Chapter 5
Detection and Mitigation of High-Rate Flooding Attacks

G. Mohay, E. Ahmed, S. Bhatia, A. Nadarajan, B. Ravindran, A.B. Tickle, and R. Vijayasarathy

5.1 Introduction

Because high-rate flooding attacks constitute such a potent threat to the delivery of Internet-based services, the early and reliable detection of the onset of such an attack together with the formulation and implementation of an effective mitigation strategy are key security goals. However, the continuously evolving nature of such attacks means that they remain an area of active research and investigation. This chapter focuses largely on our research into attack detection, with some discussion of mitigation through IP address filtering. The chapter outlines leading-edge work on developing detection techniques that have the potential to identify a high-rate flooding attack reliably and in real time or, at least, in near real time. In addition, it formulates an architecture for a DoS Mitigation Module (DMM) to provide a vehicle for integrating the elements of the solution.

Section 5.2 reviews related work on network anomaly detection, while the Sect. 5.3 addresses the key issue of feature extraction and the allied problem of determining which features of Internet traffic can best be used to characterise

G. Mohay (✉) • E. Ahmed • S. Bhatia • A.B. Tickle
Information Security Institute, Queensland University of Technology, Brisbane, Australia
e-mail: g.mohay@qut.edu.au; e.ahmed@qut.edu.au; s.bhatia@qut.edu.au; ab.tickle@qut.edu.au

A. Nadarajan
Maths and Computer Application Department, PSG College of Technology, Coimbatore, India
e-mail: anitha_nadarajan@mca.psgtech.ac.in

B. Ravindran
Department of Computer Science and Engineering, Indian Institute of Technology Madras, Chennai, India
e-mail: ravib@iitm.ac.in

R. Vijayasarathy
Network Security Research Group, Society for Electronic Transactions and Security, Chennai, India
e-mail: vijayasarathy@setsindia.net

S.V. Raghavan and E. Dawson (eds.), *An Investigation into the Detection and Mitigation of Denial of Service (DoS) Attacks: Critical Information Infrastructure Protection*,
DOI 10.1007/978-81-322-0277-6_5, © Springer India Pvt. Ltd. 2011

a DDoS attack. It also examines the problem of ranking the efficacy of the various machine-learning techniques that have been utilised in the detection process. The following three sections describe three different approaches to the detection problem. Section 5.4 shows how the Change Point Analysis (CPA) technique is used in combination with other techniques such as bit vectors and bloom filters to monitor the rate of arrival of new IP addresses which is then used as a basis to detect a high-rate flooding event. This is followed in Sect. 5.5 by a discussion of an approach to DDoS detection based on Naive Bayesian (NB) techniques that exploit characteristic features of the transmission control protocol (TCP) and User Datagram Protocol (UDP) to distinguish a high-rate flooding attack from normal traffic. It then shows how NB techniques could be used to detect a high-rate flooding attack in near real time. Section 5.6 discusses the Cumulative-sum-based Adaptive Neuro-Fuzzy Interface System (CANFIS) approach which comprises a blend of an artificial neural network (ANN), a Fuzzy Inference System (FIS) and the cumulative sum (CUSUM) algorithm. In this approach, different attacks are first modelled using CUSUM techniques, and the modelled traffic is then classified as being an attack or normal based on the Adaptive Neuro-Fuzzy Inference System (ANFIS). This is followed by Sect. 5.7 'Conclusion and Further Work'.

In addition to the approaches described in this chapter, readers should also note that Chap. 4 includes an account of some recent experiments on the Indian testbed using hop-count filtering (HCF) to mitigate flood-type attacks.

5.2 Review of Traffic Anomaly Detection Techniques

The very nature of high-rate flooding attacks suggests that the primary point of focus of corresponding attack detection techniques should be on detecting anomalies in network traffic. The intention here is to identify traffic that is likely to constitute high-rate flooding attacks. While signature detection techniques are designed to detect attacks based on signatures of attacks already learnt by them, anomaly detection techniques learn network traffic from a baseline profile and detect traffic anomalies which deviate significantly from the baseline profile. Signature detection techniques are effective against known attacks while anomaly detection has the ability to detect unknown (zero-day) attacks.

Mitigation approaches involve dropping traffic completely (and running the risk of dropping 'good' traffic), or throttling traffic (with its attendant negative effect on response times), or provisioning extra resources to cope with the high-rate traffic. Of these, the most intensively researched approach is the first, and the key requirements here necessarily are accuracy and timeliness [44]. Regarding accuracy, an alert should be raised and traffic dropped if and only if an anomalous traffic event occurs on the network [44]; furthermore, the number of false positives and false negatives must be kept to a minimum, the former so as not to disrupt the normal provision of services, the latter for obvious reasons. Regarding timeliness, the detection technique must be capable of detecting the anomalous situation in real time or in near real-time so that appropriate response/mitigation processes

may be invoked as rapidly as practicable to limit the impact of the attack. There are significant challenges in satisfying these two requirements. For example, as Kline et al. [44] observe, there is no exact definition of what constitutes anomalous traffic patterns. Normal (or so-called non-anomalous) network traffic can sometimes exhibit characteristics that, at face value, would be indicators of 'anomalous' behaviour. This includes, for example, phenomena such as so-called flash events which are sudden and extreme bursts of (legitimate) traffic [20,44,78]. Moreover, the detection problem is compounded by the fact that not only is the actual composition of normal network traffic both diverse and continuously evolving [44], the pattern of traffic used in an actual attack can be programmed to mimic such behaviour [20].

For the purpose of this discussion, the detection process is considered to comprise three main elements:

- Recording and/or measurement of certain parameters of interest
- Data analysis
- Decision-making about whether or not the observed behaviour is classified as anomalous (and the subsequent triggering of a response such as generating an alert or dropping the anomalous traffic)

5.2.1 Parameters of Interest and Approaches Used

The analysis and decision-making approaches determine the number of separate parameters to be measured and the granularity of the measurements (e.g. the sampling interval). Commonly used measurements are those of the volume of traffic entering a network (also known as ingress), and sometimes the volume of traffic departing from the network (also known as egress). Network traffic volume (or traffic flow) is typically expressed in terms of both packets/second and bits (or bytes)/second [55, 61]. Other parameters of interest include, for example, source/destination IP addresses, port addresses and protocol type (e.g. to distinguish between TCP, UDP and Internet control message protocol (ICMP) packets) [20,57]. An alternative to focusing on measurements at the packet level and one that fits the inherent interlinking and sequencing of packets in the transmission control protocol/Internet protocol (TCP/IP) suite of protocols is to consider aggregates of packets as so-called traffic-flows [13, 14]. In this context, a traffic-flow could, for example, be the sequence of packets that comprises a complete TCP-level connection session. In this case, the packets within the designated traffic-flow are linked by common Internet protocol (IP) source address and destination address as well as TCP source and destination ports.

In 2006, Carl et al. [20] undertook a systematic survey of techniques for detecting various forms of flooding attacks. They grouped the detection techniques under the following categories:

(a) Activity profiling
(b) Sequential change-point detection
(c) Wavelet analysis

They then ranked the performance of individual techniques in each category on the basis of memory use and computational complexity. Obviously, a survey of this type and the corresponding rankings only represent a snap-shot of the techniques available at the time. However, they raise a number of concerns which are important in analysing subsequent work in this field. Basically, in their view, most of the results on which they based their comparison were of limited utility to the wider networking community because the results were not generated in an environment that replicated real-world network environments.

5.2.2 Detection Performance

Since the survey undertaken by Carl et al. [20], there has been continuous activity and refinement in this field [9, 10, 44, 63, 78]. For example, in the interests of achieving computation tractability in operational environments, Kline et al. [44] restrict their measurements (and subsequent analysis and decision-making) to traffic volume only, that is, their initial technique does not require packet header or payload inspection as used in other techniques. Their particular technique uses a wavelet-based method in the analysis phase and, in the interests of computational efficiency, a Bayesian network in the decision-making phase. Similarly, a recent technique proposed by Papadopoulos [63] uses cumulative packet rates as the primary parameter of interest. (This particular technique also uses a novel combination of change-point detection and discrete Fourier transforms to identify an attack.)

The focus on performance reinforces the point made earlier that one of the key challenges in implementing a workable solution to detecting a network flooding attack is to be able to operate the detection process in real time. In essence, this means executing all of the tasks described above (i.e. monitoring the traffic stream, the possible construction of feature vector based on a predetermined set of key characteristics, using the feature vector to compute a metric to be used as the basis for comparison and then deciding if a given segment of the network traffic stream is 'normal' or 'anomalous') at a speed commensurate with that of the underlying transmission links, that is, so-called wire speed. Such speeds are a moving target and are being driven inexorably upwards by improvements in technology. To be competitive the detection processes must be currently capable of operating at speeds of at least 10 Gbps [56, 61] and preferably higher. Fortunately the emergence of high-performance parallel-processing computer hardware and associated software development environments has made feasible the task of designing and implementing sophisticated detection processes for such hardware platforms and then operating them at speeds approaching those required. Commercial examples of network processing hardware specifically designed for this purpose (e.g. the Intel IXP2400, the AMCC np7510, the EZchip NP-1 and the Agere Fast Pattern Processor and Routing Switch Processor) are listed in [84]. In addition, discussion on using the high-performance parallelism of field-programmable gate arrays (FPGAs) to achieve the same goal can be found in [18, 23, 66].

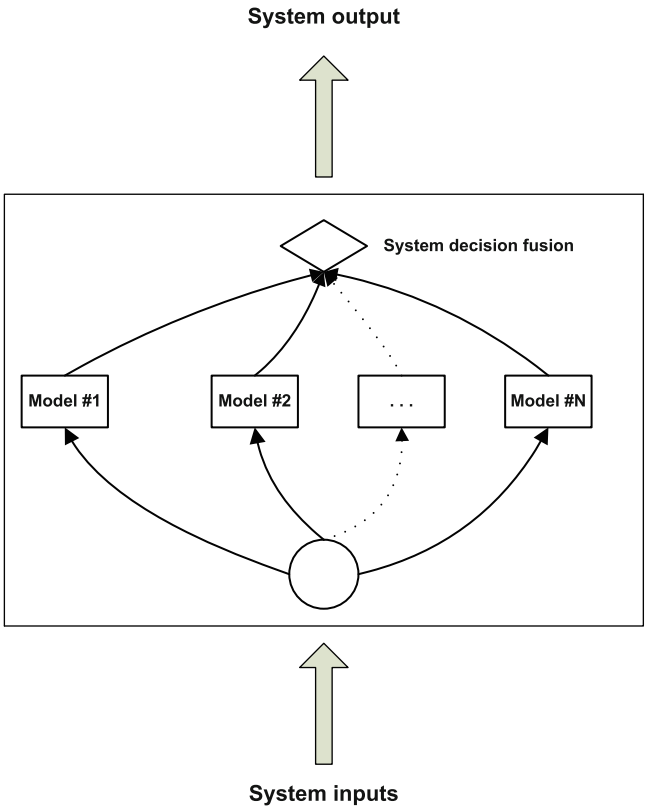

Fig. 5.1 Ensemble learning (Adapted from Wang et al. [83])

An alternative approach to using a single anomaly detection algorithm is that of Shanbhag and Wolf [73,74]. They exploit the potential of high-performance parallel processors to operate several anomaly detection algorithms in parallel. Under this regime, the output from each algorithm is normalised and then aggregated to produce a single anomaly metric. (This notion of using ensembles (Fig. 5.1) has a long history in the domain of machine learning [24, 83] and ensembles have been used successfully in a wide range of problems, e.g. see [76]).

While focusing on a simple metric such as sudden increases in traffic volume assists in maximising performance in order to meet the objective of timely detection, some approaches to detecting high-rate flooding attacks involve the inspection and analysis of both the header and content (payload) of packets. Notwithstanding their utility, there are two constraints on deploying this type of detection technique. The first, which applies, for example, at the ISP level, is that of privacy. The second is, of course, performance. Analysing both packet headers and packet contents is time- and resource-intensive. Consequently, systems that analyse both the header and content of each packet have the potential to become bottlenecks and, in the extreme case, an actual point of failure during a high-rate flooding attack.

In the interests of performance, header analysis has historically focused simply on source and destination IP addresses and packet type [11, 12, 52, 60]. In addition, a typical low-level packet analysis may involve verifying the TCP port number in the transport-layer header [17]. However, as mentioned previously, the advent of high-performance hardware (including FPGAs) and specialised network processors has created an environment in which it is now feasible to perform deeper and more complex levels of analysis in real time [18, 19, 23, 35, 66, 73, 74, 84]. For example, the capability now exists to perform 'deep-packet' inspection, including the ability to parse the application-level content of each packet [35]. At a commercial-product level, this capability appears in the form of so-called application-aware firewalls. There is now also the capability to exploit the inherent parallelism in network processing by disassembling traffic into individual connection streams and then analysing the sequence of packets in each such connection stream concurrently [66, 84]. As mentioned previously in the brief discussion on 'ensemble machine learning', the availability of high-performance parallel-processing hardware also affords the opportunity to operate a set of decision-making/filtering algorithms concurrently.

It is evident that in some circumstances, the successful identification of attack activity can be considerably assisted by having individual network devices or computers share information regarding the network traffic activity they see. Consequently a number of architectures have been proposed for allowing (primarily) sets of routers or devices of similar capability to act in concert [11, 12, 52, 77]. The focal point is often on the routers that operate on the network borders and where ingress and egress packet filtering can be performed efficiently. One final comment is that, as indicated previously, an important consideration in deploying such filters is their ability to operate at so-called wire speed lest the filters themselves adversely affect packet-forwarding performance [12]. The availability of high-speed parallel-processing hardware is an important step in mitigating this problem.

5.2.3 Decision-Making and Mitigation

The decision to accept or discard a packet can be made on a binary white-list/black-list basis. Such lists may take a variety of forms, for example, source or destination-based access control lists, real-time black-hole lists and DNS black lists. In addition, the information in such lists may be drawn from (trusted) external parties, for example, Spamhaus XBL (http://www.spamhaus.org/xbl) and CBL (http://cbl.abuseat.org). This builds upon the link between the notions underlying the creation and utilisation of such lists and the broader concept of building and deploying a reputation scheme as part of the decision-making process.

A considerable body of ideas and experience have now emerged as to how to design and execute the required mitigation strategy once an attack commences and is detected [12, 52, 55]. Whilst these defence mechanisms have become increasingly

sophisticated, central to the strategy for mitigating the impact is to reduce the traffic volume to manageable levels as quickly as possible. Necessarily, this requires that individual IP packets or packet streams be either discarded (or blocked) or delayed. The inherent difficulty is in processing the packets (and streams) in real time in order to decide which packets to filter, which packets to allow, which packets to proceed unmolested, and where (i.e. in which networking device or devices) the actual decision is to be made.

In this context, it is worth noting the existence of a degree of similarity between the current problem of identifying and filtering anomalous packet streams in real-time and earlier work in real time network congestion management [51]. For example, algorithms such as random early detection (RED) [30] and fair random early detection (FRED) [43] were developed to decide probabilistically which TCP packets should be dropped when a given network device reached a point where the volume of traffic exceeded the available buffer space. A key difference between this early work on congestion management and the current approaches is that the capability now exists to involve a greater number of information elements in the decision-making process. This opens up a number of interesting research possibilities. For example, one potential avenue of investigation could be to use a reputation scheme to assign network traffic to a particular 'class' that can then be selectively controlled using Class-Based Queuing techniques [31] from congestion management.

