IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 23, 2018, accepted January 5, 2019, date of publication January 31, 2019, date of current version February 20, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2895898

A New Architecture for Network Intrusion Detection and Prevention

WALEED BUL'AJOUL[1,2], ANNE JAMES[1], AND SIRAJ SHAIKH[3]

[1]Computing and Technology Department, New Hall, Nottingham Trent University, Clifton Campus, Nottingham NG11 8PT, U.K.
[2]Computing Department, School of Science, University of Omar Al-Mukhtar, Al Bayda' 543, Libya
[3]Systems Security Group, Institute for Future Transport and Cities, Coventry University, Coventry CV1 5FB, U.K.

Corresponding author: Waleed Bul'ajoul (bulajoul@gmail.com)

This work was supported in part by the Nottingham Trent University, Nottingham, U.K., and in part by the University of Omar Al-Mukhtar, Al-Bayda, Libya.

ABSTRACT This paper presents an investigation, involving experiments, which shows that current network intrusion, detection, and prevention systems (NIDPSs) have several shortcomings in detecting or preventing rising unwanted traffic and have several threats in high-speed environments. It shows that the NIDPS performance can be weak in the face of high-speed and high-load malicious traffic in terms of packet drops, outstanding packets without analysis, and failing to detect/prevent unwanted traffic. A novel quality of service (QoS) architecture has been designed to increase the intrusion detection and prevention performance. Our research has proposed and evaluated a solution using a novel QoS configuration in a multi-layer switch to organize packets/traffic and parallel techniques to increase the packet processing speed. The new architecture was tested under different traffic speeds, types, and tasks. The experimental results show that the architecture improves the network and security performance which is can cover up to 8 Gb/s with 0 packets dropped. This paper also shows that this number (8Gb/s) can be improved, but it depends on the system capacity which is always limited.

INDEX TERMS Computer security, computer networks, intrusion detection system, intrusion prevention system, network architecture, network security, open source, quality of service, security, switch configuration.

I. INTRODUCTION

Information technology (IT) influences almost every aspect of modern life. Today, various devices are available to meet users' requirements such as high machine processor speed, and fast networks. Alongside our increasing dependence on IT, there has unfortunately been a rise in security incidents. Threats and attacks may range from stealing personal information from a laptop or network server to stealing the most top-secret information stored on a Security Intelligence Service (SIS). Furthermore, hackers can snoop on users' online purchases by eavesdropping on their credit card details, or, even more alarmingly, safety-critical systems can be compromised. Multi-faceted attacks and threats have made the implementation of security systems more challenging. Hackers have evolved along with the sophistication of the IT industry. For example, hackers exploit the developments in computer processors and network speeds to increase the volume and speed of malicious traffic that might constitute a Denial of Service (DoS) or Distributed Denial of Service (DDoS) attack [1]–[3]. Network security is therefore extremely important and has developed into an industry aimed at improving applications and hardware platforms to identify and stop network threats.

One of the most established concepts in information security is a defense-in-depth approach which utilizes a multi-layered structural design, in which firewalls, vulnerability assessment tools (anti-viruses and worms), and IDPS (Intrusion Detection and Prevention Systems) are employed to prevent any hostile endeavours on network systems and servers. The Network Intrusion Detection and Prevention System (NIDPS) has been designed to serve as the last point of defense in the network architecture. NIDPS monitor the transportation of network traffic for any malicious and uncomfortable activities and create alerts when operating in detection mode or block packet alerts when operating in prevention node [4], [5].

The detection and prevention mechanisms of the NIDPS are grounded in observing the comparison of ingress packets (traffic) to any known attack through patterns (signature

The associate editor coordinating the review of this manuscript and approving it for publication was Ali Kashif Bashir.

TABLE 1. Snort-NIDP reaction to detect malicious packets.

Packet sent (TCP/IP flood traffic (Bps) with 255 UDP malicious packets in (1mSec))	Eth (Ethernet) packets received	ICMP packets analysed	TCP packets analysed	Malicious UDP packets analysed	Malicious UDP packets Alerts	Malicious UDP packets logged	% Malicious packets alerts	% Malicious packets logged
16 Bps	100%	35	447	9868	9820	9820	99.51%	99.51%
32 Bps	100%	59	750	8702	8654	8654	99.44%	99.44%
200 Bps	100%	88	2015	7166	7083	7083	98.84%	98.84%
1200 Bps	100%	0	5025	6024	5854	5854	97.17%	97.17%
4800 Bps	100%	91	9012	2876	1421	1421	49.40%	49.40%
60000 Bps	100%	239	93755	7560	2810	2810	35.75%	35.75%

NIDPS mechanism) or identifying unknown malicious patterns from ingress traffic (anomaly NIDPS mechanism). NIDPS are important in that they:

- counter intrusions or malicious attempts to access networks and systems;
- analyze network traffic and identify hackers' targets and techniques; and
- detect or prevent unwanted and malicious traffic.

Open source is the most common category of NIDPS software configured platforms [6]; however, its performance in high- speed networks communication remains a major issue. Irrelevant alerts (false positive alerts) occur, thus creating a more difficult job for system security managers. Moreover, despite claims of increased capabilities and efficient performances by several NIDPS dealers, research has shown that systems lack the required capabilities to monitor and analyze high-speed network traffic [7]–[9].

Innovators have created hardware IDPS to process millions of packets at the same time [10], [11], but there are limitations in the capability to perform particular software tasks. In addition, limited memory size is a problem for hardware-based NIDPS solutions. Furthermore, hardware-based NIDPS offer a high range of processing speeds but are very costly. Software solutions are popular because they are cheaper and offer more flexibility than hardware solutions. This paper focuses on open-source software solutions.

Computer network and internet security face increasing challenges and many companies rely on NIDPS to secure their data sources and systems. The need to ensure that the NIDPS can keep up with the increasing demands as a result of increased network usage, higher speed networks and increased malicious activity, makes this an interesting area of research and motivated this study.

The paper is organized as follows. Our investigation testbed is described in section II. Section III presents our proposed solution and section IV its evaluation. A discussion and comparison to related research is provided in section V. Finally, section VI gives a conclusion and recommendations for future work.

II. INVESTIGATION TESTBED

To investigate the problem, two experiments were carried out. The Snort NIDPS has been configured to NIDPS detection (NID-mode) and prevention (NIP-mode) modes. The experiments were conducted to test Snort NID and NIP modes performance in detecting and preventing malicious packets under high-speed traffic.

The experimental testbed also incorporated generator traffic tools, such as NetScanPro, Packets Generator, WinPcap, capture tool, Packets Traceroute, TCP reply and Packets flooder. The experiments used performance metrics such as number of packets analysed, number of malicious packets detected or prevented, and number of packets dropped. In this section the two experimental setups are described.

A. DETECTING MALICIOUS PACKETS

In this experiment, WinPcap, Flooder packet and TCPreplay tools were used to send flood traffic with signed (known) malicious UDP packets (255 threads per 1mSec) to a physical system at different speeds (see Table 1). The UDP malicious packets were interspersed among other packets transmitted at varying speeds. The following rule has been designed to require Snort to detect (alert and log) any UDP threads or malicious packets that contain the variables 'ab.H′..OK..cdef' and time to live (TTL) 132 that comes from any source and port address and goes to any destination address and ports:

Alert udp any any -> any any (msg: ''Detect Malicious UDP Packets''; ttl: 132; content:|' 61 62 C2 48 60 AE 97 4F 4B C3 63 64 65 66'|; Sid: 100004;).

Flood traffic TCP/IP was sent in different bandwidths (Bps) with 255 malicious UDP packets (threads) in interval packets with a delay of 1 microsecond (1 mSec). The NIDS rule was set up to check the pattern inside the packets and then detect only the malicious UDP threads when the two conditions of (TTL and content) are matched.

As shown in Table 1 and Figure 1, Snort NIDS initially analysed every packet that reached the wire. When 255 malicious UDP packets were sent at a speed of 1 mSec with TCP/IP flood traffic at 16 bytes per second (16Bps), Snort alerted and logged more than 99% of the total UDP packets that it analysed. As the flood traffic (speed) was increased to 200, 1200, 4800 and 60000 bytes per second (Bps), Snort alerted and logged packets to a decreasing degree, respectively, at 98.84, 97.17, 49.40 and 35.75% of the total malicious packets analysed (see Table 1). Figure 1 shows that the

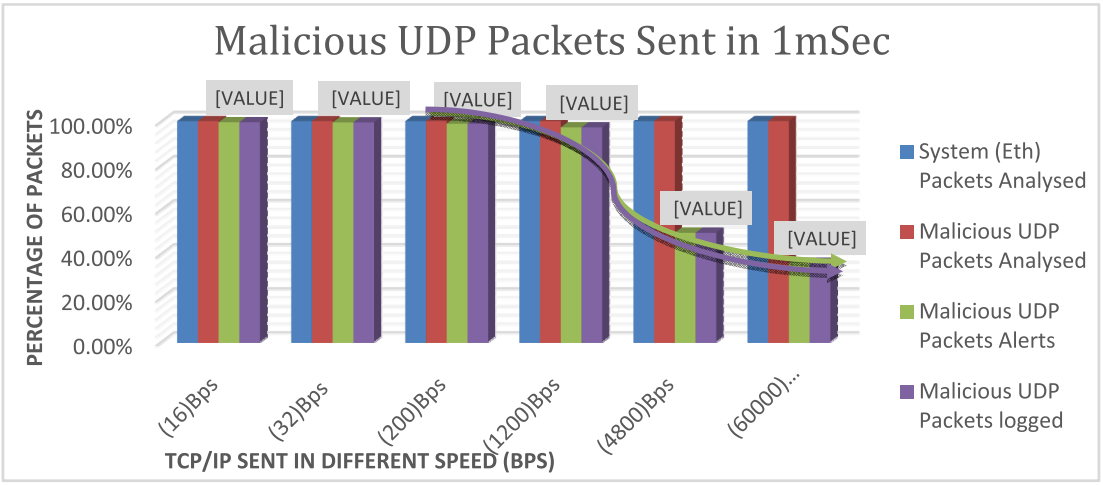

FIGURE 1. Malicious packets detection.

