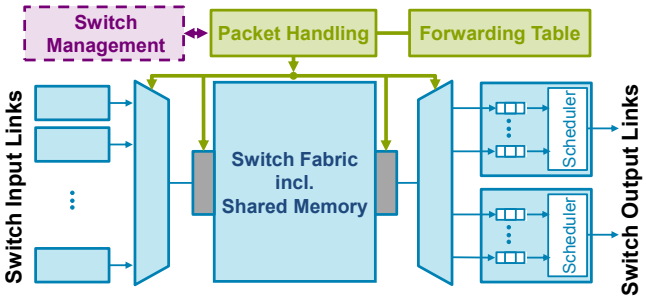

Figure 1: Typical Store-and-Forward Switch Architecture

Switch management functionality can be implemented in hardware, software or as a mixture of both. While high level protocols, that can also involve configuration of the switch, are usually handled in software others such as Medium Access Control (MAC) address learning, i.e. entering the destination port for a MAC address into the forwarding table based on received frames with the respective source address, can be hardware based. A further task of packet handling as seen in Figure 1 is to check for layer 2 protocol conformity. The goal of the remainder of this section is to raise awareness for the crucial points along the datapath and the hazards there that can cause interference between different traffic streams.

MAC address management.

Our first focus is on the forwarding table, how it is accessed and managed. Forwarding tables usually exhibit a predictable timing behavior since they are typically implemented as indexed data structures with constant access latency, e.g. as a hash table. Therefore the 48 bits wide MAC addresses are hashed by a vendor specific and often undisclosed hash function, resulting in an index space that is typically in the order of 2^7 to 2^{10} indexes. These indexes are then used to perform the lookup in the forwarding table. Obviously hash collisions can appear here due to the (intended) reduction of the value space. What is particularly challenging here is that OEMs might obtain MAC address pools and again allocate these to their suppliers in segments leading to constellations where devices with conflicting resulting hashes end up int the same car. To minimize the effect of hash collisions, often set-associative tables are used where more than one entry per index can be stored. Yet this does not fully mitigate the problem since limited index space or limited associativity of an index can always lead to forwarding table collisions, i.e. two or more values would be written to the same place in memory.

While in a commodity network this is usually not a problem, since address can be evicted from the table and learned again this leads to a loss of predictable timing in real-time networks. The reason therefore is that the usual behavior when an address can not be found in the forwarding table is to flood the frame on all egress ports until this destination address is seen as a source address and again written to the table. However, flooding a frame on an unintended link can cause congestion there, due to the unexpected frame, leading to interference.

In order to prevent that critical MAC addresses are evicted from the forwarding table, appropriate MAC address management within each Layer 2 network , i.e. collision domain, is necessary. Possible approaches are (managed) forwarding tables with static entries for critical traffic streams, or a fully associative forwarding table. However, even a fully associative forwarding table remains a security concern.

Queuing buffers & Flow Control.

The second possible spot for interference between different traffic streams is the frame buffer of the switch. As Ethernet typically embraces store-and-forward frame passing the limited buffer space of a switch can cause interference among frames since occupied buffers can lead to message drops. Furthermore this aspect is tightly linked to the effects introduced by the frame schedulers in the egress stage. Due to the fact that schedulers as for instance Credit Based Shaping (CBS) in Audio-Video Bridging (AVB) limit the rate at which frames can be sent out again, the introduced delay increases the time the frame occupies the queuing buffer. This leads to increased same-priority blocking in an egress port when multiple streams from different ingress ports compete for the same egress port. Furthermore it also contributes to increased delays for all competing traffic streams.

Blocking effects lead to unintended dependencies between traffic streams over the same switch. Since backlog of packets can ultimately lead to the occupation of the whole global frame buffer and thus inevitably to frame drop in the ingress ports since no memory can be allocated for arriving frames. This is a particular source of interference and not only if frames of higher criticality are dropped at the ingress because of lower criticality frames that filled up the frame buffer. Here a system-level analysis is necessary to ascertain that even under worst-case conditions no buffer overruns can happen. This is possible with common performance analysis tools and mechanisms, e.g. [8]. Alternatively the analysis results can be used to find a configuration of the switch such that the memory can be partitioned in a way that critical frames always have sufficient memory available. However, this is only possible if the switch allows a partitioning of memory that explicitly ties memory partitions to individual priorities or queues and ports. The effects of different schedulers on delay and thus generated backlog for switched Ethernet have been studied extensively in [21, 20, 18, 6, 9]. The importance thus not only is on the appropriate port scheduler but also on the fact if and how buffer space can be reserved for specific queues that transport high critical streams to ensure freedom from interference.

