Published in IET Networks
Received on 20th December 2011
Revised on 19th February 2012
doi: 10.1049/iet-net.2011.0048

ISSN 2047-4954

Design and evaluation of deep packet inspection system: a case study

M.-Y. Liao[1] *M.-Y. Luo*[2] *C.-S. Yang*[1] *C.-H. Chen*[3] *P.-C. Wu*[3] *Y.-W. Chen*[1]

[1]*Institute of Computer and Communication Engineering, Department of Electrical Engineering, National Cheng Kung University, Taiwan*
[2]*Department of Computer Science and Information Engineering, National Kaohsiung University of Applied Sciences, Taiwan*
[3]*Department of Computer Science and Engineering, National Sun Yat-sen University, Taiwan*
E-mail: myluo@cc.kuas.edu.tw

Abstract: An *increasing number of* Internet applications and services render network management more troublesome for bandwidth misuse and security concern. As a result, network traffic identification plays an increasingly important role in network management. Deep packet inspection (DPI) is one of the effective approaches. Conventional network devices lookup the header of a packet, but DPI means the network device is required to match a pattern in the payload of a packet. This study proposes a DPI system and WMT (Wu-Manber with trie) algorithm to classify popular network services; The Net-DPIS is developed based on Netfilter framework in Linux kernel. The authors show how to rearrange the rule policies to increase the performance of Net-DPIS. In the results, the authors show that WMT algorithm is faster than WM algorithm; Net-DPIS has higher average accuracy and performance than L7-filter.

1 Introduction

More and more Internet applications bind random port number to transmit message and data; the port number is not exactly mapping to a network service and to identify traffic with protocol type and port number is not enough [1]. As a result, there are many challenges in traffic classification. A running instance of a network application may generate a series of packets. A packet is composed with header and payload, and the header carries layering information, such as the IP address is in the layer three and the port number is in the layer four. The payload carries upper layer application protocol and data. Network services transmit data over the application protocol. HTTP uses GET or PUT as method and MSN uses NLN or BSY as user status. These strings are generally unique so that they could be thought as the patterns or signatures of network applications. DPI mechanism plays an important role in traffic characterisation and fine-grained network monitoring by searching the payload of network packets for known patterns or signatures [2].

In this paper, Net-DIPS (Netfilter [3]-based deep packet inspection system) is proposed and developed based on Linux Netfilter framework. The first part of a DPI system is to capture packets from network interface cards, and manipulate them for further detecting certain network patterns while trying to minimise packet processing latency. We extend the Netfilter framework of Linux kernel to implement an efficient mechanism to achieve this. As a result, the result of this paper can be applied to all network system equipped Linux kernel. The second part of a DPI system is to search known signature patterns in the payload of packets. Variable pattern length and location, and constantly added rules make pattern matching a difficult task. Net-DPIS improves the WM multi-pattern matching algorithm [4] with a novel design of the rule table. Rajkumar [5] indicates that feature analysis of network service is able to avoid unnecessary matching; this study also improves the rule table of Net-DPIS.

The remainder of this paper is organised as follows: Section 2 describes the related work. We describe the Netfilter framework and our extension in Section 3. We detail how to construct the rule table to increase the speed of Net-DPIS effectively in Section 4. Section 5 measures the performance of WMT algorithm and compares Net-DPIS with L7-filter [6], IPP2P [7] and Sen-Spatscheck-Wang (SSW) [8]. Finally, Section 6 concludes the paper.

2 Related works

Increasingly, network traffics are classified not only by the fields of their packet headers (e.g. port number defined by Internet Assigned Numbers Authority), but also by the content of their payloads. As a result of these trends, there has been a considerable amount of recent work on implementing signature-based DPI in software system [6, 9, 10] or networking devices [11]. Therefore some approaches [12–15] have been proposed to identify the network traffic, and DPI is one of the effective approaches.

2.1 Deep packet inspection

DPI technique inspects both the header and the payload of a packet. String matching algorithms such as KMP (Knuth–Morris–Pratt) [16], AC (Aho–Corasick) [17], BM (Boyer–Moore) [18], CW (Commentz–Walter) [19] and WM (Wu–Manber) [4] algorithms have been the foundation for many DPI systems over many years. Owing to the variable position of a signature in packet payload, the string matching algorithm directly affects the matching performance of an identification system. The authors of [8, 20, 21] suggest that the signature in packet payload can be used for generating service rules. This paper proposed a design of rule table can improve the performance of traffic identification.

