Bypassing Network Flooding Attacks using FastPass

Dan Wendlandt, David G. Andersen, Adrian Perrig
Carnegie Mellon University

Abstract

We describe the design and implementation of FastPass, a next-generation network architecture that thwarts bandwidth flooding attacks by providing destinations with fine-grained control over their upstream network capacity. Prior attempts to achieve network flood resilience have required destinations to successfully receive an initial unprotected packet (capability-based designs) or have relied upon global cooperation (filtering-based designs). FastPass requires neither. Instead, it allows destinations to distribute cryptographic *availability tokens* to potential senders that instruct routers to prioritize a limited rate of traffic from the sender in the case of network congestion. In contrast to prior architectures, we show that availability tokens provide two highly desirable DoS resilience properties: (1) hosts capable of identifying legitimate users can quickly communicate regardless of the size of the attack directed against them; and (2) hosts unable to differentiate between legitimate and malicious senders can strictly limit the ability of attackers to overwhelm incoming network capacity.

1 Introduction

An ounce of prevention is worth a pound of cure.

– Benjamin Franklin

Over the past ten years, the Internet has expanded at a frantic pace, changing the way people communicate, shop, conduct business, and find information. While this growth is likely to continue in many areas, we believe that a lack of guaranteed availability presents a significant barrier to the adoption of the Internet in many "critical infrastructure" settings that include near real-time response. For the Internet to support such applications requires a bound on the time to begin communication as well as a guarantee that subsequent communication cannot be disrupted.

Today's Internet architecture offers reachability, the ability to find a path to a particular destination, as its fundamental design goal [9]. In this paper, we advocate that the Internet of the future should position *availability* as the fundamental service provided by the network layer, giving destinations complete control over their incoming bandwidth. Such a shift would provide two strong properties: first, destinations capable of identifying legitimate users could prioritize such traffic to resist attacks of arbitrary size. Second, destinations unable to distinguish between good and bad users could require senders to commit something of value (e.g., a deposit or proof of work) before they could send traffic at even a low rate. Compared to the Internet's original sender-driven design, this "default off" approach to network forwarding priority places the destination in a significantly more powerful position to defend itself against network flooding attacks, while maintaining the traditional open architecture when no congestion exists.

FastPass provides hosts with an efficient and simple mechanism to leverage *arbitrary destination-specific policies to prioritize and rate-limit incoming network traffic at routers in the case that links become congested*. Each Internet destination, a public web-site, corporate VPN server, or a real-time device, can utilize its particular domain knowledge about incoming clients to vet incoming traffic while keeping the actual network infrastructure generic to the type of admission control performed. Furthermore, we provide these properties without introducing additional trust or cooperation requirements beyond what is assumed in today's Internet.

At a high level, FastPass enabled destinations distribute small (80 bytes) availability tokens to clients who are invited to open a connection to the server. Tokens are efficient cryptographic signatures created with the destination's private key. The corresponding public key is distributed with the destination's routing advertisements so that it reaches all Internet routers to be used for token verification. Tokens can be provided to the clients in advance or via an out-of-band channel. Potential token distribution mechanisms include:

- A user obtains a token from a large, distributed token provider, similar to today's content delivery networks. While the service being protected may be difficult to split or copy, token granting can easily be replicated and distributed for DDoS resilience.
- After a customer makes a purchase, an e-commerce site supplies the client host with a supply of tokens guaranteeing future access to the site at a limited rate.
- An online brokerage customer obtains a cryptographic authentication dongle from the company in order to securely access her account. The user's computer securely obtains a token from this dongle.

The tokens are then stored on the user's system until the client first attempts to contact the destination, at which point the client inserts the token data into the initial packet following the IP header before transmission. Using the public key distributed by the routing protocol, routers verify tokens for

each packet, prioritizing only the traffic with tokens containing a valid signature for the addressed destination. With this first packet protected, all future traffic can be protected using additional tokens supplied by the destination, or more likely, using efficient mechanisms such as router-based network capabilities (see Section 3.6).

FastPass differs from past capability schemes by not requiring an initial unprotected setup packet that is vulnerable to *Denial-of-Capability* (DoC) [3], an attack in which floods of set-up packets prevent legitimate set-up requests from reaching the destination. Furthermore, in solving this problem FastPass does not rely on remote domains to perform filtering on a victim's behalf, as is the case with coordinated filtering approaches. FastPass does not require that the destination be able to perfectly distinguish between good and bad clients, but instead gives destinations the power to effectively control the rate of incoming traffic in either case. As a result, the malicious use of tokens is self-limiting.

Contributions: FastPass is a novel architecture that protects *all* types of traffic, including connection establishment and datagram packets, yet requires no additional public key infrastructure, has no single point of failure, and requires no additional trust relationships beyond what is already required to forward traffic. While our description and discussion of FastPass raises some new research questions and design debates, the architecture presents a novel solution to the unsolved Denial of Capability vulnerability and therefore represents important progress toward building a future Internet architecture resistant to flooding attacks.

2 Related Work

Early network flood defense research used preventative approaches that traced packets back to the host or network from which they originated [6, 20, 21], or forced attackers to use their real IP addresses [11].

More recent approaches have favored attack *prevention*, by having routers make filtering or prioritization decisions in the face of congested buffers. Early router-based proposals [12, 19] considered the feasibility of routers identifying and filtering aggregates of attack traffic. With limited knowledge, however, such router-based filtering is imprecise, providing no strong guarantees that sufficient quantities of attack traffic will be blocked or that no legitimate traffic will be filtered.

