Traffic-Aware Multiple Regular Expression Matching Algorithm for Deep Packet Inspection

Kefu Xu, Jianlong Tan, Li Guo, Binxing Fang
National Engineering Laboratory for Information Security Technologies
Institute of Computing Technology, Chinese Academy of Sciences
Beijing, China
xukefu@software.ict.ac.cn

Abstract—Deep packet inspection sometimes is called application level semantic detection, which is capable of examining the content of data packets in order to provide application-specific services and improve network security. Application traffic classification based on regular expressions is an essential step for deep packet inspection. However regular expression, especially multiple regular expression matching is known to require intensive system resources and is often a performance bottleneck. Currently, the DFAs of regular expression are constructed in the preprocessing stage and the context of network streams is excluded which leads to low throughputs.

In this paper, we analyzed the application level protocols and found that the protocols are asymmetrically distributed and it is changing dynamically. From the protocol distribution characteristic, we proposed an adaptive multiple regular expression matching method for application traffic classification with deep packet inspection. The adaptive method, schedule the multiple DFAs through splay tree by matching probability other than linear scheduling in linked list, can adjust scheduling sequence according with the changing dynamic traffics. We evaluate the proposed method with the L7 rules; experiments proved that our method can improve the throughputs more than three times.

Index Terms—regular expressions, deep packet inspection, traffic adaptive, high-speed network

I. INTRODUCTION

Deep packet inspection, as the process is called, arises as networks incorporate increasingly sophisticated services into their infrastructure. Such application-aware services use specific data found in packet payloads. The standard packet inspection process (a.k.a. shallow packet inspection) extracts basic protocol information such as IP addresses (source, destination) and other low-level connection states. This information typically resides in the packet header itself and consequently reveals the principal communication intent. Deep packet inspection, on the other hand, provides application awareness. This is achieved by analyzing the content in both the packet header and the payload over a series of packet transactions, consequently, provides the ability to analyze network usage and optimize network performance, thereby playing a crucial role in the equation between supply and demand faced by every network operator.

Deep packet inspection is sometimes called application level semantic detection, which associate packets into one data stream, maintain the state when searching for the signatures of the applications. It demands on-line speed to analyze, detect and reassemble the application level packet streams for high throughputs, especially for high-speed network.

Application traffic classification is an essential step for deep packet inspection process to forbid applications, bill on the content, detect intrusion or malicious attacks. Traditional approach to classify traffics of network flow is mainly port-based, which examines the TCP and TDP server port number in packet header, and maps the port to a higher layer application using IANA (Internet Assigned Number Authority) list of registered or well-known ports [1]. For example, if the port is 80 we can identify the flow as HTTP according to IANA. However, with the continuous development of the network and emergence of new network applications, port-based method becomes ineffective for lots of applications use random ports to transmit packets, even use well-known port to hide themselves. Some recent studies show that port-based approach can only identify 30%-70% today’s Internet traffic [2].

Deep packet inspection focuses on the packet payload for application identification. It not only views the packet header, but also examines the payload deeply to determine the application level protocol.
inspection matches packet payload with signatures of network applications, which are represented by explicit string or regular expression, in order to check whether or not a pattern appears in the packet payload. It has high identification accuracy and can map a flow to an application classes. This technique has been used in many traffic classification systems, such as L7-filter [3] (the Linux Application Protocol Classifier), Ethereal [4] (the world’s most popular network protocol analyzer) and OpenDPI [5] (an open source version of Ipoque’s commercial DPI engine, released in September 2009), all of them contain a large number of signatures which describe the characters of applications.

The biggest problem for deep packet inspection is that regular expression, especially multiple regular expression matching is known to require intensive system resources and is often a performance bottleneck. It needs a high storage and computational cost to match every packet that traverses a link, which makes it impossible to classify traffic at very high-speed links online. The DFAs of the multiple regular expressions are constructed and organized in the preprocessing stage and the information of network streams is excluded, which leads the poor performances in network environments that often lead to low throughputs. In this paper, we proposed an adaptive multiple regular expression matching method for application traffic classification. The adaptive method, schedule the multiple DFAs through splay tree by matching probability other than linear scheduling in linked list, can adjust scheduling sequence according to the change of dynamic traffics, which is suitable to identify applications in online speed. We make the following contributions:

- We analyzed the application level protocol distribution and summarize the characteristic of the distribution.
- From the characteristic, we propose a more adaptive method scheduling the multiple DFAs through splay tree by matching probability other than linear scheduling in linked list.
- We evaluate the proposed method with the Linux L7 system. Experiments proved that our adaptive method can improve the throughputs more than three times.

