Towards an Evolvable Internet Architecture

Sylvia Ratnasamy
Intel Research, Berkeley
sylvia@intel-research.net

Scott Shenker
U.C.Berkeley and ICSI
shenker@icsi.berkeley.edu

Steven McCanne
Riverbed Technology
mccanne@riverbed.com

ABSTRACT

There is widespread agreement on the need for architectural change in the Internet, but very few believe that current ISPs will ever effect such changes. In this paper we ask what makes an architecture evolvable, by which we mean capable of gradual change led by the incumbent providers. This involves both technical and economic issues, since ISPs have to be able, and incented, to offer new architectures. Our study suggests that, with very minor modifications, the current Internet architecture could be evolvable.

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

General Terms: Design.

Keywords: Network Architecture, Anycast.

1. INTRODUCTION

In the early days of the commercial Internet (mid 1990's) there was great faith in Internet evolution. Many believed that whenever new versions of IP (such as IPv6), or extensions to IP (such as IntServ or IP Multicast), were evaluated on testbeds and standardized by the IETF, the ISPs would soon deploy them in the public Internet. This was not envisioned as a one-time change, but as an ongoing process of Internet evolution. This widespread optimism about change fueled ambitious efforts to extend or revise the Internet architecture, and we all looked forward to a brighter future.

The remarkable success of the Internet surpassed our wildest imagination, but our optimism about Internet evolution proved to be unfounded. The unfortunate history of IPv6, IntServ, IP Multicast, and other such proposals has turned that early optimism into a deep pessimism about evolutionary architectural change. The prevailing wisdom now is that ISPs have little incentive to deploy new architectures; since they all have to act in concert, there is no competitive advantage to such advances, and the costs of universally deploying a new architecture are immense. Thus, the ISPs, which were once thought the agents of architectural change, are now seen as the main cause of the Internet impasse [1].

This comes at an unfortunate time as the need for evolution is more apparent than ever. The Internet's success has brought with it unforeseen stresses and strains that have revealed numerous limitations in the Internet architecture. The literature is replete with calls for change, from the enduring worries about security [2–5], quality-of-service [6] and mobility [7], to more recent concerns about high-speed congestion control [8] and middleboxes [9, 10], to fundamental revisions in the basic architectural framework [9, 11–17].

This collision between the improbability and the necessity of change has produced two reactions in the community. Some have tried to *augment* the Internet architecture through overlay networks. Overlays have been proposed for a variety of services, including multicast [18], quality-of-service [19], robust routing [20], and content distribution [21, 22]. These overlays would not lead to fundamental changes in the underlying architecture, but would merely mask some of its most obvious deficiencies.

More recently, overlays have been proposed as a way to *undermine* the current ISPs. As observed in [1], overlays are not restricted to offering isolated services; they can instead be used to deploy radically new architectures (with, of course, certain limitations in the QoS and security offered). As such, overlays would enable a new entrant — that is, an aspiring rather than an incumbent ISP — to enter the market with a new architecture. This, however, is revolution not evolution; change would require destabilizing the market, with the current ISPs replaced by a new generation of service providers. Moreover, maintaining evolution would require a succession of such revolutions.

Neither of these two stories is very comforting. One offers limited change, providing additional services but not substantial revision to the fundamental architecture. The other requires repeated market destabilization, which is unlikely to be an easy or frequent occurrence for such a global, large-scale business.

In this paper we return to the original goal of Internet evolution. Rather than take ISP non-cooperation as an unchangeable given, we regard the lack of incentives for evolution as an architectural flaw and ask how this can be remedied. Hence, we ask: how can one make an architecture evolvable? The term architecture can be a somewhat fuzzy one and so, to be specific, what we're referring to here is IP. By evolving the Internet architecture, we mean the ability to change IP – IP headers, addressing, forwarding and so forth. By *evolvable* we mean capable of gradual (but unlimited) change within the current market structure: that is, change that is not restricted architecturally and that does not require a change in the incumbent ISPs.

Many have posed this question before, and yet, we find ourselves with no proposed solution. Here we want to address this question in a very concrete and pragmatic fashion. Hence rather than adopt a clean-slate approach and design the "ideal" evolvable IP, we try and create an evolvable IP starting from the confines of today's IP. Our discussion is primarily a walk-through of design options and

SIGCOMM'05 August 21–26,2005, Philadelphia, Pennsylvania, USA.

their implications with little to offer in novel mechanisms or radical insights; it is the importance of the question at hand that provides the impetus. We turn to that question in the next section.

2. REQUIREMENTS FOR EVOLVABILITY

2.1 General Discussion

To express the problem of evolvability in very concrete terms, we reduce it to a single question:

When a new version of IP, call it IPvN, becomes standardized or otherwise defined, what conditions would lead ISPs to deploy it?

This question, which the remainder of this paper will attempt to answer, is primarily one of incentives and the mechanisms needed to support them. While it is technically straightforward (though perhaps expensive) for ISPs to support a new protocol, they must have the incentive to do so. However, this need for the proper incentives influences the technical means by which IPvN will be initially offered. Thus, it is the incentive issues that drive the evolutionary requirements. In this section we motivate and identify the technical conditions necessary to provide ISPs with the incentives to deploy IPvN. The following section then presents the details of how these technical requirements might be met. As before, we don't claim novelty for the ensuing discussion, merely (we hope) relevance.

The key to ensuring the right incentives is fostering competition. Early-adopting ISPs will offer new architectures only if they believe it will give them a competitive edge, and late-adopting ISPs will do so only if they feel they are at a competitive disadvantage without it. There are several (related) factors in encouraging competition:

- **Foster independent innovation:** an ISP should not be able to block innovation by others.
- **Enable customer choice:** ISPs will then have to compete for customers.
- **Allow ISPs some degree of control:** if ISPs have no control, then they cannot recoup their expenses and are unlikely to deploy.

These overly general principles are best appreciated in the context of a specific technical decision and hence, rather than expand on them here, we will repeatedly return to these points when discussing design options. One recurring theme is the conflict between user choice and ISP control; our discussion will highlight the tradeoffs we make between these.

