Rethinking Deep Packet Inspection Design and Deployment in the era of SDN and NFV

Hanan Mustapha*, Soufiene Djahel*, Philip Perry† and Zonghua Zhang‡
*Manchester Metropolitan University, UK
†Ulster University, UK
‡Huawei France Research Center, Paris, France
{hananmustapha.96@gmail.com,s.djahel@mmu.ac.uk, p.perry@ulster.ac.uk, zonghua.zhang@huawei.com}

***Abstract*—With the advent of Software-Defined Networking (SDN) and Network Functions Virtualization (NFV), the design and deployment of DPI (Deep Packet Inspection) must be reconsidered. The programmability, global visibility and centralized control of SDN, as well as the NFV enabled lightweight service creation and migration, have potential to empower the capability of DPI tools. On the other hand, dynamic environments make the deployment of DPI challenging. Although it has been validated that some security functions like firewall, and Intrusion Detection System (IDS) can be implemented in SDN controllers or NFV, it remains unclear whether or not DPI can be done in the similar way considering its sophisticated interactions with the network traffic packets, especially for the stateful protocols and encrypted traffic. In other words, the design and deployment of DPI in an SDN and NFV architecture would not be as straightforward. Therefore, this paper aims to shed the light on the challenges facing DPI design and deployment in the context of SDN and NFV and propose a solution to overcome them.**

***Index Terms*—SDN, NFV, Deep Packet Inspection.**

I. Introduction

The rapid growth of Internet traffic types and volume along with the associated increase in network complexity have created a significant drive to change the way current network services are designed, deployed, managed and secured. That drive has seen a move away from dedicated telecommunications and data communications management nodes to a new ecosystem of management functions that leverage technologies that have been matured in distributed software systems. The two core concepts in this new paradigm are Software Defined Networking (SDN) [1] and Network Function Virtualization (NFV) [2] which are both network growth enablers due to their software-based functionality. This offers ways to economically scale networks and deploy tailored network management capabilities that can efficiently respond to changing requirements from end users, network operators and service providers.

Maintaining a secure network requires real-time observation, deep inspection and classification of network traffic to trigger adequate network reconfiguration and intrusion detection alerts to stop the detected threat or at least minimize its impact. Both NFV and SDN technologies can be very useful in achieving the desired security level for these networked systems, although they might be prone to their specific security threats as well [3], as they can enable faster response to detected security breaches [4]. The ability to forensically examine network traffic after a successful security attack is very important as it provides a learning basis that help the attacked system to be more resilient against new attack attempts and enables pursuing the perpetrators. Network forensics is a branch of computer forensics that is responsible for detecting network related crimes [5]. It uses specialist software to analyze network traffic in terms of its patterns and the content of its packets to recognize the characteristics of security attacks along with their source.

Deep Packet Inspection (DPI) is a mechanism used in network forensics to inspect the payload of a packet rather than just the meta-data that can be extracted from the headers. Existing DPI tools have certain limitations, as discussed later in the paper, that can impact their efficiency, agility and practicality but we believe that the advent of SDN and NFV could be a potential solution to these limitations. Indeed, the separation of data and control planes in SDN facilitates the extraction of the required information for DPI tools from the network. Moreover, thanks to NFV, network expansion is made easier as well as dynamic deployment of DPI tools as software services. Therefore,this paper will present an in depth investigation of the potential use of SDN and NFV to improve the efficiency, agility and scalability of the existing DPI tools.

The remainder of this paper is organized as follows. In Section II, we briefly describe the key principles of SDN and NFV and discuss their advantages and potential limitations in addition to their main application domains. In Section III, we review the most important approaches for Deep Packet Inspection and critically analyze and compare them. Afterwords, we propose an SDN-NFV architecture to enhance DPI efficiency and overcome a number of limitations of current DPI approaches in Section IV. Finally, we conclude the paper in Section V and highlight potential future works.

II. An Overview of SDN and NFV

In this Section, we will introduce the key principles, terminology and features of SDN and NFV that will have an impact on network forensics in general, and DPI in particular.

A. SDN Technology: Key Principles

SDN is a networking technology that is widely deployed and offers improved network management [6]. Although programmable networks have been evolving since 1995 [1], it is only in recent years that the widespread use of a consistent

approach has become possible. SDN has enabled large networks to efficiently cope with sudden increases in traffic or periods of congestion, since the network components can be programmed to reconfigure the network topology [6].

The key principles that govern SDN's operations are [7]:

- The control and data planes are decoupled, which means that all control decisions are removed from packet forwarding devices.
- Packet forwarding no longer depends on the destination address in each packet header; rather it depends on a flow table that contains forwarding rules for each flow.
- A traffic flow is identified by one or more characteristics of the packets - such as source Ethernet address, packet size, packet type or IP addresses.
- When a packet arrives at a switch, it is matched to a flow if possible. Any unidentified packets are forwarded to the controller and a new forwarding rule is sent from the controller to the switch's flow table.
- The network behavior is therefore dependent on the network application using it.

The main components of SDN, as shown in Figure 1, that are of interest in this paper are the following:

- SDN Applications: These are the programs connected to the SDN controller's "North-bound" interface which determine how the controller sets flow table rules.
- SDN Controller: As the name suggests, it is responsible for controlling the network traffic and responding to SDN applications' requests. The controller presents a rather abstract view of the network to the external applications.
- Control Plane: This is made up of SDN controllers (there could be more than one controller in the same architecture).
- Data Plane: This is made up of devices or resources responsible for traffic forwarding and processing.

The most widely used protocol that connects the control plane to the data plane in SDN is the OpenFlow protocol [8].

SDN has the capacity to accommodate many services that are governed by abstract policies to control flows that can be identified by a large number of parameters. Although the goal of this abstraction is to simplify network management and translate it into more "human friendly" language, it is also possible for policies to interact in an unexpected way which can impact on performance, function and security. Moreover, the move away from the well defined routing rules of conventional IP networks requires retraining of staff to enable them to understand the impact of reconfiguration of the policies that govern network behavior.