In the following section, we present some of the related work (Section II). In Section III we analyzed the asymmetry of the protocol and its distribution. Then the technique will be explained in detail and analyzed in Section IV. After that in Section VI, the technique’s performance is evaluated and discussion about its applicability is shown. Finally, we end with final remarks, conclusion and future work in Section VII.

II. RELATED WORKS

Application traffic classification is a helpful technique for Internet service providers (ISPs) and enterprises. There are 3 parallel lines of research for traffic classification: port-based, payload-based and machine learning, each of them has its own strong points and shortcomings. The researchers of machine learning strongly disapprove of deep packet inspection, because privacy laws will make the payload inaccessible and it need a high storage and computational cost.

Traditionally, the patterns of applications are represented by explicit strings. A. C. Yao analyzes the low bound of time complexity of string matching for a random string [6], and G. Navarro et al. extend the bound to exact and approximate multiple string matching [7]. With the development of network, there is an increasing requirement of high throughout and supporting large-scale patterns for string matching. Various new concepts and algorithms, either software-based or hardware based, have been proposed and implemented, such as ACBitmap [8], reconfigurable silicon hardware [9] and TCAM based method [10]. The bloom filters [11] have received much attention, and they are now being used in many systems, such as web caches, database systems etc.

Recently, regular expressions are replacing explicit strings as the choice of pattern describing language in packet payload scanning applications. For example, all protocol patterns in L7-filter [3] are expressed as regular expressions. More and more newer systems are replacing strings with regular expressions. The Snort NIDS [12] has evolved from no regular expressions in its rule set in April 2003 to 1131 (out of 4867 rules) using regular expressions as of February 2006. The widespread use is due to the expressive power, simplicity and flexibility for describing useful patterns.

In regular expression matching algorithms, the regular expression is first parsed into an expression tree, which is transformed into a Nondeterministic Finite Automaton (NFA) in several possible ways, the most interesting in practice is the Thompson construction [13] and the Glushkov construction [14]. It is possible to search directly with the NFA, and there are various ways to do that, but the process is quite slow. The algorithm consists in keeping a list of active states and updating the list each time a new character is read. The search is normally worst-case time $O(m^2)$ ($m$ is the length of packet payload), but it requires little memory. Another approach is to convert the NFA into a Deterministic Finite Automaton (DFA), which permits $O(n)$ search time by performing one direct transition per text character. On the other hand, the construction of such an automaton is worst-case time and space $O(2^n)$.

The traditional techniques to search for regular expressions can not adaptive to network traffic context. The DFAs are constructed in the preprocessing stage and the context of network streams is excluded. This leads that the performances for matching regular expressions vary in network environments and often leads to low throughputs. The network stream is naturally dynamic and protocol distribution is asymmetrical, which should be included into the design of regular expressions searching algorithms.

III. PROTOCOL DISTRIBUTION

Previous studies of Internet trace have shown that a very small percentage of flows consume most of the
network bandwidth [15]. So far, there are already many researchers such as Karagiannis [16] investigated the application level protocol distribution of different network traffics. Ipoque Company analyzed the Internet traffics in five regions of the world between August and September 2007 [17]. Figure 2 shows the protocol type distribution based on the size of protocol traffic. In figure 2 HTTP do not include any audio or video streaming content embedded in Web pages like YouTube, which is counted separately. P2P is composed of BitTorrent and eDonkey etc., and the corresponding proportion is showed in figure 2(b).

Figure 2. Deep packet inspection analyzes the application level content.

![Figure 2](image)

According to the CNCERT’s (Chinese Network Center for Emergency and Repsonses Technology) traffic statistics in 2008 [18], in TCP, there are four applications, Web exploring, P2P downloads, Emails and the chat tools that consume the most traffic bandwidth. The first ten TCP protocol port traffics are as figure 3 and the first ten TCP protocol port service are as table I.

