ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA091

A RELIABLE WHITE RABBIT NETWORK FOR THE FAIR GENERAL TIMING MACHINE

C. Prados[1], A. Hahn, J. Bai, GSI, Darmstadt Germany
A. Suresh, Hochschule Darmstadt, Darmstadt, Germany
[1] also at Technical University Darmstadt, Darmstadt, Germany

Abstract

A new timing system based on White Rabbit (WR) is being developed for the upcoming FAIR facility at GSI in collaboration with CERN and other partners. The General Timing Machine (GTM) is responsible for the synchronization of nodes and distribution of timing events, which allows the real-time control of the accelerator equipment. WR is a time-deterministic, low latency Ethernet-based network for general data transfer and sub-ns time and frequency distribution. The FAIR WR network is considered operational only if it provides deterministic and resilient data delivery and reliable time distribution. In order to achieve this level of service, methods and techniques to increase the reliability of the GTM and WR network has been studied and evaluated. Besides, GSI has developed a network monitoring and logging system to measure the performance and detect failures of the WR network. Finally, we describe the continuous integration system at GSI and how it has improve the overall reliability of the GTM.

FAIR GENERAL TIMING MACHINE

The FAIR General Timing Machine (GTM) is responsible for the synchronization of Front End Controllers (FeC) with nanosecond accuracy and distribution of Control Messages (CM) for the hard real-time control of the GSI and FAIR accelerator complex.

The hard real-time control is achieved in several steps. First, the Settings Management [1] distributes the settings of the FeCs over a standard network. Second, the activities in the FeCs are prepared by the Front-End Software FESA [2]. Finally, the GMT generates on-time actions at the FECs thanks to the CM broadcasted by the Data Master (DM) [3]. These CMs are sent over the White Rabbit (WR) [4] network, which is also responsible for the synchronization of the FeCs, DM and WR switches.

The GMT have been designed to scale up to 2000 FeCs and synchronize them in the range of 1 to 5 ns with ps precision. The GSI and FAIR accelerator facilities requires the GMT to work reliably in routine operation 24/7.

The GMT is linked to other systems and must be able to react, within 10 ms, to interlock and external signals [5].

All systems connected to the timing system depend on it's high availability. Distribution of CM must be guaranteed for commissioning and testing even when the accelerator does not produce beam. Therefore the loss rate of CM in WR network cannot go beyond 1 CM per year.

Table 1: FAIR GTM Requirements

Requirements	FAIR GMT
Time Resolution	1 to 5 ns
Precision (Std dev.)	≤10 ps
GMT Reaction Time	≤10 ms
CM Failure Rate	3.17 10^{-12}
CM Loss Rate	1 CM/year
Num FeCs	≤ 2000
Links Distance	1 to 2000 m

BUILDING A RELIABLE GENERAL TIMING MACHINE NETWORK

The Figure 1 depicts the components and topology of the GTM network. The network is established by the interconnection of WR Switches and WR Nodes using fibre optic cables. The GTM network is meant to transport CMs to the FeCs and synchronize the WR Nodes of the FeCs. According to the requirements of the GTM, Table 1, the WR network has to provide and guarantee timing and data delivery even under abnormal operations and conditions. Therefore the reliability of the GTM WR network relies on:

- Ethernet traffic delivery within upper-bond latency.
- Synchronization of the network and FeCs.

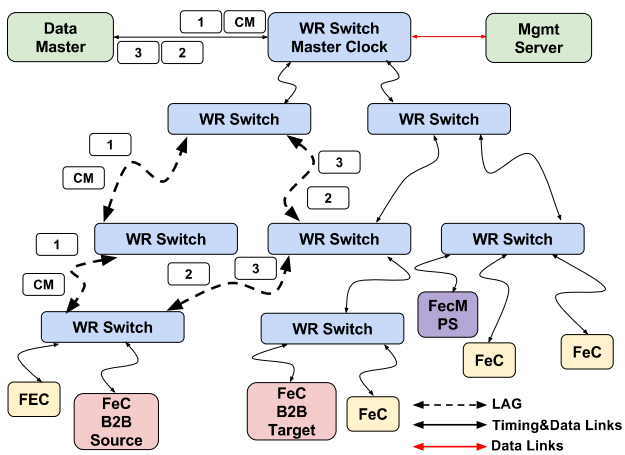

Figure 1: Overview of the FAIR general timing machine.

