Towards a Global IP Anycast Service

Hitesh Ballani
Cornell University
Ithaca, NY
hitesh@cs.cornell.edu

Paul Francis
Cornell University
Ithaca, NY
francis@cs.cornell.edu

ABSTRACT

IP anycast, with its innate ability to find nearby resources in a robust and efficient fashion, has long been considered an important means of service discovery. The growth of P2P applications presents appealing new uses for IP anycast. Unfortunately, IP anycast suffers from serious problems: it is very hard to deploy globally, it scales poorly by the number of anycast groups, and it lacks important features like load-balancing. As a result, its use is limited to a few critical infrastructure services such as DNS root servers. The primary contribution of this paper is a new IP anycast architecture, PIAS, that overcomes these problems while largely maintaining the strengths of IP anycast. PIAS makes use of a proxy overlay that advertises IP anycast addresses on behalf of group members and tunnels anycast packets to those members. The paper presents a detailed design of PIAS and evaluates its scalability and efficiency through simulation. We also present preliminary measurement results on anycasted DNS root servers that suggest that IP anycast provides good affinity. Finally, we describe how PIAS supports two important P2P and overlay applications.

Categories and Subject Descriptors: C.2.1 [Network Architecture and Design]: Network communications

General Terms: Design, Measurement.

Keywords: Anycast, Proxy, Overlay, Routing, Architecture.

1. INTRODUCTION

Ever since it was proposed in 1993, IP anycast[1][1] has been viewed as a powerful IP packet addressing and delivery mode. Because IP anycast typically routes packets to the nearest of a group of hosts, it has been seen as a way to obtain efficient, transparent and robust ***service discovery***. In cases where the service itself is a connectionless ***query/reply service***, IP anycast supports the complete service, not just discovery of the service. The best working example of the latter is the use of IP anycast to replicate root DNS servers [2][3] without modifying DNS clients. Other proposed uses include ***host auto-configuration***[1] and using anycast to reach a ***routing substrate***, such as rendezvous points for a multicast tree[4][5] or a IPv6 to IPv4 (6to4) transition device[6].

In spite of its benefits, there has been very little IP anycast deployment to date, especially on a global scale. The only global scale use of IP anycast in a production environment that we are aware of is the anycasting of DNS root servers and AS-112 servers[7][2].

The reason for this is that IP anycast has serious limitations. Foremost among these is IP anycast's poor scalability. As with IP multicast, routes for IP anycast groups cannot be aggregated—the routing infrastructure must support one route per IP anycast group. It is also very hard to deploy IP anycast globally. The network administrator must obtain an address block of adequate size (i.e. a /24), and arrange to advertise it into the BGP substrate of its upstream ISPs. Finally, the use of IP routing as the host selection mechanism means that important selection metrics such as server load cannot be used. It is important to note that while IPv6 has defined anycast as part of its addressing architecture[8], it is also afflicted by the same set of problems.

By contrast, ***application layer anycast*** provides a one-to-any service by mapping a higher-level name, such as a DNS name, into one of a group of hosts, and then informing the client of the selected host's IP address, for instance through DNS or some redirect mechanism. This approach is much easier to deploy globally, and is in some ways superior in functionality to IP anycast. For example, the fine grained control over the load across group members and the ability to incorporate other selection criteria makes DNS-based anycast the method of choice for Content Distribution Networks (CDNs) today.

In spite of this, we believe that IP anycast has compelling advantages, and its appeal increases as overlay and P2P applications increase. First, IP anycast operates at a low level. This makes it potentially useable by, and transparent to, any application that runs over IP. It also makes IP anycast the only form of anycast suitable for low-level protocols, such as DNS. Second, it automatically discovers nearby resources, eliminating the need for complex proximity discovery mechanisms [9]. Finally, packets are delivered directly to the target destination without the need for a redirect (frequently re-

[1] IP anycast is an IP addressing and delivery mode whereby an IP packet is sent to one of a group of hosts identified by the IP anycast address. Where IP unicast is one-to-one, and IP multicast is one-to-many, IP anycast is one-to-any.

[2] anycasted servers that answer PTR queries for the RFC 1918 private addresses

SIGCOMM'05, Aug 22–26, 2005, Philadelphia, Pennsylvania, USA.
Copyright 2005 ACM 1-59593-009-4/05/0008 ...$5.00.

quired by application-layer anycast approaches). This saves at least one packet round trip, which can be important for short lived exchanges. It is these advantages that have led to increased use of IP anycast within the operational community, both for providing useful services (DNS root servers), and increasingly for protecting services from unwanted packets (AS112 and DDoS sinkholes [10]).

The *primary contribution* of this paper is the detailed description of a deployment architecture for an IP anycast service that overcomes the limitations of today's "native" IP anycast while adding new features, some typically associated with application-level anycast, and some completely new. This architecture, called **PIAS (Proxy IP Anycast Service)**, is composed as an overlay, and utilizes but does not impact the IP routing infrastructure. The fact that PIAS is an IP anycast service means that *clients* use the service completely transparently—that is, with their existing IP stacks and applications.

PIAS allows an endhost in an anycast group (anycast group member, or **anycast target**) to receive anycast packets for that group via its normal unicast address (and normal protocol stack). The anycast target *joins* the anycast group simply by transmitting a request packet to an anycast address (again, via its unicast interface). The target may likewise *leave* the group through a request packet, or by simply becoming silent.

PIAS utilizes the IP address space efficiently: thousands of IP anycast groups may be identified through a single IP address. It scales well by the number of groups, group size and group churn with virtually no impact on the IP routing infrastructure. It provides fast failover in response to failures of both target hosts and PIAS infrastructure nodes.

PIAS can select targets based on criteria other than proximity to the sending host, notably including the ability to load balance among targets. PIAS has the unique feature that an anycast group member can also transmit packets to other members of the same anycast group. This is in contrast to native IP anycast, where a group member would receive its own packet if it transmitted to the group. This feature makes IP anycast available to P2P applications, something not possible if a host can't both send to and receive from the anycast group.

The remainder of the paper is organized as follows: Section 2 identifies the features of an ideal anycast service. Section 3 spells out the system design together with the goals satisfied by each design feature. Section 4 presents simulations and measurements meant to evaluate various features of the PIAS design. Section 5 discusses related work and section 6 describes a few applications made possible by PIAS. Section 7 discusses other important goals that PIAS must fulfill and section 8 presents our conclusions.

2. DESIGN GOALS

This section specifically lays out the design goals of PIAS, and briefly comments on how well PIAS meets those goals. The subsequent design description section refers back to these goals as needed. The goals are listed here in two parts. The first part lists those goals that are accomplished by native IP anycast, and that we wish to retain. The second part lists those goals that are not accomplished by native IP anycast. In this way, we effectively highlight the weaknesses of IP anycast, and the contributions of PIAS.

1. *Backwards Compatible*: Native IP anycast is completely transparent to clients and routers , and we believe that this transparency is critical to the success of a new IP anycast service. Because PIAS is an overlay technology that uses native IP anycast, it does not change clients and routers.
2. *Scale by group size*: By virtue of being totally distributed among routers, native IP anycast scales well by group size. PIAS has no inherent group size limitation. PIAS is deployed as an overlay infrastructure, and can scale arbitrarily according to the size of that infrastructure.
3. *Efficient packet transfer*: Because native IP anycast uses IP routing, its paths are naturally efficient. As an overlay, PIAS imposes some stretch penalty on the paths packets take. The penalty imposed by PIAS is small (section 4.3), and shrinks as the PIAS infrastructure grows.
4. *Robustness*: Native IP anycast's robustness properties (including packet loss) are similar to IP unicast. PIAS is engineered to be similarly robust.
5. *Fast failover*: Failover speed in Native IP anycast depends on the convergence speed of the underlying routing algorithms, and can be fast (OSPF) or somewhat slow (BGP). PIAS can be engineered to almost always rely on OSPF for certain types of failover (section 3.6). The PIAS overlay exposes additional failover situations that go beyond IP routing, and these are handled accordingly (Section 3.6).

The following are the goals that native IP anycast does not satisfy.

