A Tcl Language Extension for Accessing and Transmitting Data Link Layer Frames

Hari T.S. Narayanan[1]
Vasumathi Narayanan[2]
SP. Sathappan
St. Josephs College of Engineering, Chennai, India

Abstract

Tcl is a *simple,* interpreted, scripting language used for developing tools, prototypes, and utilities for networking besides other areas. Tcl offers an extension framework using which it can be made to offer domain or application specific commands. In this paper we describe an extenion to Tcl language by the addition of a set of commands (an interface) for *capturing and transmitting raw frames*. Unlike *socket* interface, this interface provides a lower layer access to networking data. The set of new commands added to Tcl are supported on all popular platforms (Windows, Linux, and Unix). Using this new set of commands one can develop networking tools, *cross layering* features, applications and prototypes with *faster* turn around time.

Our contribution includes specifying the syntax and semantics of this Tcl extension, designing and developing the command engine. The engine itself is developed using WinPcap and/or LibPcap C libraries. The extension is platform independent and supports frame transmit on platforms where it is supported by the corresponding packet capture C library.

[1] CEO, Netprowise, Chennai, India
[2] Director, Netprowise, Chennai, India

1 Introduction

Tool Command Language [1], Tcl, is an interpreted language used extensively in developing scripts for testing and monitoring networking devices. Besides networking, Tcl is used in other domains for its simplicity and shorter learning curve. Tcl language offers an extension framework [2], which is used to offer application specific commands. There are several such extensions to Tcl both in networking and non-networking areas. *Expect*, *Tk, Tnm/Scotty* are some of the popular Tcl extensions. A comprehensive list of this can be found in *Tcl/Tk Extensions & Information* site.

2 Motivation

Most networking applications access network through operating system primitives such as sockets. Using socket one can send a message from a host and receive a message *directed* to a host. It is easier to send and receive messages this way because:

- Operating system copes with the low level details (protocol handling, packet reassembly, etc.).
- Socket provides a familiar interface that is similar to the one used to read and write files.

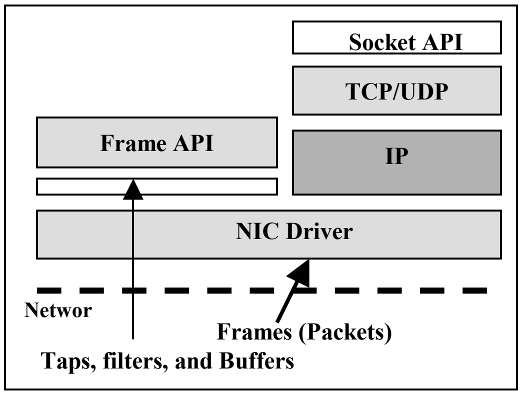


Fig 1. Frame API versus Socket API

Sometimes, however, the 'easy way' is not up to the task, since some applications require direct access to frames sent to and received from the network. That is, they need access to the "raw" frames of *data link layer* of the network without the interposition of protocol processing by the operating system. This data includes even the messages that are not necessarily sent from or directed to the host where these applications run. This issue is already addressed by a set of C libraries, WinPcap and Libpcap [3-5].

We plan to address the above shortcoming for Tcl in this paper. To be consistent with the related literature [3-5], we will use the term *packet* even though *frame* is more accurate because the capture process is done at the data-link layer. The data-link header is included in the captured data.

The set of Tcl features that are developed include:

1. Capturing raw packets, both the ones destined to the machine where it's running and the ones exchanged by other hosts (on shared media)
2. Transmitting raw packets to the network
3. Filtering the packets according to user-specified rules.
4. Gathering statistical information on the network traffic
5. Accessing network from concurrently running applications.
6. And a number of other related features

The Tcl command engine will be using either WinPcap or Libpcap depending upon the platform. The code developed can be compiled to run on multiple platforms (Windows, Linux, Unix, Mac, BSD, etc.) by following the Tcl Extension Architecture (TEA).

