Accurately Measuring Denial of Service in Simulation and Testbed Experiments

Jelena Mirkovic *Member, IEEE,* Alefiya Hussain, Sonia Fahmy *Senior Member, IEEE,*
Peter Reiher *Member, IEEE,* Roshan K. Thomas

***Abstract*—Researchers in the denial of service (DoS) field lack accurate, quantitative and versatile metrics to measure service denial in simulation and testbed experiments. Without such metrics, it is impossible to measure severity of various attacks, quantify success of proposed defenses and compare their performance. Existing DoS metrics equate service denial with slow communication, low throughput, high resource utilization and high loss rate. These metrics are not versatile because they fail to monitor *all* traffic parameters that signal service degradation. They are not quantitative because they fail to specify exact ranges of parameter values that correspond to good or poor service quality. Finally, they are not accurate since they were not proven to correspond to human perception of service denial.**

We propose several DoS impact metrics that measure the quality of service (QoS) experienced by end users during an attack. Our metrics are quantitative: they map QoS requirements for several applications into measurable traffic parameters with acceptable, scientifically-determined thresholds. They are versatile: they apply to a wide range of attack scenarios, which we demonstrate via testbed experiments and simulations. We also prove metrics' accuracy through testing with human users.

I. Introduction

Denial of service (DoS) is a major threat. DoS severely disrupts legitimate communication by exhausting some critical limited resource via packet floods or by sending malformed packets that cause network elements to crash. The large number of devices, applications and resources involved in communication offer a wide variety of mechanisms to deny service. Effects of DoS attacks are experienced by end users as a severe slowdown, service quality degradation or service disruption.

DoS attacks have been studied through network simulation or testbed experiments. Accurately measuring the impairment of service quality perceived by human clients during an attack is essential for evaluation and comparison of potential DoS defenses, and for study of novel attacks. Researchers and developers need *accurate*, *quantitative* and *versatile* DoS impact metrics whose use does not require significant changes in current simulators and experimental tools. *Accurate* metrics produce measures of service denial that closely agree with a human's perception of service impairment in a similar scenario. *Quantitative* metrics define ranges of parameter values that signify service denial, using scientific guidelines. *Versatile* metrics apply to many DoS scenarios regardless of the underlying mechanism for service denial, attack dynamics, legitimate traffic mix or network topology.

This material is based on research sponsored by the Department of Homeland Security under agreement number FA8750-05-2-0197. The views and conclusions contained herein are those of the authors only.

Existing approaches to DoS impact measurement fall short of these goals. They collect one or several traffic measurements and compare their first order statistics (e.g., mean, standard deviation, minimum or maximum) or their distributions in the baseline and the attack case. Frequently used traffic measurements include the legitimate traffic's request/response delay, legitimate transactions' durations, legitimate traffic's goodput, throughput or loss, and division of a critical resource between the legitimate and the attack traffic. If a defense is being evaluated, these metrics are also used for its collateral damage.

Lack of consensus on which measurements best reflect the DoS impact cause researchers to choose ones they feel are the most relevant. Such metrics are not versatile, since each independent traffic measurement captures only one aspect of service denial. For example, a prolonged request/response time will properly signal denial of service for two-way applications, such as Web, FTP and DNS, but not for media traffic that is sensitive to one-way delay, packet loss and jitter. The lack of common DoS impact metrics prevents comparison among published work. We further argue that the current measurement approaches are neither quantitative nor accurate. Ad-hoc comparisons of measurement statistics or distributions only show how network traffic behaves *differently* under attack, but do not quantify which services have been denied and how severely. To our knowledge, no studies show that existing metrics agree with human perception of service denial. We survey existing DoS impact metrics in Section II.

We propose a novel, user-centric approach to DoS impact measurement. Our key insight is that DoS always causes degradation of service quality, and a metric that holistically captures a human user's QoS perception will be applicable to all test scenarios. For each popular application, we specify its QoS requirements, consisting of relevant traffic measurements and corresponding thresholds that define good service ranges. We observe traffic as a collection of high-level tasks, called "transactions" (defined in Section III). Each legitimate transaction is evaluated against its application's QoS requirements; transactions that do not meet all the requirements are considered "failed." We aggregate information about transaction failure into several intuitive qualitative and quantitative composite metrics to expose the precise interaction of the DoS attack with the legitimate traffic. We describe our proposed metrics in Section III. We demonstrate that our metrics meet the goals of being accurate, quantitative and versatile (1) through testbed experiments with multiple DoS scenarios and rich legitimate traffic mixes (Section IV), (2) through NS-2 simulations (Section V) and (3) through

experiments involving human users (Section VI). We survey related work in Section VII and conclude in Section VIII.

This paper's contributions are three-fold: (1) We propose a novel approach to DoS impact measurement relying on application-specific QoS requirements. Although our proposed metrics combine several existing approaches, their novelty lies in (i) the careful specification of traffic measurements that reflect service denial for the most popular applications, and (ii) the definition of QoS thresholds for each measurement and each application class, based on extensive study of the QoS literature. (2) We aggregate multiple measurements into intuitive and informative DoS metrics that can be directly applied to existing testbed experiments and simulations, and to a variety of DoS scenarios. (3) We demonstrate that our metrics accurately capture human perception of service denial by conducting experiments with human users.

Admittedly, calculating our metrics is more complex than legacy ones. To ease this process, we have made the program used for DoS metrics calculation from network traces freely available at `http://www.isi.edu/~mirkovic/dosmetric`.

II. Existing Metrics

Prior DoS research has focused on measuring denial of service through selected legitimate traffic parameters: (a) packet loss, (b) traffic throughput or goodput, (c) request/response delay, (d) transaction duration, and (e) allocation of resources. Researchers have used both simple metrics (single traffic parameter) and combinations of them to report the impact of an attack on the network.

All existing metrics are not quantitative because they do not specify ranges of loss, throughput, delay, duration or resource shares that correspond to service denial. Indeed, such values cannot be specified in general because they highly depend on the type of application whose traffic co-exists with the attack: 10% loss of VoIP traffic is devastating, while 10% loss of DNS traffic is merely a glitch. All existing metrics are further not versatile and we point out below the cases where they fail to measure service denial. They are inaccurate since they have not been proven to correspond to a human user's perception of service denial.

Loss is defined as the number of packets or bytes lost due to the interaction of the legitimate traffic with the attack [1] or due to collateral damage from a defense's operation. The loss metric primarily measures the presence and extent of congestion in the network due to flooding attacks. It cannot be used for attacks that do not continually create congestion, or do not congest network resources at all. Examples of such attacks are pulsing attacks [2], [3], TCP SYN floods [4], attacks that target application resources and vulnerability attacks that crash applications and hosts. Further, the loss metric typically does not distinguish between the types of packets lost, while some packet losses have a more profound impact than others (for example, a lost SYN vs data packet) on service quality.

Throughput is defined as the number of bytes transferred per unit time from the source to the destination. **Goodput** is similar, but does not count retransmitted bytes [2], [5]. Both are meaningful for TCP-based traffic, which responds to congestion by lowering its sending rate. Indirectly, these metrics capture the presence and extent of congestion in the network and the prolonged duration of legitimate transactions due to congestion. They cannot be applied to applications that are sensitive to jitter or to loss of specific (e.g., control) packets, because a high throughput level may still not satisfy the quality of service required by the user. Further, these metrics do not effectively capture DoS impact on traffic mixes consisting of short connections, with a few packets to be sent to the server. Such connections already have a low throughput so service denial may be masked.

Request/response delay is defined as the interval between when a request is issued and when a complete response is received from the destination [6]. It measures service denial of interactive applications (e.g,, telnet) well, but fails to measure it for non-interactive applications (e.g., email) which have much larger thresholds for acceptable request/response delay. This metric is also inapplicable to one-way traffic (e.g., media traffic) which does not generate responses but is sensitive to one-way delay, loss and jitter.

Transaction duration is the time needed for an exchange of a meaningful set of messages between a source and a destination [7], [8], [9]. This metric depends heavily on the volume of data being transferred and whether the application is interactive and congestion-sensitive. It accurately measures service denial for interactive applications, such as Web browsing. For one-way traffic, such as media streaming that may not respond to congestion and runs over UDP, transaction duration will not be affected by the attack. Duration of many non-interactive transactions can be extended without causing service denial because humans expect that such transactions may occur with some delay.

Allocation of resources is the fraction of a critical resource (usually bandwidth) allocated to legitimate traffic vs. attack traffic [8], [10]. This metric does not provide any insight into the user-perceived service quality. It assumes the service is denied due to lack of resources, and applies only to flooding attacks. Further, it cannot capture collateral damage of a given defense. For example, a defense that drops 90% of legitimate and 100% of attack traffic, would appear perfect, since it allocates all remaining resources to legitimate traffic.

We acknowledge that the existing metrics convey some notion of denied service, especially when the denial is severe. They, however, suffer from two major drawbacks: (1) They measure a single traffic parameter assuming that its degradation always corresponds to service denial, whereas traffic parameters that signal service denial are actually application-specific and some attacks can deny service without affecting the monitored parameter. (2) They fail to define the parameter range required for acceptable service quality, which is application- and task-specific. Finally, the existing metrics predominantly capture the service denial at the network layer, en route to the victim server. While many attacks target this route, some affect the server host or the application directly, or target supporting network services (such as DNS), or the route from the server to legitimate users. Network-based metrics fail to correctly capture the impact of these attacks.

III. Proposed DoS Impact Metrics

We now introduce several definitions needed for our DoS impact metrics. The client is the host that initiates communication with another party, which we call the server.

