Study of the Time Triggered Ethernet Dataflow

by

Niclas Rosenvik

LIU-IDA/LITH-EX-G–15/011—SE

2015-07-08
Final thesis

Study of the Time Triggered Ethernet Dataflow

by

Niclas Rosenvik

LIU-IDA/LITH-EX-G–15/011—SE

2015-07-08

Supervisor: Unmesh Bordoloi
Examiner: Petru Eles
Abstract

In recent years Ethernet has caught the attention of the real-time community. The main reason for this is that it has a high data throughput, 10Mbit/s and higher, and good EMI characteristics. As a protocol that might be used in real-time environments such as control systems for cars etc, it seems to fulfil the requirements. TTEthernet is a TDMA extension to normal Ethernet, designed to meet the hard deadlines required by real-time networks. This thesis describes how TTEthernet handles frames and the mathematical formulas to calculate shuffle delay of frames in such a network. Open problems related to TTEthernet are also discussed.
# Contents

1 Introduction
   1

2 Ethernet
   2.1 Switching .............................................. 2
   2.2 Ethernet frame format .................................. 3
   2.3 The need for TTEthernet .............................. 4

3 TTEthernet
   3.1 TTEthernet spec ...................................... 6
   3.1.1 Protocol control frames ............................. 6
   3.1.2 Time-triggered frames .............................. 7
   3.1.3 Rate constrained frames ......................... 9
   3.1.4 Best effort frames ................................ 10
   3.2 Integration Algorithms .............................. 10
   3.2.1 Shuffling ............................................ 10
   3.2.2 Preemption ......................................... 10
   3.2.3 Timelyblock ........................................ 10
   3.2.4 TTEthernet requirements ....................... 11
   3.3 Shuffle delay .......................................... 11
   3.3.1 Shuffle delay of RC frames .................... 12

4 Comparison with other systems ........................ 14
   4.1 FlexRay .................................................. 14
   4.1.1 FlexRay protocol .................................... 14
   4.1.2 Comparison with FlexRay ......................... 14
   4.2 CAN ...................................................... 15
   4.2.1 CAN protocol ....................................... 15
   4.2.2 Comparison with CAN ................................ 15
   4.3 AVB ...................................................... 15
   4.3.1 AVB Ethernet protocol ........................... 15
   4.3.2 Comparison with AVB Ethernet .................. 15
Chapter 1

Introduction

In recent years Ethernet has caught the attention of the real-time community. The main reason for this is that it has a high data throughput, 10Mbit/s and higher, and good EMI characteristics. Real-time networks have hard deadlines that are to be met. There exist network protocols that achieve this but they are very specialised and based on non commodity hardware. That makes them expensive to develop since they have to be developed from ground up. The low speed also causes problems for users of these protocols, both during run-time and at production/maintenance. Reprogramming of Electronic Control Units (ECU) can take a long time. For example, uploading 81 MB to various ECU:s in some car models takes up to 10 hours[1] since the CAN bus used only achieves 1Mbit per second maximum vs Ethernet’s 100Mbits per second. One possibility is to use different buses but this is very costly because either translation gateways are needed or the ECU:s must be equipped with interfaces for each bus type. Both would increase the equipment needed and, thus, increase cost and complexity. Due to this there is an interest in using Ethernet that can achieve real-time characteristics as a common bus between ECU:s. There are different Ethernet technologies that have been developed and that might solve this problem. One of them is Time Triggered Ethernet (TTEthernet). TTEthernet is a TDMA extension to normal Ethernet.

The goal of this thesis is to understand and explain how information is passed in a TTEthernet network. A second goal is to find out how TTEthernet compares to other protocols used for real-time control applications. These goals are achieved by reading the specification and studying the protocol theoretically. Comparisons are reported by studying research papers that compare TTEthernet with different real-time network protocols and look at their conclusions.
Chapter 2

Ethernet

Ethernet is today the most common local area network standard and has had a large commercial success. Ethernet, as we know it today, has very little in common with the original standard. When Ethernet was created by Bob Metcalfe it was a shared bus network that used carrier-sensing to find out if any other node was sending. Later, star based topologies with twisted pair wiring were introduced. In this scenario collision detection is still used. The hub that is used to connect the wires simply puts the signal coming from one port on all other ports.

