Precision of Ethernet Measurements based on Software Tools

Iwan Schafer, Max Felser
Berne University of Applied Sciences, Engineering and Information Technology
Jlcoweg 1, CH-3400 Burgdorf
iwan.schafer@bfh.ch, max.felser@bfh.ch

Abstract

The use of Ethernet based solutions in automation applications raises the question of test tools. Free tools like Wireshark, formerly known as Ethereal, are available and offer decoding of real-time Ethernet like PROFINET IO. But is such a software based tool able to measure the required real-time constraints? What are the limits of such software based measurements? This report gives an overview of possible measurement setups and a choice of measured benchmarks.

1. Introduction

More and more, Ethernet based fieldbuses are used in practical applications. There is the promise that Ethernet based solutions will be more flexible and permit the end user a smooth integration of his field equipment in the IT world. Furthermore, there is hope that the theoretically faster Ethernet technology will permit faster response and cycle times in practical applications in control.

Most applications with distributed control architectures or remote inputs and outputs are using fieldbuses like the well known PROFIBUS DP. For this application, tools from different manufacturers are existent at reasonable prices. They allow checking and validating the performance of the fieldbus. Switching to an industrial Ethernet based control network raises questions about the availability of tools to verify the performance of an industrial Ethernet solution like PROFINET IO.

Well known in the world of Ethernet is the freely distributed tool Wireshark (formerly known as Ethereal). This network analyser software registers the Ethernet frames and decodes the content of the frames. The used versions of Wireshark are also able to fully decode PROFINET IO-frames. But is the performance of such a software based solution sufficient to give significant information about timing issues?

In this paper we outline the requirements of a typical PROFINET IO application. We show different possibilities to measure with Wireshark in a switched Ethernet network and finally list some results to show the limits of software based measurements.

2. The PROFINET IO-system

In a typical PROFINET IO-system an IO-controller does control one or more IO-devices. During the initialisation sequence, an Application Relation (AR) between the IO-controller and the IO-device is set up. Inside this AR different Communication Relations (CR) are defined. One CR is dedicated for the cyclic exchange of process data. We focus our study on this cyclic process data.

The cycle time of the data exchange may be different for every CR. Data is sent from the producer to the consumer over the CR. The IO-device is the producer and the IO-controller is the consumer for the input data and vice versa for the output data. Every producer is responsible to keep the cycle time within a prescribed jitter. With Class A and Class B PROFINET IO-devices, the cycles are running independent. Only with the more advanced Class C, the cycles in all producers are synchronized. To avoid problems with not synchronized cycles, only full-duplex communication and switched Ethernet at 100MBit/s is allowed for PROFINET IO-systems.

As we want to measure the jitter of the cyclic data exchange in non Class C devices, some questions arise. What are the values for different types of devices? What is the influence of the performance of the device processor and the operating system?

In a comparable PROFIBUS DP-system the DP-master does poll his allocated DP-slaves, running a bit rate of 12MBit/s with a cycle time of 1ms and a jitter of less than 1µs. Is this also possible with a PROFINET IO-system?

3. Possible measurement topologies

The cheapest and most simple solution is to use the freeware tool Wireshark. It is in fact the tool which is recommended by the PROFIBUS International (PI) organisation on their web side. This tool stores and decodes the content of the incoming and outgoing

frames. With a version higher than 0.10.10, PROFINET-frames are decoded too. Wireshark runs on Windows, but also on Linux. So, Wireshark is the tool to choose.

There are several open points about the connection of this tool to the network: Where to measure and how to connect?

On the PROFIBUS DP-bus there is one cable where all frames are going through. To monitor the bus, just plug in your tool at any place on the cable and you can see all transferred frames. In a switched Ethernet network like PROFINET, the situation is different. There is normally no place in the network where it is possible to see all traffic going through! In a switched network frames are sent only over segments they have to pass to reach their destination and there exists by definition no single point where all traffic has to go through. In our case - a PROFINET IO-system with just one IO-controller, one IO-device and no IO-supervisor or other devices connected to the network - only two stations are communicating with each other. With this setup, the point of measure can be chosen, as it makes practically no difference. But with another constellation, these facts have to be taken into consideration. For example, when using more than one IO-device, the connection to the IO-controller would be the point to choose.

