Measuring Denial Of Service

Jelena Mirkovic
University of Delaware
sunshine@cis.udel.edu

Peter Reiher
UCLA
reiher@cs.ucla.edu

Sonia Fahmy
Purdue University
fahmy@cs.purdue.edu

Roshan Thomas
SPARTA, Inc.
Roshan.Thomas@sparta.com

Alefiya Hussain
SPARTA, Inc.
Alefiya.Hussain@sparta.com

Stephen Schwab
SPARTA, Inc.
Stephen.Schwab@sparta.com

Calvin Ko
SPARTA, Inc.
Calvin.Ko@sparta.com

ABSTRACT

Denial-of-service (DoS) attacks significantly degrade service quality experienced by legitimate users, by introducing large delays, excessive losses, and service interruptions. The main goal of DoS defenses is to neutralize this effect, and to quickly and fully restore quality of various services to levels acceptable by the users. To objectively evaluate a variety of proposed defenses we must be able to precisely measure damage created by an attack, i.e., the denial of service itself, in controlled testbed experiments. Current evaluation methodologies measure DoS damage superficially and partially by measuring a single traffic parameter, such as duration, loss or throughput, and showing divergence during the attack from the baseline case. These measures do not consider quality-of-service requirements of different applications and how they map into specific thresholds for various traffic parameters. They thus fail to measure the service quality experienced by the end users.

We propose a series of DoS impact metrics that are derived from traffic traces gathered at the source and the destination networks. We segment a trace into higher-level user tasks, called *transactions*, that require a certain service quality to satisfy users' expectations. Each transaction is classified into one of several proposed application categories, and we define quality-of-service (QoS) requirements for each category via thresholds imposed on several traffic parameters. We measure DoS impact as a percentage of transactions that have not met their QoS requirements and aggregate this measure into several metrics that expose the level of service denial. We evaluate the proposed metrics on a series of experiments with a wide range of background traffic and our results show that our metrics capture the DoS impact more precisely then partial measures used in the past.

Categories and Subject Descriptors: C.4 Performance of systems: Measurement techniques

General Terms: Measurement, security, standardization.

Keywords: Denial of service, metrics.

1. INTRODUCTION

Network communication requires proper functioning of many diverse and distributed network elements: applications, protocols, operating systems, end hosts' resources, routers, network links, and critical Internet services, such as Domain Name Service. Denial-of-service (DoS) attacks can take down any of these targets, either by exploiting some vulnerability or by overwhelming a critical resource, to deny service to legitimate users. Because the Internet's popularity grows daily, DoS attacks cause great disturbance and have drawn a lot of attention from security researchers who are working to design effective defenses.

Accurately measuring denial-of-service impact is essential for evaluation of potential DoS defenses. A defense is only valuable if it provably prevents or eliminates denial of service, making DoS attacks transparent to Internet users. If we could measure which services were denied by an attack with and without the defense we could: (1) understand and express severity of various attacks, (2) characterize the effectiveness of proposed defenses, and (3) compare defenses to understand their price/performance tradeoff.

DoS attacks deny service to legitimate users because they either deplete some scarce resource needed by legitimate traffic, or because they exploit a vulnerability at the server, which slows down or disables processing of user requests. A user perceives large request-response delay in interactive traffic, low-quality audio and image due to packet loss in media and gaming traffic, and large duration of non-interactive transactions such as email transfer. When evaluating the effect of a DoS attack in a testbed or in a simulation, it is infeasible to ask human users to assess service quality, since this will not provide consistent and accurate quantitative measures of DoS impact. We thus need to define a comprehensive DoS impact metric that maps all user-perceived service quality into network traffic parameters such as packet loss, delay, etc., that can be measured objectively and in an automated fashion. Historically, several measures have been used by researchers: percentage of legitimate packets dropped, division of resources between legitimate and attack traffic, throughput or goodput of TCP connections, request/response delay and the overall transaction duration. While many of these parameters will have a different distribution during the attack, when compared with the baseline case, there is a lack of understanding how the parameter values map into user-perceived service quality. For example, a 5-minute one-way delay can be detrimental to interactive audio conversations (e.g., VoIP) [18] and an unnoticeable glitch for email transfer between servers [16]. If a defense system reduces one-way delay to 1 second, this restores service quality for Web users, who can tolerate up to 10 second request/response delays [6], but does not help game users that need less than 150 ms delays for a good service [3].

