2014-01-0245
Published 04/01/2014

doi:10.4271/2014-01-0245
saepcelec.saejournals.org

Requirements on Real-Time-Capable Automotive Ethernet Architectures

Philip Axer, Daniel Thiele, and Rolf Ernst
Technical Univ of Braunschweig

Jonas Diemer, Simon Schliecker, and Kai R. Richter
Symtavision GmbH

ABSTRACT

Ethernet is the hottest candidate for future in-car communication architecture, promising much higher bandwidth, flexibility and reduced costs. In the coming years, Ethernet will likely evolve from a separate communication medium for special applications like surround-view cameras and infotainment to a central communication infrastructure as a backbone technology. To make this transition, many difficult design decisions have to be made in order to make the technology suitable for the stricter time and safety requirements of todays and future cars. There are a lot of potential real-time effects that must be taken into account.

To guide these design decisions, it is necessary to analyze the various architecture concepts with respect to load, performance and real-time capabilities. In this paper, we present different design space axes of Ethernet and propose a methodology of assessing and comparing them. This includes a formal worst-case timing analysis approach and a set of metrics that make the timing analysis results comprehensible and comparable across different design options. Using this foundation, we explore the design space for a typical automotive scenario. Specifically, we compare different topologies and switching strategies and their effect on the performance of the network. This provides valuable insights and design guidelines, but also shows the need for further exploration and reduction of the design space by deeper standards for the automotive industry.

CITATION: Axer, P., Thiele, D., Ernst, R., Diemer, J. et al., "Requirements on Real-Time-Capable Automotive Ethernet Architectures," *SAE Int. J. Passeng. Cars – Electron. Electr. Syst.* 7(2):2014, doi:10.4271/2014-01-0245.

INTRODUCTION

Ethernet is currently added to the existing set of automotive communication standards. Due to its superior bandwidth and flexibility and the promise of sharing cost of ownership with other industrial segments, Ethernet appears ideal to address the high demands of new functions in infotainment and advanced driver assistance systems (stereo camera, surround view, etc.) or to reduce ECU flashing and updating cost. While the first generation of cars will use Ethernet as additional communication medium, it is strongly considered as a powerful future backbone technology also carrying traffic originating in CAN(−FD) or other bus subsystems. An exemplary Ethernet backbone use-case is depicted in Figure 1.

This is no easy task since Ethernet has been developed for applications which are neither time nor safety critical, but focus on bandwidth. Cars have a far richer set of real-time requirements including quality-of-service (QoS), guaranteed end-to-end timing (i.e. latency), guaranteed delivery, guaranteed bandwidth or best-effort. Furthermore, Ethernet differs from CAN(−FD) and FlexRay in many ways: Ethernet is a packet switched network with point-to-point links and switches, which add delays to the end-to-end latency. Ethernet only has (at most) 8 priority levels compared to 2^{11} / 2^{29} on CAN. Frames can even be dropped in the switches without any ECU noticing it, requiring countermeasures on higher protocol layers. There are a lot of potential real-time effects that must be taken into account.

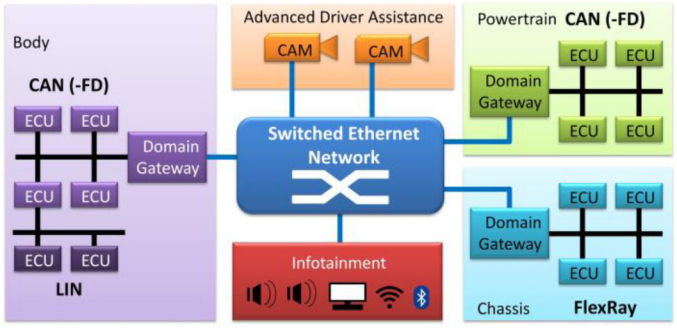

Figure 1. Typical Ethernet backbone architecture

Being able to analyze architecture concepts, load, performance and real-time capabilities of such Ethernet networks is one of the top-priority requirements. Such analyses are established standards for CAN and FlexRay networks already today, and we will need them more urgently for Ethernet-based E/E architectures because the real-time behavior will become more complex. At present, first solutions exist but there is only little experience with Ethernet-in-the-car, the existing standards do not cover all relevant aspects, and we can expect several "explosive issues" under the hood. Furthermore, the design space for Ethernet is (still) huge, which could lead to OEM-specific solutions that drive costs on all sides with no added value. To avoid this, a carefully planned joint approach would be highly beneficial for the automotive industry.

First solutions exist, showing that we can technically handle it. But what is price? On key aspect is that currently, there are very many variants of automotive Ethernet under discussion; this increases the total cost of ownership because someone has to provide the necessary equipment: switches, transceivers, lab equipment, tools, etc. When variants go up, re-use goes down, and prices increase. Hence, we need "deeper" industry-wide standards as soon as possible. Only then, we can reduce cost for components, processes, and tools. We expect an increasing awareness of this issue among experts and decision makers at the car manufacturers world-wide.

The paper is focuses on the issues on ISO/OSI layer 2 and is organized as follows: The next section provides some technical background and identifies key dimensions of the overall Ethernet design space: topology, protocol, and switching options. Then, we explain, how such Ethernet designs can be analyzed with respect to timing. We summarize key analysis techniques, and we present practically relevant metrics for the representation of the results. Next, we apply these analysis and metrics to an example Ethernet network and we compare the different strategies. Finally, we summarize lessons learned and draw our conclusions.

ETHERNET BACKGROUND

Topology Design Space

In this section, we will look into the topology design space, explain key mechanisms, possible pitfalls, and their influence on the overall performance.

Traditional busses like CAN or FlexRay are shared busses where many nodes are connected to the same physical wire. Switched Ethernet, on the other hand, can only connect two nodes directly in a peer-to-peer fashion. It must be noted, that Switched Ethernet is fundamentally different to traditional Ethernet which uses a shared medium (i.e. shared coaxial cable) and CSMA/CD for arbitration. Today, the concept of Ethernet over a shared medium is considered as obsolete.