In response, researchers proposed two classes of DoS resilience schemes utilizing explicit signaling by network destinations: network capabilities and cooperative filtering. In network capabilities schemes such as SIFF [27] and TVA [28], a source sends a connection setup packet which is cryptographically marked by each router along the path to the destination to create a source and destination-specific bit-string called a "capability". The destination can choose to echo this data-chunk to the original source for inclusion in later packets to indicate the destination's desire to receive traffic from that host. Routers cryptographically verify the authenticity of capabilities as they forward packets, giving priority to capability protected traffic in the case of congestion. Capability-based systems possess an appealing simplicity: they require no per-flow state in routers or message exchanges between routers and end-hosts and perform only local operations to mark and later verify capabilities. Nonetheless, routers acquire the knowledge needed to accurately filter unwanted traffic in a fine-grained fashion.

Unfortunately, the Achilles heel of current capability systems is the requirement that a sender's set-up packet must reach the destination without any prioritization, opening this set-up channel to Denial-of-Capability (DoC). [3]. SIFF relied on legitimate set-up packets competing with attack set-up packets until one reached the destination by random chance. TVA provided improved robustness to floods on the request channel by having each router fair-queue request packets based on its network ingress point at the congested network domain. The fair-queueing attempts to isolate requests from "good domains" from requests from domains containing attackers, yet our evaluation shows that even this current approach is vulnerable to a DoC.

As an alternative, AITF [4] proposes that destinations communicate host-specific filtering requests to remote domains to filter attack traffic. Scalability requirements, however, force AITF to make a significant security assumption in order to offer fine-grained destination specific control: filtering happens only at the network edge, meaning that the cooperation of domains hosting attacking sources is needed if the scheme is to distinguish itself from simple destination filtering. FastPass achieves both the stateless and local simplicity of network capabilities and the ability to protect set-up and single datagram traffic offered by cooperative filtering.

Similar to the "Off by Default!" work of Ballani, et al [5], the FastPass design is motivated by the long-standing security principle of *default deny*. However, unlike Ballani we apply this only in the case of network congestion and do not attempt to control non-flood traffic that could easily be filtered by a local firewall. Additionally, the default-deny portion of Ballani's architecture has properties that differ from the goals of FastPass: (1) a destination must always knows the IP address of permitted senders to "turn on"; and (2) a sender must tolerate a non-bounded average waiting time of > 30 seconds to establish a connection.

FastPass's ability to leverage third-party infrastructure for token distribution is similar in motivation to the seminal capability architecture proposed by Anderson, et al.[2], which unlike FastPass, *required* a trusted and coordinated infrastructure of servers to forward "request to send" packets all the way to destinations for approval. Overlay-based approaches such as Mayday [1] and SOS [14] aim to protect servers by deploying networks of nodes that filter traffic on their behalf. While this approach has the advantage of being compatible with today's Internet architecture, its requirement for a large network of proxies means that it is primarily suited to protecting a few well-funded services that can afford such an overlay network, and less suited to

protecting smaller services or individual clients.

While FastPass and related schemes address packet floods, other attacks target a server by targeting it with expensive application-layer requests. Approaches such as Kill-Bots use Captchas to rapidly distinguish human- and machine-generated requests at Web servers [13]. This work complements FastPass: a successful approach to DoS mitigation must address attacks at all layers.

3 FastPass Design

This section presents the FastPass architecture, describing the goals and trade-offs considered for tokens, key distribution, and router forwarding. We also describe features of the protocol that improve overall packet processing efficiency and manage the malicious replay of tokens.

3.1 Problem Statement

FastPass aims to provide connectivity in the face of attacks that flood network links upstream of an intended victim.

We define *Time to Communication* (TTC) as the time required to successfully establish an "un-interruptible" connection protected by capabilities or filters. We measure TTC beginning when the source first desires to initiation of communication (e.g., sending a TCP SYN packet). In a capability-based scheme, TTC ends when the sender receives a reply to its capability request. In a reactive filtering-based scheme like Pushback [12] or AITF [4], TTC is the time required to install enough filters to allow normal communication.

Mission-critical applications and services should obtain a guarantee for connection establishment within a small, bounded amount of time. To support this, FastPass has as its primary goal to give destinations complete control over their incoming network capacity to enable such applications to obtain a predictable and small TTC. This property should hold even in the face of large traffic floods and be independent of the network capacity, the number of attackers, and the network topology between the senders and receivers. In effect, resilience to floods should depend *only on the amount of incoming traffic already authorized by the destination, not on the resources of the attacker*. This property is not achieved by current DoS-prevention mechanisms.

3.2 Overview

In FastPass, hosts obtain a *traffic authorization token* that allows them to connect to a destination even in the case of network congestion. A host may obtain these tokens in a variety of ways, as noted in the introduction: directly from the destination at an earlier time, from a third party service, or via an out-of-band mechanism.

The basis of FastPass's tokens is a destination-specific signature, which routers can verify using the destination's public key to confirm that a destination has indeed authorized the token-holder to send traffic to it.

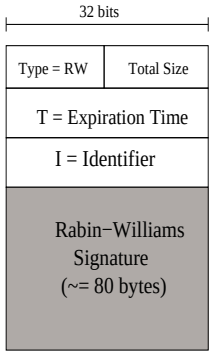

Figure 1: The FastPass token layout.

At a high level, FastPass operates as follows:

1. A destination domain D generates a public/private key pair, K_D and ${K_D}^{-1}$.
2. D distributes its public key K_D along with its reachability information in a BGP-like inter-domain routing protocol. At each AS, this key is injected into the IGP. Each router thus has the public key of all destinations in its routing table.
3. D uses its private key ${K_D}^{-1}$ to run a token granting service, or provides pre-made tokens to a trusted third party to distribute tokens on its behalf.
4. When a source S knows that it will want to connect to D in the future, it acquires a token from D's token granting service.
5. To provide a token, D uses the private key to create a digital signature proving that the token-owner has the right to contact D. The token consists primarily of the signature itself, as well as some meta-data for expiring tokens and verifying uniqueness.
6. When S contacts D, it includes the token in its first packet.
7. When forwarding packets, the router uses D's public key K_D (acquired in Step #2) and meta-data in the packet to validate the token.
8. Packets with valid tokens are placed in a high-priority traffic queue for forwarding. All other traffic is forwarded best-effort.
9. If D wishes to continue communicate with the client, its reply will include a limited number of new tokens authorizing additional traffic.

