Computer Communications 000 (2015) 1–12

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Filtration model for the detection of malicious traffic in large-scale networks

Abdulghani Ali Ahmed[a,*], Aman Jantan[b,1], Tat-Chee Wan[b,2]

[a] Faculty of Computer Systems & Software Engineering, Universiti Malaysia Pahang, Pahang, Malaysia
[b] School of Computer Sciences, Universiti Sains Malaysia, Penang, Malaysia

ARTICLE INFO

Article history:
Received 4 February 2015
Revised 14 October 2015
Accepted 31 October 2015
Available online xxx

Keywords:
ECN
Malicious traffic
QoS regulations
SLA guarantees
User violations

ABSTRACT

This study proposes a capable, scalable, and reliable edge-to-edge model for filtering malicious traffic through real-time monitoring of the impact of user behavior on quality of service (QoS) regulations. The model investigates user traffic, including that injected through distributed gateways and that destined to gateways that are experiencing actual attacks. Misbehaving traffic filtration is triggered only when the network is congested, at which point burst gateways generate an explicit congestion notification (ECN) to misbehaving users. To investigate the behavior of misbehaving user traffic, packet delay variation (PDV) ratios are actively estimated and packet transfer rates are passively measured at a unit time. Users who exceed the PDV bit rates specified in their service level agreements (SLAs) are filtered as suspicious users. In addition, suspicious users who exceed the SLA bandwidth bit rates are filtered as network intruders. Simulation results demonstrate that the proposed model efficiently filters network traffic and precisely detects malicious traffic.

1. Introduction

Network intruders continuously formulate new sophisticated tactics of network intrusion. Generating an immense volume of unsolicited malicious traffic is one of their tactics for overwhelming network resources and disrupting online services for users. A sudden surge in network traffic mostly occurs because of malicious traffic generated from single or distributed sources to prevent legitimate users from using network resources or online services. Malicious traffic may cause failure in conducting e-businesses, accessing network information, or running online services [1].

Typically, malicious traffic can be generated through several technical strategies, such as botnet [2,3], distributed denial of service (DDoS) [4], and Slashdot effect [5]. According to J. Jaeyeon, K. Balachander, and R. Michael [6], the Slashdot effect attack damages the visited site similar to the DDoS attack, although the former may not always be an attack or a malicious distribution. The abovementioned strategies create huge traffic to execute an attack. Thus, development of an effective system to filter malicious packets is valuable for all botnet, DDoS, and Slashdot effect attacks.

The success of malicious traffic filtration methods depends on accuracy, scalability, and reliability. The accuracy of these methods is challenged when malicious and legitimate traffics are similar [7]. The scalability challenge arises because filtering malicious traffic at the early stage requires extra overhead in a volume, which mostly results in performance degradation [8]. The reliability of these methods is challenged when user traffic cannot be investigated in whole or in part. Such a case can be caused by a single point of failure when making filtration decisions.

Recent studies [7–15] have proposed methods to filter malicious packets and verify intruder attempts by monitoring the impact of user behavior on the service level agreement (SLA). According to [16], SLA is an electronic contract between the service provider and its users, a contract that defines the thresholds of quality of service (QoS) metrics that the provider commits to provide the user. The service provider uses random early detection (RED) to prevent congestion at its gateway queues by policing user traffic, which may exceed the predefined bit rate in the SLA [17]. As described in [18], RED is an active queue management mechanism that performs traffic policing depending on the gateway's average queue size, which is calculated for every packet arriving at the gateway queue.

To prevent incipient traffic burst, the RED gateway notifies the user of congestion to reduce its window size once the average queue size (AQS) exceeds a predefined threshold. A bursting gateway notifies the user of congestion in a probability approximately commensurate with the bandwidth share of that user [17]. According to [19,20], notifications of RED gateways for misbehaving users can be either

* Corresponding author. Tel.: +60 9 5492110; fax: +60 9 5492144.
E-mail addresses: abdulghani@ump.edu.my (A.A. Ahmed), aman@cs.usm.my (A. Jantan), tcwan@cs.usm.my (T.-C. Wan).
[1] Tel.: +60 4 653 4642; fax: +60 4 6573335.
[2] Tel.:+60 4 653 4633, +60 4 653 3617; fax: +60 4 6573335.

http://dx.doi.org/10.1016/j.comcom.2015.10.012
0140-3664/© 2015 Elsevier B.V. All rights reserved.

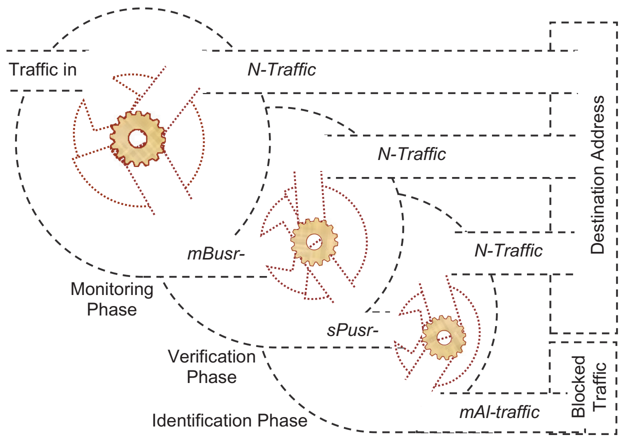


Fig. 1. Phases of traffic filtration.

implicit (dropping packets at gateway queues) or explicit (marking a bit in packet headers). According to [21–23], explicit notification is employed by modifying RED gateways to set a mechanism of TCP explicit congestion notification (ECN) in the header of IP packets. RED can deny a user from sending more bit rates than those specified in the SLA through a single gateway. However, it cannot deny the violation if the user sends it through multiple ingresses at a rate lower than the SLA bit rate.

The current paper proposes a distributed edge-to-edge model to detect and filter malicious traffic generated through several gateways and to delimit responsible users through three principal phases: monitoring, detection, and identification (Fig. 1). In the monitoring phase, ECN is primarily exploited to trigger traffic filtration. During traffic congestion, RED gateways distributed on the domain ingress edges are used to uncover a user with ECN as a misbehaving user (*mBusr*) and to classify its traffic as misbehaving user traffic (*mBusr-traffic*). In the detection phase, an *mBusr* with packet delay variation (PDV) ratios exceeding the predefined ratios in the SLA is classified as a suspicious user (*sPusr*), and its traffic is filtered as suspicious user traffic (*sPusr-traffic*). In the identification phase, an *sPusr* that exceeds the packet transmission rate (PTR) ratios guaranteed in the SLA is a malicious user, and its injected traffic is malicious traffic (*mAl-traffic*). Users who do not violate the ratios specified in the SLA are classified as legitimate users, and their traffic is filtered as normal traffic (*N-traffic*).

The novelty of this study is represented through several potential contributions. First, this study develops a scalable method for monitoring user violations of QoS regulations. The method benefits from ECNs, which are issued by RED gateways to misbehaving end users. ECNs inform misbehaving end users about the need to reduce the window size of data transmission. Thus, monitoring of user traffic is triggered only when burst gateways actually generate an ECN to *mBusrs*. Second, this study presents a reliable load-balancing technique for reducing the overhead required to make decisions on traffic filtration. This technique is deployed using multi-management units that are interconnected on the basis of a virtual overlay network. The virtual structure of overlay networks is exploited to exchange filtration messages among the management units and to finalize decisions on traffic filtration. This technique also benefits from the anycast protocol for supporting one-to-one transmission among the gateway routers and the nearest overlay network management unit. Third, this study proposes a new solution to the problem of single-point failure in the decision maker agent. This solution involves multi-management units connected through a virtual overlay network. When a particular unit fails, the overlay network maintains its structure and sets the nearest unit for the ingress edges closest to the unit of failure. Lastly, this study defines a deficiency method for improving the accuracy of traffic filtration and for distinguishing *mAl-traffic* from *N-Traffic*. The accuracy of this method depends on the passive measurement of transmission rates, although only for the *sPusr* filtered in the previous stage of ECN monitoring.

The rest of the paper is organized as follows. Section 2 discusses related works. Section 3 describes the architecture of malicious traffic filtration. Section 4 presents the deployment of agents. Section 5 describes the malicious traffic filtration policy. Section 6 presents the experimental results and analytical evaluation. Finally, Section 7 concludes and discusses the possibilities for future work.

2. Related work and comparison

Many real-time studies have investigated the impact of user behavior on QoS regulations to filter malicious traffic. Anomaly behavior is detected when its impact on QoS regulations exceeds a particular threshold. The threshold values, which distinguish normal from abnormal impact, are defined either by specifying a predefined ratio or by the threshold ratio obtained through training the system on a normal pattern. The following subsections classify the related work of this paper based on the way of specifying the thresholds and critically analyze their existing limitations.

2.1. Classification of threshold-based approaches

In this paper, related works are classified into three directions depending on how they specify threshold values: learned, adaptive, and predefined threshold-based approaches. Learned threshold-based approaches have been studied in the past [24–27]. Trained thresholds are created by learning network traffic patterns on a particular network for a specific period (e.g., days, weeks, or months). According to [23], the threshold is trained with traces of normal system behavior. Observed event streams are then fed into the trained threshold, which classifies these streams as normal (the observations match the training data) or anomalous. Adaptive threshold-based approaches have been studied in the past [17,18,28]. Adaptive threshold is a varying threshold whose value is calculated automatically. The value of this threshold is set dynamically and adaptively on the basis of a particular estimation computed from recent traffic measurements. According to [17], the adaptive threshold algorithm is a simple algorithm that detects anomalies on the basis of violations of a threshold that is adaptively set following recent traffic measurements. One advantage of using the adaptive threshold is that it improves accuracy and reduces human intervention by accurately tracking varying real-time measurements [17].

Predefined threshold-based approaches refer to preset values considered as a baseline to separate two different behaviors (i.e., normal or abnormal). This baseline method compares the observed behavior with lower and higher bounds of the threshold. Whenever the behavior goes below the lower bound threshold or above the higher bound threshold, a violation to the thresholds is detected, and the baseline method raises an alarm. The predefined thresholds are commonly used in statistical-based detection mechanisms. Table 1 describes the general limitations of the threshold-based approaches.

2.2. Critical analysis

The proposed study avoids the aforementioned limitations of learned-based approaches by using a hybrid of adaptive and predefined thresholds that are not required in the learning process for malicious traffic detection. The adaptive threshold is used in the phase of network traffic monitoring by utilizing RED thresholds. The predefined threshold is used in the following stages of malicious traffic filtration by utilizing SLA thresholds. Therefore, the aforementioned limitations of adaptive threshold are avoided by using predefined

Table 1
Limitations of threshold-based approaches.

