Universal Packet Analyzer - A Network Packet Filtering tool

Sriram K Parameswar and Udo W. Pooch

Department of Computer Science, Texas A&M University,
College Station, Tx - 77843-3112.

Technical Report 96-008 (TR_96-008)
March 1996

Abstract

The Universal Packet Analyzer is a Unix user program that filters a network packet and reports its contents. The term "Universal" is used to indicate that this program is not limited to a particular protocol stack nor is it limited by the number of layers.
The software has an intuitive windows-based graphical interface and is mouse driven. The input is a Hex packet stream read from user designated files and the output is the filtered packet, which can be viewed either online or offline. The code is written in 'C' and the graphical user interface was built using devguide, *the OpenWindows Software Developer's Guide.*
Packets are defined with the aid of a simple scripting language called the Packet Definition Language (PDL) [1] and [2]. The PDL has constructs that support the layered architecture concept of networks. The user can inspect the packet at any layer using filters. Filters are built up from filter clauses, written in a simple scripting language called the Filter Definition Language (FDL) [1] and [2].
The package has a number of supplementary services that support the main activity of filtering packets. The user has the capability of saving and naming the current configuration of packet definitions and filters. The user can return to this configuration by name, thus saving time. The user can build and edit both filters and packet definitions on the fly.

1 Introduction

Networks constantly need to be monitored, to find and eliminate bottlenecks, and for security purposes. A packet analyzer can be a useful tool to achieve these vital functions.
The Universal Packet Analyzer is a Unix user program that filters a network packet and reports its contents to the user. The term "Universal" is used to indicate that this software is not limited to a particular protocol stack like the DARPA stack nor is it limited by a specific number of layers, e.g., the seven layers of the ISO/OSI model. The user is free to inspect the packet at any layer. The entire program has an intuitive windows-based graphical interface and is mouse driven.
Research indicated that there exist some packages with similar functionality to that of the Universal Packet Analyzer, but all of them have limitations. Most packages are specialized and are targeted

towards specific protocols and architectures. The package *snoop* which comes with SunOS 5.x (Solaris) is completely line oriented and is limited to the Internet protocol stack. The Universal Packet Analyzer can handle any type of packet and provides the ease of a point and click interface. All that is required to use the Universal Packet Analyzer to filter various protocols (even proprietary protocols) is that packet definition files be set up, which is a one time effort.
Most packages in the market are dependent on the Data link layer: they target certain network protocols like ethernet or token ring. The Packet Analyzer can handle any Data link layer, including ethernet, FDDI, and any other that may be developed in the future. This would require new packet definitions and some packet capturing tool. This program allows the easy addition of new protocols using the scripting languages.
The introduction is followed by some of the features supported by this program. This program is based on the layered approach to computer communication protocols. Section 3 discusses this briefly. This is followed by an introduction to the PDL and FDL scripting languages. The services are explored in greater detail in Section 5.

2 Some Features

The program is feature-rich and has been designed to minimize the learning curve. Some of the features incorporated in this software are listed below:

- Intuitive point and click interface
- Packet Format description language
- Filter description language
- Filtering of Hex packet dumps - singly or as a stream
- Saving and loading pre-defined configurations
- Status indication and monitoring
- On-line and off-line viewing of results

Packet formats are described using a simple scripting language; a graphical user interface (GUI) and context-sensitive help make it an easy task. A large number of popular packet formats are pre-defined for the user, e.g., TCP, Sun and DCE RPC, etc. Filters use an extremely simple scripting language; a GUI tool makes it simple to define new filters.
The user can define and load commonly used configurations of packet definitions and filters. Consider the scenario where the user is filtering a large number of HEX packet dumps over a period of several days. The loading of a single hierarchical file saves the user the trouble of having to load several packet definitions and filters.

