Traffic Shaping - Application Layer Packet

Pratibha Tambewagh[1] Shankar M Patil[2]
[1]M.E. Student [2]Associate Professor
[1,2]Department of Computer Engineering
[1,2]BVCOE, Kharghar, Navi Mumbai

Abstract— The use of Application layer packet classifier and optimization of bandwidth towards QoS in Linux using netfilter, iproute2 and layer-7 Filter. As seen in the statistics the huge amount of data flows through the network, so it is necessity to apply packet-filtering rules in order to control the traffic and add firewall rules. So linux-based firewalls being highly customizable and versatile are also robust, inexpensive, and reliable. Using iproute2 package with ip and tc we can control traffic. Using Layer-7 Filter we can classify packet at using regular expressions not on the regardless of the tcp port but in the ip layer data. This gives us more flexibility to catch unknown traffic, where the tunneling protocols are used to transfer the data. Thus we can save bandwidth and drop or reject unwanted traffic.

Key words: Application layer packet, HTTP, FTP

I. Introduction

Internet has tremendous growth, Application user write programs based on their requirements; this may be with standard port or nonstandard ports. The identification of packets based on application layer data. It can classify packets as Kazaa, HTTP, Jabber, Citrix, BitTorrent, FTP, Gnucleus, eDonkey2000 etc. regardless of port. It complements existing classifiers that match on IP address, port numbers and so on. Our intent for application layer packet classification to be used for packet filtering, QoS and bandwidth arbitration with non-standard and standard ports.

There are a number of network services that require packet classification, such as routing, access-control in firewalls, policy-based routing, provision of differentiated qualities of service, and traffic billing. In each case, it is necessary to determine which flow an arriving packet belongs to so as to determine — for example — whether to forward or filter it, where to forward it to, what class of service it should receive, or how much should be charged for transporting it. The categorization function is performed by a flow classifier (also called a packet classifier) which maintains a set of rules, where each flow obeys at least one rule. The rules classify which flow a packet belongs to base on the contents of the packet header(s). For example, a flow could be defined by particular values of source and destination IP addresses, and by particular transport port numbers. Or a flow could be simply defined by a destination prefix and a range of port values. As we shall see, a number of different types of rules are used in practice. This paper describes a method for fast packet classification based on an almost arbitrary set of rules. We focus here only on the problem of identifying the class to which a packet belongs.

The simplest and most well-known form of packet classification is used in routing IP datagrams, where each rule specifies a destination prefix. The associated action is the IP address of the next-hop where the packet needs to be routed to. The classification process requires determining the longest prefix which matches the destination address of the packet.

A. Packet Classification:

Packet classification is an integral part of today’s networks. On the path from source to destination, a packet is typically classified several times based on multiple header fields and/or its content. Traditionally, firewalls classify the packet to decide whether to drop it or not, routers may classify it to determine its next hop and its QoS class, and load balancers use classification to decide appropriate destination servers. In software defined networks (SDNs), switches classify packets to forward them based on the installed flow entries, and it is common for large enterprise networks to employ complex application-aware access control mechanisms to classify ingress/egress traffic. Finally, modern datacenters use load balancers to classify tens of thousands of flows.

Packet classification is performed using a packet classifier, also called a policy database, flow classifier, or simply a classifier. A classifier is a collection of rules or policies. Each rule specifies a class† that a packet may belong to base on some criterion on F fields of the packet header, and associates with each class an identifier, classID. This identifier uniquely specifies the action associated with the rule. Each rule has F components. The ith component of rule R, referred to as R[i], is a regular expression on the ith field of the packet header (in practice, the regular expression is limited by syntax to a simple address/mask or operator/number(s) specification). A packet P is said to match a particular rule R, if "i , the ith field of the header of P satisfies the regular expression R[i]. It is convenient to think of a rule R as the set of all packet headers which could match R. When viewed in this way, two distinct rules are said to be either partially overlapping or non-overlapping, or that one is a subset of the other, with corresponding set-related definitions. When two rules are not mutually exclusive, the order in which they appear in the classifier will determine their relative priority. For example, in a packet classifier that performs longest-prefix address lookups, each destination prefix is a rule, the corresponding next hop is its action, the pointer to the next-hop is the associated classID, and the classifier is the whole forwarding table. If we assume that the forwarding table has longer prefixes appearing before shorter ones, thelookup is an example of the packet classification problem.

