Performance-driven Development of Deep Packet Inspection Systems on Commodity Platforms

Thiago Lacerda[1], Stênio Fernandes[1,2,3], Ana Cristina Oliveira[1], Djamel Sadok[1], Judith Kelner[1]

[1]Grupo de Pesquisa em Redes e Telecomunicações (GPRT) – Centro de Informática (CIn) – Universidade Federal de Pernambuco (UFPE)
Caixa Postal 7851 – 50.732-970 – Recife – PE – Brazil.

[2]Instituto Federal de Educação Ciência e Tecnologia de Alagoas (IF-AL), Maceió, Brazil

[3]School of Information Technology and Engineering (SITE) – University of Ottawa Ottawa – Canada.

{thiagobl, stenio, ana, jamel, jk}@gprt.ufpe.br

Abstract. *Deep Packet Inspection (DPI) systems have been increasingly performed on dedicated hardware, as an attempt to speed up the packet processing for high speed links. This is mainly caused by the current demand for CPU-intensive processing required by regular expression functions, which investigate the packet payload trying to match patterns of application signatures. This study proposes and evaluates techniques to optimize DPI systems using commodity hardware. At first, it is designed a new optimized software architecture. In the following, this architecture is implemented into a DPI software and those optimization techniques are then integrated. Our results show that the time spent with regular expression matching was actually improved, besides the packet loss rate when realizing online measurements meanwhile. The evaluation results state that the performance of a typical DPI process on a Linux box can be improved in almost 100%, and the amount of classified traffic may be increased 220%.*

1. Introduction

Internet Service Providers (ISP) and network administrators always had deep interest in knowing what type of traffic is traversing their network backbones. Therefore, they need to perform continuously network monitoring and traffic analysis. Such tasks are very important to provide overall information about the network status, such as network problems, protocols and applications that are being used and other information about the network infrastructure. Additionally, this monitoring must be very precise, since erroneous assumptions about the network can lead to undesirable operating cost.

In order to have updated knowledge about network traffic profile, ISPs and network administrators are commonly relying on Online Traffic Classification (OTC). OTC is a great aid to traffic management, since the ISPs can take management decisions in real time, according to the traffic that is traversing the backbone. For example, they can decrease or increase the Quality of Service (QoS) of some applications or even

block anomalous flows. In the past, OTC was exclusively relying on the well-known port-based application identification approach, where Internet applications had well-known port numbers to send and receive their packets. However, with the behavior dynamics of the Internet, mainly caused by peer-to-peer (P2P) applications [Karagiannis et al. 2004], this type of classification is not accurate anymore. Morevover, classification based on the well-known ports is not efficient to detect malicious flows and attacks over the Internet either.

Nowadays, due to the dynamic behavior of the Internet, OTC is relying on Deep Packet Inspection (DPI), which is the monitoring approach that provides the most detailed information about the packets that are traversing the network. DPI systems capture packets and perform some kind of matching between the packet payload and an application signature, commonly represented by Regular Expressions (RegEx). DPI systems are considered one of the most precise and maintainable packet analysis techniques, since RegExes can fully describe various protocol messages and can be modified or added to the system in a quite simple fashion, in order to fix a broken pattern, or recognize new applications.

On the other hand, the RegEx matching process performed by DPI systems is the most expensive task on such applications, consuming over 90% of the processing time when considering the processing of the system as a whole [Yu et al. 2006]. This high processing time forces, in counterpart, DPI systems to use specialized hardware and software solutions, striving to obtain good performance on high speed links. In general, hardware and software solutions to support DPI are expensive, since those solutions use dedicated high-end technology components to speedup performance and effectiveness as much as possible. Consequently, those systems are not financially reasonable to medium or small Internet Service Providers (ISP).

Deploying DPI systems capable to deal with high speed links in commodity hardware and software is still an open challenge, either for the industry or the academic community. Such challenges are mainly caused by the huge and rapid growth of the network link speeds. It is difficult to deploy DPI systems, performing OTC without packet loss, while maintaining an acceptable classification completeness.

This paper addresses the previous issues by proposing some architectural and software layer modifications that lead to considerable performance gains, using commodity hardware and software. With those gains, a system that was not able to deal even with 100Mbit/s without discarding packets in the path through the network interface, passing by the kernel space to the user space level, is now able to attend a throughput of 900Mbit/s with nearly 0%of packet loss, and surprisingly increasing the classification completeness. In order to verify the augmented performance, a traditional DPI system that runs on a commodity platform, i.e., Linux OS and Intel architecture was taken as a baseline.

The DPI system used as baseline has some basic features, namely sequential processing, libpcap[1] library for packet capture and a set of rules or signatures. Some optimizations are proposed and evaluated within this DPI system at several levels, in order to obtain the best performance trade-off without specialized hardware solutions. The contribution of this paper is many-fold: first an architectural level optimization is

[1] www.tcpdump.org

proposed to take advantage of all available resources of the operating system. Second, a slight modification on how packets are forwarded, at kernel level to user space is implemented. Third, this paper applies and evaluates two other techniques, presented at [Fernandes et al. 2008], that lead to a considerable improvement to this DPI system, namely packet counting (PC) and payload truncating (PT). Finally this paper also shows how subtle modifications on some RegEx (Regular Expression) signatures can increase the processing speed, by reducing the size of the generated automaton, and decreasing the packet loss rate.

