FLoc: Dependable Link Access for Legitimate Traffic in Flooding Attacks

Soo Bum Lee, Virgil D. Gligor

November 23, 2011

CMU-CyLab-11-019

CyLab
Carnegie Mellon University
Pittsburgh, PA 15213

FLoc : Dependable Link Access for Legitimate Traffic in Flooding Attacks

Soo Bum Lee Virgil D. Gligor
CyLab, Carnegie Mellon University
Email: {soobum, gligor}@cmu.edu

Abstract— **Malware-contaminated hosts organized as a "bot network" can target and flood network links (e.g., routers). Yet, none of the countermeasures to link flooding proposed to date have provided *dependable* link access (i.e., bandwidth guarantees) for legitimate traffic during such attacks. In this paper, we present a router subsystem called FLoc (Flow Localization) that confines attack effects and provides *differential bandwidth guarantees* at a congested link: (1) packet flows of uncontaminated domains (i.e., Autonomous Systems) receive better bandwidth guarantees than packet flows of contaminated ones; and (2) legitimate flows of contaminated domains are guaranteed substantially higher bandwidth than attack flows. FLoc employs new preferential packet-drop and traffic-aggregation policies that limit "collateral damage" and protect legitimate flows from a wide variety of flooding attacks. We present FLoc's analytical model for dependable link access, a router design based on it, and illustrate FLoc's effectiveness using simulations of different flooding strategies and comparisons with other flooding defense schemes. Internet-scale simulation results corroborate FLoc's effectiveness in the face of large-scale attacks in the real Internet.**

I. Introduction

Recent flooding attacks in the Internet have typically involved "bot networks" that target application-layer services. Naturally, much work has been devoted to providing continued service access during such attacks. However, a bot network can also target and flood network links (e.g., routers), not just specific application services, and yet, none of the countermeasures to these attacks proposed to date have provided *dependable* link access (i.e., bandwidth guarantees) for legitimate traffic.

In handling link-flooding attacks, "capability" schemes [1]–[3] have aimed to provide identifier authenticity for individual flows via router-generated unforgeable identifiers (i.e., capabilities) created during the connection setup phase. These schemes effectively prevent identifier (IP address) spoofing and filter unauthorized traffic by making use of strong source-authorization rules applied at a network edge (e.g., by IDSs, Firewalls). Flow-identifier authenticity, though effective in preventing address-spoofing attacks, is insufficient to counter link-flooding attacks, since a large "bot network" could acquire capabilities in a legitimate manner and then flood a targeted link. Such an attack would grow in strength at least linearly with the number of bots, and would have global side effects: legitimate flows could be denied access through the targeted link. Thus, any per-flow defense scheme, even though it may limit the bandwidth of individual aggressive (i.e., attack) flows precisely [4], is ineffective for countering large-scale attacks comprising multiple flows launched from multiple, bot-contaminated hosts – a situation in which an aggregate-based defense is needed. However, previous aggregate-based defenses [5]–[7] do not identify attack flow-aggregates accurately, nor do they limit "collateral damage" within those flow-aggregates; i.e., they may deny link access to legitimate packet flows within attack aggregates. In principle, integrating salient features of both types of flooding-defense mechanisms can produce practical countermeasures.

In this paper, we present a router-based subsystem called FLoc (Flow Localization) that confines collateral damage of link-flooding attacks within specified packet-flow "locales," namely flows of single domains (Autonomous Systems, ASs) or specific sets of domains, and provides (1) link-access (i.e., bandwidth) guarantees on a per domain basis, and (2) fair bandwidth allocation for individual flows within a domain.

FLoc distinguishes between the flows (and flow aggregates) of bot-contaminated and uncontaminated domains. This distinction is possible for two reasons. First, the distribution of host contamination with "malware" in the Internet is highly non-uniform [8]–[12]: domains that employ sufficiently strong security mechanisms and policies for individual hosts are less likely to be contaminated with malware (including attack "bots") than others. Analysis on the active bot distribution in the current Internet evidences this non-uniformity; e.g., in Composite Blocking List [13] that holds the list of IP addresses of spam-bots, 95 % of the IP addresses belong to 1.7 % of active ASs. Second, legitimate (i.e., non-attack) flows originating from uncontaminated domains have more homogeneous congestion characteristics, such as mean packet-drop rates, than flows of contaminated domains. This enables FLoc to identify attack flows of contaminated domains and confine their effects to those domains. It also enables FLoc to provide *differential bandwidth guarantees* at a congested link, in two ways: (1) uncontaminated (or lightly contaminated) domains receive better bandwidth guarantees than highly contaminated domains; and (2) legitimate flows of contaminated domains are guaranteed substantially more bandwidth than attack flows. FLoc provides differential bandwidth guarantees by two complementary rate-control policies, namely, an intra-domain preferential drop policy and inter-domain, path-identifier aggregation policy. Both are discussed in Sections IV-B and IV-C, in detail. FLoc is designed to scale to operate at high speed backbone routers and in the presence of millions of attack flows.

FLoc's effectiveness is evaluated both by simulating differ-

ent attack scenarios (e.g., Constant Bit Rate (CBR), Shrew [14], and covert [15] attacks) and by comparing the FLoc results with those of other approaches (e.g., Pushback [5] and RED-PD [16]). We first evaluate the precise FLoc's functionalities using ns2 simulations in Section VI and then evaluate its effectiveness in defending against realistic large-scale attacks using Internet-scale simulations in Section VII.

II. Related Work

Recent approaches to network-layer defenses against flooding attacks provide authenticated identifiers, namely capabilities, to packet flows as a means of making these flows accountable. Network-layer capabilities, which were initially proposed in [1] and extended in [2], [3], help prevent address spoofing and filter out unwanted traffic at packet destinations. However, in the absence of appropriate flow control at a congested router, network links remain vulnerable to flooding attacks launched by "bot networks" using valid capabilities.

Most flooding-attack defenses first identify the flows (or flow aggregates) causing link congestion and then limit their bandwidth. Traditional per-flow defense mechanisms [16]–[19] can be used in a capability-based scheme to provide legitimate flows with different types of fair bandwidth sharing. However, fair bandwidth sharing does not necessarily imply, nor is it intended to have, the ability to distinguish between legitimate and attack flows, which is a necessary requirement for achieving FLoc's goals. For example, mechanisms that use a single packet-drop rate [16], or router-queue occupancy [18], [19] cannot make this distinction. Using a single packet-drop rate will not work because flows, legitimate or not, can have different RTT delays and hence *different* drop rates (viz., Section IV-A, Eq. (IV.1), where the packet drop rate is $1/(n_i T_{S_i})$) despite fair bandwidth allocation. Using router-queue occupancy will not work either, because both individual flows of attack aggregates and legitimate flows may have the *same* router-queue occupancy for some time periods.

Neither the ability to distinguish effectively between legitimate and attack packet flows, which is necessary to identify a broad range of flooding attacks [6], nor fair bandwidth allocation is sufficient to confine the effects of "covert" adversary attacks (viz., Sections IV-B.3 and VI-D for a specific example). That is, an adversary can coordinate a large number of "bots" in one contaminated domain, or more, and send only legitimate-looking (e.g., TCP-conformant, low-rate) flows through a targeted link, concurrently. This can easily deplete the bandwidth allocated (fairly) to flows of uncontaminated domains at that link [4], [15]. Hence, additional mechanisms are necessary to confine the harmful effects of such attacks.

Aggregate-based countermeasures [5], [7], [20] mitigate the downside of per-flow defense, yet their effectiveness is highly dependent upon how flow aggregates are defined (e.g., in terms of "locales") and how bandwidth is allocated to these aggregates. Previous approaches, such as Pushback, which install filters at remote routers, do not identify attack flow aggregates precisely since their aggregates account only for the *local* flow rates of incoming links to a router. Also, they lack effective mechanisms to limit "collateral damage" within attack aggregates and do not provide incentives to domains to perform flow filtering. Furthermore, installing filters at remote routers can be susceptible to timed attacks, whereby a "bot network" changes attack strength (e.g., on-off attacks) or location (e.g., rolling attacks) in a coordinated manner to avoid detection [6].

CDF-PSP [7] isolates the bandwidth of "high priority" flow aggregates, which conform to historical traffic data, from that of non-conformant "low-priority" traffic, and limits collateral damage by allocating bandwidth proportionally to all high priority traffic first. However, CDF-PSP does not provide bandwidth guarantees. For example, some legitimate flows that exhibit uncharacteristically high rates over a historically low-rate path receive low bandwidth allocations and, conversely, attack flows over a historically high-bandwidth path receive high bandwidth allocations. Furthermore, bandwidth isolation assumes static routing and loses effectiveness whenever dynamic traffic re-routing is caused by link congestion or failure.

III. Design Overview

A. Background: Domain and Flow Identifiers

A router-based defense mechanism that controls (1) the bandwidth allocation to individual domains and (2) the service rate of attack flows within a domain needs unique domain and flow identifiers that cannot be forged or replayed by a corrupt host or "bot" that originates a packet flow. Several mechanisms for flow and domain accountability have been proposed in the past [21], which can be used within FLoc. Currently, in FLoc, a router identifies a packet's domain of origin uniquely by a path of domain (AS) numbers starting from that domain and terminating at the router's domain. This path identifier differs from the original notion of a packet's path identifier [22], where router markings are added to the packet header by all routers on the packet's forwarding path. In FLoc, a path identifier is written to a packet by the BGP speaker of the packet's domain of origin as the domain path to the destination is available in the BGP speaker's routing table. Hence, this path marking scheme can be adopted by individual domains independently and incrementally. When a packet is forwarded by a sequence of domains $AS_i, AS_{i-1}, \ldots, AS_1$, the domain-path identifier of the packet is denoted by $S_i = \{AS_i, AS_{i-1}, \ldots, AS_1\}$ as shown in Fig. 1. Henceforth, we refer to a domain-path identifier simply as the "path identifier." The security requirements (e.g., unforgeability, non-replay) of the path identification mechanism are discussed in [23].

Currently, FLoc uses network-layer capabilities [1]–[3] to authenticate flow identifiers. These capabilities are constructed as follows. During the connection establishment phase of a flow, a router R_j generates a capability for that connection by hashing the packet's source and destination addresses (IP_S, IP_D), and path identifier (S_i) with the router's secret (K_{R_j}); i.e., Hash(IP_S, IP_D, S_i, K_{R_j}). The router writes the capability in the client's connection request packet (e.g., TCP-SYN) to the server. Thus, a router issues an authenticated identifier for a flow that can be verified only by the router

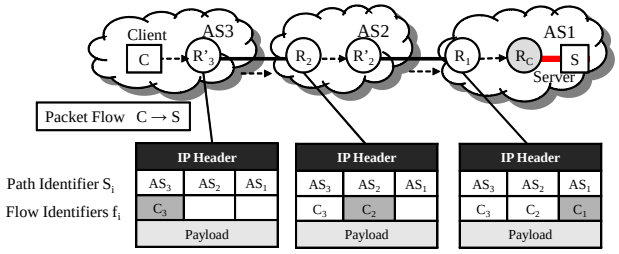


Fig. 1: A domain path identifier S_i and a flow identifier (f_i) for a packet sent from client C to server S.
Legend: S_i = {AS3, AS2, AS1 } and $f_i = C_j$ at R_j are implemented in a shim layer between the IP and Transport layers.

itself. This capability is returned to the client along with the server's acknowledgement (e.g., TCP-SYN/ACK). In Fig. 1, the flow identifier f_i at R_j is the capability C_j since C_j can only be authenticated by the R_j. These capabilities accompany all packet flows along this path and thus the flow authenticity is guaranteed at every router.

FLoc uses the path identifier as a domain identifier and the capability as a flow identifier. However, it does not require universal deployment since these identifiers are located outside the IP headers and only FLoc enabled routers interpret them.

B. Bandwidth Guarantees in Normal Mode

In normal mode of router operation, namely when a flooding attack is not in progress, a router assigns equal link bandwidth to all outstanding path identifiers (i.e., domains).[1] Whenever congestion is detected, the service rate for each path identifier is set by a separate token-bucket mechanism [24].[2] In principle, implementing separate token buckets for individual path identifiers could provide bandwidth guarantees to those identifiers. In practice, however, the effective bandwidth received by the aggregate flows of a specific identifier becomes highly dependent on the token-bucket parameters (i.e., on the token generation period, which determines the size of the transmission token generated at a time, and the bucket size) which are determined by the number of flows and the average RTT of the path identifier. The determination of the token bucket parameters per path identifier and that of the RTT in practice are presented in Sections IV-A and V below.

