An Enhancement of Advanced Random Time Queue Blocking using Active Queue Management for Mitigating Low-Rate DoS Attacks against Application Servers

Kavitha Jayabal

Sri Krish Arts and Science College, Coimbatore, Tamil Nadu, India

Abstract
Among different types of Denial-of Services (DoS), the most essential attack is Low-Rate traffic Denial-of-Service (LRDoS) attacks. This type of attacks has the ability to deny the network services by facilitating the vulnerabilities in applications. Various defense mechanisms have been developed against the LRDoS attacks. Among different defense techniques, the Advanced Random Time Queue Blocking with Traffic Prediction (ARTQB-TP) mechanism achieves better performance than the other techniques. However, this approach was not analyzed while the server is in an overflow condition. Hence, in this research paper, the congestion handling mechanism such as Active Queue Management (AQM) scheme is proposed. In this proposed mechanism, Deterministic Fair Sharing (DFS) scheme is incorporated with ARTQB-TP mechanism. The DFS is performed based on the weighted fair share (wfs) for identifying the congestion of the network by dynamically adjusts the router buffer utilization. The flow states are stored by using an efficient data structure like B+tree. Then, the malicious flows are differentiated from legitimate flows by using comprehensive repository and cache. Finally, the experimental results demonstrate that the performance of ARTQB-TP-AQM compared with the other mechanisms.

Keywords:
Denial-of-Service, LRDoS attack, ARTQB-TP, Congestion control, Deterministic Fair Sharing (DFS), Active Queue Management.

1. Introduction

An application server is defined as the software model which provides both facilities for constructing the web applications and a server environment for operating them. The application server model contains the comprehensive service layer model and acts as the group of components which are accessible to the software developer via an Application Programming Interface (API) defined by the platform itself. For wed applications, these elements are performed in the similar running environment as web servers whereas for java applications, the server acts as the extended virtual machine. Mostly the application servers are used for offering the middleware services, security and maintenance along with the data access and perseverance [1].

A Denial-of-Service (DoS) attack in networking is defined as an event which creates the machine or network resource inaccessible to its deliberate users for provisionally or indefinitely interrupting or delaying the services of the host linked to the internet. This can be categorized into crash services and flood services. Most of the attacks include foraging the IP sender addresses [2]. Hence the location of the adversary cannot simply detect. If this attack is achieved on the extremely high scale then the entire geographical areas of internet connectivity may be compromised without the attacker's awareness or target by means of imperfectly configured or poor network communications equipment. The prevention of DoS attacks is most essential to avoid network performance or degradation of internet connectivity.

Different DoS attacks detection and prevention techniques have been proposed. However, in network services, the major issue is Low-Rate Denial-of-Service (LRDoS) attacks. Among different type of DoS attacks, LRDoS attack are new type of attacks where more number of packets are transmitted in the short period of time and the process is repeated for several intervals [3]. The congestion is occurred in the network due to the increased link capacity since the packet sending rate is high. Though many detection and prevention methods for LRDoS attacks have been developed, it has very complex due to low average rate during network congestion [4].

Fernandez, G. M., et al. [5] proposed defense mechanism for LRDoS attacks against application servers. The main objectives of the defense mechanism were preventing an attacker from capturing the locations in the incoming queues of applications and randomizing the server function for removing the possible vulnerabilities due to the predictable behaviours. In this paper, different DoS detection mechanisms were analyzed Like Random Service Time (RST), Random Answer Instant (RAI), Random Time Queue Blocking (RTQB) and Improved Random Time Queue Blocking (IRTQB). Among these

Manuscript received March 5, 2026
Manuscript revised March 20, 2026
https://doi.org/10.22937/IJCSNS.2026.26.3.12

mechanisms, IRTQB has outperforms than the other methods but distinguishing between the attack requests from the legitimate user requests was not satisfied. Therefore, the IRTQB was improved by incorporating the bandwidth utilization and named as Advanced Random Time Queue Blocking (ARTQB). In addition, the traffic prediction was also included with the ARTQB mechanism (ARTQB-TP) for preventing the LRDoS attacks against application servers.

However, the congestion handling techniques at the edge routers such as Active Queue Management (AQM) mechanisms were not considered to prevent these attacks in previous approaches. Hence, in this paper, AQM mechanism is integrated with the ARTQB-TP for identifying the congestion associated with LRDoS attack traffic. In the proposed mechanism, Deterministic Fair Sharing (DFS) scheme is introduced as AQM. The proposed DFS utilizes the novel Weighted Fair Share (WFS) for dynamically determining the router buffer utilization according to the congestion level which is useful for malicious flow detection. The state of legitimate and malicious flows is maintained by using the multiple data structures like B+tree which are used for storing the flow states or managing the lookup tables. In addition, DFS has the ability for optimizing its performance like higher bandwidth flows is handled by the cache.

