Loose Source Routing as a Mechanism for Traffic Policies

Katerina Argyraki
EE Department
Stanford University
Stanford, CA 94305
argyraki@dsg.stanford.edu

David R. Cheriton
CS Department
Stanford University
Stanford, CA 94305
cheriton@dsg.stanford.edu

ABSTRACT

Internet packet delivery policies have been of concern since the earliest times of the Internet, as witnessed by the presence of the Type of Service (ToS) field in the IPv4 header. Efforts continue today with Differentiated Services (DiffServ) and Multiprotocol Label Switching (MPLS). We claim that these approaches have not succeeded because they require, either explicitly or subtly, a network-layer virtual circuit mechanism.

In this paper, we describe how adding a form of Loose Source and Record Route (LSRR) capability into the next-generation Internet provides adequate support for transmit and receive policies, including filtering, while avoiding the problems of virtual circuits and the original problems with LSRR in IPv4.

Categories and Subject Descriptors

C.2.2 [**Computer Communication Networks**]: Network Protocols—*Routing Protocols*; C.2.5 [**Computer Communication Networks**]: Local and Wide-Area Networks—*Internet*

General Terms

Design, Reliability

Keywords

Loose Source Routing, Traffic Policies, Route Control, Quality of Service, Filtering

1. INTRODUCTION

The Internet was originally designed with little support for traffic policy. The only traffic policy mechanism available was the Type of Service (ToS) field in the IPv4 header [4], which enabled a sender to specify a priority level and the top preference among low-delay, high-throughput and high-reliability treatment. Today ToS has been deprecated and its descendant, Differentiated Services (DiffServ) [8], has seen limited deployment. Yet, the need for traffic policies in the Internet is evident: an edge system often needs to control how its outgoing traffic is routed, in order to cause its traffic to bypass failure/congestion points; a public-access site often needs to control how its incoming traffic is filtered, in order to mitigate distributed denial-of-service (DDoS) attacks. So, there is a need both for *transmit policies* – to dictate how certain traffic sourced by a certain node be routed – and *receive policies* – to dictate how certain traffic destined to a certain node be filtered.

For lack of a better traffic policy mechanism, nodes often resort to first-hop or last-hop path control: When an edge system routes certain traffic through a specific Internet Service Provider (ISP), and that traffic experiences congestion/loss, the edge system often switches the traffic to a second ISP, in an attempt to force the traffic to take a different path that bypasses the failure point, i.e., the sender attempts to affect the transmission path by controlling the first hop. However, this does not guarantee that the traffic indeed bypasses the failure point. When a public-access site is under DDoS attack, the site often identifies the upstream link that forwards the most traffic and rate-limits or disables it, i.e., the receiver attempts to affect the reception paths by controlling the last hop. However, the last hop is typically within the tail circuit to the victim, so blocking attack traffic comes at the cost of sacrificing most of the "good" traffic as well. So, controlling the first hop of outgoing traffic or the last hop of incoming traffic is not a sufficient traffic policy mechanism.

In this paper, we explore traffic policy support for the next-generation Internet. We argue for enabling an edge system to control the end-to-end path of its outgoing and incoming traffic. Specifically, we propose a variant of Loose Source Record Route (LSRR) and show that it is easy to implement, while it avoids the processing overhead and security problems introduced by LSRR in its traditional IPv4-option form [4]. We argue that our mechanism provides sufficient support for maintaining reliable connectivity and quality of service (QoS) in the face of network failures, as well as filtering undesired traffic. We compare our mechanism to previous proposals, like DiffServ and Multiprotocol Label Switching (MPLS) [18], and observe that these mechanisms introduce *policy state* in the routers, i.e., they require, either explicitly or subtly, the Internet to become "connection-oriented" at the network layer. In contrast, our mechanism requires no policy state in the routers – it is "connection-less". The arguments in favor of our approach are basically the same as those that favored datagrams over virtual circuits in the original Internet design.

SIGCOMM'04 Workshops, Aug. 30+Sept. 3, 2004, Portland, Oregon, USA.

2. A SIMPLE LSRR IMPLEMENTATION

In our approach, the Loose Source Record Route (LSRR) capability is provided by the Wide-Area Relay Addressing Protocol (WRAP), a protocol over IP, which shares similar functionality and mechanism with traditional LSRR: The WRAP header includes a *forward path* and a *reverse path*. A source specifies inside the forward path a sequence of IP addresses that correspond to *relays*, i.e., WRAP-enabled routers on the packet's path. Every time a relay is traversed, its address is moved to the reverse path, so, WRAP header size remains constant.

