NetBouncer: Client-legitimacy-based High-performance DDoS Filtering

Roshan Thomas*, Brian Mark+, Tommy Johnson*, James Croall*

*Network Associates Laboratories
Network Associates, Inc.
1145 Herndon Parkway, Suite 500
Herndon, VA 20170
{rthomas, tjohnson, jcroall}@nai.com

+Dept. of Electrical and Computer Engineering
George Mason University
4400 University Drive
Fairfax, VA 22030
bmark@gmu.edu

Abstract

We describe "NetBouncer", an approach and set of technologies for providing practical and high-performance defenses against distributed denial-of-service (DDoS) attacks. The central innovation in the NetBouncer approach to filtering and mitigating DDoS attacks is the ability to distinguish legitimate traffic from illegitimate ones so as to enable the discarding of only illegitimate traffic. In particular, this allows a NetBouncer-enabled network to distinguish DDoS congestion from flash crowd congestion situations. This provides a unique advantage over other DDoS mitigation techniques such as those based on filtering and congestion control where some loss of legitimate traffic is inevitable. The NetBouncer approach is characterized as an end-point-based solution to DDoS protection. It provides localized protection at potential choke points or bottlenecks that may exist in front of hosts and servers. NetBouncer attempts to block traffic as close to the victim as possible, while upstream of the nearest bottleneck. The immediate manifestation of NetBouncer technology is as a high-speed packet processing in-line appliance based on network processor technology. However, the long-term evolution, adoption and integration of NetBouncer technology may be in the back-plane/fast path of commercial high-speed routers.

1. Introduction

Denial-of-service (DoS) and Distributed Denial-of-service (DDoS) attacks have received a lot of attention lately in the security community and the industry at large. This can be attributed to the fact that the victims of these attacks have included well known web sites and electronic commerce companies. This is exacerbated by the reality that DDoS attacks are increasing in frequency and sophistication with the current attack tools now considered to be in the fourth generation of development [13].

In this paper, we present a practical and high-performance approach to DDoS defenses. The approach, related concepts and technologies are collectively referred to hereafter as "NetBouncer". The NetBouncer project is a two-year research effort currently under funding by DARPA's Fault Tolerant Networks (FTN) program and at the time of writing this paper, the project is about half-way through its funding period. The NetBouncer approach stems from the realization that at its core, the DDoS problem is caused by the illegitimate use of network and host resources so as to cause availability problems. As such, our approach to DDoS protection relies on distinguishing legitimate and illegitimate use and ensuring that resources are made available only for legitimate use.

In its current form, NetBouncer technology consists of high-speed packet processing and filtering devices. To enable filtering of incoming packets, a NetBouncer device maintains a large legitimacy list of clients that have been proven to be legitimate. If packets are received from a client (source) not on the legitimacy list, a NetBouncer device will proceed to administer a variety of legitimacy tests to challenge the client to prove its legitimacy. If a client can pass these tests, it will be added to the legitimacy list and subsequent packets from the client will be accepted until a certain legitimacy window expires. Once accepted, the transmission of legitimate packets is controlled by a traffic management subsystem that applies various bandwidth allocation and rate limiting schemes to ensure that legitimate clients do not abuse bandwidth consumption and that target servers cannot be overwhelmed even by what appears to be legitimate traffic.

We characterize the NetBouncer approach as an *end-point-based* solution to DDoS protection and thus contrast it from network-wide approaches that rely on network-wide visibility, trace back, attack isolation and congestion control. The immediate manifestation of NetBouncer technology, as pursued in our research prototype, is in the form of a high-performance in-line packet blocking and traffic management appliance built on a network processor (NP). However, the eventual evolution and adoption of NetBouncer technology may be in a form where it can be easily integrated into the backplane and fast path of commercial routers. NetBouncer provides localized protection and should be placed upstream of potential choke points or bottlenecks that may exist in front of hosts and servers in a network. In other words, NetBouncer attempts to block or rate limit packets as close to the victim as possible while upstream of the bottleneck. Based on the capacities of the various links and the potential for choke points, we may place NetBouncer-enabled devices at various points in a network topology. At the lowest level, NetBouncer can protect an individual server or a subnet. However, if choke points can arise further upstream, NetBouncer can be used to block traffic on the links connecting the border (distribution) routers to other interior (access) routers, or in the worst case block traffic from an ISP before it reaches a border router. NetBouncer makes no attempt to characterize or analyze a DDoS attack, maintain historical data or to trace the origin of an attack across intervening ISP networks and intermediary devices such as routers and firewalls. Our claim here is that although the DDoS protection NetBouncer can offer is localized and with limited network-wide visibility, it is nevertheless adequate for most organizations and offers a practical, cost-effective and easy to deploy near-term solution.

From a high level concept of operations standpoint, the working of a NetBouncer device is very simple. On receipt of a packet, a device has to make one of three decisions: (1) accept and transmit the packet; (2) discard the packet or (3) challenge the sender of the packet. However, NetBouncer has to operate in a manner that can meet the scalability and performance needs of high bandwidth, real-world commercial and military environments. In particular, our objective is to create a solution that is easy to integrate and has low technology insertion cost, scalable in terms of network topology complexity and network speeds, imposes minimal administrative overhead, and requires minimal collaboration and information exchange across organizational network infrastructures and Internet service provider (ISP) networks.

Meeting the above needs poses several research, design and architecture challenges. As such, the NetBouncer approach and design incorporates several innovative elements including:

- Novel techniques to test for the legitimacy of network traffic using stateless legitimacy tests and the subsequent enforcement of access controls on traffic.
- Algorithms to enable efficient look-up and updates of very large legitimacy lists.
- Quality-of-service (QoS) related traffic management schemes to provide rate limiting and bandwidth management for various classes of traffic based on client legitimacy and service priorities.
- Hardware-assisted high-speed packet processing techniques and architectures using network processors to implement the above functions, so as to provide a defense against DDoS attacks that incur minimal performance degradation (as measured by packet throughputs, latencies, etc.).

Our initial research resulted in a software prototype of NetBouncer based on the Linux operating system. This paper reports on our current efforts to develop a high-speed hardware prototype using leading-edge network processor technology.

The rest of this paper is organized as follows. Section 2 gives an overview of DDoS filtering based on client legitimacy and section 3 discusses flexible traffic management and QoS mechanisms to provide rate limiting and bandwidth management. Section 4 gives an architectural overview of the NetBouncer prototype currently being built on top of the Intel IXP 1200 network processor. Section 5 concludes the paper.