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA091

Ethernet networks, like WR, don't provide by definition any mechanism to guarantee the level of reliability required by the FAIR GTM. In order to increase the reliability of the networks, companies and standardization groups have developed since years protocols and techniques. The WR Switch has already adopted some of these mechanisms.

In the next sections, we describe methodologies and strategies to achieve reliable data and timing delivery, either adopting existing features of the WR switch or proposing new approaches.

Resilient Data Delivery

The principal source of network traffic in the GMT network is the DM. The DM broadcasts CMs from the top of the network to all FeCs. It is critical for the control of the accelerator complex resilient deliver of CMs to the FeCs over the GMT. Single point of failures (SPOF) in the network or bit errors in the Ethernet frames may be the cause of unsuccessful delivery of data.

The GMT network overcomes SPOFs deploying redundant components: WR switches and links. As the Fig. 1 shows, the tree topology of the GMT network can overcome SPOFs thanks to the redundant links between the WR Switches. Unfortunately, this strategy, redundant links, creates undesired logical loops in the network.

A family of network protocols, the Spanning Tree Protocols, could be used to create spanning tree topologies. These protocols disable those links that doesn't belong to the spanning tree. In the event of link failure, SPOF, the protocol reconfigures the spanning tree and enables the redundant link to solve the SPOF. During the reconfiguration, Ethernet frames may get lost and won't be delivered. An Enhanced versions of Spanning Tree Protocol [6], supporting seamless redundancy, has been proposed and tested in the White Rabbit community.

We propose an alternative to the Spanning Tree Protocols, the Link Aggregation Protocol (LAG) [7]. LAG provides methods to combine multiple links of a network to form a single link. This approach provides redundancy in the network without loops and reconfiguration of the topology. The links compressed in a LAG are always active. In the event of SPOF, the LAG engine is notified and will forward the frames only to the active links. A set of rules defined by the users in the LAG engine, determine the output link, among the aggregate links, for every incoming frame.

Ethernet frames travel over a noise and imperfect channel, the Ethernet network. Bits of the frame may be altered in the channel for different reasons, resulting in flawed or lost frames. The loss rate of CMs required by the GTM is 1 CM per year, 3.17 10^{-12}, expressed in terms of failure rate. The global bit error rate (BER) of the GTM network is 4.10^{-11}, 3.17 10^{-10} in terms of frame error rate (FER) [8]. The GTM BER and FER are calculated taking into account: logical topology of the network, number of FeCs, BER of cabling and traffic bandwidth. The loss rate of frames in the GMT network is ten times bigger than the specified. Classical network protocols, like TCP, achieve resilient and reliable data delivery retransmitting the missing frames. Unfortunately, we can't rely on retransmission since the delivery of the CMs won't be anymore within the specified upper-bound latency. An alternative technique to retransmission, is the Forward Error Correction (FEC). This technique consists in sending, from the DM, the original CMs and redundant data generated by a coding algorithm. This redundant information allows the FeCs to detect and correct a limit number of bit errors in the frame using a decoding algorithm.

The Fig. 2 shows a FEC scheme capable of overcoming the loss of packets in the network and fix bit error in the Ethernet frames. The encoding is done in two steps. First, the CMs are encoded using the linear error correcting called LDPC [9]. The LDPC code generates redundant data using the original CM. The DM sends the original CM along with the redundant information encapsulated in *N* frames. The LDPC are able to retrieve the original CM if at least *M* out *N* frames are delivered. The selection of *N* and *M* depends on the use case and it has to be carefully calculated. In the second step, the LDPC encoded frames, in turn, are encoded using the Golay Code [10]. This code is able to detect bit errors in the Ethernet frames and repair up 3-bit errors.