The remainder of this paper is organized as follows. Section 2 presents the essential background needed for the paper understanding. Section 3 presents proposed optimizations and the methodology that is followed by the evaluations. Then, Section 4 shows the results obtained with the new architecture and proposed approaches. Section 5 presents the related work. Finally, Section 6 discusses some observed points and Section 7 points out some concluding remarks and future works.

2. Technical Background

2.1.Regular Expressions

Regular Expression (RegEx) is a set of strings, which represent a pattern used to match a certain string of characters. They provide an enormous expressiveness without necessity to express the complete desired pattern, so a RegEx can fully represent a complete protocol communication. They are also used in computer science.

When RegExes apply a formal description language checking the source code in a programming language and compiling it, with a variety of writing patterns. They are commonly represented as automata, which can be Deterministic Finite Automata (DFA), Non-Deterministic Finite Automata (NFA) and Extended Finite Automata (XFA). These automata can consume a great bunch of memory if with complex RegExes also, consuming a huge amount of time to report a successful match. Hence, one of the features that is directly correlated with the matching time is the number of greed operators used to represent one RegEx, e.g.* and +, used to match character chains,. For instance, if a pattern like **r.*e** is going to search for patterns in the string "**regular expressions**", the reported match would be "**regular expre**" even if "**re**" would be sufficient. This additional searching time may not play a big difference in applications that do not require results in real time or that even requiring it, does not have a high income rate. However, when those kind of matching are performed in real time systems, or those that deal with high traffic rates, for instance, this additional time can be crucial. Therefore, RegExes must be carefully writtenin order to have acceptable searching times.

2.2.Traffic Analyzer Module (TAM)

The DPI system used as the baseline of this paper, namely Traffic Analyzer Module (TAM), uses a set of RegExes signatures. The signature database accomplishes more than 60 applications, falling into 9 different application classes. TAM was developed using the C language, using the libpcap library for packet capture, and the C library *regex* for RegEx matching.

TAM is composed of four modules: 1) the capture module, which is responsible for capturing packets from an Ethernet device; 2) the aggregation module, which aggregates the packets in to flows and stores them in a hash table for further lookup; 3) the classification module, which compares the packet payload (only packets that belongs to unclassified flows) with the RegExes signatures, aiming to classify the network traffic flows, and 4) the cleanup module, which is responsible to go through every entry in the hash table and verify whether or not the flows have expired removing the timed out ones.

TAM has two basic/main threads, which separate the flow packet analysis from the hash table cleanup. The cleanup module runs on a thread separate from the other three modules. This thread is activated on pre-defined time intervals, i.e., 5 minutes. It is important to remark that when it runs, it locks the hash table that is accessed and updated by the other three modules. Thus, the capture module has to stop dequeueing packets from the capture buffer, resulting in undesirable packet losses.

TAM was developed at the Networking and Telecommunication Research Group (GPRT)[2], from the Federal University of Pernambuco (UFPE). The main objective of TAM is to capture all the packets that traverse the network in order to classify them on the fly. To achieve this goal, it aggregates the packets into flows with the lowest packet loss rate in case of online classification. It was initially conceived to run in a 34Mbit/s backbone, at the Point of Presence of Pernambuco (PoP-PE) with a direct link to the Point of Presence of Rio de Janeiro (PoP-RJ), working with no loss of packets at such speed.

However, the requirements have changed. Those requirements have increased to accomplish higher speeds of network links, in this case of study reaching 1Gbit/s. At this speed, the TAM tool must capture, aggregate and classify packets maintaining as few as packet losses as possible without significant decrease in the classification completeness. Although it seems that the use of TAM as a baseline may limit the generality of our proposal, it is worth emphasizing that the architecture of TAM and its performance is much similar to other commodity platform-based DPI systems (e.g. Snort[3]). Therefore, most of the appointments and conclusions made throughout this paper can be extended to such similar DPI systems.

3. Proposed Optimizations

3.1.New Architecture

In this paper, it is proposed and implemented a novel multi-threaded architecture to improve the performance of DPI systems. Each thread, which runs in parallel, will act as an independent packet inspection component, grouping up the functions of the first three modules described at Section 2, namely capture, aggregation and classification modules. Each worker thread has its own capture buffer and an ID number (starting from 0 for the first thread). These several packet buffers are filled up using the packet capture library PF_RING [Deri 2004]. This new architecture additionally has he cleanup module in a separated thread.

[2] http://www.gprt.ufpe.br
[3] http://www.snort.org/

This new architecture presents problems related to operation synchronization. It has to deal with two different levels of synchronization: 1) each thread must be locked when the cleanup module starts running, and 2) since only one thread would be working at this time, the other need to wait until it finishes, in order to avoid the access to a flow that is being cleaned up from the hash table, for instance.

One manner to create one packet buffer per worker thread is to replicate in all buffers the packets received, oncethere is no entity responsible to distribute the incoming packets among the threads. In this sense, a mechanism to detect which packets are already being analyzed by another worker thread worth it to be implemented, consequently adding some overhead. Also, even applying such mechanism, if a buffer is locked by one thread that is analyzing the packets, it may overflow, and packets will be dropped.

Those problems could be eliminated if the threads were independent, without the necessity to share information among each other. This could be achieved if packets from the same flow were treated by the same thread, that is, if different threads were not analyzing packets of the same flow. Hence, no thread synchronization would be mandatory, since each thread would have its own set of flows, leading to a complete parallel execution. Moreover, each thread would have a private hash table, decreasing the time spent to search for a flow.