2. Current DDoS Defenses versus Legitimacy-based DDoS Filtering

We now survey current approaches to DDoS defenses and then discuss the NetBouncer approach to DDoS mitigation.

2.1. Related Work

In response to the growing DDoS problem, we have seen the emergence of a variety of vendor-supplied solutions as well as research-oriented solutions and prototypes. Commercial router manufacturers suggest a variety of techniques to detect and mitigate DDoS attacks based on traffic sampling, monitoring and filtering. These include setting up of counters and access lists to monitor traffic patterns and filter unwanted packets, ingress and egress filtering of bogus IP addresses and manual tracing of incoming traffic across routers and interfaces [6, 7, 8].

More recently we have witnessed a number of solutions from start up companies (or so called DDoS vendors) [2, 3, 4]. These solutions typically import incoming traffic traces and statistics from routers using technology such as NetFlow [9] or in-line sensor devices. These traces and statistics are then compared to well-known DDoS attack signatures and baseline traffic profiles to identify potential DDoS attack conditions and recommend filters and rate limiting parameters to routers. These products have the advantage that they give more DDoS-specific visibility into the network. However, these traffic monitoring and filtering approaches have the disadvantage that when attacks are mitigated through filters and rate limiting mechanisms, some proportion of the legitimate traffic may also be discarded (as noted in [5]).

Other techniques for DDoS mitigation include host-based and intermediary-based approaches. Host-based approaches apply better resource management techniques locally within hosts so that DDoS vulnerabilities do not arise due to resource starvation conditions. Examples include better connection and timeout management (such as for the TCP connection table) as well as slowing down or throttling senders through client puzzles [10] and congestion control mechanisms. Intermediary-based approaches rely on an intermediary sitting between attackers and target hosts. The intermediary may intercept and terminate suspect connection requests or provide stateful connection binding by negotiating connections on behalf of servers. An example of the latter is the TCP-intercept feature on many CISCO devices running the IOS operating system [8].

Research directions in DDoS solutions currently being pursued include automated tracing of attacks through collaboration of network devices so as to block attacks as close to the attacker as possible [23], collaborative congestion control [17] and the use of routing information to trace attacks [22]. However, such tracing may require collaboration from multiple routers, ISPs and network administrators. Also, a number of legal and logistical challenges and privacy concerns have to be overcome before attack information can be exchanged in a timely fashion.

2.2. Client-legitimacy-based DDoS Defenses

The commercial solutions surveyed above have several limitations. For example, approaches based on access filters and rate limits will inevitably discard some proportion of legitimate traffic along with illegitimate traffic as they cannot often distinguish DDoS-based floods from flash crowd situations where a large number of legitimate users may be requesting services from a server. Intermediary-based approaches simply move DDoS vulnerabilities from target hosts to intermediate hosts. Approaches based on network-wide congestion control, collaborative attack tracing etc. have long-term potential but in their current form do not scale and face other technical, legal and organizational challenges that limit cost-effective and easy deployment. Thus, NetBouncer is a direct response to provide a practical, end-point based and short-term DDoS solution that is easy to deploy in a localized manner. It can be seen as providing a complementary approach to other solutions.

The key innovation of the NetBouncer approach is the ability to distinguish legitimate traffic from illegitimate traffic. If we examine this issue more closely, we come to the realization that determining if traffic is legitimate, in turn, requires us to determine if the origin of the traffic (the client) itself is legitimate in relation to the target of the traffic (the server). In general, determining legitimacy may require us to abstract and analyze the traffic at one or more levels of the protocol stack. Our approach thus relies on a series of legitimacy tests, with each test targeted for a particular type of traffic recognized at a specific protocol layer. Thus, the source of the traffic and the related notion of the client identity that is validated will depend on the protocol layer (e.g., network, application etc.) and the application or service (ftp, real video, etc.) for which a test is administered. Thus, it makes sense for a legitimacy test at the network layer to determine the validity of a host or router as identified by its IP address. At the transport layer, the legitimacy tests may be aimed at validating TCP connections. At the application layer, our legitimacy tests will attempt to determine valid application sessions and user processes and identifiers. Depending on the circumstances and the application, we may apply in succession, a combination of tests for various protocol layers and application level notions such as sessions.

The legitimacy tests we are developing in this project are all governed by some common principles and design objectives. These include the following:

- No changes should be required to network protocols or the configurations at clients and servers.
- No administrator intervention should be required to perform legitimacy tests in real time.
- The overall operation of a NetBouncer device and the administering of legitimacy tests have to be state-safe. The concept of state-safety can be informally defined as a property that guarantees that the amount and rate of the consumption of state at NetBouncer cannot be directly induced by the proportion and rate of illegitimate traffic. This prevents the implementation of legitimacy tests themselves from being vulnerable to state-consumption DDoS attacks.[1]

These principles and objectives also allow us to design and deploy NetBouncer devices rapidly with minimal changes and disruption to existing network configurations and services within an infrastructure.

[1] We are currently developing formal state-safety proofs for various legitimacy tests. However the presentation of this is beyond the scope of this paper.

2.3. Categories and Examples of Legitimacy Tests

Our ongoing research has led us to believe that there exist at least three categories of legitimacy tests. Specifically, these include packet-based tests, flow-based tests and service-based tests. In order to understand the specific and sometimes subtle differences between these, let us first define the notion of legitimacy tests in more detail. In particular, we model the elements common to all legitimacy tests.

Definition. A legitimacy test ***t*** is a tuple <assertion, legit-id, pre-scope, pre-state, post-state, post-scope> where:

- **assertion***:* the legitimacy assertion that is validated if the test is successfully passed.
- **legit-id:** is the identifier in the assertion that is validated and subsequently used to process requests.
- **pre-scope:** the entity (such as a packet, protocol flow, application session etc.) to which the test is applied.
- **pre-state:** the state that is maintained and examined in order to validate the legitimacy assertion. A good example of such state information is authentication and integrity check information in response to messages returned by a client after it has been challenged.
- **post-state:** the state that is required after legitimacy has been established and is maintained, examined and updated to process legitimate data traffic as it is passed through. Typically, the post-state will include the legit-id and other formatting, housekeeping and statistical information.
- **post-scope:** this refers to the entities to which the just validated legitimacy assertion and identifier applies.

We now describe each of these categories in turn and give examples of tests in each category.

2.3.1. Packet-based tests

The distinguishing characteristic of a packet-based legitimacy test is that the pre-state and post-state that needs to be examined is fully self-contained in an individual packet. In other words, a decision as to whether a packet can be passed is made solely by examining the contents of the packet. In particular, there is no information from previously seen packets that needs to be retained and consulted.