B. Stateless Packet Filters:

A border router configured to pass or reject packets based on information in the header of each individual packet. Packet filters can theoretically be configured to pass/reject based on any field but usually done based on:

- Protocol Type
- IP address
- TCP/UDP port
- Fragment number
- Source routing information

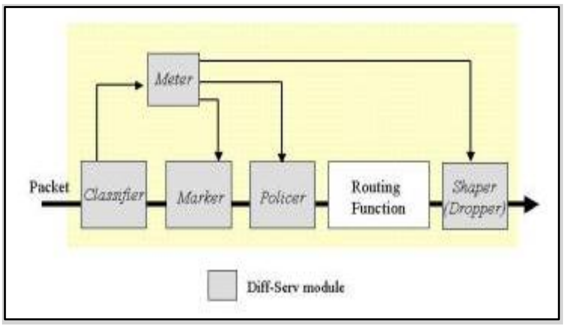

Fig. 1: Forwarding Path

C. *Protocol Filtering:*

Filtering based on the IP protocol field allows rejecting of entire protocol suites

- UDP
- TCP
- ICMP
- IGMP

The process of categorizing packets into "flows" in an Internet router is called packet classification. All packets belonging to the same flow obey a pre-defined rule and are processed in a similar manner by the router. For example, all packets with the same source and destination IP addresses may be defined to form a flow. Packet classification is needed for non "best-effort" services, such as firewalls and quality of service; services that require the capability to distinguish and isolate traffic in different flows for suitable processing. In general, packet classification on multiple fields is a difficult problem. Hence, researchers have proposed a variety of algorithms which, broadly speaking; can be categorized as "basic search algorithms," geometric algorithms, Heuristic algorithms or hardware-specific search algorithms.

Fig 1 shows some of the header fields (and their widths) that might be used for classifying the packet. Although not shown in this figure, higher layer (e.g., application-level) headers may also be used.

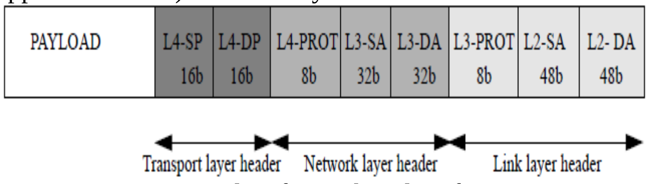

Fig. 2: Headers for packet classification

- L2 = Layer 2 (e.g., Ethernet)
- L3 = Layer 3(e.g., IP)
- L4 = Layer 4(e.g., TCP)
- DA =Destination Address
- SA = Source Address
- PROT = Protocol
- SP = Source Port
- DP =Destination Port

Until recently, Internet routers provided only "best-effort" service, servicing packets in a first-come-first-served manner. Routers are now called upon to provide different qualities of service to different applications which means routers need new mechanisms such as admission control, resource reservation, per-flow queuing, and fair scheduling. All of these mechanisms require the router to distinguish packets belonging to different flows. Flows are specified by rules applied to incoming packets. We call a collection of rules a classifier. Each rule specifies a flow that a packet may belong to based on some criteria applied to the packet header, as shown in Figure 1. To illustrate the variety of classifiers, consider some examples of how packet classification can be used by an ISP to provide different services.

The paper is organized in four sections followed by references. Section 2 describes the recent study done in application layer packet classification system. Section 3 describes the methodology used to implement packet classification. Section 4 summarizes the paper.