Our design of a token-bucket mechanism is based on the model of persistent (long) TCP flows (e.g., FTP flows) as these flows provide the reference transmission pattern of "legitimate" flows, based on which attack flows can be distinguished (viz., Section IV-B). We argue that guaranteeing the bandwidth of persistent TCP flows would guarantee that of short TCP flows as well, since short TCP flows are affected by the TCP congestion control mechanism to a lesser extent (e.g., they may finish their transmission even without experiencing packet drops or at least are less likely to experience consecutive packet drops that cause significant throughput loss [14], [23]). Additionally, our token-bucket mechanism offers features of active queue management (AQM) schemes for a path-identifier's TCP flows; e.g., early congestion notification, TCP flow de-synchronization, and fair bandwidth allocation among these flows (viz., Section V).

[1]For different domains having different numbers of sources, proportional rather than equal bandwidth allocation can be supported by FLoc (viz., per-domain token bucket mechanism in Section IV-A) provided that the number of domains with a large number of legitimate sources are known (e.g., via ISP service agreement).

[2]The basic idea of a token bucket mechanism is that link-access tokens are generated at a constant rate and are buffered in a bucket that has a limited capacity. The token generation rate determines the guaranteed bandwidth and the bucket size specifies the maximum tolerable burst size.

C. Bandwidth Guarantees in Attack Mode

While the normal-mode, per-domain bandwidth allocation helps localize the effects of link congestion within the domains where flows originate, such localization is insufficient to counter deliberate attacks for at least two reasons. First, an attack coordinated by multiple domains reduces the bandwidth available to packet flows of legitimate domains at a targeted link. Second, the bandwidth allocated to packet flows originating from legitimate clients of highly contaminated domains is severely reduced. To handle these harmful side effects, we identify attack flows and restrict their bandwidth.

Attack-Flow Identification. We identify attack flows (which keep consuming more bandwidth than the allocated amount) with two related mechanisms. First, we identify domains that originate attack flows. Our token-bucket mechanism allocates router bandwidth to individual domains and makes the necessary packet drops for a desired bandwidth utilization. As our token-bucket mechanism provides the reference drop rate for the flow aggregate of a legitimate domain, an excessive packet-drop rate signals the presence of attack or non-conformant flows within that domain.

Second, we identify attack flows independent of the flooding-attack strategies. To do this, we use a flow's average packet-drop interval, defined as the "mean time to drop (MTD)" of the flow. In Section IV-B, we show that in FLoc, the MTDs of attack flows are distinct from those of legitimate flows, and represent the strength of attack precisely, no matter what attack strategies are employed by a "bot network". In Section V we show that the MTD-based attack identification can be efficiently implemented in a router since only the states of dropped packets need to be recorded and the lifetime of those states can be precisely bounded.

Differential Bandwidth Guarantees. Once identified, attack flows are penalized by two rate-control policies, namely preferential drop and path aggregation. These policies are designed to achieve two types of *differential bandwidth guarantees*: (1) packet flows of domains that are uncontaminated or lightly contaminated by "bot networks" receive better bandwidth guarantees than flows of contaminated domains; and equally importantly, (2) within a contaminated domain, legitimate packet flows experience fewer packet drops, hence better throughput, than attack flows.

First, within a contaminated domain, the packets of the attack flows are preferentially dropped with the aim to upper

bound their throughput by their fair bandwidth allocation. Accordingly, attack flows that do not respond to per-domain, fair-bandwidth controls are penalized by increasingly more packet drops. Any misidentification of legitimate flows as attack would never result in service denial once the sources of misidentified flows respond to the packet drops by decreasing their send rate. Hence, "collateral damage" (i.e., denied service) to legitimate flows within attack domains is avoided.

Second, the path identifiers of highly contaminated domains are aggregated into a single path identifier. Since router bandwidth is assigned fairly to path identifiers, aggregation, in effect, reassigns the bandwidth of highly contaminated domains to legitimate (i.e., non-contaminated or lightly contaminated) ones. Hence, path aggregation helps provide higher bandwidth guarantees to legitimate (i.e., non-aggregated) paths by reducing the bandwidth assigned to aggregated paths even when attack sources (e.g., bots nets) are widely dispersed across multiple domains. Aggregation is triggered whenever the number of outstanding path identifiers exceeds a limit set so that all active path identifiers receive a minimum guaranteed bandwidth at a congested router.

D. Scalable Design

In high-speed Internet backbone routers, per-flow treatment of packets could become infeasible. This is not only because the memory size is not sufficient, but because frequent memory access is prohibitive. If filters can be made very size-efficient, a SRAM-based design may work. However, in the presence of millions of attack flows, individual flows' states (e.g., bandwidth and packet-drop rate) cannot be maintained at SRAM since traditional filters require at least as many filter entries as the number of flows. RLDRAM (Reduced Latency DRAM) could be a viable solution, but less memory accesses is still necessary in order not to disturb the basic functionalities of routers (routers manage RIB (Routing Information Base) and FIB (Forwarding Information Base) which involve control-plane message exchanges and substantial computation, and based on those table, they perform packet forwarding).

When flooding attacks are not in progress, even if flows' bandwidth is controlled by router's packet-drops (i.e., the bottleneck is not the service rate of an end-server, but the link capacity on its path to the service), the packet service rate at a specific link is generally much higher than the packet drop rate as shown in the Fig. 2. Provided that we can precisely count the number of flows, the packet drop ratio of a flow-aggregate S_i (γ_{S_i}) can be expressed in terms of the maximum TCP congestion window size of individual flows: $\gamma_{S_i} = \frac{8}{3W_{S_i}(W_{S_i}+2)}$ [3] (viz., Section V-B.1). This suggests that we can identify attack aggregates (i.e., flow-aggregates that contain high-rate, attack flows) by observing the packet-drop ratio of flow-aggregates and hence can handle that traffic efficiently.

In handling attack flows from their packet-drop rates, we argue that it is sufficient to work with full-sized packets (i.e., 1.5KB) since the distribution of packet sizes is *bimodal*: either 40B SYN/ACK or 1.5KB full-sized packet according to the previous study [25]. According to Fig. 3, the size of some full-sized packets is 1.3KB probably because those packets are injected into the Internet through VPN tunneling. Yet, those flows would exhibit the same congestion control characteristics as the full-size packets.

[3]We note that a flow-aggregate is a set of flows that have the same congestion characteristics (i.e., RTTs)

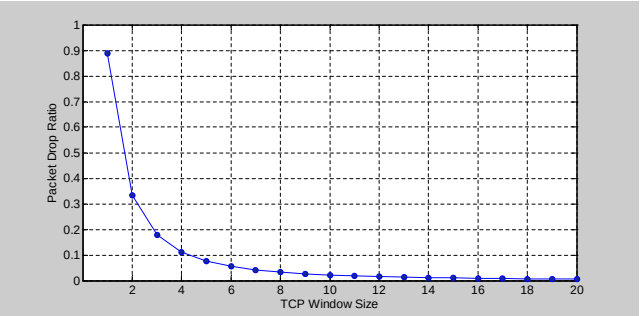


Fig. 2: TCP Congestion Window Size vs. Packet Drop Ratio

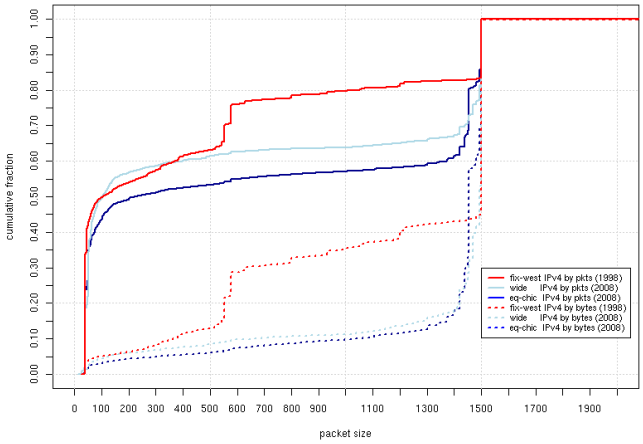


Fig. 3: Packet Size Distribution [25]

IV. Modeling Dependable Link Access

In this section we present an analytical model for the bandwidth guarantees for legitimate packet flows, which define FLoc's notion of dependable link access in flooding attacks.

A. Token-Bucket Model Revisited

We first review the standard TCP congestion-control mechanism and the aggregate traffic characteristics of multiple TCP flows at a congested router. Then, we define the parameters of a token bucket mechanism for a (per-domain) path identifier that provide guaranteed bandwidth to legitimate flows in normal (non-attack) mode of router operation.

TCP Flow Model. Let W_{f_i} be the congestion window size at the source of TCP flow f_i. The TCP congestion-avoidance protocol increases the W_{f_i} of the TCP source on every acknowledgement from the destination. That is, the source increments its W_{f_i} by one in every round trip time (RTT) until it experiences a packet drop. After the packet drop, the

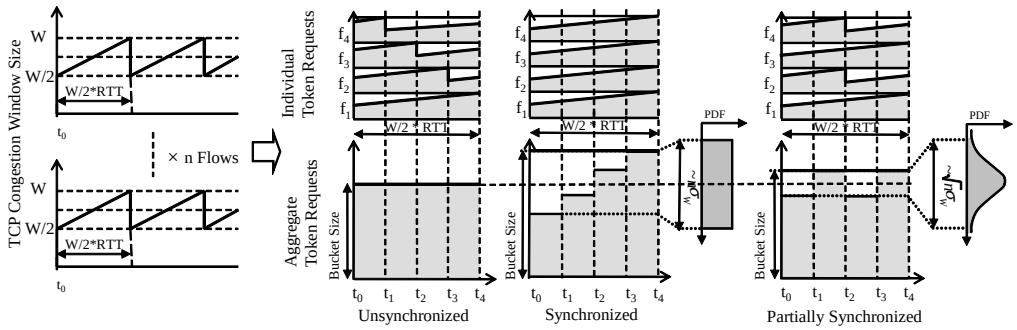


Fig. 4: TCP window variations and their aggregate effect at a congested router for unsynchronized, synchronized, and partially synchronized flows.
Legend: Upper graphs: gray areas denote individual packet-flow rates. Lower graphs: grey areas illustrate three different distributions of the same aggregate-flow rate at a congested router.

source halves its W_{f_i}, and repeats the congestion-avoidance protocol. Fig. 4 illustrates TCP window size variation for n flows. Window size W_{f_i} is typically modeled as a random variable whose distribution is uniform on $[\frac{W}{2}, W]$, where W is the peak congestion window size of a flow f_i's source, as discussed in [26]. In this ideal model, a source experiences a packet drop in every $W/2 \times RTT$ seconds; i.e., the "mean time to drop" for flow f_i (denoted as MTD(f_i)) is MTD(f_i) = $W/2 \times RTT$. Hence, for n TCP flows through a congested router, n packet drops occur during $W/2 \times RTT$ seconds if all flows share an equal bandwidth and have the same RTT.

Guaranteed Bandwidth for Legitimate Flows. To control the packet drops of TCP flows on a per-domain basis, we allocate a separate token bucket to each path identifier and customize the token generation period and bucket size for a given bandwidth guarantee. We consider three cases of TCP flow synchronization that affect traffic burstiness and hence the token bucket parameters; i.e., unsynchronized, synchronized, and partially synchronized traffic.

Let us consider the ideal (best) case, namely when the flows are completely unsynchronized; i.e., the peak window size of each source is uniformly distributed in time, as shown in the upper graph of Fig. 4 in the "Unsynchronized" case. Uniform distribution of individual sources' peak window sizes makes their aggregate token-request rate at the congested router uniform over time, leading to full consumption of tokens; i.e., to full bandwidth utilization. This is illustrated in the lower graph of Fig. 4 in the "Unsynchronized" case. Let C_{S_i} be the bandwidth guaranteed to a n_i-flow path identifier S_i, RTT_i be the average RTT of the flows in S_i, and W_i be the maximum window size of the flows in S_i. Let N_{S_i} be the bucket size measured in tokens, and T_{S_i} be the token generation period. To make a packet drop uniformly every $T_{S_i} = \frac{W_i/2 \times RTT_i}{n_i} = \frac{MTD(f_i)}{n_i}$ seconds, the bucket size is set to $N_{S_i} = C_{S_i} \times \frac{W_i/2 \times RTT_i}{n_i}$ tokens, and the bucket is filled within T_{S_i} seconds. That is, N_{S_i} tokens are generated at the start of each period, and the unused tokens of the previous period are removed. In this case, aggregate flows of a (per-domain) path identifier would run out of tokens only if their token requests exceed N_{S_i} in a period T_{S_i}. (Note that bursty requests are allowed within a period T_{S_i}.)

