A performance analysis of EtherCAT and PROFINET IRT

Gunnar Prytz
ABB AS Corporate Research Center
Bergerveien 12
NO-1396 Billingstad, Norway
Gunnar.Prytz@no.abb.com

Abstract

Ethernet is emerging as the preferred industrial communication platform upon which a multitude of protocols and solutions specially targeted at industrial communication requirements have been developed. This paper concentrates on a set of protocols popularly referred to as the Real-time Ethernet protocols and analyses two key technologies, namely PROFINET IRT and EtherCAT.

The real-time performance characteristics of PROFINET IRT and EtherCAT is compared by looking into what factors affect their performance in a set of realistic scenarios. The goal is to provide a scientifically solid study on this subject where many presentations are seriously flawed by basic mistakes, misunderstandings or biased views. The detailed analysis presented here shows that EtherCAT has a performance advantage over PROFINET IRT and the reasons for this are discussed.

1. Introduction

There is no longer any doubt that Ethernet is here to stay in the world of industrial communication systems. The traditional fieldbuses and the directly coupled signals are slowly and in many cases not so slowly being replaced by various Ethernet based communication solutions as confirmed by a recent ARC study [5].

The move from the fieldbus-based communication towards industrial Ethernet unfortunately does not mean that the industry moves from incompatible communication solutions to a scenario with all compatible Ethernet based devices. The various industrial automation protocols on top of Ethernet are in most cases not compatible with each other, e.g. a PROFINET device can not easily talk to an Ethernet/IP device. Various resolutions to this problem exist, e.g. through proxy devices or components or by running numerous communication stacks simultaneously, but there continues to be quite a degree of incompatibility among devices and protocols on industrial Ethernet.

The move towards Ethernet as the basic communication platform is based on the excellent price/performance relationship of the technology. However, there are areas where there is a need for the very highest performance that Ethernet technology is able to deliver, using either hardware or software augmentation or both. This includes devices within robotics, high performance PLC systems, motor control systems and instrumentation and I/O systems. Common to all these are challenging communication requirements on low latency, high update rates and high throughput. It is mainly for these systems and applications that the two protocols EtherCAT and PROFINET IRT have been developed.

Due to its success particularly in office communication networks there exist a large array of mature and well-proven Ethernet hardware and software solutions. These include Ethernet network infrastructure components like network interfaces, switches, routers and cables. All the little bits and pieces that together constitute the hardware in a typical Ethernet network (e.g. MACs, PHYs, cables, connectors) come at a fairly low cost due to the high volumes. In addition to this there is a wide variety of mature software solutions taking care of various aspects of the Ethernet communication such as data transport, network management, addressing, redundancy, discovery, security and safety. Another argument in favor of Ethernet technologies is the widespread competence on these technologies making it even more appealing for automation companies who have systems to develop, manufacture, commission and maintain.

However, Ethernet based technologies also represent a challenge to automation companies since these companies need to gain a high level of insight and competence within this fairly complex and wide area in a fairly short time. Some of the challenging aspects that automation companies face are discussed in more detail in [10].

EtherCAT and PROFINET IRT are two examples of hardware-augmented Real-time Ethernet (RTE) protocols. Although there exists no universally accepted definition for the term “real-time” in this context, the term “Real-time Ethernet” protocols seems to have established itself throughout the industrial arena and academia. The lack of definition is not a big problem here since most people seem to know what is meant by the term, thus there is some sort of de-facto definition in use.

EtherCAT and PROFINET IRT are discussed in detail in this paper since they are very interesting both from a technical and market perspective. EtherCAT has grown to be a market winning technology due to the combination

1-4244-1506-3/08/$25.00 ©2008 IEEE

of very high real-time performance, a large set of available functionality and the general openness and availability of the technology. PROFINET IRT on the other hand is at the moment a marginal market player. This is believed to be partly due to the fact that it was introduced to the market some years after EtherCAT but since it is part of the PROFINET protocol suite it will no doubt have momentum on the market and is thus a relevant technology to be compared with EtherCAT. It is believed that it is of key interest to know more about the relevant differences between these two protocols for automation manufacturers in need of an RTE protocol. It should also be stressed that this papers compares the present, commercially available versions of the technologies. Both technologies are surely under continuos development as indicated by a recent paper on PROFINET IRT [9].

It is a challenging task to compare two such protocols. Such an undertaking asks for critical focus, particularly from technology providers or others with strongly biased views. It is therefore the goal of this paper to base the comparisons on a scientifically strong basis by only drawing conclusions that can be argued for and by presenting results that are repeatable. The results presented are drawing upon years of experience with relevant Ethernet technologies by the author and colleagues.