3 <u>A short Survey</u>

The problem of developing a Tcl extension for packet capture is suggested in one of our earlier works [6]. Since then we have cited similar efforts to extending Tcl with either libpcap or WinPcap. Jose Nazario has written a Tcl-pcap interface (*tcap*) [7,8]. This is built and tested with libpcap in BSD UNIX. Craig French has developed a Tcl Interface for Windows environment using WinPcap [9]. We have developed a platform independent packet capture and transmission using WinPcap/Libpcap. Cliff Flynt has developed a library to writing packets in Tcl[10]. This library could be used in conjunction with our Tcl extension to transmit packets.

4 <u>Design Objective</u>

As a programming language Tcl is endowed with certain inherent characteristics. These characteristics differentiate Tcl from other programming languages. For instance, Tcl's short learning curve comes from its simple grammar. This characteristic makes it a desirable language for building utilities, tool, and prototypes. The extension features that we add keep the syntax simple to a large extent. Most of the commands work with a single parameter.

5 <u>Extension Commands</u>

Every pcap extension command is preceded by the keyword, *pcap* to indicate that it is an extended Tcl command. The term *pcap* is followed by a command option (*list, open, close*, etc.). The Tcl extension to access data link layer (networking adapters) is similar to a file access using a handle.

In general *pcap* extension commands return error value using Tcl extension architecture. Using this architecture both error values and normal values are returned.

5.1 Listing Network Adapters

Typically, the first thing that a user likes to do is to identify the list of networking adapters or interfaces in a host. The *pcap list* (extension) command supports this requirement as follows:

```
pcap list
```

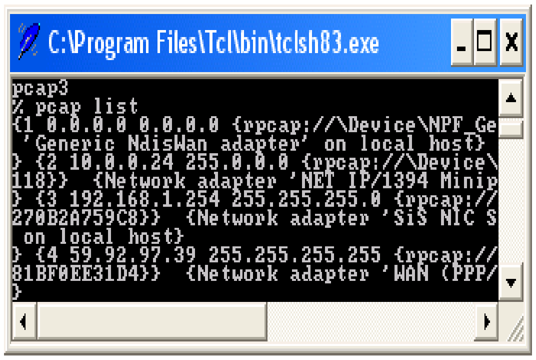
Parameters:

No parameters for the *list* command in this release. In future this could be enhanced to list the interfaces on a remote host.

This command returns the list of networking interfaces hosted by the platform. The return value for the *list* command is a (Tcl) list of *interfaces*. Each interface in the list includes the following attributes in order:

1. *Interface number*
2. *IP address of the interface*
3. *Net Mask*
4. *Interface ID*
5. *Interface description*

The list and individual items are enclosed within curly braces as shown below:



```
pcap3
% pcap list
{1 0.0.0.0 0.0.0.0 {rpcap://\Device\NPF_Ge
 'Generic NdisWan adapter' on local host}
} {2 10.0.0.24 255.0.0.0 {rpcap://\Device\
118}} {Network adapter 'NET IP/1394 Minip
} {3 192.168.1.254 255.255.255.0 {rpcap://
270B2A759C8}} {Network adapter 'SiS NIC S
 on local host}
} {4 59.92.97.39 255.255.255.255 {rpcap://
81BF0EE31D4}} {Network adapter 'WAN (PPP/
}
```

The *interface number* is provided for ease of use. *Interface number* is easier to specify in a Tcl command shell rather than the long, unique, alphanumeric *interface identity*. *Interface numbers* are likely to change when adapters are hot-swapped. So it must be used with care. The interface identity is unique and recommended for identifying an interface.

If the command fails, then the error status is set to *TCL_ERROR* and the return value includes a description of the error condition. Otherwise, the error status is set to TCL_OK and a response is returned.

5.2 Opening an interface

The extension command that opens an interface for packet capture or transmission is *pcap open*. This command is also used to open *pcap* files for reading in the packet data. The pcap *open* command of Tcl simply readies the specified interface for capturing and/or transmitting packets. The interface is specified using one of the four possible parameters

```
pcap open
    -if <ifnum> |
    -ipaddress <ipaddress> |
    -id <interfaceid> |
        -infile <filename>
```

This command when successful returns an interface handle. This handle is a positive integer prefixed with *pcap*. It is used to capture packet and/or transmit packets on that interface. This handle reduces the possible errors in typing in the long interface name. More than one handle can be associated with an interface at a given time.