The next question is how to connect the measuring PC with Wireshark to this line. The first approach would be to replace the switch close to the IO-controller with a hub. A hub copies all incoming frames to all ports and therefore permits capturing all frames in- and outgoing to the IO-controller. But the implementation of the PROFINET IO communication stack does not allow this method. It verifies that the connected Ethernet line really uses full-duplex and 100MBit/s. If this is not the case, it refuses to go online. A hub permits only half-duplex connections and that is why this setup does not work.

A trick is to use a "sandwich" setup. In this case two switches with a hub in between can be used. All the PROFINET IO-devices used in the network see the correct line configuration, and the monitoring PC connected to the hub in the middle is able to fetch all frames going from one switch to the other. This setup works fine if there is only low traffic on the network. The switches try to avoid the collisions by storing the frames and the number of collisions on the half duplex connections to the hub should stay low. The most serious problem for this setup normally is to find a 100 MBit/s hub!

Much more comfortable is using a managed switch, which is able to mirror a port. A managed switch may be more expensive than the sandwich solution with two unmanaged switches and a hub. But most switches for industrial Ethernet are managed anyway, so this is probably the better choice.

But a switch with a mirror port can store the frames to avoid collisions on the mirroring port. This storing introduces small timing changes which are undesired if the interest is focussed on measuring the cycle times.

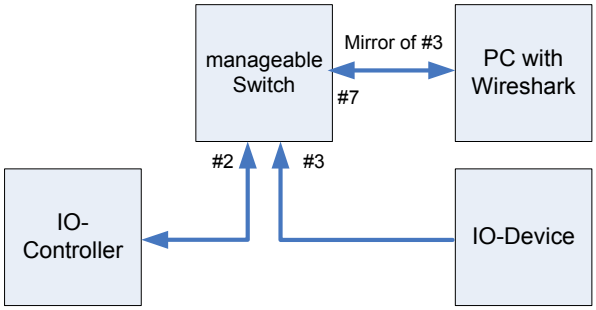

Figure 1: Setup with a Mirror-Port

A similar approach is adding a "Port Aggregator" between the switch and the IO-controller. However, this solution has the same disadvantages as mirroring the port on the managed switch.

The best solution is adding a tap into the Ethernet link. The tap copies all frames on the analysed link to two monitoring links (one for each direction). On short links passive taps work well, but to be sure not to disturb the signal on the link, an active tap is recommended. The disadvantage of this solution in combination with Wireshark is now, that both links have to be captured separately. This can be done on one PC with two Network Interface Controllers (NIC) and two Wireshark instances started. Simply merge the captured files afterwards.

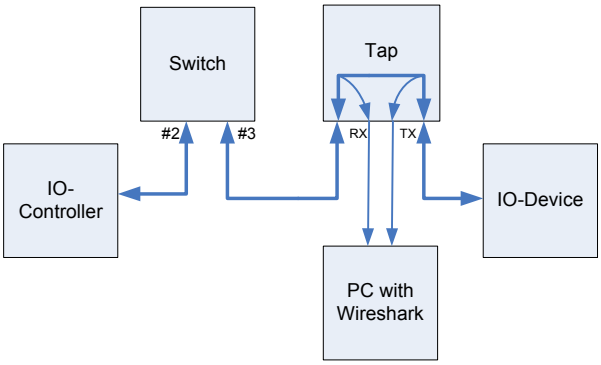

Figure 2: Setup with a Tap

All these setups are fine. But the time stamp on the Ethernet frame is made by the driver software on the measurement PC. Is this time stamping precise enough to measure cycle times of PROFINET IO?