Source host address validation at the newtwork layer

Many DDoS attacks use spoofed and bogus source IP addresses making it difficult to trace the source of the attack. Thus, one technique that can be employed to diminish or prevent certain types of DDoS attacks is the use of legitimacy tests to determine if the IP address in an incoming request is associated with a live host. One such test is based on the creative use of ICMP echo messages[2].

A NetBouncer device intercepts any incoming packet destined for a server. If the source of the packet is recognized as not being on the legitimacy list, NetBouncer will challenge the source of the packet by sending to this source an ICMP echo request. However, to avoid storing any state in NetBouncer, the original incoming request packet is encapsulated in the payload of the outgoing ICMP echo request. If the original sender of the request is a valid host, the legitimacy test host can expect to get back an ICMP echo reply packet.[3] However, we need to authenticate such replies and verify their integrity. To enable this, the payload of the ICMP echo request also includes a hashed message authentication code (HMAC) computed using a keyed hash function and taking as input a packet tag, the incoming request packet, source IP address, payload-length, expiry-time, and a nonce. If an ICMP echo reply is received and the HMAC can be verified, the authenticity and integrity of the ICMP echo reply is verified and the source IP address of the extracted incoming request packet is added to a legitimacy list consisting of validated source addresses. The HMAC is an example of the pre-state. The detailed messages for this test are given in Figure 1 with "h_k" standing for the HMAC function h using key k.

Anti-smurf filtering through egress packet tagging and ingress filtering

Some DDoS attacks (such as a smurf attack) spoof the source addresses of victims and utilize an amplification network to overwhelm a server with messages such as ICMP-echo replies. To provide a defense against such attacks, NetBouncer cryptographically signs and tags outgoing ICMP-echo requests from the trusted side of internal networks and accepts ICMP-echo replies from the untrusted side only if they contain the previously inserted tag. This tag is basically the pre-state. A flood of ICMP-echo replies generated from untrusted sources in a Smurf attack can thus be filtered and discarded. Although we have discussed this test within the context of the smurf-style attacks, this scheme of egress packet tagging combined with ingress filtering can be used for any scenario where incoming packets should be received at a

[2] We realize that a test based on ICMP echo is not very reliable as many hosts, firewalls and routers are configured to not respond to echo requests; nevertheless this provides a simple illustration.

[3] Please note that the challenge packet must be smaller than the maximum transmission unit (MTU) for this test to be applicable.

network only in response to a previously sent outgoing request.

2.3.2. Flow-based tests

We define a "flow" as a stream of packets from a particular protocol connection/session (at layer 4 in the network stack) and thus having the same (<source-address, source-port> <destination-address, destination-port>) properties. In contrast to a packet-based test, the post-state is not contained within individual packets and more significantly a single post-state applies to the entire stream of packets that belong to a flow.

Transport layer TCP connection request validation using stateless TCP SYN cookies

This layer 4 test provides a defense against denial-of-service attacks based on TCP SYN floods by providing a means to validate TCP connection requests for their legitimacy. NetBouncer basically intercepts TCP requests for connection establishment.. In traditional TCP implementations, as soon as the first TCP SYN packet is received, state is allocated for the connection. This violates state-safety.

To elaborate, NetBouncer intercepts SYN packets and generates a cryptographic checksum (cookie) of private rotating keying material and various fields in the TCP/IP header. This cookie is stored in the outgoing SYN/ACK packet as a NetBouncer-generated intial sequence number (this cookie is effectively the pre-state). The SYN/ACK is then returned to the source address of the SYN packet. When the ACK (which is the response from the client to the SYN/ACK) arrives at the NetBouncer, if the cookie (pre-state) verifies, state is instantiated for the connection. Then the original SYN along with client supplied sequence number is recreated based on the information in the ACK packet and the stored cookie and forwarded to the original destination server. However, the original destination will then return a SYN/ACK server sequence number of its own, and NetBouncer will complete the 3-way handshake. NetBouncer will also retain the offset between its sequence number and the server sequence number, allowing it to transform the sequence number on each packet on the TCP connection from and to the server through NetBouncer. This offset and related information form the post-state for the test. Our solution is an adaptation of the idea of SYN cookies proposed by [1]. The detailed messages for this test are shown in Figure 2.

If the original SYN packet was spoofed and sent from a fraudulent IP address not used by any live host, then no ACK packet will be generated and the original SYN packet is effectively ignored. Now if the spoofed address is in use by another host but one that did not send the original client SYN packet, the host will send a reset (RST) packet and no connection will be established. In summary, this technique allows us to handle TCP connection requests in a **stateless** manner. This is clearly state-safe. State is reserved only when the host generating the request has demonstrated that it is legitimate, i.e., that it received the SYN/ACK and wants to complete the connection handshake. Our stateless approach is superior to stateful connection binding or knitting approaches such as TCP-Interrupt [8] as it escapes state-based DDoS vulnerabilities. Approaches such as TCP-Interrupt are an attempt to better manage state but are truly not state-safe.

2.3.3. Application and session-oriented legitimacy tests

In contrast to packet-based and flow-based tests, this third category of tests is relevant for higher level (above layer 4) services and applications. In particular, these tests understand application level abstractions and structures as well as session semantics. Testing for legitimacy requires an understanding of the structure and semantics at the application-level. Thus, a packet-based and flow-based examination of traffic at the level of IP packets and flows will generally not provide conclusive evidence of the legitimacy of application traffic. Rather, this would require examination of application headers. In an application-level test, the pre-state may be spread across several IP packets and applying the post state may require examination of application-level headers. We now describe some application level legitimacy tests.

In reality, this category of tests may be thought of as consisting of two subcategories. The first is what we call *structured composite services* (SCS). These consist of services and protocols such as the real-time streaming protocol (RTSP). The structure of an RTSP session can be thought of as a composite one that consists of many underlying lower level protocol sessions and connections (including TCP, UDP and RTP etc.) However, the exact structure is fixed by the RFCs and standards.

The second subcategory consists of *ad-hoc composite services* (ACS). The structure of ACS services is also a composite one but varies from one environment to another rather than being defined by a standard. A simple example of an ACS service would be one where a user clicks on a URL at a web site and the server subsequently downloads one or more applets to the client machine. These applets may subsequently initiate additional network connections and services depending on the application logic and transaction structure. So the critical challenge here is to understand how legitimacy state is preserved and tracked through various components sessions and application interactions.

We now describe some application level legitimacy tests, but limit our discussion to the SCS subcategory. The ACS subcategory is currently a topic of intense research and will be reported in the future.