The remaining article is organized as follows: Section 2 describes about the related DoS attack detection and defense mechanism briefly. Section 3 explains the proposed methodologies. Section 4 illustrates the experimental results of the proposed mechanism compared with the previous mechanisms. Section 5 concludes and presents the future scope of the research work.

2. Literature Survey

Mishra, S., & Bathla, A. K. [6] proposed the Active Queue Management (AQM) techniques which improves the performance of the network by reducing the congestion inside the network. The major objective of the AQM was analyzing the effect of different queuing algorithms such as DropTail, Fair Queuing (FQ), Stochastic Fair Queuing (SFQ), Deficit Round Robin (DRR) and Random Early Detection (RED). Each algorithm has its own advantages. The congestion inside the network was detected based on the Quality of Service (QoS) to be delivered by the network. However, traffic management algorithms were also required for enhancing the AQM.

Xu, X., et al. [7] investigated about the Low-Rate Denial-of-service attacks against Application Servers (LoRDAS). In this paper, a modified strategy for forecasting and removing the LoRDAS attacks in the network by differentiating the attack behaviours. Then, the queue model was introduced for characterizing the dynamic behaviour of the attack. Furthermore, the effect of attack was measured by computing the steady-state probabilities and corresponding traffic rate of the queue system. However, the complexity of the mechanism was high and the effectiveness was less compared with other algorithms.

Piedrahita, A. F. M., et al. [8] proposed FlowFence mechanism which is lightweight and fast DoS detection and defense model for Software Defined Networking (SDN). The proposed FlowFence consists of router running daemons for monitoring the average occupation of their interfaces which helps to detect the congestion conditions and SDN controller which is utilized for coordinating the bandwidth assignment of controlled links. The prototype of FlowFence was implemented in the Future Internet Testbed with Security (FITS). However, the time required for each flow was high.

Kumar, G. D., et al. [9] proposed the multi-layered probabilistic model for Distributed DoS attacks. The most significant features of each attack were described and also the defense mechanism was explained along with the merits and demerits. The major objective of the proposed mechanism was improving the conventional attacks and defense mechanisms for understanding the DDoS attacks.

Spyridopoulos, T., et al. [10] proposed game theoretic defense model against DDoS cyber attacks. The prospects of game theory were investigated by modelling the transmission rate of each attacker's nodes including with the transmission rate of legitimate nodes. In addition, the implications for firewall design and performance were also evaluated based on the assumptions about the traffic flow modelling, host victimisation rate and other parameters. The performance of the game theoretic approach was evaluated in both theoretical and simulation.

Zhou, W., et al. [11] proposed the novel approach for detecting the Application-Layer (AL) DDoS attacks. The proposed approach was introduced for constructing the Real-time Frequency Vector (RFV) and real-timely characterizes the traffic as a set of models. These models were used for recognizing the real AL-DDoS attacks by computing the entropy of AL-DDoS attacks and flash crowds. These principals were integrated into the modularized defense model which contains the head-end sensor, detection module and the traffic filter. However, the reaction rate was not improved by this method.

Jiang, X., et al. [12] proposed the approach named Random Early Detection (RED) along with the trust value of flows for defense against DoS attacks. The flow trust was employed into the safeguard legitimate flows. The network flows were monitored the router and the trust values of flows were computed which is utilized for the relevant queue management. This approach was developed as a significant decision-making factor of AQM for improving the robustness of defense algorithm. However, the computational complexity of the method was high.

Thakur, K. [13] investigated about the issues in DoS flooding attacks and the countermeasures were provided for preventing the attacks by detecting and responding the DoS attacks. In the proposed approach, the secondary data collection method was introduced and the data was collected through the different online and offline sources. Moreover, the cyber-security TCP-SYN was detected for mitigating and blocking the DDoS attacks. However, the proposed approach was identified to be limited for different applications.

