Postprint: Research on Line-Rate Deep Packet Inspection for High-Speed Networks

Authors: Xu Kefu, Li Yang, Tan Jianlong, Guo Li

Date: 2017-03-10T00:00:00+00:00

Abstract

As the functionality of networks gradually evolves from simple shallow packet processing to complex applications based on deep packet processing, line-rate deep packet inspection technology suitable for high-speed core networks (10 Gbps) has become a research hotspot. This paper identifies two key scientific problems in deep packet inspection systems: "the contradiction between speed and performance" and "semantic distortion." Focusing on these two issues, it provides a systematic review and in-depth analysis of various optimization techniques and strategies that have emerged in detection methods, detection models, detection algorithms, and other aspects, discusses related research trends and development directions, and summarizes future research directions.

Full Text

Preamble

Vol. 9 No. 3 Information Technology Express
Research on High-Speed Network Line-Rate Deep Packet Inspection
Xu Kefu, Li Yang, Tan Jianlong, Guo Li

Abstract

Network functionality is gradually evolving from simple shallow packet processing to complex applications based on deep packet processing, making line-rate deep packet inspection technology suitable for high-speed core networks (10Gbps) a research hotspot. This paper identifies two key scientific problems in deep packet inspection systems: the "contradiction between speed and performance" and "semantic distortion." Focusing on these issues, we systematically review and analyze various optimization techniques and strategies in detection methods, detection models, and detection algorithms, discuss relevant research trends and developments, and summarize directions for further research.

Keywords: Deep Packet Inspection; Signature Set; Pattern Matching Algorithm; Performance Evaluation

1 Introduction

Deep Packet Inspection (DPI), sometimes also called Completely Packet Inspection, provides application-layer semantic awareness by analyzing the headers and especially the payload content of a series of packets. It belongs to application-level semantic detection and is one of the fundamental means and key technologies for ensuring network information security.

For network service providers, deep packet inspection can examine layers two through seven of the OSI (Open System Interconnect, the ISO reference model for open systems interconnection) reference model, enabling a range of new network security functions such as lawful interception, policy definition and enforcement, copyright protection, content filtering, and more. For enterprises, deep packet inspection has become the core theory and key technology for network information security. For example, Network Intrusion Detection/Protection Systems (IDS/IPS) need to detect whether packet payloads contain malicious computer viruses; security routers need to determine whether packets contain network worm code before forwarding; and spam filtering programs need to scan messages for spam information. Additionally, deep packet inspection allows fine-grained control and can effectively prevent buffer overflow attacks, DoS (Denial of Service) attacks, and intrusions by experienced hackers. For government departments, deep packet inspection serves as an important means of managing information dissemination, enabling investigation of network-sensitive content, network public opinion analysis and dissemination, and monitoring and review of network traffic, which is of great significance for maintaining social stability, promoting national economic development, and even strengthening national defense construction.

This research was supported by the National 973 Key Basic Research Development Program and the National Natural Science Foundation of China project "Research on Line-Rate Deep Packet Inspection for Network Environment Information Perception." This paper takes the dynamically changing network environment as its research object, analyzing the impact of various network environmental factors such as link-layer traffic environment and application-layer semantic environment on deep packet inspection performance. From the perspective of deep packet inspection mechanisms (detection methods—detection models—detection algorithms), we conduct an in-depth comparison and analysis of existing theoretical foundations, various optimization techniques, and strategies related to deep packet inspection, and discuss relevant research trends and developments.

2 Related Research

In recent years, network bandwidth and traffic have increased dramatically, with numerous new network protocols and applications continuously emerging, placing higher demands on deep packet inspection technology for real-time performance and more complex semantic support. Below, we review current research from the perspectives of deep packet inspection models, detection algorithms, and network environment information.

2.1 Deep Packet Inspection Models

Overall, existing detection models lack accuracy, flexibility, and scalability, provide insufficient support for more complex semantics, and have not yet solved the line-rate problem for deep packet inspection systems. Deep packet inspection systems are one of the core technologies for ensuring a cleaner cyberspace and more complete national sovereignty in cyberspace, involving sensitive issues such as state secrets and citizens' freedom of speech, and have received high priority from all countries. Currently known typical research programs include the following categories:

(1) Scalability: A typical example is the STREAM project supported by the US NSF (National Science Foundation). Its main research objective is to develop a universal data stream management system, including providing a universal and flexible architecture, related theoretical results and algorithms, data models, related languages and semantics, and exploring the querying and processing of multiple continuous, rapid, and variable network streams. Currently, their research objectives focus primarily on scalability and performance optimization.