Different professional Ethernet analysers with built-in taps are available on the market. According to the data sheets, they allow measurements with a resolution in the range of nanoseconds. As a reference for our measurements, we used the High Resolution Timing Analyser (HRTA) [1]. This device does not store the

frames internally, but it enables the user to make simple statistical evaluations about the timing on an Ethernet connection. The results of this device allow us to evaluate the source of the jitter in the cycle time.

4. Measurements

We set up a simple system with just an IO-controller, an IO-device and a switch. The examined cycle times were 1, 4, 16 and 64ms, each measured in both directions (input and output data). At 1, 4 and 16ms the test amount was 10'000 packets per direction, at 64ms it was 5'000 packets.

The PROFINET IO-packets used here are typically 60 bytes long (including MAC-header). One IO-device at 1ms cycle time requires 0.5 – 1% of load on a 100MBit/s full-duplex link.

4.1. Setup

First of all, the measurements were done with the HRTA, represented in all figures by the red, drawn through graph.

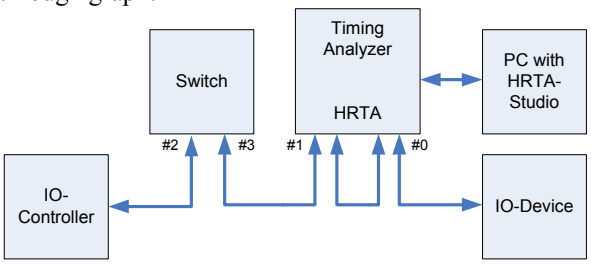

Figure 3: Setup with the HRTA

To compare this reference data with results from Wireshark, probability plots with a normal scale were used. Unlike a linear scale this helps to detect outliers.

The tests with Wireshark on Windows were conducted on following configuration:

CPU:	Intel Pentium 4 HT, 3GHz
RAM:	1GB
Chipset:	82915G
VGA:	integrated in Chipset
HD:	Maxtor 6Y080MO (80GB)
NIC1:	Broadcom NetExtreme Gbit (Measurements)
NIC2:	3Com 3C905TX (LAN-connection)
OS:	MS Windows XP SP1
Wireshark:	V. 0.10.14 (V. 0.99.0 showed no differences)

The tests with Wireshark on Linux were conducted on following configuration:

CPU:	Intel Pentium M 1.6GHz
RAM:	768MB
Chipset:	82855PM
VGA:	ATI Mobility FireGL 9000
HD:	Hitachi HTS726060M (60GB)
NIC:	Intel PRO/1000 MT Mobile GBit (Measurements)
OS:	Ubuntu Linux 6.06 LTS
Linux-Kernel:	2.6.15.6
GNOME:	2.14
X.org:	7.0
Wireshark:	V. 0.99.0

4.2. Comparing the HRTA with Wireshark

Figure 4 shows a comparison of all tested methods with the lowest possible cycle time of 1ms. The graph of the HRTA shows clear steps, no outliers and a correct averaged cycle time of 1ms. Looking at the graphs from Wireshark on Windows (Mirror-Port and