It seems to be a critical lack of unbiased and scientifically based, in-depth presentations and papers within the subject of comparisons of industrial Ethernet protocols. This is no less true for the RTE protocols. In fact, the RTE protocols were among the first industrial Ethernet protocols to reach the market so they have actually been around for quite some time (a few years in this context). The technology providers to a varying degree seem to use every opportunity not only to present the positive aspects of their own technology but also to come forward with the disadvantages of competing technologies. As such this topic has a political aspect to it, not unlike the discussions around fieldbuses. This is fine in principle but in practice it is seen that when lack of understanding of competing technologies is paired with a biased view the result is of questionable use to end users. Above all, such presentations should mostly be viewed as marketing material.

It can thus be argued that the end users of industrial Ethernet technologies, of which the author is a representative, is in need for both increased levels of competence on industrial Ethernet and RTE technologies and for unbiased scientifically based information.

2. The problem at hand

A comparison of two protocols which are designed more of less for the same problem ideally should be done by implementing both protocols in an optimal way and then to analyze the outcome. This is in practice not a feasible way, particularly since very few end users would do exactly the same thing twice without a very good reason. In addition this would only provide one example and would hardly be general enough to serve as a basis for a more general comparison. This paper thus performs a comparison based on a number of calculations and simulations and by doing some quantitative analyses based on the available information found in specifications [2] and on the available material found on the web pages of the supporting technology organizations [6], [7].

In more concrete terms a comparison of two RTE protocols has to look at one of the key variables of a such an automation system, namely the cycle time. Other issues like communication jitter, available functionality, limitations found in specifications or implementations, cost of implementations, availability of relevant software and hardware as well as products will also be touched upon in this paper. Some of the issues will be covered by comparing protocol features, some will be handled by providing example calculations of performance (e.g. minimum achievable cycle time) whereas others will be discussed in an more quantitative manner.

3. EtherCAT

EtherCAT is a protocol offering very high real-time performance and was developed by Beckhoff. Both the EtherCAT master and the slave can in principle be implemented using entirely standard network interfaces (in this context meaning any PC compatible MAC and PHY). However, in practice an EtherCAT slave is implemented with special hardware (e.g. FPGA or ASIC) to facilitate very short packet forwarding delays in the slave devices. The EtherCAT master is on the other hand normally implemented with only standard components.

A typical EtherCAT network consist of one master and a number of slaves. Currently the network speed is limited to 100 Mbps since no special slave hardware exist for gigabit Ethernet network speeds. However, EtherCAT may in principle be implemented on a switched gigabit Ethernet network using entirely standard hardware both at the master and slave side but the performance will then be limited by forwarding delays of standard switches, the network jitter and the specific topology in use.

At present there exists a fairly wide variety of master implementations for different operating systems. On the slave side there are a number of custom hardware options available including many FPGA solutions, some ASICs as well as network-on-chip solutions. A great variety of products ranging from drives to sensing nodes to I/O systems are offered throughout the market.

All EtherCAT telegrams are generated and sent by the master using full duplex transmission. The telegrams are reflected at the end of each network segment and sent back to the master. The basic operating principle is that all nodes in an EtherCAT network can read data from and write data to the EtherCAT telegram as it passes by with only a short constant delay in each slave device (independent of the size of the packet). The constant node delay is

typically below 500 ns (slight variations between different implementations). Typically, a large number of slaves can be accommodated using only one telegram, thereby optimizing bandwidth usage and decreasing interrupt rates. This very efficient usage of bandwidth is enabled by a concept called logical addressing.

Furthermore, EtherCAT is able to carry any type of standard Ethernet traffic transparently (in theory it is also possible for both EtherCAT traffic and any other Ethernet traffic to coexist when standard hardware slaves are used together with standard switching devices according to IEEE 802.1D). If the Ethernet frame is too large to be transported in the available time or if it does not fit into the available space in one EtherCAT frame it is fragmented and then reassembled at the receiving end. This is an important feature which does not limit the achievable cycle time due to time reserved for asynchronous traffic only. EtherCAT frames are either standard Ethernet IEEE 802.3 frames or encapsulated in UDP/IP telegrams, thereby enabling routing (at a lower degree of performance).

EtherCAT offers slave synchronization performance down to well below the hundreds of nanoseconds through special timing functionality found in the hardware-augmentation part of the slaves which is an attractive feature for high performance control systems and instrumentation systems. Slave devices may have from one up to at least 4 ports and any topology is in principle supported although the line (and possibly the ring) topology is the most common. Other features like profiles, redundancy, security, discovery etc. are also supported.

