PacketScore: Statistical-based Overload Control against Distributed Denial-of-Service Attacks

Yoohwan Kim[*], *Wing Cheong Lau*[**], *Mooi Choo Chuah*[**] *and Jonathan H. Chao*[***]

* EECS Department
Case Western Reserve University
Cleveland, Ohio

** Bell Labs,
Lucent Technologies
Holmdel, New Jersey

*** ECE Department
Polytechnic University,
Brooklyn, New York

Abstract **-- Distributed Denial of Service (DDoS) attack is a critical threat to the Internet. Currently, most ISPs merely rely on manual detection of DDoS attacks after which offline fine-grain traffic analysis is performed and new filtering rules are installed manually to the routers. The need of human intervention results in poor response time and fails to protect the victim before severe damages are realized. The expressiveness of existing filtering rules is also too limited and rigid when compared to the ever-evolving characteristics of the attacking packets. Recently, we have proposed a DDoS defense architecture that supports distributed detection and automated on-line attack characterization. In this paper, we will focus on the design and evaluation of the automated attack characterization, selective packet discarding and overload control portion of the proposed architecture. Our key idea is to prioritize packets based on a per-packet score which estimates the legitimacy of a packet given the attribute values it carries. Special considerations are made to ensure that the scheme is amenable to high-speed hardware implementation. Once the score of a packet is computed, we perform score-based selective packet discarding where the dropping threshold is dynamically adjusted based on (1) the score distribution of recent incoming packets and (2) the current level of overload of the system.**

Keywords— System design, Simulations, Denial-of-Service Attack, Security, Overload Control, Selective Packet Discarding, Traffic characterization.

I. MOTIVATION

One of the major threats to cyber security is Distributed Denial-of-Service (DDoS) attack in which the victim network element(s) are bombarded with high volume of fictitious, attacking packets originated from a large number of machines. The aim of the attack is to overload the victim and render it incapable of performing normal transactions. DDoS attacks can be categorized into *end-point attacks* and *infrastructure attacks*. In an end-point attack, the victim can be an individual end-host or, more typically, an entire customer stub-network served by an Internet Service Provider (ISP). In an infrastructure attack, high volume of attacking packets are forced through a port of an ISP router to create one or more choke-points within the ISP infrastructure based on the knowledge of the routing pattern within the domain. Currently, most ISPs merely rely on manual detection of DDoS attacks. Once an attack is reported, an offline fine-grain traffic analysis is performed by a subject-matter expert to identify and characterize the attacking packets. New filtering rules/ access control list are then constructed and installed manually to the routers according to the outcome of attack characterization. The need of human intervention results in poor response time and fails to protect the victim before severe damages are realized. This procedure also lacks adaptability and renders the system vulnerable towards fast-varying DDoS attacks. Further, the expressiveness of existing rule-based filtering is too limited as it requires an explicit specification of all types of packets to be discarded. As the difference between legitimate and attacking packets become increasingly subtle, the number of required filtering rules as well as the number of packet attributes included in each rule explode. Increase in rule-set complexity also poses serious scalability problems for high-speed implementation of rule-based filtering.

Recently, the DDoS problem has attracted much attention from the research community. So far, the focus has been on the design of traffic marking and traceback protocols [Be01, Pa01, Sa01, Sn01] which enable downstream routers to determine and notify the upstream routers of the attacking packets. Most of the work emphasizes the backward compatibility of protocol support for traceback under the existing Internet infrastructure. Once the upstream sources of the attack have been identified, proposed pushback mechanisms [Io02, Ya02] are used to contain the damage of the attack. However, the effectiveness of such an approach is contingent upon the ability to extract a precise characterization of the attacking packets. Without such characterization, the legitimate traffic within the suspicious flows will be equally affected by the pushback mechanism. While there has been recent work by the data-mining research community to recognize intrusion patterns using offline machine-learning approaches [Le98, Ma99], these schemes are mostly offline-oriented. An exception to this trend is the D-WARD approach [Mi02], which does perform limited statistical traffic profiling at the edge of the networks to perform online detection of new types of DDoS attacks.

Corresponding Authors: yoohwan@ieee.org, lau@bell-labs.com

By monitoring the nominal per-destination type traffic arrival and departure rate of TCP, UDP, ICMP packets, as well as any abnormal asymmetrical behavior of the two-way traffic at the edge router connecting to a stub-network, D-WARD aims at stopping DDoS attacks near their sources, i.e., the ingress routers. While such "source-side" tackling approach is attractive in terms of having less demanding operating-speed and scalability requirements, its viability hinges on the voluntary cooperation of majority of ingress network administrators internet-wide. In theory, one can circumvent this deployment problem by applying the D-WARD approach to the backbone network. However, in order to realize such a backbone approach, one must address the key scalability issues such as the large number of targets required to be protected and the high operating speed within the backbone network. This is indeed the emphasis of our proposed scheme. There are also a small set of commercial products [Mazu, Rive] which advertise limited support of statistics-based adaptive filtering techniques. However, most of these solutions do not fully automate packet differentiation or filter enforcement. Instead, they only recommend a set of binary filter rules to the network administrator to be installed in their routers or firewalls. The recommended rule set is often too complex to be comprehensible, let alone to be debugged or modified. The technical details of their statistics-based adaptive filtering schemes are not available to the public. The performance of the schemes, especially in terms of scalability and impact on legitimate traffic is not clear either. The situation is well summarized by a quote from a recent article on anti-DoS device review [Fo01]:

> *"In the end, we felt as though we were left playing Russian roulette when it came to installing the recommended filters."*

The rest of this paper is organized as follows: in Section II, we provide an overview of the entire PacketScore DDoS defense architecture. In Section III, we focus on the design and implementation of the intelligent packet differentiation, selective discarding and overload control portion of our proposal, which is the main subject of this paper. In particular, we will concentrate on a standalone implementation of these schemes, which is directly applicable for protecting infrastructure DDoS attacks. Due to limited space, the details of their distributed implementation are beyond the scope of this paper, and will be the subject of a sequel of this paper. In Section IV, we evaluate the performance of the standalone packet differentiation/ discarding scheme. The paper is concluded in Section V with a list of future investigation directions.

II. OVERVIEW OF THE PACKETSCORE APPROACH

Recently, we have proposed adefense scheme based on distributed detection and automated on-line attack characterization [La03]. The proposed scheme consists of the following 3 phases:

- **D**etect the onset of an attack and identify the victim by monitoring four key traffic statistics of each protected target while keeping minimum per-target states.
- **D**ifferentiate between legitimate and attacking packets destined towards the victim based on a readily-computed, Bayesian-theoretic metric of each packet. The metric is the so-called "Conditional Legitimate Probability" (CLP).
- **D**iscard packets selectively by comparing the CLP of each packet with a dynamic threshold. The threshold is adjusted according to (1) the distribution of CLP of all suspicious packets and (2) the congestion level of the victim.