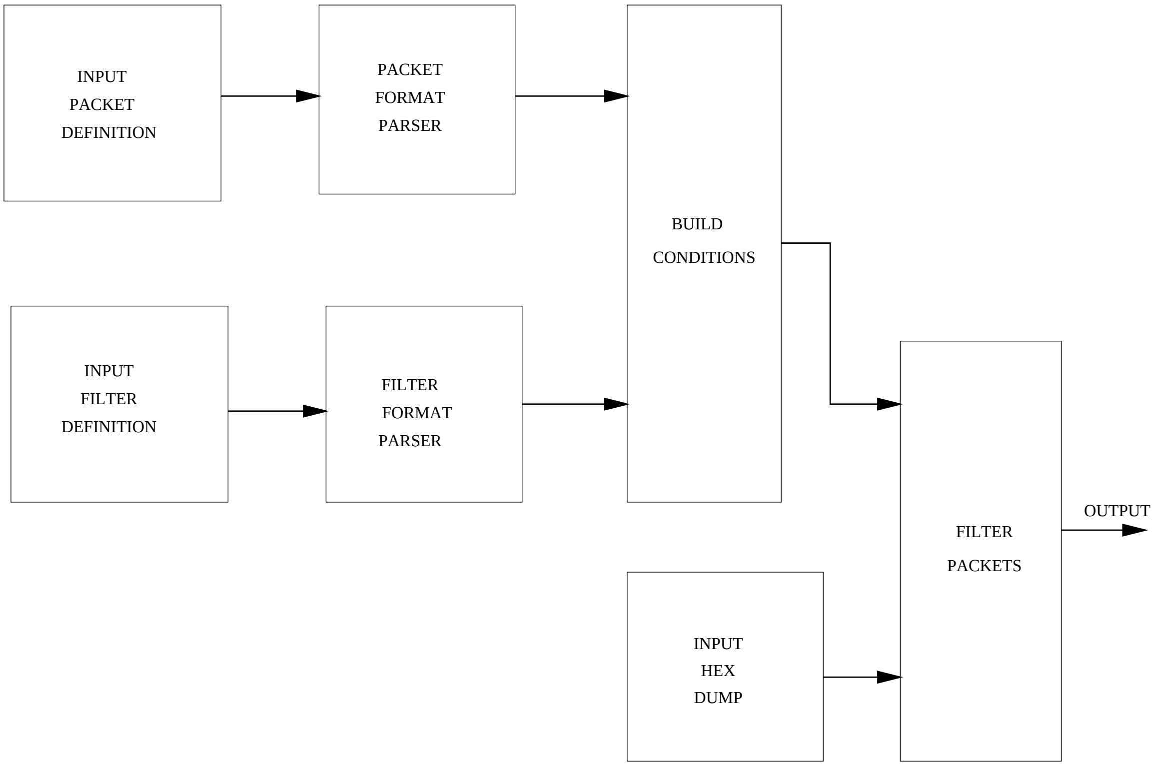

Figure 1: Block Diagram of Packet Analyzer

The user can query the current status and get a list of packet and filter definitions loaded. The packets can be filtered either singly or as an entire stream. Results can be viewed as filtering is in progress or at the end of the process, giving the user maximum flexibility.
The code is written in 'C' and the graphical user interface is built using *devguide*, the OpenWindows Software Developer's Guide.

3 Layered Architecture and Universal Packet Analyzer

Computer communication software is extremely complex. The need to make it easy to comprehend and manage led to the design of networks as a sequence of layers. Upper level protocols utilize the services offered by a subordinate layer, and by adding enhancements, offers a certain level of service to its higher layer. The lower layers shield the upper layers from the details of how the services are actually implemented.
The Packet Analyzer tool is built to filter packets in a layered architecture. The figure below is a conceptual model of how the Packet Analyzer works, and the various steps involved in the filtering of a packet.
As Figure 1 indicates, a packet definition is written in the Packet Definition Language. The packet definition is a BNF-like expression representing the packet and all its various combinations and

options. The PDL is discussed in greater detail in the next section. The PDL is designed to support the layering concept using two keywords:

- **The include statement** allows the inclusion of the lower layer files. It allows for the examination of the packet at any layer. All that is required is an include statement that points to the immediate lower layer.
- **The Decode Section** comes at the bottom of the packet definition and points to immediate upper layer(s).