Approach	Limitations
Learned threshold-based approach	– Experiencing a high false alarm rate due to the deviation of the normal behavior from the learned pattern. – If the system undergoes malicious activity during the learning time, it learns malicious behavior, which is later filtered as normal behavior.
Adaptive threshold-based approach	– Adaptive thresholding is designed specifically for real-time anomaly detectors. Other anomaly detectors that operate in non-real-time measurements (e.g., file analysis algorithms and rule-based traffic classification algorithms) cannot use adaptive thresholding. – Several anomalies can potentially go undetected if the predicted score of the adaptive thresholding algorithm is well above the observed value.
Predefined threshold-based approach	– Using predefined thresholds alone makes it difficult to determine thresholds that balance the likelihood of false positives with that of false negatives. – The robustness of predefined (fixed) threshold-based models is insufficient. A fixed threshold may fail because of regular traffic variations [17].

threshold combined with the adopted ones. The limitations of predefined thresholds are avoided by utilizing SLA thresholds, which are percentile ratios guaranteed by the service provider to the users. SLA thresholds therefore provide considerable accuracy in balancing the likelihood of false positive and false negative alarms in terms of measuring users' consumptions of QoS metrics. The second limitation is also mitigated because the SLA threshold is set to measure a fraction of the traffic behavior of each user instead of the behavior of the entire network traffic. Therefore, a variation in the entire network traffic does not lead to a failure of the SLA threshold. SLA regulations can be used to prevent users from changing the transmission rate and exceeding their share in the link bandwidth.

Most work related to adaptive thresholds has been conducted in [17] as a RED of network congestion. RED is applied to compute the AQS for every packet received at the gateway queue. RED uses adaptive minimum ($\min_{th}$) and maximum ($\max_{th}$) thresholds to prevent traffic bursts at gateways by monitoring traffic shifts in the AQS. The AQS is then compared with $\max_{th}$ and $\min_{th}$. If the calculated AQS is less than $\min_{th}$, all packets are allowed to pass to the destination. However, if the AQS value is greater than $\max_{th}$, the packets are marked to be dropped. In between, every received packet is marked with an ECN in commensurate probability. Users whose packets are marked with an ECN are notified to reduce the volume of the sent packets. The proposed study benefits from the RED algorithm to monitor the anomaly congestion in the network gateways. Users who are notified by dropping or marking excess packets with an ECN at RED gateway queues are considered misbehaving users that need further investigation. Therefore, the proposed model considers explicit notifications of bit marking in packet headers as indicators of user violations of SLA loss and delay guarantees.

Most studies related to predefined threshold-based approaches are those relying on predefined SLA thresholds [12,14,29–31]. Studies to detect SLA violation involve inspecting the SLA to detect malicious traffic in the network. Important studies related to this approach include [10,13,15]. Although these studies are significant in monitoring end-user violation of QoS edge-to-edge regulations, they still have accuracy, scalability, and reliability limitations. In these studies, QoS metrics are measured for every end-user and compared with the predefined SLA threshold to detect violations in the SLA guarantees. SLA violation is detected first by an estimated delay of end-user packets. The delay is estimated by subtracting probe packet timestamps recorded at the two edges of the sender and destination or by extracting packet round-trip time (RTT) and dividing it by 2. As described in [32,33], the limitation of using packet timestamps is the non-synchronization of the two ends, whereas the limitation of using packet RTT is the asymmetric links of the edges. The proposed model rectifies the abovementioned shortcomings by exploiting ECNs to identify misbehaving users at RED gateway queues.

In [34–36], the authors presented studies to detect unwanted malicious traffic in a QoS network domain. These studies accurately distinguish between malicious and normal traffic; however, use of passive measurements to investigate every incoming packet consumes considerable network resources. The current model saves network resources by postponing passive measurement to the last stage of bandwidth investigation. Therefore, only the traffic fractions of suspicious users that have actually violated SLA are passively investigated.

A recent work most related to the current study is presented in [37]. Violations in the SLA and malicious traffic are detected by inspecting QoS metrics such as PDV, packet loss, and PTR [38]. PDV is first estimated for all users at the domain edges using active measurement. In PDV violation, packet loss is passively measured for users with violated PDV guarantees. SLA violation is detected when the measured loss ratios exceed the guaranteed SLA ratio of a user. Finally, the PTR of suspicious users is aggregated and compared with their guaranteed ratios to differentiate legitimate from illegitimate users. Technically, the probe packets between SLA management and domain edges are separately transmitted for each stage, namely, PDV estimation, loss measurement, and PTR measurement. As a result, processing and communication overhead may increase and represent a shortcoming for this approach. In the proposed model, an SLA violation is deduced when RED gateways generate ECNs toward misbehaving users. Therefore, estimating the PDV for all users and measuring the packet loss ratios are unnecessary. Users accused of violating the SLA are notified to reduce their window size. Thereafter, the PDV and PTR of these users are measured to filter illegitimate users. Hence, the communication overhead is improved by reducing information messages.

The algorithms in [10,11,13,37,39] involve the use of central management units to gather the measured values of users from various domain edges and to finally decide on traffic filtration. These algorithms may be vulnerable to single points of failure. In contrast, the algorithm in [15] uses distributed management units on various edges to gather the measured values of users and filter malicious traffic. Although the distributed algorithm is immune against single points of failure, it is not sufficiently scalable because of the high processing and communication overhead generated in gathering distributed measured ratios. The algorithms in all these studies also do not support a reliable solution for investigating *mBusr-traffic*, which may be sent to the non-responded edge.

To solve single-point-of-failure problems, the proposed model creates a new load balancing method by using multi-management units that are connected on the basis of a virtual overlay network. This method also remedies the high overhead limitations of distributed management by employing the anycast technique as a routing protocol among domain ingress edges and overlay network management units. Thus, any ingress edge router can communicate with the nearest management unit in the overlay network through one-to-one connection. The proposed model mitigates the challenge of investigating the *mBusr-traffic* sent to non-responded edges through an alternative solution capable of recognizing the non-responded edges

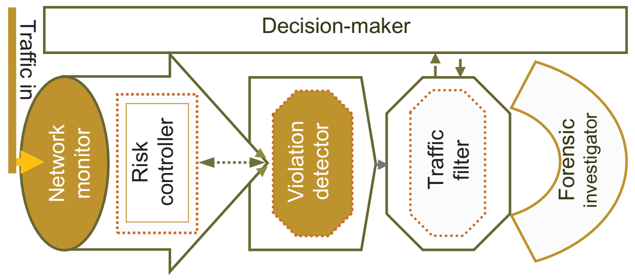


Fig. 2. Architecture of malicious traffic filtration.

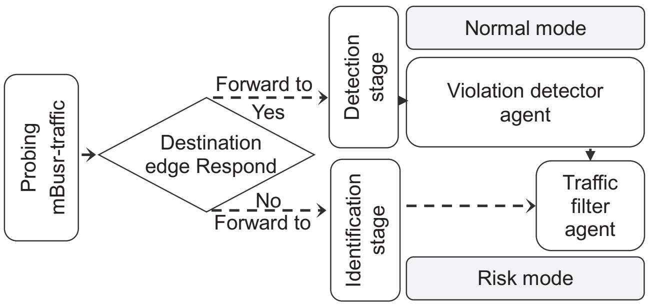


Fig. 3. Risk controller agent operation modes.

first and immediately filtering all *mBusr-traffic* sent to these edges as *sPusr-traffic*.

3. Architecture of malicious traffic filtration

Fig. 2 illustrates the architecture of the proposed model in which traffic filtration is handled as a complementary process through various integrated agents. The next sections describe in detail the design of each agent and its integration with other agents to develop the protection system.

3.1. Network monitor agent

The network monitor agent is used to recognize *mBusrs* notified with an ECN and to probe the *mBusr-traffic* for PDV estimation. Principally, this agent relies on the RED mechanism of traffic policing at domain gateways, as performed in [17]. The RED mechanism is used to prevent traffic bursts at gateways by monitoring abnormal shifts in the AQS. The AQS is computed for every packet received at the gateway queue through a low-pass filter algorithm that uses the exponential weighted moving average (EWMA) technique described in [40]. RED policy uses EWMA to smooth possible short-term increases in queue size (QS), which result from normal traffic bursts or from transitory congestion, significantly increasing the AQS. Therefore, the AQS of the burst gateway is calculated using equation (1) [17] as follows:

$$AQS = AQS \times (1 - w_q) + w_q \times q, \tag{1}$$

where q is the instantaneous buffer size of the gateway queue and w_q is an exponential weight coefficient that defines the time of the low-pass filter with a ratio much less than one. Choosing an appropriate ratio for w_q is important to efficiently calculate the AQS. If the w_q ratio is too large, the averaging transaction may not filter the transitory congestion at the gateway queue. However, if the w_q ratio is too low, the response of the AQS to shifts in the actual QS will be too slow; thus, the gateway may not detect the initial levels of congestion. As described in [17], the ratio of coefficient w_q is selected on the basis of the number of packet arrivals at the gateway queue. Therefore, the ratio of w_q should be set to satisfy the formula (2) [17]:

$$n + 1 + \left((1 - w_q)^{n+1} - 1\right)/w_q < min_{th}, \tag{2}$$

where n is the number of packets arriving at the gateway and $\min_{th}$ is the RED minimum threshold. In one RTT, $\min_{th}$ and $\max_{th}$ are defined by considering that $\max_{th} - \min_{th}$ must not be less than the typical augment in the AQS. The AQS is then compared with $\max_{th}$ and $\min_{th}$. If the calculated AQS is less than $\min_{th}$, all packets are allowed to pass to the destination. However, if the AQS value is greater than $\max_{th}$, the packets are marked to be dropped. In between, every received packet is marked with an ECN in commensurate probability. The network monitor agent therefore exploits the RED mechanism in monitoring network edge routers. The traffic of users who are connected to the edge routers whose AQS exceeds the RED thresholds is therefore filtered for further investigation.

