Institute of Embedded Systems

Zurich University of Applied Sciences

School of Engineering

InES Institute of Embedded Systems

PRP-1 stack

Authors:	S. Meier
Creation date:	08 February 2007
Last modification:	04 May 2012
Version:	1.0.7
State:	released

Contact address

Institute of Embedded Systems (InES)
Zurich University of Applied Sciences
Technikumstr. 22
Postfach
CH-8401 Winterthur

Tel: +41 58 934 75 25
Fax: +41 58 935 75 25

Mail: support.ines@zhaw.ch
http: ines.zhaw.ch/

Document

Distribution: ZHAW – InES

History

Version	Date	Author/Revisor	Description
1.0.1	08. Feb. 2007	Meier Sven	Created
1.0.2	16. Mar. 2007	Meier Sven	Changed the API etc
1.0.3	20. Mar. 2007	Meier Sven	Updated some classes
1.0.4	19. Sep. 2007	Meier Sven	SRP stuff added
1.0.5	13. Feb. 2008	Meier Sven	Updated Graphics etc
1.0.6	01. Jul. 2011	Weibel Hans	
1.0.7	01. May 2012	Walch Oliver	Updated to PRP-1

Contents

1 About this document

This is a design and implementation description of the Parallel Redundancy Protocol (PRP-1) Software.

1.1 The PRP-1 Stack

The requirements are as followed:

The Parallel Redundancy Protocol Stack shall support the whole PRP-1 standard.
It shall allow running IEEE1588/PTP.
It shall support SRP.
It shall be manageable.
It shall be easy portable to other Operating Systems.
It shall be as fast as possible.
It shall use as little resources as possible.
It shall be extendable.
It shall have a clear API.

With these requirements in mind the design described in the following chapters was chosen.

2 Design

2.1 Block Diagram

Figure 1: illustrates the relations between the different components of a PRP device.

Each PRP device owns a PRP Environment and a PRP Node Table, which is embedded into the PRP Environment. The PRP Environment provides the protocol functionalities like Discard Algorithm, Supervision or Redundancy Control Trailer as well as global and node specific data. The interfaces (PRP, PRP_xxxItf) decouple all the operating system dependent functions from the PRP software which is generic. These interfaces are implemented as singletons what means that every interface exists only once. The access layer PRP (API) is the communication path between user programs and the PRP software.

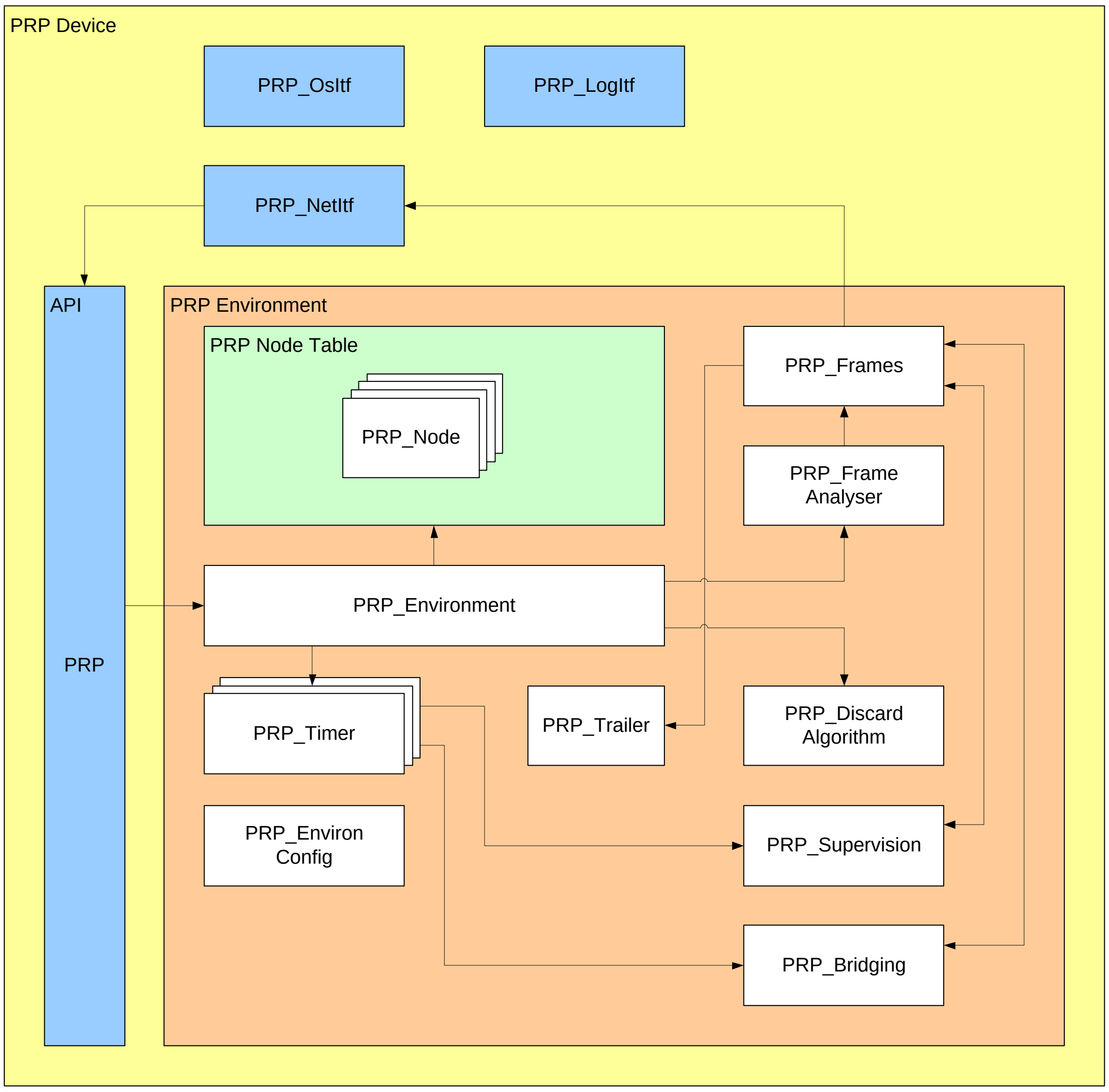

Figure 1: The components of the PRP software

2.2 Mode of operation

Basically the following steps are executed when an event occurs:

An event occurs (e.g. a frame was received).
The respective API function gets called (e.g. receive()).
The PRP_Environment invokes further actions depending on the current event (e.g. pass the received frame to the PRP_Frame_Analyser).

2.3 Layer Diagram

PRP-1 software mainly consists of two layers. One layer is the protocol engine – it's generic and hasn't got to be changed at all in order to run on a specific operating system. It's implemented according to the PRP-1 standard. The other layer is the OS abstraction layer with the interfaces – this is where all the environment dependent code is located at. The interfaces have to be adapted to the OS so the protocol engine has access to the used resources which the OS -environment provides. The layers are shown in Figure 2:

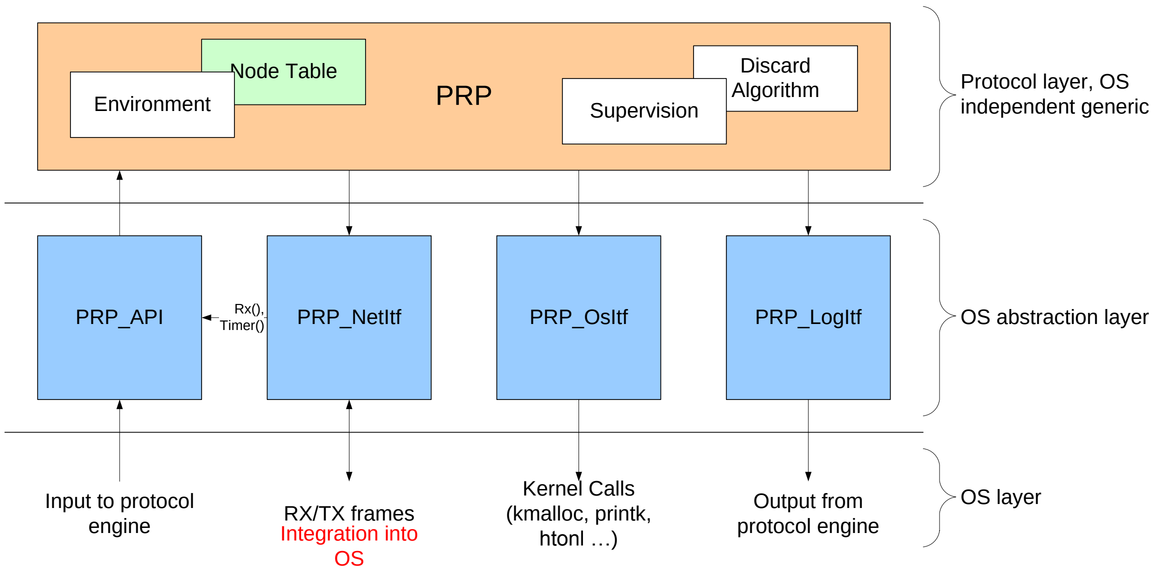

Figure 2: Layers and interaction

2.4 Components

2.4.1.1 PRP_Environment

The PRP_Environment is kind of a multiplexer. It forwards API calls coming from the PRP_API to the respective class.

2.4.1.2 PRP_Node_Table

The PRP_Node_Table is the entry point to the node table described in the standard. It has a pointer to the first and the last node (PRP_Node) in the table and provides functions to add/remove and search nodes in the table.

2.4.1.3 PRP_Node

The PRP_Node class contains general information about a remote node.

2.4.1.4 PRP_Bridging

This module implements the SRP part of the protocol. It forwards the frames back to the sender. SRP is a simplification of RSTP.

2.4.1.5 PRP_Frames

This is the actual core of the design. It gets nodes out of the table, adds or removes the RCT and creates duplicates in the transmission path.

2.4.1.6 PRP_Frame_Analyser

The PRP_Frame_Analyser checks the frame type, whether it is a normal, a ptp or a supervision frame, and forwards it to the respective handler.

2.4.1.7 PRP_Discard_Algorithm

This module implements the discard algorithm, the actual heart of the protocol.

2.4.1.8 PRP_Supervision

The PRP_Supervision block sends and receives supervision frames and checks the PRP_Node_Table for expired nodes.