6. *Ease of* joining *and* leaving: Target hosts must not have to interact with IP routing to join and leave.
7. *Scale by the number of groups*: In addition to scaling by the usual metrics of memory and bandwidth, we require that PIAS also make efficient use of the IP address space. PIAS is able to accommodate thousands of groups within a single address by incorporating TCP and UDP port numbers as part of the group address.
8. *Scale by group dynamics*: Globally, IP routing behaves very badly when routes are frequently added and withdrawn. The PIAS overlay hides member dynamics from IP routing, and can handle dynamics caused both by continuous member churn and flash crowds (including those caused by DDoS attacks).
9. *Target Selection criteria*: IP anycast can only select targets based on proximity. At a minimum, we wish to add load and connection affinity as criteria.

3. DESIGN DESCRIPTION

This section gives a detailed description of PIAS. We take a "layered" approach to the description—we start with the core concepts and basic design and then step-by-step describe additional functionality that satisfies specific goals listed in section 2.

PIAS is deployed as an overlay infrastructure. It may be deployed by a CDN company like Akamai, by multiple cooperating ISPs, or even by a single ISP (though the efficacy of proximity discovery would be limited by the ISP's geographic coverage). Multiple distinct PIAS infrastructures may be deployed. In this case, each operates using distinct blocks of IP

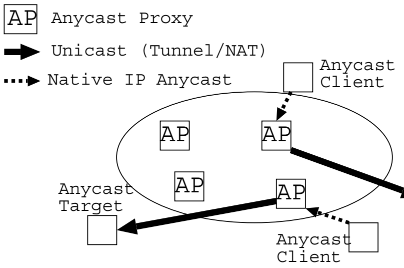


Figure 1: Proxy Architecture: the client packets reaching the proxies through native IP anycast are tunnelled to the targets

anycast addresses, and they do not interact with each other[3]. In the remainder of this document, for simplicity of exposition, we assume a single PIAS infrastructure.

The basic idea of PIAS, illustrated in Figure 1, is very simple. Router-like boxes, hereon referred to as ***anycast proxies*** (AP or simply *proxies*), are deployed at various locations in the Internet, for example at POPs (Point of Presence) of different ISPs. These proxies advertise the same block of IP addresses, referred to as the ***anycast prefix***, into the routing fabric (BGP, IGP). As such, the proxies are reachable by native IP anycast—a packet transmitted to the anycast prefix will reach the closest proxy. However, these proxies are not the actual ***anycast target destinations***(AT)[4]. Rather, true to their name, they proxy packets that reach them via native IP anycast to the true target destinations using unicast IP. This proxying can take the form of lightweight tunnels or NAT. NAT allows for backwards compatibility with the protocol stack at target hosts, but increases processing at the proxy.

This novel combination of native IP anycast with tunnelling to the unicast addresses of the targets allows PIAS to fulfill three critical design goals and drives the rest of the system design. First, it allows for efficient use of the address space as all the IP addresses in the prefix advertised by the proxies can be used by different anycast groups. In fact, PIAS does one better. It identifies an anycast group by the full ***transport address*** (TA), i.e. IP address and TCP/UDP port, thus allowing thousands of anycast groups per IP address. Second, it solves the IP routing scaling problem by allowing many anycast groups to share a single address prefix and hence, fulfills goal 7. Finally, it relieves targets from the burden of interacting with the routing substrate. They can join an anycast group by registering with a nearby proxy that is discovered using native IP anycast. This fulfills goal 6.

The reader may notice two suspicious claims in the last paragraph. First, we claim to ease deployment by running unicast at the target instead of anycast, and yet the proxies still must run anycast. So, how is this an improvement? The benefit is that the difficult work of deploying IP anycast is borne by the anycast provider once, and amortized across many anycast groups. Second, we claim to improve scaling by allowing thousands of IP anycast groups to share a single IP address prefix. All we've really done, however, is to move the scaling problem from the IP routing domain to the PIAS infrastructure domain. This is quite intentional. As we argue later on, the scaling issues are much easier to deal with in the overlay than in IP routing.

PIAS offers two primitives to the members of an anycast group, which involve sending messages to a nearby proxy:

- *join*(IP_A:$port_A$,IP_T:$port_T$,*options*): this message instructs the proxy to forward packets addressed to the anycast group identified by the TA IP_A:$port_A$ to the joining node's unicast TA IP_T:$port_T$. The *options* may specify additional information such as the selection criteria (load balance etc.), delivery semantics (scoping etc.), or security parameters needed to authenticate the target host. These are discussed later.
- *leave*(IP_A:$port_A$,IP_T:$port_T$,*options*): this message informs the proxy that the target identified by TA IP_T:$port_T$ has left the group IP_A:$port_A$. *options* are the security parameters.

The join and leave messages are transmitted to the anycast address IP_A(that belongs to the anycast prefix) at some well-known port that is dedicated to receiving registration messages. This means that no extra configuration is required for a target to discover a nearby proxy.

Note that we don't specify a "create group" primitive. For the purpose of this paper, we assume that the first join essentially results in the creation of the group. In practice, a subscriber to the service would presumably have entered into a contract with the anycast service provider, which would have resulted in the assignment of anycast TAs to that subscriber. The subscriber would also have obtained authentication information using which targets may join the group. While the issues surrounding this sort of group creation are important, they are not central to the PIAS architecture, and we don't discuss them further.

3.1 The Join Anycast Proxy (JAP)

A target may leave a group either through the leave primitive, or by simply falling silent (for instance, because the target is abruptly shut off or loses its attachment to the Internet). This means that the ***Join AP*** (JAP—the nearby proxy with which the target registers; shown in figure 2) must monitor the health of its targets, determine when they are no longer available, and treat them as having left the group. The proximity of the JAP to the target makes it ideal for this.

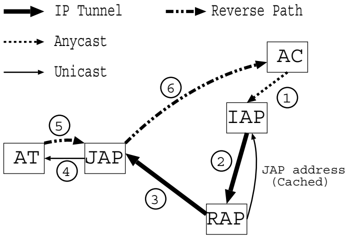
The JAP must also inform zero or more other anycast proxies (APs) of the target(s) that have registered with it. This is because not all APs may be JAPs for a given group (that is, no target joined through them), but **anycast clients** (ACs) may nevertheless send them packets destined for the group. A proxy that receives packets directly from a client is referred to as the ***Ingress AP*** (IAP)[5] for the client. Note that the client-IAP relation is established using native IP anycast. As an IAP, the proxy must know how to forward packets towards a target; even though the IAP may not explicitly know of the target.

One possible way to achieve this would have the JAP spread information about targets associated with it to all proxies. This allows the IAP to tunnel packets directly to clients (as in Figure 1). However, such an approach would hamper PIAS's ability to support a large number of groups. In fact, Figure 1 is conceptual—PIAS's approach for spreading group information is described in the next section and the actual paths taken by packets are shown in Figure 2.

[3] Indeed, a single operator could deploy multiple distinct PIAS infrastructures as a way to scale.

[4] the members of the anycast group; hereon referred to as **anycast targets** or simply **targets**

[5] in figure 1 the proxies in the client-target path are IAPs



No.	Source	Dest	Comment
1	AC:p	AA:g	Native IP Anycast
2	IAP:AC:p	RAP:AA:g	IP-IP tunnel
3	RAP:AC:p	JAP:AA:g	IP-IP tunnel
4	JAP:q	AT:r	Unicast IP JAP:q = NAT(AC:p)
5	AT:r	JAP:q	Unicast IP
6	AA:g	AC:p	Unicast IP AC:p = NAT^{-1}(JAP:q)
7	IAP:AC:p	JAP:AA:g	IP-IP tunnel

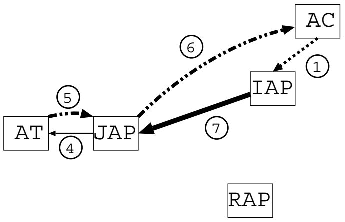


Figure 2: Initial (left) and subsequent (right) packet path. The table shows the various packet headers. Symbols in block letters represent IP addresses, small letters represent ports. AA(Anycast Address) is one address in the address block being advertised by PIAS, AA:g is the transport address assigned to the group the target belongs to, while AT:r is the transport address at which the target wants to accept packets. Here, the target joined the group by invoking join(AA:g,AT:r,options)

3.2 Scale by the number of groups

In the previous section, we mentioned the need for a scheme that would allow PIAS to manage group membership information while scaling to a large number of groups. For any given group, we designate a small number of APs (three or four) to maintain a list of JAPs for the group. When acting in this role, we call the AP a ***Rendezvous Anycast Proxy*** (RAP). All APs can act as RAPs (as well as as JAPs and IAPs).

