European Organization for Nuclear Research
Organisation Europeenne pour la Recherche Nucleaire

White Rabbit Protocol (WRP)

Preliminary specification
Version: **20090112**

Tomasz Włostowski
CERN AB-CO-HT
Tel : +41 (0)22 76 79262
Bat : 864-1-A07

Geneve, January 14, 2009

Contents

1 Changelog

2009/01/12

- converted whole thing to LaTeX
- reordered chapters
- changed WRP message protocol name to WRCMP for clarity

2008/11/24

- added CRC32 header checksum in HP frame
- added `prev_segment_crc` field in SP header
- changed HP header format (`hp_flags`)
- burst mode operation for HP
- added flags for HP collision handling
- explained SP frame reassembly
- `WRCMP_HEARTBEAT` and `WRCMP_REQUEST_REPORT` messages added
- small changes in use cases
- HP collision handling explained
- bugfixes

2008/10/22

- bugfixes (interframe gap, minimum payload)
- explained addressing /routing of HP frames
- full PTP compatibility for non-WR devices
- described network behaviour for non-WR compatible nodes

2008/10/17 Added link failure and alternate uplink paths support
2008/10/15 Initial release

2 Abbreviations

SP	Standard Priority frame
HP	High Priority frame
non-HP	Any frame type other than HP frame (e.g. standard Ethernet traffic and SP traffic)
WRP	White Rabbit Protocol
WRCMP	White Rabbit Control Message Protocol
PTPv2	Precision Time Protocol version 2
LT encoding	Luby-transform coding
CRC	Cyclic redundancy check
WR switch	Switch which supports White Rabbit protocol

3 Introduction

White Rabbit (WRP) protocol is a low-level timing and control transport protocol for White Rabbit network devices. The tasks of WRP are:

- providing reliable one-way transmission channels (without handshaking) for low-latency control and timing messages
- precise delay measurement and reporting, fine (sub-nanosecond scale) transparent time transmission based on PTPv2 protocol and synchronous Ethernet (frequency reference is embedded into Ethernet carrier).
- identification of WR-compliant devices in the network

WRP operates on 2nd (data link - MAC) network layer to make it simple and easy to implement in hardware. WRP is software-transparent protocol.

4 Network structure/organization

White Rabbit network consists of following active components:

- System timing master ? provides time and frequency reference for the whole network and sends all control and timing messages
- Switches capable of routing high priority (HP) WRP frames with integrated PTPv2 boundary clock functionality
- Slave devices (nodes) such as timing receivers, WorldFIP/Powerlink bridges, standard twisted-pair Ethernet bridges, etc.
- Network cabling - single-mode fiber (for backbone WR network or high precision timing receivers) or twisted-pair (for devices where ultrafine timing is not required)

5 White Rabbit frame types

White Rabbit protocol uses two different frame types:

- **standard traffic (SP)** - standard Ethernet frames (*Ethertype* `0xa0a1`). For this kind of traffic, switches operate in store-then-transmit mode (like standard Ethernet gear). In case of collision, data is buffered. This type of traffic is non-deterministic. There are no restrictions on usage of SP frames. SP frames can be fragmented by White Rabbit switches when switch needs to route incoming HP frame. WR compatible devices shall be able to reconstruct fragmented SP frames. Compatibility with WR frame fragmentation and reconstruction can be achieved on purely software way, without hardware modification (e.g. using standard Ethernet NICs/switches)
- **high-priority traffic (HP)** - Ethernet frames with special unique value of *Ethertype* field (`0xa0a0`). These frames have absolute priority over SP frames to maintain low and deterministic transmission delay.

Other types of frames, called non-WR frames (not used by WRP protocol - with ethertypes different than `0xa0a0` or `0xa0a1`) are treated by switches like SP frames. They can be fragmented when HP frame arrives and continued by sending proper SP frame(s). Frame structures are depicted on fig. 1.

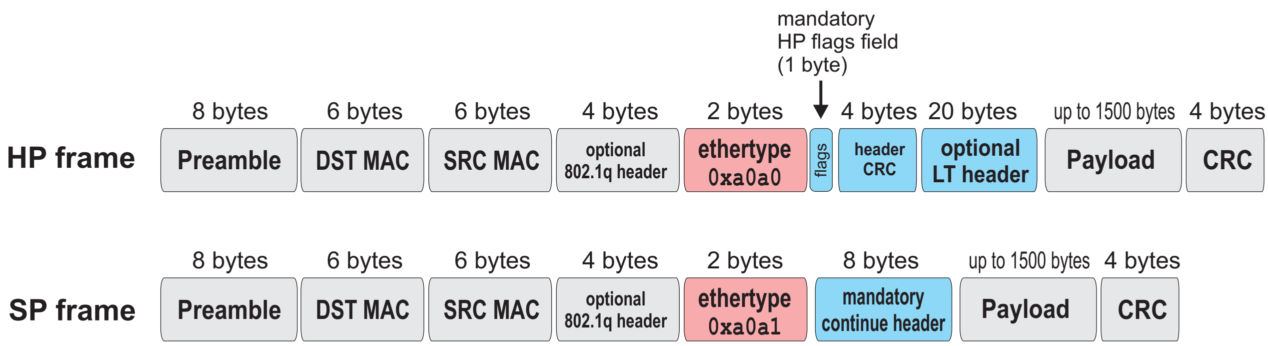

Figure 1: White Rabbit Ethernet frames structure

5.1 HP frames

HP frames are frames for time-critical control data. They are routed with as low latency as possible, forcing fragmentation of non-HP traffic if required. Structure of such frame is shown on listing 1:

Listing 1: C-like structure of WRP HP frame

```
__attribute__ ((packed)) struct WR_HP_frame {

  struct eth_header {                // standard Ethernet frame header
    uint8_t preamble[8];             // ethernet preamble (SFD, 6 x 0xaa + 0xab)
    uint8_t dst_mac[6];              // destination MAC.
    uint8_t src_mac[6];              // source MAC
    uint16_t ethertype = 0xa0a0;     // Ethertype
  };

  uint8_t hp_flags;                  // HP frame flags
```

```
  uint32_t header_crc;             // CRC32 of frame header and hp_flags

  struct lt_header {               // optional LT-coding header
    uint32_t orig_frame_id;        // unique original frame id
    uint32_t eqn;                  // equation coefficients
    uint16_t segment_size;         // size of LT segment in bytes
    uint16_t segment_num;          // number of current segment
    uint8_t md5sum[8];             // MD5 frame checksum (LT header + LT payload)
  };

  uint8_t payload[32..1500];       // frame payload
  uint32_t crc;                    // frame CRC32
};
```

Every HP frame contains two additional fields after standard Ethernet frame header:

- 1-byte `hp_flags` field, containing frame flags (see table 5.1),
- 32-bit CRC of data received so far, allowing for early detection of header corruption during HP frame routing. If corrupted header is detected frame is immediately dropped.

To maintain compatibility with ethernet gear not supporting WRP, HP frames are no different from Ethernet standard frames. WR switches recognize them **only** by unique value of *Ethertype* field. HP frames may contain optional LT encoding header, when `LT_ENABLED` flag is set, otherwise payload data begins right after header CRC. LT coding is described in detail in section **??**.

HP frames are physically broadcast through whole network (all switches are receiving them), although they reach only nodes to which they are assigned via destination MAC address (either a single node or multicast group). This is to prevent overflowing FIFOs in switches. While HP frame is being received by switch, it should pause routing of whole non-HP traffic.

5.2 SP frames

SP frames are used for non time-critical traffic:

- sending WRP messages
- continuation of transmission of fragmented SP frames or other Ethernet frames.
- PTP and link delay compensation

C-like structure of SP frame is shown on listing 2. Each SP frame has custom (WR-specific) SP header, described in table 5.2.

Listing 2: C-like structure of WRP SP frame

```
__attribute__ ((packed)) struct WR_SP_frame {
  struct eth_header {             // standard Ethernet frame header
    uint8_t preamble[8];          // ethernet preamble (7 x 0xaa + 0xab)
    uint8_t dst_mac[6];           // destination MAC.
    uint8_t src_mac[6];           // source MAC
    uint16_t ethertype = 0xa0a1;  // EtherType
};

  struct SP_mandatory_header {    // mandatory SP header
    uint16_t is_continue;
    uint16_t continue_offset;
    uint32_t prev_segment_crc;
};

  uint8_t payload[42..1500];      // frame payload
```

Bit	Flag name	Description
0 (LSB)	LT_ENABLED	1 means that frame contains LT-encoded payload and LT encoding header is present.
1	BURST_NEXT	1 indicates that current frame is part of burst of HP frames and switch/node should expect reception of another HP frame immediately after current HP frame (with respect to 12-byte minimum interframe gap). This affects the behaviour of non-HP fragmentation. If BURST_NEXT bit is set, switch/node should not attempt to send buffered non-HP frames (or their fragments) after transmission of current HP frame. BURST_NEXT flag works only for certain amount of cycles, set in hardware configuration. When last frame in burst is lost or invalid, switch will continue normal operation after specified number of clock cycles.
2	BURST_LAST	1 indicates that current frame is the last one in burst of HP frames. After routing this frame, switch can continue routing non-HP traffic.
3	DROP_ON_COLLISION	1 indicates that in case of collision of two HP frames, this frame should be dropped. The main aim of this flag is to protect sensitive control/timing data from interference caused by stray HP frames (e.g. sent by badly configured/programmed nodes).
4	DELAY_ON_COLLISION	1 indicates that in case of collision of two HP frames, this frame can be delayed and routed nondeterministically. Aim of this flag is similar to aim of DROP_ON_COLLISION. Upstream HP frames (sent by slave nodes), should probably have this flag set.
5	DELAYED	1 indicates that this HP frame was routed nondeterministically (probably because of collision with another HP frame). The bit should be set by the switch at which collision happened.

Table 1: Structure of `hp_flags` field

```
  uint32_t original_crc;          // CRC of original frame (before fragmentation)
  uint32_t crc;                   // frame CRC32
};
```

Field name	Field type	Description
is_continue	uint16_t	Non-zero value of this field informs, that payload of current SP frame contains continuation of previous non-HP frame which has been fragmented due to routing of HP frame.
continue_offset	uint16_t	Contains payload offset at which previous frame was interrupted.
prev_segment_crc	uint32_t	Value of CRC32 of previous segment of fragmented frame, allowing for detection of proper segment order during reassembly.

Table 2: Contents of SP frame header

Non-HP frames can be fragmented multiple times. In this case multiple SP continuation frames are sent. Last segment contains value of original (received) CRC checksum of fragmented frame before its own CRC. This is justified by following reasons:

- to allow for easy detection of last segment of fragmented frame (if checksum of fragmented payload equals original CRC ? this is the last segment of frame)
- to check if original frame (before fragmentation) was correct. If it wasn't, instead of being reassembled, frame is dropped or forwarded with invalid CRC (depending on settings of the switches).

5.3 HP frame handling and non-HP traffic fragmentation

As we mentioned before, HP frames have absoulte priority over any other traffic in WR network. Therefore, when HP frame arrives, currently routed non-HP traffic shall be interrupted. Unfortunately, this may lead to excessive packet loss in non-HP traffic. To avoid such situations, WRP provides frame fragmentation and reconstruction mechanism, depicted on fig. 2.

Below is description of situation shown on fig. 2.