B. NFV Technology: Key Principles

The dramatic increase in the speed of general purpose processors and Virtual Machine (VM) technologies have resulted in commodity hardware that can execute complex, real time functions at low cost. In the networking domain, the virtualization of network functions has developed greatly in recent years so that there are now a range of network functions that

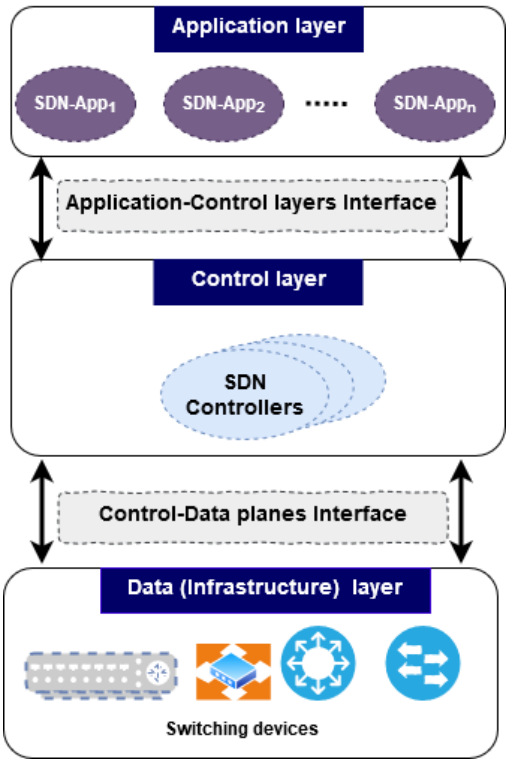

Figure 1: Three-layer model of SDN

can run on a regular Linux machine with moderate compute capability. These efforts have been assisted by standardization groups such as ETSI-NFV who have developed industry led protocols for the management and orchestration of NFV.

The key novelty of NFV is that the major networking functions are no longer physical hardware nodes, they now run as software. To gain a better understanding of the NFV architecture, as shown in Figure 2, we will outline its three main components and how they operate [9]:

- **Virtual Network Functions (VNF):** This is the actual implementation of network functions or services in software, such as (virtual) firewalls, switches, Domain Name System (DNS) servers or DPI.
- **NFV Infrastructure (NFVI):** This is the combination of physical and virtual processing, storage and connectivity resources needed to create the environment where VNFs can be deployed. NFVI includes the hypervisors as well to run the virtual machines (VMs) and share physical resources between them.
- **NFV Management and Orchestration (MANO):** As its name suggests, this is the component that handles the network's initialization, modification and other management operations. The orchestration is required for the management of a VNF over its full life cycle and any possible automation in initiating new services.

NFV is very beneficial for network operators and service providers because they no longer have to purchase bespoke, single function hardware. Instead, one general purpose com-

puting device can be used to run multiple network functions that can be tailored to specific network management requirements and only when requested. This implies improved scalability and facilitates automation [9]. Within such a system, then, the security policies and mechanisms can be automated, in particular, DPI can be targeted to specific traffic types of packets that originate in a particular geographical area.

C. NFV MANO

The integration of SDN concepts and NFV deployment methodologies have attracted a great deal of research attention coupled with industrial development - with a primary goal of standardization of the approaches [10]. One of the prime drivers has been the European Telecommunications Standards Institute (ETSI) through their initiative known as ETSI-NFV. In the context of this current paper, the subset of that initiative known as ETSI-NFV-MANO is of particular interest as it defines an architecture that is built on robust research and industry best-practice.

One important concept in NFV-MANO is the idea of Service Function Chaining [11] where a number of virtualized services or functions can be concatenated to provide a network service to create a higher level network function or deliver some business goals for the network operator. Service chaining can therefore be used to insert DPI functions in a particular network path to enable some security functionality.

This idea of network services is also discussed in [12] with particular reference to the core of the fifth generation mobile network (5GC) which is moving to a Software-Based-Architecture (SBA) that embodies SDN, NFV and MANO concepts as network services or functions. In fact, these concepts are so deeply embedded in the 5GC that they are often not identified as separate but rather are all pervasive, essential concepts.

The idea of viewing security as a suite of services that can be orchestrated to deliver adaptive or tailored security solutions using both physical security nodes or virtualized security functions has also been explored [13]. This interesting approach to combining virtualized and physical resources is of particular interest to the deployment of DPI solutions.

III. Deep Packet Inspection: State of the Art

In this Section, we provide a brief overview of Deep Packet Inspection in networking to understand how it is performed. We then discuss a number of commercial products that use DPI for network forensics. Finally, we critically analyze the different existing approaches for DPI and highlight their respective strengths and limitations.

A. An Overview on DPI

The basic function of a router is to receive a packet and forward it towards its destination address without checking its content (i.e., the payload). This makes the network vulnerable to malware, spyware or any sort of malicious data that could potentially be hidden in the payload of a packet. Therefore, the concept of DPI was introduced allowing some authority to