Especially challenging with Ethernet here is the limited number of only up to eight priorities in 802.1Q compared to e.g. Controller Area Network (CAN) with 2^{29} priority levels. Striking in this context is that not necessarily all of the priorities defined in IEEE 802.1Q must be mapped to independent queues making the situation even more complicated, as it can lead to head-of-line blocking. This implies interference between the priority level mapped to the same queue. These facts make a fine grained separation of criticality by priority impossible, which suggests that a different design strategy must be employed here. Typically switches support at least three distinct priority classes with separate priority queues per class. Here a design scheme that can provide hard real-time guarantees for time critical traffic can be combined with flexibility for less time critical traffic due to

the separation properties of priority-based scheduling.

Therefore, we suggest to classify the traffic according to its time criticality, starting with the most urgent frames on the highest priority, e.g. control traffic with a small load share to avoid shaping on this level. On the intermediate priority level more bulky medium time-critical traffic can be distributed, possibly with traffic shaping enabled to leave resources for lower priorities. A possible candidate could be camera traffic. Experiments with realistic automation control traffic show that this approach is reasonable and achieves the required latency bounds required for automotive applications [9][19]. For less time-critical traffic or traffic that is not that well defined, a "less-than worst-case" design principle is then possible on the lower priorities, just like found in ECU or CAN bus designs. Methods to validate such a design could for instance be simulative or measurement based since they are not guaranteed to capture the worst-case but allow early design-space exploration. The threshold on which priority levels worst-case design is actually applied and where the transition to a less-than worst-case design happens can of course be altered depending on the criticality of the traffic and the individual functionality backed by the Ethernet network.

2.2 Protocol Considerations

Besides the interference points in the switches themselves the network level also exhibits further sources of interference or exposes obstacles for a predictable network design.

2.2.1 Communication Protocols

Firstly, complex protocol choices have to be made in order to decide how to transport and address the data over the network. There exist a wide range of possibilities which all expose different advantages but also drawbacks that can lead to interference or loss of e.g. time predictability. While fixed length UDP packet communication on top of IP and MAC protocols allows real-time predictability, this choice of protocols might be unhandy with respect to applications that need to perform (dynamic) service discovery or transport varying length payloads. Thus middleware protocols such as SOME/IP are defined in the scope of AUTOSAR with the aim of making Ethernet communication manageable for automotive use cases [4]. However, the increasing complexity that comes with these protocols can make timing analysis of the network a hard challenge [15] especially if they also introduce connection oriented protocols such as TCP. In the case of TCP, the sliding window technique that is used for flow and congestion control as well as Transport Control Protocol (TCP)'s three-way handshake make TCP streams hard to model.

2.2.2 Packaging

Another point of possible interference on switched Ethernet is the packaging of data. This is as a particular obstacle if the Ethernet network is used as a high-capacity backbone for communication between other sub networks, often legacy buses. The challenge here is that a change in the communication paradigm happens. While signal communication on buses is typically a publisher-subscriber communication, i.e. the receiving party decides whether the message should be processed, on switched Ethernet dedicated sender-receiver pairs exist, i.e. the receiver(s) of a message have to be explicitly specified at the source.

In such setups the task of translating IDs that identify the data, e.g. in CAN frames, to destination addresses on the Ethernet network is fulfilled by bus-Ethernet gateways. The same happens vice versa if signals from Ethernet frames are fed onto other can buses, attached to the receiving gateway.

