

IMPLEMENTATION OF TBAQM ALGORITHM TO ALLEVIATE FLOODING ATTACKS

¹Amogh Venkatanarayan, ²Mainul Hasan, ³Inder Mohan, ⁴Ninni Singh, ⁵Gunjan Chhabra^{1,2,3,4,5}School of Computer Science Engineering, College of Engineering Studies,

University of Petroleum and Energy Studies, Dehradun, Uttarakhand, India

¹amogh.venkatanarayan@gmail.com, ²mainulhasan@live.com, ³imgupta1996@gmail.com,⁴ninnisingh1991@gmail.com, ⁵gchhabra@ddn.upes.ac.in

Abstract: Servers and cloud services today have an ever-growing load and demand, due to digitization of all sectors. Though the vulnerabilities have remained similar, the threat levels and impact have grown. The CIA triad of security is of utmost importance and is continuously monitored. Denial of Service attacks, distributed or otherwise, are the most destructive attacks. These attacks do not necessarily need a vulnerability to succeed. It works on the mathematics and economics of scale. Limited memory, limited processing power and limited bandwidth are a few factors that are usually attacked. A new algorithm Threshold Based Active Queue Management (TBAQM) was recently published, which is tested under various conditions and restriction to check its efficiency. The algorithm was tested on simulation to verify and quantify the results. Memory, Latency and network configuration were changed to see the impact on the functioning of the algorithm. Conclusively, this algorithm is able to work efficiently and effectively even under duress.

Keywords: Distributed Denial of Service, Queue Management, flood attacks.

1. Introduction

Denial of Service is a direct attack on the availability of a system. The system is rendered useless by occupying its resources in futile tasks. This can be performed by overwhelming the bandwidth and blocking it, or engaging it in an intensive task that occupies the processor completely. Overwhelming of bandwidth is achieved by flooding, i.e. sending a huge number of redundant packets to be considered as genuinely different packets. [1]

Flooding done on the transport layer could be perceived to be more harmful as it is the Transport layer that is responsible for establishing the communication channel between two systems. Multiple solutions involving modification of the Active Queue Management (AQM) have been proposed. Network queuing algorithms have been proposed to utilize the

bandwidth better. But these solutions pertain to only one factor or have other issues such as high processing requirement or high memory. [2][3][4]

This paper implements the Threshold Based Active Queue Management algorithm to check for its efficiency and adaptability in real world, by testing it against various conditions such as low memory, low processing and variant number of attackers. [5]

2. Literature Review

Denial of Service (DoS) is an attack possible on multiple layers of the network. TCP/IP and UDP being the most integral part while establishing the connection is the most attacked. There are two kinds of DoS attacks that could take place on the transport layer, low-rate DoS which exploits the minimum retransmission output (RTO) of TCP and flood DoS. In the flood DoS, the server or the system is overwhelmed with a huge influx of packets. [6]

Another suggested remedy to distributed DoS is the internet firewall, i.e., a firewall placed on gateway of every network, connected to a common server to collect data of all outgoing traffic, so as to ascertain whether an attack is being perpetrated. [7]

Active Queue Management (AQM), is a common method to address requests. FAVOR suggests that certain connections which have already established and are in the active queue be given temporary priority [3], which in case of small Time-To-Live could cause congestion and denial. An approach involving captcha based verification on application layer and MAC filtration and cryptography based authentication [8] is proposed, but this method requires higher memory even during normal conditions. An enhanced AQM using a smaller buffer is suggested.

Threshold Based Active Queue Management (TBAQM) is another such AQM algorithm that builds upon the shortcomings of the other algorithms, such as memory, resource utilization etc. TBAQM suggests that a threshold be placed to minimize the consumption of resource during peacetime, and to use the resource in a highly optimized manner during attack. [5]

3. Methodology

Since, the algorithm was based on the 3-Way handshake protocol, the first objective was to implement the protocol for multiple clients and single server. The attacker also attacks the protocol and needs to be identified, hence needed to be exactly similar to all the clients.