II. Recent work

A. *Scalable Packet Classification for Enabling Internet Differentiated Services (IEEE):*

Nowadays, IP networks are rapidly evolving toward QoS-enabled infrastructure. The need for packet classifications increasing in accordance with emerging differentiated services. While the new differentiated services could significantly increase the number of rules, it has been demonstrated that performing packet classification on a potentially large number of rules is difficult and has poor worst-case performance. In this work, we present an enhanced tuple pruning search algorithm called "Tuple PruningPlus" (TPP) for packet classification, which outperforms the existing schemes on the scalability. Our main idea is to simplify the lookup procedure and to avoid unnecessary tuple probing by maintaining the least-cost property of rule through pre computation and the proposed Information Marker. With extra rules added for Information Marker, only onetuple access is required in each packet Classification. In our experiments, 70 MB DRAM is used to achieve50 million packets per second (MPPS) for a 1 M-rule set, showing a performance improvement by a factor of 50. We also present heuristic to further reduce the required storage to about 20 MB. These results demonstrate the effectiveness of the TPP scheme to achieve high speed packet classification. The disadvantage of dividing the packets into tuples lead to lots of traffic, packets are more segregated into smaller bits reducing the speed.

B. *Performance Improvement over Linux Layer-7 Content Filtering(IEEE):*

Due to security reasons, many companies need firewalls to filter some mistrusted applications, like FTP or P2P software. However, some applications may hide themselves with some well-known application ports like HTTP port 80 such that some firewalls cannot distinguish mistrusted applications from well-known applications. As a result, firewalls require high performance classification engines that can efficiently inspect layer-7 contents to recognize mistrusted applications. This paper analyzes the layer-7 classification module in Linux Net filter, theL7filter package, and proposes an alternative implementation to improve the performance of L7filter.The throughput of the proposed method can remain high even in heavily-loaded network environments. The performance of the proposed method is justified by the Spirent Smart Bits 6000 testing equipment whose traffic generation speed can achieve gigabit wire-speed. Tedious procedure to work on the layer 7 for Linux increases the complexity.

III. METHODOLOGY

The OSI and TCP/IP networking models. As we saw, even if the TCP/IP model has the widest usage, the reference model is OSI.

The TCP/IP and OSI models are shown in Figure 3.

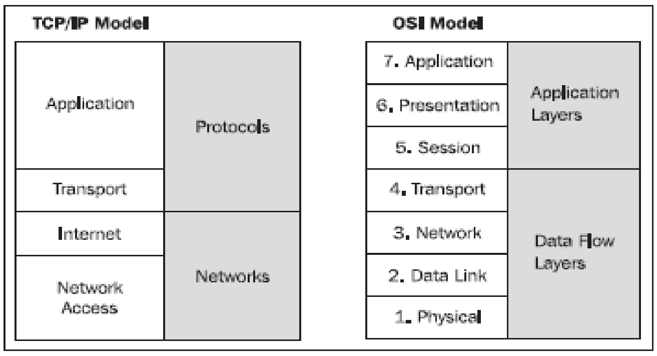

Fig. 3: TCP/IP and OSI Model Layers

At Layer 7 of the OSI model, we find Application (HTTP, FTP, SSH, etc.). As you can see from the picture above, TCP/IP compacted OSI Layers 7, 6, and 5 into one Layer, TCP/IP Layer 4 (Application), which has the same name, but different functionality.

Filtering and prioritizing traffic from some applications can be very easy and very hard at the same time. Normally, we would filter/prioritize web traffic by matching TCP packets with source or destination port 80, which is the standard HTTP port. However, web servers can be configured to use any port; so our filters/prioritizations won't work for that particular traffic.

Another big problem network administrators have is filtering traffic belonging to P2P (peer to peer) applications like Kazaa, DC++, Emule, etc., as those applications don't use standard ports and, even worse, they can be configured to use other applications' standard ports for communication (e.g. TCP port 80).

At one point, some people decided to do something about it and they did a pretty good job by starting the project named "Layer 7 Filtering" at http://l7-filter.sourceforge.net.

As you probably guessed, "Layer 7 Filtering" is a method to filter Layer 7 data. That means filtering network traffic generated by an application regardless of the protocol or port it uses at Layer 4.

L7-filter is a packet classifier for the Linux kernel that doesn't look up port numbers or Layer 4 protocols, but instead looks up the data in an IP packet and does a regular expression match on it to determine what kind of data it is, mainly what application protocol is being used.

