Analyzing the Performance of Web-services during Traffic Anomalies

Avneet Dhingra[1]
Research Scholar, Department of Computer Science and Engineering, I.K. Gujral Punjab Technical University
Kapurthala-144603, India

Monika Sachdeva[2]
Associate Professor, Department of Computer Science and Engineering, I.K. Gujral Punjab Technical University
Kapurthala-144603, India

***Abstract*—Intentional or unintentional, service denial leads to substantial economic and reputation losses to the users and the web-service provider. However, it is possible to take proper measures only if we understand and quantify the impact of such anomalies on the victim. In this paper, essential performance metrics distinguishing both transmission issues and application issues have been discussed and evaluated. The legitimate and attack traffic has been synthetically generated in hybrid testbed using open-source software tools. The experiment covers two scenarios, representing DDoS attacks and Flash Events, with varying attack strengths to analyze the impact of anomalies on the server and the network. It has been demonstrated that as the traffic surges, response time increases, and the performance of the target web-server degrades. The performance of the server and the network is measured using various network level, application level, and aggregate level metrics, including throughput, average response time, number of legitimate active connections and percentage of failed transactions.**

Keywords—Denial of service; DDoS attack; flash event; performance metrics; throughput; response time

I. Introduction

In the event of network traffic anomaly, the users get to grips with either a drastic slowdown of the service or a complete outage. Recent years have witnessed a rise in the frequency and strength of some illegitimate anomalies known as DDoS attacks. These attacks compromise the availability of the web-services of the victim server. The motive behind such activity varies from being personal to political. Whatever the cause, these attacks can be very troublesome and costly for a target. For instance, the online encyclopedia, Wikipedia, suffered a DDoS attack on September 6, 2019, that lasted for about three days [1]. The intermittent outages and performance degradation were faced by users in the Middle East, Europe, the United Kingdom, and the United States. Many such instances are confronted daily, across the globe, due to an exponential increase in the use of Internet-based applications. As per the Kaspersky report, the number of attacks has increased by 80% in the first three months of 2020 against the attacks observed in 2019 [2].

The need hence arises to generate realistic techniques to evaluate the performance and measure the impact of anomalies (legitimate or illegitimate) on the services of the web-server. Measuring the performance of the server under such anomalies can help understand the preventive techniques required to be installed along with the type of potential defenses. The importance of performance testing is also realized in the situation when multiple users generate concurrent traffic creating a heavy load on the network similar to that created during a DDoS attack. The network responding to anomalies needs to be tested repetitively with short-duration attack traffic to evaluate the overall performance of the server and the cost involved for installing the required security measures. The metric, thus calculated, provides the information related to the network traffic in case of saturation.

The literature reviewed [3,4,5] for the impact analysis highlights the use of a simulator for generating traffic and analyzing the performance of the network. However, the experiment presented in this paper makes use of emulation to generate synthetic traffic. Emulation has the advantage of using real-time OS and apps, along with the simulated elements such as virtual nodes and soft network links. Exploits. The paper also presents the exhaustive review undertaken to comprehend the concept of performance and quantifying the impact of anomalies on the web-services. Various application-level, as well as network-level and server-level performance metrics, have been identified and evaluated using the synthetically generated traffic in DDoSTB hybrid testbed [6]. The results of the study have been presented as graphs showing the effect of traffic surges and realize their impact on performance. The background traffic is mainly composed of TCP protocol. The attack traffic is composed of UDP, with varying packets per second. The HTTP traffic is generated with varying percentage of requests per second and represents the flash event (a legitimate anomaly). The paper defines performance metrics quantifying the quality of service (QoS) of the web server during normal conditions and under the increased traffic load. The experimental set-up and procedure to evaluate performance metrics of the designed network have been discussed.

The paper has been organized as follows. The related literature has been reviewed in Section 2. Section 3 gives an overview of what performance metrics are and its importance in the detection of anomalies. Section 4 describes the model of an experimental network using realistic topology and software tools used to generate legitimate and attack traffic. Section 5 discusses the metrics selected for analysis, and the results obtained are presented as graphs for better understanding. The paper concludes the observations of the experiment in Section 6. The scope for future work in the same field has been mentioned in section 7.

II. Literature Review

Researchers have studied the damage caused due to the poor performance of the victim server and suggested specific damage models. The models give an insight into the situation, the risks involved, and highlight the importance of analyzing and evaluating the performance of the system. One such model proposed by [7], divides the financial damage into characteristics like disaster recovery, loss due to downtime, liabilities involved, and losing the customers. The model can be generalized to any traffic anomaly affecting the availability of the server (like FE) and hence degrading the performance.

Vasudevan et al. [8] recommended metrics based on the cost of losing customers and the cost of SLA violations, which is from the point of view of the network users. It draws attention to the possible financial impact a DDoS attack can have on the network-provider of the affected network. The MIDAS2007NET factor has been defined based on the fact that allocated bandwidth is proportional to the volumes of traffic on the network and which in turn are proportional to the related profits.

Gade et al. [9] studied the impact of the DoS Land (Local Area Network Denial) attack to compare the memory and processor utilization during the attack. Chertov et al. [10] concluded that simulated networks produced different results from the emulated ones. The difference is because of the assumptions taken for simulation. The authors suggest the use of testbeds for accurate results. The metrics computed to measure performance are the average goodput (computed using transfer size and time for completion of transfer), size of the congestion window (calculated by dividing the average of weighted congestion window by average of segment size), percentage of CPU utilization and packets sent (and received) per second.

Mirkovic et al. [11,12,13,14] defined the various legacy metrics along with the applications where these metrics give accurate results. The metrics discussed are packet loss, throughput, request/response delay, transaction duration, and allocation of resources. Packets lost in the transit after an anomaly hits the network is termed as packet loss. In this case, a network could be either a direct hit network or nearby networks experiencing collateral damage. It measures the network congestion caused by flooding attacks. Throughput is the number of bytes transmitted or re-transmitted per time-interval. Goodput is the same as throughput with the difference that re-transmitted bytes are not counted. This metric, however, does not give accurate results for connections with few packets as the throughput, in this case, is already low.