Application layer Turing-style test for the presence of a human user

As validation methods and countermeasures against DDoS attacks become more powerful, we can expect attacks to move up from the protocol level to the application level. During a DDoS attack, it is critical to distinguish attackers from non-threatening users. Since most DDoS attacks are composed of a handler guiding a large number of automated attack agents [13], we can use the non-intelligence of the digital agents to distinguish them from intelligent human users. The key idea is to interrupt an application session and challenge the client host with a question only a human user can answer. We can test for intelligence by questioning agents with some puzzle/question that requires some basic level of human intelligence to answer; for instance, “What color is the apple?" or "How many houses do you see in the picture?"

Such techniques were used in many historical battles to expose spies. If the client answers the question correctly, its identity is entered into a list of good clients that are allowed subsequent access. To be of any value, this test must be done in a manner that cannot be defeated by automated agents. In other words, the questions posed should be sufficiently random and yet the answers to these questions should require a level of intelligence that will make it difficult for an automated agent (or computer program) to construct.

In our current hardware prototype of NetBouncer, we have completed an initial implementation of this test at the application level and integrated it with HTTP/HTML. In this implementation, a user (client) initiating an http (web) request from a web browser is confronted with a question (i.e., a puzzle). If the user can supply the correct answer, he is added to the legitimacy list.

The test works as follows. With an incoming HTTP request, NetBouncer intercepts the connection establishment TCP SYN packet from the client and responds with the SYN/ACK packet on behalf of the server. The client then responds with the TCP ACK packet to complete the TCP 3-way handshaking procedure and then follows up with an HTTP get request. NetBouncer then issues the challenge by posting an HTML form. However, the correct answer to the puzzle is cryptographically sealed and sent with the form. If the client responds correctly, as verified by comparing the user supplied answer with the cryptographically sealed and extracted answer, NetBouncer will then send an HTTP refresh request to the client's browser and this will result in the client issuing a second HTTP get request that NetBouncer routes directly to the server.

The tests discussed above represent only a small sample taken from our initial research in developing a prototype of NetBouncer. However, the diversity of these legitimacy tests should give the reader some idea of the range of possibilities with respect to complexity and applicability at various protocol layers and for newer protocols and applications. The development of additional tests as well as algorithms to determine the right combination of tests for a given situation and host and the sequence in which such tests are to be applied, all represent active areas of research for this effort. Some discussions of these issues follow.

3. Integrating Legitimacy and Quality of Service

A novel concept that is key to the NetBouncer approach is the integration of legitimacy, security, and assurance-related attributes into quality-of-service (QoS) provisioning. The objective is to provide QoS to different service classes even under DDoS conditions. NetBouncer factors legitimacy weights and service priorities into QoS-based traffic management, including rate-limiting and fair scheduling of outgoing packets. We expect traffic management features to be important requirements for many environments that could deploy NetBouncer technology.

Many ISP and enterprise environments must provide certain QoS guarantees to their customers. If ISPs could make such guarantees even when their networks are under DDoS attacks, they could easily use this feature to differentiate their services from those provided by other ISPs. E-commerce environments that distinguish varying classes of customers may wish to provide superior service to certain preferred (high class) customers. For example, a stock brokerage may want to provide faster response times and better network access (especially under DDoS conditions) to customers who have account balances that exceed a certain amount. Finally, in an enterprise environment, one may wish to reserve more bandwidth and guarantee higher packet rates for certain classes of services (such as http, video, voice, etc).

Our goal is to provide a rich set of underlying mechanisms so that NetBouncer has the flexibility to support a variety of QoS policies based on client legitimacy and service classes. Supporting these policies requires an integrated approach incorporating various rate limiting and scheduling mechanisms.

3.1. Legitimacy Weights and Costs

To manage various cost and assurance tradeoffs, we associate with every test a weight (w) and a cost (c). The weight of a test is a measure of the "goodness" of a test while the cost is a measure of the expense that can be incurred when a test is administered. When we think of characterizing the goodness of a test, we may use one or more criteria such as assurance, reliability, strength or difficulty level etc. Thus our goal is to establish the weight

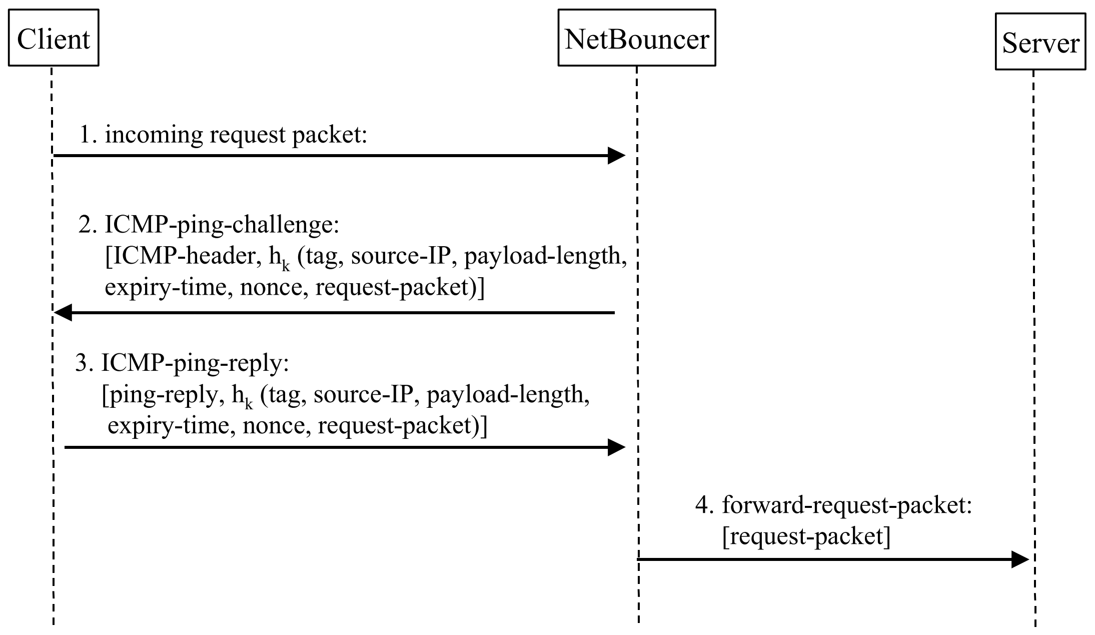
Client
NetBouncer
Server
1. incoming request packet:
2. ICMP-ping-challenge:
[ICMP-header, h_k (tag, source-IP, payload-length, expiry-time, nonce, request-packet)]
3. ICMP-ping-reply:
[ping-reply, h_k (tag, source-IP, payload-length, expiry-time, nonce, request-packet)]
4. forward-request-packet:
[request-packet]