Evidently, an accurate DoS metric must consider application-specific quality-of-service (QoS) requirements and compare measured traffic parameters with application-specific thresholds to map them into a user-perceived service quality. We propose such a metric in this paper, that speaks to the heart of the problem: did the legitimate clients receive acceptable service or not, for each task they performed during an attack. There are two possible approaches to compute application performance metrics: we could instrument each application to compute statistics such as average response time, transaction completion time, loss, etc. or we can use real, uninstrumented application programs, and then process experimental traffic traces to identify transactions, measure required parameters and compute performance metrics in a completely automated fashion. We select the second approach, since it scales better to different and new application types.

We process a traffic trace captured during an experiment to identify *transactions*, that represent higher-level tasks whose completion is meaningful to a user, such as: browsing one Web page, downloading a file or having a VoIP conversation. For each transaction, we measure five parameters: (1) one-way delay, (2) request/response delay, (3) packet loss, (4) overall transaction duration and (5) delay variation (jitter). Jointly, these parameters capture a variety of application QoS requirements as we will discuss in Section 2. A transaction is then classified into an application category, and measured parameters are compared with category-specific thresholds to determine if the transaction *succeeded* or *failed*. We calculate the percentage of failed transactions (*pft*) in each application category, as a measure of DoS impact. We aggregate this measure into several metrics to expose specifics of a DoS attack's interaction with the legitimate traffic.

The main challenge of the proposed DoS impact metric lies in categorizing applications by their QoS requirements, and specifying realistic, objective and measurable criteria for success (or failure) for each application category. Ideally, a DoS impact metric should match a legitimate user's experience during an attack so that transactions marked as failed are those that a user would find of poor quality, and similarly succeeded transactions are those that have an acceptable quality. This is very challenging, because research on QoS involving human subjects has noted that a user's subjective perception of acceptable or impaired service not only varies greatly between users but also varies for a given user depending on his intent and expectations [6]. An additional obstacle lies in the fact that only some subjective QoS perceptions have been paired with measurable parameter values [18, 7, 3], and even for these there is a large grey area of service quality that some users would label as acceptable while others would not. Further, some applications can hide a network-induced delay, delay variation or loss using variable buffering (streaming media applications [18]) or extrapolation (online games [9, 5]). For scalability reasons, we must decide to either consider all applications of a certain kind (e.g., streaming audio) or none of them, as capable of masking some specific range of delay, jitter or loss.

We acknowledge that definition of application categories and universally acceptable QoS criteria will be a major undertaking, and will require participation of a large research and commercial community. However difficult, we believe that this effort is necessary for objective evaluation of DoS defenses and their fair comparison. In Section 2.1 we propose a set of application categories and their QoS criteria. In this we largely borrow from 3GPP's specification of QoS requirements for Universal Mobile Telecommunications System, that defines acceptable service quality for various applications. 3GPP is a "collaboration agreement which brings together a number of telecommunications standards bodies" [1] from all over the world, in an effort to "produce globally applicable Technical Specifications ... for a 3rd Generation Mobile System" [1]. Thus, the proposed set of QoS specifications has an advantage of being already accepted by the world's large standards bodies. We significantly extend and formalize the specifications from [16], using findings from the recent QoS research, and with a goal to customize them for trace-based measurement.

The proposed DoS impact metric requires trace capture at the legitimate senders and at the traffic destinations. As such, it is suitable for testbed experimentation where we can capture traffic at any point, but it would not be applicable to detect attacks or measure DoS impact in real-world situations. Because our metric is trace-based it applies well to all testbed experiments. With minor extensions it could be also used for DoS measurement in simulations. We discuss how to measure the required traffic parameters in DoS experiments in Section 2.2, and how to aggregate these measures into various DoS impact metrics in Section 2.3. We illustrate our metrics with small-scale experiments in DETER testbed [4] in Section 3, survey related work in Section 4 and discuss future directions in Section 5.

2. DOS IMPACT METRIC

We propose a DoS impact metric that directly measures if a user's service was denied or not. This metric considers a set of high-level user tasks, called *transactions*, and categorizes them, based on the application that generated the traffic, into an application category. For each transaction, we measure the following five parameters: (1) one-way delay, (2) request/response delay, (3) packet loss, (4) overall transaction duration and (5) delay variation (jitter). Request/response delay captures QoS degradation of interactive applications, one-way delay, packet loss and jitter capture QoS degradation of media and game applications and transaction duration captures degradation of non-interactive tasks where a user can tolerate intermittent delay but expects a transaction to be completed in a certain time frame. Each transaction's parameters are compared to a set of thresholds specific to its application category, and the transaction is marked as successful, if it meets all QoS criteria, or failed otherwise. The main DoS impact measure we define is the percentage of failed transactions, *pft*, during some experimental interval.