Request/response delay metric cannot be applied to non-interactive applications and one-way traffic. Transaction duration captures the time required for exchanging a set of messages between a source and destination. The metric efficiently measures the services for the interactive applications (example-browsing internet) but fails to give results for one-way traffic. The allocation of resources is in proportion to a critical shared resource, like bandwidth. The ratio of resources allocated to legitimate traffic versus the attack traffic captures the service quality as viewed by the user. It measures the server load but fails to determine the collateral damage caused to the network. The authors categorized the applications into five groups: interactive applications (web-based), media applications (audio-video streaming), online games, chat applications, and non-interactive applications(email). Metrics vary across these applications and cannot be generalized. They proposed a few impact metrics keeping in mind the QoS requirements of different applications. One of them being a percentage of failed transactions (pft), which quantifies the quality of services experienced by the users. Another metric, DoS-hist, shows the resilience of an application to an attack with the help of histograms for pft.

Singh et al. [3], has applied application layer metrics and network layer metrics on the trace generated using the NS-2 simulator. The author speculates that the network level and transport-level parameters are not sufficient by themselves to detect application-layer attacks. They have checked the performance of the network for various GET-HTTP DDoS attacks.

Sachdeva et al. [4] have defined the ratio of average serve rate and average request rate to check the performance. This ratio is 1 for normal traffic and decreases as the strength of attack increases. These are evaluated using the NS-2 simulator. Sachdeva et al [15] evaluate the DDoS and FE impact on web services using DETER testbed. The metric throughput was evaluated as goodput and badput. Badput is defined as the attack traffic over bottleneck link during a given time window. Authors have experimented with traffic for response time, percentage of request packets lost, legitimate packet survival ratio, percentage of failed transactions, and bottleneck bandwidth utilization (for goodput).

Bhatia et al. [16] recommended the performance evaluation of network using server-level metrics viz, system-level CPU utilization, user-level CPU utilization, CPU load, and real memory utilization. The authors have proposed a framework for generating realistic traffic for the anomalies under study, using minimal hardware. Apart from the legacy metrics, Behal et al. [6] have used sever load metrics – CPU utilization, memory utilization, and CPU load to test the performance of the server. The researcher has developed DDoSTB testbed to generate the required synthetic traffic for experimentation.

Bhandari et al. [17] consolidate the metrics used for evaluating the performance of the defense framework and classifies them at a packet level, aggregate level, and application-level based on the level of the network they are used for. The author has also discussed the system parameters affected in the case of DDoS attacks and the different tools to measure the same.

Performance metrics have been classified as external metrics and internal metrics [6, 11, 17, 18]. The metrics that do not need privileged access to the system or its network are termed are external—for instance, attack strength or defense parameters. The internal metrics measure CPU load, CPU utilization, memory utilization, internal algorithms, and data structures. Another valid criterion for classification is the OSI layer involved in measuring the metric [6, 18]. The metric may measure the aggregate performance of networks such as throughput, or it may work on application level like transaction duration or at a packet level, such as a number of re-transmissions [4, 17].

III. Performance Metrics

As each network has a different topology and varies in purpose, different factors define the performance for each environment. Performance metrics quantify the QoS provided by the server, during normal conditions, and under the increased traffic load. The performance of a network depends on the following factors:

- The rate at which the information can be or is transferred,
- The delay between the request sent, and the response received,
- Error in transmission.

These factors are quantified with generated metrics, keeping in view the user requirements for a particular application. Mirkovic et al. [12] have explicated the response times of various kinds of applications and concluded that there is no particular set of metrics that can be considered as the benchmark for measurement of performance. The interactive applications are expected to respond in minimum time. For such applications, including VoIP, video streaming, chatting, and gaming applications, the delay is expected to be the minimum. The applications, such as email transfer, which are non-interactive, do not have any defined or tolerable limit for the delay. Also, there is no specific threshold or baseline to define the best performance. To efficiently examine the impact of anomalies in network traffic, metrics need to be accurate, quantitative, and versatile [11]. Ideally, this can be achieved using realistic networks, giving a holistic view of the network parameters. However, it is practically not possible to have such an expansive view due to specific economic, legal, and privacy issues. Thus, techniques like simulation or emulation are used to create a virtual network that can generate realistic synthetic traffic.

Table I outlines various network level, application level, and aggregate level metrics evaluated in this paper. As per [3], network level, transport level, and application-level metrics need to be assessed together to evaluate the overall performance of the network.

TABLE I. Performance Metrics

Level	Metric
Aggregate	Throughput (Goodput, Badput)
Application	Response time Number of legitimate requests dropped Number of legitimate active connections Percentage of failed transactions Average serve rate/ Average request rate Legitimate packet drop probability
Network	Percentage of link utilization
Server	CPU-utilization

IV. Experimental Set-up

The set-up of an experiment and its procedure includes three components - network topology, legitimate traffic, and attack traffic [14]. The best option is to set up an experimental environment in a real-time operational network that handles live traffic. However, the drawback involved in a real-time environment is that it cannot be reconfigured or scaled as per the experiments' needs. Thus, other options of simulated environment or emulated environment can be taken into consideration. Out of the two, simulation and emulation, the former lacks realism, and traffic replay is slow. Thus, the latter is the preferred method for testing by the researchers [3, 6, 13, 15, 19]. Emulator, for instance - DETER, is the combination of real hardware plus simulator to achieve the desired topology of a network. It provides a more stable environment as compared to a theoretical model or simulation alone. Soft routers and soft network links are used with the real systems real applications. However, the emulator is scalable only to a certain extent. Identifying the compatible tool along with the data source and its deployment on the systems can, somehow, be a challenge.

In this section, the performance of the testbed network has been carefully examined using the three components discussed in [14]. Flooding attack is generated using the realistic topology on the available testbed with the help of available and compatible software tools.