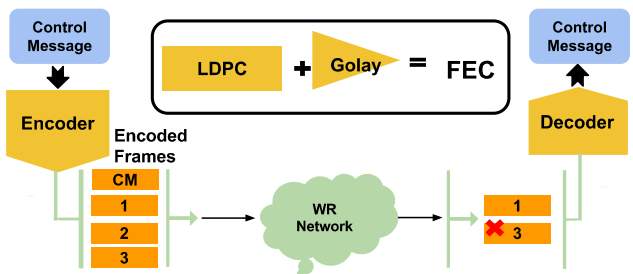

Figure 2: GTM forward error correction scheme.

In conclusion, the LDPC provide resilience against loss of frames in the network and the Golay Code repairs flawed frames. For the FAIR GMT we propose a LDPD with *N=3 and M=2.*

As we have stated before, the LAG is a suitable protocol for seamless redundancy in the event of a SPOF. Yet the LAG can not avoid losses of CMs if a network device fails (e.g. fibre optic) and CMs were in transit. The LAG uses a set of rules, defined by the users, to forward the frames among the aggregated links. These rules can be defined in such a way that the LAG forwards the encoded frames in specific links creating a multipath. The encoded frames reach the very same FECs following different paths. The Figure 1 illustrates the multipath strategy. The dashed lines represent a LAG groups, two links per WR switch. The half of the encoded frames follows a complete different path as the other half of the frames. If the LDPC encoder is capable of retrieving the original CM receiving two out four encoded frames, this

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA091

scheme overcomes SPOF in the network without loss frames or CMs.

Deterministic Data Delivery

The GMT network requires data delivery of CMs within 500 µs upper bound latency so as to guarantee the reaction of the GMT within 10 ms to interlock and external signals.

The latency of of Ethernet network has a constant and variable component. The latency of the frames throughout the network links (fibre optics or copper) is constant, with a negligible jitter. On the contrary, the switches in a network introduce a variable latency during the switching procedure of the Ethernet frames. The latency of the switches depends on the design of the switch and the amount of traffic in transit. The WR switches has been carefully designed to achieve low latency switching adopting a cut-through architecture: WR switches start forwarding a Ethernet frame to the destination port/s as soon as the frame arrives, even before the whole frame has been received. A dedicated module in the WR switch, RTU Forwarding Engine, decides based on the destination MAC address, to which port/s a frame has to be forwarded [6].

As we have pointed out already, the amount of frames in transit in an Ethernet switch influences the latency. In this regard, the Ethernet standardization group introduced to the protocol a useful scheme to bound the network latencies, quality-of-service (QoS) [11]. The QoS defines eight levels of service expressed in a special 3-bit field of the Ethernet header frame. The switch prioritizes the forwarding of the frames according to the QoS in this field. If the switch can't forward intermediately the incoming frames, it stores the frames in QoS queues according to its level of service.

In GMT, the CM frames from the DM are tagged with the highest QoS, 7th, so as to achieve the lowest latency possible. In FAIR, the DM is not the only source of network traffic in the WR network. Other subsystems of the accelerator like the Bunch to Bucket Transfer System (B2B) [12] or the Machine Protection System (MPS) [13], will probably make use of the GMT network to control their equipment. The last source of network traffic is the Management Server, responsible of the network management protocols like DHCP, SNMP, LLDP etc…

The B2B triggers the beam extraction and injection between synchrotrons at GSI. The B2B issues a handshake of frames between two special FeCs, the B2B Source and B2B Target. The MPS is responsible for the protection of the accelerator machines and safety of personnel. The MPS for FAIR is not fully specified yet. According to the last specifications, a selected group of FeCs in the network send, at regulars intervals, Ethernet frames to a specific FeC, MPS, as a heartbeat signal. The Table 2 and Table 3 summarize the traffic and the data source in the GTM.