4. PROFINET IRT

PROFINET IRT is a part of the PROFINET protocol suite developed by Siemens. PROFINET IRT was for a long time expected with quite some anticipation but the market introduction was late compared to rivalling technologies. PROFINET IRT requires special hardware support both on the controller (master) side and on the device (slave) side. At the moment there are very few options for implementing a PROFINET IRT node and an ASIC offered by Siemens is in practice the most realistic option. The ASIC comes in two versions, one supporting nodes with two ports and one supporting nodes with four ports. The product spectrum is still very limited, partly because the technology has not been around that long. Siemens provides both controllers and devices (e.g. drives) and also infrastructure components (switches).

The operating principle of PROFINET IRT is based on dividing the cycle time into an asynchronous and a synchronous part. A scheduling mechanism is used to set up both the asynchronous and the synchronous slots within the communication cycle. This scheduling is dependent on the topology and the details of how the scheduling works seem to be fairly well protected. A number of topologies are supported including line, ring and star. The star topology requires custom switches (IRT compatible cut-through switches) although standard switches may be used at the expense of performance. The jitter in a PROFINET IRT network is specified to be better than 1 µs. Cycle times from 250 µs and upwards are supported. The reason for this limitation is that at least 50% of the available time is set aside for asynchronous communication and a maximum sized Ethernet packet has to be supported (which may take up to 123 µs to forward). Since no fragmenting functionality like that in EtherCAT is available the cycle times are limited down to 250 µs (for 100 Mbps bandwidth).

The forwarding delay of a PROFINET IRT slave is the delay of the cut-through switch. In practice this will be approximately 3 µs per device [1].

5. Performance analysis

The performance of EtherCAT and PROFINET IRT will be compared by means of calculating the minimum achievable cycle times as a function of the number of slaves/devices for both technologies in a number of scenarios. Here the term “slave/device” is used to denote either slave (EtherCAT) or device (PROFINET IRT). These scenarios are defined as follows:

- **16 bytes payload per slave/device.**
- **36 bytes payload per slave/device.**
- **100 bytes payload per slave/device.**
- **EtherCAT vs. PROFINET IRT as a function of the payload in bytes per slave/device (for a given number of devices).**

The scenarios above will be evaluated both for 100 Mbps and 1 Gbps bandwidth although in practice only 100 Mbps bandwidth is supported by both protocols today. The assumptions behind the calculations for 1 Gbps bandwidth will therefore be described in detail.

Since the performance is somewhat topology dependent it is necessary to use realistic and efficient topologies in the calculations. It will be shown below that the line topology is very efficient both for EtherCAT and PROFINET IRT. PROFINET IRT may be implemented in a star or tree (complex) topology by the use of custom (cut-through) switches. Similarly, EtherCAT may be implemented in a star or tree (complex) topology by either using some EtherCAT devices with more than two ports or by using standard Ethernet switches (at the expense of some more jitter). In this context it should be noted that by using only standard EtherCAT slaves to construct complex topologies the logical topology is in practice a linear topology.

Both EtherCAT and PROFINET IRT use standard Ethernet frames as defined by IEEE 802.3 [3] with total frame lengths ranging from 64 to 1518 bytes. When packets are to be sent back-to-back (at full speed) the interpacket frame gap of 12 bytes and the preamble of 8 bytes also have to be taken into account.

5.1. EtherCAT cycle time - line topology

The EtherCAT cycle time is made up by a weighted sum of the following components:

- **Master packet forwarding time**
- **Master PHY delay**
- **Slave PHY delay**
- **Slave forwarding delay**
- **Propagation delays along the cables**
- **Network idle period**

Before continuing the following components needs to be defined:

T^{ETC} = minimum achievable EtherCAT cycle time

T_m^f = forwarding time of the packet at the master

T_P = maximum delay of the PHY (Rx+Tx).

T_c^p = propagation delay along the cables

T_s^f = forwarding time of the packet inside the slave

T_{1538} = forwarding time of a maximum sized packet

T_s^{tot} = total delay introduced by an EtherCAT slave

n = number of slaves

k = number of packets used per EtherCAT cycle

p = payload per slave in bytes

bw = bandwidth in bits per second

When calculating the minimum achievable cycle time the network idle period is obviously set to zero. The forwarding delay of the master depends on the packet length and the network speed, e.g. 100 Mbps or 1 Gbps. The forwarding delay of an EtherCAT slave depends somewhat on the actual implementation but is currently in the order of hundreds of nanoseconds. A realistic value for a common slave implementation is approximately 450 ns (the sum of the delay in both directions). Adding tho these numbers are the PHY delays which varies somewhat between different models but the sum of the receive and transmit delays of a PHY should in practice not be above 500 ns. In addition there is the cable propagation delay which is typically less than 50 ns per 10 m cable (Cat 5 and upwards). This then leads to the following formula describing the total delay per EtherCAT slave in a line topology:

$$T_s^{\mathrm{tot}} = T_P + T_c^p + T_s^f \qquad (1)$$

For simplicity the total slave forwarding delay and the cable propagation delay associated with a slave can be set to 1 µs (which is a realistic estimate). Assuming that only one packet is required to handle all the EtherCAT communication in a cycle, the minimum achievable cycle time is reduced to the following expression:

$$T_{\mathrm{line\ (1\ packet)}}^{\mathrm{ETC}} = T_m^f + T_P + n \cdot 1\ \mu s \qquad (2)$$

