VTT INFORMATION TECHNOLOGY

NIPPER

USER GUIDE

Contact information

In Nipper related issues you are welcome to contact via e-mail to address nipper@opensource.erve.vtt.fi.

More information about VTT's open source projects can be found from VTT's open source web page opensource.erve.vtt.fi.

Table of Content

1.PUBLIC LICENSE AGREEMENT

1 Definitions

"Agreement" means this VTT's Software Public License Agreement.

"Contributor(s)" means any individual or entity that creates or contributes to a Modification of the Licensed Work.

"Licensee" means a legal entity who is exercising rights under this Agreement.

"Licensed work" means the software and/or any work related to the software.

"Modification" means any addition, deletion, change, improvement, or impairment to the Licensed Work and any translation (in natural or programming languages) of the Licensed Work. Modification may not be an identifiable separate part not derived from the original software.

"Binary Code" means machine readable and executable software code.

"Source Code" means machine and human readable and compile-ready software code.

"Original Contributor" means VTT.

"Recipient" means any legal entity receiving the software under this agreement

2 Grant of Rights

Contributors (Original and others) grant the Licensee a royalty-free, non-exclusive license, subject to third party intellectual property claims to use, copy, distribute and modify the Licensed Work for any legal purposes as stated in this Agreement.

3 Copying and Distributing

Licensee may copy and distribute verbatim copies of the Licensed work as long as the notice by the Original Contributor and this Agreement are intact. Licensee must include this Agreement with each and every copy of the Licensed Work. Licensee may charge a fee for the act of transferring a copy.

Licensee may copy or distribute the Licensed Work in Binary Code without the Source Code if the Source Code will by other means be available for the Recipient.

Licensee is free to release identifiable separate parts of his modifications that are not derived from the Licensed Work under a different license. These modifications must contain a clear identification of the Contributor and a notice of the applicable license.

4 Modifying

The Licensee may make Modifications to the Licensed Work, as long as he states who changed the files and the date of Modification; all Modifications are published under this license.

5 Termination

The License and rights granted under this Agreement are terminated automatically if the Licensee fails to comply any term or condition herein, if this Agreement may not comply with applicable law, if the use of the Licensed Work would infringe any intellectual property rights of any party (including third party), or if the distribution of the Licensed Work would violate any import and/or export laws or regulations.

Upon termination, the Licensee must destroy all copies of the Licensed Work in his possession.

6 Warranty

Contributors (Original and/or others) do not grant any warranty for the Licensed Work, to the extent permitted by applicable law. Original and/or other Contributors provide this Work free of charge on an "AS IS" basis without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose or that the Licensed Work will not infringe any intellectual property rights of any party (including third party).

Unless required by applicable law will none of the Contributors be liable to the Licensee for damages including any general, special, direct, indirect, incidental or consequential damages arising out of the use or inability to use the Licensed Work. The Licensee shall be solely liable for any and all damage incurred to a third party from the Licensee's use of the Licensed Work. **VTT Public License Version 0.5 11.11.2002**

2.Introduction

This document is a user's guide describing the usage of the Nipper network testing tool. For advanced usage and for further development there is an Advanced User Guide available.

In the first place Nipper was developed as a testing tool for VTT's internal projects. But since it is considered to be practical enough to be used also more generally, it is published as open source.

3.System requirements

The listed requirements are more or less approximations that are based on the usage experiences. Nipper is used in both AMD Athlon and Intel Pentium based desktop workstations running Linux Debian operating system. Resources of these systems vary between 400MHZ and 2GHz in processor speed and between 128Mb and 512Mb in amount of memory. Moreover, Nipper is briefly tested in Windows 2000 environment and discovered to be working with some limitations (see README.windows in the Nipper package).

Java 1.4.x or newer is a mandatory requirement and must be installed because entire Nipper is implemented using Java 1.4.0.

List of system requirements:

- Java 1.4.x or newer
- GNU Make
- GNU find
- awk
- Processor power at least 400MHz
- Amount of memory at least 128MB

We urge user's to test Nipper also on platforms with modest CPU or memory resources.