A. Network Topology

For the experimentation, authors used the hybrid DDoSTB Testbed, developed by Behal et al. [6,20], consisting of real as well as emulated systems. 45 physical systems have been deployed and grouped into 3 clusters of 15 computers each. Out of the 3 clusters, 2 clusters are specified for background traffic and 1 cluster for attack traffic. The systems are run on Windows XP and Ubuntu instances. To increase the nodes, the v-core emulator [6] is used for attack cluster that is, cluster 3. One v-core node implements 30 virtual nodes. This scales the network to (15 x 30) 450 attack nodes. Each node in cluster 1 and cluster 2 is deployed with 30 instances of Apache JMeter to scale the number of legitimate nodes to a total of 900 nodes. For making identification of source easier, different pools of IP-address are assigned to the users in each cluster. The victim web server gives access to the resources to the users from both legitimate and attack clusters. It records every request attempt as a log file. The topology deployed on DDoSTB is shown in Fig. 1, and the associated parameters have been summed up in Table II. Generally, the typical link speeds are used to assign link bandwidths. In Fig. 1, the link between R1 and server acts as the bottleneck link. The bandwidth of this link is 100 Mbps, with a delay of 5ms. The rest of the links in the topology have a bandwidth of 1 Gbps. Topology for the experiment is finalized considering the principles of benchmarking, as suggested in [21]. An attempt has been made to keep it realistic and comparable to the topology of the Internet.

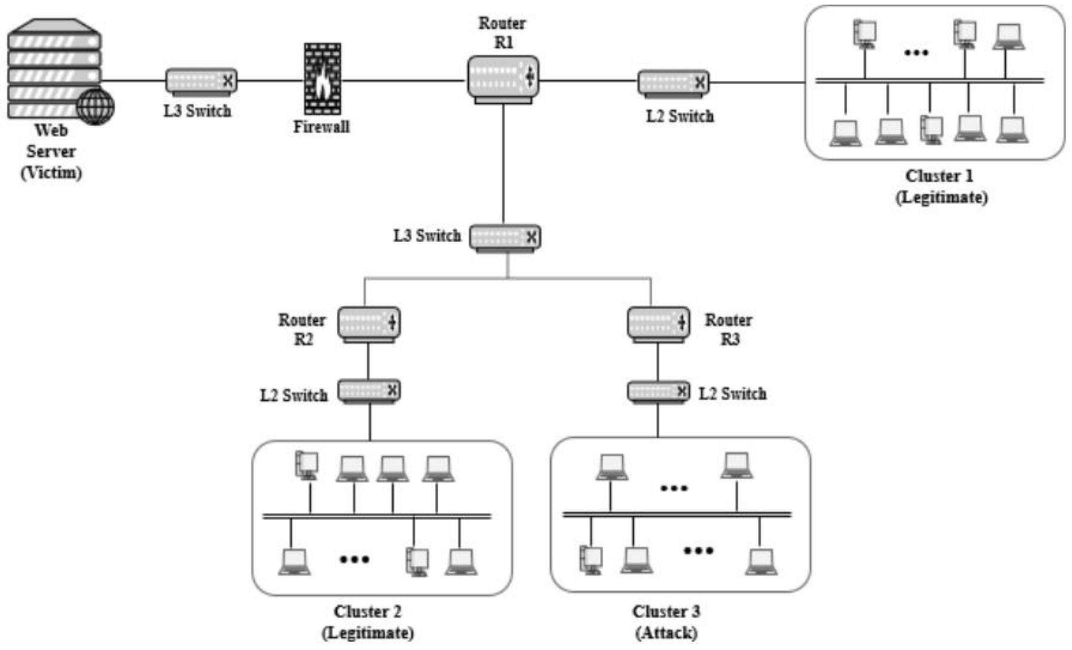


Fig. 1. Network Topology for Experimentation.

TABLE II. TOPOLOGY PARAMETERS

Parameter	**Value**
Number of legitimate nodes	2 clusters times 15 nodes times 30 VMs = 900
Number of attack nodes	1 cluster times 15 nodes times 30 VMs = 450
Number of routers	3 (R1, R2, R3)
Number of switches	5 (2 L3 Switch, 3 L2 Switch)
Bandwidth from R1 to web Server (Bottleneck Link)	100 Mbps
Delay of a link from R1 to a web server	5 ms

B. *Generating Traffic Traces*

The network traffic is generally a blend of two protocols: Transmission Control Protocol (TCP) and User Datagram Protocol (UDP); working at layer 4 of the OSI model. HTTP is an underlying generic protocol used by the World Wide Web to transfer data to and from the client and server. It operates in the application layer and relies on TCP in the transport layer for establishing a connection between the client and server. For the successful connection, TCP requires a valid IP-address. Each command is executed independently without the knowledge of the commands before and after it. GET and POST are two types of HTTP requests used for applications for which transmission time is not very critical but at the same time need reliability. These commands are not dependent on the commands that precede them or follow them. UDP, on the other hand, is a connectionless protocol as it does not require any virtual connection to be established before any data transfer. It is used for fast, efficient transmission like games and VoIP. It enables process-to-process communication by sending messages, called datagrams. It is efficiently used for applications that are loss tolerating and require low latency.

In the absence of real-time datasets, the traffic is generated synthetically to evaluate the system [6,19]. To achieve realistic trace, the client machines (nodes) interact with applications on a victim server with a non-spoofed and broad spectrum of source IP-addresses. Various open-source traffic generators are considered and studied to obtain a manageable mix of normal traffic and attack traffic for the experiment performed. Finally, Apache JMeter and D-ITG were chosen for generating background traffic and attack traffic.