2.1 Switching

Today Ethernet LAN:s use switches to connect instead of hubs. Switches don’t usually repeat the information coming from one port to all other ports. Instead they use a forwarding table and if the address that the frame is destined for is in the forwarding table it sends the frame to only one port corresponding to the destination, otherwise it forwards it to all ports. The table is normally built on the fly, frames coming from different ports are read and the source address of the frame is put into the table.

Modern Ethernet uses full duplex operation. In this mode one channel is used to send frames and another channel is used to receive frames. There is just one side that is sending on each channel so the transmission is collision free. So collision detection is not needed. However full duplex switched Ethernet is still vulnerable to packet loss due to the fact that many nodes might want to send data to the same destination.

A switch can forward frames in different ways. These ways are called ”Store and forward” and ”cut-through”. ”Store and forward” means that the switch receives the whole frame before it forwards it to a port. ”Cut-through” means that the information in the frame is forwarded as soon as the switch can determine the destination of the frame.
2.2 Ethernet frame format

Ethernet uses different but very similar frame formats depending on the speed. I will here describe the frame format for 100 Mbit operation[2].

At the beginning of the frame is the so called Preamble as seen in Table 1. It consists of Start of Stream Delimiter (SSD), Preamble and Start of Frame Delimiter (SFD). This is sent before everything else so that the receiver can lock on to the stream before receiving it. The first and last octets of the preamble are called Start of Stream Delimiter and Start of Frame Delimiter.

Then comes the Destination address and the Source address. The 2 octet Length/Type field represents different things depending on its value. If the value is between 46 and 1500 it represents the length of the data field that comes after it. If it is 1538 and above it represents what type of data is in the data field. After the Data comes a 32-bit Frame Check Sequence. And last, an one byte End of Stream Delimiter (ESD) that marks the end of a frame. In table 1 also the Inter Frame Gap (IFG) is mentioned. This is the shortest idle time between frames. The IFG is not usually included when describing an Ethernet frame. We include it here because this gap will fill up the transmission line when two frames are transmitted directly after each other. So, it must be included when calculating the delay that frames cause on other frames when in transmission. We utilize this in our calculations in chapter 3.2.1 where the maximum frame size is used.

Fast Ethernet uses so called 4b/5b encoding[2]. This means that it uses 5 bits to encode 4 bits. So 16 of the 32 possible bit patterns are used to represent the data values 0 to F. The codes are listed in Table 2. Some of the rest of the patterns are used as control characters. These control
characters are used in the SSD, ESD and the IFG. Since the Length/Type field represents different things depending on its value it might seem strange that the devices know when a frame is finished even if the length is not included in the header. The ESD takes care of this.

2.3 The need for TTEthernet

Ethernet has properties that make it unsuitable for real-time applications. When frames arrive from different ports but are forwarded to the same port they have to be queued. In the case that frames arrive too fast from different ports, the buffer that holds the queue might become full and the frames that arrive afterwards have to be discarded because there is no space for them in the output buffer. This means that frames are not guaranteed to
arrive at their destination. The queueing and frame loss leads to that there are no guaranteed worst case delays in an Ethernet network. Candidates have emerged for real-time networks based on Ethernet that try to address these issues. One of those is TTEthernet. The next chapter will describe TTEthernet.
Chapter 3

TTEthernet

3.1 TTEthernet spec

TTEthernet is defined as an extension to normal Ethernet (IEEE 802.3) [3, Chapter 3]. This allows normal Ethernet operations within a TTEthernet network if the system designer wants this. TTEthernet defines three traffic types: protocol control frames, time-triggered traffic and rate-constrained traffic. Protocol control frames and time-triggered traffic are mandatory and must be implemented, rate-constrained traffic is optional [3, section 3.5.1]. Normal Ethernet traffic is also optional [3, section 3.5.1].

