Optimising a PROFINET IRT Architecture for Maximisation of Non Real Time Traffic

Itin, Lukas. Doran, Hans Dermot.

Institute of Embedded Systems, ZHAW
Technikum Strasse 9
8401 Winterthur
itin@zhaw.ch
donn@zhaw.ch

Abstract: In this paper we examine architectures for non-real time (NRT) traffic handling in PROFINET nodes. Much work has gone into optimising communication controllers for real time traffic resulting in several implementations capable of handling cycle times down to and lower than 250us, in some cases down to 31.25us. Industry experience shows that if the system experiences problems large quantities of data must be shifted between nodes whilst isochronous fidelity is upheld. In particular architectures this can be problematic and we examine architectures typical of those on today's market, suggest improvements and back these up with measurement data.

1 Introduction

The focus on Real Time Ethernet architectures in the early days of the technology was achieving the smallest possible cycle time. PROFINET was no different and much research effort was expended in achieving optimal methods and architectures to post-hoc integrate techniques to achieve cycle times of 31.25us and under ([Ja07], [Gu10]). Whilst industry-available implementations of communication controllers capable of sustaining these cycle times are limited to that offered by Enclustra [En14A] it has become clear through discussions with industry representatives that the requirement to use as much as possible of the theoretically available non-real-time bandwidth is of great importance to design engineers. One of the first real-world applications to exert pressure on the available non-real-time bandwidth are control and monitoring systems. In the control state low cycle times are often required to uphold the system function but if an error occurs in the system, monitoring software must download large quantities of data for storage and later analysis whilst at the same time upholding real-time fidelity and guaranteeing the passage of safety critical communications. This is especially prevalent in the control and monitoring of investment-intensive equipment. An example shown below is a wind-turbine system (Figure 1) where control is based around two cycle times of 10ms and 125us and which requires a high degree of transmission time fidelity (example: Figure 2).

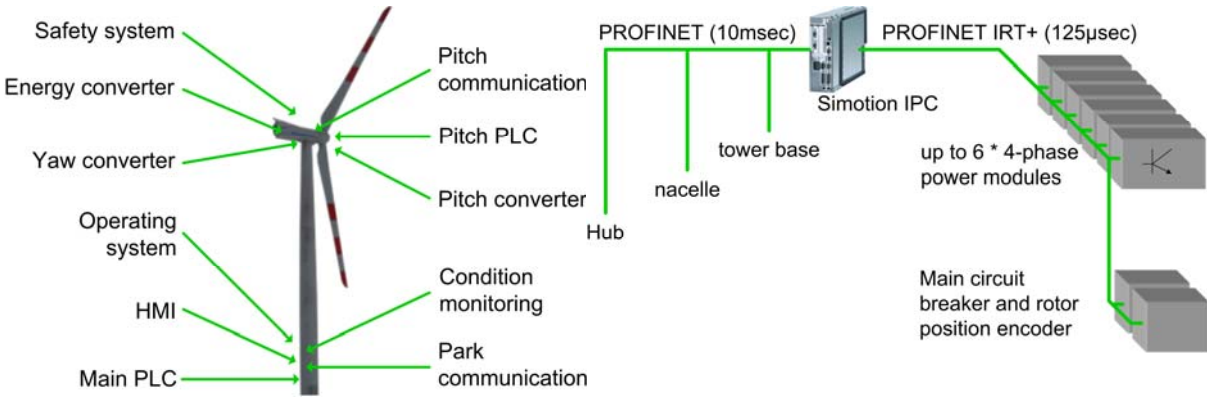

Figure 1: Wind turbine controlled using a PROFINET IRT enabled Simotion. The short cycle times are necessary to control power generation and the synchronicity of power injection into the transmission lines whereas the slow RT cycle times of 10 ms suffice for the control of the rest of the system. Should issues arise, the system state and historical data need to be transferred to a central server for archiving whilst the control functions are upheld.

In cases of system problems, larger quantities of data need to be transferred but whilst there is communication bandwidth available to transport the data, most commercially available systems are unable to either sink or source the mass of incoming data without breaking the control connection.