5.3 Evaluating Machine-Learning Algorithms for Detecting DoS Attacks

5.3.1 Related Work

Two important and challenging research problems in detecting distributed denial of service (DDoS) attacks are:

1. Extracting a valid and sufficient subset of features that can be used to build efficient models to identify a DDoS attack
2. Ranking the efficacy of the various machine-learning techniques that have been utilised in the detection process

For most problem domains, the process of feature reduction which involves extracting the most significant and relevant attributes or features prior to applying modelling techniques (such as machine-learning and statistical techniques) can lead to a major improvement in the time required in training and testing the model. However, in comparison with other problem domains, extracting a set of features that characterise Internet traffic to the point of being able to distinguish normal traffic from anomalous traffic is particularly difficult. One problem, for example, is

that nodes in the Internet experience widely differing traffic flux densities caused by the large variations in the number of users seen at each node. This makes it difficult to decide as to what constitutes 'normal' traffic on the Internet. Another problem, and one that can be seen from the discussion on the detection techniques already presented, is that there are potentially a large number of variables that can be used to characterise network traffic patterns. Nevertheless, extracting the important and relevant attributes from network traffic is crucial for modelling network behaviours so that attack behaviours can be differentiated clearly from normal behaviour. This feature-extraction problem has been studied by a number of groups. For example, Xu et al. [87] selected eight relative values as features that are independent from the network flow. Zargar et al. [91] proposed and investigated the identification of effective network features for probing attack detection using the Principal Component Analysis (PCA) method to determine an optimal feature set. Jin et al. [37] discussed the application of multivariate correlation analysis to DDoS detection and proposed a covariance analysis model for detecting flooding attacks. They used all of the flag-bits in the flag field of the TCP header as features in the covariance analysis model. The authors have demonstrated the successful use of the proposed method in detecting SYN flooding attacks which is an important form of DDoS attacks. However, the method has the major limitation that there is no guarantee that the six flags are valid or sufficient features to detect all forms of DDoS attack with consistent accuracy.

As has been discussed previously in Sect. 5.2, a widely diverse range of statistical methods and machine-learning techniques could be used to detect abnormal changes in the resource usage that are indicative of a DDoS attack. However, both approaches have their limitations. For example, one identifiable problem with statistics-based detection is that it is not possible to find out the normal network packet distribution. Rather, it can only be simulated as a uniform distribution. Some research papers suggest that this problem may be resolved by using clustering methodologies to formulate the normal patterns since one of the advantages of clustering methods over statistical methods is that they do not rely on any prior known data distribution. While machine-learning techniques, typically drawn from the allied field of data mining, have been shown to produce a high degree of accuracy in detecting DDoS attacks, they also have their limitations. For example, these techniques typically require a lengthy learning period and hence, currently, these methods typically cannot operate in real time.

Despite these current limitations, a solution to the problem of reliable DDoS detection will come from either or both these domains and considerable research effort continues to be directed to this end. For example, Seo et al. [72] have used a multi-class SVM classification model to detect DDoS attack as have Xu et al. [87]. In the work of Xu et al. [87], a group of new features was also introduced, including the composition of relative values as part of an expanded set of detection information. They also proposed a new approach of using attack intensity to detect a DDoS event. In [65], Paruchuri et al. proposed a new Probabilistic Packet Marking (PPM) scheme called TTL-based PPM scheme, where each packet is marked with

a probability inversely proportional to the distance traversed by the packet so far, enabling a victim source to trace back the attack source. In [21], Cheng et al. proposed a novel algorithm to detect DDoS attacks using IP address feature values using support vector machine (SVM) classification. Nguyen et al. [62] have developed an anti-DDoS framework for detecting DDoS attack proactively utilising K-NN Classifier. They used the k-nearest neighbour method to classify the network status into each phase of DDoS attack. However, while the K-NN approach is excellent in attack detection, the detector is computationally expensive for real-time implementation when the number of processes simultaneously increases. As has been indicated previously, the problem of computational intensity is critical in the DDoS problem as it is in other applications of data mining where large databases are analysed.

One of the key resources used to evaluate the performance of DDoS detection techniques is the KDD dataset. The set contains 14 attacks which is used for testing and model creation. Several methods have been proposed to extract useful features from this dataset, and a wide range of classifiers drawn from areas such as statistics, machine learning and pattern recognition have been evaluated against this dataset. For example, in Kim et al. [42], the 1999 KDD dataset was pre-processed followed by a learning and testing process. In the learning process, they used polynomial, kernel functions linear and Radial Basis Function (RBF). A classification accuracy of 93.56% was achieved. An SVM-based one-class classifier is also used to perform anomaly detection in [25]. The training data in the feature space was mapped into a new feature space. Yuan et al. [90] used the cross-correlation analysis to capture the traffic patterns and then to decide where and when a DDoS attack may possibly arise.

The following study discusses the extraction of a feature set from two different sources of datasets of Internet traffic. These are the public-domain CAIDA dataset [5] and traffic collected on the Smart and Secure Environment (SSE) Network. Various types of DDoS attacks were studied to select the packets and traffic parameters that change unusually during such attacks. Approximately 23 features were collected. Ranking these features is done with Information Gain and chi-square statistic [82] which enables the number of features to be reduced to eight. All the features used in this paper are calculated at an interval of 1 s. Since these classes are well divided as attack and normal, it is possible to apply various machine-learning algorithms for the detection. The approach taken is to use the feature selection mechanism discussed previously and build the classifier using various machine-learning algorithms such as SVM, K-NN, Naive Bayesian, Decision Tree, K-means and Fuzzy c-means clustering. This phase of the study shows an evaluation of the performance of the selected set of machine-learning algorithms in detecting DDoS attacks. The performance measures were the Receiver Operating Characteristic (ROC) curve and F-measure. An important result from this work is that, of the various methods used, Fuzzy c-means clustering is a useful and very efficient way to detect a DDoS attack.

5.3.2 Feature Selection and Evaluation

A list of 23 features, namely,

1. One-Way Connection Density (OWCD)
2. Average length of IP flow
3. The ratio between incoming and outgoing packets
4. Entropy of IP flow length
5. Entropy of the packet ratios of the three protocols TCP, UDP and ICMP
6. Ratio of TCP protocol
7. Ratio of UDP protocol
8. Ratio of ICMP protocol
9. Number of data bytes from source to destination
10. Number of data bytes from destination to source
11. Number of packets in which destination port is mapped to a particular service
12. Type of the protocol, e.g. TCP, UDP, ICMP
13. The number of packets having the same source IP address and destination IP address
14. Number of wrong fragments
15. Number of connections that have SYN errors
16. Number of connections to the same source IP
17. Number of connections having the same destination host
18. Number of packets where URG flag is set
19. Number of packets where SYN flag is set
20. Number of packets where FIN flag is set
21. Number of packets where ACK flag is set
22. Number of packets where PSH flag is set
23. Number of packets where RST flag is set

was selected for evaluation. After ranking these 23 features with information gain and chi-square statistics [82], the following 8 features [87] were selected:

(a) One-Way Connection Density (OWCD):
An IP packet without a corresponding reverting packet composes a One-Way Connection (OWC). In a sampling interval T, the ratio of OWC packets to all packets is called One-Way Connection Density (OWCD):

$$OWCD = \frac{\sum OWC\ Packets}{\sum IP\ Packets} \times 100 \tag{5.1}$$

(b) Average Length of IP Flow (L_{ave_flow}):
IP flow, a concept which is used widely in network analysis area, means that a packet set has a same five-element-group (source IP address, source port, destination IP address, destination port and protocol). Length of IP flow means that the number of packets belong to a certain IP flow:

$$L_{ave\_flow} = \frac{\sum IP\ Packets}{\sum IP\ Flows} \tag{5.2}$$

(c) Incoming and Outgoing Ratio of IP packets (R_{io}):
Normally the ratio between incoming and outgoing packets is steady. But in a DDoS attack, R_{io} increases quickly:

$$R_{io} = \frac{\sum incoming\ IP\ Packets}{\sum outgoing\ IP\ Packets} \tag{5.3}$$

(d) Ratio of TCP Protocol (R_t):

$$R_t = \frac{\sum TCP\ Packets}{\sum IP\ Packets} \tag{5.4}$$

(e) Ratio of UDP Protocol (R_u):

$$R_t = \frac{\sum UDP\ Packets}{\sum IP\ Packets} \tag{5.5}$$

(f) Ratio of ICMP Protocol (R_i):

$$R_t = \frac{\sum ICMP\ Packets}{\sum IP\ Packets} \tag{5.6}$$

(g) Land: The number of packets having the same source IP address and destination IP address.
(h) Protocol-type: Type of the Protocol, e.g. TCP, UDP, ICMP, etc.

These eight features, which have been selected based on the principles mentioned in Xu et al. [87], are used to classify the network status. Each variable is normalised to eliminate the effect of difference between the scales of the variables, as proposed by Lee et al. [47]. With normalisation, variables become

$$z = \frac{x - \overline{x}}{\sigma} \tag{5.7}$$

where x, $\overline{x}$, and σ denote the value of each feature, the mean of the sample dataset and the standard deviation, respectively.

Additionally, chi-square statistic and information gain is applied to measure the rank of each feature. The Information gain of a given attribute X with respect to the class Y is the reduction in uncertainty about the value of Y, after observing values of X. This is denoted as $IG(Y|X)$. The uncertainty about the value of Y is measured by its entropy defined as

$$H(Y) = -\sum_i P(y_i) \log_2(P(y_i)) \tag{5.8}$$

Table 5.1 Feature ranking

Features	Chi-squared rank	Information gain rank
Ratio ICMP	1	1
Land	2	2
Ratio UDP	3	3
One-way ratio	4	4
Ratio TCP	5	5
Protocol type	6	6
Average length TCP/IP flow	7	8
Ratio of in/out packets	8	7

where $P(y_i)$ is the prior probabilities for all values of Y. The uncertainty about the value of Y after observing values of X is given by the conditional entropy of Y given X defined as

$$H(Y|X) = -\sum_j P(x_j) \sum_i P(y_i|x_j) \log_2(P(y_i|x_j) \log_2(P(y_i|x_j)) \quad (5.9)$$

where $P(y_i|x_j)$ is the posterior probabilities of Y given the values of X. The information gain is thus defined as

$$IG(Y|X) = H(Y) - H(Y|X) \quad (5.10)$$

According to this measure, an attribute X is regarded as being more correlated (i.e. makes more contributions) to class Y than attribute Z if $IG(Y|X) > IG(Y|Z)$. By calculating information gain, the correlations of each attribute can be ranked to the class. The most important attributes can then be selected based on the ranking.

The chi-square statistic [82] measures the lack of independence between a feature X and a cluster Y. It can be compared to the chi-square distribution with one degree of freedom to judge extremeness:

$$x^2 = \sum_{i=1}^{r} \sum_{j=1}^{k} \frac{(A_{ij} - E_{ij})^2}{E_{ij}} \quad (5.11)$$

where r is the number of features and k is the number of clusters, A_{ij} is the number of instances for which the value of a feature is i and the value of the cluster is j, E_{ij} is the expected number of instances of A_{ij}. The larger the x^2 value, the more important the feature is to the cluster. Thus, ranking the importance of each feature with respect to the clusters based on the value of x^2 for the proposed work is considered. Based on the x^2 value, eight features, as shown in Table 5.1, are considered as important features.

Table 5.2 Samples collected

Network Data	Data type	Total number of packets
Trained	Attack (CAIDA)	9,45,372
	Normal	1,10,535
Unseen test data	Attack (CAIDA)	3,24,098
	Normal	36,485

Table 5.3 Classification results

Method used	Correct classification %
Fuzzy c-means	98.7
Naive Bayesian	97.2
SVM	96.4
KNN	96.6
Decision tree	95.6
K-means	96.7

Table 5.4 F-Measure details of classifiers

Method	TP	FP	TN	FN	F-measure
Fuzzy c-means	298	2	270	3	0.987
Naive Bayesian	290	10	256	17	0.972
KNN	280	20	243	30	0.969
SVM	282	18	253	20	0.964
K-means	285	15	273	0	0.9669
Decision tree	278	22	218	55	0.956

The first step is to extract these eight features from the dataset consisting of both normal and attack data patterns. In the experiments, a sampling frequency of 1 s was used. The next step is to train the machine-learning techniques with these datasets. In the detection phase, the same set of eight features were computed for the given network traffic, and the traffic is labelled as attack or normal based on the majority of the values computed by the machine-learning classifiers.

5.3.3 Experimental Results

The CAIDA dataset [5] was used in the experiments as the attack component. Data collected on the SSE network provided the normal traffic component. Classification of attack and normal traffic was done using an open-source tool called KNIME (Konstanz Information Miner) version 3 [1]. Table 5.2 shows details of the CAIDA dataset and the normal traffic collected on the SSE network. Table 5.3 shows the correct classification, Table 5.4 shows the f-measure details and Fig. 5.2 shows the evaluation results using ROC curves for the selected machine-learning techniques. Based on the results of these experiments, the FCM-based classification gives the best results in detecting DDoS attacks.

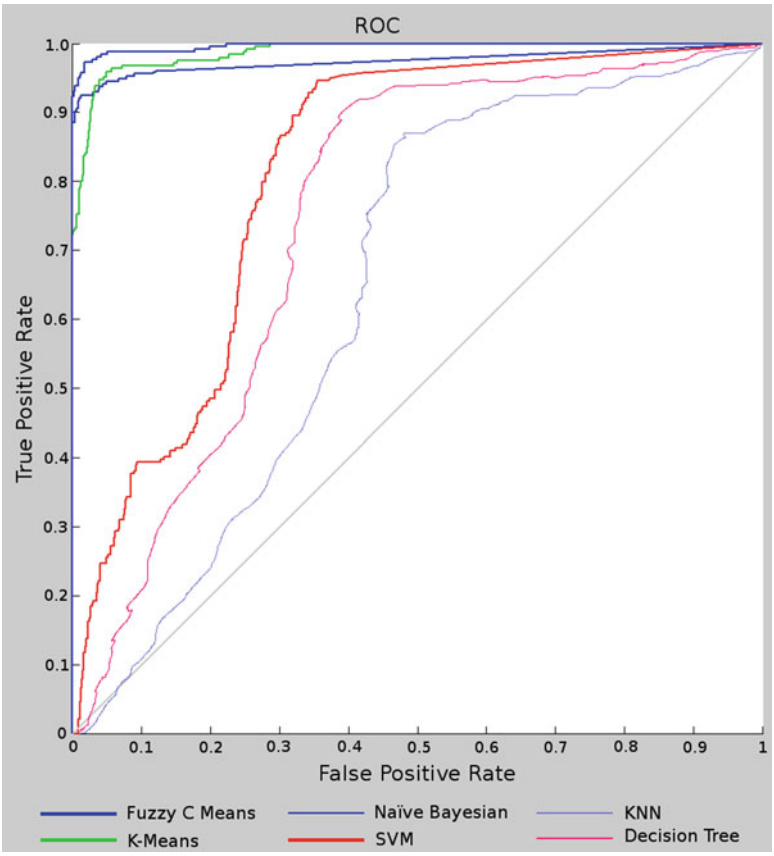

Fig. 5.2 False vs. true positive rate

5.4 DoS Detection Using Change Point Analysis (CPA) of Not Seen Previously (NSP) IP Addresses

5.4.1 Related Work

IP addresses are an integral part of communication over TCP/IP networks and are a valuable piece of information for uniquely identifying communicating entities. Given their significance, there is now a distinct body of knowledge surrounding the use of source IP address for detecting various network activities. This includes Denial of Service (DoS) and high-rate flooding attacks/DDoS [15, 21, 33, 39, 45, 67, 70, 78]. It also enables discrimination between network traffic sharing common characteristics such as anomalous network traffic and flash event [15, 39].