TTEthernet uses priorities. Traffic types of higher priorities are always sent before traffic of lower priorities. The priority order is: Protocol control frames, Time-Triggered frames, Rate-Constrained frames, Best-Effort frames. This can be illustrated as four outgoing queues.

<table>
<thead>
<tr>
<th></th>
<th>PCF</th>
<th>TT</th>
<th>RC</th>
<th>BE</th>
</tr>
</thead>
</table>

3.1.1 Protocol control frames

PCF (protocol control frames) frames take care of the time synchronisation in the system. These packets have an Ethernet type field of 0x891d and have a DATA segment of 46 bytes. They are sent periodically, just like TT frames, when the system has been set up and there is no error occurring. The period time is called an integration cycle. This cycle must be considered together with the periods and offsets of the TT frames when making a schedule.

PCF frames are sent from so called synchronisation masters (SM) to the so called compression masters (CM) and the compression masters respond
to these by sending back PCF frames to the SM:s and the synchronisation clients. The responses are sent in a multicast/broadcast fashion. So only one message is sent over a link even if there are more than one receiver of it. [3, Figure 13 and 14] The CM:s can respond in two ways. Either by sending one PCF back to each SC and SM in a so called compressed scenario or send every frame it received to each SC and SM in a so called uncompressed scenario. The frames sent periodically are responded to in a compressed manner by the CM:s [3, Table 1].

The impact from PCF:s on the other traffic types is bounded and can be calculated due to the above mentioned characteristics.

In a TTethernet network where there is only one switch and one compression master the time that the synchronisation step occupies in the beginning of each integration cycle is bounded to:

\[
\frac{(3 + SM) \times (38 + 46) \times 8}{link\text{-}speed} + CM\_response\_time
\]

Where \( SM \) is the number of synchronisation masters in the network, \( link\_speed \) is the speed (100Mbits/s) and \( CM\_response\_time \) is the time it takes for a compression master to calculate the response message that it sends to the SC:s.

### 3.1.2 Time-triggered frames

Time-triggered traffic frames are only allowed to be sent from devices to switches at certain moments in time. This is different from normal (best effort) Ethernet traffic where frames can be sent at any time.

The system must be able to differentiate between TT frames and other frames in order to have a specific flow for time-triggered traffic [3, section 3.1.1]. How this is done is up to the system designer. For example, specific source or destination addresses could be used to tag TT frames. The type field in the Ethernet frame could also be used for this purpose. In this case the type field is only allowed to be 0x88d7.
3.1. TTETHERNET SPEC

A device is only allowed to send a TT frame to a switch at a certain time. This time is called the acceptance window. The scenario is depicted in Figure 1. In order to achieve this the device and the switch have synchronised clocks. Synchronisation of clocks has some time difference that is bounded. This synchronization error is called Pi in the figure. A frame may be transmitting on the link and that frame has to finish before the TT frame can be sent. This is called send delay in the figure. max(send delay) is the maximum send delay that can occur. hw_max_send_delay is the delay in the hardware that is not caused by queueing effects. dispatch_pit (dispatch point in time) is the time when a frame is scheduled for transmission. send_pit is the time when the first bit of the symbol after the Start of Frame delimiter starts transmitting on the link. Switch B senses the first bit of the frame at receive_pit. The time difference between send_pit and receive_pit is called link latency. The acceptance window starts at accept_pit. accept_pit is defined to be dispatch_pit + link_latency - Pi. The link latency is static and is only dependent on the physical length of the link and the speed that the signal travels through the link. The specification [3, section 3.1.1] defines the size of the acceptance window to be 2Pi + max(send_delay).