2.4.1.9 PRP_Timer

This is the Timer needed for the supervision of the node table

2.4.1.10 PRP_Lock

This block is used to guarantee atomic access on shared resources.

2.4.1.11 PRP_Trailer

This module provides functions to get add and remove the RCT.

2.4.1.12 PRP_Environment_Configuration

The PRP_Environment_Configuration provides the general information about the protocol engine defined in the PRP-1 standard.

2.4.1.13 PRP

The PRP is the API of the PRP-1 software. It encapsulates the functions of the PRP_Environment that must be callable from outside. It guarantees atomic access to the protocol engine.

2.4.1.14 PRP_Operating_System_Interface

The whole software runs in user space to get an OS independent protocol core.

2.4.1.15 PRP_Network_Interface

This PRP_network_Interface module provides functions to transmit and receive frames. It is the wrapper class around the protocol core. The integration into the network stack of the operating system is done here.

2.4.1.16 PRP_Log_Interface

The PRP_log_Interface is the output interface of the software.

3 Implementation

The implementation of PRP-1 is done with object-oriented C. So the advantages of object-oriented analysis / design / programming can be applied (also methodology, encapsulating, reusability etc.). UML was used to design the software.

3.1 Object-oriented C

This clause shows how a C++ class is realised in C.

A C++ class:

```
/*** PRP_Timer class in C++ ***/
class PRP_Timer
{
   /** Attributes */
   private:
            boolean enabled_;
            Unsigned32 value_;
            Unsigned32 timeout_;

   /** Methods */                                                    Constructor
   public:
            PRP_Timer( Unsigned32 timeout );                          Destructor

            ~PRP_Timer();


            tick();
            restart();
            stop();
}
```

The same class PRP_Timer in C is realised as the following.

```
/*** PRP_Timer class in C ***/
typedef struct PRP_Timer_T PRP_Timer_T
{
   /** Attributes */
   boolean enabled_;
   Unsigned32 value_;
   Unsigned32 timeout_;
};
                                                                      Constructor
/** Methods */
PRP_Timer_Init( PRP_Timer_T *const me, Unsigned32 timeout );          Destructor

PRP_Timer_Cleanup(PRP_Timer_T *const me );


PRP_Timer_tick(PRP_Timer_T *const me );
PTP_Timer_restart(PTP_Timer_T *const me );
PTP_Timer_stop(PTP_Timer_T *const me );
```

The attributes of a class are encapsulated in a structure. To avoid the struct keyword, the class struct PRP_Timer is defined as a new type: typedef struct PRP_Timer. Because there is no this pointer in C, a pointer of type of this class is passed to every function of this class. The names of the methods (or functions) of a class shall always begin with the class name. Constructor and destructor shall be named <CLASSNAME>_Init and <CLASSNAME>_Cleanup respectively. Some classes own functions named <CLASSNAME>_Create and <CLASSNAME>_Destroy. The _Create function creates an instance of the class dynamically and the _Destroy function destroys the instance and frees the allocated memory. These two functions also call the _Init and _Cleanup functions.

3.2 UML Class Diagram

Hereafter, a class shall be a synonym for typedef struct. The class diagram shows the relations between the different classes.

Conventions for the UML diagram:
The Init (constructor), Cleanup (destructor), Create and Destroy functions are not displayed in the diagram.
Although each class has a me* pointer (this pointer in C++), it's not displayed in the UML diagram. Furthermore, for singleton classes there is no me* pointer because there is only one instance of each singleton class anyway.
In the UML diagram, the names of the methods are displayed without the class name prefix. So e.g. a method with the name PRP_Timer_T_tick() is only displayed as tick() in the diagram.

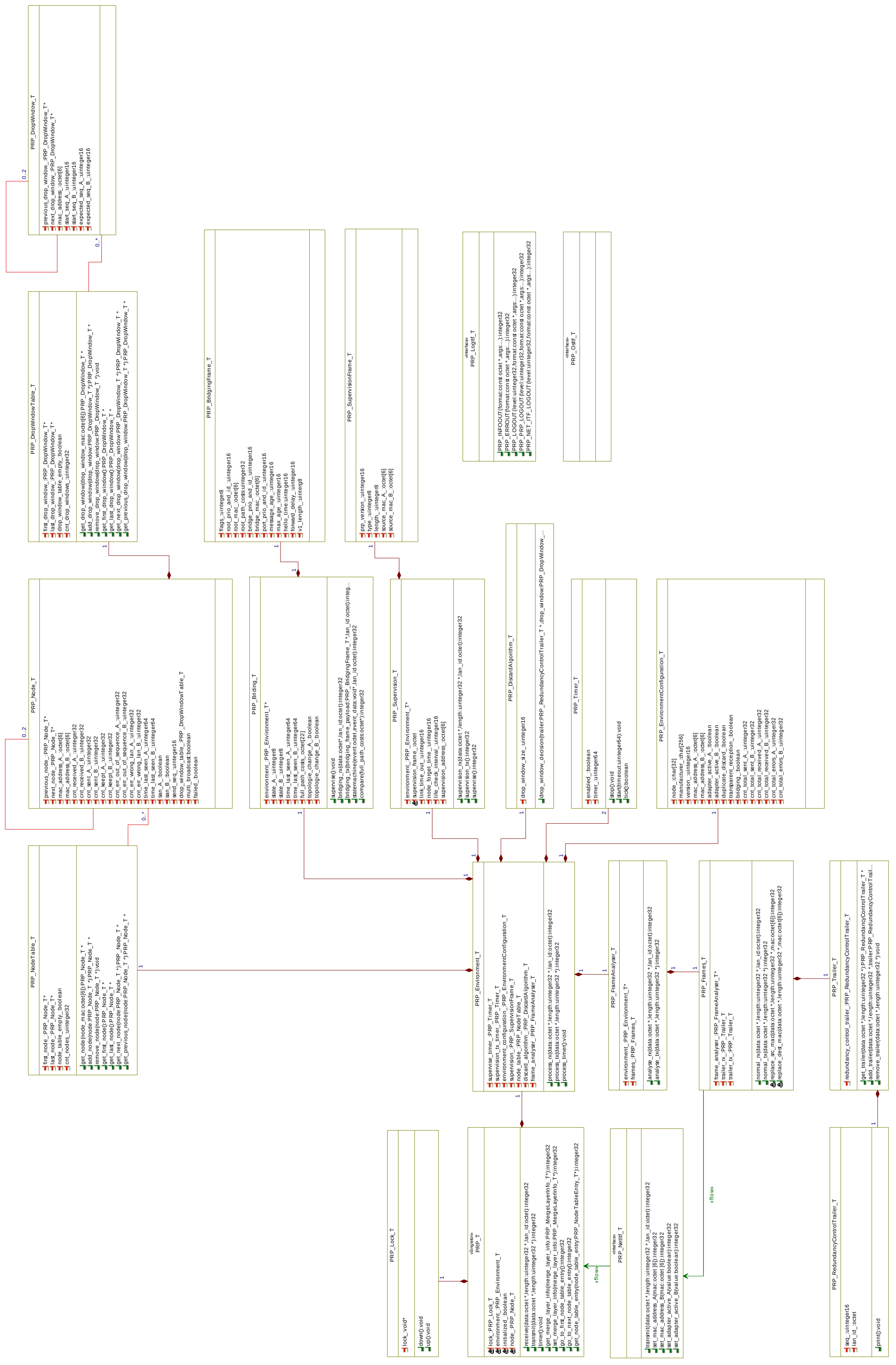

Figure 3: Class diagram

3.3 API and Interfaces (OS abstraction)

3.3.1 PRP_T

This class represents the API of the PRP-1 stack and describes the functions which can be called from outside in order to impact it. *PRP_T* handles inputs to the stack as function calls. Every time an API function is called, it will pass a lock saved section, so every API call is atomic. This class is implemented as a singleton therefore there is no me* pointer.

In order to ensure an error free operation of the PRP-1 Stack, inputs have to be passed to the stack by (only) using the API functions. PRP_T is thread save and has not to be adapted for a specific OS.

3.3.1.1 Attributes

Attribute	lock_
Description	All API functions are using this lock to have atomic access to the protocol engine
Visibility	private
Type	semaphore

Attribute	initialized_
Description	Indicates if the stack is initialized or not
Visibility	private
Type	semaphore

Attribute	node_
Description	Node to read over management
Visibility	private
Type	semaphore

Attribute	environment_
Description	Instance of the protocol engine
Visibility	private
Type	PRP_Environment_T

3.3.1.2 Functions

Function	PRP_T_receive	
Description	Receive function for all incoming frames of the PRP network adapters	
Visibility	public	
Arguments	octet * data	Pointer to the beginning of the frame (dest mac)
	uinteger32* length	Length in bytes of the frame
	octet lan_id	On which LAN it was received
Return value	integer32	1 : DROP 0 : KEEP <0 : ERROR (code)

Function	PRP_T_transmit	
Description	Transmit function for all frames of the PRP network adapters	
Visibility	public	
Arguments	octet * data	Pointer to the beginning of the frame (dest mac)
	uinteger32* length	Length in bytes of the frame

Return value	integer32	0 : OK <0 : ERROR (code)

Function	PRP_T_timer	
Description	Timer (1s) for all periodical jobs	
Visibility	public	
Arguments	void	-
Return value	void	-

Function	PRP_T_get_merge_layer_info	
Description	Copies the Merge Layer information to the object passed as argument.	
Visibility	public	
Arguments	PRP_MergeLayerInfo_T* merge_layer_info	Pointer to a Merge Layer Info object where the info shall be copied to.
Return value	integer32	0 : OK <0 : ERROR (code)

Function	PRP_T_set_merge_layer_info	
Description	Sets the Merge Layer information parameters to the values passed as argument.	
Visibility	public	
Arguments	PRP_MergeLayerInfo_T* merge_layer_info	Pointer to a Merge Layer Info object from where the info shall be copied from.
Return value	integer32	0 : OK <0 : ERROR (code)

Function	PRP_T_go_to_first_node_table_entry	
Description	Goes to the first node in the node table, has to be called before go_to_next_node_table_entry	
Visibility	public	
Arguments	void	-
Return value	integer32	0 : OK 1 : PRP_NODETABLE_END <0 : ERROR (code)

Function	PRP_T_go_to_next_node_table_entry	
Description	Goes to the next node in the node table.	
Visibility	public	
Arguments	void	-
Return value	integer32	0 : OK 1 : PRP_NODETABLE_END <0 : ERROR (code)

Function	PRP_T_get_node_table_entry	
Description	Copies the node info into to the structure passed as argument	
Visibility	public	
Arguments	PRP_NodeTableEntry_T* node_table_entry	Pointer to a Node Table Entry Info object where the info shall be copied to.
Return value	integer32	0 : OK <0 : ERROR (code)

Function	PRP_T_delete_node_table_entry	
Description	Deletes the node passed as argument	
Visibility	Private to the stack	
Arguments	PRP_Node_T* node	Pointer to a node structure
Return value	integer32	0 : OK <0 : ERROR (code)

3.3.2 PRP_NetItf_T_linux

The Network Interface integrates the protocol engine into the network stack and is therefore the wrapper class around the OS independent protocol engine. The NetItf must receive frames from the two Adapters and must be able to send to the two Adapters. All traffic to the two interfaces must pass this Interface. On receiving the OS depending receive function will call the API PTP_T_receive() function. In the sending path it will forward the frames coming from the upper layers to the API PTP_T_transmit() function. The protocol engine will call the PRP_NetItf_T_transmit() function.
This Interface also provides functions to change properties of the two adapters.