The RAPs associated with any given group are selected with a consistent hash [11] executed over all APs. This suggests that each proxy know all other proxies, and maintain their current up/down status. This is possible, however, because we can assume a relatively small number of global APs ($\leq 20,000$, a number we derive later). We also assume that, like infrastructure routers, APs are stable and rarely crash or are taken out of service. The APs can maintain each other's up/down status through flooding, gossip [12] or a hierarchical structure [13]. The current implementation uses flooding. Such an arrangement establishes a simple one-hop DHT and hence, limits the latency overhead of routing through the proxy overlay.

When a proxy becomes a JAP for the group (i.e. a target of the group registers with it), it uses consistent hashing to determine all the RAPs for the group and informs them of the join. This allows the RAP to build a table of JAPs for the group.

The concept of the RAP leads to a packet path as shown on the left side of Figure 2. When an IAP receives a packet for an anycast group that it knows nothing about, it hashes the group TA, selects the nearest RAP for the group, and transmits the packet to the RAP (path segment 2). The RAP receives the packet and selects a JAP based on whatever selection criteria is used for the group. For instance, if the criteria is proximity, it selects a JAP close to the IAP. The RAP forwards the packet to the selected JAP (path segment 3), and at the same time informs the IAP of the JAP (the RAP sends a list of JAPs, for failover purposes).

The use of RAPs unfortunately introduces another overlay hop in the path from client to target. We mitigate this cost however by having the IAP cache information about JAPs. Once the IAP has cached this information, subsequent packets (not only of this connection, but of subsequent connections too) are transmitted directly to the JAP. This is shown in the right-hand side of Figure 2. The time-to-live on this cache entry can be quite large. This is because the cache entry can be actively invalidated in one of two ways. First, if the target leaves the JAP, the JAP can inform the IAP of this when a subsequent packet arrives. Second, if the JAP disappears altogether, inter-AP monitoring will inform all APs of this event. In both cases, the IAP(s) will remove the cached entries, failover to other JAPs it knows of, or failing this, go back to the RAP. Because of this cache invalidation approach, the IAP does not need to go back to the RAP very often.

Note that in figure 2, the JAP is responsible for transmitting packets to and receiving packets from its targets. The reasoning for this is not obvious and goes as follows. We aim to support legacy clients that expect to see return packets coming from the same address and port to which they sent packets. In general, targets cannot source packets from anycast addresses and so at least one proxy must be inserted into the target-client path. Furthermore, if NAT is being used to forward packets to the target, then the proxy with the NAT state should be the proxy that handles the return packets.

This might argue for traversing the IAP in the reverse direction too, since by necessity it must be traversed in the forward direction. The argument in favor of using the JAP however, boils down to the following two points. First, it is highly convenient to keep all target state in one proxy rather than two or more. Since the JAP in any event must monitor target health, it makes sense to put all target state in the JAP. Second, the JAP is close to the target, so the cost of traversing the JAP in terms of path length is minimal (Section 4.3). Also, by seeing packets pass in both directions, the JAP is better able to monitor the health of the target. For the most part, when a packet passes from client to target, the JAP may expect to soon see a packet in the reverse direction. Rather than force the JAP to continuously ping each target, the lack of a return packet can be used to trigger pings.

The use of proxies implies that the PIAS path (AC⇒IAP⇒JAP⇒AT) might be longer than the direct path (AC⇒AT)[6]. However, the proximity of the client to the IAP and of the target to the JAP should ensure that PIAS imposes minimal stretch and hence fulfills goal 3. This has been substantiated by simulating the stretch imposed by PIAS across a tier-1 topology map of the Internet.

The introduction of the RAP to allow scaling by the number of groups is somewhat equivalent to the extra round-trip imposed by application-level anycast schemes, for instance in the form of the DNS lookup or the HTTP redirect. This is

[6] the PIAS path may actually be shorter as inter-domain routing is not optimal[14]

one aspect of PIAS that falls short of native IP anycast, which has no such extra hop. Having said that, it would be possible for a small number of groups with minimal target churn to operate without RAPS—that is, to spread JAP information among all APs. This might be appropriate, for instance, for a CDN or for 6to4 gateways. By-and-large, however, we can expect most groups to operate with RAPs as described here, and in the remainder of the design section, we assume that is the case.

3.3 Scale by group size and dynamics

If the only selection criteria used by a RAP to select a JAP were proximity to the client, then the RAP could ignore the number of targets reachable at each JAP. In order to load balance across targets, however, RAPs must know roughly how many targets are at each JAP. In this way, RAPs can select JAPs in a load balanced way, and each JAP can subsequently select targets in a load balanced way. Unfortunately, requiring that RAPs maintain counts of targets at JAPs increases the load on RAPs. This could be a problem for very large groups, or for groups with a lot of churn.

We mitigate this problem by allowing the JAP to give the RAP an approximate number of targets, for example within 25% or 50% of the exact number. For instance, if 25% error is allowed, then a JAP that reported 100 targets at one time would not need to report again until the number of targets exceeded 125 or fell below 75. This approach allows us to trade-off the granularity of load-balancing for scalability with group size and dynamics. Indeed, this trade-off can be made dynamically and on a per-group basis. A RAP that is lightly loaded, for instance, could indicate to the JAP that 100% accuracy reporting is allowed (i.e. in its acknowledgement messages). As the RAP load goes up, it would request less accuracy, thus reducing its load. The combination of the two-tiered approach with inaccurate information in a system with 2 groups is illustrated in Figure 3 (the figure assumes that there is just one RAP for each group). Section 4.2 presents simulations that show the benefits of this approach in the case of a large, dynamic group.

In any event, the number of targets is not the only measure of load. Individual targets may be more-or-less loaded due to differing loads placed by different clients. Ultimately, the JAP may simply need to send a message to the RAPs whenever its set of targets are overloaded for whatever reason.

3.4 Scale by number of proxies

Given that we have laid out the basic architecture of PIAS, we can now specifically look at PIAS deployment issues. A central question is, how many proxies may we reasonably expect in a mature PIAS deployment, and can we scale to that many proxies?

A key observation to make here is that the scaling characteristics of PIAS are fundamentally different from the scaling characteristics of IP routing. While the traffic capacity of the Internet can be increased by adding routers, the scalability of IP routing per se is not improved by adding routers. All routers must contain the appropriate routing tables. For instance, all Tier1 routers must contain the complete BGP routing table no matter how many Tier1 routers there are. For the most part, IP routing is scaled by adding hierarchy, not adding routers.

With PIAS, on the other hand, scaling does improve by adding proxies. With each additional proxy, there are lower ratios of target-to-JAP and group-to-RAP. Growth in the number of groups and targets can be absorbed by adding proxies. However, an increase in the number of proxies presents its own scaling challenge. Among other things, every proxy is expected to know the up/down status of every other proxy.

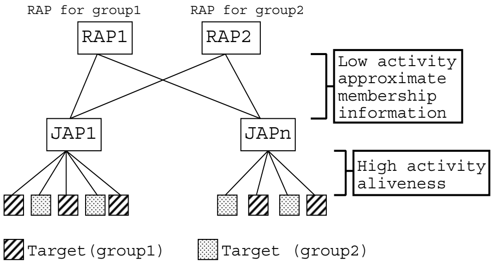


Figure 3: 2-tier membership management: the JAPs keep the aliveness status for the associated targets; the RAP for a group tracks the JAPs and an approximate number of targets associated with each JAP

The following describes a simple divide-and-conquer approach that can be used if the number of proxies grows too large. In a typical deployment, a given anycast service provider starts with one anycast prefix, and deploys proxies in enough geographically diverse POPs to achieve good proximity. As more anycast groups are created, or as existing anycast groups grow, the provider expands into more POPs, or adds additional proxies at existing POPs. With continued growth, the provider adds more proxies, but it also obtains a new address prefix (or splits the one it has), and splits its set of proxies into two distinct groups. Because the IP routing infrastructure sees one address prefix per proxy group, and because a proxy group can consist of thousands of proxies and tens of thousands of anycast groups, the provider could continue adding proxies and splitting proxy groups virtually indefinitely.

The size of a mature proxy deployment may be roughly calculated as follows. There are about 200 tier-1 and tier-2 ISPs [15]. An analysis of the ISP topologies mapped out in [16] shows that such ISPs have ∼25 POPs on average. Assuming that we'd like to place proxies in all of these POPs, this leads to 5000 POPs. Assuming 3-4 proxies per POP (for reliability, dicussed later), we get a conservative total of roughly 20,000 proxies before the infrastructure can be split.