2.1 Application QoS Criteria

Table 1 lists the application categories we propose, mostly borrowed from [16], and the corresponding QoS requirements. We note the following differences from [16]: (1) Because many media applications can sustain higher jitter than 1 ms [16] using variable-size buffers, we adopt the jitter threshold value of 50 ms as defined in [2]. (2) We differentiate between first-person shooter (FPS) and real time strategy (RTS) games, because research has shown that their QoS requirements differ. We use [3] (FPS) and [17] (RTS) as sources for specifying delay and loss bounds. (3) Research on human perception of Web traffic delay has shown that people can tolerate higher latencies for entire task completion if some data is served incrementally [6]. We specify two types of delay requirements for interactive transactions (email, Web, FTP) where a user can utilize a partial response: (a) *any delay* measured between receipt of any two data packets from the server. For the first data packet, any delay is measured from the end of a user's request, and (b) *whole delay* measured from the end of a user's request until the entire response has been received. We also specify an overall duration requirement for these transactions. We use 60 s [6] for Web-transaction duration threshold, and like [11], we assume that an acceptable duration for email and FTP should not exceed three times the expected duration, given the amount of data being transferred. (4) We add DNS and ICMP services, and specify 4 s whole delay requirement. This is the maximum tolerable delay for interactive tasks [16]. (5) We use 4 hours for expected email (server-to-server) and Usenet transaction duration, instead of *several hours* [16]. (6) We capture request/response delay at the transport level, considering all data packets going to a server, between two responses, as a request, and similarly, all data packets sent by the server between two requests as a reply. This measure will miss the delay that occurs if some request packets are dropped and retransmitted, but this delay is noticed by a user and must be included in success calculation. We capture this delay by calculating the maximum RTT for each TCP-based transaction, and use this as additional parameter for QoS calculation. (7) For TCP-based applications, we ignore loss bounds specified in [16] because losses will either be handled through TCP retransmissions or will lead to a high request/response delay or RTT that exceeds the specified threshold. For UDP and ICMP transactions we map packet loss into delay, by setting request/response delay to a large, fixed value (= 2 times the delay threshold) if the request has been lost. We keep loss bounds for audio, video and FPS games.

Category	One-way delay	Req/resp delay	Loss	Dur.	Jitter
email (srv/srv)		whole, RTT $<$ 4 h			
Usenet		whole, RTT $<$ 4 h			
Chat	$<$ 30 s				
Web		any, RTT $<$ 4 s		$<$ 60s	
FTP		any, RTT $<$ 10 s		$<$ 300%	
FPS games	$<$ 150 ms		$<$ 3 %		
RTS games	$<$ 500 ms				
Telnet		any, RTT $<$ 250 ms			
email (usr/srv)		any, RTT $<$ 4 s		$<$ 300%	
DNS		whole $<$ 4 s			
ICMP		whole $<$ 4 s			
	media	control	media		media
Audio, conv.	$<$ 150 ms	whole, RTT $<$ 4 s	$<$ 3%		$<$ 50 ms
Audio, messg.	$<$ 2 s	whole, RTT $<$ 4 s	$<$ 3%		$<$ 50 ms
Audio, stream	$<$ 10 s	whole, RTT $<$ 4 s	$<$ 1%		$<$ 50 ms
Videophone	$<$ 150 ms	whole, RTT $<$ 4 s	$<$ 3%		
Video, stream	$<$ 10 s	whole, RTT $<$ 4 s	$<$ 1%		

Table 1: Application categories and their QoS requirements

2.2 Measuring Traffic Parameters

We collect a traffic trace at each legitimate sender, and identify user-initiated conversations by looking for SYN packets sent from this machine for the TCP traffic, UDP packets sent to a well-known UDP service port for UDP traffic and ICMP request packets. We also categorize conversations into application categories from Table 1 using destination port numbers for the mapping. In these conversations, we identify transactions as smaller user/server exchanges that have some meaning to a user. We introduce a notion of a transaction to properly evaluate success or failure of lengthy conversations. For example, if a user opens an FTP connection to a server and uses it to transfer 100 files over an hour, one by one, the failure of the last transfer does not indicate the failure of the entire FTP session but instead a failure rate of $1/100$. If we instrumented different applications we could precisely identify transactions that produce some meaningful output to a user. Instead, we opt for a trace-based approach and attempt to identify transactions through a traffic trace analysis. This approach is imperfect, but results in a more portable measurement strategy that can be applied to experiments that use off-the-shelf applications. Table 2 shows how we identify transactions in the trace data. A flow is defined as all traffic between two IP addresses and port numbers. For interactive applications, an inactive time (user think time) followed by a new user's request denotes a new transaction. A transaction is usually a part of or the whole flow, e.g., if a user opened a TCP connection to an FTP server, downloaded one file and closed the connection, this would be recognized as one transaction. Downloading 3 files in the same session, with think time in between, would be recognized as 3 transactions. In case of media traffic, both the media stream (UDP flow) and the control stream (TCP flow) are part of a single transaction.