TABLE 2. Snort-NIDPS reaction to prevent malicious packets.

flood traffic(Bps) with 255 UDP malicious packets in (1mSec)	Number of packets analysed	Eth packets received of packets analysed	Ip4 analysed of Eth packets analysed	ICMP packets analysed	TCP packets analysed	UDP malicious packets analysed	UDP malicious packets reject	% Malicious packets prevent
100 Bps	267032	100.00%	89.066%	28	995	236795	236795	100.00%
1000 Bps	266863	100.00%	99.991%	7	3572	263260	263260	100.00%
10000 Bps	329926	100.00%	99.988%	522	114260	215104	108107	50.258%
60000 Bps	335143	100.00%	99.992%	784	147518	186814	32812	17.564%

number of missed malicious packet alerts increased when the speed increased. The experiment shows that, when the speed was 60000 Bps, Snort detected less than 36% of the malicious packets analysed (see Table 1).

B. PREVENTING MALICIOUS PACKETS

In this experiment, TCP/IP flood traffic was sent at differing speeds (see Table 2) with 255 malicious UDP packets (threads) also sent at 1 microsecond (1 mSec) intervals. Snort was set to prevent UDP threads by using two rule conditions (TTL and content) as follows:

reject udp any any ->any any (msg: "Prevent Malicious UDP Packets"; ttl: 120; content:|' C2 48 60 AE 97 4F 4B C3 '|; Sid: 100007;).

Use of these options will prevent any UDP malicious packet that is matched with the TTL value equal to 120 and a data pattern inside the malicious packet with content ".H'..OK.". The hexadecimal number ('C2, 48, 60, AE, 97, 4F, 4B, C3'), which the rule contained, is equal to the ASCII characters ('., H′,,.,., O, K,.').

As shown in Table 2 Figure 2, When 255 malicious UDP packets were sent at a speed of 1 mSec and TCP/IP flood traffic at 100 bytes per second (Bps), Snort prevented 100% of the total UDP packets that it analysed. As the flood traffic (speed) was increased to 10000 bytes per second (10000Bps), Snort prevented less than 51% of the total malicious packets analysed (see Table 2). Figure 2 shows that the number of missed malicious packets increased when the speed increased. The experiment shows that, when the speed was 60000 Bps, Snort only prevented less than 18% of 100% of the malicious packets analysed (see Table 2).

III. PROPOSED SOLUTION

The results of the experiments described above in section (II) show that the NIDPS's performance decreases when faced with heavy and high-speed attacks. This section analyses the problem and then outlines a novel solution to increase NIDPS performance in the analysis, detection, and prevention of malicious attacks.

A. NOVEL NIDPS ARCHITECTURE

Critical analyses were done for the experiments presented in sections II(A) and II(B) (see Figures 1 and 2, respectively). The figures show that performance of NIDPS throughput is affected when NIDPS is exposed to a high-volume and speed of traffic; more packets will be dropped and left outstanding as the speed of traffic increases. Figure 1 shows that the NIDPS's detection performance decreased when the traffic speed increased. There were more missed alerts and missed logs for packets as the speed of traffic increased.

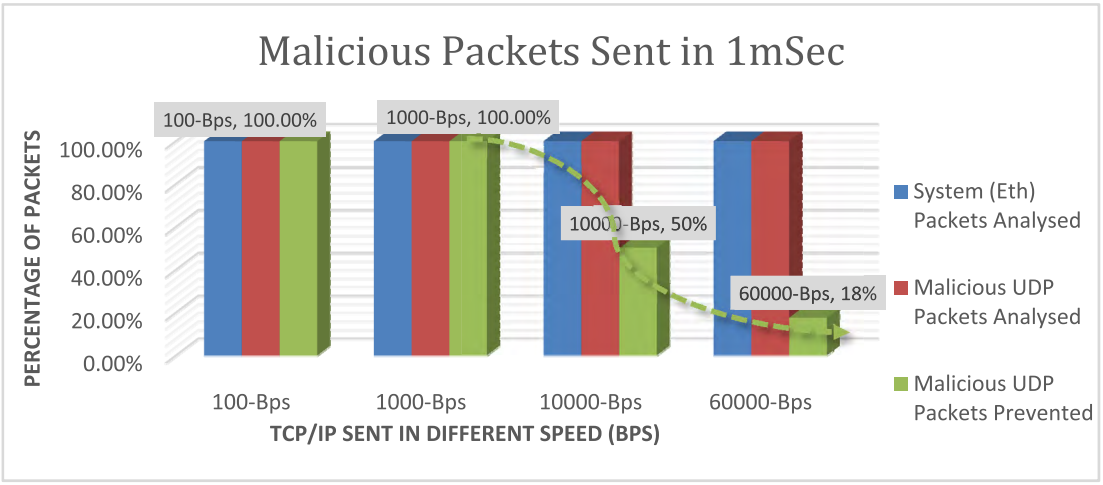

FIGURE 2. Prevent malicious packets.

Figure 2 shows that the NIDPS prevention performance decreased when traffic speed increased.

When traffic moves through the network interface card (NIC) to the NIDPS node, the packets are stored in the buffer until the other relevant packets have completed transmission to processing nodes. In the event of high-speed and heavy traffic in multiple directions, the buffer will fill up. Then packets may be dropped or left outstanding [13]–[15]. In this case, there is no security concern about the packets dropped; the packets are dropped outside the system. The existence of outstanding packets that are waiting or have not been processed by a security system (i.e. NIDPS node) affects the system efficiency however.

Packets can also be lost in a host-based IDPS. Most software tools use a computer program such as the kernel, which manages input/output (I/O) requests from software and decodes the requests into instructions to direct the CPU's data processing. When traffic moves from the interface (NIC) through the kernel's buffer to the processor space, where most of processing nodes are executed, the packets will be held in the kernel buffer before being processed by the CPU. When some nodes experience a high-volume of data, the buffer will fill up and packets may be dropped.

There are therefore three (3) places where packets could be dropped: in the network, in the host or in the processor, because all of them are dependent on buffer size and processing speed. If the arrival packet speed rate (λ) is greater than the network or host buffer speed rate (β), dropped packets (λ_d) may occur (see Figure 3), and even increasing the buffer speed can affect processor speed and cause packet drop.

For network-based packet loss, the NIDPS node fails to analyze this traffic (packets) because the network drops packets and the node cannot see them. Packet loss has no negative impact on the node's ability to detect or prevent received malicious packets, but it does have an impact on the receiving system in that useful packets would not be delivered. In host-based and processor-based packet loss, the NIDPS node has analysed this traffic because these packets have reached the

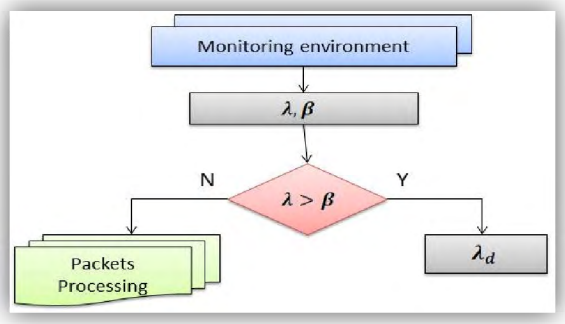

FIGURE 3. General model of buffer packets drops.

host system but the NIDPS node has not been able to process them. Therefore, useful packets can be lost. In order to solve the problem of lost packets in NIDPS, our study investigated the use of a QoS configuration in layer 3 switches with parallel NIDPS technology to organize and improve the processing performance. Our study develops a novel QoS architecture (see Figure 4) based on a layer 3 network switch.

A layer 3 switch enables a network to get the best performance effort from a network traffic delivery system. Through it, packets of various priorities can be delivered on a network in a timely manner. When networks experience high-speed and heavy traffic, each packet has a similar chance of being dropped or modified.

Implementing QoS methods, such as queueing, memory reservation, congestion-management, and congestion-avoidance techniques, can yield preferential treatment to priority traffic according to its relative importance. Furthermore, QoS technology ensures that network performance is more predictable, and that bandwidth utilization is more effective. QoS is used to improve performance in high speed network events [16]–[18]. QoS can be configured on physical interfaces such as ports and switch virtual interfaces (SVI) [16, pp. 294–299], [19], [20], [21, p. 826].

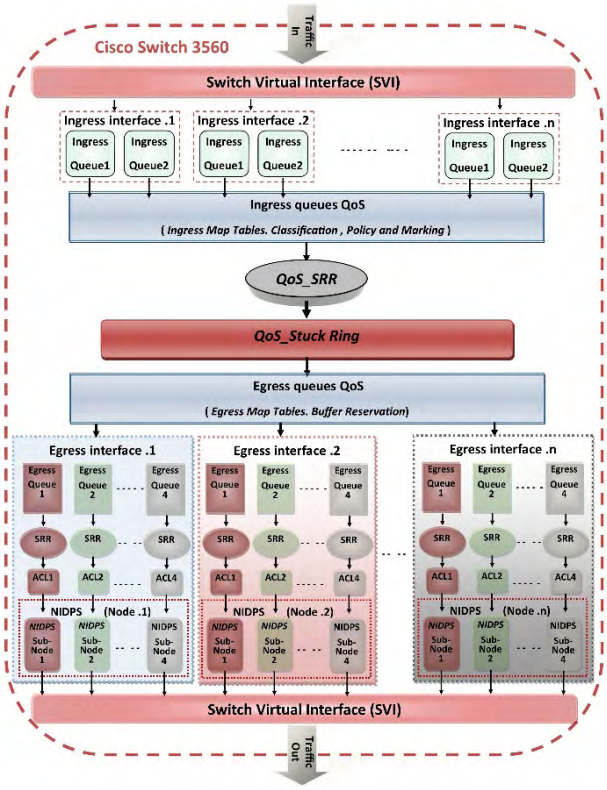

FIGURE 4. Novel NIDPS architecture.

In our study, QoS has been used to configure a novel architecture in order to improve overall network traffic and security performance. As shown in Figure 4, the system (switch) interface has been configured to have two input queues and four output queues. The queues' parameters were configured to allow queues to process traffic as a group of bytes. These load a set of packets equally among the queues and divide traffic into parallel streams in order to increase the rate of packet processing. The system then uses parallel NIDPS nodes to increase the NIDPS throughput performance and analyses each egress queue separately to determine whether it is free of malicious codes.