Automotive Ethernet networks are established by connecting several ECUs to one another with one or more switches in between. This enables many different topologies, see Figure 2. A thorough analysis is required to choose the best fitting topology for a network. In this section four commonly used topologies are discussed.

The *star topology*, where all ECUs are connected to one big central switch, is well-known from office Ethernet networks. This topology only requires a single switch and reduces network contention, by minimizing shared links. On the other hand, a star topology can result in expensive wiring costs, depending on the location of the ECUs.

The *line topology* strives to reduce overall cable length. Here, each ECU comprises a 3-port switch to which it is connected as well. This increases the cost of the ECUs as each unit must contain a switch, and complicates the network configuration. Most importantly, contention increases as links are shared between different ECUs.

Clustered topologies represent a mixture between star and line, allowing a trade-off of the individual strengths and weaknesses. Typically, ECUs exchange most of the data locally in their system domains (i.e. body control). Thus, to reduce contention, one switch per domain is introduced. This reduces latency for intra-domain communication without excessive wiring cost.

In case the number of ECUs exceeds the number of ports per switch, a *tree* Ethernet topology can be chosen. This reduces the number of ports per switch. Special care must be taken with respect of the depth of the tree and location of ECUs in this topology as the latency strongly depends on the number of hops in the network. Note that in most automotive networks, ECUs will integrate the switch to decrease the number of physical nodes. In this sense Figure 2 shows the logical topology and not the physical setup.

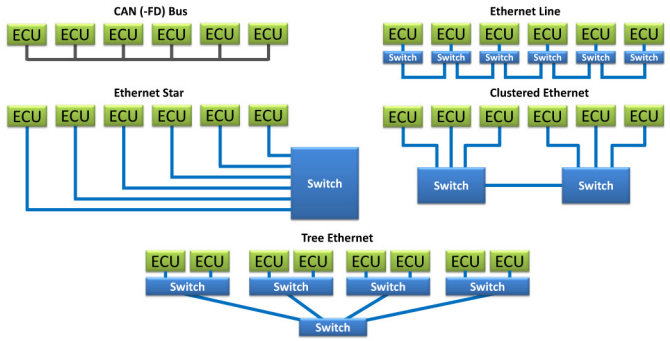

Figure 2. Typical topology candidates

Switching Strategy

The Ethernet switch obviously is the most important part of the entire network design. The switch receives frames on its ingress ports and forwards them the proper egress ports. This is conceptually shown in Figure 3.

There are two sources of delays: The static switching delay is caused by the store and forward architecture. An Ethernet frame is fully received, written to the output buffer(s) of the destination output port(s) and then transmitted. Static delays can be considered fairly easy and are mostly in the order of a few clock cycles of the switch (i.e. mostly internal memory access). The dynamic queuing delay occurs when frames arrive faster than they can be forwarded. This typically happens if frames arrive on multiple ingress ports which must be forwarded to the same egress port. The queuing delay is complex to predict and strongly depends on the interfering traffic pattern, used scheduling policies, shaper settings and available buffer sizes.

The switch pipeline consists of two units: an internal switch fabric and the egress stage. The switch fabric contains the address resolution logic, which resolved the output port from the frame's MAC address and forwards the frame to the correct egress port. The egress stage contains one or multiple FIFO queues, one for each service class, as well as the output scheduling logic. If frames are pending on multiple queues, the scheduler decides on the order of transmission.

The original Ethernet standard (IEEE 802.3), initially indented for best-effort transmission, does not specify any particular scheduling policy, but standard implementations typically implement one FIFO queue per port. Such an implementation makes it difficult or even impossible to make any timing predictions for different traffic classes. Because queues are shared, misbehaving or bursty streams easily over-occupy the buffer space and lead to dropped frames (tail-drop). This can significantly degrade the performance and reliability.

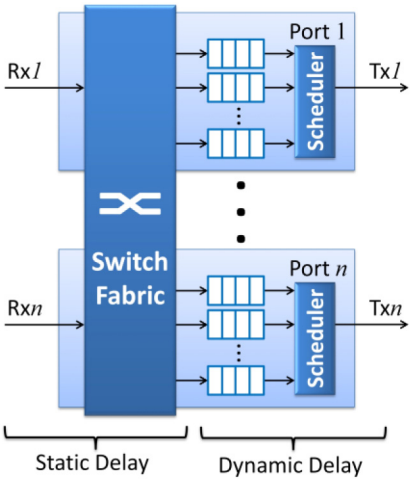

Figure 3. Internal switch architecture.

IEEE 802.1Q VLAN adds the concept of traffic classes and the switches can have up to 8 separate FIFO queues per port, each with a distinct *strict priority* (SP). In such switches, higher-priority traffic has always precedence over lower-priority traffic, just like on CAN buses. However, CAN has 2^{11} (base) or 2^{29} (extended) priority levels, while VLAN supports a maximum of 8 priority levels which requires merging the fine grain priority structure of CAN onto few separate communication streams. This entails a lot of potential real-time effects (e.g. latency) that must be taken into account. Also note, that most switches do not implement all 8 priority levels due to the required memory footprint, this leaves even less design freedom.

The recent *Ethernet AVB* Standard [1] extends the strict-priority scheduler by traffic shapers in the egress stage that facilitate bandwidth reservation. This way, higher-priority traffic classes can be prevented from blocking lower priority traffic for a long duration, while still meeting their bandwidth constraints.

Other switches provide a *weighted round-robin* (WRR) policy [2], where bandwidth is distributed according to weights that are associated with each traffic class. *Time-Triggered Ethernet* (TTEthernet/SAE AS6802) [3] uses a different approach based on a time-triggered schedule that is globally synchronized. It thereby strives to avoid collisions by design instead of resolving them on the fly. Here (just as with FlexRay), the guaranteed latency times come at the cost of reduced flexibility and efficiency. Finally, there is a new standard on the horizon: *AVB Generation 2* or also called *time-sensitive networks* (TSN). This standard borrows concepts from the existing ones to combine the “best of all worlds”, eventually resulting in a real improvement - if done right, and not too late, and not too costly.