WRAP is different from traditional LSRR in that it is a "shim" protocol between the IP and transport layers rather than an IP option, i.e., the WRAP header is included as the beginning of the IP payload. Each WRAP packet carries (i) the IP addresses of the communicating end-points and (ii) the IP addresses of all the relays on the path. These addresses exist partly in the IP header and partly in the WRAP header. The IP header always carries the addresses of the last end-point or relay that forwarded the packet and the next relay or end-point that will forward it. The reverse path in the WRAP header carries the addresses of the end-point and relays already traversed; the forward path carries the addresses of the relays and end-point still ahead. Apart from forwarding packets appropriately, a relay also updates packet headers, so that the IP header always refers to the last and next "hops", and the WRAP header to past and future "hops".

Implementing WRAP as a shim protocol versus an IP option has two significant advantages: First, relaying of WRAP packets is easy to implement in hardware. Second, filtering of WRAP packets can be done with conventional wire-speed filters (similar to TCP/UDP-level filters). In contrast, relaying or filtering of traditional LSRR packets incurs the overhead of processing the variable-length IP options field, which typically requires forwarding the packet to the router's CPU [3]. As a result, traditional LSRR introduces security vulnerabilities with hosts that do not have the necessary filtering capabilities.

For example, consider victim node V that filters incoming packets by IP source address and accepts only packets generated by trusted node T. Suppose malicious node M sends to V LSRR packets that appear to have been generated by T and source-routed via M, in order to hijack V-T communication. Traditional LSRR implementations have two characteristics that make it easy for M to succeed: First, the IP source address refers to the original sender of the packet, not the last relay[1]. Second, the receiver of a source-routed packet typically responds by reversing the recorded route. As a result, victim node V accepts attacker node M's packets and sends a response to trusted node T via M. At this point, M has successfully hijacked V-T communication. With WRAP, filtering packets by IP source address is enough to drop any packets not relayed through the designated trusted relays. In general, a WRAP receiver can drop packets received through untrusted paths at wire-speed, by filtering packets based on their reverse path.

WRAP relays correspond to border routers located on administrative domain boundaries – an "administrative domain" being an edge network or an ISP. So, in a WRAP-enabled Internet, an edge system can specify the set of border routers that relay its outgoing traffic and limit the set of border routers that relay its incoming traffic; the latter is accomplished by filtering traffic received through undesired paths.

[1] This is according to RFC 1812 [7]; the original IPv4 RFC 791 [4] (at least as we interpret it) seems to suggest a WRAP-like algorithm, where the IP source address is updated by each relay to refer to the relay's interface that forwards the packet.

3. TRAFFIC POLICIES WITH LSRR

3.1 Transmit Policies

WRAP enables a node to specify a transmit policy for each packet, namely the end-to-end domain-level path that the packet is to follow. An edge system can benefit from this capability to route around failure/congestion points and untrusted regions of the Internet. More specifically, the edge system can compute more than one paths to each potential destination and monitor the paths to decide whether they are functional and meet the desired quality-of-service (QoS) requirements. In this way, if the currently used path fails or performs inadequately, failover time to another path is practically zero. The question is whether it is feasible for an edge system to maintain the required information, i.e., compute and monitor multiple paths to each potential destination.

Zhu et al. show that it is feasible to maintain such information at the edges [26]. They process Internet routing table dumps[2] and count 124,330 network prefixes and 155,706 inter-domain links. They state that a 950MHz processor takes less than 2 seconds to compute one path to each network prefix, and another 30 minutes to compute secondary paths, as independent as possible from the primary ones. In their scheme (Feedback Based Routing or FBR), each access router that connects an edge network to the Internet processes received routing advertisements and computes two paths to each network prefix (the two paths are as independent as possible). The access router further monitors each computed path to ensure that it meets certain pre-configured loss, delay and throughput standards. It is the responsibility of the access router to specify the paths of outgoing traffic by filling in the WRAP headers. In contrast, transit routers at the Internet core simply propagate routing advertisements and relay packets according to WRAP headers[3].

WRAP also enables an alternative approach, where the responsibility to specify outgoing traffic paths is passed to the end-user – one could argue that it is not the access router's, but the end-user's role to specify transmit policies, because the end-user knows the particular QoS requirements of her application. Apart from specifying the domain-level path, the end-user can further tag each packet with a ToS-style label (e.g., requesting low-delay or low-jitter treatment), in order to affect intra-domain packet handling. However, backbone networks today demonstrate "binary" behavior: There are "good" paths, which introduce negligible delay/jitter (due to over-provisioning) and exceed any typical QoS requirements, and "bad" paths, which introduce significant

[2] collected by the RouteViews project on April 2002

[3] Transit routers do not blindly "obey" WRAP headers. In order to relay a packet from neighbor provider A to neighbor provider B, a transit router needs an appropriate forwarding table entry. Such an entry is configured on the router only if the router's provider has agreed to transit traffic from neighbor A to neighbor B.

delay/jitter and loss (due to periodic outages) [17]. In the rare cases that good paths fail, they do so for seconds or minutes [16], in which case they become unacceptable independently of QoS requirements. These trends favor FBR-like schemes, where one entity per edge network monitors paths and ensures that a functional, low-delay, high-throughput path to each destination always exists, versus more complicated schemes, where each application chooses its path and tags its packets according to its own proprietary QoS standards.

