

MANAGING YOUR TIMING SYSTEM AS A STANDARD ETHERNET NETWORK

A. A. Wujek*, G. Daniluk, M. M. Lipinski, CERN, Geneva, Switzerland
A. Rubini, GNUDD, Pavia, Italy

Abstract

White Rabbit (WR) is an extension of Ethernet which allows deterministic data delivery and remote synchronization of nodes with accuracies below 1 nanosecond and jitter better than 10 ps. Because WR is Ethernet, a WR-based timing system can benefit from all standard network protocols and tools available in the Ethernet ecosystem. This paper describes the configuration, monitoring and diagnostics of a WR network using standard tools. Using the Simple Network Management Protocol (SNMP), clients can easily monitor with standard monitoring tools like Nagios, Icinga and Grafana e.g. the quality of the data link and synchronization. The former involves e.g. the number of dropped frames; The latter concerns parameters such as the latency of frame distribution and fibre delay compensation. The Link Layer Discovery Protocol (LLDP) allows discovery of the actual topology of a network. Wireshark and PTP Track Hound can intercept and help with analysis of the content of WR frames of live traffic. In order to benefit from time-proven, scalable, standard monitoring solutions, some development was needed in the WR switch and nodes. The paper describes these developments and shows many examples of the benefits brought about by this strategy.

INTRODUCTION

Timing networks are custom networks, with limited bandwidth and use of custom technologies. This limits possibilities of using many standard network tools and protocols.

For example, the General Machine Timing (GMT) used at CERN is based on uni-directional 500 kb/s RS422 links, and allows operators and users to synchronize different processes in CERN's accelerator network. The system has a number of shortcomings though, among which the most important are the limited bandwidth and the impossibility of dynamically evaluating the delay induced by the data links. White Rabbit is foreseen as a successor technology for GMT.

THE WHITE RABBIT NETWORK

The White Rabbit network [1] [2] achieves sub-nanosecond timing accuracy by implementing both layer 1 synchronization and an extension of The IEEE 1588-2008 (Precision Time Protocol, PTP) standard [3]. Layer 1 synchronization enables reference frequency distribution among devices in a WR network. On the other hand, IEEE 1588-2008 is a packet-based-protocol that performs the time synchronization among the WR nodes. The fact that data transmission

is independent of clock transmission can reduce the cost of infrastructure needed for monitoring of end-nodes.

WR Network Architecture

Figure 1 shows the layout of a typical WR network. Data-wise it is a standard Ethernet switched network, i.e. there is no hierarchy. Any node can talk to any other node. Regarding synchronization, there is a hierarchy established by the fact that switches have downlink and uplink ports. A switch uses its downlink ports to connect to uplink ports of other switches and discipline their time. The uppermost switch (Grand Master) in the hierarchy receives its notion of time through external TTL Pulse Per Second (PPS) and 10 MHz inputs, along with a time code to initialize its internal International Atomic Time (TAI) counter.

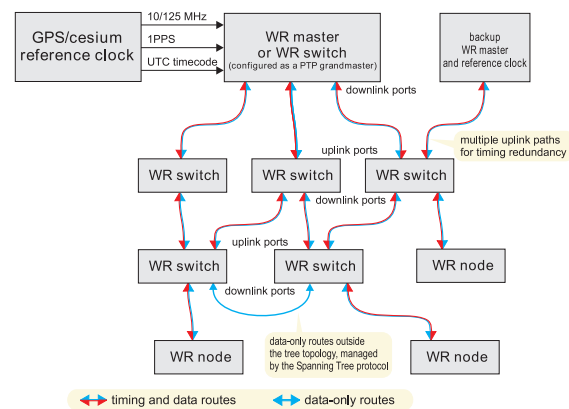


Figure 1: Layout of a typical WR network.

The WR Switch [4] is the main component of WR networks. It is in the form of a standard 19" chassis with 18 GbE ports and one management port in the front panel.