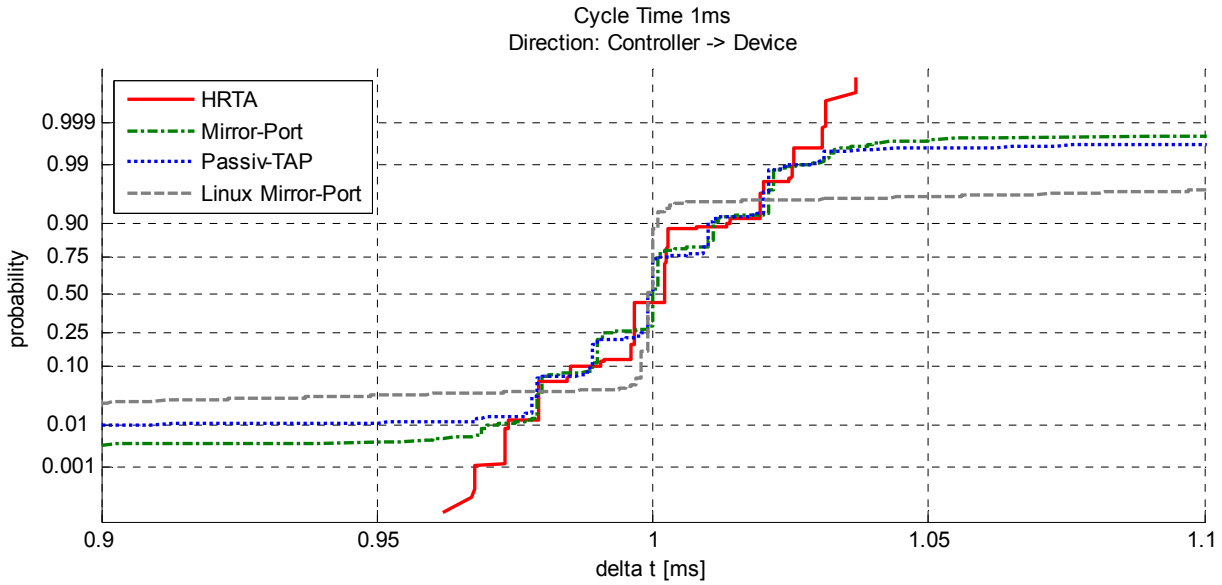

Figure 4: Comparison of all tested methods

Passive-Tap), the steps are less clear and there are outliers, but the average is correct nonetheless. With Wireshark on Linux only the averaged cycle time is correct, the steps are not comparable to the other graphs.

With higher cycle times, the results of Wireshark on Windows are better; the percentage of the outliers is getting smaller. Figure 5 and 6 show the same setup with a cycle time of 16ms. There are still some outliers, even though it is now less than one per mille. So, the quality of measurements with Wireshark is dependent from the load of the PROFINET-system.

The reasons for this have to be the FIFO-buffers on the NIC. With more load, the software becomes more vulnerable to overload-situations, which results in packets being buffered before they receive a time stamp. The HRTA with its time stamping unit in hardware does not suffer from this problem.

4.3. Measuring with different topologies

Figure 5 shows the comparison of the measured cycle times with the HRTA and with Wireshark (Windows) using a Mirror-Port of the connected switch. Figure 6 is the equivalent measurement done with a Passive-Tap in place of the Mirror-Port. The used setups are shown in Figure 1 and 2

The comparison of the Mirror-Port with the Passive-Tap shows, that in contrast to the expectations, the