Our technical discussion begins with four very basic assumptions that must be incorporated in any discussion of Internet evolvability. The first three limit our expectations, while the fourth offers hope:

(A1) Assume partial ISP deployment. Clearly, we cannot expect that all ISPs will simultaneously adopt IPvN. Requiring ISPs to move in lockstep allows individual ISPs to block deployment, and prevents early-adopters from gaining a competitive advantage. Thus, we should assume that only a subset of the ISPs, perhaps only one, will initially deploy IPvN. Moreover, we cannot even expect an adopting ISP to initially deploy IPvN in all of its routers. Hence, all mechanisms must work with only a subset of an ISP's routers implementing IPvN.

(A2) Assume partial ISP participation. While an ISP might choose not to deploy IPvN in its network, one might hope that it would participate in a larger, general plan for evolvability (for example, through general configuration tools that would help its clients access IPvN services deployed elsewhere). However, for the same reasons as above, any proposal requiring explicit participation by all ISPs is unlikely to achieve ubiquity. Hence, while any technical solution should accommodate ISP participation where available, we require that any plan for evolution to IPvN not rely on such global participation.

(A3) Assume the existing market structure and contractual agreements. For example, we avoid requiring that clients enter new contractual agreements beyond their current ones with access providers for basic Internet connectivity. In particular, we do not require that clients sign up with specific (possibly in addition to their current ISP) providers in order to obtain IPvN service. This constraint is to some extent self-imposed because of our desire for solutions that are practical in today's Internet. More importantly however, this assumption follows from our "evolution, not revolution" argument; *i.e.*, the general process of transition should avoid requiring such upheaval. Note though that we do not preclude such change; we merely avoid *requiring* it.

(A4) Assume revenue flow. If there is no financial gain to be realized from offering IPvN, then it won't be deployed. What we assume here is that if IPvN attracts users, then revenue will flow towards those ISPs offering IPvN. An ISP that attracts new customers would obviously increase revenue. We also posit that an ISP that attracts new traffic, by offering IPvN, will also gain revenue possibly due to increased settlement payments (traffic from non-offering ISPs to offering ISPs would increase).

The need for certain technical mechanisms follow from these assumptions. Tolerating partial deployment, both within an ISP and across ISPs, immediately implies the obvious conclusion that evolution will initially require some form of overlay or virtual network to bridge across ISPs that do not support IPvN. While overlays are common, the IPvN overlay is not administered as a single unit (unlike, for example, RON [20], PlanetLab [23], Akamai [24], OverCast [21]), but instead is formed by peering arrangements across the IPvN overlays within individual ISPs (akin to the MBone [25], XBone [26], *etc.*). We sketch an example of how this might be done in Section 3.3.

The assumption of revenue flow means that an ISP would be rewarded for attracting usage. This is a crucial component in fostering competition, and points to the single most important technical requirement for evolvability, universal access (UA), which we summarize below:

Require Universal Access All clients can use IPvN if they so choose, regardless of whether their ISP deploys IPvN or assists their clients in accessing IPvN.

Even if a single ISP deploys IPvN, every client has access to it. Thus, no ISP can block use of IPvN. In this way, universal access fosters innovation.

Universal access also provides for customer choice; customers can access IPvN no matter what their ISP does. If access was restricted such that clients could only access IPvN with the assistance of their ISP, then customer choice would be severely limited. This would, in turn, decrease competition between the ISPs since it

would require changing ISPs, which is a significant burden, before a customer could experiment with IPvN.

On a more positive note, universal access creates a positive feedback cycle for evolution. As long as a single ISP was willing to deploy IPvN, application developers could have a large market for IPvN-aware software, and this in turn would create more demand for IPvN. A virtuous cycle between application demand (for IPvN-aware software) and service demand (for IPvN deployment) could lead to rapid deployment.

The flip side of this scenario was most clearly exemplified in the attempted deployment of IP Multicast. There are myriad reasons why multicast, despite being supported by most major router vendors, was never deployed at scale. One was, we believe, the lack of universal access. Even had a major ISP (say Sprint) deployed multicast, this new functionality would only have been available to Sprint's customers. Application developers on the other hand (*e.g.*, content providers such as CNN), were reluctant to develop multicast applications that could only service a fraction of Internet users. This led to a chicken-and-egg scenario where ISPs were reluctant to deploy a service for which there was no apparent application demand, while application developers were reluctant to rely on a service that was not universally available. If instead, *any* endhost had been able to access Sprint's multicast services, then application developers might have been more willing to experiment with the service. Thus, requiring universal access of a partially deployed service fosters demand and encourages independent innovation (by both ISPs and applications).

Given partial deployment, the need for universal access leads to our second technical requirement – *redirection*. To tolerate partial IPvN deployment (A1) and participation (A2), IPvN packets leaving a host must find their way to the virtual IPvN network whether or not their local ISP supports IPvN (A1) or supports the configuration needed to redirect packets to IPvN domains (A2). While redirection is always a requirement for overlay networks, here we require that redirection cannot be through application-specific or manual configuration on the host. Manual configuration is outside the capabilities of most Internet users, and it would be doubly difficult to configure if one's own ISP is not willing to assist in doing so. Thus, for evolution, the redirection challenge is not how to intercept packets, which is a significant issue itself (though there has been recent progress in general techniques to do this [27]), but knowing where to redirect them to. This problem has not received much attention, but it turns out to be crucial to our story. Moreover, incentive issues, rather than technical ones, are the dominant factor.

Note that enabling universal access leads to a balance between the competing needs of user choice and ISP control. Users are free to choose whether or not to use IPvN (which drives ISP competition), but the operation of the IPvN virtual network and the process of redirecting IPvN packets is left under the control of ISPs. A further tilt to this balance would be to offer users the choice of which IPvN service provider their IPvN packets are redirected to. We do not explore this option in detail but note that the technical framework we describe in the following sections could, with few modifications, be adapted to such scenarios.

There are two obvious approaches to redirection: application-level controlled by third-party brokers and network-level controlled by ISPs. We discuss each in turn.

2.2 Application-Level Redirection

For application-level redirection, one might have a lookup service that tracks the state of IPvN deployment in terms of which ISP domains and/or routers support IPvN. When queried by an endhost, the lookup service would return an IP address for a nearby IPvN router. The client can then tunnel IPvN packets to that router, which injects them into the IPvN overlay.

