TCNOpen	Open Source Developement

TRDP

Train Real Time Data Protocol

TRDP-SPY Architecture and Design

Document reference no: TCN-TRDP3-D-BOM-024-02

Author :	Florian Weispfenning
Organisation :	Bombardier Transportation
Document date:	17 Sept 2013
Revision:	2
Status:	issued

Dissemination Level		
PU	Public	
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

DOCUMENT SUMMARY SHEET

This document describes the design specification of the TRDP-PD and TRDP-MD plug-in for Wireshark. The IP-Train project is in need of a TRDP Wire Protocol Analysis Tool which shall be made available as a plug-in to the popular Wireshark network protocol analyser.

Participants		
Name and Surname	Organisation	Role
Florian Weispfenning	BOM	Participant

History			
V1	11 Sept 13	F. Weispfenning	Initial version
V2	17 Sept 13	Armin-H. Weiss	License note added, issued

Table of Contents

Table of Figures

1. Introduction

1.1. Purpose

As part of the IP-Train project, two new protocols namely TRDP-PD (Process Data) and TRDP-MD (Message Data) are intended to be supported by the Wireshark tool. The support is envisaged to be made available in the form of a plug-in.

The existing GUI of the Wireshark V1.8.3 shall not be modified. The plug-in TRDP-SPY shall be available as a DLL for Windows platform and shared library for TRDP-spy for Linux platform.

1.2. Intended Audience

The TRDP-SPY will be used primarily by TRDP Engineers.

1.3. References/Related Documents

Reference	Number	Title
[Wire]	IEC51375-3-2	TRDP Protocol (Annex A)
[TCN]	61375-1/FDIS	Train Communication Network
[WS-UM]	user-guide-a4.pdf	Wireshark User Manual
[WS-Web]	http://www.wireshark.org	Web link for Wireshark
[WS-Setup]	http://www.wireshark.org/docs/wsdg_html_chunked/ChSetupWin32.html	Setup Wireshark on Windows
[TRDP-UM]	TCN-TRDP2-D-BOM-011-21	TRDP User's Manual

1.4. Abbreviations and Definitions

Abbreviation	Definition
API	Application Program Interface
BT	Bombardier Transportation
DLL	Dynamic Link Library
FCS	Frame Check Sequence
GUI	Graphic User Interface
IEC	International Electrotechnical Commission
IP	Internet Protocol
TRDP - MD	TRDP – Message Data
TRDP - PD	TRDP – Process Data
TRDP	Train Real Time Data Protocol
MD	Message Data
PD	Process Data
SOE	Standard Operating Environment
QoS	Quality of Service
T2W	Train to Wayside
TCMS	Train Control and Management System
TCN	Train Communication Network
TCP/IP	Transmission Control Protocol/ Internet Protocol
WP	Wire Protocol (on IP Train Network)
XML	eXtended Markup Language

2. Overview of the Requirements

2.1.1. Description

Wireshark is a popular network protocol analyser used by network professionals around the world. Wireshark is an open source and provides flexibility for plug-in development.

A TRDP Analysis Tool is needed, which shall be made available as a plug-in to the popular Wireshark network protocol analyser.

The TRDP-SPY plug-in shall fulfil the following requirements:

- shall be available for Wireshark v1.8.3
- shall be as far as possible also support later versions of Wireshark
- shall be available for both Windows XP and Linux
- shall decode all TRDP PD and TRDP MD frames in the live capture window
- shall support all functionalities of Live Capture window for real time analysis (capture mode) and for recorded mode (messages saved on hard disk)
- shall allow declaring filters on any field of TRDP PD and TRDP MD frames
- shall provide statistical functionalities
- shall be able to decode and interpret the application data, using the configuration file "TRDP_config.xml"

3. Architecture

This chapter describes the architecture of the TRDP-SPY plugin itself.

The plugin consists out of three parts:

3.1. Analysis

The analysis of Ethernet packets is expanded for the TRDP specific part.

3.2. Visualization

The visualisation is already realized by Wireshark that will also integrate the new TRDP packets in its graphical user interface.

3.3. XML configuration

The description of the transmitted telegrams by the XML configuration is optional. This necessary information is described in Figure 1, visualizing the connection between ComId [1] and Datasets[2].

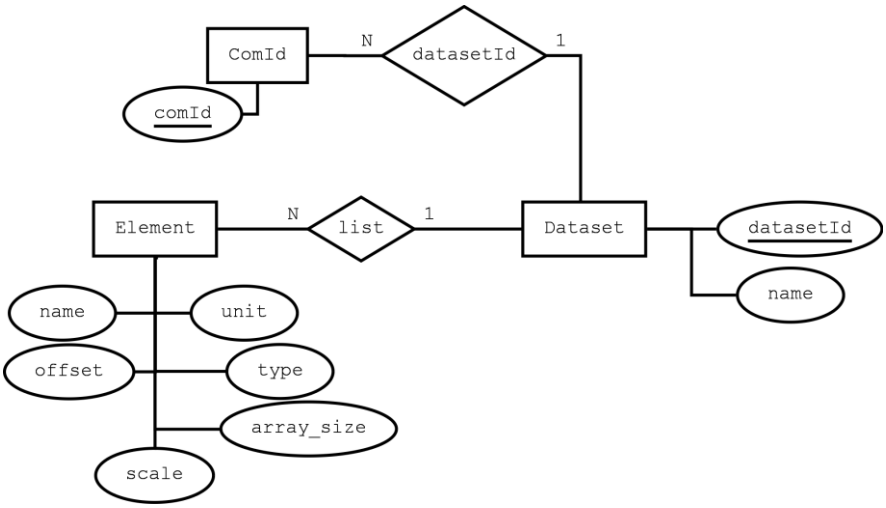

Figure 1 Relation between datasets and a ComId

On start-up, the necessary information is loaded into the memory in order to get a faster decoding.
The Datasets can contain out of multiple variables (Elements) or even of themselves. This is realized by referring in `type` of `Element` to the same `datasetId`.
The other attributes can be used to decode the values in a human readable format. Table 55 in the User's Manual[TRDP-UM] describes these.

[1]Unique identifier of an TRDP packet

[2] Structure of the transported variables.