Apache JMeter [1] is a pure Java open-source software designed to measure the performance of web services. It is a multi-platform Java desktop application. It has a friendly and easy to use Graphical User Interface (GUI). The results can be visualized as a log file and using a chart, table, or tree. Multiple virtual users can be created to generate heavy load against the victim server. JMeter supports all basic protocols, such as HTTP and FTP. Therefore, JMeter has been the choice of researchers of late [20, 22]. For the experiment described in this paper, the same version of JMeter is loaded on all the systems in cluster 1 and cluster 2. The distributed testing in JMeter requires one master, number of slaves, and one target machine, as shown in Fig. 2. One of the nodes in cluster 1 is configured as a master. The GUI runs on master and controls the rest of the nodes in cluster 1 and 2. Slaves run JMeter-server and take commands from the GUI and send requests to the target system. In short, each cluster has 15 nodes, and 30 virtual machines (VMs are defined as users in JMeter) are added hierarchically in the topology. Thus, the number of users generating traffic is scaled up to 900. The ramp-up time is set to 10 seconds so that each VM sends 3 requests per second. Similarly, the master-slave model is configured in cluster 3 with JMeter to generate HTTP traffic as attack traffic. Users in each cluster are assigned IP-addresses from different pools to distinguish between legitimate and illegitimate users [23].

[1] https://jmeter.apache.org/

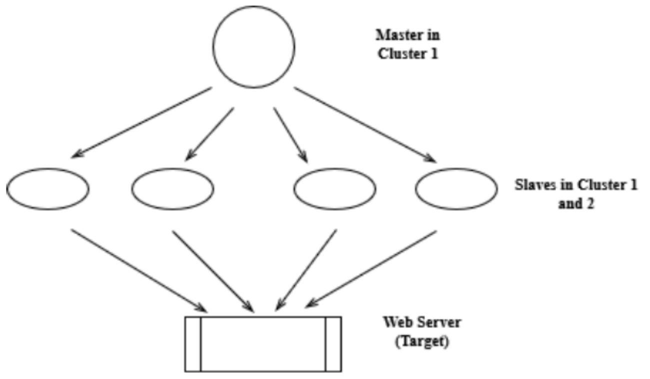


Fig. 2. Master- Slave Model.

Distributed Internet Traffic Generator (D-ITG) is a software platform capable of producing traffic that accurately adheres to patterns defined by the inter-departure time (IDT) between packets and packet size stochastic processes [24]. It supports both Linux and windows-based OS. It generates traffic having layer 3, layer 4, as well as layer 7 features. It emulates the sources of protocols like TCP, UDP, ICMP, DNS, Telnet, and VoIP [6, 24, 25]. The header fields like packet size, source, and destination IP-address, source, and destination port numbers can be customized as per the requirements. Due to its wide-ranging features, it has been widely used in research to generate synthetic attack traffic [8, 25] and hence, is the choice for generating attack traffic for the experiment mentioned in this paper. Cluster 3, as shown in Fig. 1, contains 15 nodes and 30 VMs on each node. Thus, attack traffic is generated from 450 users in cluster 3.

V. Results and Discussions

The performance of the victim server has been analyzed under two experimental set-ups. In the first set-up, the UDP traffic (representing DDoS attack) is generated as attack traffic in cluster 3 using the D-ITG tool and mixed with the normal traffic from 60th second to 120th second of the experiment targeting the victim server. Traffic is studied with IDT of 50 pps, 100 pps, and 200 pps.

In the second set-up, the HTTP traffic (representing FE) is generated in cluster 3 using Apache JMeter from 60th second to 120th second of the experiment. The attack traffic generated is mixed uniformly with the legitimate traffic generated by cluster 1 and 2 and targeted towards the victim server. Performance is observed for three instances- when attack traffic is 10%(approx.), 30%(approx.) and 75%(approx.) of total traffic. Parameters of experimental set-ups 1 and 2 have been outlined in Table III. The performance metrics used for analyzing the impact of an attack and FE are discussed:

A. Throughput

It is the amount of data transferred (in packets, in bits or bytes) in the unit time interval. It gives insight to the congestion of the network. Goodput is computed as the number of bits per second of legitimate traffic received at the server. Higher the goodput, the higher is the efficiency of the system being tested. These can be expressed as:

Throughput=Total traffic reaching server /Time-interval

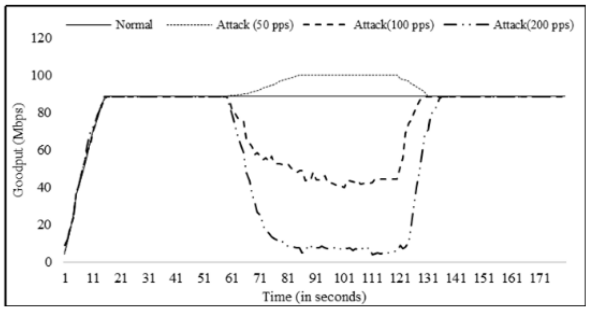
Goodput=Legitimate traffic reaching server/ Time interval

Fig. 3(a) and Fig. 3(b) show that for normal traffic, goodput increases slowly in the beginning until it either reaches the maximum traffic or the bandwidth limit before sit stabilizes. The slow start can be attributed to the congestion control strategy used by TCP. The transmission rate is increased by the slow-start algorithm until either a loss is detected or the receiver server's bottleneck link bandwidth is reached. In case loss occurs, the algorithm assumes that the network is congested and takes measures to reduce load. Otherwise, goodput increases exponentially until it reaches the bandwidth limit.

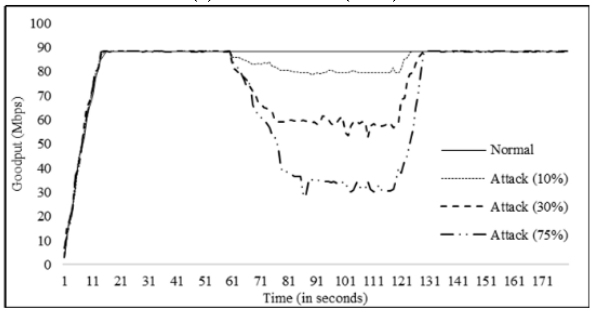
The traffic through the bottleneck link increases suddenly during the 60th second. As the attack strength increases, the value of goodput decreases. This is because as TCP senses packet loss, it decreases the transmission rate and hence decreasing the goodput. Hence, it is concluded that as soon as the traffic increases, whether UDP or HTTP, the attack traffic reaching the server increases. This rise plummets the goodput of the server to almost 7 Mbps.