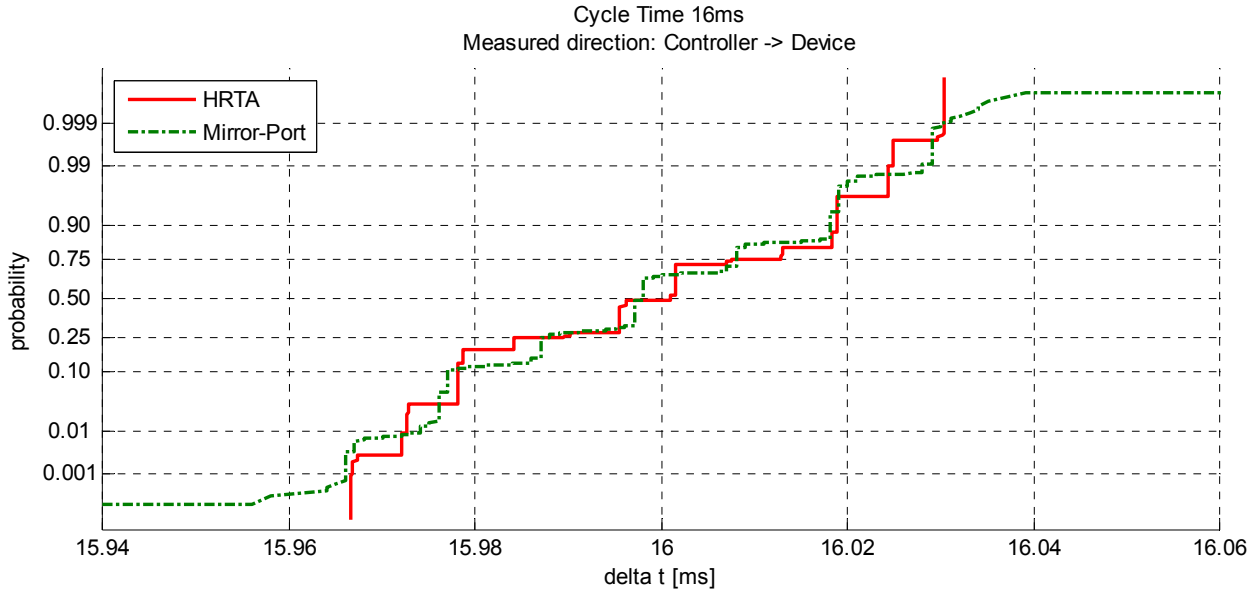

Figure 5: Comparison HRTA <> Windows Mirror-Port

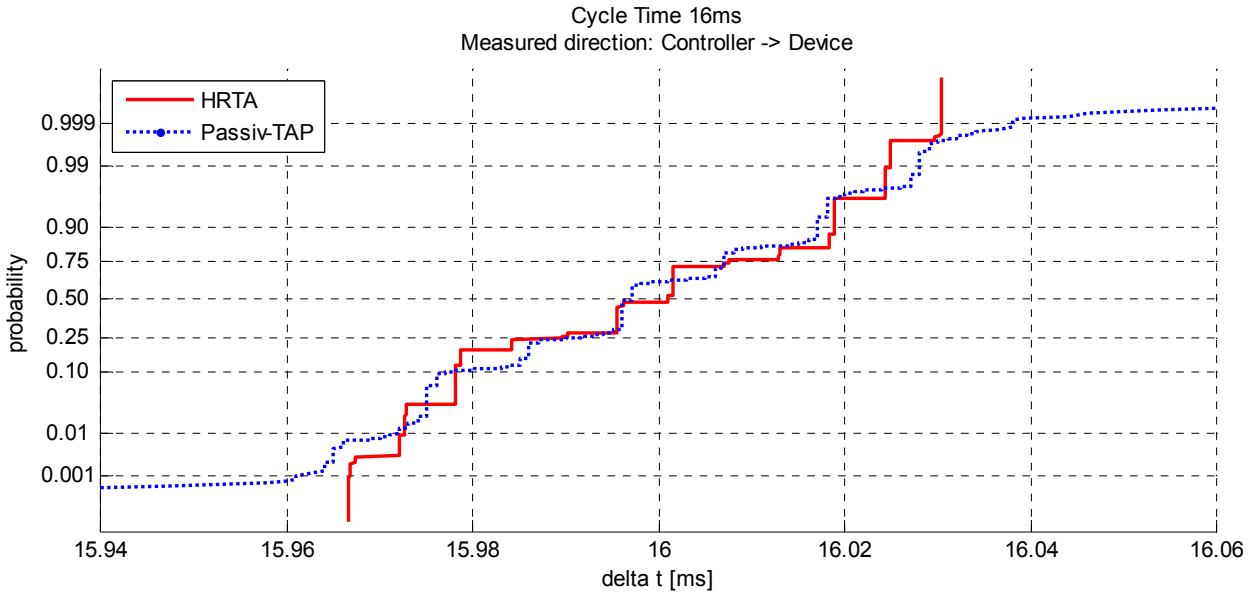

Figure 6: Comparison HRTA <> Windows Passive-Tap

results of the Passive-Tap are not better. In this example they are even a little bit worse, but this is likely to be based on the measurement inaccuracy.