The crucial question is: who runs this lookup service? In one scenario, this would be offered by ISPs themselves; *i.e.*, ISPs could exchange deployment information with each other and then each provide such a lookup service. However, in that case, non-offering ISPs would have little incentive to provide such a lookup service (assumptions A1 and A2) and hence a customer of a non-offering ISP would have to use the lookup service of another ISP. This violates our assumption that customers enter no new contractual agreements. This is also technically difficult because, in the absence of any additional redirection services, it would require endusers to know which ISPs offered such lookup services (at any point in time, they would have to know which ISPs were offering IPvN) and which offering ISP is best suited to serving the enduser in terms of network proximity. Thus, relying on ISPs to provide a lookup service would likely interfere with universal access.

Another possibility is that the lookup service could be run by third-party brokers who gather deployment information from each of the ISPs. At a technical level, this would be consistent with universal access since any client could reach such brokers. This option however alters the current market structure and, in terms of incentives, is riddled with open questions. Chief among these is that under this arrangement, brokers become a crucial component of the infrastructure, significantly influencing the routing of IPvN packets, and hence it isn't clear whether and why ISPs would be willing to relinquish control to such brokers. At the same time, third party-brokers are dependent on ISPs for the deployment information needed to effect redirection. One (positive) possibility here is that if some ISPs did enter agreements with brokers, the rest would have to follow to compete. Yet another question is who would pay these brokers: they could be paid by ISPs to direct traffic to them, or by customers to provide good referrals, or both. This in turn leads to the question of how one would ensure competition at the broker level. Presumably there would be multiple brokers that would likely peer with each other, or only provide partial visibility into the IPvN overlay.

In summary, both the above options for application-level redirection are less than satisfactory: ISP-based redirection is difficult in the event of partial participation requiring manual configuration by endusers, while redirection by third-party brokers upsets the current market structure. This leads us to explore an alternate option, namely that of network-level support for IPvN redirection.

2.3 Network-level Redirection

Network-level redirection involves no lookup to find a nearby IPvN router. Instead, every router in the network (whether IPvN or not) is equipped with the knowledge needed to forward an IPvN packet towards an IPvN router; *i.e.*, the network naturally routes IPvN packets to an appropriate destination. Ignoring the technical difficulties for a moment (we discuss them in detail in the following section), this has the nice property that it works within the current market structure. Such redirection is under the shared control of ISPs and, as we discuss later, doesn't involve establishing new brokers or even making substantial changes in routing. While it may not have the full flexibility that a lookup service could have, it would thus be easier to achieve with incremental changes. The main problem is this: if we allow redirection to be done by ISP routing, how can a client in a non-offering ISP be guaranteed access to IPvN? Can't the ISP block such access through its routing algorithms? That is the question we address in the next section.

3. MECHANISMS FOR EVOLVABILITY

The previous section argued for network-level redirection as a vital primitive in supporting the evolvability of a multi-provider network infrastructure. In this section, we propose the use of IP Anycast as a candidate mechanism by which to effect this network-level redirection. We show that IP Anycast is both well-suited to the task and a practical choice as support for anycast routing can be deployed with little-to-no change in today's routing infrastructure. We discuss anycast and its deployment scenarios in Section 3.1.

The second required component identified in Section 2 is the set of mechanisms used in the construction and maintenance of multi-provider virtual networks (vN-Bones). We explore candidate solutions for this in Section 3.3. Finally, Section 3.4 describes how packet forwarding is implemented through the combination of IP anycast and vN-Bone tunneling.

We stress that there is remarkably little innovation in the details of the technical discussion that follows. Rather, our contribution is one of synthesis – in identifying the necessary components and piecing them together to construct a plausible scenario of IP evolvability. At the same time, the incremental nature of the individual pieces lends hope that our proposal for evolvability is practical and within grasp of real deployment.

3.1 IP Anycast as Network-level Redirector

RFC 1546 [28] defines anycasting as a network service in which a host transmits a datagram to an anycast address and the network is responsible for providing best effort delivery of the datagram to one of possibly multiple servers that accept datagrams for that anycast address. Typically, a datagram will be delivered to the server closest to the client host where "closest" is defined in terms of the network's measure of routing distance.

Since its proposal in 1993, IP Anycast has been deployed within individual domains primarily for service discovery (*e.g.*, to locate rendezvous points in PIM-SM [29]) and on a global scale for the robust implementation of root DNS name servers RFC3258 [30].

Here, we propose the use of IP Anycast as the mechanism for network-level redirection in support of the deployment of successive generations of IP. As described in Section 2, in a network where the ubiquitously supported IP service is (say) IPv4, and the next generation IP being deployed is (say) IPv8, network-level redirection is needed to steer IPv8 packets towards IPv8 routers. Using anycast, this is easily achieved as follows: a well-known IPv4 anycast address A_4 is assigned to the deployment of IPv8 and every IPv8 router accepts delivery of packets destined to A_4. To use IPv8, *any* endhost can simply encapsulate an IPv8 packet in an IPv4 packet with destination A_4. Anycast ensures this packet will be delivered to the closest IPv8 router, from which point on the packet is in the hands of IPv8 routers.

Our choice of anycast stems from two key reasons –

1. the abstraction of an anycast address enables the *seamless spread* of deployment. By this we mean that the use of an anycast address allows endhosts and (if desired) ISPs to remain ignorant of the state of IPv8 deployment in terms of which ISP domains/routers currently support IPv8. Were this not the case, then deployment by one ISP could trigger widespread reconfiguration at possibly remote endhosts.[1]

2. because anycast reuses the existing unicast routing infrastructure, it inherits the highly decentralized control structure of IP routing and hence individual ISPs can independently configure and control the redirection process.

[1] In fact, it isn't clear how an ISP might, without global knowledge, even *identify* the set of endhosts that need reconfiguration.