The WR switch implements methods, along with the QoS, to guarantee that CMs are forwarded in within an upper-bond latency. The forwarding engine of the WR switch introduces constant latencies for CMs with priory 7th and broadcast destination MAC address. On top of that, the WR switch implements a strict priority (SP) queue algorithm to schedule the forwarding of frames from the QoS queues. The WR switch dispatch first CM frames from the 7th QoS, as long as the queue is not empty. Then the frames in lower QoS queues are scheduled to be forwarded. This scheme guarantees the deterministic delivery of CMs in the GTM with an upper-bond latency of 500 µs in a network with 5 layers of switches [14].

Table 2: GTM WR Network Traffic and Priorities

Application	Traffic Bandwith	Prio	VLAN ID
DM Broadcast	100 Mbit/s	7	100
DM Unicast	10 Mbit/s	7	100
B2B	25 kbit/s	6	100, 200
MPS	900kbit/s	5	100, 300
Mgmt Traffic	~10 Mbit/s	4-0	400

The B2B and MPS network traffic require also upper-bond latency. In order to measure the latency for the traffic of these two systems, we have intensively tested the WR Switch. The test set-up simulates the FAIR use case in terms of layer of switches and traffic in the GTM network. The Table 2 and Table 3 specifies the bandwidth, VLAN Id and upper bond latency of every traffic. The tests [14] were carried out with the Xena Traffic [15] generator and we have followed the testing methodology for GbE switches, RFC 2889 [16]. The results show latencies beyond the specified requirements for the B2B and MPS, Tabe 3. There are peaks of latency going beyond the upper-bound and even, at times, loss of frames, presumably by queue overrun. The SP queue algorithm, in the case of B2B MPS traffic, doesn't fulfil the requirements. We are in the process of simulating and testing alternative queue algorithms like Priority-Based Deficit Weighted Round Robin.

Table 3: GTM WR Network Traffic Bandwidth

Application	Max Latency/ Layer of Switches	Payload
DM	≤500 us / ≤ 5	200-500 bytes
B2B	≤10 ms / 3 – 5	100 bytes
MPS	≤ 1ms / 3	64 bytes
Mgmt Traffic	Best effort	64 – 1518 bytes

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA091

Reliable Timing Delivery

WR extends the protocol IEEE 1588 [17], for achieving sub-nanosecond synchronization with picoseconds jitter. WR characterizes the asymmetries of the link and measures the phase shift between the master and slave clocks [18]. The continuous synchronization of all the FeCs during operation is critical for the control of FAIR and GSI accelerators. The GTM production network [19], made of 8 WR switches and 20 FeCs, exhibits high level of reliability. More than three months in operation and no degradation or lost of synchronization has been detected.

The protocol IEEE 1588 supports different topologies of redundant networks, although the quality of the synchronization can be affected during the reconfiguration of the network in the event of SPOFs. The GTM requires continuous phase alignment and synchronization of all the FeCs without lost of quality. In the Phd thesis [6], the author studies and characterize the problem and proposes a scheme for seamless reconfiguration in redundant WR network preserving subns accuracy of synchronisation.

MONITORING AND LOGGING THE GENERAL TIMING MACHINE

The stated requirements of the FAIR CTS are not only achieved implementing protocols and network techniques, but also supervising the performance of the system during operation. Monitoring and logging systems provide the means to Timing Network Managers (TNM) to detect, diagnose and prevent failures and malfunctions of the system on real-time and analyse the performance of the system offline.

The performance of the system is evaluated using the status information of the GTM network and its evolution and changes. This information can be classify in two levels: events and alarms. The events represent changes in the status of devices of the GTM network (e.g. FeC synchronized) or change of the status of network peers (e.g. new PTP Clock Master). Alarms are triggered when the devices reach specific status (e.g. Whiter Rabbit port down).

In a distributed system like the timing network, the monitoring and logging information are generated in the components and is propagated passively or actively to a central point creating a global status of the GTM network.