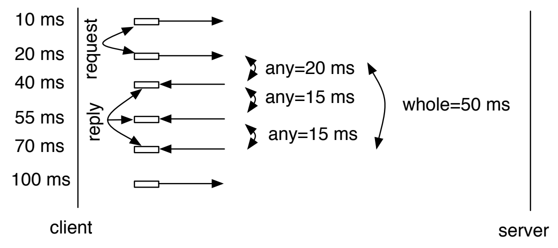


Figure 1: Illustration of request/response identification

Application	Transaction
email (srv/srv), Usenet	TCP flow
Chat, Web, FTP, Telnet, emal (usr/srv)	TCP flow and inactive time $>$ 4s
Games	UDP flow and inactive time $>$ 4s
DNS, ICMP	One request/response exchange
Audio and video	TCP flow (control channel) and a corresponding UDP flow (media traffic)

Table 2: Transaction identification

We measure request/response delay and transaction duration using sender-collected trace and we correlate sender/receiver traces to measure one-way delay, loss and jitter. The correlation is done by matching source IP, port and packet identification (if available) of the packets at the sender with the packets at the receiver side, and synchronizing sender and receiver clocks at the beginning of the experiment. We use tcpdump to capture a traffic trace during the experiment.

We identify requests and responses using the data flow between senders and receivers. Let A be a client that initiates some conversation with a server B. A *request* is identified as all data packets sent from A to B, before any data packet from B. A *reply* is identified as all data packets sent from B to A, before any new request from A. Figure 1 illustrates request and reply identification, and measurement of any delay and whole delay values.

Email and Usenet applications have a delay bound of 4 hours and will retry a failed transaction for a limited number of times. It would be infeasible to run several-hour long experiments so we need to extrapolate transaction success for these applications using short experiment data. DoS impact usually stabilizes shortly after the onset of an attack or after the defense's activation, unless the attack or the defense exhibit time-varying behavior. We can thus use the *pft* value measured for transactions that originate after the stabilization point as a predictor of *pft* in a longer experiment. Let r be a total number of retries within 4 hours and let s be the stabilized *pft* for email (or Usenet) transactions during a short experiment. The predicted *pft* for a long experiment is then: $pft_p = s^r$.

Finally, FTP and email success criteria require comparing a transaction duration during an attack with its expected duration without the attack. Since transaction duration depends on the volume of data being transferred and network load, we cannot set an absolute duration threshold. If we have perfectly repeatable experiments, we could measure the expected duration directly, running the experiment without the attack. However, some traffic generators may have built-in randomness that prevents repeatable experiments. In this case we must estimate the expected transaction duration, using information about the throughput of transactions from the same application category, that complete prior to the attack. Let us observe a transaction T that has completed in t_r seconds, sending B bytes of data, and whose duration overlaps an attack. Let *Th* be the average throughput of transactions generated by the same application as transaction T, and completed prior to the attack's start. We calculate the expected duration for the transaction T as $t_e = B/Th$. If $t_r > 3 \cdot t_e$ (see Table 1) the transaction will be labeled as failed.

2.3 Aggregating Results

Because many DoS attacks inflict damage only while they are active, and the impact ceases when the attack is aborted, we suggest that only transactions that overlap the attack be used for *pft* calculation.

We define *DoS-hist* measure as the histogram of *pft* measures across application categories. DoS-hist measure is especially useful to capture the impact of attacks that target only one application, e.g., TCP SYN attack at Web server port 80.

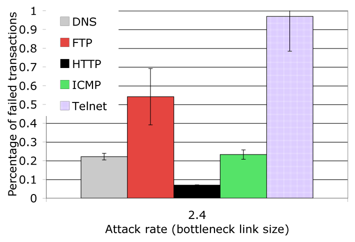
Sometimes it may be useful to aggregate *DoS-hist* measures into a single number, we call *DoS-level*, applying a different weight for each application: *DoS-level* $= \sum_k$ *pft(k)* $\cdot w_k$, where k goes over all application categories, and w_k is a weight associated with a category k. Note that *DoS-level* is highly dependent on the chosen set of application weights. Unless there is a broad consensus on the appropriate set of weights, using *DoS-level* for defense performance comparison could lead to false conclusions as weights can be chosen to drive the results in favor of any defense.