For the sake of this study, bandwidth and delay were taken to be constraints in the network. These two constraints could lead to variation in the rate of congestion. Since the algorithm doesn't deal with the network directly, these constraints play an important role to determine the impact on the legitimate user before and after congestion.

On the sever end, the memory is limited and so is the processing power, it is hence imperative that this algorithm maintain the server working in stress conditions. Hence, memory constraints and processing constraints were placed to test the algorithm. The flooding that was performed was a SYN flood. SYN packets are a part of 3 Way handshake protocol which is illustrated below.

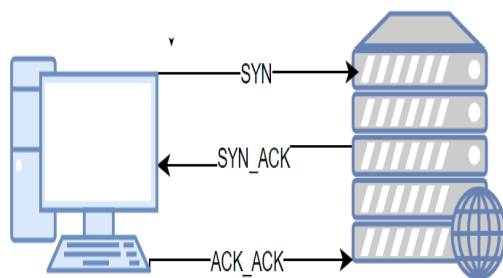


Figure 1. Regular 3 Way Handshake Protocol

The 3 Way handshake protocol can be misused in the below manner by an attacker.

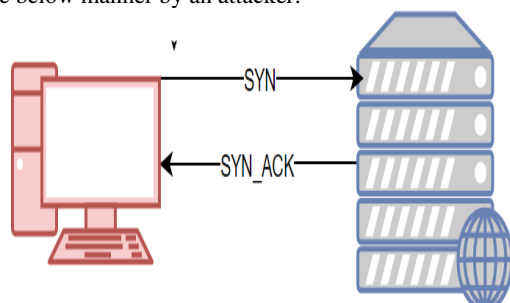


Figure 2. Attacker's 3 Way Handshake Protocol

The server allots some memory for the requested connection after receiving the SYN packet. In this attack the attacker is directly trying to consume memory resources by keeping the allotted space allotted unused and then requesting a new connection.

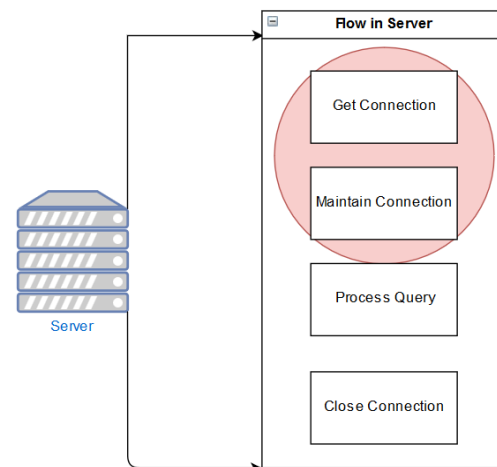


Figure 3. Working of Server

A server performs multiple functions, such as receiving packet, establishing connection, processing application queries and terminating connection. The TBAQM algorithm only focuses on the establishing the connection and maintaining the connection.

A. AQM modification

When a server receives packet, it has to ascertain the kind of the packet it has received. For establishing the connection, a SYN packet is expected. Other kinds of packets could be, FIN, ACK, ACK-ACK. Also, query packets for the Application layer could be present. The server only has to look at the flags set in the packet to ascertain the kind of packet after which the action on the packet is determined.

For a SYN packet, the server has to respond with the ACK packet and allot some memory for the processing after the connection is established. Since, a server receives multiple connection requests, it maintains a queue structure to keep track of the connections and its processing. Since, queues are FIFO in nature, and expecting no bandwidth problems, the server generally receives the packets in a FIFO manner. In SYN flood, this queue gets filled up by unused and wasteful packets and connections. The utilization of the resources is very low, yet the server is occupied. TBAQM is an algorithm that works on the queue management and subsequent DoS detection. This algorithm can also be used for DDoS with slight modifications. The algorithm not only alerts the Network Operations Center (NOC) about an ongoing DoS but also takes active preliminary measures to alleviate the problem without human intervention.

Both single source DoS and DDoS have been implemented to check the capability of this algorithm so as to prove its efficiency.