If more than one packet is required the expression gets slightly more complex, thus the general expression for the minimum achievable EtherCAT cycle times is given by the following formula:

$$T_{\mathrm{line}}^{\mathrm{ETC}} = (k-1)T_{1538} + T_{rest} + T_P + n \cdot 1\mu s \qquad (3)$$

The number of maximum sized packets k required is given by the quotient of the total amount of data to be sent and the maximum EtherCAT payload in each packet which is 1488 bytes (for details see the EtherCAT specification [2]):

$$k - 1 = np \ \mathrm{DIV}\ 1488 = \left\lfloor \frac{np}{1488} \right\rfloor \qquad (4)$$

Similarly, T_{rest} is the forwarding time of the one packet which is not maximum sized and is given by the following expression:

$$T_{rest} = \frac{np \ \mathrm{MOD}\ 1488}{bw} = \frac{np - 1488\left\lfloor \frac{np}{1488} \right\rfloor}{1488} \qquad (5)$$

Finally T_{1538} is given by

$$T_{1538} = \frac{1538}{\mathrm{bw}} \qquad (6)$$

The formula given by Equation 3 above will be used in the calculations of the EtherCAT cycle time in the line topology in this paper.

5.2. EtherCAT cycle time - star topology

It is possible to construct very complex topologies which logically are linear topologies (due to the forwarding nature of a hardware-augmented EtherCAT slave). The minimum achievable cycle times will therefore be nearly (a small additional delay of typically around 200 nanoseconds is introduced by any device with more than two ports) equal to the line topology case (Equation 4) with the equal number of slaves.

In addition it is also possible to use a star topology with standard store-and-forward switches. In practice this could be realized by having the EtherCAT master simply organizing the data into one packet per segment (here a segment is the set of devices connected to one switch port). The calculations get slightly more tricky here and due to constraints on the paper length the formula for the minimum achievable cycle time will not be developed in detail. The resulting cycle time consist of a weighted sum of the forwarding times for all packets from the master, the forwarding time of the switching devices and the forwarding delays of the slaves. For a star topology with a number of line topologies connected to a switch which is again connected to the EtherCAT master the resulting minimum cycle time is given by the following equation:

$$T_{\text{star}}^{\text{ETC}} = \sum_{k} T_m^{\,f} + 2T_{\text{switch}}^{\,f} + T_{line}^{\text{ETC}} + \sum T_n \quad (7)$$

Here $\sum T_n$ is the sum of all delays related to PHYs and cable propagation on the network.

5.3. EtherCAT and Gigabit Ethernet

Although no hardware assisted implementation of EtherCAT exists for 1 Gbps Ethernet today this will presumably come within some time, in line with a general move from 100 Mbps to 1 Gbps in the industry (such a move is well on the way in office networks).

The calculations for EtherCAT on 1 Gbps bandwidth are based on the following assumptions (only changes from 100 Mbps shown):

- **Packet forwarding delays reduced by a factor of 10**
- **The forwarding delay of a slave including PHY delays (Rx and Tx) is reduced from 1 μs to 0.6 μs.**
- **The MTU was raised from 1500 to 9000 bytes.**

5.4. PROFINET IRT cycle time - line topology

The situation for PROFINET IRT is in general that one packet is sent from the master (controller) to each slave device in the network during a cycle and every device will also send a packet to the master. Where EtherCAT uses only one packet per cycle a PROFINET IRT networks thus will send a number of packets proportional to the number of slave devices. To compensate for some of the overhead introduced by this drawback the so-called slipstreaming effect has been developed to minimize the cycle time in this situation [1]. Slipstreaming is simply said a way where the packets are scheduled in a way that minimizes the total time spent, e.g. by having the master send packets to the devices furthest away from the master first. The slipstreaming effect was used in the calculations for PROFINET IRT in this paper.

By adopting a more complex topology (e.g. a star topology) the forwarding delay of the switching device also has to be taken into account. These two cases will be handled separately in the following analysis.