Either way, enabling edge systems to compute and specify the entire path of their outgoing traffic is scalable, in the sense that it relieves Internet core routers from computing and monitoring routes from an arbitrary number of sources to an arbitrary number of destinations – the Internet core becomes purely a forwarding engine. An ancillary benefit of providing such path visibility and control to an edge system is that it cannot be fooled by malicious routing advertisements into using a link that does not exist.

3.2 Receive Policies

WRAP enables a node to specify a receive policy for each packet, namely have it accepted, blocked or rate-limited according to its end-to-end path. An edge system can benefit from this capability to selectively filter distributed denial-of-service (DDoS) traffic, without affecting its "good" traffic. More specifically, the victim end-host or router can process the WRAP headers of incoming traffic and identify the end-hosts or domains that forward traffic at excessive rates (see Appendix, Section A.3); the victim can then request from its closest relay (the victim's "gateway") to selectively block traffic received through these paths.

However, as attacks become increasingly distributed, it is unlikely that the victim's gateway alone has enough filtering resources to selectively block attack traffic from each individual attack source. For example, a worm can infect millions of hosts [21] and use them as attack sources against a single victim. The victim's gateway cannot possibly filter millions of flows – today's most sophisticated routers can accommodate up to tens of thousands of filters [2].

On the other hand, a sophisticated attack involves a large number of routers, which can share the "filtering load" with the victim's gateway; and the closer we get to the attack sources, the larger the amount of filtering resources that are available per attack source. If the victim can identify the relay located closest to each attack source, the victim can request from that relay to block traffic from the corresponding attack source to the victim for a certain amount of time (and renew the request, if attack traffic recurs). This approach raises the following questions: (i) Can it be done securely, i.e., without enabling a malicious node to abuse the mechanism to disrupt other nodes' communications? (ii) Why would the domains hosting attack sources cooperate and help the victim?

A filter management scheme that addresses these two questions (called Active Internet Traffic Filtering or AITF) is presented in [6]: (i) A relay that receives a request to block certain traffic to an alleged victim verifies that the request is real by exchanging a three-way handshake with the victim's gateway. The handshake guarantees that malicious node M cannot disrupt the communication between nodes A and B, unless M is located *on the path* between A and B (which means that it can interfere with their communication anyway, e.g., by dropping their packets). (ii) If a relay does not cooperate and continues to forward attack traffic, the victim's gateway can selectively block all traffic from that non-cooperating relay to the victim.

This scheme "motivates" the domains that host attack sources to help block attack traffic, or else they lose their connectivity to the victim; this can be especially effective if the victim is a popular public-access site. Of course, in the unfortunate scenario that a significant portion of the Internet, including core and edge routers, has been compromised and is launching an attack, none of the relays close to the attack sources will help block attack traffic. In that case, the victim has to block all traffic from the compromised region. On one hand, this may cause lots of "good" traffic to be lost. On the other hand, given that the region's routers are compromised (and "good" traffic is at their mercy anyway), the Internet service provided by the compromised region is uncertain and can be regarded as having failed – so, blocking all its traffic may be the best reaction.

The probability of significant Internet portions (including routers) being compromised is, fortunately, low. In the typical DDoS attack scenarios, it is a large set of compromised end-hosts that send attack traffic. In these cases, independently of which specific filter management scheme is used, knowing the domain-level path of attack traffic enables the victim to push filtering close to the attack sources, by contacting their gateways directly, i.e., skipping over the Internet core. This approach is scalable, in the sense that it relieves the core routers from managing and propagating filters from an arbitrary number of victims to an arbitrary number of attack sources.

4. THE ALTERNATIVES TO LSRR

4.1 Transmit Policies with Labels

Our transmit policy mechanism is arguably burdensome: the edge system must know the entire domain-level path to be followed by its outgoing traffic. The alternative is for the edge system to tag each packet with a *policy label* that indicates how the packet should be routed and let each domain map this policy label to an appropriate next hop domain – the classic examples being DiffServ and MPLS.