Algorithm**Algorithm** Threshold based AQM

```

1: procedure tBAQM Procedure
2: if  $tBAQ.fillCap$  greater than  $tBAQ.Cap/0.8$  then
3:    $timeout \leftarrow timeout/3$ 
4:   for each packet  $P \in tBAQ$  do
5:     if  $b \leftarrow insertTemp(P, tempArr)$  then -> Boolean
       return
6:   continue
7:   else
8:     break
9:   end if
10:  end for
11:   $packet.Common \leftarrow SearchPack(tempArr)$ 
12:  end if
13:   $IP \leftarrow grabIP(packet.Common)$  -> to get IP
14:   $blacklist(IP) \rightarrow$  temporary blacklisting
15: end procedure

```

S

b Check for completion of data transfer from queue to array

IP IP address to be blacklisted

P Packet in TBAQ

packet.Common Common Packet

tBAQ Active Queue

tBAQ.Cap Total capacity of Queue

tBAQ.fillCap Capacity filled in the queue

tempArr Temporary data structure to search

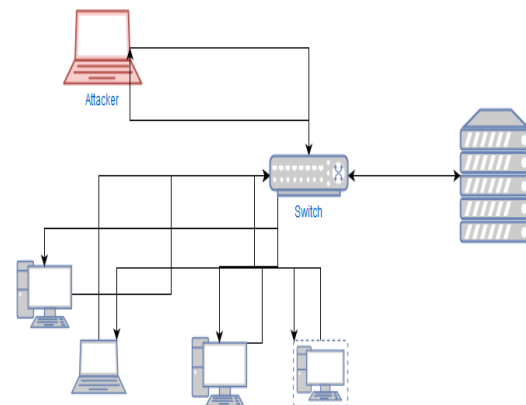
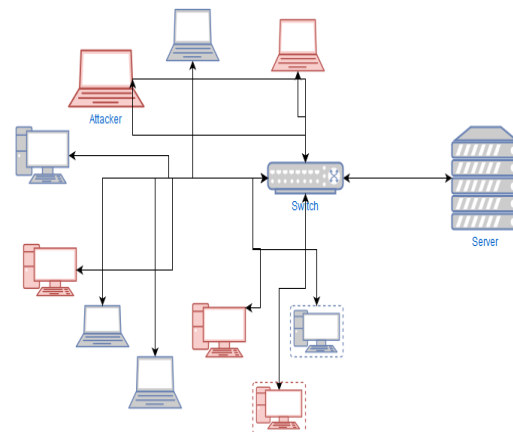
Timeout Time after which connection will be

The simulation of the algorithm was done on the OMNet++ simulator, which allowed for doing event by event simulation. The modularity provided ease in simulating only the network and transport layer, instead of the full stack.

OMNet++ allows having custom messages and custom gates to handle the messages. Each message handled by all the network nodes are specified according the algorithm. The attacker node(s) needed a different algorithm for its approach towards the attack. The time between the successive packets it throws needed to be set and modified for each run. The normal user had a rather simple implementation of the 3-Way handshake protocol. The Switch in the network was responsible for forwarding the packets to and from the server and clients. The bandwidth congestion occurred at the switch.

B. Network Configuration

Two main network configurations were designed to study the effects intricately. First is where there is only one rogue system in the network, which tries to inundate the server with malicious requests. Second, there are equal number of attackers as clients in the network. This situation can help to study the efficiency when there is equal probability of a system being a rogue, and the consequent performance of the algorithm.

**Figure 4.** Single attacker Network**Figure 5.** 5C-5A Network**C. Flowchart**

This flowchart shows the exact flow of events according to the algorithm. It is the same flow that was programmed in OMNet++ to check for efficiency.