The components that constitute the PROFINET IRT cycle time are thus the following:

- **Controller packet forwarding time**
- **Controller PHY delay**
- **Device PHY delay**
- **Switch node forwarding delay (complex topology)**
- **Device forwarding delay (receive)**
- **Propagation delays along the cables**
- **Network idle period**

Now the following components needs to be defined:

T^{IRT} = minimum achievable PROFINET IRT cycle time

$T_c^{\,f}$ = forwarding time of the packet at the controller

T_P = maximum delay of the PHY (Rx+Tx).

$T_c^{\,p}$ = propagation delay along the cables

$T_d^{\,f}$ = forwarding time of the packet in a device

n = number of slaves
p = payload per slave in bytes
bw = bandwidth in bits per second

The term $T_d^{\,f}$ is assumed to be the same for both an end node and a switching device.

When calculating the minimum achievable cycle time the network idle period is again set to zero. Only the time required to communicate from the controller to the device is calculated as it is assumed that the communication from the device to the controller takes place simultaneously (full duplex). As for EtherCAT the forwarding delay for the controller depends on the packet length and the bandwidth. The forwarding delay of a PROFINET IRT switching device is estimated to be 3 μs [1], a realistic value for a cut-through switch. The PHY delay (receive plus transmit) is set to approximately 500 ns as for EtherCAT. In addition a small cable propagation delay of 50 ns per 10 m cable should be added.

For the line topology the minimum achievable cycle time for PROFINET IRT is then (taking advantage of the slipstreaming effect):

$$T_{\text{line}}^{\,IRT} = nT_c^{\,f} + T_c^{\,p} + T_d^{\,f} + T_P \quad (8)$$

By utilizing the slipstreaming effect and thus sending the packet to the last device on the line first only the forwarding delay per packet from the controller needs to be multiplied by the number of packets in these calculations.

5.5. PROFINET IRT cycle time - star topology

Due to the slipstreaming effect the delays for the complex (star) topology will be fairly similar to the expression for the line topology in Equation 8 but some components get multiplied by 2:

$$T_{\text{star}}^{\,IRT} = nT_c^{\,f} + 2T_c^{\,p} + 2T_d^{\,f} + 2T_P \quad (9)$$

For other topologies with more complex branching the minimum achievable cycle time of PROFINET IRT would be different (i.e. larger) and it will also be increasingly difficult to schedule the slipstreaming effect. Comparing Equation 8 and Equation 9 it is also seen that the line topology is the most efficient for PROFINET IRT due to the slipstreaming effect benefit in the line topology. For this reason only results from the PROFINET IRT line topology using Equation 8 are presented in this paper.

5.6. PROFINET IRT and Gigabit Ethernet

As for EtherCAT, no hardware assisted implementation of PROFINET IRT for 1 Gbps Ethernet is available today but this will presumably change in the future.

The calculations for PROFINET IRT on 1 Gbps bandwidth are based on the following assumptions (only changes from 100 Mbps shown):

- **Packet forwarding delays reduced by a factor of 10**
- **The forwarding delay of a cut-through switch is reduced from 3 μs to 0.6 μs**
- **The MTU was raised from 1500 to 9000 bytes.**

6. Results

A number of calculations were performed according to the previously defined scenarios. The same calculations were performed both for 100 Mbps and 1 Gbps bandwidth, with the assumptions given earlier in this paper. The results were as follows:

6.1. 16 bytes payload per slave/device.

The minimum achievable cycle time for both EtherCAT and PROFINET IRT was calculated with a payload of 16 bytes per slave/device. In this section the term device is used for both slaves and devices for simplicity. The number of devices ranged from 1 to 200. The results for the bandwidths 100 Mbps and 1 Gbps are shown in Figure 1a and Figure 1b, respectively.