3. Proposed Methodology

3.1 Overview of ARTQB-TP Mechanism

In ARTQB-TP mechanism, the Spatial Similarity Metric (SSM) was computed by considering the record timestamps, source IP addresses, bandwidth utilization between two requests and the difference between the attack traffic and legitimate traffic. The SSM is computed as,

$$SSM(A_i, A_j) =\neq \_consecutive_{bits_1}(A_i XNOR A_j) + BW(A_i, A_j) + T \quad (1)$$

In equation (1), $consecutive_{bits_1}(A_i XNOR A_j)$ is the number of consecutive bits set to 1 in the Bitwise XNOR of two addresses A_i and A_j, BW refers the bandwidth between A_i and A_j requests and T denotes the classification threshold which defines whether the traffic is attack traffic or legitimate traffic. However, any mechanism related to congestion control was not considered since the overflow at the edge router may also affect the legitimate flows. Therefore, the operational overhead was also increased.

To reduce the operational overhead and control the congestion, ARTQB-TP is included with the AQM technique. Here, the low operation overhead is provided by using heuristic and probabilistic measures.

3.2 Deterministic Fair Sharing (DFS)

Deterministic Fair Sharing (DFS) is utilized as AQM technique for identifying the malicious flows. When legitimate flows are protected, DoS for legitimate flows drop all incoming traffic from the malicious flows for ensuring the least damage [14]. When incoming packets from malicious flows are not allowed in the buffer, then it must have the ability to acknowledge their behavioural changes if they become legitimate in the future. If there is no attack, then an ideal AQM technique has the ability for providing the high degree of fairness among competing legitimate flows when overall queue stability is maintained. This is achieved by keeping per-flow state which reduces the operational overhead. Figure 1 illustrates that the block diagram of AQM mechanism. If there is no attack, then the incoming traffic consists of only legitimate flows which are responsive (TCP) flows and unresponsive (UDP) flows within the fair rate. An additive Increase/Multiplicative Decrease (AIMD) is utilized as feedback control algorithm by TCP for providing these TCP flows highly sensitive to packet drops. Therefore, high degree of fairness among such competing flows is ensured by packet drop or Explicit Congestion Notification (ECN) mark decisions which are analyzed for each incoming packet, and flow. The optimal solution is not guaranteed by separating the available buffer space fairly among all incoming flows. Therefore, an AQM scheme is required for sharing the buffer fairly among all competing flows when the queue length is always stable. In addition, the buffer should not be under utilized or overflown. Hence, the weighted fair share (wfs) method is introduced for obtaining these properties.

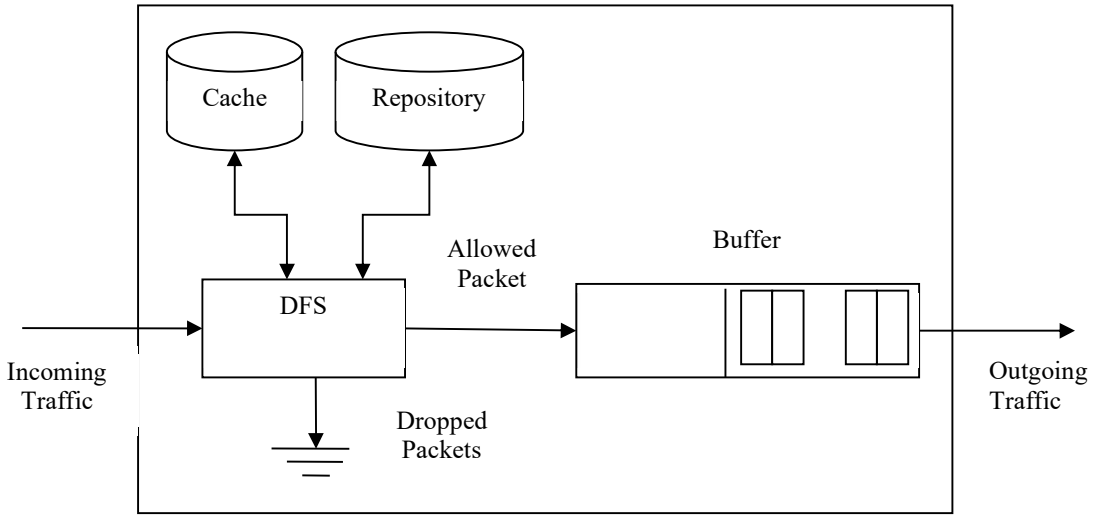

Figure.1 Block Diagram of AQM Mechanism

The fair buffer share is determined by wfs for each competing flow based on the scaling factor which is according to the available empty buffer space. It is defined as the dynamic value updated for each incoming packet and is utilized for deciding whether the packet is allowed or dropped.

$$wfs = \frac{b}{n} \cdot \left(\frac{100}{q_p} - 1\right) \qquad (2)$$

In equation (2), q_p refers the instantaneous current queue length (q) in percentage, b is the router buffer capacity and n is the total number of active flows. The fair share of buffer that each flow is allowed to have is represented by $\frac{b}{n}$ and it is useful for providing the fairness among competing flows. The scaling factor is denoted as $\frac{100}{q_p} - 1$ which is helpful for assigning the dynamic weight to the fair share of buffer $\frac{b}{n}$. Hence, the queue is stabilized and preserved from overflow at all times and therefore the latency is minimized. The queue stability and fairness is provided by dynamically adjusting the queue length of each incoming packet. Moreover, the DFS utilizes the Exponential Weighted Moving Average (EWMA) of wfs for smoothing the queue length and reducing its high variance [15].