The packet definition is a representation of the layered architecture - the include statement links the immediate lower layer while the decode section points to the next upper layer level protocol.
The packet definition is parsed using the *packet format parser* and the packets details are stored. The packet format parser written with the UNIX tools *lex* and *yacc*.
The filter definition file consists of a number of distinct *filter clauses*. These filter clauses allow the user the option of inspecting the different layers either individually or in the same output file. The filter definition is parsed using the *filter format parser* and the filter conditions are stored. The filter definition parser was written in *lex* and *yacc* also.
The parsed packet definition and filter definition are used to build stacks of conditions. Condition stacks consist of each clause in the filter and the packet layer to which it relates. The filter clauses may point to specific protocols at the different layers. The hex dump file is filtered by reading it, running it through the condition stacks, separating out the data that match the conditions and outputting the results to files indicated in the clauses.
The filtering process starts at the lowest layer and progresses upwards till it is done with all the layers in the chain of include statements and decode sections. The feature that stands out in this program is that it is completely independent of the number of layers and the kind of layered architecture.

4 Packet and Filter Definition Languages

This section contains a brief overview of the *Packet Definition Language (PDL)* and the *Filter Definition Language (FDL)* with illustrative examples. For a more detailed explanation of these scripting languages the reader is referred to [1] and [2].
The PDL is a simple scripting language that is used to define packets. The structure of the language is given in Figure 2.
In the packet definition, *field details* contain the *field type* which indicates the nature of the field as shown in Figure 3. These concepts will become clearer in the following packet definition example.
The PDL supports three data types:

- **Integer** - Any sequence of digits

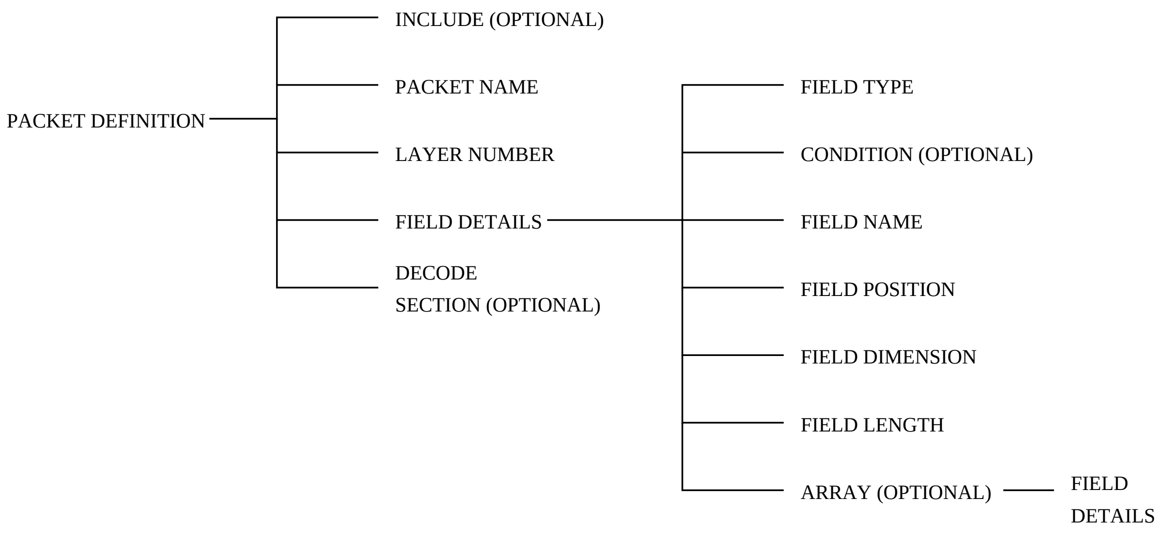

Figure 2: The Packet Definition Language Structure

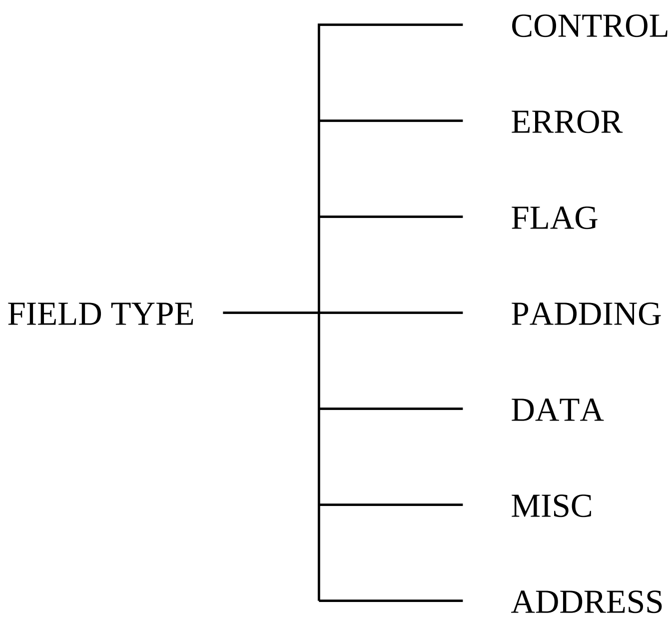

Figure 3: The Field Types

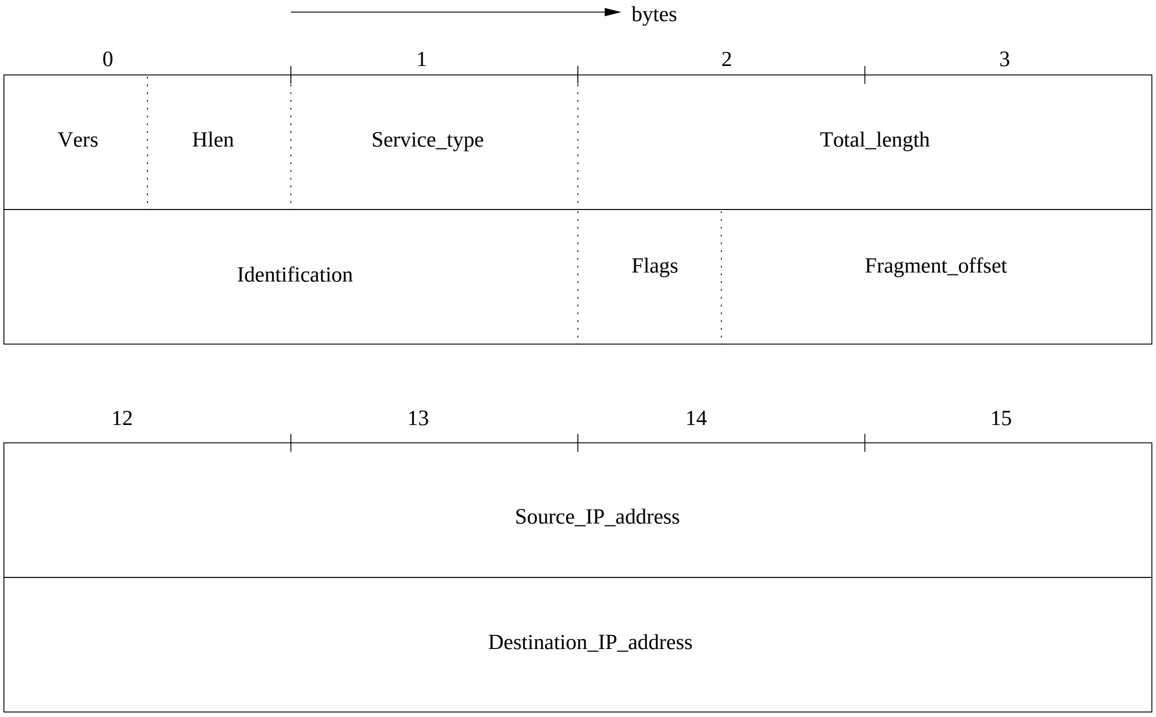

Figure 4: Part of the Internet Protocol (IP) Packet Structure

- **String** - Any sequence of alphanumeric characters enclosed within double quotes.
- **Hexadecimal Values** - Any sequence of digits or characters from A though F, preceded by **0x**.