Many DPI systems use regular expressions to depict pattern signatures as more general cases. There have been numerous studies on regular expression matching [22–24]. L7-filter is a packet identification system whose regular expression is based on Henry Spencer's Bell Version 8 Regular Expression. This version of the regular expression has the following limitations; Bound, Character Class and Back Reference are not used. Currently, L7-filter supports to identify 120 types of services, including 23 types of P2P applications. Different platforms have been used to implement DPI system, including ASICs [25], field programmable gate arrays [26], network processors [27] and even cloud platform [28].

As shown in Fig. 1, the depth (N) means that the first N bytes of the payload are required to be searched by DPI system; Hsu [29] indicates that Net-DPIS finds 99% of identifiable network flows in the first 600 bytes of a payload, DPI system could set a suitable depth (N) according to the tradeoff of accuracy and cost time.

2.2 Netfilter framework

The Netfilter framework is located in the Linux kernel IP layer; it provides a set of hooks to intercept and manipulate the packets. Netfilter framework provides the packet processing function such as packet filtering, packet forwarding, connection tracking, network address translation (NAT), packet mangling for packet modification etc. The Netfilter framework for kernel version 2.6 implements five hooks to intercept and manipulate packets as illustrated in Fig. 2. If the packets are forwarded to the next hop, they go through the path of PREROUTING, FORWARD and POSTROUTING chains. The packets are received to local network service via the PREROUTING and INPUT chains. Outgoing packets are sent out via OUTPUT and POSTROUTING chains. Netfilter firewall is registered at INPUT chain for end-host servers.

Netfilter framework provides the *iptables* utilities for users to configure the Netfilter framework, for example, firewall rules configuration. The *iptables* utilities in user space communicate with the Netfilter framework in the kernel space via the Netlink socket. Netlink socket is socket-like system calls for accessing the kernel space. Unlike other system calls, Netlink socket has the benefits that support asynchronous operations, duplex characteristics, multicasting and short response time for user-space applications etc. Netfilter firewall manages the firewall rules using the linked-list data structure. So every packet must check all firewall rules until it finds the rule-matching result. As a consequence, the number of rules and incoming packets determine firewall's computation complexity. With the growth of rules and incoming packets, CPUs would spend considerable time on the Netfilter firewall; this situation would influence the overall performance of network application.

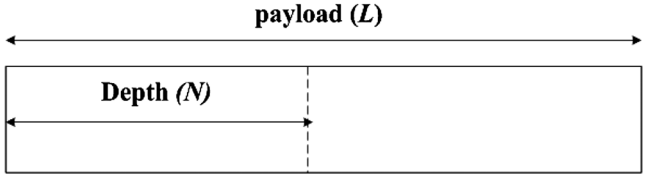

Fig. 1 *Inspection depth length*

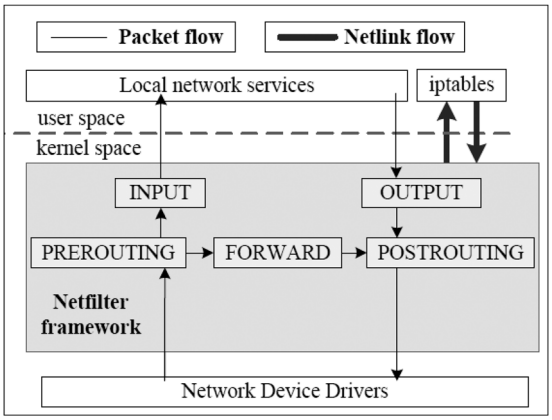

Fig. 2 *Netfilter framework*

3 Design and implementation of Net-DPIS

Net-DPIS is a network traffic classification system; Fig. 3 is the network topology. The core switch clones the network traffic between local area network (LAN) and Internet to a mirror port. The advantage of using port mirror is that we do not need to change the network topology and the Net-DPIS will not be the bottleneck while network traffic is large. What we have to do is to enable the port mirror feature of the core switch and to connect Net-DPIS with the mirror port of the core switch. If we use online mode, we have to make the network traffic go through Net-DPIS; the processing delay of Net-DPIS may cause packet lose. We show the details of design and implementation in this section.