(2) Semantic Support: A typical example is the NIAGARA project, also supported by the US NSF, whose main research objective is an XML (Extensible Markup Language) data retrieval and filtering system in Internet environments. This system collects and monitors information on the Internet, then packages it as XML data streams for retrieval and filtering. By utilizing the semantic information of XML, more accurate data stream retrieval and filtering can be provided.

(3) Adaptability: A typical example is the Telegraph project at UC Berkeley. Its research objective is to provide adaptive queries for data such as network monitor output and web data. The project's feature is adaptive query processing for data streams, including adaptive joins and adaptive operation adjustments. Another collaborative project between MIT and Brown University—Aurora—also aims to monitor data streams from various embedded devices. Cornell University also has a project called Cougar for querying and monitoring sensor data.

(4) Line-Rate Processing: The line-rate processing of network equipment is currently a hot issue of concern in both academia and industry. So-called line-rate processing means that input data from a processing device can be

transmitted to the output without delay after processing (when draining is completed), i.e., it can be processed at "online transmission rate." This is almost impossible in practice because any processing requires time. In reality, line-rate processing generally means that input delay is controlled within a certain range, such as within 20% of the input speed. The effectiveness of line-rate processing methods is closely related to two factors: processing complexity and input speed. For complex processing, achieving line-rate processing is obviously much more difficult; for processing of the same complexity, higher input speed clearly makes line-rate processing more difficult. So far, only limited papers have addressed the system architecture issues of line-rate processing, especially for information flow line-rate processing related to network environments.

2.2 Deep Packet Inspection Algorithms

There have been dozens of deep packet inspection algorithms to date, with different principles. Traditional detection algorithms typically have two phases: the first phase is the preprocessing stage, where given a pattern set P, all patterns are preprocessed according to different algorithm principles using different data structures to generate specific detection data structures (generically called "detection automaton"); the second phase is the search stage, where given text T is searched to find all results. Currently, detection algorithms remain the system' s speed bottleneck for two reasons, arising from the above two phases:

(1) Detection algorithms cannot adapt to changing network data streams. The core of detection algorithms—the detection automaton—is generated during the preprocessing stage and is completed solely through processing the pattern set itself, without considering the network environment information where the algorithm runs. This creates a problem: under different conditions, detection algorithm performance varies dramatically, with theoretical analysis results not matching actual running effects.

In previous research, detection data was always assumed to be static and random, i.e., like the pattern set, it follows an "equal-probability uniform distribution." Therefore, detection data was considered not a given condition of the algorithm and theoretically did not need to be considered. Based on this premise, researchers established the time complexity lower bound for single-pattern matching algorithms as O(n/m), and for multi-pattern matching algorithms as O(n/m). Many algorithms have already achieved or approached this optimal value.

However, network data streams are inputs during application and have natural dynamic characteristics. Considering the characteristics of data to be detected in detection algorithm research is only recent work. Reference [1] analyzed the relationship between detection algorithms and random data entropy, presented detection algorithms for entropy-bounded text, and proved their average time complexity and worst-case time complexity. This research concluded that the traditional assumption of "equal-probability uniform distribution" for detection

data is unreasonable. However, while this work considered the probability distribution of patterns and random data, it did not consider their relationship in practical application environments. The impact of dynamic application data streams on detection algorithm performance requires further research.

(2) The state space of detection algorithms during detection is enormous. Among the many factors affecting detection algorithm performance, storage space size occupies an increasingly important position. This is because automata are the main data structures of pattern detection algorithms, and as the scale of pattern sets increases, the huge storage space required leads to poor cache locality, thereby reducing algorithm detection speed. Therefore, how to reduce the storage space of pattern detection algorithms and optimize cache locality is also a research direction in recent years.