The problem with policy labels is that each ISP only has visibility into the performance of its own private network. So, if a packet's policy label specifies, for example, "low-delay delivery", the first ISP may know how to select a path in its network with this property, but not whether the path from the ISP's exit point to the destination is an overall low-delay path, i.e., the first ISP may make a locally optimal, but globally pessimal choice. However, any designation that some packet gets lower delay forwarding than others requires some economic incentive to drive it, plus some means to ensure end-to-end low-delay delivery. Simply stated, there is no reason the ISP should favor a particular packet just because a client requests it, unless that client is paying more; and there is no reason for a client to pay more if the end-to-end performance is not improved.

To improve end-to-end performance, the ISP must make a globally good choice in forwarding the packet. To do this, the ISP needs to maintain policy state[4] that maps the "low-delay delivery" label and the specific destination to a spe-

[4] "Label Information Base" in MPLS

cific next-hop ISP. Not only must ISPs collect such state, but also constantly verify it – in an untrusted environment, where every ISP could potentially be misconfigured or compromised, it is dangerous to blindly trust an advertisement (e.g., for a low-delay path) without verifying it through monitoring/probing. Consequently, each ISP must participate in multiple network-layer "connections", which monitor the performance of entire paths and dictate to the ISP the local choices that result in the desired end-to-end performance.

We use "network-layer connection" to refer to end-to-end state maintained in all the domains across an Internet path, whether this state is initiated through explicit connection setup messages – e.g., using the Resource Reservation Protocol (RSVP) [25] – or dynamically created through measurements. Conversely, our LSRR approach can be viewed as a way to move transmit policy state from the network to the edges, maintaining the end-to-end nature of the Internet architecture.

4.2 Receive Policies via Hop-by-Hop Traceback

Our receive policy mechanism appears equally burdensome: the edge system must process the entire domain-level path followed by its incoming traffic, identify the relays closest to the attack sources, and request that they block attack traffic. As with transmit policies, the alternative is to delegate this responsibility to the network: a DDoS attack victim identifies the most congested links and requests from the corresponding routers to rate-limit attack traffic; the last-hop routers propagate similar requests further upstream. In this way, the network gradually pushes filtering of attack traffic close to its sources. A scheme that follows this approach (called Pushback) is presented in [15].

This alternative can indeed help mitigate DDoS attacks and is, perhaps, the best we can do in the current Internet architecture. However, it implicates Internet core routers in every end-to-end filter propagation. As a result, the Internet core becomes a potential filtering bottleneck.

To avoid overloading the Internet core, the solution is coarser-grained filtering, i.e., rate-limit *aggregates* of attack flows instead of blocking each attack source individually. For example, Pushback propagates only one filter per victim, i.e., each router is requested to rate-limit *all* traffic to each victim. However, if attack traffic volume exceeds "good" traffic volume (which is the case in DDoS attacks), rate-limiting all traffic to the victim results in dropping most of its "good" traffic as well. As a result, filtering becomes selective only when it gets pushed to the routers close to the attack sources.

Conversely, our LSRR approach enables selective filtering of attack traffic upfront, before filtering is pushed close to the attack sources. Moreover, it enables the victim to identify the relays close to the attack sources and contact them directly, bypassing the Internet core. In this sense, our approach can be viewed as a way to move receive policy state from the network to the edges.

5. RELATED WORK

We organize related work into three categories. The first one references architectures that enable edge systems to control the path of their outgoing traffic. The second one references traceback and packet marking schemes, which enable edge systems to identify the paths of their incoming traffic. The third category references addressing protocols that, like WRAP, add path information inside each packet (but do so with different objectives). Finally, we should note that WRAP first appeared as part of the TRIAD architecture [9], where it served different objectives than presented in this paper.

5.1 Route Control

RouteScience Inc. [1] follows an approach similar to ours, in that it involves entities located at the edges of the Internet, which monitor and evaluate multiple paths to each potential destination. Their approach differs from ours, in that a sender does not control the entire path, only the first ISP. Controlling the first ISP can sometimes help improve communication quality and requires no changes to the current Internet architecture. However, it does not always enable a sender to route around failing or undesired regions of the Internet, even if an appropriate route exists.

The Resilient Overlay Networks (RON) architecture [5] is related to WRAP in a similar way: RON nodes constantly monitor and evaluate the paths that interconnect them; when the path between two nodes fails, the nodes resume contact by establishing indirect communication through other overlay nodes. WRAP provides higher visibility and more control to the edge system, but RON was designed to work with the limited route control provided by the current Internet architecture.

The New Internet Routing Architecture (NIRA) [24] provides similar route control with WRAP, namely enables an edge system to specify the entire domain-level path of its outgoing traffic. However, NIRA forces provider-rooted hierarchical addressing: each top-level ISP is allocated a globally unique prefix, which it uses to recursively sub-allocate prefixes to its customer domains. Compared to this hierarchical addressing, WRAP facilitates incremental deployment, by using conventional IP addresses.