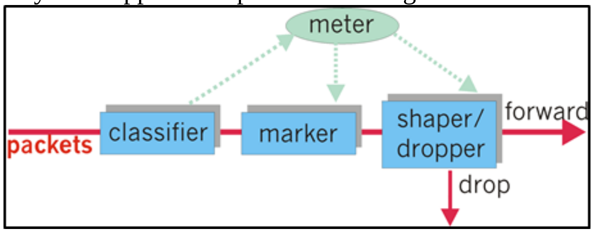

Fig. 4: Packets classifier

A. *When to use L7-Filter:*

1) *Features:*

– Patches for Linux 2.4 and 2.6
– Support for TCP, UDP and ICMP over IPv4
– Uses Netfilter's connection tracking of FTP, IRC, etc
– Examines data across multiple packets
– Number of packets examined tunable on the fly through /proc
– Number of bytes examined tunable at module load time
– Distinguishes between new connections (those still being tested) and old unidentified connections
– Gives access to both Netfilter and QoS (rate limiting) features
– With the Netfilter "helper" match, you can distinguish between parent and child connections (e.g. ftp command/data)

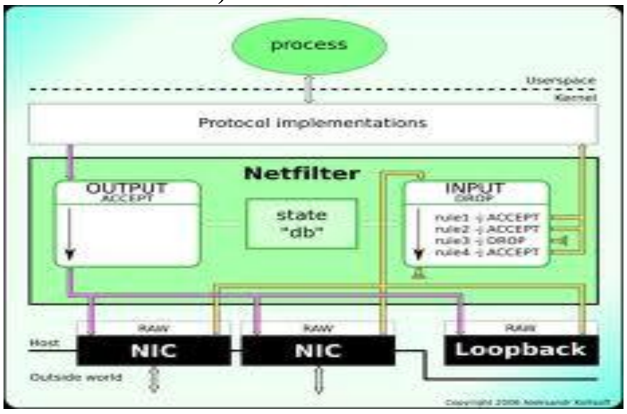

Fig. 5: L7 Netfilter

L7-filter is a great solution for matching application data in a network, but, as with almost every good thing, there are downsides to it too. As we will go deeper into how L7-filter works, you will see that it has to actually analyse data contained in IP packets, so it is quite obvious that this can eat up a lot of CPU power. So, using L7-filter on a Linux router with high traffic is not quite recommended, as L7-filter is CPU-consuming and can thus introduce latency and packet loss in the network. However, it really depends on the type of traffic passing through that router rather than the amount of traffic. For example, if you have 20 Mbps average of WWW data, L7-filter can work pretty well, but for 2 Mbps of VoIP traffic, the router's performance would dramatically drop.

Normally, we would consider using L7-filter for SOHO environments. In this case, L7-filter is very good for filtering viruses, limiting the bandwidth consumed by children when downloading music, etc.

Another situation when L7-filter is recommended would be for small-to-medium-sized companies that want to shape and prioritize traffic consumed by applications that are not vital or productive to the company. For example, in a company that has a 10 Mbps internet connection, an employee using P2P software can fill up the bandwidth and other traffic like Web, SSH, database replication, and so on would suffer because of that.

B. *How Does L7-Filter Work?*

What L7-filter does is provides a way for iptables to match packets based on the application they belong to.

The TCP/IP model contains four layers and, before the L7-filter project, netfilter could match data by the first three layers:

– Network access layer: iptables -A CHAIN -m mac --mac-source …"
– Internet: iptables -A CHAIN -s IP_ADDRESS …"

– Transport: iptables -A CHAIN -p tcp--dport 80 …

At the network access layer, netfilter uses -m mac to match packets from or to a MAC address in the network. At the layer above, the Internet layer, we have the IP protocol; netfilter matches packets from or to an IP address, regardless of the transport protocol, port number, or application the packet uses. At the transport layer, we have TCP or UDP, and netfilter can match packets by protocol, and more specifically, by port number within the protocol.

Any combination of the three lower layers is permitted; for instance:

iptables –A FORWARD –s 192.168.0.2 –p tcp -–dport 80 –m mac –-mac-source 00:01:BC:2D:EF:2A –j DROP"

Will drop all packets from the IP address 192.168.0.2 if the source MAC address is 00:01:BC:2D:EF:2A and the packets use the TCP protocol and have the destination TCP port 80.