Architecture

The Figure 3 describes the architecture of the Monitoring and Logging System at GSI. The source of status information are the FeCs and WR Switches. The Monitoring and Logging Server (MLS) gathers the status of the network and with the help of the frameworks Icinga[20] and LogStash[21]. A secondary network, the Management network, interconnects the management interfaces of all devices and transports the monitoring information. The WR Network, besides timing and control events, transports logging information and alarms. Finally, a set of protocols: SNMP, Etherbone [23] and Syslog, stablish the communication and transport the information from the devices to the MLS.

Monitoring

A monitoring systems basically observe and checks the state of a system and notifies events. The system at FAIR, uses the Icinga Framework to gather, display and digest the monitoring information. The Icinga framework is programmed to issue periodically SNMP requests to read out the information from the WR Switches. In the case of the FeCs, the host CPU reads out the status of the WR Nodes over the host bus. All this information is sent periodically from the CPU to the Icinga server using NSCA Passive Check Daemon [22]. At GSI, we have chosen this strategy over other alternatives because we want to keep the monitoring network traffic out of GTM Network. The GTM network will interconnect up to 2.000 FeCs. The simultaneous generation of monitoring information upstream, from these FeCs to the MLS, could bring about severe network congestion. The TNM can configure Icinga to display this information in different levels and create alarms if a status of the network reaches of a defined value.

Logging

A logging system records the events and status of the timing network. This information can be used either for triggering real-time alarms or off-line analysis of the information. Both devices, WR switches and FeCs, generate logging information from the OS of the host system and transmitted over the Management Network. Alternatively, the WR TR generates only events related to WR and are sent over GTM network, the amount of logging traffic is almost negligible. GSI uses the framework LogStash [21] to collect, parse and transform the log information.

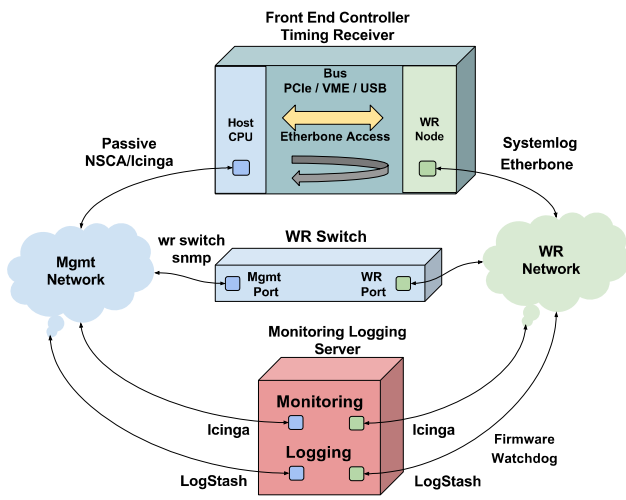

Figure 3: Monitoring and logging architecture.

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA091

System-wide Firmware Watchdog

In the GTM network the maintenance and support of the WR Switches and FeCs is distributed among the TNM and the users, respectively. Both parties try their best to maintain their systems up-to-date with specific compatible versions of the firmware. Otherwise, the timing and data distribution can not be guarantee. During the last years, at GSI, we have faced several updates of the GTM and we have learnt a lesson. We can't guarantee that all the components of GTM network are up-to-date. There were always situations where some of FeCs couldn't be reached and updated during the maintenance period. In order to improve the update procedure and guarantee that all the FeCs in the WR network are up-to-date, we have implemented a system-wide firmware watchdog to verify that all the FeCs are running compatible firmware. Using the architecture and infrastructure of the Monitoring and Logging System, we have developed an application that verifies the firmware version of the the FeCs connected to the timing network. If the version of the firmware is not compatible, the firmware watchdog disables the port of the WR Switch where the FeCs is connected to. By doing this, we prevent potential operations problems of the FeCs and in the WR network. In addition, the application communicates the user the event and instruct him to update the firmware. The status and actions of the firmware watchdog is logged in the MLS.