3.3 FastPass Availability Tokens

The primary goal of FastPass is to provide control over the rate of *all incoming traffic*. This requires routers to verify that a destination wishes to receive a packet without any active input from the destination (e.g., echoing a capability in the case of route-based capability schemes).

FastPass tokens (Figure 1) consist of a short message along with a digital signature of that message, which guar-

antees the authenticity of the token when it is verified by a router. The message consists of two values: a 32-bit expiration time T and I, a 32-bit identifier field unique to all currently valid tokens distributed by a given destination. The signature S is a Rabin-Williams (RW) [17] signature of these two values under the private key of D: $\{T \mid\mid I\}_{K_D^{-1}}$. The signature ensures integrity and authenticity of the token information, preventing alteration of the identifier or the expiration timestamp. Rabin-Williams is known for extremely fast signature verification, which requires only a single modular multiplication.

Tokens deliberately do not contain a source address. As a result, tokens can be given to an individual without knowing anything about how that person will eventually contact the destination. This design choice is fundamental to our goal of supporting a flexible notion of identity: specifically, that the identifier used by a destination to inform the infrastructure about its preference for or against a flow should be flexible and topology independent. Architectures like TVA use a router-based packet-marking scheme similar to Pi [26] to identify network aggregates for filtering. However, this approach often results in *false positives* when legitimate traffic located near a malicious source is also filtered. FastPass, in contrast, can even differentiate between different hosts sharing a single NAT IP address.

Tokens can be generated by standard PCs and may be distributed either *preemptively* before an attack occurs or *on-line* through the use of an always available out-of-band mechanism. Destinations may wish to apply vastly different policies for allocating network resources. Private sites may allow access only to expressly authorized users, while public sites may require proof of computation or a Captcha [22] (reverse Turing test) before granting access. Each of these policies has merit in different circumstances; an infrastructure should not mandate a particular one. A Captcha, for example, would be inappropriate for a service accessed primarily by other programs, and computational puzzles can be biased against small clients such as PDAs. This generality is a major strength of the design (we discuss potential policies in greater detail in Section 6).

In addition to standard capability functionality, end-hosts will need to manage their tokens, storing acquired tokens and potentially refreshing tokens that are about to expire (no client cryptographic processing is needed). Token management software might also assist in the transaction between the host and the token granting service, e.g., by prompting a user for a user-name and password or performing computation as a proof-of-work. We make no assumptions that tokens are protected in the case that end-hosts are compromised.

3.4 FastPass Keys and Trust Model

From a security perspective, our primary design goal was to create an architecture that requires no additional trust beyond what is already needed to forward traffic and has no heavyweight public key infrastructure (PKI) or single point of failure.

FastPass keys operate at a one key per domain granularity; roughly what is today represented by a stub atomic system (AS). More specifically, we borrow from the idea of a "Failure Atomic Unit" in routing [23], and assume that the indivisible units of routing on the core Internet are directly tied to a particular organization and to a group of machines that are "linked by failure" (e.g., they share a common bottleneck, access link, etc.). While this granularity is not strictly necessary—FastPass could operate at smaller granularities—per-domain is a better match in the face of route aggregation and limited routing table sizes.

Although the Rabin-Williams public keys used in FastPass could be signed by a "root of trust" and distributed as public-key certificates using a PKI, we propose to distribute the public keys directly through the inter-domain routing protocol along with the reachability information. In general, overloading a protocol with additional functionality and requirements is not advisable. However, in this case, the public key information can itself be viewed as additional reachability information that is required in the case of congestion. Since the token verification process is intimately tied to the forwarding process for reaching the destination, distribution through the routing protocol is preferred to avoid inconsistencies between routing state and public-key information. We can avoid a PKI because traffic's "trust" mostly follows the reverse path of route updates in BGP—thus, a malicious router that could modify the public key would already be on the path towards that destination. While a secure routing protocol is needed to prevent other serious control-plane attacks against destination availability, FastPass requires no additional security.

With FastPass, only routing prefixes from the same AS (i.e., using the same public key) can be aggregated and still be used to perform destination-based public key look-up operations. While some compelling proposals for future router architectures make this point moot by utilizing a flat address-space [23], we recognize this as a potential downside of the use of routing to distribute keys. If only some packets require token verification (see Section 3.6), routers could still aggregate locally to keep forwarding tables small, and perform key-lookups in a larger table with higher memory latency which impacts only packets subject to token verification.

3.5 FastPass Router Processing

FastPass-enabled routers must prioritize and forward packets in a manner that is simple to understand, thereby allowing destinations to easily reason about its token granting policies. Conceptually, FastPass routers have two output queues per interface: authorized traffic and best effort traffic. The former is strictly prioritized over the latter.

When a router receives a packet, it checks to see if the packet contains a token. If no token exists, a next-hop lookup is performed and the request is placed in the best-

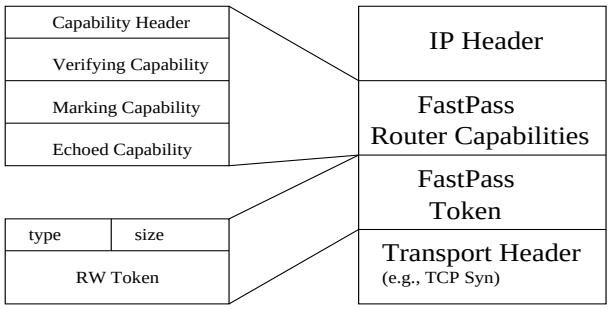

Figure 2: Using router-based capabilities as a path-specific optimization, FastPass adds two headers to a packet following the IP header. The capability header contains fields for three capabilities: one to be verified for priority forwarding, one to be marked by the router to build a new capability, and one to be echoed back to the remote host.

effort queue for forwarding. If there is a token, the router first checks the expiration time T to see if the token is still valid. This function requires coarse time synchronization depending on the granularity required for token expiration (we envision this to be on the order of minutes). Next, the router uses a Bloom Filter-based [10] duplicate detection scheme (Section 3.7) to test if it has seen the token ID I before. If the token is expired or a duplicate, a route lookup is performed and the packet is placed in the best-effort queue with the token removed.