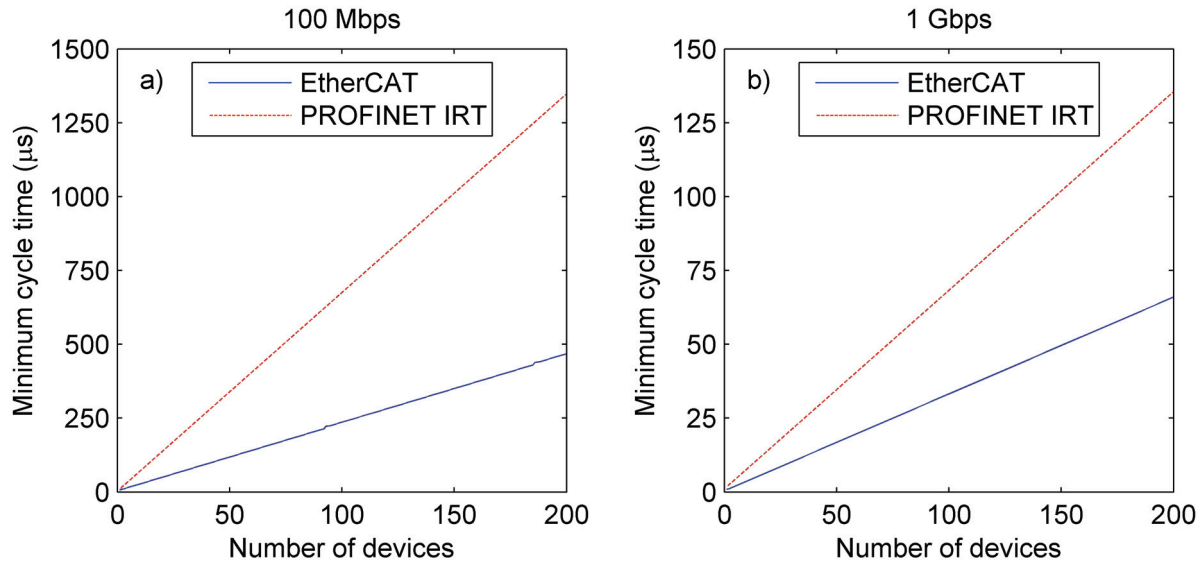

Figure 1. Minimum achievable cycle times at bandwidths of 100 Mbps and 1 Gbps for EtherCAT and PROFINET IRT on a line topology network with 16 bytes payload per device as a function of the number of devices. (The 250 μs limit for PROFINET IRT was not applied.)

6.2. 36 bytes payload per slave/device.

The calculations from Section 6.1. were repeated with 36 bytes payload per slave/device. The results for the bandwidths 100 Mbps and 1 Gbps are shown in Figure 2a and Figure 2b, respectively.

6.3. 100 bytes payload per slave/device.

The calculations from Section 6.1. were repeated with 100 bytes payload per slave/device. The number of devices ranged from 1 to 200. The results for the bandwidths 100 Mbps and 1 Gbps are shown in Figure 3a and Figure 3b, respectively.

6.4. EtherCAT vs. PROFINET IRT as a function of the payload in bytes per slave/device.

The relationship between the minimum achievable cycle times of EtherCAT and PROFINET IRT were calculated as a function of the payload in bytes per slave/device for a networks with 50 slaves/devices. In this calculation only the line topology was used. The results for the bandwidths 100 Mbps and 1 Gbps are shown in Figure 4a and Figure 4b, respectively.

7. Discussion

Comparing the real-time performance of two protocols like EtherCAT and PROFINET IRT is actually quite complex. The results presented here are somewhat different than those presented by Jasperneite et al. [1] although similar scenarios and conditions have been used. This is believed to be partly due to some oversimplifications in the Jasperneite et al. paper [1].

This study has presented a real-time performance comparison of EtherCAT and PROFINET IRT by calculating the cycle time in some scenarios. The analysis shows that the line topology is very efficient for both technologies but that EtherCAT has a clear advantage performance-wise. This should not come as a surprise since EtherCAT uses much less overhead per slave/