Definition 1: A ***conversation*** between a client and a server is the set of all packets exchanged between these two hosts to provide a specific service to the client, at a given time. A conversation is explicitly initiated by a client application (e.g., by opening a TCP connection or sending a UDP packet to a well-known service) and ends either explicitly (a TCP connection is closed, UDP service is rendered to the client) or after a long period of inactivity. If several channels are needed to render service to a client, such as FTP control and data channels, all related channels are part of a single conversation.

Definition 2: A ***transaction*** is the part of a conversation that represents a higher-level task whose completion is perceptible and meaningful to a user. A transaction usually involves a single request-reply exchange between a client and a server, or several such exchanges that occur close in time. A conversation may contain one or several transactions.

Definition 3: A transaction is ***successful*** if it meets all the QoS requirements of its corresponding application. If at least one QoS requirement is not met, a transaction has ***failed***. Transaction success/failure is the core of our proposed metrics.

A. Application-Specific QoS Requirements

Our first step was to identify traffic measurements that are important for service quality for the most popular applications today. Several organizations that collect and publish traffic traces [11], [12] analyze Internet applications and the ratio of the packets and bytes that they contribute to these traces. We surveyed their findings to assemble a list of popular applications. Further, we leverage the findings of the 3GPP consortium on defining application QoS requirements [13], complemented with findings from contemporary QoS research [14], [15], [16], [17].

Table I summarizes the application categories we propose, and their corresponding QoS requirements. Should novel applications become popular in the future, the proposed application categories will need to be extended, but our DoS impact metrics will be immediately applicable to new applications.

Interactive applications such as Web, file transfer, telnet, email (between a user and a server), DNS and ping involve a human user requesting a service from a remote server, and waiting for a response. Their primary QoS requirement is that a response is served within a user-acceptable delay. Research on human perception of Web traffic delay shows that people can tolerate higher latencies for entire task completion if some data is served incrementally [14]. We specify two types of delay requirements for email, Web, telnet and file transfer transactions where a user can utilize a partial response: (a) *partial delay* measured between receipt of any two data packets from the server. For the first data packet, partial delay is measured from the end of a user request, and (b) *whole delay* measured from the end of a user request until the entire response has been received. Additionally, a telnet application serves two types of responses to a user: it echoes characters that a user types, and then generates a response to the user request. The echo generation must be faster than the rest of the response, so we define the *echo delay* requirement for telnet transactions. We identify the echo delay as the delay between a user's request and the first response packet.

We use 250 ms as the telnet echo delay requirement [13]. We use 4 s as the partial-delay threshold for Web, telnet and email applications [13], and 10 s for file transfer applications [13]. We use 60 s as the whole-delay requirement for Web [14], and require that the delay for email and file transfer not exceed three times the expected delay [18], given the amount of data being transferred. The expected delay is defined as the delay experienced by the same transaction in the absence of an attack. For DNS and ping services, we adopt a 4 s whole-delay requirement. This is the maximum human-acceptable delay for interactive tasks [13]. We consider peer-to-peer applications to be file transfers.

Media applications such as conversational and streaming audio and video have strict requirements for low loss, low jitter and low one-way delay. These applications further involve a media channel (where the audio and video traffic are sent, usually via UDP) and a control channel (for media control). Both of these channels must provide satisfactory service to the user. We adopt the one-way delay and loss requirements for media traffic from [13]. Because many media applications can sustain higher jitter than 1 ms [13] using variable-size buffers, we adopt the jitter threshold value of 50 ms [19]. We treat control traffic as interactive traffic requiring a 4 s partial-delay.

Online games have strict requirements for low one-way delay and loss [13]. We differentiate between first-person shooter (FPS) and real time strategy (RTS) games, because research has shown that their QoS requirements differ. We use [16] (FPS) and [17] (RTS) as sources for specifying delay and loss bounds (see Table I for specific values).

Chat applications can be used for text and media transfer between two human users. While the request/response delays depend on human conversation dynamics, the receipt of user messages by the server must be acknowledged within a certain time. We express this delay requirement as a 4 s threshold on the round-trip time between the client and the server. Additionally, we apply the QoS requirements for media applications to the media channel of the chat application.

Non-interactive services such as email transfer between servers and Usenet do not have a strict delay requirement. Users will accept long delays as long as the transactions complete within a given interval. 3GPP [13] specifies the transaction duration threshold as *several hours* for email and Usenet. We quantify this as 4 hours, since this value is commonly used by mail servers to notify a user about a failure to deliver mail to the destination server.

We impose an additional requirement on services that run over TCP, which was not included in our previous work [20]. We require that data must be exchanged between the client and the server during a transaction. This is important to detect failures where an application aborts its TCP connection without transmitting any data due to adverse network conditions.

Category	One-way delay	Req/rep delay	Loss	Duration	Jitter
email (srv/srv)		whole, RTT <4 h			
Usenet		whole, RTT <4 h			
chat, typing		RTT <4 s			
chat, typing	some data must be sent to server				
chat, audio	<150 ms	whole, RTT <4 s	<3%		<50 ms
chat, video	<150 ms	whole, RTT <4 s	<3%		
Web		part, RTT <4 s		<60 s	
Web	some data must be received from server				
FTP Data		part, RTT <10 s		<300%	
FTP Control		part, RTT <4 s			
FTP	some data must be exchanged on data channel				
FPS games	<150 ms		<3%		
RTS games	<500 ms				
telnet		part, RTT <250 ms			
telnet	some data must be received from server				
email (usr/srv)		part, RTT <4 s		<300%	
DNS		whole <4 s			
ping		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 I
APPLICATION CATEGORIES AND THEIR QOS REQUIREMENTS.

Application	Transaction
email (srv/srv), Usenet	TCP flow
chat, Web, telnet, email (usr/srv)	TCP flow and inactive $>$ 4 s
FTP	TCP flow and inactive $>$ 4 s on both control and data channel
games	UDP flow and inactive $>$ 4 s
DNS, ping	One request/response exchange with unique request ID
audio and video	TCP flow (control channel) and matching UDP flow (media traffic)

TABLE II
TRANSACTION IDENTIFICATION.

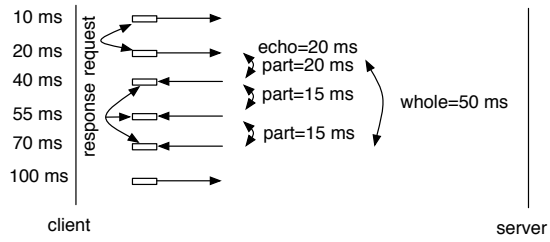


Fig. 1. Illustration of request/response identification.

B. Measurement Approach

During simulation, collection of necessary traffic measurements usually implies slight simulator modification. Such collection is a challenge in testbed experimentation, and we explored two possible approaches: (i) Instrumented-clients: instrumenting each client application to compute required measurements, or (ii) Trace-based: using real, uninstrumented applications and traffic generators, identifying transactions in collected packet traces and computing traffic measurements. The instrumented client approach can precisely identify transactions, but it limits the metrics' usability to open-source clients. We thus decided to use the trace-based approach, since it is easily applicable to most test scenarios and immediately usable by other researchers. In implementing trace-based QoS evaluation, we encountered several challenges in transaction and request/response identification. We summarize our handling of these challenges here; more details are in [20]. Table II shows how we identify transactions in the trace data. For interactive applications, an inactive time (user think time) followed by a new user's request denotes a new transaction. A transaction is either a partial or entire flow, where flow is defined as all traffic exchanged between two IP addresses and port numbers. For traffic requiring multiple flows, such as media or FTP traffic, a transaction spans both flows.

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

C. DoS Metrics

We aggregate the transaction success/failure measures into several intuitive composite metrics.

Percentage of failed transactions (pft) per application type. This metric directly captures the impact of a DoS attack on network services by quantifying the QoS experienced by end users. For each transaction *that overlaps with the attack*, we evaluate transaction success or failure applying definition 3. A straightforward approach to the *pft* calculation is dividing the number of failed transactions by the number of all transactions during the attack. This produces biased results for clients that generate transactions serially. If a client does not generate each request in a dedicated thread, timing of subsequent requests depends on the completion of previous requests. In this case, transaction density during an attack will be lower than without an attack, since transactions overlapping the attack will last longer. This skews the *pft* calculation because each success or failure has a higher influence on the *pft* value during an attack than in its absence. In our experiments, IRC and telnet clients suffered from this deficiency. To remedy this problem, we calculate the *pft* value as the difference between 1 (100%) and the ratio of the number of successful transactions divided by the number of all transactions *that would have been initiated by a given application during the same time if the attack were not present*.

The ***DoS-hist*** metric shows the histogram of *pft* measures across applications, and is helpful to understand each application's resilience to the attack.

The ***DoS-level*** metric is the weighted average of *pft* measures for all applications of interest: *DoS-level* $= \sum_k pft(k) \cdot w_k$, where k spans all application categories, and w_k is a weight associated with a category k. We introduced this metrics because in some experiments it may be useful to produce a single number that describes the DoS impact. But

we caution that *DoS-level* is highly dependent on the chosen application weights and thus can be biased.

QoS-ratio is the ratio of the difference between a transaction's traffic measurement and its corresponding threshold, divided by this threshold. The ***QoS*** metric for each successful transaction shows the user-perceived service quality, in the range $(0, 1]$, where higher numbers indicate better quality. It is useful to evaluate service quality degradation during attacks. We compute it by averaging *QoS-ratios* for all traffic measurements of a given transaction that have defined thresholds.

For failed transactions, we compute the related ***QoS-degrade*** metric, to quantify severity of service denial. *QoS-degrade* is the absolute value of *QoS-ratio* of that transaction's measurement that exceeded its QoS threshold by the largest margin. This metric is in the range $[0, +\infty)$. Intuitively, a value N of *QoS-degrade* means that the service of failed transactions was N times worse than a user could tolerate. While arguably any denial is significant and there is no need to quantify its severity, perception of DoS is highly subjective. Low values of *QoS-degrade* (e.g., < 1) may signify service quality that is acceptable to some users.