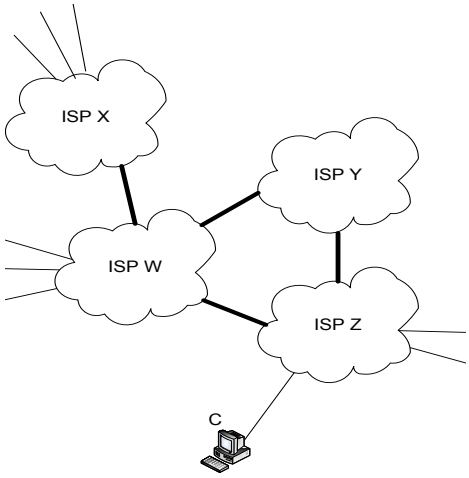

Figure 1: ***Anycast enables the seamless spread of deployment: IPv8 is deployed successively in ISPs X, then Y and finally Z. Throughout, client C is seamlessly redirected to the closest IPv8 provider.***

To see this, consider the deployment of IPv8 in Figure 1. Initially, provider X is the only ISP to deploy IPv8 and hence IPv8 packets from client C, with local ISP Z, are routed through X's domain. However, once ISP Y deploys IPv8, C's packets are routed through Y instead of X and finally, when Z deploys IPv8, C's IPv8 packets are handled by its own local ISP. Note that throughout this deployment process, client C's packets are seamlessly redirected through the appropriate providers without requiring any form of reconfiguration or indeed any awareness of the state of IPv8 deployment on the part of endhosts. The same is largely true for ISPs as well. The natural inter-domain routing protocols (with possible support for anycast, as described below) ensures that packets are routed to the appropriate provider without ISPs having to explicitly monitor, and adapt to, the global state of IPv8 deployment.

Moreover, note that despite its seamless nature, ISPs can, to some extent, control the process of redirection through policy choices in their inter-domain routing. For example, in Figure 1, ISP W might, based on peering policies, choose to route anycast packets to ISP X before Y. As with regular inter-domain routing, this control is both partial (*i.e.*, shared across ISPs) and implemented in a decentralized manner.

Thus, unlike the case of application-level redirection described in Section 2, here the network, in a completely decentralized manner, "self-manages" redirection without the need for higher-layer, third party brokers that track and/or control the global state of deployment.

In summary, anycast implements network-level redirection in a manner that addresses the issues of incentives raised in Section 2. Specifically:

- universal access is achieved even under partial deployment
- the existing ISP control structure is preserved
- operation is not dependent on all ISPs participating (we discuss anycast routing with partial ISP participation below)
- through peering policies, ISPs can control but not gate deployment
- as with regular unicast routing, control is decentralized and shared across ISPs

The above discussion assumes the existence of a global IP Anycast service. Unfortunately, anycast deployment today is typically limited to individual domains. In what follows, we discuss two options to deploying a global IP anycast service. In keeping with our goals, we would rather not assume that all ISPs will participate either in the deployment of a new generation of IP (IPvN), nor in our plan for evolvability. Hence while we assume that a participant ISP (by which we mean an ISP that is willing to deploy IPvN) will also be willing to deploy mechanisms to support anycast, we are unwilling to assume that all ISPs shall do so. The question then is what minimal degree of support for anycast can we require of non-participant ISPs. The two options we describe below differ slightly in this respect: our first option requires that a non-participant ISP be willing to propagate a small number of non-aggregatable (*i.e.*, with prefix longer than the /22 deemed acceptable for propagation in today's routing infrastructure) anycast addresses in its inter-domain routing protocol. Note that this is a change in *policy* on the part of an ISP and does not require the deployment of any new mechanism. Our second option requires no change (neither policy nor deployment) by non-participant ISPs but does result in a somewhat less "pure" form of anycast.

Before proceeding, we note that our discussion here implicitly promotes a somewhat stripped down anycast service model. Instead of the fully dynamic framework assumed in RFC 1546 where arbitrary hosts can join and leave anycast groups dynamically we envision a more statically provisioned framework where only configured hosts within the network infrastructure are members of an anycast group and ISPs explicitly control the allocation and advertisement of anycast addresses.

3.2 Anycast Routing

Intra-Domain. Standard intra-domain unicast routing algorithms, whether distance-vector or link-state, are naturally amenable to routing anycast. As described in [31], for link-state protocols such as OSPF, the only modification required is that IPvN routers also advertise a high-cost "link" to the corresponding anycast address. This high cost is necessary to prevent routers from attempting to route *through* an anycast address. Note that from these link state advertisements, an IPvN router can easily identify every other IPvN router within its domain. With distance-vector protocols such as RIP [32], anycast routing merely requires that an IPvN router advertise a distance of zero to its anycast address; standard distance-vector then ensures that every router will discover the next hop to its closest IPvN router. Note that here, unlike link-state routing, an IPvN router cannot easily identify other IPvN routers.

An alternate approach to both the above is simply to have an IPvN router indicate this in its standard unicast route advertisement by, for example, explicitly listing its anycast address. Because intra-domain routing algorithms build a complete (*i.e.*, non-aggregated) routing table, this makes anycast routing trivial – a router merely checks its unicast routing table for the closest anycast-addressed router. This involves a small modification to existing intra-domain routing algorithms but makes it trivial for IPvN routers to discover one another. As described in Section 3.3, this knowledge enables very simple intra-domain virtual topology construction.[2]

[2]While the remainder of this paper will assume that the intra-domain protocol does allow IPvN routers to discover one another we stress that this is merely a simplification and in no way a necessary requirement. In its absence (*i.e.*, for domains that use unmodified RIP [32]), the intra-domain virtual topology construction will merely have to implement some additional discovery mechanisms such as those described in Section 3.3.

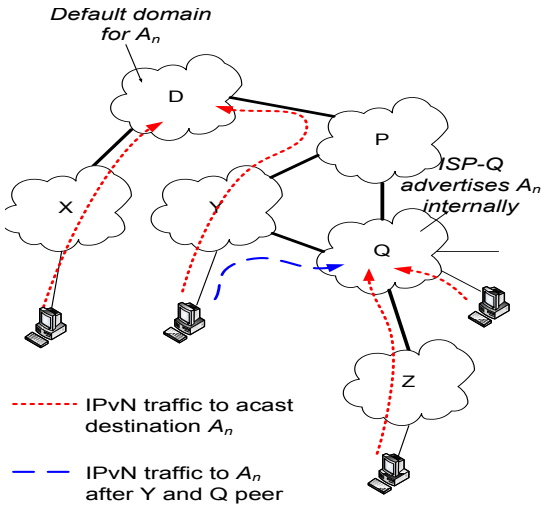

Figure 2: ***Inter-domain anycast routing using ISP-rooted unicast addresses and "default" routes.***