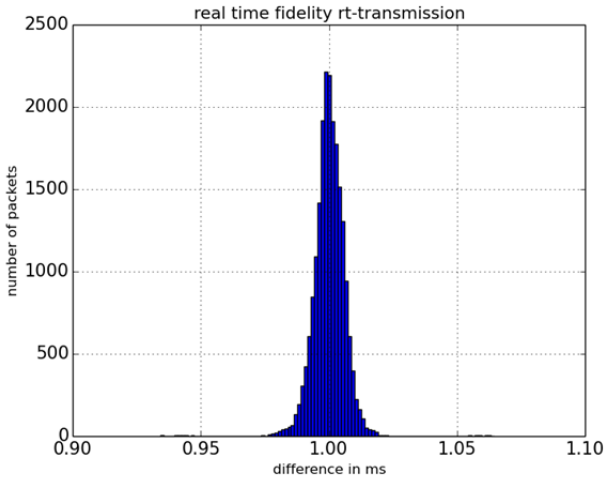

Figure 2: Real-Time Frame Generation Fidelity. This graphic describes the distribution of actual transmission times for a set nominal cycle (transmission) time of 1ms. It was measured using the easyIRT architecture described below, also used in the system of Figure 1.

This paper looks at this issue. Post this introduction we look at the typical architecture of a communication controller node and point out where the communication bottlenecks within the typical architectures are. We show how these can be refactored to allow an optimal, under the circumstances, usage of CPU time. We also point out that using state of the art architectures splitting communication between protocol processors and application processors is not only feasible but also results in higher performance and back this assertion up with measurement data. We end by drawing suitable conclusions and proposals for further work.

2 Communication Controller Architectures

Typical distributed nodes consist of a communication hardware attached to a microcontroller upon which run an operating system (OS) with integrated IP-stack, a PROFINET stack and an application. Such architectures, using a 32-bit processors running at speeds of 100MHz can, if carefully optimised, handle cycle times of 1ms. Typically an OS will run with a time-slice of 500us or 1ms and will still be able to uphold the fidelity of the cycle. We can implement a typical configuration of many PROFINET ASICS and FPGA based communication controllers ([Hm14], [Si14]) with the easyIRT platform [En14A] using a single receive buffer capable of holding multiple frames. Our experimental setup uses a Xilinx Zynq SoC platform with software running on an ARM A9 core at 666 MHz, eCOS as an operating system and the Molex [Mo14] PROFINET RT stack. A block-diagram is shown below (Figure 3).

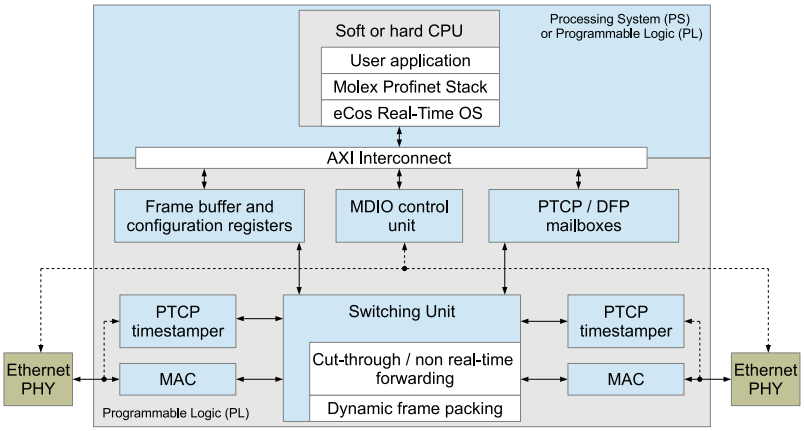

Figure 3: Experimental Setup Configured as a Typically-Available PROFINET node

In a single-buffer system, frame processing is sequential and the limit of real-time fidelity is given by the total amount of directed traffic to the communication node. A first optimisation is to separate RT and NRT traffic on buffer level. This allows NRT traffic handling at a priority lower than the handling of RT traffic. Nevertheless a PROFINET communication relationship is established an upheld using NRT traffic which is

normally separated from application NRT traffic by the PROFINET stack. This is described in the sequence diagram below (Figure 4) which illustrates the path a NRT frame destined for a socket opened by an application must take. In this case our application is a simple loopback application that opens a socket, waits for a frame and retransmits it on reception. For these measurements we use minimum sized Ethernet frames, i.e. a payload of 46 bytes.