To transport CAN frames (i.e. their ID and signal payload) over an Ethernet network, various strategies are possible. The two extrema are: (a) sending each CAN frame via a dedicated Ethernet frame and (b) multiplexing (packing) as many CAN frames as possible into an Ethernet frame. While the first approach intuitively results in low latencies (each CAN frame is sent immediately), it has a large Ethernet protocol overhead ($\approx 90\%$) and introduces a high load on the domain gateways and the Ethernet (packing and sending many small frames). The latter approach only has a minimal Ethernet protocol overhead, however, suffers from potentially large latencies as Ethernet frames are only sent when they are full, which might take some time. As a consequence various ways to group signals or can frames exist as well as different triggering mechanisms when a buffer for a group of signals (or frames) is released [11]. Challenging in these setups is that all variants that group frames suffer from the fact that they introduce dependencies between individual and otherwise independent frames which can lead to interference, e.g. critical impact on response-times.

In summary interference on the domain gateway can either originate from blocking in one of the buffers that awaits its trigger condition or from packed and already triggered frames being in the transmit queue of the gateway before the respective frame. An overview of different packaging strategies and their impact on end-to-end latencies is presented in [23].

3. TOWARDS FAIL-OPERATIONAL

Ethernet networks are a promising candidate for future vehicle networks because they provide the bandwidth required for future automotive Advanced Driver Assistance System (ADAS) systems. Yet, these systems strive more towards highly automated and autonomous driving. However, with increasing autonomy of the vehicle and longer reaction or take-over times of the driver, the vehicle network must be capable of delivering fault-free service. This property adds another challenge to the design goals of the network, as it requires fail-operational properties rather than fail-safe as in legacy driver assistance systems. This is due-to the longer reaction times until a driver can be brought back into the loop to fully perceive the current driving situation and react accordingly, which is not possible by simply switching off a faulty functionality. In the context of network and platform design, this means that faults have to be considered in order to derive mechanisms for fail-operational capability. In order to address fault events and their impact on the network, a classification of the fault types and an understanding of their probability of occurrence is necessary.

Here we distinguish fault events according to their behavior over time: *transient* and *permanent* faults. Faults are classified as transient if normal, i.e. fault free, operation is restored after a bounded time interval without any special repair effort. Permanent faults leave an affected component in the corrupted, i.e. faulty, state for the rest of a system's lifetime or until repair.

3.1 Transient Fault Effects

Typical transient faults are bit flips due to electromagnetic interference or other sources. This leads to frame loss since the frames are detected as corrupted frames by a Cyclic Redundancy Check (CRC) logic on the receiving switch. The

reason why this type of fault is particularly important when designing a fail-operational network sub system is the Bit Error Rate (BER) one can expect. For the two-wire Ethernet standard BroadR Reach for example a Physical Layer Transceiver (PHY) chip is compliant with the standard if it exhibits a BER of less than or equal to 10^{-10} [7]. While appearing reasonably low for a single link this corresponds to a bit error approximately every two minutes on a 100 Mbit/s Ethernet link. Such a high transient error frequency is only acceptable if transient error handling is part of regular operation including timing constraints.

As fault events are not a rare event, a fail-operational strategy should be efficient and simple w.r.t. the introduced overhead. The situation is comparable to transmissions on bus based connections. These also experience electromagnetic interference and thus also frame loss. CAN and FlexRay are designed with this challenge in mind and thus handle bit errors directly on the protocol level. For small overhead strong error detection coding is combined with repeated transmission in case of an error. This approach mitigates the risk of undetected (residual) errors at low overhead in regular operation. For instance, if an erroneous bit is detected on CAN either by the sender or through the CRC sequence at the end of a frame, the error is directly signaled to all bus participants which then can discard the message. Assuming only a transient fault on the wire, the sending node can then again enter arbitration to retransmit the frame. In case of Ethernet, a resend mechanism must cover the complete network.

Furthermore, large volume data are typically not sent in bare Ethernet frames but are encapsulated in a higher level transport protocol that can fragment messages over more than one Ethernet frame. Thus further fault classes can contribute to packet loss in an Ethernet network. One particular issue in this context is tail drop due to frame buffer overrun as discussed in Section 2.1. Consequently, switched Ethernet requires an end-to-end error-control mechanism that addresses transient fault scenarios, tail drop and transmission bit errors. This can be found in higher level transport protocols such as Automatic Repeat ReQuest (ARQ) that provide reliable data transfer with guaranteed in-order delivery [17] [13].