PRP_NetItf_T has to be adapted for a specific OS.

3.3.2.1 Attributes

There are no attributes.

3.3.2.2 Functions

Function	PRP_NetItf_T_transmit	
Description	Transmits the Frame to the actual adapter specified in the lan_id.	
Visibility	public	
Arguments	octet * data	Pointer to the beginning of the frame (dest mac)
	uinteger32* length	Length in bytes of the frame
	octet lan_id	LAN to send the frame
Return value	integer32	0 : OK <0 : ERROR (code)

Function	PRP_NetItf_T_set_supervision_address	
Description	Sets the MAC address for the supervision frames, MACs have to know the multicast mac, or they go into promiscuous mode	
Visibility	public	
Arguments	octet* mac (array[6])	Mac address to set
Return value	integer32	0 : OK <0 : ERROR (code)

Function	PRP_NetItf_T_set_mac_address_A	
Description	Sets the MAC address of the adapter A	
Visibility	public	
Arguments	octet* mac (array[6])	Mac address to set
Return value	integer32	0 : OK <0 : ERROR (code)

Function	PRP_NetItf_T_set_mac_address_B	
Description	Sets the MAC address of the adapter B	
Visibility	public	
Arguments	octet* mac (array[6])	Mac address to set
Return value	integer32	0 : OK <0 : ERROR (code)

Function	PRP_NetItf_T_set_active_A	
Description	Enable or disable adapter A	
Visibility	public	
Arguments	boolean value	TRUE activate the adapter
Return value	integer32	0 : OK <0 : ERROR (code)

Function	PRP_NetItf_T_set_active_B	
Description	Enable or disable adapter B	
Visibility	public	
Arguments	boolean value	TRUE activate the adapter
Return value	integer32	0 : OK <0 : ERROR (code)

3.3.3 PRP_LogItf_T

This interface provides log macros for outputs of the protocol engine. There are several different log levels.

PRP_LogItf_T has not to be adapted for a specific OS (but maybe for a not GNU compliant C compiler because list of arguments is not supported by all compilers).

3.3.3.1 Attributes

There are no attributes.

3.3.3.2 Functions

Function	PRP_INFOOUT	
Description	Informational output	
Visibility	public	
Arguments	const octet *format (string)	Display string
	args…	Variables etc.
Return value	integer32	>=0 : OK (nr of bytes written) <0 : ERROR (code)

Function	PRP_ERROUT	
Description	Error output	
Visibility	public	
Arguments	const octet *format (string)	Display string
	args…	Variables etc.
Return value	integer32	>=0 : OK (nr of bytes written) <0 : ERROR (code)

Function	PRP_LOGOUT	
Description	General output	
Visibility	public	
Arguments	uinteger32 level	importance
	const octet *format (string)	Display string
	args…	Variables etc.
Return value	integer32	>=0 : OK (nr of bytes written) <0 : ERROR (code)

Function	PRP_PRP_LOGOUT
Description	Protocol engine output
Visibility	public

Arguments	uinteger32 level	importance
	const octet *format (string)	Display string
	args…	Variables etc.
Return value	integer32	>=0 : OK (nr of bytes written) <0 : ERROR (code)

Function	PRP_NET_ITF_LOGOUT	
Description	Network interface output	
Visibility	public	
Arguments	uinteger32 level	importance
	const octet *format (string)	Display string
	args…	Variables etc.
Return value	integer32	>=0 : OK (nr of bytes written) <0 : ERROR (code)

3.3.4 PRP_OsItf_T

The Operating System Interface abstracts functions like malloc, free etc from the OS depending kernel functions like kmalloc, kfree etc.

PRP_OsItf_T has to be adapted for a specific OS.

3.3.4.1 Attributes

There are no attributes.

3.3.4.2 Functions

Function	prp_malloc	
Description	Allocates memory and returns a pointer to the memory region	
Visibility	public	
Arguments	uinteger32 size	Size of the memory to allocate
Return value	void *	!NULL : OK (pointer to the memory) NULL : ERROR

Function	prp_free	
Description	Frees the dynamicly allocated memory.	
Visibility	public	
Arguments	void *ptr	Pointer to the memory region
Return value	void	-

Function	prp_printf	
Description	Prints to the output of the Kernel	
Visibility	public	
Arguments	const octet *format	Display sting
	…	Arguments
Return value	integer32	>=0 : OK (nr of bytes written) <0 : ERROR (code)

Function	prp_htonl	
Description	Converts a long from host to network order	
Visibility	public	
Arguments	uinteger32 net	Long in host byte order
Return value	uinteger32	Long in network byte order

Function	prp_htons

Description	Converts a short from host to network order	
Visibility	public	
Arguments	uinteger16 net	Short in host byte order
Return value	uinteger16	Short in network byte order

Function	prp_ntohl	
Description	Converts a long from network to host order	
Visibility	public	
Arguments	uinteger16 net	Long in network byte order
Return value	integer32	Long in host byte order

Function	prp_ntohs	
Description	Converts a short from network to host order	
Visibility	public	
Arguments	uinteger16 net	Short in network byte order
Return value	uinteger16	Short in host byte order

Function	create_lock	
Description	Creates a semaphore structure	
Visibility	public	
Arguments	void* sem	Pointer to the semaphore
Return value	integer32	>=0 : OK (nr of bytes written) <0 : ERROR (code)

Function	destroy_lock	
Description	Destroys the semaphore structure	
Visibility	public	
Arguments	void* sem	Pointer to the semaphore
Return value	integer32	>=0 : OK (nr of bytes written) <0 : ERROR (code)

Function	lock_down	
Description	Blocks on the semaphore if in use	
Visibility	public	
Arguments	void* sem	Pointer to the semaphore
Return value	integer32	>=0 : OK (nr of bytes written) <0 : ERROR (code)

Function	lock_up	
Description	Unlocks the semaphore	
Visibility	public	
Arguments	void* sem	Pointer to the semaphore
Return value	integer32	>=0 : OK (nr of bytes written) <0 : ERROR (code)

Function	prp_memcpy	
Description	Copies size bytes from src_ptr to dest_ptr.	
Visibility	public	
Arguments	void *dest_ptr	Destination
	const void *src_ptr	Source
	uinteger32 size	Number of bytes to copy
Return value	integer32	>=0 : OK (nr of bytes copied) <0 : ERROR (code)

Function	prp_memcmp	
Description	Compares size bytes from left_ptr with right_ptr.	
Visibility	public	
Arguments	const void * left _ptr	Left pointer
	const void * right _ptr	Right pointer
	uinteger32 size	Number of bytes to compare
Return value	integer32	>0 bigger =0 equal <0 less

Function	prp_memset	
Description	Set size values to the set value,	
Visibility	public	
Arguments	void * ptr	Destination
	Octet set	Octet to set
	uinteger32 size	Number of bytes to set
Return value	integer32	>=0 : OK (nr of bytes copied) <0 : ERROR (code)

3.4 Core (OS independent)

3.4.1 PRP_Environment_T

The Environment is the central part of the protocol engine. It distributes function calls coming from the API to the respective classes. It also runs the timers.

3.4.1.1 Attributes

Attribute	supervise_timer_
Description	Instance of the timer to invoke the check of the Node table for timed out nodes.
Visibility	public
Type	PRP_Timer_T

Attribute	bridging_timer_
Description	Instance of the timer to invoke the check of the link, for bridging.
Visibility	public
Type	PRP_Timer_T

Attribute	supervision_tx_timer_
Description	Instance of the timer to invoke the sending of the supervision frame.
Visibility	public
Type	PRP_Timer_T

Attribute	environment_configuration_
Description	Instance of the environment configuration.
Visibility	public
Type	PRP_EnvironmentConfiguration_T

Attribute	supervision_
Description	Instance of the supervision part of the protocol engine.
Visibility	public
Type	PRP_SupervisionFrame_T

Attribute	bridging_
Description	Instance of the bridging part of the protocol engine.
Visibility	public
Type	PRP_SupervisionFrame_T

Attribute	node_table_
Description	Instance of the node table.
Visibility	public
Type	PRP_NodeTable_T

Attribute	discard_algorithm_
Description	Instance of the discard algorithm part of the protocol engine.
Visibility	public
Type	PRP_DiscardAlgorithm_T

Attribute	frame_analyser_
Description	Instance of the frame analyzer.
Visibility	public

Type	PRP_FrameAnalyser_T

3.4.1.2 Functions

Function	PRP_Evironment_T_process_rx	
Description	Forwards the API call receive to the frame analyser	
Visibility	public	
Arguments	PRP_Environment_T* const me	This pointer
	octet * data	Pointer to the beginning of the frame (dest mac)
	uinteger32* length	Length in bytes of the frame
	octet lan_id	On which LAN it was received
Return value	integer32	1 : DROP 0 : KEEP <0 : ERROR (code)

Function	PRP_Evironment_T_process_tx	
Description	Forwards the API call transmit to the frame analyser	
Visibility	public	
Arguments	PRP_Environment_T* const me	This pointer
	octet * data	Pointer to the beginning of the frame (dest mac)
	uinteger32* length	Length in bytes of the frame
Return value	integer32	0 : OK <0 : ERROR (code)

Function	PRP_Evironment_T_process_timer	
Description	Runs the timers and if the timer expired calls the respective supervision function	
Visibility	public	
Arguments	PRP_Environment_T* me	This pointer
Return value	void	-

3.4.2 PRP_EnvironmentConfiguration_T

The Environment Configuration contains general information about the protocol engine and its environment.