We name our scheme the PacketScore approach because CLP can be viewed as a score which estimates the legitimacy of a suspicious packet. By taking a score-based filtering approach, we avoid the problems of conventional binary rule-based filtering discussed in Section I. The score-based approach also enables the prioritization of different types of suspicious packets. It is much more difficult, if not impossible, for rule-based filtering to support such prioritization. The ability to prioritize becomes even more important when a full characterization of the attacking packets becomes infeasible. By linking the CLP discard threshold to the congestion level of the victim, our approach allows the victim system to opportunistically accept more potentially legitimate traffic as its capacity permits. In contrast, once a rule-based filtering scheme is configured to discard a specific type of packets, it does so regardless of the victim utilization.

For end-point attacks, we employ a scalable, distributed attack detection process using Bloom filter/ leaky bucket arrays (BFLBA) similar to those proposed by [Fe01, Es02] to monitor key traffic statistics of each protected target. The BFLBA's allow us to simultaneously monitor such statistics for a large number of protected targets while keeping minimal per-target state information. Distributed attack detection is realized via a DDoS control server (DCS) which correlates and consolidates

possible incidents reported by routers residing along a security perimeter. We refer such routers as Detecting-Differentiating-Discarding routers (3D-R). Once an attack victim is identified, the 3D-Rs collaborate with the DCS to perform a distributed, online characterization of the attacking traffic by comparing the fine-grain characteristics of the suspicious traffic with a nominal traffic profile of the victim. The result enables each 3D-R to compute a "score", i.e., the CLP, for each suspicious packet at wire-speed which ranks the likelihood of the packet being an attacking packet, given the attribute values it carries, using a Bayesian-theoretic approach. Based on a dynamic thresholding mechanism against such score, the 3D-Rs perform selective packet discarding and overload control for the victim in a distributed manner. The DCS coordinates this distributed overload control process by adjusting the threshold dynamically based on the arrival rate of suspicious traffic and score distributions reported by different 3D-Rs. Fig. 1 depicts the support of distributed detection and overload control by a set of 3D-Rs and DCSs.

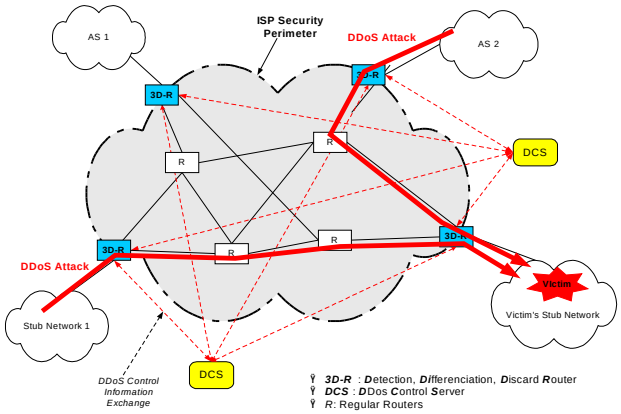

Figure 1: Deployment of 3D-Rs and DCSs to tackle DDoS Attacks

From here onwards, we focus the design and implementation of the intelligent packet differentiation, selective discarding and overload control portion of our proposal, which is the main subject of this paper. In particular, we will concentrate on a standalone implementation of these schemes, which is directly applicable for protecting infrastructure DDoS attacks.

III. DETAILED PACKETSCORE METHODOLOGIES

In this section, we first discuss the design issues as well as implementation details related to the packet differentiation, selective packet discarding and overload control the proposed scheme.

A. Packet Differentiation via Fine-grain Traffic Profile Comparison

Once a DDoS attack is detected, the next step is to distinguish the attacking packets from the legitimate ones amongst the suspicious traffic. Our approach is to perform online profiling of the suspicious traffic and compare the findings with the nominal traffic profile of the victim. The viability of this approach is based on the premise that there are some traffic characteristics that are inherently stable during normal network operations of a target network, in the absence of DDoS attacks.

A disproportional increase in the relative frequency of a particular packet attribute value is an indication that the attacking packets also share the same value for that particular attribute. The greater the disproportional increase, the stronger the indication. The more "abnormal" attribute values a packet possesses, the higher the probability that the packet is an attacking packet. For example, if it is found via that the suspicious packets contain abnormally high percentage of (1) UDP packets and (2) packets of size *S* and (3) packets with TTL value *T*, then UDP packets of size *S* and TTL value *T* destined to the DDoS victim should be treated as prime suspects and given lower priority upon selective packet discarding during overload.

Candidate packet attributes considered to be used for traffic profiling include: the marginal distributions of the *fraction*[1] of recently arrived packets having various (1) IP protocol-type values, (2) packet size, (3) *server*[2] port numbers, (4) source/ destination IP prefixes[3], (5) Time-to-Live (TTL) values, (6) IP/TCP header length[4], (7) TCP flag patterns. We are also interested in the fraction of packets which (8) use IP fragmentation and (9) bear incorrect IP/TCP/UDP checksums. It is worthwhile to consider the joint distribution of the fraction of packets having various combinations of (10) TTL value and source IP prefix, (11) packet-size and

[1] Profiling against relative frequency of different attribute values (instead of absolute packet arrival rates) helps to alleviate the difficulties caused by the expected fluctuation of nominal traffic arrival rates due to time-of-the-day and day-of-the-week behavior.

[2] We employ the heuristics of taking the server port number to be the minimum of the source and destination port numbers carried by the packet. This eliminate the need of identifying whether the packet is client-bound or server-bound. Also, since client port number is usually selected in random by the client operating system, it does not meet the "invariant" criteria to be used for profiling.

[3] In our study, we have used the 16-bit IP prefix as an approximation of the IP subnet. In practice, we can extract the actual prefix-length of the subnet from routing tables and/or route-server databases.

[4] This is to detect possible abuse of IP/TCP options.

protocol-type, (12) server port number and protocol-type, as well as (13) source IP prefix and TTL value.

To validate our claim of the relatively "invariant" nature of the distribution of the above packet attributes, we have conducted extensive statistical analysis on real-life Internet traces collected from the traffic archive of the WIDE-project [WIDE]. Fig. 2(a)-(d) show the time variation of the distribution of various packet attributes values observed from a moderately loaded wide area network link. For each attribute, the relative frequency of its values are computed every 10 minutes for the period between May 10, 1999 8:00pm and May 11, 2:00pm for a total of 108 non-overlapping periods. Fig. 2(a) shows the time-variation of the distribution of TTL values. In particular, the ends of the error-bar correspond to the maximum and minimum fraction observed for the given TTL value over the aforementioned 18-hour interval and the black-dot represents the average. The corresponding time-varying distributions for protocol-type, packet-size, TCP-flag pattern, server port number and 16-bit source IP prefix are shown in Fig. 2 (b)-(f) respectively. Notice from Fig. 2 that while the fraction of an attribute value does vary over the 18-hour period, the variation is always within a few percentage of the total number of packets arrived over a 10-minute window. Due to the overwhelming volume of DDoS attack packets compared to normal ones, the formers are expected to increase the fraction of particular attribute values they carry by more than a few percentage and change the overall distribution substantially. Furthermore, the variability of nominal attribute value distribution can be substantially reduced if hourly time-of-the-day profile are used. One may argue that it is relatively straightforward for a sophisticated attacker to learn the approximated distribution of some attributes, e.g. protocol-type, TCP-flag pattern and packet-size, based on publicly available data on Internet traffic characteristics, and thus be able to generate the attribute distributions for the attacking packets accordingly to circumvent our profile-based differentiation scheme. Note however that distributions of other attributes such as TTL and source IP-prefixes, and to a lesser extent, server-port distribution, are expected to be site-dependent (or link/port dependent) and thus more difficult for an outside attacker to collect such information. For instance, it is quite difficult for an outsider to determine the joint-distribution of source-IP-prefix and the TTL value for a given site. As long as there exists profiling information which is known only to the site/network-operator but not to the attacker, our scheme can use it as the information edge to differentiate among attacking and legitimate packets.