Central to the functioning of the Internet is its IP protocol within layer 3 of the TCP/IP suite. The IP protocol was designed for interconnection of hosts to networks without any support for the verification/authentication of IP header fields [22]. This allows attackers to inject falsified information into the IP header (such as source address spoofing), enabling them to conceal their identity and yet traverse the IP packet to the intended target. Historically, this has been a primary vector of attack. Hence, considerable amount of research has been carried out to detect such attacks with the assumption that, source address of the attack is being spoofed [29,46,50]. More recently, attackers have been able to gain control of a large number of computing resources in a so-called botnet herd. These do not need to spoof the source IP addresses. This allows them to imitate the behaviour of a legitimate user making it difficult to distinguish between normal network traffic and attack traffic [21,69,70]. An additional problem is that the traffic generated by individual bots can be varied dynamically so that the bot does not exhibit an easily predictable pattern of behaviour.

For the detection of attacks with such adaptive behaviour, it is important to use features that are difficult or impossible for an attacker to alter without being detected [69]. In this regard, considerable research has been conducted in utilising source IP address – related features as the primary means of detecting such attacks [21,39,67, 68, 70]. One of the key issues associated with using the source IP address features to identify DDoS attacks is the difficulty of storing statistical data for all 2^{32} IP addresses. This can further be complicated due to a sharp increase in rate of arrival of new IP addresses during an attack [69].

To address this scalability issue, Gil and Poletto [33] proposed a scheme called MULTIOPS, consisting of a dynamic 4-byte 256-ary tree. The proposed solution was later found to be vulnerable to memory exhaustion attack by Peng et al. [69]. The memory issue can be addressed by storing information about only a subset of source IP addresses in a set of 2^{32} available addresses such as those completing the 3-way TCP handshake or sending more than a predetermined number of packets [67, 69, 78]. A somewhat similar solution based on aggregating the source IP addresses was proposed by Peng et al. [70]. These alternatives solve the memory exhaustion (also known as scalability) issue identified previously but can still be vulnerable to similar problems under the next-generation of IP address (IPv6), where the number of addresses is many orders of magnitude larger than the IPv4 address space.

As discussed in [8], the key challenges in developing an adaptive solution for detecting DDoS attacks are:

- Operation of detection process at high speed
- Activation of response in real time to mitigate the impact of the attack

Ahmed et al. [8] provide the proof-of-concept implementation of the DMM architecture which integrates the DDoS detection and mitigation capability into a single architecture. This architecture is discussed briefly in Sect. 5.4.2, following which we present a detailed description of our detection approach.

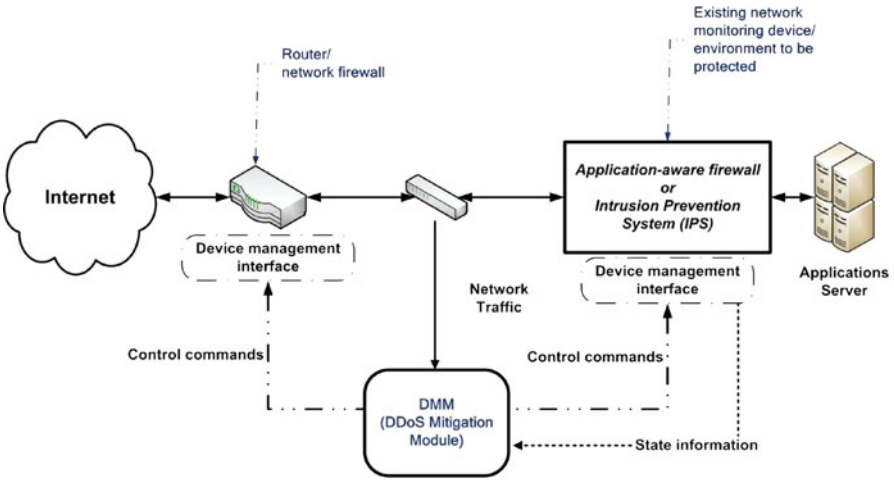

Fig. 5.3 A conceptual architecture of a DDoS Mitigation Module (DMM)

5.4.2 DMM Architecture

As has been indicated previously, reliable and timely attack detection is the crucial first step in successfully managing DDoS attacks. However, once the onset of an attack has been detected, the next step is to protect the system as a whole from failure by protecting the individual components. For example, a typical host application environment comprises not only of the actual application server but also devices whose role is to protect the application from attack. These include, for example, application-aware firewalls and intrusion prevention system (IPS). Typically, the role of the application-aware firewall is to filter packets using a set of rules based on incoming IP addresses, ports, payload, etc. An IPS performs a deep-packet analysis of network packets to detect any malicious payload targeted at the application-layer services (such as HTTP payload) again based on a set of rules. In both cases, the rule sets need to be updated dynamically so that the devices can respond to the continuously evolving threat environment. Whilst the primary focus is on protecting the applications host/server, it is equally important to protect these security devices which may themselves be at risk. In order to protect the set of components at risk, a security architecture, also known as the DDoS mitigation module (DMM) – is proposed in this chapter.

Figure 5.3 shows a schematic of a version of the proposed DMM. At a conceptual level, the design goal is for the DMM to have the capability of developing a continuous profile of the network traffic. This information is then used with a predictive model to form real-time estimates of the impact of the traffic on the processing capacity of the devices to be protected, viz., an application-aware firewall or a network-based intrusion detection system. In the event that DMM determines that the devices to be protected are at risk of failing, it would then initiate a

traffic-management strategy to mitigate the situation. For example this could take the form of using a white list to allow access only to legitimate sources whilst the network is experiencing a period of heavy load and potentially when an attack is actually in progress. This capability helps the DMM to predict and then react to imminent failure of the network security/monitoring devices to be protected. Interested readers may refer to [8] for further details. Section 5.4.3 provides a detailed description of a potential high-rate flooding attack detection algorithm used in the implementation of DMM.

5.4.3 Detection Approach

The proposed detection algorithm to operate within the DMM, consists of two functions:

- Function *ipac* for classifying IP addresses
- A *ddos* function to identify an attack

The *ipac* function extracts the source IP address of each incoming packet and determines if it is a new IP address (not seen previously (NSP)). The resulting time series is then analysed using the *ddos* function to identify if the system is under attack. The *ddos* function maintains two states: NA (not under attack) and A (under attack).

In order to learn the behaviour of network under normal operating conditions, the *ipac* classification function is first applied on normal network traffic traces without the *ddos* function being activated. After training, the *ddos* function is invoked periodically (intervals of the order of 1 to 10 s), and based on the rate of arrival of new IP addresses calculated by the *ipac* function, the *ddos* function determines a transition between two states NA and A. A description of detection algorithm used in the *ddos* function is provided in Sect. 5.4.5.

Once a transition from state NA to A signals that an attack has been detected, the *ddos* function generates a white list which is then used to define a mitigation strategy, that is, only allow traffic from IP addresses within a white list. A transition from state A to NA signals the end of an attack and will result in the relaxation of mitigation strategy i.e. stop using white list as a filtering policy. It is to be noted that in the current implementation, the white list will contain all the IP addresses observed on the targeted network during normal network conditions. The *ddos* function does as follows:

```
if (in state NA) then
    if NOT (StateChange(NA)) then //no state change
    Update White-list (Add IP address/es to white-list)
else //state change to A
     state = A
```

```
    communicate White-list to the protected security
     device
if ((in state A) then
    if (StateChange(A)) then    //state change to NA
    state = NA
    communicate to the protected security device to
      stop
    using the white-list
```

There are also two obvious limitations in this approach. The first is that when the system is deemed to be under attack, new IP addresses are treated as malicious. This can give rise to false positives. Use of other features of IP addresses including IP address distribution can be used to limit the false positives and is part of the future work. The second limitation concerns the malicious IP addresses observed during normal network operations. For example, an attacker performing reconnaissance of the target network by sending a small number of packets (such as ICMP echo requests) would result in these IP addresses being added to the white-list addresses, giving rise to false negatives, before actually flooding the target with large amounts of packets. This can be addressed not only by using the other IP address features described previously but also by using smaller historical time periods, for example, only using IP addresses observed during the last 24 h as being trusted IP addresses.

5.4.4 IP Address Classification

The implementation of the IP address classification function (*ipac*) necessitates the use of a compact and efficient data structure. Using the rate of change of new IP addresses as a primary feature to detect the onset of an attack requires keeping track of source IP addresses already observed over the network to be protected. During normal network conditions, the rate of arrival of new IP addresses is relatively low, that is, only a small number of new IP addresses appear at the user site. A substantial increase in the rate of arrival of a number of new IP addresses is usually observed during a high-rate flooding attack [69]. Keeping in view the scalability of *ipac* function during normal and attack conditions, two different implementations of the IP address classification function (*ipac*) are provided:

- Bit vector
- Bloom filter

The classification of 32-bit IPv4 addresses using a bit vector would require an array of length 2^{29} with each array element of size 1 byte. This results in 0.5 GB storage space for representing the entire set of IPv4 addresses. In the *ipac* function, whenever a packet is received, a bit at the specific position in the bit vector is marked to indicate the presence of a given IP address. Using a separate counter, the number of new IP addresses observed during an interval is calculated resulting in a time

series of the rate of change of IP addresses. Further details of *ipac* classification function using the bit-vector approach can be found in [8].

In contrast to IPv4, classification of 128-bit IPv6 addresses using a bit vector requires an array of length of at least 2^{125} with each array element of size 1 byte. The increase in address size from 32 bits to 128 bits results in a significantly larger storage requirement for representing the entire set of IPv6 addresses. One solution to the problem of scaling the *ipac* function to accommodate the storage requirement for the IPv6 address space is to use efficient data structures like bloom filters [16].

Bloom filters are data structures that offer inexact but compact representation of the elements within a set. In contrast to bit vectors, the benefits of using bloom filters are twofold. Firstly, they offer a compact representation of the elements within a given set whereas similar elements would have sparse representation if a bit vector were used. Secondly, for similar network traffic, a bloom filter can reduce the storage requirement significantly as compared to a bit vector. Given their advantages, bloom filters can be used to efficiently identify the presence or absence of elements within a set. For example, in case of IPv6 using bloom filters for membership query (in *ipac* function) of a given IPv6 address requires identification of an IP address as being seen (old) or not seen (new) previously. For a given IP address, the corresponding positions (array index) within a bloom filter are marked as set. The array indexes are identified by applying a hash function on the IP address.

The inexact and compact representation offered by the bloom filters comes at the cost of an increased number of false alarms as compared to bit vectors which have no false alarms. The bloom filter can identify an element as being within a set when it is not (collision). The false alarm rate can be reduced as a function of bloom filter size, the number of elements within a given set and the number of hash functions used. In order to identify the relationship between these three parameters, a theoretical analysis presented by Ripeanu et al. [71] has been conducted. The theoretical analysis helps analyse the relationship between these three parameters and helps in selecting the optimal size of a bloom filter for a given number of hash functions and false alarm rates. For the purpose of this discussion, different sizes of bloom filters with different numbers of hash functions for a given false alarm rate have been analysed.

Figure 5.4 shows the relationship between false positive rate, number of hash functions and bloom filter size (in terms of number of bits per set element). In this figure, the number of hash functions is plotted along the horizontal axis and the false positive rate (in log scale) along the vertical axis. Each graph represents a bloom filter of different size ranging from 10 to 40 bits per set element. In the graph legend, B represents the number of bits per set element. For example, B10 means bloom filter with 10 bits per set element. As shown in Fig. 5.4, the false positive rate can be reduced to a specific limit by increasing the number of hash functions, after which an increase in the number of hash functions as no significant effect on the false positive rate. On the basis of this analysis, a bloom filter of size 30 bits per set element with 10 hash functions is selected for experimentation (described in Sect. 5.4.6) which gives a false alarm rate of less than .00001%.

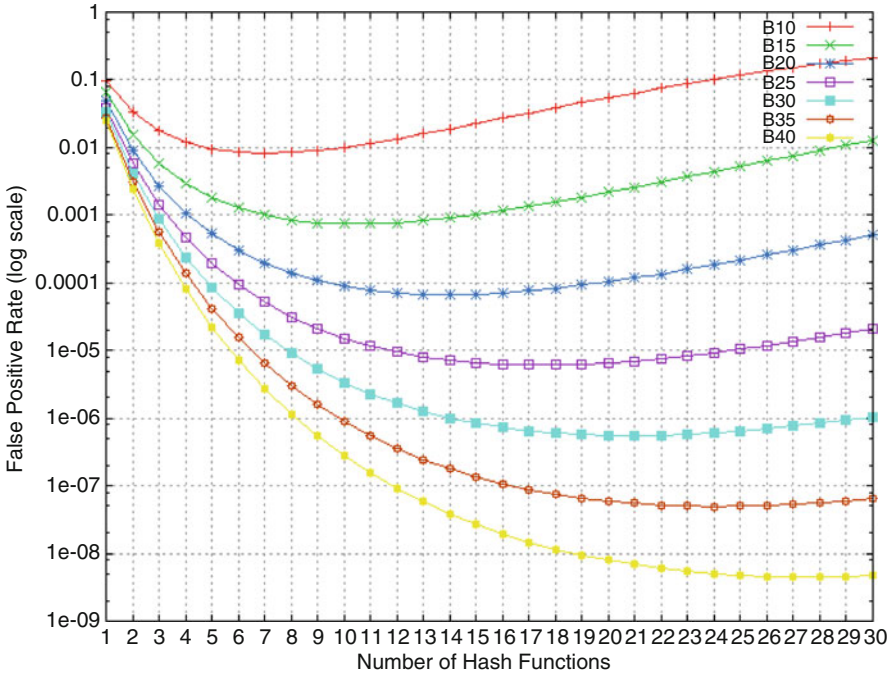

Fig. 5.4 Bloom filter: false positive rate as a function of the number of hash functions used. For example, in the case of a bloom filter with 30 bits per entry (graph third from the bottom), the false positive rate is at the minimum using 21 hash functions. Adding further hash functions does not significantly decrease the false positive rate

For the purpose of identifying the group of 10 hash functions (to be used with bloom filters) having small collision rates, an experimental analysis has been performed with 14 general purpose hash functions. A description of some of these hash functions can be found [64]. In this regard, two experiments – one without specifying the size of a bloom filter and the other with a specific size bloom filter – were conducted. In the latter case, the size was selected based on the results of the theoretical analysis performed previously; see Fig. 5.4 (a bloom filter of size 30 array indexes per set element is selected which gives less than .00001% false alarm rate). For the experimental analysis, different set sizes (number of source IP addresses) have been used with the minimum being 10 and the maximum being 400,000. Table 5.5 shows results of the number of collisions observed for different numbers of IP addresses.