3.4.2.1 Attributes

Attribute	node_
Description	Name of the node
Visibility	public
Type	char[32]

Attribute	manufacturer_
Description	Name of the manufacturer
Visibility	public
Type	char[256]

Attribute	version_
Description	Version of the implementation
Visibility	public
Type	uinteger16

Attribute	mac_address_A_
Description	Mac address of adapter A
Visibility	public
Type	octet[6]

Attribute	mac_address_B_
Description	Mac address of adapter B
Visibility	public
Type	octet[6]

Attribute	adapter_active_A_
Description	Status of adapter A
Visibility	public
Type	boolean

Attribute	adapter_active_B_
Description	Status of adapter B
Visibility	public
Type	boolean

Attribute	duplicate_discard_
Description	Duplicate discard modus enabled?
Visibility	public
Type	boolean

Attribute	transparent_reception_
Description	Transparent reception modus enabled?
Visibility	public
Type	boolean

Attribute	bridging_
Description	Bridging modus enabled?
Visibility	public
Type	boolean

Attribute	cnt_total_sent_A_
Description	Total number of frames sent over adapter A
Visibility	public
Type	uinteger32

Attribute	cnt_total_sent_B_
Description	Total number of frames sent over adapter B
Visibility	public
Type	uinteger32

Attribute	cnt_total_received_A_
Description	Total number of frames received over adapter A
Visibility	public
Type	uinteger32

Attribute	cnt_total_received_B_
Description	Total number of frames received over adapter B
Visibility	public

Type	uinteger32

Attribute	cnt_total_errors_A_
Description	Total number of errors on adapter A
Visibility	public
Type	uinteger32

Attribute	cnt_total_errors_B_
Description	Total number of errors on adapter B
Visibility	public
Type	uinteger32

3.4.2.2 Functions

There are no functions.

3.4.3 PRP_NodeTable_T

The Node Table consists of four sub classes, PRP_NodeTable_T, PRP_Node_T, PRP_DropWindowTable_T and PRP_DropWindow_T. This class provides functionality to search, add, delete, etc. nodes. The table is designed as a double linked, unsorted list. Therefore a search is always a linear search trough the table. The design of the Node table can easily be changed without affecting the rest of the stack.

3.4.3.1 Attributes

Attribute	first_node_
Description	Pointer to the first node. If NULL, table is empty.
Visibility	public
Type	PRP_Node_T*

Attribute	last_node_
Description	Pointer to the last node. If NULL, table is empty.
Visibility	public
Type	PRP_Node_T*

Attribute	node_table_empty_
Description	Shows whether table is empty or not.
Visibility	public
Type	boolean

Attribute	cnt_nodes
Description	Number of nodes in the table.
Visibility	public
Type	uinteger32

3.4.3.2 Functions

Function	PRP_NodeTable_T_get_node	
Description	Searches for a node in the table which matches the mac address passed as argument. If found returns a pointer to the Node object else NULL.	
Visibility	public	
Arguments	PRP_NodeTable_T* const me	This pointer
	octet* mac (array[6])	Mac address to search for.
Return value	PRP_Node_T *	!NULL : OK (pointer to the memory)

		NULL : ERROR (or there is no node with this mac)

Function	PRP_NodeTable_T_add_node	
Description	Creates a new node with the values passed as arguments. Returns a pointer to the new Node object.	
Visibility	public	
Arguments	PRP_NodeTable_T* const me	This pointer
	PRP_Node_T * node	Node to add to the table. Mac address A or B in the node structure are mandatory.
Return value	PRP_Node_T *	!NULL : OK (pointer to the memory of the new added node) NULL : ERROR

Function	PRP_NodeTable_T_remove_node	
Description	Removes the node passed as argument from the table.	
Visibility	public	
Arguments	PRP_NodeTable_T* const me	This pointer
	PRP_Node_T * node	Pointer to the node to remove.
Return value	void	-

Function	PRP_NodeTable_T_get_first_node	
Description	If table is not empty returns a pointer to the first Node object in the table, else NULL.	
Visibility	public	
Arguments	PRP_NodeTable_T* const me	This pointer
Return value	PRP_Node_T *	!NULL : OK (pointer to the memory) NULL : ERROR (or table is empty)

Function	PRP_NodeTable_T_get_last_node	
Description	If table is not empty returns a pointer to the last Node object in the table, else NULL.	
Visibility	public	
Arguments	PRP_NodeTable_T* const me	This pointer
Return value	PRP_Node_T *	!NULL : OK (pointer to the memory) NULL : ERROR (or table is empty)

Function	PRP_NodeTable_T_get_next_node	
Description	Returns a pointer to the next Node object from the current object, passed as argument. Node that is passed has to be received by previously call get_first or get_last node (and call get_next or get_previous node)	
Visibility	public	
Arguments	PRP_NodeTable_T* const me	This pointer
	PRP_Node_T * node	Pointer to a node
Return value	PRP_Node_T *	!NULL : OK (pointer to the memory) NULL : ERROR (or end of table)

Function	PRP_NodeTable_T_get_previous_node	
Description	Returns a pointer to the previous Node object from the current object, passed as argument. Node that is passed has to be received by previously call get_first or get_last node (and call get_next or get_previous node)	
Visibility	public	
Arguments	PRP_NodeTable_T* const me	This pointer
	PRP_Node_T * node	Pointer to a node
Return value	PRP_Node_T *	!NULL : OK (pointer to the memory) NULL : ERROR (or start of table)

3.4.4 PRP_Node_T

The Node class contains general information about a remote node. It also has an instance of a PRP_DropWindowTable_T. It is an entry of the node table.

3.4.4.1 Attributes

Attribute	previous_node_
Description	Pointer to the previous node. If NULL, start of the table.
Visibility	public
Type	PRP_Node_T*

Attribute	next_node_
Description	Pointer to the next node. If NULL, end of the table.
Visibility	public
Type	PRP_Node_T*

Attribute	busy_
Description	If >0 it is use by some application that reads the node table over the API. As long as it is >0 the node gets not deleted by the supervision.
Visibility	uinteger32
Type	PRP_Node_T*

Attribute	mac_address_A_
Description	Mac address of adapter A of the remote node
Visibility	public
Type	octet[6]

Attribute	mac_address_B_
Description	Mac address of adapter B of the remote node
Visibility	public
Type	octet[6]

Attribute	cnt_received_A_
Description	Total number of frames received from this node over adapter A.
Visibility	public
Type	uinteger32

Attribute	cnt_received_B_
Description	Total number of frames received from this node over adapter B.
Visibility	public

Type	uinteger32

Attribute	cnt_sent_A_
Description	Total number of frames sent to this node over adapter A.
Visibility	public
Type	PRP_Node_T*

Attribute	cnt_sent_B_
Description	Total number of frames sent to this node over adapter B.
Visibility	public
Type	PRP_Node_T*

Attribute	cnt_keept_A_
Description	Total number of frames kept from this node which were received over adapter A.
Visibility	public
Type	PRP_Node_T*

Attribute	cnt_keept_B_
Description	Total number of frames kept from this node which were received over adapter B.
Visibility	public
Type	PRP_Node_T*

Attribute	cnt_err_out_of_sequence_A_
Description	Total number of out of sequence errors from this node received on adapter A.
Visibility	public
Type	PRP_Node_T*

Attribute	cnt_err_out_of_sequence_B_
Description	Total number of out of sequence errors from this node received on adapter B.
Visibility	public
Type	PRP_Node_T*

Attribute	cnt_err_wrong_lan_A_
Description	Total number of wrong LAN errors from this node received on adapter A.
Visibility	public
Type	uinteger32

Attribute	cnt_err_wrong_lan_B_
Description	Total number of wrong LAN errors from this node received on adapter B.
Visibility	public
Type	uinteger32

Attribute	time_last_seen_A_
Description	Time when the last frame was received from this node over LAN A
Visibility	public
Type	time

Attribute	time_last_seen_B_
Description	Time when the last frame was received from this node over LAN B
Visibility	public
Type	time

Attribute	san_A_
Description	TRUE if this node is a SAN on LAN A
Visibility	public
Type	boolean

Attribute	san_B_
Description	TRUE if this node is a SAN on LAN B
Visibility	public
Type	boolean

Attribute	send_seq_
Description	Sending sequence number for this node
Visibility	public
Type	uinteger16

Attribute	drop_window_table_
Description	Instance of a drop window table with all multicast and unicast MAC drop windows.
Visibility	public
Type	PRP_DropWindowTable_T

Attribute	multi_broadcast_
Description	Shows whether the node is a real node or a multicast destination.
Visibility	public
Type	boolean

Attribute	failed_
Description	Indicates whether no frame was received for link_timeout_time
Visibility	public
Type	boolean

Attribute	failed_A_
Description	Indicates whether no frame was received for link_timeout_time on LAN A
Visibility	public
Type	boolean

Attribute	failed_B_
Description	Indicates whether no frame was received for link_timeout_time on LAN B
Visibility	public
Type	boolean

3.4.4.2 Functions

There are no functions.

3.4.5 PRP_Lock_T

This class provides the functions to create locked sections.

3.4.5.1 Attributes

Attribute	lock_
Description	Pointer to the first drop window in the table. If NULL table is empty.
Visibility	public
Type	void*

3.4.5.2 Functions

Function	PRP_Lock_T_down	
Description	Takes the lock, this is blocking if already locked	
Visibility	public	
Arguments	PRP_Lock_T* const me	This pointer
Return value	-	

Function	PRP_Lock_T_up	
Description	Releases the lock, this is non-blocking	
Visibility	public	
Arguments	PRP_Lock_T* const me	This pointer
Return value	-	

3.4.5.3 Functions

There are no functions.

3.4.6 PRP_Supervision_T

The Supervision class is one of the core classes it processes received Supervision frames as well as it send periodically Supervision by itself. This class also does the Supervision of the Node Table.

