will describe in detail how fuzzy logic can be utilized in DDoS flood attack intensity decision.

4.3 Attack intensity decision

Based on the basic theory and method of fuzzy mathematics, we propose an intelligent DDoS flood attack intensity decision system. DDoS flood attack intensity itself includes fuzziness, because the boundary between the light attack, moderate attack and severe attack is not well defined. So when judging the intensity of attack, one should take the intensity of background traffic into consideration. For example, a DDoS flood attack is considered as light attack if it causes slight decline of the network performance when the traffic load is high, but is considered as severe attack if it causes serious decline of the network performance when the network load is light [17]. In the proposed decision system, the DDoS flood attack intensity decision rules and operations are expressed by fuzzy sets, and then we feed these fuzzy decision rules and related information into knowledge repository.

The network elements take the dynamic process of actual attack into consideration, and then use fuzzy reasoning to determine the intensity of attack dynamically and intelligently. In this paper, the structure of fuzzy decision is two dimensional input and one-dimensional output. The two inputs are the Hurst parameter and its changing rate. The Hurst parameter reflects the influence of dynamic normal traffic on attack intensity and the changing rate reflects the influence of attack on normal traffic. The output is the intensity of the attack. As shown in Figure 3, the fuzzy decision process of the intensity of the attack consists of three parts: Hurst parameter and its changing rate fuzzification, fuzzy decision rules of attack intensity and fuzzy reasoning of attack intensity.

The description of each part of the fuzzy decision process is as follows:

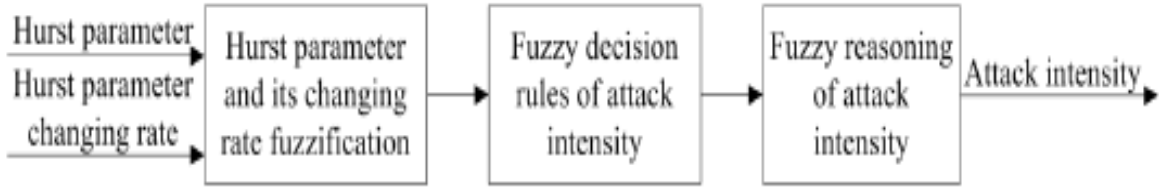

Figure 3: Fuzzy Decision Process

4.4 Hurst Parameter and its Changing Rate

Fuzzification- Fuzzification makes the real input parameters of Hurst parameter and its changing rate as fuzzy sets[15]. According to the change scope of Hurst parameter and its changing rate, we define the universe of discourse of the Hurst parameter as *UH*={0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1}; the universe of discourse of the changing rate as *UHC*={0.05, 0.10, 0.15, 0.20, 0.25, 0.30, 0.35, 0.40, 0.45, 0.50}. The fuzzy sets of *UH* and *UHC* are *H'*={*S*, *M*, *B*} and *HC'*={*S*, *M*, *B*}, where "*S*" stands for small, "*M*" the moderate, and "*B*" the big. The variable's membership degree function of each fuzzy language satisfies normality assumption

$$\mu(x)=\exp\left[-(x-v)^2/b^2\right]$$

where *v* and *b2* are the mean and variance of the membership degree function. Through the above equation, we can obtain fuzzy judgment model of every parameter as well as the membership degree assignments of every fuzzy subset.

Fuzzy decision rules of attack intensity- The decision rules take note of the relationship between input fuzzy sets and output fuzzy sets [18]. Define the fuzzy decision result of DDoS flood attack intensity as a variable *L*, and the fuzzy set of *L* as *L'*={*LA*, *MA*, *SA*}, where "*LA*", "*MA*" and "*SA*" represent light DDoS flood attack, moderate DDoS flood attack and severe DDoS flood attack, respectively. Considering the relationship between Hurst parameter, its changing rate and DDoS flood attack intensity, we can get the fuzzy decision rules displayed in Table 1.

Table 1: Fuzzy decision rules

HC'	*H'*		
	S	*M*	*B*
S	*MA*	*LA*	*LA*
M	*SA*	*MA*	*LA*
B	*SA*	*SA*	*MA*

Fuzzy reasoning of attack intensity- After fuzzifying the input parameters Hurst parameter and its changing rate, we can reason the intensity of attack according to decision rules presented in Table 1. For example, when the Hurst parameter is considered moderate, we infer there is a light DDoS flood attack if the changing rate of the Hurst parameter is considered small. In a similar way, there is a moderate DDoS flood attack if the changing rate of the Hurst parameter is moderate, and severe DDoS flood attack if the changing rate of the Hurst parameter is big.