To identify the maximum collision size for a given number of IP addresses, the maximum number of IP addresses that have been hashed to a same location (array index) in a bloom filter has been calculated (see Table 5.6). It was observed that almost all 14 hash functions performed extremely well with, on the average, 2–4 IP addresses being hashed to the same array index within a bloom filter. It should

Table 5.5 Number of collisions for bloom filter size 30 ranked in ascending order based on the last column

	Number of IP addresses to be stored in the bloom filter											
Hash functions	10	50	100	500	1,000	5,000	10,000	50,000	100,000	200,000	300,000	400,000
Fowler Noll Vo (FNV) algorithm	0	0	0	0	0	0	6	202	832	3,143	7,271	12,785
Arash Partow (AP) algorithm	0	0	0	0	0	2	12	238	828	3,225	7,250	12,958
Brian Kernighan and Dennis Ritchie (BKDR) algorithm	0	0	0	0	0	2	12	204	817	3,289	7,282	13,094
Robert Sedgwick (RS) algorithm	0	0	0	0	0	0	4	168	793	3,342	7,449	13,188
Rotation (ROT13) algorithm	0	0	0	0	0	0	10	232	891	3,267	7,386	13,201
Cyclic Redundancy Check (CRC32) algorithm	0	0	0	0	0	0	6	212	898	3,499	7,498	13,216
Justin Sobel (JS) algorithm	0	0	0	0	0	4	12	215	828	3,389	7,597	13,236
Bob Jenkins one-at-a-time algorithm	0	0	0	0	0	2	10	214	886	3,257	7,498	13,245
Leonid Yuriev (LY) algorithm	0	0	0	0	0	2	8	244	821	3,309	7,584	13,287
Daniel J. Bernstein (DJB) algorithm	0	0	0	0	0	2	10	216	891	3,587	7,952	14,272
Donald E. Knuth (DEK) algorithm	0	0	0	0	0	2	30	334	1,279	5,023	9,807	16,409
Substitute DBM (SDBM) algorithm	0	0	0	0	0	0	14	288	1,520	6,335	14,487	25,416
Executable and Linkable Format (ELF) algorithm	0	0	0	0	0	0	10	378	1,625	6,752	14,652	26,468
BP algorithm	0	0	0	6	14	329	1,333	25,377	73,059	182,172	290,372	395,012

Table 5.6 Maximum collision size for bloom filter size 30: the maximum number of IP addresses that have been hashed to a same location

	Number of IP addresses											
Hash Functions	10	50	100	500	1,000	5,000	10,000	50,000	100,000	200,000	300,000	400,000
Fowler Noll Vo (FNV) algorithm	1	1	1	1	1	1	2	2	2	3	3	4
Arash Partow (AP) algorithm	1	1	1	1	1	2	2	2	2	3	3	3
Brian Kernighan and Dennis Ritchie (BKDR) algorithm	1	1	1	1	1	2	2	2	3	3	4	4
Robert Sedgwick (RS) algorithm	1	1	1	1	1	1	2	2	3	3	4	4
Rotation (ROT13) algorithm	1	1	1	1	1	1	2	2	3	3	3	3
Cyclic Redundancy Check (CRC32) algorithm	1	1	1	1	1	1	2	2	2	3	4	4
Justin Sobel (JS) algorithm	1	1	1	1	1	2	2	3	3	3	3	4
Bob Jenkins one-at-a-time algorithm	1	1	1	1	1	2	2	2	3	3	3	3
Leonid Yuriev (LY) algorithm	1	1	1	1	1	2	2	2	3	3	3	3
Daniel J. Bernstein (DJB) algorithm	1	1	1	1	1	2	2	2	3	4	4	4
Donald E. Knuth (DEK) algorithm	1	1	1	1	1	2	2	2	3	4	4	4
Substitute DBM (SDBM) algorithm	1	1	1	1	1	1	2	2	3	4	4	4
Executable and Linkable Format (ELF) algorithm	1	1	1	1	1	1	2	3	3	4	4	5
BP algorithm	1	1	1	2	2	3	4	9	11	15	21	26

be noted that each hash function has been applied independently. Applying all hash functions on each source IP address and then setting the corresponding bits in a bloom filter has not been done and is a part of the future work.

Based on the experimental analysis, the first 10 hash functions have been used in the implementation of the DMM. In the current implementation, the user has been given the choice of selecting a number of hash functions (from 1 to 10) and the size of the bloom filter. The program captures the network traffic in real time and performs the analysis of the network traffic for a given measurement frequency (duration specified in terms of seconds).

The current implementation makes use of a counting bloom filter [26], which uses an array of n counters instead of bits. The counters track the number of elements currently hashed to that location. In counting bloom filters, the counters should be large enough to avoid overflow. Although it has been observed that a 4-bit-long counter is enough for the majority of the applications [26], an 8-bit counter has been selected in the current implementation of the DMM. Currently, the counting bloom filter is being used as a traditional bloom filter and can be enhanced in the future to minimise the number of collisions. This can be achieved by using linked lists together with a counting bloom filter to exactly identify the source IP addresses being hashed to a given array index.

5.4.5 DDoS Detection

An outbreak of anomalous activities including DDoS/high-rate flooding attack results in an increase in the rate of unsolicited packets arriving at the victim machine. This increase in the number of unsolicited packets usually results from a large number of compromised hosts sending voluminous traffic to the target. This results in not only an increase in traffic volume but also an increase in the number of sources sending such traffic. The problem of identifying malicious network activity in the presence of DDoS/high-rate flooding attacks can thus be formulated as a change detection problem. This involves identifying the change in the statistical properties of the network traffic parameter under investigation, that is, the rate of arrival of new IP addresses.

This abrupt increase in the rate of arrival of new IP addresses is being used in the DMM as a key observation in identifying and investigating potential malicious network activity. For change detection, a sliding-window-based non-parametric CUSUM proposed by Ahmed et al. [1, 7] has been used. The *ddos* function in DMM takes input from the IP classification function (*ipac*). The bloom filters in *ipac* identify the number of new source IP addresses during a given measurement frequency which then calls the *ddos* function (change detection algorithm) to identify an attack.

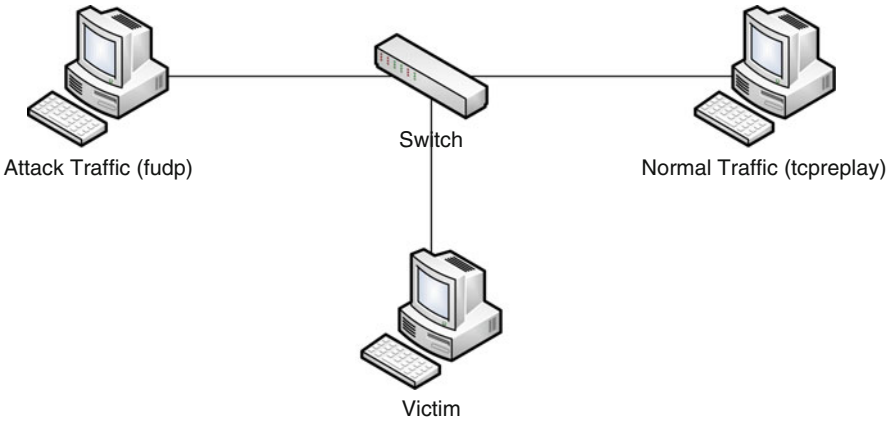

Fig. 5.5 The testbed architecture

5.4.6 Experimental Results

A proof-of-concept evaluation of the proposed Not Seen Previously (NSP) and CPA algorithms has been performed with both bit vector and bloom filter classification techniques. The testbed shown in Fig. 5.5 was used to evaluate the NSP algorithm using bit vector and bloom filter classification techniques. A detailed description of the testbed is provided in Chap. 4.

For the purpose of the experiments, it is necessary to construct a dataset with two distinct components, viz., background/normal traffic and the anomalous traffic. The background network traffic component is drawn from the data collected by the University of Auckland (Auckland VIII) [3]. It comprises 20 h of continuous bidirectional real-world network traffic collected on 12-03-2006. The IP addresses in the traffic trace have been mapped into 10.*.*.* using one-to-one hash mapping for privacy. Before using the data, it is required to be pre-processed. The dataset is first analysed to select the traffic destined for the busiest web server (destination IP address 10.0.0.63). The selected traffic is then processed to remove TCP flows with less than three packets (SYN attacks) and also flows with no data from the web server. The processed data is then converted into a one-way incoming traffic to destination web server 10.0.0.63.

For the analysis of NSP algorithm using bit-vector classification technique, the processed data is reproduced over the testbed using TCPREPLAY utility. The traffic is replayed at the rate of around 300 packets per second. The attack traffic component was constructed using the FUDP utility. It was used to flood the victim machine (see Fig. 5.5) with UDP packets having varying number of spoofed source IP addresses ranging from 35 to 150 sources. The rationale for using a different number of source IP addresses for the attack traffic was to enable an analysis of the behaviour of the proposed algorithm under different attack conditions. It should be noted that attacks with a large number of source IP addresses (e.g. UDP flooding attack with 150 sources per second) are trivial to detect as compared to attacks

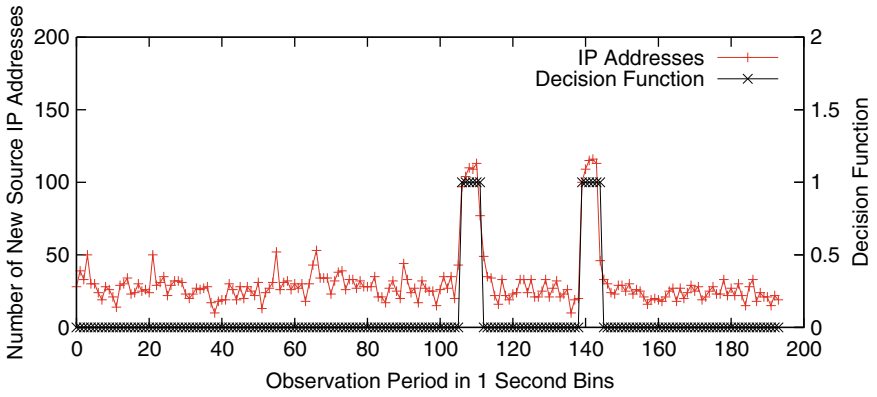

Fig. 5.6 Successful detection of two UDP flooding attacks using CPA (right y-axis). Each attack uses 75 source IP addresses vs. ca. 25–35 source IP addresses in the background traffic (left y-axis)

with a small number of sources. The average attack traffic rate is set to around 3000 packets per second (10 times the normal background traffic). The algorithm successfully detected all the attacks and generated a list of IP addresses observed under normal network conditions. A detailed analysis and evaluation of bit-vector-based NSP algorithm can be found in [8].

For the analysis of NSP algorithm using bloom filter classification technique, the processed data described above is replayed using TCPREPLAY utility at the rate of 4 Mbps, that is, nearly 6,500 packets per second. The attack traffic component was constructed using the FUDP utility with varying number of source IP addresses, including 75, 100 and 300 sources per second, being generated. Based on the theoretical analysis of the bloom filters described previously, for these experiments 10 hash functions with 30 bits per set element were used. This would give an expected false positive rate of less than .00001%. Unless otherwise specified, a measurement frequency of 1 s is used in all the experiments. Figure 5.6 shows the result of UDP flood using 75 unique source IP addresses.

In Fig. 5.6, the horizontal axis represents the observation period in 1 s bins, the left vertical axis is the total number of new source IP addresses in the measurement interval and the right vertical axis represents the CPA decision function with 1 being attack and 0 being no attack. Two instances of UDP flooding attacks were generated, as shown in the figure, and both attacks were successfully detected. The NSP algorithm was able to generate a list of legitimate IP addresses also known as the white list (IP addresses appearing under normal network operations). It is to be noted that the detection delay is bound by the measurement interval which in this case is 1 s and can further be reduced using smaller intervals.

Figure 5.7 shows the result of flooding attack with 100 source IP addresses. Similarly, Fig. 5.8 shows the result of UDP flooding attack with 300 source IP addresses. For UDP floods with 100 and 300 source IP addresses, only one attack instance was generated. The proposed algorithm was successful in identifying attacks in both cases.

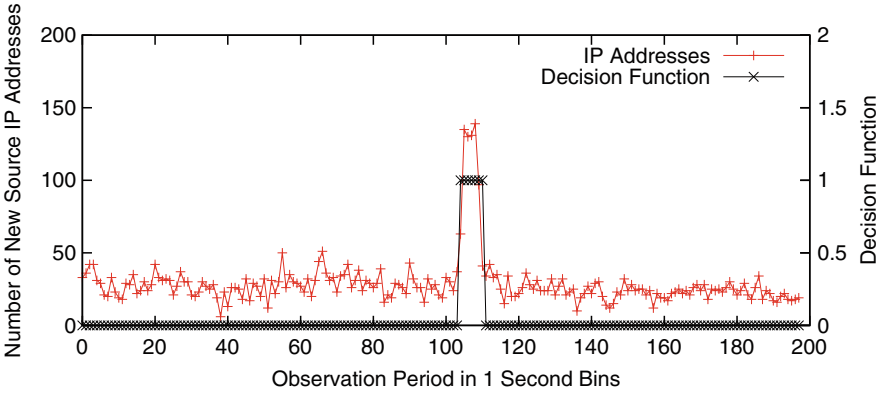

Fig. 5.7 Successful detection of UDP flooding attack using CPA (right y-axis). The UDP attack uses 100 source IP addresses vs. ca. 25–35 source IP addresses in the background traffic (left y-axis)

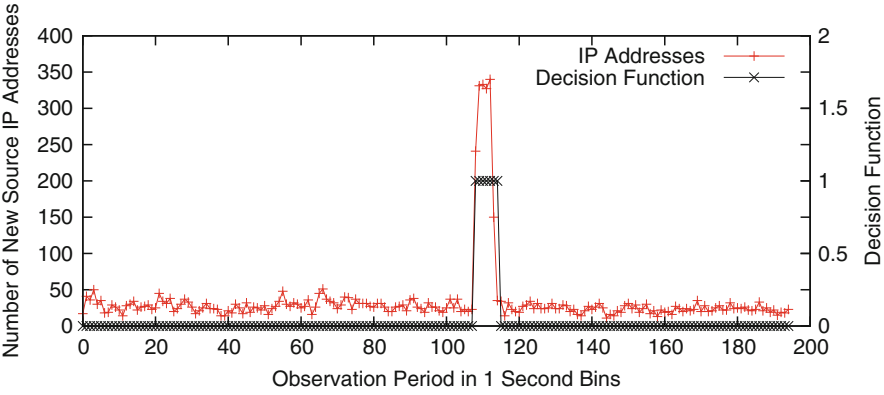

Fig. 5.8 Successful detection of UDP flooding attack using CPA. The UDP attack uses 300 source IP addresses vs. ca. 25–35 source IP addresses in the background traffic (left y-axis)

5.5 DoS Detection Using Naïve Bayesian Classifiers

5.5.1 Related Work

Recently, a substantial amount of work has been done on using various machine-learning techniques and other general statistical approaches in the detection of DoS attacks. In these works, DoS attacks have been classified into the broad category of intrusion attacks. Hence, solutions have been oriented more towards formal

intrusions, including root escalation, scripting attacks, etc. A major factor that is often missed by classifying DoS attacks into intrusion attacks is the enormous volume that DoS detection solutions have to handle in comparison with intrusion attacks. While computationally intensive models could be used to detect intrusions, this extreme processing workload renders learning models such as hidden Markov models (HMMs) and ANNs impractical when it comes to real-time attack detection. These models would either be too space intensive or too time intensive. The detection mechanism for DoS attacks is required to be ultralightweight to support 'wire speeds'.

Another practical drawback of supervised/unsupervised learning models for DoS detection is the accuracy of (supposedly normal) traffic used in the building of the models. While it is common to assume that all traffic used is absolutely normal, for data collected at actual user sites, this may not necessarily be true and the training data may itself contain abnormalities. This may cause false negatives when the model is applied to other real-world data.

Another common problem for the entire research community on DoS attacks is the lack of data in all forms and, in particular, the unavailability of training data. Currently, the DoS research community is heavily dependent on two standard datasets, that is, the KDD dataset and the DARPA dataset [4, 34], for the purpose of learning and analysis. However, these have a number of inherent limitations and are better suited to the analysis of intrusion attacks [79].