3.2. Risk controller agent

The failure of one end of the connection to exchange information probe packets prevents the coordination of other edges, thus inhibiting the system from performing a reliable filtration process. The risk controller agent supports an alternative solution for the early identification of *mBusr-traffic* sent to the non-responded edges and its immediate filtration as *sPusr-traffic*. The risk controller agent activates the risk mode of the algorithm if an edge fails to exchange information probe packets with other edges. Such a case may occur when a sophisticated intrusion, such as DDoS, botnet, or Slashdot, deters the victim destination edge in delivering the probe packets of coordination. This phenomenon affects filtration in case the intrusion is devastating enough to deny the destination edges to respond. However, this phenomenon does not apply to simple intrusions or to some normal congestion cases where the destination edge is still able to respond and thus coordinate with the source edges for traffic filtration.

In this study, coordination with the destination edge is required only for the PDV estimation to filter *sPusr-traffic* from *mBusr-traffic*. Thus, the risk controller function activates only the risk mode when sending *mBusr-traffic* to non-responded edges that cannot receive or respond to coordination probe packets of PDV estimation. In the risk mode, filtration ignores the detection stage that requires coordination with the destination edge, going directly instead to the identification stage. Thus, *mBusr-traffic* sent to non-responded edges is immediately and completely filtered as *sPusr-traffic* without PDV estimation as illustrated in Fig 3.

In conclusion, the proposed model may continuously operate with the risk controller, but with specific users, not with all users involved in the inspection process. A particular edge router can be non-responded to some users while responded to other users. The user who experiences a non-responded case is not necessarily an attacker or malicious user. Usually, the victim users are the most users who experience non-responded case.

3.3. Violation detector agent

The violation detector agent detects SLA violations by estimating the PDV metric of *mBusr-traffic* at egress edges. In computer networking, PDV means the difference between one-way delays of selected packets in a flow. However, PDV estimation is preferred to not be dependent on delay measurement to avoid the limitations of non-synchronized ends and asymmetrical links (see Section 2). Thus, this paper proposes the use of a four-packet train method [33] that does not require time synchronization or link symmetry for PDV estimation. The four-packet train method estimates PDV ratios by using the received times of the pairs of the four-packet train and the interval time between the packet pairs. Thus, the end-to-end PDV of the kth four-packet train is calculated as follows:

$$PDV_k = (T_{kx2} - T_{kx1}) - T\gamma, \quad (1 \le k \le n), \tag{3}$$

where T_{X1} and T_{X2} are the received times of consecutive packets x1 and x2, respectively, k is the serial number of the four-packet train, and T_γ is the interval time between the packet pairs. To guarantee the validity of the PDV's statistical calculation, T_1 must satisfy the following formula:

$$s(P)/T_1 < min(b_i), \quad (1 \le i \le n), \tag{4}$$

where b_i is the bandwidth of the four-packet train links and $s(p)$ is the size of the packets in the four-packet train. Once Formula (4) is satisfied, the same packet pair is guaranteed to be observed over all links. Otherwise, packet x_1 for the lowest bandwidth link in the first packet pair cannot complete the transmission before packet x_2 when the second packet pair arrives. Consequently, the edge-to-edge PDV of the lowest bandwidth link and all subsequent links downstream of the bottleneck is extremely high, which may invalidate the statistical calculations.

To obtain meaningful results for detecting SLA violations, the PDV ratio of each user is calculated using EWMA. EWMA is used to smooth the possible shifts caused by normal traffic bursts or transient congestion, which may considerably increase the average PDV ratios of packets trains. Through EWMA, the PDV average of the *mBusr* is computed by employing the following formula:

$$avg\_PDV_{mBusr} = avg\_PDV_{mBusr} \times w + PDV_y^k \times (1 - w), \tag{5}$$

where PDV_y^k is the PDV ratio computed from the four-packet train k at the egress edge y of the *mBusr* over the time interval Δt, and w is a small adaptation factor to emphasize the recent history instead of the current sample alone.

3.4. Traffic filter agent

The traffic filter agent measures the PTR fractions of *sPusrs* and reports the measured rates to the decision maker agent. The PTR measurement has two considerations. First, measurement is performed using passive measurement, which requires counting all user packets transmitted over network links. Second, the PTR fraction of each *sPusr* is measured at the ingress gateways where user-sent packets can be captured completely. In view of these considerations, the PTR of every *sPusr* at each ingress edge can be adequately measured by counting the average number of packets generated by a user. According to [41], the PTR can be accurately measured by multiplying the total sent packets with the packet size. The bandwidth consumed by every *sPusr* is computed by measuring the PTR of that user at each ingress edge using Eq. (6) [51] as follow:

$$PTR^i_{sPusr} = \left(\rho s \times \left(avg\_sent^i_{sPusr}\right)\right)/\Delta t, \tag{6}$$

where $avg\_sent^i_{sPusr}$ is the average number of packets sent by the *sPusr* at ingress edge *i*, ρs is the packet size in bits, and Δt is the time interval. Conclusions on malicious traffic sources are made once the end-user sends more than its preset bandwidth share in the SLA.

3.5. Decision maker agent

The decision maker agent provides a mechanism for managing the filtration of malicious traffic. This agent computes the average rates of the PTR for *sPusrs* on the basis of user details gathered from various ingress edges. Therefore, malicious traffic is differentiated from legitimate traffic through the recognition of malicious users who exceed the SLA bit rate of the PTR. The decision-maker agent comprises various management routers (MRs) connected through a virtual overlay network. Authors in [42] define overlay network as a virtual network that is built on top of a real network. Peer-to-peer and client–server networks are examples of overlay networks whose nodes run on top of the Internet. Nodes in the overlay network are connected to each other through virtual or logical links, where each node corresponds to a path in the underlying network through several physical links.

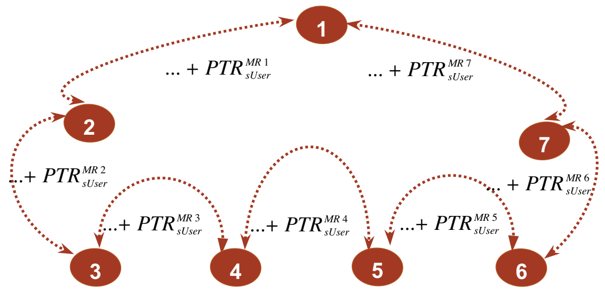

1
2
3
4
5
6
7
... + $PTR^{MR\,1}_{sUser}$
... + $PTR^{MR\,7}_{sUser}$
...+ $PTR^{MR\,2}_{sUser}$
...+ $PTR^{MR\,3}_{sUser}$
...+ $PTR^{MR\,4}_{sUser}$
...+ $PTR^{MR\,5}_{sUser}$
...+ $PTR^{MR\,6}_{sUser}$

Fig. 4. PTR aggregation in overlay MRs.

In the overlay network, each node probes its neighbors in clockwise and counterclockwise directions. Neighbors are determined by visiting the tree through the depth-first search algorithm that starts from any edge node and putting all edge nodes in an ordered sequence. Referring to [43,44], the virtual overlay network is a self-organizing network topology that does not fail in case of node failure. The virtual overlay network provides fast failure detection and recovery mechanisms that enable the network to recover and maintain an appropriate network structure in case of node or link failures [45]. According to [46], failure detection and recovery in the overlay network relies on probing mechanisms for IP-path performance monitoring and failure detection.

In this paper, the overlay network manages MRs as membership in its virtual structure. It treats MRs as peers, similar to a load balancing system. Once a particular MR fails, the overlay network maintains its structure and sets the nearest MR for the ingress edges closest to the MR of failure. Thus, multiple MRs are used in this study to avoid single-point-of-failure problems. Moreover, the overlay network is used to conduct lightweight and scalable communication among various MRs. Accordingly; the decision maker agent manages the process of malicious traffic filtration. Once the *sPusr-traffic* exceeds the PDV ratio predefined in the SLA, every ingress edge measures the PTRs of the *sPusrs* and reports them to the nearest MR. Ingress edges communicate with the overlay-based MR in a scalable manner in which an edge ingress router forwards its PTR to the nearest MR by one-to-one connection. The anycast technique in IPv6 networks allows a system to connect to the nearest server using only one anycast address and is used as a routing protocol among ingress edges and various MR units. In the proposed overlay network, MRs exchange messages among themselves to resolve traffic filtration decisions, where the nearest MR probed from the ingress edges is considered a root node of the overlay network. Thus, the root MR immediately probes the reported PTR_{sPusr} to its neighbor MRs in the right and left directions. If neighbors are reported from the ingress edges, the MRs add their reported PTR^{MR}_{sPusr} to the aggregated PTR_{sPusr} using the following formula:

$$PTR_{sPusr} = PTR_{sPusr} + PTR^{MR}_{sPusr}. \tag{7}$$

The process of aggregating the PTR and sending the subtotal to their right and left neighbors is repeated for every MR. Fig. 4 shows how MRs exploit the virtual structure of overlay networks to exchange PTR messages with each other and resolve the decision of PTR violations.

For every *sPusr*, the total number of PTRs is computed by aggregating the PTR fractions from the corresponding MRs. To compare the PTR measured ratios with the SLA bandwidth shares and to resolve the decision of PTR violations, the root MR transfers the total PTR_{sPusr} to percentile ratios (β_{sPusr}) through the following formula:

$$\beta_{sPusr} = (PTR_{sPusr}/Bnd_{Link}) \times 100\% \tag{8}$$

where Bnd_{Link} is the bandwidth of links in the domain. Therefore, the root node compares the PTR percentile of *sPusrs* with the bandwidth

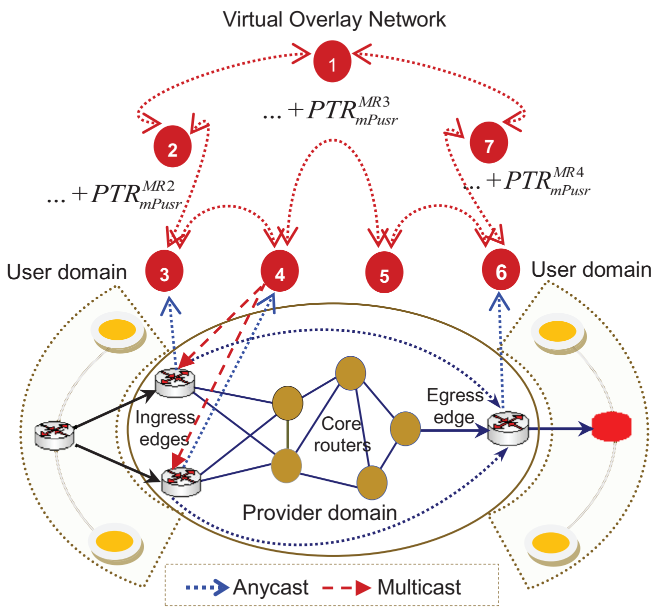
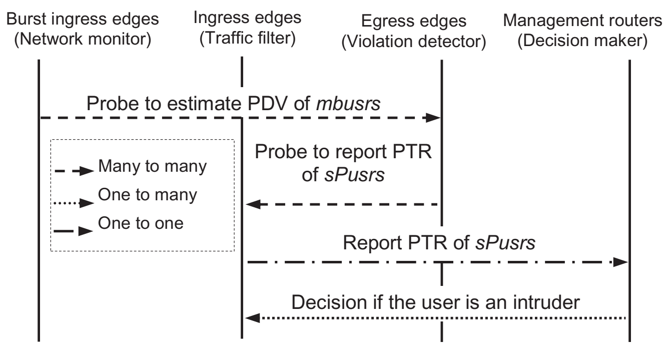


Fig. 5. Deployment of model agents at the provider domain.