A typical WR Node [5] is a device with one GbE link, two voltage controlled oscillators and an FPGA that contains the WR PTP Core (WRPC) [6] together with application specific IP cores.

CONFIGURATION OF A WR NETWORK

For small networks it is convenient to configure and monitor switches and nodes manually, by using ssh or the web interface for switches and a direct console connection for nodes. However, such an approach does not scale well for bigger networks.

White Rabbit Switch Configuration

A White Rabbit Switch applies the configuration at boot time. The configuration file is in the format of Kconfig, the same format used by the Linux kernel for configuration at

* adam.wujek@cern.ch

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2017). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI.

build time [7]. The configuration file in the Kconfig format can be stored locally on a switch or downloaded at boot time. The remote location can be pre-configured or can be sent to a switch in the DHCP response together with an assigned IP address. The switch tries to download a configuration file from a given location (via e.g. TFTP) and applies it (Fig. 2). If the downloaded config file contains errors, it is reported via SNMP and the last, locally stored, configuration file is used instead.

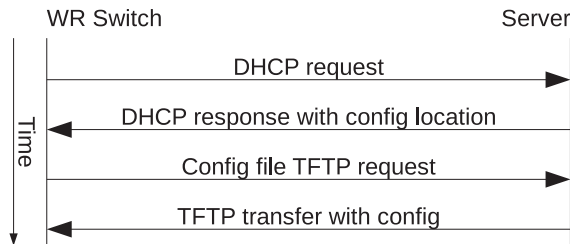


Figure 2: Simplified flow showing how the White Rabbit Switch retrieves its configuration.

When a WR switch breaks, it can be replaced on-site by a unit taken directly from the stock without any pre-configuration. Only the MAC address of a switch’s management port has to be updated in the DHCP database to be able to get the same IP, hostname and configuration file. Such an approach can significantly reduce down time and does not require in-depth WR knowledge to fix the network.

Controls Configuration Data Editor (CCDE)

To simplify the management of configuration files, a new web-based tool, the Controls Configuration Data Editor (CCDE), was developed (Fig. 3) at CERN.

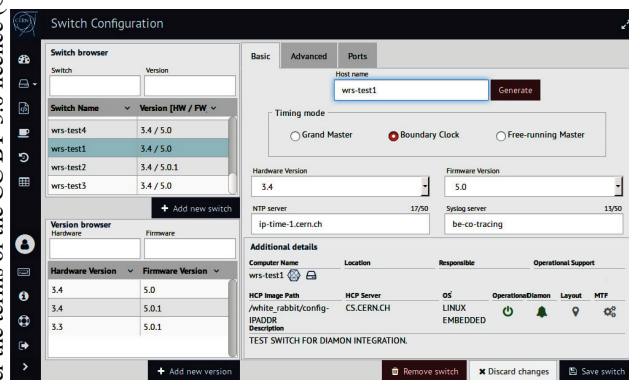


Figure 3: White Rabbit Switch configuration in CCDE.

A new switch can be installed following these simple steps:

- Add a new switch to the CCDE
 - Pick the hostname of the switch
 - Select Hardware and Software versions
 - Optionally change the default configuration
- Click *Generate* in CCDE to generate a config file
- Power-on or restart the switch to trigger retrieval of the new configuration file

Now the White Rabbit Switch is configured and ready to operate.

Node Configuration

Contrary to a WR Switch’s configuration, configuration on a WR node is done at build time. The main reason for such design decision is lack of resources (especially memory) on nodes. Unfortunately, not all parameters are known at build time. For example SFP¹ calibration values are not known. These parameters are needed to achieve sub-ns synchronization accuracy and are stored in an external flash memory on each WR node. They can be written directly to the flash through JTAG or by logging in via a serial console to a soft CPU implemented in the FPGA and manually setting the SFP values. These two possibilities do not scale well for bigger installations. To address this issue, an SNMP agent was implemented, which makes possible to configure SFP calibration values by sending SNMP packets in the WR network to a given node. Such an architecture also makes it possible to design a stand-alone node with only one WR link connected, reducing the cost of cabling and additional machines to host WR node cards.