3.4.6.1 Attributes

Attribute	environment_
Description	Pointer to the environment object.
Visibility	public
Type	PRP_Environment_T*

Attribute	supervision_frame_
Description	Space for a supervision frame.
Visibility	private
Type	octet[64]

Attribute	link_time_out_
Description	Time when the link is considered down.
Visibility	public
Type	uinteger16

Attribute	node_forget_time_
Description	Time when a node is considered down and should be removed from the node table
Visibility	public
Type	uinteger16

Attribute	life_check_interval_
Description	Interval to check for the existence of nodes.
Visibility	public

Type	uinteger16

Attribute	supervision_address_
Description	Destination address to send supervision frames to.
Visibility	public
Type	octet[6]

3.4.6.2 Functions

Function	PRP_ Supervision _T_ supervision_rx	
Description	Receives a supervision frame and does the actual processing. Check 3.5.3 for more details.	
Visibility	public	
Arguments	PRP_Supervision_T* const me	This pointer
	octet * data	Pointer to the beginning of the frame (dest mac)
	uinteger32* length	Length in bytes of the frame
	octet lan_id	On which LAN it was received
Return value	integer32	0 : OK <0 : ERROR (code)

Function	PRP_ Supervision _T_ supervision_tx	
Description	Transmission of a supervision frame. Check 3.5.4 for more details.	
Visibility	public	
Arguments	PRP_Supervision_T* const me	This pointer
Return value	integer32	0 : OK <0 : ERROR (code)

Function	PRP_ Supervision _T_ supervise	
Description	Supervision of the node table. Check 3.5.5 for more details	
Visibility	public	
Arguments	PRP_Supervision_T* const me	This pointer
Return value	integer32	0 : OK <0 : ERROR (code)

3.4.7 PRP_SupervisionFrame_T

This is an instance of a Supervision frame. It provides a function to set some specific field in the frame.

3.4.7.1 Attributes

Attribute	prp_version_
Description	PRP version : 0x0000
Visibility	public
Type	uinteger16

Attribute	type_
Description	Mode: 0x20 for Duplicate Discard, 0x21 for Duplicate Accept.
Visibility	public
Type	uinteger8

Attribute	length_
Description	LSPDU: 12
Visibility	public
Type	uinteger8

Attribute	source_mac_A_
Description	Source mac of adapter A of the local node
Visibility	public
Type	octet[6]

Attribute	source_mac_B_
Description	Source mac of adapter B of the local node
Visibility	public
Type	octet[6]

3.4.7.2 Functions

There are no functions.

3.4.8 PRP_Bridging_T

The Bridiging class is one of the core classes it processes received RSTP-BPDU frames as well as it answers them. It implements the SRP

3.4.8.1 Attributes

Attribute	environment_
Description	Pointer to the environment object.
Visibility	public
Type	PRP_Environment_T*

Attribute	state_A_
Description	state for the statemachine
Visibility	private
Type	uinteger8

Attribute	state_B_
Description	state for the statemachine
Visibility	public
Type	uinteger8

Attribute	time_last_seen_A_
Description	Last time a BPDU was received on LAN A
Visibility	public
Type	uinteger64

Attribute	time_last_seen_B_
Description	Last time a BPDU was received on LAN B
Visibility	public
Type	uinteger64

Attribute	full_path_costs_
Description	Path costs to the root, updates with every frame
Visibility	public
Type	octet[22]

Attribute	topologie_change_A_
Description	Flag for the statemachine
Visibility	public
Type	boolean

Attribute	topologie_change_B_
Description	Flag for the statemachine
Visibility	public
Type	boolean

3.4.8.2 Functions

Function	PRP_Bridging_T_bridging_rx	
Description	Receives a BPDU and does the actual processing.	
Visibility	public	
Arguments	PRP_Bridging_T* const me	This pointer
	octet * data	Pointer to the beginning of the frame (dest mac)
	uinteger32* length	Length in bytes of the frame
	octet lan_id	On which LAN it was received
Return value	integer32	0 : OK <0 : ERROR (code)

Function	PRP_Bridging_T_bridging_tx	
Description	Transmission of a supervision frame. Check 3.5.4 for more details.	
Visibility	public	
Arguments	PRP_Bridging_T* const me	This pointer
	PRP_BridgingFrame_T* bridging_frame_payload	Frame to send
	octet lan_id	LAN id
Return value	integer32	0 : OK <0 : ERROR (code)

Function	PRP_Bridging_T_supervise	
Description	Supervision of the received BPDU, check for timeout	
Visibility	public	
Arguments	PRP_Bridging_T* const me	This pointer
Return value	integer32	0 : OK <0 : ERROR (code)

Function	PRP_Bridging_T_statemachine	
Description	Processes the events to the statemachine	
Visibility	private	
Arguments	PRP_Bridging_T* const me	This pointer
	octet event	Event type
	void* event_data	Data that comes with this event
	octet lan_id	On which LAN it was received
Return value	integer32	0 : OK <0 : ERROR (code)

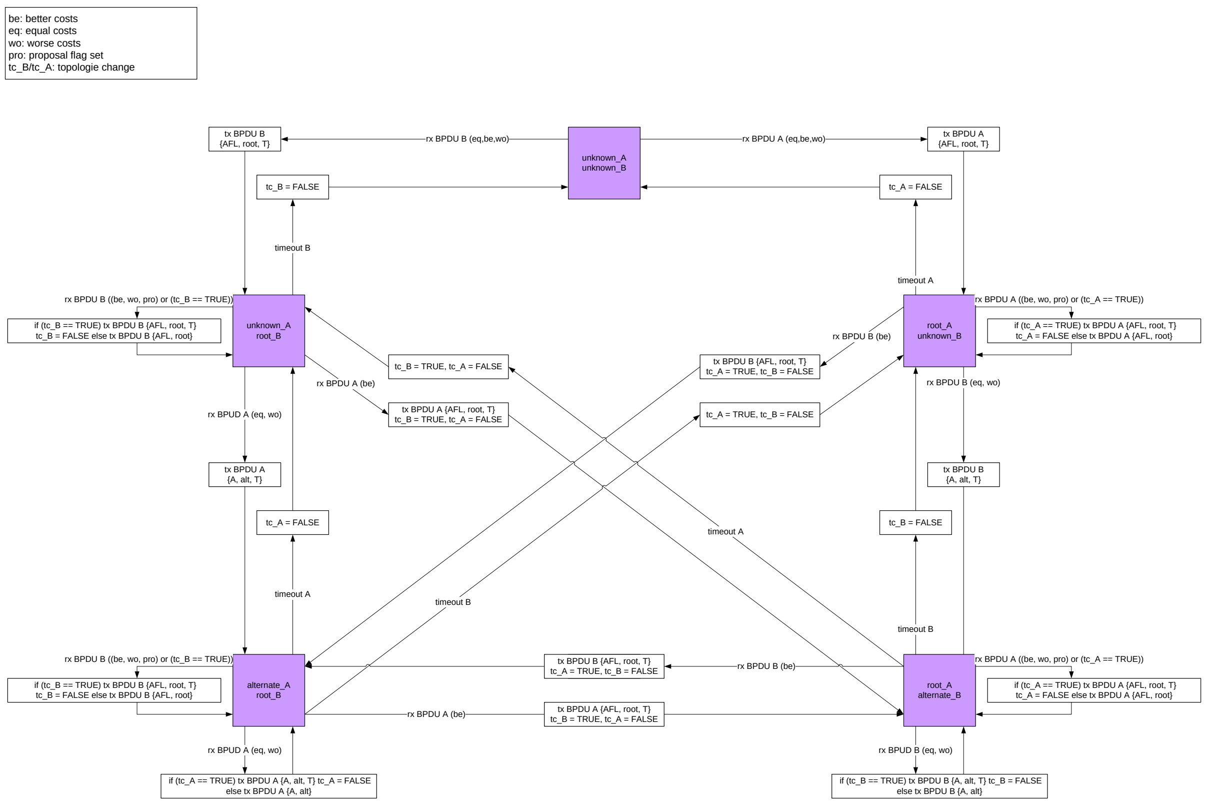
be: better costs
eq: equal costs
wo: worse costs
pro: proposal flag set
tc_B/tc_A: topologie change
unknown_A unknown_B
rx BPDU B (eq,be,wo)
rx BPDU A (eq,be,wo)
tx BPDU B {AFL, root, T}
tx BPDU A {AFL, root, T}
tc_B = FALSE
tc_A = FALSE
timeout B
timeout A
rx BPDU B ((be, wo, pro) or (tc_B == TRUE))
rx BPDU A ((be, wo, pro) or (tc_A == TRUE))
if (tc_B == TRUE) tx BPDU B {AFL, root, T} tc_B = FALSE else tx BPDU B {AFL, root}
if (tc_A == TRUE) tx BPDU A {AFL, root, T} tc_A = FALSE else tx BPDU A {AFL, root}
unknown_A root_B
root_A unknown_B
tc_B = TRUE, tc_A = FALSE
tx BPDU B {AFL, root, T} tc_A = TRUE, tc_B = FALSE
rx BPDU B (be)
rx BPDU A (be)
tx BPDU A {AFL, root, T} tc_B = TRUE, tc_A = FALSE
tc_A = TRUE, tc_B = FALSE
rx BPUD A (eq, wo)
rx BPDU B (eq, wo)
tx BPDU A {A, alt, T}
tx BPDU B {A, alt, T}
tc_A = FALSE
tc_B = FALSE
timeout A
timeout B
timeout A
timeout B
alternate_A root_B
root_A alternate_B
tx BPDU B {AFL, root, T} tc_A = TRUE, tc_B = FALSE
rx BPDU B (be)
tx BPDU A {AFL, root, T} tc_B = TRUE, tc_A = FALSE
rx BPDU A (be)
rx BPUD A (eq, wo)
rx BPUD B (eq, wo)
if (tc_A == TRUE) tx BPDU A {A, alt, T} tc_A = FALSE else tx BPDU A {A, alt}
if (tc_B == TRUE) tx BPDU B {A, alt, T} tc_B = FALSE else tx BPDU B {A, alt}

Figure 4: SRP statemachine

3.4.9 PRP_BridgingFrame_T

This is an instance of a Supervision frame. It provides a function to set some specific field in the frame.