Burst ingress edges (Network monitor)
Ingress edges (Traffic filter)
Egress edges (Violation detector)
Management routers (Decision maker)
Probe to estimate PDV of mbusrs
Many to many
One to many
One to one
Probe to report PTR of sPusrs
Report PTR of sPusrs
Decision if the user is an intruder
Fig. 6. Transmission protocol among model agents.

shared with a user in the SLA to detect possible malicious traffic. The decision maker agent then sends notification packets to all ingress edges to filter as malicious the traffic of the user who exceeds the PTR share.

4. Agent deployment

Sensor placement is a key factor in the proposed technique. Sensor placement not only produces an early warning system but also deters network intruders. For this purpose, model agents are optimally placed to guarantee edge-to-edge network security (Fig. 5). Traffic from the Internet enters the network through an ingress edge; hence, assured service traffic classification and policing occur at ingress routers. The network monitor agent is then deployed at ingress routers to detect when the AQS exceeds the RED threshold and when packets are marked with ECNs or drop.

A risk controller agent is installed at the ingress edges to investigate traffic of burst egress edges that do not report their PDV estimation details of the four-packet trains. The risk controller is activated only when the network monitor reports any non-responded destination edge. In contrast, the violation detector is deployed at the egress routers of the domain to estimate PDV metrics at the receiver end using the four-packet train. The violation detector agent is not activated if no dropping or ECN marking is found at the domain gateways. However, the traffic filter agent is activated later to compute the PTR of users who violate their PDV guarantee in the SLA. Bandwidth ratios are measured at ingress edges as the data transfer rate for each user; hence, the traffic filter agent is deployed at the ingress routers. The traffic filter agent must be deployed at the ingress routers, not at the egress routers, to detect local intrusion that may be launched synchronous with another global intrusion, as described in [37].

The decision maker agent is a separate agent activated when it is probed by the traffic filter agent for making decisions on the basis of PTR reports of users from distributed RED gateways. The decision maker agent (Fig. 5) is deployed on separate routers and installed in a virtual overlay network. Although ingress edges are the first tactic for abnormality detection at domain edges, abnormal traffic can still be injected through several gateways by an illegitimate user or through profile-exceeding traffic by a legitimate user. Therefore, coordination among model agents at provider edges (ingress and egress) and the decision maker agent is required to detect attacks that escape detection at single ingress edges. In this section, the transmission protocol is conducted among various agents that perform traffic filtration. The protocol steps executed during the filtration phases of attack monitoring, detection, and identification are illustrated in Fig. 6.

In this protocol, the network monitor agent at burst ingress edges probes the violation detector agent at egress edges for PDV estimation. In turn, the violation detector agent probes the traffic filter agent at ingress edges for PTR measurement. The traffic filter agent at ingress edges reports the PTR of *sPusr-traffic* to the decision maker agent. Finally, the decision maker agent uncovers the *mAls* and probes the ingress edges to filter *mAl-traffic*.

This section also discusses the ability of edge routers to operate the proposed agents, particularly under limited memory resources and processor speed. We analyzed the complexity of extra probe packets transmitted among several agents, where the total number of probing represents the extra overhead required on the corresponding edge router. The proposed protocol indicates that the network monitor agent probes the violation detector agents on all edge routers. Thus, the network monitor agent requires $O(n)$ probing at each ingress edge, where n is the number of edge routers. At each egress edge, the violation detector agent requires $O(n)$ probing to probe traffic filter agents at all edge routers. The traffic filter agent at each ingress edge requires only one packet to probe its nearest MR at the decision maker agent. However, the decision maker agent through the nearest MR requires $O(n)$ probing to probe all ingress edges during intrusion actualization. Each edge router requires only $O(n)$ probing to handle any agent; thus, conclusions on the proposed agents are scalable on edge routers as long as these routers have adequate resources and high performance.

5. Malicious traffic filtration

The following subsections describe the policy of filtering malicious traffic injected from several gateways and delimiting the responsible users through three main phases: monitoring, detection, and identification. Fig. 7 describes the process of checking user's traffic behaviors and filtering malicious traffic. Fig. 8 shows the overlapped algorithms of the complementary agents responsible for detecting service violation and recognizing malicious traffic.

5.1. *Monitoring* mBusr-traffic

In the proposed model, the network monitor agent employs ECNs to recognize misbehaving users and filter misbehaving traffic. Therefore, user traffic marked with ECNs is *mBusr-traffic*, which is requested for further investigation to verify suspicious behavior. However, user traffic not notified by the ECN is *N-traffic*, which is normally allowed to be transmitted to the destination.

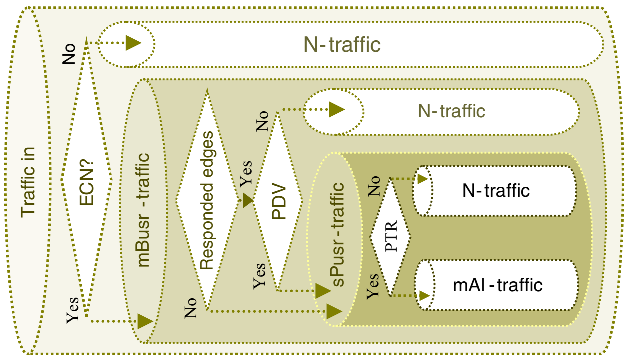


Fig. 7. Malicious traffic filtration.

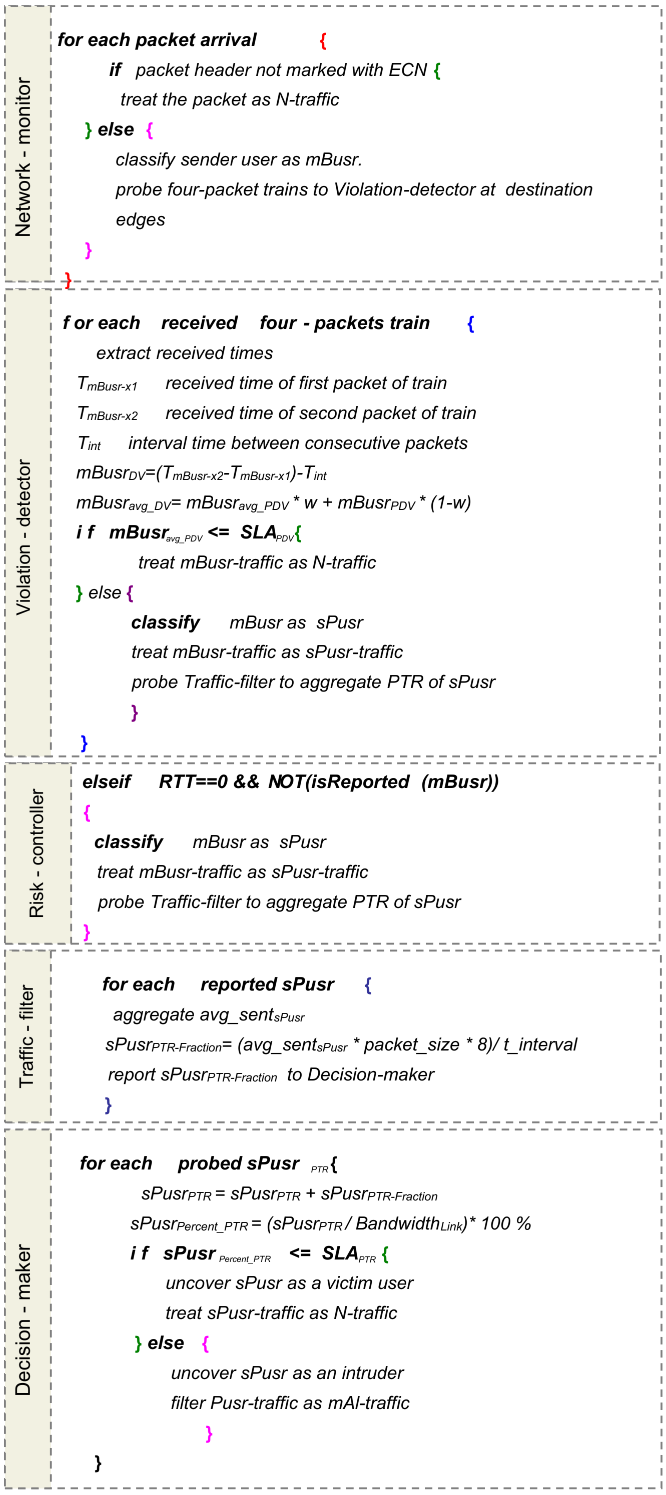
5.2. *Detecting sPusr-traffic*

Further investigation is performed on the *mBusr-traffic* to verify if it is suspicious by estimating PDV ratios using the active measurement technique. At the same time, estimation of only the *mBusr-traffic*, which is the traffic of the misbehaving users, keeps network performance scalable during the investigation. During traffic congestion, the network-monitor agent on RED-based gateways is used to report misbehaving users to the egress edges in order to estimate the PDV ratios of the *mBusr-traffic*. Thus, samples of the four-packet train are probed to the violation-detector agent. Once the violation-detector is probed, the PDV ratios of the *mBusr-traffic* are computed and compared with SLA_{DV}. When the PDV of *mBusr-traffic* exceeds the bit rate of the PDV metric in the SLA, an SLA breach occurs. Therefore, the *mBusr* is classified as *sPusr* and the *mBusr-traffic* is filtered as *sPusr-traffic*. The violation-detector agent then probes the *sPusr-traffic* on the traffic-filter agent at the ingress edges for further investigation. However, the *mBusr-traffic* that does not exceed SLA_{PDV} is classified back to *N-traffic* and is allowed to reach its destination.