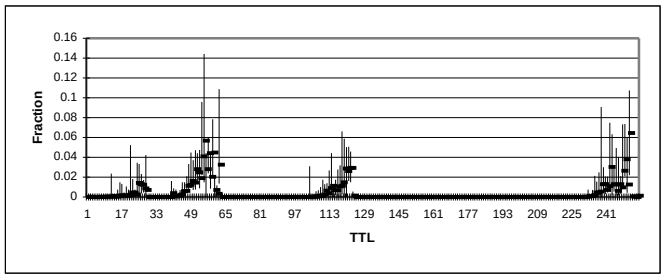

Figure 2 (a): Time variation of TTL value distribution

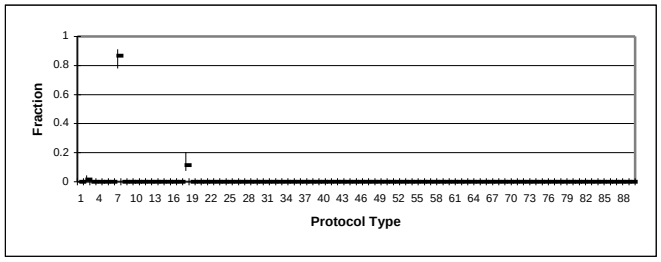

Figure 2 (b): Time variation of Protocol Type distribution

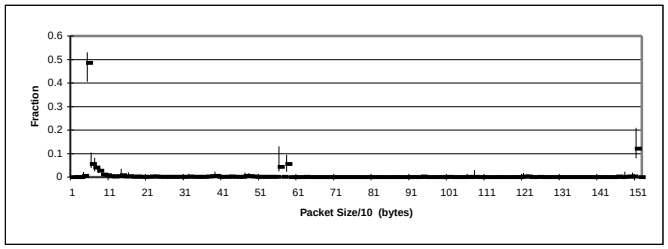

Figure 2 (c): Time variation of Packet-size distribution

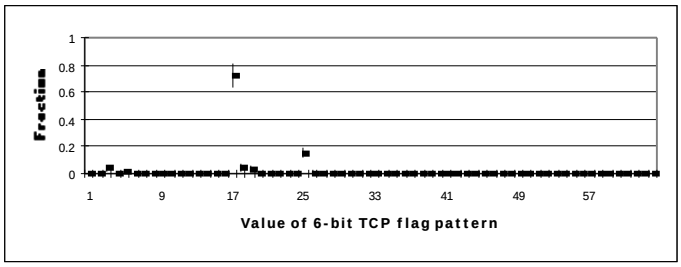

Figure 2 (d): Time variation of 6-bit TCP flag pattern distribution (e.g. SYN = 000010 = 2 ; ACK = 010000 = 16)

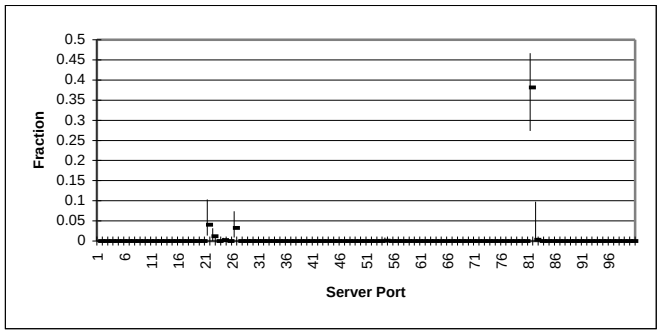

Figure 2 (e): Server Port distribution

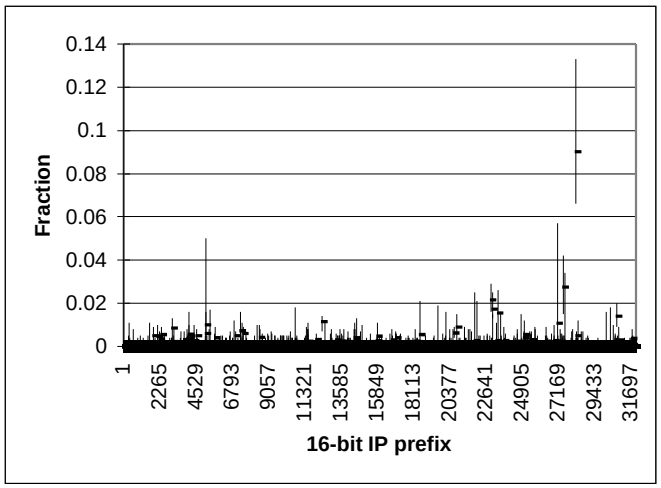

Figure 2 (f): Source IP prefix distribution

Figure 2: Time Variation of Packet Attribute Values Distribution

1. *Conditional Legitimate Probability*

In this section, we formalize the notion of ***conditional legitimate probability*** of a suspicious packet which measures the likelihood of the packet being a legitimate (instead of an attacking) one given the attribute values it possesses. Consider all the packets destined towards a DDoS attack target. Each packet carries a set of discrete-valued attributes *A, B, C, ...* For example, *A* can be the protocol-type, *B* can be the packet-size, *C* can be the TTL values etc. Let $JP_n(A,B,C,\cdots)$ be the joint probability mass function of attribute values under normal operations. The probability of a legitimate (attacking) packet having values *a, b, c, ...* for attributes *A, B, C,...,* is given by $JP_n(A=a,B=b,C=c,\cdots)$ (and $JP_a(A=a,B=b,C=c,\cdots)$ respectively). Similarly, we use $JP_m(A,B,C,\cdots)$ to denote the joint probability mass function of packet attributes measured during an attack. Define the conditional legitimate probability (*CLP*) of packet p as:

$$CLP(p) = \text{Prob}(p \text{ is a legitmate packet} \mid \text{Attributes } A, B, C, \ldots \text{ of packet } p \text{ are equal to } a_p, b_p, c_p, \cdots, \text{respectively})$$