B. Average Response Time

It is defined as the time between the request being sent from the client and receiving the first response [4]. It is also known as average latency [3]. This attribute is directly proportional to the amount of congestion in the network—lower the value of response time, less the congestion, and vice versa. In [15], the authors suggest that in the case of HTTP transactions, it is essential that the response time is less than 10 seconds to involve the users in the service and make the transaction successful. If the request crosses that limit, it is considered to be failed.



(a) Attack Traffic (UDP).



(b) Flash Event (HTTP).

Fig. 3. Goodput.

TABLE III. PARAMETERS OF EXPERIMENTAL SETUP-1 AND SETUP-2

Parameter	**Tool**	**Value**
Background/Legitimate traffic	JMeter	2 clusters x15 nodes x30 VMs = 900
Experiment time		180 seconds
Attack duration		60 seconds (from 60th sec to 120th sec)
Legitimate requests generated		Cluster 1: 15 x 30 x 3 = 1350 requests/sec Cluster 2: 15 x 30 x 3 = 1350 requests/sec Total = 2700 requests/sec
Packet Size		4096 bytes
Set-up 1:		
Attack traffic	D-ITG	UDP
Attack type		Constant rate
Packet size		512 bytes
Attack generated		Cluster 3: At 50pps per user Attack traffic = 50 x 450 = 22500 pps = 92.16 Mbps At 100 pps per user Attack traffic = 100 x 450 = 45000 pps = 184.32 Mbps At 200 pps per user Attack traffic = 200 x 450 = 90000 pps = 368.64 Mbps
Set-up 2:		
FE traffic	JMeter	HTTP
Traffic type		Uniformly distributed
Packet Size		500 bytes- 1000 bytes
Traffic generated		Cluster 3: For 10% FE traffic: VMs =10, 2 request/sec for each VM Total requests/sec = 15 x10 x 2 = 300 10% (approx.) of total traffic reaching server For 30% attack traffic: VMs =30, 6 request/sec for each VM Total requests/sec = 15 x 30 x 6 = 1200 30% (approx.) of total traffic reaching server For 75% attack traffic: VMs =50, 10 request/sec for each VM Total requests/sec = 15 x 50 x10 = 7500 75% (approx.) of total traffic reaching server

Average response time can be computed as

$$Response\ time = \frac{\sum(Tc + Td + Ts)}{N}$$

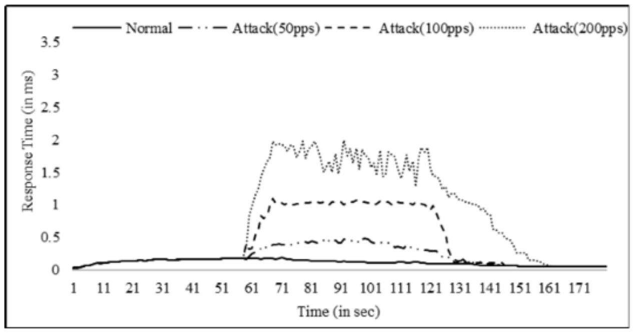
Where T_c is time for request sent from client to server, T_s is time for response sent from server to client, T_d is the time taken by the server to process the request, and N the number of time-intervals. Fig. 4(a) and Fig. 4(b) show an increase in response time with an increase in strength of attack in case of UDP traffic and HTTP traffic, respectively. HTTP traffic has a higher response time as compared to UDP traffic. The reason for the same is that in UDP, a connection is not formed between user and server, a datagram is just sent. Thus, UDP is faster than TCP, where the next packet is sent only once the acknowledgment is received for the previous one leading to the wait time and hence, an increase in the response time.

C. *Number of Legitimate Requests Dropped*

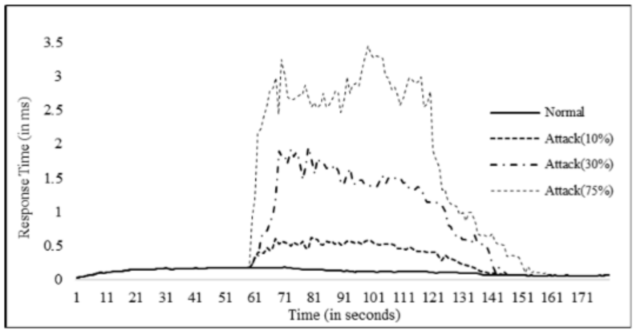
The number of requests dropped, due to an attack, measures the amount of congestion in the bottleneck link. For experiment 1, as UDP traffic increases, the congestion of the bottleneck link leads to the dropping of a large number of requests to the server. Fig. 5 shows the scenario of experiment setup-1, where the number of legitimate requests dropped increases with an increase in the attack strength.

D. *Number of Legitimate Active Connections*

Clients who have successfully connected themselves to the server and started sending the data are considered active connections. In the case of TCP connections, the active connections complete the 3-way handshake. When the attack packets of type HTTP surge the traffic, several packets endure time-out and hence reduce the window-size to almost one. According to the slow-start algorithm, the network reduces the load on the server by dropping the requests. Thus, the number of active connections decreases with an increase in the traffic beyond the capacity of the bottleneck link. The number of connections can reach as low as 90%, as suggested in Fig. 6. It can be observed that a large number of legitimate clients are denied services as the strength of attack increases. Thus, with a decrease in the number of active connections, the percentage of failed transactions increases.



(a) Attack Traffic (UDP).



(b) Flash Event (HTTP).

Fig. 4. Response Time.

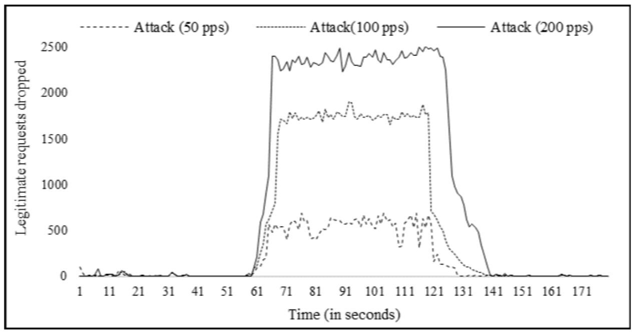


Fig. 5. Number of Legitimate Requests Dropped.