Frame Forwarding Logic

Each incoming frame must be forwarded to one or multiple output ports. The Ethernet frame does not contain any intrinsic route information that can be used by switches for the forwarding process, as it only contains a source and a destination MAC (Media Access Control) address. Thus, either a switch dynamically learns the topology by listening to the traffic or it must be statically configured during design time. The possible number of MAC addresses (2^{48}) is too large to provide forwarding entries for all possible addresses. Thus a caching technique is used in which destination addresses compete for a limited number of physically available forwarding entries. Forwarding entries (cf. Figure 4) are organized in a Content Addressable Memory (CAM) hardware structure. The number of entries is in the order of a few thousand entries. A consumer grade unmanaged five-port switch (BCM5325) supports up to 2k entries according to the product brief.

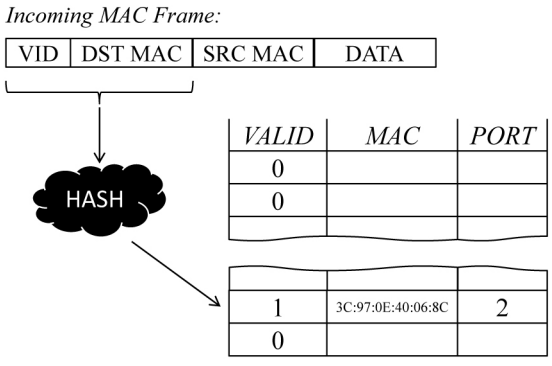

Figure 4. Forwarding logic: CAM table

The actual forwarding resolution works in the following way: The incoming frame is decoded and the fields are stored in a memory buffer. Depending on the implementation, the destination MAC and the virtual LAN identifier are hashed (i.e. by a CRC) and the hash value is used as an index to the entry table. This table stores control flags (i.e. for aging) and the destination port(s). Obviously, as the number of entries is limited, hash collisions will occur. Depending on the switch configuration, unresolved frames will be dropped or flooded on all ports.

If MAC and multicast addresses are not collision free, some entries do not fit into the CAM table. This can have significant impact on the overall network timing and functionality. Either frames do not arrive at all, or additional broadcast traffic is imposed, which is especially problematic if a high data-rate stream (i.e. HD video) is affected. In an open environment, MAC addresses cannot be assigned freely because the pool of available addresses is managed by the IEEE. Thus collisions cannot be out ruled. In a closed system, like a car, the situation is different, as a careful selection of MAC addresses used in one network can be done provided MAC addresses are the same for each individual car. However, this is not trivial, as the Ethernet standard does not recommend any specific hashing policy and a set of addresses which are collision-free on switch A can be lead to collisions on switch B which implements a different hashing algorithm. This issue needs to be addressed by the OEM integrating the ECUs. For the remainder of the paper, we will assume a collision free mapping.

TIMING ANALYSIS METHODOLOGY FOR ETHERNET

In this chapter, we briefly outline a timing analysis methodology for Ethernet networks. It is based on the Compositional Performance Analysis approach, which we introduce in the first section. To capture the timing of Ethernet, several extensions are required which we present in the following section.

Compositional Performance Analysis

The Compositional Performance Analysis (CPA) approach [4], which is used to derive formal performance metrics for Ethernet, reduces the timing analysis complexity by decomposing a system into independent resources. These resources are analyzed locally in isolation and the results are propagated to dependent tasks by a global analysis loop (cf. Figure 5). This approach makes CPA very fast and scalable compared to other formal approaches like model checking. The CPA approach is implemented in the commercially available SymTA/S tool, but the basic algorithms are also available by the open source tool pyCPA [5]. A similar compositional approach is also used by Real-Time Calculus (RTC) [6]. The local analyses between CPA and RTC, however, differ.

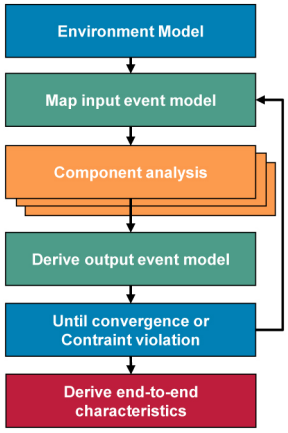

Figure 5. System analysis flow. Event-model propagation and component analysis are repeated, until convergence.

The CPA system model comprises three main components: resources, tasks, and event models. Resources abstractly model the actual communication and processing resources in a system. By means of a scheduling policy, they provide service (e.g. processing time, network bandwidth) to the tasks, which are mapped to them. These tasks τ_i are characterized by best-case and worst-case execution times C_i^- and C_i^+, which represent lower and upper bounds on how much service a single task activation consumes on its associated resource. Tasks are activated by events originating from other tasks or external sources. Event models are used to provide bounds on the arrival of events. In such an event model, the arrival curves $\eta^-(\Delta t)$ and $\eta^+(\Delta t)$ describe the lower and upper bound of the number of events that can arrive in any time interval of length Δt. An example for an event model is given in Figure 6. Their pseudo-inverse counterparts, the minimum distance functions $\delta^+(n)$ and $\delta^-(n)$, give the maximum and minimum time interval between the first and the last event in any sequence of n events. Thus, compared to a single event trace, an event model is a more general model, which captures all possible event arrivals between its bounds.

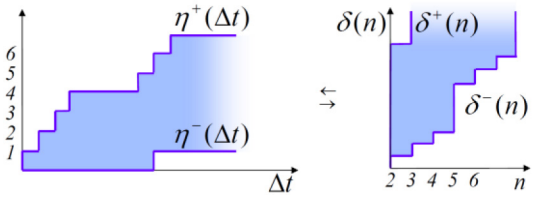

Figure 6. Illustrative example for an event model. The η and δ functions are pseudo inverse.