Assume that there are N_m packets in total within a measurement interval among which N_n are from legitimate ones, and N_a are attacking ones. Using standard Bayesian argument, we have:

$$CLP(p) = \frac{N_n \cdot JP_n(A=a_p,B=b_p,C=c_p,\cdots)}{N_n \cdot JP_n(A=a_p,B=b_p,C=c_p,\cdots)+N_a \cdot JP_a(A=a_p,B=b_p,C=c_p,\cdots)}$$

$$= \frac{N_n \cdot JP_n(A=a_p,B=b_p,C=c_p,\cdots)}{N_m \cdot JP_m(A=a_p,B=b_p,C=c_p,\cdots)}$$

$$= \frac{r_n}{r_m} \cdot \frac{JP_n(A=a_p,B=b_p,C=c_p,\cdots)}{JP_m(A=a_p,B=b_p,C=c_p,\cdots)} \quad \text{.....Eq.(1)}$$

where r_n (r_m) is the nominal (currently measured) utilization of the system, respectively. Here we have used r_n / r_m to estimate N_n / N_m. Observe that, since r_n / r_m is constant for all packets within the same observation period, one can even ignore its contribution when comparing and prioritizing packets based on their CLP values as long as the packets arrive within the same observation period. If we assume the attributes to be independent of each other, Eq.(1) can be rewritten as,

$$CLP(p) = \frac{r_n}{r_m} \cdot \frac{P_n(A=a_p)}{P_m(A=a_p)} \cdot \frac{P_n(B=b_p)}{P_m(B=b_p)} \cdot \frac{P_n(C=c_p)}{P_m(C=c_p)} \cdots, \quad \text{.....Eq.(2)}$$

where $P_n(X)$ ($P_m(X)$) is the marginal probability mass function of packet attribute *X* under nominal (currently measured) traffic conditions, respectively. To achieve a compromise between profile storage requirement and the need to capture important inter-attribute dependency, we use joint distribution(s) for the strongly-correlated attributes while using marginal distribution(s) for the remaining ones. The CLP is therefore expressed in the form of a product of marginal and joint probability mass function values. In Section IV, we will compare the performance impact and storage requirements for different combinations of marginal/ joint distributions.

2. *Variation of Nominal Profiles*

In the above formulation, we have assumed that the nominal profiles, i.e., $JP_n(A,B,C,\cdots)$ and $P_n(X)$'s are constant for ease of illustration. In general, the nominal traffic profile is a function of time which exhibits periodical time-of-the-day, e.g., diurnal, day-of-the-week variations as well as long term trend changes. While long-term profile changes can be handled via periodical re-calibration using standard time-series forecast and extrapolation techniques [Br00], the daily or weekly variation between successive re-calibration may require time-of-the-day, day-of-the-week specific traffic profiles. To reduce storage and maintenance requirement of a large set of time-specific nominal

profiles, our approach is to use a high percentile, say 95-percentile, of the fraction of each attribute value observed amongst the multiple time-of-the-day nominal profiles as the corresponding reference value. In Section IV, we will investigate the performance impact due to inherent variation of nominal traffic profile.

3. *Managing Nominal Traffic Profiles using Iceberg-style Histograms*

We expect that a nominal traffic profile of each target to be consisted of a set of marginal and joint distributions of various packet attributes. This profiling information will be stored in the form of normalized histograms of one or higher dimensions. Due to the number of attributes to be incorporated in profile (in the order of ten or more) and the large number of possible values of each attribute (as much as tens of thousands or more, e.g., in the case of possible source IP prefixes), an efficient data structure is required to implement such histograms. This is particularly important for the case of distributed overload control because traffic profiles have to be exchanged between the 3D-Rs and the DCS. Towards this end, we propose to use *iceberg-style* histograms [Ba02]. By "iceberg-style", it means that the histogram only includes those entries in the population which appear more frequently than a preset percentage threshold, say *x%*. This guarantees that there are no more than *100/x* entries in the histogram. For entries which are absent from the iceberg-style histogram, we will use the upper bound, i.e., *x%* as their relative frequency. Due to the vast dimensions of joint distribution functions, an iceberg-style implementation is particularly important. With iceberg-style histograms, a fine-grain per-target profile can be kept to a manageable size. As we will demonstrate in Section IV, in practice, most packet attributes are dominated by a small set of attribute values. As such, the actual number of non-null values, the so-called number of icebergs, in the corresponding iceberg histograms are much smaller than the maximum bound given above. More importantly, one-pass iceberg-style histogram maintainance/ updates can be implemented efficiently in hardware, e.g. by applying a two-stage pipelined approximation of the scheme proposed in [Ka03]. Tradeoffs between iceberg-threshold, histogram storage requirement and packet differentiation performance are discussed Section IV.

To handle infrastructure attacks, each 3D-R stores and maintains the nominal traffic profile of each of its egress ports. Since there is a limited number of ports per 3D-R, this should not be an excessive burden. For the case of end-point attacks, a large number of nominal profiles, namely, one per protected target, has to be stored and maintained. By having a DCS to coordinate distributed fine-grain traffic profiling, the maintenance of per-target nominal profiles is offloaded to the DCS. Upon the detection of an end-point attack, each 3D-R simply measures the fine-grain profile of traffic destined towards the victim and forwards the local measurements to the DCS for aggregation and comparison with their nominal counterparts. In fact, the same distributed measurement and aggregation mechanism is used to establish the nominal traffic profile of each end-point target at during initial and periodical calibrations[5]. The management of nominal profiles of different target end-points within a domain can be further partitioned among multiple DCSs for enhanced scalability.

4. *Real-time Traffic Profiling and Per-packet CLP Computation*

According to Eqs. (1) and (2), the real-time per-packet processing of a naive implementation of the CLP computation seems formidable: The current packet attribute distributions have to be updated as a result of the arriving packet. The CLP for the incoming packet can be computed only after the packet attribute distributions have been updated. To make wire-speed per-packet CLP computation possible, we decouple the update of packet attribute distribution from that of CLP computation to allow CLP computation and packet attribute distribution to be conducted in parallel, but at different time-scales. With such decoupling, the CLP computation is based on a snapshot of "recently" measured histograms while every packet arrival (unless additional sampling is employed) will incur changes to the current packet attribute histograms. To be more specific, a frozen set of recent histograms is used to generate a set of "scorebooks" which maps a specific combination of attribute values to its corresponding "score". The scorebooks are updated periodically in a time-scale longer than the per-packet arrival time-scale, or upon detection of significant change of the measured traffic profile. By assuming attribute independence and using the logarithmic version of Eq. (2) as shown below,

[5] In practice, an ISP may choose to perform comprehensive nominal traffic profiling for the set of "premium-paying" stub-networks only. For the rest of the end-points, the ISP may choose their profiles from a set of standard templates based on their business nature, ingress access speed as well as their size.

$$\log[CLP(p)] = \left\{ \begin{array}{l} [\log(r_n) - \log(r_m)] + \\ [\log(P_n(A=a_p) - \log(P_m(A=a_p)] + \\ [\log(P_n(B=b_p)) - \log(P_m(B=b_p))] + \\ [\log(P_n(C=c_p)) - \log(P_m(B=b_p))] + \ldots \end{array} \right\} \ldots\ldots\ldots\ldots\ldots Eq.(3)$$

we can construct a scorebook for each attribute that maps different values of the attribute to a specific partial score. For instance, the partial score of a packet with attribute A equal to a_p is given by $[\log(P_n(A=a_p) - \log(P_m(A=a_p)]$. According to Eq.3, we can sum up the partial scores of different attributes to yield the logarithm of the overall CLP of the packet. This scorebook approach enables hardware-based computation of per-packet CLP by replacing numerous floating-point multiplications and divisions in Eq.(2) with simple additions and table lookups. This scorebook approach can be readily extended to handle nominal profiles which contain of a mixture of marginal and joint packet attribute distributions. Of course, the scorebook for a multiple-attribute joint-distribution will be larger. The size of the scorebook can be further reduced by adjusting (1) the iceberg threshold and (2) quantization steps of the score.