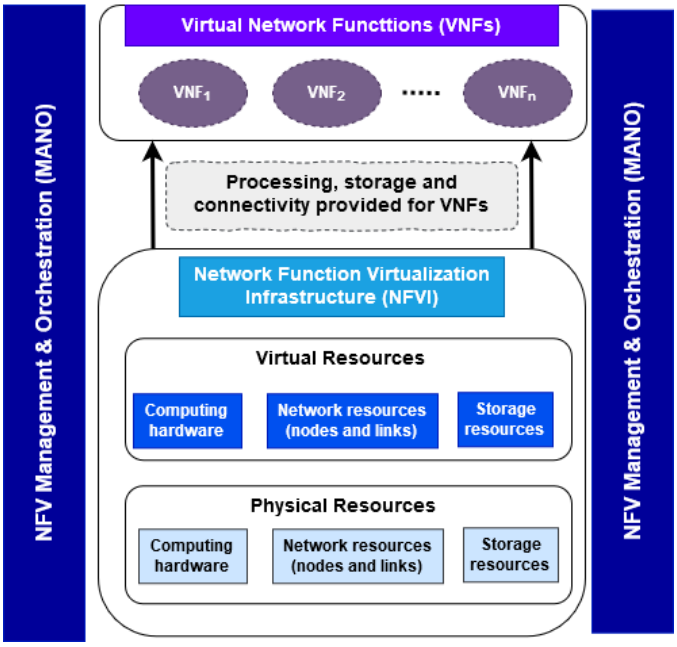

Figure 2: NFV Architecture according to ETSI

open a packet, inspects its content and decide on whether this packet should be forwarded or it presents a threat that should be either flagged, dealt with or both. DPI enables inspecting layers 2 to 7 from the Open Systems Interconnection (OSI) model [14]. The most common inspection used consists in comparing the packet content against certain attributes and checking for hits in any of the accessible layers. Depending on the application in which DPI is used the inspection process could be customized in terms of what needs to be matched, such as traffic from a certain IP address etc. DPI includes port-based analysis that enables the identification of the port number by inspecting the TCP or UDP headers and thus determining which protocol was used to generate the packet. Statistical analysis [8] can also be employed as a DPI method, but it is not payload specific and rather uses several approaches to classify traffic based on port numbers or timestamps. We argue, however, that both port-based and statistical analysis methods are not exactly DPI but could be classified as what is known as Shallow Packet Inspection (SPI) [14] because the payload is not involved at any stage of the inspection process. There exist multiple DPI methods, the variation is determined based on how the payload is matched to keywords. Understanding the logic behind DPI is an important step to find out which method should be avoided and which can be improved. Some of the methods can be performed as either string matching or regular expression matching [15]. We are mainly interested in two methods and the ones that are most commonly used among all [15], they are as follows:

- **Automaton-based:** This involves both regular expressions and string matching. A finite state automaton is created for each string or packet. The input can take multiple states until it is able to match itself to the

required expression or string.

- **Heuristics-based:** A trial-and-error approach to find a specific string or expression in the payload.

When explaining the Automaton-based DPI later in Section III-C we shall introduce Non-deterministic Finite Automaton (NFA) and Deterministic Finite Automaton (DFA) and how they perform differently. Inspecting the packet payload will naturally decrease the network speed since every packet will undergo this process, which is the first issue associated with DPI and there are more. Another issue that arises is data and user privacy since the packet's content or a portion of it are being revealed to an unknown (or even known) third party other than the intended recipient. Achieving a secure and fast network is yet to be real; research has been expanding to reach a consent that has the best of both. We believe that DPI is an important tool that enforces network security and is highly important for governments. Internet Service Providers (ISP) and other enterprises both implement DPI in their network infrastructures, but usually they are less concerned with security. DPI does have a high risk of being illegal to conduct and many privacy concerns, especially with encrypted traffic, that could make it a data breach rather than a network security mechanism.

B. Commercial Applications of DPI

DPI is an application that is used on a daily basis by networks' users without their knowledge. The most common DPI personas are firewalls, parental or data filtering and Intrusion Detections Systems (IDS) [16]. The purpose of DPI in networks is determined by the organisation it is serving. Security benefits of DPI applications are not limited to firewalls and IDS only; we shall discuss the most significant DPI models by classifying them at a government level, enterprise level or personal level. We will also show how encryption is no longer an inspection preventer in most of the known DPI systems.

Governments take the responsibility, whether users approve or not, and closely monitor all Internet traffic and can even block certain services. In fact, Virtual Private Networks (VPN) are blocked in China, Iran and The United Arab Emirates. Governments do not like users surfing the web while being connected to VPN for two reasons, the first is that they can no longer monitor what a user is doing and the second is that the VPN server could be hosted in a hostile country, which could leak important and sensitive data. Some large online vendors and social media websites are not accessible in parts of the world for reasons that are not fully justified or accepted by users. Governments can even censor the data that is sent over the network, and this is where DPI is useful for them as it enables them to identify and eliminate packets with unwanted data. The reasons for which this is done include preventing attacks at higher network level, others remain political. Some DPI deployments are purely business related, for example if an enterprise provides a certain video streaming services, they could block traffic coming from other video streaming websites or applications either by checking source IP addresses or a certain payload pattern in the packets. One could argue that censoring and blocking actions are not intended to secure the network, but the ability to use them in such scenarios implies that they can be leveraged to make the network safer and is a good starting point for research.

Firewalls are widely used software for packet inspection that protect and filter traffic between two points in a network [16]. Firewalls play an important role in protecting the end-point device from malware and supporting IDS by blocking predefined IP addresses known as a security threat. How deep a firewall inspects a packet would depend on the firewall type and configuration. There are many software vendors that worked on and produced different or multiple DPI solutions in the form of network management or IDS services; among these vendors, we have Cisco, Allot and Qosmos. Some of these technologies have even been patented due to their uniqueness in DPI. A previous survey on existing DPI software was an insight for this paper, further information can be found in [17]. A software developed by Ipoque named Protocol and Application Classification Engine (PACE) uses DPI for both network security and network management [18]. It performs DPI on OSI layers 4 to 7 and uses analysis techniques for traffic classification. It has built-in firewalls, IDS and other network security applications.

Allot has developed Predictive DPI (PDPI) technology that is able to successfully perform DPI on encrypted data to prevent certain network attacks and perform successful URL filtering[1]. Symantec developed a service named DeepSightTM Intelligence which is a fully cloud-based cyber threat system to ensure network security. It is able to identify vulnerabilities in the networks or any malware files. In fact, there are other companies, such as Sandvine, that use DeepSightTM Intelligence in their own network security solution. Qosmos ixEngine is another DPI Software Development Kit (SDK) that can be used by developers to employ and configure DPI functionality into networks. Qosmos states that more than 70% of network vendors use their SDK to produce end products [19]. ixEngine is useful for inspecting OSI layers 2 till 7 and can analyze traffic and also perform metadata extraction for improved traffic classification. Cisco also uses DPI for application classification through the Application Visibility and Control (AVC) used in their routers and other networking hardware. Cisco produced multiple network devices that are able to perform DPI. They do not elaborate on the security benefits, rather they emphasize on the network management ability and control [20]. Another network management solution, known as BIG-IP Policy Enforcement Manager[2], was developed by F5 Networks to perform URL filtering and inspection on packet headers and OSI layers 4 to 7.