Figure 2. The ICMP echo (ping) test

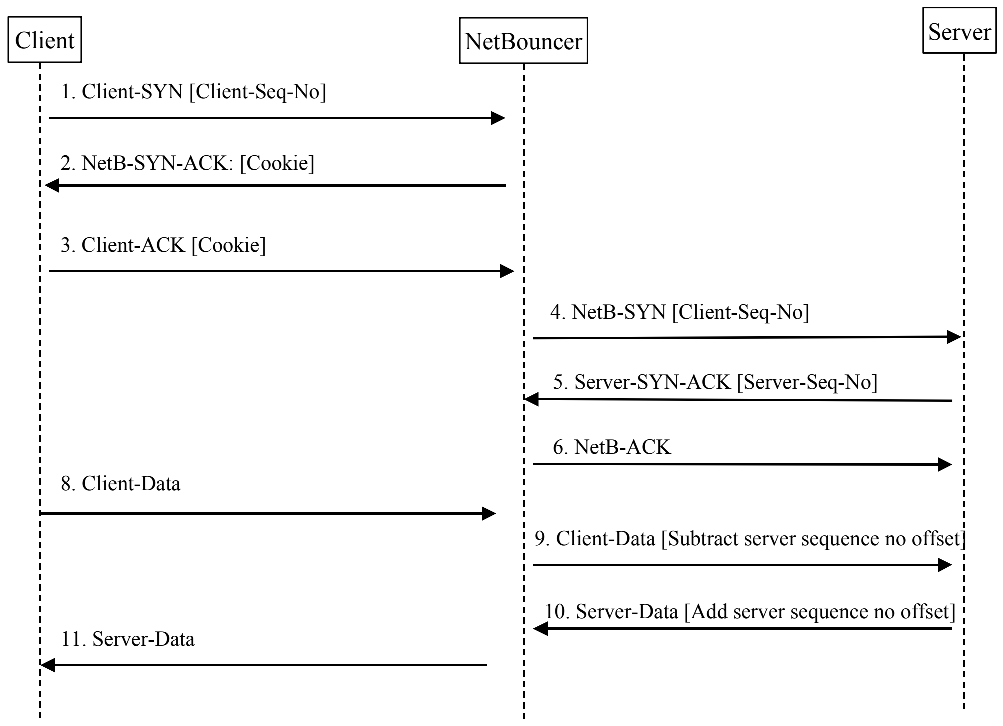
Client
NetBouncer
Server
1. Client-SYN [Client-Seq-No]
2. NetB-SYN-ACK: [Cookie]
3. Client-ACK [Cookie]
4. NetB-SYN [Client-Seq-No]
5. Server-SYN-ACK [Server-Seq-No]
6. NetB-ACK
8. Client-Data
9. Client-Data [Subtract server sequence no offset]
10. Server-Data [Add server sequence no offset]
11. Server-Data

Figure 1. The TCP SYN Cookies legitimacy test

of a test as a function of assurance, reliability and strength, i.e.,

Legitimacy-weight = *f* (assurance, reliability, strength) … (1)

Intuitively, a high degree of assurance (or confidence) in a test should be indicative of the fact that the test cannot be circumvented or compromised easily. This in turn relates to the soundness of the implementation as well as the security of the messages and protocol steps required to administer the test. The reliability of a test relates to a variety of qualities including the ability to get consistent and accurate results. The strength of a test relates to the rigor of the test challenges and the difficulty level and specificity of the answers required. The idea here is that a stronger test will establish legitimacy with more certainty by narrowing down the respondents to a test.

To model the cost of administering a legitimacy test, we consider the various factors that contribute to the overall overhead and expense of the test. Our initial investigations have identified three obvious contributors. These include packet processing, message transmission and miscellaneous computational overhead. Thus we can express cost as:

Cost = *f* (packet-processing, message-transmission, misc-computational-overhead) … (2)

To elaborate, packet-processing costs include the overhead in processing various headers and the payload and the overhead to queue and dequeue packets. Message transmission costs include the cost to transmit and receive messages where a message may consist of one or more packets. We also have to factor in additional computational overhead for operations such as those involving cryptography.

3.2. Service Classes based on Legitimacy Weights and Service priorities

Central to our QoS approach is the notion of service class. The service class determines the rate limiting and bandwidth allocation policies that are applied to the packets. The service class assigned to a packet arriving at NetBouncer port is determined by: (1) the legitimacy weight of the source that sent the packet (2) a preassigned service priority for the service to which the packet belongs. For each packet, a *service-class-score* is computed as a function of its legitimacy-weight and a *service-priority-weight*:

Service-class-score = *f* (legitimacy-weight, service-priority-weight) … (3)

In formulating a function to compute the service-class-score, it is important to consider the implications of processing packets that are destined for services that have a high service priority but have very low legitimacy. In particular, these packets should not be assigned to the high service priority queues. To achieve this, a *service-class-threshold* is defined for each service class. A packet's service-class-score must meet or exceed a service-class-threshold for it to be assigned to the corresponding service class queue. An initial choice for a function to compute the service-class-score is:

Service-class-score = ((legitimacy-weight x service-priority-weight) + service-priority-weight) … (4)

In general, the service-priority-weights should be chosen so that the service-class score intervals do not overlap. All packets whose service-class-score falls below the lowest service-class-threshold are assigned to a default low priority service queue. This queue will be serviced only when there are no other packets in the system.

3.3. Two-tiered Adaptive Framework for Traffic Management

NetBouncer employs a two-tiered framework that can support a rich variety of security-based QoS policies. Tiers 1 and 2 provide, *inter-class* and *intra-class* traffic management, respectively. Inter-class packet scheduling provides QoS differentiation among different service classes, while intra-class scheduling ensures a fair allocation of resources to clients belonging to the same service class.

At any given time, a NetBouncer installation may be required to support a variety of QoS policies and the traffic management policies may be modified adaptively based on ongoing network congestion and its impact on QoS. The adaptive aspect of our framework refers to a system implementation that constantly monitors the policies that need to be supported and dynamically provides the appropriate traffic management at each tier in order to support the current policies. Ideally, this adaptation should be done in an automated fashion with minimal human/operator intervention.

3.3.1. Inter-Class Scheduling

Strict priority (SP) scheduling is a simple way of providing service differentiation, but is not sufficient to provide QoS guarantees to service classes. Under SP, the highest priority service class is effectively guaranteed all of the available bandwidth, while the remaining classes receive no bandwidth guarantees. To provide QoS guarantees to different service classes, we have adapted the use of dynamic rate scheduling (DRS) as reported in [14]. DRS allows a NetBouncer system the capability to support a rich variety of policies for QoS and rate limiting. In particular, it offers the ability to set hard rate limits for various classes of service and to adjust these rates dynamically to take advantage of available excess bandwidth.