B. Selective Packet Discarding and Overload Control

Once the CLP is computed for each suspicious packet via fine-grain real-time traffic profiling, selective packet discarding and overload control can be conducted using CLP as the differentiating metric. The key idea is to prioritize packets based on their CLP values. Since an exact prioritization would require offline, multiple-pass operations, e.g., sorting, we take the following alternative approach to realize an online, one-pass operation: First, we maintain the cumulative distribution function (CDF) of the CLP of all incoming suspicious packets using one-pass quantile computation techniques described in [Ch00,Gr01][6]. We then discard a suspicious packet if its CLP value is below a dynamically adjusted threshold[7]. If there is a need to guarantee certain minimum throughput for particular types of packets, we can incorporate such "immunity" rules by artificially boosting the scores of a given portion of these specific types of packets.

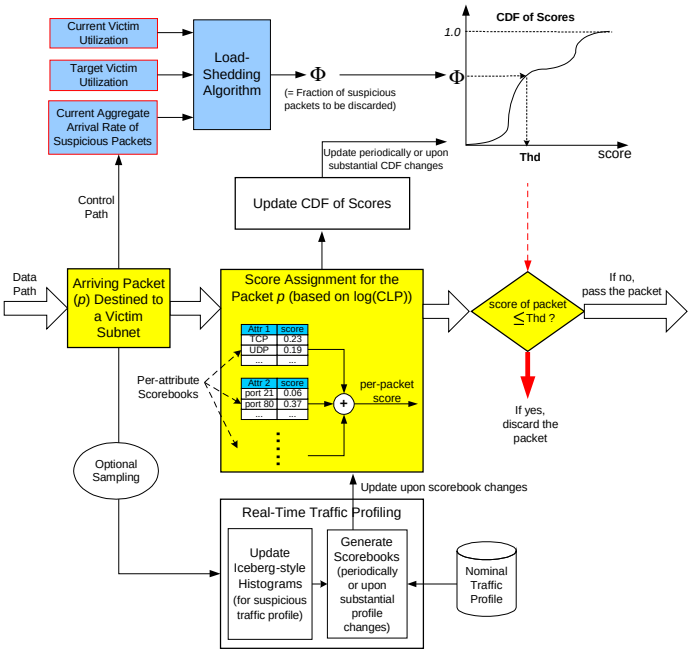

Figure 3: Packet Differentiation and Overload Control

Fig. 3 depicts the integrated operation between CLP computation and the determination of dynamic discarding threshold for CLP. First, a load-shedding algorithm, such as those described in [Ka01], is used to determine the fraction (Φ) of arriving suspicious packets required to be discarded in order to control the utilization of the victim to be below a target value. Typical inputs to a load-shedding algorithm include: current utilization of the victim, maximum (target) utilization allowed for the victim as well as the current aggregated arrival rate of suspicious traffic. (See Appendix I for a description of the actual load-shedding algorithm used in PacketScore.) Once the required packet-discarding percentage, Φ , is determined, the corresponding CLP discarding threshold, Thd, is looked up from a recent snapshot of the CDF of the CLP values of all suspicious packets. The use of a snapshot version of the CDF (instead of the most up-to-date one) eliminates possible race-conditions between discarding threshold updates and CDF changes upon new packet arrivals. The snapshot is updated periodically or upon significant changes of the packet score distribution. The adjustment of the CLP discarding threshold, as well as the load-shedding algorithm, are expected to operate at a time-scale which is considerably longer than the packet

[6] Comparing to a score CDF representation using constant width score-buckets, a 100-quantile score CDF representation works much better due to the unpredictable spacing of packet scores in advance. Also, a 1% resolution in system utilization is already fine enough for overload control purpose.

[7] For practical implementation, we actually keep the CDF of log (CLP) of all suspicious packets and apply the discarding threshold against log (CLP). This is to eliminate the need of performing real-time inverse logarithm after the partial scores of various attributes are summed up according to Eq. (3).

arrival time-scale. When a suspicious packet[8] arrives, the following tasks are performed in parallel:

(1) The aggregate arrival rate towards the victim is adjusted. This, in turn, changes the input of the load-shedding algorithm.

(2) The packet attribute values are used for updating the fine-grain traffic profile, i.e. measured histograms, of the suspicious traffic.

(3) The CLP-based score is computed for the arriving packet using frozen scorebooks generated from a recent snapshot of suspicious traffic profile.

Once the score of an arriving packet is computed, the score CDF is updated. The packet is then discarded if its score is below the current discarding threshold, Thd. Notice that the use of frozen scorebooks is essential for the parallelization of tasks (2) and (3). It is also important to re-emphasize that, while CLP-computation is always performed for each incoming packet, selective packet discarding only happens when the system is operating beyond its safe (target) utilization level r_{target}. Otherwise, the overload control scheme will set Φ to zero.

IV. PERFORMANCE EVALUATION

In this section, we will evaluate the performance of the proposed CLP-based packet-discarding scheme in a stand-alone setting via simulation. Unless stated otherwise, the default settings for the simulation is summarized in Table 1.

Attack Type	Generic attack in Section IV A
Scorebook/ CDF Update Interval	Every 60 seconds
Baseline Profile	Five 10-minute windows of traffic, collected between 8:00pm to 8:10pm, Monday through Friday, in the week of May 10, 1999, by the WIDE project [WIDE].
Target Max. Load (r_{target})	Set to the maximum incoming load of the baseline profile observed over any 10-minute period. This corresponds to 1660 pps for the default baseline profile.
Legitimate traffic	Use a trace of the same link, collected between 8:00pm to 10:00pm, Tue, May 11, 1999. Its average arrival rate is 900pps.
Attack Intensity	10 times the nominal arrival rate
Scoring Strateg	Option 5 in Section IV D
Iceberg Thd	90%-adaptive-coverage scheme in Section IV E

Table 1: Default Simulation Settings

[8] For endpoint attacks, the suspicious packets are the ones which destinate to the victim subnet. For infrastructure attacks, all the packets passing through the victim choke-point are considered to be suspicious.

Performance Criteria: First, we examine the differences in the score distribution for attack and legitimate packets. Such differences are quantified using 2 metrics, namely, R_A and R_L as illustrated in Fig. 4.

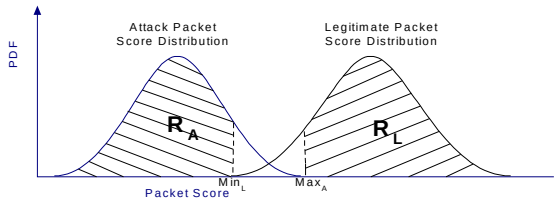

Figure 4: Characterizing difference in Score Distribution between legitimate and attacking packets

Let Min_L (Max_A) be the lowest (highest) score observed for the incoming legitimate (attacking) packets. Define R_A (R_L) to be the fraction of attacking (legitimate) packets which have a score below Min_L (above Max_A). The closer of the values of R_A and R_L to 100%, the better the score-differentiation power. In practice, the score distributions usually have long but thin tails due to very few outliner packets with extreme scores. To avoid the masking effect of such outliers, we have taken Min_L (Max_A) to be the 1st (99th) percentile of the score distribution of legitimate (attacking) packets. A typical set of score distributions for the attacking and legitimate packets are shown in Fig 5.