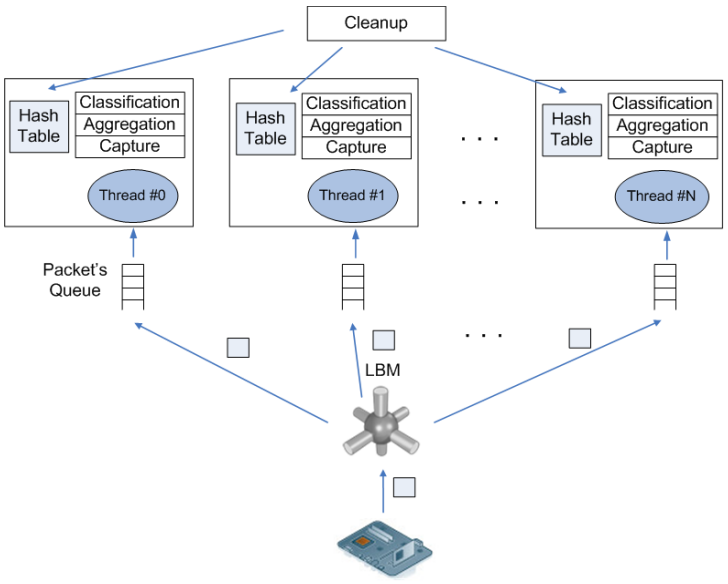

Figure 3.1 – New Architecture

On this purpose, a *Load Balancing Module* (LBM) was integrated within the architecture. The LBM was implemented as a Linux kernel module. This module distributes the packet load among the worker threads in a way that different threads will not receive packets of the same flow, avoiding many synchronization problems. This module intercepts all packets that arrive at the NIC before their entrance on the capture buffers, assigning the packets to the thread buffer that they belong to. Since each thread has its own set of flows, there is no need to share the flow information between threads, therefore, it avoids synchronization costs. This new architecture is shown in Figure 3.1.

The key point of the LBM is a simple and effective hashing function that is calculated whenever a packet arrives at the NIC. Such function is described in Equation 1. With this simple hashing function, LBM can guarantee that a given flow, and its reverse, will be always forwarded to the same thread.

$$threadID = (IPSrc + IPDst + layer4SrcPort + layer4DstPort + layer3Protocol) \ \mathbf{mod} \ numberOfThreads$$

Equation 1 – Hashing Function

As mentioned in Section 2.2, most commodity platform-based DPI systems, like TAM software, have unacceptable packet losses, because of the lock of the classifier thread at the flow cleanup moment. Since this novel proposal has a private hash table per thread, then it can obtain an important performance gain with a unique cleanup module, which is depicted at Figure 3.2. In this example, the cleanup process is activated every 5 minutes. The cleanup module will only update one hash table a time, thus, the system still has N-1 threads working in parallel. Therefore, the entire system would not be locked out, increasing the packet capture rate.

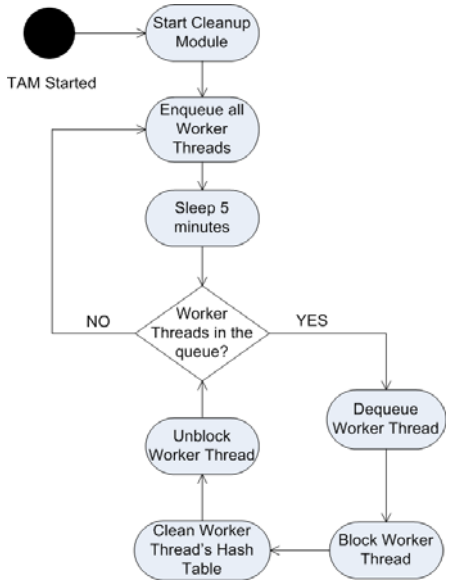

Figure 3.2 – Unique Cleanup Module

RegEx matching performance is directly dependant on the size of the generated automaton that represents the expression. Thus, if a RegEx is full of wildcards (*), the associated automaton size can become enormous, dramatically decreasing the RegEx matching time. In order to reduce such latency, some RegEx signatures are going to be rewritten, reducing the size of the generated automaton without loss of precision.

The final optimizations to be incorporated at this architecture were proposed by Fernandes et al. [Fernandes et al. 2008], namely PC and PT. It is then expected a reduction in the packet loss rate when the DPI system is handling with high throughputs by analyzing: (i) only the first 7 packets of a given flow; and (ii) the first 750 of the packet payload. Those thresholds were set for PC and PT based on the results of the study presented in [Fernandes et al. 2008]. Afterwards, PC and PT techniques are going to be part of the rewritten patterns of the new architecture.

3.2.Methodology

In order to evaluate the packet capture rate when the system is submitted to different traffic generation loads, an experimental testbed was built, according to what is described in Figure 3.3. The testbed comprises four machines and one switch, each one playing the following roles: 1) a *measurement machine* (M) that will receive the generated traffic; 2) three *traffic generators machines* (S1, S2 and S3), also called *slave machines*, which replay a previously captured packet trace and send it through the machine NIC, using tcpreplay[4]; and 3) a 1Gbps *switch*, which in charge of aggregating all traffic and forward it to the measurement machine. The full machines' configuration of the built testbed, used for all tests, is described at Table 3.1.