To illustrate the basic framework for DRS, consider two classes of service: high (H) and low (L). Under DRS, the rates for these classes are governed by:

$$R_H = M_H + W_H E, \quad R_L = M_L + W_L E \qquad \ldots (5)$$

where R_H and R_L are the effective service rates for the classes H and L, M_H and M_L are guaranteed minimum rates for H and L, E is the excess bandwidth available. The weights W_H and W_L determine the fractions of E that will be assigned to H and L, respectively.

The essential idea is that of a minimum guaranteed rate for each service class. Intuitively, service classes requiring more bandwidth are assigned higher minimum rates. These minimum rates should be chosen after careful analysis so as to ensure that the system can guarantee these rates. Once the minimum rates, M, are assigned, the effective rates R are computed dynamically as the sum of M, a class weight W, and the available excess bandwidth E. To ensure a hard rate limit for a service class, its associated class weight W can be set to zero. The excess bandwidth E is computed dynamically based on traffic utilization and/or aggregate queue lengths in order to ensure that service classes with strictly positive weights, W, can use any available bandwidth. Thus, DRS adapts to changing traffic conditions.

3.3.2. Intra-class Scheduling

Intra-class scheduling provides fair bandwidth allocation within a service class. Let us discuss why such a scheme is needed. Consider the scenario where there are two legitimate sources (clients) S1 and S2 that have been assigned to the same service class. If S1 sends 95% of the packets for this service class, it will clearly dominate the bandwidth allocated to the service class. From S2's perspective this is essentially a bandwidth-based denial of service condition.

To address the above problem, tier 2 of the NetBouncer traffic management framework consists of an intra-class weighted fair queuing (ICWFQ) scheme for bandwidth management within service classes. Weighted Fair Queueing (WFQ) is a packet scheduling scheme that approximates ideal fair scheduling among a set of connections by assigning timestamps to arriving packets [16, 21]. The timestamp values are proportional to the finishing times of the packets under an ideal fair scheduler. Packets are then served in increasing order of their timestamp values. Several variants of WFQ have been devised that achieve a sufficient degree of fairness while also being computationally efficient. The basic form of the timestamp computation for WFQ and its variants is as follows:

$$TS(i,k) = \max \{TS(i,k-1), VT\} + P(i,k)/W(i), \quad \ldots (6)$$

where *TS(i,k)* and *P(i,k)* denote, respectively, the timestamp value and packet length of the *k*th packet arriving from the *i*th source, and *W(i)* is a weight value assigned to the *i*th source. The weight *W(i)* is proportional to the bandwidth share received by the *i*th source, i.e., WFQ scheduling guarantees that the *i*th source will receive a minimum bandwidth share proportional to *W(i)*. From (6), it can be seen that a packet belonging to a source with a small value of *W(i)* will be assigned a larger timestamp than a packet associated with a smaller value of *W(i)*, assuming all other variables to be the same for both packets. Hence, the first packet will be served before the second. In WFQ, the variable *VT* (virtual time) represents the normalized finishing time of the *k*th packet from the *i*th source under an idealized fair scheduler. A popular variant of WFQ known as Self-Clocked Fair Queueing (SCFQ) [16] sets the variable *VT* equal to the timestamp value of the packet that is being served at the time of the current packet arrival. At the end of a busy period, *VT* is reset to zero.

Our innovation here is that ICWFQ employs a fair queueing scheme (i.e., WFQ or one of its variants) to guarantee clients within a given service class a fair share of the bandwidth allocated to the class at tier 1, with respect to the client's level of legitimacy. The weight *W(i)* associated with the *i*th client is set equal to the client's legitimacy weight. Thus, client legitimacy is factored into the QoS provisioning within a service class.

Implementation of ICWFQ requires a priority queue to be maintained for each service class. In such a priority queue, the packets are ordered in increasing value of their associated timestamp, which can be stored as part of the packet descriptor. The most recent timestamp value associated with a (legitimate) client must be stored in a list indexed by the legitimacy list. We are currently exploring how ICWFQ can be supported within the IXP-1200 based hardware prototype of NetBouncer.

4. Architecture of the NetBouncer Hardware Prototype

We now discuss the architecture of the NetBouncer hardware prototype implemented on the Intel IXP1200 network processor. This prototype is a network transparent inline device with Gigabit input and output ports connected by fiber cables. We start with an overview of our hardware platform.

4.1. Overview of the Intel IXP1200 Network Processor

The IXP1200 chip and network processor development system consists of a StrongArm processor and six microengines. The Intel StrongArm Core processor, is a 32-bit RISC processor currently available at an operating frequency of 200 MHz and acts as the master general-purpose processor and supports the control plane. It would be responsible for performing the bulk of the signaling and network management functions comprising the control plane. The actual movement of data and packet processing operations is performed by six 32-bit RISC Microengines running at 200MHz and acting as specialized slave processors. Each microengine can execute four parallel threads by means of four independent program counters, zero overhead context switching and hardware semaphores. In addition, the Microengine contains an ALU and shifter, 4 Kbytes of RAM control store, 128 32-bit general-purpose registers and 128 32-bit transfer registers for accessing the SRAM and SDRAM units. The IXP1200 also contains a hash unit for generating 48- and 64-bit hash keys. The IXP1200 supports two Gigabit Ethernet ports and eight fast Ethernet (100 Mbps) ports. For our current prototype, only the Gigabit ports are used.

4.2. Prototype Architecture

Figure 3 illustrates in detail the architecture of our current hardware prototype. Our prototype uses only the two Gigabit ports. The thick arrows shows the *fast path* in the architecture - the path through which legitimate packets are processed and transmitted. The dotted arrows shows the *test path* which is the path through which packets that failed legitimacy, as well as test packets associated with sending and verifying legitimacy tests, are processed and transmitted. The figure also shows the careful allocation of various NetBouncer packet filtering and legitimacy administration functions to the available microengines (labeled μE1 through μE6). Packets are received from the media access control (MAC) layer, processed, and transmitted. To enable this, a variety of queues are used. As a packet is received from the MAC layer, it is stored in SRAM. However, vital information from the packet (such as critical header information) is extracted and stored in a structure called a *packet descriptor*. The packet descriptor also stores a pointer to the physical SRAM address where the packet is stored. The queues are used to pass the packet descriptors from one microengine to another as processing of the packet progresses. This is more efficient than passing the actual packet from one queue to another.