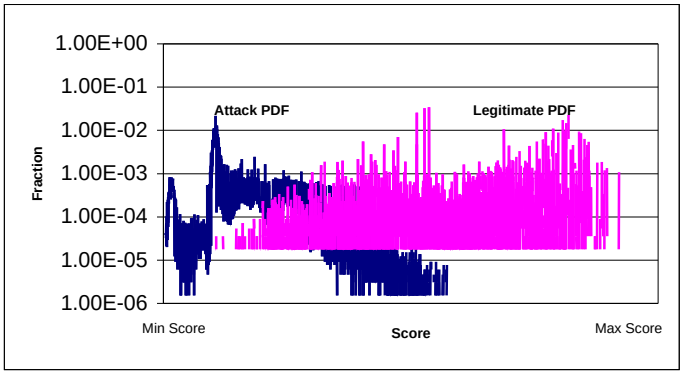

Figure 5: Sample Score Distributions

While R_A and R_L can quantify the score differentiation power, the final outcome of selective discarding also depends on the dynamics of the threshold update mechanisms. We therefore also measure the false positive (i.e., fraction of legitimate packets got falsely discarded), and false negative (i.e., fraction of attacking packets got falsely admitted) ratios of the proposed scheme. To check the effectiveness of the overload control scheme, we compare the actual output utilization r_{out} against the target maximum utilization r_{target} set by the scheme.

A. Different Attack Types

We have evaluated the performance of PacketScore in defending against the following types of attacks:

- *Generic attack*: all attribute values of the attacking packets are uniformly randomized over their corresponding allowable ranges.
- *TCP-SYN Flood attack*
- *SQL Slammer Worm attack*
- *Nominal attack*: all attacking packets resemble the most dominant type of legitimate packets observed in practice, i.e. 1500-byte TCP packets with server-port 80 and TCP-flag set to ACK, with uniformly random source IP addresses.
- *Mixed attack*: equally combines the above 4 types of attacks while keeping the overall attack rate to the 10 times of that of the target rate
- *Changing attack*: Similar to the *Mixed attack* except that the different types of attacks take turns. An attack type is randomly selected and continues for an exponentially distributed period.

The corresponding results are depicted in Table 2. In general, the proposed packet scoring scheme can successfully distinguish between attacking and legitimate packets. In all cases except the *Changing* attacks, R_A and R_L are above 99%. It is noteworthy that the false positive probability for the TCP-SYN flood attack is kept at a very low level (0 and 0.39%). Although the signature of the TCP-SYN flood packets can easily be derived by the PacketScore scheme, the ability of PacketScore to prioritize legitimate TCP-SYN packets over attacking ones based on other packet attributes, e.g. source IP prefix and TTL, is an essential feature. Without such priorization, e.g. in the case of stateless rule/signature-based filtering, all TCP-SYN packets would have been discarded and thus ensure the success of the DDoS attack towards the victim.

Changing attacks are more challenging due to their complex/ time-varying attacking packet characteristics. When the average change-period of the attack is much longer than the measurement/ scorebook generation interval (300 sec vs. 60 sec in our case), the change in attacking packet characteristics can readily be tracked. However, when such changes occur at the same (or shorter) time-scale of the measurement update interval, the PacketScore scheme can be misled to defend against some no-longer-exist attack packets. A possible remedy is to shorten the measurement update interval or apply more sophisticated change-detection techniques [Ke93] on the current profile measurements to trigger and speed up scorebooks/ CDF updates[9]. However, even in the worst case, the proposed scheme can still successfully discard more than 94% of the attacking packets (together with about 11% legitimate ones). This is substantially better than random packet dropping as the aggregate arrival rate is more than 10 times of the target load of the system. Furthermore, r_{out} is successfully kept close to its target value in all cases.

Attack Type	% False + ve	% False - ve	PDF Separation		$\frac{r_{out}}{r_{target}}$
			% R_A	% R_L	
Generic	0.31	4.49	99.49	99.54	0.99
SYN flood	0	4.09	100	100	0.95
SQL Worm	0	4.38	100	100	0.98
Nominal	0	4.26	100	100	0.97
Mixed	0.02	4.49	99.83	99.67	0.99
Changing (Ave. ON period = 60 sec)	10.82	5.84	95.15	97.43	1.07
Changing (Ave. ON period = 300 sec)	2.01	4.53	100	100	0.98

Table 2a: $r_{target} / r_{legitimate} = 1660/900 = 185\%$, based on the default r_{target} setting

Attack Type	% False + ve	% False - ve	PDF Separation		$\frac{r_{out}}{r_{target}}$
			% R_A	% R_L	
Generic	1.03	0.93	99.49	99.54	0.98
SYN flood	0.39	1.12	100	100	1.01
SQL Worm	0.39	0.91	100	100	0.99
Nominal	0.26	0.74	100	100	0.97
Mixed	0.75	0.99	99.82	99.67	0.99
Changing (Ave. ON period = 60 sec)	11.26	2.56	95.15	97.43	1.05
Changing (Ave. ON period = 300 sec)	4.75	1.29	100	100	0.98

Table 2b: $r_{target} / r_{legitimate} = 1000/900 = 110\%$, based on a lower r_{target} setting

Table 2: Performance against various Attack Types under different target maximum system utilization

The differences between Table 2a and 2b illustrate the tradeoffs of accepting more suspicious packets opportunistically by raising r_{target} beyond the legitimate traffic load $r_{legitimate}$: When r_{target} is increased from 110% to 185% of $r_{legitimate}$, the fraction of falsely discarded legitimate packets is reduced at the expense of the admission of more attacking traffic. Conversely, this indicates that the 4-5% false negative rate in Table 2a is mainly due to the gap between r_{target} and $r_{legitimate}$, i.e. the extra system capacity left over by the legitimate packets allows some attacking packets to slip through.

[9] However, there is a limit on the minimum number of packets to be observed before valid histogram statistics can be derived.

B. *Increasing Attack Intensity*

Fig. 6 shows the proposed scheme can effectively provide overload control as attack intensifies. Even when the volume of attacking packets increases from one time to 25 times of the nominal load, the scheme still consistently allow more than 99.5% of legitimate packets to pass through unaffected. The attacking packets are admitted only due to the gap between r_{target} and $r_{\text{legitimate}}$ as discussed before.