5.2 Traceback and Packet Marking

An IP traceback mechanism enables an edge system to identify the path followed by its incoming traffic. Most mechanisms do that by requiring routers to mark packets; the receiving edge system can then process/combine the marks and reconstruct the path [20, 10]. Packet marking can also be used to provide each packet with a "fingerprint" that depends on the path followed by the packet. This kind of packet marking may not necessarily enable traceback, but it does enable filtering of packets based on their path [23].

The current Internet architecture does not provide room for packet marking. As a result, traceback and packet marking research has focused on inventing intelligent marking algorithms, which fit the full path information in lightly utilized IP header fields. Our approach is simpler: First, we argue that the receiver does not need to know the entire set of routers that forwarded a packet; knowing the domain-level path is enough, both to trace back and to filter. Second, we argue that all the research effort invested in packet marking signals the importance of route recording and suggests that the next-generation Internet architecture should directly provide this capability.

5.3 Addressing Protocols

WRAP is similar to the IP-next-layer (IPNL) [12] and IPv4+4 [22] addressing protocols, in that it (i) is a "shim" protocol between the IP and transport layers, (ii) involves

an overlay of upgraded routers that relay packets to each other, and (iii) specifies inside each packet's header the set of such routers on the packet's path. However, WRAP-enabled routers map to border routers between administrative domains. As a result, and unlike IPNL and IPv4+4, WRAP enables an edge system to control the full domain-level path of its outgoing and incoming traffic.

6. CONCLUSIONS

We described a mechanism that provides traffic policy support for the next-generation Internet. Our mechanism enables an edge system to control the domain-level path followed by its traffic, by adding state in each packet. This "datagram" approach provides all the benefits of datagrams over virtual circuits, while incurring relatively modest forwarding complication or packet header overhead (the typical costs with datagrams). In contrast, previous approaches to traffic policy require some form of network-layer virtual circuits, by adding policy state in the routers.

We presented WRAP, an addressing protocol that specifies the domain-level path followed by a packet inside the packet's headers. We compared WRAP to the traditional Loose Source Record Route (LSRR) IPv4 option and showed that WRAP provides similar functionality, while avoiding the processing overhead and security problems introduced by traditional LSRR.

We described how WRAP can be used by edge systems to specify both transmit and receive policies: first, we presented elements of an inter-domain routing scheme (Feedback Based Routing), which enables edge systems to choose good paths for their outgoing traffic; second, we drew the outline of a filter management scheme (Active Internet Traffic Filtering), which enables edge systems to mitigate highly distributed DDoS attacks. We did not prove that these two schemes are scalable, but we consider [26] and [6] to make a strong case for their scalability.

Finally, we compared our approach to alternative mechanisms, which place policy state in the routers. Relative to these mechanisms, our approach moves policy state to the edges, avoiding a "connection-oriented" network layer. Therefore, we expect our approach to avoid the scalability problems associated with network-layer virtual circuits.

7. ACKNOWLEDGMENTS

We would like to thank TJ Giuli, Athina Markopoulou, George Candea, and the FDNA reviewers for their constructive feedback.

APPENDIX

A. THE WRAP PROTOCOL

A.1 Header

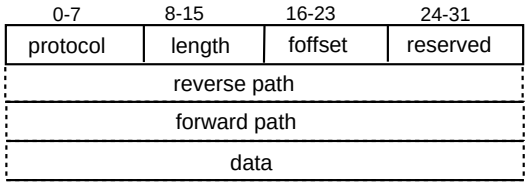

Figure 1: WRAP Header

The format of the WRAP header appears in Figure 1. The *protocol*, *length*, *foffset* and *reserved* fields are each 8 bits long. The *reverse path* and *forward path* fields are variable-length lists of 32-bit IP addresses.

- The *protocol* field specifies the higher layer protocol, e.g. TCP, UDP etc.
- The *length* field is the number of 32-bit addresses in the *reverse path* and *forward path* fields.
- The *foffset* field is the offset into the list of 32-bit addresses, where the *forward path* field starts.
- The *reverse path* field carries the addresses of the end-point and relays already traversed.
- The *forward path* field carries the addresses of the relays and end-point still ahead.
- The *data* field contains a TCP, UDP or other transport protocol packet.

A.2 Relaying

The relaying algorithm is best explained through an example. In Figure 2, host S sends a packet to host D. The packet crosses three realms before reaching its destination. The figure includes part of the IP and WRAP header of the packet as it travels in each of the three realms.