In UDP, the ports consume the most traffic bandwidth are the P2P downloads applications, such as eMule, BT. The DNS services also consume large bandwidth, 1.34% of the total. The first ten UDP protocol port traffics are as figure 4 and The first ten UDP protocol port services is as table II.

Many Internet and private traces shows that the frequency distribution for some of the traffic properties appears to be highly skewed [19]. Utilizing traffic characteristics to the optimization process was addressed by many researchers [20]. In [21] the work was extended.

---

**Table I. The first ten TCP protocol port services**

<table>
<thead>
<tr>
<th>UDP ports</th>
<th>Traffic ranking</th>
<th>Percentage</th>
<th>Main services</th>
</tr>
</thead>
<tbody>
<tr>
<td>15000</td>
<td>1</td>
<td>4.43%</td>
<td>P2P download</td>
</tr>
<tr>
<td>53</td>
<td>2</td>
<td>1.34%</td>
<td>DNS service</td>
</tr>
<tr>
<td>80</td>
<td>3</td>
<td>1.22%</td>
<td>Web pages</td>
</tr>
<tr>
<td>8000</td>
<td>4</td>
<td>1.20%</td>
<td>QQ communication</td>
</tr>
<tr>
<td>29909</td>
<td>5</td>
<td>1.12%</td>
<td>Unknown port</td>
</tr>
<tr>
<td>7100</td>
<td>6</td>
<td>0.95%</td>
<td>Online games</td>
</tr>
<tr>
<td>1026</td>
<td>7</td>
<td>0.88%</td>
<td>MS Messenger</td>
</tr>
<tr>
<td>1027</td>
<td>8</td>
<td>0.75%</td>
<td>MS Messenger</td>
</tr>
<tr>
<td>6881</td>
<td>9</td>
<td>0.41%</td>
<td>P2P download</td>
</tr>
<tr>
<td>1434</td>
<td>10</td>
<td>0.38%</td>
<td>SQL Server Resolution</td>
</tr>
</tbody>
</table>

**Table II. The first ten UDP protocol port services**

<table>
<thead>
<tr>
<th>UDP ports</th>
<th>Traffic ranking</th>
<th>Percentage</th>
<th>Main services</th>
</tr>
</thead>
<tbody>
<tr>
<td>15000</td>
<td>1</td>
<td>4.43%</td>
<td>P2P download</td>
</tr>
<tr>
<td>53</td>
<td>2</td>
<td>1.34%</td>
<td>DNS service</td>
</tr>
<tr>
<td>80</td>
<td>3</td>
<td>1.22%</td>
<td>Web pages</td>
</tr>
<tr>
<td>8000</td>
<td>4</td>
<td>1.20%</td>
<td>QQ communication</td>
</tr>
<tr>
<td>29909</td>
<td>5</td>
<td>1.12%</td>
<td>Unknown port</td>
</tr>
<tr>
<td>7100</td>
<td>6</td>
<td>0.95%</td>
<td>Online games</td>
</tr>
<tr>
<td>1026</td>
<td>7</td>
<td>0.88%</td>
<td>MS Messenger</td>
</tr>
<tr>
<td>1027</td>
<td>8</td>
<td>0.75%</td>
<td>MS Messenger</td>
</tr>
<tr>
<td>6881</td>
<td>9</td>
<td>0.41%</td>
<td>P2P download</td>
</tr>
<tr>
<td>1434</td>
<td>10</td>
<td>0.38%</td>
<td>SQL Server Resolution</td>
</tr>
</tbody>
</table>
to multiple fields in firewall policies with the capability of parallel processing. The authors in [22] proposed a technique that is based on a specialized policy encoding (i.e., policy segments) in order to build Huffman trees that adapt to the traffic statistics. The technique can also be parallelized and its worst case could be bounded. An approach to find an optimal ordering of rules while maintaining policy semantics was addressed in [23]. The maintained form of the rules makes it a plausible preprocessing phase to any other technique. In [24], a hybrid approach between software and hardware was proposed; it also incorporates the traffic statistics to dynamically build new rules in the form of a cache. These new rules have better matching ratio than using original rules from the rule set. In [25], depth-constrained alphabetic trees are used to reduce lookup time of destination IP addresses of packets against entries in the routing table. The authors show that using statistical data structures can significantly improve the average-case lookup time. As the focus of the paper is on routing lookup, the scheme is limited on search trees of a single field with arbitrary statistics. In addition, the paper provides no further details on traffic statistics collection and dynamic update of the statistical tree.