Obviously, for or a fail-operational real-time network architecture the transient error correction time must be predictable in order to also give guarantees under the presence of transmission errors. To predictably analyze the response times of ARQ mechanisms for bus-based communication such as CAN, several mechanisms exist, however, they are not applicable to switched Ethernet as for bus-based communication all participants directly detect the packet corruption. In a switched network the unsuccessful delivery of a packet is only detected after an acknowledgment, timeout or negative acknowledgment (depending on the actual protocol), i.e. detection efforts by the sender or receiver. In the literature various forms of ARQ protocols are described (cf. [17] or [13]), among which the most popular are *Stop and Wait*, *Go-back-N* and *Selective Repeat*.

Stop and Wait is illustrated in Figure 2. For Stop and Wait ARQ the receiver acknowledges every packet of the sender with a positive acknowledgment packet (ACK). Only after the sender has received the ACK a new packet is emitted. If no ACK is received after a timeout of t_{out}, the packet is retransmitted.

Go-back-N ARQ is an extension thereof and is illustrated in Figure 3. In go-back-n ARQ there can be up to n_{SW}

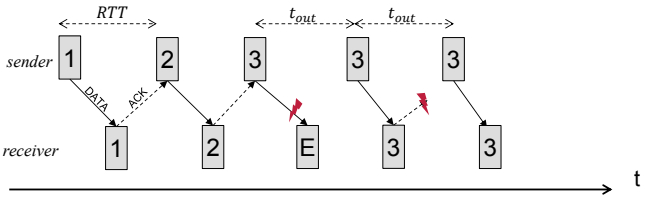

Figure 2: Example for Stop and Wait ARQ (c.f. [5])

packets in-flight from the sender that are unacknowledged by the receiver. If a data packet is lost, the sender will resend the last packets again in-order, starting with the unacknowledged packet, i.e. max. n_{SW} packets.

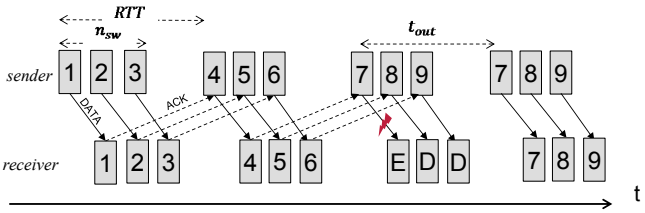

Figure 3: Example of Go-Back-N ARQ (c.f. [5])

For a worst-case analysis of both protocols the worst-case round-trip time RTT has to be computed. For Stop and Wait ARQ this is the longest time between release of the frame from the application to reception at the sink. The reason therefore is that the RTT dictates the minimum distance in time two packet releases can be apart in Stop and Wait ARQ. Note that the artificial delays that are introduced by the ARQ protocol alter the event models that describe the emission of packets from the application. In cases where no ARQ is used these event models would be used for a worst-case analysis. In the case of Go-Back-N the RTT, however, describes the time between sending the first packet of a window of up to n_{SW} packets and the reception of the Acknowledgement (ACK) for this first packet. Knowledge of the RTT then allows to alter the event models such that at most n_{SW} packets are in flight.

Despite the fact that ARQ protocols have been regarded as inapplicable for hard real-time communication in the past since in the worst-case the latency must be bounded, Axer et.al. in [5] present and efficient end-to-end latency analysis based on this considerations for Stop and Wait as well as Go-Back-N ARQ enabling low overhead selective repeat in a few milliseconds. The latency bounds provided are for the fault free case as well as for lossy communication with a parameterizable amount of lost packets k. The selective repeat protocol as used in TCP however is more complex. To the extent of the authors knowledge so far no analysis that can provide formal worst-case bounds exists.

Predictable and fast end-to-end handling of transient errors in switched Ethernet networks can match that of existing bus standards. This is highly important as it avoids special error modes for regular operation.

3.2 Permanent Fault Effects

While, due to their high frequency, transient faults are a primary concern, permanent fault effect handling is also needed to reach the required functional safety. However, since it is many orders of magintude more unlikely, it may include switching to a fail operational mode.