The objective of any practical system to detect DoS attacks-including high rate flooding DDoS attacks, should be to ensure that the resultant detection system is (a) light weight, (b) accommodative of practical difficulties in learning, and (c) operable at close to line speeds.

Proposed DoS/DDoS detection techniques differ from each other in terms of the objectives to which they cater, the strategy used, the features chosen and the (reported) performance. These detection techniques include statistical approaches like [28] which proposes a chi-square-test on the entropy values of the packet headers. Jin and Yeung [37] discusses the effects of multivariate correlation analysis on DDoS detection and presents an example of detecting a SYN flooding attack. Jin and Yeung [38] uses a covariance matrix to present the relationship between each pair of network features in order to identify an attack. In [92] the CUSUM algorithm is used to detect a change in the number of packets directed towards the destination to identify the attack. Other techniques taken from pattern analysis and machine learning have also been proposed in the literature. For example, Xie et al. [86] consider application layer DDoS attacks and use hidden semi-Markov models to detect these types of attacks. Other classification algorithms such as support vector machines [72], genetic algorithms [75], artificial neural networks (ANN) [32, 85] and Bayesian learning [54] have also been applied. Hybrid modelling techniques such as [75] also provide interesting results. Hidden Markov model–based DoS attack solutions have also been proposed in [40, 88]. A taxonomy of DDoS attacks and defence mechanisms has been documented in [58]. Recent works [27, 41] have

discussed the use of Bayesian classifiers towards intrusion detection, in general, which includes DoS attacks. Some especially relevant works are as follows:

- In [88], the authors model the rate of change of new and old IP addresses at nodes in the network using an hidden Markov model (HMM) and determine attacks based on the above characteristic. They also explain how the results appear to improve with attack information exchange between distributed nodes using Cooperative Reinforcement Learning techniques. The solution is a target end solution.
- Seo et al. [72] use the traffic rate analyser (TRA) to model the TCP flag rate which is a measure of the proportion of packets with chosen flag set to the total number of TCP packets. In addition, they use the protocol rate which is a measure of the proportion of packets from a selected protocol to the total number of incoming packets. Ten features, each representative of either of the above, are monitored and separately modelled for both normal packets and attack packets as a series of SVMs. Attacks are labelled as DoS, DDoS or DrDoS attacks based on the class to which the pattern belongs. The solution is a source end solution.
- Jin et al. [37] present a generalised view of multivariate correlation analysis for DDoS detection using SYN flooding attacks as an example. The idea projected in the paper is to differentiate between normal and attack traffic by considering the correlation between possible pairs of selected features among a set of features $f_1 \ldots f_p$ at different intervals, and comparing the distance between this correlation matrix and the average of correlation matrices seen during training. The authors consider the six flags in the TCP field as separate features and show the effectiveness of the method under these conditions. The solution is a target end solution.
- Kang et al. [40] proposed a solution which is deployed at the source end and uses three features, viz., a three-tuple {*source IP, destination IP, destination port*}, a 16-bit packet identifier and the TCP header flags on the same dataset. Independent HMMs is trained for each of these three features and then the HMMs are coupled to form the multi-stream fused HMM. If the probability of the traffic pattern is smaller than the threshold probability, the pattern is flagged as anomalous, and a suitable thresholding on the proportion of anomalous patterns to total patterns determines an attack. The solution is a source end solution.

5.5.2 Detection Approach

The detection approach described here is likewise to be situated and to operate within the DMM architecture described in Sect. 5.4.2, as is the case with the CPA NSP approach presented in Sect. 5.4.

While signature-based detection techniques use signatures to identify attacks, anomaly-based detection models build models of normal traffic and determine attacks as traffic which deviates significantly from these models. The latter is more suited to detecting flooding attacks and hence was chosen.

General design elements

- **(Traffic separation)**. Network traffic needs to be separated into streams in order to facilitate applying DoS handling techniques in the case of an attack. We define a stream by the two-tuple value *destination ip, destination port*.
- **(Windowing)**. Windowing essentially means splitting the input traffic into traffic subsets which fit into logical entities called windows. Windows may either be time windows or packet windows. Packet windows were chosen in order to decrease the reaction time and also to have control on the number of events that have to be modelled.
- **(Flagging an attack)**. Flagging an abnormal situation as an attack is a function of a series of abnormal windows, not necessarily consecutive. For example, if an attack were to be flagged after observing five consecutive abnormal windows, the attacker can cleverly escape attack detection by keeping the attack traffic just below 5. The target would still be reasonably safe from this action by the attacker. However, the detection mechanism will miss such attack attempts in the build-up of an attack. In order to overcome this problem, a step-based mechanism is used. At any point (until a reasonable timeout period) in the course of deployment, if the number of abnormal windows goes beyond a particular number (quantified as a parameter called abnormal window count (AWC)), then the system flags an attack. This will ensure that such attacks will be caught.

5.5.3 Modelling TCP Traffic

For attack detection, the modelling is done based on the protocol headers. The choice of parameter is driven by the fact that it should be able to differentiate between a normal and attack pattern. Among a set of various other header parameters, TCP flags appear to be best suited. At the time of selecting parameters, it should also be noted that keeping down the number of such parameters will reduce the computation load on the detection system.

In [80], the authors present a detailed account of the proposed methodology for TCP traffic modelling. Central to modelling TCP traffic is the TCP flags field, available as a field in the TCP header. The TCP flags field is a collection of 8-bits, each bit representing one flag. Individual flags or combinations of flags symbolise specific actions in TCP – for example, connection establishment, connection closure, requesting data, etc. The different TCP flags are: SYN, ACK, PSH, FIN, URG, RST (standard TCP flags) and ECE, CWR. The last two flags, it was observed, are seldom used in contemporary Internet traffic, and so we are concerned only about events raised from six flags. Hence, there are 2^6 different observables. A packet window is technically defined as a collection of flag sets observed with every packet in the window. The probability of a given packet window is a function of individual probabilities of different flag sets observed inside the window. Even though some of these flag sets may be dependent on each other (TCP is a causal system), inside small windows (in comparison to the total training traffic) the dependency between individual flags may not exist and are assumed to be totally independent.

However, TCP traffic is observed to be highly skewed in nature. This is evidenced by the presence (and prescription) of only very few flag combinations in inbound traffic. This gives scope to modelling traffic with fewer than 2^6 events as mentioned above. Also inducing some form of valid grouping will reduce the number of events and will be more space efficient. After experiments on various datasets to determine a reasonable grouping of these flags, the following groups were arrived at:

1. T_1: Packets with RST bit set (irrespective of other bits) – 32 packet types
2. T_2: SYN packets – 1 packet type
3. T_3: ACK packets – 1 packet type
4. T_4: FIN/ACK packets – 1 packet type
5. T_5: PSH/ACK packets – 1 packet type
6. T_6: Rest of the packets – 28 packet types. Includes seldom used packets and invalid packets

For a packet window of size N, there are $(N + 1)$ instances to be monitored, thereby making the total number of instances in probability space (and hence the total number of floating points in memory) $6 * (N + 1)$. However, as mentioned above, many instances are very unlikely to occur in normal traffic because TCP traffic is highly skewed in terms of its use of TCP flags. Hence, a fewer number of events could be sufficient in modelling the traffic, and this offers a reason for choosing NB classifiers with simplistic assumptions. However, the existence of instances that are unlikely to occur may result in events with zero probability. To avoid zero probability events, a simple technique like Laplacian smoothing (used for speedy smoothing) can be used. However, this may produce inaccurate results when applied to a large number of instances, particularly given that we started with the assumption of working with smaller amounts of training data.

We reduce the $N + 1$ number of possible events per observable group (N is the packet window size) by using a constant number of bands K, where each band groups together events of similar probability. This reduces the number of zero probability events and also improves smoothing. To achieve this, the following procedure is used:

1. Observe occurrences for each event and consider jumps between event occurrences in descending order. Arranging events in descending order of their occurrences brings like events closer. This will help grouping like events together. The jump between two observations of events in the array is denoted as $J_i = O_i - O_{i+1}$.
2. Consider top $K - 1$ jumps in J_i s and form K bands bordering them.
   To illustrate:
   Band grouping – An Example

   - Let the set representing number of windows during learning against each event (from $n = 0$ to $n = 10$) be $A = \{500, 291, 271, 36, 222, 111, 1211, 3, 1, 31, 1\}$, corresponding to the packet type T_i. Let the fixed number of bands be $K = 5$.
   - Sorted array $A_s = \{1211, 500, 291, 271, 222, 111, 36, 31, 3, 1, 1\}$, which brings like events closer.

- Jump array $J = \{711, 209, 20, 49, 111, 75, 5, 28, 2, 0\}$.
- Top 4 jumps $J_s[0] \ldots J_s[3] = \{711, 209, 111, 75\}$.
- Index array $I = \{0, 1, 4, 5\}$.
- Sorted index array $I_s = \{0, 1, 4, 5\}$.
- Band groups: $Bg_1 : A_s[0]$ (1 value), $Bg_2 : A_s[1]$ (1 value), $Bg_3 : A_s[2] - A_s[4]$ (3 values), $Bg_4 : A_s[5]$ (1 value), $Bg_5 : A_s[6] - A_s[10]$ (5 values).
- Bands[indices]: B_1: {6}, B_2: {0}, B_3: {1, 2, 4}, B_4: {5}, B_5: {3, 9, 7, 8, 10}.

After clustering the $N + 1$ instances per observable group T_i into K clusters, we have reduced the storage requirements from 6*($N + 1$) to 6*K floating points for the detection system. Interested readers may refer to [80] for further details.

5.5.3.1 Training Goals for TCP

In essence, the objectives of the training phase for TCP traffic are as follows:

- To group the flag instances into bands for each observable.
- To learn events from input traffic based on the above set of events and determine probabilities for each event.
- To determine an *appropriate* threshold probability, a probability below which a window will be classified as abnormal.

5.5.3.2 Setting Threshold Probabilities

The important consideration for setting threshold probabilities is the assumption that abnormal traffic may be present, although in small quantities, within the training traffic. Based on this consideration, a technique called cross-validation is used, along with sieving of the lower t percent of probabilities during threshold determination. The input parameter t is the sensitivity factor or error proportion which denotes the proportion of abnormal traffic which may be present in the training traffic.

5.5.4 Modelling UDP Traffic

The objective of monitoring protocol headers is to evolve a distinguishing pattern in them, which will subsequently help detecting DoS attacks against the victim. A characteristic feature of UDP is that it is connection-less, and hence ensures speedy communication. However, unlike TCP, the smaller UDP header does not contain fields like flags, which will determine the state of the communication. Since UDP is connection-less, most DoS attacks performed using UDP are only bandwidth-based attacks, essentially trying to exhaust bandwidth resources available to the server. Consequently, UDP header information cannot be used as a critical parameter for detecting DoS attacks using UDP.

Hence, the parameter considered here is the window arrival time (WAT) of a packet window. WAT of a packet window is the duration in which a packet window has arrived. Technically, it is the difference in time between packets P_1 and P_N of a window, where N is the size of the window. Due to various constraints, only non-overlapping windows are considered. During the training phase, the WATs of windows are monitored, and a model evolved to accommodate these WAT events. During the deployment phase, the model probabilities are used to determine the probability of an incoming window. If the probability is less than a threshold probability (determined from of the learning input itself), then the window is classified as abnormal.

As is the case with TCP, the events are so sparse in nature that there appears to be a lot of scope to group these events in order to reduce the number of probabilities handled by the model. This requires that WATs be further grouped into a constant number of bands defined by time bounds. Hence, each band is a collection of a contiguous range of WATs. The probability of a band will be common to all WATs falling under the band. During deployment, the WAT of each window is computed, and the probability of the window is the probability of the band inside which the WAT falls. If this probability were to be smaller than a threshold probability, then the window is considered abnormal. The threshold probability is determined by adopting cross-validation methods similar to that done in TCP.

5.5.4.1 Training Goals for UDP

The primary goals of the training phase with respect to UDP are as follows:

1. To compute per UDP stream, the upper and lower intervals of each band of intervals of WATs, given that a constant number of bands(K) exist. For example, if the window size N = 100, then the different bands could be $B_1 = (0 - 10)$ s,[1] $B_2 = (11 - 235)$ s, $B_3 = (236 - 499)$ s and so on. It should be noted that the band intervals span the entire time.
2. To compute per stream, the probabilities of each of these bands (e.g. probability of WAT in B_2, etc).
3. To compute per stream, the dynamic threshold probability associated with the stream. This threshold probability is the probability below which the window will be considered abnormal.

The definition of a band is fundamentally different for UDP due to the way UDP traffic is modelled. The variables *upperbound* and *lowerbound* are the time bounds for the band of WATs, and the variable probability denotes the probability of WAT falling into the time band.

[1] Unit of time is assumed to be seconds.

The problems in the case of UDP are very similar to that of TCP, and similar mechanisms are adopted.

5.5.4.2 UDP Bands

Given an array of numbers WATs with size of array equivalent to the total number of windows during learning, $A = \{a_0, a_1, a_2, a_3, \ldots a_{TW}\}$, where TW = The total number of windows seen during learning divide the array of WATs into B time bands.

The traffic is modelled in terms of the WAT, that is, the time taken for a (non-overlapping) window of packets to arrive. An approach similar to the one used in TCP is utilised. In this case, only contiguous time ranges are considered for grouping since the learning traffic may not have captured all time variations. This is different from TCP where there is only a finite set of events to model. However, the approach of making contiguous bands has a drawback – inaccuracy of estimation of probabilities is possible if events with like probabilities are distributed non-contiguously. This was the case in TCP where equally likely events were put into different bands and were estimated by different probabilities. However, it is assumed that in the case of UDP, consecutive WAT events (e.g. WAT = 500, WAT = 501, WAT = 502, etc.) are most likely to be equally probable, and hence this method of dividing along the X-axis can be adopted.

In summary, a simple clustering method is used to solve the problem, as follows:

1. Observe occurrences for each event and consider jumps between event occurrences in descending order. Arranging events in descending order of their occurrences brings like events closer, which will help grouping like events together. The jump between two observations of events in the array is denoted as $J_i = O_i - O_{i+1}$.
2. Consider top $B - 2$ jumps in J_i s and form B bands bordering them.

5.5.4.3 Band Expansion

The above algorithm describes the method of clustering and grouping UDP traffic based on WATs of incoming windows. However, it has to be noted that the exact boundaries of these bands may not be accurately known from the WATs observed during learning, since the learning traffic may not have captured all possible values of WATs. Hence, there should be a mechanism for meaningful expansion of bands in order to accommodate events which have not occurred before. Only after expansion can the entire time be spanned. A bad choice of expansion strategy can result in inaccurate probabilities.

A simple strategy to expand bands is followed: The difference between the upper bound of a band and the lower bound of its neighbouring band (the conflict zone) is equally shared between the consecutive bands. Hence, every dynamic band (bands except the first and the last band, which are already expanded to accommodate unseen events) can be expanded on both its bounds to accommodate the entire time.

5.5.4.4 Updating of Learning Probabilities

Updating of learning probabilities for UDP streams is straightforward, after time bands have been frozen. It involves re-counting the number of windows in the learning traffic falling under each band (according to the WAT of the window) and estimating the probability of each band as the number of windows seen in the band by the total number of windows seen during learning. It is to be seen that the window counters against each band have to be smoothed before estimating probabilities.

5.5.4.5 UDP Threshold Probability Determination

Given learning traffic statistics for a site, the objective here is to determine the threshold probability. The same strategy of 10-fold cross validation is employed to arrive at a threshold probability for UDP streams based on the learning traffic. Vijaysarathy et al. [80] describe in detail the cross-validation technique for TCP. The probabilities, in this case, would be the UDP window probabilities.