Description-Distributed denial-of-service detection:
*start listener
*create a node and enter node name as 'server'
*create two or more nodes and enter its name as client1, client2...

The concept is that, we create a network containing a server and multiple clients. Client nodes will be sending data to server node. The server node will display the bandwidth (speed) at which the client nodes sending the data. The client nodes can send data in two modes: request mode and attack mode. While sending in request mode, the data will be sent

Licensed Under Creative Commons Attribution CC BY

at normal speed. While sending in attacking mode, the data will be sent in abrupt speed. That is, too much requests will be sent to the server which slows down the available network bandwidth of the server. After some time, click the prevent button in the server node. The client nodes which are sending data at bandwidth greater than threshold bandwidth are kicked out from the network. That is, the attack nodes are identified and are removed from the network. They no longer can communicate with the server. The data we are sending is a big matrix. The user needs to specify the number of rows and columns and a big matrix will be created with random 'double' data type numbers.

Network Setup- A network is setup with configurable number of nodes in the system. The nodes in the setup can communicate each other. We generate huge data in the form of matrices in different nodes, and the data are sent between the nodes.

Packet Sniffer- We capture the packets sent between the computers or through any network interface available in the computer. The sniffer works by calling WinPcap DLL, through JNI (java native interface).

Packets Data Parser- Here, we decode captured or stored (tcpdump files) packets for obtaining the packet data and its header fields like source ip, destination ip, source port, destination port, ttl, length, etc. Both incoming and outgoing data are captured so that data routed to and from a specific computer can be separated.

Attack Detection- Here, we implement the proposed method to detect the attack nodes. The nodes, which send data in abrupt speed are isolated in the network, by analyzing the network traffic between each node in the network.

Results and Analysis- The proposed method is tested with various network traffics, and a log containing the statistical data like time, kbps, etc. is generated.

5. Experimental Results

In this section, we conduct sets of experiments on data sets in terms of the classification error using the optimum training sets for testing. The classification performance is investigated by considering the performance with the lowest error rate and the corresponding value of k.

5.1 Evaluation of Attack Detection

The performance evaluation of proposed system is shown using accuracy rate and recall rate. Its measurement using F-measure with matrix confusion standard are shown in the Table 2.

Table 2: Confusion matrix for a binary classification problem in which the classes are not equally important

Actual class	*Predicted Class*	
	+	-
+	TP	FN
-	FP	TN

- The **True Positive (TP):** corresponds to the number of positive examples correctly predicted by the classification model.
- The **False Negative (FN):** corresponds to the number of positive examples wrongly predicted as negative by the classification model.
- The **False Positive (FP):** corresponds to the number of negative examples wrongly predicted as positive by the classification model.
- The **True Negative (TN):** corresponds to the number of negative examples correctly predicted by the classification model.

Accuracy means probability that the algorithms can correctly predict positive and negative examples. Sensitivity means probability that the algorithms can correctly predict positive examples. Specificity means probability that the algorithms can correctly predict negative examples.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

$$\text{Sensitivity} = \frac{TP}{TP + FN} \quad (2)$$

$$\text{Specificity} = \frac{TN}{TN + FP} \quad (3)$$

The purpose of Experimental results is to show synflood attacks in the high speed network. In the High speed network attacker can attack server system by sending more requests as a authorized host. Finally attacker denial the service of actual user.

The following are the Fuzzy PCF experimental results:

Table 3: Attacks on Victim IP

Source IP	*Victim IP*	*Attack Name*	*No.of Syns*
192.168.1.1	192.168.1.2	Syn-flood	512
192.168.1.4	192.168.1.2	Syn-flood	512
192.168.1.5	192.168.1.2	Syn-flood	512
192.168.1.10	192.168.1.2	Syn-flood	512
192.168.1.12	192.168.1.2	Syn-flood	512
192.168.1.14	192.168.1.2	Syn-flood	512
192.168.1.16	192.168.1.2	Syn-flood	512

The table shows the evaluation of the proposed method Fuzzy Partial Completion Filter, a prototype of DDoS attack system has been established, as an example, the SYN Flooding, which is the most well-known DDoS attacks, is employed in our method. We used this proposed approach for attacking detection, and the proposed method works as the packet filter at the victim side. For contrast, we also implement the 32 bits strict filtering algorithm of FPCF.