CPA derives the worst-case timing of a system by first performing the busy-window analysis [7] on each resource to derive the local worst-case timing and event models of each task. The busy-window analysis starts with a critical instant scenario, where the worst-case arrival times of all interfering tasks are assumed to maximally delay the currently analyzed task. From the delay a task experienced, (new) output event models can be derived. During the global analysis loop, these output event models are then propagated to dependent tasks,

i.e. tasks that are activated by the completion of other tasks, to serve as updated input event models [8]. The global analysis iterates through these steps until a fixed-point is found, i.e. all event models have converged, or until predefined constraints (e.g. deadline, latency, jitter) are violated. The system is deemed unschedulable in the latter case. In addition to schedulability analysis, CPA also provides upper bounds on the worst-case response time of tasks and the end-to-end latency along a chain of tasks. Also, the required worst-case buffer sizes at each resource can be computed. This is especially important in Ethernet networks as switches only have limited internal buffer capacity and will drop frames if these buffers are exhausted.

Worst-Case Analysis for Ethernet

Classic Ethernet uses a single FIFO queue for outgoing traffic. Thus, a FIFO scheduling analysis is required to derive worst-case latencies. This is similar to the approach taken in earliest deadline first (EDF) scheduling analysis [9]. A worst-case delay analysis for Ethernet without priorities is presented in [10].

Strict-priority Ethernet matches a classical scheduling strategy, static-priority non-preemptive (SPNP). SPNP scheduling, in general, is well understood [11] and can be adapted to analyze Ethernet [12]. In Ethernet, however, multiple traffic streams may (and usually have to) share the same priority, which is typically resolved by FIFO scheduling. This requires an extension to SPNP analysis [13] by a FIFO analysis. In [24], Real-Time Calculus (RTC) [6] is used to analyze an Ethernet network, but this analysis suffers from large overestimations. A CPA-based Ethernet analysis is presented in

Ethernet AVB is a promising candidate for arbitration in future automotive Ethernet networks. Analyses are presented in [14], [15], and [13]. The first one only derives timing properties per traffic class, whereas the latter two derive these properties per-stream. Ethernet AVB was designed to provide bandwidth guarantees, but automotive and control applications typically require latency guarantees. [13] shows that, as a consequence, the end-to-end latencies of high priority traffic classes can suffer heavily from AVB's traffic shaper implementation and that care must be taken when configuring the parameters of the shapers.

In [16], RTC is used for the timing analysis of both strict-priority and weighted round robin (WRR) scheduled Ethernet. The proposed WRR analysis, however, suffers from a large overestimation. [17] provides an improved CPA-based WRR analysis, which provides considerably tighter timing guarantees. In [17], WRR is also compared against SPNP scheduled Ethernet.

An analysis for the worst-case end-to-end latencies for AFDX, which also uses SPNP, is described in [18][19][20], of which the first two also present methods to compute the buffer sizes throughout the network.

Time Triggered Ethernet introduces three message classes to address the requirements of timing deterministic Ethernet network behavior. The timing analysis of time-triggered messages is, by principle, trivial, once a schedule is constructed. [21] provides a method to optimize TTEthernet schedules for a minimum end-to-end delay of rate-constrained messages.

Metrics

When exploring different design options such as topology or switching strategy, there needs to be a way to compare them efficiently. There are two basic dimensions for comparison: load, i.e. the utilization of the network, and latency, i.e. the delay imposed by the network. These metrics can be obtained by simulation or formal analysis. In bus-based systems, these metrics are well known and rather straight forward. For Ethernet, however, metrics requires additional considerations [22].

Load

For a specific CAN bus configuration, for example, we can compute a single load number because there is only a single “resource”, the bus itself. For multi-hop switched networks, there are multiple resources. Specifically, for Ethernet, each output port (of an ECU or a switch) is a resource in itself with an associated load. Simply computing the loads of all Ethernet ports oftentimes does not yield significant insight, as ports are used differently depending on the actual traffic. Just knowing how much each individual port is utilized is meaningless without the knowledge of network topology and routing.

First of all, we can characterize the injected load by sender and receiver. This yields a first overview of the available communication, i.e. how much each individual ECUs sends and receives. This can be extended into a matrix to break down the individual communication partners.

From the individual sender/recipient pairs, we can derive the load of each port that is traversed by packets from the sender to the recipient. We call the maximum of these individual port loads the *squeeze load* [22]. It gives us an idea of the utilization and hence the free capacity between the communication partners. If required, the maximum of these loads over all communication pairs can be used as a single number characterizing the load of the entire Ethernet network.

Latency

Load only characterizes the utilization of the network, it does not contain any information about the timing of the individual messages. For bus-based systems, computation of the maximum latency (either observed in simulation or computed by formal worst-case analysis) is established to verify that timing requirements can be met.

In order to assess the timing in relation to the deadline, the maximum latencies of the individual messages are reported relative to their deadlines. If no deadlines are available, the message periods (for periodic messages) can be used instead.

This can be done on Ethernet networks just as it has been done on CAN. If required, the maximum of all relative deadlines can be computed over all messages, yielding a single number characterizing the real-time quality of the network. If this number is below 1, all timing requirements are met. If it is a lot less than 1, there is good headroom for all messages, but potentially also an over design of the network. If this number is larger than 1, at least one message does not meet its deadline and the timing of the system needs to be inspected in more detail.

USE-CASE AND EXPLORATION

We will now take a look at different design options for a concrete use case scenario. This use case is a system of 16 ECUs that communicate with each other. Out of these, 4 ECUs send video data (e.g. from a camera) to one other ECU for processing.