3.4.9.1 Attributes

Attribute	flags_
Description	Flags
Visibility	public
Type	uinteger8

Attribute	root_prio_and_id_
Description	Root priority and ID
Visibility	public
Type	uinteger16

Attribute	root_mac_
Description	Root's mac adress
Visibility	public
Type	Octet[6]

Attribute	root_path_costs_
Description	Path costs to root
Visibility	public
Type	uinteger32

Attribute	bridge_prio_and_id_
Description	Priority and ID of the Brdige who sent frame
Visibility	public
Type	uinteger16

Attribute	bridge_mac_
Description	The bridge’s mac address
Visibility	public
Type	octet[6]

Attribute	port_prio_and_id_
Description	Port where the message was sent
Visibility	public
Type	uinteger16

Attribute	message_age_
Description	Age of the message
Visibility	public
Type	uinteger16

Attribute	max_age_
Description	Maximum age
Visibility	public
Type	uinteger16

Attribute	hello_time_
Description	How often a BPDU is sent
Visibility	public
Type	uinteger16

Attribute	v1_length_
Description	For STP
Visibility	public
Type	uinteger8

3.4.9.2 Functions

There are no functions.

3.4.10 PRP_DiscardAlgorithm_T

This class is another core class it implements the discard algorithm specified in the standard. The decision whether to keep a frame or not is done here.

3.4.11 PRP_FrameAnalyser_T

The Frame Analyzer checks the frame type at forwards the frame to the respective frame handler.

3.4.11.1 Attributes

Attribute	environment_
Description	Pointer to the environment
Visibility	public
Type	PRP_Environment_T*

Attribute	frames_
Description	Instance of a frames object.

Visibility	public
Type	PRP_Frames_T

3.4.11.2 Functions

Function	PRP_FrameAnalyser_T_analyse_rx	
Description	Checks the frame type and forwards it to the respective frame handler for the receive path.	
Visibility	public	
Arguments	PRP_FrameAnalyser_T* const me	This pointer
	octet * data	Pointer to the beginning of the frame (dest mac)
	uinteger32* length	Length in bytes of the frame
	octet lan_id	On which LAN it was received
Return value	integer32	1 : DROP 0 : KEEP <0 : ERROR (code)

Function	PRP_FrameAnalyser_T_analyse_tx	
Description	Checks the frame type and forwards it to the respective frame handler for the transmit path.	
Visibility	public	
Arguments	PRP_FrameAnalyser_T* const me	This pointer
	octet * data	Pointer to the beginning of the frame (dest mac)
	uinteger32* length	Length in bytes of the frame
Return value	integer32	0 : OK <0 : ERROR (code)

3.4.12 PRP_Frames_T

The Frames class is the core class of the whole protocol engine. It adds/removes the RCT gets/adds nodes and drop windows to the Node Table, runs the discard algorithm, etc.

3.4.12.1 Attributes

Attribute	frame_analyser_
Description	Pointer to the frame analyser
Visibility	public
Type	PRP_FrameAnalyser_T*

Attribute	trailer_
Description	Instance of a trailer object.
Visibility	public
Type	PRP_Trailer_T

3.4.12.2 Functions

Function	PRP_Frames_T_normal_rx	
Description	Actual processing of a received frame. Check 3.5.1 for more details.	
Visibility	public	
Arguments	PRP_Frames_T* const me	This pointer
	octet * data	Pointer to the beginning of the frame (dest mac)
	uinteger32* length	Length in bytes of the frame

	octet lan_id	On which LAN it was received
Return value	integer32	1 : DROP 0 : KEEP <0 : ERROR (code)

Function	PRP_Frames_T_ptp_rx	
Description	Actual processing of a received PTP frame. TODO	
Visibility	public	
Arguments	PRP_Frames_T* const me	This pointer
	octet * data	Pointer to the beginning of the frame (dest mac)
	uinteger32* length	Length in bytes of the frame
	octet lan_id	On which LAN it was received
Return value	integer32	1 : DROP 0 : KEEP <0 : ERROR (code)

Function	PRP_Frames_T_normal_tx	
Description	Actual processing of a transmitting frame. Check 3.5.2 for more details.	
Visibility	public	
Arguments	PRP_Frames_T* const me	This pointer
	octet * data	Pointer to the beginning of the frame (dest mac)
	uinteger32* length	Length in bytes of the frame
Return value	integer32	0 : OK <0 : ERROR (code)

Function	PRP_Frames_T_ptp_tx	
Description	Actual processing of a transmitting PTP frame. TODO	
Visibility	public	
Arguments	PRP_Frames_T* const me	This pointer
	octet * data	Pointer to the beginning of the frame (dest mac)
	uinteger32* length	Length in bytes of the frame
Return value	integer32	0 : OK <0 : ERROR (code)

Function	PRP_Frames_T_replace_src_mac	
Description	Replaces the source mac address in the frame with the mac passed as argument.	
Visibility	private	
Arguments	PRP_Frames_T* const me	This pointer
	octet * data	Pointer to the beginning of the frame (dest mac)
	uinteger32* length	Length in bytes of the frame
	octet* mac (array[6])	Mac address to set
Return value	integer32	0 : OK <0 : ERROR (code)

Function	PRP_Frames_T_replace_dest_mac	
Description	Replaces the destination mac address in the frame with the mac passed as argument.	
Visibility	private	
Arguments	PRP_Frames_T* const me	This pointer

	octet * data	Pointer to the beginning of the frame (dest mac)
	uinteger32* length	Length in bytes of the frame
	octet* mac (array[6])	Mac address to set
Return value	integer32	0 : OK <0 : ERROR (code)

3.4.13 PRP_Trailer_T

The Trailer class provides functions to add/remove/get the RCT to/from the frame.

3.4.13.1 Attributes

Attribute	redundancy_control_trailer_
Description	Instance of a redundancy control trailer
Visibility	public
Type	PRP_RedundancyControlTrailer_T

3.4.13.2 Functions

Function	PRP_Trailer_T_get_trailer	
Description	Extracts the Trailer out the frame, and fills it into a structure in host byte order. Returns a Pointer to a RCT object if there is a Trailer, else NULL.	
Visibility	public	
Arguments	PRP_Trailer_T* const me	This pointer
	octet * data	Pointer to the beginning of the frame (dest mac)
	uinteger32* length	Length in bytes of the frame
Return value	PRP_RedundancyControlTrailer_T*	!NULL : OK (pointer to the memory) NULL : ERROR (or no trailer)

Function	PRP_Trailer_T_add_trailer	
Description	Adds a Trailer to the frame. Changes the values passed as argument into network byte order and changes the field length etc.	
Visibility	public	
Arguments	PRP_Trailer_T* const me	This pointer
	octet * data	Pointer to the beginning of the frame (dest mac)
	uinteger32* length	Length in bytes of the frame
	PRP_RedundancyControlTrailer_T* trailer	
Return value	void	-

Function	PRP_Trailer_T_remove_trailer	
Description	Removes a RCT from a frame, if there is one.	
Visibility	public	
Arguments	PRP_Trailer_T* const me	This pointer
	octet * data	Pointer to the beginning of the frame (dest mac)
	uinteger32* length	Length in bytes of the frame
Return value	void	-

3.4.14 PRP_RedundancyControlTrailer_T

This is an instance of an RCT.

3.4.14.1 Attributes

Attribute	seq_
Description	Sequence number.
Visibility	public
Type	uinteger16

Attribute	lan_id_
Description	LAN identifier.
Visibility	public
Type	octet

Attribute	size_
Description	Size of the LSPDU.
Visibility	public
Type	uinteger16

3.4.14.2 Functions

Function	PRP_RedundancyControlTrailer_T_print	
Description	Prints the redundancy control trailer.	
Visibility	public	
Arguments	PRP_RedundancyControlTrailer_T* const me	This pointer
Return value	void	-

3.4.15 PRP_Timer_T

The Timer class is to run scheduled task like Supervision frame transmission or supervision of the Node Table.

3.4.15.1 Attributes

Attribute	enabled_
Description	If a tick decrements the timer.
Visibility	public
Type	boolean

Attribute	timer_
Description	Current tick of the timer
Visibility	public
Type	Uinteger64

3.4.15.2 Functions

Function	PRP_Timer_T_stop	
Description	Stops a running timer.	
Visibility	public	
Arguments	PRP_Timer_T* const me	This pointer
Return value	void	-

Function	PRP_Timer_T_start	
Description	Starts or restarts a timer.	
Visibility	public	
Arguments	PRP_Timer_T* const me	This pointer

	uinteger64 timeout	Timeout in nanoseconds
Return value	void	-

Function	PRP_Timer_T_tick	
Description	Decrements the value_ variable until timeout happened.	
Visibility	public	
Arguments	PRP_Timer_T* const me	This pointer
Return value	boolean	TRUE : Timer expired FALSE : Timer decremented