Even during an attack, the properties of fairness and stability are ensured by maintaining the variable p_m for each flow which refers its behaviour. Higher p_m is assigned to higher unfair behaviour by the particular flow. Moreover, the behaviour of malicious flows is identified by tracking the bit rates of each incoming flow while allowing into the buffer.

DFS has two operations before an incoming packet enters into the buffer. Such operations are Enque and Deque. For each incoming packet p, different attributes are extracted such as $fid_p, time_p, size_p$ and BW_p. In which, fid_p is the flow ID used for identifying the flows. It includes the sender and receiver's IP addresses and port numbers. The packet timestamp and size are denoted as $time_p$ and $size_p$ respectively. The bandwidth utilization between two flows is represented as BW_p. The behaviour of the specific flow is determined by it's p_m which denotes its degree of unfairness. The particular flow is identified as unfair if its space in buffer (sz) exceeds wfs at a given time. The p_m value is adjusted by utilizing the set of parameters. The p_m value is incremented and decremented by the variables δ_i and δ_d. The maximum and minimum p_m value is obtained by two thresholds min_{th} and max_{th}.

By using higher values for δ_i, DFS is provided as more aggressive towards high bandwidth flows whereas by using higher values for δ_d it is provided as more lenient. The variable $freeze\_time$ is represented as time period for waiting before p_m is updated again. This value is assigned according to the RTT of flows multiplexed together. If the flow is identified as unfair for long time period, its p_m value reaches max_{th} and marked as malicious by setting their Boolean mark. Therefore, DFS drops all incoming packets from these flows. Moreover, these malicious flows are corrected in future by tracking their bit rates and reduced it to the fair rates. The Boolean $store_{br}$ is used for determining the flows whose bit rates are required to be stored. The attribute br is used for keeping the stored bit rate. By keeping track of bit rates, the malicious flows are disabled without affecting the functions of DFS.

3.3 Enque and Deque Operation

During Enque operation, fid_p of each incoming packet is extracted. The B+tree and Cache are searched in sequence for identifying any potential matches [16]. Since these two data structures are mutually exclusive for storing the flows. If the match is identified in Cache, then B+tree are not searched. If match is identified then the corresponding flow is updated and determined whether the current packet is allowed or not by using ProcessPacketAndDecide operation. If the packet is allowed, then the value of matching flow i size in buffer (sz_i) is incremented by the packet's size $(size_p)$. Similarly, the Deque function decrements (sz_i) by $(size_p)$, when the packet is dequeued from the buffer.

The flows are moved between B+tree and Cache according to their behaviour which is defined by their p_m value. If the flow is identified in B+tree and has $p_m = max_{th}$, then it is shifted to the Cache since it is known as high bandwidth flow. If the flow is identified in Cache and has $p_m = min_{th}$, then it is shifted to B+tree since this flow has changed from being malicious to the legitimate flow. If fid_p of the incoming packet is not matched with any flow in B+tree or Cache, then the new flow with matching fid_p is generated in the B+tree.

The inactive flows from Cache and B+tree are deleted by Purge operation which is performed periodically. Its execution frequency is adjusted by the variable $prune\_interval$. During each function, Purge deletes all inactive flows effectively.

Algorithm: ARTQB-TP-AQM Mechanism

1. For each incoming Request R_a
2. Compute attack interval t_I and bandwidth utilization b_I
3. For each incoming packet do
4. Procedure $ENQUE(p)$ // p- incoming packet

5. $if(q \geq b_l)$ then
6. $Drop(p)$
7. $if(matching\ flow\ i\ is\ in\ Cache\ or\ B+tree)$ then
8. $if\ ProcessPacketAndDecide(p,i)$ then
9. $Drop(p)$
10. $else$
11. $sz_i \leftarrow sz_i + size_p$
12. $Allow(p)$
13. $if\left(p_{m_i} \leq min_{th} \& flow\ i \in Cache\right)$ then
14. Shift flow i to B+tree
15. $else\ if\left(p_{m_i} \geq max_{th} \& flow\ i \in B+tree\right)$ then
16. Shift flow i to Cache
17. else
18. $InsertNewFlow(p)$
19. $if(Current\_time - last\_prune > prune\_interval)$ then
20. $Prune(last\_prune)$
21. $last\_prune = current\_time$
22. End for
23. End Procedure