1. At the beginning, WR switch is routing non-HP frame.

2. HP frame arrives. Switch immediately issues broken frame marker (to indicate that this frame is fragmented) followed by CRC of data sent so far. After then, switch issues end-of-packet delimeter (K29.7 8b10b control character) to terminate currently transmitted packet, then waits 12 byte clock cycles (required minimum frame-to-frame gap) and forwards the HP frame. In the meanwhile, data of incoming non-HP frame is buffered.

3. After HP frame has been sent and if it wasn't a part of burst of HP frames (`BURST_NEXT` flag not set), switch sends SP frame with `is_continue` field set to non-zero value, containing rest of fragmented frame payload and its original (received) CRC.

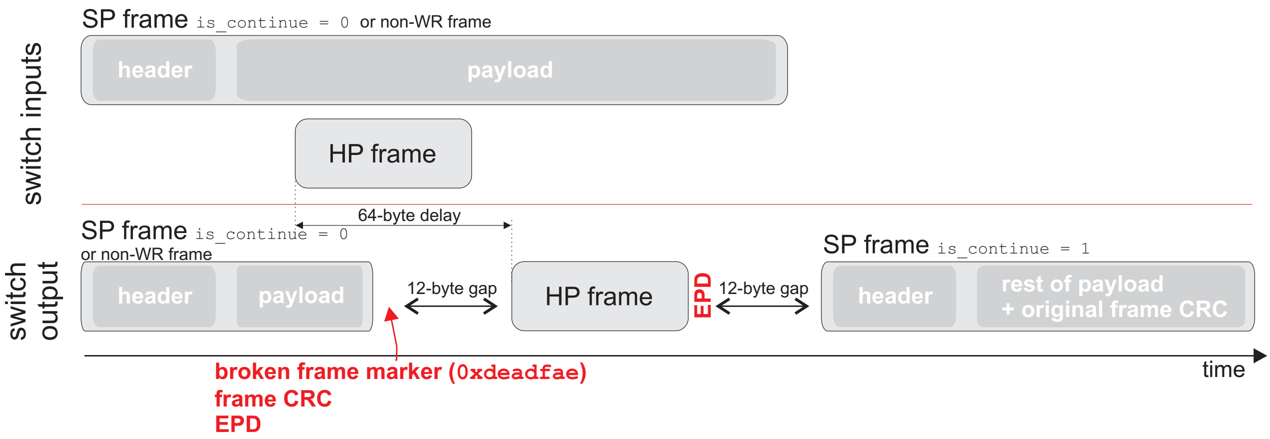

Figure 2: Fragmentation of non-HP frames by single HP frame

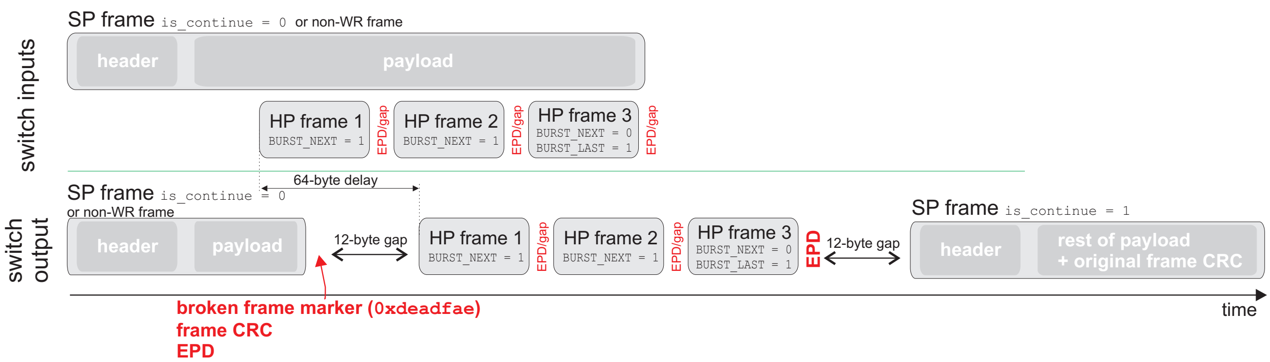

Figure 3: Fragmentation of non-HP frames by burst of HP frames

Sometimes (althouth very rarely) nonfragmented (full) frames may end with broken frame markers. In such situation node shall wait for reception of next frame and check if it is a continuation of previous frame. Frames can be fragmented multiple times.

In general, continuation packets shall be handled with highest priority (e.g. if certain non-HP frame has been fragmented and there are other non-HP frames waiting in the queue, continuation packets shall be sent first). Unfortunately, due to nondeterministic behaviour of network gear for non-HP traffic, we cannot guarantee that next SP packet received by node after fragmented frame will always contain its continuation. To make reassembly possible in such situation, the CRC of previous segment (`prev_segment_crc`) is encapsulated in continuation packet header. Using this information, along with `continue_offset` field we can reconstruct the original frame. Linking frame segments both by `continue_offset` and CRC fields, minimizes the risk of reassembling frame incorrectly. Fragmentation/reassembling example (where frame is split into 3 segments) is depicted on fig. 4.

HP frames are always delayed by switches by constant value of 64 byte clock cycles. This is to prevent fragmentation of headers of SP/non-WR frames. If HP frame arrives when SP/non-WR frame header data is being forwarded, we have enough time to finish transmission of SP/non-WR header before start of transmission of HP frame. This approach should also make it possible to implement frame fragmentation feature in software on standard (non-WR) Ethernet gear – fragmentation may never cause corruption of frame headers, so they should be accepted by ordinary network controllers/switches.