The ***life diagram*** shows the birth and death of each transaction in the experiment with horizontal bars. The x-axis is time and the bar position shows a transaction's birth (start of the bar) and death (its end). We show failed and successful transactions on separate diagrams, for clarity. This metric can help quickly show which transactions failed and indicate clusters that may point to a common cause.

The ***failure ratio*** shows the percentage of live transactions in the current (1-second) interval that will fail in the future. The failure ratio is useful for evaluation of DoS defenses, to capture the speed of a defense's response, and for time-varying attacks [2]. Transactions that are born during the attack are considered live until they complete successfully or fail. Transactions that are born before the attack are considered live after the attack starts. A failed transaction contributes to the failed transaction count in all intervals where it was live.

IV. Evaluation in Testbed Experiments

We first evaluate our metrics in experiments on the DETER testbed [21]. The testbed is located at the USC Information Sciences Institute and UC Berkeley, and allows security researchers to evaluate attacks and defenses in a controlled environment.

A. Topology

Fig. 2 shows our experimental topology. Four legitimate networks and two attack networks are connected via four core routers. Each legitimate network has four server nodes and two client nodes, and is connected to the core via an access router. Links between the access router and the core have 100 Mbps bandwidth and 10–40 ms delay, while other links have 1 Gbps bandwidth and no added delay. The location of bottlenecks is chosen to mimic high-bandwidth local networks that connect over a limited access link to an over-provisioned core. Attack networks host two attackers each, and connect directly to core routers.

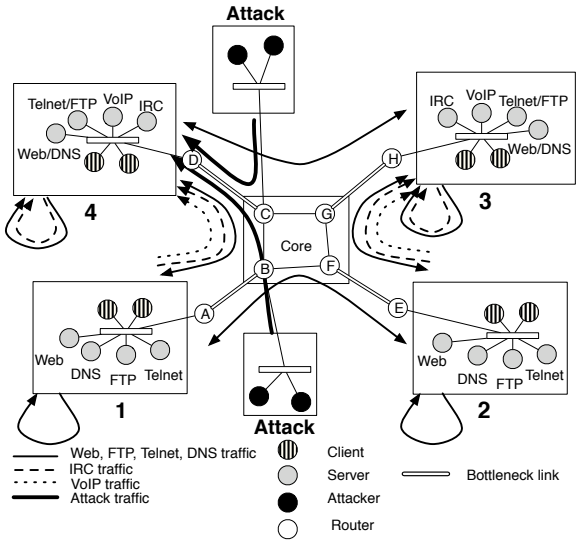


Fig. 2. Experimental topology.

B. Background Traffic

Each client generates a mixture of Web, DNS, FTP, IRC, VoIP, ping and telnet traffic. We used open-source servers and clients when possible to generate realistic traffic at the application, transport and network level. For example, we used an `Apache` server and `wget` client for Web traffic, `bind` server and `dig` client for DNS traffic, etc. Telnet, IRC and VoIP clients and the VoIP server were custom-built in Perl.

Clients talk with servers in their own and adjacent networks. Fig. 2 shows the traffic patterns. Traffic patterns for IRC and VoIP differ because those application clients could not support multiple simultaneous connections. All attacks target the Web server in network 4 and cross its bottleneck link, so only this network's traffic should be impacted by the attacks.

Our previous work [20] used a similar experimental setup to illustrate our metrics in realistic traffic scenarios for various attacks. Here, we show a different set of experiments with one novel attack scenario (Section IV-D). We modified the topology from [20] to ensure that bottlenecks occur only before the attack target, to create more realistic attack conditions. We used a more artificial traffic mix than in [20], with regular service request arrivals and identical file sizes for each application, to clearly isolate and illustrate features of our metrics. Traffic parameters are chosen to produce the same transaction density in each application category (Table III): roughly 100 transactions for each application during 1,300 seconds, which is the attack duration. All transactions succeed in the absence of the attack.

C. UDP Bandwidth Flood

Our first experiment is a UDP flood attack, frequently used in the literature and frequently observed in the Internet. This attack can deny service in two ways: (1) by generating a large traffic volume that exhausts bandwidth on bottleneck links (more frequent variant), (2) by generating a high packet rate that exhausts the CPU at a router leading to the target. We generate the first attack type: a UDP bandwidth flood. Packet

Type	Parameter (unit)	Distribution
telnet	Request interarrival time	10 s
	Response size	4 KB
	Session duration	60 s
	Time between sessions (s)	15 s
FTP	Request interarrival time	13 s
	File size	10 KB
Web	Request interarrival time	13 s
	File size	1 KB
DNS	Request interarrival time	13 s
ping	Request interarrival time	13 s
IRC	Request interarrival time	5 s
	Message size	10 KB
VoIP	Packet interarrival time	0.03 s
	Talk time	8 s
	Think time	5 s

TABLE III
LEGITIMATE TRAFFIC PARAMETERS AND THEIR VALUES.

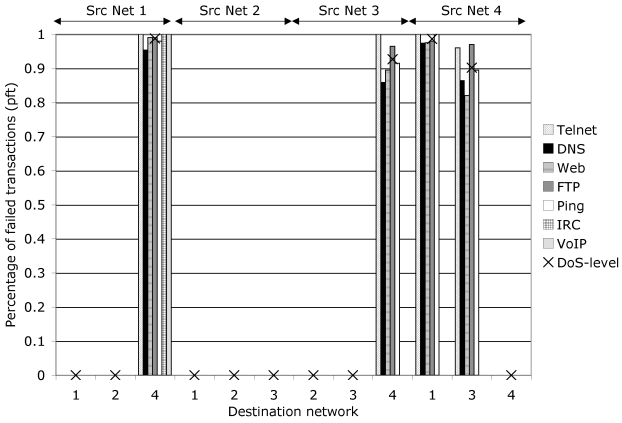


Fig. 3. UDP bandwidth flood: *DoS-hist* and *DoS-level* measures.

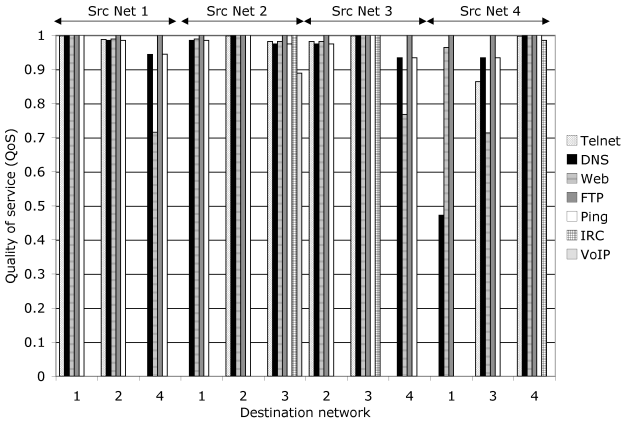


Fig. 4. UDP bandwidth flood: *QoS* measures for successful transactions.

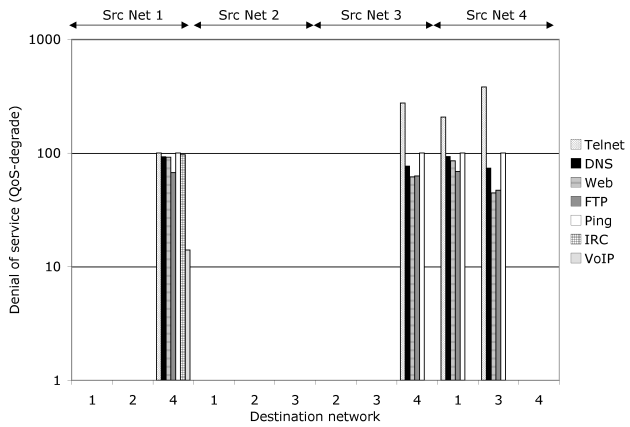


Fig. 5. UDP bandwidth flood: *QoS-degrade* measures for failed transactions.

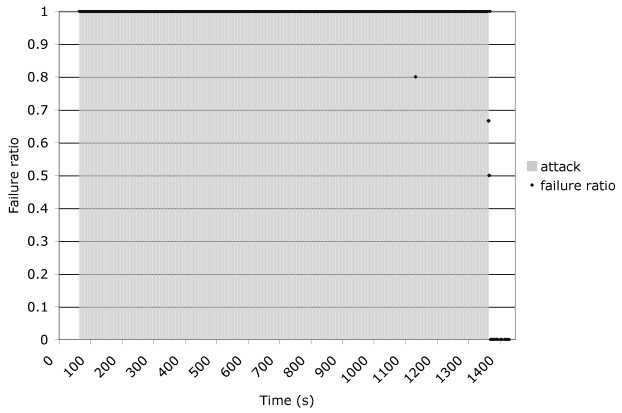


Fig. 6. UDP bandwidth flood: *Failure ratio* for transactions from network 4 to network 1.

sizes had range [750 B, 1.25 KB] and total packet rate was 200 Kpps. This generates a volume that is roughly 16 times the bottleneck bandwidth. The expected effect is that access link of network 4 will become congested and traffic between networks 1 and 4, and networks 3 and 4 will be denied service.

Fig. 3 shows the *DoS-hist* measures for all source and destination networks, and the *DoS-level* measure assuming equal application weights. Labels at the top of the graph show measures that belong to the same source network, x-axis labels denote the destination network, and the y-axis shows the *pft* per application. As expected, only traffic to and from network 4 is affected. Transactions between networks 1 and 4 have somewhat higher *pft* than transactions between networks 3 and 4. A similar trend is also noticeable in other experiments, and occurs because traffic between networks 1 and 4 shares one more router with the attack (router B) than does traffic between 3 and 4 (crosses C and D but not B). DoS-level is around 98% for traffic between 1 and 4, and around 91% for traffic between 3 and 4.