4.4. Influence of the operating system

Figure 7 and 8 show the comparison of the measured cycle times using a Mirror-Port and Wireshark on Windows with the same measurement done on Linux. Both figures show the reference data from the HRTA too.

Clearly visible, the results of the setup using Linux are much worse than they are on Windows. As with 1ms cycle time, only the averaged value is correct. The distribution is influenced too much from the components of the operating system. Even with a cycle time of 64ms, the differences between the graphs did not get noticeably smaller.

However, these results can not be transferred on all measuring methods using Linux. With an optimized configuration it is likely to score better results, but this involves widespread tests.

5. Conclusion

Measuring real-time Ethernet without special hardware is not that simple. When using short cycle times, the risk of overloading the measuring system is too high. Results under such conditions are only of use,

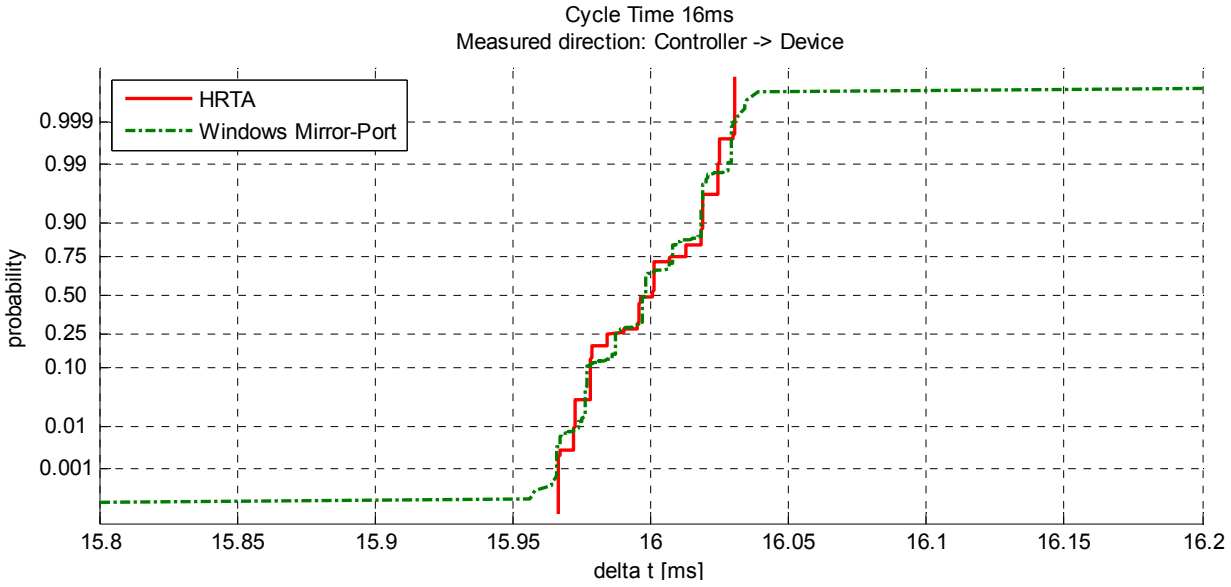

Figure 7: Comparison HRTA <> Windows Mirror-Port

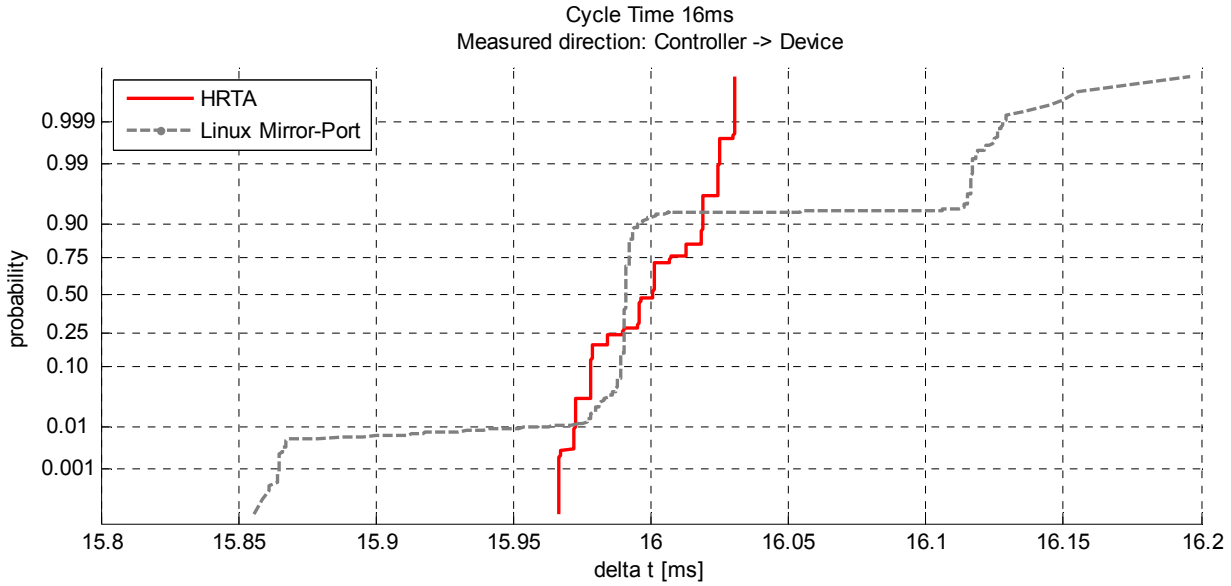

Figure 8: Comparison HRTA <> Linux Mirror-Port

when the tests are focused on averaged cycle times, not on single values. E.g. it is not suited to detect outliers caused by the tested devices.

With longer cycle times, results were getting better using Windows XP. However, the uncertainty stays, as every shortage of resources can influence the measurement. This means, every measured cycle time out of bound is possibly caused by the software, not by the tested device.

The use of a custom-built operating system could be a solution for this problem. But, as shown, a generic Linux-installation, optimized for desktop-use, did not help but rather changed the results to the worse. More work and knowledge is needed to configure a complex open source operating system for a custom application like timing measurements.