Traffic Description

We keep the injected traffic constant during the exploration. We consider three types of traffic:

1. Control traffic between individual ECUs: This traffic is the most latency critical, but typically has relatively low bandwidth. This traffic class includes both single- and multi-cast messages. The total injected control traffic amounts to ca. 15Mbit/s.
2. Video traffic: Four ECUs (ECU number 4, 8, 15, and 16) contain a camera which sends to a central video processing ECU (ECU10). Each camera ECU injects around 23.5Mbit/s of traffic as unicast to ECU10, so the total video traffic amounts to ca. 94Mbit/s. For this reason, we connect ECU10 via a 1GBit/s port, while all other ports are 100MBit/s.
3. Broad-cast status traffic: Most ECUs send low-bandwidth broadcast messages with status information. This traffic class is least important. The total injected broadcast traffic amounts to 1.5Mbit/s.

Table 1 shows the injected and received bandwidths for each ECU.

The communication is further illustrated in Figure 7, which shows a graph of the system-wide traffic. ECUs are represented by nodes, and communication is represented by edges whose width is determined by the bandwidth. The video traffic is clearly visible. On top of that, there is unicast communication between other ECUs. Furthermore, all ECUs send low-bandwidth broadcast messages. All messages are transferred strictly periodically. We set the message deadlines to three times their period.

Table 1. Injected Bandwidth

ECU	TX (Kbit/s)	RX (Kbit/s)
ECU01	1571.84	6461.92
ECU02	113.12	3054.24
ECU03	1571.84	1560.16
ECU04	23640.6	1459.2
ECU05	241.6	6856
ECU06	120.64	2307.2
ECU07	2715.2	4808
ECU08	23761.6	1528
ECU09	2715.2	5417.92
ECU10	4656	96318.4
ECU11	1507.2	1633.6
ECU12	521.92	2550.4
ECU13	486.4	2507.2
ECU14	15.84	1699.36
ECU15	23640.6	1608
ECU16	23640.6	1614.72

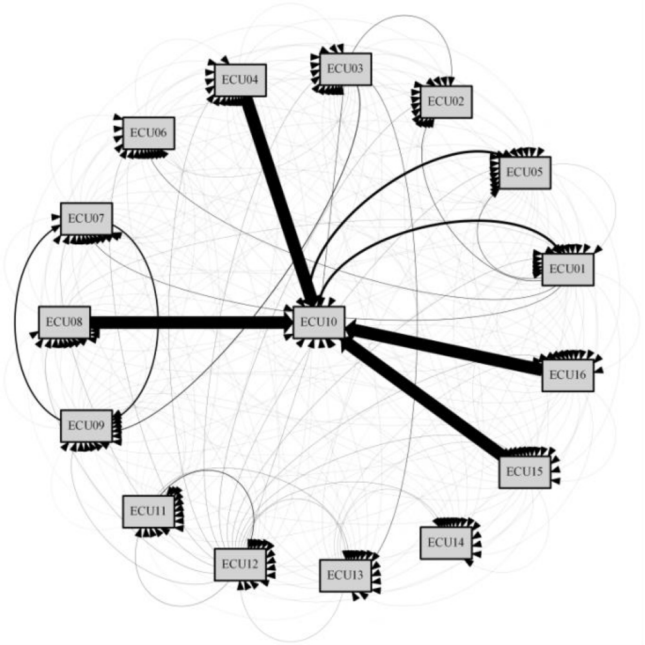

Figure 7. Overview of the traffic

Comparison of Topologies

We first compare the different topologies to each other. Specifically, these are

1. Star: All 16 ECUs are connected to a central 16-port switch.
2. Line: Each ECU is connected to a 3-port switch, which connects to 1 or two neighboring switches. The ECUs are lined up according to their number, i.e. ECU1 is connected to ECU2 and so forth.

3. Clustered Line: The ECUs are connected in groups of four to one of four switches, which themselves are connected in a line. This requires two 5-port and two 6-port switches. Again, the ECUs are grouped and lined up according to their number.
4. Tree: The ECUs are connected to a two-level tree. This topology is similar to the clustered line, but the switches of the clustered line are connected to a 4-port root switch instead of to each other.

Regardless of the switching/scheduling strategy used, we can determine the maximum port load between each pair of ECUs, which is an indicator for the utilization of the network. For this, we model the traffic scenario and the different topologies in SymTA/S [23] and analyze the squeeze loads. Figure 8 shows the squeeze load each ECU observes when transmitting to any of its recipients for the different topologies. For example, ECU1 observes a maximum port load of about 10% for the star topology, which in this case results from the load on the port to ECU10. As expected, the star topology shows the lowest utilization, as each ECU can be reached via a dedicated port, so there is no contention for messages with different recipients. The line topology exhibits a lot of sharing at the ports connecting the individual switches. As a result, almost all ECUs see a load of about 50% to at least one of their recipients - caused by interference with two of the four video streams. The clustered line topology fares better, as some ECUs can reach all of their recipients without contending with a video stream. The tree topology lies in between. Especially for the low ECU numbers, it shows significantly less load (interference with only one video stream). For high ECU numbers, it performs the same as line and clustered line.

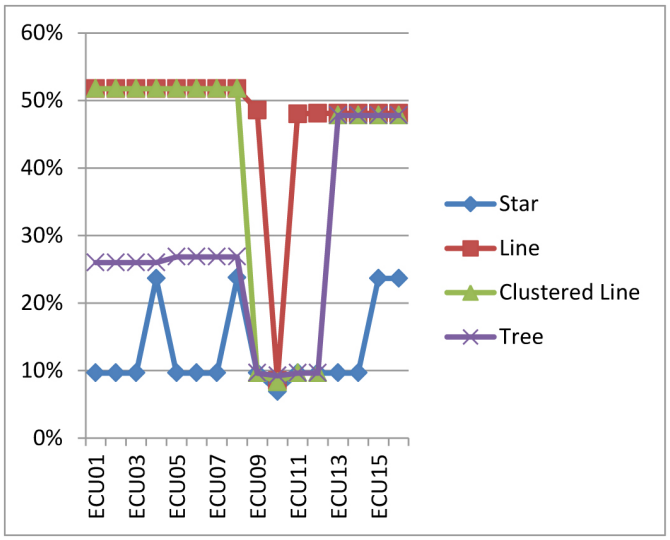

Figure 8. Squeeze Loads by Transmitting ECUs

It should be noted that the squeeze load depends on the assignment of the individual ECUs to ports for the line, clustered line and star topology. Thus, it could potentially be optimized by remapping the ECUs. However, this may not always be feasible due to other topological constraints (e.g. position of ECU in the car).