4.Downloading Nipper

Nipper tarball (Nipper-x.x.x.tar.gz) is available for downloading from VTT's open source website opensource.erve.vtt.fi.

5.Nipper Installation

After downloading, next thing to do is extract files from Nipper package. This is done by command: `tar -xzf Nipper-x.x.x.tar.gz`

This will create new directory named `Nipper-x.x.x` in your working directory.

Then go to `./Nipper-x.x.x/src` -directory and use your favorit editor to modify `Makefile.defs` to meet your system properties.

Finally run `make` in the `src`-directory. Now you should have compiled Nipper, please proceed to the next chapter.

6.Executing Nipper

In order to run Nipper you should give a command `make run` in the `Nipper-x.x.x/src`-directory. For RAW-sockets you should have a root privilege.

7.Packets and packet sets

Building packets and packet sets

In the Nipper environment, test packets are built by using a graphical user interface. Steps of building of a packet (set) are:

- Create a new packet.
- Set a sending time of the packet.
- Select required protocol header or payload data.
- Fill fields of the protocol header / payload data.
- Add required header / payload to the packet.
- Repeat until the packet is ready.

This list can be gone through until a required amount of packets are constructed.

More detailed packet building instructions

Creation of the packet

A new packet is created by simply clicking a *New packet*-button or selecting *New packet* from *Packet*-menu.

Selection of protocol headers / payload

Header / payload can be selected by selecting a proper tab. Tabs are divided to two levels. First one contains higher level protocols and another contains tabs that accede to one of the higher level tabs. Higher level tabs are:

- IPv4. IP version 4.
- Ipv6. IP version 6.
- ICMPv4.
- TCP/UDP TCP and UDP packets.
- MIPv4. Mobile IP version 4.
- Other. Packet's payload.

Filling fields of the protocol header

All of the fillable tabs have similar layout. The leftmost field is *Tag*-field, middle field is *Description*-field and the rightmost field is editable *Value*-field. Values can be altered by clicking the *Value* field with mouse and then typing a new value to it. The type of the value depends on the parameter, but the descriptions should be explanatory enough.

Adding headers to the packet

When fields are filled and header is ready to be added to the packet, select the packet to which the header is to be added. Then click *Add to packet*-button or select *Add to packet* from the *Packet*-menu.

Entry type	*Entry examples*
Booleans	TRUE; true; 1 / FALSE; false; 0
IP addresses	127.0.0.1; foo.bar.org
Integers	1; 0x1; 0x0000001f
Multiple values	Value, Value, Value;

Table 1Header entry examples

Setting a sending time

The packet's sending time can be set by right-clicking packet and selecting *Set sending time* from appearing pop-up menu. Another way to set sending time is to select *Set packet's sending time* from *Packet*-menu. Note that the sending time is in milliseconds and it can not be less than 1. It is also notable that the real sending times of the packets are not accurate. There can be errors of a few milliseconds.

Saving and loading packet (sets)

Packet sets can be saved for future use. Because it's quite boring and time consuming task to build packets, it is recommend to save all the sets you make. There are, in the *File*-menu same *Save* and *Save as* items as in almost all common applications. *Open* is also located in the *File*-menu. The functionalities of *Open* and both saves are obvious.

8.Sending packets

Now it is assumed that a packet or packet set is now ready for sending. That is, it is constructed from scratch or loaded from a previously made file. Next thing to do is to select the network interface. By default there are two choices; PacketDumper and RawSender. The first one is, as name tells, dummy and actually does nothing. It is suitable for testing of the Packet Editor and Packet Engine. The second one, RawSender, is the NIF that sends packets to the network. It is suitable for various testing anywhere there is a need for custom-made IP packets.

Figure 1 represents the network interface selection dialog window. The dialog appears when *Set network interface* from *Packetengine*-menu is selected. In the dialog window there are radio buttons that are used to select the interface and a text box that is used to set parameters for the selected network IF. Parameters are depending on the selected interface. Both of the currently implemented NIFs do not require any options.