The same *open* command can also be used to open a file using –infile option. The file should have been created with *tcpdump* format either by pcap or any similar application. The handle returned is used with pcap *capture* and *loop* commands. The pcap *send* and *stats* commands are not applicable for captured files.

Parameters:

-if (interface)

Specifies an interface using the number listed in the output of *pcap list* command.

-ipaddress

Specifies an interface using its IP address.

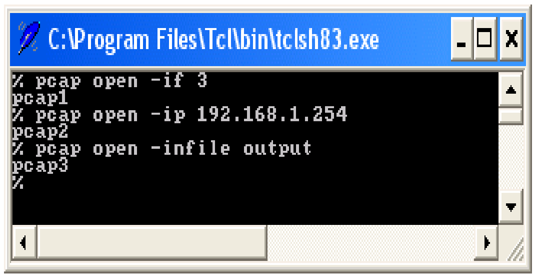
-id

Specifies an interface using its alphanumeric Identity. Pcap list command lists identities of all the interfaces on a given host.

-infile

Specifies a *tcpdump* file name. The filename specification is limited to 1024 characters. Full path name of the file is required.

If this command fails, then the error status is set to TCL_ERROR and the return value includes a description of the error condition. Otherwise, the error status is set to TCL_OK, and this command returns an interface handle.



```
% pcap open -if 3
pcap1
% pcap open -ip 192.168.1.254
pcap2
% pcap open -infile output
pcap3
%
```

5.3 Updating Handle Attributes

In order to simplify the use of *pcap* commands, attributes of a handle are updated using *set* command rather than specifying the attributes with *open* and other handle-based commands. This separation simplifies the syntax. But at the same time it needs invocation of one or more *set* commands to update the handle attributes before an *action* command (*loop, capture, stats, and send*) is invoked.

```
pcap set handle
    [-callback <cbscript>]
    [-count <pktcount>]
    [-filter <filterstr>]
    [-format hexa|decimal]
    [-outfile <filename>]
    [-promiscuous y/n]
    [-snaplen <int>]
    [-timeout <readtimeout>]
```

All of the above parameters are optional. The *promiscuous* parameter is used to capture either all packets in the wire or those directed to and leaving from the host. All packets are captured by default. This parameter is set to '*y*' to capture packets in *promiscuous* mode. The *snap length* indicates the maximum size of the packet to be returned. This is in bytes, and read *timeout* is in milli-seconds. The default value for snap length is 65535 bytes, and default timeout is 1 second (1000 msec). The filter specification uses the syntax described in http://www.tcpdump.com Callback script is any legitimate Tcl script. Using the *outfile* option, one can redirect the output to a file. This file saves the captured packets in *tcpdump* file format. The *count* parameter is used to specify the number of packets to capture. The *format* option allows the packet data to be represented in either *hexadecimal* or *decimal* form for both sending and receiving. The default format is hexadecimal.

-callback

Specifies the Tcl script to be invoked when the interface driver returns a packet. There is no default value for this parameter. This parameter is used only with *loop* command.

-count:

Specifies the number of packets to be captured with *loop* or *stats* command. By default this is set to 0 – indicates that the user sets no limit. The range of count is 0 to $2^{32}-1$. This parameter is used in *pcap-loop and pcap-send* commands.

-filter:

Specifies the filter used in capturing. It follows the *tcpdump* filter format. The size of the filter specification is limited to 1024 characters including the space. This parameter is used for *capture*, *loop, and stats* commands.

-format

The *format* parameter allows the packet data to be represented in either hexadecimal or decimal form for both sending and receiving. The default format value is hex. This parameter is used for *capture*, *loop, and send* commands.

-outfile

This option *is used to* specify the filename. The captured packets are redirected to this file. This file saves the captured packets in *tcpdump* format. The default file name is *<handle>DumpFile. The file appears* in Tcl/bin(Windows) or current directory where Tcl script is launched. The file name is limited to 1024 characters. This parameter is used for *loop* command.

-promiscuous

The *promiscuous* parameter is used to capture either all packets that are seen by the interface or those that are sent and received by the host. All packets are captured by default; this is indicated by *promiscuous* mode of a handle set to *y*. The possible values are *y* and *n*. This parameter is used for *capture*, *loop, and stats* commands.