3.1 System architecture

A core switch enables the port mirror function to clone all packets in LAN and all of the cloned packets are sent to a Net-DPIS. The system components are shown in Fig. 4. The Net-DPIS sets the network interface card (NIC) in promiscuous mode and captures all the cloned packets from the core switch. Net-DPIS is developed based on Netfilter

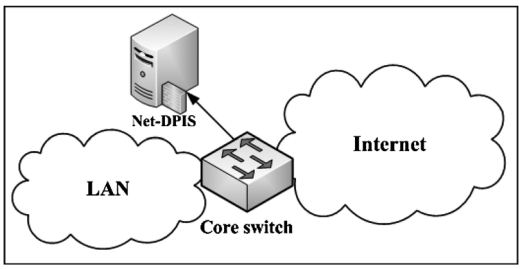

Fig. 3 *Network topology (port mirror)*

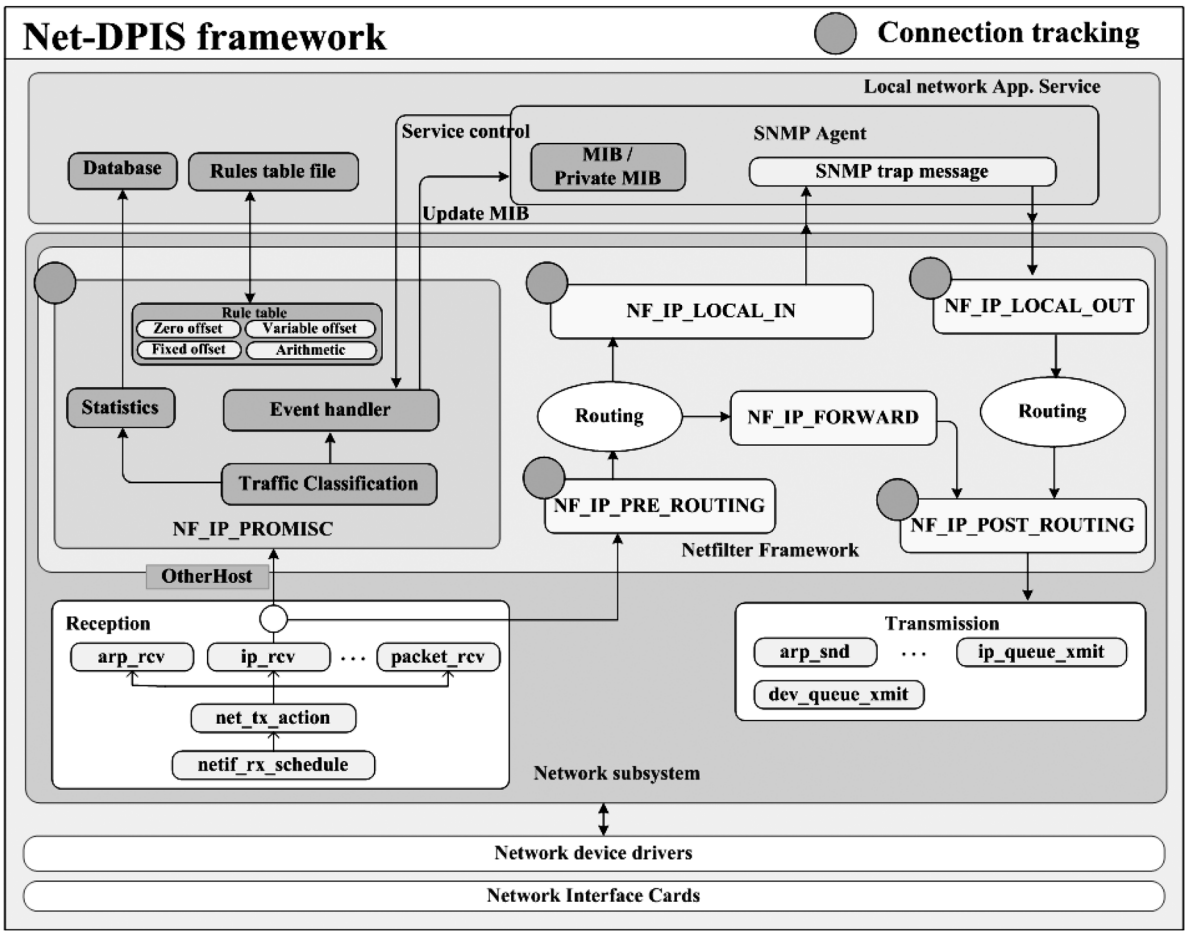

Fig. 4 *System architecture*

[3] framework in Linux kernel. A PROMISC hook is added to Netfilter framework; the PROMISC hook cloned packets from NIC. Connection tracking module is a native module on Netfilter framework and maintains the connection information. Cloned packets are classified in traffic classification module and the classified results are recorded in database. Net-DPIS uses simple network management protocol (SNMP) trap message to communicate with network management system (NMS) [30, 31]. While event handler sends the results of traffic classification with updating the private management information base (MIB), Net-DPIS sends connection information and service types to NMS over the SNMP trap message.

3.2 PROMISC hook

In Netfilter framework, there are five native hooks which are called NF_IP_PRE_ROUTING, NF_IP_LOCAL_IN, NF_IP_FORWARD, NF_IP_LOCAL_OUT and NF_IP_POST_ROUTING as shown in Fig. 3. While a NIC runs in promiscuous mode, the NIC captures all the incoming packets. If the destination media access control (MAC) address of a packet is not the same with the NIC of Net-DPIS, the packet is dropped; if the destination MAC address of a packet is the same with the NIC of Net-DPIS, the packet is allowed to enter Netfilter framework [32, 33]. In order to allow all packets from NIC to enter Netfilter framework, we modified receive module and added a PROMISC hook on Netfilter framework to receive the packets that should be dropped. Shishlov [34] proposed a patch of PROMISC hook for Linux kernel 2.4; in this study, we proposed a patch of PROMISC hook for Linux kernel 2.6.31 [35].

3.3 Traffic classification