Fig. 4 shows the *QoS* measure, averaged over successful transactions. Service quality degrades among transactions involving servers or clients in network 4. Other transactions have consistently high service quality. The *QoS-degrade* measure is shown in Fig. 5, averaged over failed transactions. While a single large value could bias this metric, values in our experiments were fairly balanced over failed transactions in the same application category. Transactions with network 4 experience large service denial, receiving a service with 10-300 times worse quality than expected.

Fig. 6 shows the failure ratio for transactions originating

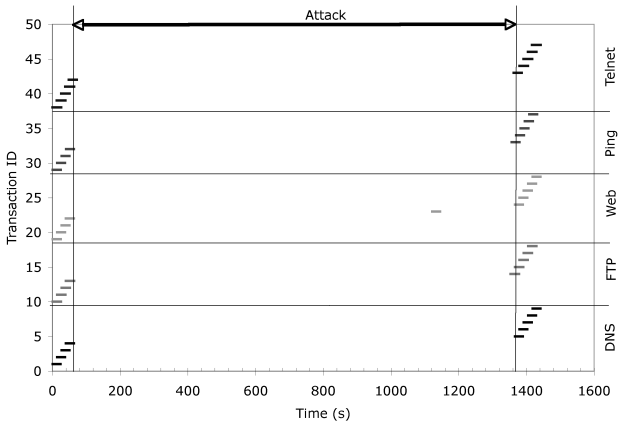


Fig. 7. UDP bandwidth flood: *Life diagram* of successful transactions from network 4 to network 1.

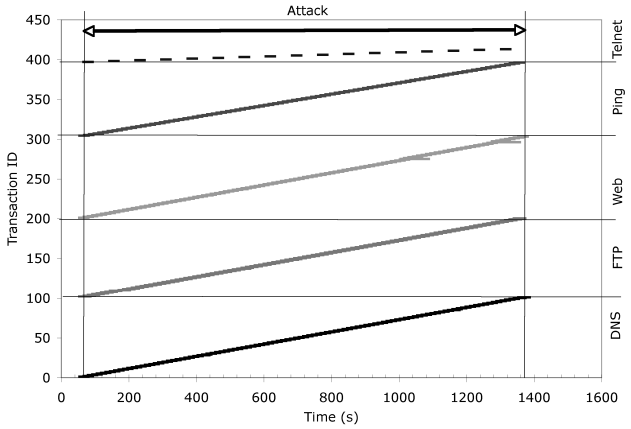


Fig. 8. UDP bandwidth flood: *Life diagram* of failed transactions from network 4 to network 1.

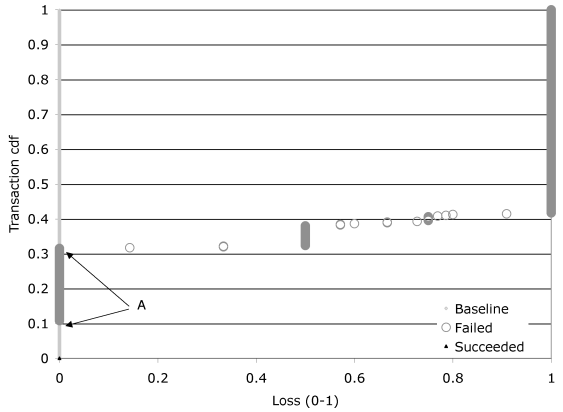


Fig. 9. UDP bandwidth flood: Transaction cdf with respect to loss; transactions originated by network 4 with network 1.

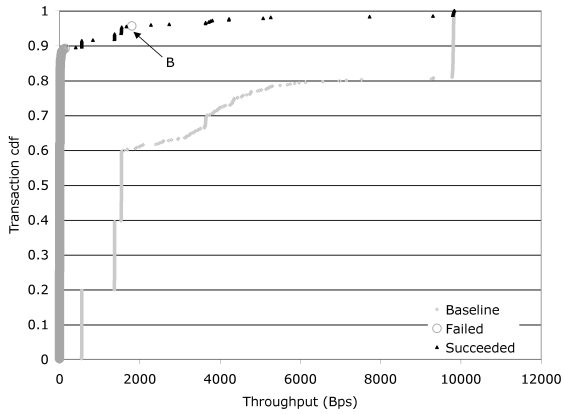


Fig. 10. UDP bandwidth flood: Transaction cdf with respect to average data throughput; transactions originated by network 4 with network 1.

from network 4 to network 1. Throughout the attack, the failure ratio value stays close to 1, illustrating that nearly all service between these two networks is denied.

Fig. 7 and 8 show the life diagrams of successful and failed transactions. The x-axis shows the start and end time of a transaction, the bar length represents transaction duration, and the y-axis shows the transaction ID. We assign consecutive IDs to transactions of the same type. All failures occur during the attack, and all transactions fail regardless of their application type. One Web transaction succeeds during the attack because it obtains enough bandwidth by chance in competition with the attack. Note the difference in transaction density during the attack between telnet and other applications (Fig. 8). Telnet and IRC clients in our experiments generate transactions serially and thus their transaction density reduces when an attack prolongs transactions.

We now contrast our metrics with the legacy metrics: transaction duration, request/response delay, throughput, loss and resource allocation. Since the UDP bandwidth flood is the simplest form of DoS attack that denies service through excessive congestion, we expect that many existing metrics will do well in predicting transaction failure. An effective metric would have a clear separation of values for successful and for failed transactions.

The cumulative distribution function (cdf) of maximum loss within a 5-sec interval for all transactions originated by network 4 with network 1 is shown in Fig. 9. We also show the cdf in the baseline case. Baseline transactions all have zero loss, and are clustered at the origin on the graph. Successful transactions also all have zero loss, and quite a few failed transactions have high loss (between 0.5 and 1). However, many failed transactions have zero loss, as shown in area A in the figure, and fail because their other QoS requirements are not met. This overlap between values for successful and failed transactions makes the loss metric insufficient for DoS measurement.

Fig. 10 shows the cdf of average data throughput (control packets are not counted) for all transactions during the attack and for the baseline case. The attack clearly lowers the transaction throughput — many failed transactions have throughput close to zero (and zero, not shown on the log-scale graph) and all successful transactions have a higher throughput. However, there is one transaction that failed despite high throughput, shown in the area B in the figure. This was a Web transaction that managed to quickly deliver a request to the server; the request was acknowledged but the data reply was lost.

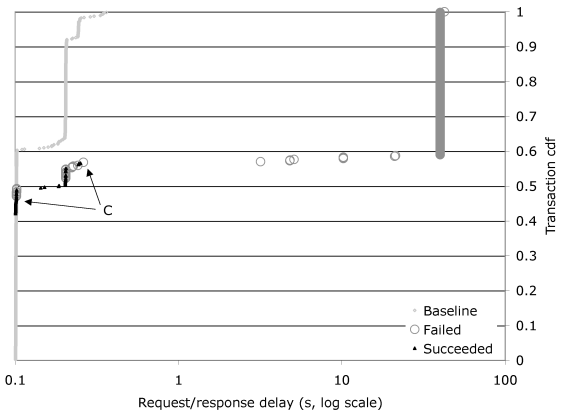


Fig. 11. UDP bandwidth flood: Transaction cdf with respect to request/reply delay; transactions originated by network 4 with network 1.

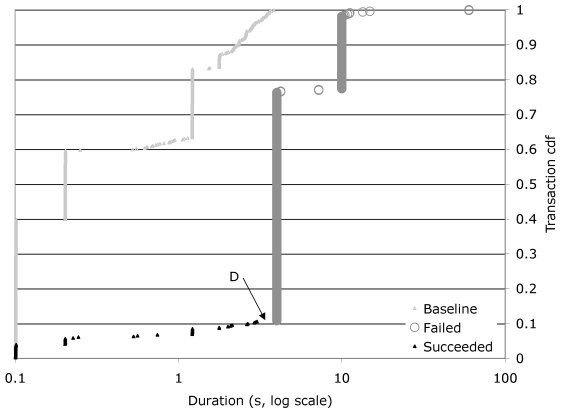


Fig. 12. UDP bandwidth flood: Transaction cdf with respect to duration; transactions originated by network 4 with network 1.

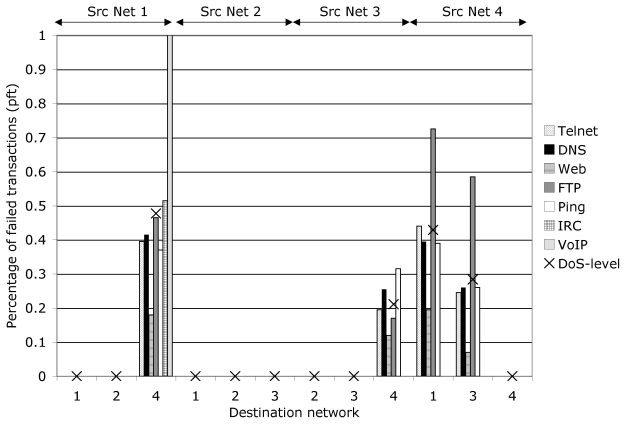


Fig. 13. UDP bandwidth flood — low-rate: *DoS-hist* and *DoS-level* measures.

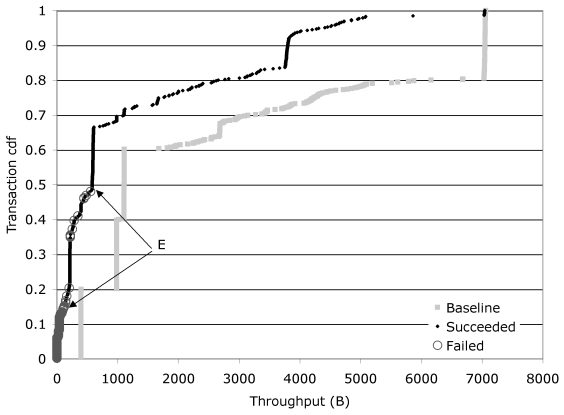


Fig. 14. UDP bandwidth flood — low-rate: Transaction cdf with respect to average throughput; transactions originated by network 3 with network 4.