If T and I are valid, the router looks up the next hop and the public key associated with the destination of the packet. The router then checks the Rabin-Williams signature of the message consisting of T and I and places packets with verified tokens in the authorized traffic queue if capacity exist. If no capacity exists, the token is forwarded best-effort. Packets with invalid tokens are stripped of their token and also transmitted best effort. We discuss the feasibility of per-packet cryptographic processing in Section 4.3.

Once the packet arrives, the destination determines whether or not to echo one or more new tokens to the client in a reply packet. Containing a valid token in no way implies that the destination will grant additional tokens for the client to send more traffic in the future. If a destination wishes to reduce the rate of an incoming flow, it can simply reduce the number of tokens it hands out to this host. If desired, the token-protected packets from the client can also include as data a return-path token to protect the server's reply packets.

3.6 Path-specific Optimizations

While simple and effective, the token-based scheme as described above is resource intensive for both routers verifying tokens and the Internet destinations generating new tokens for clients. However, once connection setup is complete, the use of tokens can easily be replaced by a path-specific protection scheme such as the one described in TVA. When using this optimization, each packet (including the initial token request, depicted in Figure 2) would include a capability header, with tokens protecting any packets that have not yet acquired valid capabilities.

TVA assumes that capability request traffic is limited to 5% of link capacity, with the remainder of the link available for capability protected and best-effort traffic. For the remainder of the paper we take a similar approach, but estimate the percentage of overall traffic containing tokens to be 10% in order to include datagram traffic such as DNS. For each outgoing interface, a router can only transmit packets with tokens at 10% of the link rate. If more valid requests exist for an outgoing queue then capacity to transmit them, as many as possible are checked and the remainder have their tokens stripped and are forwarded best-effort.

Router-based capability schemes allow destinations to specify a "permitted sending rate" for each client, which the destination can modify every few seconds with a refreshed capability in order to throttle incoming traffic. Thus, for long-lived flows, capabilities provide the same rate-limiting protection offered by tokens in the face of malicious hosts with previously granted access.

3.7 Throttling Token Replay

An attacker with a single token must not be able to reuse this token to mount a flooding attack by repeatedly sending the same valid token. FastPass ensures that a token can only be used infrequently on any given path, and as a result, only a small number of duplicated tokens will reach the bottleneck links near the victim.

FastPass provides these properties by hashing the token's ID and the destination AS into a Bloom filter and rejecting ID/destinations pairs that have appeared in the previous t seconds. This approach limits token reuse to once per t seconds through any Internet router. Even if k tokens are acquired and shared among many different hosts in a massive "bot-network", attackers will be limited to flooding $\frac{k}{t}$ packets per second through any network bottleneck that may exists near the destination. This effectively renders even large bot-networks unable to significantly benefit from token replication.

Bloom filters provide compact lookups, have no false negatives, and allow false positive probabilities to be driven to arbitrarily low rates with the use of additional memory. To illustrate this, consider a router with a gigabit link on which 10% of the capacity is allocated to token requests and a circular buffer of t Bloom filters, with each filter containing one second worth of token traffic. The router will process up to 80,000 tokens/sec, with each of the t filters being checked to see if a token is duplicated. If k different hash functions are used, inserting n tokens into a filter of size m bits gives a false positive probability of $(1-(1-\frac{1}{m})^{kn})^k \approx (1-e^{\frac{kn}{m}})^k$. With the optimal k value of $k = ln\ 2(\frac{m}{n})$, this is approximated by $(0.6185)^{\frac{m}{n}}$. Thus, a filter of 300 KB can prevent duplicates for one second with a false positive probability of under $(0.6185)^{\frac{2,400,000}{80,000}} < \frac{1}{10^6}$

per packet. A circular buffer configuration of t Bloom filters can therefore filter traffic for t seconds with less than $(1-(1-\frac{1}{10^6})^t) \approx \frac{t}{10^6}$ false positive probability per packet.

The duration of duplicate suppression within all Internet routers must be bounded from above and below in order to assist destinations in planning token distribution (see Section 6).

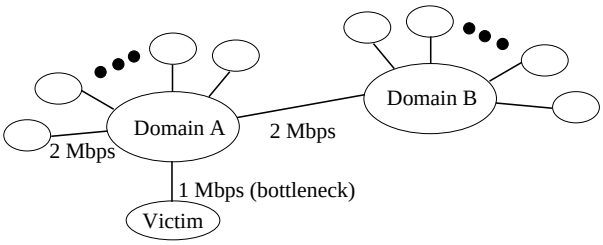

Figure 3: The dual domain topology, each with 20 stub networks. Domains A and B are connected via a 2 Mbps link, while Domain A is the sole provider of access for the victim over a link of of 1 Mbps.

4 Evaluation & Analysis

This section presents an experimental evaluation of FastPass compared to the recent capability-based DDoS defense architecture outlined in TVA [28].

4.1 Click Router Implementation

We have implemented FastPass router and end-host processing in the Click modular router [15], running on top of a generic capability system. The implementation uses the Rabin-Williams implementation using SFS [16] and the GNU MP library. Due to the lack of public-key cryptographic support in the Linux kernel, we run Click at user-level, which limits our implementation's forwarding capacity to 19,200 152 byte non-token packet-per-second (pps). When forwarding token traffic, the router is bounded by cryptographic processing to a maximum rate of 12,400 pps. Additionally, we implemented a simple single-threaded token-granting server for use in the evaluation. The router, end-host, and server code is open-source and available at `http://www.cs.cmu.edu/~dwendlan/fastpass/`.