Discussion: The author of this thesis believes that the specification misses some things. One such thing is hw_max_send_latency. send_pit might occur at dispatch_pit + max(send_latency) + hw_max_send_latency. The acceptance window only takes into consideration the synchronisation error, max(send_delay) and the link latency. In the case of maximum synchronisa-
tion error receive.pit might occur at dispatch.pit + Pi + max(send_latency) + hw_max_send_latency + link_latency. The acceptance window ends at accept.pit + 2Pi + max(send_delay) = dispatch.pit + link_latency - Pi + 2Pi + max(send_delay) = dispatch.pit + link_latency + Pi + max(send_delay). Since this is earlier, receive.pit occurs outside of the acceptance window. To solve this hw_max_send_latency should be included in the acceptance window. Another issue is that receive.pit is defined to be when the first bit of the frame arrives at the receiver. Do they mean the leading edge of the first bit after the Start of Frame delimiter? That is the same bit that triggers send.pit. Otherwise the link latency is not the time it takes for the signal to propagate through the link.

Figure 2 depicts the scenario with updated values.

3.1.3 Rate constrained frames

RC (rate constrained) traffic frames can be sent at any time but not directly after each other. The time that a frame is not allowed to be sent is called bandwidth allocation gap (BAG). RC traffic adheres to the AFDX standard. RC frames can be forwarded to one or many destinations. AFDX uses dedicated Ethernet addresses to route packets to a set of destinations. The route is in AFDX terms called ”virtual link”. These Ethernet addresses have
one 24-bit constant part and one 16-bit part that identifies what virtual link
the frame belongs to. AFDX allows only one end system to be the source of
frames that are sent over a virtual link [4, page 11]. Each virtual link also
has a maximum frame size. The length of the BAG is allowed to be a value
of the power of 2 in ms between 1 ms and 128 ms [4, page 14]. Because of
the BAG and the maximum frame size the bandwidth that each virtual link
consumes on a physical link is bounded.

3.1.4 Best effort frames
BE (best effort) traffic frames are normal Ethernet frames that are acted on
just as they would in a normal Ethernet network.

3.2 Integration Algorithms
How frames should impact each other when scheduled on a link depending
on their priority is called an integration algorithm.

3.2.1 Shuffling
When using the Shuffling integration algorithm a higher priority frame waits
until the frame in transmission is finished. This introduces a delay to the
high priority packet. But there is no loss of bandwidth. In this case the high
priority packet is delayed for a time that is by at most the maximum frame
size divided by the link speed. In the case of a full size frame at 100Mbits/s
this is \( \frac{1538 \times 8}{100000000} = 0.00012304s = 123\mu s \). Higher speeds give a lower value
on this. [5, section 8.2.1.4]

3.2.2 Preemption
The preemption algorithm works by cutting off the lower priority frame
while it is in transmission. This means that the lower priority frame has to
be retransmitted again later. This has the consequence that the bandwidth
used for the lower priority frame will be wasted. [5, section 8.2.1.4]

3.2.3 Timelyblock
Timelyblock works by using the time triggered schedule to calculate when a
frame is expected to arrive. [5, section 8.2.1.4] The frames are then sched-
uled so that there is no frame in transmission when the high priority frame is
to be sent. If the integration algorithm does not know the size of the frames
it schedules, then the integration algorithm has to expect that the last frame
being sent before the high priority frame is of the maximum frame size. This
3.3. SHUFFLE DELAY

CHAPTER 3. TTETHERNET

means that the last frame must be sent at such a time that even if it has a maximum size it will be finished before the high priority frame is expected to be sent. If the last frame is not of maximum size then bandwidth will be lost. The size of the frames are known in the case that low priority frames (RC, BC) are forwarded using store and forward. It is then possible for the switch to check if any frames could be sent so that they are finished before the high priority frame is expected to be sent.