Traffic classification is the core module of Net-DPIS for identifying the network traffic; the purpose of traffic classification is to match signatures in payload. Notations used in this paper are explained in Table 1.

Formula (1) is used to define how to classify the signatures into the four types of matching policies:

$$t = \begin{cases} \alpha & \text{if } |p| = 1 \text{ and } x = 0 \\ \beta & \text{if } |p| = 1 \text{ and } 0 \leqq x \leqq (L - S) \\ \gamma & \text{if } |p| > 1 \\ \lambda & \text{if } |p| = 1 \text{ and } c = L \end{cases} \quad (1)$$

Net-DPIS uses four matching policies to increase the performance of pattern matching; the following statements describe four matching policies: As shown in Fig. 5, zero offset policy and fixed offset policy match signatures at a static offset x, zero offset policy is a case of fixed offset policy whereas the offset x is at 0. 'PASV' is one of zero offset signatures; 'BitTorrent protocol' is one of fixed offset signatures and starts at offset 1. Variable offset policy matches signatures at dynamic offset x; 'SMTP' signature belongs to variable offset type. Arithmetic policy computes the message header; an example of arithmetic policy, the message header of eDonkey has the following data structure: protocol field, message length field and message

Table 1 Notations definition

Symbol	Definition
L	length of the payload of a packet
S	length of the signature
c	content of the signature
t	type of the signature
α	zero offset signature type
β	fixed offset signature type
γ	variable offset signature type
λ	arithmetic signature type
x	position of the signature
p	the position set of the signature, $p = \{x \in 0 \leqq x \leqq (l - s)\}$
$\|p\|$	number of the elements in p set, $\|p\| > 1$ means that the signature occurs at more than one position
N	the deep length of payload which is inspected by Net-DPIS
R	the rule of identifying a service type
m	length of shortest pattern
B	the number of characters is matched in a matching cycle

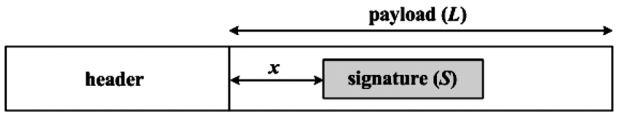

Fig. 5 *Locating the signature*

ID field [35]; eDonkey traffic is identified with counting the message size in byte and compares the counting result with the message length field in message header.