Table 3.1 – Testbed Configuration

Machine	Processor	RAM Memory	Administrative NIC	Traffic Generator/Receiver NIC	HD	Operating System
M	Intel Xeon X3210 Quad-core	4GB DDR	Onboard Gigabit	Offboard, 3Com Gigabit	3x 500GB Sata HDs	Linux, 2.6
S1	Intel Xeon 5110 Dual-core	2GB DDR	Onboard Gigabit	Offboard, 3Com Gigabit	1x 250GB Sata HD	Linux, 2.6
S2	AMD Athlon 64x2 Dual-core	1GB DDR	Offboard 10/100Mbit	Onboard, nVidia Gigabit	1x 300GB Sata HD	Linux, 2.6
S3	AMD Athlon 64x2 Dual-core	1GB DDR	Onboard Gigabit	Offboard , Intel Gigabit	1x 300GB Sata HD	Linux, 2.6

It is worth remarking that all experiments were performed using real packet-level traces collected at one of the largest commercial ISP in Brazil. The passive probe used to collect the traffic was listening to a router port that was mirrored, in order to avoid interference in the regular traffic transit, which consists of traffic from/to around 50.000 ADSL subscribers. To make the collected data more representative in terms of traffic diversity, the network was sniffed for several days, accumulating almost 6TB of real Internet traffic in different periods of the day. A representative sample of this collection was selected to be replayed to the measurement machine at these experiments.

The following metrics are considered in the experiments:

1. *Packet Loss Rate*

Loss of packets is one of major concerns when applying DPI at the wire rate, especially when using commodity hardware. It also impacts the classification completeness. The rate of the traffic generated by the slave machines was varied, in order to evaluate how much traffic each DPI version can handle. The starting point was 100Mbit/s, after that the rate was increased by a factor of 100, up to 900Mbit/s.

2. *Classification Completeness*

Classification completeness is the percentage of volume or flows that is classified by the system. It is important to develop a DPI system that has no packet loss, besides providing a good level of with completeness.

[4] http://tcpreplay.synfin.net/trac/

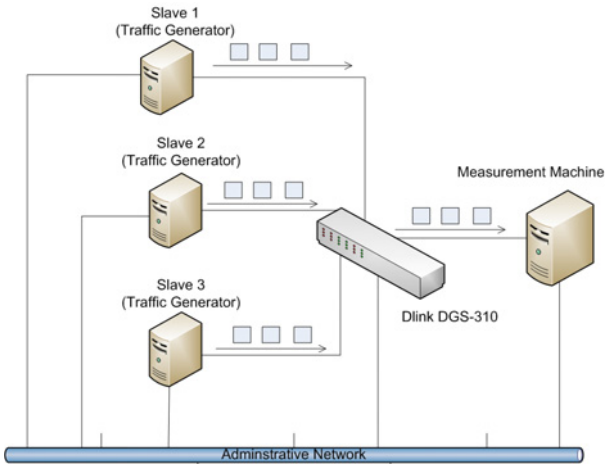

Figure 3.3 – Testbed Architecture

4. Evaluation and Results

At first, it is shown in Figure 4.1 (a) how the DPI baseline system performs on the lowest packet incoming rate (100Mbit/s). Each point of the line corresponds to the percentage of the packets that were lost after received at the NIC in the period of one second. Additionally, Figure 4.1 (b) contains the results obtained when the novel architecture proposed in this work is evaluated at the same rate, 100Mbit/s. As shown in the figure, even with a lower packet incoming rate, the original DPI system has severe losses of packets, reaching on average almost 60%. It is also important to remark that at time that the cleanup thread is triggered, i.e. in intervals of 5 minutes, the system loses 100% of the packets received.

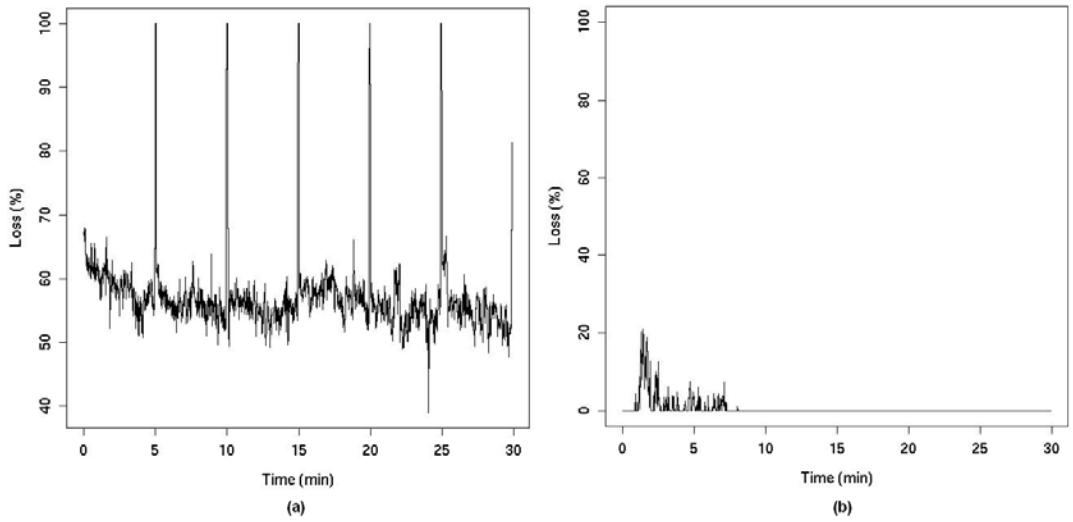

Figure 4.1 – Packet Loss, Original DPI (a) vs. New Architecture (b) (100Mbit/s)

The New Architecture has a new packet capture mechanism, switching from the libpcap to a fine-tuned PF_RING. From now on, the shared circular buffer between kernel and user space will avoid unnecessary packet copies between them, therefore it will increase the system throughput.