We ran experiments using the Emulab [25] pc3000 hosts equipped with 3GHz 64-bit Intel Xeon processors and 2GB RAM. Because of userlevel Click's limited forwarding capacity, we artificially constrain the emulated "access" link capacities and *simulate only the token channel* (i.e., the 10% of the link used to forward token packets). In the scenarios described below, the token channel for a "transit link" is 2 Mbps and the channel the bottleneck link is 1 Mbps. With 10% of a link dedicated to requests, this setup represents 20Mbps and 10Mbps links, respectively. We limit each attackers to a quarter of the victim's token capacity (250 kbps). Each stub network is also assumed to be connected to the token granting server by a connection (not shown) that is already protected either by another token or through over-provisioning. Links between transit domains have a latency of 10 msec and links to the token granting server 5 msec, while stub networks experience no delay in reaching their transit domain.

While many factors, including the link rates, attack rates, and total number of attackers are scaled down from a realistic Internet attack, the high-eve trends exhibited in these experiments are sufficient to highlight several key distinctions between FastPass and TVA.

In our experiments, each Emulab host emulates a single large Internet domain, similar to a transit AS. Domains represent different *trust domains* in the network, and so both TVA and FastPass make prioritization decisions for packets as they enter a new domain. As described in Section 2, TVA fair-queues requests based upon their ingress interface into the current domain. To provide a more realistic topology, each domain connects to a number of stub networks that are emulated on the same physical machine for our experiments, but are treated as separate networks by TVA's filtering. These stub networks represent customers or peers of the transit networks, and are connected to the transit domain with 2Mbps links.

For each experiment run, both transit domains host the same number of attackers (the number of which varies by trial) and legitimate hosts (held constant at 50 for each trial). Legitimate hosts are randomly assigned to a stub network of the transit domain. Attackers constantly flood packets to the victim, while legitimate clients send their initial packet at a random time, and resend with an aggressive but reasonable timeout interval of 500ms until a connection is established.

We limit the number of stub networks that can contain attackers to at most one-half of the stub networks for each transit domain. This helps TVA, which attempts to isolated bad traffic by inspecting its last-hop domain. Without this, TVA effectively converges to the same best-effort forwarding of set-up requests provided by SIFF.

4.2 Time to Communication

We calculate Time to Communication (TTC) as the time from when a destination first desires to connect to the victim until the time the reply (with protecting capability) is received. TTC thus includes the time the packet is queued, time spent waiting for a timeout as a result of packet loss, as well as propagation delay. For token-protected traffic, TTC also includes the time to acquire a token from the token granting service.

We compare the TTC values provided by FastPass to those achieved using TVA's AS-input marking to highlight key differences between the two architectures. Our topology is a simple chain of two "transit domains", each with 20 connected "stub networks". We chose this topology because

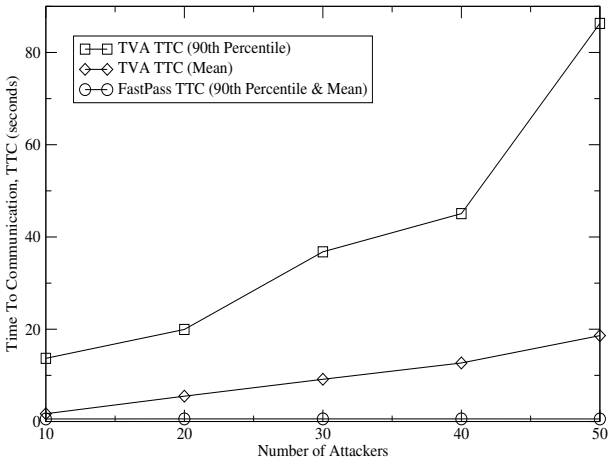

Figure 4: Average and 90th percentile TTC value for FastPass and TVA as attack size increases. Because FastPass limits the incoming attackers to a very low rate of prioritized packets, it scales significantly better in the face of attacks.

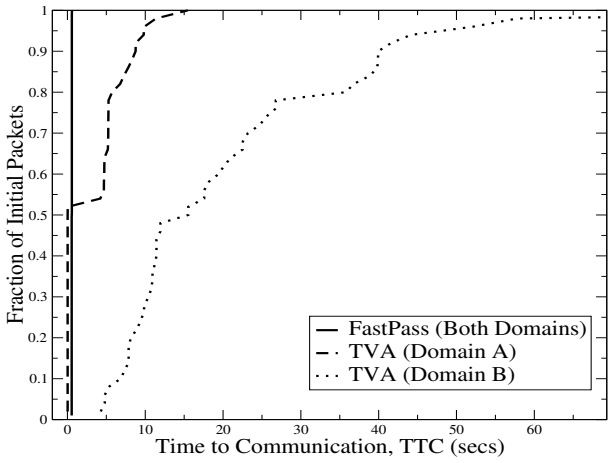

Figure 5: CDF showing FastPass and TVA performance for the 40 attacker trial. The FastPass lines for both domains overlap, but the TTC for TVA clients in domain B is significantly degraded.

it is capable of showing TVA's best and worst cases while also demonstrating FastPass's topology independence.

For FastPass experiments, *we do not assume that the token granting server can distinguish between good and bad clients*. Instead, the server implements a simple policy of limiting clients to a single new token per minute with tokens valid for one minute (effectively, the duration of the experiment). Routers suppress duplicates for 5 seconds, meaning the the vast majority of attackers packets to not receive priority because they are detected as duplicates.