C. DPI approaches: a taxonomy

The research on DPI has been active for almost two decades to develop more accurate IDSs. Since DPI is performed

[1]https://www.allot.com/service-providers/traffic-management/encrypted-traffic-classification/

[2]https://www.f5.com/products/big-ip-services/policy-enforcement-manager

by inspecting the packet headers or payload and searching for some known keywords, research efforts have focused on making this matching process faster and more efficient using regular expressions and pattern matching algorithms as well as dedicated hardware devices. Other researches have attempted to apply DPI on encrypted payload and even onion router (TOR) traffic. In this section, we will critically analyze the most important related research works by providing some insights about their methodology and solutions.

*1) **Bloom Filter based DPI with Hardware support**:* Bloom Filters have been tested to perform string matching in a predictive manner rather than algorithmic. The Bloom Filters theory was founded in 1970 by Burton Howard Bloom [21] and is in fact string matching data structure that used probability when achieving results, hence 'filtering' a stream of data. In theory, this can be particularly useful to speed up payload inspection. Multiple Bloom Filters were used by Dharmapurikar et al., which they refer to as Parallel Bloom Filters in order to improve the speed of DPI [22]. They developed a prototype and inspected data coming from the internet to their network via a Washington University Gigabit Switch (WUGS)[3] with a capacity of 20 gigabits per second (Gb/s). The obtained results show that 10,000 string were captured using their prototype, however they do not state how many payloads were inspected nor the total number of strings searched. Parallel Bloom Filters were not used in this prototype, only a single filter; however, the concept is very feasible. They also used reconfigurable hardware to create their network which is a good concept to consider as it can support the use of virtualisation in DPI. The authors take on another attempt with using Bloom Filters and this time in a network based IDS. They combined the Longest Prefix Match (LPM) algorithm and the Aho-Corasick algorithm [23] to improve the string-matching rate. Their system is based on Cisco's open source IDS, named Snort, using on-chip memory hardware to reach a data rate of 10 Gb/s.

In conclusion, we believe that applying this experiment again using a memory hardware that could even reach up to two tera bytes would make the process much faster than 10 Gb/s. If we were to compare this rate with the latest version of PACE which is 4 Gb/s, we can see that this mechanism, even being an open source appears more promising. One might argue whether this is realistic, ixEngine can achieve a highest rate of 10Gb/s on each x86 core and Allot Service Gateway 9700 [24] cluster can go up to 2 Tb/s. The main advantage of [22] is that it achieves a reasonable data rate which is competitive with commercial DPI. However, the hardware used can be costly and could be a huge obstruction for network expansion.

*2) **DFA-based DPI enhanced with regular expressions rewrite rules**:* Regular expressions are the key concept behind the design of any string-matching algorithm, and should be given great importance when attempting to improve any DPI mechanism. Yu et al. proposed a novel fast and memory-efficient regular expressions matching technique to improve DPI in [25]. In this work, the authors designed novel regular expression rewrite techniques to effectively reduce the extremely high memory requirements of the existing schemes. Then, they developed a grouping technique that enables compiling a set of regular expressions into several engines, leading to a significant enhancement of the matching process speed without excessive increase in memory usage. The above techniques were used in a new unique implementation of DFA-based matching to perform DPI. The general concept of expressions rewrite rules used in this work consists in reducing the total number of hops a system requires to retrieve a certain character combination, and thus reducing the required memory space. Although DFA approach is better than NFA in terms of the achieved speed and memory usage required it is still unclear how the rewrite rules are being applied on certain combination of expressions.

*3) **Delayed Input DFA based DPI**:* Regular expression sets that arise in various advanced network applications, such as security appliances from CISCO systems, and are represented by DFA still require high memory usage, which significantly limit their practicality. To overcome this issue, Kumar et al. have taken DFA to a further level and designed a new technique named Delayed Input DFA (D^2FA) which aims to reduce the memory usage during a DPI process [26]. Their idea consists in reducing the number of transitions performed to one only, where a transition is defined as the number of matching functions executed. This reduction is achieved by splitting tasks between multiple memory chips during each inspection made by the DPI process. Such approach would make the process faster since each memory chip is dealing with less data. It is worth to note that the term "Delayed Input" comes from waiting for every chip to complete processing and then combining each chip transition with the corresponding transition. For cost-effectiveness purpose, the proposed D^2FA technique was implemented using multiple low storage memory devices but still achieve faster results.

*4) **Content Addressed Delayed Input DFA based DPI**:* The main limitation associated with the use of NFA or even D^2FA in DPI is the reduced network throughput. In [16], Kumar et al. proposed a novel technique, named Content Addressed Delayed Input DFA (CD^2FA), that provides compact representation of regular expressions and thus achieves a throughput equals to that of uncompressed DFA, without the need of extra memory. CD^2FA is built upon the same idea of D^2FA with the exception that state numbers are replaced by content labels. The information contained within these labels enable CD^2FA to avoid any default traversal, meaning that unnecessary memory accesses are prevented and higher throughput can therefore be achieved. Although the memory access rate in CD^2FA is the same as in uncompressed DFA when carrying out a search for specific keywords, CD^2FA achieves higher throughput than DFAs in systems with a small data cache because of their small memory footprint and high cache hit rate. According to the results shown in [16], we conclude that CD^2FA is capable of implementing regular

[3]https://www.arl.wustl.edu/projects/archive/gigabitkits/switch.html

expressions in a more economic way as well as enhancing the achieved throughput and scalability in the number of rules. Moreover, it only requires off-the-shelf computing peripherals and avoids complexity, making it simple to implement and more realistic for use in DPI process.

