

University of Crete
Computer Science Department

*Gigabit monitoring API - An API for monitoring
gigabit networks*

Konstantinos Xinidis

September 2003
Heraklion , Greece

1. Table of Contents

1. Table of Contents.....	2
3. Introduction.....	3
2. Design.....	3
3. Supplied abstraction.....	3
4. MAPI functions.....	3
5. Gigabit Monitoring API.....	4
6. Low level.....	4
7. High level.....	4
8. Implementation.....	4
9. Gigabit monitoring API core.....	4
10. Gigabit monitoring API functions.....	5
11. Predefined functions.....	5
1. Examples.....	7
2. MAPI vs pcap.....	7
3. Future work.....	7

1. Introduction

The primary goal of Gigabit monitoring API (MAPI) is to make possible the monitoring of network traffic at gigabit links. In order to achieve this goal MAPI pushes much of the computation required by network monitoring applications down to Linux kernel. This is based on the observation that network-monitoring applications, with high probability, want only a portion of the packets (that's why BPF filtering is done inside Linux kernel) and only to do a simple operation on them. Such operations are counting the packets, searching the packets for a signature (this is the case for intrusion detection-prevention applications) etc. So, for example Linux kernel takes the burden of counting the packets and gives the results to the application when requested; also, Linux kernel searches all the packets for the signature and if a match is found is queued for later processing from the application. The gain from this "movement of responsibility" is much less memory copies and context switches. This reduction is critical and enables MAPI to be able to monitor much more traffic than other APIs such as pcap.

2. Design

a) Supplied abstraction

Each API should provide its users with a suitable abstraction which, on the one hand, is simple enough for humans to use and understand, and on the other hand, is powerful enough to express complex user requirements. Such an abstraction is the *network flow*. A network flow is defined to be a sequence of packets that satisfy a given condition. For example, the simplest flow one can think of, is the flow that is composed of all network packets. Another flow may be composed of all packets directed to a particular web server. Another more complex flow may be composed of all packets sent from a given source IP subnet to a given destination IP subnet that use the TCP/IP protocol suite and contain a particular signature. Note that this definition of flow differs from the traditional that usually refers to the traffic between two hosts, or even between two hosts using specific ports.

b) MAPI functions

The innovation behind MAPI is the ability to apply functions on packets. These functions can do anything from collecting statistics, filtering based on headers, searching for a particular signature, checking packet's headers for correctness to modifying the packet's contents or dropping it.

MAPI functions are organized into chunks. Every time a packet arrives the functions registered run in the order of their registration. The first one registered will be the first one to run. Because of the fact that a MAPI function can modify or drop a packet the order of registration does matter.

3. Gigabit Monitoring API

a) Low level

In this level the entity that represents the network flow is the socket descriptor. This socket is created with the usual way having as a protocol family the PF_MAPI protocol family. Functions are added and removed through ioctls. For every MAPI function there are at least one ioctl for adding this function to the list of functions to be applied to incoming packets and one for removing this function. In addition many MAPI functions have ioctls for providing the results, resetting statistics, setting asynchronous notification etc.

b) High level

The high level API is a direct extension of the low level one. The entity that represents the network flow here is a *mapi_flow_t* descriptor. The network flow is created by calling the function *mapi_create_flow* and destroyed by calling the function *mapi_destroy_flow*. There are functions for adding and removing functions (*mapi_apply_function*, *mapi_remove_function*), setting and getting various network flow options (*mapi_set_flow_option*, *mapi_get_flow_option*), reading results (*mapi_read_results*), reading packets (*mapi_get_next_packet*), setting callback functions for each packet read (*mapi_loop*) and some other not so interesting.