IV. ADAPTIVE MULTIPLE REGULAR EXPRESSION MATCHING METHOD

A. Multiple Regular Expression

Definition 1 (Finite Automaton). Finite automaton is a finite set of states \( Q \), among which one is initial (state \( I \) \( Q \)) and some are final or terminal (state set \( F \subseteq Q \)). Transitions between states are labeled by elements of \( \Sigma \) \{\( \varepsilon \)\}. These are formally defined by a transition function \( D \), which associates to each state \( q \in Q \) a set \{\( q_1 \), \( q_2 \), \( \ldots \), \( q_k \)\} of states of \( Q \) for each \( \alpha \in \Sigma \). An automaton is then totally defined by \( A = (Q, \Sigma, I, F, D) \).

In practice, there are two general types of automata, depending on the form of the transition function.

Definition 2 (Nondeterministic Finite Automaton). If the function \( D \) is such that there exists a state \( q \) associated by a given character \( \alpha \) to more than one state, say \( D(q, \alpha) = \{ q_1, q_2, \ldots, q_k \} \), \( k > 1 \), or there is some transition labeled by \( \varepsilon \), then the automaton is called a nondeterministic finite automaton (NFA), and the transition function \( D \) is denoted by the set of triples \( \Delta = \{ (q, \alpha, q') | q \in Q, \alpha \in \Sigma, \{ \varepsilon \} \} \).

Definition 3 (Deterministic Finite Automaton). Deterministic finite automaton \( D \) is denoted by a partial function \( \delta : Q \times \Sigma \to Q \), such that if \( D(q, \alpha) = \{ q' \} \), then \( \delta(q, \alpha) = q' \). We give examples of both types of automata in figure 5. In both, the state 0 is initial state and the double-circled states are terminal. The left automaton is nondeterministic since from the state 0 by \( T \) we reach 2 and 6. The right one is deterministic because for a fixed transition character all the states lead to at most one state.

Definition 4 (Pattern Recognized by Automaton). A pattern is recognized by the automaton \( A = (Q, \Sigma, I, F, \delta) \) if it labels a path from an initial to a final state. The language recognized by an automaton is the set of patterns it recognizes. For instance, the language recognized by the automaton in figure 5 (a) is the set of patterns: \( A \) in state 8, \( ATAT \) in states 7 and 8, \( T \) in state 6, \( TC \) in state 8, \( TAG \) in state 6, and finally \( TAGC \) in state 8.

Application traffic classification by regular expressions can commonly divided into several steps. Firstly, the signatures of each application protocol are analyzed, extracted and written in the form of regular expressions. Then the regular expressions are compiled into NFAs and the NFAs are converted into DFAs. The DFAs require too much storage space, it is impossible to compile all the NFAs into one DFA. So NFAs are accordingly compiled into individual DFAs. The DFAs are organized into a linked list and when the packets arrive, DFAs in the linked list are selected one by one from left to right linearly to search for the signatures in the packets, as is showed in figure 6.

B. The Problem of DFA Scheduling

As we analyzed in the section III, application level protocol distribution is asymmetric, DFAs in the linked list has different probability to matching successfully. Currently, DFAs are ordered in fixed sequence, and are scheduled in fixed linear sequence to identify the traffic; it will try too many times to make a successfully matching. We know that the regular expression algorithms matching are slow and need much computing resource; linear scheduling will consume much more time and slow down the throughputs. To reduce the regular expression matching time and improve the throughputs, a method is to adjust the schedule sequence to make the highest matching probability DFA be select to check the
traffic firstly. Another problem is that the network stream is naturally change dynamically, sometimes, some DFAs have the higher probability to be matched, in the other time, the probability changed and the schedule sequence should be changed accordingly.