-snaplen

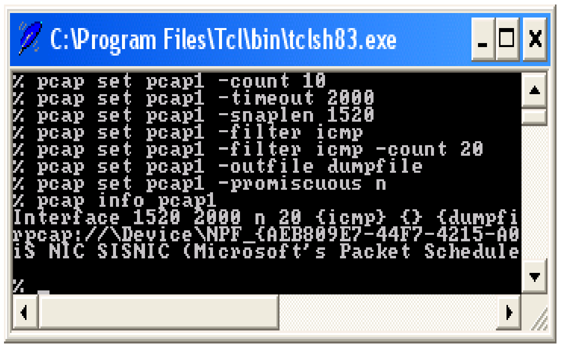
Specifies the maximum size of the captured packet in number of bytes. By default this is set to 64K. The range for *snaplen is 0* to 2^{32}-1. This parameter is used with *capture* and *loop* commands.

-timeout

The maximum time to wait for the arrival of a packet for 'capture' command. This is not applicable to other action commands. If no packet arrives within the specified time then, the command (*capture*) returns.

By default this is set to 1 second. The resolution is milliseconds.

The return value is 0 (TCL_OK) if the command is successful. Otherwise, the *error status* is set to TCL_ERROR and the return value includes a description of the error condition. The pcap *info* command is used to list the attributes of a handle.



```
% pcap set pcap1 -count 10
% pcap set pcap1 -timeout 2000
% pcap set pcap1 -snaplen 1520
% pcap set pcap1 -filter icmp
% pcap set pcap1 -filter icmp -count 20
% pcap set pcap1 -outfile dumpfile
% pcap set pcap1 -promiscuous n
% pcap info pcap1
Interface 1520 2000 n 20 {icmp} {} {dumpfi
rpcap://\Device\NPF_{AEB809E7-44F7-4215-A0
iS NIC SISNIC (Microsoft's Packet Schedule

% _
```

5.4 Capturing Packets

Tcl extension provides two different capture commands. The first command, *pcap capture*, is a simple Tcl extension that offers synchronous packet capture. The only input for this command is an interface handle. This command is blocked on Tcl event-loop until a packet arrives at the specified interface or a timeout occurs. If this command is successful, a list with two items is returned. The first item in the list is packet header and the second item in the list is a partial or full packet-dump. The packet header is a list with three items: *time stamp*, length of the packet-dump, and the length of this packet (off wire).

pcap capture handle

pcap loop handle

The second Tcl extension, *pcap loop*, includes an option to use a callback function. This callback function is invoked when a packet is received. There is no timeout associated with *loop* command. This command returns after the specified number of packets arrive. The *count and callback* parameters are applicable only for loop command. That is, they are not used for capture command. Similarly, the output file is used only with the *loop* command.

The *loop* command also populates two Tcl global variables %H and %D. The %H is the header information and %D is the packet data. The %H includes timestamp with the following format: "HH:MM:SS microseconds". Packet length and captured length follow the timestamp. Both are in number of bytes. The %D contains the packet data. Each byte value is represented by a hexadecimal or a decimal number and separated from each other by a space.

The filter specification and promiscuous mode are used for both *loop* and *capture* commands. The filtering specification is same as the one used in *tcpdump* syntax. The original version of this specification can be found at www.tcpdump.org. Filters are based on a declarative predicate syntax. A filter is a string containing a filtering *expression*. The expression selects which packets to be captured. If no expression is given, all packets on the net will be captured. Otherwise, only packets for which *expression* is `true' are captured.

Both the commands work seamlessly with *tcpdump* files. Packets are retrieved from the specified *tcpdump* file instead of an interface. If successful, this command returns TCL_OK, otherwise returns an appropriate error code with the error status set to TCL_ERROR.

Breaking the loop: If the packet count attribute of the handle is set to 0 then the *pcap loop* continues perpetually. Otherwise, it terminates after the specified number of packets are captured. The only way to break the loop is to set the global variable ***pcap_bloop*** to 1. By default, this variable starts with the value 0. The engine checks the value of this variable when the callback script is called to see if the loop needs to be terminated. If the value is non-zero then the loop is terminated, otherwise it continues. This type of interruption works only in Windows platform at present.