While the proposed transaction success measure is binary, and therefore easy to grasp, for a deeper analysis it is useful to measure a service quality in a continuous manner, e.g., to understand if a failed transaction's delay was just above the threshold or several times bigger. We calculate *QoS-degrade* for each failed transaction by locating a parameter that exceeded its QoS threshold and calculating the ratio of their difference and the threshold. For example, if d is the measured delay that exceeds the threshold value t, *QoS-degrade*=$(d - t)/t$. If more than one parameter violates its threshold we choose the largest *QoS-degrade*. In experiments, we report the average of the *QoS-degrade* measures for transactions in the same application category. Intuitively, a value N of *QoS-degrade* means that the service of failed transactions was N times worse than a user could tolerate.

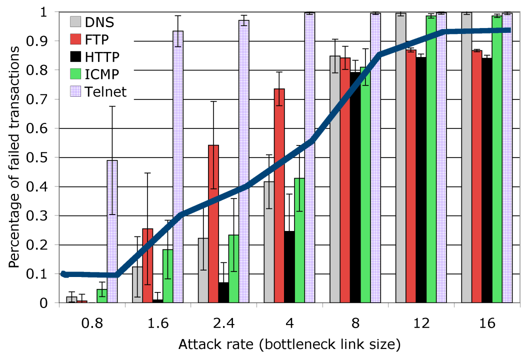
3. EXPERIMENTS

We now illustrate our proposed DoS impact metrics in small-scale experiments in DETER testbed [4]. We use a simple network topology, with a single legitimate client, an attacker and a server. The server, attacker and client are connected to a router and the server's link bandwidth is 10 times smaller than the other links, creating a bottleneck. DoS attacks usually engage numerous attack machines and target networks with many legitimate clients and a rich topology. While our tested scenario is much simpler than the real world attacks we deliberately chose such low-complexity setting so we could properly expose, attribute and explain specifics of our DoS metrics. Our continuing research focuses on more realistic attack scenarios.

We generate the following legitimate traffic between the client and the server: (1) HTTP and FTP traffic with file sizes distributed according to Pareto(100K, 2.5) and Pareto(100K, 1.92), and exponentially distributed connection arrivals with 1 s mean, (2) Telnet traffic with Pareto(50,1.18)-distributed duration and Pareto(10K,2.14)-distributed traffic volume, and exponentially distributed connection arrivals with 1 s mean, (4) DNS and ICMP traffic with exponentially distributed request arrivals, with 1 s and 5 s mean, respectively. We wrote a customized Telnet application that sends a small random number of characters per second and receives a similar-volume reply. We generated the rest of the traffic using real applications: scp for FTP, ping for ICMP, wget for HTTP and dig for DNS. The server machine is running Apache Web server, bind



(a) DoS-hist measure for attack volume of 2.4 times the bottleneck link



(b) DoS-hist measure vs attack strength

Figure 2: DoS-hist measure for UDP flood, and DoS-level (blue line) with equal weights

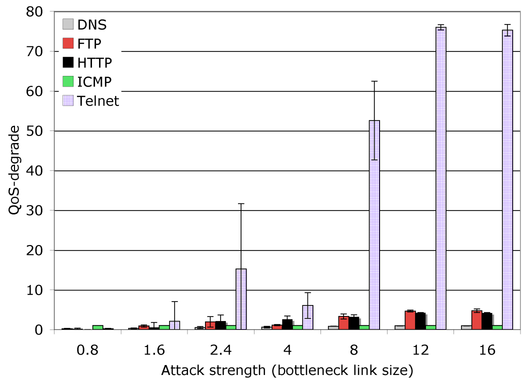
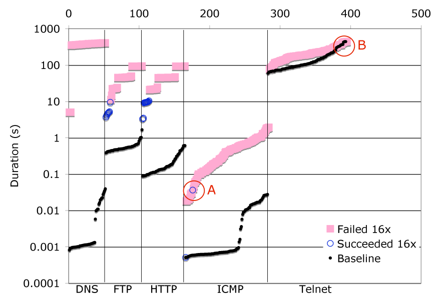


Figure 3: QoS-degrade measure for UDP flood

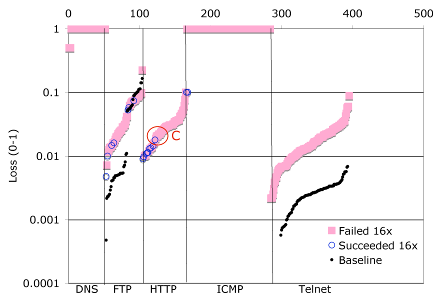
DNS server, Red Hat 9 operating system and has a 733MHz Pentium III CPU. All requests are sent from the client to the server, but half of FTP file transfers are uploads, and the other half are downloads. Legitimate traffic is started at the beginning of an experiment, and an attack is launched from 50 s to 110 s. The *pft* measure without an attack was zero for all application categories.