5) ***Cache-based and prediction assisted DPI****:* Although DFAs are widely used to perform multiple regular expression matching in linear time, their implementation in modern memories affects negatively the matching speed, especially when the size of DFA gets larger. Tang et al. [27] investigated the use of both cache memory and prediction techniques to improve DPI speed. Their work consists mainly in finding a way to make DFA behavior predictable and improve cache memory access. They introduced a new concept called local prediction which enables predicting the memory accesses to the DFA and thus increases the cache hit rate. Such prediction helps in speeding up the process by taking decisions on whether the rest of the payload should be inspected or skipped. They developed a prototype consisting of an IDS that uses Snort data on real traffic. The objective of this work was to improve the performance of DPI process by improving the hit rate and the robustness level. To conclude, we believe that it would be useful to develop different prediction approaches that could improve the prediction accuracy, or apply local prediction to other than DFA, such as Content Addressed D^2FA. Using different or large cache memory could also improve the accuracy of the evaluation process. Moreover, the performance evaluation of prediction approaches should also consider other metrics such as the processing speed. In fact, there is also a vendor competition, Allots' PDPI. A lot of work can be done to keep up with the pace of vendor based DPI.

6) ***Speculative Parallel Pattern Matching based DPI****:* The large set of patterns used by modern DPIs to inspect network packets are usually defined using regular expressions parsed by DFAs. Since these packets are inspected one byte at a time, this will slow down the DPI process. To avoid this issue, an interesting research work conducted by Najam et al. proposes to use speculation and multi-stride (stride-k) to improve data packets pattern matching in DPI [28]. The main idea consists in splitting each packet into two chunks, then inspecting the bytes of each chunk simultaneously using speculation and multi-stride DFA, where a stride represents the number of bytes processed per state transition. This will certainly improve the processing speed but leads to high memory usage, which requires some sort of data compression to reduce it. To this end, Najam et al. chose a compression scheme that uses alphabet compression tables and data addressing built upon the work of Kong et al. to reduce the memory consumption [29]. Experimental results demonstrated the effectiveness of this scheme as it successfully reduced the memory usage by 65% compared to uncompressed DFA. Attempting to inspect chunks of data bytes rather than a single byte is not impressive since enterprise DPI has with all means reached very fast throughput, making the market congested and competitive. In conclusion, packet payload must be studied with more depth and focused on when developing efficient DPI methods. A similar research work was conducted by Luchaup et al. to tackle multiple regular expressions and speculation, further information can be found in [30].

D. DPI vs TOR and Encrypted Traffic

Cryptography was introduced into computing to help keep data inaccessible by unauthorized users. This brought up two challenges for DPI; the first is how to decrypt the data and the second being the issue of data privacy. Our main concern is the first challenge, how can data be decrypted for inspection, or is there another means of inspecting packets without the need of decryption. Another challenge is linked to onion routing (or TOR) which is a mechanism used to make packets untraceable to hide their origin. TOR software encrypts the data being sent and the next destination address since the routing is distributed among many other routers. Although tracing the source is not the focus of this work, it is an important aspect to look into if suspicious packets are identified. Some vendor DPI software managed to tackle encrypted data and have produced successful results such as PDPI by *Allot*.

1) ***DPI vs TOR Traffic****:* To the best of our knowledge, current IDS systems are not able to efficiently handle TOR traffic, however, Saputra et al. attempted to unveil certain TOR trademarks that can help in blocking them completely. In [31], Saputra et al. performed network analysis with the open source software Bro. They managed to determine certain identifiers in Transport Layer Security (TLS) handshakes such as cipher suite used by the browser. This work is not particularly targeting payload inspection; however, it sheds light on the fact that even the most intelligent ways of staying anonymous on the internet can be identified easily if network data is properly analyzed. Since there is no certitude that all TOR traffic is bad or harmful, there is no need to block all data transmission through TOR. Saputra et al. did not clearly state what the cipher suite TOR uses or any of the other TOR identifiers they determined, so even if their proposed TOR blocking mechanism was implemented by an ISP, it is not clear what they must block. To conclude, we believe that more work must be done in order to correctly identify TOR traffic, which could be extremely important for DPI at a government level.

2) ***BlindBox****:* Sherry et al. proposed a novel way of performing DPI over encrypted Hyper Text Transfer Protocol (HTTP) traffic, named BlindBox [32]. They proposed an architecture that involves a middle box to perform DPI. The traffic that enters the middle box is encrypted using an encryption method named DPIEnc. Both BlindBox and DPIEnc are technologies developed by Sherry et al. and work together to achieve the sought results. The basic model consists in encrypting the traffic twice, once with Secure Sockets Layer (SSL) and the second time with DPIEnc. The traffic is then split into smaller strings based on some token; in their case, they used 8 bytes per token. They reinvent a new HTTP secure (HTTPS) scheme that is meant for use in DPI which acts in a similar way to a proxy server. In order to do so, BlindBox

is designed to operate using three protocols, which are the following:

- **Basic Detection:** this is word to word matching or matching traffic with keywords.
- **Limited IDS:** this works like an IDS and uses the first protocol as detection scheme.
- **Full IDS with Probable Cause Privacy:** this uses the first and second protocols, but is able to decrypt the data when necessary.

They also conducted a detailed evaluation of the three protocols by testing BlinBox performance on real traffic (functionality) and checking whether it is plausible to deploy it in real networks by assessing the generated overhead. The protocols are designed to work together or separately depending on the environment and requirements of the network. BlindBox inspection achieved a detection rate of 186 Mb/s, when deployed on single core memory, which is quite reasonable. This DPI scheme respects users' privacy since it does not decrypt any of the payload data, making it reliably protective of user and data privacy that cryptography is designed to provide. Since there is no decryption involved, the network speed is not significantly affected. Since the encryption is done in-house, it is easy to retrieve any necessary decryption keys if any maliciousness is detected (the third protocol).

In conclusion, BlindBox appears to be a potential IDS that can perform DPI over encrypted traffic without ruining any privacy agreements made by cryptography. We believe that it could be turned into a commercial product, however this depends on the adaptation of the BlindBox-based HTTPS. It also appears that there could be some vendor locking if BlindBox is used since it can only operate with its own hardware and protocols (i.e., BlindBox requires all its components and protocols to work, including a middle box).