Now let us consider the worst-case latencies of the individual messages for the different topologies. For this, we analyze the different topologies using the compositional performance analysis for Ethernet [13] implemented in pyCPA [5]. Figure 9 shows the latencies for strict-priority Ethernet relative to the message deadline. The messages are sorted by their relative deadline in the star topology. As expected, the relative deadlines for star are best. The clustered line and tree are very similar, with the tree being slightly better than the clustered line in many cases. The line topology yields the worst latencies, sometimes far beyond the messages' deadlines.

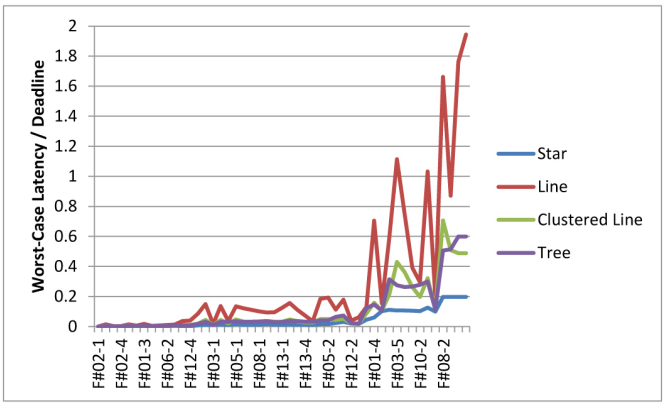

Figure 9. Latency profile of the different messages for varying topologies

The maximum absolute latencies observed for the use case were 0.46ms for the star topology, 5.96ms for the line, 1.49ms for the clustered line and 1.21ms for the tree.

Comparison of Switching Strategies

Now, we will explore the different switching strategies, specifically FIFO of classic Ethernet, strict priority (SP) of Ethernet VLAN and the credit-based shaping algorithm of Ethernet AVB. For these, we configure the parameters as follows:

1. FIFO: For FIFO scheduling, no scheduling parameters are required, as all traffic is mapped to a single FIFO queue.
2. Strict-Priority (SP): For SP scheduling, we assign priorities to messages based on their traffic class. We assign the lowest priority (priority 1) to broadcast status messages, a medium priority (priority 2) to video traffic and the highest priority (priority 3) to uni-/multicast control traffic.
3. AVB: For AVB scheduling, we keep the priority assignment of SP, but add traffic shaping to the video traffic (on priority level 2). We explicitly do not shape the high-priority control traffic due to adversarial effects on latency as shown in [13]. We configure the idleSlopes, i.e. the amount of traffic permitted by the traffic shaper, to 100Mbit/s for the port connecting ECU10, which receives all video traffic. All other

ports have an idleSlope of 50Mbit/s for the video traffic, which is enough to accommodate two parallel video streams (the maximum on any port except the one to ECU10).

Figure 10 shows the worst-case latencies of the messages (sorted by their message ID) for the different switching strategies in the star topology. The latencies are plotted relative to the latencies of the FIFO switching. It can be seen that SP often significantly reduces the latencies compared to FIFO, as prioritization reduces interference for the higher priority messages. Consequently, the lower-priority broadcast messages see higher worst-case latencies. AVB shows a significant increase of the latencies for the video traffic due to shaping delay. This shaping does not improve the latencies of the low-priority broadcast traffic, because the video frames were already injected periodically (i.e. “shaped” even for FIFO and SP switching) and hence shaping at the injection ports does not reduce the burstiness.

Figure 11 shows the same results for the line topology. Prioritization improves the latencies for many messages with little adversarial effects. For AVB, the traffic shaping increases the latencies for video traffic even more (2.5 - 4.7x of the FIFO latency). Recall that the line topology already resulted in relatively high latencies (see Figure 9). Compared to SP, this high additional worst-case delay results in marginal improvements for a few of the broad-cast messages. AVB latencies for non-video traffic are often better and never much worse than FIFO.

For the clustered line topology, the results are similar but less dramatic. For SP, latencies are usually better and never worse than those of FIFO scheduling. Video traffic sees a similar impact from the traffic shaping of AVB, but there is no positive effect on the broadcast traffic.

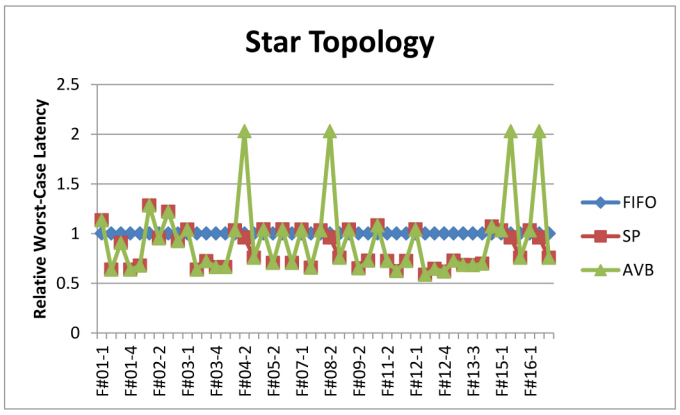

Figure 10. Latencies for the star topology with different switching strategies, relative to the FIFO switching