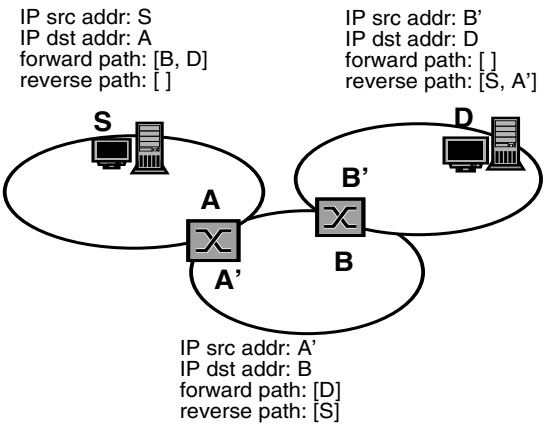

Figure 2: Relaying Example

When S sends a packet, it sets the IP source address to its own address and the IP destination address to the address of the first relay. The forward path includes the address of the second relay and the address of host D. The reverse path is empty.

Relay A rearranges the IP and WRAP headers, setting the IP source address to its own address in the intermediate realm and the IP destination address to the address of the next relay. Similarly, relay B rearranges the headers, setting the IP source address to its own address in the destination realm and the IP destination address to the address of host D.

Host D knows that the received packet is destined to it, because the forward path is empty.

A.3 Limited Path Spoofing

In the IP world, if a sender puts the wrong destination address in the IP header, the packet never reaches its destination. Similarly, in the WRAP world, if a sender puts a fake forward path in the WRAP header, the packet never reaches its destination. Thus, a sender is *forced* to specify the correct forward path in the WRAP header. For example, in Figure 2, host S must specify IP destination address A and forward path [B, D], otherwise the packet does not reach host D.

A malicious sender is free to fake the source address in the IP header and the reverse path in the WRAP header. I.e., a malicious sender can pretend to be either another host in the same realm or a host in a realm further away from the receiver than she really is. For example, in Figure 3, S can spoof the identity of host V, specifying IP source address R' and reverse path [V].

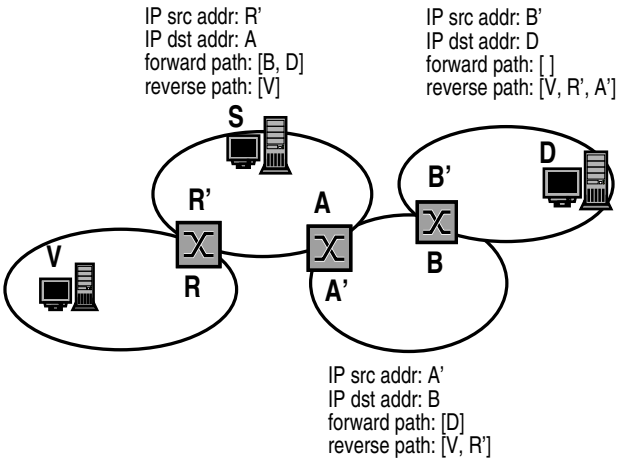

Figure 3: Reverse Path Spoofing Example

However, this sort of spoofing does not prevent the receiver from identifying the "gateways" of spoofed traffic. In Figure 3, S can spoof a large number of addresses and reverse paths and launch a DDoS attack against D. D can still inspect the reverse paths specified in the incoming traffic and identify relay A' as the "gateway" of attack traffic.

A.4 Address Space Flexibility

It is possible for each realm to have an independent address space, thus allowing two nodes to have the same IP address as long as they are located in different realms. In this way, the number of available addresses per administrative domain expands to approximately 4.5 billion.

In this scenario, a relay address specifies not only a specific router interface, but also a specific outgoing realm. For example, in Figure 2, when host S sends out a packet addressed to A, A specifies both the corresponding router interface and the realm where the next relay, B, is located. This is necessary, because B is not a globally unique address, so a relay cannot process B and deduce which realm it refers to.

A.5 Low Packet Overhead

In their study of Autonomous System (AS) topology [14], Magoni and Pansiot have plotted the distribution of AS pairs by shortest path length. According to their measurements, the mean AS path distance is between 3 and 4 and, as the authors themselves comment, the distribution looks Gaussian. Around the mean, the distribution drops pretty quickly: about 75% of AS pairs have a shortest path length less than 4 and about 95% of them have a shortest path length less than 6.