L7-filter adds a new feature to netfilter by matching packets that belong to an application that is found at the TCP/IP Layer 4. A very important thing is that L7-filter is just another match option for iptables, and so all the rules of the other match options apply in this case. Therefore, you can do all the iptables operations with the packets matched by L7-filter.

The short and easy version of L7-filter is that after installing it you have -m layer7 --l7proto [http|ftp|...]. This is the match option we were talking about.

In order to match Layer 7 data, netfilter looks deeper into an IP packet than just at its header. However, the actual data contained in the packet doesn't just say "I'm a P2P packet; filter me!"; so the data is matched against a set of regular expressions that are common to different applications. This set of regular expressions is probably the most important part of this project, and is called "protocol definitions".

The L7-filter project contains three important parts:

1) A kernel patch, which provides a way for the kernel to look into the IP packets
2) An iptables patch, which provides the match option for iptables
3) A collection of pattern files that contain the regular expressions for supported protocol(protocol definitions).

C. How to Build Packet Classifier?

The two things needed to build firewalls and Quality of Service (QoS) with Linux are two packages named netfilter and iproute. While netfilter is a packet filtering framework included in the Linux kernels 2.4 and 2.6, iproute is a package containing a few utilities that allow Linux users to do advanced routing and traffic shaping.

1) netfilter/iptables:

Netfilter is a very important part of the Linux kernel in terms of security, packet mangling, and manipulation. The front end for netfilter is iptables, which "tells" the kernel what the user wants to do with the IP packets arriving into, passing through, or leaving the Linux box.

The most used features of netfilter are packet filtering and network address translation, but there are a lot of other things that we can do with netfilter, such as packet mangling Layer 7 filtering.

A rough explanation on how netfilter works is like this:

1) The user instructs the kernel about what it needs to do with the IP packets that flow through the Linux box using the iptables tool.
2) The Linux box then analyzes the IP headers on all packets flowing through it.
3) If, when looking at the IP headers, the kernel finds matching rules, then the packet is manipulated according to the matching rule.

IV. Conclusion

It might look very simple at the beginning, but actually is a lot more complicated process. Net filter has a few tables, each containing a default set of rules, which are called chains. The default table loaded into the kernel is the filter table, which contains three chains, that contains rules for packets destined to the Linux machine itself, for packets that the Linux machine routes to another IP address, rules for packets generated by the Linux machine. The advantage is the low cost and works on nonstandard protocol.

References

[1] Lucian Gheorghe “Designing and Implementing Linux Firewalls and QoS using netfilter, iproute2, NAT, and LFilter” Packt Publishing, October 2006
[2] Application Layer Packet Classifier for Linux website “http://l7filter.sourceforge.net/”
[3] Netfilter, firewalling, nat and packet mangling for linux website http://www.netfilter.org
[4] Nigel Kukard “Bandwidth Management and Optimization” International Network INASP, Opensource Bandwidth Solutions March 2006
[5] Lukas Kencl, Christian Schwarzer, “Traffic Adaptive Packet Filtering of Denial of Service Attacks” Intel Research laboratories, World of Wireless, Mobile and Multimedia Networks,2006. WoWMoM 2006.
[6] J. McCann and Satish Chandra, “Packet Types: Abstract Specification of Network Protocol Messages” Bell Laboratories, ACM SIGCOMM Computer Communication Review Volume 30, Issue 4 October 2000
[7] Jeffrey C. Mogul, “The Packet Filter An Efficient Mechanism for User level Network Code ”Digital Equipment Corporation Western Research Laboratory, ACM Operating Systems Review, SIGOPS
[8] HolgerDreger, AnjaFeldmann, Michael Mai, Vern Paxson, Robin Sommer “Dynamic Application Layer Protocol Analysis for Network Intrusion Detection” USENIX Security
[9] Pankaj Gupta ,Nick McKeown “Packet Classification on Multiple Fields”, Proc. Sigcomm, Computer Communication Review, vol. 29, no. 4, pp 14760,September 1999, Harvard University.
[10] Florin Baboescu George Varghese “Scalable Packet Classification”, University of California, San Diego Proceedings of ACM Sigcomm, pages 199210, August, 2001