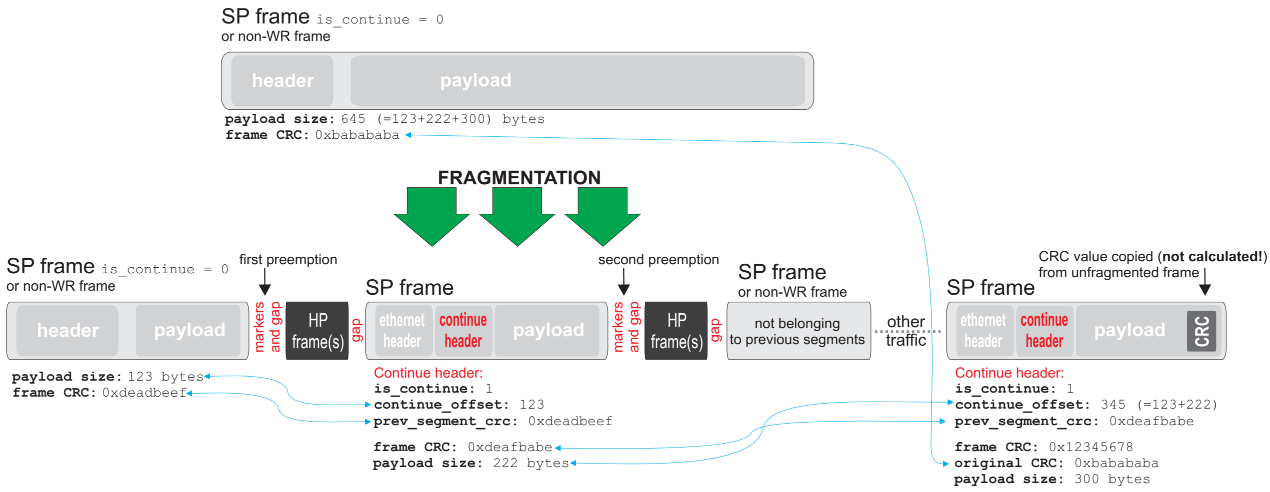

Figure 4: SP frame fragmentation/reassembly example

In order to prevent overflows in switch FIFOs, all nodes should immediately pause transmission of current SP/non-WR data when HP is being received.

Please note that fragmentation feature has been introduced only to minimize bandwidth loss caused by retransmission of interrupted SP/non-WR packets. It hasn't been designed to replace TCP/IP (or other high-layer protocol) reassembly engine. Packets which are incorrectly reassembled by White Rabbit hardware therefore should be detected and handled by higher-layer protocols.

5.4 Collision handling on HP frames

As the HP frames are routed deterministically, with highest priority, collisions may occur when multiple devices try to send HP frames independently. To avoid such situations, higher layer protocols should schedule HP frame transmissions. Nevertheless, collisions may still happen, caused by stray HP frames sent by faulty or non-WR nodes. WRP protocol provides simple collision management for HP frames by adding `DROP_ON_COLLISION`, `DELAY_ON_COLLISION` and `DELAYED` flags in each HP frame header. The switch may behave in multiple ways (see figure 5), depending on state of these flags:

1. Both packets have `DROP_ON_COLLISION` = 1 – both packets are dropped.
2. 1st packet: `DROP_ON_COLLISION` = 1, 2nd packet: `DROP_ON_COLLISION = 0`: 1st packet is immediately terminated and 2nd packet is routed deterministically.
3. 1st packet: `DROP_ON_COLLISION` = 0, 2nd packet: `DROP_ON_COLLISION = 1`: 1st packet is routed deterministically, 2nd packet is dropped.
4. Both packet have `DROP_ON_COLLISION` = 0: 1st packet is routed deterministically while 2nd packet is buffered. After transmission of 1st packet, 2nd packet is routed nondeterministically with `DELAYED` flag set.

The `DELAY_ON_COLLISION` flag allows the switch to delay colliding frames when the its value is set to 1. Switches should interpret `DELAY_ON_COLLISION` in similar way as described above, except no packets should be dropped. If both colliding frames have `DELAY_ON_COLLISION`

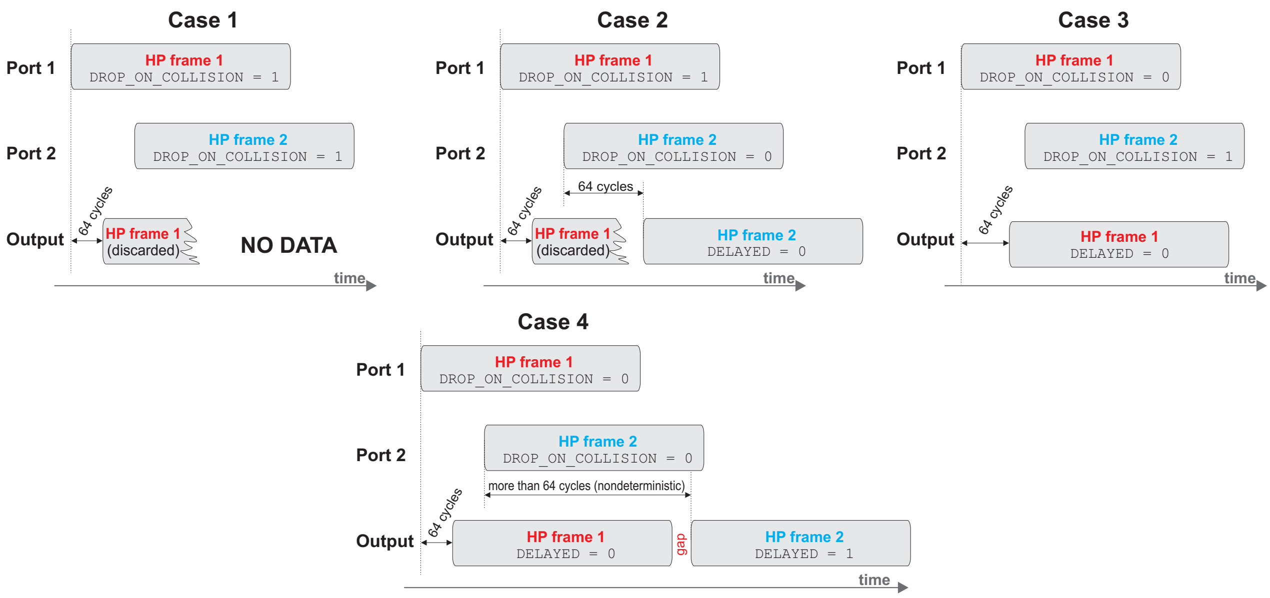

Figure 5: Collision handling for HP frames

= 0 and `DROP_ON_COLLISION` = 0, the second frame is delayed without respect to flags and appropriate `DELAYED` bit is set.