//ProcessPacketAndDecide Function

24. Procedure $PROCESSPACKETANDDECIDE(p,i)$
25. $bool\ drop \leftarrow false$
26. $time\_gap \leftarrow time_p - p_{time_i}$ // p_{time_i} - Timestamp when p_m was last updated
27. $if\left(p_{m_i} \leq min_{th} + \delta_i\right)$ then
28. $if\left(sz_i > EWMA(wfs)\right)$ then
29. $ECN\ Mark(p)$
30. $if\ (time\_gap > freeze\_time)$ then
31. $store_{br_i} \leftarrow 1$
32. $p_{m_i} \leftarrow p_{m_i} + \delta_i$
33. $p_{time_i} \leftarrow time_p$
34. $else\ if\ sz_i = 0$ then
35. $p_{m_i} \leftarrow p_{m_i} - \delta_d$
36. $p_{time_i} \leftarrow time_p$
37. $if\left(p_{m_i} \leq min_{th}\right)$ then
38. $ResetFlow(i)$
39. $else$
40. $ECN\ Mark(p)$
41. $if(mark == malicious)$ then
42. $drop \leftarrow true$
43. $if(time\_gap > freeze\_time\ \&store_{br_i} \neq 0)$ then
44. $if(br_i > fair_{br})$ then
45. $\delta \leftarrow \frac{br_i}{fair_{br}}$
46. $p_{m_i} \leftarrow p_{m_i} + (\delta.\delta_i)$
47. $else$
48. $\delta \leftarrow \frac{fair_{br}}{br_i}$
49. $p_{m_i} \leftarrow p_{m_i} - (\delta.\delta_d)$
50. $p_{time_i} \leftarrow time_p$
51. $if\left(p_{m_i} > max_{th}\right)$ then
52. $mark \leftarrow 1$
53. Return drop
54. End Procedure
55.

4. Experimental Evaluation

In this section, the proposed ARTQB-TP-AQM mechanism is evaluated experimentally by using Network Simulator-2 (NS2). The performance of LRDoS attack is analyzed by measuring the mean-in-system time and attack efficiency under different scenarios. The LRDoS attack is employed in the application server for evaluating the attack impact in the server. The different configurations values for attack and server parameters are shown in Table 1.

Table 1: Configuration Values for the Attack and Server Parameters

Parameter	Value
Duration of attack burst, B	0.4s
Time between attack packets in a burst	0.2s
Mean service time, Ts	12s
Variance of server, var[Ts]	0(S_1), 0.2(S_2,S_3)
Interval between legitimate users requests	3s (S_1,S_2),0.95s (S_3)
Number of server threads	1(S_1,S_2), 4 (S_3)
Number of positions in service queue, N	4 (S_1,S_2),8 (S_3)
Number of attack threads	-N
Round trip time, RTT	1s
Similarity metric, ST	32

Scenario 1 (S_1): the server is mono-threaded and the variances of the service time for attack requests, var[T_s], and var[B_s] are 0. Scenario 2 (S_2): the server variances of var[T_s], and var[B_s] are modified. Scenario 3 (S_3): the main objective of this scenario is checking how a multithread operation in the server affects the performance of a given defense technique.

A). Attack Efficiency

Attack efficiency is defined as the percentage of service queue locations which are captured by the attacker over the sum amount of locations captured during the attack execution.

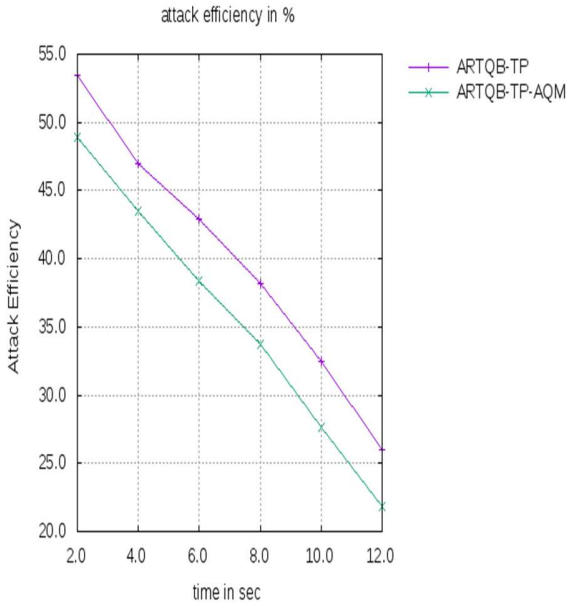

Figure.2. Scenario 1 Attack Efficiency (%)