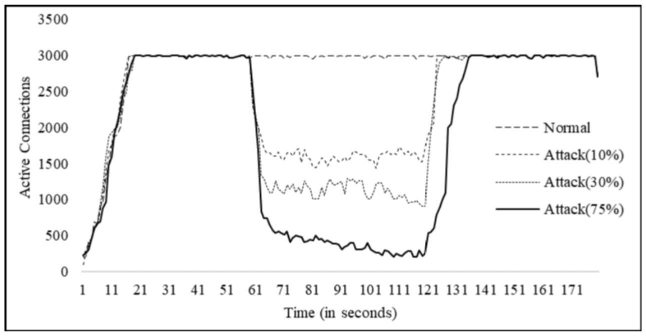


Fig. 6. Number of Legitimate Active Connection.

E. Percentage of Failed Transactions

It gives an insight to a number of requests timed-out or not being served by the server for whatever reason. In case the user sends requests using HTTP, the transaction is complete only if the response is received within the defined time (3-way handshake). When traffic increases during the 60th second, the large number of transactions fail due to congested bottleneck link. This decreases the throughput and also increases the response time, increasing the percentage of transactions that fail. It is directly proportional to the attack strength and can be represented in the equation as

$$\%age\ of\ failed\ transactions = \frac{T_{sent} - R_{recvd}}{T_{sent}} \times 100$$

where R_{recvd} is the number of responses received, T_{sent} is the total transactions sent.

F. Average Serve Rate/ Average Request Rate

It is the ratio of the rate at which the server serves the requests to rate at which the request is generated. The value of 1 indicates that all the requests generated are being served. As the strength of the attack increases, the ratio decreases. Higher the attack strength, the lesser the ratio. Fig. 7(a) and Fig. 7(b) show the pattern the ratio follows when under load in case of UDP traffic and HTTP traffic, respectively.

G. Percentage of Link Utilization

It is the percentage of bandwidth link used by legitimate requests. It can be computed as

$$\%age\ of\ link\ utilization = \frac{BW_{used}}{BW_{Total}} \times 100$$

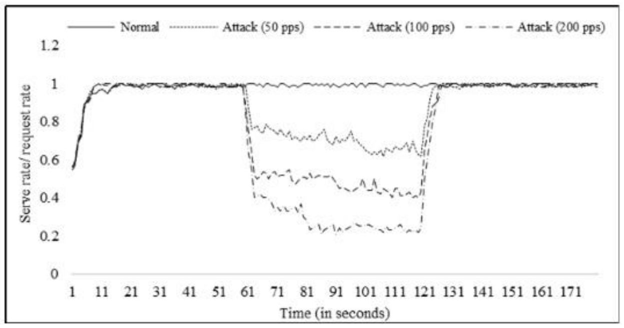
where *BW* is the link bandwidth. Fig. 8(a) and Fig. 8(b) show the link utilization (LU) in the case of UDP traffic and HTTP traffic, respectively. LU is 100% in case of normal conditions, but as soon as the traffic increases, LU reduces as a lesser number of legitimate requests reach the server. Legitimate traffic follows a congestion control protocol, so when the bandwidth gets clogged, the legitimate packets are dropped to decongest the bandwidth in the case of FE (HTTP traffic).

H. Legitimate Packet Drop Probability

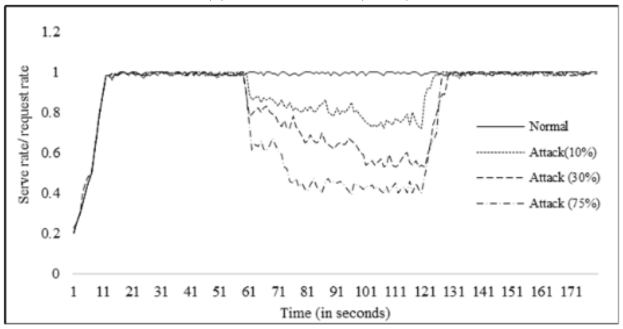
A packet-level metric that compares the number of legitimate packets dropped with the total number of legitimate packets in the network. During normal traffic conditions, many dropped legitimate packets is negligible, making the ratio value to be zero. With an increase in traffic, the number of dropped legitimate packets increases, thus, increasing the ratio. Fig. 9(a) and Fig. 9(b) show the response of the server for packet drop in the case of DDoS and FE, respectively. As can be seen, the performance of the server degrades with an increase in traffic, whether UDP or HTTP.

I. CPU Utilization

It is the server level metric that quantifies the utilization of CPU of the victim server. As the traffic increases in case of an anomaly, the total utilization of CPU increases though variations are seen for different applications. In the case of UDP attack traffic as in experiment 1, CPU utilization increases with an increase in strength of the attack, as shown in Fig. 10(a). If the attack is layer 7 attack (HTTP), the level of CPU utilization is more as compared to scenario-1, as shown in Fig. 10(b). This is because the request made to some running application has to be processed by a victim server. This leads to higher CPU utilization as compared to the UDP attack, where the requests are just dropped.

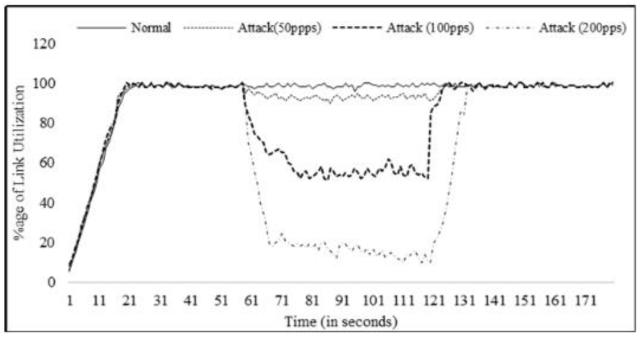


(a) Attack Traffic (UDP).