While 20,000 proxies is not an outrageous number, it is large enough that we should pay attention to it. One concern not yet addressed is the effect of the number of proxies on IP routing dynamics. In particular, BGP reacts to route dynamics (flapping) of a single prefix by "holding down" that prefix—ignoring any advertisements about the prefix for a period of at most one hour [17]. A naive proxy deployment where each proxy advertises the anycast prefix directly into BGP would imply that a proxy failure necessitates a BGP withdrawal for the prefix (from the site where the proxy is located) that could lead to hold downs. While the proxy stability ensures that such events do not occur often, even the occasional prefix instability and the consequent service disruptions that a large proxy deployment would entail are not acceptable.

Hence, the deployment model involves more than one proxy being placed inside every POP where the proxies are deployed. Such an arrangement is referred to as an ***anycast***

Segment	Failure of	Failover through	Section
AC⇒IAP	IAP	IGP, onto a proxy within the same cluster	3.6
IAP⇒JAP	JAP	proxy health monitoring system	3.6
JAP⇒AT	AT	pings between target and JAP, passive monitoring by JAP	3.1,3.2
AT⇒JAP	JAP	pings routed to a different proxy who becomes JAP	3.6
JAP⇒AC	AC	no failover needed	-

Table 1: Failover along the PIAS forward path (AC⇒IAP⇒JAP⇒AT) and reverse path (AT⇒JAP⇒AC)

cluster[7] and is based on the model used by the anycasted f-root server[18]. The approach involves connecting one or more routers and more than one proxy to a common subnet. All the proxies in the cluster advertise the anycast prefix into IGP while the routers advertise it into BGP and hence, a proxy-failure does not lead to a BGP withdrawal.

3.5 Proximity

The introduction of the proxies into the IP path negates the natural ability of native IP anycast to find the nearest target. Therefore, we require explicit mechanisms in PIAS to regain this capability.

As mentioned before, native IP anycast sets the client-IAP and target-JAP path segments. The RAP, on the other hand, selects the JAP, and therefore sets the IAP-JAP path segment (on forward packets) and the JAP-client path segment (on return packets). To ensure the proximity of the target to the client, the RAP must choose a JAP close to the IAP and hence, every AP must know the distance (in terms of latency) between every pair of APs. This could be accomplished using a proximity addressing scheme like GNP [19] or Vivaldi [20].

Another possibility is to use a simple, brute-force approach whereby every AP occasionally pings every other AP and advertises the minimum measured round trip time (RTT) to all other APs. This is feasible because, with the cluster deployment approach, RAPs only need to know the distance between each pair of clusters. While validating the above claim would require experimentation with the actual deployment, back of the envelope calculations do paint a promising picture for the simple approach.

3.6 Robustness and fast failover

The introduction of proxies between client and target might have a negative impact on the robustness of PIAS as compared to native IP anycast. On the other hand, RON[14] has shown how an overlay structure can be used to improve the resiliency of communication between any two overlay members. Extending the same thought, PIAS, by ensuring the robustness of packet traversal through the proxy overlay, can improve the resiliency of communication between clients and group members. We believe that given the stable nature of the proxies, their deployment in well connected parts of the Internet (tier-1 and tier-2 ISPs) and the engineering that would go into their set-up, PIAS should be able to match, if not better, the robustness offered by native IP anycast.

A related requirement is that of fast fail-over. "E2E" native IP anycast has to achieve failover when a group member crashes, so that clients that were earlier accessing this member are served by some other group member. Given the way native IP anycast works, this failover is tied to IP routing convergence. Specifically, in case of a globally distributed group, the failover is tied to BGP convergence, which in some cases can extend to a few minutes[14]. Since PIAS uses native IP anycast to reach the proxies, it is subject to the same issues. The process of overcoming the failure of a proxy is termed as **proxy failover**. In addition, the proxies must themselves be able to fail over from one target to another which is termed as **target failover**. Thus the failover problem seems worse with PIAS than with native IP anycast; however, this is not the case.

3.6.1 Target failover

As discussed in Sections 3.1 and 3.2, the JAP is responsible for monitoring the aliveness of its targets. It does this through pinging and tracking data packets to and from the target. The JAP is also responsible for directing IAPs to delete their cache entries when enough targets have failed.

3.6.2 Proxy failover

There is still the question of clients failing over onto a different proxy when their IAP crashes, and targets failing over when their JAP crashes. And there are two levels at which this must be achieved: at the routing level and at the overlay level.

At the routing level, the system must be engineered such that when a proxy fails, clients that were using this proxy as an IAP are rerouted to some other proxy quickly. PIAS's deployment of proxies in a cluster means that this failover is across proxies within the same cluster. Also, since the proxies advertise the prefix into IGP, PIAS relies on IGP for convergence after a proxy failure and hence can achieve faster failover. Typically, this is of the order of a few seconds and can be reduced to sub-second times[21].

At the overlay level, to monitor the health of proxies, we use a 2-tier health monitoring system. At the first tier, the proxies within the same proxy cluster are responsible for monitoring each other. At the next level, each proxy in a cluster monitors the health of a small number of other clusters. When either an individual proxy or an entire cluster fails, it is detected quickly and communicated to all remaining proxies.

Section 3.2 had described IAP behavior when a JAP goes down. The only thing left to discuss is target behavior when a JAP goes down. In this case, native IP anycast routing will cause ping packets from the target to reach another JAP, which will ask the target to re-register. Table 1 sums up the way PIAS achieves failover across various segments of the client-target path.

3.7 Target selection criteria

As described earlier, the RAP may select the JAP based on a number of criteria, including proximity, load balancing, and connection affinity[8]. The JAP subsequently selects a target. It is this selection process, divorced from IP routing, that allows PIAS to offer richer target selection criteria

How PIAS achieves load balance and proximity has already been discussed. Connection affinity is discussed later in this section. We wish to point out here that these three important selection criteria are in fact at odds with each other. For

[7] hereon referred to as proxy cluster or simply, cluster

[8] Connection affinity—all packets from a given connection or flow are delivered to the same target.

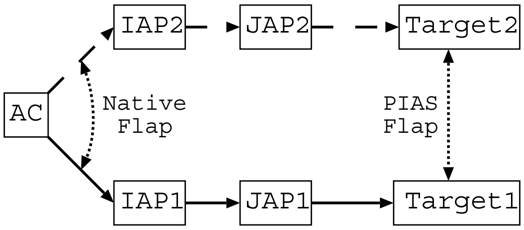


Figure 4: Lack of native IP anycast affinity can cause flaps in the PIAS model

exampl, if both load balance and proximity are important criteria, and the JAP nearest to the IAP is heavily loaded, then one of the other criteria must be compromised. This basic set of trade-offs applies to application-level anycast as well.

By never selecting the source of a packet as the target, PIAS allows a host to be both a target and a client for a given group. Packets sent by the target to the group address would be forwarded to some group target other than the sender. Note that this is not possible with native IP anycast and it allows PIAS to support new P2P applications (section 6.1).

Proxies could potentially base their target selection on various scoping criteria. These selection criteria can be expressed by overloading the transport address, i.e. a group can have separate TAs for each type of scoping. For instance, an anycast packet could be administratively scoped. That is, it could indicate that the target should be in the same site, belong to the same DNS domain, or have the same IP address prefix (or be from different sites, DNS domains, or IP prefixes). While how this would be configured and operated is a good topic for further study, the selection functionality of the RAP allows for the possibility of many such features.

Another form of selection would be to pick a random target rather than the nearest target - the RAP would pick a random JAP who would then pick a random target. Random selection among a group can be useful for various purposes such as spreading gossip [22] or selecting partners in multicast content distribution [23]. Indeed, in the PIAS architecture, there is no reason an anycast packet cannot be replicated by the RAP and delivered to a small number of multiple targets. The salient point here is that, once IP anycast functionality is divorced from IP routing, any number of new delivery semantics are possible if the benefits justify the cost and complexity.