5.5.5 Experimental Results

Experiments were conducted primarily on the following datasets:

1. DARPA dataset for TCP – Consists of many streams at various levels of traffic. The stream with the most number of packets was chosen. The stream was a telnet stream with roughly 70,600 inbound packets.
2. SETS dataset for TCP – This represents the user dataset which contains traffic addressed to the SETS web server on port 80. The stream consists of roughly 240,900 inbound packets.
3. SETS UDP dataset for UDP – This represents the user dataset for UDP which contains traffic addressed to the SETS internal DNS server on port 53. The stream consists of roughly 60,400 packets.

Critical system parameters, namely, accuracy (Acc), false alarm rate (FAR), and miss rate (MR) were computed based on the number of true and false predictions. To understand the effect of window size on the problem, experiments were conducted with different window sizes. The results are tabulated in Tables 5.7 and 5.8.

Table 5.7 Experimental results for TCP

	DARPA at t =1%			SETS at t =5%		
WS	Acc	FAR	MR	Acc	FAR	MR
50	98.3%	2.3%	0	97%	7%	0
100	98.6%	2%	0	97.4%	6.2%	0.06%
200	98.7%	1.8%	0	97.1%	5%	1.1%

Table 5.8 Experimental results for UDP

	SETS at t =1%		
WS	Acc	FAR	MR
100	99.5%	0.4%	0.1%
200	99.2%	0.5%	0.6%

For the tagged DARPA dataset and at an error proportion of 1%, the accuracy (Acc) improves with increasing window size. There is a corresponding decrease in the false alarm rate (FAR) and the miss rate (MR) is zero, that is, no attacks were missed. For the untagged SETS dataset, it is safe to assume a slightly bigger error proportion of 5%. In this case, the false alarm rate falls with increasing window size but the miss rate increases marginally. Hence, although untagged datasets show a slightly different pattern in performance, it can be concluded that larger window sizes are bound to improve system performance.

5.6 DoS Detection Using CUSUM and Adaptive Neuro-Fuzzy Inference System

Our approach in this work has been to apply the CUSUM algorithm to track variations of the attack characteristic variable *X(n)* from the observed traffic (specific to different kinds of attacks) and raise an alarm when the cumulative sum gets too large (based on a threshold value). But often this threshold mechanism produces many false alarms. A fuzzy inference system (FIS) can be employed instead of a threshold-based system as it removes the abrupt separation between normality and abnormality and thereby reduces the number of false alarms comparatively. The output of a FIS depends on the membership function chosen and its parameters. Hence, for a better output it is important to appropriately select the membership function parameters; ANFIS serves as a suitable technique for this purpose. In ANFIS, the membership function parameters of FIS are fine-tuned based on training data collected in a real-time environment.

5.6.1 Related Work

DoS/DDoS defence mechanisms can be broadly categorised based on the attack detection strategy into pattern detection and anomaly detection. Further, in anomaly

detection we have two Normal Behaviour Specifications: Standard and Trained [4]. In pattern detection mechanism, the signatures of known attacks are stored in a database, and each communication is monitored for the presence of these patterns. The drawback here is that only known attacks can be detected, whereas attacks with slight variations of old attacks go unobserved. On the contrary, known attacks are easily and unfailingly detected, and no false positives come across. Snort [2] provides some examples of a DDoS detection system that uses pattern attack detection. An identical approach has been supportive in controlling computer viruses also. Similar to virus detection software, signature databases must be frequently updated to account for new attacks.

Methods that set up anomaly detection have a representation of normal system behaviour, such as normal traffic dynamics or anticipated system performance. The current state of the system is periodically compared with the models to detect anomalies. Mirkovic et al. [58] proposed a technique called D_WARD (DDoS Network Attack Recognition and Defence). It is a source end solution whose goal is to autonomously detect and stop outgoing attacks from the deploying network. It provides a dynamic response that is self-adjusting. By carefully choosing the criteria for adjustment, D_WARD is able to promptly react to network conditions while being resilient to an attacker's attempts. It is efficient in autonomous mode. However, the authors note that their approach sometimes results in wrong actions.

Mahajan et al. [53] proposed the aggregate congestion control (ACC) system, a mechanism which reacts as soon as attacks are detected, but does not give a mechanism to detect ongoing attacks. For both traffic monitoring and attack detection, it may suffice to focus on large flows. XenoServices [89] is an infrastructure for a distributed network of web hosts that responds to an attack on any website by replicating the website rapidly and widely among the XenoService servers, thereby allowing the attacked site to acquire more network connectivity to absorb a packet flood. Although such infrastructure can ensure QoS during DDoS attacks, it is doubtful that a large number of ISPs will adopt such an infrastructure quickly. Multops [33] proposes a heuristic and a data structure that network devices maintain a multi-level tree, monitoring certain traffic characteristics and storing data in nodes corresponding to subnet prefixes. The tree expands and contracts within a fixed budget. The attack is detected by abnormal packet ratio values and offending flows are rate-limited. The system is designed so that it can operate as either a source-end or victim-end DDoS defence system.

Methods presented in [33, 53, 59, 89] provide examples of anomaly detection approaches. The advantage of anomaly detection over pattern detection is that previously unknown attacks can also be discovered in anomaly detection.

Based on the specification of a normal behaviour, we can segregate anomaly detection mechanisms into standard and trained. Mechanisms that employ standard specifications of normal behaviour depend on some protocol standard or a set of rules. For example, the TCP protocol specification describes a three-way handshake that has to be performed for TCP connection establishment. An attack detection mechanism can make use of this specification to detect half-open TCP connections and discard them from the queue. Some protocol stack enhancement approaches

like SYN cookies and SYN cache are proposed, as described in [48] and [14, 15] respectively, to mitigate attacks like the SYN flood which uses TCP protocol. The advantage of a standard-based specification is that it produces no false positives; all legitimate traffic must meet the terms of the specified behaviour. The disadvantage is that attackers can still perform complicated attacks which, on the surface, seem amenable to the standard and hence unnoticed.

Mechanisms that use trained specifications of normal behaviour observe network traffic and system behaviour and produce threshold values for different parameters. All connections exceeding one or more (depending on the approach) of these values are considered as anomalous. One such widely used threshold-based approach is the CUSUM algorithm. Wang et al. [81] proposed a detection mechanism very specifically for SYN flood attack; Peng et al. [67] proposed a detection mechanism by monitoring Source IP address; Zhou et al. [92] proposed a detection based on CUSUM and space similarity at each host in a P2P network; Leu et al. [49] proposed an intrusion prevention system named Cumulative Sum–based Intrusion Prevention System (CSIPS). Though all the above-mentioned work have their own novel ideas, one common drawback is the selection of threshold, which is important since selection of low thresholds lead to a lot of false positives, whereas high thresholds reduce the sensitivity of the detection mechanism. For this reason, in our proposed approach, more intelligent ANFIS engines are used.

5.6.2 Detection Approach

The detection approach described here is likewise to be situated and to operate within the DMM architecture described in Sect. 5.4.2, as is the case with the CPA NSP approach presented in Sect. 5.4.

The proposed model involves the CUSUM technique and ANFIS. The CUSUM measure for each attack is found and passed on to their respective ANFIS. The classification of the analysed traffic as an attack or normal is decided based on the output of ANFIS. Figure 5.9 gives the block diagram of the proposed model. As shown in Fig. 5.9, the real-time traffic data is collected and the characteristics needed for modelling the attack using CUSUM is extracted (pre-processing) which is then passed on to the corresponding ANFIS engine.

The characteristics differ for different types of attacks which is explained elaborately in this section. Considering the fact that it may not be possible to model the traffic in dynamic and complex systems like the Internet using simple parametric description, we chose the non-parametric version of the CUSUM algorithm. The CUSUM algorithm dynamically checks if the observed time series is statistically homogenous and, if not, it finds the point at which the change happens and responds accordingly. Consider $X(n)$, which denotes the value of the attack characteristic variable during the nth time interval and its corresponding weights $W(n)$, which is a derived value from $X(n)$.

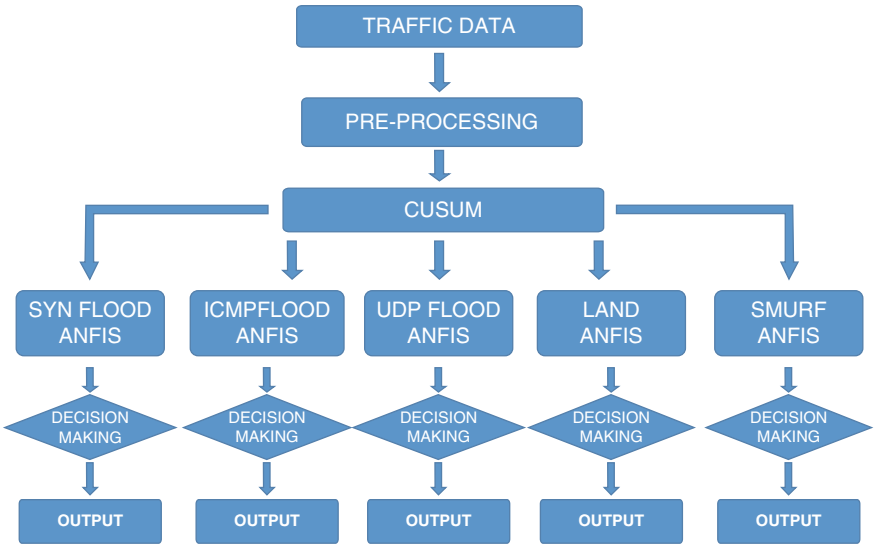

Fig. 5.9 Proposed model

We have modelled $X(n)$ for five different attacks where $XSYN(n)$, $XLAND(n)$, $XSMURF(n)$, $XUDP(n)$ and $XICMP(n)$ represent SYN Flood, Land, Smurf, UDP Flood and ICMP Flood attacks, respectively, and are collectively termed as attack characteristic variables. These variables are premeditated in such a way that it shows a sudden increase in value only during an attack, thus allowing the CUSUM algorithm to detect sudden variations in traffic more accurately and restricting false alarms to a large extent. Some of the existing DDoS detection techniques using CUSUM keep record of only the packet counts observed in a sample interval [49]. But in the proposed technique, CUSUM modelling is done in such a way that it reflects exactly the attack behaviour. The variable $X(n)$ for a specific attack category keeps track of a unique quantity corresponding to the same. Its value will be large when there is a high attack.

We model the above five different attacks as follows:

- SYN flood attack where we are taking into consideration the counts of RST, SYN and SYN/ACK packets:

$$X_{SYN}(n) = (N_{RST} + N_{SYN}) - N_{SYN/ACK} \tag{5.12}$$

- Land attack where $N_{[(SRC\_IP=DST\_IP)\&(SYNset)]}$ represents the number of incoming packets having the same source IP address and destination IP address with its SYN FLAG set:

$$X_{LAND}(n) = N_{[(SRC\_IP=DST\_IP)\&(SYNset)]} \tag{5.13}$$

- Smurf attack where $N_{(DEST\_ADDR=BADDR)}$ denotes the number of ICMP requests made to the broadcast address, exploits the vulnerability in the ICMP protocol:

$$X_{SMURF}(n) = N_{(DEST\_ADDR=BADDR)} \tag{5.14}$$

- UDP flooding attack where $N_{(DEST\_ADDR=HOST\_IP)}$ denotes the number of incoming UDP packets, $N_{(SRC\_ADDR=HOST\_IP)}$ denotes the number of outgoing UDP Packets and N_{ICMP_error} denotes the number of ICMP Destination Port Unreachable Error packets:

$$X_{UDP}(n) = (N_{(DEST\_ADDR=HOST\_IP)} - N_{(SRC\_ADDR=HOST\_IP)}) + N_{ICMP\_error} \tag{5.15}$$

- ICMP flood attack:

$$X_{ICMP}(n) = \text{Total payload size of the ICMP request packets} \tag{5.16}$$

(The above characteristic variable keeps a record of the summation of the sizes of incoming ICMP request packets since the time of observation.)

After modelling the attack characteristic variables, the values that are collected from the real-time traffic are passed on to their corresponding trained ANFIS engines which is explained in the next section.

5.6.3 ANFIS Engines

Fuzzy logic and neural networks help to embark upon issues such as vagueness and unknown variations in parameters more efficiently, and hence improve the robustness of the overall defence mechanism. Neuro-fuzzy techniques have been developed by a blend of the Artificial Neural Network (ANN) and the Fuzzy Inference System (FIS) and is termed 'ANFIS'. Using ANFIS with the CUSUM algorithm provides twofold advantage. First 3 it helps in removing the crisp threshold-based alarm-raising mechanism of CUSUM by a more comprehensible fuzzy logic–based mechanism, and second it fine-tunes the membership function parameters involved in FIS using the neural network–based learning technique. An adaptive network (Fig. 5.10) is a five-layer feed-forward network in which each node performs a particular function (*node function*) on incoming signals as well as having a set of fuzzy membership parameters pertaining to this node [36]. Following are the fuzzy rules base of Mamdani type which we have modelled for each type of attacks discussed above.

RULE 1: If (X(n) is HIGH) then attack is HIGH
RULE 2: If (X(n) is MEDIUM) then attack is MEDIUM
RULE 3: If (X(n) is LOW) then attack is LOW

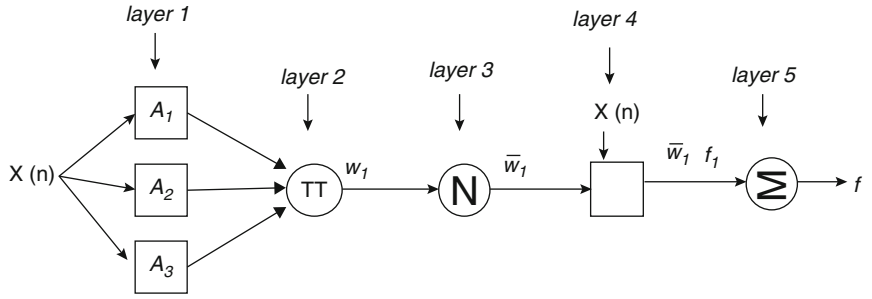

Fig. 5.10 Feed-forward networks

where $X(n)$ can either be $X_{SYN}(n)$, or $X_{LAND}(n)$, or $X_{SMURF}(n)$, or $X_{UDP}(n)$, or $X_{ICMP}(n)$ for SYN Flood, Land, Smurf, UDP Flood and ICMP Flood attacks, respectively.