A class map and a policy map were made for each input queue. The class map recognizes and classifies a certain type of traffic for each input queue, while the policy map controls and organizes the speed limit for each input queue and applies the limit to all interfaces. The bandwidth, threshold, buffer, memory reservation, and priority (queue and traffic) were configured for all ingress and egress queues to treat and control traffic in order to help prevent congestion or complete failure through overload.

One queue was configured as an expedited queue. It received prioritized QoS services and other queues were not serviced until the bandwidth of prioritized queue reached its limit. A memory buffer reservation technique was implemented on our novel QoS architecture for each queue to guarantee that each queue's buffer could attain more space once it reached its limit. This was achieved by reserving space from an available queue buffer, from the SVI, or port interface memory buffers, or by switching to a common memory pool buffer.

ICMP, TCP, and UDP packets as well as malicious packets have different characteristics and require different processing. The Shaped or Share Round Robin (SRR), threshold, and priority methods for each output queue offer opportunities to manage differently various packet types and behaviors. For example, when all input and output queue buffers are flooded with traffic, priority queue and threshold map values can deny buffer overflow [19], [20].

The main idea of our novel QoS architecture is to manage and allocate a specific traffic weight, or set of bytes, into each input queue and process each output queue individually in parallel, thereby increasing NIDPS processor speed and reducing traffic congestion, even if the traffic is high-load and high-speed. The next section (B) gives more detailed information about our QoS configuring methods, i.e., class and policy map, ingress and egress queues, SRR, bandwidth, threshold, buffer, queue memory reservation and priority.

B. THEORETICAL AND TECHNICAL BACKGROUND OF NOVEL ARCHITECTURE

NIDPSs process packets which are carried by IP protocols, e.g. UDP, TCP and ICMP. The IP protocols are checked by NIDPS rules based on a signature database (known signature/attacks). However, to get the best NIDPS performance, the NIDPS should be implemented in a system which can manage the layer 3 network protocol (IP layer). In our study, a layer 3 switch has been used to support and improve NIDPS performance. The switch supports QoS configuration as well as Differentiated services (DiffServ) architecture.

1) MAPPING TO LAYER 3

Most of the switches work in layer 2, which is the data link layer. The switches use the class of services (CoS) value (see Figure 5), which enables differentiation of the packets [16], [21, pp. 827–828]. However, layer 2 provides insufficient methods to support switch features such as QoS features, dynamic access control lists (ACLs), VLAN features, static IP routing, and policy-based routing (PBR) [5], [21], [22].

Other mechanisms operate at OSI (Open Systems Interconnection) model layer 3 where DiffServ architecture can implemented (see Figure 5). For example, DiffServ allows different types of services to be offered depending on a code [18]. DiffServ allows a policy that gives priority to a certain type of package [16], [21, pp. 827–828]. DiffServ architecture is the basis for the QoS implementation in this research. It assigns each packet a classification upon entry that states its priority and its likelihood of being delivered into a network before packets are distributed. It adjusts each packet for different traffic speeds to ensure timely delivery.

Figure 5 illustrates the relative layers at which CoS and Differentiated Services Code Point (DSCP) operate. It also illustrates the relevant models (i.e., TCP/IP, Proto-

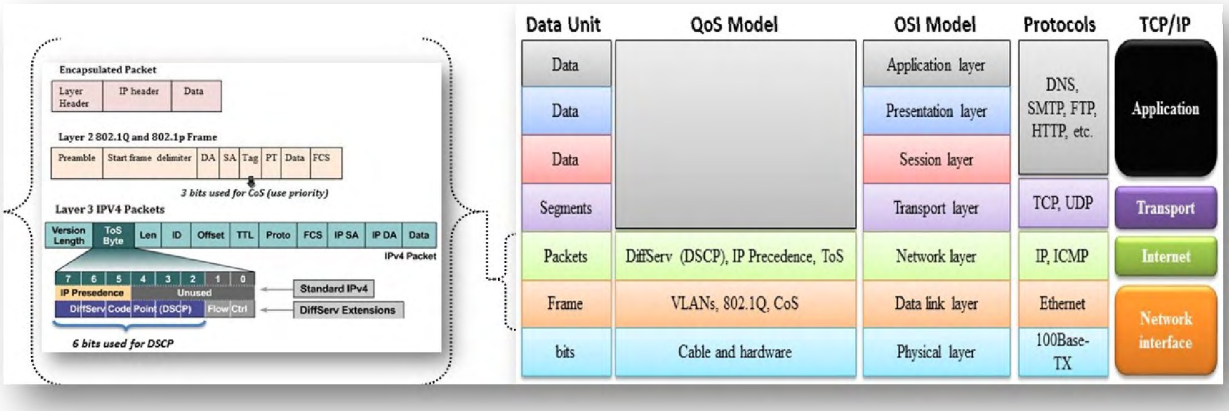

FIGURE 5. Positioning of CoS and DSCP values.

cols, QoS, and Data Unit) based on OSI layers [16, p. 191], [21, pp. 827 and 830–833].

Setting the type of service (ToS) field in the IP header can be used to achieve a simple classification which can be carried with the packet across the network [16:p32,21:p830-833]. In layer 2, 802.1Q and 802.1p frames use 3 bits (see Figure 5) for the IP ToS field; in layer 3 IPV4 packets use 6 bits (see Figure 5) for DSCP in the ToS field to carry the classification information. Regardless of a network's capability to identify and classify IP packets, hops can offer each IP packet a QoS service.

In the configuration method utilized by this research, the first action changes the switch frame from layer 2 to layer 3 by mapping values from CoS to DSCP. We considered DSCP to be the best choice for the intended usage because differentiated services technology can offer more precise handling of traffic on the network, can classify each packet upon entry into the network interfaces, and allows adjustments to be made for different traffic speeds and loads. The mapping action between values determines the delay priority of the packets. CoS has 8 values and DSCP has 64 values. Thus, the DSCP values allow for a higher degree of differentiation [5], [16].

2) QOS CLASSIFICATION AND POLICY METHODS

Classification is the process of identifying the data packets to a class or group in order to manage the packet appropriately [16]. QoS features such as a policy map and class map can be used to achieve this. The class information can be assigned by switch, router, or end host. Policing involves creating a policy that defines a group weight (the number of bytes to be processed together) for the traffic and applies it to the interface. Policing can be applied to a packet per direction and can occur on the ingress and egress interfaces. Different types of traffic can be recognized in terms of type, and ports and differentiated policies can be set accordingly.

Network QoS technology enables the implementation of a new logical and throughput-traffic-forwarding plan in the switch. For the purpose of this research, a physical interface was configured to two input queues and four output queues (see Figure 4). This configuration helps to prevent congestion traffic (which would cause buffer overflow) and helps to improve buffer throughput performance. A buffer was set for each queue and a memory reservation method using a dynamic memory reservation technology was implemented in order to manage higher traffic loads. After packets were placed into input queues, class and policy maps were implemented to handle packets based on their QoS requirements. Appropriate services were then provided, including bandwidth guarantees, thresholds, queue setting, and priority servicing through an intelligent ingress and egress queueing mechanism.

The class map information is assigned along the path of a switch and can be used to limit the volume of incoming packets distributed to each traffic class. The default behavior in layer 3 switches using the DiffServ architecture is the ''per-hop'' method [16, pp. 6 and 940–941], [21, p. 828]. If a switch along the path does not provide a consistent behavior per hop, QoS provides a conceptual and constructed solution, such as an end-to-end queue solution. The solution is based on a configurable policy map that allows the system to examine packet information closer to the edge of the network, which prevents the core switch from experiencing overload. The output queues are processed individually where parallel Snort NIDPS nodes are implemented. Each output queue has own NIDPS node (see Figure 4).

3) PARALLEL TE CHNOLOGY WITH QOS.

Parallel NIDPS is a form of computation where many NIDPS nodes work simultaneously, operating on the principle that the large incoming data can be divided into smaller sets, which are processed at the same time. Parallelism of NIDPS

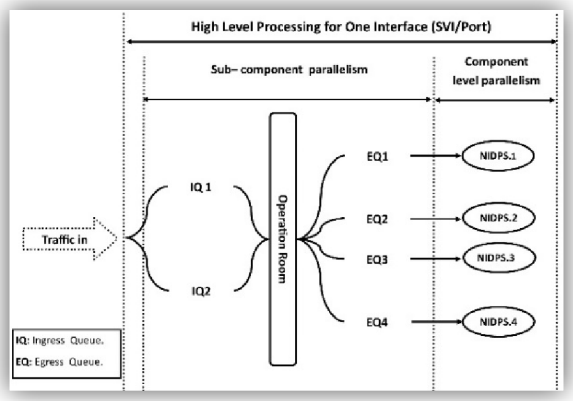

FIGURE 6. High level parallel process for one interface.

can occur at three general levels: the high-level processing node (entire system), the component level (specific tasks are isolated and parallelized) and the sub-component level parallelism (function within a specific task) [23]. The handling of data can also be parallelized with traffic being split into separate streams to be processed by parallel nodes or components. This is data parallelism which can occur in various ways with the three general levels of parallelization. In our architecture, parallel management of traffic was implemented through the use of queues (2 input queues and 4 output queues) on an SVI where component level parallelism of NIDPS nodes was implemented (see Figure 4 and Figure 6).

The parallelization of data (traffic) that was distributed through ingress and egress queues into critical and non-critical is viewed as multiple traffic parallelism (MTP). Critical pre-processing of traffic is performed on queues to create particular groups of packets (threads) before the traffic is examined by an ingress queue algorithm. Non-critical pre-processing occurred after the packets had been matched to ingress and egress queues policies. The NIDPS node component can be parallelized in an either non-functional or functional manner. Component level parallelism is defined as function parallelism of the NIDPS processing node. Individual components of NIDPS were isolated, and each output queue was given its own processing element (see Figure 4 and Figure 6). The NIDPS node was configured from a single-node NIDPS to a multi-node NIDPS. Each node was configured to check for a certain type of packet (e.g. UDP, TCP and ICMP) and was able to access discrete parts of a centralized, common rule base to order to carry out its task. The kernel buffer parameters for each NIDPS node was configured as each output queue rate.