Figure 2 shows that the attack efficiency comparison of ARTQB-TP and ARTQB-TP-AQM during scenario 1 (mono-threaded with zero variance). In scenario 1, the attack efficiency of ARTQB-TP-AQM has been much reduced than the ARTQB-TP mechanism since, consideration of the flow state. It shows that when the time period is 12sec, the attack efficiency of ARTQB-TP-AQM is 16.15% which is lower than the ARTQB-TP mechanism.

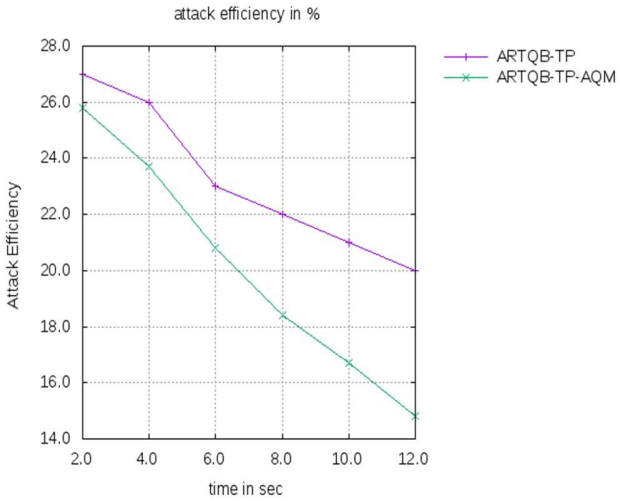

Figure.3. Scenario 2 Attack Efficiency (%)

Figure 3 shows that the attack efficiency comparison of ARTQB-TP and ARTQB-TP-AQM during scenario 2 (di- threaded with variance 0.2). The attack efficiency of ARTQB-TP-AQM has been much reduced than the ARTQB-TP mechanism since, consideration of the flow state. It shows that when the time period is 12sec, the attack efficiency of ARTQB-TP-AQM is 26% which is lower than the ARTQB-TP mechanism.

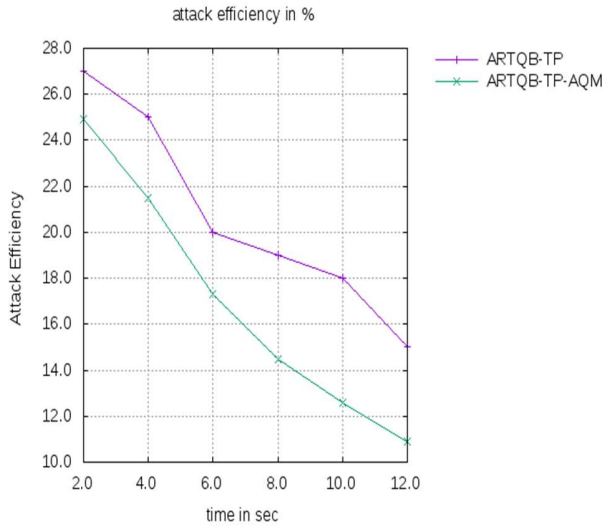

Figure.4. Scenario 3 Attack Efficiency (%)

Figure 4 shows that the attack efficiency comparison of ARTQB-TP and ARTQB-TP-AQM during scenario 3 (multi-threaded with variance 0.2). The attack efficiency of ARTQB-TP-AQM has been much reduced than the ARTQB-TP mechanism since, consideration of the flow state. It shows that when the time period is 12sec, the attack efficiency of ARTQB-TP-AQM is 27.33% which is lower than the ARTQB-TP mechanism.

B). Mean in-system Time

Mean in-system time is defined as the time from when the request enters the server to the instant at which its corresponding answer is transmitted.