If the network-monitor agent reports particular misbehaving users to the violation-detector agent but receives no reply from the destined edges (i.e., egress edges do not probe the details of PDV estimation to the traffic-filter agent), the risk-controller agent concludes that these egress edges are under attack. Thus, the algorithm is transferred to the risk mode. In the risk mode, the PDV estimation of *mBusr-traffic* destined to nonresponded edges is omitted. The risk-controller agent immediately filters the fraction of *mBusr-traffic* as *sPusr-traffic*.

5.3. *Identifying mAl-traffic*

In this study, the intruder is detected by identifying the suspicious users that strip the resources of others. Thus, *sPusr-traffic* with PTR higher than their bandwidth shares in the SLA is identified as *mAl-traffic* by measuring the PTR of suspicious users. Meanwhile, the *sPusr-traffic* of those with normal PTR is *N-traffic*, and its resources are actually abused by the *mAl-traffic*. In the case of PDV violation in the previous phase, the oscillatory change in the PDV activates the passive measurement phase to measure the PTR of *sPusr-traffic*. The violation-detector agent probes the traffic-filter agent to report the total bandwidth consumed by each *sPusr-traffic* user at ingress gateways. Thus, the PTR of each user is calculated and reported to the decision-maker agent for comparisons with SLA_{PTR}. The decision-maker agent considers as intruders the users that exceed the PTR ratios specified in the SLA and their injected traffic as *mAl-traffic*. Users within the bandwidth ratio guaranteed in the SLA are victims, and their traffic is classified back to *N-traffic*. A local attack, which may be simultaneous with a global attack, may be detected by measuring the PTR of users at the ingress edges instead of at the egress edges.


```
Network - monitor
for each packet arrival {
    if packet header not marked with ECN {
        treat the packet as N-traffic
    } else {
        classify sender user as mBusr.
        probe four-packet trains to Violation-detector at destination edges
    }
}

Violation - detector
for each received four - packets train {
    extract received times
    T_mBusr-x1   received time of first packet of train
    T_mBusr-x2   received time of second packet of train
    T_int   interval time between consecutive packets
    mBusr_DV=(T_mBusr-x2-T_mBusr-x1)-T_int
    mBusr_avg_DV= mBusr_avg_PDV * w + mBusr_PDV * (1-w)
    if mBusr_avg_PDV <= SLA_PDV {
        treat mBusr-traffic as N-traffic
    } else {
        classify mBusr as sPusr
        treat mBusr-traffic as sPusr-traffic
        probe Traffic-filter to aggregate PTR of sPusr
    }
}

Risk - controller
elseif RTT==0 && NOT(isReported (mBusr))
{
    classify mBusr as sPusr
    treat mBusr-traffic as sPusr-traffic
    probe Traffic-filter to aggregate PTR of sPusr
}

Traffic - filter
for each reported sPusr {
    aggregate avg_sent_sPusr
    sPusr_PTR-Fraction= (avg_sent_sPusr * packet_size * 8)/ t_interval
    report sPusr_PTR-Fraction to Decision-maker
}

Decision - maker
for each probed sPusr_PTR {
    sPusr_PTR = sPusr_PTR + sPusr_PTR-Fraction
    sPusr_Percent_PTR = (sPusr_PTR / Bandwidth_Link)* 100 %
    if sPusr_Percent_PTR <= SLA_PTR {
        uncover sPusr as a victim user
        treat sPusr-traffic as N-traffic
    } else {
        uncover sPusr as an intruder
        filter Pusr-traffic as mAl-traffic
    }
}
```

Fig. 8. Agent-overlapped algorithms.

6. Experiment results

This section presents the experimental results of the proposed model. A comparative analysis is also conducted to evaluate the performance of the proposed model with recent existing schemes.

6.1. *Simulation setup*

A simulation experiment was conducted by using a network simulator NS-2.35. As shown in Fig. 9, the network topology is composed

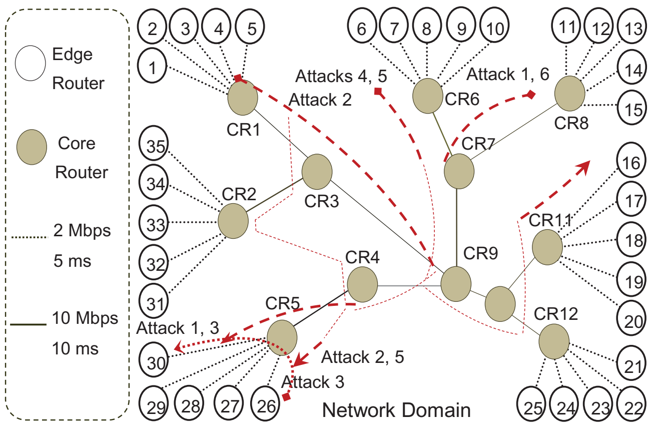


Fig. 9. Simulated network topology.

Table 2
Background traffic setting of end-users.

End-users	Connected to	Destined to	N-traffic (Mbps/user)
U1–U10	E1–E5	E26	1
U11–U20	E6–E10	E20	0.7
U21–U30	E11–E15	E30	1
U31–U40	E16–E20	E10	0.5
U41–U50	E21–E25	E33	0.5
U51–U60	E26–E30	E1	0.7
U61–U70	E31–E35	E25	1

Table 3
Parameter values of RED algorithm.

Parameter Name	Description	Value
Queue limit	Router maximum buffer size	100 packets
Min_{th}	Minimum threshold	10–15 packets
Max_{th}	Maximum threshold	30–45 packets
Queue weight	Emphasizing factor	0.002

Table 4
SLS values of end-users.

Metric	Description	Maximum Value
Delay	One-way delay	Total of links delay
DV	Delay variation	10% of the links delay
Loss	Packet loss	0.01 of each user PTR
Bandwidth	Packet transfer rate	20% of link bandwidth

Table 5
Details of simulated attacks.

Attack	Attack time	Source user	Injected through	Intended for	PTR (Mbps)
Attack 1	10–25 45–70	U30	E11–E15	E30	>10
Attack 2	25–45 70–85	U10	E1–E5	E26	>10
Attack 3	30–60	U55	E26–E27	E30	>10
Attack 4	5–80	U15	E6–E10	E16	>10
Attack 5	5–80	U12	E6–E10	E26	>8
Attack 6	5–80	U25	E11–E15	E16	>8

of 47 nodes [35 edge routers (E1–E35) and 12 core routers (CR1–CR12)]. Network traffic is FTP over TCP traffic generated by 70 end-users (U1–U70). Each user could use several active sources to send several flows through one or more ingress edges. Each of the five edges was gathered as a fivefold set; likewise, each of the 10 end-users was gathered to obtain a denary set. Each end-user could send its data through one fivefold set of ingress edges as maximum, and each ingress edge could be used only by one denary set of end-users as maximum. For instance, U1 can send its data through E1, E2, E3, E4, and E5; however, E1 can be used by U1–U10. Further details of the description of user traffic setting, such as volume of traffic actually generated, source, and destination machine, are presented in Table 2.

RED gateways were modified to indicate congestion. Table 3 shows the values of the RED parameters selected to guarantee an efficient calculation of gateway AQSs. The maximum TCP flow windows were 256 and 512 packets, and the maximum packet size was 1024 bytes. QoS ratios guaranteed for end-users was predefined by the SLA. The values of delay, PDV, packet loss, and bandwidth metrics specified for end-users were registered in service level specifications (SLS), as shown in Table 4. The simulated experiment lasted for 100 s.

6.2. Result and discussion

In this simulated experiment, network traffic was monitored under light load and traffic burst. Light load was observed at 0–9 and 86–100 s. Within these periods, end-users did not consume more than their bandwidth share; therefore, the network was properly provisioned. Traffic burst was observed at 10–85 s, when attacks were executed and the link bandwidth could no longer accommodate all user traffic. Within this period, six mutual attacks were simulated as malicious traffic increased from 1 Mbps to more than 10 Mbps. In this experiment, malicious traffic is generated as mix of constant bit rate (CBR), exponential (EXP), and variable bit rate (VBR) with real-time transport protocol (RTP) traffic. Fig. 9 illustrates from which ingress edges the attacks were injected and for which edges the attacks were intended. Table 5 presents details of these attacks and the users that generated them. It is worth mentioning that experiment was run several times. In the following graphics the mean values across several test repetitions are reported. The mean values of the test repetitions were also normalized to draw general conclusion about the findings of this study.

6.2.1. Scenario for monitoring mBusr-traffic

This scenario demonstrates the ability of the RED technique to recognize misbehaving user traffic in the network gateways. It also demonstrates the ability of the proposed model to filter different types of traffic and exclude the misbehaving ones. To this end, this scenario was executed in three modes with different settings. In the first mode, TCP window size was set to 256 packets and type of malicious traffic was set to CBR/EXP traffic. In the second mode, TCP window was set to 512 packets while malicious traffic was kept as CBR/EXP type. In the third mode, TCP window was kept as 512 packets, while malicious traffic type was set to VBR with RTP traffic. The results obtained from these modes are similar in terms of detecting the misbehaving traffic as shown in Fig 10a, b, and c.

In these figures, the network gateways with an AQS between 10 and 45 packets marked the packet headers of misbehaving users with ECN notifications in the period (6 s to 87 s) which is the period of attacks. In Fig. 10a, the gateways e1–e15 and e26–e30 of AQS between 10 and 30 marked packets of U1-U30 and U50-U60 with ECN notifications. In Fig. 10b, the gateways e1–e15 and e26–e30 of AQS between 15 and 45 packets marked the same users with ECN notifications in the same period. Fig. 10c shows that the same gateways marked the same users in the same period. Table 6 shows the details of ECN notifications issued for misbehaving users. E6–E10 marked U11–U20 with ECN from 6 s to 81 s. E11–E15 marked U21–U30 with ECN from 11 s to 26 s and from 46 s to 71 s. E1–E5 marked U1–U10 with ECN from 28 s to 47 s and from 72 s up to 87 s. E26–E30 marked U51–U60 with ECN from 31 s to 61 s. The ECN notifications were used in this experiment to identify *mBusrs* with traffic that required further investigation. During the period of attacks, the number of *mBusrs* notified

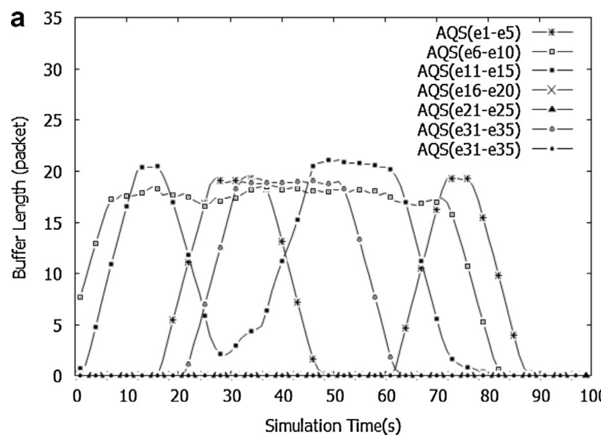


Fig. 10a. Domain RED gateways AQS using $Min_{th} = 10$, $Max_{th} = 30$, TCP window 265, and CBR/EXP over UDP as attack traffic, and FTP over TCP as background traffic.