Licensed Under Creative Commons Attribution CC BY

Table 4: Performance Comparison based on Testing Criterion with Statistical, Partial Completion and Fuzzy Partial completion Filters

Testing Criterion	Statistical Filter		Partial Completion Filter		Fuzzy Partial Completion Filter	
	Correctly Classified Instances	Time to build Model (seconds)	Correctly Classified Instances	Time to build Model (seconds)	Correctly Classified Instances	Time to build Model (seconds)
Training set itself	51.28	0.15	91.81	0.74	76.55	0.01
Percentage split (50% train 50% test)	59.67	0.07	69.91	0.57	70.8	0.01
Percentage split (70% train 30% test)	58.09	0.08	74.26	0.44	75	0.01
Percentage split (80% train 20% test	56.04	0.08	67.03	0.42	74.73	0.02

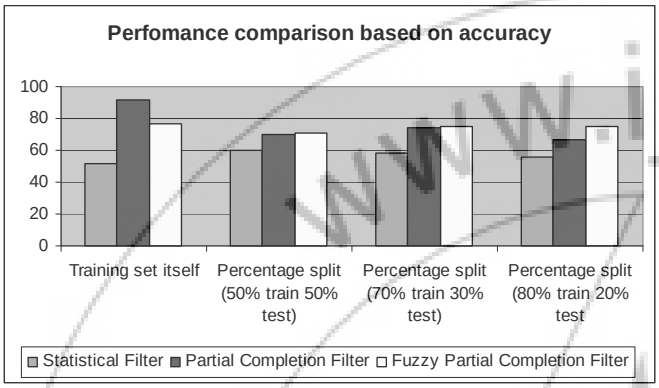

Figure 4: Comparing Accuracy among Statistical, Partial Completion and Fuzzy Partial completion Filters

As shown in Figure 4, the highest accuracy was observed. Despite the high accuracy rate of Fuzzy Partial Completion filter, the accuracy curve is unstable when the data is spilt into training and test, whereas Statistical and Partial Completion filter show stable accuracy for the same dataset. The accuracy rate of Statistical Filter is the lowest among the three algorithms.

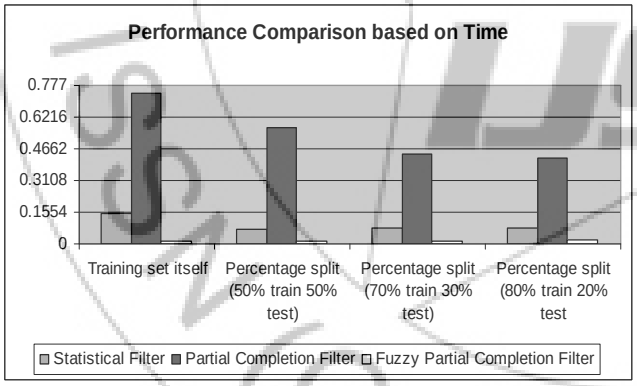

Figure 5: Comparing Learning Time among Statistical, Partial Completion and Fuzzy Partial completion Filters

Figure 5 illustrates the learning time comparison of the algorithms. The Fuzzy Partial completion Filter algorithm consumes far more learning time than the other algorithms. The learning time of Fuzzy Partial completion Filter drops drastically at percentage split of 50% and 70%. The learning time of Statistical drops at percentage split of 50%. The differences in learning time for Partial Completion filter for different percentage split was found to be not significant.

Table 5: Classification Result of Fuzzy Partial Completion Filter

Category	*Number of Testing Packets*	*Correct Classification*	*Incorrect Classification*
Normal	4813	4566	247
Attack	124562	123159	1403

The classification result of testing dataset using proposed algorithm only shows in Table 5. According to the statistics the system can correctly detect 94.87% for normal traffic and 98.87% for attack traffic. It incorrectly classified traffic in 5.13% for normal class and 1.13% for attack class. Out of total number of packets (Attack + Normal packets) 129375, 1650 packets are classified as others.