3.1 UDP flood

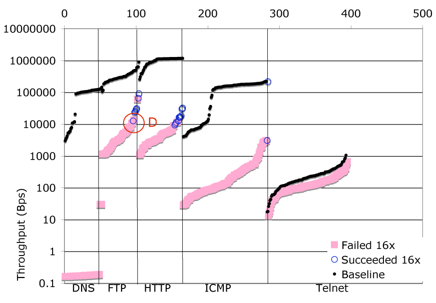
In the first experiment, we generate a UDP flood that overwhelms the bottleneck link. Figure 2(a) shows the *DoS-hist* measure for the attack whose volume is 2.4 times the bottleneck link size. To preserve space, we show *DoS-hist* measures for various attack strengths in the single Figure 2(b). The x-axis shows the attack strength in multiples of the bottleneck link size, the column height denotes the average of 20 test runs and the error bars denote the standard deviation. We also show the *DoS-level* measure using equal application weights with the blue line. Telnet application fails earlier than others because it has a very low delay threshold (250 ms) that is easily



(a) Duration distribution



(b) Loss distribution



(c) Throughput distribution

Figure 4: Duration, loss and throughput distribution for UDP flood

violated even when packet loss is small. FTP transactions also exhibit high failure rates, mostly because a lengthy data transfer increases a chance of multiple packet loss, which lowers the throughput and increases transaction duration beyond the specified threshold. DNS and ICMP transactions generate small single-packet requests, that have a good chance of winning against small-rate attacks. These transactions fail mostly because they do not retransmit lost packets a sufficient number of times (zero for ICMP, three for DNS). HTTP is most competitive, because it generates short requests, retransmits them vigorously upon loss, and has a generous (10 s) delay threshold. Large variability of *pft* measure for moderate attack rates can be attributed to a luck effect when packets compete for limited bandwidth. Because the attack rate is comparable to the legitimate traffic rate, in repeated experiments packet arrivals at the bottleneck queue occur in different order, resulting in a different probability of legitimate packet loss. At high attack rates the variability is reduced because the attack always wins. The worst *pft* of FTP and HTTP transactions is lower than 100% because transactions started near the attack's end, which make $10 - 15\%$ of all transactions in the given category, recover after attack ends, before violating their delay bounds.

Figure 3 shows the average *QoS-degrade* measure for the failed transactions in each application category. The measure grows with the attack's strength, and is most severe for Telnet transactions that experience more than 70 times the allowed delay.

To illustrate the inadequacy of single-parameter measures for DoS impact, used in many research papers, we show in Figures 4 (a)—(c) the distributions of transaction duration, loss and throughput for no-attack (baseline) case, and for failed and succeeded transactions in case of the highest attack rate. We show these distributions per application category. Note that y-axis is in log scale, so zero values will not be visible, e.g., baseline loss values for DNS, HTTP and ICMP are zero and do not show in the Figure 4 (b). In many cases the markers for failed and succeeded transactions or for baseline and failed transactions overlap, indicating that a given measure cannot capture the DoS impact. We have circled several such cases. In Figure 4 (a), point A emphasizes a successful ICMP transaction whose duration is larger than for many failed transactions, illustrating the fact that transaction duration alone is not a good predictor of QoS for ICMP transactions. Point B emphasizes that many telnet transactions in the baseline have longer duration than telnet transactions during the attack, that fail because the request/response delay threshold has been exceeded. In Figure 4 (b), point C emphasizes a successful HTTP transaction whose loss is larger than for many failed HTTP transactions, that exceed their request/response delay threshold. The baseline packet loss of FTP transactions overlaps the packet loss of failed transactions during the attack. Both these facts prove that packet loss alone is not a good predictor of QoS. In Figure 4 (c), point D emphasizes a successful FTP transaction whose throughput is lower than for some failed FTP transactions. The throughput of Telnet transactions during the attack is barely lower than the baseline throughput. Thus, throughput alone is not a good QoS predictor.