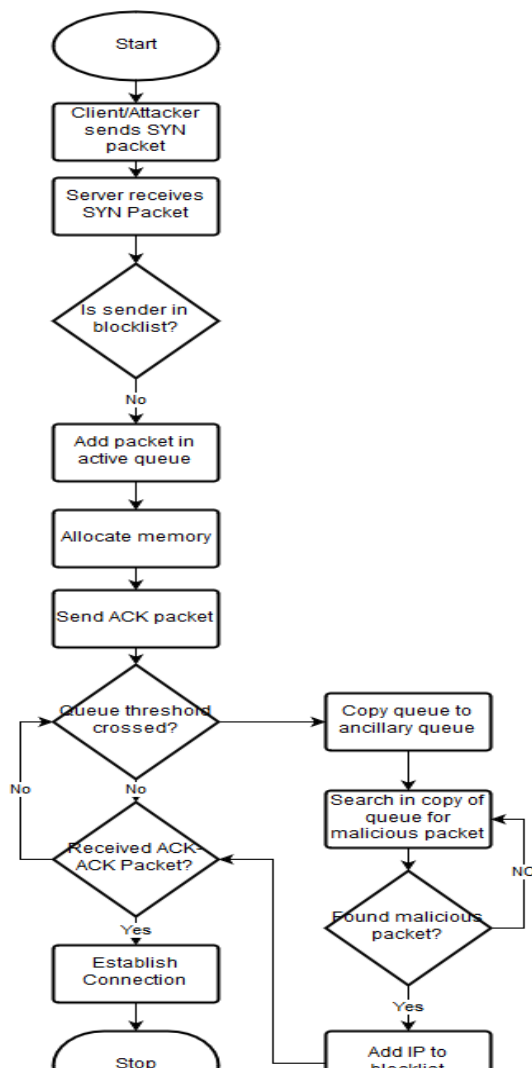


Figure 6. Flow of control for TBAQM

4. Results

The above proposed mechanism was implemented on OMNET++ simulator, as shown in Fig. 6 & 7. OMNET++ provides the framework to create modular and discrete events in order to study them in detail.

A basic template of the event and network model to be used was made. The basic template consisted of fixed number of clients and one attacker in the model. The attacker was programmed to create a flood attack on the server. All the clients were programmed to function according to the 3-Way handshake protocol.

The server allocated resources and added all packets it received into the active queue.

The results of the simulation suggest that the algorithm is able to handle the traffic when there is a surge without any breakdown of the server. The bandwidth plays a very crucial role, it directly affects the rate of achieving the congestion.

As mentioned in the methodology, the parameters and constraints placed, were number of systems, latency in the connection and server memory.

4.1 Number of Systems

This initial study tried to check the variation in the performance of algorithm when the number of systems, legitimate and rogue, are connected to the server. This was done to check, how the server deals with the variant rate of influx after the blocking of the rogue senders.

The first graph shows the working of the algorithm without implementing the alleviating part of the algorithm, i.e. blocking the attacker. The second graph, after implementing alleviating, shows how DoS, i.e. single attacker doesn't cross the threshold once blocked, but the one with 5 clients doesn't show such trend. This is because two attackers were programmed to be activated at the start of the simulation, whereas the remaining three, got activated at only 500ms after the start.

The load on the server memory is high when the queue gets filled to the capacity, but since most of it is unutilized the server is able to handle the duress of searching for the sender.

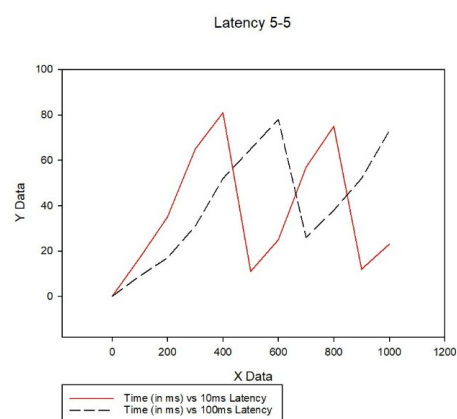


Figure 7. Latency in 5C-5A

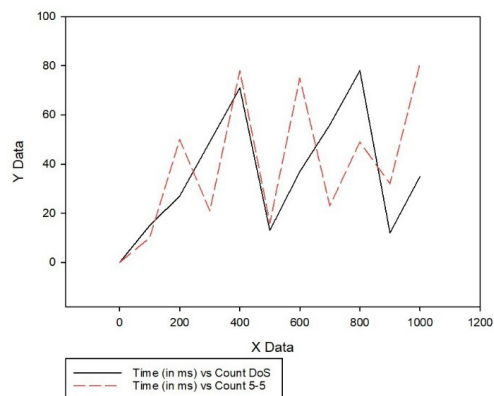


Figure 8. Without alleviation

4.2 Latency

Latency refers to the speed of the connection and the time it would take a packet to travel from one node to another. OMNet++ allows for modification of latency for each node, which was used to test the efficiency of algorithm, in case of successive threshold breaches.