3.7.1 *Connection affinity*

Lack of connection affinity in native IP anycast has long been considered one of its primary weak points. This issue spills over into PIAS. Specifically, the issue is how to maintain affinity when native IP anycast causes a different IAP to be selected during a given client connection. If the same IAP is always used, then packets will be sent to the same JAP that was initially cached by the IAP. However, a change in the IAP could lead to a change in the target the packets are delivered to, as shown by Figure 4. Application-layer anycast doesn't have this problem, because it always makes its target selection decision at connection start time, and subsequently uses unicast.

A simple solution would be to have RAPs select JAPs based on the identity of the client, such as the hash of its IP address. This way, even if IP routing caused packets from a given client to select a different IAP, they would be routed to the same JAP and hence the same target. Unfortunately, this approach completely sacrifices proximity and load balance. Broadly, another approach would be to modify the host application by making it anycast aware, and redirect the host to the unicast address of a selected target (either PIAS or the target itself could do this redirect). There are some security issues here—the redirect must be hard to spoof—but these are surmountable.

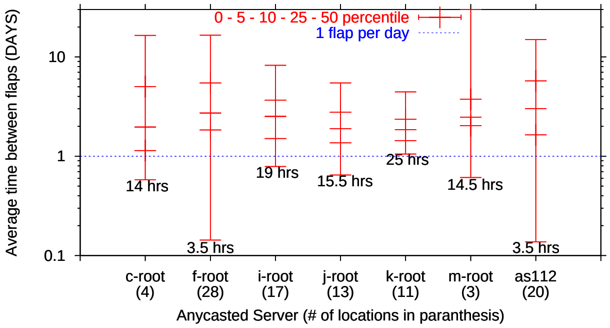


Figure 5: Percentiles for the average time between flaps for all the anycasted destinations

We can also imagine complex schemes whereby JAPs and IAPs coordinate to insure affinity. However, a fundamental question that still has not been answered is, how good or bad is the affinity offered by native IP anycast? It might be the case that the affinity offered by native IP anycast is very good; i.e. the probability that a connection breaks due to a routing flap is very small as compared to the probability of the connection breaking due to other factors. This would imply that we do not need the complex mechanisms stated above. In this regard, we did some measurements to find out the affinity offered by native IP anycast. Our results, while preliminary, suggest that native IP anycast affinity is quite good, and PIAS need not do anything extra to provide reasonable connection affinity. Details of these measurements are presented in section 4.1

4. EVALUATION

In this section we evaluate the PIAS architecture using measurements and simulations. Section 4.1 describes the measurements made using the Planetlab[24] testbed and the anycasted DNS root servers to argue for the sufficiency of the affinity offered by native IP anycast and hence, PIAS. Sections 4.2 and 4.3 present simulation results that show the scalability (by group characteristics) and the efficiency of the PIAS deployment. Finally, section 4.4 discusses our PIAS implementation. We also measured the quality of proximity selection offered by the anycasted DNS server deployments. These are briefly discussed in section 7.

4.1 Connection Affinity measurements

As mentioned earlier, it is important to determine the affinity offered by native IP anycast in order to understand the need for mechanisms to ensure affinity in PIAS. This section presents the results of our measurement study aimed to do so. The goal of the study was to determine how often IP routing selected different locations when sending packets to a native IP anycast address. We used the anycasted root servers and the AS-112 servers as the anycast destinations. For clients, we used 129 Planetlab nodes belonging to 112 sites.

For each anycast destination, the clients probed the associated anycast address every 10 seconds to determine the

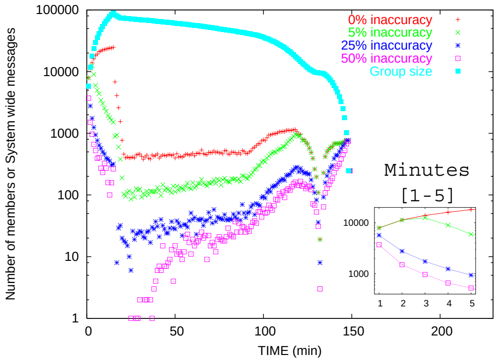


Figure 6: System wide messages from the all the JAPs to the 4 RAPs during the event for varying degrees of inaccuracy

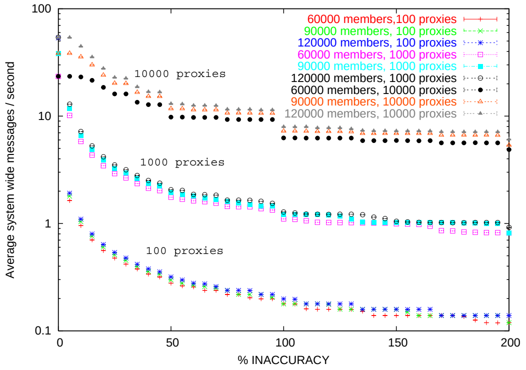


Figure 7: Average system wide messages (per second) versus the percentage of inaccuracy with varying number of proxies and varying maximum group size.

location they are routed too. The servers at different locations have been configured by their operators to respond to a TXT type DNS query with their location[25] and hence, the probes were DNS queries generated using *dig*. This data was collected for a period of 30 continuous days in Dec'04-Jan'05.

The probing of the anycasted destinations reveals changes in routing or 'flaps' that cause packets to be delivered to different locations of an anycasted server. So, a pair of probes from a given Planetlab node switching from the San Jose f-root server to the Palo Alto f-root server[9] would be counted as one flap. Using our measurement data, we determined the average time between flaps to a given root server for each probing node. Figure 5 plots various percentiles for the average time between flaps when probing various anycasted servers. The figure shows that the anycasted services are very stable as viewed from almost all locations. For example, more than 95% of the nodes observed less than a flap per day for all the anycasted destinations. Similarly, ∼48% of the nodes never observed a flap when probing the f-root during the entire 30 day period.

Also, the few nodes that observed frequent flaps (i.e. an average inter-flap duration of less than a day) had their average skewed by tiny bursts of instability in between large periods of stability. For example, the Planetlab node that experienced most flaps (208) over the month when probing j-root was in Leixip, Ireland. Of these, 180 flaps occurred in a 3-hour period. We conjecture that such phenomena can be attributed to ephemeral issues specific to the sites to which these nodes belong. While a more rigorous analysis of the collected data and correlation with BGP-updates for the prefixes representing these anycasted destinations would be needed for determining the causes and patterns amongst these flaps, the overall figures do paint an encouraging picture. These measurements reveal that the probability that a two minute connection breaks due to a flap is about 1 in 4500 and the probability that an hour long connection breaks is about 1 in 150. Note that it is the short connections that, in order to avoid the overhead of anycast to unicast redirect, need to rely on anycast affinity. Long connections can incur the overhead of a redirect and hence, could use anycast for discovery and unicast for the actual communication.

We admit that the limited number(129) and variety of vantage points and the number of locations of the anycast destinations makes our study preliminary. Also, the operators of j-root, based on their observations, have come to the opposite conclusion regarding the ability of native IP anycast to support stateful connections[26]. While their results are being debated by many in the operational community[27], we are trying to acquire the relevant data-sets so as to find the reason for the flapping observed by them (something that the authors of the j-root study have not analyzed).

4.2 Scalability by group size and dynamics

In this experiment, we evaluate PIAS's ability to handle large and dynamic groups (as described in 3.3). We simulate the load imposed by a large group with high churn on the proxy infrastructure. The dynamics of the simulated group - the arrival rate of group members and the session duration cumulative distribution function - resemble the dynamics of the largest event observed in a study of large-scale streaming applications[28]. Simulation of just one such group is sufficient as the load imposed varies linearly with the number of such groups supported.

The PIAS infrastructure in the simulation has varying number of proxies and maximum group size. We simulate four RAPs per group. We want to measure the number of messages required to keep the 2-tier membership hierarchy updated in face of the group dynamics. This is the number of messages from the JAPs of the group to the 4 RAPs and is referred to as **'system wide messages'**.

Figure 6 plots the system wide messages produced with a proxy deployment of size 1000 and the group size bounded by 90000. The topmost curve in the figure shows how the group size varies with the time. A flash crowd, at a rate of ∼100 members/second, leads to a sudden rise in the group size in the first 10 minutes. The other curves plot the number of messages produced in the corresponding minute (as plotted along the X-axis) for varying degrees of inaccuracy. The degree of inaccuracy, as explained in section 3.3, implies that a JAP only informs a RAP of a change in the number of members associated with it if the change is more than a certain percentage of the last value sent.