Each node i in layer 1 is a square node with node function $O_i^1 = \mu_{A_i}(X(n))$, where A_i is the linguistic label (LOW, MEDIUM, HIGH) associated with this node function. In other words, O_i^1 is the membership function of A_i, and it specifies the degree to which the given *X(n)* satisfies the quantifier A_i . In our case all the membership functions are Gaussian,

$$O_i^1 = \mu_{A_i}(X(n)) = \exp\left(-\frac{(c_i - X(n))^2}{2\sigma_i^2}\right) \tag{5.17}$$

where $\{c_i, \sigma_i\}$ is the parameter set. As the values of these parameters change, the Gaussian function varies accordingly, thus exhibiting various forms of membership functions on the linguistic label A_i. Every node in layer 2 is a circle node labelled II which multiplies the incoming signal and sends the product out, but since in our model there is only one incoming signal, we get $w_i = \mu_{A_i}(X(n)), i = 1$. Each node output represents the firing strength of a rule. In layer 3, every node is a circle node labelled N. The *i th* node calculates the ratio of the *i th* ruleś firing strength to the sum of all ruleś firing strength:

$$\bar{w_i} = \frac{w_i}{w_1} \tag{5.18}$$

The output of this layer is called normalised firing strengths. Every node i in layer 2 is a square node with a node function

$$O_i^4 = \bar{w_i} f_i = \bar{w_i}(p_i X(n) + q_i) \tag{5.19}$$

where $\bar{w_i}$ is the output of layer 3, and $\{p_i, q_i\}$ is the parameter set known as consequent parameters. Variable f_i is not explained here. The single node in layer 5 is a circle node labelled $\sum$ that computes the overall output as the summation of all incoming signals, that is,

$$O_i^5 = overall\_output = \sum \bar{w_i} f_i = \frac{\sum_i w_i f_i}{\sum_i w_i} \tag{5.20}$$

All the membership functions used are Gaussian functions whose parameters are trained using the above-explained network with a hybrid learning algorithm. More specifically, in the forward pass of the hybrid learning algorithm, functional signal goes forward until layer 4 and the consequent parameters are identified by the least square estimate. In the backward pass, the error rates propagate backward and the premise parameters are updated by the gradient descent. Thus, it enables the FIS to adjust its parameters from the given training data. Also, it affords best trade-off between fuzzy and neural network by providing both smoothness and adaptability. In the proposed model, a separate adaptive network, as explained above, is formed for each of the five different attacks. Since the proposed system is intended to provide real-time detection, the training phase of ANFIS is carried out in offline mode over and over again to come out with a perfect training model, and during the deployment of this system the testing is done. Thus, the five ANFIS engines would not turn out to be a heavy computation because it is only the FIS that will be evaluated during the detection phase and the number of ANFIS engines can be further increased if required.

5.6.4 Decision-Making

After the evaluation of all the five ANFIS engines with CUSUM measures, the output (defuzzified value) of each ANFIS engine is collected and a decision is made based on those defuzzified values. Depending on the intensity of the attack, the final decision is given as LOW, MEDIUM or HIGH, which denotes the risk level of the network being monitored. The intensity of each attack can be known from the output of its corresponding ANFIS engine. The risk level 'LOW' means that the observed traffic is a NORMAL traffic. The risk level 'MEDIUM' warns the administrator of a possible attack which might occur. However, such an attack might not occur sometimes. Also, since the attack level is MEDIUM, the system may not be affected much. In this case, the administrator may not be sure if he can take any action because he would not know if it is going to be a serious attack. Therefore, the administrator can wait for some time and can take necessary action after he receives the next alarm. Thus, in these situations, it is solely the decision of the administrator to take necessary countermeasures. The risk level 'HIGH', alarms the administrator to immediately take necessary actions.

5.6.5 Experimental Results

All the experiments were carried out in the eight node SSE (Smart and Secure Environment) network laboratories distributed across different geographic locations and connected through an MPLS VPN cloud with each lab consisting of around nine workstations and one server. A detailed description of the testbed is provided

in Chap. 4. During a period of 10 days, 1,000 sample sets S_i, $1 \leq i \leq 1000$ with each set consisting of 16 different attack characteristic variables (i.e. features) $s_{ij}, 1 \leq i \leq 1000, 1 \leq j \leq 16$ are collected from a network traffic consisting of a mixture of attack and normal traffic. Each attack characteristic variable in a sample set is a count of the number of packets collected in a 5 s interval. These features are related to a particular characteristic of traffic packets such as packets with SYN flag set, RST flag set, SYN/ACK flags set, UDP packets, ICMP packets and Smurf packets.

From the collected sample sets, the attack characteristic variable of each attack is then used for training its corresponding ANFIS engine. The training process is done many times with different sample sets (S_i) in order to fine-tune the corresponding ANFIS engine, as mentioned in Sect. 5.6.3. The training process is carried out in offline mode, thus not adding any extra computation cost during the deployment in real-time traffic. Since the SSE network is a closed network, normally there will not be any attacks, and the training datasets are assumed to be clean.

Once the ANFIS is trained, many trials of testing were done in real-time traffic against the trained model, and comparisons are made with one of the well-known open source network Intrusion Detection System, (IDS) called Snort. For testing, 1,100 sample sets are collected comprising 200 sample sets each of SYN flood, LAND, SMURF, ICMP flood and UDP flood attack traffic along with 100 samples of normal traffic. The detection delay, DP, which is the time difference between the attack sample given as an input to the proposed scheme and its detection for each attack sample, is observed. Similarly DS, the detection delay using Snort for the same samples, is also observed. After observing the detection delays whenever the attack is detected, the maximum detection delay in each category is considered for DS as well as DP.

Table 5.9 compares the detection time of CANFIS and Snort by testing them using online samples. The specification of each sample regarding its average incoming packets per second and the type of attack traffic injected along is tabulated. A sampling interval of 0.5 s is fixed for analysing the samples.

As shown in Table 5.9, the detection time of the proposed technique is less than that of Snort. Table 5.10 compares the detection time of CANFIS and Snort by testing them offline. For this purpose, a well-known DDoS attack dataset is used [5]. The specification of each sample regarding its average packets per second, the attack duration and the type of attacks involved are tabulated above. The datasets are of varying attack levels – most of the datasets correspond to high-level DoS attacks. In most of the above-mentioned cases, the detection time of the proposed technique is less than that of the Snort. Thus, it is reasonable to infer from these facts that the proposed technique is faster when compared to Snort. In Table 5.11, the accuracy of the proposed technique is shown. The values of various evaluation parameters for each attack, namely True Positive (TP), True Negative (TN), False Positive (FP), False Negative (FN), True Positive Rate (TPR $=$ TP/(TP $+$ FN)), False Positive Rate (FPR $=$ FP/(FP $+$ TN)) and the Accuracy (ACC $=$ (TP $+$ TN)/(FP $+$ TP $+$ FN $+$ TN)) are shown.

Table 5.9 Comparison of detection time of CANFIS and Snort when tested online

S. no.	Sample name	Average packets/s	Attacks injected	Detection time CANFIS (s)	Detection time SNORT (s)
1	Sample_17	1,955	SYN flood, ICMP flood, Smurf	2.1	3.1
2	Sample_238	1,404	ICMP flood, SYN flood, Land, UDP flood	1.8	2.8
3	Sample_394	1,897	Land, Smurf	2.6	Not detected
4	Sample_431	1,540	UDP flood, ICMP flood, Land	1.8	3.1
5	Sample_479	1,478	SYN flood, UDP flood, Smurf	2.1	3.1
6	Sample_563	1,562	ICMP flood, SYN flood, UDP flood	2.5	2.8
7	Sample_704	1,457	Smurf, Land, ICMP flood	2.7	3.4
8	Sample_859	1,784	Smurf, Land, SYN flood	2.3	2.6
9	Sample_947	1,609	Smurf, SYN flood, Land, ICMP flood, UDP flood	2.8	3.8

Table 5.10 Comparison of detection time of CANFIS and Snort – using CAIDA dataset (offline)

S. no.	Sample name	Average packets/s	Attack duration (s)	Attacks injected	Detection time CANFIS (s)	Detection time SNORT (s)
1	ddos.2007 0804_134936	555	300	ICMP flood	4.6	5.1
2	ddostrace.2007 0804_135436	362	300	ICMP flood	5.1	5.5
3	ddostrace.2007 0804_143936	15,2619	300	ICMP flood, SYN flood	4.2	5.6
4	ddostrace.2007 0804_142936	153,180	300	ICMP flood, SYN flood	4.5	5.1
5	ddostrace.2007 0804_143436	147,740	300	ICMP flood, SYN flood, UDP flood	4.2	4.4
6	ddostrace.2007 0804_144436	165,226	300	ICMP flood, SYN flood	4.8	4.8
7	ddostrace.2007 0804_145436	172,566	55	ICMP flood, SYN flood	4.4	4.5
8	ddos.2007 0804_142436	164,824	300	ICMP flood, SYN flood, UDP flood	5.5	5.6

Table 5.11 Accuracy of CANFIS and Snort

Attack		FP	FN	TP	TN	TPR (%)	FPR (%)	Accuracy (%)
SYN flood	CANFIS	2	5	98	95	95.2	2.1	96.5
	SNORT	9	20	81	90	80.2	9.1	85.5
Land	CANFIS	1	3	97	99	97.0	1.0	98.0
	SNORT	10	25	75	90	75.0	10.0	82.5
Smurf	CANFIS	3	6	92	99	93.9	2.9	95.5
	SNORT	8	26	82	84	75.9	8.7	83.0
ICMP flood	CANFIS	4	1	98	97	99.0	3.9	97.5
	SNORT	11	17	92	80	84.4	12.1	86.0
UDP flood	CANFIS	2	1	99	98	99.0	2.0	98.5
	SNORT	7	23	80	90	77.7	7.2	85.0

For calculating the various evaluation parameters as listed in Table 5.11, a test sample containing 100 sample sets of attack traffic of each type and 100 sample sets of normal traffic is created. For example, for a SYN flood attack, 100 sample sets each containing traffic related to SYN flood and 100 sample sets of normal traffic are taken, and the various evaluation parameters were calculated. For all the attacks, the accuracy of CANFIS is quite better than Snort, and the false alarms also are also comparatively less when compared to it. It can be inferred from Tables 5.9 to 5.11 that our proposed scheme is computationally faster and has less false alarms when compared to Snort IDS.

5.7 Conclusion and Further Work

As has been discussed elsewhere, high-rate flooding attacks remain a potent threat to the delivery of Internet-based services. Moreover, the early and reliable detection of the onset of such an attack remains a challenging problem as does the formulation and implementation of an effective mitigation strategy. The preceding discussion has covered both the detection and mitigation elements of the solution strategy and includes leading-edge work on developing detection techniques that can reliably identify a high-rate flooding attack in real time or, at least, in near real time.

The initial discussion was on the extraction and ranking of a feature set that characterises Internet traffic in a way that distinguishes normal traffic from anomalous traffic. These features were then used to evaluate the performance of the selected set of machine-learning algorithms in detecting DDoS attacks. An important result from this work is that, of the various methods used, Fuzzy c-means clustering is a useful and very efficient way to detect a DDoS attack. The next technique discussed was based on Change Point Analysis. The specific focus was on exploiting a characteristic of high-rate flooding attacks, viz., a marked increase in the number of new users (i.e. IP addresses) attempting to interact with the target site. The discussion showed how the Change Point Analysis technique could be used in

combination with other techniques such as bit vectors and bloom filters to monitor the rate of arrival of new IP addresses and then use this as a basis to detect a high-rate flooding event. This was followed by discussion on an approach that exploited characteristic features of the TCP and UDP protocols which potentially distinguish a high-rate flooding attack from normal traffic. This approach also established that Naive Bayesian techniques could be used to detect a high-rate flooding attack in near real time. The final approach is to apply the CUSUM algorithm to track variations in the characteristic variable from the observed traffic that is specific to different kinds of attacks. Instead of using a threshold system to signify an attack, the discussion shows how a FIS can be employed which removes the abrupt separation between normality and abnormality and thereby reduces the number of false alarms.

The other key element of the discussion was on mitigation techniques. Central to this was the formulation of an architecture for a DDoS Mitigation Module (DMM). In particular, the discussion showed how the creation and maintenance of a white list could, in principle, meet one of the design goals for an effective DMM. Having established a proof of concept for the key elements of a DMM architecture, the next phase is to investigate the performance of both bit vector and bloom filter classification techniques under high-speed network flooding attacks in both IPv4 and IPv6 networks. Moreover, the analysis of detection algorithms under different and diverse flooding attacks needs to be investigated. We also expect to extend the basis for rejection of packets beyond the simple property of IPs Not Seen Previously. Further work will attempt to identify malicious traffic using other features to be used in tandem with NSP – features like subnet source addresses, IP distribution, traffic volume, traffic volume per IP address, packet inter-arrival time, etc. In addition, another line of investigation is the realisation of key detection algorithms in hardware so that they are genuinely capable of detecting the onset of high-rate flooding attacks in realtime.

References

1. KNIME. 2011. http://www.knime.org. Accessed 7 Feb 2011.
2. Snort: The open source network intrusion detection systems. http://www.snort.org/. Accessed 31 Aug 2011.
3. Waikato Applied Network Dynamic Research Group. http://wand.cs.waikato.ac.nz/. Accessed 1st Oct 2010.
4. DARPA Intrusion Detection Datasets, 1991. http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/data/. Accessed 31 Aug 2011.
5. UCSD Network Telescope – Code-Red Worms Dataset, 2001. The Cooperative Association for Internet Data Analysis http://www.caida.org/data/passive/codered_worms_dataset.xml. Accessed 7 Feb 2009.
6. Ahmed, E., A. Clark, and G. Mohay. 2008. A novel sliding window based change detection algorithm for asymmetric traffic. In *Proceedings of the IFIP International Conference on Network and Parallel Computing*, 168–175, Oct 2008.
7. Ahmed, E., A. Clark, and G. Mohay. 2009. Effective change detection in large repositories of unsolicited traffic. In *Proceedings of the Fourth International Conference on Internet Monitoring and Protection*, May 2009.

8. Ahmed, E., G. Mohay, A. Tickle, and S. Bhatia. 2010. Use of IP addresses for high rate flooding attack detection. In *Security and Privacy Silver Linings in the Cloud*, vol. 330, 124–135. Boston: Springer.
9. Almotairi, S., A. Clark, G. Mohay, and J. Zimmermann. 2008. Characterization of attackers' activities in honeypot traffic using principal component analysis. In *Proceedings of the IFIP International Conference on Network and Parallel Computing*, 147–154, Washington, DC, 2008. IEEE Computer Society.
10. Almotairi, S., A. Clark, G. Mohay, and J. Zimmermann. 2009. A technique for detecting new attacks in low-interaction honeypot traffic. In *Proceedings of the Fourth International Conference on Internet Monitoring and Protection*, 7–13, Washington, DC, 2009. IEEE Computer Society.
11. Argyraki, K. and D.R. Cheriton. 2005. Active internet traffic filtering: Real-time response to denial-of-service attacks. In *Proceedings of the Annual Conference on USENIX Annual Technical Conference*, ATEC '05, 10–10, Berkeley, 2005 . USENIX Association.
12. Argyraki, K. and D.R. Cheriton. 2009. Scalable network-layer defense against internet bandwidth-flooding attacks. *IEEE/ACM Transactions on Networking* 17: 1284–1297.
13. Baldi, M., E. Baralis, and F. Risso. 2004. Data mining techniques for effective flow-based analysis of multi-gigabit network traffic. In *Proceedings of IEEE 12th International Conference on Software, Telecommunications and Computer Networks*, 330–334, Split, Croatia, 2004.
14. Baldi, M., E. Baralis, and F. Risso. 2005. Data mining techniques for effective and scalable traffic analysis. In *Proceedings of the Ninth IFIP/IEEE International Symposium on Integrated Network Management*, 105–118, Nice, France, 2005.
15. Barford, P. and D. Plonka. 2001. Characteristics of network traffic flow anomalies. In *Proceedings of ACM SIGCOMM Internet Measurement Workshop*, 2001.
16. Bloom, B. 1970. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM* 13: 422–426.
17. Bocan, V. 2004. Developments in DoS research and mitigating technologiess. *Transactions on AUTOMATIC CONTROL and COMPUTER SCIENCE* 49(63): 1–6.
18. Bos, H. and K. Huang. 2005. Towards software-based signature detection for intrusion prevention on the network card. In *Proceedings of Eighth International Symposium on Recent Advances in Intrusion Detection*, Seattle, WA, 2005.
19. Bruijn, W.D., A. Slowinska, K. Reeuwijk, T. Hruby, L. Xu, and H. Bos. 2006. Safecard: A gigabit IPS on the network card. In *Proceedings of Ninth International Symposium on Recent Advances in Intrusion Detection*, Hamburg, 2006.
20. Carl, G., G. Kesidis, R.R. Brooks, and S. Rai. 2006. Denial-of-service attack - detection techniques. *IEEE Internet Computing* 10(1): 82–89.
21. Cheng, J., J. Yin, Y. Liu, Z. Cai, and M. Li. 2009. DDoS attack detection algorithm using IP address features. In *Frontiers in Algorithmics*, eds. X. Deng, J. Hopcroft, and J. Xue, vol. 5598, *Lecture notes in computer science*, 207–215. Berlin: Springer.
22. Clark, D.D. 1995. The design philosophy of the darpa internet protocols. *SIGCOMM Computer Communication Review* 25: 102–111.
23. Deri, L. 2007. High-speed dynamic packet filtering. *Journal of Network and Systems Management* 15(3): 401–415.
24. Dietterich, T.G. 2000. Ensemble methods in machine learning. In *Proceedings of the First International Workshop on Multiple Classifier Systems, MCS '00*, London, 1–15. Springer-Verlag.
25. Erskin, E., A. Arnold, M. Prerau, and L. Portnoy. 2002. A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data. In *Applications of Data Mining in Computer Security*, eds. D. Barbará and S. Jajodia, 77–102. Kluwer.
26. Fan, L., P. Cao, J. Almeida, and A.Z. Broder. 2000. Summary cache: A scalable wide-area web cache sharing protocol. *IEEE/ACM Transactions on Networking* 8: 281–293.
27. Farid, D.M., N. Harbi, and M.Z. Rahman. 2010. Combining naive bayes and decision tree for adaptive intrusion detection. *CoRR*, abs/1005.4496.