MONITORING AND DIAGNOSTICS PROTOCOLS USED IN WR NETWORKS

Timing networks, due to their specific nature, usually require new protocols to be defined and implemented. Since White Rabbit is based on Ethernet, all standard protocols of the IP family can be used. No new protocols were designed neither to monitor or to diagnose White Rabbit networks.

Simple Network Management Protocol (SNMP)

The Simple Network Management Protocol (SNMP) [8] is a standard and widely used protocol for monitoring networks. It defines two types of roles: SNMP manager, which sends requests, and SNMP agent, which responds to the manager with the requested information.

Managers and agents exchange data as objects, which are organized into a hierarchical tree. The objects are defined in Management Information Base files (MIBs), and addressed by object ID (OID) in the form of dot-separated natural numbers. For example, the OID for the uptime of a system is 1.3.6.1.2.1.1.3, which thanks to MIBs can be translated into a more readable form SNMPv2-MIB::sysUpTime.

OIDs implemented in the WR Switch are organized in two groups:

- *General status objects* – are intended for operators. They provide a summary about the status of a switch and several subsystems (like timing, networking, Operating System). Each piece of information is exported as a separate object (Fig. 4). These should be used by control system operators and users without a comprehensive knowledge of White Rabbit internals. These exports provide a general status of the device and high

¹ Small Form-factor Pluggable transceiver

level errors, which is enough in most cases to perform a quick repair.

- *Expert objects* – can be used by White Rabbit experts for the in-depth diagnosis of switch failures. These values are verbose and normally should not be used by operators. This group contains various parameters including the quality of the data link (like the number of dropped frames) and synchronization (like the latency of frame distribution and fibre delay compensation).

The SNMP implementation on the WR PTP core supports only a subset of expert objects due to size constraints.

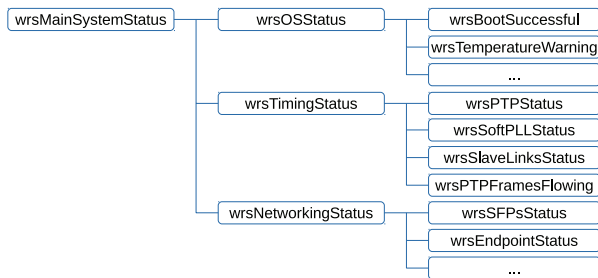


Figure 4: The structure of general status objects for operators exported via SNMP.

The White Rabbit Switch uses Net-SNMP [9], a standard, widely used implementation of an SNMP agent. The same solution could not fit into the White Rabbit PTP Core due to the tight memory constraints. Instead a custom SNMP agent with a small footprint was implemented.

Syslog

Both the WR Switch and WR nodes support sending syslog messages. The switch uses a standard syslog [10] implementation and is capable of sending messages generated by the kernel and user space applications. Additionally, when any of the SNMP general statuses is not in the “OK” state, a syslog message is sent with more details.

WRPC’s custom implementation of a syslog is very limited and sends messages only on critical events like loss of synchronization or link going up.

Link Layer Discovery Protocol (LLDP)

The Link Layer Discovery Protocol (LLDP) is a protocol defined in standard IEEE 802.1ab [11]. LLDP can be used to discover neighbors connected directly to a particular device. LLDP is implemented as a link layer protocol, it is not propagated via layer 2 network devices (like switches, bridges). All the data sent in LLDP frames is in the form of Type Length Value (TLV) fields. LLDP defines a set of obligatory TLVs (chassis ID, port ID and time to live) and a set of optional TLVs (including: port description, system name, system description, system capabilities, management, VLAN info). Every valid LLDP frame has to contain all obligatory and any number of optional TLVs.

An LLDP-capable device can work in two modes: receive and transmit. These two modes can be enabled or disabled independently. Because there is neither a request-response

mechanism nor an acknowledgement embedded into the protocol, all transmit capable devices have to send LLDP frames periodically.