In the Ethernet context permanent faults are effects that leave a specific path along switches and links in an erroneous state, such that even if continuous resending of messages, e.g. due to an ARQ protocol, is unable to convey the message along the path. The affected element of such a path can be either the switch or the connection in between, i.e. the cabling including the connectors. Permanent faults are thus all types of influence that leaves an element of a route permanently unusable. This explicitly also includes faulty switches as they lead to permanent loss of all routes that traverse the particular switch. Although this effects are less likely by orders of magnitude compared to transient faults, but they are still significant over the lifetime of a fleet of cars.

First of all, in order to be able to handle such situations faults must be detectable to mitigate their possibly safety-critical effects. Errors can be detected in each switch in the network by monitoring packet loss and corruption. Possible detection mechanisms are error detecting code fields that are part of packets anyway, e.g. by monitoring the CRCs of Ethernet frame and User Datagram Protocol (UDP) for that purpose. Alternatively timeout-based mechanisms are possible if guaranteed arrival times of packets are known, e.g. form worst-case analysis. Another well known protocol to detect link faults is link-based bidirectional detection as described in RFC 5880 [10].

Second, to consider any real-time network that is subject to strict fail-operational timing requirements, an upper bound on the time between the occurrence of the fault and until it is resolved is necessary. This time is denominated as fault recovery time. For critical automotive control applications fault recovery must thus be fast. The AVnu Alliance suggests fault recovery times of less than 100 ms [16], whereas other sources even suggest times of less than 50 ms [2].

In order to survive a permanent network error routing around the defective network element is necessary. This is obviously only possible if sufficient redundancy, i.e. alternative paths, exist in the network to reach all end stations with a fail-operational requirement. The first option to achieve this and to fulfill the fault recovery time requirement is to design the network with direct dual modular redundancy with error detection for the required terminals. This implies that each required stream is sent over two fully independent routes from the source to its sink. That is possible and used in AFDX [3] (and as an option in FlexRay buses). However, this is not cost effective as the redundancy is in the physical network as well as the logical layers above, i.e. packets are also duplicated. Especially in the fault free case this setup leads to a significant waste of resources since the physical network needs to provide twice as much capacity as logically can be used. As an alternative to this approach physical layer redundancy can be used in an on demand fashion for streams that require to stay operational after a fault occurs, i.e reconfigure the network such that the critical streams utilize the remaining links.

3.2.1 Network Reconfiguration

Network reconfiguration allows the network to return to an operational state in the presence and detection of a component (link or switch) failure by switching to an alternative, valid configuration. After a successful reconfiguration, the network resumes regular or degraded operation and lost packets can be retransmitted e.g. on the basis ofARQ as described in Section 3.1. This could imply that after a reconfiguration (after failure), the network operates with reduced performance where some traffic streams of low importance are disabled in order to provide the required guaranteed service for highly critical streams.

There are two requirements to reconfiguration, the availability of redundant connections, and the ability to adapt the network service to using the redundancy. The first problem can be modeled as a standard graph problem with many available solutions, the second requirement needs a network control mechanism that is fast and reliable enough to be used as part of a fail-operational mechanism. Standard Ethernet would start a set-up sequence to find new routes but this is not fast nor predictable enough for our application.

An example of a possible network reconfiguration is Software Defined Networking (SDN), which is an emerging topic in the time-sensitive Ethernet community. SDN leverages that a network comprises a control and a data plane and introduces network-wide interfaces for managing the control plane. While the data plane is responsible for frame forwarding based on forwarding tables, the control plane is in charge of configuring the data plane accordingly. In SDN the control plane is not distributed as the data plane, i.e. present in every switch, it is instead merged in a (logically) centralized SDN controller. However, SDN is not a fault-recovery technology in the first place, it is rather a generic network management framework. In the context of fault tolerance SDN can be used to reroute the critical streams (after a failure) via different links. The SDN controller receives the reconfiguration requests, grants or dismisses them and afterwards issues reconfiguration commands to the data plane, i.e. mainly to update the forwarding tables accordingly. The reconfiguration commands are then put into action by SDN agents that are able to control the switch fabrics, e.g. are part of the switch management. Controller redundancy can be added to protect against controller faults. An comprehensive survey of SDN techniques that is yet not particularly targeted at real-time systems is given by Kreutz et. al. in [12]. It covers interfaces and controllers, current fields of application as well as standardization efforts.