Now we can relate the window size of a flow's source with that flow's assigned bandwidth and RTT_i. The uniform distribution of the window size on the interval $[\frac{W_i}{2}, W_i]$ implies that the average window size is $\frac{3}{4}W_i$. Since the bandwidth C_{S_i} guaranteed to path identifier S_i is allocated fairly (i.e., divided equally) among the n_i flows of S_i, each flow's bandwidth is C_{S_i}/n_i. Thus the relationship between a flow's bandwidth and its window size, $bw_{f_i} = \frac{W_{f_i}}{RTT}$, implies that $C_{S_i} = \frac{3n_iW_i}{4RTT_i}$. Consequently, the token bucket parameters (i.e., token generation period and the corresponding bucket size) can be expressed as follows.

$$T_{S_i} = \frac{W_i/2 \times RTT_i}{n_i} = \frac{MTD(f_i)}{n_i} = \frac{2}{3}\frac{C_{S_i}}{n_i^2}RTT_i^2 \qquad \text{(IV.1)}$$

$$N_{S_i} = \frac{2}{3}\frac{C_{S_i}^2}{n_i^2}RTT_i^2 \qquad \text{(IV.2)}$$

In contrast, in the worst case, namely when all flows are completely synchronized, they would experience packet drops in the same period T_{S_i}; viz., the interval denoted by $[t_3, t_4]$ in upper graph of Fig. 4 in the "Synchronized" case. In this case, the peak token-request rate of aggregate flows at the congested router, that occurs when the window size of every flow reaches W_i, would be twice its minimum that occurs when every flow halves its window size after experiencing a packet drop. Only 3/4 of the generated tokens can be consumed in this case; e.g., the shaded area of the lower graph of Fig. 4 in the "Synchronized" case represents only 3/4 of the token generated in the interval $[t_0, t_4]$. Hence, the ideal token-bucket size needs to be increased by 1/3 to accommodate the peak flows in the worst case; i.e., it should be increased to $\frac{4}{3}N_{S_i}$.

In normal (non-attack) mode of operation, flow sources are independent (e.g., flows are not synchronized by adversaries) and they share a token-bucket's bandwidth fairly (i.e., equally). This implies that the window size W_{f_i} of a flow f_i's source can be modeled as an i.i.d. random variable whose distribution is uniform on $[\frac{W_i}{2}, W_i]$ (i.e., $\mu_{W_{f_i}} = \frac{3}{4}W_i$ and $\sigma_{W_{f_i}} = \frac{1}{\sqrt{12}}\frac{Wi}{2}$). Thus, the token-request rate of a n_i TCP-flow aggregate (i.e.,

$\sum_{i=1}^{n_i} W_{f_i}$) has a gaussian distribution with mean $n_i\mu_{W_{f_i}}$ and standard deviation $\sqrt{n_i}\sigma_{W_{f_i}}$ (by the Central Limit Theorem). Intuitively, this can be interpreted to mean that (1) only $\sqrt{n_i}$ flows are synchronized, and (2) $\sqrt{n_i}$ subaggregate flows exist on a path. This is illustrated in the "Partially Synchronized" case of Fig. 4. As the lower graph of this figure illustrates, partially synchronized flows do not consume all tokens of their bucket either, yet they utilize the tokens (i.e., the allocated bandwidth) better than the completely synchronized flows. This is the case because the request rate of partially synchronized flows fluctuates less than that of completely synchronized flows. This suggests that the token bucket size should be increased as a function of the flow-synchronization degree, which determines the standard deviation of a path identifier's token-request rate.[4] Thus, expected traffic bursts are tolerated.

We define the new, increased size of the token bucket as $N^{\mathcal{I}}_{S_i} = (1 + \frac{\epsilon\sigma_{S_i}}{\mu_{S_i}})N_{S_i}$, where μ_{S_i} and σ_{S_i} are the mean and standard deviation of the token request rate of S_i, and ϵ is the increase factor. For i.i.d flows that we consider, $\sigma_{S_i} = \sqrt{n_i}\sigma_{W_{f_i}} = \frac{\sqrt{n_i}}{\sqrt{12}}\frac{W_i}{2}$. We set the increase factor to $\epsilon = \sqrt{12}$ as this would bound the peak token requests with probability 99.97%; i.e., $\Pr(\sum_{i=1}^{n_i} W_{f_i} \leq \mu_{S_i} + \sqrt{12}\sigma_{S_i}) = 0.9997$. Accordingly, the new, increased token-bucket size for i.i.d. flows becomes

$$N^{\mathcal{I}}_{S_i} = (1 + \frac{\epsilon\sqrt{n_i}\sigma_{W_{f_i}}}{n_i\mu_{W_{f_i}}})N_{S_i} = \frac{2}{3}(1 + \frac{2}{3\sqrt{n_i}})\frac{C^2_{S_i}}{n_i^2}RTT_i^2. \quad \text{(IV.3)}$$

In summary, to guarantee a certain bandwidth, C_{S_i}, to a (domain) path identifier S_i in normal mode of router operation (i.e., when all flows, including bursty ones, are legitimate), we count the number of active flows, n_i, measure the average RTT_i for that path identifier (viz., Section V), and then set the token bucket parameters, namely the token generation period, T_{S_i}, and corresponding token bucket size, $N^{\mathcal{I}}_{S_i}$, as specified by Eqs. (IV.1) and (IV.3), respectively.

B. Attack-Flow Identification and Confinement

In this subsection, we describe how to identify attack paths and attack flows within those paths by the packet drop intervals of flows, and how to limit the bandwidth of those attack flows. Let $\mathcal{F}_{S_i}$ be a set of flows carrying path identifier S_i, and $\text{MTD}(f_i)$ be the *"mean time to packet drop"* of flow $f_i \in \mathcal{F}_{S_i}$. Then, $\text{MTD}(f_i)$ can be written as $\text{MTD}(f_i) = \frac{W_i}{2}RTT_i = n_iT_{S_i}$ in normal mode of operation (as shown in the previous section). We define $n_iT_{S_i}$ as the *reference* MTD of a flow carrying (domain) path identifier S_i.

1) Attack (Domain) Paths: For the paths that deliver attack flows (which we call "attack paths"), the MTD of aggregate flows is lower than the token generation period while the request rate of S_i (λ_{S_i}) is higher than the allocated bandwidth added by the reference drop rate of S_i; i.e., $\text{MTD}(\mathcal{F}_{S_i}) < T_{S_i}$ and $\lambda_{S_i} > C_{S_i} + 1/T_{S_i}$. This is because the MTDs of legitimate flows are less than the reference MTD (due to the decrease of available bandwidth), yet they are greater than those of attack flows; i.e., the MTDs of all flows are less than the reference MTD. Hence, attack paths can be identified by estimating the *mean packet drop rate* of path-identifiers, which is the inverse of MTD. If allowed, attack paths would over-utilize their bandwidth by exhausting the extra tokens made available by the new, increased bucket size (defined above). To avoid such bandwidth over-utilization, the fixed bucket size (N_{S_i}) is applied to path identifiers containing attack flows instead of the new, increased ones ($N^{\mathcal{I}}_{S_i}$). This strictly limits the bandwidth available to attack paths.

2) Attack Flows: Though effective in localizing attack effects, bandwidth control on a (per-domain) path identifier basis does not prevent all "collateral damage;" i.e., does not protect legitimate flows that happen to be within an attack path. To protect these flows, we introduce an attack-flow identification and control mechanism.

MTD Measurement. Let $D^{f_i}_{S_i}(t_j)$ be the number of packet drops of flow f_i in interval $(t_{j-1}, t_j]$. If we set $t_j - t_{j-1} = T_{S_i}$, for some $k \geq n_i$, $\text{MTD}(f_i)$ under our token-based packet admission policy is measured as:

$$\text{MTD}(f_i) = \frac{k \cdot T_{S_i}}{\sum_{j=1}^{k} D^{f_i}_{S_i}(t_j)}. \quad \text{(IV.4)}$$

Since $\text{MTD}(f_i)$ is inversely proportional to the packet-drop rate of f_i (which is proportional to its send rate), the MTD of an attack flow is *always lower* than that of a legitimate flow. This definition of MTD could identify attack flows showing vastly different drop patterns (i.e., employing different attack strategies), since it is measured over sufficient periods kT_{S_i} (viz., Eq. (IV.4)) for estimating flows' send rates while fine-grained bandwidth control (in a period of T_{S_i}) is performed.

For the attack flows identified by their MTD, the congested router applies the following packet admission policy to limit the bandwidths of those flows.

$$\Pr(f_i \text{ is serviced}) = I^T_{S_i}(f_i) \cdot Min\{1, \frac{\text{MTD}(f_i)}{n_iT_{S_i}}\} \quad \text{(IV.5)}$$

where $I^T_{S_i}(f_i)$ equals 1, if a token is available to flow f_i, and 0, otherwise.

This packet admission policy preferentially drops the packets belonging to attack flows in proportion to their *send* rates, and more aggressively penalizes the flows whose MTDs keep decreasing (i.e., flows that do not respond to packet drops by decreasing their send rate). When an attack flow which sends traffic with rate $\alpha\frac{C_{S_i}}{n_i}$ ($\alpha > 1$) experiences d preferential packet drops, its effective bandwidth at the congested link is $\alpha\frac{C_{S_i}}{n_i} \cdot \Pr(f_i \text{ is serviced}) \leq \alpha^{1-d}\frac{C_{S_i}}{n_i}$, since $\text{MTD}(f_i)$ decreases proportionally to $\frac{1}{\alpha}$ on each preferential drop. Hence, whenever all flows actively compete for the bandwidth allocated to a path identifier (i.e., no spontaneously under-subscribing flow exists), flow f_i cannot use more bandwidth than its fair amount within that path identifier allocation.

[4]Note that, in general, for a given flow synchronization model (i.e., a window-size distribution), the number of additional tokens that need to be provided is proportional to the standard deviation of the aggregate token request rate.

Note that the above packet admission policy would never deny service to the misidentified (attack) flows since service to those flows would resume once the sources of those flows respond to packet drops by decreasing their *send* rates; i.e., their MTD(f_i) would keep increasing and so would Pr(f_i is serviced), as shown in Eq. (IV.5).

3) Confinement of Covert Attacks: Recall that, in a "covert" attack, each "bot" may coordinate a large number of legitimate-looking (low-rate) flows to traverse a target link, concurrently. Covert attacks can be extremely potent. For example, each of N bots on one side of the targeted link can coordinate sending messages to and receiving messages from each of N "bots" on the other side of that link so as to cause $O(N^2)$ link flows [15]. Individually, these $O(N^2)$ flows appear to be perfectly legitimate and yet collectively they may deplete most of, if not all, that link's bandwidth.

FLoc counters the effects of covert attacks in two ways. First, FLoc limits the number of flows that a source can make with different IP destinations through the flooded link by constructing a flow's capability as follows.

Let $C_{f_{s,d}}$ be the capability for a flow $f_{s,d}$ between a source s and a destination d. $C_{f_{s,d}}$ consists of two parts, namely $C_{f_{s,d}} = C^0_{f_{s,d}} || C^1_{f_{s,d}}$. Here, $C^k_{f_{s,d}}$ for k $\in \{0,1\}$ is defined as:

$$\begin{aligned} C^0_{f_{s,d}} &= \text{Hash}(\text{IP}_s, \text{IP}_d, S_i, K^1_R) \\ C^1_{f_{s,d}} &= \text{Hash}(\text{IP}_s, \text{F}(\text{IP}_d), S_i, K^2_R) \end{aligned}$$

where IP_s and IP_d are the source and destination IP addresses, K^0_R and K^1_R are the router's secret keys, and F($\cdot$) is a function whose output is randomly uniform on [0, n_{max}-1].