The WR Switch supports both receiving and transmitting of LLDP frames in a WR network. Due to memory constraints, the WR node’s implementation of LLDP is limited only to transmit mode.

All data gathered by LLDP can be exported by a standard SNMP daemon. Thanks to this it is possible to discover the full topology of a WR network. This graph (Fig. 5) was generated by LLDP-SNMP software [12].

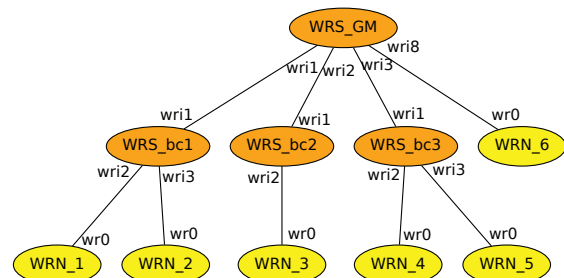


Figure 5: Network layout based on information gathered via LLDP. WR Switches are represented as orange ellipses, WR nodes as yellow ellipses. Labels next to nodes represent network interfaces used by particular links.

To generate such a graph, the root node of a graph/network has to be identified; the natural choice of root node in a WR network is the Grand Master switch. Starting from the root switch LLDP-SNMP software recursively queries all switches via SNMP for their neighbors discovered via LLDP. If a neighbor is a switch it is also queried. All queries go through the management port of the WR Switch. Since the traffic on the management port is separate from the WR network, no traffic is generated in the WR network.

After all devices in a WR network are discovered, a representation of the network is stored in JSON format. Then LLDP-SNMP translates the JSON representation into a DOT language [13] file and using *graphviz* [14] generates a network topology graph.

For networks that are intended to be static (like CERN’s) LLDP can be used to compare the configured network topology with the actual state. All differences can be easily tracked and reported to a network administrator.

MONITORING AND DIAGNOSTICS TOOLS FOR WR NETWORKS

Sniffing Traffic in a WR Network

Since a WR network transmits frames like a regular Ethernet network, it is possible to use already existing tools to intercept and analyze traffic in a WR timing network. However, due to the way the PTP protocol, WR switch and user application itself work, there are several possible scenarios of sniffer usage:

- An obvious scenario is to run a sniffer on a WR node connected to a network. It could help to analyze the

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2017). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI.

incoming traffic. Unfortunately, current implementation of a WR node have little support to forward traffic from the WR network to the host system. The WR-NIC project [15] allows that with the exception of WR-PTP traffic itself.

- The simplest scenario is to connect a sniffing machine to a free port in a WR switch. This is useful for intercepting broadcast traffic. In this scenario it is impossible to analyze WR-PTP traffic, since by design it is point-to-point. In the future, WR switches will support a port mirroring functionality, so even unicast traffic will be intercepted using this method.
- Another possibility is a “men in the middle” kind of approach. Inserting a host with two network interfaces between two devices allows to intercept all the traffic running between them. However, due to the way WR-PTP is designed, the synchronization cannot be preserved.
- The best way to intercept WR-PTP traffic is to run a sniffer on a WR switch, then forward its output (live or captured in a file) to a tool capable of analyzing WR traffic (see the Wireshark and PTP Track Hound sections for concrete examples). This method is also not ideal. It allows only to intercept WR-PTP traffic. Any other kind of traffic is not forwarded from the WR Switch ports to the management CPU (ARM), running Linux in the switch.

Unfortunately, none of the methods presented above allows to intercept data and synchronization (WR-PTP) traffic at the same time, but using them separately or in combination can greatly improve debugging abilities.

Wireshark and *PTP Track Hound* are good examples of tools that can be used for sniffing and analyzing WR-PTP network traffic.