4) QOS CLASSIFICATION, POLICING AND MARKING FOR INGRESS AND EGRESS INTERFACES (QUEUES).

Queues, class, and policy technologies can use access control lists (ACLs) to allow the processing management of different types and patterns of incoming and outgoing

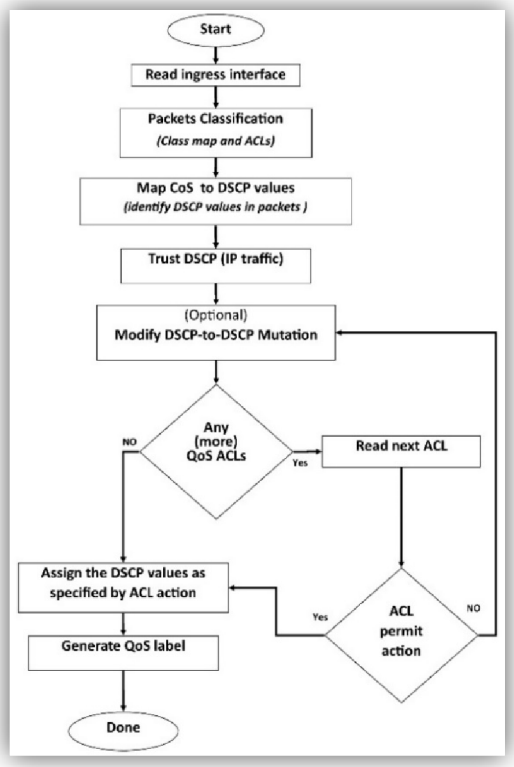

FIGURE 7. Packets classification and marking.

packets [22, p. 1]. The novel configuration proposed in this paper uses an ACL technology with a class map and SVI queues, as well as a policy map that specifies each type of IP traffic (e.g., ICMP, TCP and UDP) to be processed by implementing parallel output queues with associated parallel NIDPS nodes.

When traffic arrives at the ingress interface of system, packets will be classified through a class map (see Figure 7) that will enable packets to be processed as a group of bytes defined by a policy and ACLs that were matched with DSCP values. A policy map (see Figure 8) was made to specify required action for each class. The following procedures constitute the method:

- Classify the traffic with a class map for SVI and ports. Set ACLs rules depending on the kind of traffic/attacks to be detected or prevented. In our experiment, we detect and also prevent UDP malicious packets which came with random high-speed flood traffic. We allowed UDP traffic to be processed in a separate egress interface (queue) and then analysed by a parallel NIDPS node. The other traffic (e.g. TCP, ICMP, etc.) was processed in the other egress queues.
- Organize a rate-limit for the system ingress interface processing speed (Setting a set group of packets in bytes) for the class traffic. The rate depends on the maximum limit of SVI bandwidth including memory. In our system we set "1.124 million" bytes (nearly 1Gb of packets) for

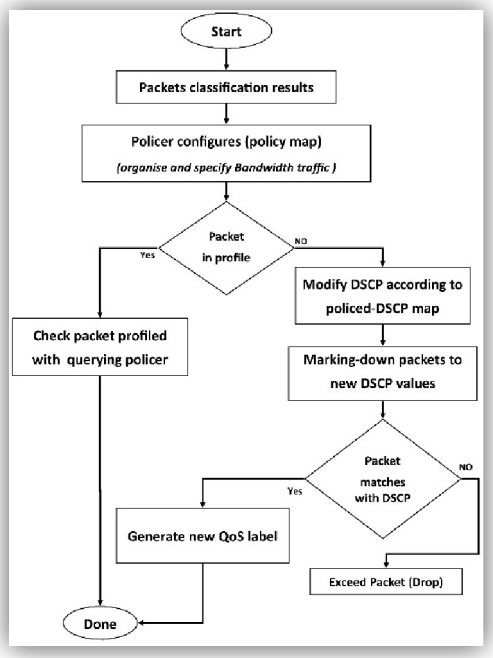

FIGURE 8. Packet policing and marking.

the set of classes because the maximum limit for each interface in our system is 1Gbps.

- Each class of traffic matches to a DSCP value so packets in that class are marked down to a new DSCP label.

After packets were classified and policed with a specific bandwidth, some were dropped out of the profile (fabric). Each policy can specify what actions should be carried out [16], [21, p. 833], including dropping packets, allowing dropped packets to be modified, allowing packets to pass through without modification, and deciding on a packet-by-packet basis whether a packet is in or outside the profile. The novel QoS policy map architecture (see Figure 8) was proposed as follows:

- Packets dropped were modified to be re-processed again and mapped with new DSCP values based on the original QoS label. This modification helps for correcting and reducing dropped packets inside the profile.
- When packets are reprocessed, they may get out of order. To prevent this, a policy was designed to allow packets to be re-processed in the same queue as the original QoS label.
- The system has the ability to mark up a limit speed (as a set of bytes) for each input queue.
- If packets are not matched with DSCP values, packets will be dropped. See Figure 8 for an illustration of the architecture.

A hierarchical policy map was created and applied to the traffic inside the ingress queues. The policy map targeted SVIs and ports. Two types of QoS policy were created: individual and aggregate. Individual QoS applies a separate policy to specify a bandwidth limit for each traffic class. Aggregate QoS specifies an aggregate policy with which to apply a bandwidth limit to all matched traffic flows. The individual policer only affects packets on a physical interface (i.e., SVI/port). Furthermore, if more than one type of traffic needs to be classified, it is possible to create more ACLs, class maps, and policy maps [16]. In our experiments, three types of traffic (TCP, UDP, and ICMP) were classified using ACL, class map, and policy map methods.

Switches receive each traffic frame in a token bucket [16, p. 62], [21, p. 835], where an algorithm is used to check leaks of data transmissions. The token bucket processing speed is set at the same rate as the configured average packets rate and conforms to defined limits on bandwidth to allow a burst of traffic for short periods. Each time a token is added to the bucket, the algorithm checks to see if sufficient room is available in the bucket. If not, the packet will be marked as non-conforming, and the specified policy action will be taken. In our QoS architecture (see Figure 8), packets dropped out of profile were marked down with new DSCP values and the DSCP value was modified to generate a new QoS label.

5) QOS THRESHOLDS FOR INGRESS AND EGRESS INTERFACES (QUEUES).

QoS stores packets in input and output queues according to the QoS label, which has been defined and identified by the DSCP values in the packets. Threshold map values can be assigned to the queues [16, p. 260], [21, p. 838], [22]. Our architecture has employed weighted tail drop (WTD) thresholds on ingress and egress queues to cope the bandwidth length of each queue and deliver the drop precedence for different classifications of packets. If the available space on a destination queue is less than the volume of packets, the threshold is exceeded for that QoS label and the switch drops the packet. DSCP values (which are carried by packets) were mapped to ingress and egress queues which all have a located buffer space. WTD thresholds for input and output queues were set (see Figure 9). Each queue has three drop thresholds. This means that different thresholds can be set for different types of packets. Each value of the threshold represents a percentage of the queue's total buffer.

In our configuration, each ingress queue was assigned a various threshold value. One of the ingress queues was assigned to be a high-priority queue with a maximum queue threshold (the queue can hold up to its limit of frames at up to 100% threshold). The other ingress queue was assigned to be a lower-priority queue with a lower queue threshold (threshold $<100\%$). Higher priority packets were directed to the high-priority queue.

6) QOS BUFFER RESERVATION FOR INGRESS EGRESS INTERFACES (QUEUES).

Buffers are universal throughout the software and hardware layers of any network computer system. They are valuable in reducing the impact of traffic rate variability on the network especially in the case of traffic rate points. By having

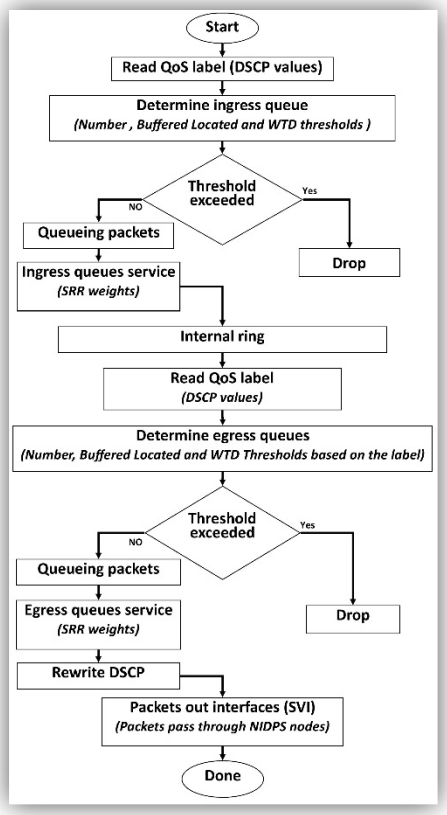

FIGURE 9. Novel scheduling architecture for ingress and egress queues.

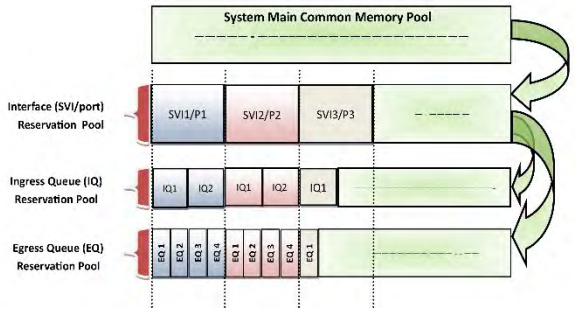

FIGURE 10. QoS buffer reservation.

sufficiently large buffers to absorb traffic rate spike, high latencies associated with retransmissions and lost data can be avoided. They are also useful if there is a temporary difference in the rate at which traffic is produced and consumed. However, increasing the buffer size cannot compensate for packet processing that is perpetually slower than the packet arrival rate. Systems (e.g., switches, routers, etc.) may have different buffer configurations. The total rate of all buffers is β; and each ingress and egress buffer of an interface is limited to rate α. The same α applies to all interfaces. The rate of a buffer is the speed at which packets move through it and this depends on the underlying processing system.