Figure 4.1 (b) shows that, at 100Mbps the New Architecture outperforms the original one and has some losses at the beginning of the measurement. The mains reason for this behavior is that when the DPI is initiated, almost every packet that arrives could represent a new flow. Therefore, the DPI will try to classify almost every packet, which can lead to such loss peaks. However, as long as those flows are classified by the DPI, the arrival rate of new flows starts reducing and it has minimal packets losses.

Figure 4.2 shows that the New Architecture dramatically improves the packet capture rate up to 300Mbit/s. However, with incoming rates greater than 300Mbit/s, the new DPI .still faces severe losses.

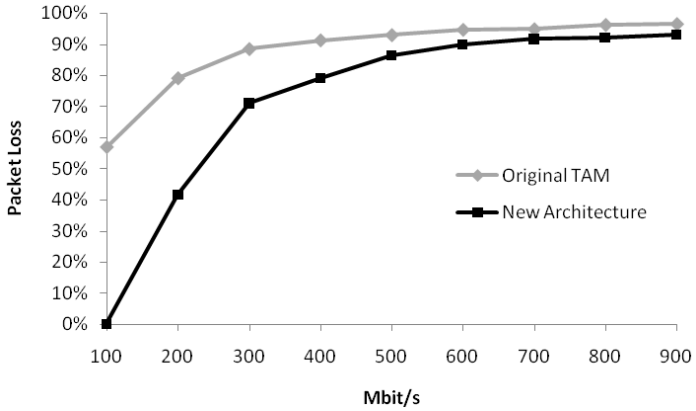

Figure 4.2 – Packet Loss, original DPI vs. New Architecture

One of the key proposed optimization technique was the new multithread architecture, which distributes the capture and analysis tasks among several threads. Please note that only deploying this multithread feature would not be enough to minimize losses, without the help of LBM. By providing a load balancing component, the new DPI can eliminate synchronization problems, by making sure that there will exist a N:1 relation between flows and threads. In other words, the worker threads will effectively work in parallel. Additionally, the single cleanup module can ensure that, even when it starts its operation, there will be, at least, N-1 threads working.

4.1.Packet Counting (PC) and Payload Truncating (PT)

By inspecting only the first 7 packets of a given flow (PC feature), we expect that packet losses will be minimized even at a incoming rate beyond 300Mbps. Although, it has an impact on the DPI accuracy, such approach provides good completeness levels [Fernandes et al. 2008]. Then, another improvement in the new DPI was integrated, namely the use of the PT technique, which takes into account only the first 750 bytes of the packet's payload. The performance evaluation results are presented at Figure 4.3.

The results show that the packet loss rate can be dramatically reduced when the RegEx matching is limited to the inspection of the first 7 packets of a given flow. At 900Mbps, the New Architecture presented only 38.58% of packet loss rate, against 96.57% of the original DPI.

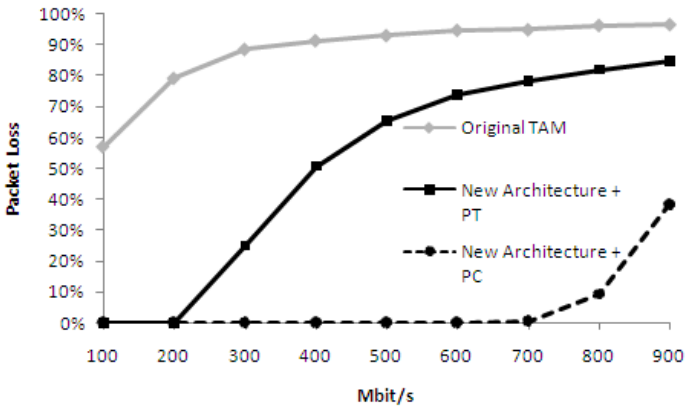

Figure 4.3 – Packet Loss: original DPI vs. New Architecture with PC and PT

The packet loss rate had a considerable reduction when PT technique was applied, and the results obtained with the PC technique are remarkably better. Both of these improvements do not present packet losses until an income rate of 200Mbps, against 56.94% and 79.11% for the original DPI system, in a rate of 100Mbps and 200Mbps respectively. The dropped packets at the highest rate income rate, 900Mbit/s, have suffered a slightly reduction for the original DPI technique from 96.57% to 84.72%.