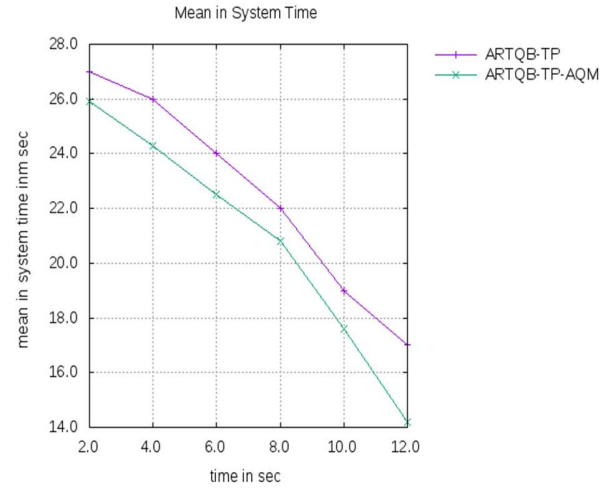

Figure.5. Scenario 1 Mean in-system time (s)

Figure 5 shows that the mean in-system time comparison of ARTQB-TP and ARTQB-TP-AQM during scenario 1 (mono-threaded with zero variance). The mean in-system time of ARTQB-TP-AQM has been much reduced than the ARTQB-TP mechanism since, consideration of the flow state. It shows that when the time period is 12sec, the mean in-system time of ARTQB-TP-AQM is 16.47% which is lower than the ARTQB-TP mechanism.

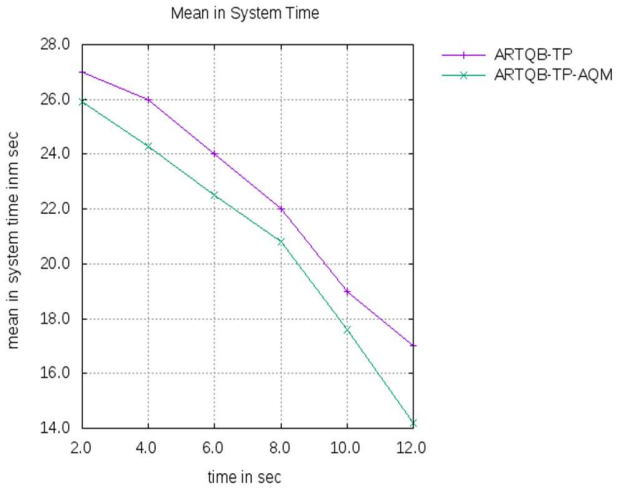

Figure.6. Scenario 2 Mean in-system time (s)

Figure 6 shows that the mean in-system time comparison of ARTQB-TP and ARTQB-TP-AQM during scenario 2 (di- threaded with variance 0.2). The mean in-system time of ARTQB-TP-AQM has been much reduced than the ARTQB-TP mechanism since, consideration of the flow state. It shows that when the time period is 12sec, the mean in-system time of ARTQB-TP-AQM is 19% which is lower than the ARTQB-TP mechanism.

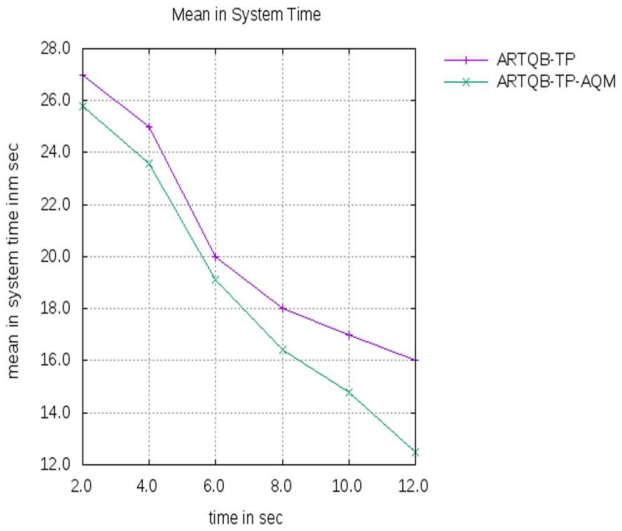

Figure.7. Scenario 3 Mean in-system time (s)

Figure 7 shows that the mean in-system time comparison of ARTQB-TP and ARTQB-TP-AQM during scenario 3 (multi-threaded with variance 0.2). The mean in-system time of ARTQB-TP-AQM has been much reduced than the ARTQB-TP mechanism since, consideration of the flow state. It shows that when the time period is 12sec, the mean in-system time of ARTQB-TP-AQM is 21.88% which is lower than the ARTQB-TP mechanism.