Fig. 6 shows the flowchart of traffic classification. While traffic classification module gets a packet, four policies are used to search if there are any signatures in the packet; the rule table identifies the service type of the packet according to the results of four policies.

3.4 Pattern matching

WMT (WM with trie) is an evolution from WM pattern matching algorithm. WM algorithm uses hash structure to lookup patterns and the maximum offset of shift is $(m - B + 1)$. WMT algorithm uses a trie structure to reduce the hash collisions; shift distance table (SDT) of fast string matching algorithm for network processor (FNP) algorithm [36] is referred by WMT to design the shift table of WM.

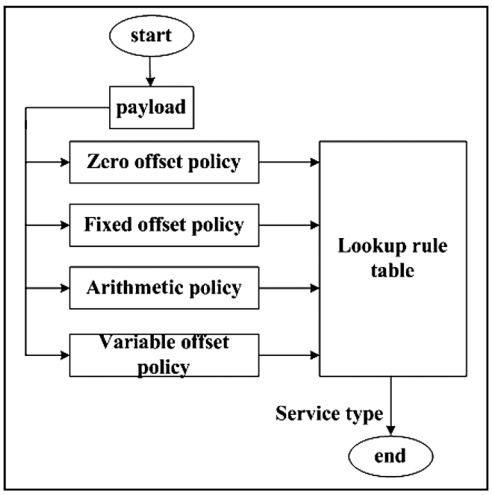

Fig. 6 *Traffic classification flowchart*

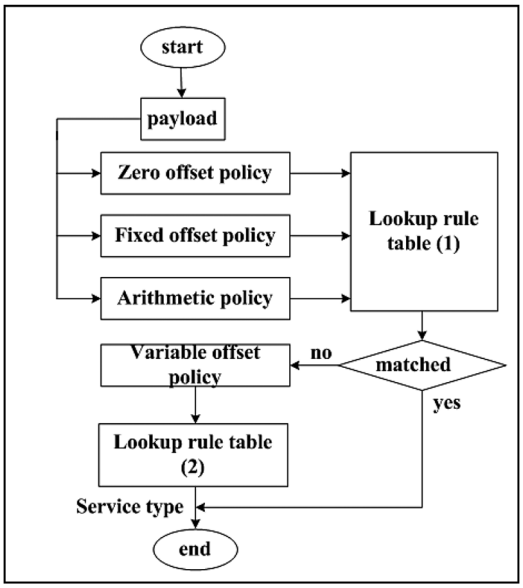

Fig. 7 *Refined traffic classification flowchart*

In the latter result of this paper will show that WMT effectively improves the cost time in pattern matching.

3.5 Rule table

To understand the features of network services is useful to prevent the unnecessary pattern matching. The rules in rule table are used to identify the service type of the network traffic; there are more than 300 rules in Net-DPIS. We rearrange the rule policies of the rule table in order to effectively reduce the time of lookup table. In the pattern matching of the four policies, the variable offset policy is the most time consuming; thus, to separate the variable signatures can reduce the time. The improved traffic classification procedure is as shown in Fig. 7. Zero offset, fixed offset and arithmetical signatures are first used for traffic classification state one. In cases of failure to search, the name of the corresponding service is returned; the variable offset policy is not required.

3.6 Connection tracking table

In general, there are more than one packet in a connection; the packets in an identified connection should not be inspected in the reason of saving computing performance. In order to prevent to match the remainder packets in an identified connection, a connection tracking table is used and records the following records: (i) source IP address, (ii)