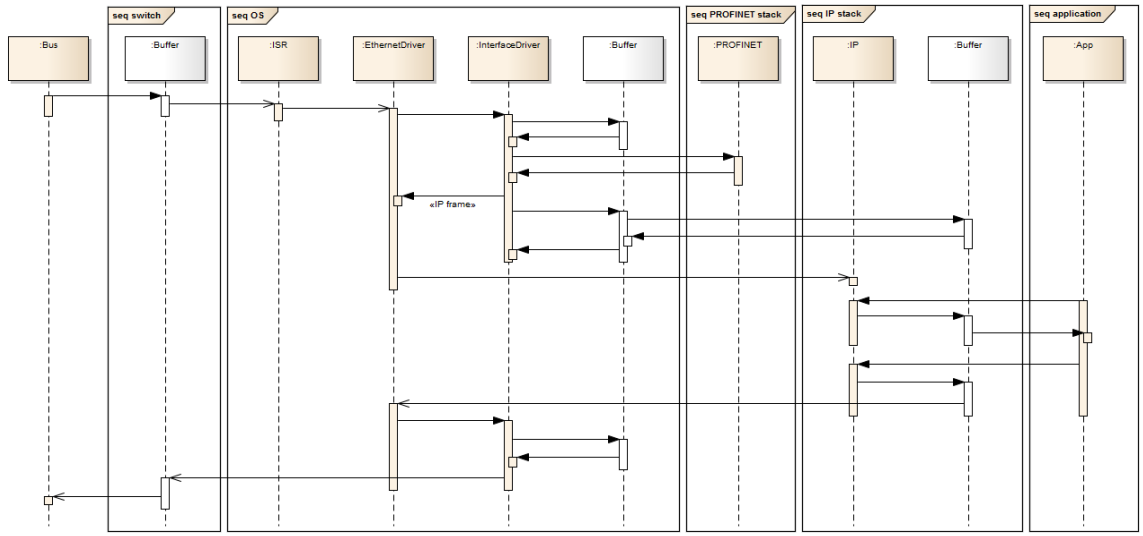

Figure 4: Sequence Diagram of the Processing of a NRT Frame by PROFINET Stack and IP Stack. In the domain of the OS the ISR calls a driver which calls a Deferred Service Routine (DSR) which handles initial frame processing by calling the PROFINET Stack and then the IP stack. The frame is copied twice.

The loopback times were measured using a Hilscher NANL-B500E-RE netAnalyzer with timestamp precision of 10ns and converted into architecture specific frame processing rate, which are shown below (Figure 5). The histogram on the left is the sink/source bandwidth time for minimum sized frames and that on the right for maximum sized frames (payload 1500 bytes). The loopback time minus the time-on-wire for the frame (because the timestamp on the frame occurs on the SFD of the incoming frame) are therefore the retention times of the architecture and hence a measure of the maximum sink/source rates the architecture can sustain for sequential processing of frames. The sink/source bandwidth can be expressed as:

$$2 * frame\_size * 8 \,/\tau_{loopback} - \tau_{wire-time}$$

Where $frame\_size$ is in bytes and can represent either the entire frames size, including overhead or only the payload. For comparability with other research results we take the entire frame size. $wire - time$ represents the time the frame is on the wire and includes the Inter Frame Gap (IFG). The factor 2 is necessary as the loopback time takes Rx and Tx into account and we assume as a first approximation that the Tx and Rx paths suffer the same delays. The measured figures indicate an average sink/source bandwidth of ~2.94Mbit/s, far below what the communication channel can actually carry. Surprisingly these values seem to be the industry norm and compare very favourably with some industry products as the experiments with the Siemens ET200S appear to show [Be13]. Repeating these experiments with maximum size frames indicates a sink/source rate of ~25.36 Mbit/s.