Reference [2] proposes a fragmentation algorithm that selects fragments causing DFA (Deterministic Finite Automaton) state explosion and isolates them, thereby reducing the storage requirements of individual regular expressions. Additionally, the literature proposes a selective grouping algorithm based on the combinatorial relationships of regular expressions, which significantly reduces the number of state machines under acceptable total storage requirements, effectively reducing the complexity of matching algorithms. Reference [3] uses a data structure based on Bloom filter to map multiple rules onto a one-dimensional vector space, effectively reducing storage space requirements. Reference [4] uses statistics on character frequency in patterns to design new detection automaton structures, reducing the data structure space usage of the classic AC algorithm by 75% to 84%. Reference [5] studied the relationship between classic detection algorithm performance and cache, and proposed corresponding cache optimization strategies for different types of algorithms. Experimental results show that automaton storage space can be compressed to no more than 5% of the original storage space, with speed improvements of 2-3 times.

Analyzing the above two phases, we consider that automata should be created or have states added/removed adaptively during the detection phase rather than the preprocessing phase, according to changes in network traffic and based on the input data stream. This way, automata can both adapt to dynamic network data streams and greatly reduce space requirements, thereby eliminating the algorithm' s speed bottleneck.

2.3 Network Environment Information

Research trends and developments show that traffic environment information and application-layer semantic environment information have significant impacts on both the speed and accuracy of deep packet inspection.

(1) Traffic Environment Information (Improving Detection Speed): Reference [6] research shows that the frequency distribution of network traffic characteristics is non-uniform, with a few traffic characteristics appearing at very high frequencies. Reference [7] points out that identifying and spe-

cially treating the network flows that occupy most of the network traffic is beneficial for optimizing resource utilization. References [8] and [9] show that a large proportion of network traffic carries identical or similar network information content. In our previous research [10, 11], we conducted statistics on application-layer protocol distribution in China' s high-speed backbone network traffic and found that application-layer protocol type distribution is extremely non-uniform, while application-layer data content distribution also shows imbalance. Popular, widely-used content is clicked by large numbers of users, and the traffic carrying this content occupies a large proportion of network traffic. Our analysis of observed data found that content transmitted in network traffic is skewed and concentrated in some currently popular, commonly used files, including: Web pages, music files, peer-to-peer (P2P) shared files, worm viruses, and spam. Therefore, traditional deep packet inspection models and detection algorithms based on static random data can no longer adapt to dynamically changing network traffic in terms of theoretical design and performance analysis.

Reference [6] utilizes research on network traffic attribute characteristics from reference [12] to propose optimization methods for deep packet inspection. Reference [6] proposes a specialized encoding technique for building adaptive traffic statistics Huffman trees, which can use parallel processing and has a lower bound in its worst case. Reference [13] uses statistical search trees in the form of caches to dynamically establish new rules. These new rules have higher hit probabilities and support adaptive adjustment based on predicted network traffic changes—traffic characteristics are reflected in the imbalance during tree traversal, where the final results of extensive search traversals are always concentrated on a few leaf nodes. Reference [14] studied adaptive traffic packet filtering, a real-time packet classification mechanism that achieved good results.

In references [15, 16], we conducted statistics on application protocol distribution in China' s high-speed backbone network traffic and found that application protocol type distribution is extremely non-uniform. Although there are more than 100 application-layer protocols, the top 10 most used protocols account for over 95% of network traffic, as shown in Table 1 . At the same time, as mentioned earlier, application-layer data content distribution also shows imbalance. Traditional methods can no longer adapt to dynamically changing network traffic.

(2) Application-Layer Semantic Environment Information (Improving Detection Accuracy): Reference [17] points out that the requirement for context-related semantic information makes traditional pattern matching methods for finding specific patterns in detection data no longer sufficient. Deep packet inspection programs rely on application-layer semantics to make effective decisions, requiring more research to establish scalable and efficient pattern matching solutions. At the same time, more and more network security systems rely on protocol analysis to extract application-layer semantic environment information from network data streams.

Application-layer protocol analysis is increasingly becoming an important component of deep packet inspection. Reference [18] proposes an intrusion detection system based on dynamic application-layer protocol analysis. For each connection, this method first determines the possible application-layer protocol type, then uses an appropriate analyzer to analyze the protocol and extract application-layer semantics for decision-making, achieving good results. Reference [19] proposes a General Application-level Protocol Analysis (GAPA) architecture and prototype. GAPA can quickly perform online protocol analysis, and its work is worth learning from, though it is slightly insufficient in adaptability, flexibility, and decision accuracy.