4. Design Description

4.1. Environment

4.1.1. System

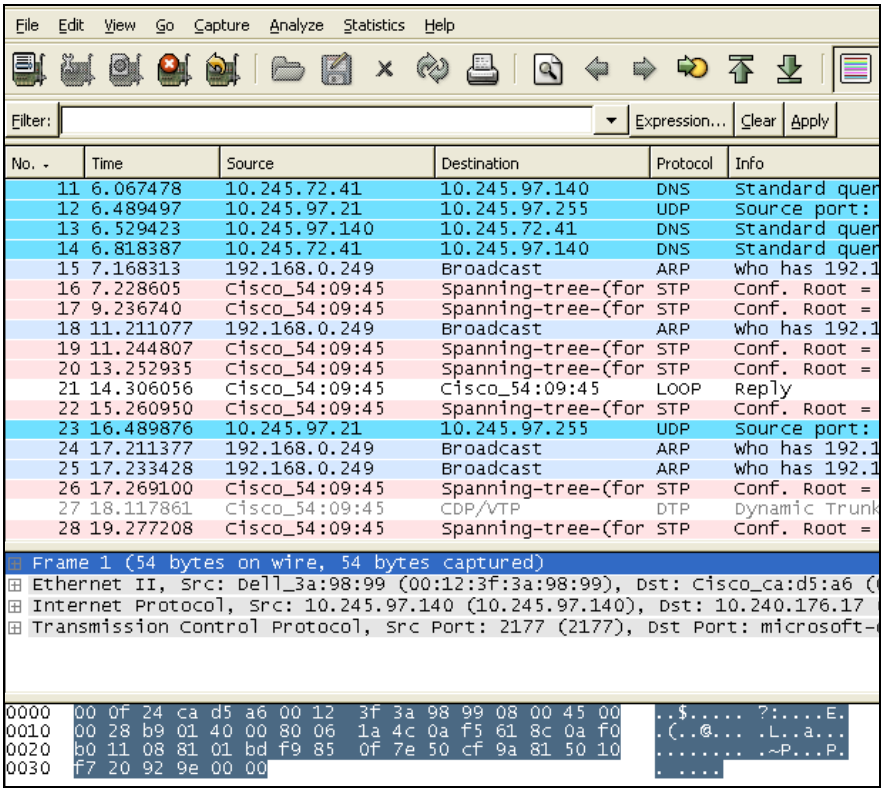

Figure 2 Example view of Wireshark

TRDP Wire Protocol Analysis tool (TRDP-SPY) shall provide qualitative and quantitative analysis of TRDP streams, in order to verify system behaviour during qualification tests (level 2 and level 3) and provide help in problem analysis during train integration and debugging.

4.1.2. Operational Environment

The plug-in shall be compatible with Windows XP and Linux implementation of Wireshark.

Standard behaviour of Wireshark for all other protocols than WP shall not be influenced in any way by the TRDPWP analysis plug-in.

The plug-in shall be delivered as a DLL (Windows) along with the Wireshark-setup.exe and shared Library (.la, .lai and .so files or Linux) along with the minimal source – Wireshark-1.8.3.

4.1.3. Development Environment for Windows

Following specifications are used for development of the TRDP PD and TRDP MD plug-in for Wireshark.

- Operating System: Windows XP
- Tool : Wireshark V1.8.3
- Programming Language: C
- TRDP Wire Protocol

4.1.3.1. Steps to compile for Windows

Prerequisites:

- Wireshark minimal source (wireshark-1.8.3.tar.bz2).
- `TRDP-SPY_src.zip` source.
- Follow the online guide [WS-Setup]

Steps:

- Unzip wireshark-1.8.3.tar.bz2 to `c:\` and rename it to `wireshark`.
- Unzip `TRDP-SPY_src.zip`.
- From TRDP-SPY_src source copy folders `trdp_spy` to `c:\wireshark/plugins`.
- Also copy `config.nmake` from `TRDP-SPY` to `c:\wireshark` (overwrite the existing one).
- First clean it using command "`nmake -f makefile.nmake distclean`" or run `clean.bat` from `c:\wireshark\plugins\trdp_spy`.
- Then compile using command "`nmake -f makefile.nmake`" or run `build.bat` from `c:\wireshark\plugins\trdp_spy`.

4.1.3.2. Files to be changed when a new plug-in is made

`wireshark` source directory:

`config.nmake`

Files to be newly create for a new plug-in: Create a folder `TRDP_spy` in `c:\wireshark\plugins\trpd-spy\`

Plug-in should be placed in `wireshark/plugins` and should contain the following files:

- `Makefile.am`
- `Makefile.common`
- `Makefile.nmake`
- `moduleinfo.nmake`
- `moduleinfo.h`
- `plugin.rc.in`
- `packet-TRDP_spy.c`
- `packet-TRDP_spy.h`
- `parsebody.c`
- `parsebody.h`
- `lookuptype.c`
- `lookuptype.h`

Please refer `wireshark/docs/readme.plugins` and `wireshark/docs/readme.developer` for more references.

4.1.3.3. Prerequisite for Build Procedure

1. MS-VC++ 9.0 (Visual Studio 2008) needs to be pre-installed.
2. Place the source of the `TRDP_spy` plug-in in the `plugins` folder of the `Wireshark` source files.
3. The `TRDP_spy` folder should contain the files as shown in Figure 3 below:

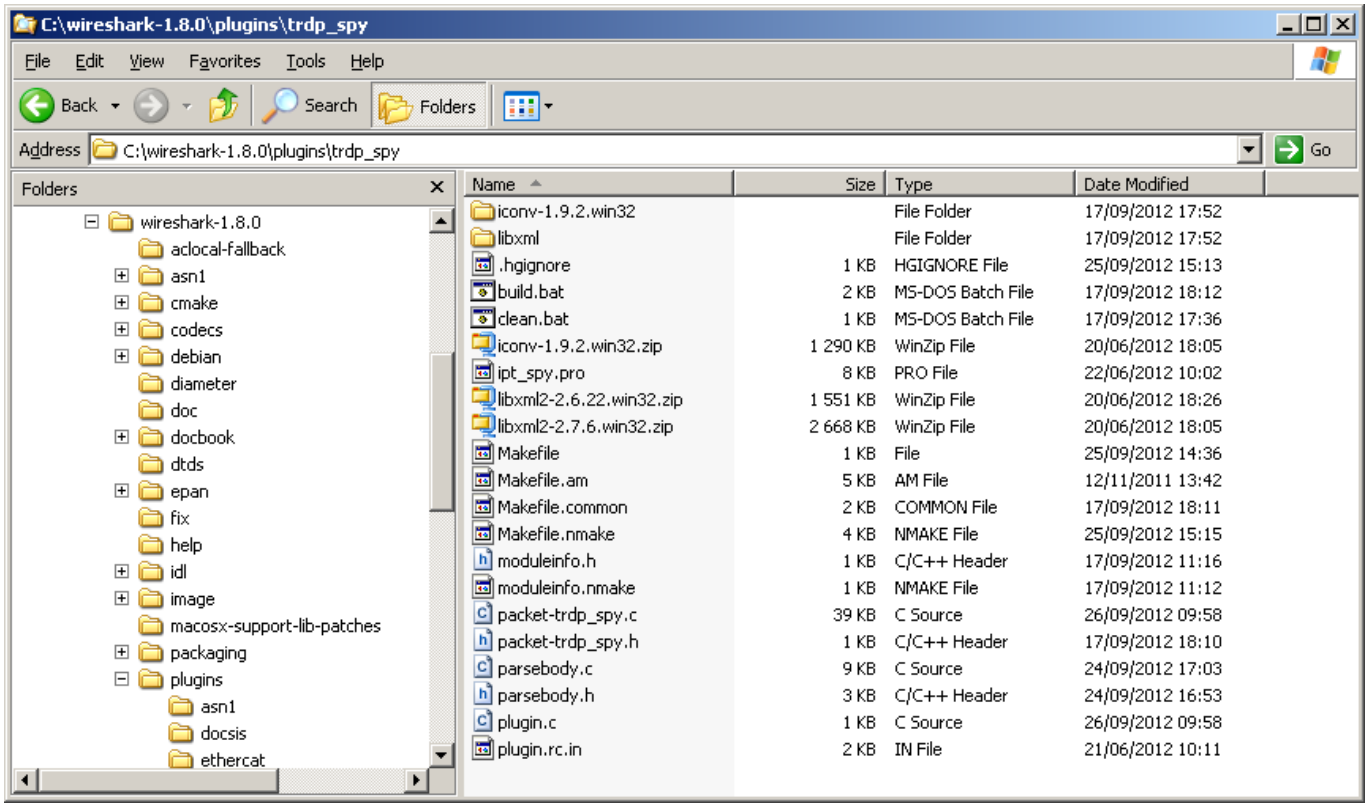

Figure 3 Files within trdp_spy folder

4.1.3.4. Build Procedure

Open a Terminal and navigate to the plugin location `\Wireshark\plugins\trpd_spy`. Enter the command `build.bat` on the DOS prompt at the `TRDP_spy` plug-in source location `\Wireshark\plugins\TRDP_spy`. This will generate the `TRDP_spy.dll` and the related output files as shown in below:

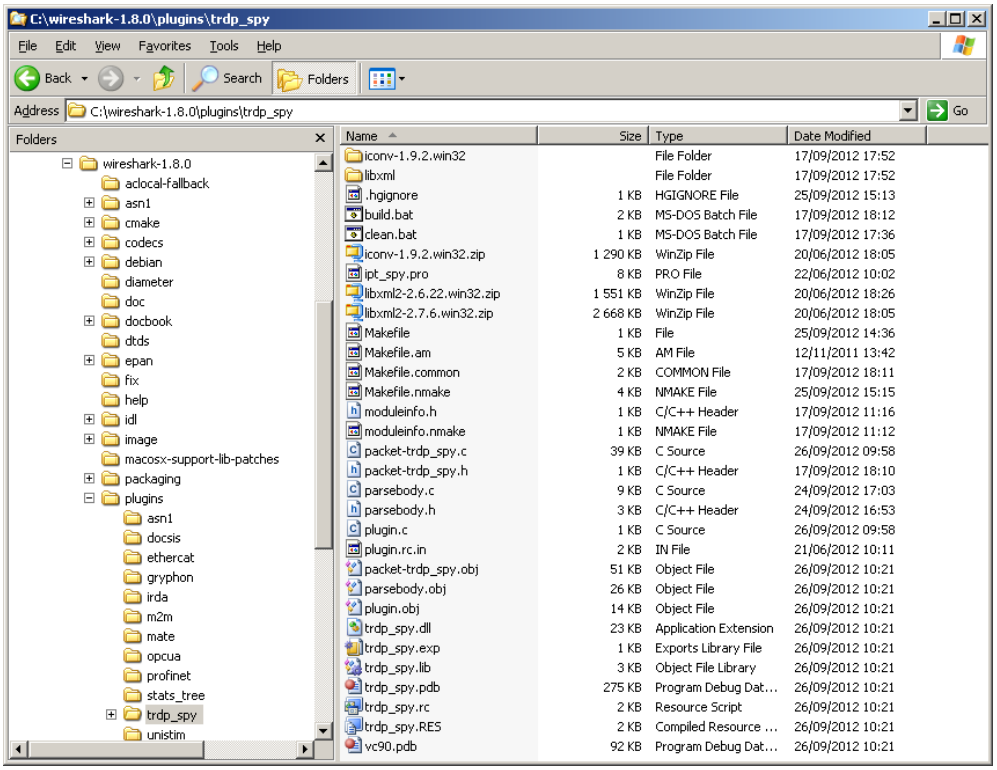

Figure 4: Files within TRDP_spy after build procedure

4.1.4. Development Environment for Linux

Following specifications are used for development of the TRDP PD and TRDP MD plug-in for Wireshark.

- Operating System: Ubuntu 12.04 LTS-Linux
- Tool : Wireshark v 1.8.3
- Programming Language: C

4.1.4.1. Steps to compile and install Wireshark on Linux:

Prerequisites:

- Wireshark source (`wireshark-1.8.3.tar.bz`).
- `TRDP-SPY_src.zip`.

Steps:

Unzip `wireshark-1.8.3.tar.bz` with command on Console and not by using win-zip>unzip :

```
$ tar xjvf Wireshark-1.8.3.tar.bz
```

Now execute the following commands to compile and install Wireshark

```
$ cd wireshark-1.8.3
$ ./configure --prefix=/opt/local
$ make
$ make install
```

`$ /opt/local/bin/wireshark` (to launch Wireshark)
Unzip `TRDP-SPY.zip`.
Now copy folder `TRDP_spy` location `wireshark-1.8.3/plugins/`. Then give following commands.

```
$ cd ../wireshark-1.8.3/plugins/trdp_spy
$ make clean
$ make
```

Please refer `wireshark-1.8.3/readme` and `wireshark-1.8.3/install` for more reference.

4.1.4.2. Files to be changed when a new plug-in is made.

The same files as for Windows are used to create the plugin. The structure for windows was described in chapter 4.1.3.2.

4.1.4.3. Prerequisite for Build Procedure

1. Place the source of the `TRDP_spy` plug-in in the `plugins` folder of the `wireshark-1.8.3` source files.
2. The `TRDP_spy` folder should contain the files as shown in Figure 5 below:

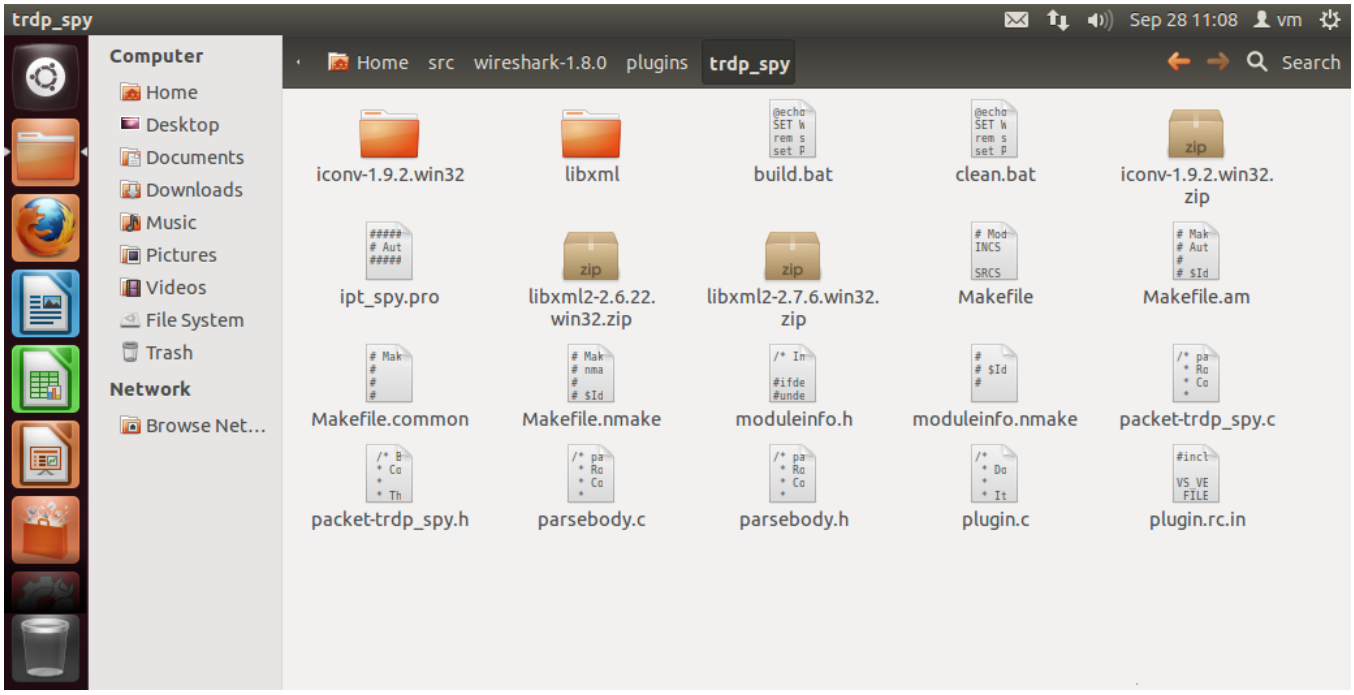

Figure 5 Files within trdp_spy folder

4.1.4.4. Build Procedure

Enter the command `make` on the console at the `TRDP_spy` plug-in source location `\Wireshark\plugins\TRDP_spy`. This will generate the `packet-trdp_spy.so` and related output files as shown in the Figure 6 below:

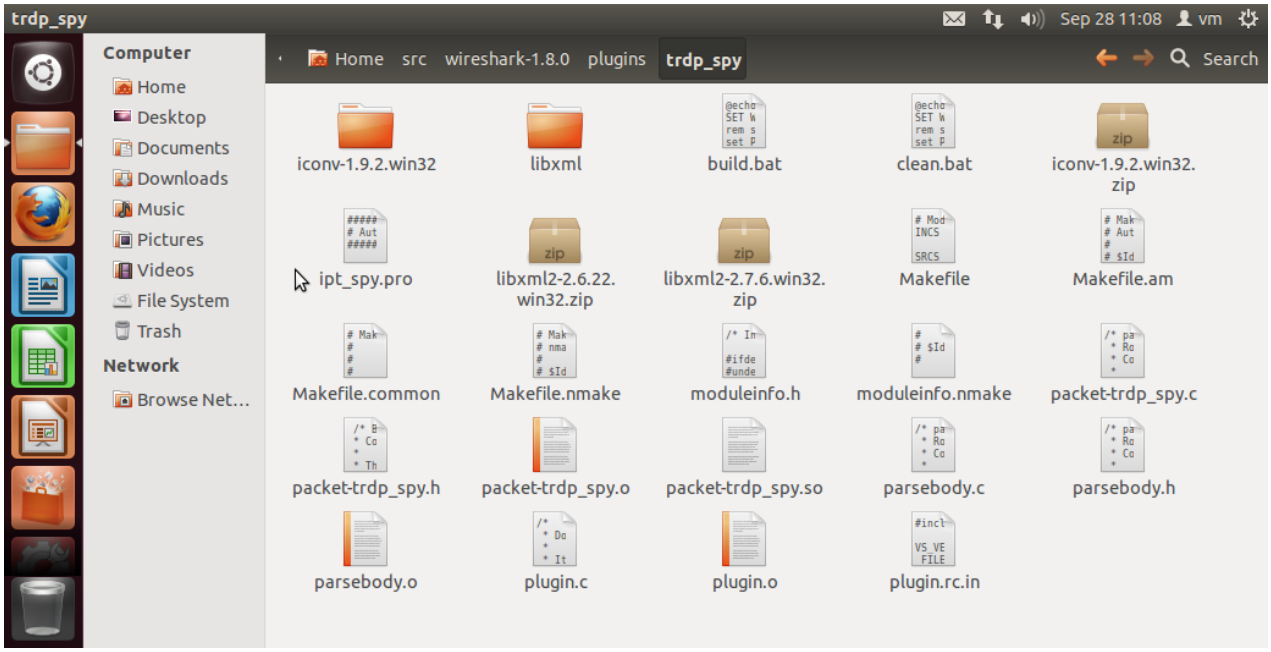

Figure 6 Files within trdp-spy after build procedure

Note: The Makefile copies the library to the */opt/local/libs/Wireshark/plugins/1.8.3* and then launch Wireshark (`/opt/local/bin/wireshark`) to see the working of the TRDP_spy.

4.2. Interfaces

The plug-in shall be delivered as a DLL i.e. `TRDP_spy.dll` for Windows platform and shared library i.e. `TRDP_spy.la` and `TRDP_spy.so` files for Linux platform.
For Application Data decoding additional `libxml2.dll` for Windows and `libxml2.a`, `libxml2.la and libxml2.so` for Linux are required which functions to parse the `TRDP_config.xml` file that contains the details of the Data-sets corresponding to each frame that is captured or logged by Wireshark.

Overall interface of the system can be explained as shown in the figure below:

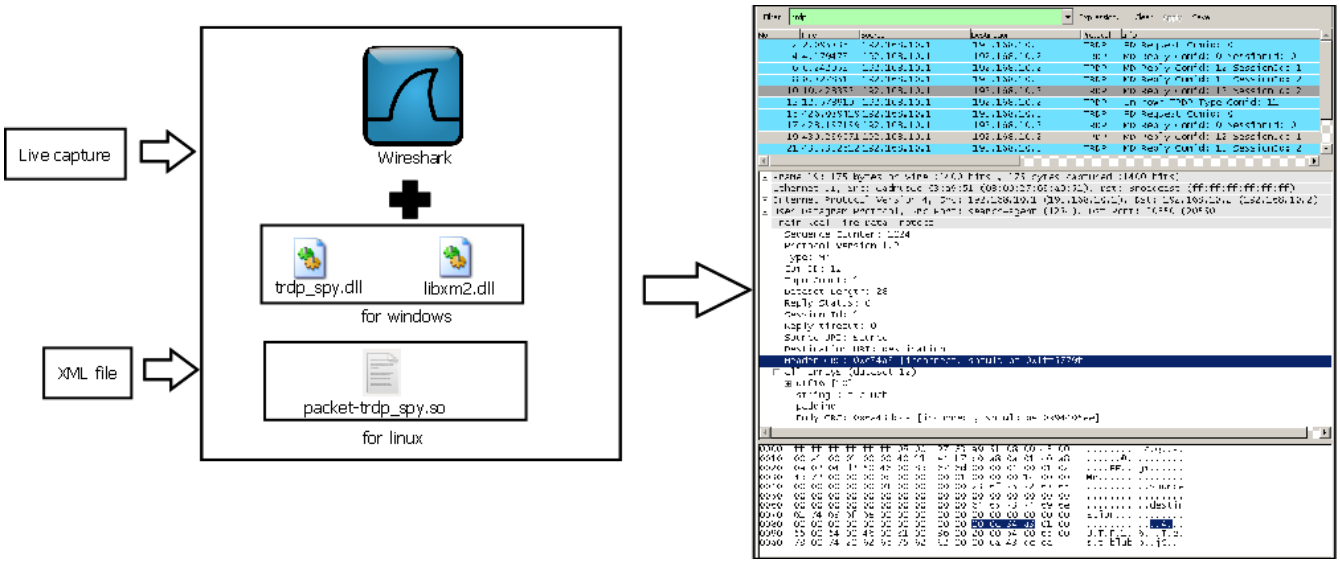

Figure 7 Interface Diagram

4.2.1. System Interface

In context of this document System Interface shall mean plug-in's interface to Wireshark.

A protocol dissector can be called in 2 different ways:

- Operational dissection
  In this mode Wireshark does not build a "protocol tree".

- Detailed dissection
  In this mode Wireshark builds a "protocol tree" giving all details

Wireshark distinguishes between the two modes with the proto_tree pointer.

For the TRDPWP plug-in "protocol tree" shall be built.

There shall be a protocol register routine which shall be called when Wireshark starts. The code to call register routines is generated automatically by Wireshark, to arrange to call the protocol register routine at start-up.

void proto_register_trdp (void)

The protocol will be forward declared using ***void proto_reg_handoff_trdp (void)***.

Function which actually does the dissection of the packets will be
static void dissect_trdp (tvbuff_t *tvb, packet_info *pinfo, proto_tree *tree)

Display sub-tree shall be built using ***build_trdp_tree*** and ***dissect_trdp_body***

Protocol description shall be registered as
proto_trdp_spy = proto_register_protocol ("Train Real Time Data Protocol","TRDP","trdp")

Wireshark defines a conversation as a series of data packet between two address: port combinations. Conversation routines listed below shall be used to work with conversations.

- ***conversation_new,***
- ***find_conversation,***
- ***conversation_add_proto_data,***
- ***conversation_get_proto_data,***
- ***conversation_delete_proto_data***

Once the dissector is prepared it will be "pluginized".

4.2.2. User Interface

The user on selection TRDP from "*Analyse/ Decode As*" menu is able to decode TRDP PD and TRDP MD frames in Live Capture mode as well as save them for later use.

The user shall be able to define filters on TRDP PD and TRDP MD, from *Capture* → *Capture Filters Menu*.

4.3. Modularization

4.3.1. Display Filter

The tool shall allow declaring filter on any field of TRDP PD and TRDP MD frames.
New filters can be defined using this Dialog Box. For example “TRDP” filter is added in the following Figure 8.

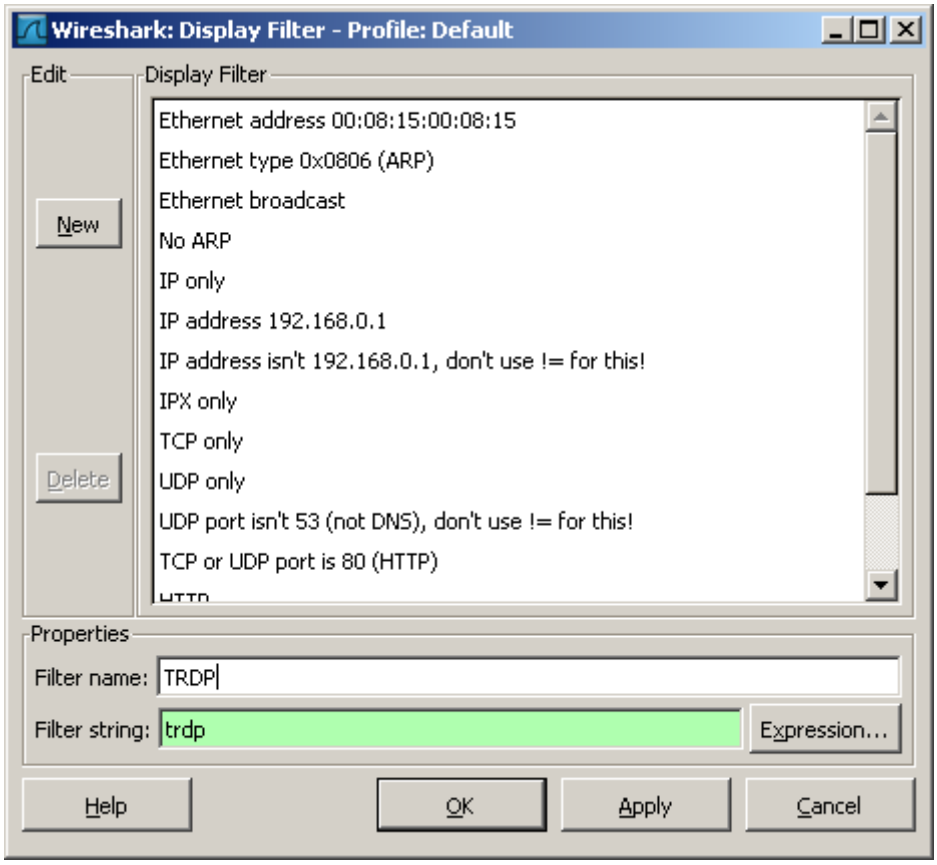

Figure 8 Display Filter

The TRDP shall be accessible in the list of available protocol in filter expression dialog box as TRDPWP.

The Dialog pops-up on clicking the *Expression* button on the *Display Filter* Dialog box. Through this Dialog, filters can even be set to the fields within the Data Frame as shown for the “Destination URI “of TRDP in the following Figure 9.