Comparing to this, the influence of the used measuring topology turned out to be of less importance – at least with only one IO-device and therefore a low load on the network link.

References

[1] HRTA: Device developed by the Institute of Embedded Systems of the University of Applied Sciences in Winterthur, more information available at http://www.ines.zhwin.ch/

[2] Felser, M.: Real-time Ethernet - industry prospective, Proceedings of the IEEE, Volume 93, Issue 6, June 2005, Page(s): 1118 - 1129

[3] Ferrari, P.; Flammini, A.; Marioli, D.; Taroni, A.: Experimental evaluation of PROFINET performance, 2004 IEEE International Workshop on Factory Communication Systems, Proceedings, 22-24 Sept. 2004, Page(s): 331 – 334

[4] Ferrari, P.; Flammini, A.; Vitturi, S.: Response Times Evaluation of PROFINET Networks, Industrial Electronics, ISIE 2005, Proceedings of the IEEE International Symposium on Volume 4, June 20-23 2005, Page(s): 1371 – 1376

[5] Decotignie, J.-D.: Ethernet-based real-time and industrial communications, Proceedings of the IEEE Volume 93, Issue 6, June 2005, Page(s): 1102 – 1117

[6] Poschraann, A.; Neumann, P.: Architecture and model of PROFINET IO, AFRICON 2004, Page(s): 1213 - 1218 Vol.2

[7] Feld, J.: PROFINET - scalable factory communication for all applications, 2004 IEEE International Workshop on Factory Communication Systems, Proceedings, 22-24 Sept. 2004, Page(s): 33 – 38

[8] Cena, G.; Cibrario, B.I.; Valenzano, A.: Inexpensive tools for measuring Ethernet performance, 2nd IEEE International Conference on Industrial Informatics, INDIN '04, 24-26 June 2004, Page(s): 303 - 308