5.5 LT encoding

To maintain high reliability for non-handshaked HP protocols for sensitive timing or control data, special encoding scheme, called LT coding is provided. Its idea is described below:

1. Sender splits original frame into n segments of size `segment_size`, varying from 32 to 384 bytes. These segments are named $A1 \ldots An$

2. Sender generates $m > n$ blocks $X1 \ldots X2n$ by XORing randomly chosen segments from set $A1 \ldots An$. All blocks must be different. For example:
   $X1 = A1\ xor\ A3\ xor\ A7\ xor\ A8$
   $X2 = A2\ xor\ A4\ xor\ A6\ xor\ A8$
   $Xm = \ldots$

3. Sender transmits $X1..Xm$ in burst of separate HP frames with `LT_ENABLED` flag set and proper LT coding header containing:

   - 32 bit unique identifier of original frame (`orig_frame_id`) allowing to identify to which frame each block belongs)
   - 32-bit integer with equation coefficients (`eqn`). For example, value of *100101* means that $Xi = A1\ xor\ A4\ xor\ A6$.
   - Segment size in bytes (`segment_size`)
   - Index of segment (`segment_num`)
   - MD5 checksum of payload and LT header (`md5sum`)

4. Assuming that at least N blocks reached the receiver without errors, receiver can solve the equation system from p. 2. and reconstruct the original frame.

Reliability of this encoding system depends on:

- effectiveness of used checksum algorithm (MD5 should be more than sufficient)
- bit error rate of link. Assuming that BER will be the same as observed in test GbE link (very faulty – about 1 broken packet per second) this scheme will work perfectly well.

6 The WRCMP Protocol

In this section, the White Rabbit Control Message Protocol (WRCMP) format is described along with some use cases. WRCMP along with PTPv2 is used by WR-compatibile network devices for detection, low-level maintenance and synchronization of network nodes.

6.1 WRCMP message encapsulation

All WRCMP messages are sent using SP frames. HP frames provide only transport layer for timing-sensitivie traffic, and they shall never be used for WRCMP messages. Every WRCMP packet has following NTLV (name-type-length-value) structure:

Listing 3: C-like structure of WCMRP message

```
#define FIELD_SIGNED 0x40    // field type modifier flag, stating that
                             // field value is signed

#define FIELD_ARRAY 0x80     // field type modifier flag, stating that
                             // field value is array of field_type

__attribute__((packed)) struct wrp_message {

  uint16_t num_fields;       // Number of NTLV fields in packet

  struct field_table[] {     // Table of NTLV fields (num_fields entries)
    char field_name[4];      // 4-character unique name of field
    uint8_t field_type;      // field type (see table 1)
    uint16_t field_length;   // total field data size (in bytes)
  };

  char field_values[...];
};
```

Table 3 shows available `field_types`. All multibyte binary data is encoded in big-endian format. An example of WRCMP message containing 4 different NTLV fields is provided in table 4:

- message ID 'MSID' = 0
- 32-bit signed integer 'INT1' = -100000
- array of 4 16-bit unsigned integers 'ARR1' = 10, 2000, 7777, 54321
- ASCII string 'STR1' = 'Hello, world'

Table 3: WRCMP message field types

Type ID	Flags	C type	Base size	Description
0x00	none	msgid_t	2	16-bit unsigned integer, containing unique type identifier of the message. Can be used only once in each packet
0x01 0x41	FIELD_SIGNED FIELD_ARRAY	sint8_t uint8_t	1	8-bit unsigned/signed integer in two's complement format
0x02 0x42	FIELD_SIGNED FIELD_ARRAY	sint16_t uint16_t	2	16-bit unsigned/signed integer in two's complement format
0x03 0x43	FIELD_SIGNED FIELD_ARRAY	sint32_t uint32_t	4	32-bit unsigned/signed integer in two's complement format
0x04 0x44	FIELD_SIGNED FIELD_ARRAY	sint64_t uint64_t	8	64-bit unsigned/signed integer in two's complement format
0x05	FIELD_ARRAY	float	4	IEEE754 32-bit (single precision) floating point
0x06	FIELD_ARRAY	double	8	IEEE754 64-bit (double precision) floating point
0x07	none	char *	variable	ASCII null-terminated string, UTF8 encoding

Table 4: Example WRCMP message

+00:	00 04	- num˙fields (4)
+02:	4D 53 49 44	- field 0 name ('MSID')
+06:	00	- field 0 type (msgid˙t)
+07:	00 02	- field 0 length (2 bytes)
+09:	49 4E 54 31	- field 1 name ('INT1')
+0D:	43	- field 1 type (sint32˙t)
+0E:	00 04	- field 1 length (4 bytes)
+10:	41 52 52 31	- field 2 name ('ARR1')
+14:	82	- field 2 type (array of uint16˙t)
+15:	00 08	- field 2 length (8 bytes = 4 uint16˙t's)
+17:	53 54 52 31	- field 3 name ('STR1')
+1B:	07	- field 3 type (UTF8 string)
+1C:	00 0D	- field 3 length
+1E:	04 D2	- field 0 value (MSID = 1234 decimal)
+20:	FF FE 79 80	- field 1 value (INT1 = -100000)
+24:	00 0A 07 D0	
+28:	1E 61 D4 31	- field 2 value (ARR1 = 10, 2000, 7777, 54321)
+2C:	48 65 6C 6C	- field 3 value (STR1 = 'Hello, world')
+30:	6F 2C 20 77	
+34:	6F 72 6C 64 00	