$C^0_{f_{s,d}}$ provides identifier authenticity to flows [2], [3]. And, $C^1_{f_{s,d}}$ enables a router to perform two tasks: (1) restrict the number of capabilities (i.e., flows) per source to n_{max}, and (2) account for the total bandwidth requested using those capabilities concurrently. Thus sources with a high fanout of legitimate, low-rate, concurrent flows would be identified as sources of high-rate, covert attack flows within a router.

Second, FLoc confines such covert attacks to individual domains by avoiding aggregation of legitimate path identifiers that have widely different numbers of flows (viz., Section IV-C.2). A brief analysis of FLoc's handling of covert attacks is presented is Section VI-D.

C. Differential Bandwidth Guarantees

In this subsection, we present a mechanism that provides path identifiers with differential bandwidth guarantees based on a "conformance" measure for path identifiers. We define the "conformance" of a path identifier S_i starting from router R_i in a time interval $(t_{k-1}, \quad t_k]$ as the fraction of the legitimate flows in S_j. We denoted this "path-conformance" by $\mathcal{E}_{R_i}$ and express it as a moving average of $\mathcal{E}_{R_i}$ values. That is,

$$\mathcal{E}_{R_i}(t_k) = \beta(1 - \frac{n^a_i}{n_i}) + (1-\beta)\mathcal{E}_{R_i}(t_{k-1}) \qquad \text{(IV.6)}$$

where n_i and n^a_i are the number of active flows and attack flows forwarded by R_i; and β ($0 < \beta < 1$) is a constant smoothing factor (e.g, β = 0.2 in our simulations).

Based on this path-conformance measure, a router performs (1) attack-path aggregation for goodput maximization at the flooded link and (2) legitimate-path aggregation for fair bandwidth allocation to flows of different (legitimate) paths. For these aggregations, a congested router R_0 builds a traffic tree ($\mathcal{T}_{R_0}$) using the path identifiers carried in the "active" flows and decomposes it into two sub-trees, namely an attack tree ($\mathcal{T}^{\mathcal{A}}_{R_0}$) and a legitimate tree ($\mathcal{T}^{\mathcal{L}}_{R_0}$). $\mathcal{T}^{\mathcal{A}}_{R_0}$ consists of the path identifiers that have lower path conformance than a certain threshold ($\mathcal{E}_{th}$), and $\mathcal{T}^{\mathcal{L}}_{R_0}$ consists of the other ones in $\mathcal{T}_{R_0}$.

1) Attack-Path Aggregation: The goal of attack-path aggregation is to provide bandwidth guarantees to legitimate paths despite wide dispersion of attack "bots" over a large number of domains. This is achieved by aggregating the path identifiers of highly contaminated domains and hence limiting the number of bandwidth-guaranteed path identifiers. Path identifier aggregation starts from nearby domains (i.e., domains with longest postfix-matching path identifiers) to (1) localize attack effects within these domains and (2) avoid mixing flows having highly different RTT delays (as this would affect FLoc's precision in estimating token-bucket parameters). Whenever the number of path identifiers is bounded by $|\mathcal{S}|_{max}$,[5] the attack-path aggregation problem can be defined as the path-conformance maximization problem below.

Let $\mathcal{R}$ be the set of all nodes in $\mathcal{T}^{\mathcal{A}}_{R_0}$, and $\mathcal{R}_i$ be the set of leaf nodes of a subtree rooted at $R_i \in \mathcal{T}^{\mathcal{A}}_{R_0}$ (i.e., $\mathcal{T}^{\mathcal{A}}_{R_i}$). And, let $\mathcal{S}^{\mathcal{L}}$ and $\mathcal{S}^{\mathcal{A}}$ be the set of legitimate and attack path-identifiers respectively. Then, the path-conformance maximization problem is defined as:

$$\max O(\mathcal{T}^{\mathcal{A}}_{R_0}) = \sum_{R_i \in \mathcal{R}} \frac{1}{|\mathcal{R}_i|} \sum_{R_j \in \mathcal{R}_i} \mathcal{E}_{R_j} \qquad \text{(IV.7)}$$

$$\text{subject to } \sum_{R_i \in \mathcal{R}} I_{R_i} \le |\mathcal{S}|_{max} - |\mathcal{S}^{\mathcal{L}}| \text{ and } \bigsqcup_{R_i \in \mathcal{R}} \mathcal{R}_i = \mathcal{R}_0$$

where I_{R_i} is the indicator function which equals 1, if paths are aggregated at R_i, and 0, otherwise. For a non-aggregated path, I_{R_i} is 1 at the leaf node. Since $\sum_{\mathcal{S}_i \in \mathcal{S}} I_{\mathcal{R}_i}$ is the total number of attack path identifiers seen at node R_0, it should be bounded by $|\mathcal{S}|_{max} - |\mathcal{S}^{\mathcal{L}}|$ for the bandwidth guarantees on $\mathcal{S}^{\mathcal{L}}$.

In the above equation, aggregation at router R_i decreases the path-conformance by $\frac{|\mathcal{R}_i|-1}{|\mathcal{R}_i|} \sum_{R_j \in \mathcal{R}_i} \mathcal{E}_{R_j}$, which we define as the "aggregation cost" and denote by $C^{\mathcal{A}}(R_i)$. Hence, a set of nodes at which aggregating path-identifiers has the minimum (overall) aggregation cost and reduces at least $|\mathcal{S}^{\mathcal{A}}| - (|\mathcal{S}|_{max} - |\mathcal{S}^{\mathcal{L}}|)$ path-identifiers, would be a solution to the above problem.

Whenever necessary, attack-path aggregation is performed as summarized in **Algorithm 1** below. Let $\mathcal{O}$ be the solution set whose elements are the nodes at which paths are aggregated, and $\mathcal{C}$ be the candidate set whose elements could be

[5]Aggregation can be optimized with respect to $|\mathcal{S}|_{max}$, yet this may cause unnecessary path aggregation in the presence of under-subscribing or short-lived paths. Hence, we leave $|\mathcal{S}|_{max}$ as a configurable parameter determined by individual routers (e.g., the maximum number of concurrent paths observed statistically).

a solution and initially consist of all non-leaf nodes of $\mathcal{T}^{\mathcal{A}}_{R_0}$. Then, the algorithm works as follows. First, the node having the lowest aggregation cost in $\mathcal{C}$ is added to $\mathcal{O}$. Second, a node $R_i \in \mathcal{C}$ replaces the current solution set if its aggregation cost is less than the total aggregation cost of the nodes in $\mathcal{O}$. This procedure continues until the constraint on the number of path identifiers is satisfied. **Algorithm 1** is a "greedy" approximation algorithm. However, the second step ensures the minimum cost decrease in $\mathcal{C}$ at each iteration, and this helps bound the distance of its solution from the optimal by the product of $\mathcal{E}_{th}$ and the degree of the last added node [23].

Algorithm 1: Aggregation

1: $\mathcal{O} \leftarrow \text{argmin}_{R_i}\{C^{\mathcal{A}}(R_i)|R_i \in \mathcal{C}\}$
2: A node $R_i \in \mathcal{C}$ replaces $\mathcal{O}$ if it satisfies :
- $C^{\mathcal{A}}(R_i) < \sum_{R_j \in \mathcal{O}} C^{\mathcal{A}}(R_j)$
- $C^{\mathcal{A}}(R_i) > \max_{R_j \in \mathcal{O}} C^{\mathcal{A}}(R_j)$

3: goto Step 1 if $|\mathcal{S}^{\mathcal{A}}| > |\mathcal{S}|_{max} - |\mathcal{S}^{\mathcal{L}}|$.

2) Legitimate-Path Aggregation: The aggregation of legitimate paths is intended to achieve proportional bandwidth allocation to "legitimate" path identifiers that have different numbers of flows; i.e., fair bandwidth allocation to flows. In aggregation, a (congested) router assigns a new path-identifier to the aggregated paths and allocates its bandwidth in proportion to the number of aggregated paths. Consequently, the path conformance of the aggregated path would be the weighted average of individual paths' conformance measures, where the weighting factor is the number of domains' flows.

Let $C^{\mathcal{L}}(R_i)$ be the *net change* of path-conformance after the aggregation of paths at R_i. Then, $C^{\mathcal{L}}(R_i)$ is defined as:

$$C^{\mathcal{L}}(R_i) = \frac{1}{|\mathcal{R}_i|} \sum_{R_j \in \mathcal{R}_i} \mathcal{E}_{R_j} - \frac{\sum_{R_j \in \mathcal{R}_i} \mathcal{E}_{R_j} n_j}{\sum_{R_j \in \mathcal{R}_i} n_j}. \quad \text{(IV.8)}$$

A negative value of $C^{\mathcal{L}}(R_i)$ means that the aggregation of paths at R_i would increase the path-conformance, and, eventually, the goodput of the flooded link. Hence, aggregation would be performed at all nodes in $\mathcal{T}^{\mathcal{L}}_{R_0}$ whose cost is negative.[6]

However, if a covert-attack path (viz., Section IV-B.3 and [15]) is inadvertently aggregated with a (truly) legitimate path, a large number of legitimate-looking flows of the covert path may soak the bandwidth [6] of (truly) legitimate flows. To avoid this, FLoc does not aggregate legitimate paths whenever aggregation would increase bandwidth allocation to any path by more than a fraction of its current value; e.g., in our simulation this fraction is set to 50%.[7] Thus aggregation would never take place for path identifiers that have widely different numbers of flows.

[6]However, FLoc does not aggregate paths that have significant discrepancies in RTT delays as such aggregations would lead to false identification of attack flows.

[7]Typically, ISPs over-provision network link capacity to maintain link utilization under 40 – 50% in normal operating conditions, and under 75% in the presence of a single link failure. Hence, increasing the bandwidth of covert-attack paths by 50% or less will not exceed link capacity in practice.

V. Router Design

A. Token Bucket Management

In this section, we describe how our model of dependable link-access is implemented in a router. Specifically, we describe the packet admission (and drop) policies based on the token-bucket mechanism and the management of the router's buffer queue.

The token bucket parameters (i.e., token generation pattern and bucket size) for a path-identifier depend quadratically on the *actual* RTT_i of a path identifier S_i (viz., Eqs. (IV.1) – (IV.3)). Since the actual RTT_i can only be approximated, we estimate its value by (1) averaging the measured RTT of individual flows in a path, and (2) adjusting that average downward to avoid an over-estimate (e.g., we divide the average RTT of a path by 2 in the simulations reported in the next section). Note that an over-estimate of the actual RTT_i would inflate the token generation period and bucket size substantially, thereby causing bandwidth over-utilization and overflows of the router buffer queue. In contrast, an under-estimate would deflate the token-bucket parameters and potentially cause unnecessary packet drops for a path identifier. FLoc implements a control mechanism that adjusts the packet-drop rate and compensates for any unnecessary packet drops (discussed below).

$\mathbf{RTT_i}$ measurement. We measure the average RTT of a path S_i at a router by averaging the individual flows' RTTs for that path. A flow's RTT is measured as the time between a client's connection (i.e., capability) request (SYN) and its first data transmission (with ACK). The elapsed time from a capability issue by a router to a client to the client's first use of the issued capability at that router is a fairly accurate measurement of an individual flow's RTT.

Router-Queue Management. The management of a router's buffer queue establishes the service rate for path identifiers and, as such, it implements the bandwidth guarantees provided by the token-bucket mechanism for each path identifier. FLoc implements a router's (FIFO) queue whose size varies in the interval $[Q_{min}, Q_{max}]$. Q_{min} is a configurable parameter chosen to avoid both link under-utilization (which could be caused by short bursts of packets) and long queuing delays; viz., the RED queue. (We set Q_{min} to 20% of buffer size in the simulations of Section VI.) To determine Q_{max}, recall that the token requests of partially synchronized flows of a path S_i oscillate between $(\frac{3n_i}{4} - \frac{\sqrt{n_i}}{2})W_i$ and $(\frac{3n_i}{4} + \frac{\sqrt{n_i}}{2})W_i$. This requires at least $\sqrt{n_i}W_i$ packet buffer space for S_i to avoid link under-utilization. Hence, we set $Q_{max} = Q_{min} + \sum_{S_i \in \mathcal{S}} \sqrt{n_i}W_i$.