Figure 4 shows part of the Internet Protocol (IP) packet. This is followed by the corresponding packet definition.
The packet definition for the IP packet in Figure 4 follows:

```
include : "ethernet.pkt"
include : "llc.pkt"

packet : "ip"
layer : 3
MISC
field_name : "Vers"
field_pos : 0
field_dimension : 0.4
field_length : 0.4
MISC
field_name : "Hlen"
field_pos : 0.4
```

```
field_dimension : 0.4
field_length : 0.4
MISC
field_name : "Service_type"
field_pos : 1
field_dimension : 1
field_length : 1
    ⋮
ADDRESS
field_name : "Source_IP_address"
field_pos : 12
field_dimension : 4
field_length : 4
ADDRESS
field_name : "Destination_IP_address"
field_pos : 16
field_dimension : 4
field_length : 4
```

PDL support for bit fields is illustrated by the Vers (Version) and Hlen (Header Length) fields, which are each 4 bits long. This is also useful for specifying flags that are normally one bit in size. Fields of varying lengths and fields that exist under special condition are supported by PDL.

Filter files are made up of *filter clauses*. Each clause contains the following constructs:

- *Physical Layer Name*
- *Filter Condition* - can use relational operators
- *Filter output file name* - which holds the results. This is optional.

An example will clarify these concepts. Consider the following filter file:

```
physical_layer : "ethernet"
(value("ethernet.Frametype") = 0x0800) || (value("llc.control") = 0x80) ||
(value("ip.source_ip_address")=0x80c28601)
file : "ip.out"

physical_layer : "ethernet"
(value("sunrpc.msg_type")=0x00) && (value("sunrpc.rpc_vers")=0x02)
file : "sun_rpc.out"
```

The filter file above has two clauses. The first clause uses the relational operator **or** to filter packets that have a frametype value of **2048** or the link layer control field with a value of **128**. The frametype value of 2048 indicates that the upper layer protocol is going to be IP. The third part of this clause is looking for a packets from a specific source **128.194.134.1** - which is the machine *photon.cs.tamu.edu*.
The second clause uses the relational operator **and** to filter packets that are Sun RPC calls. The message type of value '0' indicates a call, the user is specifically looking for RPC version '2'.

5 Using the Universal Packet Analyzer

The services offered by the Packet Analyzer can be split into two classes:

- **Filtering Services**
- **Supplementary Services**

The *filtering service* forms the main service component and is composed of the following steps:

- Load Packet Definition
- Load Filter Definition
- Load HEX packet
- Filter the Packet singly or as a stream
- View the results

The *supplementary services* support the main task of filtering packets and provide vital functions. The support services make the Packet Analyzer a complete tool. Some of the services provided are:

- Packet Definition Scripting Interface
- Packet Definition Editing Interface
- Filter Definition Scripting Interface
- Filter Definition Editing Interface
- Configuration Saving Interface
- Configuration Loading Interface
- Configuration Editing Interface