28. Feinstein, L., D. Schnackenberg, R. Balupari, and D. Kindred. 2003. Statistical approaches to ddos attack detection and response. In *Proceedings of the DARPA Information Survivability Conference and Exposition*, vol. 1, 303–314, 2003.
29. Ferguson, P. and D. Senie. 2000. Network ingress filtering: Defeating denial of service attacks which employ IP address spoofing, BCP 38, RFC 2827, May 2000.
30. Floyd, S. and V. Jacobson. 1993. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking* 1(4): 397–413.
31. Floyd, S. and V. Jacobson. 1995. Link-sharing and resource management models for packet networks. *IEEE/ACM Transactions on Networking* 3(4): 365–386.
32. Gavrilis, D. and E. Dermatas. 2005. Real-time detection of distributed denial-of-service attacks using rbf networks and statistical features. *Computer Networks* 48(2): 235 – 245.
33. Gil, T.M. and M. Poletto. 2001. Multops: A data-structure for bandwidth attack detection. In *Proceedings of the Tenth Conference on USENIX Security Symposium*, 3–3. USENIX Association.
34. Hettich, S. and S. D. Bay. 1999. The UCI KDD archive [http://kdd.ics.uci.edu]. University of California, Department of Information and Computer Science.
35. Hruby, T., K.V. Reeuwijk, and H. Bos. 2007. Ruler: high-speed packet matching and rewriting on npus. In *Proceedings of the Third ACM/IEEE Symposium on Architecture for Networking and Communications Systems, ANCS '07*, 1–10, New York, 2007. ACM.
36. Jang, J.S.R. 1993. ANFIS: adaptive-network-based fuzzy inference system. *IEEE Transactions on Systems, Man and Cybernetics* 23(3): 665–685.
37. Jin, S. and D. Yeung. 2004a. A covariance analysis model for DDOS attack detection. In *Proceedings of IEEE International Conference on Communications*, vol. 4, 1882–1886,20–24 June 2004.
38. Jin, S.Y. and D.S. Yeung. 2004b. DDoS detection based on feature space modeling. In *Proceedings of 2004 International Conference on Machine Learning and Cybernetics*, vol. 7, 4210–4215, 2004.
39. Jung, J., B. Krishnamurthy, and M. Rabinovich. 2002. Flash crowds and denial of service attacks: Characterization and implications for CDNs and web sites. In *Proceeding of 11th World Wide Web Conference*, 252–262, Honolulu, 2002.
40. Kang, J., Y. Zhang, and J.B. Jus. 2006. Detecting DDoS attacks based on multi-stream fused HMM in source-end network. In *Cryptology and Network Security*, vol. 4301, *Lecture Notes in Computer Science*, eds. D. Pointcheval, Y. Mu, and K. Chen, 342–353. Berlin: Springer.
41. Khor, K.C., C.T. Ting, and S.P. Amnuaisuk. 2009. From feature selection to building of bayesian classifiers: A network intrusion detection perspective. *American Journal of Applied Sciences* 6(11): 1949–1960.
42. Kim, D. and J. Park. 2003. *Network-based intrusion detection with support vector machines*, *Lecture Notes in Computer Science*, vol. 2662, 747–756. Springer, Berlin.
43. Kim, W.J. and B.G. Lee. 1998. Fred – fair random early detection algorithm for tcp over atm networks. *Electronic Letters* 34(2): 152–153.
44. Kline, J., S. Nam, P. Barford, D. Plonka, and A. Ron. 2008. Traffic anomaly detection at fine time scales with bayes nets. In *Proceedings of the Third International Conference on Internet Monitoring and Protection*, 37–46, Washington, DC 2008. IEEE Computer Society.
45. Le, Q., M. Zhanikeev, and Y. Tanaka. 2007. Methods of distinguishing flash crowds from spoofed dos attacks. In *Proceedings of the Third EuroNGI Conference on Next Generation Internet Networks*, 167–173, 2007.
46. Lee, H. and K. Park. 2001. On the effectiveness of probabilistic packet marking for ip traceback under denial of service attack. In *Proceedings of the IEEE INFOCOM*, 338–347, 2001.
47. Lee, K., J. Kim, K.H. Kwon, Y. Han, and S. Kim. 2008. DDoS attack detection method using cluster analysis. *Expert Systems with Applications* 34(3): 1659–1665.
48. Lemon, J. 2002. Resisting syn flood dos attacks with a syn cache. In *Proceedings of the BSD Conference, BSDC'02*, 10–10, Berkeley, 2002. USENIX Association.
49. Leu, F.Y. and Z.Y. Li. 2009. Detecting dos and ddos attacks by using an intrusion detection and remote prevention system. In *Proceedings of the Fifth International Conference on Information Assurance and Security*, vol. 2, 251–254.

50. Li, J., J. Mirkovic, M. Wang, P. Reiher, and L. Zhang. 2002. Save: Source address validity enforcement protocol. In *Proceedings of the IEEE INFOCOM*, 1557–1566, 2002.
51. Lin, D. and R. Morris. 1997. Dynamics of random early detection. *SIGCOMM Computer Communication Review* 27(4): 127–137
52. Liu, X., X. Yang, and Y. Lu. 2008. To filter or to authorize: Network-layer DoS defense against multimillion-node botnets. *SIGCOMM Computer Communication Review* 38(4): 195–206.
53. Mahajan, R., S.M. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker. 2002. Controlling high bandwidth aggregates in the network. *ACM Computer Communication Review* 32: 62–73.
54. Mahoney, M. and P. Chan. 2002. Learning nonstationary models of normal network traffic for detecting novel attacks. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '02*, 376–385, New York, 2002. ACM.
55. McPherson, D., C. Labovitz, M. Hollyman, J. Nazario, and G.R. Malan. 2008. Worldwide infrastructure security report. Technical report, Arbor Networks.
56. Miercom. 2008. Enterprise firewall: Lab test summary report. Technical report.
57. Mirkovic, J., G. Prier, and P.L. Reiher. 2002. Attacking DDoS at the source. In *Proceedings of the Tenth IEEE International Conference on Network Protocols, ICNP '02*, 312–321, Washington, DC, 2002. IEEE Computer Society.
58. Mirkovic, J. and P. Reiher. 2004. A taxonomy of DDoS attack and DDoS defense mechanisms. *SIGCOMM Computer Communication Review* 34:39–53.
59. Mirkovic, J. and P. Reiher. 2005. D_WARD: A source-end defense against flooding denial-of-service attacks. *IEEE Transactions on Dependable and Secure Computing* 2: 216–232.
60. Molsa, J. 2005. Mitigating denial of service attacks: a tutorial. *Journal of Computer Security* 13(6): 807–837.
61. Nazario, J. 2008. Political ddos: Estonia and beyond (invited talk). In *Proceedings of the Seventeenth USENIX Security Symposium*, San Josa, 2008.
62. Nguyen, H.V. and Y. Choi. 2009. Proactive detection of DDoS attacks utilizing K-NN classifier in an anti-DDos framework. *International Journal of Electrical and Electronics Engineering* 4(4): 247–252.
63. Papadopoulos, C., A.G. Tartakovsky, and A.S. Polunchenko. 2008. A hybrid approach to efficient detection of distributed denial-of-service attacks. Technical Report, June 2008.
64. Partow, A. 2008. General purpose hash function algorithms. http://www.partow.net/programming/hashfunctions/. Accessed 25 Feb 2011.
65. Paruchuri, V., A. Durresi, and S. Chellappan. 2008. TTL based packet marking for IP traceback. In *Proceedings of the IEEE Global Telecommunications Conference*, 2552–2556, Los Angels, 30 Nov–4 Dec 2008. IEEE.
66. Paxson, V., K. Asanovic, S. Dharmapurikar, J. Lockwood, R. Pang, R. Sommer, and N. Weaver. 2006. Rethinking hardware support for network analysis and intrusion prevention. In *Proceedings of the First USENIX Workshop on Hot Topics in Security*, 63–68.
67. Peng, T., C. Leckie, and K. Ramamohanarao. 2004. Proactively detecting distributed denial of service attacks using source IP address monitoring. In *Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communications: NETWORKING 2004*, 771–782, 2004.
68. Peng, T., C. Leckie, and K. Ramamohanarao. 2007. Information sharing for distributed intrusion detection systems. *Journal of Network and Computer Applications* 30(3): 877–899. 1231771.
69. Peng, T., C. Leckie, and K. Ramamohanarao. 2007. Survey of network-based defense mechanisms countering the DoS and DDoS problems. *ACM Computing Surveys* 39(1): 3. 1216373.
70. Peng, T., C. Leckie, and K. Ramamohanarao. 2008. System and process for detecting anomalous network traffic. United States Patent Application 20100138919. http://www.freepatentsonline.com/y2010/0138919.html. Accessed 31 Aug 2011.
71. Ripeanu, M. and A. Iamnitchi. 2001. Bloom filters – Short tutorial. Technical report, Dept. of Computer Science, University of Chicago.

72. Seo, J., C. Lee, T. Shon, K.H. Cho, and J. Moon. 2005. A new DDoS detection model using multiple SVMs and TRA. *Lecture notes in computer science*, vol. 3823, 976–985. Berlin: Springer.
73. Shanbhag, S. and T. Wolf. 2008. Evaluation of an online parallel anomaly detection system. In *Proceedings of the IEEE Global Telecommunications Conference*, 1–6, 2008.
74. Shanbhag, S. and T. Wolf. 2008. Massively parallel anomaly detection in online network measurement. In *Proceedings of Seventeenth International Conference on Computer Communications and Networks*, 1–6.
75. Shon, T., Y. Kim, C. Lee, and J. Moon. 2005. A machine learning framework for network anomaly detection using svm and ga. In *Proceedings of the Sixth Annual IEEE Information Assurance Workshop*, 176–183, 2005.
76. Simmons, K., J. Kinney, A. Owens, D.A. Kleier, K. Bloch, D. Argentar, A. Walsh, and G. Vaidyanathan. 2008. Practical outcomes of applying ensemble machine learning classifiers to high-throughput screening (hts) data analysis and screening. *Journal of Chemical Information and Modeling* 48(11): 2196–2206.
77. Sterne, D.F., K. Djahandari, B. Wilson, B. Babsonl, D. Schnackenberg, H. Holliday, and T. Reid. 2001. Autonomic response to distributed denial of service attacks. In *Proceedings of the Fourth International Symposium on Recent Advances in Intrusion Detection, RAID '00*, 134–149, London, 2001. Springer-Verlag.
78. Takada, H.H. and A. Anzaloni. 2006. Protecting servers against DDoS attacks with improved source IP address monitoring scheme. In *Proceedings of the Second Conference on Next Generation Internet Design and Engineering*, p. 6, 2006.
79. Tavallaee, M., E. Bagheri, W. Lu, and A.A. Ghorbani. 2009. A detailed analysis of the KDD CUP 99 data set. In *Proceedings of the Second IEEE International Conference on Computational Intelligence for Security and Defense Applications, CISDA'09*, 53–58, Piscataway, 2009. IEEE Press.
80. Vijayasarathy, R., B. Ravindran, and S.V. Raghavan. 2011. A systems approach to network modeling for DDoS detection using naive Bayesian classifier. In *Proceedings of the Third International Conference on Communication and Networks*, 2011.
81. Wang, H., D. Zhang, and K.G. Shin. 2002. Detecting SYN flooding attacks. In *Proceedings of the IEEE Infocom*, 1530–1539, 2002. IEEE.
82. Wang, W. and S. Gombault. 2008. Efficient detection of DDoS attacks with important attributes. In *Proceedings of the Third International Conference on Risks and Security of Internet and Systems*, 61–67, Oct 2008.
83. Wang, W., G.R. Guile, J.A. Shaqsi, A.A. Aulamie, R. Harrison, and W. Zhang. 2007. Machine learning ensemble methodology, 2007. http://www.uea.ac.uk/cmp/research/mma/kdd/projects/ensemble-methods/Machine+Learning+Ensemble+Methodology. Accessed 31 Aug 2011.
84. Weng, N. and T. Wolf. 2009. Analytic modeling of network processors for parallel workload mapping. *ACM Transactions in Embedded Computing Systems* 8(3): 1–29.
85. Xiang, Y. and W. Zhou. 2005. Mark-aided distributed filtering by using neural network for DDoS defense. In *Proceedings of the IEEE Global Telecommunications Conference*, vol. 3, 5.
86. Xie, Y. and S. Yu. 2006. A novel model for detecting application layer DDoS attacks. In *Proceedings of the First International Multi-Symposiums on Computer and Computational Sciences, IMSCCS '06*, 56–63, Washington, DC, 2006. IEEE Computer Society.
87. Xu, T., D. He, and Y. Luo. 2007. DDoS attack detection based on RLT features. In *Proceedings of the International Conference on Computational Intelligence and Security*, 697–701, China, 15–19 Dec 2007.
88. Xu, X., Y. Sun, and Z. Huang. 2007. Defending DDoS attacks using hidden Markov models and cooperative reinforcement learning. In *Intelligence and Security Informatics*, *Lecture notes in computer science*, vol. 4430, 196–207, 2007. Springer, Berlin.
89. Yan, J., S. Early, and R. Anderson. 2000. The xenoservice – A distributed defeat for distributed denial of service. In *Proceedings of the Information Survivability Workshop*, Oct 2000.
90. Yuan, J. and K. Mills. 2005. Monitoring the macroscopic effect of DDoS flooding attacks. *IEEE Transactions on Dependable and Secure Computing* 2: 324–335.

91. Zargar, G.R. and P. Kabiri. 2009. Identification of effective network features for probing attack detection. In *Proceedings of the First International Conference on Networked Digital Technologies*, 392–397, July 2009.
92. Zhou, Z., D. Xie, and W. Xiong. 2009. Novel distributed detection scheme against DDoS attack. *Journal of Networks* 4: 921–928.