*3) **GINTATE**:* A similar research work has been conducted by Miura et al. in [33] using a monitoring framework named GINTATE capable of performing DPI over TLS traffic. In this work, TLS communications are decrypted using a shared key between the framework and the client device and the results are then scrutinized in DPI processing. GINTATE consists of three main components responsible for performing different tasks as described below:

- **GINGATE:** is responsible for intercepting the packets carrying TLS related communications and forwarding them to GINPEEK.
- **GINPEEK:** perform DPI processing by analyzing the packets received from GINGATE after decrypting them.
- **GINFRIEND:** holds the session keys and information required for decryption to share with GINPEEK.

One of the distinguishing features of GINTATE is its scalability which is achieved by splitting the DPI processing across several computing servers for every TLS session. This is done as a response to the increase of the volume of captured packets where GINGATE adds new GINPEEKS to avoid saturation of the available computing resources. Moreover, GINTATE offers an independent analysis module that can be extended by programs in any language and enables achieving detailed protocol analysis. In GINTATE, decrypting TLS traffic adds an extra computing overhead in addition to the communication overhead incurred by key sharing between GINFRIEND and GINPEEK. The critical nature of key sharing adds strict constraints on their storage method to prevent any leakage. We do not recommend the decryption of data before inspection since it does not preserve users' privacy, other than the fact that it slows down the inspection process while doing a non-DPI task. Furthermore, TLS 1.3 was launched as of August 2018 which included a few major changes from TLS 1.2, including a change in the cipher suite [34]. Although it is not stated as to which TLS version was used by Miura et al., but this could mean that GINTATE needs upgrading.

E. Discussion

In [35], Becchi & Crowley argued whether using DFA is 'practical' in DPI due to the high memory requirement associated with it, or should DPI be restricted to the use of NFA only for regular expression matching. They introduced a new concept named Hybrid Finite Automaton (Hybrid-FA), which combines the best of both DFA and NFA and tested its performance using Snort IDS on real traffic. Hybrid-FA has been found to use higher amounts of memory than NFA but lower than that required by DFA. DPI has now advanced through the use of data compression methods and cache memory, making DFA very practical. It is also not clear how the FA is able to alternate between using NFA and DFA, that is; when is better to use each of them on a certain expression. We strongly believe that there is a potential to develop a fast and low cost mechanism that improves the throughput in NFA but we still do not know the best way to achieve that. In Table I, we compare the DPI solutions discussed in Sections III-C and III-D by highlighting their respective strengths and limitations.

IV. SDN-NFV Architecture for Enhanced DPI

In previous sections, we have introduced and described various DPI solutions; some are fully software based while others require hardware enhancements. We have also highlighted the benefits of combining SDN and NFV to create a unified network infrastructure that can improve agility and scalability. In this section, we will examine how DPI can benefit from an SDN-NFV architecture.

*A. **SDN-NFV Architecture***

The reduced network throughput rate that DPI causes was the major issue faced by most DPI solutions. We believe that SDN and NFV, combined, have the potential to overcome this problem based on these key factors:

- SDN separates data from control, making it easy to extract the information from the network that is required for inspection.
- NFV facilitates the expansion of a network, making it easy to dynamically deploy software services.
- DPI exists as software that lends itself to virtualization.

DPI Solution	Strengths	Limitations
Bloom Filter based DPI with hardware support [22]	Reasonable data rate	Hardware usage
DFA-based DPI enhanced with regular expressions rewrite rules [25]	Lower memory consumption	Rewrite rules usage
Delayed Input DFA based DPI (D^2FA) [26]	Requires one memory chip	Slow inspection
Content Addressed D^2FA [16]	Off-the-shelf resources usage	High memory access rate requirement
Cache-based and prediction assisted DPI [27]	Cache memory and prediction usage	Narrow evaluation performed
Speculative parallel pattern matching based DPI [28]	Capability of inspect multiple strings	Not challenging at an industrial level
DPI vs TOR Traffic [31]	Detect TOR traffic	No payload inspection
BlindBox [32]	Respecting data privacy	Vendor lock-in
GINTATE [33]	Network independent	Slow inspection

Table I: Summary of the main strengths and limitations of the existing (DPI) approaches

We use a simplified and abstracted model of the SDN/NFV architecture, shown in Figure 3, which includes only the most essential parts that are required to understand how applications can be deployed as network functions [36]. Starting from the top of the figure the application layer contains the VNF, in our case DPI would reside in this layer, meaning that DPI is deployed as a virtual function. The main reason for implementing DPI as a VNF is to take advantage of virtualization, which implies the benefits of being more agile, better resource utilization and lower costs. For example, an ISP can have more than one DPI solution available that can be selected depending on the customer's request. This is not only useful for DPI, but can be important for running IoT services. We previously highlighted that firewalls are being improved to include DPI mechanisms, but this is not universal, so that it is best to have DPI software deployed as part of the application layer instead of control layer. Furthermore, an ISP can provide a dedicated and tailored DPI solution to customers who request it at an additional fee.

The NFV MANO resides in the control layer along with the SDN controller which can be virtual or physical. This layer is responsible for the management of the network functions and other resources. The control layer is connected to the application layer through an API so that the MANO functionality can deploy applications dynamically. One would naturally assume that the SDN controller would implement some sort of security mechanisms by default, however, having the controller making all decisions would slow down the functionality and effectiveness of the networks and impact on the end user's performance. Hence, we believe that we can exploit the flexibility and efficiency of NFV MANO by separating these functions from the SDN controller functionality.

Finally, the data layer or infrastructure layer would be made up of any hardware that is required, including storage. This is particularly useful for DPI that requires Bloom Filter hardware, or even any additional external hardware. Note that the Bloom Filter is optional and is only deployed if a specific DPI solution required it. Other DPI solutions, such as BlindBox, which is a middle box, could be used as part of the data layer; however, it might need some reconfiguration to adapt into the new SDN/NFV environment. This is assuming that the middle box is hardware based, however it can be deployed as a VNF.

The OpenFlow protocol is used for communication between the data and control layers so that there is no direct communication between data and application layers, the control layer acts like a mediator between the two.