Wireshark

“*Wireshark is the world’s foremost and widely-used network protocol analyzer.*” [16] In the first three scenarios described above Wireshark can be run on a host system and be used at the same time as a sniffer and an analyzer. In the latter scenario we recommend running *tcpdump* [17] on a switch (it is available there already), then redirect *tcpdump*’s output over an ssh tunnel to Wireshark for further analysis². Alternatively, *tcpdump*’s output can be stored locally on a switch first, then transferred to a host machine for post analysis.

There is an ongoing development by an author of this paper of a dissector to extend Wireshark’s abilities to be able to understand the WR-PTP protocol. Wireshark’s dissection of an announce message is presented in Fig. 6.

² To start *tcpdump* on a WR Switch and redirect its output to local a Wireshark instance, run the following command on a host with a connection to the management port of the WR switch (not WR network):

```
$ ssh root@<switch_IP> /usr/sbin/tcpdump -U -s0 -i <sw_port> -w - | wireshark -k -i -
```

Where <switch_IP> is the IP address of the WR switch management port; <sw_port> is the port whose traffic is to be analyzed.

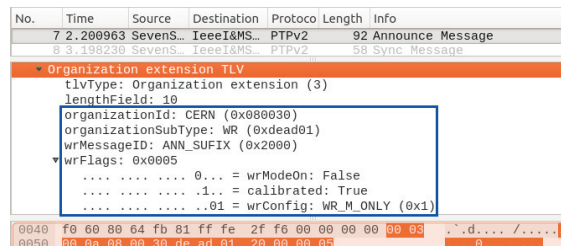


Figure 6: Dissected WR announce frame in Wireshark.

Recent versions of Wireshark are able to visualize flow between hosts with a *Flow Graph*. An example of message exchange between two WR switches to get synchronization is presented in Fig. 7. Please note that the WR PTP implementation does not use unicast addresses for communication, but a multicast address (00:1B:19:00:00:00) as destination.

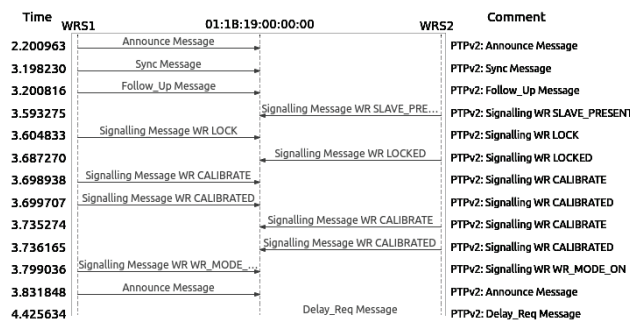


Figure 7: Flow of WR frames in Wireshark.

A similar dissector to the one mentioned above can be written for any application-specific protocol used in a WR network.

PTP Track Hound

PTP Track Hound is another tool [18] that can be used to sniff and analyze WR traffic [19] (Fig. 8). Similar to Wireshark, this tool can sniff and analyze local traffic or use *tcpdump*’s output from a remote machine to analyze it on a local host live or as post analysis.

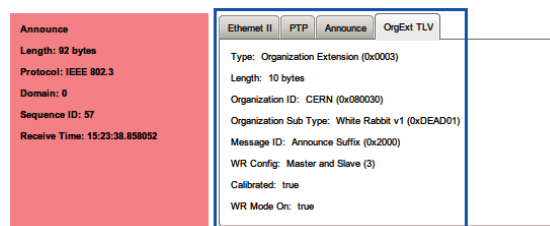


Figure 8: Dissected WR announce frame in PTP Track Hound.

Nagios/Icinga

Nagios [20] and its fork Icinga [21] are applications that monitor systems, networks and infrastructure. A sample screen-shot of Nagios with the General status objects of a WR switch is presented in Fig. 9. When an error occurs,

the corresponding status box changes color and notification like sending e-mail or SMS can be issued. Nagios also has different views showing an overview of multiple devices and their state.