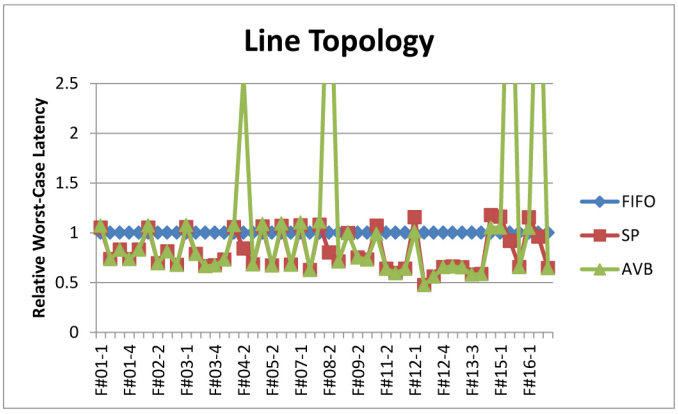

Figure 11. Latencies for the line topology with different switching strategies, relative to the FIFO switching

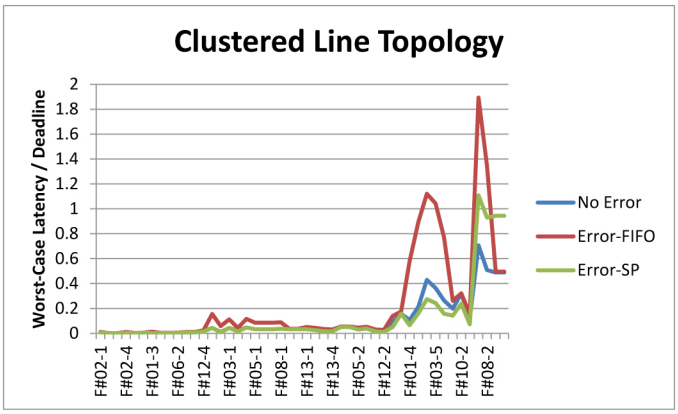

Figure 12. Latency profiles for mixed-criticality scenario using the clustered line topology

Priorities in SP and AVB scheduling can also be used to isolate different traffic classes from each other. This way, interference from lower-criticality traffic to critical traffic can be prevented. This is a requirement to for mixed-critical systems. Figure 12 shows the latencies of all messages (sorted by deadline) relative to their respective deadline for three scenarios. The error-free case (blue line) uses FIFO scheduling and matches the scenario seen before. Now, we assume an error into a low-criticality application (i.e. one that has no fault-tolerance and lower test coverage), causing it to erroneously inject a lot of low-critical traffic. This causes the latencies of many messages to increase drastically if FIFO scheduling is used (red line). With SP scheduling, the faulty message can effectively be isolated, so that only the messages on the low priority level are affected (green line).

LESSONS LEARNED

The timing of Ethernet systems depends on many factors, such as topology, switching strategy, buffer organization. In order to make educated decisions on the usability of Ethernet solutions for different in-car use cases, and to find the optimal configuration of an Ethernet network, timing analysis methodologies and tools are required.

In the previous chapter, we have explored different design options for Ethernet networks for a typical in-car communication architecture. Comparing different topologies, the star is optimal from a timing perspective as it minimizes contention. This results in the lowest squeeze load which translates to the largest margin for extensibility, but also results in the lowest latencies. However, a single central switch may not always be feasible or optimal when considering other factors like cost or weight. A line topology may be more suited from these aspects, but shows the worst squeeze load and latencies. Tree and clustered line topologies lie in between and offer trade-offs in the individual aspects.

When looking at the absolute worst-case latencies, Ethernet does indeed look capable of meeting the latency requirements for many time-critical applications, as latencies were usually well below 2ms (except for the line topology).

Regarding the switching technology, we have compared FIFO, strict priority and AVB. FIFO offers no way of differentiating between traffic and thus is sub-optimal for non-uniform traffic. With priorities, we can improve the latencies of time-critical traffic at the cost of less critical ones. On top of that, priorities can also be used to isolate safety-critical from non-critical traffic. At first sight, Ethernet AVB looks promising as its traffic shaping promises to reduce the interference caused by high-bandwidth high-priority traffic. Upon closer examination, however, the impact of traffic shaping on the shaped traffic seems much more than the actual benefit of lower-priority traffic.

CONCLUSIONS

Leading OEMs such as Volkswagen, Daimler, BMW and others have made clear that Ethernet is on its way into automotive electronics. While first applications will, for the most part, use Ethernet as an add-on to existing network structures, Ethernet is also a hot candidate as future backbone technology. There are many good arguments, such as cost sharing with other industries, high bandwidth and new protocol functions for Internet access. Ethernet has a proven record of picking up new technological developments towards permanently increasing bandwidth, and Internet and wireless communication compatibility with the promise of a long lasting backbone standard that the automotive industry has been looking for. However, the way is rough as Ethernet is conceptually different from legacy automotive networks in various aspects.

In this paper, we have presented different flavors of Ethernet with their unique strengths and weaknesses, and their different design options. We have presented a methodology to assess the timing of Ethernet networks which includes a formal worst-case analysis and a set of metrics. The former allows the computation of the worst-case timing of different Ethernet design options. The latter compress the data gained from timing analysis into comprehensible form and produce comparable numbers that are essential to efficiently explore and compare the design space. We have presented a case study of a typical automotive use case and explored different topology options and switching strategies. This shows that the necessary research into real-time and performance analysis generates valuable results, which we will continue on the open questions. TU Braunschweig via its innovation center, iTUBS, is collaborating with Daimler, Volkswagen and others to develop the foundations. Symtavision is working towards commercial solutions and tools for Ethernet timing.

So technically it is complex but possible. But how about the commercial aspects? When the number of Ethernet variants is growing, the number of customers per variants goes down, significantly increasing prices for components, equipment and tools. Therefore, we need “deeper” standards as soon as possible. Otherwise, we will find us lost in a zoo of manufacturer-specific solutions; expensive and inflexible. With few deep standards, we can reduce cost for components, processes, and tools. Here, a strategy must be planned well ahead, as decisions have far-reaching cost and productivity implications for the entire automotive supply chain, from the OEMs through the tier-1 suppliers down to the device and communication stack suppliers, as well as associated tool providers.