The inaccuracy of information offers only a small benefit in the nascent stages of the group (the first minute). This is because no matter what inaccuracy percentage we use, the JAP must inform the RAP of the first group member that contacts it. In the next couple of minutes, as the group increases in

[9] San Jose and Palo Alto are two locations of the f-root server

size and more members join their corresponding JAPs, the inaccuracy causes the traffic towards the 4 RAPs to drop rapidly (see the embedded graph in figure 6). Overall, the average number of messages over the duration of the entire event reduces from 2300 per min. with the naive approach to 117 per min. with 50% inaccuracy.

Figure 7 plots the average system wide messages (per second) versus the percentage of inaccuracy for varying number of proxies and varying maximum group size. Each plotted point is obtained by averaging across 20 runs. All curves tend to knee around an inaccuracy mark of 50%−60%. The closeness of the curves for different sized groups (given a fixed number of proxies) points to the scalability of the system by the group size even in the face of high churn.

More interesting is the variation of the load on the RAPs with the number of proxies. As the number of proxies increase, the number of JAPs increase; an offshoot of the assumption that the group members are evenly distributed across the proxy infrastructure. For a given group size, each JAP is associated with lesser number of group members. Hence, there is lesser benefit due to the inaccuracy approach. This shows up as the increase in the average number of messages directed towards the RAPs with the number of proxies.

The figure shows that such an extreme group in a 100 proxy deployment with 100% inaccuracy would require an average of ∼0.18 messages/second. As a contrast the same setup in a 10000 proxy deployment would necessitate an average of ∼7.25 messages/second. The low message overhead substantiates the PIAS scalability claim. Note that a larger number of proxies implies that each proxy is a RAP for a smaller number of groups. The number of targets associated with each proxy (as a JAP) reduces too. Thus, increasing the number of proxies would indeed reduce the overall load on the individual proxies.

4.3 Stretch

PIAS causes packets to follow a longer path (client ⇒ IAP ⇒ JAP ⇒ target). We have argued that the combination of native IP anycast and proxy-to-proxy latency measurements minimizes the effect of this longer path. This section simulates the stretch introduced by PIAS along the end-to-end path.

For the simulation, we use a subset of the actual tier-1 topology of the Internet, as mapped out in the Rocketfuel project [16]. This subset consists of 22 ISPs, 687 POPs, and 2825 inter-POP links (details in [29]). The use of only the tier-1 topology can be justified on two grounds. First, a large proportion of traffic between a randomly chosen client-target pair on the Internet would pass through a tier-1 ISP. Second, such a simulation gives us an approximate idea about the overhead that a PIAS deployment restricted to tier-1 ISPs would entail.

The topology was annotated with the actual distance between POPs (in Kms) based on their geographical locations. We then used SSFNET[30] to simulate BGP route convergence. This allowed us to construct forwarding tables at each of the POPs and hence, determine the forwarding path between any two POPs.

The simulated PIAS deployment involves placing a variable number of proxies at random POPs, one proxy per POP. These POPs are referred to as the *proxy POPs*. For every client-target pair to be simulated, we choose a POP through which the client's packets enter the topology (the *client POP*) and a POP through which the target's packets enter the topology (the *target POP*). The forwarding paths between the client and the target through these POPs represents the *direct path*. The IAP is assumed to be in the *proxy POP* closest to the client POP—this is the *IAP POP*. Similarly, the JAP is in the *proxy POP* closest to the target POP—this is the *JAP POP*. The *PIAS path* comprises of the following three segments: from the *client POP* to the *IAP POP*, from the *IAP POP* to the *JAP POP* and from the *JAP POP* to the *target POP*.

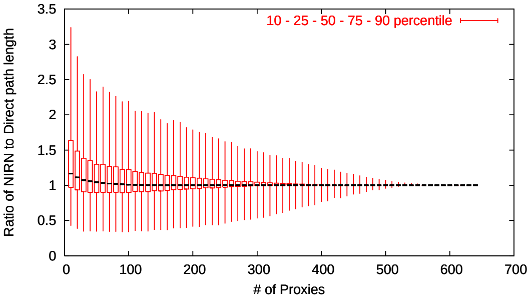


Figure 8: Percentiles for the stretch with varying number of proxies

Figure 8 plots the percentiles for the stretch with varying number of proxies. For a given number of proxies, we simulated 100000 runs. Each run comprised of simulating a client-target pair and finding the *direct* and the *PIAS path* length (in kms). Note that the well-documented non-optimal nature of inter-domain routing[14] is reflected in the cases where the PIAS path turns out to be shorter than the direct path. The figure shows that with a deployment of just 100 proxies (a mature deployment might encompass 50 times more POPs), the median stretch is 1.01 with the 90^{th} percentile being 2.2. Hence, even with a small size deployment, PIAS performs well with regards to the direct path.

4.4 Implementation

We have implemented the PIAS system and are in the process of deploying it. The current implementation of PIAS proxies comprises of a user-space component responsible for the overlay management tasks, such as handling proxy failures, target join/leaves, health monitoring etc. and a kernel-space component responsible for the actual forwarding of packets through the use of Netfilter hooks[31]. This involves tunnelling of the packets when sending them between 2 proxy nodes, and using a NAT when handling packets to/from a target.

5. RELATED WORK

Table 2 summarizes the pros and cons of PIAS, application level anycast, and other related approaches described below.

Partridge et. al. [1] originally proposed the IPv4 anycast service. It involves assigning an otherwise unicast IP address IP_A to multiple hosts, and advertising it into the routing infrastructure from all the hosts. Packets addressed to IP_A will be forwarded to the host nearest to the packet source in terms of metrics used by the routing protocol. Later, IPv6 incorporated anycast into its addressing architecture[8]. It allowed for scoped anycast addresses for groups confined to a topological region, which does not burden the global routing system. However, a globally spread group still poses scalability problems. Besides, IPv6 anycast also inherits all the other

Criterion (related to goal number)	IPv4	IPv6	IP + GIA	App. Level	*i*3	PIAS
Router Modification(1)	No	No	**Yes**	No	No	No
Client Modification(1)	No	No	No	No	**Yes**	No
Scalability by group size(2)	Very Good	Very Good	Very Good	**Poor**	**Poor/Good**[10]	Good
Stretch(3)	No	No	Little/No	No	Little	Little
Robustness(4)	No Issues	No Issues	No Issue	Mixed	Mixed	Mixed[11]
Failover(5)	Fast[12]	Fast[12]	Fast[12]	Fast	Fast	Fast
Target Deployment(6)	**Difficult**	**Difficult**	**Difficult**	Easy	Easy	Easy
Scalability by no. of groups(7)	**No**	**No**	Yes	Yes	Yes	Yes
Scalability by group dynamics(8)	**Poor**	**Poor**	**Poor**	**Poor**	**Poor/Good**[10]	Good
Cost of Proximity(9)	None	None	Small	**Large**	**Large**	Small
Low-level access	Yes	Yes	Yes	**No**	Yes	Yes

Table 2: The Anycast Design Space

limitations of IPv4 anycast. Despite the shortcomings, there has been work detailing the relevance of anycast as a tool for service discovery and other applications, both for IPv4[32] and for IPv6[33].

Katabi and Wroclawski[34] proposed an architecture that allows IP anycast to scale by the number of groups. Their approach is based on the observation that services have a skewed popularity distribution. Hence, making sure that the unpopular groups do not impose any load on the routing infrastructure addresses the scalability issue. However, the need to change routers puts a severe dent on the practical appeal of the approach. Besides, being a router-based approach, it suffers from most other limitations of IPv4 anycast.

Because of the limitations of these approaches, anycast today is typically implemented at the application layer. This offers what is essentially anycast service discovery—DNS-based approaches use DNS redirection while URL-rewriting approaches dynamically rewrite the URL links as part of redirecting a client to the appropriate server. Related proposals in the academic community include [35][36]. The idea behind these is to identify the group using an application level name that, at the beginning of the communication, is mapped to the unicast address of a group member. The reliance on unicast support from the underlying IP layer implies that these approaches circumvent all limitations of IP anycast. The challenge here is to collect the relevant selection metrics about the group members in an efficient and robust fashion.