Parameter:
<handle>

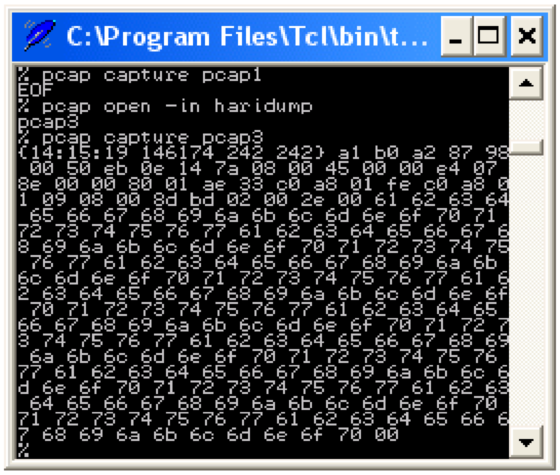
The *handle* is a mandatory parameter for both the commands. This handle can represent an *interface* or a *tcpdump* file

The response to capture includes two Tcl lists.

The first list contains three attributes:

- Timestamp: HH:MM:SS and microseconds
- Length of the packet off wire.
- Length of the packet presented (different from the length of the packet off wire).

The second list contains the packet data. Each byte value is represented by hexadecimal or decimal and separated from each other by a space.



```
% pcap capture pcap1
EOF
% pcap open -in haridump
pcap3
% pcap capture pcap3
{14:15:19 146174 242 242} a1 b0 a2 87 98
 00 50 eb 0e 14 7a 08 00 45 00 00 e4 07
8e 00 00 80 01 ae 33 c0 a8 01 fe c0 a8 0
1 09 08 00 8d bd 02 00 2e 00 61 62 63 64
 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71
72 73 74 75 76 77 61 62 63 64 65 66 67 6
8 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75
 76 77 61 62 63 64 65 66 67 68 69 6a 6b
6c 6d 6e 6f 70 71 72 73 74 75 76 77 61 6
2 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f
 70 71 72 73 74 75 76 77 61 62 63 64 65
66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 7
3 74 75 76 77 61 62 63 64 65 66 67 68 69
 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75 76
77 61 62 63 64 65 66 67 68 69 6a 6b 6c 6
d 6e 6f 70 71 72 73 74 75 76 77 61 62 63
 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70
71 72 73 74 75 76 77 61 62 63 64 65 66 6
7 68 69 6a 6b 6c 6d 6e 6f 70 00
%
```

5.5 Transmitting Packets

The pcap send command transmits a raw packet through the specified networking interface. The MAC CRC doesn't need to be included, because it is transparently calculated and added by the network interface driver. The return value is 0 (TCL_OK) if the packet is transmitted successfully; otherwise it returns appropriate TCL_ERROR status. User program should rely on the error status to interpret the return value.

The first parameter is an interface handle. The second parameter is a hexadecimal or decimal byte value string separated by space. The *send* command is not supported for handles that represent *tcpdump* files. This command is also not supported on non-Windows platforms.

pcap send handle "pktstring "

Parameters:
<handle>

The handle is a mandatory parameter. It represents only an interface.

<pktstring>

The packet data is a hexadecimal or decimal byte value string separated by space. The *format* attribute is used to specify whether the value is in hexadecimal or decimal form.

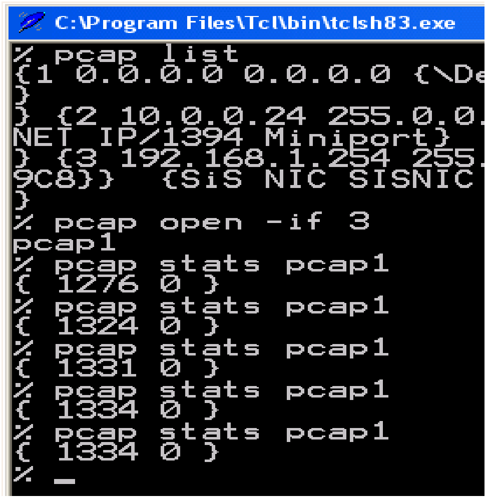
5.6 Listing Statistics

The pcap *stats* command implements the function to retrieve statistics for an interface (handle). At present the following two counters are supported: number of frames received by the *interface* (sent and received to/from network*)*, and the number of frames dropped due to errors.