Whilst one could optimise the implementation of OS, IP stack, PROFINET stack and application this is not without potentially substantial costs associated with unknown outcomes. There is also the consideration that in single-processor systems where the application itself has real-time constraints that need to be met, many design engineers have enough issues handling these constraints without integrating a PROFINET stack into the system. This has led to the rise of split architectures where communication is mapped to one processor and application to the second. This is the use case of most commercial ASIC/FPGA PROFINET solutions ([Hm14], So14]). However well this works for RT communications the application processor still has to open a socket via some interface on the communication processor, with all ancillary costs and in effect the NRT architecture and processing times are the same/comparable to those shown in Figures 4 and 5, hence possible NRT bandwidth is similarly constrained.

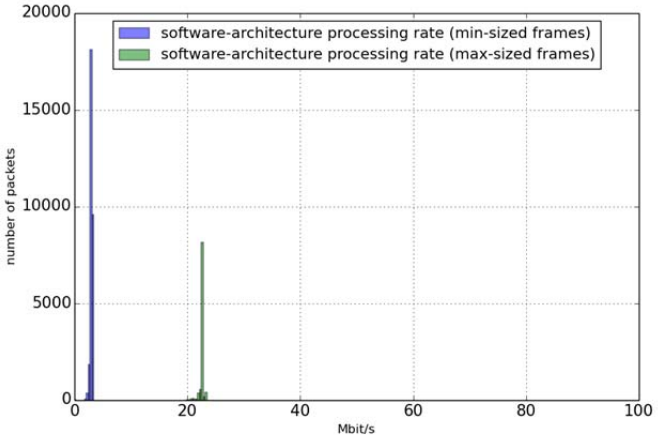

Figure 5: Source and Sink Rates for Minimum and Maximum Sized NRT Ethernet Frames by an Industry Typical PROFINET node

One potential solution is to split communication between the PROFINET communication controller and an application controller. We achieve this by setting filters that distinguish between PROFINET NRT communication and other NRT communication. A block diagram is shown in Figure 6.

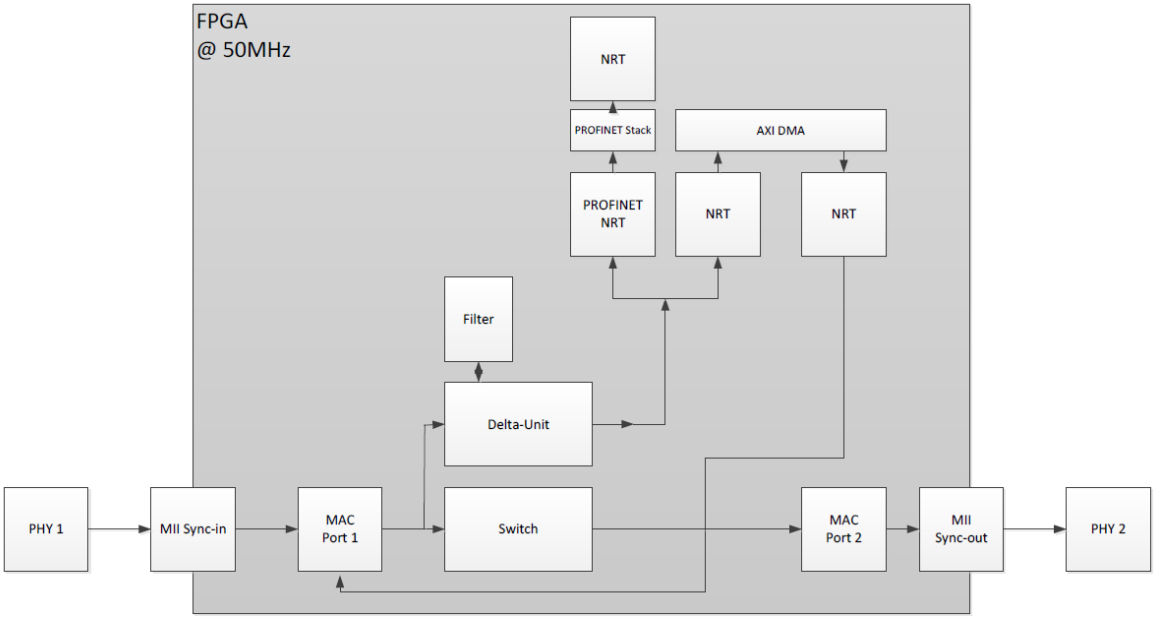

Figure 6: IRT Node-Architecture Splitting PROFINET-NRT and non-PROFINET-NRT Traffic