DEVELOPMENT AND CONTINUOUS INTEGRATION OF THE GENERAL TIMING MACHINE

The GTM is a highly complex development made up of a large number of sub-components, protocols and technologies. As example, the Fig. 4 shows the architecture and some sub-components of a FeC. The sub-components of the FeCs are being developed by different groups at GSI, collaborations with other institutes and Open Software communities. The majority of these developments are still in constant evolution and change. New releases come to life asynchronously fixing bugs and providing new required features to the system. On the other hand, new releases of the sub-components may introduce new bugs or blunders. In addition, collaborative shared developments are really beneficial, but may be difficult to modify and upgrade without breaking other functionality. A small and harmless change, done by one developer in one isolate component of the system, may expose or bring about performance problems or failures in other parts of the system.

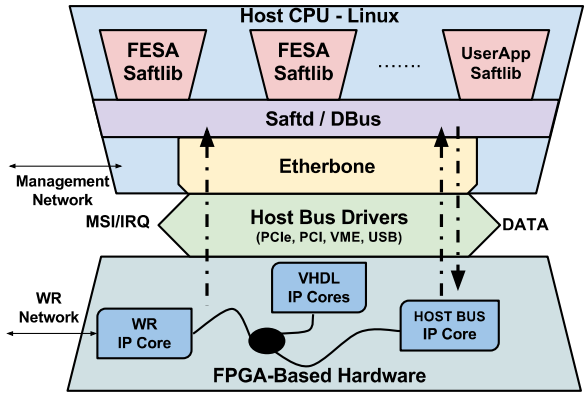

Figure 4: FeCs architecture and sub-components.

The constant and mingled evolution of the developments, along with the high complexity of the FeCs, makes the integration of the sub-components a challenging and at times, frustrating task. The time invested in debugging and fixing new corner cases increases considerably. On top of that, the FeCs, at GSI, are made up of FPGA-based hardware in different form-factors, with specific firmware and drivers, which multiplies the amount of components to update, test and integrate. Classical development flows are useful in situations where the development of sub-components is stable and the requirements and functionalities are clearly defined. As we stated above, this is not the case in the development of the CTS and FeCs. In order to delivery a reliable system under these circumstances and reduce the testing and debugging effort, we have adopted a development flow known continuous integration (CI).

The Figure 5 depicts how we have integrated, at GSI, along with our classical development flow, a CI, where the updates of the sub-components are automatically built, deployed and tested without the intervention of the developers. The architecture and benefits of our CI system are detailed below.

Continuous Integration System Architecture

Our continuous integration system is made of two sub-systems. The building server, and a testing environment.

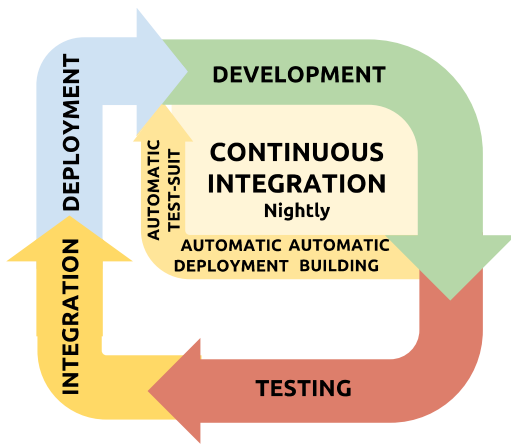

Figure 5: Continuous integration flow.

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA091

In the building server we have set up a development environment with all the tools and compilers required for the building. We use the popular Jenkies [24] to automatically download and build the code of the timing firmware stack. The code of the timing firmware stack is stored and managed in a Git repository, Bel Projects [25]. The timing firmware is a complex stack of VHDL code, kernel modules and user spaces tools. We use a hierarchical structure of makefiles to steer the building process and resolve dependencies among the projects.

Jenkins starts and logs, every night, the building of the stack, and notify per email the developers in case of failures. After building, Jenkins initiate the deployment of firmware on the testing environment, which recreates a minimal timing network. Once the deployment of the timing firmware has concluded successfully (otherwise Jenkins will notify the developers) the tests start automatically. We have achieve a moderate level of automation in the interpretation of the test's results. The most remarkable is the notification, to the developers, of the degradation of the synchronization during the tests.