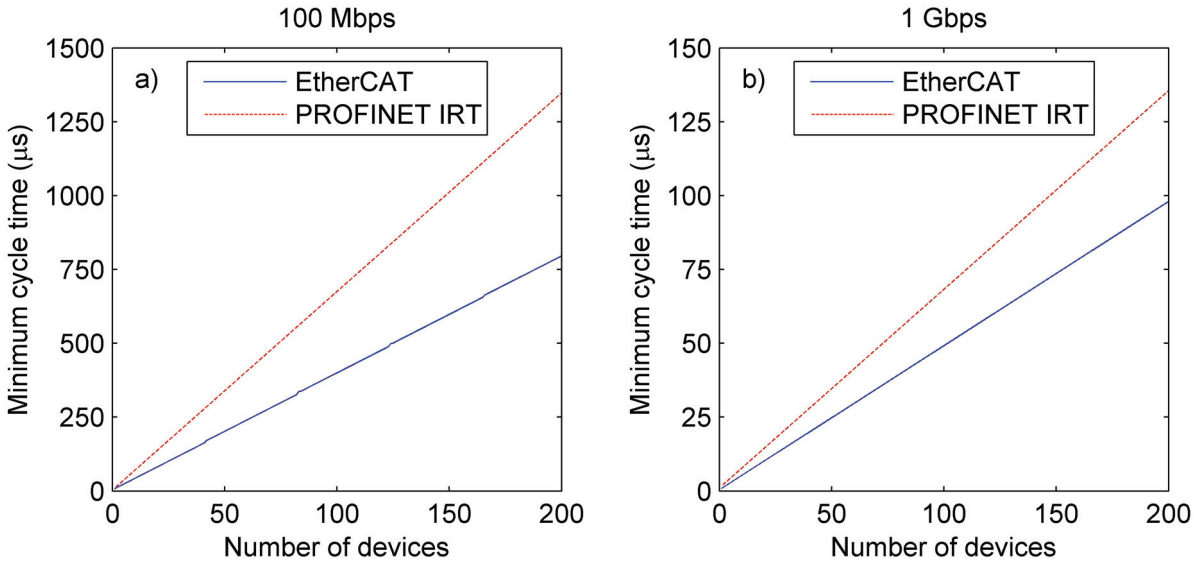

Figure 2. Minimum achievable cycle times at bandwidths of 100 Mbps and 1 Gbps for EtherCAT and PROFINET IRT on a line topology network with 36 bytes payload per device as a function of the number of devices. (The 250 µs limit for PROFINET IRT was not considered.)

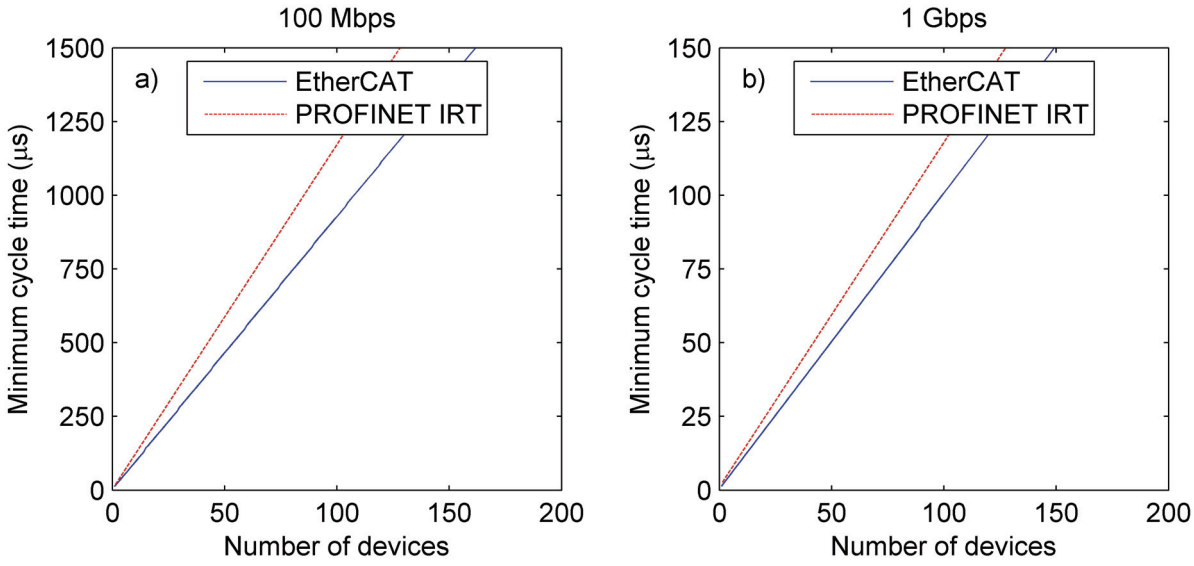

Figure 3. Minimum achievable cycle times at bandwidths of 100 Mbps and 1 Gbps for EtherCAT and PROFINET IRT on a line topology network with 100 bytes payload per device as a function of the number of devices. (The 250 µs limit for PROFINET IRT was not applied.)

device than PROFINET IRT. Any other conclusion would in fact be very surprising due to the fact that where EtherCAT may use only one packet to accommodate the data for a great number of slaves, PROFINET IRT needs one packet per slave device.

The multi-slave frame approach by EtherCAT further benefits from using jumbo frames (larger than 1538 bytes) on gigabit Ethernet, e.g. frames of approximately 9000 bytes which are common in present gigabit Ethernet networks.

A different aspect that should be kept in mind is the load that the two protocols put both on the master/controller and the slaves/devices in the network. In a network with 100 slaves/devices each with 10 bytes payload running at a cycle time of 100 µs the PROFINET IRT controller would need to handle 2 million packets per