4. Implementation

a) Gigabit monitoring API core

The part of MAPI that resides inside Linux kernel is implemented as a new protocol family (PF_MAPI). In reality it complements packet protocol family (PF_PACKET). It has all the mechanisms for registering, unregistering and running MAPI functions. Also, supports the memory mapping of packets in user-space.

b) Gigabit monitoring API functions

Every MAPI function is implemented as a separate module, so it is possible to load a MAPI function only when this is necessary and unload it when it is not needed. Each MAPI function when is loaded registers itself with the MAPI core. The registration provides to the MAPI core some information. Some of them are which C function to run when the MAPI function is applied to a packet, which C function to run when this MAPI function is added or removed etc. Also, almost every MAPI function creates a file in the /proc filesystem.

5. Predefined functions

Until now there are 40 implemented MAPI functions (see Table 1).

<i>Gigabit monitoring API function</i>	<i>Short description</i>
COUNT_PACKETS	Counts total packets received until know.
COUNT_BYTES	Counts total bytes received until know.
SAMPLE_PACKETS	Samples the stream of packets. Application can select between probabilistic or deterministic sampling and the period of sampling.
SUBSTRING_SEARCH	Boyer Moore string search algorithm. Selects only the packets with the given payload.
HASH	Rotating and additive hash. Selects or ignores packets depending on the return value of the hash function.
SUBFLOW	Collects information about sub-flows. A sub-flow is defined to contain all packets (within the parent flow) that have a common 5 or 7-tuple of protocol number, source IP address, source port, destination IP address, destination port , input interface and output interface.
LOGGING	Logs all packets in a specified file (currently only tcpdump format is supported).
EXB	Exb string search algorithm. Selects only the packets with the given payload.
PACKETS_IN_INTERVAL	Counts all packets received in a specified interval (msec).
BYTES_IN_INTERVAL	Counts all bytes received in a specified interval (msec).

<i>Gigabit monitoring API function</i>	<i>Short description</i>
PACKET_DISTRIBUTION	This is a rather elaborate function that can be used to find the distribution of the length of the packets, the ports of the packets etc. The function takes as arguments an offset and a mask. For each incoming packet the function takes a sequential number of bits starting at byte offset. This number of bits which is the value of mask should not be larger than 16. The resulting 16-bit number is then used as an index to an array to increase the respected position.
PACKET_SAVE	Trims packets according to the specified parameters.
COOK_IP	Reassembles IP traffic, ignores packet if IP checksum is not valid etc.
COOK_UDP	Removes UDP headers after the packet has already successfully passed from IP reassembly function.
PKT_TYPE	Selects packets of the given packet type (PACKET_HOST, PACKET_OUTGOING ...)
METER	Computes the rate of arriving packets in packets/sec.
BAND_METER	Computes the rate of arriving traffic in bytes/sec.
CHECK_IP_HDR	Checks the correctness of the IP header of the packet.
CHECK_UDP_HDR	Checks the correctness of the UDP header of the packet.
CHECK_TCP_HDR	Checks the correctness of the TCP header of the packet.
PRINT_ETHER	Prints the Ethernet header of the packet (Used for debugging).
PRINT_IP	Prints the IP header of the packet (Used for debugging).
PRINT_UDP	Prints the UDP header of the packet (Used for debugging).
PRINT_TCP	Prints the TCP header of the packet (Used for debugging).
NETDEV_STATS	Selects packets according to the statistics collected so far from the network interface from which packet was captured (Used for debugging).
SET_PERF_COUNTER	
ACCUM_PERF_COUNTER	
SET_CYCLE_COUNTER	
ACCUM_CYCLE_COUNTER	
BPF_FILTER	
CACHED_BPF_FILTER	
FLOW_REPORT	
FLOW_KEY	
FLOW_RAW	
DECIDE	This is a function that transforms the list of MAPI functions to a tree of MAPI functions. To accomplish this needs the help of some helper functions listed below.

<i>Gigabit monitoring API function</i>	<i>Short description</i>
DECIDE_BPF_HOOK	
DECIDE_ACTION_HOOK	
DECIDE_TEE_HOOK	

Table 1 Implemented gigabit monitoring API functions.

Two of them deserve special attention. The first one is a MAPI function called DECIDE. This is a function that transforms the list of MAPI functions to a tree of MAPI functions.

6.Examples

7.MAPI vs pcap

8.Future work