3.2.4 TTEtherNet requirements

The integration algorithms described above can be divided into two classes, preemptive and non-preemptive integration algorithms. In a non-preemptive algorithm a packet of higher priority has to wait for a packet of lower priority that is already in transfer on the link. In preemptive algorithms the device will make it possible for the high priority frame to be transferred directly if there is no frame that has the same or higher priority that is to be transferred or is in transfer. Shuffling is non-preemptive, Preemption and Timelyblock are preemptive. Devices in an TTEtherNet network must implement a non-preemptive integration algorithm. The devices may implement a preemptive integration algorithm.[3, Section 3.5.1]

3.3 Shuffle delay

As mentioned above TTEtherNet uses a system of output queues of different priorities. When a new frame is to be transmitted over a link the first frame from the highest priority non empty queue is selected. The delay created because of higher priority frames, and frames that are laying before a certain frame in a queue, is called shuffle delay. Realtime messages have deadlines that must be met and, therefore we must have an upper limit to the time it takes for a frame to reach its destination. Calculating the shuffle delay is important because of its impact on the time it takes for a frame to reach its destination. According to the TTEtherNet [3, Section 11.1 Shuffle Delay] specification, it is possible to calculate the worst case (bounded) shuffle delay like this: For a frame going from a Time-Triggered Ethernet device \( k \) to an outgoing link \([v_k, v_y]\), the worst case delay is \(\max(shuffle\_delay[v_k \cdot v_y])\).

A frame that uses outgoing link \([v_k, v_y]\) is called \(f_{i[v_k, v_y]}\). The priority of the frame (that is which queue it belongs to) is called \(f_{i[v_k, v_y]}\).priority and the length of the frame \(f_{i[v_k, v_y]}\).length. The worst case queue length(depth) for priority \(p\) is called \(Q_{i[v_k, v_y]}^{\text{priority}=p}\). The sum of the lengths of all frames in the queue is:
3.3. SHUFFLE DELAY

In the worst case a frame is sent last of all the frames in it’s queue. So it is delayed by its own queue and all queues of a higher priority. This leads to the following equation:

\[ Q_{\text{priority}=p}^{[v_k, v_y]} = \sum_{f_i^{[v_k, v_y]} \cdot \text{priority}=p} f_i^{[v_k, v_y]} \cdot \text{length} \]

The maximum shuffle delay is dependent on the integration policy that is used in the system. If shuffling is used it’s possible that a frame of lower priority is transmitting. So this has to be considered. The worst case is a frame of maximum length. \( \max_{f_i^{[v_k, v_y]} \cdot \text{priority}<p} (f_i^{[v_k, v_y]} \cdot \text{length}) \). So the maximum shuffle delay for a frame that is in a queue with priority \( p \) that is sent over a link is the sizes of the queues with a higher and equal priority of the frame divided by the link speed plus the maximum frame size of a lower priority frame divided by the link speed.

\[
\text{max}(\text{shuffle\_delay}^{[v_k, v_y]}_p) = \frac{Q_{\text{priority} \geq p}^{[v_k, v_y]} + \max_{f_i^{[v_k, v_y]} \cdot \text{priority}<p} (f_i^{[v_k, v_y]} \cdot \text{length})}{\text{link\_speed}}
\]

3.3.1 Shuffle delay of RC frames

RC frames can be sent at any time if a certain time has passed from when the last frame was sent. In the worst case scenario frames that are routed over a certain link might arrive to be sent over the link at about the same time. So they will be queued and this will create a shuffle delay. The AFDX specification specifies limits for shuffle delay of its (RC) frames [6, Equation 1]. It demands that frames should not be delayed more than 500 microseconds and that the delay is less than 40 microseconds + the time it takes to send a frame of maximum size from each virtual link queued after each other. This can be put into an equation system. “Set of VL:s” is the number of virtual links passing through an output port. Payload is the size in octets of the DATA section of an Ethernet frame.