Let us look at the details of how packets are processed on the prototype. We start with the fast path in the architecture. Processing starts with the "Internet-Receive" microengine (μE1) reading incoming packets from the MAC layer and depositing their packet descriptors into the queue labeled Q-fast-in. A second microengine labeled "Fast-Path-Manager" (μE2) dequeues these descriptors, extracts the source address and checks on the legitimacy list to see if the packet is coming from a legitimate source. If the source address is on the legitimacy list, we consider the packet to have passed the legitimacy test and the corresponding packet descriptor is queued into the queue labeled Q-fast-out. A third microengine labeled "Protected-Transmit" (μE3) consumes descriptors from this queue and uses the pointer in the descriptor to extract the packet contents and transmit them to the appropriate destination on the trusted side.

Now if a packet is not on the legitimacy list, its descriptor is queued into the queue labeled Q-test-in and the packet is now processed through the test path. A microengine labeled "Test-Manager" (μE5) empties entries from this queue and issues one or more legitimacy tests. Each legitimacy test challenge packet is queued into the queue Q-test-out from where the microengine "Source-transmit" (μE4) transmits the challenge back to the client source. Now our design accommodates the scenario where if a legitimacy test is too complicated for data plane processing in a microengine, the appropriate descriptor is forwarded to the StrongArm processor for further processing via the queue labeled Q-strong-arm. The StrongArm may do more complex processing such as those involving very sophisticated legitimacy tests. Finally, if a packet can be identified by μE1 as a response from a client to a previously issued challenge, its descriptor is put directly into the queue Q-test-in and thereby avoiding the Fast-Path-Manager" (μE2) microengine. These response packets will eventually be processed and validated by the Test-Manager" (μE5).

Implementing an HMAC function on the IXP1200 requires careful thought due to the limited instruction store and memory and other efficiency concerns. Common schemes such as MD5 and SHA1 are too complex and inefficient. So we have chosen to implement an alternate algorithm called Michael [15] to generate HMACs for the various legitimacy test packets. Michael is simple to implement and offers acceptable tradeoffs between performance and security strength given NetBouncer's design intention to change keys frequently. By default, Michael generates 64-bit authentication tags. NetBouncer performs some modest post-processing of the Michael output by applying the XOR operation to the two 32-bit halves to generate a final 32-bit result that is encapsulated in outgoing legitimacy test packets.

4.3. Support for High-speed and Memory-efficient IP Address Lookup

In the NetBouncer architecture, the legitimacy list may contain hundreds of thousands of entries. Since the

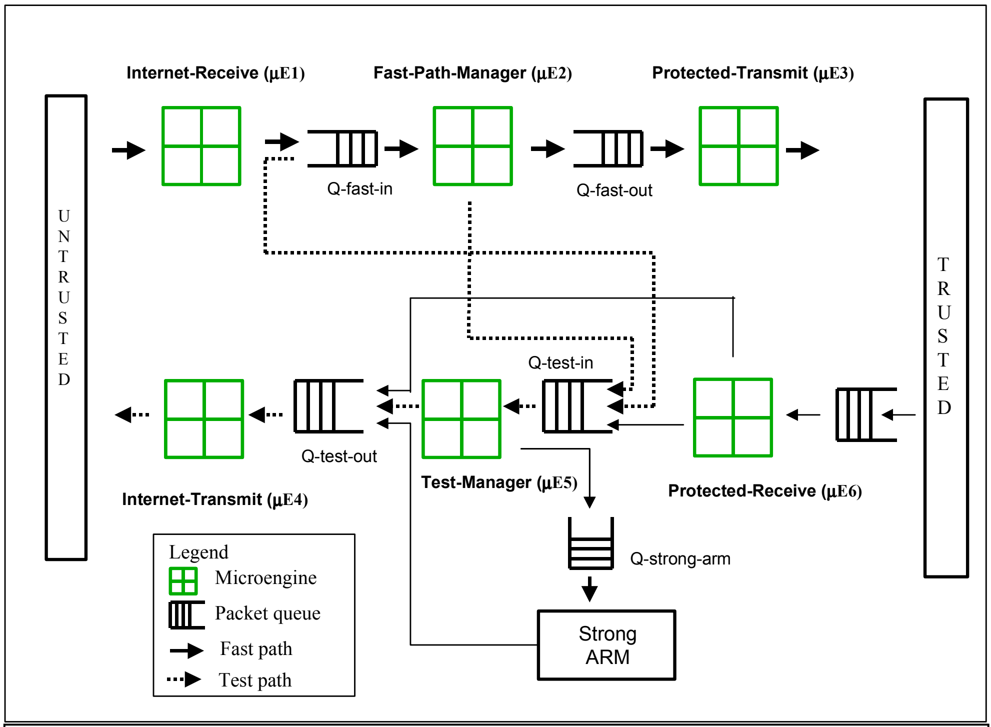

Figure 3. Architecture of the NetBouncer hardware prototype

lookup function lies on the fast path of the hardware architecture and is performed on every incoming packet, it can become a major system bottleneck, reducing packet throughput dramatically. Moreover, NetBouncer must be able to process a very high frequency of updates to the legitimacy table. We expect the legitimacy list to be updated every few seconds. In contrast, routing tables on routers are updated far less frequently. Another unique requirement is the size of the legitimacy list. Since clients correspond to full source IP addresses rather than subnets indicated by prefixes, the size of the legitimacy list could be significantly larger than typical routing tables. These requirements pose some unique challenges.

Various approaches to fast IP address lookup have been proposed in the literature [11, 19, 20, 25]. The LC-trie structure proposed in [20] (a variation of tries [18]) offers very good lookup performance with minimal storage requirements but is not optimized to support dynamic updates. By combining features of hash tables and LC-tries, we have developed a structure called the hash-trie that can support efficient insertion and deletion to and from the legitimacy list. Further details are presented in [24].

5. Conclusions and Future Work

We have presented a practical and high-performance approach to DDoS filtering. The key elements to the NetBouncer approach include the use of legitimacy tests to distinguish legitimate traffic, high performance through the use of network processors and a flexible two-tiered QoS-oriented traffic management scheme. NetBouncer differs from other approaches to DDoS protection in that it focuses on distinguishing legitimate traffic from illegitimates ones. This will allow NetBouncer to be placed in environments where even under DDoS attack conditions, legitimate requests from high priority clients to critical services can be guaranteed availability and quality-of-service. This is a big benefit for e-commerce sites since they would ideally prefer not to incur any loss of legitimate traffic as this translates directly to loss of revenue. In this context, the NetBouncer approach represents a clear and unique advancement over the current state-of-the art.