B. *Potential Benefits*

It is important to identify the benefits associated with the SDN-NFV architecture described in the above section to understand how it can benefit DPI and overcome some of its associated limitations outlined in Table I.

a) ***Data Layer****:*

- All physical and virtual resources reside at this layer. The virtualisation infrastructure is of particular interest here, including hypervisors, compute platforms, storage and applications specific hardware.
- It is abstracted away from the network control and applications so that additional resources can be added.
- These resources can also be dynamically reconfigured to accommodate multiple services if needed.

b) ***Control Layer****:*

- The control requirements for the SDN part and the NFV part of the architecture are fully independent from each other.
- The control layer can be completely virtual (or logical) to reduce costs and increase flexibility.
- Additional security mechanisms can be embedded within the network here.

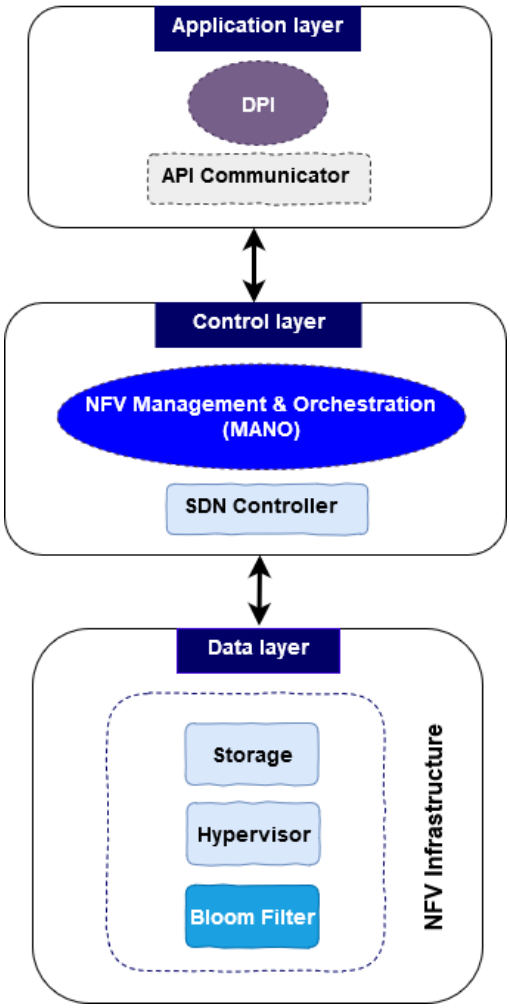

Figure 3: Hybrid SDN-NFV architecture

- The control layer is the key behind the agility and scalability of the entire network.

*c) **Application Layer**:*

- The software for each of the VNF's resides here.
- DPI can be deployed as a VNF so that it can be independent from other network functions.
- Multiple DPI tools can be added, whether they are open source or vendor based.

C. *Impact on DPI limitations*

After identifying all the possible benefits that an SDN-NFV architecture can bring, we will now explain how it can tackle the current limitations we found in existing DPI solutions/tools (see Section III-E). The nature of some of these limitations prevents SDN/NFV from being a solution so we shall not include them. We discuss these in the same order they were described in Table I. A summary of these corrections can be found in Table II.

*a) **Hardware resources usage**:* The first limitation we identified was hardware usage, which specifically required some sort of Bloom Filter processing. SDN-NFV architecture can directly improve this by creating a VNF that can connect the hardware for the Bloom Filter into the data path when needed. Moreover, Bloom Filters can be implemented as a purely software function that could be used by other components.

*b) **Speed (slow inspection)**:* The inspection speed of D^2FA can be improved with the SDN-NFV architecture since there is no need for DPI to run on a single physical machine. This means that the virtual capacity can be expanded as required to adjust to the optimum speed and no longer affect other services running on the application layer. NFV MANO can also be useful to ensure the consistency of the prediction rules, which we briefly mentioned in Section III-C3. Network speed was also a limitation to GINTATE (see Section III-D3) since it involved decrypting all traffic. By using SDN to steer traffic towards a scalable compute platform running the decrypting component (GINPEEK) could improve the speed.

*c) **High memory access rate**:* Content Addressed D^2FA was one of the favorite DPI solutions we explored since it did not require customized equipment. However, in terms of performance, it did require high access rate to the memory. This would affect the overall performance of the network regardless of the end result. As before, using SDN to steer the traffic to one or more virtual functions with scalable fast access memory and assembling the ensemble with NFV MANO resolves the problem.

*d) **Vendor lock-in**:* BlindBox was an overall DPI solution for inspecting encrypted data packets without the need of decrypting them. The major limitation resulting from BlindBox was the fact that it requires all its components to function and uses its own HTTPS mechanism resulting in vendor lock-in. SDN/NFV cannot help in overcoming this limitation, but can ease the integration of a third party cloud-based solution into the network.

Limitations	**Solved by our SDN-NFV architecture?**
Hardware usage	✓
Rewrite rules usage	N/A
Slow inspection	✓
High memory access rate requirement	✓
Narrow evaluation	N/A
Not challenging at an industrial level	N/A
No payload inspection	N/A
Vendor lock-in	×

Table II: DPI limitations correction

V. Conclusion and Future Work

We focused in this paper on exploring the great opportunities that SDN and NFV could offer to improve network forensics functions, in particular DPI. After analyzing the benefits

and limitations of both SDN and NFV and investigating the most important existing DPI approaches and identifying their key limitations, we explored the possibility for developing an SDN-NFV architecture to overcome them. The SDN-NFV architectural model we used is made of three layers; application layer, control layer and the data layer. We suggested that DPI should be deployed as a VNF in the application layer to scale up its usage as a service. The architecture is also flexible so that hardware-based DPI can be connected into any point in the network by using the NFV MANO to orchestrate the topology. The flexibility of the SDN/NFV approach also can ensure that the network speed is not affected by DPI. As a future work, we aim to investigate how DPI could be used to prevent from cyber attacks in IoT environments. We are also interested in improving network firewalls to perform some sort of DPI, to reduce the load on IDS tools, and thorough data inspection mechanism that could be part of a network or SDN controller.