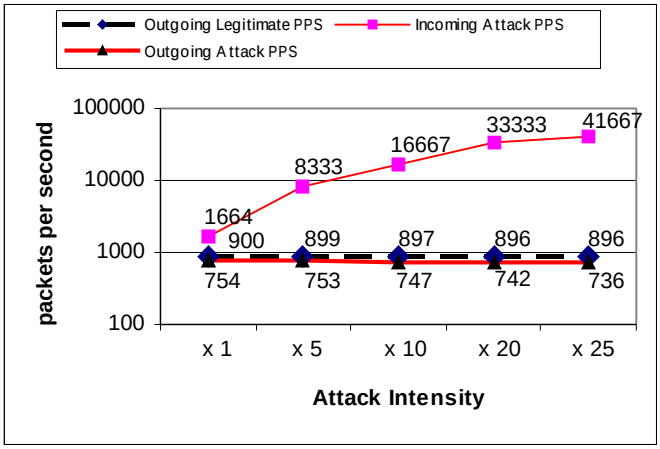

Figure 6: Effect of Increasing Attack Intensity

By design, the differentiation power of PacketScore improves as the DDoS attack intensifies. This is because as attack traffic volume increases, the difference between the current traffic profile and the nominal one also increases. Conversely, this reveals a limitation of the PacketScore approach: it is designed to protect against DDoS attacks which create their damage by overloading the victim with their sheer-volume of traffic. PacketScore is not effective against attacks which are based on very-low traffic volume, e.g. in *Tear-Drop* or *Ping-of-Death* attacks where a single carefully crafted packet is used to crash the entire system. Signature-based filtering would be more appropriate for such types of attacks.

C. *Nominal Profile Sensitivity*

In this subsection, we study the effect of the choice of nominal profile. In the first case, we build the nominal profile based on an hourly trace collected in during Monday, May 10, 8:00PM—9:00 pm, 1999. This is then used as the baseline for scoring 5 different 2-hour long traces (including itself), with attacking packets added, collected at the same hours but different days of the same week. The results are depicted in Table 3.

	% False +ve	% False -ve	PDF Separation		$\frac{r_{out}}{r_{target}}$
			% R_A	% R_L	
Mon	0.42	0.86	99.65	99.71	1.00
Tue	1.78	1.65	87.61	98.54	0.99
Wed	1.56	1.45	87.9	98.57	0.99
Thu	38.08	0.36	0.01	66.97	1.00
Fri	36.53	0.31	0	72.77	1.00

Table 3: Mon 1-hour profile applied to other days of the week

	% False +ve	% False -ve	PDF Separation		$\frac{r_{out}}{r_{target}}$
			% R_A	% R_L	
Mon	0.38	4.03	99.5	99.49	0.99
Tue	0.31	4.49	99.49	99.54	0.99
Wed	0.31	4.34	99.5	99.55	0.99
Thu	2.51	0.25	99.58	99.6	1.00
Fri	1.98	0.33	99.52	99.55	1.00

Table 4: Weekly profile applied to weekdays of the week

Observed from Table 3 that while the false negative probability is always maintained at a very low level (at most 1.65%), the false positive probability for Thursday and Friday 1-hour traces using the Monday profile is unacceptably high (> 38%). Upon further examination of the data, we find that this is an artifact of our default setting of r_{target} according to the maximum traffic rate of the baseline profile observed over any 10-minute window (which is about 1100pps for the Monday trace). Since the incoming traffic rate for the Thursday and Friday traces are significantly greater than the Monday one, such default choice of r_{target} inadvertently forces the system to discard a significant portion of legitimate packets. As shown in Table 4, the poor performance due to the mismatch among different daily legitimate traffic profile and r_{target} can be overcome by using the default weekly profile as described in Table 1.

D. *Different Scoring Strategies*

In this subsection, we explore the trade-offs of using different combinations of marginal and joint attribute distributions in establishing the nominal profile. Table 5 describes the options for baseline profile/scorebook generation:

Scoring Strategy	Description
1	Assume independence between each attribute and include the marginal distributions of packet-size, protocol-type, server-port number, TCP-flag pattern, TTL value in the nominal profile generation while excluding the source IP prefix distribution.
2	Same as (1) except using the 3-dimensional joint-distribution of packet-size, protocol-type and server-

	port number to replace their corresponding marginal distributions.
3	Same as (1) except including the marginal distribution of 16-bit source IP prefixes during baseline profiling.
4	Same as (2) except including the marginal distribution of 16-bit source IP prefixes during baseline profiling.
5	Same as (4) except using the 2-dimensional joint-distribution of 16-bit source IP prefixe and TTL value to replace their corresponding marginal distributions.

Table 5: Different Options of Scoring Strategies

Fig. 7 shows the differentiation performance and storage requirements of these five scoring options. As expected, among the five options, (5) yields the best scoring/ differentiation performance at the expense of increased storage size for baseline-profile and scorebook, while (1) has the smallest baseline-profile/ scorebook footprint. The performance improvement of (2) over (1), (as well as (4) over (3)) is due to the exploit of dependency among packet-size, protocol-type and server-port number. The improvement of (3) over (1) (as well as (4) over (2)) reflects the information value of source IP prefixes, even at a very coarse 16-bit granularity. The advantage of (5) over (4) illustrates the value of dependency information between source IP prefix and TTL value, which, to a large extent, captures the nominal "distance" between the source and the site/port/link to be protected.

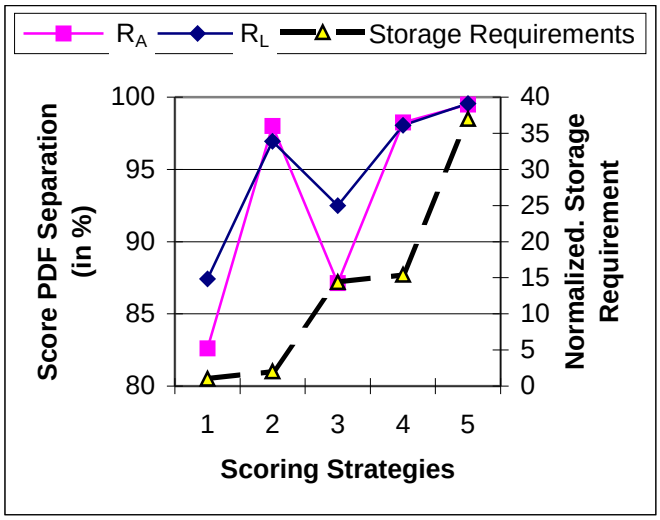

Figure 7: Comparison of scoring strategies

E. Setting the Iceberg Thresholds

In this subsection, we investigate the impact of iceberg threshold value on packet differentiation performance and profile/ scorebook storage requirements. We have considered two different iceberg threshold setting strategies. Under the *static* strategy, we fix the iceberg thresholds for all single attribute marginal distributions, 2-dimensional and 3-dimensional joint distributions at 0.01, 0.001 and 0.0001 respectively. Under the *adaptive* strategy, the iceberg threshold value is determined separately for each marginal/ joint distribution of interest so that 90%, 95% or 99% of the overall entries observed in the baseline trace are covered by the corresponding iceberg histograms. Table 6 summarizes the results of these various approaches. As shown in Table 6, there is no significant difference in the differentiation power of all the approaches. However, since the adaptive iceberg-threshold setting strategy should be more robust against possible changes in nominal profile traces, it is recommended over the static strategy. Among different coverage of the adaptive strategy, the 90%-coverage produces the best balance between storage requirement and differentiation performance. It also shows that with iceberg histograms, each nominal profile (as well as its corresponding set of scorebooks) requires less than 100Kbyte of memory.