Reference [17] believes that in deep packet inspection pattern matching algorithms, understanding complex protocols and application-layer semantics is very beneficial for interpreting the significance of successful matches. Surveys show that nearly 80% of attacks come from the application layer [20]. Dealing with these attacks is impossible with only data syntax-level detection; it is necessary to analyze the content of data and its communication purpose and intent at the application layer. Reference [21] improves detection capability by using decision tree technology to implement application protocol analysis, as shown in Figure 1 [Figure 1: see original paper]. This protocol analysis extracts specific parts of protocols, which can greatly reduce the feature search space. At the same time, this research points out that application-layer semantic-based processing methods provide high-level abstraction and are a suitable choice for improving detection accuracy.

3.1 Key Scientific Problems

Due to the sharp increase in network bandwidth and traffic and the continuous emergence of numerous new protocols and applications, deep packet inspection must support higher real-time requirements and more complex semantic support requirements. The “contradiction between speed and performance”and “semantic distortion” are two key scientific problems and technical difficulties in current deep packet inspection systems.

The goal of deep packet inspection is to deeply examine the essence of interactions between hosts and make decisions based on application-layer semantics. However, existing deep packet packet inspection models and algorithms assume static random data at the syntax level in both theoretical design and performance analysis. In actual environments, within high-speed network traffic, the object of detection—network data—changes constantly, network traffic characteristics continue to evolve, various new network protocols emerge endlessly, and the application-layer semantic environment becomes increasingly complex and diverse. Static random data can neither reflect the characteristics and changes of real network traffic nor help understand complex protocols and application-layer semantics, nor explain the significance of successful detection, often causing serious “semantic distortion.”

At the same time, due to the rapid expansion of network information, complex processing of network information through deep packet inspection causes significant delays in information output, reducing network transmission speed and causing serious bandwidth loss. This delay generally grows as a non-linear function of processing complexity. For high-speed networks such as IPv6, ATM, and Gigabit Ethernet, high-speed arriving network packets must be inspected without delay at line-rate (On-line Speed). If the inspection speed cannot keep up with network data transmission speed, it will cause serious congestion, and the inspection system will miss and misinspect some of the data, resulting in false negatives and false positives that affect system accuracy and effectiveness.

3.2 Technical Challenges

Below, we conduct an in-depth comparison and analysis of existing theoretical foundations, various optimization techniques, and strategies related to deep packet inspection from the perspective of deep packet inspection mechanisms (detection methods—detection models—detection algorithms).

3.2.1 Deep Packet Inspection Methods

3.2.1.1 Signature Sets "Signature" is a key concept in deep packet inspection. In law, fingerprints are used to identify whether a citizen (or individual) is involved in a criminal event; in deep packet inspection, signatures are used to identify applications or malicious behaviors. Analyzing thousands of network applications requires providing organized, systematic identification methods. Broadly speaking, signatures are pattern fragments. These pattern fragments are selected and should uniquely identify the associated application (as shown in Figure 2 [Figure 2: see original paper]). For example, application protocols use special headers to initialize and control information transmission; network intrusions, worms, and viruses contain specific code fragments; specific information to be queried can also be considered signatures. These signatures are generally represented in the form of regular expressions. When new applications or malicious behaviors are discovered, they are analyzed, appropriate signatures are designed, and added to the database. This database is generally called a signature library. Due to space limitations, this paper does not comment on methods and techniques for analyzing and discovering new signatures; see references [22, 23] for details.

The complexity of signature sets and their description methods have a serious impact on detection speed and accuracy. The main characteristics of signature sets that make deep packet inspection systems difficult to implement at line-rate and affect detection accuracy and practicality are the following four aspects:

(1) Network protocols are diverse, with new protocols continuously emerging and new security attacks endless, resulting in a huge number of signatures. For example, the Snort intrusion detection system already contained 4,867 rules by April 2006, and each rule contains multiple signatures. Addi-

tionally, the ever-changing network environment causes signature sets to constantly change and update, exhibiting dynamic characteristics.