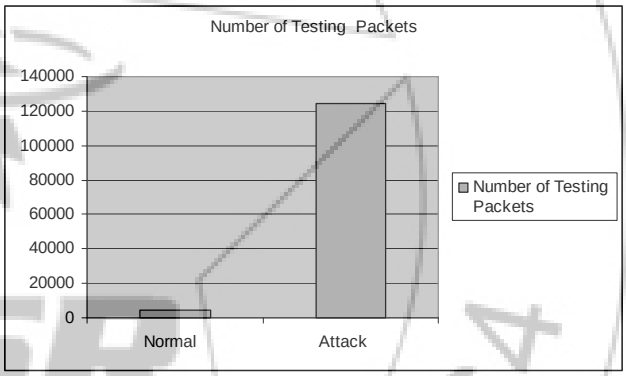

Figure 6: Number of Testing Packets

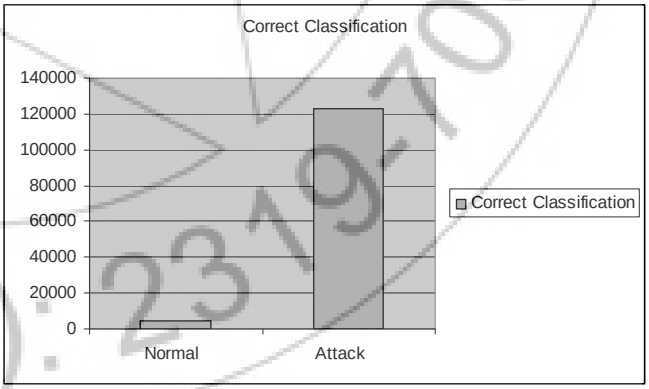

Figure 7: Correct Classification

Licensed Under Creative Commons Attribution CC BY

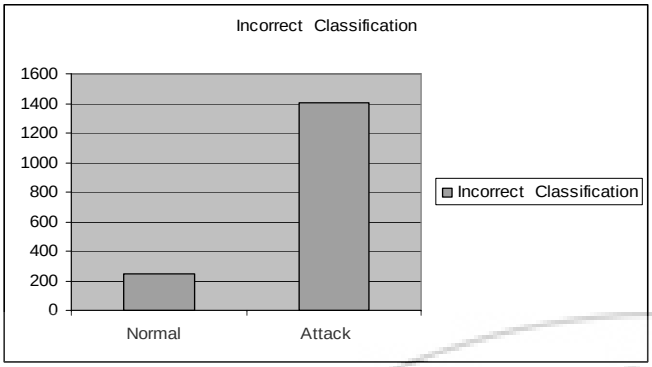

Figure 8: Incorrect Classification

5.2 Recall and Precision:

These are two widely used metrics employed in applications where successful detection of one of the classes is considered more significant than detection of the other classes. A formal definition of these metrics is given below. The recall and Precision values are already classified.

$$\text{Precision, } p = \frac{TP}{TP + FP} \quad (1)$$

$$\text{Recall, } r = \frac{TP}{TP + FN} \quad (2)$$

Table 6: Performance comparison based on Precision and Recall

Algorithm	*Precision*	*Recall*
Statistical Filter	68.2	69
Partial Completion Filter	75.6	76
Fuzzy Partial Completion Filter	96.3	98.2

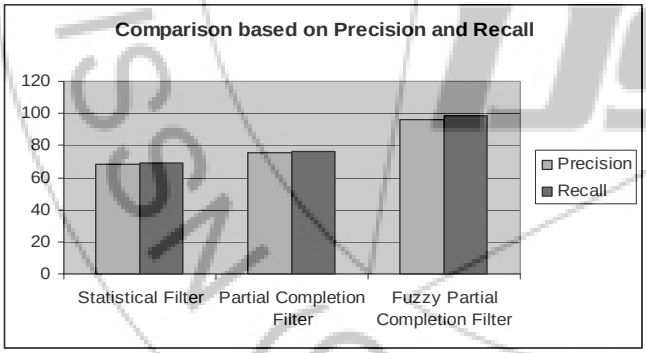

Figure 9: Comparison based on Precision and Recall

6. Conclusion

It appears to be widely perceived that detecting intrusions scalably within the network is a bad idea. Unfortunately, that causes security devices to choose between performance (which requires low memory) and completeness (which appears to require per-flow state). This paper is a gentle first step towards suggesting that this tradeoff may not be as Draconian as is commonly thought. While the general problem is still very hard (and indeed for attacks such as evasion attacks, we believe that aggregated solutions cannot work without causing unacceptably high false positives), our paper shows some progress for bandwidth- based and partial completion DoS attacks, and scan-based attacks including worms.

This paper explores this possibility in the specific context of DoS attacks and scan attacks by introducing fuzzy partial completion filter. While the existing approaches have not harped on this point of membership value, doing DoS detection in the network also finesses the need for traceback and/or manual intervention, and allows enterprise networks and ISPs to automatically filter out attacks before they enter (or leave) their networks. More fundamental than the specific techniques discussed in this paper is the general question of scalable behavior-based detection of attacks within the network.