ACKNOWLEDGMENT

This research has been supported in part by the BT Ireland Innovation Centre (BTIIC) project, funded by BT and Invest Northern Ireland.

REFERENCES

[1] N. Feamster et al. The road to sdn: An intellectual history of programmable networks. *SIGCOMM Comput. Commun. Rev.*, 44(2):87–98, April 2014.

[2] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee. Network function virtualization: Challenges and opportunities for innovations. *IEEE Communications Magazine*, 53(2):90–97, 2015.

[3] M. Pattaranantakul et al. Nfv security survey: From use case driven threat analysis to state-of-the-art countermeasures. *IEEE Communications Surveys Tutorials*, 20(4):3330–3368, 2018.

[4] B. Siniarski et al. Real-time monitoring of sdn networks using non-invasive cloud-based logging platforms. In *2016 IEEE 27th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pages 1–6, 2016.

[5] R. Hunt and S. Zeadally. Network forensics: An analysis of techniques, tools, and trends. *Computer*, 45(12):36–43, 2012.

[6] H. Farhady et al. Software-defined networking: A survey. *Computer Networks*, 81:79 – 95, 2015.

[7] D. Kreutz et al. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76, 2015.

[8] W. Xia et al. A survey on software-defined networking. *IEEE Communications Surveys Tutorials*, 17(1):27–51, 2015.

[9] R. Mijumbi et al. Network function virtualization: State-of-the-art and research challenges. *IEEE Communications Surveys Tutorials*, 18(1):236–262, 2016.

[10] C. Rotsos et al. Network service orchestration standardization: A technology survey. *Computer Standards & Interfaces*, 54:203 – 215, 2017. SI: Standardization SDN & NFV.

[11] M. Mechtri et al. Nfv orchestration framework addressing sfc challenges. *IEEE Communications Magazine*, 55(6):16–23, 2017.

[12] B. Nogales et al. Design and deployment of an open management and orchestration platform for multi-site nfv experimentation. *IEEE Communications Magazine*, 57(1):20–27, 2019.

[13] Bernd Jaeger. Security orchestrator: Introducing a security orchestrator in the context of the etsi nfv reference architecture. In *Proceedings of the 2015 IEEE Trustcom/BigDataSE/ISPA - Volume 01*, TRUSTCOM '15, page 1255–1260, USA, 2015. IEEE Computer Society.

[14] K. Mochalski and H. Schulze. White paper on deep packet inspection. itu-t study groups com13, 2009.

[15] C. Xu et al. A survey on regular expression matching for deep packet inspection: Applications, algorithms, and hardware platforms. *IEEE Communications Surveys Tutorials*, 18(4):2991–3029, 2016.

[16] S. Kumar et al. Advanced algorithms for fast and scalable deep packet inspection. In *2006 Symposium on Architecture For Networking And Communications Systems*, pages 81–92, 2006.

[17] T. Bujlow et al. Independent comparison of popular dpi tools for traffic classification. *Computer Networks*, 76:75 – 89, 2015.

[18] Ipoque GmbH. R & s pace 2 - solution guide. rohde & schwarz gmbh & co. kg. leipzig., 2015.

[19] Qosmos. Qosmos ixengine: Classification & metadata engine. qosmos. paris., 2016.

[20] Application visibility and control. cisco systems inc. california., 2011.

[21] Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426, July 1970.

[22] S. Dharmapurikar et al. Deep packet inspection using parallel bloom filters. *IEEE Micro*, 24(1):52–61, 2004.

[23] Alfred V. Aho and Margaret J. Corasick. Efficient string matching: An aid to bibliographic search. *Commun. ACM*, 18(6):333–340, June 1975.

[24] Service gateway 9700. allot communications ltd. hodhasharon., 2018.

[25] F. Yu et al. Fast and memory-efficient regular expression matching for deep packet inspection. In *2006 Symposium on Architecture For Networking And Communications Systems*, pages 93–102, 2006.

[26] S. Kumar et al. 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*, SIGCOMM '06, page 339–350, New York, NY, USA, 2006. Association for Computing Machinery.

[27] Y. Tang et al. Cache-based scalable deep packet inspection with predictive automaton. In *2010 IEEE Global Telecommunications Conference GLOBECOM 2010*, pages 1–5, 2010.

[28] M. Najam, , et al. Speculative parallel pattern matching using stride-k dfa for deep packet inspection. *Journal of Network and Computer Applications*, 54:78 – 87, 2015.

[29] S. Kong et al. Efficient signature matching with multiple alphabet compression tables. In *Proceedings of the 4th International Conference on Security and Privacy in Communication Netowrks*, SecureComm '08, New York, NY, USA, 2008. Association for Computing Machinery.

[30] D. Luchaup et al. Multi-byte regular expression matching with speculation. In *Recent Advances in Intrusion Detection*, pages 284–303, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[31] F. A. Saputra et al. Detecting and blocking onion router traffic using deep packet inspection. In *2016 International Electronics Symposium (IES)*, pages 283–288, 2016.

[32] J. Sherry et al. Blindbox: Deep packet inspection over encrypted traffic. *SIGCOMM Comput. Commun. Rev.*, 45(4):213–226, August 2015.

[33] R. Miura et al. Gintate: Scalable and extensible deep packet inspection system for encrypted network traffic: Session resumption in transport layer security communication considered harmful to dpi. In *Proceedings of the Eighth International Symposium on Information and Communication Technology*, SoICT 2017, page 234–241, New York, NY, USA, 2017. Association for Computing Machinery.

[34] E. Rescorla. The transport layer security (tls) protocol version 1.3. internet engineering steering group (iesg). california., 2018.

[35] M. Becchi and P. Crowley. A hybrid finite automaton for practical deep packet inspection. In *Proceedings of the 2007 ACM CoNEXT Conference*, CoNEXT '07, New York, NY, USA, 2007. Association for Computing Machinery.

[36] A.L.V. Caraguay et al. An overview of integration of mobile infrastructure with sdn/nfv networks. In *Proceedings of the 7th International Conference on Information Technology*, pages 250–265, 2016.