6.2 WRCMP INVITE

Direction: switch/master → node
Sent in response to: detection of new node in network
Expected response: `WRCMP_INVITE_RESPONSE`
Source address: sender address
Destination address: slow protocol default (`01-80-c2-00-00-02`)
Format: see table 5

`WRCMP_INVITE` is sent by switch after detection that new node has been connected. Its purpose is to inform new node about network/timing configuration and request for its capabilities. To make sure that the message will not be routed further, destination address is the slow protocol default address (`01-80-c2-00-00-02`). The message is received by freshly connected node and acknowledged by sending `WRCMP_INVITE_RESPONSE` packet. Invite packet contains all addresses that node needs to know and important timing information.

6.3 WRCMP INVITE RESPONSE

Direction: node → switch/master
Sent in response to: `WRCMP_INVITE`
Expected response: `WRCMP_ACK`
Source address: sender address
Destination address: switch address from `WRCMP_INVITE`
Format: see table 6

WRCMP_INVITE_RESPONSE packet contains capabilities of freshly connected device, allowing for switches/master to check if its WR-compliant. WRP˙INVITE˙RESPONSE is sent to closest switch, and then forwarded by switch to System Timing Master.

6.4 WRCMP ACK

Direction: any
Sent in response to: WRCMP_INVITE_RESP, WRCMP_REPORT_NODE, WRCMP_REPORT_DELAY, WRCMP_HEARTBEAT
Expected response: none
Source address: sender address
Destination address: any
Format: see table 7

WRCMP_ACK packet is used to acknowledge reception of packets listed above or to indicate error.

6.5 WRCMP REPORT NODE

Direction: node → switch, switch → STM
Sent in response to: node/switch event or state change
Expected response: WRCMP˙ACK
Source address: sender address
Destination address: switch or master
Format: see tables 8 and 9

WRCMP_REPORT_NODE packet is used to report results of delay measurement and current node state to:

- System Timing Master (used for calculating delays between HP frames to prevent collisions and reporting state of nodes).
- closest switch

WRCMP_REPORT_NODE is sent by each node to closest switch when node state changes. It can be issued when:

- node have perfomed initial delay compensation,
- node has chaged its uplink port,
- amount of corrupted packets received by node exceeded certain limit.
- when switch/master requests immediate report by sending WRCMP_REQUEST_REPORT message.

The switch can also independently issue WRCMP_REPORT_NODE message to STM in response to state change of one of connected nodes:

- when new node has just been physically connected/disconnected to any of switch ports,
- when node stopped sending WRCMP_HEARTBEAT messages and seems dead,
- if amount of corrupted packets received from node exceeds certain limit,

- when link delay between switch and any of nodes changed significantly.

Switches shall forward all ingress `WRCMP_REPORT_NODE` messages to System Timing Master.

6.6 WRCMP REPORT DELAY

Direction: switch → node
Sent in response to: after delay measurement (periodic event)
Expected response: `WRCMP_ACK`
Source address: switch/master address
Destination address: measured node
Format: see table see table 10

`WRCMP_REPORT_DELAY` packet is sent to node after the switch had finished link delay measurement. Using information in this packet, slave can perform delay compensation. These packets are never forwarded by switches.

6.7 WRCMP HEARTBEAT

Direction: node → switch
Sent in response to: none (periodic event)
Expected response: `WRCMP_ACK`
Source address: node address
Destination address: switch/master address
Format: see table see table 11

`WRCMP_HEARTBEAT` packet is sent periodically by node to closest switch to indicate that the node is alive. The frequency boundaries of sending heartbeat messages are defined by in `WRCMP_INVITE` message. If the node had not been sending WRCMP˙HEARTBEAT messages for several periods, switch may assume that it is dead (hanged up or faulty) and issue proper `WRCMP_REPORT_NODE` message for STM.

To avoid making excessive traffic in network, heartbeat packets are never forwarded by switches.

6.8 WRCMP REQUEST REPORT

Direction: switch → node
Sent in response to: none
Expected response: `WRCMP_REPORT_NODE`
Source address: switch/master address
Destination address: node address
Format: see table see table 12

`WRP_REQUEST_REPORT` packet is issued by STM/switch to force node to send unconditional `WRP_REPORT_NODE` message (e.g. not in response to state change or event).

7 WRCMP use cases

7.1 New node connected to switch

Detection of just connected nodes is done by looking for presence carrier (e.g. valid 8b10b symbols) on switch's downlink ports. If carrier is present, the following procedure is executed:

1. Switch waits for 1 second and sends `WRCMP_INVITE` packet to freshly connected node
2. Node responds with `WRCMP_INVITE_RESPONSE` packet containing its capabilities
3. If correct `WRCMP_INVITE_RESPONSE` was received, switch sends ACK packet to the node. Otherwise switch sends `WRCMP_INVITE` packet again. Retransmission can be performed many times, depending on switch settings.
4. If there is no response from node, or the response is malformed, switch assumes that device is nonsynchronous and non-WR capable. Switch issues `WRCMP_REPORT_NODE` message to System Timing Master, stating that node has been `JUST_CONNECTED` and is not WR-compatible.
5. If the node replied with proper `WRCMP_INVITE_RESPONSE`:
   - `WRCMP_ACK` packet is sent to node,
   - `WRCMP_INVITE_RESPONSE` packet received from node is forwarded to System Timing Master. Master replies with `WRCMP_ACK`. If there is no response, retransmission is performed.
   - `WRCMP_REPORT_NODE` packet is issued to STM with proper node state. Master replies with `WRCMP_ACK`. If there is no response, switch retransmits `WRCMP_REPORT_NODE` message.
6. Switch starts performing delay measurements if the slave is WR-compliant.