3.5 UML Sequence Diagrams

3.5.1 Normal frame receiving

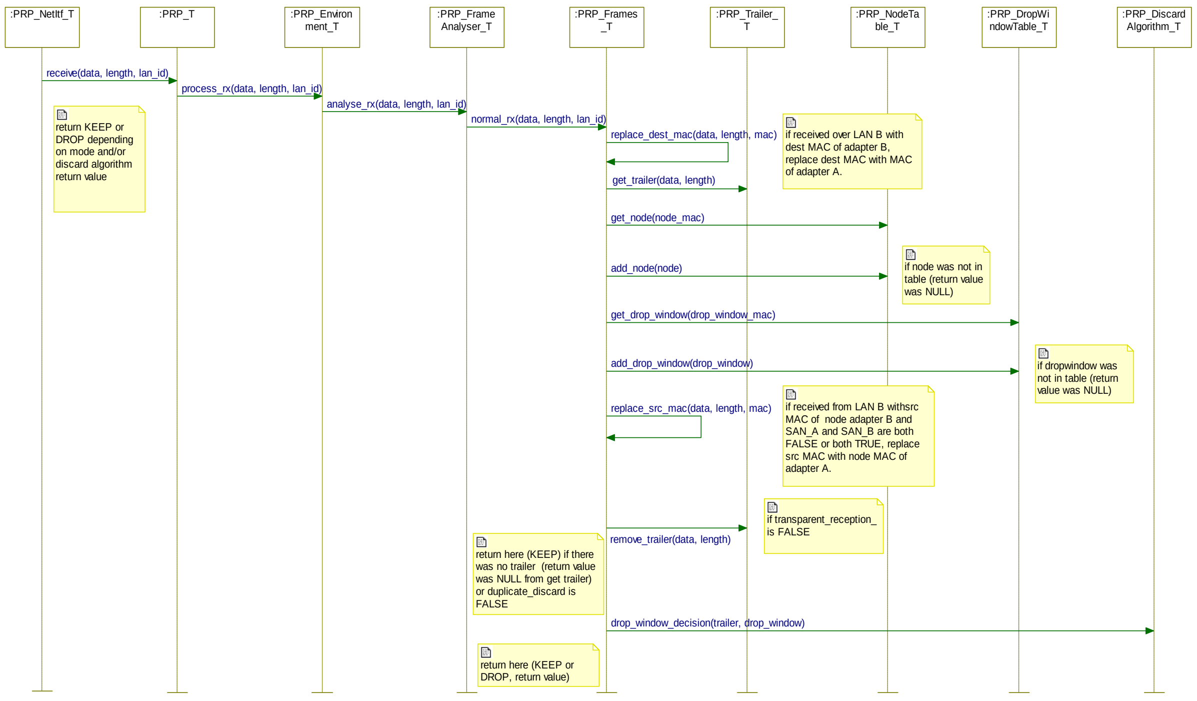
:PRP_NetItf_T
:PRP_T
:PRP_Environment_T
:PRP_FrameAnalyser_T
:PRP_Frames_T
:PRP_Trailer_T
:PRP_NodeTable_T
:PRP_DropWindowTable_T
:PRP_DiscardAlgorithm_T
receive(data, length, lan_id)
process_rx(data, length, lan_id)
analyse_rx(data, length, lan_id)
normal_rx(data, length, lan_id)
return KEEP or DROP depending on mode and/or discard algorithm return value
replace_dest_mac(data, length, mac)
if received over LAN B with dest MAC of adapter B, replace dest MAC with MAC of adapter A.
get_trailer(data, length)
get_node(node_mac)
add_node(node)
if node was not in table (return value was NULL)
get_drop_window(drop_window_mac)
add_drop_window(drop_window)
if dropwindow was not in table (return value was NULL)
replace_src_mac(data, length, mac)
if received from LAN B withsrc MAC of node adapter B and SAN_A and SAN_B are both FALSE or both TRUE, replace src MAC with node MAC of adapter A.
if transparent_reception_ is FALSE
remove_trailer(data, length)
return here (KEEP) if there was no trailer (return value was NULL from get trailer) or duplicate_discard is FALSE
drop_window_decision(trailer, drop_window)
return here (KEEP or DROP, return value)

Figure 5: Reception of a normal frame

3.5.2 Normal frame transmitting

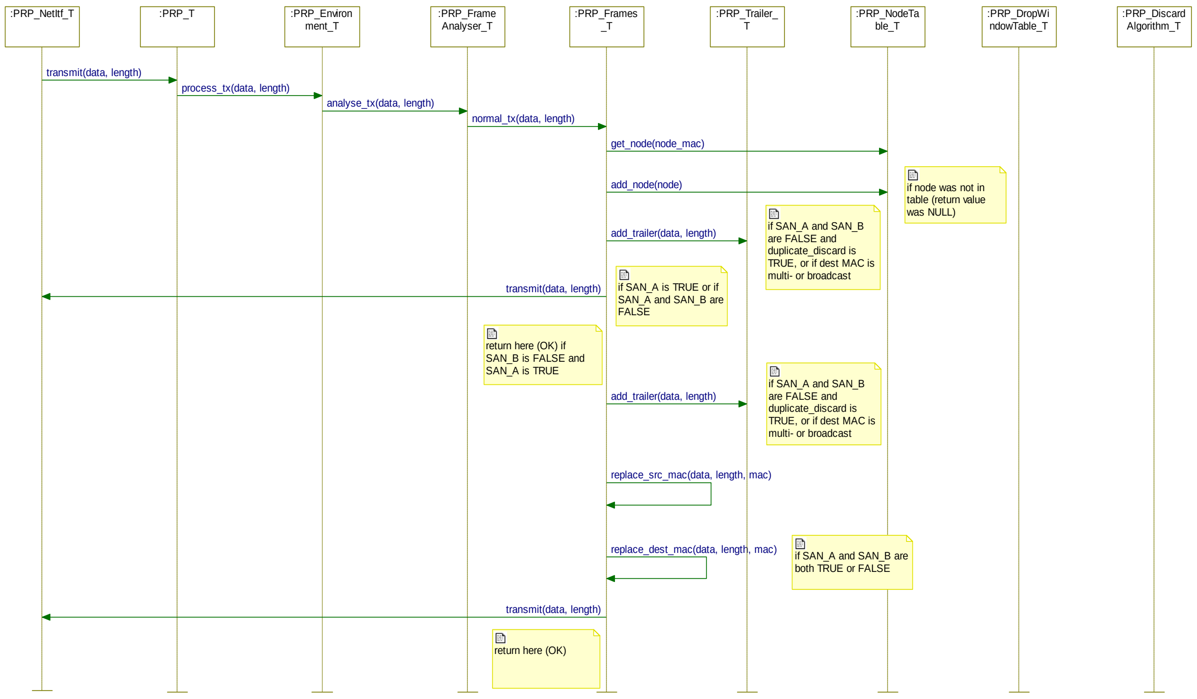
:PRP_NetItf_T
:PRP_T
:PRP_Environment_T
:PRP_FrameAnalyser_T
:PRP_Frames_T
:PRP_Trailer_T
:PRP_NodeTable_T
:PRP_DropWindowTable_T
:PRP_DiscardAlgorithm_T
transmit(data, length)
process_tx(data, length)
analyse_tx(data, length)
normal_tx(data, length)
get_node(node_mac)
add_node(node)
if node was not in table (return value was NULL)
add_trailer(data, length)
if SAN_A and SAN_B are FALSE and duplicate_discard is TRUE, or if dest MAC is multi- or broadcast
transmit(data, length)
if SAN_A is TRUE or if SAN_A and SAN_B are FALSE
return here (OK) if SAN_B is FALSE and SAN_A is TRUE
add_trailer(data, length)
if SAN_A and SAN_B are FALSE and duplicate_discard is TRUE, or if dest MAC is multi- or broadcast
replace_src_mac(data, length, mac)
replace_dest_mac(data, length, mac)
if SAN_A and SAN_B are both TRUE or FALSE
transmit(data, length)
return here (OK)

Figure 6: Transmission of a normal frame

3.5.3 Supervision frame receiving

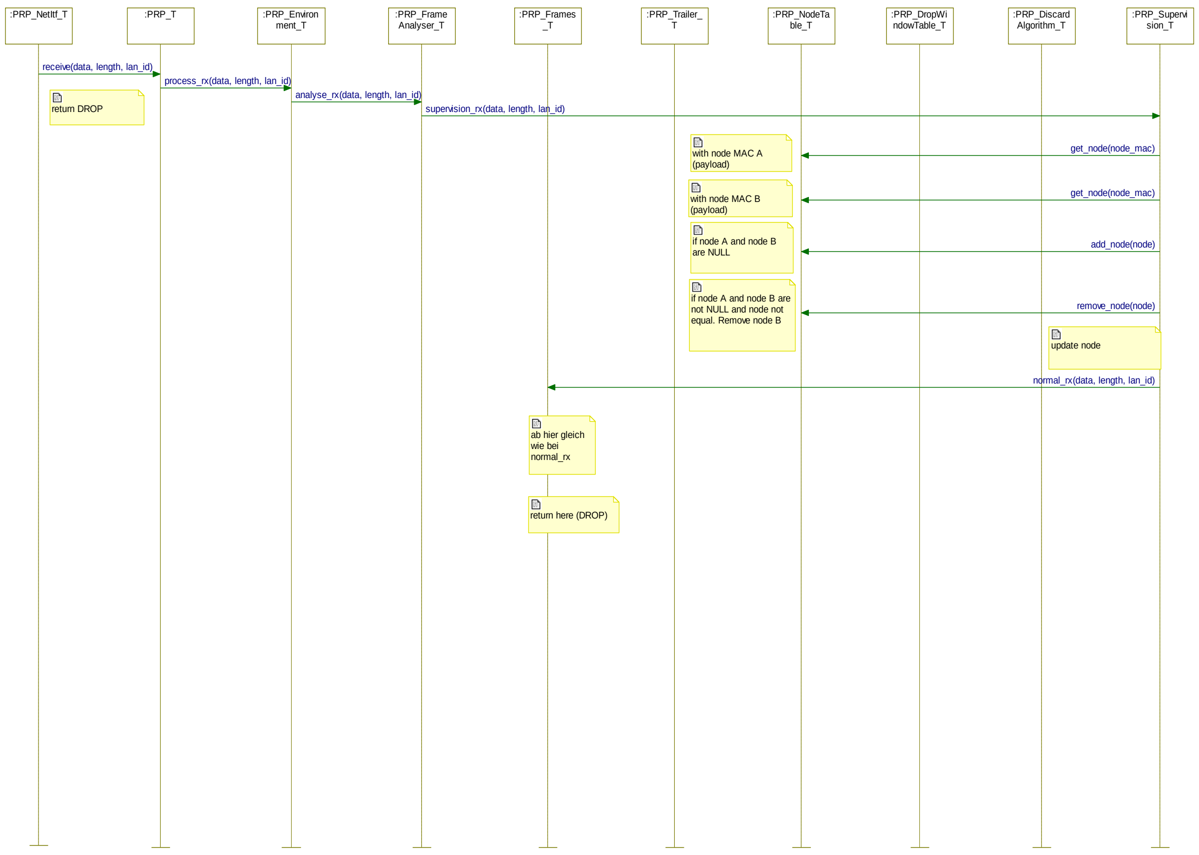
:PRP_NetItf_T
:PRP_T
:PRP_Environment_T
:PRP_FrameAnalyser_T
:PRP_Frames_T
:PRP_Trailer_T
:PRP_NodeTable_T
:PRP_DropWindowTable_T
:PRP_DiscardAlgorithm_T
:PRP_Supervision_T
receive(data, length, lan_id)
process_rx(data, length, lan_id)
analyse_rx(data, length, lan_id)
return DROP
supervision_rx(data, length, lan_id)
get_node(node_mac)
with node MAC A (payload)
get_node(node_mac)
with node MAC B (payload)
add_node(node)
if node A and node B are NULL
remove_node(node)
if node A and node B are not NULL and node not equal. Remove node B
update node
normal_rx(data, length, lan_id)
ab hier gleich wie bei normal_rx
return here (DROP)