Now we have packets ready and the interface selected. Next thing to do is to set the debug level and hit the *Send*-button. The debug level adjusts verbosity and it can be changed by clicking *Set debug level* in the *Packetengine*-menu. When the dialog (Figure 2) appears, pick the level and hit *OK*. Five (5) is the most informative level and 0-level prints out least. Packets are sent when *Send packet(s)*-button or *Send*

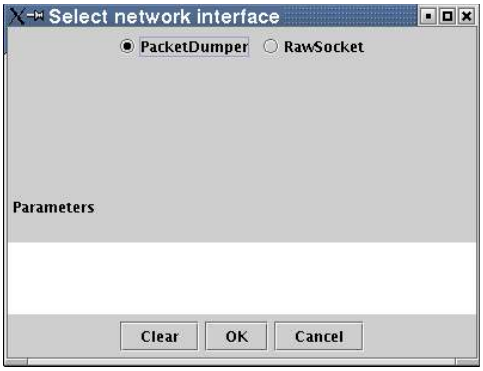

Figure 1 Interface selection dialog

packet from *Packet*-menu is clicked.

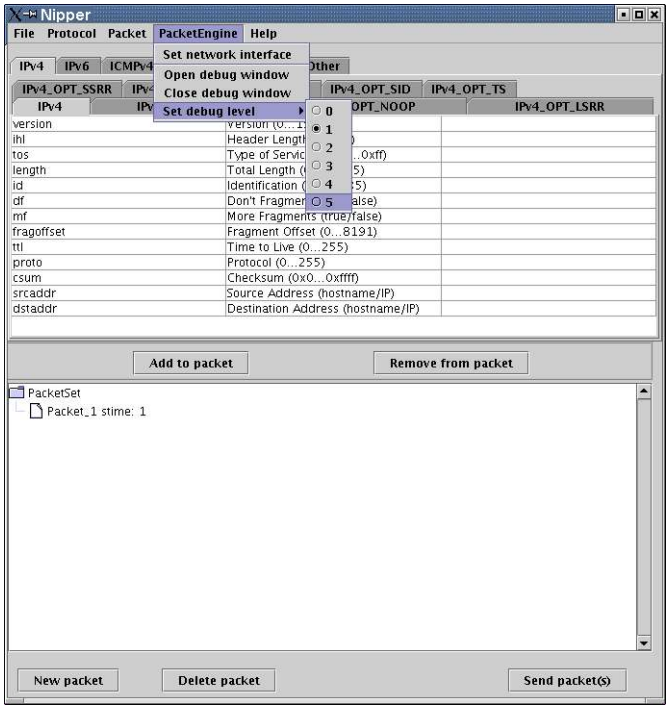

Figure 2 Debug level dialog

Information about state of Nipper, packets, sending times etc. is presented in the output window. Output window is viewed in Figure 3. The content of the window depends on the debug level and on the used network interface.