Because transactions can fail and still have high throughput, the throughput metric by itself cannot accurately measure DoS.

The cdf of request/reply delay for all transactions is shown in Fig. 11, during the attack and for the baseline case. Many failed transactions have high delay but there is a significant overlap in delay values between failed and successful transactions, in the area C in the figure. This overlap makes request/reply delay insufficient for DoS measurement.

Fig. 12 shows the cdf of transaction duration for all transactions during the attack, and for the baseline. The attack prolongs durations, and successful transactions finish sooner than failed ones. There is a narrow but clear separation of values, in the area D in the figure. Thus the duration metric could predict DoS in this particular experiment but we will show it fails in experiments with other attack types.

Considering resource allocation, around 97% of bandwidth on network 4's access link was consumed by the attack. This is close to some *DoS-hist* and *DoS-level* values for transactions between networks 1 and 4, in Fig. 3, but higher than the DoS impact on transactions between networks 3 and 4. Hence, resource allocation metric indicates DoS impact in this case, but is not completely accurate in predicting its severity.

The remaining experiments discuss a subset of the metrics.

D. UDP Bandwidth Flood — Low Rate

We now illustrate the inadequacy of metrics that were adequate for high-rate UDP bandwidth floods — duration, resource allocation, and throughput (if we ignore the one failed transaction with high throughput). We reduce the rate of the UDP flood attack to 80% of the bottleneck link bandwidth.

Fig. 13 shows the *DoS-hist* and *DoS-level* measures. Traffic to and from network 4 suffers service denial, but the percentage of impaired traffic varies greatly depending on application. Web transactions suffer the least service denial (8-20%), followed by telnet, DNS and ping. FTP is less impacted when the server is in network 4 than when the clients are there, because our FTP transactions are downloads, so most data flows from server to client. This is also why FTP suffers more than Web, telnet, DNS and ping when network 4 is the source network. About 50% of IRC transactions fail and 100% of VoIP transactions fail. For the *QoS-degrade* metric (not shown in graph due to space), telnet, DNS, FTP and ping traffic have 10-100 times worse QoS than required. When the server is in network 4, Web traffic has 2-10 times degraded service and VoIP has only 0.15 times degradation. Clearly, resource allocation metrics cannot predict such variability in service denial: 20% of resources are allocated to legitimate traffic.

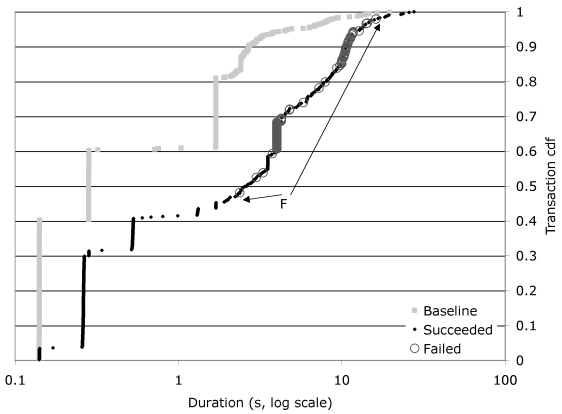


Fig. 15. UDP bandwidth flood — low-rate: Transaction cdf with respect to duration; transactions originated by network 3 with network 4.

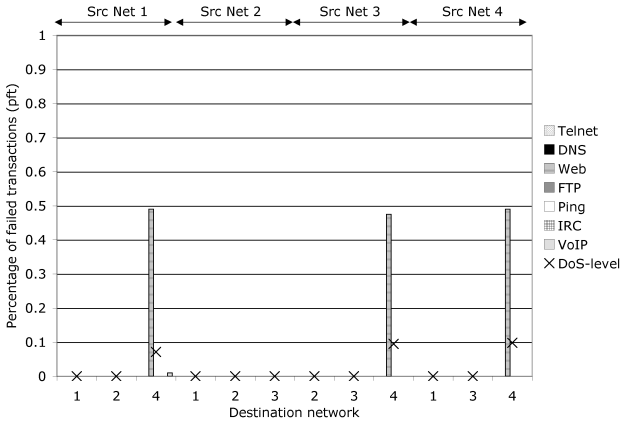


Fig. 16. TCP SYN flood with syn-cookies: *DoS-hist* and *DoS-level* measures.

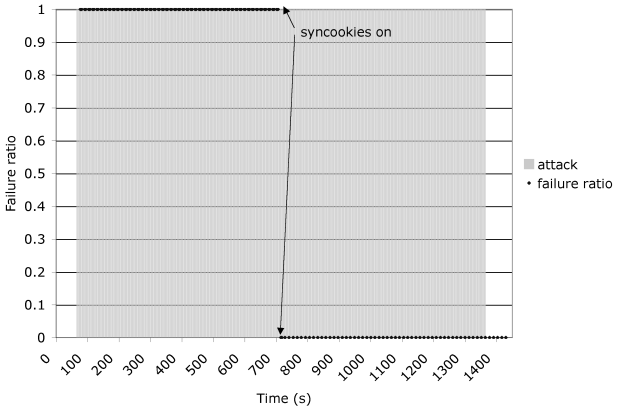


Fig. 17. TCP SYN flood with syn-cookies: Failure ratio for Web traffic from network 1 to network 4.

Fig. 14 shows the cdf of average throughput for all transactions during the attack initiated by network 3 with network 4, and for the corresponding baseline case. There is a significant overlap of throughput values for successful and failed transactions in the area E, which shows that throughput by itself cannot accurately measure DoS.

Fig. 15 shows the cdf of duration for all transactions initiated by network 3 with network 4 during the attack, and for the corresponding baseline case. Durations of failed and successful transactions overlap in area F in the graph, showing that duration by itself cannot accurately measure DoS.

Loss and request/reply delay do not adequately capture DoS impact due to a large overlap in values for failed and successful transactions. We omit these graphs due to space.

E. TCP SYN Flood with Syn-cookie Defense

Another popular attack with both attackers and researchers is the TCP SYN flood [4]. It denies service by sending a TCP SYN flood that consumes OS memory at the target. This attack can be largely countered if the target deploys the TCP syn-cookie defense [22], which allocates memory only after the 3-way handshake is completed. Since attackers do not complete the handshake, the attack is thwarted. We generated a TCP SYN flood to port 80 on the Web server in network 4, sending 500 pps. We turned syn-cookies on 650 seconds after the start of attack, at time 715 seconds.

The *DoS-hist* and *DoS-level* measures are shown in Fig. 16. As expected, all traffic to network 4's Web server suffers service denial. The severity is around 50%, in line with the expectation that almost all transactions were denied service before the syn-cookie defense was turned on, and none afterward. There is a slight DoS for the VoIP traffic from network 1 to network 4, when 1 out of 100 transactions fails because of excessive loss. The loss is due to aggressive TCP retransmissions and is minor (3.3%) but higher than the 3% QoS threshold for VoIP.

Fig. 17 shows the Web transaction failure ratio from network 1 to network 4. During the attack, the value goes to 1, but reverts to zero when syn-cookies are deployed.

The life diagrams of successful and failed transactions are shown in Fig. 18. Only Web transactions fail during the attack, and only during a period when syn-cookies are off.

We summarize legacy metrics, for space reasons. Duration and loss metrics capture DoS impact well in this case, but throughput and request/reply delay produce overlapping regions for failed and successful Web transactions and thus cannot measure DoS accurately. Only 18% of bandwidth is consumed by the SYN flood, yet 100% of web transactions are denied service when syncookies are off. The better approach to resource allocation measurement would be to measure occupancy of the TCP connection table at the Web server. We lacked tools to obtain this information easily from the OS, but we infer from the Web transaction success/failure metrics that the table would mostly be occupied by attack connections.

V. Evaluation in NS-2 Simulations

To extend the application of our proposed metrics to simulated DDoS defense evaluation, we have ported the metrics to the popular NS-2 simulator [23]. We illustrate the DoS impact metrics in small-scale experiments using NS-2 (version 2.29), and compare the results with identical experiments on the DETER testbed. During simulations, we generate flows that

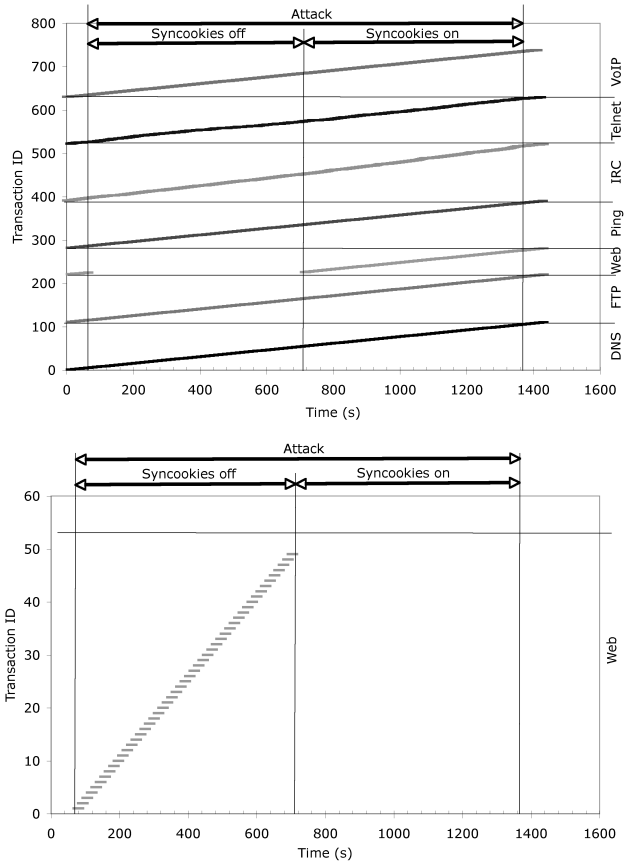
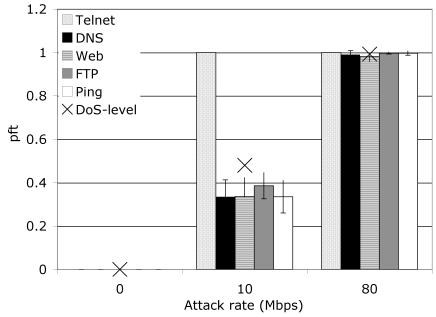
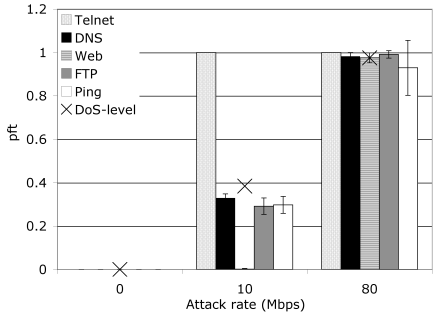