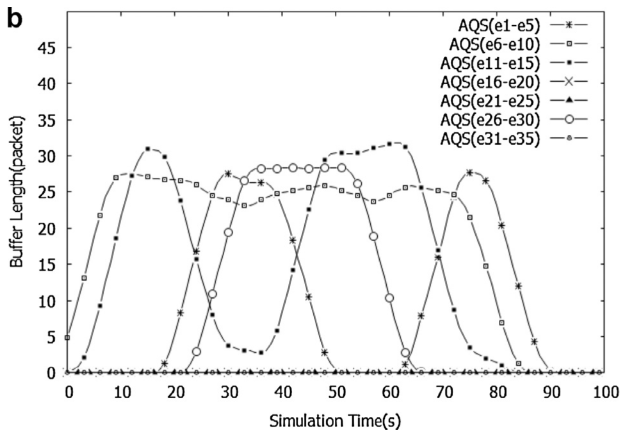


Fig. 10b. Domain RED gateways AQS using $Min_{th} = 15$, $Max_{th} = 45$, TCP window 512, CBR/EXP over UDP as attack traffic, and FTP over TCP as background traffic.

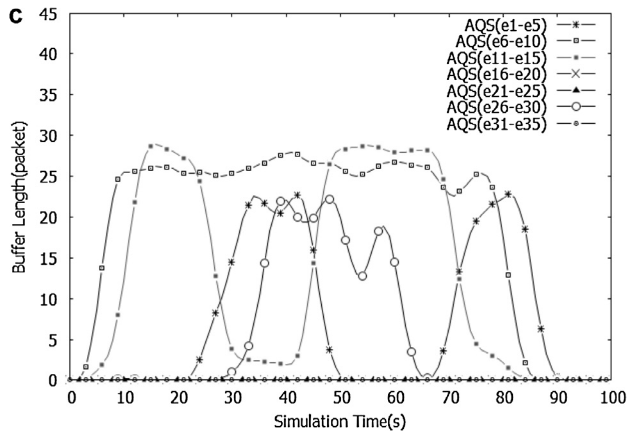


Fig. 10c. Domain RED gateways AQS using $Min_{th} = 15$, $Max_{th} = 45$, TCP window 512, VBR over RTP as attack traffic, and FTP over TCP as background traffic.

Table 6
Details of ECNs issued for *mBusr.*

Notified users	Generator edges	ECN marking started at	ECN marking stopped at
U21–U30	E11–E15	11 s	26 s
U21–U30	E11–E15	46 s	71 s
U1–U10	E1–E5	28 s	47 s
U1–U10	E1–E5	72 s	87 s
U51–U60	E26–E30	31 s	61 s
U11–U20	E6–E10	6 s	81 s

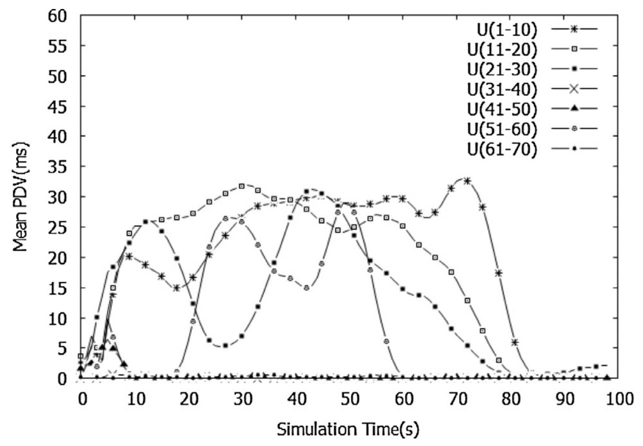


Fig. 11. Mean PDV of mBusrs.

by the burst ingress edges with ECN oscillated to reach a maximum value of 40 *mBusrs*. Therefore, the ECN strategy filtered 40 *mBusrs* of 70 users in the worst case. This strategy also involved the determination of filtration time of the burst gateway traffic of these *mBusrs*.

6.2.2. *Scenario for detecting* sPusr-traffic

Further investigation for every user with mBusr-traffic was conducted by estimating the PDV of the mBusr-traffic using Formulas (3)–(5) of the four-packet train method, where train length is four packets, packets type is CBR, and maximum train packet size is 40 B. In Fig. 11, the PDV estimations of mBusr-traffic show that the average rates of U(1–10), U(11–20), U(21–30), and U(51–60) exceeded their PDV ratios in the SLA.

The PDV ratios of these users increased from 5 ms to more than 33 ms between 5 and 82 s of simulation time. Thus, the traffic of these users was classified as *sPusr-traffic* and filtered for further investigation. However, those with PDV average rates of approximately 5 ms throughout the experiment time were legitimate users. Therefore, their traffic was classified as *N-traffic*.

6.2.3. *Scenario for filtering out* mAl-traffic

We showed that only users whose traffic was classified as *sPusr-traffic* violated the SLA. Thus, the PTR measurement was required only for the traffic of 40 users, not 70. Using the passive model, the PTRs of *sPusrs* were measured at the ingress edges using Formula (6) and reported to the decision-maker agent. The decision-maker agent used the reported PTR fractions to calculate the total PTR ratio of each *sPusr* using Formula (7). Thereafter, the total PTR ratios were used to compute the link bandwidth consumption percentage of each *sPusr* using Formula (8). The computed PTR percentages were compared with the preset SLA thresholds to differentiate legitimate from malicious traffic and intruders from legitimate users. Fig. 12 shows the link bandwidth percentages consumed by every *sPusr*. This implies that U10, U12, U15, U25, U30, and U55 were intruders who launched the attack. Consequently, the excessive traffic sent by these intruders was filtered as *mAl-traffic*. Conversely, other *mBusrs* were all victims plundered by the attacks of U10, U12, U15, U25, U30, and U55.

6.2.4. *Efficiency of using ECNs*

As explained in Section 2, the existing schemes investigate user traffic by inferring QoS metrics, such as delay, and comparing these with the SLA [11,35,47,48] or training [24,25,49,50] thresholds. Delay rates were measured throughout the simulation time to evaluate the efficiency of using ECNs upon the methods of existing schemes. The results were analyzed by comparing with the ECN results. For all domain users, delay was estimated using the active measurement

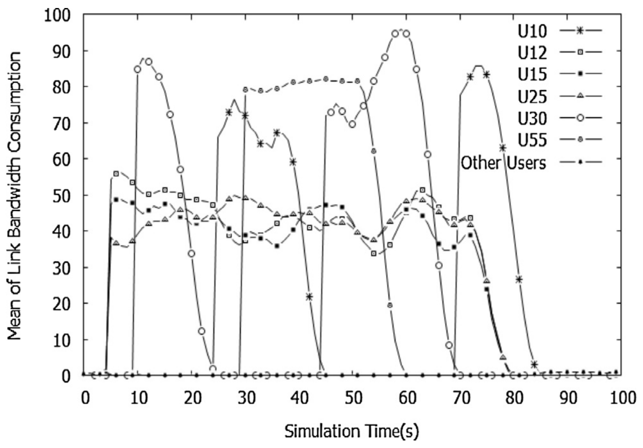


Fig. 12. Mean PTR of *sPusrs* at ingress edges.

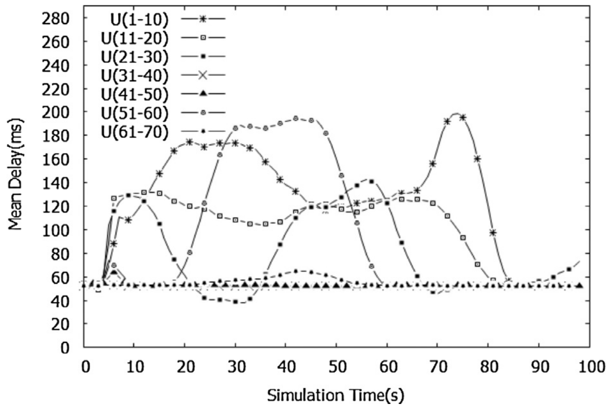


Fig. 13. Mean delay of users.

technique. Fig. 13 shows that the mean delay of users U(1-10), U(11-20), U(21-30), and U(51-60) exceeded the delay bit rates in the SLA. Thus, delay estimation filtered 40 mBusrs, whereas ECN monitoring filtered 40 mBusrs. Notably, all the mBusrs filtered by delay estimation had already been filtered by ECN monitoring. The key advantage of this model is that misbehaving users that could be filtered by the delay method with high communication cost and processing overhead could be filtered by ECN monitoring with low communication cost and processing overhead. Therefore, the use of ECNs in detecting and filtering malicious traffic is a significant and novel contribution.

6.3. Analysis and evaluation

Effective intrusion detection systems should be able to support high detection accuracy with low false alarm rate and a large-scale network in a scalable manner, and should be reliable for monitoring the entire network traffic. The proposed model was therefore evaluated by comparing accuracy, scalability, and reliability. Fig. 9 shows the topology used for the comparison analysis. Performance was measured for a network domain with *U* users and *n* edge routers. Table 7 lists the values of variables used in the comparison equations.

6.3.1. Accuracy evaluation

Accuracy was evaluated by comparing the detection accuracy and false alarm rate of the proposed FModel with each Stripe-based [11,13] and Com-based [37] approach. To compare the detection accuracy of malicious and non-malicious traffic, accuracy was measured using the following formula [49]:

$$Accuracy = \left(\frac{TP+TN}{TP+TN+FP+FN}\right) \times 100\%, \tag{9}$$

Table 7
Symbols and values used in the comparison equations.