In deep packet inspection applications, the DFAs scheduling problem can be described as follows:

- Is it returned when any rule is matched, or it needs to find all rules? If it is the former, the signatures searching will stop when any DFA is successfully matched; otherwise, all the DFAs need to be checked. Here we assume it is the former, as is used in most network security applications.
- Supposing there are k (1 ≤ i ≤ k) DFAs, and the i-th DFA has the probability \( p_i \) to be matched, then the probability of all the DFAs that not matched is \( q = 1 - \sum_{1 \leq i \leq k} p_i \). If the value of \( q \) is large, the time for the matching is nearly to the worst case, on the other words, it needs to traverse all the DFAs.
- If the \( p_i \) is changed dynamically, how to schedule the DFAs to adaptive to the flows?

We suppose that:
(a) DFAs are scheduled in linear sequence;
(b) It is returned when any rule is matched.

The DFAs scheduling can be illustrated as figure 7.

As is showed in figure 7, the mean time price for scanning is:

\[
T = p_1 + p_2 + \cdots + p_k + (1 - p_1 - p_2 - \cdots - p_k)
\]  

(1)

It can be proved that to minimize the mean time price, the optimal scheduling sequence must suffice the following conditions:

\[
\frac{c_1}{p_1} > \frac{c_2}{p_2} > \cdots > \frac{c_k}{p_k}
\]  

(2)

If \( c_1 = c_2 = \cdots = c_k = O(n) \), the mean time price is:

\[
T = p_1 + 2p_2 + \cdots + kp_k + h(1 - p_1 - p_2 - \cdots - p_k)
\]  

(3)

The optimal scheduling sequence is:

\[
p_1 > p_2 > \cdots > p_k
\]  

(4)

C Adaptive DFA Scheduling Method Based on Improved Splay Tree

According to the dynamic and asymmetric distribution network traffic, the traditional DFA scheduling method should be adapted. Here we use an adaptive DFA scheduling method based on improved splay tree.

Assuming that there are a serial lookup operations in tree-based data structures, to minimize the times of the lookup operations, the most frequent items should be near to root of the tree. We need to adjust the structure of the tree every time after the lookup operations and try to move the item that currently matched near to the root. After some time, the high frequent items will centralized on the root of the tree.

Splay tree is a binary sort tree, and is constructed by Daniel Sleator and Robert Tarjan [27]. Splay trees are standard example of self-adjusting binary search trees. They have great advantages over explicitly balanced trees, as they automatically adapt to various non-uniform access patterns. Nodes of splay balanced trees have the relative values while DFAs are equality, so we improved the splay tree.

At first, we construct a complete binary tree to stand for a splay tree, compile the rules into DFAs and insert every DFA as nodes into the tree, as is shown in figure 8. Then levelly traverse the tree and find the DFA that can identify the current application protocol. If any DFA is successfully matched, stop the traverse and adjust the tree according to adaptive policy by moving the node contain the DFA up nearly to the root of the tree.

After a period of time adjusting, the higher probability DFAs will centralized on the root, the others will sinking down to the bottom of the tree and the scheduling could also be adaptive to dynamic traffics. By statistics, the times for searching DFAs will reduce and the throughput will increase.

The new algorithm may be described as follows:

Procedure Adaptive-Multiple-RE(TCP-links, Splay-tree, DFA)

(1) Compile Regular Expressions Into NFAs;
(2) Convert NFAs into DFAs;
(3) Construct the Improved Splay-tree;
(4) Insert the DFAs into the Splay-tree;
(5) Searching // Check each TCP-links with the DFAs;
(6) Read the packets from the TCP-links;
(7) For (TCP-link0, TCP-link-Last) output protocol
of search of TCP-links
(8) Do
(9) Level traverse the splay tree from the root;
(10) If any DFA matching successfully, report the DFA;
(11) Stop the traverse;
(12) Adjust the tree according to adaptive policy in advance;
(13) Go back to the root of the splay tree;
(14) End If;
(15) End Do;
(16) End For;
(17) End;

V. EVALUATION AND ANALYSIS

A Experiments Environment
We tested the two scheduling method and contrasted the scheduling number of times and throughputs between them. The testing machine is VMware Workstation 6.5-7.0 over windows XP OS, the CPU is Intel(R) Core(TM)2 Duo 2.40GHZ, 1.00GB of RAM. The traffic classification rules are extracted from Linux L7-filter. Three are total 125 regular expressions. We gather 350GB data set from the Chinese core network; there are total 615,924 TCP stream links in the data set.