3.2.2 SDN and Real-Time Guarantees

The SDN idea can also be found in the upcoming Ethernet Time-Sensitive Networking (TSN) standards family which aims to introduce real-time capability to IEEE 802.1. With the IEEE P802.1Qac draft standard a (central) path computation element (PCE) is introduced to explicitly compute (potentially redundant) paths [24]. Where the PCE fulfills the functionality of an SDN controller. Further, in IEEE P802.1Qcc centralized network configuration for time-sensitive networks is proposed [25].

A standard for a SDN interface in commodity networks is OpenFlow [14]. However, it is built on TCP as transport protocol, which is unsuitable for latency critical real-time traffic, as pointed out before.

In consequence other protocols are necessary to be able to obtain formal worst-case guarantees for the configuration latency R^+_{SDN}. The fault-recovery time then comprises the time for fault detection and the configuration latency.

In contrast to OpenFlow's TCP-based protocols, [22] proposes a UDP-based protocol for network reconfiguration. As illustrated in Figure 4, the contributing factors for R^+_{SDN} are the request by an SDN agent to the controller, the necessary computation and distribution time of a new configuration to all agents as well as an enabling the configuration at the requesting agent, to finally redirect the traffic stream. Yet for

real-time automotive networks precomputed solutions are preferable since they upper bound the latency τ_c^C of the SDN controller. Possible precomputations are for example all possible fault scenarios for one failed link. Such sim-

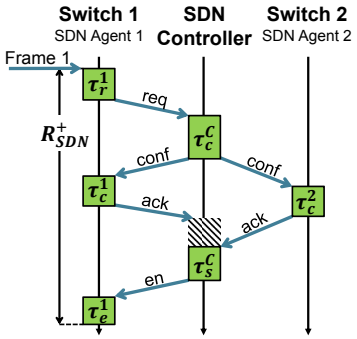

Figure 4: SDN configuration flow (cf. [22])

ple UDP based protocols can then be analyzed with performance analysis methods. A full evaluation thereof is available in [22]. It further demonstrates the suitability of the SDN concept for real-time networks and for fault recovery in such networks. SDN is also a candidate to improve network security in case of denial-of-service attacks on network terminals. This application is investigated in the European SAFURE project [1].

4. CONCLUSION

Switched Ethernet is a suitable basis for future automotive applications requiring fail-operational behavior. However, many traps in switch and protocol design require a systematic design and verification approach to avoid violations of safety guarantees. Robust network operation with real-time transient error handling is possible using low overhead end-to-end protocols. Permanent errors and security challenges can be addressed using appropriate SDN techniques with real-time capabilities. These topics require further research, as in the European project SAFURE [1].

5. ACKNOWLEDGEMENTS

This work has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 644080.

6. REFERENCES

[1] SAFURE - Safety And Security By Design For Interconnected Mixed-Critical Cyber-Physical Systems. https://safure.eu/.

[2] N. L. M. v. Adrichem, B. J. v. Asten, and F. A. Kuipers. Fast Recovery in Software-Defined Networks. In *2014 Third European Workshop on Software Defined Networks (EWSDN)*, Sept. 2014.

[3] ARINC. *ARINC 664, P7 - Avionics Full-Duplex Switched Ethernet (AFDX) Network*, 2009.

[4] AUTOSAR. *AUTOSAR Specification Release 4.2*, 2015.

[5] P. Axer, D. Thiele, and R. Ernst. Formal timing analysis of automatic repeat request for switched real-time networks. In *2014 9th IEEE International Symposium on Industrial Embedded Systems (SIES)*, June 2014.

[6] P. Axer, D. Thiele, R. Ernst, and J. Diemer. Exploiting Shaper Context to Improve Performance Bounds of Ethernet AVB Networks. In *Proceedings of the 51st Annual Design Automation Conference*, DAC '14, New York, NY, USA, 2014. ACM.