Symbol	Description	Values used
εn	Bursting gateways	57% of n
μU	Misbehaving user	60% of u
ρs	Probe packet size	40 B
S	Stripe length	3
T	Four-packet train length	4
Mop	Mega operations	

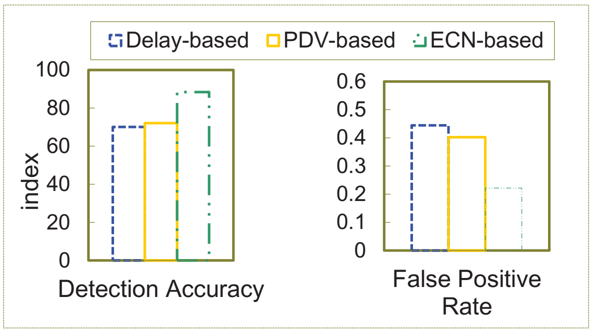


Fig. 14. Accuracy and false positive rate of the schemes.

where *TP* is true positive, *TN* is true negative, *FP* is false positive, and *FN* is false negative. For each scheme, detection accuracy was measured in an average, whereas accuracy was computed first for each end-user using Formula (9), after which the average user accuracy for each scheme was determined. The comparison of scheme accuracy shown in Fig. 14 demonstrates that FModel has the highest accuracy among all schemes.

False alarm rate refers to the percentage of legitimate flows falsely filtered as misbehaving flows with respect to the percentage of all nonmisbehaving flows. The false alarm rate was measured using the following formula, as described in [49]:

$$FalsePstv = \left(\frac{FP}{FP+TN}\right) \times 100\%. \tag{10}$$

Similar to detection accuracy, false positive rate was measured first for each end-user separately, and then an average of the users' false alarm rate was computed for each scheme. Comparing the existing schemes in terms of accuracy (Fig. 16) shows that ECN monitoring to detect *mAl-traffic* has the highest accuracy among the schemes that depend on delay or PDV metrics.

The obtained results also show that ECN monitoring has the least percentage of false alarm rates. The percentage of accuracy of FModel is approximately 22% higher than that of the delay-based approach and approximately 19.9% higher than that of thePDV -based approach. The percentage of false positive rates of the FModel is approximately 43.9% less than that of the delay-based approach and 40.2% less than that of thePDV -based approach. The figure also reveals that, compared with the existing schemes, FModel is more capable of recognizing service violations and detecting simultaneous attacks with a low false negative rate.

6.3.2. Scalability evaluation

Scalability evaluation was achieved by measuring the processing overhead *POH* and communication overhead *COD* of each scheme with variable domain sizes. To compute the *COD*, the total number of probe packets injected per unit time for investigating network traffic was multiplied by the size of probe packets ρs. However, the *POD* was computed by considering the extra processing δ at all hops *h*, through which a packet passes per unit time. For each probe packet in the monitoring schemes, a *POD* is required to change some fields in the packet header, such as address lookup, checksum computation, and any other central processing unit processing overhead.

In this subsection, *POD* and *COD* are required to be computed for each investigated QoS metric to evaluate the scalability of the corresponding schemes. The *POD* and *COD* of the PTR measurement are equivalent in all schemes. Each scheme approximately requires $O(n)$ probing overhead to measure the PTR metric, as described in their algorithms. In addition, the Stripe-based or Com-based approach requires the approximate $O(n)$ probing overhead to measure metric loss. However, FModel is conclusively improved in terms of loss measurement overhead. It does not measure metric loss. For delay or PDV metrics, Stripe-based and Com-based approaches estimate one of these metrics at the monitoring phase, which is the first phase of filtration where network traffic is not yet filtered. Thus, the *POD* and *COD* in these schemes are required in large volumes to estimate the delay or PDV of all user traffic. However, FModel estimates PDV at the detection phase, which is the second phase of filtration where *mBusr-traffic* is already filtered and *POD* and *COD* are required to estimate the PDV of *mBusrs* only.

Each of the evaluated schemes requires a different volume of probing overhead based on its method for delay or PDV estimation. Thus, the communication protocols of these schemes are analyzed in detail to create mathematical formulas that can be used to calculate the *POD* and *COD* required for delay or PDV estimation on each scheme. In the Stripe-based approach, every edge injects a stripe of *S* packets to every egress edge pair for each user. The egress edge pair sends a complementary stripe in reverse. Thus, the total number of injected probes is $S \times (n-1) \times (n-2) \times U$, where *U* is the number of domain end-users. Formulas (11) and (12) show the *POD* and *COD* in the Stripe-based monitoring approach respectively:

$$POD_{Stripe} = S \times (n-1) \times (n-2) \times h \times \delta \times U, \tag{11}$$

$$COD_{Stripe} = S \times (n-1) \times (n-2) \times \rho s \times U. \tag{12}$$

To monitor the network in the Com-based approach, every edge injects four-packet trains *T* to every egress edge of each user. Thus, the total number of injected probes is $T \times (n-1) \times (n-1) \times \varepsilon U$, where εU is the number of users filtered by the Com-based approach algorithm as suspicious end-users. The *POD* and *COD* in the Com-based approach are computed using Formulas (13) and (14) respectively:

$$POD_{Com} = T \times (n-1) \times (n-1) \times h \times \delta \times \varepsilon U, \tag{13}$$

$$COD_{Com} = T \times (n-1) \times (n-1) \times \rho s \times \varepsilon U. \tag{14}$$

In the proposed FModel, every bursting edge *bN* probes egress edges with the four-packet trains to report the edge-to-edge PDV estimation of the corresponding *mBusrs*. In response, the egress edge routers report the PDV estimation of the *mBusrs* to the ingress edges. The total number of transmitted probes in the network domain is $(\varepsilon n - 1) \times (n-1) \times \mu U$, where εn is the number of bursting gateways and μU is the number of misbehaving end-users. The values of εn and μU are computed based on the result shown in Fig. 10. Thus, the *POD* and *COD* of FModel are computed using Formulas (15) and (16) respectively:

$$POD_{FModel} = T \times (\varepsilon n - 1) \times (n-1) \times h \times \delta \times \mu U, \tag{15}$$

$$COD_{FModel} = T \times (\varepsilon n - 1) \times (n-1) \times \rho s \times \mu U. \tag{16}$$

The scalability of each scheme was evaluated using a variable number of gateways ranging from 35 to 2000, while the number of users was fixed at 70. Evaluation of the scalability of schemes with thousands of edge routers also showed that FModel, when compared with the existing schemes, can support a large-scale network in a scalable manner. A comparison of the *POD* and *COD* among the three schemes showed that of all the comparison scenarios FModel exhibits the lowest values. Fig. 15 shows the POD of each scheme in the monitoring phase for a network comprising 2000 edge routers and 70 end users. The Stripe-based schemes require approximately 5.E+03 Mega operations (Mop). The Com-approach schemes require approximately 4.E+03 Mop, and the FModel requires 2.3E+03 Mop. Fig. 16 shows the COD of each scheme required for monitoring a network with 2000 edge routers and 70 end users. The Stripe-based scheme requires 2.6E + 05 Mbps, Com-approach schemes require 2.2E + 05 Mbps, and the FModel requires 1.3E + 04 Mbps.

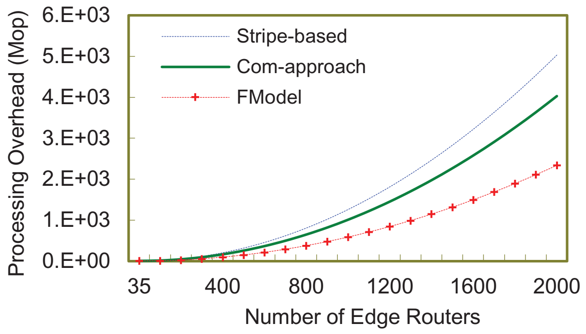


Fig. 15. POD comparison.

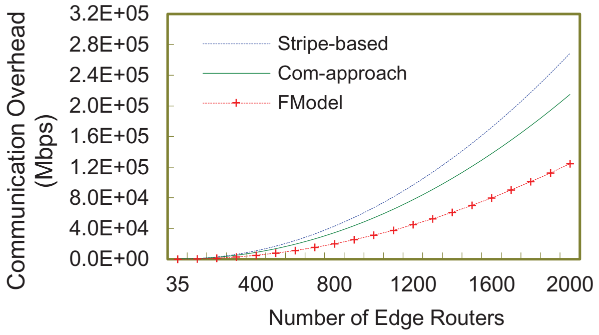


Fig. 16. COD comparison.

6.3.3. *Reliability evaluation*

Reliability is measured as the probability that a system will not fail and will complete its intended function in the expected manner over time. The failure of the scheme to investigate all or a fraction of network traffic conclusively affects its performance reliability in detecting an actual attack. In existing studies, reliability challenges occur once the algorithm fails to make traffic filtration decisions or once a particular edge fails to coordinate with other domain edges for traffic investigations. The reliability of the scheme was evaluated based on its ability to reliably exchange information probe packets among domain edges and to continually achieve a reliable filtration process in the expected manner.

The Stripe-based and Com-based approaches did not propose a solution for single-point-of-failure problems to maintain SLA management in case of large volumes of malicious traffic. Furthermore, neither of these schemes proposed a solution to investigate traffic fractions that were sent to nonresponded edges that fail to receive or respond to probe packets. The proposed FModel addressed these limitations via the functions of the decision-maker and risk-controller agents (Section 3).

7. Conclusion and future work

The model proposed in this study detects the attack before it happens with early warning notifications to uncover intruders while still in the planning stages of an attack. Monitoring ECNs as an early notification when anomaly congestion surfaces is beneficial for filtering malicious traffic and minimizing potential overhead and resources associated with intrusions. Although the results are based on simulation scenarios, a comparison of approximate results indicates that the use of ECNs to trigger traffic filtration reduces more than 40% of traffic investigation overhead and assists the proposed algorithm in supporting a large-scale network in a scalable manner. Estimation of the PDV of misbehaving users ensures that suspicious users are filtered and that the scope of bandwidth measurement is reduced from 40 to 36 suspicious users. The strategy of the proposed model to postpone passive measurements until the final stages and limit the scope to suspicious users also results in a continuously accurate and scalable system performance throughout the measurement period.

By providing overlay network-based multi-MRs as a reliable solution for single-point-of-failure problems, this model can obviate failures in making traffic filtration decisions. Conducting an alternative solution for investigating the mBusr-traffic destined to non-responded edges also increases the reliability of the proposed model to investigate all user traffic, thus resulting in accurate filtration.

Future research will extend the role of the forensic investigator agent to identify how intruders generate malicious traffic at an early stage. This objective can be achieved by storing the output of the forensic investigation process to a historical database. This historical information may be used in the preliminary phases of filtration to filter traffic with similar malicious traffic behavior at an early time.