This paper concentrates on considerable attention in the research and product literature, and solutions that scale are developed under java platform. As security functions become more prevalent in the edge first and then the core, it is natural to expect the same attention to be paid to scalable security solutions. More than just introducing the question and suggesting a specific mechanism for some problems, our paper shows that the issues of *behavioral aliasing* and *spoofing* are key questions that must be addressed in any scalable solution, even if the only response is to simply ignore the problem. These two provide a simple lens to view existing and future work in attack detection, and can perhaps suggest new solutions to an even broader class of attacks. The experimental result concludes that the proposed fuzzy based partial completion filter performs the detection of IDS very effectively comparing the existing approaches represented in this work.

References

[1] M. Roesch, Snort. [Online]. Available: http://www.snort.org
[2] P. Barford, J. Kline, D. Plonka, and A. Ron, "A signal analysis of network traffic anomalies," in *Proc. 2nd ACM SIGCOMM Internet Measuremen Workshop*, 2002, pp. 71–82.
[3] B. Krishnamurthy, S. Sen, Y. Zhang, and Y. Chen, "Sketch-based change detection: Methods, evaluation, and applications," in *Proc. 3rd ACM SIGCOMM Internet Measurement Conf.*, 2003, pp. 234–247.
[4] S. J. Staniford, "Containment of scanning worms in enterprise networks," *J. Computer Security*, 2004, to be published.
[5] ForeScout Technologies. [Online]. Available: http://www.forescout.com
[6] D. Moore, G. Voelker, and S. Savage, "Inferring Internet denial of service activity," in *Proc. 10th USENIX Security Symp.*, Aug. 2001, pp. 9–22.
[7] Mazu Publishing. [Online]. Available: http://www.mazu.com
[8] Arbor Networks. [Online]. Available: http://www.arbornetworks.com
[9] H.Wang, D. Zhang, and K. Shin, "Detecting SYN flooding attacks," in *Proc. IEEE INFOCOM*, 2002, pp. 1530–1539.
[10] V. Paxson, "Bro: A system for detecting network intruders in realtime," *Computer Networks*, vol. 31, no. 23–24, pp. 2435–2463, 1999.

Licensed Under Creative Commons Attribution CC BY

[11] K. Levchenko, R. Paturi, and G.Varghese, "On the difficulty of scalably detecting network attacks," in Proc. 11th ACM Conf. Computer and Communications Security, 2004, pp. 12–20.
[12] R. Keyes, "The Naptha DoS vulnerabilities," [Online]. Available:http://www.cert.org/advisories/CA-2000-21.html
[13] N. Weaver, V. Paxson, S. Staniford, and R. Cunningham, "A taxonomy of computer worms," in Proc. ACM Workshop of Rapid Malcode (WORM), 2003, pp. 11–18.
[14] S. Staniford, V. Paxson, and N. Weaver, "How to 0wn the Internet in your spare time," in Proc. 11th USENIX Security Symp., Aug. 2002, pp. 149–167.
[15] MyDoom.B Virus. [Online]. Available: http://www.us-cert.gov/cas/techalerts/TA04-028A.html
[16] CERT Advisory CA-2001-19, "Code Red" Worm Exploiting Buffer Overflow In IIS Indexing Service DLL, [Online]. Available:http://www.cert.org/advisories/CA-2001-19.html
[17] CERT Advisory CA-2001-26 NimdaWorm, [Online].Available: http:// www.cert.org/advisories/CA-2001-26.html
[18] CERT Advisory CA-1998-01 Smurf IP Denial-of-Service Attacks,[Online]. Available: http://www.cert.org/advisories/CA-1998-01.html
[19] V. Paxson, "An analysis of using reflectors for distributed denial-ofservice attacks," Comput. Commun. Rev., vol. 31, no. 3, Jul. 2001.
[20] T. M. Gill and M. Poletto, "MULTOPS: A data-structure for bandwidth attack detection," in Proc. 10th USENIX Security Symp., 2001, pp. 23–38.

Author Profile

Abilesh received the M.Sc. Degree in Information Technology from Shri Nehru Maha Vidyalaya College of Arts & Science, Affiliated to Bharathiar University, Coimbatore in 2012 and pursuing M.Phil. degree in Computer Science from Sri Ramalinga Sowdambigai College of Science & Commerce (Affiliated to Bharathiar University), Coimbatore.

Licensed Under Creative Commons Attribution CC BY