CONCLUSION

Thanks to the WR techonology the GTM at GSI deliveries CMs well below the GSI's latency requirement and synchronizes steadily a mid-size production network. Yet there are still open issues regarding the reliability of the systems: data and timing delivery in the event of SPOF, resiliant transmission of data and scheduling of non-CMs frames. As we have presented, there are already solid technical proposal to improve the in-built reliability of the GTM.

In addition to the in-built reliability, the monitoring and logging of the GTM is critical for the reliability of system in production. Finally, we recomend to use continous integration practics to reduce the time of integration and testing of complex system like GTM. Our continous integration system helps us to maintain thet quality and reliablity of the system under development.

We would like to ackondlege and thank the great work done by the WR community, the Experimental Electronics and Beam Diagnostic departments at GSI.

REFERENCES

[1] J. Fitzek *et al.*, "Settings Management within the FAIR Control System Based on the CERN LSA Framework", *Proceedings of PCaPAC'10*, Saskatoon, Canada, 2010, WEPL008.

[2] Al. Schwinn *et al.*, "FESA3-The New Front-End Software Framework at CERN and the FAIR Facility", *Proceedings of PCaPAC'10*, Saskatoon, Canada, 2010, WECOAA03.

[3] M. Kreider *et al.*, "New developments on the FAIR Timing Master", *Proceedings of PCaPAC'14*, Karlsruhe, Germany, 2014, FPO022.

[4] J. Serrano, P. Alvarez, M. Cattin, E. G. Cota *et al.*, "The White Rabbit Project", ICALEPCS'09, Kobe, Japan, 2009.

[5] J. Fitzek *et al.*, F-DS-C-08e, FAIR Detailed Specification Interlock System.

[6] M. Lipinski, "Methods to Increase Reliability and Ensure Determinism in a White Rabbit Network", PhD Thesis, Warsaw University of Technology, 2016.

[7] IEEE 802.3ad Link Aggregation

[8] C.Prados and M.Lipinski, "White Rabbit and Robustness", http://www.ohwr.org/documents/103

[9] RFC 5170 "Low Density Parity Check (LDP) Staircase and Triangle Forward Error Correction (FEC) Schemes"

[10] S. B. Wicker, Error Control Systems for Digital Communication and Storage. Prentice Hall, 1995.

[11] IEEE Std 802.1Q Standard for Local and metropolitan area networks Bridges and Bridged Networks.

[12] J. Bai, "Development of the Timing System for the Bunch-to-Bucket Transfer between the FAIR Accelerators", PhD Thesis, Frankfurt University, 2017.

[13] F-DS-C-31e, FAIR Detailed Specification "Machine protection systems".

[14] RFC 2889 Benchmarking Methodology for LAN Switching Devices.

[15] Xena Traffic Generator, https://xenanetworks.com

[16] C. Prados and J. Bai. "Testing the WR Network of the FAIR General Machine".

[17] IEEE 1588-2008. IEEE Standard for Precision Clock Synchronization Protocol for Networked Measurment and Control Systems.

[18] T. Włostowski. Precise time and frequency transfer in a White Rabbit network. Master's thesis, Warsaw University of Technology, Warsaw, Poland, May 2011.

[19] M. Kreider *et al.*,"Launching the Fair Timing System with CRYRING", *Proceedings of PCaPAC'14*, Karlsruhe, Germany.

[20] Icinga, https://www.icinga.com/

[21] Logstash, https://www.elastic.co/guide/en/logstash

[22] NSCA, https://docs.nsclient.org/howto/nsca

[23] M. Kreider *et al.*, "Etherbone - A Network Layer for the Wishbone SoC BusS" *Proc. of ICALEPCS 2011*, Grenoble, France, 2011.

[24] Jenkins, https://jenkins.io

[25] Bel Projects, https://github.com/GSI-CSCO/bel_projects