Figure 4 shows the TTC each scheme provides for a given number of attackers. Because TVA provides no mechanism for limiting incoming traffic, TTC increases along with the number of attackers, quickly yielding significant waiting times. In contrast, FastPass's TTC is constant and small since the total amount of prioritized traffic from both legitimate senders and attackers is strictly limited. While the inefficiency of our user-level implementation limits the size of attacks we could measure, the TTC for FastPass would have continued to remain largely constant until enough attacker existed to overwhelm the 1 Mbps link with non-duplicated tokens (see Section 6 for related analysis).

Figure 5 shows a CDF of TTC values broken down by domain for the trial with 40 attackers. From this we see that TVA TTC values have a heavy-tailed distribution, which is the result of legitimate senders being forced to share per-domain queues with attack traffic from the same stub domain. This highlights TVA's inability to perform fine-grained filtering.

Figure 5 also demonstrates the impact of topology on TVA. Notice that domain A is ideal for TVA, because the bottleneck router can perfectly distinguish "good" and "bad" domains. Because we only allowed attackers in half of all stub domains, half of the TTC values in domain A are essentially zero. The significantly degraded performance for hosts in domain B results from the fact that both legitimate and illegitimate traffic "mix" and become indistinguishable prior to reaching the bottleneck, where prioritization is most desirable. In fact, if we consider a source d domains away from the victim, where each domain connects to i other domains, we see that for TVA the worst case TTC scales *exponentially* as $O(i^d)$.

Because FastPass does not prioritize traffic based on topology, its TTC is the same for both domains. The TTC for all FastPass requests is slightly larger than the best TVA time because of the additional latency of communicating with the token-granting service and generating a new token.

4.3 Implementation Considerations

Our implementation of the FastPass tokens uses the extremely fast Rabin-Williams signature scheme, which requires only a single modular multiplication for signature verification. FastPass is also practical because tokens are included only in a small fraction of total traffic—initial setup packets or datagram traffic (e.g. DNS). Most traffic is protected by path-specific and highly efficient router-based capabilities, leaving perhaps 10% of the capacity of the network links for token traffic. Furthermore, tokens must be verified only once per network "trust domain" (i.e., an AS or collection of cooperative AS's), with internal routers trusting the verification of the border routers.

FastPass in Hardware: How fast could a dedicated hardware implementation of a FastPass router perform? Rabin-Williams verification is based upon modular multiplication of very large numbers. Most hardware implementations of RW use optimized Montgomery multiplication [18] to achieve very fast ASIC and FPGA results. Our conservative back-of-the-envelope calculations estimate that a contemporary implementation could achieve over $\frac{1}{2}$ million ops/sec in a high-end FPGA, and nearly 2.5 million ops/sec

Operation	single op	rate
RW verify	61 μsec	16,459 / sec
RW sign	3.4 msec	297 / sec

Table 1: 1024-bit Rabin-Williams software benchmarks.

in a custom ASIC [24]. This rate is 10% of the forwarding rate of the fastest OC-192 line card available for Cisco routers [8]. While an ASIC implementation may be expensive, the line cards that would require it already cost in excess of $100,000.

FastPass in Software: While hardware implementations are the likely choice for high-speed routers, lower-end routers may also need to verify tokens. In addition, tokens will often be generated using RW signing by servers running commodity hardware. Table 1 shows micro-benchmarks on a 3.2Ghz Pentium-IV based PC with 1GB of RAM. Kernel Click routers can forward in excess of 1Mpps [7], meaning signature verification is the clear bottleneck for a software router. We therefore conservatively assume that a kernel-Click router performing FastPass token verification could support 16kpps of token traffic, which then permits 160kpps of data traffic. At an average packet size of 512 bytes, such a router could support 655Mbps.

Recent hardware trends are also very favorable for a software FastPass implementation. First, newer 64-bit architectures allow modular multiplications to be performed in fewer total iterations. Second, each packet's RW signatures can be verified independently. This type of "embarrassingly easy" parallelism is ideally suited to emerging multi-core processors.

5 Oversubscription & Provisioning

FastPass provides a reliable mechanism for receivers to inform the network whether incoming traffic is desired or not, enabling destinations to protect legitimate traffic but quickly limit and stop malicious network floods. However, no DoS defense system can let a destination handle more than its upstream capacity allows, even if all requests are legitimate. Additionally, since most links in the Internet are shared by multiple destination entities, these entities "share fate" in that they are impacted if the group collectively allows in more traffic than the shared link can carry. We refer to this problem as *link oversubscription* and term the deliberate creation of such congestion by malicious parties a *collusion attack*.

We note that in general, any system that determines priority based on destination authorization is vulnerable to this type of attack. For example, filtering schemes such as AITF permit victims to filter only traffic that is addressed to them; they cannot block traffic destined to other hosts, even if that traffic congests their bottleneck link. The same problem was noted as a challenge to TVA. Broadly, the existence of over-subscription and collusion attacks can be thought of as a trade-off required to achieve the highly desirable properties of fine-grain filters and low false positives provided by schemes leveraging destination-specific input.

Because of this "fate-sharing" property, architectures that use destination specific input are best suited to protect links near the network edge and cannot strictly guarantee availability in the core of the Internet, where capacity is shared by a large number of destinations. Fortunately, destination-based fairness or per-destination capacity limits offer a good—though not perfect—solution to this problem for networks looking to provide strict guarantees to particular customers. ISPs can use a number of simple, effective tactics to reduce the power of collusion attacks. For example, the ISP can push a set of limits to its border routers. Examples of effective limits include:

Per-destination-AS fair queueing: Ensure that traffic is shared between destination autonomous systems, perhaps with appropriate weights to account for large and small customers, or those who pay different amounts.

Customer capacity limit propagation: If a customer has an access link capacity of x, then no border router should pass traffic to the customer at a rate no higher than x.

Elephant squashing: Like the monitoring scheme used in SIFF and TVA, monitor the largest F flows and limit them so that no destination AS consumes more than $\frac{1}{F}$ of the total link capacity.