In this first architecture, a NRT-frame enters the FPGA via the PHY1 and MAC and the first bytes are buffered in the delta unit. Filters working on the delta unit trigger the further forwarding of the NRT frame either to the PROFINET NRT buffer or a holding buffer for the AXI-bus interface. We expect the application processor, not shown in the above picture, to interface with this AXI-bus and collect any stored frame(s) via its own IP stack. In our tests we loop the received frames back using an AXI-DMA entity and so measure the possible NRT sink-source rate. In this architecture the application NRT traffic is thus no longer dependent on the forwarding characteristics of a communication controller optimised for real time traffic. Using this architecture we can present non-PROFINET NRT frames to an application processor at the rate of entry – the sink/source rate is dependent on the performance of the application processor and software. We test the filter-set using frame sequences from a PLC to ensure that PROFINET communications are not compromised.

3 Conclusions

We have shown how industry standard devices which we presume to have been optimised for RT traffic show less encouraging characteristics in the handling of NRT traffic and we have been able to reproduce these architectures in the laboratory. We have proposed an alternative architecture and show that the first results show significant promise. The architecture we have implemented implies two IP stacks acting on the same MAC address and there are a few issues that need to be resolved. One easily identifiable issue is the (bidirectional) handling of ARP requests/responses and the actual setting of the IP addresses. First experiments indicate that the delegation of the IP address handling can be carried out by the communication controller since in PROFINET Siemens PLC's tend to set the IP addresses using DCP. The IP address needs to be made available to the IP stack of the application processor via some interface, which would mean that the application processor cannot communicate before the PLC has set the device IP address. An alternative would be to implement a global register handling the IP address and apply data coherence management allowing each IP stack to set/reset the IP address. In a similar vein a first experiment with the PROFINET communication controller handling ARP requests/responses has shown promise.
For future work we would propose investigating the interface of an IP stack implemented in hardware. These exist [En14B] and may enable implementation of a more elegant architecture especially in cases where there is no second application processor but the processing time of the path as shown in Figure 4 above needs to be optimised.

4 References

[Be13] Belic, F.; Martinovic, G., "Model of influence of MRP on network performances," Computers and Communications (ISCC), 2013 IEEE Symposium on , vol., no., pp.000874,000879, 7-10 July 2013

[En14A] http://www.enclustra.com/en/products/ip-cores/profinet-irt-ip-core/ Last accessed 20.10.2014

[En14B] http://www.enclustra.com/en/products/ip-cores/udp-ip-ethernet/ Last accessed 20.10.2014

[Gu10] Gunzinger, D.; Kuenzle, C.; Schwarz, A.; Doran, H.D.; Weber, K., "Optimising PROFINET IRT for fast cycle times: A proof of concept," Factory Communication Systems (WFCS), 2010 8th IEEE International Workshop on, vol., no., pp.35,42, 18-21 May 2010

[Hm14] http://www.anybus.de/products/abcc_np40.shtml Last accessed 20.10.2014

[Js07] Jasperneite, J.; Schumacher, M.; Weber, K., "Limits of increasing the performance of Industrial Ethernet protocols," Emerging Technologies and Factory Automation, 2007. ETFA. IEEE Conference on, vol., no., pp.17,24, 25-28 Sept. 2007

[Mo14] http://www.deutsch.molex.com/molex/products/family?key=profinet_solutions&channel=products&chanName=family&pageTitle=Introduction Last accessed 20.10.2014

[Si14] http://www.automation.siemens.com/salesmaterial-as/brochure/de/datasheet_ertec200p_de.pdf Last accessed 20.10.2014

[So14] http://industrial.softing.com/uploads/softing_downloads/IE001D_201110_Industrial_Ethernet_Geraete_Firmware_Altera_FPGA-1323083039.pdf Last accessed 20.10.2014.