If the latency is high it can cause congestion in the network, but low latency can cause congestion in the system. The packets arrive faster than what the system can be handled. Successive threshold breaches could be difficult to resolve if the earlier conflict is still being resolved and a new breach has occurred.

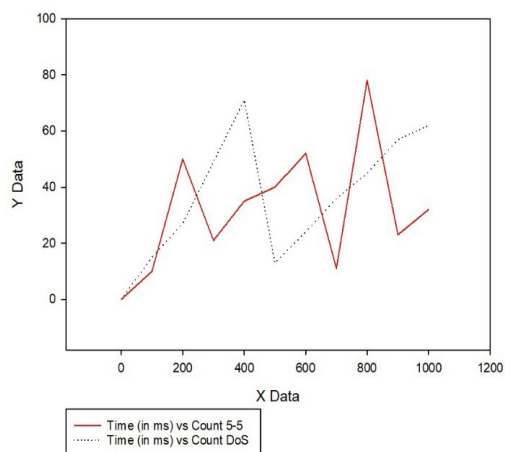


Figure 9. With Alleviation

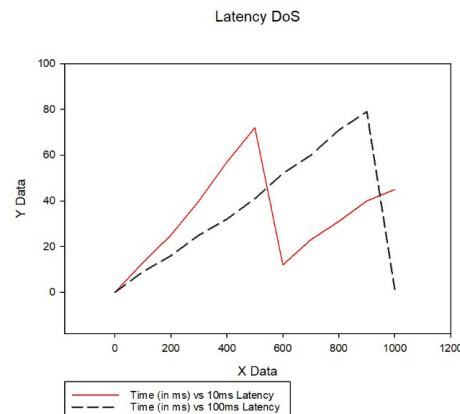


Figure 10. Latency in Single Attacker

4.3 Threshold

This AQM algorithm, is based on threshold which can be different according to differing business requirements. Hence, does change in threshold change the efficiency of the algorithm and the load on the server.

Higher the threshold, lesser the memory available to run the search feature. Hence, it was imperative that the algorithm be checked with varying thresholds to check how it holds up in high-stress conditions. The number of attackers has hugely impacted the load that is faced by the server.

5. Conclusions

Threshold Based Active Queue Management, is efficient, not just on the paper but also can be proven through simulation. This paper has done a comprehensive study on the algorithm's working, and the results show the same thing. Irrespective of changes in the configuration of the network, latency, bandwidth or memory the algorithm works smoothly in all conditions.

During the simulation in OMNet++, it was apparent that this algorithm would need its own implementation depending on the usage of the server. This study was majorly based only on the 3-Way handshake protocol, and conclusively it can be held that this algorithm is efficient and effective. This algorithm is prudent in solving the problem of flooding attacks of all natures.

6. Acknowledgement

This work was supported by the University of Petroleum and Energy Studies. We would like to specially acknowledge the support and guidance of Mr. Gunjan Chhabra, Asst. Professor, UPES. We would

also like to thank Ms. Ninni Singh for her guidance and support throughout the project.