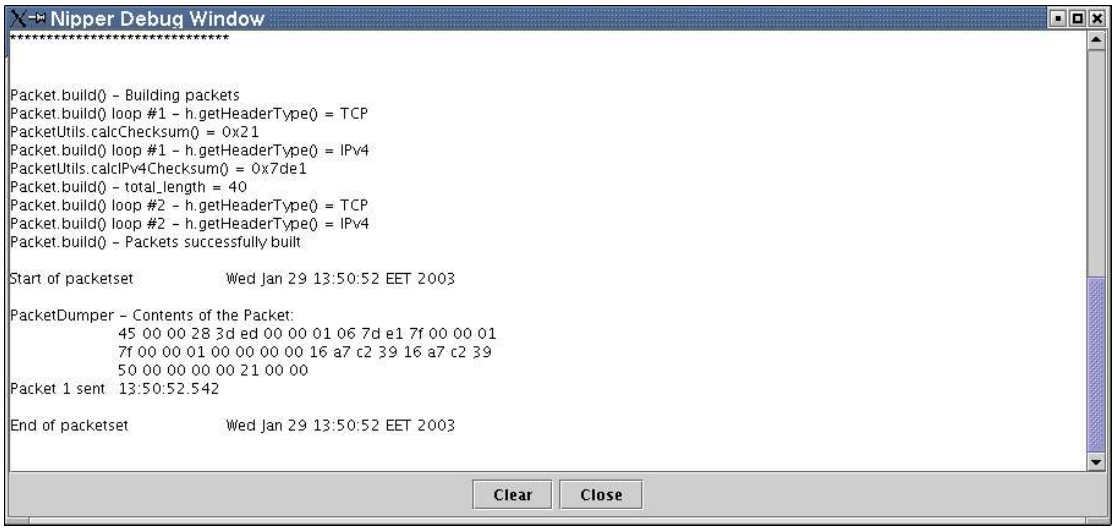

Figure 3 Nipper output window

9.An example

In this chapter we illustrates how to use Nipper with a little, simple, example. In the case, an example packet, presented in pictures below, is sent to the localhost. The packet is a simple IPv4 / TCP packet with a string as its payload.

In Figure 4 is IPv4 header presented. Source and destination addresses are set and other fields are left empty. Nipper automatically fills empty fields with proper default values.

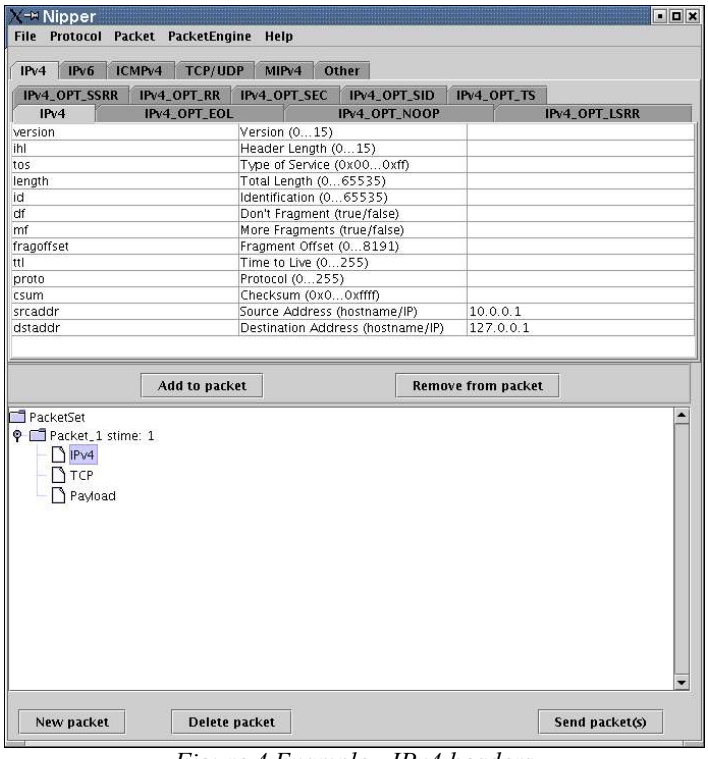

Figure 4 Example - IPv4 headers

Figure 5 presents a TCP header of the packet. As can be seen from figure, only source and destination ports are filled and the rest is left for Nipper to fill with defaults.