Acknowledgments

RDU grant number RDU1403162, Faculty of Computer System & Software Engineering, Universiti Malaysia Pahang supported this work.

References

[1] D. Dittrich, J. Mirkovic, P. Reiher, S. Dietrich, Internet Denial of Service: Attack and Defense Mechanisms, Pearson Education, 2004.
[2] S. Khattak, N.R. Ramay, K.R. Khan, A. Syed, S.A. Khayam, A taxonomy of botnet behavior, detection, and defense, IEEE Commun. Surv. Tutor. 16 (2014) 898–924.
[3] S.S.M. Gyanchandani2, Analysis of Botnet behavior using queuing theory, Int. J. Comput. Sci. Commun. (IJCS) 1 (2010) 239–241.
[4] L. Qin, L. Hong, K. Songlin, L. Chuchu, Feature extraction and construction of application layer DDoS attack based on user behavior, in: Proceedings of the 33rd Chinese Control Conference (CCC), 2014, pp. 5492–5497.
[5] B.H. Ismail Ari, Ethan L. Miller, Scott A. Brandt, Darrell D.E. Long, Managing flash crowds on the Internet, in: Proceedings of the 11th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS'03), 12-15 Oct. 2003, pp. 246–249.
[6] J. Jung, B. Krishnamurthy, M. Rabinovich, Flash crowds and denial of service attacks: characterization and implications for CDNs and web sites, in: Proceedings of the 11th International Conference on World Wide Web, 2002, pp. 293–304.
[7] O. Al-Jarrah, A. Siddiqui, M. Elsalamouny, P.D. Yoo, S. Muhaidat, K. Kim, Machine-learning-based feature selection techniques for large-scale network intrusion detection, in: Proceedings of the IEEE 34th International Conference on Distributed Computing Systems Workshops (ICDCSW), 2014, pp. 177–181.
[8] D.R. Choffnes, Service-Level Network Event Detection from Edge Systems, Northwestern University, 2010.
[9] Y. Xuan, I. Shin, M.T. Thai, T. Znati, Detecting application denial-of-service attacks: A group-testing- based approach, IEEE Trans. Parallel Distrib. Syst. 21 (2010) 1203–1216.
[10] A. Habib, S. Fahmy, B. Bhargava, Monitoring and controlling QoS network domains, J. Network Mgmt. 15 (2005) 11–29.
[11] A. Habib, S. Fahmy, S.R. Avasarala, V. Prabhakar, B. Bhargava, On detecting service violations and bandwidth theft in QoS network domains, Comput. Commun. 26 (2003) 861–871.
[12] Y.L. Ren´e Serral-Graci'a, Jordi Domingo-Pascual, Philippe Owezarski, et al., "Towards end-to-end SLA assessment", in: Proceedings of the 28th Conference on Computer Communications. IEEE INFOCOM, Apr 2009, pp. 2581–2585.
[13] A. Habib, M. Hefeeda, B.K. Bhargava, Detecting service violations and DoS attacks, in: Proceedings of the NDSS, 2003.
[14] S. Joel, B. Paul, D. Nick, R. Amos, Accurate and efficient SLA compliance monitoring, in: presented at the Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications, Kyoto, Japan, 2007, pp. 109–120.
[15] A. Habib, M. Khan, B. Bhargava, Edge-to-edge measurement-based distributed network monitoring, Comput. Netw. 44 (2004) 211–233.
[16] C.-K. Tham, Y. Liu, Assured end-to-end QoS through adaptive marking in multi-domain differentiated services networks, Comput. Commun. 28 (2005) 2009–2019.
[17] S. Floyd, V. Jacobson, Random early detection gateways for congestion avoidance, IEEE/ACM Trans. Netw. 1 (1993) 397–413.
[18] B. Abbasov, S. Korukoglu, Effective RED: An algorithm to improve RED's performance by reducing packet loss rate, J. Netw. Comput. Appl. 32 (2009) 703–709.
[19] C. Kulatunga, G. Fairhurst, Enforcing layered multicast congestion control using ECN-nonce, Comput. Netw. 54 (2010) 489–505.
[20] V. Santhi, A. Natarajan, A new approach to Active Queue Management for TCP with ECN, in: Proceedings of the First International Conference on Advanced Computing, 2009. ICAC, 2009, pp. 76–81.
[21] S. Floyd, TCP and explicit congestion notification, ACM SIGCOMM Comput. Commun. Rev., 24 (1994) 8–23.
[22] H. Bai, M. Atiquzzaman, Enhancing TCP throughput over lossy links using ECN-capable RED gateways, in: Proceedings of the IEEE 58th Vehicular Technology Conference, 2003. VTC 2003-Fall, 2003, pp. 2721–2725.
[23] R.R. Chodorek, A simulation of ECN-capable multicast multimedia delivery in ns-2 environment, in: Proceedings of 14th European Simulation, ESS, 2002, pp. 233–237.
[24] D.P.T.M. Thangavel, K. Saravanan, Defend against Anomaly Intrusion Detection using SWT Mechanism, Int. J. Innov., Manag. Technol. 1 (2) (2010) 209–213.
[25] M.T.P. Thangaraj, Cluster based Statistical Anomaly Intrusion Detection for Varied Attack Intensities, Int. J. Comput. Appl. 24 (2011) 27–33.
[26] A. Garg, A.N. Reddy, Mitigation of DoS attacks through QoS regulation, Microprocess. Microsyst. 28 (2004) 521–530.
[27] V. Das, V. Pathak, S. Sharma, M. Srikanth, and G. Kumar, "Network intrusion detection system based on machine learning algorithms," 2010.
[28] M. Chonggang Wang, Jiangchuan Liu, Bo Li, S.M. Kazem Sohraby, Y. Thomas Hou, LRED: a robust and responsive AQM algorithm using packet loss ratio measurement, IEEE Trans. Parallel Distributed Syst. 18 (2007) 29–43.
[29] K. Jacek, R. Dominik, Z. Krzysztof, Z. Slawomir, P. Grzegorz, N. Pawel, Definition and evaluation of penalty functions in SLA management framework, in: presented at the Proceedings of the Fourth International Conference on Networking and Services, Gosier, 2008, pp. 176–181.
[30] D.C. Verma, Service level agreements on IP networks, 92, 2004, pp. 1382–1388.
[31] G.Serral Ren, Y. Marcelo, L. Yann, O. Philippe, M.-B. Xavi, An efficient and lightweight method for service level agreement assessment, Comput. Netw.: Int. J. Comput. Telecommun. Netw. 54 (2010) 3144–3158.
[32] W.-Z. Lu, W.-X. Gu, S.-Z. Yu, One-way queuing delay measurement and its application on detecting DDoS attack, J. Netw. Comput. Appl. 32 (2009) 367–376.
[33] Z. Hong-Hua and C. Ming, "Network topology inference based on delay variation," in: *Proceedings of the International Conference on Advanced Computer Control, 2009, ICACC*, Singapore 2009, pp. 772–776.
[34] G. An, P. Joon, Packet marking based cooperative attack response service for effectively handling suspicious traffic, In Information Security and Cryptology, 4318, Springer Berlin Heidelberg, January 2006, pp. 182–195.
[35] A.J. Abdulghani Ali Ahmed, Ghassan Ahmed Ali, A potent model for unwanted traffic detection in QoS network domain, JDCTA: Int. J. Dig. Content Technol. Appl. 4 (2010) 122–130.
[36] A. Jantan, T.-C. Wan, Real-time detection of intrusive traffic in QoS network domains, IEEE Secur. Priv. 11 (2013) 45–53.
[37] A.A. Ahmed, A. Jantan, T. C Wan, SLA-based complementary approach for network intrusion detection, Comput. Commun. 34 (14) (2011) 1738–1749.
[38] A. Botta, D. Emma, A. Pescapé, G. Ventre, Systematic performance modeling and characterization of heterogeneous IP networks, J. Comput. Syst. Sci. 72 (7) (2006) 1134–1143.
[39] A.A. Ahmed, A. Jantan, M. Rasmi, Service violation monitoring model for detecting and tracing bandwidth abuse, J. Netw. Syst. Manag. 21 (2013) 218–237.
[40] F.R. Johnston, Exponentially Weighted Moved Average (EWMA) with Irregular Updating Periods, J. Operat. Res. Soc. 44 (1993) 711–716.
[41] P. Ningning Hu Steenkiste, Evaluation and characterization of available bandwidth probing techniques, IEEE J. Commun. 21 (Aug. 2003) 879–894.
[42] Y. Amir, M. Miskin-Amir, Y. Javadi, M. Khan, J. Stanton, Scalable Flow Transport and Delivery Network and Associated Methods and Systems, Google Patents, 2013.
[43] D. Doval, D. O'Mahony, Overlay networks: A scalable alternative for P2P, IEEE Internet Comput. 4 (2003) 79–82.
[44] I. Dragos, P. Adrian, Unicast QoS routing in overlay networks, in: D.K. Demetres (Ed.), Network Performance Engineering, Springer-Verlag, 2011, pp. 1017–1038.
[45] D. Andersen, H. Balakrishnan, F. Kaashoek, R. Morris, Resilient Overlay Networks, ACM.SIGOPS Oper. Syst. Rev., vol. 35, 2001, pp. 131–145.
[46] Z. Li, L. Yuan, P. Mohapatra, C.-N. Chuah, On the analysis of overlay failure detection and recovery, Comput. Netw. 51 (2007) 3828–3843.
[47] A. Habib, Mohamed Hefeeda, B. Bhargava, Detecting Service Violations and DoS Attacks, in: Proceeding of Network and Distributed System Security Symposium (NDSS'03), San Diego, California, Februray 2003, pp. 177–189.
[48] H. Ahsan, K. Maleq, B. Bharat, Edge-to-Edge Measurement-Based Distributed Network Monitoring, 44, Elsevier North-Holland, Inc., 2004, pp. 211–233.
[49] A. Osareh, B. Shadgar, Intrusion detection in computer networks based on machine learning algorithms, Int. J. Comput. Sci. Netw. Secur. (IJCSNS) 8 (2008) 15–23.
[50] V. Das, V. Pathak, S. Sharma, M. Srikanth, G. Kumar, A. Vidyapeetham, T. Nadu, Network intrusion detection system based on machine learning algorithms, Int. J. Comput. Sci. Inf. Technol. (IJCSIT) 2 (2010) 138–151.