7.2 Delay measurement

Delay/phase measurement is done using PTPv2 peer delay message exchange as following:

1. Master (switch or STM) sends `Pdelay_req` packet to slave
2. Slave (node) responds with `Pdelay_resp` packet containing value of `Pdelay_req` reception timestamp (t2) and `Pdelay_resp_followup` containing value of `Pdelay_resp` send timestamp (t3).
3. Points (1-3) can be repeated several times, so the master could determine the phase shift introduced by link.
4. Master calculates link delay and asymmetry and reports it to node using `WRCMP_REPORT_DELAY` message.

7.3 Node reporting and maintenance

For network maintenenance and timing model optimization, nodes are obliged to report their state. Reporting is done following way:

1. If state of the node has changed, node must immediately issue proper `WRCMP_REPORT_NODE` message to switch. Switch responds with `WRCMP_ACK`. If there is no reply, `WRCMP_REPORT_NODE` is sent again. These packets are exchanged only between node and the closest switch.

2. If the switch received proper `WRCMP_REPORT_NODE` message and everything seems OK it just forwards it to System Timing Master.

3. If node is not sending `WRCMP_HEARTBEAT` packets, but valid carrier is present, switch issues `WRCMP_REPORT_NODE` packet with `NO_RESPONSE` state and forwards it to System Timing Master. `If` carrier has been lost, switch issues `WRCMP_REPORT_NODE` packet with `DISCONNECTED` state to System Timing Master. `STM` acknowledges reception of `WRCMP_REPORT_NODE` by sending `WRCMP_ACK`. If there is no response from STM, reporting packet is sent again.

Switch/STM can also force nodes to send `WRCMP_REPORT_NODE` messages immediately by sending `WRCMP_REQUEST_REPORT` message.

7.4 Interoperability with non-WR compliant nodes

WR protocol was meant to be compatible with non-WR nodes/network segments. If the new node doesn't respond to invitation message with proper `WRP_INVITE_RESPONSE`, it is assumed to be incompatible with WR. If such situation occurs, the switch:

- disables deterministic HP frame routing for incompatible node. HP frames delivered for the node are treated like normal frames to prevent fragmentation.
- (optional) enables on-the-fly fragmented frame reconstruction on port to which incompatible node is connected or drops fragmented frames (depending on configuration)

Non-WR nodes can still use synchronous mode and PTP delay measurements with precision of single Ethernet clock cycle (8 ns)

7.5 Link idle

For timing transmission, WR network uses reference clock embedded into serial Ethernet data stream. Therefore, PLLs in timing receivers must be locked for the whole time and there is no such thing like link idle state. If there is no data to send, transmitters shall send either commas (K28.5) or link idle sequences (K28.5/D5.6).

8 WRP and PTPv2 compatibility

We have chosen subset of PTP version 2 protocol as a method for delay measurement and compensation in WR network. Therefore, each WR-compatible device shall implement PTP-compatible packet timestamping and at least following set of PTPv2 messages:

- `Pdelay_req`
- `Pdelay_resp`
- `Pdelay_resp_followup`

These messages are necessary to synchronize two WR-compatible devices. However, to keep interoperability with non-WR PTP slaves, all switches must implement fully featured PTPv2 daemon.

PTPv2 messages used by WR-compatible devices are encapsulated into Ethernet frames according to IEEE1588.D2.2 annex F. Such frames are distinguished by unique value of Ether-Type (0x88f7), and non-forwardable destination address `01-80-C2-00-00-0E`.

All delay calculations and phase shift measurements must be performed by master side (e.g. switch or STM). Slaves (both WR-compatible and incompatible with WR) should be agnostic of delay/phase measurement method used. Phase measurement algorithm must not not rely on slave capabilities (e.g. slave-side phase shifting) except for operation in synchronous mode.

Table 5: WRCMP INVITE message format

Field	Type	Size	Description
msg_id	msgid_t	MSID	Unique message identifier for WRCMP_INVITE. Equals to 0x01.
switch_mac	uint8_t[6]	SWMA	MAC address of switch to which node is directly connected.
master_mac	uint8_t[6]	STMM	MAC address of System Timing Master
switch_port_id	uint16_t	SPRT	Number of switch port to which node is connected
switch_layer_id	uint16_t	SLAY	Layer of network to which device is connected: 0 – directly to master, 1 – 1 hop to master, 2 – 2 hops to master, etc...
min_delay_sync_period	uint32_t	MIDS	Minimal period of delay compensations performed by switch (in milliseconds)
max_delay_sync_period	uint32_t	MADS	Maximum period of delay compensations performed by switch (in milliseconds)
min_heartbeat_period	uint32_t	MIRP	Minimal period of sending WRCMP_HEARTBEAT mesasges to closest switch
max_heartbeat_period	uint32_t	MARP	Maximum period of sending WRCMP_HEARTBEAT mesasges to closest switch
cap_wr_switch_type	uint8_t	CPST	Type of switch to which node is connected: • 0x01 – SWITCH_TYPE_BB - backbone (fiber) WRP switch, • 0x02 – SWITCH_TYPE_TP_100M - twisted-pair 100 Mbps WRP switch, • 0x03 – SWITCH_TYPE_TP_1G - twisted-pair 1 Gbps WRP switch