[7] Broadcom Corporation. BroadR-Reach Physical Layer Transceiver Specification For Automotive Applications - Version 3.0, May 2014.

[8] J. Diemer, P. Axer, and Ernst, Rolf. Compositional Performance Analysis in Python with pyCPA. In *Proceedings of the 3rd International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems*, ECRTS, Pisa, July 2012.

[9] J. Diemer, J. Rox, R. Ernst, F. Chen, K. Kremer, and K. Richter. Exploring the worst-case timing of Ethernet AVB for industrial applications. In *IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society*, 2012.

[10] D. Katz and D. Ward. Bidirectional forwarding detection (BFD). RFC 5880, RFC Editor, June 2010.

[11] A. Kern, D. Reinhard, T. Streichert, and J. Teich. Gateway Strategies for Embedding of Automotive CAN-Frames into Ethernet-Packets and Vice Versa. In M. Berekovic, W. Fornaciari, U. Brinkschulte, and C. Silvano, editors, *Architecture of Computing Systems - ARCS 2011*, number 6566 in Lecture Notes in Computer Science. Springer Berlin Heidelberg, Jan. 2011.

[12] D. Kreutz, F. M. Ramos, P. Esteves Verissimo, C. Esteve Rothenberg, S. Azodolmolky, and S. Uhlig. Software-defined networking: A comprehensive survey. *proceedings of the IEEE*, 103(1), 2015.

[13] S. Lin, D. J. Costello, and M. J. Miller. Automatic-repeat-request error-control schemes. *IEEE Communications Magazine*, 22(12), Dec. 1984.

[14] Open Networking Foundation. Openflow. [Online]. Available: https://www.opennetworking.org/sdn-resources/openflow.

[15] J. R. Seyler, T. Streichert, M. Glaß, N. Navet, and J. Teich. Formal analysis of the startup delay of SOME/IP service discovery. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2015*, Mar. 2015.

[16] J. Takeuchi. Requirements for automotive AVB system profiles. Technical report, AVnu Alliance, April 2011.

[17] A. S. Tanenbaum. *Computer networks*. Prentice Hall PTR, 2003.

[18] D. Thiele, P. Axer, and R. Ernst. Improving Formal Timing Analysis of Switched Ethernet by Exploiting FIFO Scheduling. In *Proceedings of the 52Nd Annual Design Automation Conference*, DAC '15, New York, NY, USA, 2015. ACM.

[19] D. Thiele, P. Axer, R. Ernst, J. Diemer, and K. Richter. Cooperating on Real-Time Capable Ethernet Architecture in Vehicles. In *VDI Elektronik im Fahrzeug, Baden Baden*, October 2013.

[20] D. Thiele, P. Axer, R. Ernst, and J. R. Seyler. Improving Formal Timing Analysis of Switched Ethernet by Exploiting Traffic Stream Correlations. In *Proceedings of the 2014 International Conference on Hardware/Software Codesign and System Synthesis*, CODES '14, New York, NY, USA, 2014. ACM.

[21] D. Thiele, J. Diemer, P. Axer, R. Ernst, and J. Seyler. Improved Formal Worst-case Timing Analysis of Weighted Round Robin Scheduling for Ethernet. In *Proceedings of the Ninth IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*, CODES+ISSS '13, Piscataway, NJ, USA, 2013. IEEE Press.

[22] D. Thiele and R. Ernst. Formal Analysis Based Evaluation of Software Defined Networking for Time-Sensitive Ethernet. In *Proceedings of the Design Automation and Test in Europe*, DATE '16, 2016.

[23] D. Thiele, J. Schlatow, P. Axer, and R. Ernst. Formal Timing Analysis of CAN-to-Ethernet Gateway Strategies in Automotive Networks. *Real-Time Syst.*, 52(1), Jan. 2016.

[24] Time-Sensitive Networking Task Group of IEEE 802.1. P802.1Qca: Standard for Local and metropolitan area networks – Path Control and Reservation (Draft 2.1), June 2015.

[25] Time-Sensitive Networking Task Group of IEEE 802.1. P802.1Qcc: Standard for Local and metropolitan area networks – Stream Reservation Protocol (Draft 0.4), June 2015.