This data gives us a rough estimate of how the WRAP header length histogram would look like. Mapping ASs to realms, the reverse and forward path fields inside a WRAP packet would have together as many addresses as the number of ASs in the path connecting the end-points. However, AS topology studies do not take into account Network Address Translation (NAT) boxes [11]. In order for our overhead estimation to be conservative, we increment all AS shortest paths by two. This accounts for each AS being a collection of networks placed behind NAT boxes. Therefore, 75% of paths would require $4 + 2 = 6$ addresses to be specified and, thus, they would need a WRAP header of at most $4 + 4 * 6 = 28$ bytes. About 95% of them would require $6 + 2 = 8$ addresses to be specified and, thus, they would need a WRAP header of at most $4 + 4 * 8 = 36$ bytes.

In the same study, the authors state an empirical law: *Currently, the average distance, the diameter and the radius of the inter-domain graph of AS network stay constant.* It is worth noting that, according to their study, from November 1997 to May 2000 the number of ASs grew by 40% while the average distance, the diameter and the radius remained constant. As long as this empirical law holds, packet overhead introduced by WRAP does not increase with Internet size.

A.6 Incremental Deployment

Deploying WRAP in the current Internet would be similar to placing every administrative domain behind NAT boxes. I.e., each candidate realm would have to (i) upgrade its border gateways to support WRAP relaying, and (ii) potentially upgrade its end-hosts to support WRAP transmission and reception. However, it is not necessary for all end-hosts in a WRAP realm to be WRAP-aware. It is possible to add "translating" functionality to relays, so that they can convert IP packets to IP/WRAP packets and vice-versa. We call a relay with such functionality a WRAPID (WRAP to IP Domain) gateway.

A WRAPID gateway intercepts outgoing/incoming packets from/to WRAP-unaware hosts and converts them into the format expected by the recipient. Therefore, a WRAPID gateway maintains per-connection state, in the same way

that a NAT box does. This is best illustrated through an example: In Figure 4, WRAP-unaware host S sends a packet to WRAP-unaware host D. The packet crosses three realms before reaching its destination. The figure includes part of the IP and WRAP header of the packet as it travels in each of the three realms.

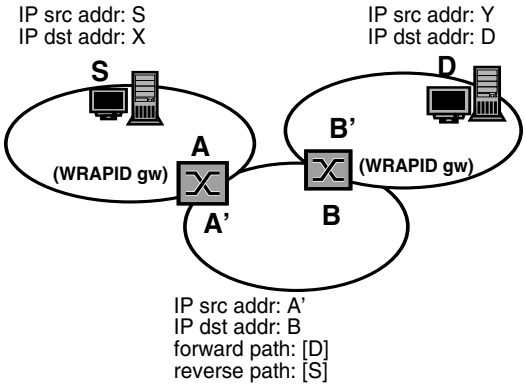

Figure 4: WRAPID Translation Example

When S sends the packet, it sets the IP destination address to "special" address X. WRAPID gateway A interprets X and maps it to IP destination address B and forward path $[D]$. WRAPID gateway B strips the packet of its WRAP header and sets the IP source address to "special" address Y. This scenario requires gateway A to already have a mapping

$$\text{IP dst addr} = X \rightarrow \{\text{IP dst addr} = B, \text{forward path} = [D]\}$$

and gateway B to set up a mapping

$$\{\text{IP src addr} = A', \text{reverse path} = [S]\} \rightarrow \text{IP src addr} = Y$$

The question of how gateway A obtains the necessary mapping in the first place is addressed in the next section.

B. NAME-TO-PATH RESOLUTION

WRAP does require an extension to the traditional Domain Name Service (DNS) architecture [19]: current DNS servers provide mappings from domain names to IP addresses; WRAP requires mappings from domain names to domain-level paths. A scheme that can easily support name-to-path resolution is presented in [13]. This section outlines a simplified version, just to convince the reader that such a scheme is reasonable to design and deploy.

Consider that each realm has an internal and an external DNS server. The internal server answers requests coming from the realm's hosts and the external server answers requests from outside. The internal server provides mappings from domain names to WRAP paths. The external server provides mappings from domain names to {global prefix, IP address} pairs. We illustrate how name-to-path resolution works with an example.

In Figure 2, routers A and B play the role of WRAP relays as well as internal and external DNS servers. Note that the following steps would be the same if A and B were many realms away from each other.

1. Host S does a name lookup for host D's name.
2. Relay/DNS server A propagates a request to relay/DNS server B.
3. B responds with {prefix = P, IP = D}. As described in Section 3.1, A knows the domain-level path to any destination prefix.
4. A responds to S with the path $[A,B,D]$.

Of course, DNS servers do not necessarily have to be collocated with relays. However, they do need to communicate, so implementing them on the same box may be the simplest solution.

Concerning WRAPID gateways, the necessary state can be set up during name resolution. For example, in Figure 4, when host S does a name lookup for host D's name, WRAPID gateway/DNS server A sets up the mapping

$$\text{IP dst addr} = X \rightarrow \{\text{IP dst addr} = B, \text{forward path} = [D]\}$$

and responds to S with IP address X.