Fig. 18. TCP SYN flood with syn-cookies: *Life diagram* of successful and failed transactions from network 1 to network 4.



(a) NS-2



(b) DETER

Fig. 19. *DoS-hist* and *DoS-level* measures in NS-2 and DETER experiments.

each represent a transaction and we compute required traffic measurements from NS-2 logs.

We use a simple network topology with a single legitimate client, an attacker, and a server. All nodes are connected to the same router. The link between the server and router is 10 Mbps with 10 ms delay. The other two links are 100 Mbps bandwidth with 10 ms delay. We use a queue size of 100 packets, with a drop-tail queuing strategy. We generate the following legitimate traffic between the client and the server: (1) Web and FTP traffic with file size 1000 bytes and 20 s request interarrival period. (2) Telnet traffic with 10 pps and a 100-byte packet size. During the simulation, we start a new telnet connection every 60 s with duration of 120 s. (3) DNS and ping traffic with 10 s request interarrival period. We use the following modules in NS-2 to generate the traffic: Application/FTP for FTP, PagePool/WebTraf for Web, Application/Telnet for telnet, Agent/Ping for ICMP, and a modified version of Agent/Ping with a maximum of 3 retransmissions with 5-s timeouts for DNS. We generate a UDP flood that overwhelms the bottleneck link with 10 Mbps (moderate attack) or 80 Mbps (large attack) rate.

Fig. 19 shows the *DoS-hist* measure for the client's traffic to the server during the two attacks for the NS-2 and DETER experiments, and in no-attack case. The x-axis shows the attack strength, and the column height denotes the result of 10 test runs, with error bars shown. Since the legitimate traffic pattern is fixed for the NS-2 simulation, we achieve variability by randomly choosing a small delay (10-100 ms) to apply to the attack start time. The traffic pattern in testbed experiments depends on a random seed. We also show the DoS-level measure using equal application weights. The telnet application is the most affected by the attack due to its small echo-delay bound (250 ms). Denial of service is similar for DNS and ping, even though DNS can retransmit requests up to three times, because these retransmissions occur after the DNS request/response delay threshold is exceeded (4 s). Web transactions survive the attack best because of the generous (4 s) delay threshold and because the lost packets are retransmitted by TCP. At high attack rate (80 Mbps), the *pft* of all applications goes to almost 100%.

Comparing simulation results with testbed results (Fig. 19(a) vs 19(b)), we find that trends in both graphs are similar but more transactions fail in simulations. This is because the software routers used on the testbed can handle the attack traffic much better than the simple output queuing model used in NS-2. The results are consistent with [24], which shows much higher throughput and TCP congestion window sizes in testbed experiments compared to the same experiments in NS-2.

VI. Evaluation in Human-User Experiments

To evaluate our metrics' ability to capture a human user's perception of service denial, we have conducted an experiment where users interact with a server that is occasionally subjected to denial-of-service attacks. After each interaction, a user rates her satisfaction with service quality and we compare this rating with two of our denial-of-service metrics: the transaction success/failure metric and the *QoS* metric.

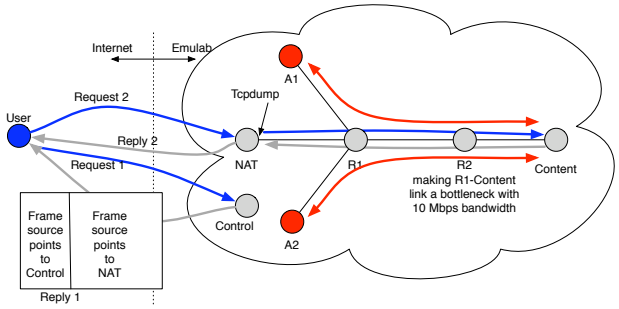


Fig. 20. Topology for Human-user DoS experiments.

A. Service and Content

We limited legitimate traffic to a single application, Web browsing, to simplify user interaction with the server and facilitate wide participation in our experiment. Users interact with the server by browsing through a set of Web pages. They rate their satisfaction with the loading speed of each page by filling in a Web form shown to the left of the page, in a separate frame.

We wanted to provide interesting and copyright-free content to attract participants and achieve reasonably long interactions with the server. We downloaded 21 select pages from Wikipedia [25], which is a highly popular online encyclopedia that allows content copying and modification under the terms of the GNU Free Documentation License. These 21 pages were grouped into four content categories: Sports (geocaching, abseiling, aerobatics, fell running, Chilean rodeo, paintball), Music (blues, hip hop, rock and roll, heavy metal, the relationship between music and mathematics), Film (Star Wars, Godfather, Lord of the Rings, Casablanca, An Inconvenient Truth) and Famous People (Walt Disney, Shakespeare, Christopher Columbus, Benjamin Franklin, Mozart). We modified each page to fit into 1-2 screens of text.

B. Experiment Setup

Because the Web server had to be subjected to occasional DoS attacks, we needed a controlled, isolated environment such as the DETER testbed [21]. However, our desire to attract many survey participants dictated the need for experimental machines to be reachable by users from outside the testbed. DETER currently prohibits any communication between external machines and experimental ones, and thus could not host our experiment. Instead, we used the Emulab testbed [26], which is similar to the DETER testbed but it allows external Web requests to experimental nodes.

A naive experimental topology would use one Web server in the Emulab testbed, and the user traffic would reach the server directly. Such a topology was inadequate for our purposes, for the following reasons:
(1) Users must reach two types of pages (a) Wikipedia content on a server that may be a DoS target, (b) welcome and thank-you pages, and pages with QoS rating forms that must always be loaded promptly regardless of an attack. We used two Web servers — one to host control information for the survey (welcome, thank-you pages and rating forms) and one to host the content and be the DoS target.
(2) For DoS attacks that target bandwidth, user traffic must share the bottleneck link with the attack. Thus, user Web requests must be tunneled to the content Web server instead of reaching it directly. Fig. 20 shows the topology used in the experiment. User traffic first reaches the Web server *Control*, which hosts control information. When the survey starts, the right frame of the pages displayed to the user points to the host *NAT*, which acts as a network address translator and tunnels the user's Web requests to the Web server *Content* over the bottleneck link, which is shared with attack traffic from hosts *A1* and *A2*. Machine *R1* is an aggregation router and machine *R2* emulates a 10 Mbps link using Click [27]. All physical links are 100 Mbps. We run tcpdump for each user on the link leading from *NAT* to *R1*, anonymizing the output. We use this output to calculate our DoS measures.

The first page displayed to a user is the registration page, with only one button labeled "Register." A click on this button assigns a sequential ID to the user and starts tcpdump; user ratings will be saved under this ID and tcpdump output will bear the name derived from the ID. We next generate a random number in the range 1–4. Number 1 triggers a UDP flood attack on the bottleneck link, number 2 triggers this same attack but at a smaller rate, which aims to degrade but not to deny service. Number 3 triggers a SYN flood attack on the *Content* Web server and number 4 does not trigger any attack, i.e., users in this category form the control group.

The next page is the welcome page, loaded from *Control* server, that explains experiment goals and setup, and gives instructions to the user. Users are asked to click on at least 5 pages of their choice, not to repeat clicks and not to follow external links from the Wikipedia pages. Repeated clicks can lead to erroneous perception of service quality because they display the content from the browser's cache, and external-page clicks bypass the testbed.

The welcome page also displays 21 buttons for the content pages, and a button to quit the survey. Clicking on a content button generates a web page with content on the right and the rating form on the left. If the *Content* server were under attack, the page in the right frame may not load, or it may take a long time to load. Users rate their satisfaction with service quality on 1–4 scale, where 4 means "Excellent", 3 means "Mostly OK", 2 means "Poor, but acceptable" and 1 means "Unacceptable." The rating form is multiple-choice and allows only a single item to be selected. The users can browse naturally: they are allowed to rate their satisfaction at any time; i.e., they did not have to wait for the page to load completely and they did not have to read any content. When the user clicks the "Submit" button on the rating page, the content file name and the rating are saved in a log file. After the rating is submitted, the welcome page is displayed again. The survey ends when the user clicks the "Quit" button on the welcome page. The thank-you page is then displayed and user ratings are shown side by side with our DoS metrics. We describe the process of mapping our measures to the same rating scale as used by human users in Section VI-C.