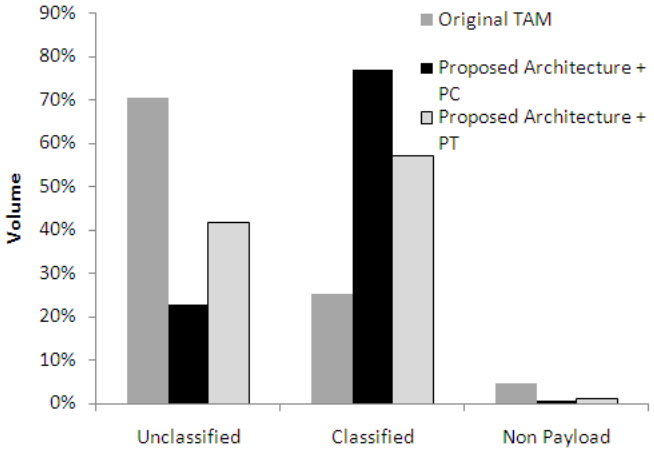

Figure 4.4 – Classification Completeness

The original DPI system analyzes all packets from a given flow, although the classification completeness is approximately 25% in terms of byte volume, as shown in Figure 4.4. . Such a high level of unclassified traffic is justified because of its high packet loss rate, which is 96.57% at 900Mbit/s. Those dropped packets carry the signatures of applications in their payload that are recognized by TAM. The release that analyses only the first 7 packets could classify around 77% of the volume share On the other hand, this release captures packets from more flows with a better regularity, besides, there is also a high probability of the signature be located at the first packets too. The same behavior on the classification completeness occurs for the PT technique. The amount of volume that was classified when inspecting only the first 750 bytes of the packet payload is approximately 57%, which is greater than twice as much as the classification completeness of the original DPI. In the cases that PC and PT techniques were used, a smaller amount of data being analyzed contributed to better classification

completeness when the packet income rate was greater than the rate that the system can actually deal with.

Table 4.1 – Analyzed traffic statistics (700 Mbit/s)

Version	Number of Captured Flows	Captured Volume (GB)
Analyzing all packets	3,222,939	6.56
Analyzing first 7 packets	13,149,126	136.86

The Table 4.1 summarizes a comparison of two TAM releases in terms of the number of flows and the amount of volume (GB) that could be actually captured at 700Mbit/s speed. The releases are the one that represents the original DPI system and the other implements the PC optimization. It shows how the number of flows and the byte volume captured by the New Architecture, analyzing only the first 7 packets, is much greater than what is captured by the original DPI system. In terms of flows, it is 4 times greater and about 21 times as much as volume.

4.2.Evaluation with Rewritten Patterns

The main bottleneck of a common DPI is the RegEx matching operation. In order to improve performance some patterns were rewritten, mostly related to use of unnecessary greedy quantifiers like * and +. Such modification on those patterns have reduced its generated automata size and consequently reduced its searching time.

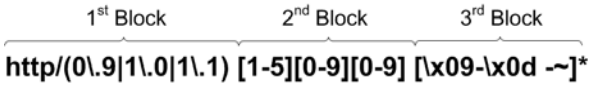

Figure 4.5 – HTTP RegEx

For instance, a common pattern that recognizes the HTTP protocol is described at Figure 4.5. This pattern has a greedy quantifier at its end, which denotes zero or more repetitions of any characters in the range specified inside the brackets (3rd Block). This quantifier is completely unnecessary, since that a payload to successfully match this RegEx only needs to have the first two blocks of the expression, making no difference whether there are more characters or not after these blocks. This indifference is caused by the quantifier *****, denoting zero or more repetitions. Thus, removing the final block of this expression can lead to a considerable gain in the matching speed, since the greedy quantifier will not be present and spending unnecessary time, looking for more characters. Therefore, the 3rd block of this RegEx was removed, resulting in a new pattern. Several RegExes for a number of protocols were revised, although we do not show here due to lack of space.

Figure 4.6 shows that the packet loss rate was reduced from 56.94% to 0%, in 100Mbit/s, and from 96.57% to 90.32%, in 900Mbit/s. As expected, the classification completeness increased since more packets were captured at the user space. Figure 4.7 shows that the classified volume has increased from 25%, in the original DPI, to 51.78%, in the new architecture with rewritten patterns.

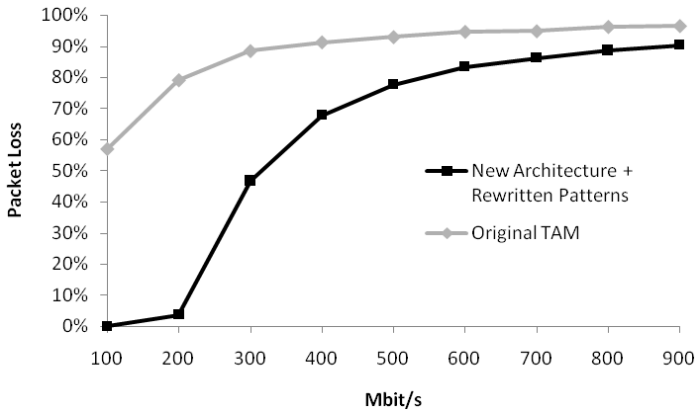

Figure 4.6 – Packet Loss, original DPI vs. New Architecture (Rewritten RegEx)

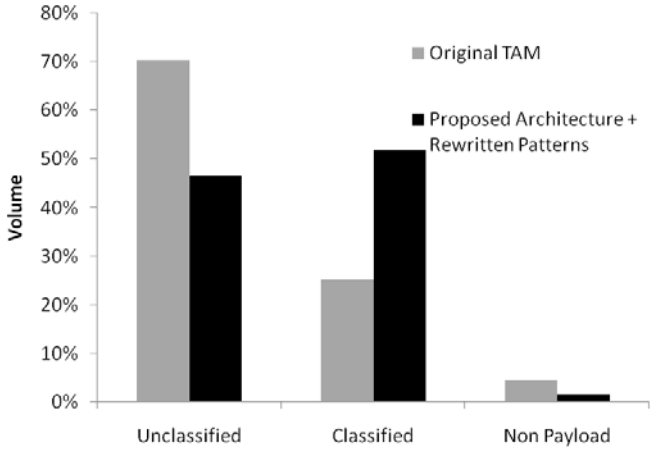

Figure 4.7 – Classification Completeness, Rewritten Patterns (Bytes)

4.2.1.All Techniques Together

In order to obtain the highest packet capture rate, the techniques mentioned on this paper were combined with the New Architecture, leading to a high performance DPI.