	% False +ve	% False -ve	PDF Separation		$\frac{r_{out}}{r_{target}}$	Normalized Storage Req. (absolute size)
			% R_A	% R_L		
Static	0.10	4.51	99.41	99.87	0.99	1.0 (13.6 Kbyte)
Adaptive 90%	0.30	4.48	99.49	99.53	0.99	5.6 (76 Kbyte)
Adaptive 95%	0.07	4.50	99.62	99.90	0.99	9.4 (127.8 Kbyte)
Adaptive 99%	0.04	4.54	99.95	99.92	1.00	21.2 (288.3 Kbyte)

Table 6: Performance results against different thresholding methods

V. CONCLUSIONS AND FUTURE WORK

In this paper, we have outlined an architecture using a set of collaborating 3D-Rs and DCSs to defend against DDoS attacks. The proposed scheme leverages hardware implementation of advanced data-stream processing techniques, including one-pass operations of iceberg-style histograms and quantile (CDF) computations, to enable scalable, high-speed fine-grain traffic profiling and per-packet scoring. We have studied the performance and design tradeoffs of the proposed packet scoring scheme in the context of a stand-alone implementation. Such scheme can tackle never-seen-before DDoS attack types by providing a statistical-based adaptive differentiation between attacking and legitimate packets to drive selective packet discarding and overload control at high-speed. In a sequel of this paper, we will study the performance of a distributed implementation of the proposed scheme. In particular, we will investigate the effects of update and feedback

delays, as well as the impact of profile and score CDF resolutions on the performance of the distributed implementation. We will also study the ability and possible enhancements of the proposed scheme for defending against more sophisticated DDoS attacks. Another investigation topic is on how the time-scale of updates of the scorebooks, score CDF, and dynamic discarding threshold, will impact the response time and decision error of the proposed selective packet discarding scheme when subject to more orchestrated synchronized DDoS attacks. While the current CLP-based packet differentiation is theoretically attractive due to its Bayesian roots, it is conceivable to use a surrogate packet differentiating metric to replace CLP and design an even more hardware-amenable scheme based on a rudimentary per-attribute scoring mechanism, e.g. using an array of leaky-buckets. We intend to study the performance and complexity trade-offs of such alternatives for hardware implementation purpose.

REFERENCES

[Ba02] B. Babcock et al, "Models and Issues in DataStream Systems," *ACM Symp. on Principles of Database Sys.,* Jun 2002.

[Be01] S. Bellovin, M. Leech, T. Taylor, "ICMP Traceback Messages," draft-ietf-itrace-01.txt, *Internet draft,* Oct 2001.

[Br00] J.D. Brutlag, "Aberrant Behavior Detection in Time Series for Network Monitoring," *the 14th USENIX .Conf.,* Dec 2000.

[Ch00] F. Chen, D. Lambert, J.C. Pinheiro, "Incremental Quantile Estimation for Massive Tracking", *the 6th International Conf. in Knowledge Discovery and Data Mining*, Aug 2000.

[Es02] C. Estan, G. Varghese, "New Directions in Traffic Measurement and Accounting," *SIGCOMM,* Aug 2002.

[Fe01] W. Feng, D.D. Kandlur, D. Saha, K.G. Shin, "Stochastic Fair Blue: A Queue Management Algorithm for Enforcing Fairness," *Infocom, Mar* 2001.

[Fo01] J. Forristal, "Fireproofing against DoS Attack,"*Network Computing*, Dec 10, 2001.

[Gr01] M. Greenwald, S. Khanna, "Space-Efficient Online Computation of Quantile Summaries", *SIGMOD*, May 2001.

[Io02] J. Ioannidis, S.M. Bellovin, "Implementing Pushback: Router-Based Defense Against DDoS Attacks", *Network and Distributed System Security Symp.*, Feb. 2002.

[Ka03] R.M. Karp, C.H. Papadimitriou, S. Shenker, "A Simple Algorithm for Finding Frequent Elements in Streams and Bags", *ACM Trans. on Database Systems*, to appear.

[Ka01] S. Kasera et al, "Fast and Robust Signaling Overload Control," *ICNP,* Nov. 2001.

[Ke93] F. Keresteciogl*u, Change detection and input design in dynamical systems*, John Wiley 1993.

[La03] W.C. Lau, M.C. Chuah, H.J. Chao, Y. Kim, "PacketScore – a proactive defense scheme against Distributed Denial of Service Attacks," *NSF proposal under submission*.

[Le98] W. Lee, S.J. Stolfo, "Data Mining Approaches for Intrusion Detection," *the 7th USENIX Security Symp.*, Jan 1998.

[Ma99] D. Marchette, "A Statistical Method for Profiling Network Traffic," *the 1st USENIX Workshop on Intrusion Detection and Network Monitoring* , Apr 1999.

[Mazu] Mazu Networks Inc. http://www.mazunetworks.com

[Mi02] J. Mirkovic, G. Prier, P. Reiher, "Attacking DDoS at the Source," *ICNP*, Nov. 2002.

[Pa01] K. Park and H. Lee, "On the Effectiveness of Probabilistic Packet Marking for IP Traceback under Denial of Service Attack," *Infocom,* 2001.

[Rive] Riverhead Networks Inc. http://www.riverheadnetworks.com

[Sa01] S. Savage, D. Wetherall, A. Karlin, and T. Anderson, "Network Support for IP Traceback," *IEEE/ACM TON,* Vol. 9, no. 3, June 2001.

[Sn01] A. Snoeren et al, "Hash-based IP Traceback," *SIGCOMM,* Aug. 2001.

[WIDE] MAWI Traffic Archive, http://tracer.csl.sony.co.jp/mawi/

[Ya02] D.K.Y. Yau, J.C.S. Lui, F. Liang, "Defending Against Distributed Denial-of-Service Attacks with Max-min Fair Server-centric Router Throttles," *IWQoS*, 2002.

Appendix I

Here, we describe the load-shedding algorithm by Kaufman [Ka01], which is used as a sub-module in the PacketScore scheme. Let $\Psi_i = (1-\Phi_i)$ denote the fraction of packets permitted to pass the throttle point during the $(i+1)^{th}$ interval. Let $\Psi_0 = 1$ and Ψ_i be always constrained within the interval $[\Psi_{\min}, 1]$, where $\Psi_{\min}$ is a small but non-zero number which prevents the throttle from shutting off all incoming packets. At the end of the i^{th} measurement interval, the load estimate r_i is available and we calculate $f_i = r_{\text{target}} / r_i$, where r_{target} is the target maximum utilization allowed by the server (or port) which is chosen to permit the server to maintain a reasonable delay for all incoming packets. If $r_i = 0$, we set $f_i = f_{\max}$ where $f_{\max}$ is a large number whose precise value is unimportant. After f_i has been computed, the throttle ratio for in the next interval, denoted by Ψ_i is given by: $\Psi_i = \Psi_{i-1} f_i$. Since Ψ_i must be truncated to lie in the interval $[\Psi_{\min}, 1]$, we can rewrite the above as:

$$\Psi_i = \max\left\{\min\left\{\Psi_0 \prod_{j=1}^{i} f_j, 1\right\}, \Psi_{\min}\right\}.$$