FLoc uses the current queue length, Q_{curr}, to manage the buffer queue in three modes of operation, namely (a) uncongested mode, where $Q_{curr} \leq Q_{min}$; (b) congested mode, where $Q_{min} < Q_{curr} \leq Q_{max}$; and flooding mode, where $Q_{curr} > Q_{max}$. The activation of the token-bucket mechanism

begins in the congested mode (b) with the initial parameters for a path identifier S_i set to T_{S_i} and $N^{\mathcal{T}}_{S_i}$, respectively (viz., Eqs. (IV.1) and (IV.3)).

Uncongested Mode. If $Q_{curr} \leq Q_{min}$, all packets are serviced regardless of token availability. The router's buffer queue tolerates temporary bursts of traffic until packet arrivals fill it. Link under-utilization, which may be caused by unnecessary packet drops, is avoided. However, attack-path flows may still appear in this mode and consume more buffers (and higher bandwidth) than legitimate flows until the router queue reaches Q_{min}. In this case, FLoc forces entry in congested mode as soon as $Q_{curr} > Q_{min} \times \min\{1, \frac{C_{S_i}}{\lambda_{S_i}}\}$, where λ_{S_i} is the request rate of S_i. This test leads to the activation of the token-bucket mechanism for attack path identifiers early, and causes them to experience packet drops before legitimate ones.

Congested Mode. If $Q_{min} < Q_{curr} \leq Q_{max}$, the token-bucket controls are activated for all path identifiers in the queue. However, since FLoc underestimates the token-buffer parameters (as discussed above), some path identifiers may experience unnecessary packet drops. To avoid penalizing legitimate path identifiers with unnecessary drops, FLoc implements a *random-drop* (i.e., neutral) policy in congested mode, instead of a targeted per-path drop policy, as required by the (under-estimated) token-buffer parameters. That is, if a packet does not get a token on its arrival, a queue threshold value, Q_{th}, is picked at random between Q_{min} and Q_{max}, and the packet is dropped only if $Q_{curr} > Q_{th}$.[8] The random drops end when the *uncongested mode* is re-entered, namely when $Q_{curr} \leq Q_{min}$.

Flooding Mode. If $Q_{curr} > Q_{max}$, then either traffic bursts or unresponsive and/or attack flows persist. In either case, FLoc applies the packet-admission (drop) policy defined by the token-bucket mechanism with the bucket size N_{S_i} instead of $N^{\mathcal{T}}_{S_i}$.

B. Efficient Attack-flow Handling

In this section, we present an efficient design of FLoc that scales to operate at high-speed backbone routers (e.g., OC-192). For this, routers first measure the packet drop rate and RTT of a path, by which they compute the number of flows competing the scarce bandwidth. If the number of distinct flows that have packet drops is less than the computed number of flows (i.e., the reference value), there certainly exist attack flows that either send packets at higher rates than or are not conforming to the TCP congestion control mechanism. For the identified attack paths, routers make further distinction between legitimate and attack flows using their packet-drop intervals, and limit the bandwidth of attack flows. These functions are designed in consideration of router's resource constraints.

[8]The random threshold (Q_{th}) functions as an early congestion notification much like the RED queue (the drop probability goes up as the queue size grows), yet it does not require complex parameter calibration as the RED queue does.

1) Attack Path Identification: For bandwidth guarantees on legitimate traffic, we first identify the contaminated domains that originate attack traffic and then identify responsible flows for the congestion within those domains. If we assume all TCP flows compete for a common bottleneck bandwidth (i.e., that of an attack-target link), the packet drop ratio of a path, which is denoted by γ_{S_i}, can be expresses as $\gamma_{S_i} = \frac{8}{3W_i(W_i+2)}$. And, the packet drop rate (denoted by δ_{S_i}) is $\delta_{S_i} = \frac{2 \cdot n_{S_i}}{W_i \cdot Rtt_i}$ [27]. Since $\frac{C_{S_i}}{n_{S_i}} = \frac{3}{4}\frac{W_i}{Rtt_i}$, $\delta_{S_i} = \frac{8C_{S_i}}{3W^2_{S_i}} = \frac{8C_{S_i}}{3\left(-1+\sqrt{1+\frac{8}{3\gamma_{S_i}}}\right)^2}$. Given these relationships, we can estimate the number of flows in the path as $\hat{n} = \delta_{S_i}\frac{W_i}{2}\text{RTT}_i = \frac{1}{2}\delta_{S_i}(-1+\sqrt{1+\frac{8}{3\gamma_{S_i}}})\text{RTT}_i$. Since we assume that all flows are actively competing for C_{S_i}, $\hat{n}$ gives the lower bound of the number of flows. Hence, if the number of distinct flows that experience drops during $\frac{W_i}{2}\text{RTT}_i$ is less than $\hat{n}$, there certainly exist attack flows.

2) Attack Flow Accounting: A router keeps track of the packet drops of attack paths to identify attack flows and those packet drops are recorded in a bloom filter for efficiency. Each record of a bloom filter consists of three fields: sequence number (since a record created), last-update time, and the number of extra packet drops, which are denoted by t_s, t_l, and d_i respectively. For space efficiency, t_s and t_l are recorded in a certain time granularity t_{base} (e.g., 10ms). Sequence number is the number of congestion epochs (i.e., $\frac{W_i}{2}\text{RTT}_i$) elapsed since the record was created. Since a flow would have a single packet drop during a congestion epoch in normal operation, the number of extra packet drops normalized by the sequence number indicates the strength of attack and determines the flow's preferential drop rate. The last-update time is used to remove a legitimate flow's normal drop from the filter.

Let t_c and sz_t be the current time in ticks and the largest time that can be presented (i.e., 2^{bits}) respectively. Let RTT_i be the path RTT of S_i. Whenever a packet drop occurs, its flow-identifier (i.e., source and destination addresses) is hashed to find the corresponding entries in the filter. Each field is updated by the algorithm shown in Algorithm V-B.2. The algorithm describes that the counter values of a flow are increased on every packet drop, yet are decreased by one in every RTT. This counts the *extra* drops that each flow have, which in effect, approximates the flow's send rate as they have linear dependency. For high-rate flows, we increase the start-time as well, which allows to measure flows' bandwidth up to d_i times its fair bandwidth. Also, we can identify flows that send $1/\text{sz}_t$ times more packets than the fair amount.

3) Preferential Drops: We determine the preferential drop ratio such that the bandwidth used by attack flows would not exceed that of legitimate TCP flows.

As the excess bandwidth of an attack flow can be approximated by $\text{P}_e = \frac{d_i}{t_s - 1}$, the preferential drop ratio on the flow is expressed as follows.

$$\text{P}_{p.d} = \frac{d_i}{t_s + d_i - 1} = \frac{t_s \cdot 2^k - 1}{(1+2^k) \cdot t_s - 1} \qquad \text{(V.1)}$$

When the above packet admission policy is applied, attack

Algorithm 2: Filter Update

```
if f_i is a new entry then
    /* record time with a counter where a single tick
    represents t_base sec. */
    t_l = t_c / t_base mod sz_t;
    t_s = 1;
    d_i = 1;
    Exit;
if t_c ≥ t_l + RTT_i then
    t_temp = t_c / t_base mod sz_t;
    /* decrase the number in every congestion epoch */
    d_i = d_i + 1 - ⌊(t_c − t_l)/RTT_i⌋;
    /* a renewed or new entry */
    if d_i == 1 then
        t_l = t_temp;
        t_s = 1;
    else
        /* for high-rate flows */
        if 2^k · t_s ≥ d_i then
            t_s -= (t_temp − t_l);
        t_l = t_temp;
else
    d_i = d_i + 1;
```

flows which send more than P_e of the fair bandwidth would be identified and rate-limited. For example, if we allocate 4 bits for t_s, the maximum of P_e would be 6.25% and the corresponding $P_{p.d}$ would be 5.88%. On the other hand, for attack flows, the maximum drop ratio becomes $P_{p.d} = \frac{2^k \cdot t_s - 1}{2^k \cdot t_s}$ since we do not increase t_s if $2^k \cdot t_s \geq d_i$. Hence, if we use k=2 (i.e., two bits for the drop count per period), we can limit the bandwidth of a flow which sends 64 times of its fair bandwidth (in this case, $P_{p.d} = 0.984$). We block those high-rate flows for a period of time without applying the preferential drop policy. (A lower threshold for filtering can be applied by design) .

4) Probabilistic Filter Update: Though our filter is updated by dropped packets, those *dropped* packets degrade router's performance significantly since they cause frequent memory accesses. To avoid this, we record packet drops proportional to the drop rate; i.e., the service probability of a flows that sends four times its fair bandwidth is 1/4, and the frequency of memory update is 1/4 of its actual drop (with prob. 1/4, update in every one of 4 drops)

5) Probabilistic Array Selection: As the number of flows grows, the false positive ratio of the flow filter increases exponentially since the ratio is expressed as $P_{f_j} = \left(1 - e^{-\frac{n}{2^b}}\right)^m$, where n is the number of flows, m is the number of arrays, and 2^b is the size of arrays. To prevent the filter from being spoiled by a large number of attack flows (which produces a high false-positive ratio), flows that belong to highly populated attack domain only update k of m arrays in the filter.

Let $\mathcal{A}$ and $n^{\mathcal{A}}$ be the set of attack domains and the number of flows originating from those domains respectively. Then, the false positive ratio of the filter on the flows of $\mathcal{A}$ and that of the remaining domains (denoted by $\mathcal{G}$) can be expressed as follows.

$$P_{f_j} = \left(1 - e^{-\frac{n - n^{\mathcal{A}} + \frac{k \cdot n^{\mathcal{A}}}{m}}{2^b}}\right)^i, i = \begin{cases} k & f_i \in \mathcal{A} \\ m & f_i \in \mathcal{G} \end{cases} \quad \text{(V.2)}$$

We find k such that $n - n^{\mathcal{A}} + \frac{n^{\mathcal{A}}}{k} \leq n^{th}$, so that the false positive ratio of the legitimate flows would be lower than the value determined by n^{th}.

If four 24-bit arrays are used for the filter (i.e., m = 4, b = 24), the false positive ratio for 0.5 million flows is only 7.4×10^{-7}. This ratio will be as low as 1.12×10^{-5} even in the presence of 4 million attack flows. And, such filter needs 24 × 4 × 2 Bytes = 128 MBytes memory space. Also, in order to reduce the overhead caused by frequent memory accesses of a large number of attack flows, we update the filter proportional to k/m. That is, the flows of $\mathcal{A}$ updates the filter with probability k/m and with the value of m/k. In this way, the memory-access frequency is limited.

VI. Functional Evaluation

In this section, we present our simulation results for various attack scenarios to evaluate our design. We use the tree topology shown in Fig. 5, where both the height and degree of the tree are set to three (i.e., 27 paths). We attach 30 legitimate (TCP) sources to every leaf node, and attach 60 additional attack sources to each of 6 leaf nodes designated as attack nodes (i.e., we use 360 attack sources). Each legitimate source is configured to send a 12MB file to a destination server located across the link targeted for flooding and randomly starts its transmission between zero and five seconds. Each attack source is configured to change the *send* rate from one to ten times its fair bandwidth depending on the simulation scenario. The target-link capacity is set to 500 Mbps.

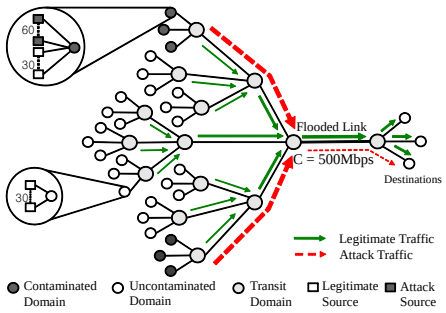


Fig. 5: Simulation Topology

A. Attack Confinement

We illustrate the attack-confinement effects (e.g., bandwidth guarantees to legitimate path identifiers) during link flooding for three different types of attacks: a high-population TCP attack (Fig. 6(a)), a Constant Bit Rate (CBR) attack (Fig.

6(b)), and a Shrew attack (Fig. 6(c)) [6]. The aim of the high-population TCP attack is to reduce the bandwidth of legitimate flows at a congested router. These flows adapt their *send* rate to the available bandwidth and as a consequence become indistinguishable from legitimate flows at that router. However, FLoc confines the effect of this attack to a single path identifier, since bandwidth is separately guaranteed to each path. As FLoc guarantees the same bandwidth allocation to each of the 27 paths shown in Fig. 5 (i.e., 500 Mbps/27 = 18.5 Mbps per path), the bandwidths received by individual path identifiers shown in Fig. 6(a) are almost identical regardless of their (legitimate or attack) population.