(2) Signatures have duplication phenomena, and each packet may be related to multiple attacks. For instance, in the Snort system, an HTTP packet may have 1,096 attack vulnerabilities.

(3) The location of signatures in packets is uncertain because application formats are complex and variable, making it usually impossible to accurately determine the location of signatures in packets. Therefore, every byte of packet headers and payloads must be inspected at high speed.

(4) Regular expressions can describe more complex patterns than single strings, string sets, or extended strings. Due to their strong descriptive capability and flexibility, they have been widely used in deep packet inspection systems. Famous open-source systems such as Snort and Bro use signature sets in regular expression form. However, there is a gap between regular expressions and the semantics of application protocols, making it difficult to make application semantic-level decisions based solely on regular expression matching results.

3.2.1.2 Signature Fragments and Packet Reordering In deep packet inspection, searching for malicious patterns is generally based on predefined signature sets, using regular expression matching algorithms to perform exact pattern matching on arbitrary positions in packet content. In a network communication session, communication data is split into multiple parts and encapsulated in different packets. The target pattern to be detected often spans multiple packets, so detecting individual packets as units cannot detect the entire pattern.

As shown in Figure 3 [Figure 3: see original paper], detecting individual packets cannot detect the pattern “attack”because it is split across three different packets. For security applications requiring complete detection, such as NIDS (Network Intrusion Detection System), this problem is very serious because attackers often deliberately split attack patterns to evade detection. To discover these pattern fragments, session-level detection must be performed.

Another important issue in deep packet inspection is packet reordering handling and session reassembly [24]. Due to the complexity of network environments, for pattern matching units, the physical arrival order of packets belonging to the same session is often not logically continuous, which is called Out of Order. When subsequent packets are out of order, the pattern matching process will terminate, as shown in Figure 4 [Figure 4: see original paper].

Research on packet reordering problems has been conducted in different network environments. Reference [25] uses the method of sending ICMP (Internet Control Message Protocol) packets and analyzing their responses. The main problem with this method is that different networks treat ICMP traffic differently. Reference [26] uses different methods to study end-to-end TCP (Transmission

Control Protocol) sessions; reference [27] conducts broader analysis using a single capture point in the network to detect out-of-order packets and TCP sessions; reference [28] uses a method similar to [27] to analyze TCP session throughput, focusing on the impact of TCP reassembly windows on throughput and communication performance; reference [29] studies UDP (User Datagram Protocol) session reassembly problems, using flow-based reassembly, with main work focused on internal reassembly mechanisms within flows and related research on burst traffic.

To obtain accurate matching results, subsequent packets need appropriate handling. The general methods are buffering subsequent out-of-order packets or discarding subsequent out-of-order packets.

The method of buffering subsequent out-of-order packets [29, 30] stores out-of-order packets in a buffer, reorders them according to packet sequence numbers, and sends the reordered packets sequentially to the pattern matching unit for detection. When the required next logical packet has not arrived, the detection unit stops and waits for subsequent logically continuous packets to arrive. The buffer space required for buffering subsequent out-of-order packets is [Round Trip Time (RTT)] × [Network Bandwidth]. Assuming RTT = 200ms and Bandwidth = 40Gbps, the required storage space reaches 1GB. In the worst case, buffering high-speed arriving out-of-order packets will cause required storage space to continuously increase. Since network growth rates exceed storage capacity growth, this problem will become more serious in the foreseeable future.

The other method is to discard all subsequent packets with inconsistent logical order [31]. Due to TCP protocol' s delayed retransmission mechanism, the pattern matching unit will eventually obtain logically consistent subsequent packets. The disadvantage of this method is that due to large numbers of packets being discarded, TCP protocol' s flow control mechanism will cause network throughput to drop sharply. In TCP protocol, recovery time caused by packet loss is RTT × number of packet losses. Moreover, if the retransmission timeout (RTO) is exceeded, the congestion window (CWND) will be reset to the minimum.

3.3.1 Modeling

Here we take application protocol classification as an example to model and evaluate the performance of deep packet inspection; other types of deep packet inspection are the same or similar. The structure of the deep packet inspection protocol classification model is shown in Figure 5 [Figure 5: see original paper]. Through observation, each application protocol uses special headers to initialize and control information transmission. The model consists of two parts: a slow path and a fast path. The slow path defines the processing method and specific algorithms used for unclassified sessions (deep packet inspection uses regular expressions), while the fast path associates subsequent packets (belonging to the same session) to the correct protocol based on the slow path' s results [32].