The switch manages buffering across a number of interfaces. An ingress interface has ingress buffers connected to common egress buffers. The switch algorithm is also responsible for scheduling. At each event of scheduling, the switch algorithm selects one of the ingress buffers. The packet at the head of the selected buffer is then transmitted to the inside at the switch and via the egress buffer to the target system. There are some formulations that model the entire switch rather than just one interface [24]. For example: a switch consists of m ingress and n egress interfaces, where each interface has buffer. Arrival events (packets) arrive at the ingress interfaces (which have specified destination egress interfaces). At the scheduling event, packets at the top end of the ingress interface buffers are sent to the egress interface buffers. Here, the switch algorithm matches the packets in the ingress and egress interfaces. According to this matching, the packets in the ingress interfaces will be transmitted to the corresponding egress interfaces. In this scheduling task, there is also a buffer for each pair of ingress and egress interfaces. Thus, there arises anther buffer management problem at scheduling phases. In the implementation of QoS architecture proposed in this study, QoS DiffServ was used to assign a value to each packet according to its importance and then it decides the order of packets to be processed through queues based on the value of packets. Additional buffers were provided dynamically to the ingress and egress interfaces. A QoS switch was used to control the input and output traffic. A priority queue was implemented for one of the ingress queues in the interface.

By default, the ingress buffer rate is the same as egress buffer rate. However, when the arrival event of traffic rate (λ_{in}) is more than total rate of egress buffers, or one of n egress buffers already reached α, the packets would be dropped ($\lambda_{\text{d}} > 0$). In the novel configuration, the sharing policy was configured for each ingress-queue's buffer which corresponds to rate $\frac{\alpha}{2}$ and the egress-queue's buffer was to rate $\frac{\alpha}{4}$, where α is the maximum rate for the interface's buffer. All buffers were assumed to have the same maximum rate.

A buffer reservation technique (see Figure 10) has been used to increase buffer size along with implemented parallel nodes of buffers to increase buffer speed performance. The total buffer memory space was provided by the system main common memory pool with subdivisions for the SVI pools and further for ingress queue pools and egress queue pools (see Figure 10). A buffer distribution scheme was implemented to reserve more space for each egress buffer. Thus, all buffers cannot be consumed by one egress queue, and the system can manage matching buffer space to queue demand. The remaining free common pool interfaces were set to reserve up to 50% of the available switch memory pool.

A switch identifies whether the target queue has consumed less buffer space than its reserved volume (under-limit), whether the target queue has consumed all of its reserved amount (over-limit), and whether the system (switch) memory pool has free buffers. If the queue is not over-limit, the switch can reserve a buffer space from the interface

pool or from the switch main common memory pool. If no more space is available on the common pool or if a queue is over-limit, the switch drops the packet.

7) QOS SHAPED/SHARED ROUND ROBIN (SRR).

After traffic has been classified, marked, and policed in two ingress queues, each packet is processed into four output queues that implement parallel NIDPS nodes. QoS also offers SRR technologies, which can vary the interface queue bandwidth, and control the buffer rate at which packets are sent [16, p. 260], [21, p. 840]. The Shaped function SRR can guarantee each queue a bandwidth limit, but queues cannot share with each other if one or more queues reach their bandwidth limits. The Share function SRR can guarantee a bandwidth limit for each queue, and the other queues can share with each other if one or more queues reach their bandwidth limits. This research utilized Share in the ingress queues and Shaped in the first three egress queues. In the fourth egress queue, the Share mode was set to allow the queue to share traffic with other available output queues.

Queue technology is placed at specific points in a system (e.g., Layer 3 switches) to help prevent congestion. The total inbound bandwidth of all interfaces may exceed a ring space of internal bandwidth. After packets are processed through classification, policing, and marking, and before packets pass into the system (switch) fabric, the system allocates them to input queues. Because multiple input queue interfaces can simultaneously send packets to output queue interfaces, ingress packets exit to an internal ring, and outbound queues are allocated from the internal ring. This avoids congestion. The SRR ingress queue transmits packets to the internal ring, while the SRR egress queue sends the packets to the output link.

The novel configurable architecture has a large limit of buffer space and a generous bandwidth allocation for each queue. One of the ingress queues was set as a priority queue, which allowed the system to priorities packets with particular DSCP values and thereby allocate a large buffer. It also allows buffer space to be used dynamically, and then adjusts the thresholds for each queue so that packets with inferior priorities are dropped when queues are full. This allows the system to ensure that high priority traffic is not dropped.

When the traffic comes to the interface, the packets are buffered in the priority ingress buffer (priority queue) and if the priority buffer is getting full, the traffic will transmit to the second ingress buffer. If all ingress buffers are getting full, the packets will be dropped, or the switch can reserve another ingress buffer with the same priority up to n, where n is the maximum number of ingress buffers.

When the arrival packets pass through ingress buffers, the traffic speed was re-controlled as an interface speed limit (μ) (group of bytes per second) and the traffic is stored in the SVI QoS-ring space, where packets were arranged and managed to exit the interface through egress queues. μ should be equal to the maximum bandwidth limit of each system interface.

Every arrival packet needs to be sent out of an ingress interface and then placed in egress buffers which permit an interface to hold packets when there are more packets to be transmitted than can physically be sent (experiencing congestion). If the system (switch) cannot allocate sufficient buffers to hold all incoming traffic, the packets will be dropped. Availability of egress buffers determine if a packet is transmitted or not. When it comes to reducing packet drop, the system does not concern itself with packets. Rather, it is concerned with the number of requested (reservation) buffers to which unbuffered packets can be added. The volume of egress buffering differs from platform to platform. Typically, two (2) reservation pools are available for layer 3 network switches. These pools are the SVI reserved pool and the switch memory common pool (see Figure 10). The switch memory common pool is used when the SVI reservation pool has previously been consumed.

When packets (traffic) go through the output queues, the switch reserves buffer from the SVI reservation pool for all egress buffers and then if one egress buffer is fully consumed, the consumed egress buffer can reserve buffer space from the available buffers of other egress queues. When the SVI reservation pool is consumed by all egress buffers, it reserves more buffers from the switch common memory pool. If all the switches' buffers are consumed, the packets will be dropped because no more space will be available.

All ingress and egress buffers above are collectively called one node of the switch's buffer (η_i). The common memory switch pool is the ''holdup'' storage area, where packets are held until they can be processed. If the holdup area is full, more reservation buffer can be achieved by reserving memory from another switch memory pool in the LAN (see Figure 11) and even can be from WLANs. However, all egress buffers were controlled to limit rate β_k(kernel buffer speed) to prevent host based dropped packets. The kernel buffer rate (speed) should be equal to output queue rate.

C. SUMMARY OF PROPOSED SOLUTION

NIDPSs are often unable to detect or prevent all unwanted traffic or malicious activities when traffic comes in at high-speeds and volumes. As a solution, this paper describes a novel architecture. Layer 3 Cisco switch technology is combined with parallel NIDPS nodes, to create queues with specific buffer and bandwidth sizes.

The system thus increases queue buffer size automatically up to a network limit. It also services buffer space from an available queue buffer, port buffer, or switch pool memory buffer to hold more packets. This allows the system to organize and increase the processing speed of arriving packets (which have been reconfigured and reordered as groups of bytes) by setting a number of parallel egress queues to be processed by parallel NIDPS nodes. The number of parallel processing NIDPS nodes needed in any particular system depends on network arrival rates. Therefore, it was necessary to operate with the class and QoS technologies within the network switch.

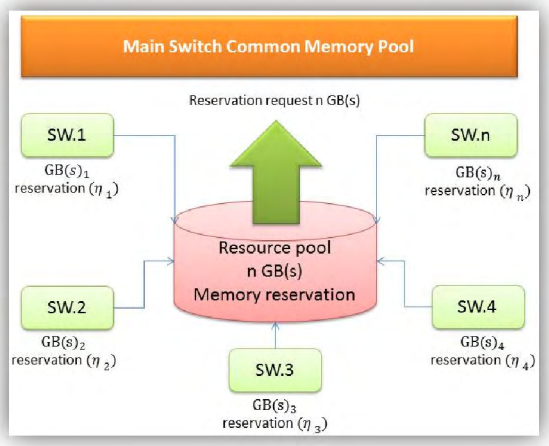

FIGURE 11. Reservation buffer from n node of switches.

An assumption is that there will be an underlying parallel implementation of the target destination (NIDPS in this case) and for each egress buffer commissioned there will be a port to a parallel node of the target system. This enables better performance and higher volumes of traffic to be processed successfully. The difference between the previous studies [5], [12] is that this study gives a clear picture of how QoS architecture along with parallel technology can improve NIDPS performance. The QoS configuration boosts the NIDPS performance with regard to its congestion management and its congestion avoidance. Congestion management creates balanced queuing by evaluating the internal DSCP and determining in which of the four egress queues to place the packets.

Other items related to queuing are also configured to reduce dropped buffer packets in interfaces in order to improve NIDPS performance, e.g., defining the priority queue, defining a queue set, guaranteeing buffer availability, limiting memory allocation, specifying buffer allocation, setting drop thresholds, mapping the CoS to the DSCP value, configuring SRR, and limiting the bandwidth on each of the outbound queues. The congestion avoidance method also helps with the performance of the NIDPS, by, e.g., setting output queuing, and configuring WTD parameters for the ingress and egress queues. The use of parallel NIDPS nodes to match each of system egress queues enables NIDPS packet checking to keep up with increased arrival rates typical of an attack.

IV. EVALUATION OF THE PROPOSED SOLUTION

This section presents an evaluation of the proposed architecture.

A. EVALUATION OF NOVEL NIDPS ARCHITECTURE

The experiments that were described in section II are repeated, but here the novel architecture is implemented to test performance in terms of throughput with the support of the proposed solution (QoS and parallel technologies). Each experiment tested Snort NIDPS throughput when analyzing traffic such as TCP/IP headers and then detecting or preventing unwanted traffic (UDP malicious packets) arriving at a high-speed.