\[
\begin{align*}
\text{max\_jitter} & \leq 40 \mu s + \frac{\sum_{j \in \{\text{set of VL:s}\}} ((38 + \text{payload}) \times 8)}{\text{link\_speed}} \\
\text{max\_jitter} & \leq 500 \mu s \\
0 & \leq 460 \mu s - \frac{\sum_{j \in \{\text{set of VL:s}\}} ((38 + \text{payload}) \times 8)}{\text{link\_speed}} \\
\frac{\sum_{j \in \{\text{set of VL:s}\}} ((38 + \text{payload}) \times 8)}{\text{link\_speed}} & \leq 460 \mu s
\end{align*}
\]
As seen in the equation above the maximum bits in the output buffer to a output port is 460 micro seconds multiplied with the link speed. AFDX specifies the smallest BAG to be 1 ms. In the worst case scenario this buffer will fill up directly after one millisecond. 1 ms is larger than 460 micro seconds so the buffer will be emptied before any new frames arrive. This means there is no risk of packet loss due to filled buffers in this scenario. In a worst case scenario the output link is used for 460 microseconds every millisecond. This leads to a network utilization of $\frac{460 \mu s}{1000 \mu s} = 46/100 = 46\%$. In TTEthernet RC traffic might also be affected by higher priority frames. These frames has to be considered and added to the equations above. In the case of one Compression Master the PCF frames occupy the network flow with as many frames as there are Synchronisation Masters, as mentioned in the description of PCF frames. In the worst case scenario all of these will delay the RC frames. The equation will look like this then PCF frames are considered.

$$46 \times 8 \times SM + \sum_{j \in \{\text{setofVL:s}\}} ((38 + \text{payload}) \times 8) \leq 460\mu s \times \text{link\_speed}$$

$SM$ here is the number of Synchronisation Masters. The size of a PCF frame is 46 bytes hence $46 \times 8$. 
Chapter 4

Comparison with other systems

4.1 FlexRay

4.1.1 FlexRay protocol

FlexRay is a network protocol that uses a shared bus. It can have star topology but the bus is shared. This is similar to how Ethernet originally worked. All devices in a FlexRay network are time synchronized. Time is divided into cyclic periods. These periods are divided into a static segment and a dynamic segment. The static segment uses TDMA scheduling and the dynamic segment uses FTDMA (flexible time division multiple access). FTDMA uses minislots and a priority system. The device with the highest priority is allowed to send directly on the first minislot. If it does not send in this first slot, it’s an indication to the next device that it is allowed to send in the next mini slot and so on. This is similar to the way the CAN bus acts. So for every device that does not choose to send a frame only a mini slot is lost not a whole frame.

4.1.2 Comparison with FlexRay

An important difference between TT Ethernet and FlexRay is that FlexRay uses a shared bus while TT Ethernet uses switched networking. What TT Ethernet tries to accomplish with TT frames is similar to what FlexRay accomplishes with its static segment. Both TT Ethernet and FlexRay use a TDMA based scheme when sending real-time data.

From a fail-safe perspective TT Ethernet isolates faulty ECU:s since TT frames have allocated slots in the network. So, these faulty units don’t interfere with other units traffic. FlexRay also has a problem with signal propagation over long cables when on a shared bus.
There are papers that compare TTEthernet with FlexRay[7]. They conclude that TTEthernet can be used to replace FlexRay and that the latencies in TTEthernet are low enough for transporting the same data as FlexRay.

4.2 CAN

4.2.1 CAN protocol

CAN is a shared bus network architecture that uses CSMA/CR (Carrier sense multiple access/conflict resolution). In CSMA/CR every device has a priority. Before a packet is sent, there is a global arbitration period. During this period all devices that want to send a packet send out their priority and, at the same time, listen to the line. The devices sense if they are the one with the highest priority that want to send. If not, they try again during the next arbitration period.

4.2.2 Comparison with CAN

The CAN bus is a shared bus. CAN also has quite a low bandwidth, while Ethernet has a high bandwidth. This is one of the reasons the industry is so interested in Ethernet. CAN uses priority based scheduling which is different to the TDMA approach used by TTEthernet. There exists schedulability analysis on CAN[8] based on the idea that the priority based scheduling is similar to priority based scheduling of processes in an operating system.