We experimented with the following attack dynamics before

settling on one of them: (1) The attack starts immediately upon registration and lasts for a long time. This generates predictable results because either all user transactions are affected by the attack or none are. This scenario was too simplistic to validate our metrics; we preferred to have each user experience some good and some poor service. (2) The attack starts when a user requests a content page displaying Wikipedia content. This scenario would be ideal but it had timing problems. A Web request is contained in very few packets sent to server — two to open a TCP connection and one to request a Web page — and only this path is affected by the attack. Request packets are sent rapidly when the user clicks on the content button. If the attack is triggered simultaneously, there was a race condition between creating sufficient congestion and user packets reaching the server. If, on the other hand, we delayed page load until the attack has started, this would affect the user's perception of service quality and skew the rating toward lower values. (3) The attack is triggered upon the registration click. It runs periodically, thus a given user may experience some high quality and some low quality transactions. We opted for this scenario since it was rich enough to generate interesting test cases for our measures and did not suffer from the timing problems present in scenario 2. Each attack starts 60 seconds after the registration click, lasts for 30 seconds and repeats every 60 seconds for total of 10 times. We carefully selected the attack period and duration to maximize the chance that attack traffic overlaps with user requests. The attack is aborted when the user quits the survey.

C. Mapping our DoS Metrics to User-Compatible Ratings

User ratings of service quality are on the scale 1–4, where 2, 3, 4 ratings denote successful transactions with increasing degrees of user satisfaction and 1 denotes failed transactions. Two of our DoS metrics are comparable with user ratings: the transaction success/failure metric and the *QoS* metric. The *QoS* metric is on the scale $(0, 1]$ and is calculated only for successful transactions; we set it to zero for failed transactions. Higher *QoS* values denote higher service quality.

We mapped transaction success/failure and the *QoS* measure into the 1–4 scale as follows (summarized in Table IV). If a transaction failed, our rating of its service quality was set to 1 and its QoS measure was set to 0. If it succeeded, we run an optimization algorithm to find the best values for thresholds on the QoS measure that denote the limits between ratings 2 and 3, and ratings 3 and 4. For our experimental results, these thresholds were 0.87 and 0.88, respectively.

QoS	Rating
0	1
≥ 0 but < 0.87	2
≥ 0.87 but < 0.88	3
≥ 0.88	4

TABLE IV
MAPPING OF *QoS* METRIC TO USER RATING SCALE.

We also had to map transactions into clicks. Our measures are calculated per transaction, which in case of Web service may denote the event of initiating the communication with the server, partially or completely loading a page, or loading each embedded object in a page. Thus one user click usually maps into several transactions. We map clicks into transactions by first identifying TCP connections in the tcpdump output associated with one Web page load, then relating our transactions to these connections (and thus to page loads), and finally pairing the page loads with the user clicks recorded in our rating log file, as explained next.

Identification of TCP connections associated with one page load proceeds as follows: (1) Identify TCP connections in the collected tcpdump file by looking for a 3-way handshake and all subsequent traffic between the same IP addresses and port numbers until either a FIN or a RESET. (2) If a connection contains a packet with an HTTP GET directive in the content field, parse the file name following this directive. For files ending in `.html` this connection denotes a new page load. For other files, look for the `Referer` field in the packet containing HTTP GET, and parse the name of the referring file, which in our case always ends in `.html`. This connection is added to the page load of the referring file. (3) If a connection does not contain a packet with an HTTP GET directive, it is associated with a "NO URL" page load. These connections usually contain a partial or full 3-way handshake, but the service denial was so large that the connection never advanced to data exchange.

Relating our transactions to TCP connections involved selecting the TCP connection that had the same port numbers as the given transaction and encompassed its start and end times. After all transactions were paired with TCP connections and thus with page loads, we calculate the success/failure and the QoS measure for each page load. A load's success/failure measure is a "success" only if all transactions that are mapped to this load were successful, otherwise it is a "failure." A load's *QoS* measure is 0 if its success/failure measure is "failure." Otherwise, the *QoS* measure is the average of *QoS* measures of transactions associated with this page load.

Pairing Web page loads with user clicks from the rating log file was performed by pairing the file names from the loads with URLs in the log file. If we cannot find the name from the log file among our page loads, we next attempt to pair this click with our "NO URL" load based on timing. If this fails, the click is marked invalid. Repeated clicks are also considered invalid because they may be served from a client's cache; an action invisible in network traces.

D. Results

We recruited experiment participants from the following populations: (1) graduate students and faculty at the University of Delaware, (2) graduate students at UCLA, (3) graduate students at Purdue University, (4) attendees of SIGMETRICS 2007, and (5) subscribers of the TCCC mailing list. We kept the survey open for four months (July-October 2007) and had 101 participants and 840 clicks. 32 (3.8%) clicks were invalid, leaving 808 valid clicks for 100 users.

Assignment of users to attack categories was balanced: 23 experienced a UDP flood attack, 28 experienced a low-rate

UDP flood, 29 experienced a SYN flood attack, and 20 were in a control group with no attack being launched. UDP flood is the most severe attack, because it affects all request traffic. SYN flood has lower severity — it only prevents connection setup, but once a user's TCP SYN packet is accepted by the server, communication proceeds normally. We expected that the low-rate UDP flood would have modest to no impact on service quality, and that the best service quality will be assigned to clicks in the control group.

Users who experience poor service usually lose interest in interacting with the server. The average number of clicks per user in different groups was 6.17 for UDP flood, 8.03 for low-rate UDP flood, 7.86 for SYN flood and 10.2 for the control group. These results agree with our expectations and indicate that users in the control group had the best service, followed by users in the low-rate UDP flood group, users in the SYN flood group, and finally users in the UDP flood group.

	Users' ratings						Our ratings					
Attack	1	2	3	4	μ	σ	1	2	3	4	μ	σ
UDP	85	34	12	20	1.78	1.06	107	17	4	23	1.62	1.1
low-UDP	0	11	89	125	3.51	0.59	1	0	33	191	3.84	0.40
SYN	78	27	49	74	2.52	1.26	74	16	16	122	2.81	1.37
No	4	14	95	91	3.34	0.69	1	2	19	182	3.87	0.4

TABLE V
SUMMARY OF RATINGS, MEANS (μ) AND STANDARD DEVIATIONS (σ)

Table V shows the distribution of users' and our ratings. Users' ratings agree with our expectations — satisfaction with service quality was lowest for the UDP flood group, followed by the SYN flood group, and the control and low-rate UDP flood groups experienced the best service quality. Attack groups had some percentage of high ratings (3 and 4) — this is because overlap of an attack with user traffic was random, thus some transactions completed without service denial. Surprisingly, clicks in the low-rate UDP flood group had a slightly higher user rating than clicks in the control group. This was mostly due to more 4 than 3 ratings in the low-rate UDP group, versus the control group. The best explanation we have for this is the subjectivity of human perception of QoS, which was observed in [14] but is not quantified.

Comparing our success/failure metric with user ratings, we had 728 matches, which constitutes 90% of total valid clicks. Out of the 80 mismatches, 40 were cases where we considered a click to be failed while a user gave it a successful rating. These were all cases where an embedded picture in the page did not load quickly enough or at all. Some users did not consider this significant enough for a failure, while we did. Given the high subjectivity of human QoS perception [14], we view a 90% match as proof of high accuracy of our success/failure metric.

Comparing our *QoS* metric with user ratings, both mapped to 1–4 scale, we had 441 matches (54.5% accuracy) and 367 mismatches. If we leave out the 80 mismatches where our success/failure metric disagreed with a user's, of the 287 remaining mismatches, in 240 (29.7% of all clicks) a user assigned a rating 4 (Excellent) and we assigned a 3 (Mostly OK), or vice versa. This is expected, since categories "Excellent" and "Mostly OK" are very similar and humans find it more difficult to distinguish between an excellent and a slightly impaired service, than between an excellent and a poor service. Together, matches and four/three mismatches make 84.2% of all clicks. If we merge the "Excellent" and "Mostly OK" ratings, then, we have 84.2% accuracy of our *QoS* metric, which is fairly high given human subjectivity in rating QoS. Overall, our metrics accurately predicted human perception of service denial (success/failure metric) and service quality (*QoS* metric).

VII. RELATED WORK

A number of computer science fields have developed systematic, standardized approaches to performance measurement. Two examples of this are the TPC benchmarks for application servers and web services [28], and the SPEC benchmarks for a variety of application categories [29]. These efforts adopt a representative workload mix for an application of interest, and a set of performance measures with thresholds that signify success or failure. Measures are calculated at the application level. Our measures are similar to those in TPC and SPEC, but are less diversified, since our inference of application-specific tasks at the network level is difficult and imperfect.

In the quality of service field, there is an initiative, led by the 3GPP partnership, to define a universally accepted set of QoS requirements for network applications [30]. While we reuse many of the specified requirements in our work, we extend, modify and formalize these requirements as explained in Section III-A.

The networking research community has separated applications into categories based on their sensitivity to delay, loss and jitter [31]. That work focuses on providing applications guaranteed service, rather than measuring service denial. The Internet Research Task Force Transport Modeling Research Group (TMRG) discussed using user-based QoS metrics for measuring congestion, but did not specify such metrics in any detail [32].

In [33], the authors measure DoS impact on real-world traffic via the distributions of several parameters: the throughput of FTP applications, round-trip times of FTP and Web flows, and latency of Web flows and the DNS lookup service in real world traces before, during, and after an attack. Our paper strives to define a more formal threshold-based model for these and other parameters that can be extended to a broader variety of services and attacks. In [7], the authors measure the percentage of "completed" transactions, which may appear similar to our *pft* metric and our "successful" transactions. However, their transactions count as success if they complete at all during the simulation, however late, while our transactions are successful only if they complete *while meeting all their QoS requirements*. Our metric thus more accurately captures user perception of service quality.