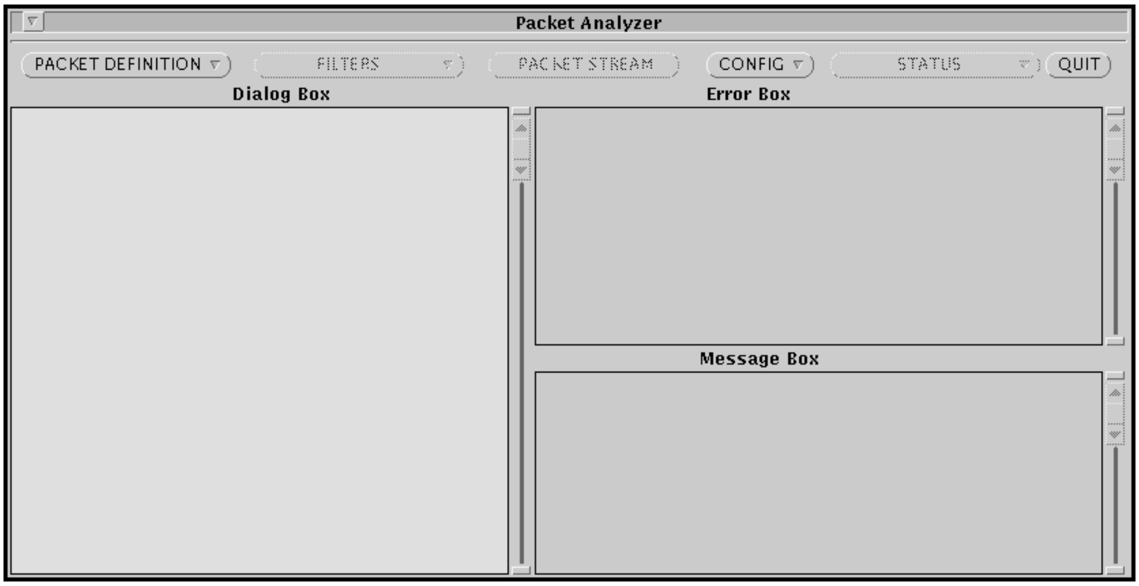

Figure 5: The Main Window

- Status Viewing Interface

A detailed description of how these two class of services are accomplished can be found in [2], Appendix A also contains a quick reference card to filtering services.

The Packet Analyzer is started by typing **Pkt_tool** at the prompt. The user is presented with the Main Window as shown in Figure 4. The Main Window contains three information windows and a set of buttons. Some of the buttons are active and some are inactive and guide the users initial choice.

Some of the buttons are in an inactive state and can be accessed only after one of the two active buttons has been used. The Main Window remains on the screen as long as the Packet Analyzer is running; however, it can be minimized to an icon, if required.

Each of the boxes in the Main Window is a read-only pane that give the user textual information - these are discussed below.

DIALOG BOX - This is the pane where the result of filtering is displayed when the user opts to view the results *online*.

ERROR BOX - As the name suggests, this is the box where the error messages appear.

MESSAGE BOX - General user information messages appear in this pane. A typical message informs a user that a packet definition has been loaded up with no errors.

A portion of a result file is listed below, the field names are followed by their values in hexadecimal.

```
Packet# : 1
"ethernet"
```

```
"dstaddr" : 00003C00299B
"srcaddr" : 080020718C58
"frametype" : 0800
"ip"
"source_ip_address" : 80C284DD
"destination_ip_address" : 80C28402
"vers" : 4
"hlen" : 5
"service_type" : 00
"total_length" : 05DC
"identification" : 563F
"fragment_offset" : 000
"time_to_live" : FF

    ⋮
```

The reader is refered to [2] for a detailed discussion of the user interface and the functionality supported by this program.

6 Conclusions

The Universal Packet Analyzer has a number of pre-defined protocols. The main advantage is that it allows the easy addition of new protocols. The Graphical User Interface adds to its usability, making it an effective tool for network administrators.

A few features are identified here that would make the Universal Packet Analyzer a more powerful program. If a tool for capturing packets is integrated into this program, then it could be used to filter packets dynamically. Result file interpretation would make the output of filtering accessible to a wider range of computer users.

References

[1] S. Prakashkumar, "Packet Analyzer Tool," Project Report, Department of Computer Science, Texas A&M University, College Station, TX, 77843, Spring 1994.

[2] S. Parameswar, "Universal Packet Analyzer - A Network Packet Filtering Tool," Project Report, Department of Computer Science, Texas A&M University, College Station, TX, 77843, Fall 1995.

[3] A. S. Tannenbaum, "Computer Networks," Prentice Hall of India, August 1993.

[4] Sun Microsystems Inc, "RPC: Remote Procedure Call Protocol specifications version 2," Request For Comment (RFC) 1057, Internet Publication, June 1988.

[5] Sun Microsystems Inc, "XDR: External Data Representation Standard," Request For Comment (RFC) 1014, Internet Publication, June 1987.

[6] R. Stevens, G. Wright, Illustrated TCP/IP volume 1 and 2, Reading, MA: Adison-Wesley, 1994.

[7] DigiNet Corporation, "Ethernet/IEEE 802.3 Protocol Reference Guide", Reference Card, Broomfield, Colorado.

[8] DigiNet Corporation, "TCP/IP Protocol Reference Guide", Reference Card, Broomfield, Colorado.