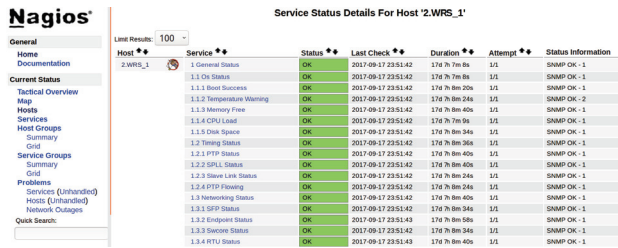


Figure 9: General status objects of a WR switch presented in Nagios.

Grafana

Grafana [22] is a web based general purpose dashboard and graph composer (Fig. 10). It can display data gathered from WR switches and WR nodes via e.g. SNMP.



Figure 10: An example grafana's dashboard.

Fluctuations of the round-trip time (RTT) between two WR switches installed at CERN and visualized in Grafana (Fig. 11) prove that changes between day and night have to be compensated to achieve sub-nanosecond synchronization. The data of the right part of the graph was gathered during lower temperature periods and with less fluctuations compared to the left part.

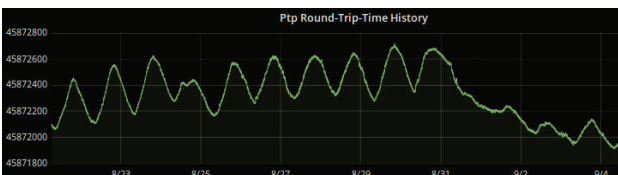


Figure 11: Time series of round-trip time between two WR switches in Grafana

CONCLUSION

Since timing networks based on WR provide data and synchronization distribution at the same time using Ethernet technology, timing networks can benefit from many already existing tools for management, monitoring and debugging. Using standard protocols like SNMP and LLDP can reduce

the amount of required new development and reduce the risk of vendor lock-in for monitoring and management software. Introducing new users to WR network technology is much easier compared to custom networks as it is based on already existing and well known tools.

As these tools become more and more powerful, WR users benefit from even more advanced monitoring and diagnostics capabilities without any additional investment.

REFERENCES

- [1] J. Serrano, P. Alvarez, M. Cattin, E. G. Cota et al., "The White Rabbit Project," in ICALEPCS, Kobe, Japan, 2009.
- [2] White Rabbit project website, <http://cern.ch/white-rabbit>
- [3] 1588-2008 - IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems
- [4] The White Rabbit Switch, <https://www.ohwr.org/projects/wr-switch-hw>
- [5] The White Rabbit Node, <https://www.ohwr.org/projects/white-rabbit/wiki/node>
- [6] White Rabbit PTP Core, <https://www.ohwr.org/projects/wr-cores/wiki/wrpc-core>
- [7] Kconfig language specification <https://www.kernel.org/doc/Documentation/kbuild/kconfig-language.txt>
- [8] Introduction to version 2 of the Internet-standard Network Management Framework, RFC 1441 and others <https://tools.ietf.org/html/rfc1441>
- [9] Net-SNMP project homepage, <http://www.net-snmp.org>
- [10] rsyslog project, <http://www.rsyslog.com>
- [11] IEEE Standard Local and Metropolitan Area Networks. Station and Media Access Control Connectivity Discovery, in IEEE Std 802.1AB-2005
- [12] SNMP LLDP project, <https://github.com/stanislabv/snmp-lldp>
- [13] Specification of a DOT Language, <http://www.graphviz.org/content/dot-language>
- [14] Graphviz - Graph Visualization Software, <http://www.graphviz.org>
- [15] WR-NIC project's homepage, <https://www.ohwr.org/projects/wr-nic>
- [16] Wireshark project homepage, <https://www.wireshark.org>
- [17] tcpdump's homepage, <http://www.tcpdump.org>
- [18] PTP Track Hound's homepage, <https://www.ptptrackhound.com>
- [19] PTP Track Hound 1.0.1 release notes, <https://www.ptptrackhound.com/release-notes.html>
- [20] Nagios's homepage, <https://nagios.org>
- [21] Icinga's homepage, <https://www.icinga.com>
- [22] Grafana's homepage, <https://grafana.com>