Figure 7: Reception of a supervision frame

3.5.4 Supervision frame transmitting

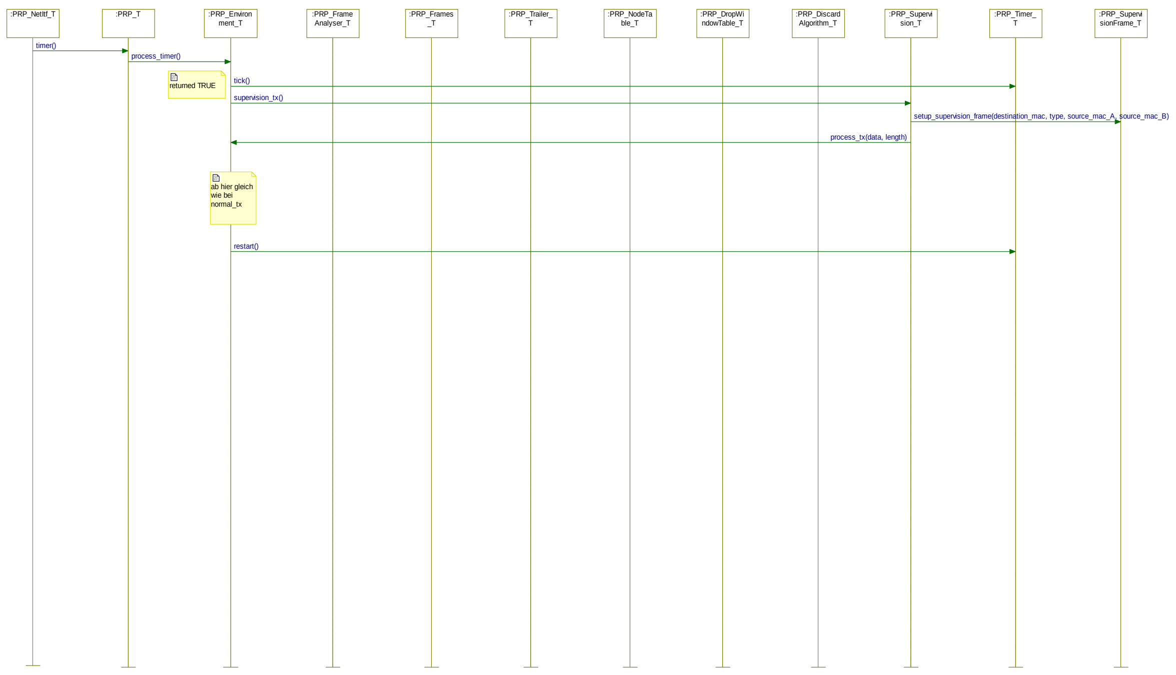

Figure 8: Transmission of a supervision frame

3.5.5 Supervise

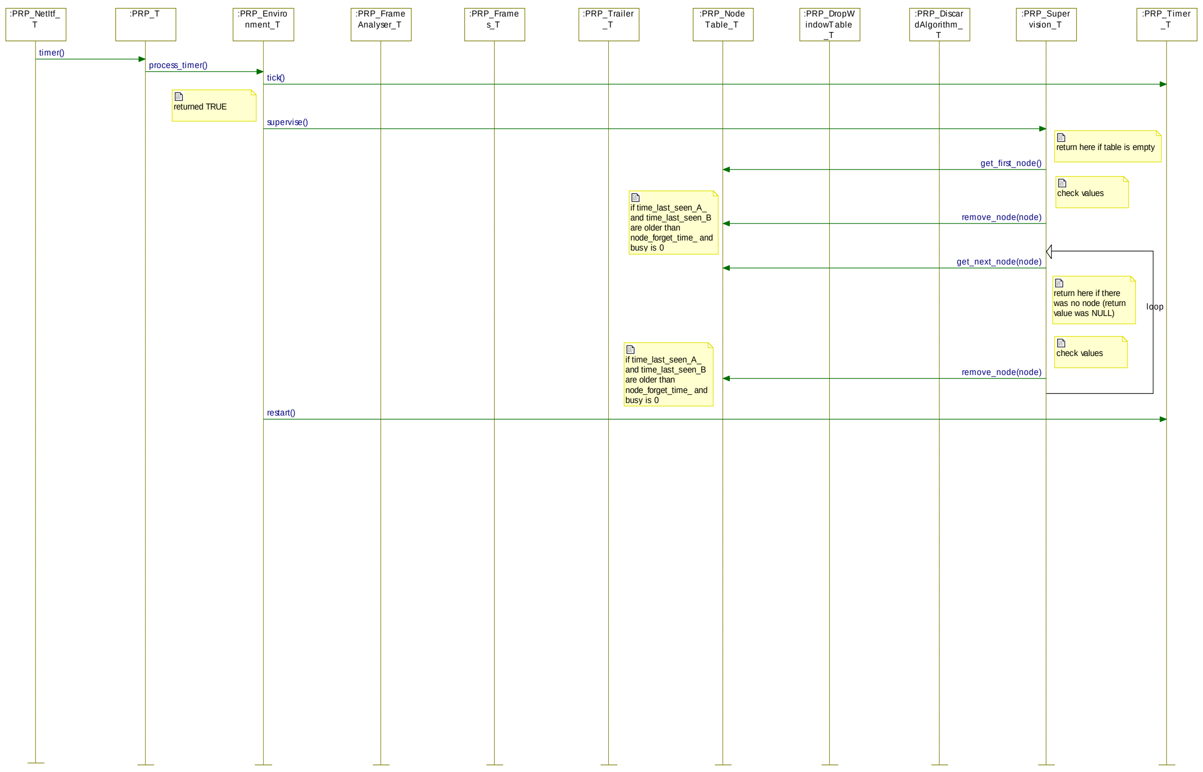

Figure 9: Supervision of the node table

3.5.6 Bridging receiving

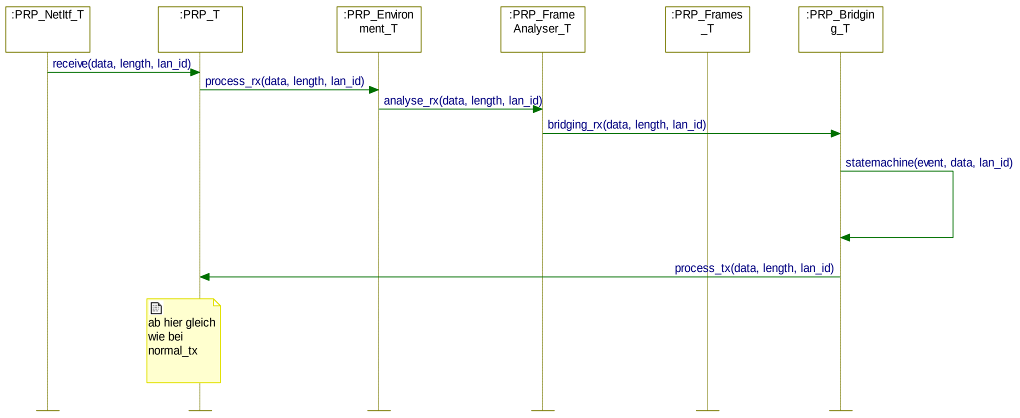
:PRP_NetItf_T
:PRP_T
:PRP_Environment_T
:PRP_FrameAnalyser_T
:PRP_Frames_T
:PRP_Bridging_T
receive(data, length, lan_id)
process_rx(data, length, lan_id)
analyse_rx(data, length, lan_id)
bridging_rx(data, length, lan_id)
statemachine(event, data, lan_id)
process_tx(data, length, lan_id)
ab hier gleich wie bei normal_tx

Figure 10: Receiving of a RSTP-BPDU

4 Integration in Linux

4.1 Example Userspace integration in Linux

The PRP protocol engine receives the frames from the two real Ethernet interfaces does the Duplicate elimination, and forwards it to a virtual network interface which forwards the frames back to the operating systems protocol engine. Therefore all applications are talking with the virtual network device.

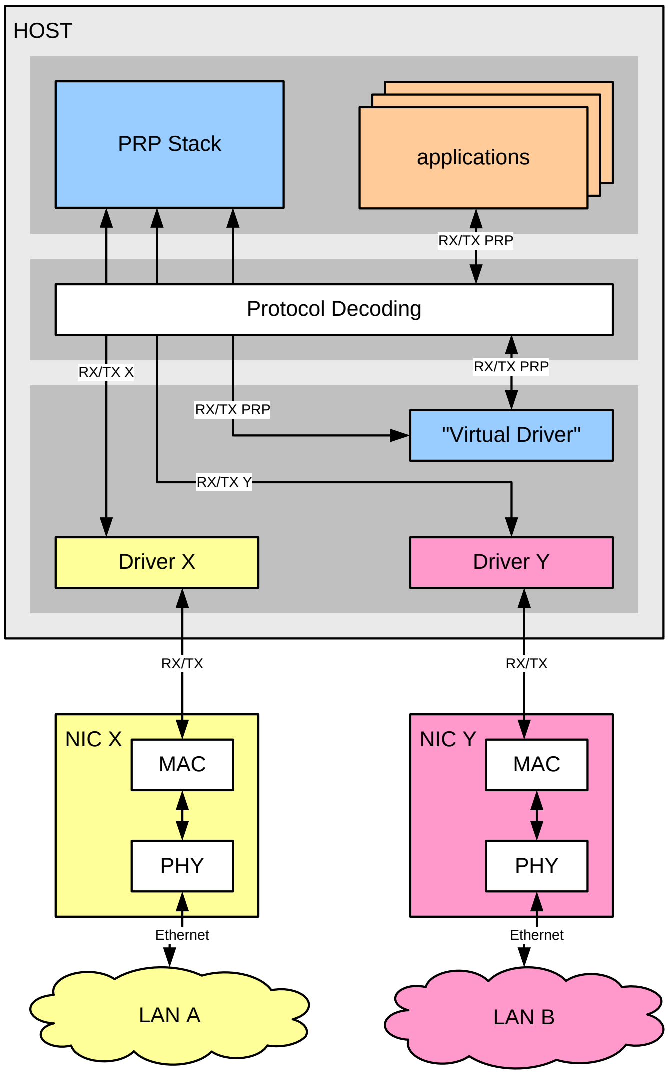

Figure 11: Linux Userspace integration

5 Graphic index