4.3 AVB

4.3.1 AVB Ethernet protocol

AVB (Audio Video Bridging)[9] is an extension to full-duplex Ethernet. AVB uses a priority based model. Frames belong to different traffic classes. Each class has a priority. The priority scheduling is non-preemptive. AVB also uses a credit based shaping algorithm (CBSA)[10, Section 8.6.8.2]. The CBSA has a variable that is reduced with the number of bits that was sent. The variable is replenished at a constant bits/second rate. Frames are only scheduled for sending if this variable is non negative. If there are no frames in the output queue and the variable is positive it is set to zero. If more than one output queue has scheduled frames, the frame in the queue with the highest priority is sent. Having a CBSA on an traffic class is not mandatory.

4.3.2 Comparison with AVB Ethernet

AVB involves a traffic shaper to ensure that low priority frames are not starved. TTEthernet achives this by time-scheduling of TT frames and by the AFDX BAG mechanism of RC frames. There are research papers that
compare TTEthernet with AVB[11][12]. They conclude that TTEthernet provides better latencies for cross-domain control traffic and that AVB has features that benefit multimedia applications. The authors want to combine TT traffic with AVB in the same network. This has been tested by the authors in later simulations[13].
Chapter 5

Open Problems

Even though TTEthernet is a promising candidate, there are still several open problems that need to be solved before it matures as a de-facto automotive protocol.

5.1 Incremental design

Adding a new control unit is easy, in TTEthernet it is as easy as adding a new computer to an Ethernet network. This is true if the unit is to send and receive plain Ethernet frames. This is not true for a TTEthernet device. The schedule that guarantees that messages are sent without collision and that dictates when a device is allowed to send its frames is calculated offline and adding a device might lead to recalculating this schedule. How to design a schedule where you can add a device without changing the scheduling of the already attached devices? One way to guarantee that a TTEthernet device can be added to the system without effecting the scheduling of the already attached devices is to have space left at the end of the lowest common period.

In the above scenario a TTEthernet device can be added that needs to send a frame of full size over this link, independent of its period. Leaving slots unused for future use can lead to low utilization of the network. Leaving one full size frame available at every 1ms period leaves 12.5% of the network unused (of course this space can be filled with BE frames if used) when
running at 100Mbit/s. A constructor might want to have as high utilization as possible to meet constraints set by the project. To have a high utilization factor and be able to add devices without affecting the scheduling of the other devices remains an open problem.

5.2 Mixed frame types

Unlike FlexRay, where static frames must be sent first in the cycle, TTEthernet allows TT frames and RC frames to be sent anywhere in a cycle. One problem that arises is to guarantee the delay of RC messages when configuring the TT schedule. Solutions to such problems must bear in mind the formulas mentioned in section 3.3.1. The impact of integration algorithms such as shuffling and preemption must be taken into account in case they are used. There are research papers that are trying to solve this [14].
TTEthernet is an extension to normal Ethernet that adds time scheduled traffic in order to reduce delay caused by queueing in the output ports. It allows usage of AFDX traffic as well as normal best effort Ethernet traffic. Traffic is prioritized in descending order TT, AFDX and normal Ethernet. TT frames are put on outgoing links either by waiting on the lower priority frame already in transmission, cutting of it’s transmission or making sure that lower priority frames are not in transmission when the TT frame is to be sent. As a protocol that might be used in real-time environments such as control systems for cars etc, it seems to fulfil the requirements. Simulations show that it can achieve guaranteed delays below the $100\mu s$ that is needed for real-time control applications[12]. This is dependent on the configuration of the TTEthernet network. A 100Mbit/s network that uses the shuffling integration algorithm and includes BE traffic with an MTU of 1500 octets might get delays over $100\mu s$ for TT frames. Due to this, with speeds at 100Mbit/s it is needed to use a preemptive integration algorithm to get guaranteed delays below $100\mu s$.

Ethernet provides high link speeds, 100 Mbit/s and upwards. This is to TTEthernets advantage compared with other real-time networking protocols not based on Ethernet, such as CAN or FlexRay. TTEthernet has a low enough worst case delay to transport data intended for FlexRay networks. It provides lower delays for real-time control traffic than AVB Ethernet.