The classifier model includes five processing units: Session ID Extraction Unit,

Session Lookup Unit, Pattern Matching Unit, Session Update Unit, and Session Association Unit. When the protocol classifier runs, if a packet from a session cannot yet be determined to belong to which protocol, the packet is passed to the deep packet inspection classifier. The classifier performs pattern matching on the packet using the protocol signature set. Once a signature matches successfully, the packet (and the entire TCP session) is identified as the corresponding application protocol category, and other subsequent packets belonging to that session skip the signature matching unit. When a new session is successfully identified, a new entry is created in the session table. This entry includes the 5-tuple, application protocol category, and timestamp. The timestamp records the time when the classifier last observed a packet from that session. The main function of the fast path is to update the timestamp of the current session in the session table and delete inactive sessions from the session table to prevent sessions from remaining in the session table after they end. This is particularly common for UDP data streams, where a 10-minute session timeout is generally considered to indicate the session has ended [33].

According to automaton and formal language theory, regular expressions have descriptive capabilities equivalent to automata, so regular expression languages can be recognized by constructing automata corresponding to regular expressions. Automata come in two forms: one is NFA (Nondeterministic Finite Automaton), and the other is DFA (Deterministic Finite Automaton). The difference between NFA and DFA is that for a given state and input, NFA allows multiple successor states, while DFA can only have a unique successor state. NFA and DFA are equivalent in descriptive power.

Figure 6 [Figure 6: see original paper] shows the general process of regular expression matching using automata. First, the regular expression is parsed into a parse tree, which is then converted into an NFA. Currently, common methods for constructing NFAs from parse trees include the Thompson construction method and the Glushkov construction method. After constructing the NFA, it can be used directly for text matching, or it can be determinized and minimized to construct the corresponding DFA, which is then used for text matching. Currently, there are three methods:

(1) NFA-based matching method: As previously introduced, for a given state and input, NFA allows multiple reachable states. Therefore, NFA-based matching methods need to store the current active states. For each new character read, each current active state is checked sequentially to obtain newly activated states; these new states are then added to a new active state set.

Assuming the regular expression length is m, each state can have at most m reachable states, and there can be at most m active states at any time. By using a bit vector method, the state transition time complexity of NFA-based matching methods is O(m). When m is relatively large, matching speed is slower, but its space complexity is only O(m), and a pure NFA matching engine is easy to implement.

(2) DFA-based matching method: DFA-based matching methods utilize the characteristic of DFA: for a given state and input, DFA can only have a unique reachable state. Therefore, the state transition time complexity of this class of methods is O(1), greatly improving matching speed. However, compared with NFA, DFA may cause exponential space growth, with worst-case space complexity of O(2^m) (where m is the regular expression length). In current application requirements, m is often relatively large, which is intolerable.

(3) Hybrid method: To address the slow matching speed of NFA and the large storage space of DFA, researchers have proposed a compromise method [34] that lies between NFA and DFA. The core idea of this method is: first divide the NFA into k modules, then build a DFA for each module separately. The divided NFA is equivalent to having at most m/k (where m is the regular expression size) active states, and each module requires O(2^k) space in the worst case. Therefore, the state transition time complexity of this method is O(m/k), and the space complexity is O(2^k × m/k). One disadvantage of this method is the partitioning of the NFA—deciding how to choose an appropriate k value and which module each state should be assigned to—is relatively difficult. Table 2 compares the complexity of the above three methods.

4 Future Research Directions

Although many research ideas, methods, and algorithms have emerged in recent years to solve the key problems and technical difficulties of deep packet inspection, there are still many issues requiring further in-depth research:

(1) How to further improve processing capability: With increasingly high-speed processing requirements, it can be expected that future deep packet inspection applications will be more hardware-implemented. However, to maintain flexibility, corresponding auxiliary software strategies will also be introduced. Currently, implementations on common hardware using the Snort rule set as the pattern set generally have throughput not exceeding 10G, which may be further improved through improved auxiliary components and optimization strategies.