In the CBR attack, each of the 360 attack sources (i.e., "bots") sends 2.0 Mbps CBR traffic through the targeted link. Thus the overall attack strength reaches 720 Mbps – an amount that would disrupt most legitimate TCP flows through a 500 Mbps link. Fig. 6(b) shows that the legitimate-path flows get higher bandwidth in this attack than in the high-population TCP attack. This is because the token-bucket mechanism is activated early for attack paths and the fixed (non-increased) token-bucket sizes limit the traffic on these paths. At the same time, attack flows are easily identified by their low MTDs and are rate-limited accordingly.

In the Shrew attack, each attack source sends 2.0 Mbps traffic only during 0.25RTT seconds within an interval of RTT seconds. Also, we coordinate all attack sources to maximize the attack strength. Fig. 6(c) shows that the bandwidths received by the flows of a legitimate path are almost identical to (or slightly higher than) those received by legitimate flows in the CBR attack. This means that the Shrew attack is handled at least as well as the CBR attack. Yet, the service rate has a higher variance during the Shrew attack. This is because flows that experience packet drops synchronized with the attack traffic utilize less bandwidth than the unsynchronized flows.

B. Robustness of Bandwidth Guarantees

We use the distribution of the bandwidth received by legitimate flows in legitimate paths to illustrate the robustness of FLoc's bandwidth guarantees under various attack strengths. The strength of FLoc in this area is compared with that of an aggregate-based defense scheme (i.e., Pushback [5]) and a flow-based defense scheme (i.e., RED-PD [16]). Fig. 7 shows the cumulative distribution function (CDF) of the bandwidth received by the flows of legitimate paths (measured in a 20 to 80 second interval) for various attack strengths.[9] In FLoc's case (Fig. 7(a)), the bandwidth distributions to flows of legitimate paths are nearly identical for various attack strengths, and the mean bandwidth is close to the ideal fair bandwidth; i.e., 18.5 Mbps for each of the 30 legitimate paths yields 0.617 Mbps per legitimate flow. Also, FLoc provides

[9]We illustrate the bandwidth distribution of all flows in legitimate paths since the link bandwidth is allocated in equal amounts to all 27 paths (i.e., 18.5 Mbps per path). Also, we increase the send rate of each attack source starting from 0.2 Mbps, since this is the fair per-flow bandwidth in attack paths (i.e., 18.5 Mbps / (60 attack + 30 legitimate flows) = 0.205 Mbps / flow).

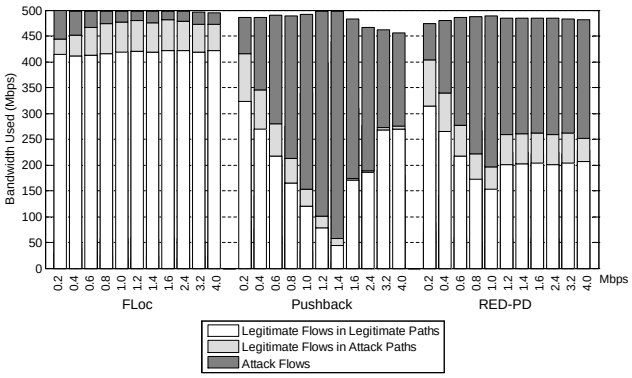


Fig. 8: Differential bandwidth used at flooded link. (Flow rates of each attack source are shown on the horizontal axis.)

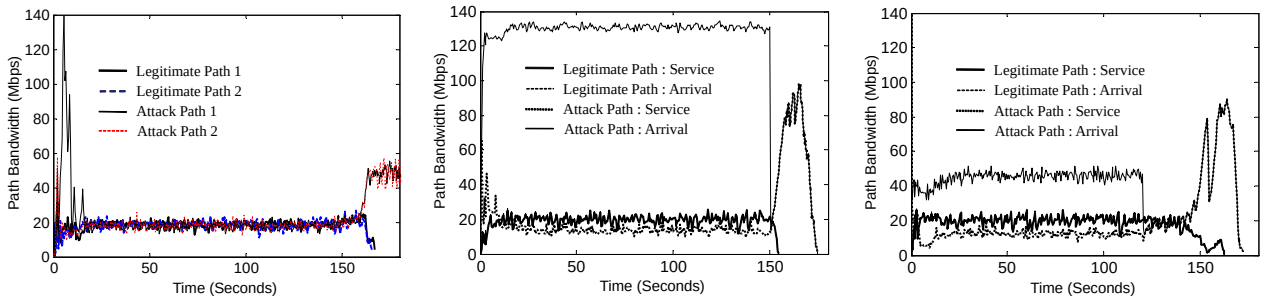
per-flow fairness comparable to that of the RED queue in the normal (no-attack) case (viz., Fig. 7(c)). RED-PD outperforms Pushback slightly in low-rate attacks, namely when overall rates are less than 500 Mbps, yet both RED-PD and Pushback do not provide effective protection to legitimate flows; viz., Figs. 7(b), 7(c) where their CDF curves move left and drop below the "no attack" curve.

C. Differential Bandwidth Guarantees

Next, we evaluate the differential bandwidth guarantees achieved by FLoc's path-aggregation policies. We set the maximum number of bandwidth-guaranteed paths to 25 (i.e., $|\mathcal{S}|_{max}$ = 25 of the 27 paths), which allocates 20 Mbps bandwidth to each of them. This requires at least four out of six attack path identifiers of the contaminated domains in Fig. 5 to be aggregated at the congested router.

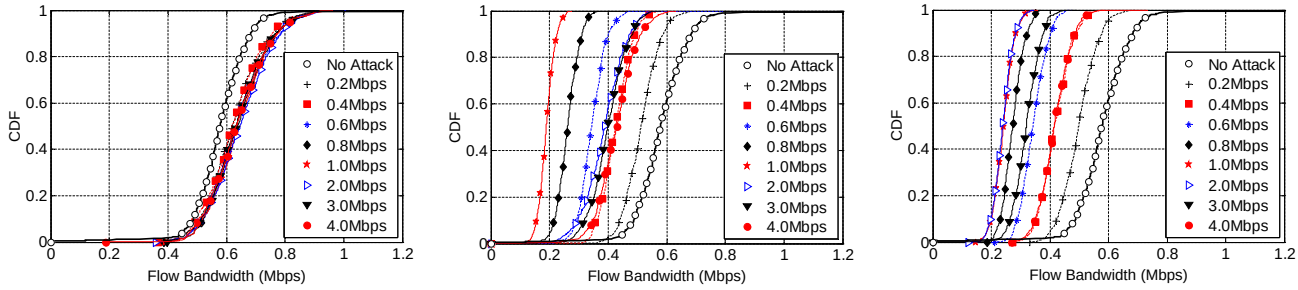
Fig. 8 illustrates a comparison between the differential bandwidth guarantees provided by FLoc, and the bandwidths provided by Pushback and RED-PD, at different attack rates.

With FLoc, the bandwidth used by the flows of legitimate paths is over 80% of the link bandwidth, which is nearly identical to the proportion of legitimate paths (i.e., 21/25 = 0.84). Recall that there are twice as many attack sources as legitimate sources in contaminated domains. Consequently, the total bandwidth used by attack flows is higher than that of legitimate flows in the same paths (i.e., attack paths) even though per-flow bandwidth is higher for legitimate flows (viz., FLoc under 0.2 and 0.4 Mbps attacks in Fig. 8). As attack sources increase their *send* rates, the traffic from those sources are more aggressively rate-limited by FLoc's preferential drop policy. This leaves more bandwidth for legitimate flows in attack paths, while the bandwidth of legitimate-path flows remains unaffected. With Pushback, the bandwidth of legitimate-path flows decreases until the attack traffic dominates the link bandwidth and the packet-drop rate triggers the activation of rate throttling (i.e., in a "bandwidth soaking" attack [6]). Once rate throttling is performed, the bandwidth of the legitimate-path flows increases (viz., the last four bars for Pushback in Fig. 8). However, the bandwidth of the legitimate flows in attack paths decreases significantly, since Pushback does not



(a) High-population TCP attack (b) CBR attack (c) Shrew attack

Fig. 6: Localization effects for three different attacks. Legitimate and attack paths are randomly chosen from 27 paths. Legend: For each path identifier, "Service" denotes the bandwidth received, and "Arrival" the bandwidth requested



(a) FLoc (b) Pushback (c) RED-PD

Fig. 7: Cumulative Distribution Function (CDF) of legitimate path flows' bandwidth for various CBR attack rates. Note: For FLoc, the bandwidth of legitimate-path flows has nearly identical distribution independent of attack strength. In contrast, for Pushback and RED-PD, this bandwidth decreases significantly (i.e., the CDF curve moves left) as the attack strength increases.

implement any per-flow measure to counter attack traffic; i.e., "collateral damage" within attack aggregates is unavoidable. RED-PD limits the bandwidth of attack flows more than Pushback for low-rate attacks, and protects legitimate flows in attack paths for all attack strengths. However, RED-PD is less effective in protecting legitimate-path flows (whose bandwidth is shown as white bars in the figure) than Pushback, when the send rates of attack sources are very high (e.g., 3.2 and 4.0 Mbps). This is because RED-PD allocates the same bandwidth to flows (including extraordinarily high-rate, attack flows) regardless of their send rates. FLoc outperforms both Pushback and RED-PD, both in terms of the bandwidth guaranteed to legitimate traffic and link-bandwidth utilization for all attacks rates.

Legitimate-Path Aggregation. For legitimate-path aggregation, we place 15 legitimate-flow sources in each one of three sibling nodes (domains) and 30 legitimate sources in the other nodes. Since a third of 21 uncontaminated domains have 15 sources (i.e., 105 sources in all) and the others have 30 sources (i.e., 420 sources in all), there would be 525 flows originating from legitimate domains at the congested router.

Fig. 9 shows that without aggregation, nearly 80% of the legitimate-path flows receive less bandwidth than the other 20% of the flows (viz., the CDF marked by upward-pointing triangles). This implies that the flows of less populated paths

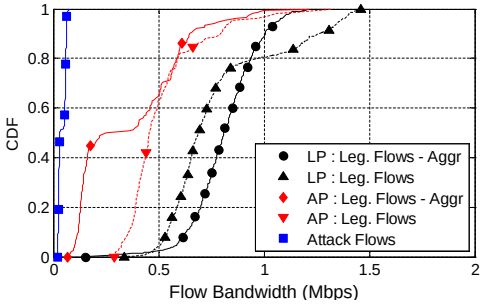


Fig. 9: Differential Guarantees: Legitimate vs. Attack Flows. Legend: The CDFs labeled by "Aggr" illustrate the results of path-identifier aggregation. LP and AP denote Legitimate and Attack Paths, respectively.

(i.e., domains), which account for 105/525 = 20% of all legitimate flows, consume much more bandwidth (i.e., two times more in our simulation) than those of highly populated paths (domains). This uneven bandwidth distribution disappears when legitimate-path aggregation is performed (viz., the CDF marked by circles).

Attack-path aggregation unavoidably penalizes legitimate

flows of an attack path to some extent[10] (but *never* denies link access to them). Since three attack paths are aggregated by the constraint of $|\mathcal{S}|_{max}$=25, the legitimate flows of the *aggregated* attack paths only get a third of the fair bandwidth allocated to each path identifier. As the figure shows, these flows, which account for half of all legitimate flows of attack paths, receive somewhat less bandwidth than the legitimate flows of *non-aggregated* attack paths, and certainly less bandwidth than the legitimate-path flows – the expected result of differential bandwidth guarantees.

D. Covert Attacks

We evaluate the effect of covert attacks, where attack sources establish a large number of "legitimate-looking" flows through the congested link with "different IP destinations", on FLoc, Pushback and RED-PD. For this simulation, we configure attack sources to connect with multiple destinations and to send low-rate CBR traffic (i.e., 0.2 Mbps per flow) to each destination to make individual attack flows look "legitimate". (Recall that 0.2 Mbps is the fair bandwidth of each flow in attack paths; viz., Section VI-B.) The number of destinations to which an attack source connects concurrently within a router is increased from 1 to 20, which increases the send rate of individual attack sources from 0.2 Mbps to 4.0 Mbps. Note that since we use 360 attack sources in this simulation, the targeted link is already completely flooded at 7 connections per source (i.e., $360 \times 7 \times 0.2$ Mbps = 504 Mbps which exceeds the link capacity of 500 Mbps). To illustrate the use of our covert attack countermeasures, we restrict the maximum number of concurrent connections per single source within a single router to 2 (i.e., two capabilities are made available to each multi-flow source, namely n_{max} = 2),[11] thereby limiting the bandwidth available to attack sources to 28.8% of the total link bandwidth. Of course, a source's multiple connections through multiple routers are not affected by this restriction.