Inter-Domain, option 1: non-aggregatable addresses, global routes. One approach to supporting inter-domain anycast is to designate a portion of the regular unicast address space to serve as anycast addresses and require that ISPs propagate route advertisements for anycast addresses in their inter-domain routing protocols. This approach is certainly implementable even today – as suggested by [28,31], a designated portion of the unicast address space could be assigned to anycast and propagating these routes in BGP would require a change in policy but not mechanism on the part of ISPs – and yet there is, with one exception [30], little deployment of global IP anycast. One reason for this narrow adoption is concern over the scalability of such an approach, particularly under RFC1546's fully general and dynamic IP anycast service model. Anycast addresses, as described above, are not aggregatable and must hence be advertised individually by routing protocols and lead to routing state that grows in direct proportion to the number of anycast groups. However, for our proposed use of anycast, scalability is unlikely to be a concern. Recall that a single anycast address is needed to serve each new generation of IP. Given the cost and effort for an ISP to roll out a new generation of routers, we imagine that the number of simultaneous attempts to deploy different IP versions is likely to be very small (ideally one) and will not lead to a problematic growth in routing state. Moreover, unlike the more commonly advocated uses of anycast (server selection, *etc.*), here the consumers of anycast addresses are not arbitrary endusers but rather the ISPs themselves who have an incentive to use these addresses sparingly. To further ensure this, ISPs might even charge to route anycast.

Instead, our concern with this approach is that it requires that all ISPs eventually support the propagation of anycast routes. While this seems like a not unreasonable hope given that the only change required is a simple modification to policy, we would rather not rely on this assumption and hence explore alternate approaches.

Inter-Domain, option 2: aggregatable addresses, default routes. To address the poor scaling of traditional anycast architectures, Katabi *et al.* propose GIA [31]. In GIA, scalability is achieved by introducing the notion of a "home" ISP domain associated with an anycast group. GIA still allocates anycast its

own portion of the IP address space – all addresses prefixed by a well-known "Anycast Indicator" sequence of bits. However the remaining address bits are drawn from the unicast address space of the home domain. This allows for simple "default" routes; a router with no anycast routing entry for a given address can look up the home domain's prefix in its unicast routing table and forward the packet towards the home domain. GIA requires that the home domain include at least one member of the anycast group and hence this ensures the packet will reach a group member although not necessarily the closest. For more optimized anycast routes, Katabi *et al.* propose an extension to BGP whereby border routers can initiate searches for nearby members of an anycast group.

While GIA offers an elegant solution to scalable anycast, its deployment requires modifying the border routers at client domains. Given the current lack of deployment of GIA by ISPs, and to satisfy our stated required assumption of no global participation (Section 2), we present here an approach to anycast that requires no change by non-participant ISPs. We stress however that our proposal is somewhat motivated by expediency and open to eventual replacement by GIA (or a similar design) and/or the use of a limited number of non-aggregatable addresses as described above.

Our proposal, along the lines described in [33], is to avoid (at least for now) introducing a special type of anycast address and instead just reuse a piece of the existing unicast address space. We borrow the basic insight behind GIA and advocate that anycast addresses be allocated from the unicast address space of a "default" ISP (*e.g.*, the first ISP to initiate deployment of IPvN) and IPvN routers are configured to advertise the anycast address in their IGP as described earlier. Additional ISPs that adopt IPvN also configure their IPvN routers to advertise the same anycast address internally. Standard unicast routing will deliver anycast packets to the closest IPvN router along the path from the source to the default ISP. For example, in Figure 2, ISPs Q and D deploy IPvN and D is the default domain; anycast packets from domains X and Y terminate in domain D while those from Z reach Q. To widen their reach, non-default domains can peer with neighboring domains to advertise their anycast route. For example, in Figure 2, Q can peer with Y to advertise its path for the anycast address in question; Y's packets will then be delivered to Q rather than D.

Thus the final picture in our proposal is not unlike that using non-aggregatable addresses. The key difference is that our use of a default ISP allows us to transition to that final picture through the *optional and independent* participation of ISPs. Even with no cooperation from non-IPvN domains, the above scheme will route anycast correctly, although imperfectly in terms of proximity to, IPvN routers. Here, the use of inter-domain advertising is an optimization that leads to more improved anycasting. A potential failing of our approach is that the default provider owns the anycast address and receives a larger than normal share of IPvN traffic. Ideally though, this could incite other ISPs to pursue inter-domain advertising of anycast addresses.

Given its practicality, our discussion from here on will assume the the use of anycast addresses rooted in default ISPs.

3.3 vN-Bone Formation

Sections 3.1 and 3.2 described how IPvN packets are steered to IPvN routers. We now describe how these IPvN routers cooperate to form a "virtual" IPvN network, or vN-Bone, overlaid on an Internet where IPv(N-1) is ubiquitously deployed.

Before delving into the details of our mechanisms, we make two observations: the first is that, unlike the case of network redirection which must be ubiquitously supported (whether explicitly as in option 1 for inter-domain anycast, or implicitly as in option 2 for inter-domain anycast, as described in Section 3.2) by both participant and non-participant providers, vN-Bones are implemented entirely by participant ISPs and hence this design space is much less constrained. Indeed, many of the techniques from the literature on overlays and testbeds [18, 20, 21, 21, 23] could likely find use here and, as such, our proposals are best viewed as one set of candidate solutions. The second observation is that virtual networks that span multiple ISPs are not new. Networks like the MBone [25] and 6Bone were all pioneering efforts in this respect. These networks relied greatly on manual configuration and, while the solutions we present do automate much of the topology construction and maintenance process, we readily accept that many ISPs might, as in the past, simply choose to configure their networks by hand.

There are two main components to a virtual network:

1. virtual topology construction, and
2. routing over this virtual network

Note that because we do not assume ubiquitous deployment even within a participant ISP, each of the above must be addressed at both the intra and inter-domain level.