B Scheduling Number of Times
We tested the scheduling number of times in linear scheduling and adaptive scheduling. At the start, the method of adaptive splay-tree needs a little time to adjust to the traffic. Figure 9 is the scheduling number of times for the first five hundreds TCP links of the linked list method. The DFAs organized in the linked list need to try average 36.2 times. The method base on splay tree adjusting speed is fast, after a little time for adjusting, the scheduling number of times reduced notably and the meaning scheduling times is 13.7 as is shown in figure 10.

Figure 9. Scheduling times for the first five hundreds TCP links based on linked list

Figure 10. Scheduling times for the first five hundreds TCP links based on splay tree

Figure 11 is the scheduling number of times contrast between the two methods. At the beginning, there are nearly the same scheduling times for the two methods, but after a little time adjusting, the scheduling times reduced rapidly for the adaptive method based on splay tree. The method based on linked list need more than two times scheduling numbers before a successfully matching. After a little more time, the times for adaptive scheduling will be even less.

In figure 12, 13 and 14, we give out the scheduling times and the contrast for one thousand TCP links from 300,001th to 301000th. The adaptive scheduling times is less than 10 after adjusting and the scheduling number of
times is obviously less than the linked list, as is shown in figure 14.

![Figure 14. Scheduling times contrast between the two methods for one thousand stream links.](image)

C Throughputs Contrast

There are total 615,924 TCP stream links in the data set. We observe the run time from the fixed linked list method and the traffic-adaptive method. Above 90% TCP stream links could be recognized. The throughputs could be raised more than 2.5 times by the traffic-adaptive method than the fixed linked list method as is shown in table III. Consider that the 2.5 times performance upgrade is of the whole system, there are many resource and time consuming operations include the data reading, TCP stream assembly, stream state maintain, stream searching. So only consider the scheduling algorithms, the actually performance optimization by traffic-adaptive scheduling algorithms should be about 3.5-4 times.

VI. CONCLUSIONS

In this paper, we analyzed the application level protocol distribution and proposed an adaptive multiple regular expression matching method for application traffic classification with deep packet inspection. The adaptive method, schedule the multiple DFAs through splay tree by matching probability substituted as linear scheduling in linked list, can adjust scheduling sequence according to the change of dynamic traffics. We evaluate the proposed and compare it with the L7 system and the actually performance improve by traffic-adaptive method is about 3.5-4 times. The Traffic adaptive is suitable to identify applications online.

<table>
<thead>
<tr>
<th>Numbers of TCP links</th>
<th>Time for Linked list (s)</th>
<th>Time for Splay tree (s)</th>
<th>Recognition ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test1 615924</td>
<td>450</td>
<td>188</td>
<td>89.1%</td>
</tr>
<tr>
<td>Test2 615424</td>
<td>421</td>
<td>190</td>
<td>89.1%</td>
</tr>
<tr>
<td>Test3 615924</td>
<td>474</td>
<td>209</td>
<td>89.1%</td>
</tr>
<tr>
<td>Mean result 615924</td>
<td>445</td>
<td>195.6</td>
<td>89.1%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>REFERENCES</th>
</tr>
</thead>
</table>

Kefu Xu received the Ph.D. degree in computer science from The Center South University of Technology, Guangdong, China, in 2009. He is now a researcher as postdoctoral at Institute of Computing Technology, Chinese Academy of Sciences. He has published over 10 research papers in journals and conferences in the field of computer networks. His research interests include network measurement and monitoring, network security. Particularly, his research interests include algorithm design, high-speed deep packet processing systems and converged networks.

Jianlong Tan received the Ph.D. degree in computer science from The Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 2003. He is an associate researcher and master supervisor at Institute of Computing Technology, Chinese Academy of Sciences. His main research interests include algorithm design, data stream management and information security. He has published over 30 research papers in journals and conferences in computer network.

Li Guo received the Master degree in computer science from The Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 1999. She is now a senior research engineer and master supervisor at Institute of Computing Technology, Chinese Academy of Sciences. Her main research interests include information security, and data stream processing. She has published over 40 research papers in leading journals and conferences in computer network and security.