Another element in this design space is anycast built on top of the indirection architecture offered by i3[37]. i3 uses identifiers as a layer of indirection that generically gives the receiver tremendous control over how it may (or may not) be reached by senders. One of the services i3 can provide is anycast. There are two main advantages of PIAS over i3 for the anycast service. First, PIAS requires no changes in the protocol stack, whereas i3 requires a new layer inserted below transport. A PIAS client, on the other hand, can use PIAS with no changes whatsoever. Second, because PIAS uses native IP anycast, it is easier to derive proximity from PIAS than from i3. PIAS only has to measure distances between proxies—i3 has to measure distances to clients and targets. The main advantage of i3 over PIAS is that it is easier to deploy an i3 infrastructure than a PIAS infrastructure, precisely because i3 doesn't require IP anycast. Indeed, this has been a source of frustration for us—we can't just stick a PIAS proxy on Planetlab and start a service.

As far as the broader notion of indirection is concerned, there is no question that i3 is more general. Its ability for both the sender or receiver to chain services is very powerful. The addressing space is essentially infinite, and hosts can create addresses locally. Finally the security model (that supports the chaining) is elegant and powerful. Having said that, PIAS does provide indirection from which benefits other than just anycast derive. For unicast communications, it could be used to provide mobility, anonymity, DoS protection, and global connectivity through NATs. In the best of all worlds, we'd want something like i3 running over PIAS. But IPv6 and NAT have taught us that you don't always get the best of all worlds, and considering PIAS's backwards compatibility, it may after all be the more compelling story.

6. ANYCAST APPLICATIONS

Given that PIAS offers an easy-to-use global IP anycast service that combines the positive aspects of both native IP anycast and application-layer anycast, it is interesting to consider new ways in which such a service could be used.

6.1 Peer Discovery

Though IP anycast has long been regarded as a means of service discovery, this has always been in the context of clients finding servers. PIAS opens up discovery for P2P networks, where not only is there no client/server distinction, but peers must often find (and be found by) multiple peers, and those peers can come and go rapidly. Examples of such cases include BitTorrent and network games.

One reason that traditional IP anycast has not worked for peer discovery (other than difficulty of deployment), is that an IP anycast group member cannot send to the group—packets are just routed back to themselves. With the right selection characteristics, PIAS can support a wide-range of P2P applications. Random selection would allow peers to find arbitrary other peers, and is useful to insure that unstructured P2P networks are not partitioned. Proximity is obviously also important, but to insure that a peer can find multiple nearby peers (rather than the same peer over and over), a selection service whereby a node can provide a short list of targets to exclude (i.e. already-discovered targets) could be used.

6.2 Reaching an Overlay network

A very compelling application of PIAS would allow a RON[14] network to scale to many thousands of members, and would allow those members to use RON not only for exchanging packets with each other, but with any host on the Internet! What follows is a high-level description of the approach. Assume a set of 50-100 RON "infrastructure" nodes that serve

[10] Note that the way i3 has described their anycast, it wouldn't scale to very large or very dynamic groups, because a single node holds all the targets and receives pings from the targets. It may be possible that i3 could achieve this with a model closer to how they do multicast, but we're not sure.

[11] for reasons described in first paragraph of section 3.6

[12] they can be engineered to be fast by relying on IGP for convergence

many thousands of RON clients. The RON nodes all join a large set of anycast groups—large enough that there is an anycast transport address (TA) for every possible client connection. The RON nodes also partition the anycast TAs so that each TA maps to a single RON node. Clients discover nearby RON nodes (or a couple of them) using one of the anycast groups, and establish a unicast tunnel (for instance, a VPN tunnel) with the RON node. We call this the *RON tunnel*, and the RON node is referred to as the *local RON*.

When a client wishes to establish a connection with some remote host on the Internet, it does so through its RON tunnel. The local RON assigns one of its TAs to the connection using NAT, and forwards the packet to the remote host. When the remote host returns a packet, it reaches a nearby RON node, called the *remote RON*. Because the transport address of the return packet maps to the local RON node, the remote RON node can identify the local RON node. The remote RON tags the packet with its own identity, and transmits the packet through the RON network to the local RON node, which caches the identity of the remote RON, and delivers the packet to the client. Now subsequent packets from the client to the remote host can also traverse the RON network.

This trick isn't limited to RONs. It could also work for route optimization in Mobile IP[13] (for v4 or v6, see [38] for a description of the problem), or simply as a way to anonymize traffic without sacrificing performance.

7. DISCUSSION

In this paper, we have presented the basic aspects of PIAS. A "practical" IP anycast service, however, requires a number of features that we don't have space to describe in detail. For example, the need for scoping whereby packets from clients in a domain (enterprise) are always served by targets within the domain. This can be achieved by deploying a PIAS proxy in the domain, or simply by deploying intra-domain native IP anycast.

Another important issue is security. The IP routing infrastructure is secured router-by-router through human supervision of router configuration. This makes routing security error-prone and unreliable. Since PIAS involves advertising a prefix into inter-domain routing, it is afflicted by the same issues. However, it is important to note that PIAS does not worsen the situation. Also, the fact that from the routing point of view, an anycasted autonomous system is akin to a multi-homed autonomous system implies that any future solution for routing security would apply directly to the PIAS deployment.

PIAS, however, does need to explicitly secure its *join* and *leave* primitives. The fact that these primitives are to be used by group members who have an explicit contract with the anycast service provider implies that we could use standard admission control schemes; for example PIAS could adapt any of a number of network or wireless authentication protocols like EAP [39]. Previous work on using overlays to protect specific targets from DOS attacks [40] described some approaches to allow controlled access to the overlay.

An assumption implicit in PIAS's claim of incurring minimal stretch (section 4.3) is the proximity of the client to the IAP and of the server to the JAP. This assumption is justified by the fact that these relations are discovered using native IP anycast and hence, the distances are small in terms of metrics used by inter-domain routing. However, this does not necessarily imply that the distances are small in terms of latency. As a matter of fact, preliminary measurements done by us show that the assumption does not hold for the j-root server anycast deployment. We found that native IP anycast does not do a great job of selecting close-by locations, at least not for the j-root server deployment. For example, 40% of the measured clients experienced a stretch of more than 4 when accessing the anycasted j-root. The measurement methodology and the results are detailed in [41].

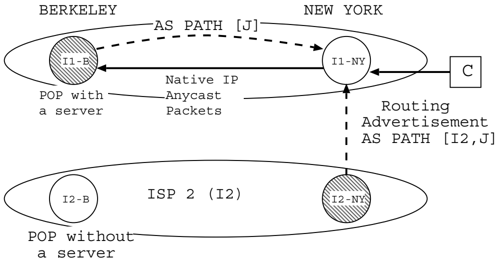


Figure 9: Native IP anycast inefficiency - packets from client C in New York destined to the native IP anycast address are routed to the anycast server in Berkeley, even though there is a server in New York

We believe the inefficacy of anycast when selecting close-by root-servers might be due to the way the j-root servers have been deployed - all 13 anycasted servers for j-root are placed in POPs of different ISPs. A possible problem with this approach is illustrated in figure 9. The figure shows 2 ISP networks- I1 and I2, each having a POP in New York and in Berkeley. It also shows a native IP anycast deployment (AS number J) with two servers - one hosted at the New York POP of I2 (I2-NY) and the other at the Berkeley POP of I1 (I1-B). The figure has these POPs highlighted. The anycast servers have an EBGP relation with the routers of the hosting POP; hence, the anycast prefix is advertised with J as the origin AS. Now, if a client (C) in the New York area sends packets to the anycast address and these reach POP I1-NY, they will be routed to the server hosted at I1-B. This is because the routers in I1-NY would prefer the 1 AS-hop path ([J]) through I1-B to the anycasted server over the 2 AS-hop path ([I2,J]) through I2-NY. Note that the anycasted server hosted at I1-B represents a customer of I1 and so, it would be very uncommon for I1 to steer these packets towards I2-NY due to local policies (local preference values); rather the AS path length would dictate the path.

Although negative, the importance of the result cannot be overemphasized. It brings out the fact that a naive proxy deployment might not achieve low-latency client-IAP and JAP-target paths. Also, an unverified implication of the above analysis is that for good performance, an ISP that is part of the deployment[14] should be sufficiently covered, i.e., there should be clusters at a decent number of POPs of the ISP. For example, deployment of the two servers in the figure at both of the POPs of I1 (I1-NY and I1-B) or I2 (I2-NY and I2-B) would avoid the problem of long paths. We believe that such an approach would ensure that the client-IAP and the target-JAP segments are latency-wise small - something that can only be substantiated when we get the PIAS deployment going

[13] Details withheld for lack of space.