5. Conclusion

The LRDoS attacks detection is the most significant for ensuring the application server behaviours. Though, ARTQB-TP reduces the attack efficiency of LRDoS, still congestion constraints are not handled in order to reduce the malicious flows through the network. In this paper, Active Queue Management (AQM) mechanism is proposed for identifying and mitigating the congestion oriented DoS attack traffic. The DFS scheme is introduced included with the ARTQB-TP which utilizes the group of data structures for providing low operational overhead by maintaining the limited per-flow state. In addition, the proposed mechanism offers high DoS attack identification capability. The performance of ARTQB-TP-AQM mechanism with different scenarios is analyzed through NS2. Finally, the experimental results prove that the proposed ARTQB-TP-AQM mechanism reduces the attack efficiency, traffic overhead and improves the degree of fairness, throughput to legitimate flows by allowing the low bandwidth to the attack traffic and stabilizing the router queue length.

References

[1] Mantas, G., Stakhanova, N., Gonzalez, H., Jazi, H. H., & Ghorbani, A. A. (2015). Application-layer denial of service attacks: taxonomy and survey. *International Journal of Information and Computer Security*, *7*(2-4), 216-239.

[2] Akiwate, B., Desai, M., & Surpurmath, S. (2016). Detection and prevention of DoS attack. *International Journal of Advanced Research in Computer Science and Software Engineering, 6*(5), 639-642.

[3] Shashidhara, H. V., & Balaji, D. S. (2014). Low Rate Denial of Service (LDoS) attack–A Survey. *International Journal of Engineering Technology and Advanced Engineering Volume*, *4*.

[4] Bhuyan, M. H., Bhattacharyya, D. K., & Kalita, J. K. (2015). An empirical evaluation of information metrics for low-rate and high-rate DDoS attack detection. *Pattern Recognition Letters*, *51*, 1-7.

[5] Maciá-Fernández, G., Rodríguez-Gómez, R. A., & Díaz-Verdejo, J. E. (2010). Defense techniques for low-rate DoS attacks against application servers. *Computer Networks*, *54*(15), 2711-2727. Mishra, S., & Bathla, A. K. (2015, September). Active Queue Management based Solution for Improving Performance under DDOS Attacks. *IJCA Proceedings on International Conference on Computer Technology (ICCT),* 6, 1-5.

[6] Mishra, S., & Bathla, A. K. (2015, September). Active Queue Management based Solution for Improving Performance under DDOS Attacks. *IJCA Proceedings on International Conference on Computer Technology (ICCT),* 6, 1-5.

[7] Xu, X., Guo, X., & Zhu, S. (2010, June). A queuing analysis for low-rate DoS attacks against application servers. In *Wireless Communications, Networking and Information Security (WCNIS), 2010 IEEE International Conference on* (pp. 500-504). IEEE.

[8] Piedrahita, A. F. M., Rueda, S., Mattos, D. M., & Duarte, O. C. M. (2015, October). FlowFence: a denial of service defense system for software defined networking. In *Global Information Infrastructure and Networking Symposium (GIIS), 2015* (pp. 1-6). IEEE.

[9] Kumar, G. D., Rao, C. G., Ahmed, M. W., & Satyanarayana, G. (2013, June). Multilayered Probabilistic Model for

Distributed DoS Attacks. In *Computational Science and Its Applications (ICCSA), 2013 13th International Conference on* (pp. 99-104). IEEE.

[10] Spyridopoulos, T., Karanikas, G., Tryfonas, T., & Oikonomou, G. (2013). A game theoretic defence framework against DoS/DDoS cyber attacks. *Computers & Security*, *38*, 39-50.

[11] Zhou, W., Jia, W., Wen, S., Xiang, Y., & Zhou, W. (2014). Detection and defense of application-layer DDoS attacks in backbone web traffic. *Future Generation Computer Systems*, *38*, 36-46.

[12] Jiang, X., Yang, J., Jin, G., & Wei, W. (2013). RED-FT: A scalable random early detection scheme with flow trust against DoS attacks. *IEEE Communications letters*, *17*(5), 1032-1035.

[13] Thakur, K. (2015, September). Analysis of Denial of Services (DOS) Attacks and Prevention Techniques. In *International Journal of Engineering Research and Technology* (Vol. 4, No. 07, July-2015). IJERT.

[14] Bedi, H., Roy, S., & Shiva, S. (2015). Mitigating congestion based DoS attacks with an enhanced AQM technique. *Computer Communications*, *56*, 60-73.

[15] Singh, N. G. (2015). Comparing Different Active Queue Management Techniques. *International Journal of Emerging Research in Management &Technology ISSN*, 2278-9359.

[16] Chen, S., Gibbons, P. B., Mowry, T. C., & Valentin, G. (2002). Fractal prefetching B+-trees: Optimizing both cache and disk performance. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data* (pp. 157-168). ACM.