While such limits are not perfect, they greatly reduce the effects of collusion attacks. Interestingly, these tactics are completely *ineffective* against normal DDoS attacks that come from widely dispersed sources: under such an attack, they merely push the packet loss to the edge, but good flows still experience vastly increased loss. Only in conjunction with an architecture such as FastPass do they show their value. With each of these schemes, the fairness decisions can be made on a *per-AS* basis instead of a per-flow basis, requiring far fewer router resources to implement. The limits would likely be tied in with a customer's service level agreement (SLA) that already exists to govern latency and throughput guarantees.

Closer to the source, protection for individual senders can easily be provided by fair queueing on a per-sender basis. At this point, the level of aggregation is relatively low, and ISPs already have per-sender configuration information making this process extremely practical.

Finally, indirect attacks are most powerful near the victim – originating from hosts sharing the same upstream ISP, for instance – where they are also more susceptible to correction through local means. In contrast to directed flooding attacks which may originate from anywhere in the world, indirect attacks are significantly more amenable to correction by the enforcement of AUPs and applicable laws.

6 Managing Token Distribution

FastPass intentionally does not specify a particular policy for deciding what a client must do to merit a token, how

long that token is valid for, or how tokens are distributed. Leaving this decision to the end-point is a strength of the FastPass design, but we discuss potential strategies in this section to give the reader an intuition for why tokens enable a powerful defense against network floods.

6.1 Token Accounting

Recall from Section 3 that each router blocks duplicate tokens for t seconds. To assist destinations in making token distribution decisions this value should have global upper and lower bounds.

The upper bound (t_{upper}) is the maximum time a legitimate user that has been granted only a single token must wait before reusing the token if the initial packet carrying the token is lost due to token-channel oversubscription or network errors. More highly trusted clients may receive several tokens to guard against the impact of single token loss.

The lower bound (t_{lower}) guarantees that a (perhaps malicious) client with a single token is limited to the rate of $\frac{1}{t_{lower}}$ pps through any router until the token expires. This rate, combined with knowledge of the capacity provisioned by its upstream provider, allows a destination to make intelligent token granting decisions.

If the upstream provider guarantees x tokens per second of forwarding capacity through any potential bottleneck, then to maximize availability, a destination should not issue more than x tokens for any particular time-period. Token replication and replay by a distributed attacker provides no increased threat, since any particular bottleneck cannot be flooded using duplicate tokens. The use underlying capabilities, which can easily be revoked on the order of seconds, reduces the number of tokens that a destination must hand out and greatly simplifies token management for destinations.

Such a strong availability guarantee may be ideal for highly critical network services, but a more common website token-granting policy would likely allow some statistical multi-plexing over time, depending on the level of trust the destination has in its policies for distinguishing legitimate users during token distribution.

Viewing tokens (and traffic permitted by the underlying capabilities) as the destination admitting rate-limited flows demonstrates how FastPass can be useful even to public sites. For example, a site may hand-out a token to any user when lightly loaded, but require increasingly difficult computational puzzles per token granted as the total number of requests increases so that it never distributes more tokens than it has capacity to accept. Such an approach could even be effective in maximizing link good-put in the case of a flash-crowd, since the maximum sustainable number of users can be admitted and have guaranteed bandwidth. More generally, the scheduling of a scarce resource is a well-studied problem and tokens allow destinations to choose an arbitrary scheduling policy for their incoming bandwidth.

Token accounting can also leverage accountability if the token-granter records information linking an individual to the ID field of token that is later used in a token flood. This fact opens the door for interesting economic incentive schemes in which users (or their ISP) pay a deposit and then freely acquire tokens, forfeiting the deposit only if tokens are misused.

6.2 Third-Party Token Granting

The discussion of token accounting also highlights an important fact impacting a destination's choice of token distribution mechanisms. An on-line token granting service (i.e., any service such as Akamai that is always available to grant tokens) provides better control for a destination managing its incoming bandwidth. This is because on-line token services can hand-out tokens valid only for a short duration, since the user can always simply return to acquire another one once a token expires. In contrast, if a destination relies on giving out tokens ahead of time for use in future days or even weeks, it must have a better idea of who its legitimate users are. Thus, preemptive token distribution is likely to be most valuable for private destinations that can easily identify users, while public destinations would be better served by an on-line model.

Having a limited number of third-party availability providers with extremely well-provisioned and geographically diverse resources is also attractive from a resource conservation perspective, as it requires only a very few Internet destinations to buy the network capacity to sustain massive attacks of unwanted traffic, while the remainder of the Internet uses these sites for token distribution. This design in part mimics the root-servers of the DNS system which have achieved strong network flood resilience through the use of of over-provisioning, replication, and anycast.

Third-Party servers make use of the fact that token distribution, unlike many of the complex services offered on the Internet, is easily replicated and distributed. These third-party servers may serve simply as an initial gate-keeper by enforcing a simple criteria for granting tokens — a reverse turing test or the completion of a computational puzzle. More sophisticated validation could also occur without any potentially undesirable secret-sharing, if the destination keeps its token-granting key private and instead periodically provides the third-party with a new supply of signed tokens to distribute. Likewise, if the third-party was using a login to authenticate legitimate users (as might be the case with an e-commerce merchant) the site could supply the third-party with only a one-way hash of both the customer's user-id and password as authenticators without disclosing any sensitive information.

6.3 Heterogeneous Destination Domains

Up to this point, our discussion treated a destination primarily as a single entity, when in reality a single key may

cover a large number of different participating parties, as is the case for a web-hosting provider or a large university. Potentially, all destinations with the same key would be at risk if a single party is compromised or is otherwise promiscuous with granting tokens. However, such parties almost certainly already have an existing relationship with the key-owning entity running the AS. This leads to the straightforward use of a system in which the AS keeps the private token-signing key secret but provides tokens to different parties in its domain using a quota system to minimize security dependencies among the parties. For added security, the AS possessing the private key could use a bastion host that is only locally addressable and configured to run minimal services.