We have built an initial hardware-based prototype of NetBouncer as an in-line packet filtering device. Initial tests of the prototype have demonstrated promising performance but also exposed the limitations of a low-end network processor such as the IXP1200 in supporting functions beyond simple packet forwarding. A detailed presentation of these initial performance results is beyond the scope of this paper but is reported in [24].

Most notable is the fact that applying HMAC and cryptography increases the per packet overhead reducing throughputs and increasing latencies. The results we obtained used an initial implementation of the Michael HMAC algorithm. Additional optimizations and the use of more powerful network processors in the future should allow NetBouncer technology to scale up to 10 Gbps (OC 192) line rates and allow integration with edge and core and routers. For example, the anticipated release of Intel's second-generation network processor, the IXP2800, will feature 16 microengines and offer much more processing capacity and throughputs approaching 10Gbps.

As for ongoing and future work, we are currently completing the design of our two-tiered traffic management scheme and expect to implement it soon. We are also expanding our performance analysis experiments to understand the efficiency of various legitimacy tests. We will also be investigating the invention of additional service and session-based legitimacy tests. To aid in this task, we are currently collecting data and studying the most popular application-level protocols and services on the Internet. Our future work, we plan to study the performance of NetBouncer on real network traffic. We will also investigate the issues associated with integrating NetBouncer technology in various network topologies. In particular, issues associated with network address translation and asymmetric and multi-path routing require more study. We will also study deployment architectures where several NetBouncer devices may be deployed in a network and out-of-band commands are sent to one or more devices to "wake them up" to provide on demand DDoS filtering based on knowledge of which parts of the network are experiencing DDoS attacks.

References

[1] D.J. Bernstein, SYN Cookies, http://cr.yp.to/syncookies.html

[2] Peakflow/DoS, Arbor networks, http://www.arbornetworks.com/up_media/up_files/Pflow_Enter_datasheet2.1.pdf

[3] DDoS Enforcer, Mazu Networks, http://www.mazunetworks.com/solutions/product_overview.html#300

[4] Asta vantage System, Asta Networks, http://www.astanetworks.com/products/vantage/

[5] S. Capshaw, Minimizing the Effects of DoS Attacks, Application Note, Juniper networks, November 2000, http://www.juniper.net/techcenter/app_note/350001.pdf

[6] Strategies to Protect Against Distributed Denial of Service (DDoS) Attacks, White Paper, CISCO Systems, http://www.cisco.com/warp/public/707/newsflash.html

[7] Characterizing and Tracing Packet Floods Using Cisco Routers, Technical Note, CISCO Systems, July 1999, http://www.cisco.com/warp/public/707/22.html

[8] Configuring TCP Intercept (Prevent Denial-of Service Attacks), CISCO Systems, http://www.cisco.com/univercd/cc/td/doc/product/software/ios113ed/113ed_cr/secur_c/scprt3/scdenial.htm

[9] Cisco IOS Netflow, http://www.cisco.com/warp/public/732/Tech/nmp/netflow

[10] D. Dean and A. Stubblefield, Using client puzzles to protect TLS, *Proceedings of the 10th USENIX Security Symposium*, Washington, D.C, August 13-17, 2001.

[11] M. Degermark, A. Brodnik, S. Carlsson and S. Pink, Small forwarding tables for fast routing slookups, *ACM Computer Communication Review*, 27 (4), pages 3 -14, October 1997.

[12] A. Demers, S. Keshav, and S. Shenker, Analysis and simulation of a fair queueing algorithm, *Internet Res. and experience*, Volume 1, 1990.

[13] S. Dietrich, N. Long and D. Dittrich, Analyzing Distributed Denial of Service Tools: The Shaft Case, *Proceedings of the LISA XIV*, December 3-8, 2000, New Orleans, LA.

[14] R. Fan, A. Ishii, A. Itoh, M. Kobayashi, B. Mark, T. Muira, G. Ramamurthy, Q. Ren, S. Shibuya, H. Shimonishi and K. Yamada, ARC-LITE: An integrated quality-of-service ATM/IP switching-routing engine, In Proc. *IEEE ATM Workshop '99*, Koichi City, pages 161-166, May 1999.

[15] N. Ferugson, Michael: an improved MIC for 802.11 WEP, IEEE P802.11, Wireless LANs, MacFergus, Bart de Ligtstraat 64, 1097 JE Amsterdam, Netherlands, January 17, 2002.

[16] S. Golestani, A Self-Clocked Fair Queuing Scheme for Broadband Applications, *Proc. IEEE INFOCOM'94*, April 1994, pp. 636-646.

[17] R. Mahajan, S. Bellovin, D. Floyd, J. Ioannidis, V. Paxson, and S. Shenker, Controlling High Bandwidth Aggregates in the network, AT&T Center for Internet research at ICSI (ACIRI), DRAFT, February 5, 2001, http://www.research.att.com/~smb/papers/ddos-lacc.pdf

[18] D. P. Morrison, Practical algorithm to retrieve information coded in Alfanumeric, *Journal of ACM* 15, 4 (October 1968), pages 514-534.

[19] A. McAuley and P. Francis, Fast routing table lookup using CAMs, In *Proceedings of INFOCOM*, pages 1382-1391, March-April, 1993.

[20] S. Nilsson and G. Karlsson, Fast Address Lookup for Internet Routers, In Proceedings of the *International Conference on Broadband Communication*, April 1998.

[21] A.K. Parekh and R.G. Gallager, A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks - The Single Node Case, *IEEE Trans. on Networking*, June 1993, pages 344-357.

[22] K. Park and H. Lee, "On the Effectiveness of Route-based Packet Filtering for Distributed DoS Attack Prevention in Power-law Internets," in *Proceeding of the ACM SIGCOMM '01*, pages 15-26. 2

[23] D. Schnackenberg, K. Djahandari, and D. Sterne, "Infrastructure for Intrusion Detection and Response", *Proceedings of the DARPA Information Survivability Conference and Exposition*, Hilton Head, SC, January 2000.

[24] R. K. Thomas, B. Mark, T. Johnson, J. Croall, High-speed Legitimacy-based DDoS Packet Filtering with Network Processors: A Case Study and Implementation on the Intel IXP1200, To appear in the *Proceedings of the second Workshop on Network Processors - NP2*, February 8-9, 2003, Anaheim, CA.

[25] M. Waldvogel, G. Varghese, J. Turner, Bernhard Plattner**,** Scalable High Speed IP Routing Lookups, In *Proceedings of SIGCOMM '97*.