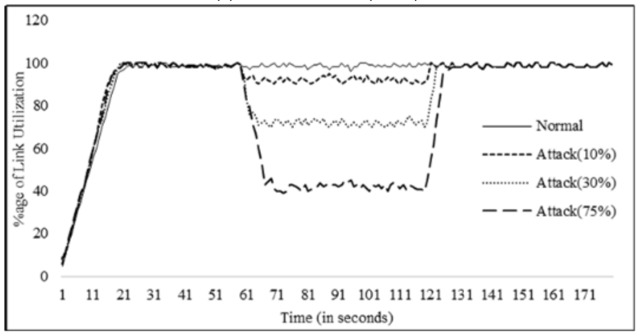


(b) Flash Event (HTTP).

Fig. 7. Average Serve Rate/ Average Request Rate.

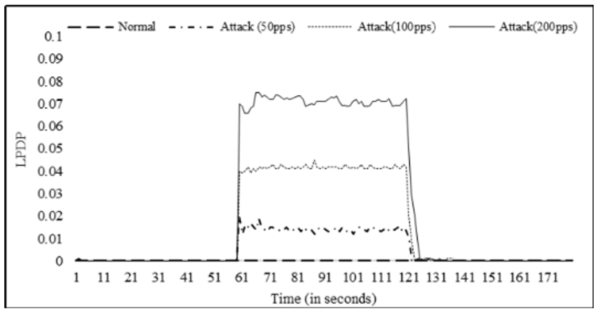


(a) Attack Traffic (UDP).

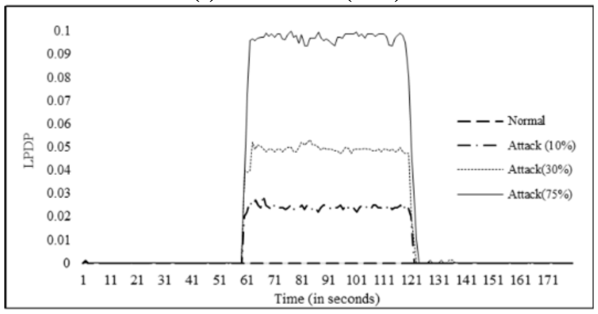


(b) Flash Event (HTTP).

Fig. 8. Link Utilization.

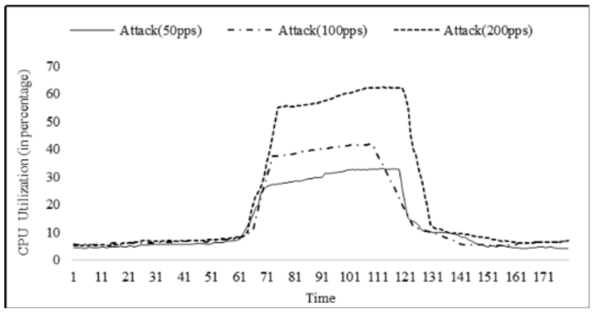


(a) Attack Traffic (UDP).

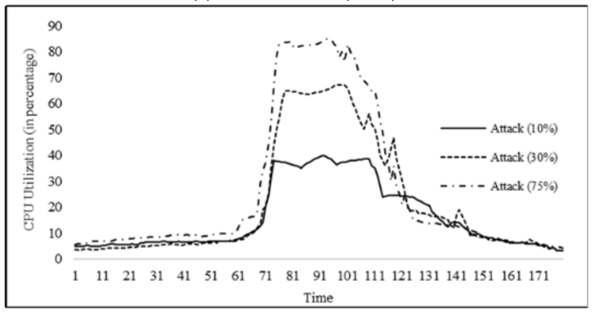


(b) Flash Event (HTTP).

Fig. 9. Legitimate Packet Drop Probability.



(a) Attack Traffic (UDP).



(b) Flash Event (HTTP).

Fig. 10. CPU Utilization.

VI. CONCLUSIONS

The paper attempts to define and evaluate the performance metrics for the network, assuming that the traffic consists mainly of TCP, HTTP, and UDP protocols. The impact metrics have been quantified using throughput, response time, number of active connections, percentage of failed transactions, percentage of link utilization, serve rate/request rate, and legitimate packet drop probability. The experiment was done under two set-ups built within a hybrid testbed. The first set-up creates the scenario of a DDoS attack, and the other creates the

FE effect. Each traffic is generated with varying strengths and has been analyzed for the system with a realistic topology using the testbed. The analysis of the impact indicated that the performance of the system degraded with the surge in traffic due to anomalies. As the number of requests increased, the link bandwidth choked, CPU utilization increased, the number of legitimate active connections decreased, legitimate requests reaching the server decreased, thus, increasing the response time. An increase in response time degraded the services. However, it is worth mentioning that the performance is also affected by the hardware (server speed, HDD, the available RAM) used at the server. The response time gets affected by the increase in throughput as well as the number of intermediate points through which the user and server are connected. The user's experience is affected by a change in response time, especially the commercial websites where performance degrades with an increase in response time.

VII. Future Scope

Each network environment uses different metrics for evaluating performance. Also, the metric chosen reveals the users' scenario when service is denied due to the attack. Such a metric compares the values with the baseline values defined during normal conditions of a particular network. There is a need to form a shared repository of the metrics being used for various applications and to generalize the performance measures. Future work is focused on explicating these metrics with the baseline model and also defining the detection metrics.

Acknowledgment

The authors would like to acknowledge I. K. Gujral Punjab Technical University, Kapurthala, India, for the great support and assistance rendered to carry out this research work.

References

[1] S. Newman, "DDoS attack on Wikipedia site," Available at: https://www.corero.com/blog/934-ddos-attack-on-wikipedia-site-smacks-of-hacktivism.html.

[2] O. Kupreev, E. Badovskya, and A. Gutnikov, "DDoS attacks in Q1 2020," Available online: https://securelist.com/ddos-attacks-in-q1-2020/96837/.

[3] K. Singh, K. K. Saluja, and P. Singh, "Impact analysis of application layer DDoS attacks on web services: a simulation study," Intl. J. of Intl. Engg. Informatics, vol. 5, no. 1, 2017, pp. 80-100, https://doi.org/10.1504/IJIEI.2017.10003432.