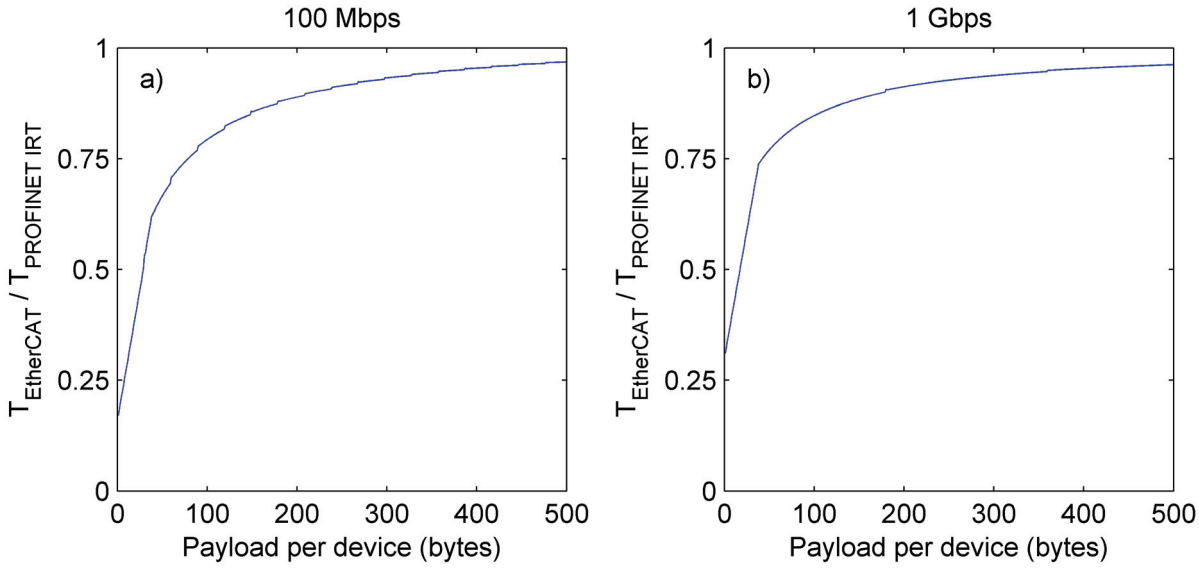

Figure 4. A comparison of the performance of EtherCAT and PROFINET IRT as a function of the payload per device at bandwidths of 100 Mbps and 1 Gbps shows an EtherCAT advantage in both situations. The calculations were performed with 50 devices on the network. (The 250 μs limit for PROFINET IRT was not applied.) The results are similar for other payloads from 0 bytes up to and beyond 50 bytes as can be seen indirectly from Figure 1 to Figure 3.

second, which is much higher than any present CPU can handle without very special measures [8]. In a similar network using EtherCAT the master would only have to handle 20.000 packets per second which is feasible.

8. Conclusion

This paper has focused on hardware-augmented Real-time Ethernet protocols and has presented a performance comparison of the two protocols EtherCAT and PROFINET IRT. The comparison was made by calculating the minimum achievable cycle times in a number of scenarios which are believed to be optimal for both technologies. The results show that EtherCAT offer significantly better performance than PROFINET IRT in all these situations. The reasons for this was discussed and a number of key factors were identified.

It is important to stress that this study does not point out any of the two protocols as a "winner" but merely states that EtherCAT outperforms PROFINET IRT on real-time performance in the realistic scenarios used here. Both technologies may have pros and cons related to factors like integration with other parts of the system and specific functionality requirements and any decision to go for one or the other of the technologies have to take these into account. It should also be noted that both protocols provide a high level of real-time performance.

As a final forward looking and optimistic remark it is both expected and necessary that the development of both technologies continues and that new implementations and products will show increased performance in the years to come. Competition is in the best interest of the end users here as well as most other places.

References

[1] Jasperneite, J, Schumacher M. and Weber, K: Limits of increasing the performance of industrial Ethernet protocols. ETFA 2007. Proceedings, pp. 17-24, 2007.

[2] IEC 61158 ed. 4. Industrial Communication Networks - Fieldbus Specifications. IEC 2007.

[3] IEEE 802.3-2005. Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications. IEEE 2005.

[4] IEEE 802.1D-2004. MAC Bridges. IEEE 2004.

[5] Ethernet-Based Device Networks Worldwide Outlook. Market analysis and forecast through 2012. ARC Advisory Group 2008.

[6] http://www.ethercat.org

[7] http://www.profibus.org

[8] Prytz, G. and Johannessen, S: Real-time Performance Measurements using UDP on Windows and Linux. ETFA 2005. Proceedings, Vol. 2, pp. 925-932, 2005.

[9] Schumacher, M, Jasperneite, J. and Weber, K: A new approach for increasing the performance of the industrial Ethernet system PROFINET. WFCS 2008. Proceedings, 159-167, 2008.

[10] Skaalvik, J. and Prytz, G: Challenges related to Automation Devices with inbuilt Switches. WFCS 2008. Proceedings, pp. 331-339,2008.

[11] Seifert, Rich: The Switch Book. The complete guide to LAN switching technology. John Wiley & Sons, Inc. 2000.