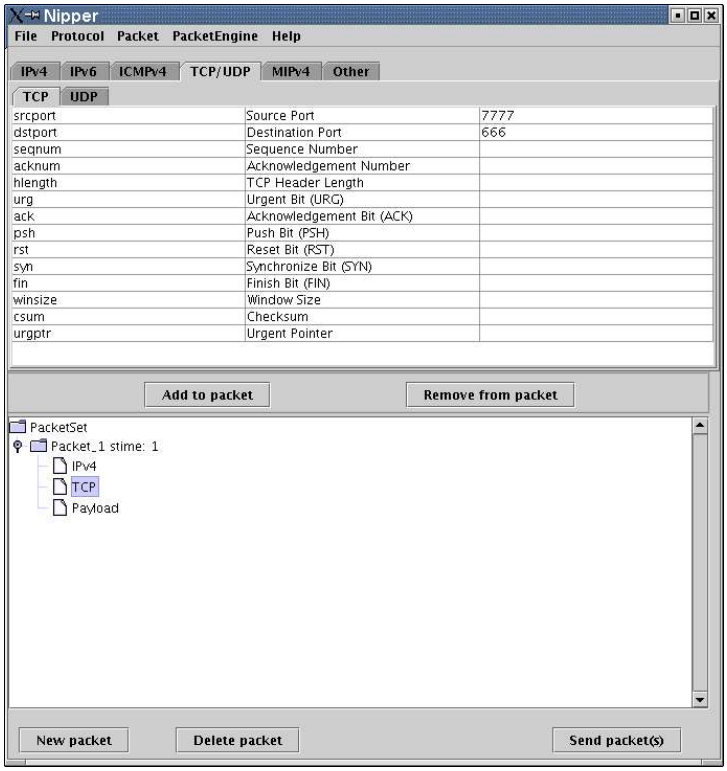

Figure 5 Example - TCP headers

Figure 6 presents the payload of the packet. In this case, the payload is a simple string "Hello cruel World". Two headers (IPv4 and TCP) plus payload and the packet is ready to be sent to the network.

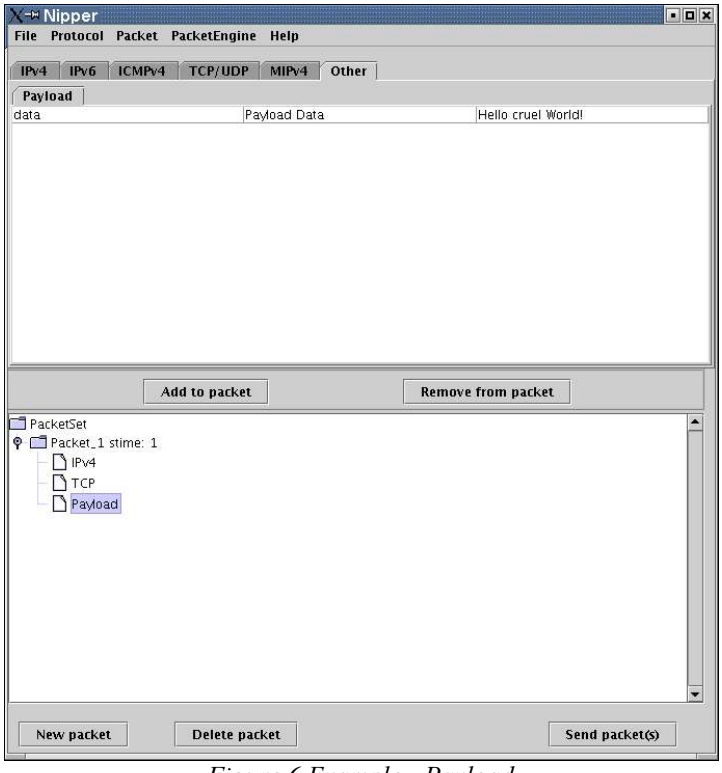

Figure 6 Example - Payload

Figure 7 presents the Nipper's output window in this example case. During the example run, the level of verbosity was 3.