This command needs only one parameter, that is, pcap handle The Tcl *stats* command returns a list with two counter values. If there is any error in completing the command then an empty or NULL list is returned with error status set to TCL_ERROR.

pcap stats handle

Parameters:
<handle>
The *handle* is a mandatory parameter. This handle can represent only an *interface.* That is, there are no statistics for files.


C:\Program Files\Tcl\bin\tclsh83.exe

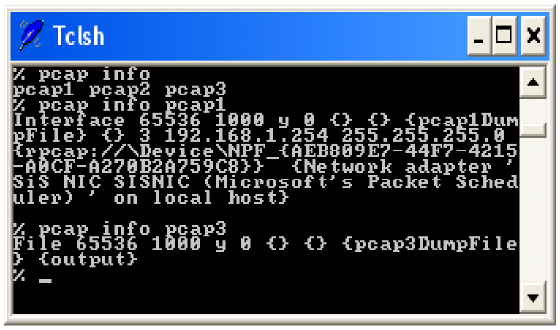
```
% pcap list
{1 0.0.0.0 0.0.0.0 {\De
}
} {2 10.0.0.24 255.0.0.
NET IP/1394 Miniport}
} {3 192.168.1.254 255.
9C8}}  {SiS NIC SISNIC
}
% pcap open -if 3
pcap1
% pcap stats pcap1
{ 1276 0 }
% pcap stats pcap1
{ 1324 0 }
% pcap stats pcap1
{ 1331 0 }
% pcap stats pcap1
{ 1334 0 }
% pcap stats pcap1
{ 1334 0 }
% _
```

5.7 The *info* Command

The pcap *info* command provides detailed information about the currently opened handles and their attributes. The info command syntax is shown in the following Fig. If no handle is specified in the command then currently open handles are listed (Tcl list), otherwise the attributes of the specified handle are listed (Tcl list).

pcap info [handle]

Parameters:
<handle>
The *handle* is an optional parameter. This handle either represents an *interface* or a *tcpdump* file.


Tclsh

```
% pcap info
pcap1 pcap2 pcap3
% pcap info pcap1
Interface 65536 1000 y 0 {} {} {pcap1Dum
pFile} {} 3 192.168.1.254 255.255.255.0
{rpcap://\Device\NPF_{AEB809E7-44F7-4215
-A0CF-A270B2A759C8}}  {Network adapter '
SiS NIC SISNIC (Microsoft's Packet Sched
uler) ' on local host}

% pcap info pcap3
File 65536 1000 y 0 {} {} {pcap3DumpFile
} {output}
% _
```

The pcap info command lists the following values for a handle in Tcl List form:

- Handle type (Interface)
- Snap length
- Read Timeout
- Promiscuous option (y/n)
- Packet Count
- Filter String
- Callback Script
- Output Dump Filename
- Interface Number
- IP Address
- Net mask
- Interface ID

- Interface Description

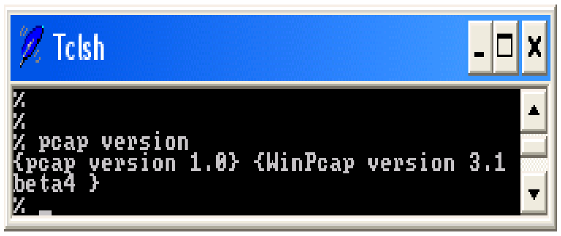
The pcap info command lists the following values for a *file* type in Tcl list form:

- Handle type (File)
- Snap length
- Read Timeout
- Promiscuous option (y/n)
- Packet Count
- Filter String
- Callback Script
- Output Dump Filename
- Input Filename

5.8 The *version* Command

The pcap *version* command provides the version number of pcap extension and WinPcap/Libpcap libraries. The syntax is as follows:

pcap version



```
%
%
% pcap version
{pcap version 1.0} {WinPcap version 3.1
beta4 }
% _
```