7 Conclusion

Time-to-communication (TTC, i.e., the duration to set up a protected channel of communication) is a likely to be a critical metric for any DDoS resilient architecture as Internet services become increasingly critical. Recent capability systems provide many useful properties but are insufficient to protect connection set-up and datagram traffic.

FastPass provides a comprehensive and robust DDoS defense capable of protecting all traffic, yet free of assumptions about inter-ISP cooperation that hamper deployment and security. Our analysis and evaluation show that FastPass drastically outperforms prior systems, even when a destination cannot easily identify legitimate clients, and that our design could be implemented with modest hardware updates. While much further work and evaluation remains, a token-based network flood resilience scheme like FastPass provides a compelling concept for the foundation of a future availability-based network architecture.

References

[1] D. G. Andersen. Mayday: Distributed Filtering for Internet Services. In *Proc. 4th USENIX Symposium on Internet Technologies and Systems (USITS)*, Mar. 2003.

[2] T. Anderson, T. Roscoe, and D. Wetherall. Preventing Internet denial of service with capabilities. In *Proc. 2nd ACM Workshop on Hot Topics in Networks (Hotnets-II)*, Nov. 2003.

[3] K. Argyraki and D. R. Cheriton. Network capabilities: The good, the bad, and the ugly. In *Proc. ACM Workshop on Hot Topics in Networks (Hotnets-IV)*, Nov. 2005.

[4] K. Argyraki and D. R. Cheriton. Active Internet traffic filtering: Real-time response to denial-of-service attacks. In *Proc. USENIX Annual Technical Conference*, Anaheim, CA, Apr. 2005.

[5] H. Ballani, Y. Chawathe, S. Ratnasamy, T. Roscoe, and S. Shenker. Off by default! In *Proc. ACM Workshop on Hot Topics in Networks (Hotnets-IV)*, Nov. 2005.

[6] S. Bellovin. *ICMP Traceback Messages, Internet-Draft, draft-bellovin-itrace-00.txt, Work in Progress*, Mar. 2000.

[7] G. Calarco. High performance click router. `https://amsterdam.lcs.mit.edu/pipermail/click/2004-November/003364.html`, Nov. 2004. Message to the click-users mailing list.

[8] Cisco Systems. Cisco 12000 series one-port oc192c/stm-64c dpt line card, Feb. 2006.

[9] D. Clark. The Design Philosophy of the DARPA Internet Protocols. In *Proc. ACM SIGCOMM*, pages 109–114, Stanford, CA, Aug. 1988.

[10] L. Fan, P. Cao, J. Almeida, and A. Z. Broder. Summary cache: A scalable wide-area Web cache sharing protocol. In *Proc. ACM SIGCOMM*, pages 254–265, Sept. 1998.

[11] P. Ferguson and D. Senie. *Network Ingress Filtering*. Internet Engineering Task Force, May 2000. Best Current Practice 38, RFC 2827.

[12] J. Ioannidis and S. M. Bellovin. Implementing Pushback: Router-Based Defense Against DDoS Attacks. In *Proc. Network and Distributed System Security Symposium (NDSS)*, Feb. 2002.

[13] S. Kandula, D. Katabi, M. Jacob, and A. Berger. Botz-4-Sale: Surviving Organized DDoS Attacks That Mimic Flash Crowds. In *Proc. NSDI*, May 2005.

[14] A. D. Keromytis, V. Misra, and D. Rubenstein. SOS: Secure overlay services. In *Proc. ACM SIGCOMM*, pages 61–72, Aug. 2002.

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

[16] D. Mazières, M. Kaminsky, M. F. Kaashoek, and E. Witchel. Separating key management from file system security. In *SOSP*, pages 124–139, Dec. 1999.

[17] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press Series on Discrete Mathematics and its Applications. CRC Press, 1997.

[18] P. L. Montgomery. Modular multiplication without trial division. *Math. Computation*, (44):519–521, 1985.

[19] K. Park and H. Lee. On the effectiveness of route-based packet filtering for distributed DoS attack prevention in power-law Internets. In *Proc. ACM SIGCOMM*, Aug. 2001.

[20] S. Savage, D. Wetherall, A. Karlin, and T. Anderson. Network support for IP traceback. *IEEE/ACM Transactions on Networking*, 9(3), June 2001.

[21] A. C. Snoeren, C. Partridge, et al. Single-packet IP traceback. *IEEE/ACM Transactions on Networking*, 10(6), Dec. 2002.

[22] L. von Ahn, M. Blum, N. Hopper, and J. Langford. CAPTCHA: Using hard AI problems for security. In *Advances in Cryptology – EuroCrypt*, 2003.

[23] M. Vutukuru, N. Feamster, M. Walfish, H. Balakrishnan, and S. Shenker. Revisiting Internet address: Back to the future! Technical Report 025, MIT Computer Science and Artificial Intelligence Laboratory, Apr. 2006.

[24] D. Wendlandt, D. G. Andersen, and A. Perrig. Fastpass: Providing first-packet delivery. Technical Report 005, CMU CyLab, Mar. 2006.

[25] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar. An integrated experimental environment for distributed systems and networks. In *Proc. USENIX OSDI*, pages 255–270, Dec. 2002.

[26] A. Yaar, A. Perrig, and D. Song. Pi: A path identification mechanism to defend against DDoS attacks. In *Proc. IEEE Symposium on Security and Privacy*, 2003.

[27] A. Yaar, A. Perrig, and D. Song. SIFF: A stateless Internet flow filter to mitigate DDoS flooding attacks. In *Proc. IEEE Symposium on Security and Privacy*, May 2004.

[28] X. Yang, D. Wetherall, and T. Anderson. A DoS-limiting network architecture. In *Proc. ACM SIGCOMM*, Aug. 2005.