[4] M. Sachdeva, K. Kumar, G. Singh, and K. Singh, "Performance analysis of web service under DDoS attacks," Proc. IEEE Intl. Advance Comput. Conf., IEEE, March 2009, pp. 1002–1007, https://doi.org/10.1109/IADCC.2009.4809152

[5] M. Kumar, and A. Bhandari, "Performance evaluation of web server's request queue against AL-DDoS attacks in NS-2." International Journal of Information Security and Privacy, vol. 11, no.4, 2017, pp. 29-46. https://doi.org/10.4018/IJISP.2017100103.

[6] S. Behal, and K. Kumar, "Measuring the impact of DDoS attacks on web services - A Real-time experimentation," Intl. J. of Comp. Sci. Info. Security (IJCSIS), vol. 14, no. 9, 2016.

[7] T. Dubendorfer, A. Wagner, and B. Plattner, "An economic damage model for large-scale Internet attacks," Proc. 13th IEEE Intl. Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, IEEE Comput. Soc, 2004, pp. 223–228, https://doi.org/10.1109/ENABL.2004.11

[8] R. Vasudevan, Z. M. Mao, O. Spatscheck, and J. Van der Merwe, "MIDAS: An impact scale for DDoS attacks," Proc. 15th IEEE Workshop on Local and Metropolitan Area Networks, Princeton, NJ, 2007, pp. 200-205. https://doi.org/10.1109/LANMAN.2007.4295999

[9] R. S. R. Gade, H. Vellalacheruvu, and S. Kumar, "Performance of windows XP, windows vista and Apple's leopard computers under a Denial of Service Attack," Proc. Fourth International Conference on Digital Society, IEEE, February 2010, pp. 188–191, https://doi.org/10.1109/ICDS.2010.39

[10] R. Chertov, S. Fahmy, and N. B. Shroff, "Emulation versus simulation: A case study TCP-targeted denial of service attacks," Proc. 2nd International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities, 2006, TRIDENTCOM 2006, pp. 316–325, https://doi.org/10.1109/TRIDNT.2006.1649164.

[11] J. Mirkovic, A. Hussain, S. Fahmy, P. Reiher, and R. K. Thomas, "Accurately measuring the denial of service in simulation and testbed experiments," IEEE Trans. on Dependable and Secure Comput. vol. 6, no. 2, 2009, pp. 81-95. https://doi.org/10.1109/TDSC.2008.73.

[12] J. Mirkovic, A. Hussain, B. Wilson, S. Fahmy, P. Reiher et al, "Towards user-centric metrics for denial-of-service measurement," Proc. 2007 workshop on experimental computer science, San Diego, California, 2007. https://doi.org/10.1145/1281700.1281708.

[13] J. Mirkovic, S. Fahmy, P. Reiher, and R. K. Thomas, "How to test DoS defenses," Proc. 2009 Cybersecurity Applications and Technology Conference for Homeland Security, March 2009, pp. 103–117. https://doi.org/10.1109/CATCH.2009.23.

[14] J. Mirkovic, S. Wei, A. Hussain, B. Wilson, R. Thomas et al, "DDoS benchmarks and experimenter's workbench for the DETER Testbed," Proc. 3rd International Conference on Testbeds and Research Infrastructure for the Development of Networks and Communities, Lake Buena Vista, FL, 2007, pp.1-7, https://doi.org/10.1109/TRIDENT COM.2007.4444680.

[15] M. Sachdeva, G. Singh, and K. Kumar, "An emulation based impact analysis of DDoS attacks on web services during flash events," Proc. 2nd International Conference on Computer and Communication Technology (ICCCT-2011), pp 479–484, 2011. https://doi.org/10. 1109/ICCCT.2011.6075134.

[16] S. Bhatia, "Ensemble-based model for DDoS attack detection and flash event separation," Proc. Future Technologies Conference 2016, San Francisco, US, 2016, pp. 958-967, https://doi.org/10.1109/FTC.2016. 7821720.

[17] A. Bhandari, A.L. Sangal, and K. Kumar, "Performance metrics for defense framework against distributed denial of service attacks," Intl. J. of Netw. Security, vol. 6, April 2014.

[18] C. Bannwart, "Predicting the impact of denial of service attacks," Master thesis submitted to ETH, Zurich, Semantics scholar, 2012.

[19] S. Bhatia, D. Schmidt, G. Mohay, and A. Tickle, "A framework for generating realistic traffic for Distributed Denial-of-Service attacks and flash events," Comp. and Sec. vol. 40, 2014, pp. 95-107. https://doi.org/10.1016/j.cose.2013.11.005.

[20] K. Singh, P. Singh, and K. Kumar, "User behavior analytics-based classification of application layer HTTP-GET flood attacks. J. Netw. Comput. Appl. vol. 112, C (June 2018), 97–114. https://doi.org/10.1016/j.jnca.2018.03.030K.

[21] J.T.J. Midgley, "The linux HTTP benchmarking HOWTO," Available online: http://www.xenoclast.org/doc/benchmark/ HTTPbenchmarking-HOWTO/node3.html, 2001.

[22] M. A. Saleh, and A. A. Manaf, "A novel protective framework for defeating HTTP-based denial of service and distributed denial of service attacks," Sci. World Journal, 2015, https://doi.org/10.1155/2015/238230.

[23] Cisco Prime Network User Guide, 4.2.3, https://www.cisco.com/c/en/us/td/docs/net_mgmt/prime/network/4-2-/user/guide/CiscoPrimeNetwork423UserGuide/ip-pool.pdf.

[24] A. Avallone, S. Guadagno, D. Emma, A. Pescape, and G. Ventre, "D-ITG- distributed internet traffic generator," Proc. First Intl. Conf. on the Quantitative Evaluation of Systems (QEST '04), IEEE, Computer Society, 2004. https://doi.org/10.1109/QEST.2004.1348045.

[25] A. Botta, A. Dainotti, and A. Pescapè, "A tool for the generation of realistic network workload for emerging networking scenarios," Comp. Netw. vol. 56, no. 15, 2012, pp.3531-3547. https://doi.org/10.1016 /j.comnet.2012.02.019.