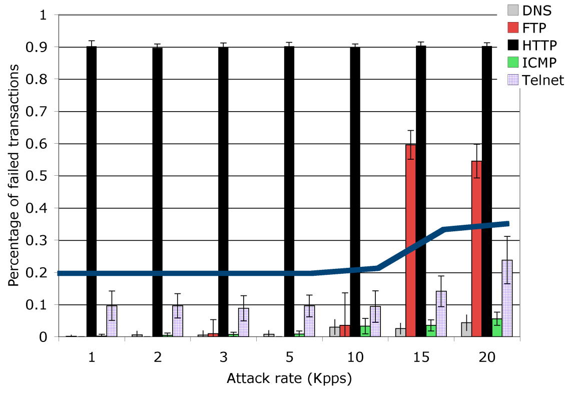
3.2 TCP SYN flood

In the next experiment we generate a SYN flood to the Web service port. This flood should deny service only to HTTP transactions that originate during the attack. We show the *DoS-hist* and *DoS-level* measures in Figure 5(a) vs the attack strength measured in thousands of packets per second. HTTP service is severely denied even by small attacks. The maximum *pft* measure is around 90%, which is the same as the ratio of HTTP transactions that start during the attack to all HTTP transactions that overlap the attack. In addition to this, Telnet service is denied even by small attacks, and DNS, FTP and ICMP degrade when stronger attacks are launched. These side effects occur because attack's packet rate interferes with end host's ability to process packets at a high rate, resulting in intermittent packet loss at the destination. FTP degrades severely when packet rates exceed 10Kpps. A closer inspection of the packet trace showed that this occurs because FTP server slows down considerably and takes a long time to generate replies, leading to violation of FTP's request/reply threshold. Since we are using SSH for FTP transfer (invoking `scp` command), the bottleneck occurs at CPU, when high packet rates cause too many interrupts that steal CPU cycles from SSH's cryptographic operations. To test this hypothesis we rerun our experiments using `vsftpd` at the server and the client machine, and running `wget` for file download instead of `scp`. We kept all other settings identical. Resulting *DoS-hist* and *DoS-level* measures are shown in Figure 5 (b). Since we have removed heavy cryptographic requirement on CPU, it could process high packet rates more efficiently which resulted in zero DoS impact on FTP, DNS and ICMP. The only seriously affected application was HTTP. Telnet's service was also slightly degraded because its low request/response threshold was violated even with low-rate attacks.

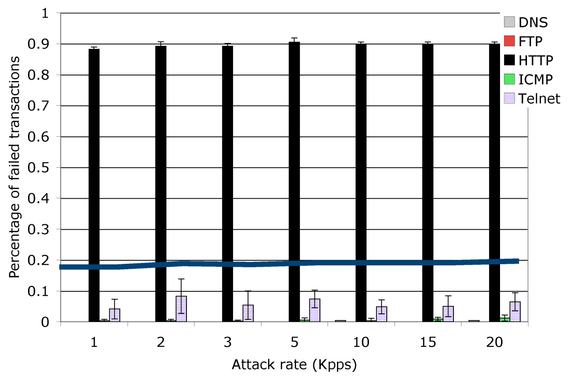
Figure 6 shows the *QoS-degrade* measure in experiments with SSH. The service of non-HTTP transactions was not severely degraded, exceeding the thresholds by at most 100%. Meanwhile, HTTP transactions exceed their threshold more than 4 times.

4. RELATED WORK

For brevity we only provide a short overview of the work related to DoS impact measurement. In the quality of service field there is an initiative to define a universally accepted set of QoS requirements for applications. This initiative is lead by the 3GPP partnership including large standards bodies from all over the world [1]. While many of the specified requirements apply to our work, we extend, modify and formalize these requirements as explained in Section 2.1.



(a) With SSH



(b) With vsftp

Figure 5: DoS-hist and DoS-level measures for TCP SYN flood

The differentiated services framework separates applications into several categories based on their sensitivity to delay, loss and jitter [14]. The Internet's integrated services framework maps these application types onto three service categories: the guaranteed service, the controlled load service and the currently available best effort service [8]. In the early 1990s, ATM networks provided different service categories for different applications and the traffic management specifications [13]. All mentioned research focuses on providing guaranteed service to applications by the network rather than measuring if the service was denied during a DoS attack.

In [10] researchers measure the impact of a DoS attack on network traffic. They study the distribution of several parameters: the throughput of FTP applications, roundtrip times of FTP and Web flows, and latency of Web flows and the DNS lookup service. Our work concerns specifying the acceptable-service thresholds for a broader variety of parameters and services.

Recently, IRTF's Transport Modeling Research Group (TMRG) has been chartered to standardize evaluation of transport protocols and develop a benchmark suite of tests [15]. TMRG's drafts discuss the possibility of using user-based QoS metrics for measuring congestion but do not specify such metrics in any detail.

5. CONCLUSIONS

Denial of service is a complex phenomenon that interacts with network's services, applications and end users in various ways. Precise and objective measurement of DoS impact is necessary for testbed DoS experimentation and for realistic evaluation of DoS defenses. We have proposed a multilevel QoS framework that can be used to precisely capture the DoS impact on various network applications by comparing several traffic parameters against application-specific thresholds. We use this framework to devise a series of comprehensive and intuitive DoS impact metrics. While much work remains to be done on refining the proposed metrics and applying them in different experimental settings, this paper is a major first step towards objective evaluation of DoS defenses.