In [34], a user satisfaction index is computed from Skype traces and validated via analysis of other call characteristics, such as conversation interactivity. Our work provides a framework where this index can be easily incorporated into a

DoS metric for Skype and other VoIP traffic. In slow-motion benchmarking [35], the authors use network traces collected at the client to measure performance of thin clients. Their only performance measure is the sum of transaction durations in the benchmark.

VIII. Conclusions and Future Work

One cannot understand a complex phenomenon like denial of service without being able to measure it in an objective, accurate way. The work described here represents the first attempt to define accurate, quantitative and versatile metrics for measuring effectiveness of denial of service attacks and defenses. By focusing on the issue of measuring human user perception of application-level service quality, the metrics cut to the heart of the problem and avoid issues of the specific form of the attack and legitimate traffic mix. Our approach is objective, reproducible, and applicable to a wide variety of attack and defense methodologies. Its value has been demonstrated in both testbeds and simulation environments. Further, we have addressed the main concern of metrics that focus on an application-level phenomenon – the accuracy of the metric compared to human perceptions – via tests with human subjects that validated our results.

Our metrics are usable by other researchers in their own work. They offer the first real opportunity to compare and contrast different denial of service attacks and defenses on an objective head-to-head basis. We expect that this work will advance denial-of-service research by providing a clear measure of success for any proposed defense, and helping researchers gain insight into strengths and weaknesses of their solutions.

While our DoS metrics are a necessary condition for performance comparison of DoS defenses, they are not sufficient. A related problem is devising standardized benchmarks for DoS defense testing, so all products are tested under the same conditions. We have done some pioneering work in this area [36] but ours is just a first, small step and an engagement of a wider research community is needed to completely address this problem.

References

[1] A. Yaar, A. Perrig, and D. Song. SIFF: A Stateless Internet Flow Filter to Mitigate DDoS Flooding Attacks. In *Proceedings of the IEEE Security and Privacy Symposium*, 2004.

[2] A. Kuzmanovic and E. W. Knightly. Low-Rate TCP-Targeted Denial of Service Attacks (The Shrew vs. the Mice and Elephants). In *Proc. of ACM SIGCOMM*, August 2003.

[3] M. Guirguis, A. Bestavros, and I. Matta. Exploiting the Transients of Adaptation for RoQ Attacks on Internet Resources. In *Proceedings of ICNP*, Oct 2004.

[4] CERT CC. CERT Advisory CA-1996-21 TCP SYN Flooding and IP Spoofing Attacks. http://www.cert.org/advisories/CA-1996-21.html, 1996.

[5] Srikanth Kandula, Dina Katabi, Matthias Jacob, and Arthur Berger. Botz-4-Sale: Surviving Organized DDoS Attacks that Mimic Flash Crowds. In *NSDI*, 2005.

[6] Hani Jamjoom and Kang Shin. Persistent Dropping: A Efficient Control of Traffic Aggregates. In *ACM SIGCOMM Conference*, 2003.

[7] X. Yang, D. Wetherall, and T. Anderson. A DoS-limiting Network Architecture. In *ACM SIGCOMM Conference*, 2005.

[8] Ratul Mahajan, Steven M. Bellovin, Sally Floyd, John Ioannidis, Vern Paxson, and Scott Shenker. Controlling high bandwidth aggregates in the network. In *ACM Computer Communication Review*, July 2001.

[9] Angelos Stavrou, Angelos D. Keromytis, Jason Nieh, Vishal Misra, and Dan Rubenstein. MOVE: An End-to-End Solution to Network Denial of Service. In *NDSS*, 2005.

[10] G. Oikonomou, J. Mirkovic, P. Reiher, and M. Robinson. A Framework for Collaborative DDoS Defense. In *Proceedings of ACSAC*, December 2006.

[11] Cooperative Association for Internet Data Analysis. CAIDA Web page. http://www.caida.org.

[12] WIDE Project. MAWI Working Group Traffic Archive. http://tracer.csl.sony.co.jp/mawi/.

[13] 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.

[14] Nina Bhatti, Anna Bouch, and Allan 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.

[15] L. Yamamoto and J. G. Beerends. Impact of network performance parameters on the end-to-end perceived speech quality. In *In Proceedings of EXPERT ATM Traffic Symposium*, September 1997.

[16] 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 *In Proceedings of ACM NetGames 2004*.

[17] Nathan Sheldon, Eric Girard, Seth Borg, Mark Claypool, and Emmanuel Agu. The Effect of Latency on User Performance in Warcraft III. In *In Proceedings of ACM NetGames 2003*.

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

[19] 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.

[20] J. Mirkovic, A. Hussain, B. Wilson, S. Fahmy, P. Reiher, R. Thomas, W. Yao, and S. Schwab. Towards User-Centric Metrics for Denial-Of-Service Measurement. In *In Proceedings of the Workshop on Experimental Computer Science*, June 2007.

[21] 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 TridentCom Conference*, March 2006.

[22] D. J. Bernstein. TCP syncookies. http://cr.yp.to/syncookies.html.

[23] The Network Simulator ns 2. NS-2 Web page. http://www.isi.edu/nsnam/ns/.

[24] R. Chertov, S. Fahmy, and N. Shroff. Emulation versus Simulation: A Case Study of TCP-Targeted Denial of Service Attacks. In *Proceedings of the 2nd International IEEE CreateNet TridentCom Conference*, February 2006.

[25] Wikipedia, the Free Encyclopedia. http://www.wikipedia.com.

[26] University of Utah. Emulab testbed. http://www.emulab.net.

[27] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. The Click Modular Router. *ACM Transactions on Computer Systems*, 18(3):263–297, August 2000.

[28] Transaction Processing Performance Council. TPC Benchmarks. http://www.tpc.org/information/benchmarks.asp.

[29] Standard Performance Evaluation Corporation. SPEC Benchmarks and Published Results. http://www.spec.org/benchmarks.html.

[30] 3GPP. *The 3rd Generation Partnership Project (3GPP)*.

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

[32] IRTF TMRG group. The Transport Modeling Research Group's Web Page. http://www.icir.org/tmrg/.

[33] Kun chan Lan, Alefiya Hussain, and Debojyoti Dutta. The Effect of Malicious Traffic on the Network. In *Passive and Active Measurement Workshop (PAM)*, April 2003.

[34] Kuan-Ta Chen, Chun-Ying Huang, Polly Huang, and Chin-Laung Lei. Quantifying Skype User Satisfaction. In *Proceedings of the ACM SIGCOMM*, September 2006.

[35] Jason Nieh, S. Jae Yang, and Naomi Novik. Measuring Thin-Client Performance Using Slow-Motion Benchmarking. *ACM Transactions on Computer Systems*, 21(1), February 2003.

[36] J. Mirkovic, S. Wei, A. Hussain, B. Wilson, R. Thomas, S. Schwab, S. Fahmy, R. Chertov, and P. Reiher. DDoS Benchmarks and Experimenter's Workbench for the DETER Testbed. In *Proceedings of Tridentcom*, 2007.

Dr. Jelena Mirkovic is a Computer Scientist at the USC Information Sciences Institute, which she joined in 2007. Prior to this she was an Assistant Professor at the Computer and Information Sciences Department, University of Delaware, 2003-2007. She received her M.S. and Ph.D. from UCLA, and her B.S. in Computer Science and Engineering from the School of Electrical Engineering, University of Belgrade, Serbia. Her current research is focused on accountability, safe sharing of network data, denial-of-service attacks, and IP spoofing. Her research is funded by the National Science Foundation and the Department of Homeland Security.

Dr. Alefiya Hussain is a Senior Principal Scientist at Sparta Inc. Her research interests include statistical signal processing, protocol design, security, and network measurements. She received a Bachelor of Engineering degree from Pune Institute of Computer Technology, and a M.S. and Ph.D. in Computer Science from the University of Southern California. She is a member of ACM and Upsilon Pi Epsilon.

Dr. Sonia Fahmy is an Associate Professor at the Computer Science department at Purdue University. She received her PhD degree from the Ohio State University in 1999. Her current research interests lie in the areas of Internet tomography, network security, and wireless sensor networks. She received the National Science Foundation CAREER award in 2003, and the Schlumberger technical merit award in 2000. She is a member of the ACM. For more information, please see: http://www.cs.purdue.edu/∼fahmy/

Dr. Peter Reiher received his B.S. in Electrical Engineering and Computer Science from the University of Notre Dame in 1979. He received his M.S. and Ph.D. in Computer Science from UCLA in 1984 and 1987, respectively. He has done research in the fields of distributed operating systems, security for networks and distributed computing, file systems, optimistic parallel discrete event simulation, ubiquitous computing, naming issues in distributed systems, active networks, and systems software for mobile computing. Dr. Reiher is an Adjunct Associate Professor in the Computer Science Department at UCLA.

Dr. Roshan Thomas is a Senior Principal Scientist at Sparta, Inc. He has over thirteen years of experience as a researcher at the Principal Investigator level in various aspects of computer security including access control models, network security, policy languages, secure distributed database management and multilevel-secure object-oriented distributed computing. As part of the research into cross domain solutions for the Collaborative Technology Alliance (CTA), he is developing policy languages for cross domain information release. He is also currently a PI on an IARPA-funded project called TDOC that is looking at advanced dissemination controls models to prevent insider leaks in the computer systems for the intelligence community. In the past, he has been a Principal Investigator and researcher on a variety of DARPA and NSF-funded research projects that investigated active and collaborative access control models, network security, and security in mobile ad-hoc networks. Dr. Thomas served as the co-founder of the First IEEE International Workshop on Pervasive Computing and Communication Security (PerSec 2004) and served as the PC co-chair for the second workshop (PerSec 2005). Dr. Thomas holds bachelor's and master's degrees in Computer Science and earned a Ph.D in Information Technology with a specialization in computer security from George Mason University, Fairfax, VA, U.S.A in May 1994.