References

- [1] L. C. Giralto, C. Conde, I. M. de Diago and E. Cabello, "Detecting denial of service by modelling web-server behaviour," *Computer and Electrical Engineering*, pp. 1-11, 2012.
- [2] H. Bedi, S. Roy and S. Shiva, "Mitigating congestion based DoS attacks with an enhanced AQM technique," *Computer Communications*, vol. 56, pp. 60-73, 2014.
- [3] P. Anelli, R. Diana and E. Lochin, "FavorQueue: A parameterless active queue management to improve TCP traffic performance," *Computer Networks*, vol. 60, pp. 171-186, 2014.
- [4] Maurizio, C. A. Grazia, M. Klapez and N. Patriciello, "QRM: A queue rate management for fairness and TCP flooding protection in mission critical networks," *Computer Networks*, vol. 93, pp. 54-65, 2015.
- [5] A. Venkatanarayan, I. Mohan, M. Hasan, N. Singh and G. Chhabra, "Threshold Based Active Queue Management (TBAQM) for Alleviating DoS/Flooding Attacks," *Journal of Engineering and Applied Sciences*, vol. 12, no. 11, pp. 3002-3007, 2017.
- [6] L. Xiao-Ming, C. Gong, L. Qi and Z. Miao, "A comparative study of flood DoS and low-rate DoS attacks," *The Journal of China Universities of Posts and Telecommunications*, vol. 19, pp. 116-121, 2012.
- [7] R. K. C. Chang, "Defending against Flooding-Based Distributed Denial-of-Service Attacks - A tutorial," *IEEE Communications magazine*, pp. 42-51, 2002.
- [8] P. A, S. M, B. S. S. T and B. N, "Detection and Mitigation of Denial of Service Attacks using Stratified Architecture," *Procedia Computer Science*, vol. 87, pp. 275-280, 2016.
- [9] S. Jamali and G. Shaker, "PSO-SFDD: Defense against SYN flooding DoS attacks by employing PSO algorithm," *Computers and Mathematics with Applications*, vol. 63, no. 1, pp. 214-221, 2012.
- [10] H. Safa, M. Chouman, H. Artail and m. Karam, "A collaborative defense mechanism against SYN flooding attacks in IP networks," *Journal of Network and Computer Applications*, vol. 31, no. 4, pp. 509-534, 2008.
- [11] S. T. Zargar, J. B. D. Joshi and D. Tipper, "A Survey of Defense Mechanisms Against Distributed Denial of Service Flooding Attacks," *IEEE Communications Surveys & Tutorials*, pp. 2046-2069, November 2013.
- [12] Lau, S. H. Rubin, M. H. Smith and L. Trajkovic, "Distributed Denial of Service Attacks," *IEEE International Conference on Systems, Man & Cybernetics - "Cybernetics Evolving to Systems, Humans, Organizations, and their Complex Interactions"*, pp. 2275 - 2280, 2000.
- [13] S. Noh, G. Jung, K. Choi and C. Lee, "Compiling network traffic into rules using soft computing," *Applied Soft Computing*, vol. 8, no. 3, pp. 1200-1211, 2008.
- [14] L. LI and S.-B. SHEN, "Packet track and traceback mechanism against denial of service attacks," *The Journal of China Universities of Posts and Telecommunications*, vol. 15, no. 3, pp. 51-58, 2008.
- [15] P. Fradet and S. Hong Tuan Ha, "Aspects of availability: Enforcing timed properties to prevent denial of service," *Science of Computer Computing*, vol. 75, no. 7, pp. 516-542, 2010.
- [16] L. Xiao-Ming, C. Gong, L. Qi and Z. Miao, "A comparative study on flood DoS and low-rate DoS attacks," *Science Direct*, pp. 116-121, 19 June 2012.
- [17] W. Eddy, "TCP SYN Flooding and Common Mitigations," *RFC*, August 2007
- [18] S. Bhattacharya, C. Diot, J. Jetcheva and T. N, "Pop-level and access level traffic dynamics," *1st ACM SIGCOMM Workshop*, pp. 39-53, 2001.
- [19] Rajesh, M., and J. M. Gnanasekar. "Congestion Control Using AODV Protocol Scheme For Wireless AD-HOC Network." *Advances in Computer Science and Engineering* 16.1/2 (2016): 19.
- [20] S.V.Manikanthan and K.Baskaran "Low Cost VLSI Design Implementation of Sorting Network for ACSFD in Wireless Sensor Network", *CiiT International Journal of Programmable Device Circuits and Systems*, Print: ISSN 0974 – 973X & Online: ISSN 0974 – 9624, Issue :November 2011, PDCS112011008.
- [21] T. Padmapriya, V.Saminadan, "Performance Improvement in long term Evolution-advanced network using multiple input multiple output technique", *Journal of Advanced Research in Dynamical and Control Systems*, Vol. 9, Sp-6, pp: 990-1010, 2017.