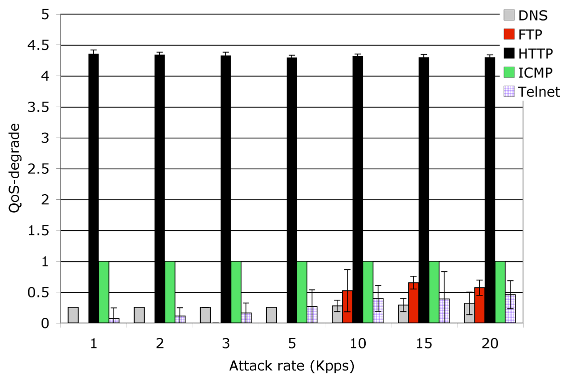


Figure 6: QoS-degrade measure for TCP SYN flood

6. REFERENCES

[1] 3GPP. *The 3rd Generation Partnership Project (3GPP)*.

[2] J. Ash, M. Dolly, C. Dvorak, A. Morton, P. Taraporte, and Y. E. Mghazli. Y.1541-QOSM – Y.1541 QoS Model for Networks Using Y.1541 QoS Classes. NSIS Working Group, Internet Draft, Work in progress, May 2006.

[3] T. Beigbeder, R. Coughlan, C. Lusher, J. Plunkett, E. Agu, and M. Claypool. The Effects of Loss and Latency on User Performance in Unreal Tournament 2003. In *Proceedings of ACM Network and System Support for Games Workshop (NetGames)*, September 2004.

[4] T. Benzel, R. Braden, D. Kim, C. Neuman, A. Joseph, K. Sklower, R. Ostrenga, and S. Schwab. Experiences with DETER: A Testbed for Security Research. In *2nd IEEE Conference on Testbeds and Research Infrastructure for the Development of Networks and Communities (TridentCom 2006)*, March 2006.

[5] A. R. Bharambe, V. N. Padmanabhan, and S. Seshan. Supporting Spectators in Online Multiplayer Games. In *Proceedings of ACM SIGCOMM Workshop on Hot Topic in Networks*, November 2004.

[6] N. Bhatti, A. Bouch, and A. Kuchinsky. Quality Is in the Eye of the Beholder: Meeting Users' Requirements for Internet Quality of Service. Technical Report HPL-2000-4, Hewlett Packard, 2000.

[7] A. Bouch and M. A. Sasse. Why Value is Everything: A User-Centered Approach to Internet Quality of Service and Pricing. In *Proceedings of International Workshop on Quality of Service*, June 2001.

[8] R. Braden, D. Clark, and S. Shenker. Integrated services in the Internet Architecture: An Overview. RFC 1633, June 1994. http://www.ietf.org/rfc/rfc1633.txt.

[9] R. F. Buccheit. Delay Compensation in Networked Computer Games. M.S. project, Case Western Reserve University, 2004.

[10] K. C. Lan, A. Hussain, and D. Dutta. The Effect of Malicious Traffic on the Network. In *Passive and Active Measurement Workshop (PAM)*, April 2003.

[11] B. N. Chun and D. E. Culler. User-Centric Performance Analysis of Market-Based Cluster Batch schedulers. In *Proceedings of the 2nd IEEE International Symposium on Cluster Computing and the Grid*, May 2002.

[12] L. Deri. Improving passive packet capture:beyond device polling. In *Proceedings of SANE 2004*, October 2004.

[13] The ATM Forum. The ATM Forum Traffic Management Specification version 4.0. April 1996.

[14] M. W. Garrett. Service architecture for ATM: from applications to scheduling. *IEEE Network*, 10(3):6–14, May/June 1996.

[15] IRTF TMRG group. The transport modeling research group's web page. `http://www.icir.org/tmrg/`.

[16] Nortel Networks. *QoS Performance requirements for UMTS*. The 3rd Generation Partnership Project (3GPP). `http://www.3gpp.org/ftp/tsg_sa/WG1_Serv/TSGS1_03-HCourt/Docs/Docs/s1-99362.pdf`.

[17] N. Sheldon, E. Girard, S. Borg, M. Claypool, and E. Agu. The Effect of Latency on User Performance in Warcraft III. In *Proceedings of ACM Network and System Support for Games (NetGames)*, May 2003.

[18] L.A.R. Yamamoto and J.G. Beerends. Impact of Network Performance Parameters on the End-to-End Perceived Speech Quality. In *Proceedings of EXPERT ATM Traffic Symposium*, September 1997.