Figure 4.8 shows the loss rate of the new DPI with all techniques combined. It starts to drop packets only at 800Mbit/s. At 900Mbit/s the loss rate is around 0.15%.

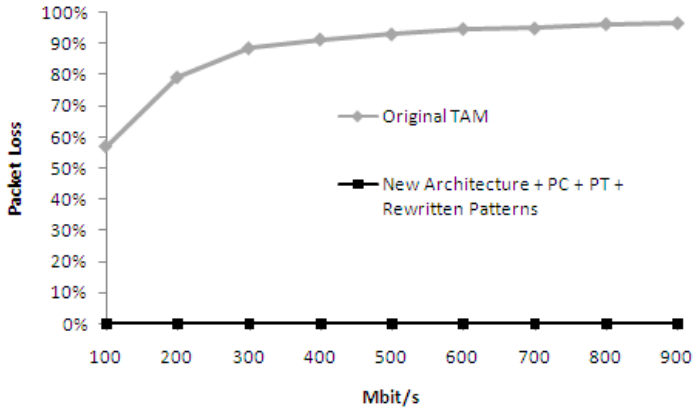

Figure 4.8 – Packet Loss, Original DPI vs. New Architecture (all techniques)

Additionally, the classification completeness reached the greatest classified volume. The Figure 4.9 shows that the classified byte volume has increased from 25%, with original DPI, to almost 80%, with the New Architecture and mixed techniques, thus representing a gain of 220%.

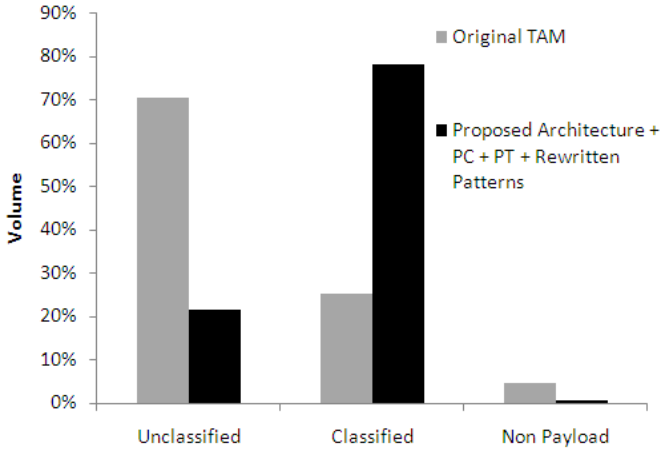

Figure 4.9 – Classification Completeness, Mixed Techniques (Bytes)

5. Related Work

There are several works in the scientific community that bring up a great contribution to the achievement of higher DPI performance and accuracy for traffic analysis and classification. Deri [Deri 2004] proposes a new type of kernel socket, called PF_RING, which is able to improve drastically the packet capture rate on Linux-based systems. Using PF_RING sockets, one packet does not have to go through the entire delivery protocol stack from the NIC to the user application, but it takes a straight path from kernel to user space. This new socket is based on a circular buffer, which is shared between kernel and user spaces, and handles incoming packets. His technique has outperformed other known packet capture approaches. He also extended his previous work with PF_RING and proposed nCap [Deri 2005]. It is seen as a further improvement to the packet capture rate in Linux-based systems. The author created a solution for packet capture and transmission at wire speed. His evaluation showed that the proposed approach has better performance than PF_RING. However, as nCap is at the driver layer, it is bound to a specific kind of NIC, the Intel GE 32-bit.

Bernaille et al. [Bernaille et al. 2006] have studied TCP flow classification with low CPU power consumption. His approach consisted of analyzing the size of the first five packets of a TCP flow, achieving a high packet capture rate since RegEx matching is not performed. Although, that approach has several drawbacks including: 1) it does not perform well for short flows (with less than five packets); 2) it needs to track the reverse flow, and sometimes the reverse flow does not follow the same path, thus, it is not suitable for measurements at ISP routers, for instance; 3) the packet size is not a precise classification criterion as different applications can have the same packet size distribution; and 4) it is not applicable to UDP flows.

Sen et al. in [Sen et al. 2004] have done a study on peer-to-peer (P2P) application classification, using signature matching and performing analysis on five widely known P2P protocols. They used fixed string signatures of those protocols,

aiming to evaluate the accuracy of those signatures. They were able to show that the packet examination is only needed on the first packets of a given flow, i.e. less than 10 packets, which led to less than 5% of false positive results in some protocols.

Yu et al. [Yu et al. 2006] proposed a fast and memory-efficient solution to RegEx matching. They have analyzed the computational and storage cost of building individual DFAs for matching RegEx. Thus, based on such analysis, they proposed two rewriting rules that can dramatically reduce the size of the resulting DFAs, reaching 99% of reduction. Following, techniques to combine multiple DFAs into a small number of groups were developed, improving the matching speed.

In [Kumar et al. 2006], the authors introduced a new representation for RegEx, called Delayed Input DFA (D^2FA). This modification in the original DFA substantially reduces its space requirements. The authors explored the outgoing transitions shared by sets of states and introduced a special kind of transitions called default transitions. It showed that even reducing the number of edges of a given automaton, the same set of patterns could be recognized. That approach reduced the number of edges by more than 95%.