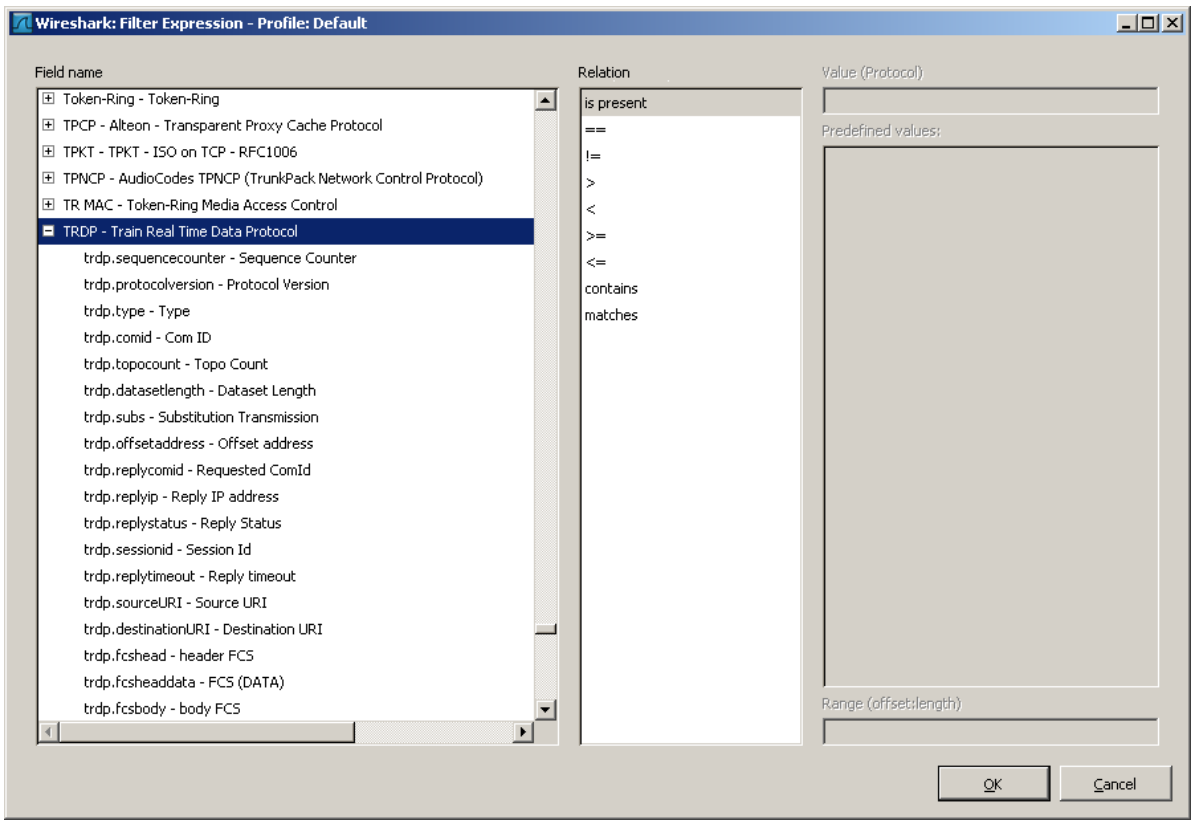

Figure 9 Filter Expression

Routines used for dialog boxes-filter editing:
For creating a "*Display Filter*" dialog box caused by a button click (*Filter* button on the main window).
void display_filter_construct_cb(GtkWidget *widget, gpointer user_data)

It destroys any filter dialog created by button widget.
void filter_button_destroy_cb(GtkWidget *widget, gpointer user_data)

User requested the "*Display Filter*" dialog box by menu or toolbar.
void dfilter_dialog_cb(GtkWidget *widget)

Create an "*Add expression*" dialog box caused by a button click.
void filter_add_expr_bt_cb(GtkWidget *widget, gpointer main_w_arg)

Colorize a text entry for various conditions:
void colorize_filter_te_as_empty(GtkWidget *widget): as empty
void colorize_filter_te_as_invalid(GtkWidget *widget): as a invalid
void colorize_filter_te_as_valid(GtkWidget *widget): as a valid

void filter_te_syntax_check_cb(GtkWidget *widget): depending on "validity"

When User requests the "*Add Expression*" dialog box
void dfilter_expr_dlg_new(GtkWidget *widget)

4.3.1.1. Data

None

4.3.2. Live Capture Functionality

The analysis shall provide a supplementary entry, named TRDP Wire Protocol or TRDP-SPY in the “Analyze/Decode As” menu of Wireshark.

When user selects a particular TRDP message, and activates decoding, all already and future received messages shall be displayed in a decoded form in the *Live capture* window.

All functionalities described for the Live Capture window shall be available for real time analysis (capture mode) and for recorded mode (messages saved on hard disk)

The Protocol column in the live window shall display TRDP as protocol name for all TRDP received messages, when decoding has been enabled. If decoding has not been enabled in the Analysis menu, protocol should be indicated as UDP or TCP according to the used protocol.

The Information column shall display following parameters for messages on which TRDP decoding has been activated:

4.3.2.1. Data

- Source IP
  Source IP address

- Destination IP
  Destination IP address

- Type
  Type of message (Pr, Pd, Mn, Mr, Mp, Mq, Mc, Me)

- Timestamp value
  Timestamp in µs

- Protocol version
  Protocol version of TRDP- PD / TRDP-MD

- Topo counter value
  Topology Counter

- ComId value

Com – Id of the data-set

- Sequence counter value
  SequenceCounter

Note: If a message is received with an unsupported protocol version (from Wireshark point of view), the information column shall not try to decode the message, but displays an error message ("Unknown TRDP Wire Protocol version")

4.3.2.2. Action

Start Capturing using menu option Capture/Start.

4.3.2.3. Sequence Diagram

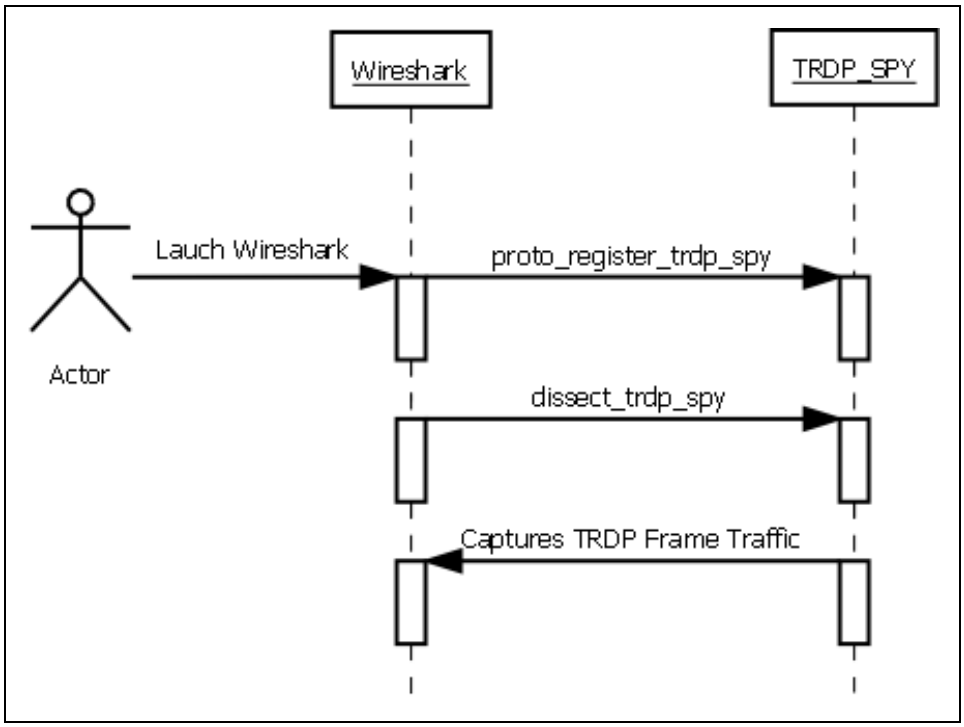

Figure 10 Live Functionality Sequence Diagram

4.3.3. Detailed Analysis of TRDP PD Frames

The detailed analysis of TRDP PD frames shall contain detailed values for:

4.3.3.1. Data

Frame details

- Arrival time
  Date and time of arrival of the frame
- Time delta from previous packet
  Time in seconds since last frame was received
- Time since reference of first frame
  Time in seconds since first frame was received

- Frame number
  Sequence number of frame
- Packet Length
  Length of the packet
- Capture length
  Length of the captured packet

Ethernet protocol details

- Destination MAC Address
  MAC address of destination
- Source MAC address
  MAC address of the source
- Protocol type
  Protocol identifier tag

IP protocol details

(All fields implemented by Wireshark for standard IP messages shall be displayed)

UDP header details

- Source port
  Source port number
- Destination port
  Destination port number
- Length
  Length of UPD header and data in bytes
- Checksum (with correctness indication)
  Checksum of UPD header and UDP data

TRDP- PD

- Sequence counter value
  Unique automatically incremented counter
- Protocol version
  Protocol version of TRDP-PD
- Type
  Type of a message of a TRDP-PD (0x5072 ‘Pr’ or 0x5064 ‘Pd’)
- ComId
  Com-Id of the data-set
- TopoCount
  Topology Counter
- Dataset length
  Length in 4 bytes of the process data dataset
- Subs
  Flag indicating substitution transmission
- OffsetAddress
  Offset for process data
- ReplyComId
  The requested ComId – only used in a PD request

- ReplyIpAddress
  Reply address used in a PD request (otherwise set to 0)
- Header length
  Length in 8 bit bytes of the process data header excluding its frame check sequence
- Header FCS
  Frame check sequence for the header
- Dataset
  the transmitting data itself.
- Padding
  One, two or three padding bytes if necessary
- Frame Check Sequence
  Check value of this sub message

Each sub-field of the message shall be displayed in collapsible tree format

4.3.3.2. Action

- Select any TRDP Frame from the captured traffic
- Extend *TRDP* Level in the Frame Detail Pane (middle Pane) as shown in fig below:

```
⊞ Frame 23: 175 bytes on wire (1400 bits), 175 bytes captured (1400 bits) on interface 0
⊞ Ethernet II, Src: CadmusCo_63:a9:51 (08:00:27:63:a9:51), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
⊞ Internet Protocol Version 4, Src: 192.168.10.1 (192.168.10.1), Dst: 192.168.10.2 (192.168.10.2)
⊞ User Datagram Protocol, Src Port: search-agent (1234), Dst Port: 20550 (20550)
⊟ Train Real Time Data Protocol
    Sequence Counter: 1024
    Protocol Version 1.2
    Type: Mr
    Com ID: 12
    Topo Count: 1
    Dataset Length: 28
    Reply Status: 0
    Session Id: 2
    Reply timeout: 0
    Source URI: source
    Destination URI: destination
    Header CRC: 0xc7c8f [incorrect, should be 0xbb8bf55a]
  ⊟ all_arrays (dataset 12)
    ⊞ utf16 [10]
      string : t blubb
      padding
      Body CRC: 0x6a43dcca [incorrect, should be 0xf38f25dc]

0000  ff ff ff ff ff ff 08 00  27 63 a9 51 08 00 45 00   ........ 'c.Q..E.
0010  00 a1 00 01 00 00 40 11  e4 f7 c0 a8 0a 01 c0 a8   ......@. ........
0020  0a 02 04 d2 50 46 00 8d  89 7f 00 00 04 00 01 02   ....PF.. ........
0030  4d 72 00 00 00 0c 00 00  00 01 00 00 00 1c 00 00   Mr...... ........
0040  00 00 00 00 00 02 00 00  00 00 73 6f 75 72 63 65   ........ ..source
0050  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0060  00 00 00 00 00 00 00 00  00 00 64 65 73 74 69 6e   ........ ..destin
0070  61 74 69 6f 6e 00 00 00  00 00 00 00 00 00 00 00   ation... ........
0080  00 00 00 00 00 00 00 00  00 00 00 0c 7c 8f 97 00   ........ ....|...
0090  55 00 54 00 46 00 31 00  36 00 20 00 54 00 65 00   U.T.F.1. 6. .T.e.
00a0  73 00 74 20 62 6c 75 62  62 00 00 6a 43 dc ca      s.t blub b..jC..
```

Figure 11 Expansaion of TRDP frame in middle pane

4.3.3.3. Sequence Diagram

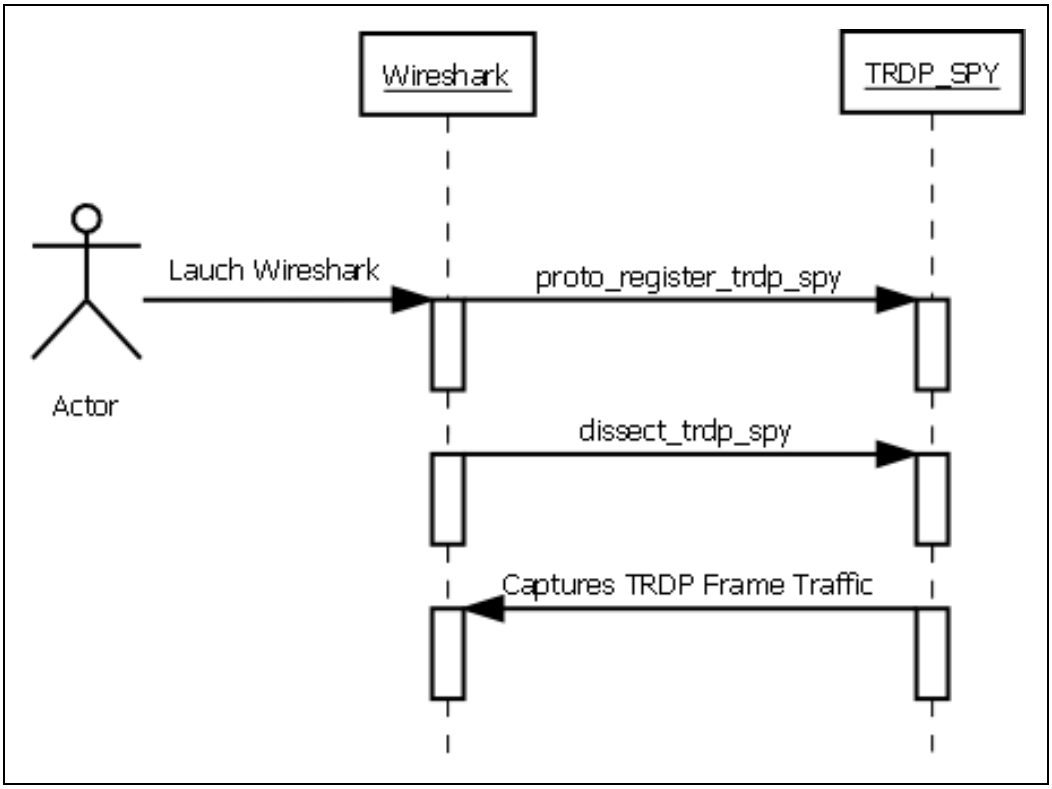

Figure 12 Detail Analysis of TRDP Frame - Sequence Diagram

4.3.4. Detailed Analysis of TRDP MD Frames

The detailed analysis of TRDP MD data frames shall contain detailed values for:

4.3.4.1. Data

Frame details

- Arrival time
  Date and time of arrival of the frame
- Time delta from previous packet
  Time in seconds since last frame was received
  - Time since reference of first frame
    Time in seconds since first frame was received
- Frame number
  Sequence number of frame
- Packet Length
  Length of the packet
- Capture length
  Length of the captured packet

Ethernet protocol details

- Destination MAC Address

MAC address of destination
- Source MAC address
  MAC address of the source
- Protocol type
  Protocol identifier tag

IP protocol details
(all fields implemented by Wireshark for standard IP messages shall be displayed)

UDP header details
- Source port
  Source port number
- Destination port
  Destination port number
- Length
  Length of UPD header and data in bytes
- Checksum (with correctness indication)
  Checksum of UPD header and UDP data

TCP header details
- Source port
  Source port number
- Destination port
  Destination port number
- Sequence number
  Destination port number
- Length
  Length of TCP header and data in bytes
- Checksum (with correctness indication)
  Checksum of UPD header and UDP data

TRDP-MD messages with data

- Timestamp
  Timestamp in µs
- Protocol version
  Protocol version of TRDP-MD
- TopoCount
  Topology counter
- ComId
  Com-id of data-set
- Dataset length
  Length in 8 bit bytes of the dataset
- Type
  Type of datagram (0x4D44 MD TRDP MD data, 0x4D41 MA TRDP MD acknowledgement)

- User Status
  The status value is set by the responding application to report the execution result of a request message. The execution result is supplied by the responding application and transmitted to the requesting application in addition to the response message itself.
- Header length
  Length in 8 bit bytes of the message data dataset excluding frame check sequence
- SrcURILen
  Source URL length in 32 bit words
- DstURILen
  Destination URL length in 32 bit words
- Index
  Index used to send large message split up into multiple datagrams
- Sequence number
  Unique sequence number
- MsgLength
  Length of the complete message in blocks of 1024 bytes excluding frame check sequence
- Session Id
  Identity of the session
- Source URI
  Source URL
- Destination URI
  Destination URL
- Response Timeout
  The response timeout used in a request /response session.
- FCS
  Frame check sequence
- Dataset
  the transmitting data itself
- Padding
  One, two or three padding bytes if necessary
- Frame Check Sequence
  Check value of this sub message

Each sub-field of the message shall be displayed in collapsible tree format.

4.3.4.2. Action

Same as 4.3.3.1

4.3.4.3. Sequence Diagram

Same as 4.3.3.2

4.3.5. Analysing Application Data

4.3.5.1. Action

- Select any TRDP Frame from the captured traffic

- Extend TRDP Protocol Level in the Frame Detail Pane (middle Pane).
- Extend the Data Field within TRDP as shown in figure below:

```
⊞ Frame 23: 175 bytes on wire (1400 bits), 175 bytes captured (1400 bits) on interface 0
⊞ Ethernet II, Src: CadmusCo_63:a9:51 (08:00:27:63:a9:51), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
⊞ Internet Protocol Version 4, Src: 192.168.10.1 (192.168.10.1), Dst: 192.168.10.2 (192.168.10.2)
⊞ User Datagram Protocol, Src Port: search-agent (1234), Dst Port: 20550 (20550)
⊟ Train Real Time Data Protocol
    Sequence Counter: 1024
    Protocol Version 1.2
    Type: Mr
    Com ID: 12
    Topo Count: 1
    Dataset Length: 28
    Reply Status: 0
    Session Id: 2
    Reply timeout: 0
    Source URI: source
    Destination URI: destination
    Header CRC: 0xc7c8f [incorrect, should be 0xbb8bf55a]
  ⊟ all_arrays (dataset 12)
    ⊞ utf16 [10]
      string : t blubb
      padding
      Body CRC: 0x6a43dcca [incorrect, should be 0xf38f25dc]

0000  ff ff ff ff ff ff 08 00  27 63 a9 51 08 00 45 00   ........ 'c.Q..E.
0010  00 a1 00 01 00 00 40 11  e4 f7 c0 a8 0a 01 c0 a8   ......@. ........
0020  0a 02 04 d2 50 46 00 8d  89 7f 00 00 04 00 01 02   ....PF.. ........
0030  4d 72 00 00 00 0c 00 00  00 01 00 00 00 1c 00 00   Mr...... ........
0040  00 00 00 00 00 02 00 00  00 00 73 6f 75 72 63 65   ........ ..source
0050  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0060  00 00 00 00 00 00 00 00  00 00 64 65 73 74 69 6e   ........ ..destin
0070  61 74 69 6f 6e 00 00 00  00 00 00 00 00 00 00 00   ation... ........
0080  00 00 00 00 00 00 00 00  00 00 00 0c 7c 8f 97 00   ........ ....|...
0090  55 00 54 00 46 00 31 00  36 00 20 00 54 00 65 00   U.T.F.1. 6. .T.e.
00a0  73 00 74 20 62 6c 75 62  62 00 00 6a 43 dc ca      s.t blub b..jC..
```

Figure 13 Application Data decoding in middle pane

4.3.5.2. Sequence Diagram

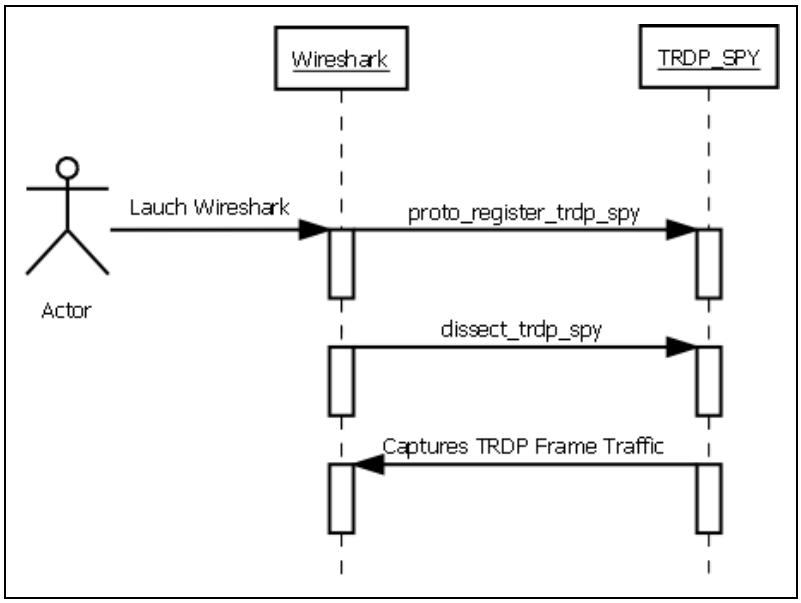

Figure 14 Application Data Decoding - Sequence Diagram