As shown in Figure 1, when malicious UDP packets were sent at a speed of 1 mSec with different TCP/IP flood traffic at 16 to 60000 bytes per second (Bps), Snort NIDPS started effectively but overall it missed detecting up to 65% of malicious packets that system received (see Table 1). Furthermore, it was unable to prevent all unwanted packets. The experiment shows that, when the speed was 60000 Bps, Snort prevented less than 18% of the malicious packets analysed (see Figure 2 and Table 2). When QoS architecture was implemented, Snort NIDPS detected almost 100% of malicious packets that system received (see Table 3 and Figure 12). The experiment results show that Snort NIDPS performance increased greatly when QoS is used. It prevented almost100% of malicious packets that it analysed (see Table 4 and Figure 13).

B. EVALUATION AT HIGHER SPEED

In this section, two tests were carried out for NIDPS analytic performance under various high-speed traffic. The first test was for NIDPS with no novel architecture, and the second test was for our novel NIDPS architecture. In this experiment, TCP replay tool was used to generate traffic at different speeds (Gbps) through the system. The same system (Cisco Switch 3560) was used. As the results show in Figure 14, when we tested NIDPS (no novel architecture), Snort NIDPS processed every single packet that reached the wire. The results show that when packets are sent at 1 Gbps, Snort analyses nearly 100% of received packets, but when traffic speed increases, NIDPS starts losing/dropping packets (see Figure 14). Furthermore, when speed was 8Gbps NIDPS analyses just 599818 of 10994568 packets which is less than 6% of total packets received (see Figure 14).

When we implemented our novel architecture, Snort processed 100% of packets that were received while the traffic speed was 8 Gbps (see Figure 15). When the traffic speed was increased to 10 Gbps, Snort started to drop packets.

By using 2x 1Gb interfaces, the experiment results showed that the Snort NIDPS processed up to 8Gbps with 0 packets dropped, but without using our architecture, Snort dropped more than 94% of total packets that it received (see Figure 14) whereas when using the novel architecture, NIDPS dropped 0. However, successful processing of more than 8Gbps depends on the system capacity which always has some limit and cost. The total capacity of our system's main memory buffer is 32 Gbps. The system used (Cisco switch 3560) can hold up to 32 queues. Each queue can has up to 1Gbps buffer.

V. DISCUSSION AND RELATED RESEARCH

This section discusses the proposed solution and compares it to related research in parallelism in intrusion detection.

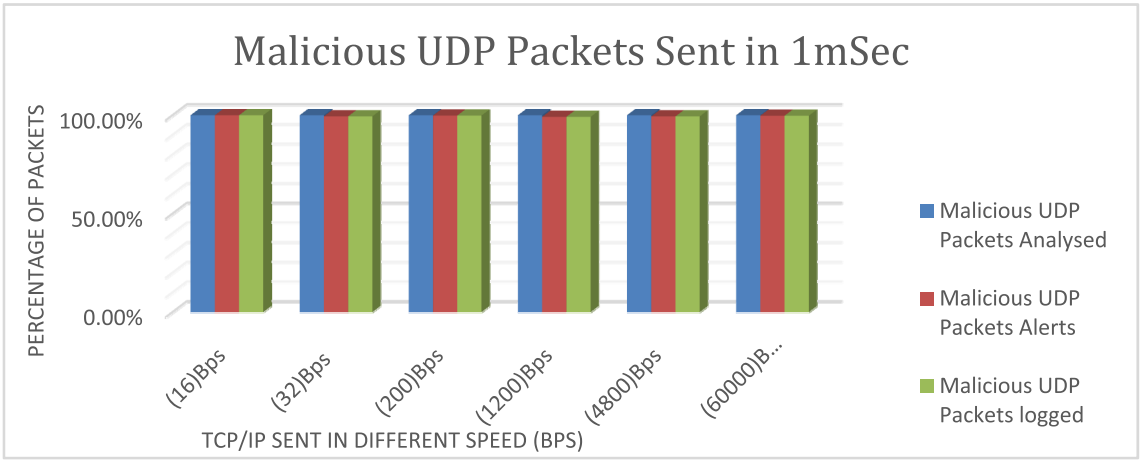

FIGURE 12. Detecting malicious packets in high-speed traffic.

TABLE 3. Novel NIDPS architecture reaction to detect malicious packets.

Packet sent (TCP/IP flood traffic (Bps) with 255 UDP malicious packets in (1mSec))	Eth packets received	Ip4 analysed of Eth packets analysed	ICMP packets analysed	TCP packets analysed	Malicious UDP packets analysed	Malicious UDP packets Alerts	Malicious UDP packets logged	% Malicious packets alert and logged
16 Bps	100%	99.174%	99	1680	999866	999866	999866	100.00%
32 Bps	100%	99.693%	105	4751	899338	894351	894351	99.44%
200 Bps	100%	99.899%	1511	200015	759092	757877	757877	99.84%
1200 Bps	100%	99.999%	1130	565025	433681	430081	430081	99.17%
4800 Bps	100%	98.376%	1003	799012	200995	199789	199789	99.40%
60000 Bps	100%	99.881%	1339	973755	27560	27491	27491	99.75%

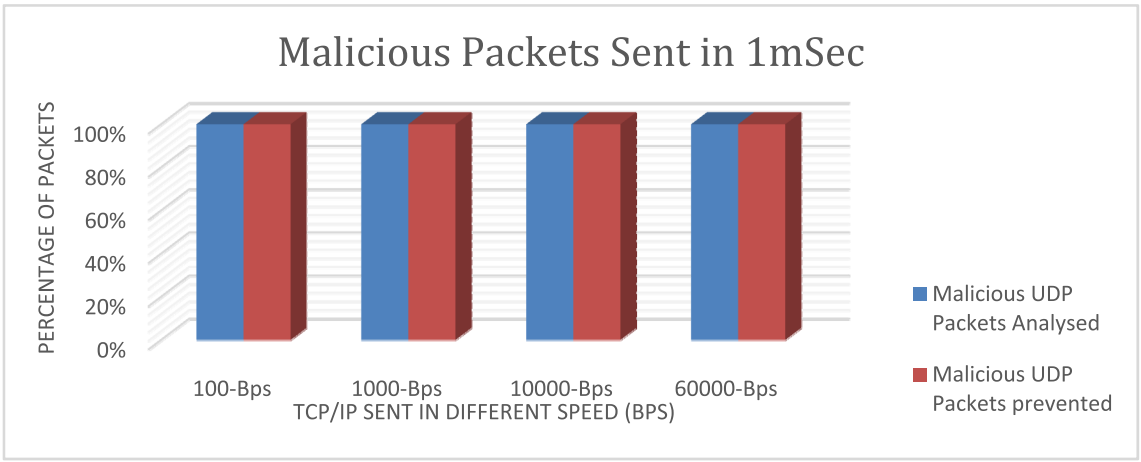

FIGURE 13. Preventing malicious UDP packets in high-speed traffic.

Vasiliadis *et al.* [25] proposed a new model for a multi-parallel IDS architecture (MIDeA) for high-performance processing and stateful analysis of network traffic. Their solution offers parallelism at a subcomponent level, with NICs, CPUs and GPUs doing specialized tasks to improve scalability and running time. They showed that processing speeds can reach up to 5.2Gbps with zero packet loss in a multi-processor system. Jiang *et al.* [26] proposed a parallel design for NIDS on a TILERAGX36 many-core processor. They explored data and pipeline parallelism and optimized the architecture by exploiting existing features of TILERAGX36 to break the bottlenecks in the parallel design. They designed a system for parallel network traffic processing by implementing a NIDS on the TILERAGX36, which has a 36 core processor.

TABLE 4. Novel NIDPS architecture reaction to prevent malicious packets.

flood traffic (Bps) with 255 UDP malicious packets in (1mSec)	Number of packets analysed	Eth packets received of packets analysed	Ip4 analysed of Eth packets analysed	ICMP packets analysed	TCP packets analysed	UDP malicious packets analysed	UDP Malicious packets reject	% UDP malicious packets prevent
100 Bps	1013836	100.00%	98.076%	228	262037	738999	738999	100.00%
1000 Bps	1502809	100.00%	99.891%	117	823198	678401	678400	99.999%
10000 Bps	1993125	100.00%	99.899%	522	1161022	830578	830578	100.00%
60000 Bps	2505935	100.00%	99.998%	384	1725890	779641	779641	100.00%

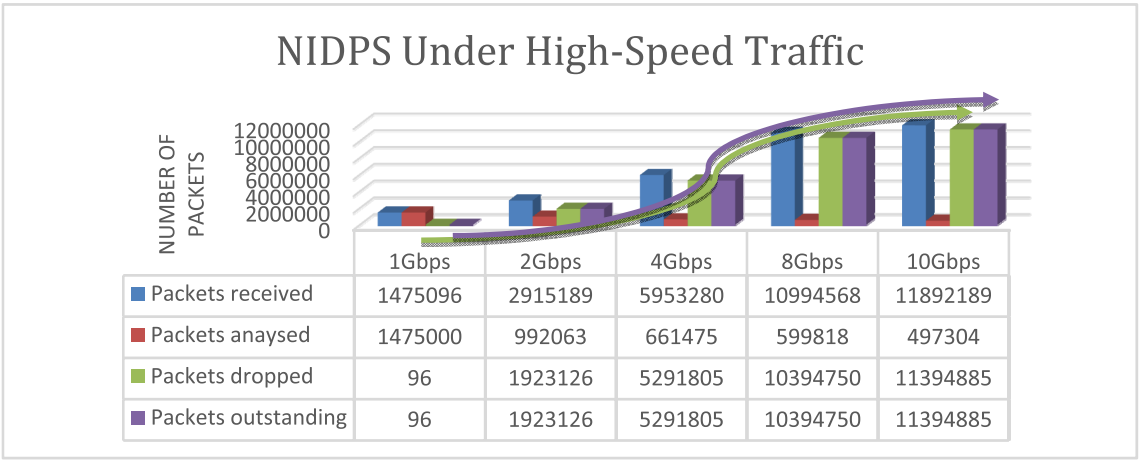

FIGURE 14. NIDPS at >= 8Gbps traffic speed.