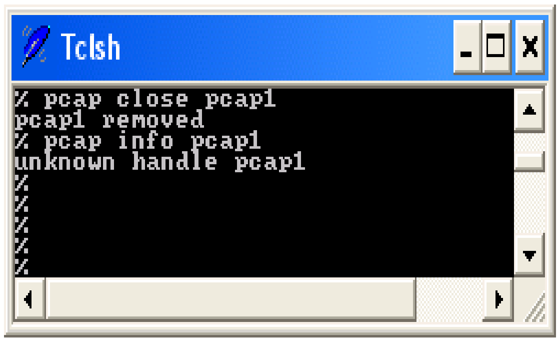
5.9 The *close* Command

The pcap *close* command removes a handle from active use and releases all its resources. It also closes the corresponding output file (e.g. pcap1DumpFile) if there is one.

pcap close handle

Parameters:
<handle>
The *handle* is a mandatory parameter. This handle either represents an *interface* or a *tcpdump* file.

The return value is 0 (TCL_OK) if the handle is closed successfully; otherwise it returns appropriate error message.



```
% pcap close pcap1
pcap1 removed
% pcap info pcap1
unknown handle pcap1
%
%
%
%
%
```

5.10 The *help* Command

The pcap *help* command lists all pcap commands with brief help messages.

pcap –help

6 Status, Limitations, and Future work

The *pcap* extension includes packet capture and packet injector (transmit) for networking interface; packet capture for *tcpdump* files, and a host of other commands. The Tcl language extension is tested on Windows, Linux, and BSD UNIX platforms. The command execution examples presented throughout this paper are Windows screen shots.

One must have the *root* user privilege to use pcap in Unix like platforms. The number of networking interface and the number of open handles are limited to 16 at present. The packet send command is supported only on Windows and there is no support for callback with stats command.

This Tcl extension enables a class of networking tools to be *easily* developed. We plan to develop some of these tools ourselves. A simple, demonstration tool is listed in Appendix B. A similar extension to *Python* language has been initiated already. Our future development also includes remote packet capture feature and an extension to stop packets reaching higher level layers.

7 Acknowledgement

This development has been made possible by the support extended by *Netprowise* and *St. Josephs College of Engineering, Chennai, India*. We thank these two organizations for their support.

8 References

1. John Ousterhout's "*Book Tcl and the Tk Toolkit*", Chapters 28-34, Addison-Wesley Professional.
2. The Tcl Extension Architecture, Brent Welch, Michael Thomas *Scriptics Corporation* http://www.tcl.tk/doc/tea/
3. Fulvio Risso, Loris Degioanni, An Architecture for High Performance Network Analysis, *Proceedings of the 6th IEEE Symposium on Computers and Communications (ISCC 2001)*, Hammamet, Tunisia, July 2001
4. Loris Degioanni, Mario Baldi, Fulvio Risso and Gianluca Varenni, Profiling and Optimization of Software-Based Network-Analysis Applications, *Proceedings of the 15th IEEE Symposium on Computer Architecture and High Performance Computing (SBAC-PAD 2003)*, Sao Paulo, Brasil, November 2003
5. Loris Degioanni, Development of an Architecture for Packet Capture and Network Traffic Analysis, Graduation Thesis, Politecnico Di Torino (Turin, Italy, Mar. 2000)
6. Hari and Barani, *Projects in Networking,* SciTech Publishing, Chennai, India. Dec, 2005.
7. Jose Nazario's *tcap download*: http://monkey.org/~jose/software/tcap/
8. Jose Nazario's *tcap*: http://wiki.tcl.tk/13515
9. Craig French's Tcl Interface to WinPcap: http://wiki.tcl.tk/13150
10. Clif Flynt's packet writer: http://noucorp.com/tcl/login/L/packetmstr0_1.zip

9 Appendix A:

Installation on Windows Platform:

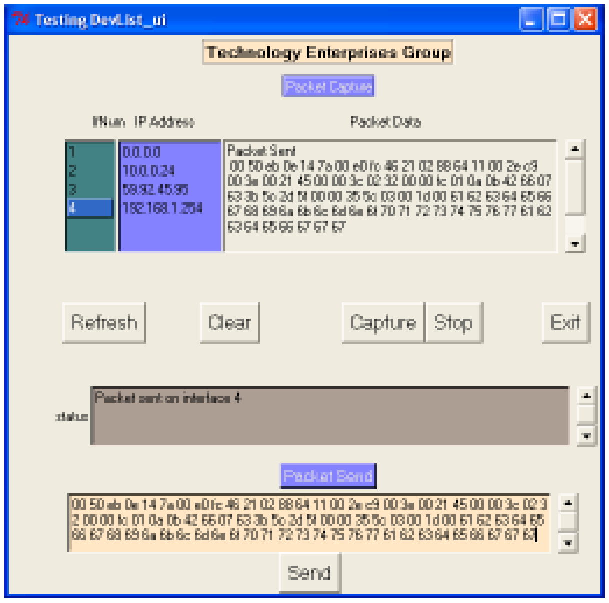
1. Install WinPcap
2. Install Tcl, preferably Scotty
3. Copy pcap.dll to Tcl/bin directory
4. Start Tcl shell
5. Issue the following command to load pcap library. Note: The following command is case sensitive:

```
load pcap.dll Pcap
```

6. Start using pcap commands:

10 Appendix B

The following figures show a live Packet Capture tool developed using platform independent Tk and *pcap* extension. Packets are sent from the top window and captured by the lower window. This tool includes about 1000 lines of code developed in about 2 days by someone who knows Tcl well but a beginner toTk. Our current distribution includes pcap.dll, some demo tools, and pcap user documentation.

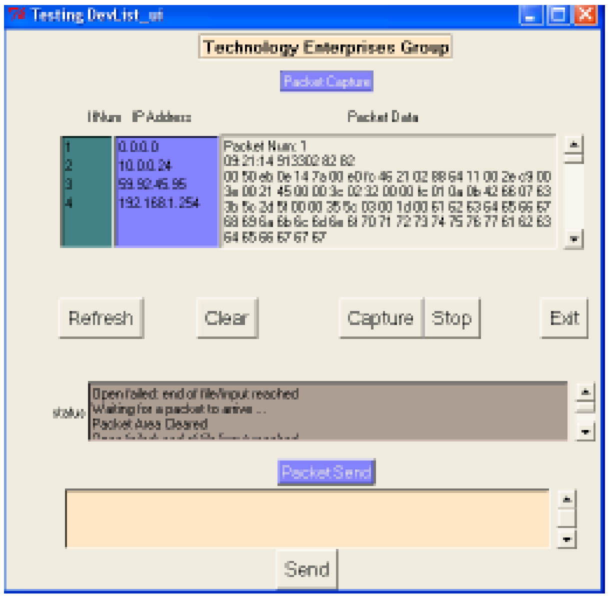


IfNum	IP Address
1	0.0.0.0
2	10.0.0.24
3	59.92.45.95
4	192.168.1.254

```
Packet Sent
 00 50 eb 0e 14 7a 00 e0 fc 46 21 02 88 64 11 00 2e c9
00 3e 00 21 45 00 00 3c 02 32 00 00 fc 01 0a 0b 42 66 07
63 3b 5c 2d 5f 00 00 35 5c 03 00 1d 00 61 62 63 64 65 66
67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75 76 77 61 62
63 64 65 66 67 67 67
```

```
Packet sent on interface 4
```

```
00 50 eb 0e 14 7a 00 e0 fc 46 21 02 88 64 11 00 2e c9 00 3e 00 21 45 00 00 3c 02 3
2 00 00 fc 01 0a 0b 42 66 07 63 3b 5c 2d 5f 00 00 35 5c 03 00 1d 00 61 62 63 64 65
66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75 76 77 61 62 63 64 65 66 67 67 67
```



IfNum	IP Address
1	0.0.0.0
2	10.0.0.24
3	59.92.45.95
4	192.168.1.254

```
Packet Num: 1
09:21:14 913302 82 82
00 50 eb 0e 14 7a 00 e0 fc 46 21 02 88 64 11 00 2e c9 00
3e 00 21 45 00 00 3c 02 32 00 00 fc 01 0a 0b 42 66 07 63
3b 5c 2d 5f 00 00 35 5c 03 00 1d 00 61 62 63 64 65 66 67
68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75 76 77 61 62 63
64 65 66 67 67 67
```

```
Open failed: end of file/input reached
Waiting for a packet to arrive ...
Packet Area Cleared
```