The authors Fernandes et al. [Fernandes et al. 2008] have proposed techniques to obtain considerable performance gains in DPI systems. Their techniques consisted in analyzying only the first packets of a given flow and truncating the packet payload. They reported that if analyzing only the first 7 packets, it would be sufficient to successfully classify a flow, hence, reducing the processing time in almost 80%. Additionally, they have shown that analyzing only first 750 bytes of the packet payload can lead to a performance and accuracy trade-off.

In [Smith et al. 2008], the authors show a solution that addresses the known problem of state explosion related to automata, which are commonly used to represent RegExes. They propose techniques aiming to combine XFAs, eliminate ambiguity and optimize the memory usage and performance of those automata. They performed tests on Snort and on Cisco Systems, in order to validate their techniques achieving impressive results.

The authors in [Po-Ching Lin et al.] present the importance of string matching for traffic identification and analysis, also citing some approaches on string matching algorithms development. They cite the Automaton-based (using DFAs and NFAs), Heuristic-based (that can make shifts while searching for a string in a payload, in order to avoid unnecessary comparisons) and the Filtering-based approaches (that makes a pre-filtering of the payload, to exclude patterns that definitely do not match). They discussed the pros and cons of those techniques.

6. Discussion

In the previous sections, some optimizations were discussed, deployed and evaluated. One of the key points was to confirm that problem solving always starts with a carefully architecture and design planning. Those points were reflected on the combination of the multi-threaded approach and LBM. Hence, without those preliminary optimizations, none of the presented gains would be possible to achieve in the original DPI.

However, PC and PT optimizations have to be carefully used. For instance, PT technique can change the traffic profile of the analyzed network. Chances are that

signatures may perform a successful match, even against the correct one that would match at the truncated payload block. Additionally, if PC is performed with a small threshold for the number of packets, a significant amount of traffic can become unclassified, since some application protocols do not write their signatures in the first packets of a flow.

7. Concluding Remarks

This paper has shown how system developers can fine-tune DPI systems, by implementing careful modifications that will lead to considerable performance gains. By doing so, a DPI system that is not able to cope with high packet incoming rates (e.g. at 1Gbit/s) can handle such traffic load with no decrease in classification completeness.

Those mentioned performance gains are summarized in the steps made in each optimization phase, thus resulting in a gain of almost 100%, reducing from 96.57% of packet losses, in the Original DPI at 1Gbit/s, to 0.15%, in the New Architecture with all optimizations combined. Additionally, this paper has shown how DPI systems can take a great advantage when analyzing the first packets of a given flow, and truncated payload, with considerable gain in classification completeness, around 220%.

In Section 4.2, the results have shown that RegEx rewriting is a valuable technique that can lead to considerable performance, without a huge effort. However, only few RegExes were rewritten, those that presented the most critical problems, in a groupof about 40 RegExes. Due to this reason, we can envisage as a future work the evaluation of other RegExes, the identification and revision of challenging signatures can lead to additional improvements. Additionally, there are other RegEx libraries that can be evaluated and compared with the one used in the original DPI.

References

Bernaille, L., Teixeira, R., Akodkenou, I., Soule, A., and Salamatian, K. 2006. “Traffic classification on the fly,” In: SIGCOMM Comput. Commun. Rev. 36, 2 (Apr. 2006), 23-26.

Deri, L., “Improving Passive Packet Capture: Beyond Device Polling,” In: Proceedings of SANE 2004, 2004.

Deri, L., “nCap: wire-speed packet capture and transmission,” In: Proceedings of the End-to-End Monitoring Techniques and Services. Workshop, p.47-55, May 15-April 30, 2005.

Fernandes, S., Antonello, R., Lacerda, T., Santos, A., Westholm, T. and Sadok, D., “Performance Optimization for Deep Packet Inspection Systems,” In: Proceedings of the 12th IEEE Global Internet Symposium 2009.

Karagiannis, T., Broido, A., Brownlee, N., claffy, kc, and Faloutsos, M., “Is P2P dying or just hiding?,” In: IEEE Globecom 2004 - Global Internet and Next Generation Networks, 2004.

Kumar, S., Dharmapurikar, S., Yu, F., Crowley, P., and Turner, J. 2006. “Algorithms to accelerate multiple regular expressions matching for deep packet inspection,” In: Proceedings of the 2006 Conference on Applications, Technologies, Architectures,

and Protocols For Computer Communications (Pisa, Italy, September 11 - 15, 2006). SIGCOMM '06. ACM, New York, NY, 339-350.

Po-Ching Lin, Ying-Dar Lin, Yuan-Cheng Lai, Tsern-Huei Lee, "Using String Matching for Deep Packet Inspection," In: Computer, vol. 41, no. 4, pp. 23-28, Apr., 2008.

Sen, S., Spatscheck, O. and Wang, D. “Accurate, scalable in-network identification of p2p traffic using application signatures,” In: Proceedings of the 13th international Conference on World Wide Web. WWW '04. ACM, New York, NY, 512-521.

Smith, R., Estan, C., Jha, S., and Kong, S. “Deflating the big bang: fast and scalable deep packet inspection with extended finite automata,” In: *SIGCOMM Comput. Commun. Rev.* 38, 4 (Oct. 2008), 207-218.

Yu, F., Chen, Z., Diao, Y., Lakshman, T. V. and Katz, R. H., “Fast and memory-efficient regular expression matching for deep packet inspection,” In: Proceedings of the 2006 ACM/IEEE Symposium on Architecture for Networking and Communications Systems. ANCS '06. ACM, New York, NY, 93-102, 2006.