[14] the ISP has at least one POP hosting a proxy cluster

8. CONCLUSIONS

In this paper, we propose a proxy based IP anycast service that addresses most of the limitations of native IP anycast. Specifically, the primary contribution of this paper is the design of PIAS, a practically deployable IP anycast architecture. The unique features of PIAS such as the scalability by the size and dynamics of groups mean that it opens up new avenues of anycast usage. The purported scalability has been substantiated through simulations representing extreme, but real, workloads. Simulations on the real tier-1 topology of the Internet point to the efficiency of our approach.

The fact that PIAS uses native IP anycast means that it can be used as a simple and general means of discovery and bootstrapping. Internet measurements against the anycasted DNS root servers show that the reliance on native IP anycast does not undermine PIAS's ability to support connection oriented services. A PIAS prototype has been built and the deployment efforts are underway. We feel confident that PIAS has the potential of fulfilling the need for a generic Internet-wide anycast service that can serve as a building block of many applications, both old and new.

Acknowledgements

We are grateful to Xinyang Zhang for help with the simulations and to David Anderson for design discussions. We would also like to thank the anonymous reviewers for their feedback. This material is based upon work supported by AFOSR MURI under Award No. F49620-02-1-0233. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of AFOSR MURI.

9. REFERENCES

[1] C. Partridge, T. Mendez, and W. Milliken, "RFC 1546 - Host Anycasting Service," November 1993.

[2] T. Hardy, "RFC 3258 - Distributing Authoritative Name Servers via Shared Unicast Addresses," April 2002.

[3] J. Abley, "Hierarchical Anycast for Global Service Distribution," ISC Technical Note ISC-TN-2003-1 www.isc.org/tn/isc-tn-2003-1.html.

[4] D. Kim, D. Meyer, H. Kilmer, and D. Farinacci, "RFC 3446 - Anycast Rendevous Point (RP) mechanism using Protocol Independent Multicast (PIM) and Multicast Source Discovery Protocol (MSDP)," January 2003.

[5] D. Katabi, "The Use of IP-Anycast for Building Efficient Multicast Trees," in *Proc. of Global Telecommunications Conference*, 1999.

[6] C. Huitema, "RFC 3068 - An Anycast Prefix for 6to4 Relay Routers," June 2001.

[7] "AS112 Project Home Page," www.as112.net.

[8] R. Hinden and S. Deering, "RFC 3513 - Internet Protocol Version 6 (IPv6) Addressing Architecture," April 2003.

[9] Akamai Technologies Inc., "Internet Bottlenecks: the Case for Edge Delivery Services," 2000, www.akamai.com/en/resources/pdf/whitepapers/Akamai_Internet_Bottlenecks_Whitepaper.pdf.

[10] B. Greene and D. McPherson, "ISP Security: Deploying and Using Sinkholes," www.nanog.org/mtg-0306/sink.html, June 2003, NANOG TALK.

[11] D. R. Karger, E. Lehman, F. T. Leighton, R. Panigrahy, M. S. Levine, and D. Lewin, "Consistent Hashing and Random Trees: Distributed Caching Protocols for Relieving Hot Spots on the World Wide Web." in *Proc. of STOC*, 1997.

[12] R. Rodrigues, B. Liskov, and L. Shrira, "The design of a robust peer-to-peer system," in *Proc. of the Tenth ACM SIGOPS European Workshop*, September 2002.

[13] A. Gupta, B. Liskov, and R. Rodrigues, "One Hop Lookups for Peer-to-Peer Overlays," in *Proc. of 9th Workshop on Hot Topics in Operating Systems*, May 2003.

[14] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris, "Resilient overlay networks," in *Proc. of the eighteenth ACM Symposium on Operating Systems Principles*, 2001.

[15] L. Subramanian, S. Agarwal, J. Rexford, and R. H. Katz, "Characterizing the Internet Hierarchy from Multiple Vantage Points." in *Proc. of INFOCOM*, 2002.

[16] N. Spring, R. Mahajan, and T. Anderson, "Quantifying the Causes of Path Inflation," in *Proc. of ACM SIGCOMM*, August 2003.

[17] Z. M. Mao, R. Govindan, G. Varghese, and R. H. Katz, "Route flap damping exacerbates Internet routing convergence," in *Proc. of ACM SIGCOMM*, 2002.

[18] J. Abley, "A Software Approach to Distributing Requests for DNS Service Using GNU Zebra, ISC BIND 9, and FreeBSD," in *Proc. of USENIX Annual Technical Conference, FREENIX Track*, 2004.

[19] T. S. E. Ng and H. Zhang, "Predicting Internet Network Distance with Coordinates-Based Approaches." in *Proc. of INFOCOM*, 2002.

[20] F. Dabek, R. Cox, F. Kaashoek, and R. Morris, "Vivaldi: a decentralized network coordinate system," in *Proc. of ACM SIGCOMM*, 2004.

[21] C. Alaettinoglu and S. Casner, "Detailed Analysis of ISIS Routing Protocol on the Qwest Backbone," February 2002, NANOG TALK.

[22] A. J. Ganesh, A.-M. Kermarrec, and L. Massoulie, "SCAMP: Peer-to-Peer Lightweight Membership Service for Large-Scale Group Communication," in *Proc. of the Third International COST264 Workshop on Networked Group Communication*, 2001.

[23] D. Kostic, A. Rodriguez, J. Albrecht, and A. Vahdat, "Bullet: high bandwidth data dissemination using an overlay mesh," in *Proc. of the Nineteenth ACM Symposium on Operating Systems Principles*, 2003.

[24] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman, "PlanetLab: An Overlay Testbed for Broad-Coverage Services," *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 3, pp. 3–12, July 2003.

[25] "ISC F-Root Sites," www.isc.org/index.pl?/ops/f-root/.

[26] P. Barber, M. Larson, M. Kosters, and P. Toscano, "Life and Times of J-Root," www.nanog.org/mtg-0410/kosters.html, October 2004, NANOG TALK.

[27] R. Bush, Mailing list posting www.ripe.net/ripe/maillists/archives/routing-wg/2004/msg00183.html.

[28] K. Sripanidkulchai, A. Ganjam, B. Maggs, and H. Zhang, "The feasibility of supporting large-scale live streaming applications with dynamic application end-points," in *Proc. of ACM SIGCOMM*, 2004.

[29] X. Zhang, J. Wang, and P. Francis, "Scaling the Internet through Tunnels," pias.gforge.cis.cornell.edu/tbgp.pdf.

[30] "SSFNet," www.ssfnet.org/homePage.html.

[31] "Netfilter," www.netfilter.org.

[32] E. Basturk, R. Haas, R. Engel, D. Kandlur, V. Peris, and D. Saha, "Using IP Anycast For Load Distribution And Server Location," in *Proc. of IEEE Globecom Global Internet Mini Conference*, November 1998.

[33] S. Matsunaga, S. Ata, H. Kitamura, and M. Murata, "Applications of IPv6 Anycasting," draft-ata-ipv6-anycast-app-00, February 2005.

[34] D. Katabi and J. Wroclawski, "A framework for scalable global IP-anycast (GIA)," in *Proc. of ACM SIGCOMM*, 2000.

[35] E. W. Zegura, M. H. Ammar, Z. Fei, and S. Bhattacharjee, "Application-layer anycasting: a server selection architecture and use in a replicated Web service," *IEEE/ACM Trans. Netw.*, vol. 8, no. 4, pp. 455–466, 2000.

[36] Z. Fei, S. Bhattacharjee, E. W. Zegura, and M. H. Ammar, "A Novel Server Selection Technique for Improving the Response Time of a Replicated Service." in *Proc. of INFOCOM*, 1998.

[37] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana, "Internet Indirection Infrastructure," in *Proc. of ACM SIGCOMM*, 2002.

[38] "Mobility for IPv6 (mip6), IETF Working Group Charter," www.ripe.net/ripe/maillists/archives/routing-wg/2004/msg00183.html.

[39] B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson, and H. Levkowetz, "RFC 3748 - Extensible Authentication Protocol (EAP)," June 2004.

[40] A. D. Keromytis, V. Misra, and D. Rubenstein, "SOS: secure overlay services." in *Proc. of ACM SIGCOMM*, 2002.

[41] H. Ballani and P. Francis, "Root-Server Anycast Deployment: A Meaurement Study," pias.gforge.cis.cornell.edu/am.pdf.