Fig. 10 illustrates the bandwidths used by legitimate and attack flows at the flooded link. Whenever an attack source increases the number of concurrent connections (i.e. flows) through a single target link, its legitimate-looking flows are classified as a single high-rate flow by FLoc. Hence, packets of the attack source are preferentially dropped at that router, much like those of CBR attack sources illustrated in the previous section. Pushback's reaction to these attacks is too late to make a difference. Its rate-control mechanism is triggered only at 12 flows per attack source when the maximum available link bandwidth is already exceeded by 52% (i.e., 360 $\times$ 12 $\times$ 0.2 Mbps = 864 Mbps vs. 500 Mbps maximum link bandwidth). Furthermore, Pushback neither prevents collateral damage within attack paths nor does it handle low-rate attacks

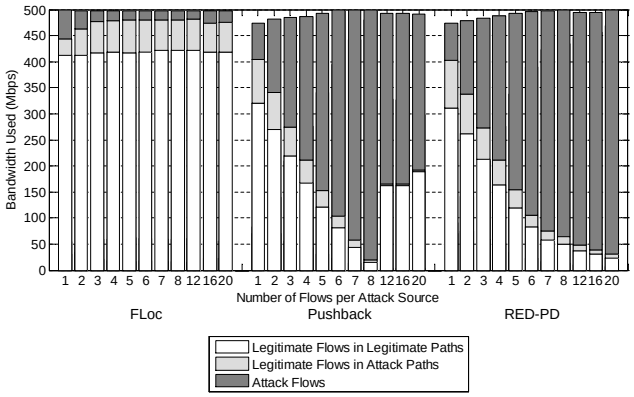


Fig. 10: Bandwidth used at flooded link under covert attacks.

(e.g., covert attacks whose total send rate is well below the maximum link bandwidth). RED-PD fails to counter covert attacks since bandwidth it provides to legitimate flows decreases as the number of attack flows increases. For example, when an attack source directs 20 concurrent connections through a single router, 810 legitimate and 20×360 = 7200 attack flows co-exist in that router. Hence, per-flow, fair bandwidth allocation provides more than 89.9% of the link bandwidth to attack flows.[12] This illustrates the lethality of covert attacks against typical schemes that act on either flow-aggregate or individual-flow basis to counter flooding attacks. Clearly, fair bandwidth allocation mechanisms cannot possibly counter such covert attacks.

VII. Internet-scale Simulations

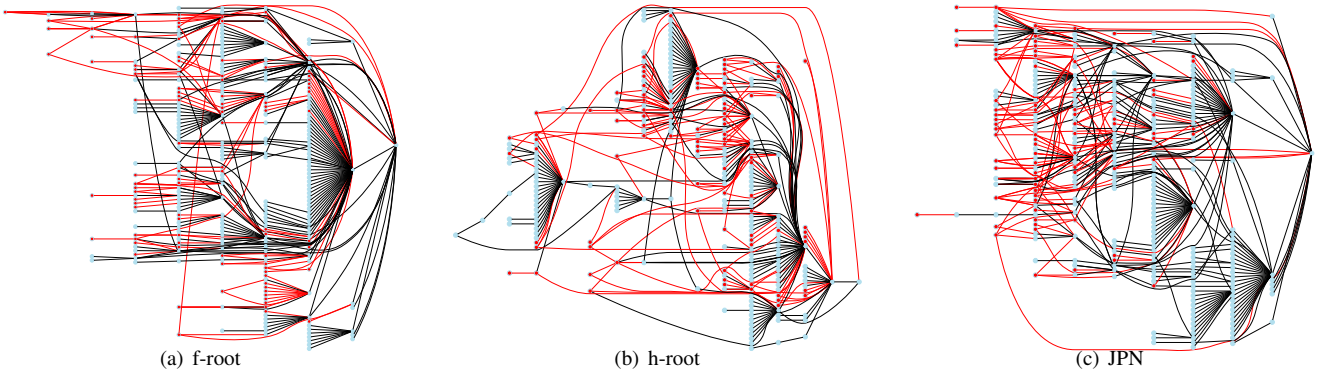
A. Topologies

We use several datasets for realistic Internet-scale simulations: CAIDA Skitter-Maps [28], Composite Blocking List (CBL) [13] and GeoLite ASN [29]. A Skitter-Map consists of a large set of routing paths measured from a root-DNS server to randomly-chosen hosts (300 $\sim$ 400 thousand hosts) in the Internet. Skitter-Map is used as a reference topology for generating simulation topologies. CBL contains the list of IP addresses of active spam-bots in the Internet and GeoLite ASN is a database that maps IP address block to the corresponding ASN. Using these datasets, we construct an AS' bot-distribution and based on which, we construct a simulation topology for a given simulation size. For example, when a given simulation size (i.e., the number of legitimate and attack sources) is set, a simulation topology is generated as follows. Host IP addresses and the corresponding paths (starting with the same subnet address with hosts) are selected from the Skitter-Map with the following distribution: the attack hosts are selected such that they have the same AS distribution as that of CBL; and legitimate hosts are randomly selected in proportion to AS population (i.e., hosts in larger ASs have a

[10]This is the case for *all* aggregate-based defenses, including those where *a priori* information regarding the legitimacy of a flow path is given, such as CDF-PSP [7].

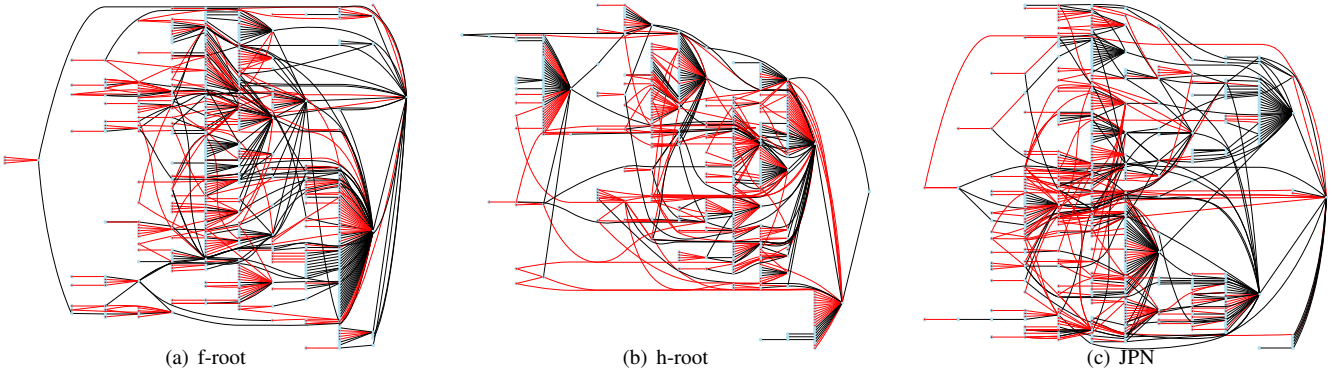
[11]We note that n_{max} = 2 is used only for the purposes of illustrating comparative performance analysis. We let n_{max} be a configurable parameter that can be differently chosen at different locations (i.e., routers).

[12]Legitimate TCP flows cannot fully utilize the allocated bandwidth due to their congestion control mechanism. This is why they use much less than 10% (i.e., $\approx$ 5%) of the link bandwidth in the simulation.



(a) f-root (b) h-root (c) JPN

Fig. 11: Topologies used for simulations (# of attack ASs is 100).



(a) f-root (b) h-root (c) JPN

Fig. 12: Topologies used for widely distributed attack simulations (# of attack ASs is 300).

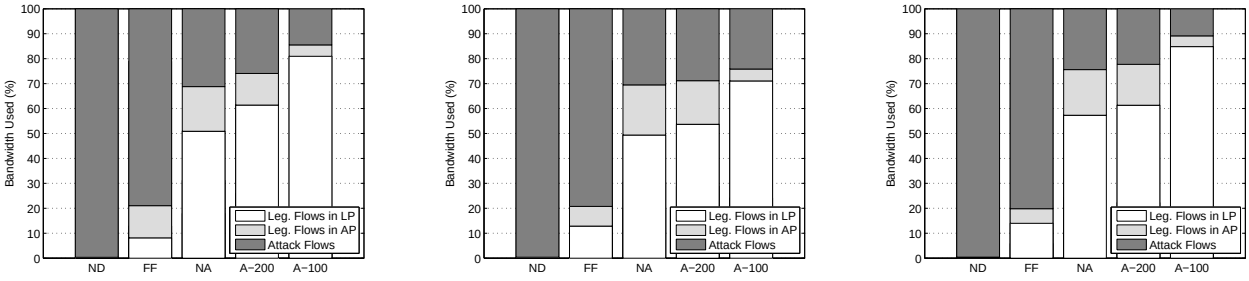
higher chance to be chosen)[13]. We note that though CBL is the list of spam-bots, they would represent the geographical distribution of bots that can be potentially used for flooding attacks. Among several distinct Skitter-Maps available, we use three different Maps (i.e., f-root, h-root, and JPN) to observe the consistency of simulation results under different topologies. In all simulations, we place 10,000 legitimate sources in 200 ASs randomly and 100,000 attack sources with several different distributions. Fig. 11 shows the topologies where attack sources are distributed in 100 ASs (i.e., for localized attack simulations) and Fig. 12 shows the topologies where attack sources are distributed to a wider extent (i.e., 300 ASs). In the figures, red lines represent the link that is directly connected to an attack AS (that contains attack sources); black lines represent all other links. And, individual nodes (ASs) are aligned by their distance (in AS hops) to the attack target located at the rightmost node. In generating these topologies, we intentionally attach 30 % of legitimate sources to attack ASs in order to observe differential guarantees; i.e., how FLoc separates the bandwidth of legitimate flows from that of attack flows within a path.

As the figures illustrate, the network topologies, even if they are presented in a AS graph, are very complicated and are widely different in terms of the degree and distance according to the location where they are constructed. Because of this wide topological diversity, any bandwidth allocation made by a router's (or an AS's) local information on traffic origin would not differentiate legitimate traffic from attack traffic.

[13]AS population is found from GeoLite ASN.

B. Simulator

For Internet-scale simulation, we design a new simulator that scales to include millions of network elements (i.e., hosts, routers and links). Our simulator runs in a discrete-time fashion: individual packets advance a single router-hop in a time tick and a router handles all the packets arrived during the tick at the same time. Hence, if we assume a 5ms time tick, the end-to-end delay for a source located 30-hops from the destination would be 150ms. Whenever a packet drop is necessary, a router randomly selects a packet from the all queued packets during a time tick so as to approximate the behavior of the router's queue in a finer granularity. Though our simulator does not (and is not intended to) capture the network characteristics precisely (e.g., router's queue size variation, queueing delay in a microsecond precision), its time precision and corresponding approximation would be fine enough to describe the general bandwidth characteristics of individual TCP flows during flooding attacks (as the queue size and queueing delay are stable). In simulations, the bottleneck-link capacity is set to 16000 packets per tick, which corresponds to 40 Gbps link (i.e., OC-768) if a 5ms tick is assumed.



(a) f-root (b) h-root (c) JPN

Fig. 13: Bandwidth used at the congested link under localized attacks.