C. REFERENCES

[1] RouteScience. `http://www.routescience.com`.

[2] SiberCore SCT1842 features. `http://www.sibercore.com/products_siberCAM.htm#3`.

[3] Troubleshooting High CPU Utilization in Cisco 7500 Routers. `http://www.cisco.com/en/US/products/hw/routers/ps359/products_tech_note%09186a00801c2af3.shtml`.

[4] RFC 791 - Internet Protocol, September 1981.

[5] David G. Andersen, Hari Balakrishnan, M. Frans Kaashoek, and Robert Morris. Resilient Overlay Networks. In *Proceedings of the ACM Symposium on Operating Systems and Principles (SOSP)*, Banff, Canada, October 2001.

[6] Katerina Argyraki and David R. Cheriton. Protecting Public-Access Sites Against DDoS Attacks. Technical report, Stanford University, May 2004. Available at `http://arxiv.org/cs.NI/0403042`.

[7] F. Baker. RFC 1812 - Requirements for IP Version 4 Routers, June 1995.

[8] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. RFC 2475 - An Architecture for Differentiated Services, December 1998.

[9] David R. Cheriton and Mark Gritter. TRIAD: A Scalable Deployable NAT-based Internet Architecture. Technical report, Stanford University, January 2000.

[10] Drew Dean, Matt Franklin, and Adam Stubblefield. An Algebraic Approach to IP Traceback. *Information and System Security*, 5(2):119–137, 2001.

[11] K. Egevang and P. Francis. RFC 1631 - The IP Network Address Translator, May 1994.

[12] Paul Francis and Ramakrishna Gummadi. IPNL: A NAT-Extended Internet Architecture. In *Proceedings of the ACM SIGCOMM Conference*, San Diego, California, USA, August 2001.

[13] Mark Gritter and David R. Cheriton. An Architecture for Content Routing Support in the Internet. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems (USITS)*, San Francisco, California, USA, March 2001.

[14] Damien Magoni and Jean Jacques Pansiot. Analysis of the Autonomous System Network Topology. *ACM*

SIGCOMM Computer Communication Review, 31(3):26–37, July 2001.

[15] R. Mahajan, S. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker. Controlling High Bandwidth Aggregates in the Network. `http://www.icir.org/pushback/`, July 2001.

[16] Athina Markopoulou, Gianluca Iannaccone, Supratik Bhattacharyya, Chen-Nee Chuah, and Christophe Diot. Characterization of Failures in an IP Backbone Network. In *Proceedings of the IEEE INFOCOM Conference*, Hong Kong, March 2004.

[17] Athina P. Markopoulou, Fouad A. Tobagi, and Mansour J. Karam. Assessing the Quality of Voice Communication over Internet Backbones. *IEEE Transactions on Networking*, 11(5):747–760, October 2003.

[18] Cristopher Metz. Ingredients for Better Routing? Read the Label. *IEEE Internet Computing*, 2(5):10–15, September/October 1998.

[19] P. Mockapetris. RFC 1035 - Domain Names - Implementation and Specification, November 1987.

[20] Stefan Savage, David Wetherall, Anna Karlin, and Tom Anderson. Practical Network Support for IP Traceback. In *Proceedings of the ACM SIGCOMM Conference*, Cambridge, Massachusetts, USA, August 2000.

[21] Stuart Staniford, Vern Paxson, and Nicholas Weaver. How to 0wn the Internet in Your Spare Time. In *Proceedings of the USENIX Security Symposium*, San Francisco, California, USA, August 2002.

[22] Z. Turanyi and A. Valko. IPv4+4. In *Proceedings of the IEEE International Conference on Network Protocols (ICNP)*, Paris, France, November 2002.

[23] Abraham Yaar, Adrian Perrig, and Dawn Song. Pi: A Path Identification Mechanism to Defend against DDoS Attacks. In *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, California, USA, May 2003.

[24] Xiaowei Yang. NIRA: A New Internet Routing Architecture. In *Proceedings of the ACM SIGCOMM Workshop on Future Directions in Network Architecture*, Karlsruhe, Germany, August 2003.

[25] Lixia Zhang, Steve Deering, Deborah Estrin, Scott Shenker, and Daniel Zappala. RSVP: A New Resource ReSerVation Protocol. *IEEE Network Magazine*, 7(5):8–18, September 1993.

[26] Dapeng Zhu, Mark Gritter, and David R. Cheriton. Feedback Based Routing. `http://www.stanford.edu/~dapengz/papers/zhu-FBR-SigComm03.pdf`, February 2003.