We would like to conclude with one of the key objectives of AUTOSAR, which applies quite well to the current situation of automotive Ethernet: *We suggest a strong global partnership that creates this common standard: “Cooperate on standards, compete on implementation”*.

REFERENCES

1. Forwarding and Queuing Enhancements for Time-Sensitive Streams, IEEE Standard 802.1Qav-2009, 2009
2. Katevenis, M, Sidiropoulos, S., Courcoubetis, C.: Weighted Round-Robin Cell Multiplexing in a General-Purpose ATM Switch Chip. *IEEE Journal on Selected Areas in Communications*, vol. 9, no. 8, 1991
3. Kopetz, H., Ademaj, A., Grillinger, P., and Steinhammer, K., “The time-triggered Ethernet (TTE) design,” In IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC), pages 22-33, 2005.
4. Henia, R, Hamann, A., Jersak, M., Racu, R., et al., “System Level Performance Analysis - the SymTA/S Approach,” In IEE Proceedings Computers and Digital Techniques, 2005.
5. Diemer, J., Axer, P., and Ernst, R., “Compositional Performance Analysis in Python with pyCPA,” In 3rd International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS), July 2012.
6. Thiele, L., Chakraborty, S., and Naedele, M., “Real-time calculus for scheduling hard real-time systems,” In The 27th Annual International Symposium on Computer Architecture (ISCA), Volume 4, 2000.
7. Tindell, K.W., Bruns, A., and Wellings, A.J. “An extensible approach for analysing fixed priority hard real-time tasks,” RTSS, 1994.
8. Schliecker, S., Rox, J., Ivers, M., and Ernst, R., “Providing Accurate Event Models for the Analysis of Heterogeneous Multiprocessor Systems,” In Proc. 6th International Conference on Hardware Software Codesign and System Synthesis (CODES-ISSS), Atlanta, GA, October 2008.
9. Palencia, J. and Harbour, M., “Offset-based response time analysis of distributed systems scheduled under EDF,” In Proceedings of the Euromicro Conference on Real-Time Systems, pages 3-12, July 2003.

10. Lee, K.C., Lee, S., and Lee, M.H., "Worst Case Communication Delay of Real-Time Industrial Switched Ethernet With Multiple Levels," IEEE Transactions on Industrial Electronics, October 2006.
11. Davis, R. I. and Burns, A., "Controller Area Network (CAN) Schedulability Analysis: Refuted, Revisited and Revised," Real-Time Systems, 35, 2007.
12. Rox, J. and Ernst, R., "Formal Timing Analysis of Full Duplex Switched Based Ethernet Network Architectures," SAE Technical Paper 2010-01-0455, 2010, doi:10.4271/2010-01-0455.
13. Diemer, J., Thiele, D., and Ernst, R., "Formal Worst-Case Timing Analysis of Ethernet Topologies with Strict-Priority and AVB Switching," In IEEE International Symposium on Industrial Embedded Systems, 2012.
14. Imtiaz, J., Jasperneite, J., and Han, L., "A performance study of Ethernet Audio Video Bridging (AVB) for Industrial real-time communication," In IEEE Conference on Emerging Technologies Factory Automation, 2009.
15. Diemer, J., Rox, J., and Ernst, R., "Modeling of Ethernet AVB Networks for Worst-Case Timing Analysis," In MATHMOD - Vienna International Conference on Mathematical Modelling, Vienna, Austria, 2012.
16. Georges, J.-P., Divoux, T., and Rondeau, E., "Strict Priority versus Weighted Fair Queueing in Switched Ethernet Networks for Time Critical Applications," In Proceedings of the IEEE International Parallel and Distributed Processing Symposium, April 2005.
17. Thiele, D., Diemer, J., Axer, P., Ernst, R., et al., "Improved Formal Worst-Case Timing Analysis of Weighted Round Robin Scheduling for Ethernet," In Proc. of CODES+ISSS, September, 2013.
18. Bauer, H., Scharbarg, J.-L., and Fraboul, C., "Worst-case end-to-end delay analysis of an avionics AFDX network," In Design, Automation Test in Europe Conference Exhibition (DATE), 2010.
19. Bauer, H., Scharbarg, J.-L., and Fraboul, C., "Worst-Case Backlog Evaluation of Avionics Switched Ethernet Networks with the Trajectory Approach," In Euromicro Conference on Real-Time Systems (ECRTS), July 2012.
20. Gutiérrez, J.J., Palencia, J.C.P, and Harbour, M.G.H., "Response Time Analysis in AFDX Networks," In XIV Jornadas de Tiempo Real, February 2011.
21. Tamas-Selicean, D., Pop, P., and Steiner, W., "Synthesis of communication schedules for TTEthernet-based mixed-criticality systems," In Proceedings of the international conference on Hardware/software codesign and system synthesis, CODES+ISSS '12. ACM, 2012.
22. Schliecker, S., Diemer, J., "Quantifying the Timing Quality of Ethernet-based Network Configurations - What is the 'Bus Load' of my Ethernet network?," Presentation at the 3rd Ethernet & IP @ Automotive Technology Day, Germany, September 2013. Available Online: http://www.ethemettechnologyday.com/downloads/archive/3rd/14 Schlicker_Symtavision.pdf
23. SymTA/S by Symtavision GmbH, www.symtavision.com
24. Revsbech, K., Schiøler, H., Madsen T.K., and Nielsen, J.J., "Worst-case traversal time modelling of Ethernet based in-car networks using real time calculus," In Proceedings of the international conferences on Smart spaces and next generation wired/wireless networking, NEW2AN'11/ruSMART'11, Springer-Verlag, 2011