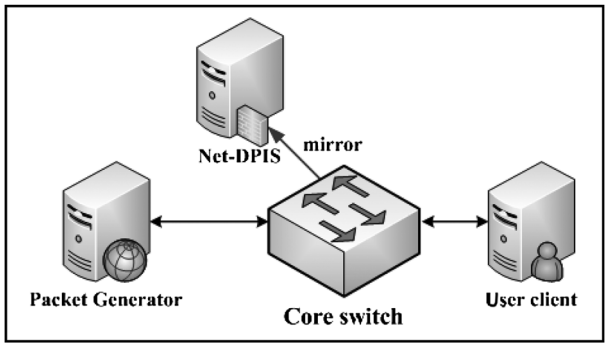

Fig. 8 *Experimental environment*

destination IP address, (iii) source port, (iv) destination port, (v) protocol, (vi) packet count, (vii) byte count, (viii) connection state, (ix) connection expired time, (x) service type. When a connection is identified, the connection information will be recorded in the connection tracking table. If a packet belongs to the identified connection, the traffic classification module is not required to match the packet; on the contrary, if the connection is not identified yet, the connection information does not exist in the connection tracking table and the traffic classification is necessary. The connection recorded by connection tracking is two-way information; when Net-DPIS receives the first packet in one connection to be identified; opposite direction information would be generated. The packet count and byte count of a connection can be obtained with the sum of both ways.

4 Performance evaluation of Net-DPIS

This section describes that we measure the time cost in each stage and reconstruct the policies of Net-DPIS to increase the performance effectively.

4.1 Experimental environment

Fig. 8 shows the experimental environment and the specification of the Net-DPIS and packet generator will be described in the following statements. The Net-DPIS is a personal computer with dual-core CPU, Ubuntu Linux 10.04 operating system, a Linux kernel version 2.6.31 and a *D*-Link DGE-530T Gigabit Ethernet Adapter; the packet generator is a personal computer with dual-core CPU, Ubuntu Linux 10.10 operating system and a Intel(R) 82567LM Gigabit Ethernet Adapter; the switch supports Gigabit Ethernet and port mirror. The packet generator generates the network traffic to the user client, and the switch will mirror the traffic between the packet generator and the user client to Net-DPIS with port mirror function.

4.2 Time measurement of Net-DPIS

This experiment is conducted to observe the time cost. The packet generator sends UDP packets to the user client. Each experiment is repeated ten times and lasts 10 s. The packet size is the sum of L2 header, L3 header, L4 header and payload length in byte; the Net-DPIS matches the signatures in the payload and is configured to inspect the payload in the depth of 60, 300, 900 and 1500 bytes. The packet generator sends user datagram protocol (UDP) packets at 1514 bytes to the user client. The results are as shown in Table 2; the unit of time is microsecond. From the results, we can find that most of time is used to lookup rule table.

4.3 Time cost of traffic classification

In this experiment, the inspection length of the Net-DPIS is at 60, 300, 900 and 1500 bytes; the packet generator sends UDP packets at 82, 342, 937 and 1514 bytes to the user client; each experiment is repeated ten times. Table 3 shows a comparison of the time cost in traffic classification; the unit of data is microsecond. The time cost of the Net-DPIS and the modified Net-DPIS are reduced with the reduction in packet length, when the packet length is fixed; the total time cost of the modified Net-DPIS is 23∼35% less than the Net-DPIS.

4.4 Time cost of look-up table

Table 4 shows a comparison of the look-up table time. The average time of the Net-DPIS is 79.4−88.1 μs. The

Table 2 Time cost in different inspection depths

	Zero offset time, ms	Fixed offset time, ms	Variable offset time, ms	Arithmetic time, ms	Lookup rule time, ms	Total time, ms
60	14.3	9.7	7.14	5.2	80.3	116.6
300	14.2	9.8	14.5	5.4	79.9	123.8
900	14.3	9.8	33.6	5.6	80.2	143.5
1500	14.4	9.9	55.1	5.3	79.4	164.1

Table 3 Time cost of traffic classification

	1514		937		342		82	
	Net-DPIS	Modified	Net-DPIS	Modified	Net-DPIS	Modified	Net-DPIS	Modified
60	116.6	79.9	117.7	80.9	124.2	80.2	117.3	79.3
300	123.8	87.1	123.2	85.4	125.9	82.4	–	–
900	143.5	104.7	130.3	94.3	–	–	–	–
1500	164.1	125.1	–	–	–	–	–	–

Table 4 Time cost of lookup table

	1514		937		342		82	
	Net-DPIS	Modified	Net-DPIS	Modified	Net-DPIS	Modified	Net-DPIS	Modified
60	80.3	43.3	80.2	43.5	88.1	44.3	81.6	43.7
300	79.9	43.7	80.4	43.4	87.9	44.0	–	–
900	80.2	43.2	79.4	43.4	–	–	–	–
1500	79.4	42.9	–	–	–	–	–	–

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.