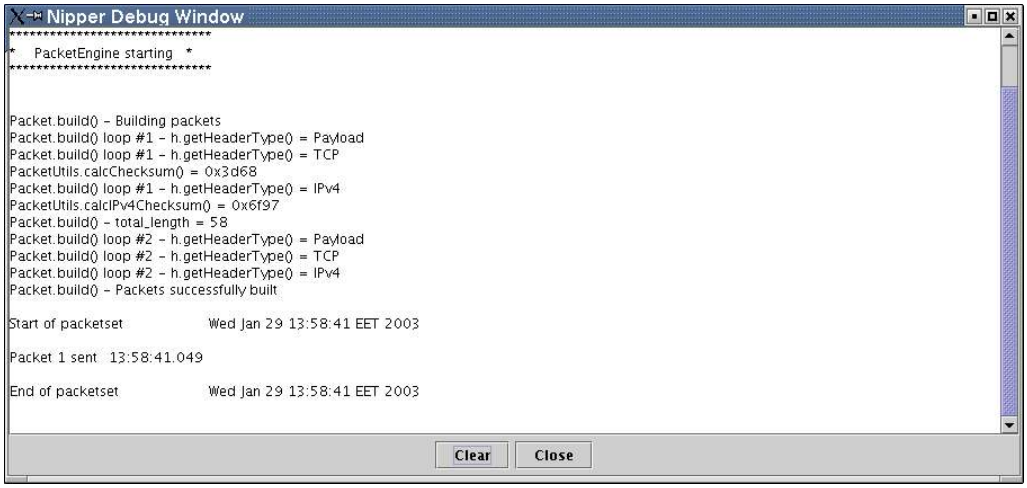

Figure 7 Example - Nipper output

Figure 8 presents a view in which the sent packet is captured and analysed. The packet is sent by using a RawSocket network interface. (Usage of RawSocket assumes root privileges.) The capture software is Ethereal. In the picture we can see that the packet (destination address 127.0.0.1 == localhost) has arrived and TCP stack has replied with a reset.

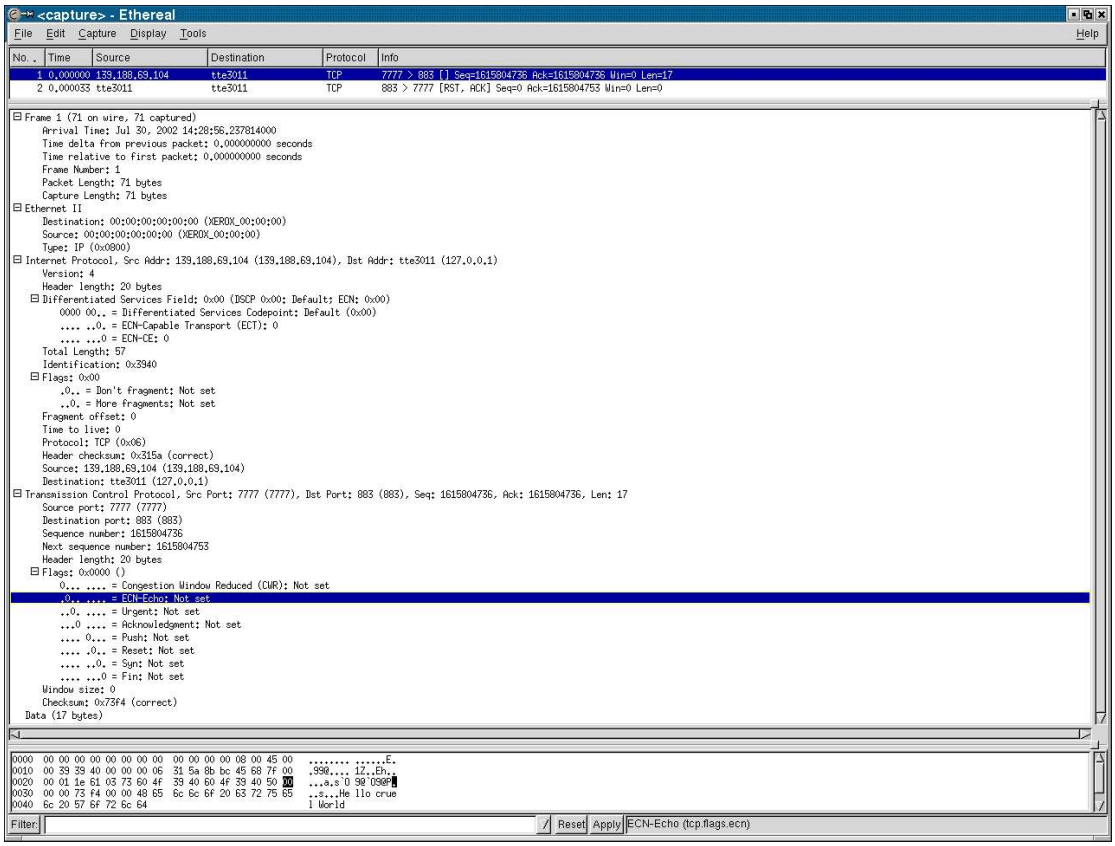

Figure 8 Example - Ethereal capture