Table 6: WRCMP INVITE RESPONSE message format

Field	Type	Size	Description
msg_id	msgid_t	MSID	Unique message identifier for WRCMP_INVITE_RESPONSE. Equals to 0x02.
node_mac	uint8_t[6]	NOMA	Node's own MAC address.
node_version	uint32_t	VRSN	32-bit identifier of node protocol version
node_type	uint16_t	NOTY	Type of node: 0x01 - NODE_BB_SWITCH (backbone WR fiber switch), 0x02 - NODE_BB_SWITCH_ALTERNATE (backbone WR fiber switch, alternate uplink port), 0x03 - NODE_BB_SLAVE (generic WR slave device), 0x04 - NODE_TP_GATEWAY (WR twisted pair gateway), 0x05 - NODE_TP_SWITCH (twisted-pair WR switch)
cap_sync_mode	uint8_t	CPSY	Nonzero value means that node is capable of synchronous operation
cap_phase_measurement	uint8_t	CPPM	Nonzero value means that node can perform fine phase shift measurement for delay compensation
cap_ptp	uint8_t	CPPT	Nonzero value means that node supports delay compensation via PTPv2
cap_hp_recv	uint8_t	CPHR	Nonzero value means that node is capable of receiving HP frames
cap_hp_send	uint8_t	CPHS	Nonzero value means that node is capable of sending HP frames
cap_fragmentation	uint8_t	CPFR	Nonzero value means that node is capable of sending and receiving fragmented SP/non-WR frames
min_hp_routing_delay	uint32_t	MIHR	(switch/gateway only) Minimum and maximum values of HP packet routing delays (in 8ns clock cycles). If they are equal, routing time is assumed to be constant.
max_hp_routing_delay	uint32_t	MAHR	See above
max_hp_delay_uncert	uint32_t	MADU	(switch/gateway only) Maximal value of nondeterministic (caused by clock misalignment) delay fluctuations for HP routing in units of picoseconds.
max_slave_asymmetry	uint32_t	MASS	Maximal value of slave node RX/TX path asymmetry in picoseconds.

Table 7: WRCMP ACK message format

Field	Type	Size	Description
msg_id	msgid_t	MSID	Unique message identifier for WRCMP_ACK. Equals to 0x03.
ack_value	uint16_t	ACKV	Zero means that packet previous request was acknowledged. Non-zero value means that error occurred and the message was rejected. ack_value field may be used to send value of error code.

Table 8: WRCMP REPORT NODE message format

Field	Type	Size	Description
msg_id	msgid_t	MSID	Unique message identifier for WRCMP_REPORT_NODE. Equals to 0x06.
node_mac	uint8_t[6]	NOMA	MAC address of reported node
switch_mac	uint8_t[6]	SWMA	MAC address of switch to which reported node is directly connected.
current_uplink_port	uint8_t	CUUP	(switch/gateway only) Current uplink port used. 0 = primary, 1 = alternate
uptime	uint64_t	UPTM	Node uptime in seconds.
rx_total	ARRAY of uint64_t	RXTO	Total number of received packets (including HP)
rx_hp	ARRAY of uint64_t	RXHP	Total number of received HP packets
rx_errors	ARRAY of uint64_t	RXER	Total number of invalid packets received (including HP)
rx_errors_hp	ARRAY of uint64_t	RXEH	Total number of invalid HP packets received
event	uint16_t	EVNT	Reported event. Detailed description in table ??
delay_twoway	uint64_t	DETW	Two-way delay (STM – node – STM). In picoseconds.
delay_twoway_alternate	uint64_t	DETA	(switch/gateway only) Two-way delay on alternate uplink port (STM – node – STM). In picoseconds.
max_asymmetry	uint32_t	MAAS	Maximal uplink path asymmetry in picoseconds.

Table 9: WRCMP REPORT NODE event field values

Value	Name	Description
0x01	JUST_CONNECTED	node have just been connected to switch
0x02	UNSYNCHRONIZED	node is connected but it haven't yet performed delay compensation
0x03	READY	node is ready
0x04	UPLINK_PORT_CHANGE	(switch/gateway only) uplink port has been changed due to primary link failure or when primary link is working again
0x05	NODE_NOT_RESPONDING	issued by switch when node which seems to be connected – carrier is presnet – didn't report itself in expected time
0x06	DISCONNECTED	carrier lost, node is disconnected
0x07	ERROR_LIMIT_EXCEEDED	amount of errorneusly received packets received preprogrammed limit

Table 10: WRCMP REPORT DELAY message format

Field	Type	Size	Description
msg_id	msgid_t	MSID	Unique message identifier for WRCMP_REPORT_DELAY. Equals to 0x07.
utc_timestamp	uint64_t	UTCT	UTC send timestamp of this packet.
ts_value	uint32_t	TSVA	PTP counter send timestamp of this packet (in 8ns clock cycles)
delay_twoway	uint64_t	DSN2	Measurement result - two-way overall link delay between switch and node (in picoseconds)
delay_downlink master_to_switch	uint64_t	DMSD	Overall downlink delay between System Timing Master and switch to which node is directly connected (in picoseconds)
delay_uplink switch_to_master	uint64_t	DSMU	Overall uplink delay between switch and System Timing Master to which node is directly connected (in picoseconds)
master_asymmetry	uint32_t	MASY	RX/TX path asymmetry of switch to which node is directly connected (in picoseconds)
link_asymmetry	uint32_t	LASY	Link asymmetry in picoseconds

Table 11: WRCMP HEARTBEAT message format

Field	Type	Size	Description
msg_id	msgid_t	MSID	Unique message identifier for WRCMP_HEARTBEAT. Equals to 0x08.

Table 12: WRCMP REQUEST REPORT message format

Field	Type	Size	Description
`msg_id`	`msgid_t`	`MSID`	Unique message identifier for `WRCMP_REQUEST_REPORT`. Equals to 0x09.