3.3.1 Topology Construction

The first component – vN-Bone construction – is fairly straightforward as it largely builds off the connectivity information revealed by the underlying IPv(N-1) routing protocols. For example, the IPv(N-1) intra-domain routing, whether link-state or distance-vector (and assuming the anycast extensions described in the previous section), ensures that every IPvN router has complete knowledge of the set of IPvN routers within its domain.[3] The intra-domain vN-Bone topology can then be constructed through simple rules such as: every IPvN router picks its k closest IPvN routers as neighbors on the vN-Bone. In the event that such rules leads to partitions, these can be easily detected and repaired because every router has complete knowledge of all other IPvN routers.

At the inter-domain level, the most likely scenario is that ISPs will set up inter-domain tunnels based on their peering policies with other ISPs. In the absence of such configuration, a newly joined ISP could reuse the anycast mechanism as the initial bootstrap by which to discover at least one other ISP that currently supports IPvN; having done this, the new ISP can discover additional neighbors through the inter-domain vN-Bone routing (described below).[4] In terms of preventing partitions of the inter-domain vN-Bone topology, one simple approach is for every domain to ensure that it is connected (either directly or indirectly) to the "default" provider of the anycast address.

Finally, as deployment spreads, the vN-Bone topology should evolve to be congruent with that of the underlying physical topology. This is easily achieved using the connectivity information revealed by the v(N-1) routing protocols at both the intra and inter-domain levels.

3.3.2 Routing in vN-Bones

[3] Recall our discussion in Section 3.2 about how such global knowledge can be achieved even in distance-vector protocols like RIP with one minor modification. In the absence of this modification, intra-domain vN-Bone construction over RIP would have to be implemented along the lines of the inter-domain vN-Bone construction; *i.e.*, through explicit neighbor discovery leveraging anycast for the initial bootstrap.

[4] Note that this use of anycast is only possible for a new ISP that isn't yet actively advertising the anycast. Otherwise, the anycast route would simply loop back to the initiator.

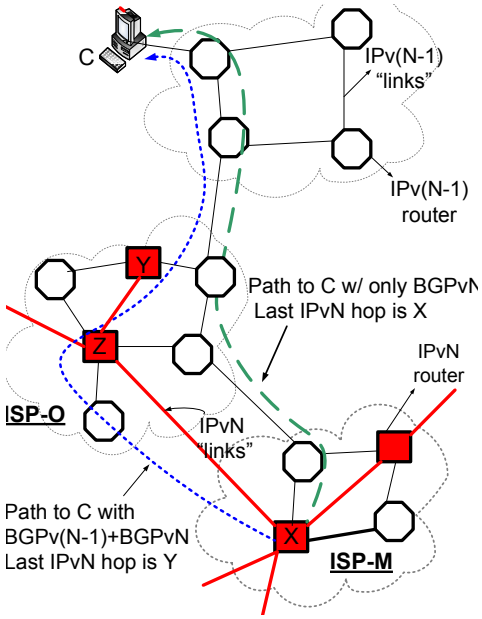

Figure 3: ***Inter-domain vN-Bone Routing: IPvN border routers must run BGPvN and, in addition, obtain BGPv(N-1) information from IPv(N-1) border routers.***

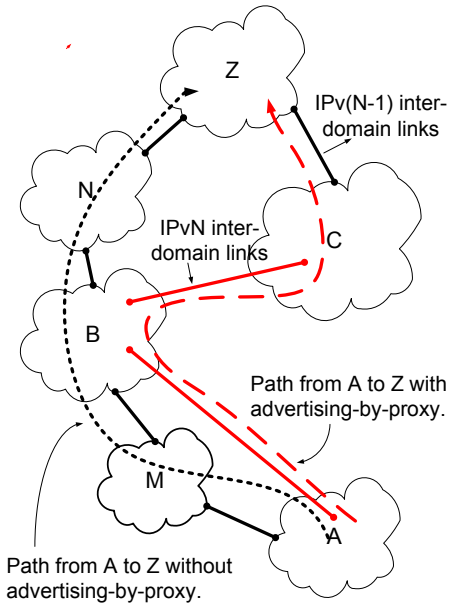

Figure 4: ***Inter-domain vN-Bone Routing: advertising-by-proxy; ISPs A, B and C support IPvN. ISPs M, N and Z support only IPv(N-1). Hence, B and C advertise their distance to Z into the BGPvN routing protocol.***

Addressing. The issue of routing is closely tied to that of host addressing. There are at least three aspects to addressing that to be considered: (1) the format or structure of addresses, (2) address allocation and, (3) advertising addresses into the routing fabric.

In today's Internet, the allocation and advertisement of an endhost's IPv4 address is handled by its local access provider. If future IPvN architectures adopt a similar model then supporting universal access raises the question of how an endhost might obtain an IPvN address if its access provider does not yet support IPvN. A possible solution, along the lines proposed in RFC 3056 [34], is to have the endhost assign itself a unique IPvN address. This can be done, for example, by using one address bit to indicate such "self addressing" and deriving the remaining IPvN address bits from the endhost's unique IPv(N-1) address. Note that these self-addresses are very likely temporary and such endhosts will have to relabel if and when their access providers do adopt IPvN. This leaves us then with the question of how such temporary IPvN addresses are advertised and routed on. We explore this question in detail in the discussion on routing that follows. Finally, we note that this need for self-addressing arises in the case where address assignment is handled by an endhost's local provider. More generally however, we place no particular constraints on the addressing structure or allocation policy a next-generation IPvN may adopt.

Routing. In considering routing on this virtual network, we have to do so at two levels:

- routing between IPvN routers on the vN-Bone, and
- routing between any two IPvN endhosts

The two issues are closely related – given the ability to route between IPvN routers, routing between two IPvN endhosts is primarily a question of how we find the appropriate ingress and egress IPvN routers for a given source and destination IPvN endhosts.

Note that most discussions of routing, whether application layer as in overlays, or at the IPv4 network layer, need not distinguish between the two cases above. The current network layer assumes that an endhost's ingress or egress router is simply its access router and hence this distinction in unnecessary. Unfortunately, our need for universal client access under partial IPvN deployment makes this assumption invalid (at the IPvN layer). Proposed overlay-based routing systems [9, 20] on the other hand, assume some form of higher-layer (*e.g.*, DNS) or out-of-band translation between an endhost identifier and its "attachment" point in the overlay. This translation can be invoked prior to communication between two endhosts and hence the issues of routing between endhosts is easily mapped to that of routing between two overlay routers. In our case however, there are a number of reasons why we might not want to make a similar assumption. First, endhosts are not assigned explicit attachment points in the vN-Bone. Moreover, an endhost might have different attachment points depending on the network location of the endhost it is communicating with and these attachment points will change as deployment spreads. Most importantly, this option raises issues similar to those with application-level redirection (Section 2.2) – given our self-imposed reluctance to assume the introduction of new services, it isn't clear who can effect this translation or how, because doing so would require intimate knowledge of the state of IPvN deployment.