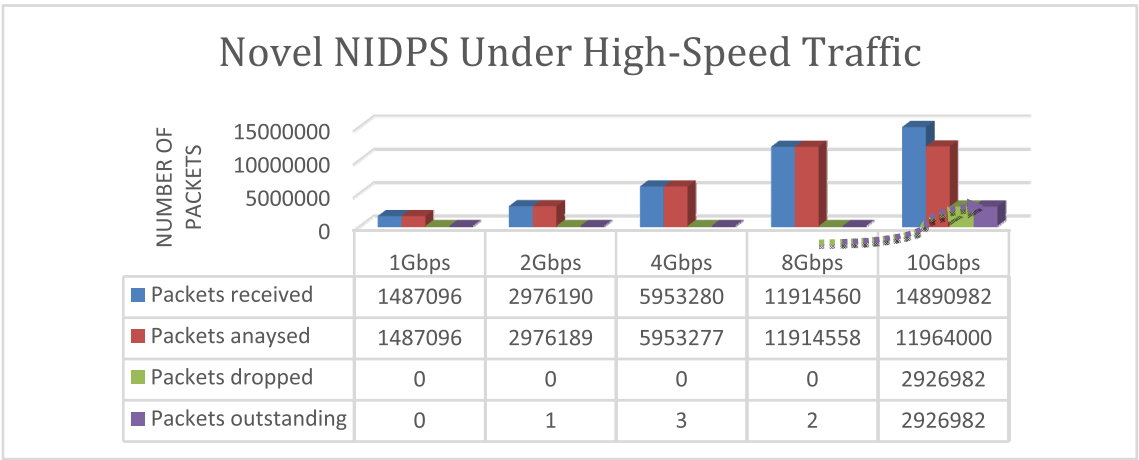

FIGURE 15. Novel NIDPS architecture at >= 8Gbps traffic speed.

The system was designed according to two strategies: first a hybrid parallel architecture was used, combining data and pipeline parallelism; and secondly a hybrid load-balancing scheme was used. They took advantage of the parallelism offered by combining data, pipeline parallelism and multiple cores, using both rule-set and flow space partitioning. They showed that processing speeds can handle and reach up to 13.5 Gbps for 512-bytes. Jamshed *et al.* [27] presented the Kargus system which exploits high processing parallelism by balancing the pattern matching workloads with multi-core CPUs and heterogeneous GPUs. Kargus adapts its resource usage depending on the input rate, to save power. The research shows that Kargus handles up to 33 Gbps of normal traffic and achieves 9 to 10 Gbps even when all packets contain attack signatures. The two approaches described in this paragraph are not directly comparable in terms of throughput as different numbers of processors are used in each. However, the experiments show that high gains can be made by parallelizing NIDPSs in order to combat problems of higher speeds and increasing traffic. Our research uses a multi-layer switch

along with parallel technology to improve packets processing performance which increases the ability to handle different speeds and data volumes. Further enhancements occur when queuing is combined with parallel processor technologies. The approach of this study has shown how parallelism at a higher level of granularity, which is simpler to implement, can also make impressive improvements for security performance in terms of throughput and the number of dropped packages. By using 2 machines connected to two interfaces, our NIDPS processed up to 8 Gbps with 0 drop for 1KB packets. This number can be increased up to 32Gbps which is the full system capacity forward bandwidth by implementing more nodes of NIDPS.

Chen *et al*. [28] proposed an application-specific integrated circuit (ASIC) design with parallel exact matching (PEM) architecture to accelerate processor packets speed. The ASIC hardware has been designed to operate at 435MHz to perform up to 13.9 Gbps throughput to manage the requirements of high-speed and high accuracy for IDS, which resolves the issue of the information security limitation to manage data received from the 10Gbps core network. They proposed SRA (Snort Rule Accelerator) with parallel rules to increase the performance of the IDS. The SRA is proposed with a stateless parallel-matching scheme to perform high throughput packet filtering as an accelerator of the Snort detection engine. The ASIC is composed of five major modules, namely the Inspector, Counter, Parallel Matching, Conformity, and Compare modules. The parallel matching scheme compares a packet's payload with the stored rule. When an entry packet is matched with Snort rules, the ASIC is in an idle state and sends a compare signal to the conformity module, which integrates all signals and determines whether an abnormal payload is presented. Here the authors designed a half mesh architecture in the parallel matching rules module, which allows the traffic to be compared with several rules. Our research addressed performance in a different way. It embeds an open source security software along with layer 3 switch technology (QoS, memory and buffer dynamic reservation and parallel queue technologies) to improve network forward-throughput-traffic architecture and, hence, security performance. It configured an interface into queues (interface-to-queues), which allows packets to be processed through the component level parallel NIDPS nodes. The approach is designed to deal with the limitation of real networks speed and finds solutions to the problems that caused the NIDPS performance. The approach can deal with any incoming traffic-speed that may allow malicious packets to enter the system and prevent NIDPS from detecting or preventing them. It does this by imposing advanced management of network packet traffic. The advantage of the proposed approach is that every-day equipment can be utilized in a new way to achieve improvements and it is also more scalable than the proposal of Chen *et al*. [28].

In the context of big data and distributed systems, Zhao *et al*. [29] have developed a security framework in G-Hadoop. This work focuses on authentication and access rather than intrusion detection but offers an interesting new direction. The framework could be enhanced with intrusion detection and protection functionality to create a more complete solution. Our research has focused on standard business infrastructure whereas the work of Zhao *et al*. has concentrated on single cluster across cloud data centers. Cross-cluster security services in a high performance environment such as that afforded by G-Hadoop is an area where attention is welcome.

Vendor companies are aiming to develop security solutions to protect the enterprise network. Equipment has been designed to meet connectivity speed and load standards. The improvements in the throughput of NIDPS shown in this research are achieved by pairing the ASA (Adaptive Security Appliance) Cisco equipment [10] with multiple implementations of Snort. The principles of the method proposed in this research could be applied to other equipment combinations where similar facilities are offered.

To summarize, our research differs from previous research in terms of the architecture used. The research investigates how QoS including DiffServ technology and parallelism can have impact in high-speed and heavy traffic networks using an industry standard switch and standard desktop processors. This solution is a more accessible way of receiving good results as it can be activated at a higher level, namely at the level of configuring the CISCO switch software and replicating Snort on standard machines. Further improvements could be made if higher performance equipment was used. Cost is generally an important concern. The design proposed in this research benefits the network security requirements at low cost.

VI. CONCLUSION, RECOMMENDATIONS FOR FUTURE WORK

This section summarizes the outcomes achieved in the paper and then provides recommendations for future research.

A. CONCLUSION

A new architecture for NIDPS deployment was designed, implemented and evaluated. There has recently been massive development in computer networks regarding their ability to handle different speeds and data volumes. As a result of this rapid development, computer networks are now more vulnerable than ever to high-speed attacks and threats. These can cause considerable trouble to computer networks and systems. Network intrusions can be categorized at various levels. Many high-speed attacks can be classified as being difficult to detect or prevent. It will become ever more difficult to analyze increasing volumes of traffic due to the rapid shifts in technology that are increasing network speed.

Recently, various open-source tools have become available to cover security requirements for network systems and users. In this paper, the performance of an open source NIDPS has been evaluated in the context of high-speed and volume attacks. The purpose of the evaluation was to determine the performance of the NIDPS under high-speed traffic when restricted by off-the-shelf hardware, and then find ways to improve it.

This study focused on the weakness of such security systems, i.e. NIDPS in high-speed network connectivity. We proposed a solution for reducing this weakness and presented a novel architecture in NIDPS development that utilizes QoS and parallel technologies to organize and improve network management and traffic processing performance in order to improve the performance of the NIDPS.

With our novel architecture, Snort's performance improved markedly, allowing more packets to be checked before they were delivered into the network. The performance (analysis, detection and prevention rate) of Snort NIDPS increased to more than 99%. By using 2 machines (PCs) connected to two 1Gb interfaces, Snort NIDPS processed up to 8 Gbps with 0 drop. This number can be increased up to 32Gbps which is the full system capacity forward bandwidth by implementing more nodes of NIDPS.

The research focused on establishing a technical solution with a theoretical foundation. This information generalizes the problem and solution and thus enables the proposed approach to be applied more easily to infrastructures that are different to the testbed used in this research.

B. RECOMMENDATION AND FURTHER RESEARCH

NIDPSs are used to capture data and detect malicious packets travelling on network media and match them to a database of signatures. Signature-based NIDPS are able to detect known attacks, but the major problem of the signature-based approach is that every signature should have an entry in a database in order to compare with the incoming packets. New signatures arise constantly, and an issue is how to keep track up with new signatures. Another problem is processing time required to check all signatures. Knowledge sharing may provide a solution. Cloud computing which provides for massive processing distribution and sharing is a possible future direction, but this also raises issues of trust. An avenue of future investigation should aim to develop a trusted cloud solution to NIDPS deployment such that if the threshold monitoring tool indicates that traffic is increasing then extra Snort nodes can be brought into play from the cloud. Future work should investigate the use of specialized and trustworthy security clouds e.g. a parallel node of NIDPS implemented on a multi-core/multi-processing cloud environment which can increase the NIDPS processing speed in order to improve its performance.

Statistical based anomaly detection is designed to detect deviations from a baseline model of network behaviour. When the rate of ''malicious'' packet transmission is very high, the attack will almost certainly be detected by a statistical anomaly detector. Therefore, statistical anomaly detection can be used with the proposed novel architecture to redirect traffic to a secured haven for processing when attacks are detected. The secure haven can use the proposed architecture to enable throughput of good traffic while bad traffic is halted. We consider this area of development needs further investigating.

In the testbed, we identified that there is limitation for the number of packets processing which is 8.0 Gbps with 0% packets dropped. The idea has been examined further in terms of performance limitation above 8.0 Gbps, and therefore modification may be made for better response. As experiment 4 showed, packets started to be dropped when load-balancing for traffic exceeding 8.0 Gbps. Analysis is still in development and shall be covered in the future efforts.

Establishing a relationship between traffic size and number of NIDPS cluster nodes for an efficient performance is also an interesting research area. Defining parameters to identify the number of nodes for a scalable response to network speed type and will be a good addition.

REFERENCES

[1] B. Wang, Y. Zheng, W. Lou, and Y. T. Hou, ''DDoS attack protection in the era of cloud computing and software-defined networking,'' *Comput. Netw.*, vol. 81, pp. 308–319, Mar. 2015.