There are open problems to create optimal implementations of TTEthernet that need to be solved. For example, the problem of how to add new TTEthernet devices without changing the schedule of the already connected TTEthernet devices. There is also the problem of how to create schedules that guarantee the delay of RC frames.
Copyright
acknowledgements

Table 1 is based on Figure 3 and Table 2 is a replica of Table 3 in the document "Ethernet Theory of Operation"[2], used by permission from Microchip inc. under the following conditions:
If you use Microchip copyrighted material solely for educational (non-profit) purposes falling under the “fair use” exception of the U.S. Copyright Act of 1976 then you do not need Microchip’s written permission. For example, Microchip’s permission is not required when using copyrighted material in: (1) an academic report, thesis, or dissertation; (2) classroom handouts or textbook; or (3) a presentation or article that is solely educational in nature (e.g., technical article published in a magazine). Please note that offering Microchip copyrighted material at a trade show or industry conference for the purpose of promoting product sales does require Microchip’s permission.
Source:
References

[1] Lucia Lo Bello. 
“The case for ethernet in automotive communications”. 

“Ethernet Theory of Operation”. 
In: (2008). 


“AFDX/ARINC 664 protocol tutorial”. 
In: (2007). 

Time-Triggered Communication. 
CRC Press, 2011, 
Pp. 181–220. 

“Comparison of IEEE AVB and AFDX”. 
2012, 
DOI: 10.1109/DASC.2012.6382405.
“Comparing time-triggered Ethernet with FlexRay: An evaluation of competing approaches to real-time for in-vehicle networks”.
Nancy, France, 2010,
DOI: 10.1109/WFCS.2010.5548606.

[8] Robert I. Davis et al.
“Controller Area Network (CAN) schedulability analysis: Refuted, revisited and revised”.
ISSN: 0922-6443.
DOI: 10.1007/s11241-007-9012-7.
URL: http://dx.doi.org/10.1007/s11241-007-9012-7.

DOI: 10.1109/IEEESTD.2011.6032690.

[10] “IEEE Standard for Local and metropolitan area networks–Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks”.
DOI: 10.1109/IEEESTD.2011.6009146.

“Tomorrow’s In-Car Interconnect? A Competitive Evaluation of IEEE 802.1 AVB and Time-Triggered Ethernet (AS6802).”
ISSN: 1090-3038.
DOI: 10.1109/VTCFall.2012.6398932.
URL: http://papers.till-steinbach.de/slkh-tiice-12a.pdf.

“Simulative assessments of IEEE 802.1 Ethernet AVB and Time-Triggered Ethernet for Advanced Driver Assistance Systems and in-car infotainment”.
In: Vehicular Networking Conference (VNC), 2012 IEEE.
2012,
DOI: 10.1109/VNC.2012.6407430.
In: *2013 IEEE Vehicular Networking Conference (VNC)*.  
Boston, Massachusetts: IEEE Press, Dec. 2013,  
DOI: 10.1109/VNC.2013.6737589.  

“Synthesis of Communication Schedules for TTEthernet-based Mixed-criticality Systems”.  
Tampere, Finland: ACM, 2012,  
Pp. 473–482.  
DOI: 10.1145/2380445.2380518.  
URL: http://doi.acm.org/10.1145/2380445.2380518.
På svenska

Detta dokument hålls tillgängligt på Internet – eller dess framtida ersättare – under en längre tid från publiceringsdatum under förutsättning att inga extra-ordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns det lösningar av teknisk och administrativ art.

Uphovsmannens ideella rätt innefattar rätt att bli nämd som upphovsmann i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida http://www.ep.liu.se/

In English

The publishers will keep this document online on the Internet – or its possible replacement – for a considerable time from the date of publication barring exceptional circumstances.

The online availability of the document implies a permanent permission for anyone to read, to download, to print out single copies for your own use and to use it unchanged for any non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional on the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its WWW home page: http://www.ep.liu.se/

©Niclas Rosenvik