Between routers:. The topology construction in Section 3.3.1 described the global vN-Bone as composed of intra-domain vN-Bone topologies interconnected by inter-domain tunnels. Given this topology, establishing routes between IPvN routers is achieved by IPvN routing protocols and will thus depend on the specifics of a particular IPvN. The space of possible routing solutions here is fairly unconstrained as the participant nodes are all IPvN routers. In the discussion that follows, we assume the existence of sepa-

rate intra and inter-domain IPvN routing protocols but assume no specific routing algorithm. For simplicity, we use the notation BGPvN to denote the IPvN inter-domain routing protocol even though BGPvN need not strictly resemble today's BGP.

Between endhosts:. We consider the problem of routing between endhosts in the face of partial IPvN deployment at the inter-domain level; the intra-domain case follows along similar lines.

The question of finding an appropriate ingress router is easily resolved by our use of anycast for redirection. An IPvN packet injected by a source host will, through anycast, find its way to an ingress IPvN router without special support or configuration by the endhost. Hence the main question is that of finding an appropriate egress IPvN router. If the destination is in an IPvN domain, this is simple. In this case, the destination has a routable IPvN address, its home domain advertises this address into the IPvN-Bone routing topology and hence all IPvN routers know how to forward the packet through to that address. In other words, in this case, routing is based entirely on the IPvN destination address, using the IPvN routing protocols.

The case where the destination IPvN client is not in an IPvN enabled domain is less obvious. Recall that such a client has an unadvertised, temporary IPvN address. One apparently simple option would be to have the IPvN client use anycast to locate a closeby IPvN router and have that router advertise the client's temporary IPvN address. An endhost would periodically repeat this process in order to adapt to spread in deployment as also to router and network dynamics. While simple, this is somewhat of a departure from existing norms for route advertisement. Endhosts today are not individually responsible for route advertisement nor do routers typically accept direct route advertisements from remote endhosts (or even remote routers); the security and policy implications of such a change are an open question. Moreover, this introduces a form of fate-sharing between an endhost and its route advertisement that, again, isn't common today. Finally, such routes for temporary IPvN addresses would be injected into the IPvN routing protocol and it isn't clear how this would constrain the design space for routing and addressing at the IPvN layer. For these reasons, in the remainder of this paper we do not adopt the above solution. Instead, we look for solutions that place the burden of locating an appropriate egress IPvN router on the IPvN routers themselves rather than on endhosts. Nonetheless, the simplicity of this anycast-based scheme is appealing and should be considered in the case of IPvNs where the above issues turn out to not be problematic.

A different option is to leverage the destination's IPv(N-1) address and IPv(N-1) routing information. The destination's IPv(N-1) address could either be inferred from its temporary IPvN address or might be carried in a separate option field in the IPvN header. Here again, there are multiple possible options.

The simplest option is to simply exit the vN-Bone and forward the packet directly to the destination's IPv(N-1) address. This however fails to fully exploit IPvN deployment. Consider for example, the scenario in Figure 3. Here, the IPvN border router X in ISP domain M does not contain an IPvN-level route to client C and hence would just deliver the packet directly to C over IPv(N-1) effectively "exiting" the vN-Bone at X. Ideally however, X could have forwarded the packet over the vN-Bone to Z in ISP-O and from there to Y at which point the packet would exit the vN-Bone and be tunneled to C. This would be possible were M's IPvN border routers aware of the IPv(N-1) domain-level path between ISP M and C's domain. This is easily achieved by having an IPvN border router acquire BGPv(N-1) routing tables from its domain's IPv(N-1) border router. In addition, the IPvN router must know the IPv(N-1) domain associated with the different IPvN border routers (in this case Z). This too can be easily achieved by having IPvN border routers add this information in their IPvN route advertisements.

Domain	BGPv(N-1) path	IPvN?	w/ adv-by-proxy
C	C:Z	yes	C:Z
N	N:Z	no	–
B	B:N:Z	yes	B:C:Z
M	M:B:N:Z	no	–
A	A:M:B:N:Z	yes	A:B:C:Z

Table 1: BGPv(N-1) and BGPvN routing entries corresponding to Figure 4 ***advertising by proxy leads A and B to route to Z through IPvN domain C***

The above is an improvement but does not ensure that all possible IPvN paths to the destination have been considered. For example, consider the scenario in Figure 4. For simplicity, we assume here that the BGPvN protocol uses a path-vector protocol similar to the current BGP. Here, Z is the non-IPvN destination domain, A, B and C are IPvN domains and M and O are non-IPvN domains. Each domain's BGPv(N-1) entries to Z and their BGPvN entries are shown in Table 1. With the routing we've developed this far, domain A is ignorant of the path from C to Z (because C does not lie on A's BGPv(N-1) path to Z) and hence will not route through C to Z. To rectify this we propose the following "advertise-by-proxy" BGPvN rule: an IPvN border router advertises an IPv(N-1) destination prefix if it is the only IPvN domain along the BGPv(N-1) path from itself to the destination domain. This can be done by having an IPvN router include a list of "on-behalf-of" IPv(N-1) domains in its IPvN inter-domain route advertisements. Note that this is actually a minor change to the above proposal – in addition to its own IPv(N-1) domain, an IPvN router merely adds a list of additional IPv(N-1) domains for which it serves as proxy. In making routing decisions, an IPvN router can now combine this IPv(N-1) level information with its IPvN information. In our example, one could simply add the number of (domain-level) hops. Hence, both B and C would inject route advertisements for Z with distances of 2 and 1 respectively and regular BGPvN inter-domain routing can then compute routes to Z as normal. Note that effectively, the routing "distance" in this case is the sum of the (1) BGPvN routing distance on the vN-Bone and, (2) the domain-level hops between the IPvN egress and the destination on the IPv(N-1) topology with ties broken (as in our example) to favor IPvN paths. While this metric is certainly open to adjustment, this would seem to achieve a good tradeoff between maximizing routing through IPvN domains, while avoiding excessively long routes in order to do so. Note that our example here was an easy case because the notion of routing distance at the BGPvN and BGPv(N-1) level were easily compatible. In general, the appropriate manner in which the two routing metrics might be combined would have to be determined based on the specific BGPvN and BGPv(N-1).