[2] K. Chauhan and V. Prasad, ''Distributed denial of service (DDoS) attack techniques and prevention on cloud environment,'' *Int. J. Innov. Advancement Comput. Sci.*, vol. 4, pp. 210–215, Sep. 2015.

[3] M. D. Samani, M. Karamta, J. Bhatia, and M. B. Potdar, ''Intrusion detection system for DoS attack in cloud,'' *International Journal of Applied Information Systems* (Foundation of Computer Science), vol. 10, no. 5. New York, NY, USA: FCS, 2016.

[4] S. H. Vasudeo, P. Patil, and R. V. Kumar, ''IMMIX-intrusion detection and prevention system,'' in *Proc. Int. Conf. Smart Technol. Manage. Comput., Commun., Controls, Energy Mater. (ICSTM)*, May 2015, pp. 96–101.

[5] W. Bul'ajoul, A. James, and M. Pannu, ''Improving network intrusion detection system performance through quality of service configuration and parallel technology,'' *J. Comput. Syst. Sci.*, vol. 81, no. 6, pp. 981–999, 2015.

[6] N. Akhtar, I. Matta, and Y. Wang, ''Managing NFV using SDN and control theory,'' Dept. CS, Boston Univ., Boston, MA, USA, Tech. Rep. BUCS-TR-2015-013, 2015.

[7] P. S. Kenkre, A. Pai, and L. Colaco, ''Real time intrusion detection and prevention system,'' in *Proc. 3rd Int. Conf. Frontiers Intell. Comput., Theory Appl. (FICTA)*. Bhubaneswar, India: Springer, 2015, pp. 405–411.

[8] M. Li, J. Deng, L. Liu, Y. Long, and Z. Shen, ''Evacuation simulation and evaluation of different scenarios based on traffic grid model and high performance computing,'' *Int. Rev. Spatial Planning Sustain. Develop.*, vol. 3, no. 3, pp. 4–15, 2015.

[9] J.-M. Kim, A.-Y. Kim, J.-S. Yuk, and H.-K. Jung, ''A study on wireless intrusion prevention system based on snort,'' *Int. J. Softw. Eng. Appl.*, vol. 9, no. 2, pp. 1–12, 2015.

[10] Cisco. (2016). *Cisco Interfaces and Modules, Cisco Security Modules for Security Appliances*. Accessed: Feb. 30, 2018. [Online]. Available: http://www.cisco.com/c/en/us/support/interfaces-modules/security-modules-security-appliances/tsd-products-support-series-home.html

[11] M. Trevisan, A. Finamore, M. Mellia, M. Munafò, and D. Rossi, ''DPDKStat: 40Gbps statistical traffic analysis with off-the-shelf hardware,'' Telecom, Paris, France, Tech. Rep. 318627, 2016.

[12] W. Bul'ajoul, A. James, S. Shaikh, and M. Pannu, ''Using Cisco network components to improve NIDPS performance,'' *Comput. Sci. Inf. Technol.*, pp. 137–157, Aug. 2016.

[13] K. R. Kishore, A. Hendel, and M. V. Kalkunte, ''System, method and apparatus for network congestion management and network resource isolation,'' U.S. Patent 9 762 497, Sep. 12, 2017.

[14] Y. Naouri, and R. Perlman, (2015). ''Network congestion management by packet circulation,'' U.S. Patent 8 989 017 B2, Mar. 24, 2015.

[15] Y. Zhu *et al.*, ''Packet-level telemetry in large datacenter networks,'' in *Proc. ACM Conf. Special Interest Group Data Commun.* New York, NY, USA: ACM, 2015, pp. 479–491.

[16] T. Szigeti, C. Hattingh, R. Barton, and K. Briley, Jr., *End-to-End QoS Network Design: Quality of Service for Rich-Media & Cloud Networks*. London, U.K.: Pearson Education, 2013.

[17] M. K. Testicioglu and S. K. Keith, ''Method for prioritizing network packets at high bandwidth speeds,'' U.S. Patent 15 804 940, Nov. 6, 2017.

[18] T. Szigeti, J. Henry, and F. Baker, *Mapping Diffserv to IEEE 802.11 Yes, Tatil*, document RFC 8325, 2018.
[19] D. Melman, I. Mayer-Wolf, C. Arad, and R. Zemach, "Egress flow mirroring in a network device," U.S. Patent 15 599 199, May 18, 2017.
[20] K. K. Kulkarni, and R. O. Nambiar, "Distributed application framework for prioritizing network traffic using application priority awareness," U.S. Patent 15 792 635, Oct. 24, 2017.
[21] Cisco, "Catalyst 3560 switch software configuration guide. Cisco IOS release 15.0(2)," SE and Later Edn., Cisco, San Jose, CA, USA, White Paper OL-26641-03, 2016, Accessed: May 31, 2016. [Online]. Available: https://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst3560/software/release/15-0_2_se/configuration/guide/scg3560.pdf
[22] Cisco (2014) *Security Configuration Guide: Access Control Lists, Cisco IOS Release 15SY*. Accessed: Mar. 20, 2018. [Online]. Available: http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/sec_data_acl/configuration/15-sy/sec-data-acl-15-sy-book.pdf
[23] P. Wheeler and E. Fulp, "A taxonomy of parallel techniques for intrusion detection," in *Proc. 45th Annu. Southeast Regional Conf.* New York, NY, USA: ACM, Mar. 2007, pp. 278–282.
[24] J. Kawahara, K. M. Kobayashi, and T. Maeda, "Tight analysis of priority queuing for egress traffic," *Comput. Netw.*, vol. 91, pp. 614–624, Nov. 2015.
[25] G. Vasiliadis, M. Polychronakis, and S. Ioannidis, "MIDeA: A multi-parallel intrusion detection architecture," in *Proc. 18th ACM Conf. Comput. Commun. Secur.* New York, NY, USA: ACM, 2011, pp. 297–308.
[26] H. Jiang, G. Zhang, G. Xie, K. Salamatian, and L. Mathy, "Scalable high-performance parallel design for network intrusion detection systems on many-core processors," in *Proc. 9th ACM/IEEE Symp. Archit. Netw. Commun. Syst.* Piscataway, NJ, USA: IEEE Press, 2013, pp. 137–146.
[27] M. A. Jamshed *et al.*, "Kargus: A highly-scalable software-based intrusion detection system," in *Proc. ACM Conf. Comput. Commun. Secur.* New York, NY, USA: ACM, 2012, pp. 317–328.
[28] M.-J. Chen, Y.-M. Hsiao, H.-K. Su, and Y.-S. Chu, "High-throughput ASIC design for e-mail and web intrusion detection," *IEICE Electron. Express*, vol. 12, no. 3, pp. 1–6, Jan. 2015.
[29] J. Zhao *et al.*, "A security framework in G-Hadoop for big data computing across distributed Cloud data centres," *J. Comput. Syst. Sci.*, vol. 80, no. 5, pp. 994–1007, 2014.

WALEED BUL'AJOUL received the M.Sc. and Ph.D. degrees, specializing in computer networking and cybersecurity, from Coventry University, U.K. He joined as a Program Developer and a Lecturer with the Computer Science Department, Omar Al-Mukhtar University, Libya. From 2012 to 2017, he was a Researcher with Coventry University. He is currently a Lecturer/Senior Lecturer with Nottingham Trent University, U.K., and a Researcher in Industry 4.0 and Cyberphysical System with the Computing and Technology Department. His first research project was done during his undergraduate at the university. The project was supported by educations training at Libyan high education institutions. The project achieved the Highest Achievement Award in the University, in 2002. In 2010, he embarked on another project privacy in mobile computing. His research interests include the general area of computer networks and cybersecurity performance, including wireless and network communications, modeling, and simulation. An innovative contribution of his work is the establishment of design a new architecture to improve security and privacy. Most recently, his research focused on network architecture and security. His research was about improving security performance for high-speed environments based on intrusion detection and prevention systems, QoS, and parallel technologies; new security architecture was designed and evaluated. His research acquired four achievements award from different institutions and from external and internal events.

ANNE JAMES received the B.Sc. degree from Aston University, U.K., and the Ph.D. degree, specializing in data processing, from the University of Wolverhampton, U.K. She was a Professor of data systems architecture with Coventry University. Her work involves ensuring that excellent teaching, which covers the latest developments in the field, is delivered at undergraduate and postgraduate levels. She promotes research within the department encouraging investigation into innovative systems that advance and support society. She is currently a Professor and the Head of the Department of Computing and Technology. Examples of her current projects are the development of enhanced methods for the detection of biometric identity fraud, the construction of new methods for cloud forensics, the use of block chain technology, and natural language processing for document content analysis. She has successfully supervised over 30 research degree students and has published about 200 papers in peer-reviewed journals or conferences. Her research interests include the general area of creating distributed and intelligent systems to meet new challenges, particularly in the area of cyber security.

SIRAJ SHAIKH was seconded to the Knowledge Transfer Network (KTN), which is the innovation network in Britain, from 2015 to 2016. He served as a Cyber Security Lead for the KTN coordinating activities across academia, industry, and national policy. From 2015 to 2016, he was also seconded to HORIBA MIRA, as part of the Royal Academy of Engineering's Industrial Secondment Scheme. In 2016, he co-founded CyberOwl, which is a VC-backed commercial venture involved in developing early warning system for cyber threats, cyber-physical platform health, and prognostics; CyberOwl has been a part of the U.K.'s first GCHQ Cyber Accelerator, in 2017. He is currently a Professor of systems security with the Future Transport and Cities Research Institute, Coventry University, where he leads the Systems Security Group. He is currently involved in the EPSRC-Funded project ECSEPA, jointly with University College London, investigating evidence-based policy-making for cyber security, working with policy partners including UK's GCHQ/NCSC. He has been involved in the research, development, and evaluation of large-scale distributed secure systems for nearly 20 years. His Doctoral and Postdoctoral Research involved the design and verification of security and safety-critical systems. From 2015 to 2017, he was involved in an EPSRC-Funded research project ACiD, investigating collusion attacks on smart phone platforms, in collaboration with City University and with Swansea University, U.K., and Intel Security as an industrial partner.