(2) How to quickly detect encrypted data: Encrypted data generally cannot be directly detected by deep packet inspection. There are already corresponding methods, which involve adding decryption plugins to deep packet inspection components. However, decryption algorithms are time-consuming and greatly impact line-rate detection. How to quickly achieve deep detection of encrypted data is a problem requiring further research.

(3) How to real-time detect attacks without signatures: This paper assumes that signature sets are known in advance. However, in some cases, such as after worm outbreaks, timely dissemination of new signatures for detection is not very easy. Because certain worm outbreaks, such as the Slammer worm devastation, can occupy most of the network bandwidth and cause network congestion when they occur. Therefore, new algorithms are needed to detect

suspicious traffic without signature sets and limit the rate of suspicious traffic before new signatures are obtained.

References

[1] Shenfeng Chen, J.H. Reif, "Fast pattern matching for entropy bounded text," DCC, pp: 282-288, Data Compression Conference (DCC07), 2007
[2] Xu Qian, E Yuepeng, Ge Jingguo, et al. An Efficient Regular Expression Compression Algorithm in Deep Packet Inspection. Journal of Software, Vol.20, No.8, August 2009, pp.214-226
[3] Ye Mingjiang, Cui Yong, Xu Ke, et al. High-Speed Packet Detection Based on Stateful Bloom Filter Engine. Journal of Software, Vol.18, No.1, January 2007, pp:117-126
[4] YU Jianming, XUE Yibo, LI Jun. Memory Efficient String Matching Algorithm for Network Intrusion Management System, TSinghua Science and Technology, 2007, 12(5).
[5] T. Jian-long, L. Yan-bing, L. Ping. Accelerating Multiple String Matching By Using Cache-efficient Strategy. The Ninth International Conference on Web-Age Information Management (WAIM), 2008
[6] A. El-Atawy, T. Samak, E. Al-Shaer, and H. Li. On using online traffic statistical matching for optimizing packet filtering performance. In IEEE INFOCOM' 07, May 2007.
[7] E. Cohen, C. Lund. Packet classification in large ISPs: design and evaluation of decision tree classifiers. In Proceedings of SIGMETRICS, Banff, Canada, 2005.
[8] A. Madhukar and C. Williamson. "A Longitudinal Study of P2P Traffic Classification" . MASCOTS
[9] Ma, Levchenko, Kreibich, Savage, and Voelker. "Unexpected means of protocol inference" . Internet Measurement Conference, 2006.
[10] Xu Kefu, Fang Binxing, Guo Li, Tan Jianlong. Traffic-Aware Frequent Elements Matching Algorithms for Deep Packet Inspection. NSWCTC2010 (Accepted)
[11] Tingwen Liu, Yifu Yang, Yong Sun, Li Guo. Fast and Memory-Efficient Traffic Classification with Deep Packet Inspection in CMP Architecture. (Accepted)
[12] H. Hamed, A. El-Atawy, and E. Al-Shaer. Adaptive statistical optimization techniques for firewall packet filtering. In IEEE INFOCOM' 06, April 2006.
[13] Qunfeng Dong, Suman Banerjee, Jia Wang, and Dheeraj Agrawal. Wire speed packet classification without tcams: a few more registers (and a bit of logic) are enough. SIGMETRICS Perform. Eval. Rev., 35(1):253-264, 2007.
[14] Kencl L, Schwarzer C. Traffic adaptive packet filtering of denial of service attacks. Proceedings of the 2006 International Symposium on world of Wireless, Mobile and Multimedia Networks, Washington, 2006:485-489
[15] Xu Kefu, Guo Li, Tan Jianlong, et al. Traffic-Aware Frequent Elements Matching Algorithms for Deep Packet Inspection. Proc of the IEEE International Conference on Networks, Security, Wireless Communications and