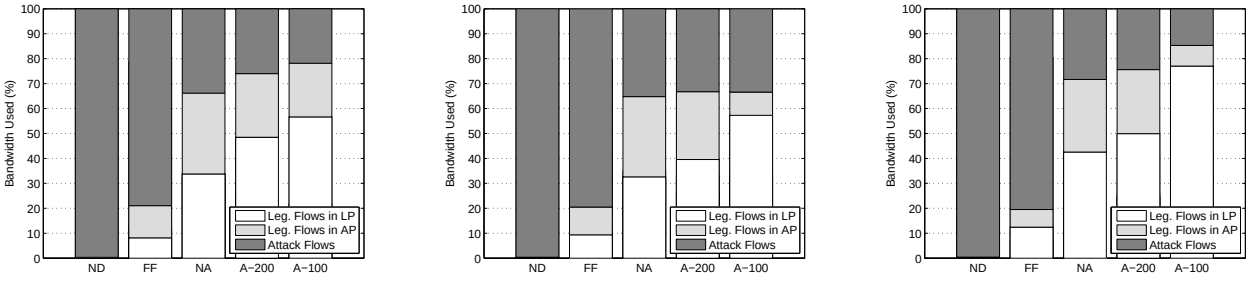
C. Bandwidth Guarantees

The effectiveness of FLoc in large-scale attacks is evaluated using the topologies shown in Fig. 11 and by comparing FLoc's results with those of no defense and per-flow bandwidth fairness strategy. In our simulator, a per-flow fairness scheme is implemented in such a way that legitimate TCP flows are allocated *at least* as much bandwidth as that of attack flows: all packets of legitimate flows are assigned a high priority yet those of attack flows are assigned a high priority up to their fair bandwidth; and routers process the high priority packets ahead of other normal priority (attack) packets. That is, high priority packets compete bandwidth only with themselves and normal priority packets are serviced only when network links become idle.

Fig. 13 show the bandwidth used by legitimate flows in legitimate paths (whose origin AS only contains legitimate sources), legitimate flows in attack paths (whose origin AS contains attack sources), and attack flows. The results of no defense scenarios illustrate the lethality of flooding attacks, and are used for comparative evaluation. The results show that, without attack defense (denoted by ND in the figures), legitimate flows, regardless of whether they originate from legitimate or attack ASs, are completely denied service at the target link. When all active flows share fair bandwidth (denoted by FF), legitimate flows use about 20 % of the congested link bandwidth. The bandwidth used by legitimate flows is substantially higher than their fair bandwidth (which is about 9 %) because all legitimate TCP flows are provided higher priority service and high priority attack packets from highly contaminated ASs are dropped on the way to the target (as they clog some other links on the path). The bandwidth used by legitimate flows in attack paths indicates the proportion of legitimate sources in attack ASs (i.e., overlap between legitimate and attack ASs). FLoc is evaluated with three different scenarios: no aggregation, aggregation with 200 bandwidth guaranteed paths, and aggregation with 100 bandwidth guaranteed paths. Each of those scenarios are denoted by NA, A-200, and A-100 respectively. Note that no aggregation means all ASs are allocated fair bandwidth and aggregation with smaller number of bandwidth guaranteed paths means aggregation is processed to a further extent. Even without aggregation, the bandwidth used by legitimate flows is close to 70 % in f-root and h-root topologies, and over 75 % in JPN topology. These results ensure that FLoc *localizes* the effects of large-scale flooding attacks effectively. In attack paths, attack flows used more bandwidth than legitimate flows in all three topologies. However, in a flow basis, legitimate flows use much more bandwidth than attack flows since the number of legitimate sources (which is 3 $\tilde{4}$K) is much lower than that of attack sources (i.e., 100K). As aggregation proceeds, legitimate flows in legitimate paths get more bandwidth allocation, yet legitimate flows in attack paths get reduced bandwidth allocation like attack flows. Aggregation produces better results in JPN topology, since most attack ASs are located farther from the destination and their paths are better separated from those of legitimate AS as Fig. 11(c) shows. These simulation results corroborate FLoc's differential guarantees – better bandwidth guarantees for legitimate domains and better bandwidth guarantees for legitimate flows in attack domains – in realistic large-scale attacks.

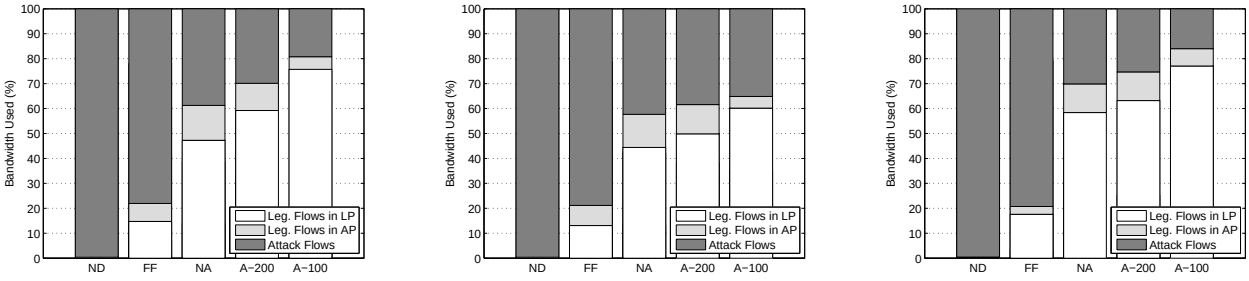
Next, we generate more complex topologies where attack sources are distributed more widely, namely to 300 ASs, and run simulations. As one can imagine, simulation results of no defense and per-flow fairness strategy are similar with those of previous simulations (viz., Fig. 14). In FLoc, the bandwidth used by legitimate flows in legitimate paths decreased significantly since (1) the bandwidth allocation to those paths decreased inversely proportional to the active path number and (2) more legitimate ASs turn into attack ASs as attack sources spread. Meanwhile, the bandwidth used by legitimate flows in attack paths increased as they are allocated more bandwidth than attack flows. We note that even in wide dispersion of attack sources, the bandwidth used by *all* legitimate sources decreased slightly due to differential bandwidth guarantees. Aggregation is more effective in defending against widely distributed attacks as it favors legitimate flows in legitimate paths.

Finally, we extend our previous simulations with new topologies where legitimate ASs are better separated from attack ASs by discarding intentional placement of legitimate sources at attack ASs (viz., Section VII-A). That is, all legitimate sources and attack sources are attached to the topol-



(a) f-root (b) h-root (c) JPN

Fig. 14: Bandwidth used at the congested link for distributed attacks.



(a) f-root (b) h-root (c) JPN

Fig. 15: Bandwidth used at the congested link when more legitimate ASs are isolated from attack ASs.

ogy based on the reference distributions; i.e., AS population and bot distribution respectively. In this scenario, FLoc can increase bandwidth allocation to legitimate flows in legitimate paths since more legitimate paths are guaranteed bandwidth yet, at the same time, more attack paths are limited bandwidth via aggregation. Fig. 15 shows that under this isolated attack scenario, attack traffic can be more precisely distinguished from legitimate traffic and hence their effects can be *localized*.

VIII. Conclusions

In this paper, we presented a per-domain bandwidth control mechanism called "FLoc" that provides precise bandwidth guarantees to the aggregate-flows of the same origin. In addition to the bandwidth guarantees, our mechanism can identify and rate-limit a variety of attack flows: flows of high-rate attacks, low-rate Shrew attacks, and more sophisticated covert attacks. Comprehensive simulations under those attack scenarios and comparisons of the results with those of per-aggregate based (i.e., Pushback) and per-flow based (i.e., RED-PD) mechanisms show the effectiveness and robustness of FLoc especially in defending against low-rate and covert attacks.

Furthermore, we evaluated FLoc under realistic network topologies and attack distributions. The simulation results show that FLoc defends against large-scale bot attacks very effectively, via localizing the effects of attacks within bot-contaminated domains. With FLoc, flows of uncontaminated (or lightly contaminated) domains by bots were able to use guaranteed bandwidth in spite of large-scale attacks, and legitimate flows of contaminated domains used substantially higher bandwidth than attack flows of the same domain.

Acknowledgment

This research was supported in part by US Army Research Laboratory and the UK Ministry of Defence under Agreement Number W911NF-06-3-0001 and by the US Army Research Office under Contract W911NF-07-1-0287 at the University of Maryland. The work on Internet simulations was supported by Northrop Grumman Cyber Research Consortium under Contract NGIT2009100109 at CyLab, Carnegie Mellon University. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the US Army Research Laboratory, US Army Research Office, the U.S. Government, the UK Ministry of Defense, the UK Government, or Northrop Grumman.

References

[1] T. Anderson, T. Roscoe, and D. Wetherall, "Preventing Internet denial-of-service with capabilities," in *Proc. of Hotnets-II*, 2003.

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

[3] X. Yang, D. Wetherall, and T. Anderson, "A DoS-limiting network architecture," in *SIGCOMM '05*, 2005.

[4] Y. Xu and R. Guérin, "A double horizon defense design for robust regulation of malicious traffic," *SecureComm*, 2006.

[5] R. Mahajan, S. M. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker, "Controlling high bandwidth aggregates in the network," *Comput. Commun. Rev.*, vol. 32, no. 3, 2002.
[6] Y. Xu and R. Guérin, "On the robustness of router-based denial-of-service (dos) defense systems," *Comput. Commun. Rev.*, vol. 35, no. 3, 2005.
[7] J. Chou, B. Lin, S. Sen, and O. Spatscheck, "Proactive surge protection: a defense mechanism for bandwidth-based attacks," in *Proc. of the 17th USENIX Security symposium*, 2008.
[8] S. Staniford, V. Paxson, and N. Weaver, "How to Own the Internet in Your Spare Time," in *Proceedings of the 11th USENIX Security Symposium*, 2002.
[9] M. A. Rajab, F. Monrose, and A. Terzis, "On the impact of dynamic addressing on malware propagation," in *WORM '06*, 2006.
[10] D. Dagon, C. Zou, and W. Lee, "Modeling Botnet Propagation Using Time Zone," *Network and Distributed System Security Symposium*, 2006.
[11] M. Casado and T. Garfinkel, "Opportunistic measurement: Extracting insight from spurious traffic," in *HotNets*, 2005.
[12] Z. Chen, C. Ji, and P. Barford, "Spatial-temporal characteristics of internet malicious sources," in *Proceedings of IEEE INFOCOM*, April, 2008.
[13] "http://cbl.abuseat.org/."
[14] A. Kuzmanovic and E. W. Knightly, "Low-rate tcp-targeted denial of service attacks: the shrew vs. the mice and elephants," in *SIGCOMM '03*. New York, NY, USA: ACM, 2003.
[15] A. Studer and A. Perrig, "The coremelt attack," in *ESORICS*, Saint Malo, France, September 2009.
[16] R. Mahajan, S. Floyd, and D. Wetherall, "Controlling high-bandwidth flows at the congested router," in *ICNP '01*, 2001.
[17] I. Stoica, S. Shenker, and H. Zhang, "Core-Stateless Fair Queueing: Achieving Approximately Fair Bandwidth Allocations in High Speed Networks," in *SIGCOMM*, 1998.
[18] W. C. Feng, D. D. Kandlur, D. Saha, and K. G. Shin, "Stochastic Fair Blue: A Queue Management Algorithm for Enforcing Fairness," in *INFOCOM*, 2001.
[19] R. Pan, B. Prabhakar, and K. Psounis, "Choke, a stateless active queue management scheme for approximating fair bandwidth allocation," in *INFOCOM*, 2000, pp. 942–951.
[20] J. Ioannidis and S. M. Bellovin, "Implementing Pushback: Router-Based Defense Against DDoS Attacks," in *NDSS*, 2002.
[21] D. G. Andersen, H. Balakrishnan, N. Feamster, T. Koponen, D. Moon, and S. Shenker, "Accountable Internet Protocol (AIP)," in *Proc. ACM SIGCOMM*, Seattle, WA, Aug. 2008.
[22] A. Yaar, A. Perrig, and D. Song, "Pi: A Path Identification Mechanism to Defend against DDoS Attacks," in *IEEE Symposium on Security and Privacy*, 2003.
[23] S. B. Lee, "Localizing the effects of link flooding attacks in the Internet." in *Ph.D Thesis, University of Maryland*, 2009.
[24] C. Partridge, "A Proposed Flow Specification," *RFC 1363*, 1992.
[25] ""http://www.caida.org/research/traffic-analysis/pkt_size_distribution/graphs.xml"."
[26] G. Appenzeller, I. Keslassy, and N. McKeown, "Sizing router buffers," in *SIGCOMM*, 2004.
[27] S. B. Lee and V. Gligor, "FLoc : Dependable Link Access for Legitimate Traffic in Flooding Attacks," in *The 30th International Conference on Distributed Computing Systems*, 2010.
[28] "http://www.caida.org/."
[29] "http://www.maxmind.com/app/asnum."