Trusted Computing. Page(s): Vol(2):93-96, 2010.
[16] Tingwen Liu, Yong Sun, Li Guo, "Fast and Memory-Efficient Traffic Classification with Deep Packet Inspection in CMP Architecture" nas, pp.208-217, 2010 Fifth International Conference on Networking, Architecture, and Storage, 2010
[17] PC Lin, YD Lin, YC Lai, TH Lee. Using string matching for deep packet inspection. Computer, 2008 - doi.ieeecomputersociety.org 2008
[18] H Dreger, A Feldmann, M Mai, V Paxson, Dynamic application-layer protocol analysis for network intrusion detection. 15th USENIX Security Symposium Pp. 257-272. 2006.
[19] Vaidehi, V.; Srinivasan, N.; Anand, P.; Balaji, A.P.; Prashanth, V.; Sangeetha, S. A Semantics Based Application Level Intrusion Detection System. International Conference on Signal Processing, Communications and Networking, 2007. ICSCN ' 07. 22-24 Feb. 2007 Page(s):338-343
[20] www.lsi.com/documentation/working/tarari_content_processors/TarariRAX Whitepaper.pdf
[21] A. Anitha and V. Vaidehi. Context based Application Level Intrusion Detection System. International conference on Networking and Services, 2006. ICNS ' 06. 16-18 July 2006 Page(s):16-
[22] Alessandro Finamore, Marco Mellia, Michela Meo, et al. KISS: Stochastic Packet Inspection Classifier for UDP Traffic. IEEE/ACM TRANSACTIONS ON NETWORKING, VOL.18(5), pp:1505-1515, 2010
[23] Tang Yong, Lu Xicheng, Wang Yongjun. Survey on Attack Signature Automatic Extraction Technology. Journal on Communications, Vol.3(2), pp:96-105, 2009
[24] V. Paxson, "End-to-end internet packet dynamics" in IEEE/ACCM Transactions on Networking, 1999, pp. 277-292.
[25] S. Jaiswal, G. Iannaccone, C. Diot, J. Kurose, and D. Towsley, "Measurement and Classification of Out-of-Sequence Packet in a Tier-1 IP Backbone" in Internet Measurements Workshop (IMW), 2002.
[26] M. Laor and L. Gendel, "The effect of packet reordering in a backbone link on application throughput" in IEEE Network, 2002.
[27] X. Zhou and P. V. Mieghem, "Reordering of IP Packets in Internet" in C. Barakat, I. Pratt (Eds.): Fifth annual Passive and Active Measurement Workshop PAM2004, LNCS 3015, pp. 207-218, 2005. Springer-Verlag Berlin Heidelberg 2004.
[28] M. Necker, D. Contis, D. Schimmel, "Tcp-stream reassembly and state tracking in hardware" in Proc. of 10th Annual IEEE Symp. on Field-Programmable Custom Computing Machines (FCCM' 02), September 2002, pp. 286-287.
[29] S. Li, J. Tørresen, and O. Sorasen, "Exploiting Stateful Inspection of Network Security in Reconfigurable Hardware" in Proc. of 13th Int. Conf. on Field Programmable Logic and Applications (FPL ' 03), September 2003, pp. 1153-1157.
[30] D. V. Schuehler, J. Moscola, J. Lockwood, "Architecture for Hardware Based TCP/IP Content Scanning System" in Proc. of 11th IEEE Symp. on

High Performance Interconnects, August 2003, pp:89-94.
[31] M. Fish and G. Varghese, "Fast Content-Based Packet Handling for Intrusion Detection" UCSD technical report CS2001-0670
[32] Niccol Cascarano, Alice Este, Francesco Gringoli, et al. An Experimental Evaluation of the Computational Cost of a DPI Traffic Classifier. Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE, Honolulu, HI, page(s): 1-8
[33] N. Brownlee, Traffic flow measurement: Meter MIB, Request for Comments RFC 2064, Internet Engineering Task Force, January 1997.
[34] Xu Qian, E Yuepeng, Ge Jingguo, et al. An Efficient Regular Expression Compression Algorithm in DPI. Journal of Software, Vol.20, No.8, August 2009, pp.214-226

Author Biographies

Xu Kefu: Assistant Researcher at Institute of Computing Technology, Chinese Academy of Sciences. xukefu@software.ict.ac.cn
Li Yang: Ph.D. Candidate at Institute of Computing Technology, Chinese Academy of Sciences
Tan Jianlong: Associate Researcher at Institute of Computing Technology, Chinese Academy of Sciences
Guo Li: Director of Information Security Research Center, Institute of Computing Technology, Chinese Academy of Sciences; Senior Engineer at Professor Level

Note: Figure translations are in progress. See original paper for figures.