In summary, for a destination in an IPvN domain, routing is effected using its IPvN address and IPvN routing information. For an IPvN destination in an IPv(N-1) domain, routing is on the destination's IPv(N-1) address using a combination of IPvN and IPv(N-1) routing information. Thus (under this design!) the requirement on IPvN that allow smooth transitioning are: (1) hosts must be able to create temporary and unique IPvN addresses, (2) a temporary address should reveal the host's IPv(N-1) address or the IPvN header should allow that information to be carried, and (3) IPvN routers should be able to annotate their route advertisements with IPv(N-1) topology information.

Note that, unlike in typical overlays, our routing does not guar-

antee a unique egress point to a destination and nor should it. For example, routes from domain C to Z should exit the vN-Bone at C while those in B should exit at N. This can lead to route asymmetry because routes *from* C will always select the same ingress point (because the anycast mechanism will always select the closest IPvN router independent of the destination). Thus, while asymmetry is not unusual even today, our proposal is likely to exacerbate asymmetric routing in the early stages of deployment.

3.4 Forwarding

We now briefly review the end-to-end data path taken by a packet. Assume IPv(N-1) is the current ubiquitously deployed version of IP, IPvN is the next generation IP and all IPvN routers form a virtual vN-Bone. We use A_{n-1} to denote the IPv(N-1) anycast address assigned to the deployment of IPvN. Then, end-to-end forwarding of an IPvN packet works as follows:

- the source S encapsulates the IPvN packet in an IPv(N-1) header with destination A_{n-1}.
- using anycast, the packet is forwarded over legacy IPv(N-1) routers to the closest IPvN router, R1.
- R1 strips off the IPv(N-1) header, processes the packet as needed, looks up the next hop (R2) to the destination using the vN-Bone forwarding tables, and forwards the packet to R2, once again encapsulating the packet in an IPv(N-1) header if required.
- this is repeated until the packet reaches the egress IPvN router which tunnels the packet through to the destination.

In addition, the source, either through configuration or an ARP-like protocol, discovers whether its first hop router supports IPvN and, if so, does not encapsulate the packet. Similarly, every intermediate router will only invoke encapsulation if its next hop IPvN router is not an immediate (*i.e.,* physical layer) neighbor. Thus, as deployment spreads, the use of IPv(N-1) is gradually phased out.

3.5 Discussion

This section described a series of mechanisms that, taken together, provide a framework for transitioning between successive generations of IP. Supporting these mechanisms places additional demands on future generations of routers. Specifically, an IPvN router must: (1) participate in the IPv(N-1) unicast and anycast routing algorithms, (2) perform IPv(N-1) forwarding, (3) participate in the construction of the virtual vN-Bone network, (4) participate in IPvN unicast and anycast routing and finally, (5) perform IPvN forwarding.

Participation in IPv(N-1) and IPvN routing and forwarding seems unavoidable for any transition path and hence the specific additions here involve support for anycast routing and the construction of the virtual IPvN network which, as described here, do not seem unduly complex.

We stress two crucial features of our framework: the first is that we do not require that *all* routers support the above mechanisms; rather, to evolve from IPv(N-1) to IPvN, only IPvN routers need support the entire suite of mechanisms. This means that our framework for evolvability is not itself gated by issues of non-cooperation from ISPs with no interest in deploying IPvN. Second, we point out that our framework adheres to the general design style of the existing Internet with no per-client state within the network, no significant complexity on the packet forwarding path, decentralized control and so forth.

Unfortunately, our approach does not assist in the deployment of architectures that, by definition, require support from every router along the path. This includes certain QoS proposals though recent work on supporting such features in overlays might assist in this regard [19]. Also unclear, is whether the potential routing inefficiencies due to anycast (at least in the early stages of deployment), might diminish the usefulness of certain IPvN architectures. Of particular concern here would be architectures with the primary goal of improving performance properties such as path loss or delay. However, this is likely to be less of an issue for the (many) proposals that seek to add IP-level support for security, mobility, addressing, robustness, and so forth [2, 3, 7, 12, 13, 16, 17, 35–37].

Our discussion in this paper addresses how IPvN packets are delivered between two IPvN endhosts across networks with limited support for IPvN. A related open question is whether general guidelines exist for how an IPvN endhost may inter-operate with an IPv(N-1) endhost over IPvN. This requires support from the endpoints and, depending on the specifics of an IPvN, may range in complexity from simple header translation to more complex translations of even higher-layer protocols. Depending on their complexity, such translations might be effected by simple NAT-like functionality in IPvN routers, client-side proxies or special IPvN transition boxes. A detailed discussion on endhost interoperability is beyond the scope of this paper; for a relevant discussion, we refer the interested reader to the work on OCALA [27], a proxy-based solution for supporting legacy applications over overlay networks.

Finally, it is worth pondering what, in the larger network architecture, must remain "invariant" in the sense of functionality that must be retained through successive architectural generations to ensure continued evolvability. Examination of our mechanisms would suggest just two: support for global unicast and anycast routing.

4. DEPLOYING SOURCE-SPECIFIC MULTICAST

The previous section presented an overall framework for evolvability based on the use of IP Anycast. In this section, we take IP Multicast as an example of a new IP service and work through its deployment under this framework. In so doing, we quite deliberately do not attempt to innovate on the details of the multicast protocols themselves; instead we take existing standards and describe how our framework might support their deployment.

We focus our discussion on the deployment of source-specific multicast. A detailed description of deploying any-source multicast which uses a somewhat larger suite of protocols (IGMP, MSDP, MBGP, PIM-SM and PIM-DM), while we believe would follow along similar lines, is beyond the most masochistic tendencies of the authors.

Source Specific Multicast. Source Specific Multicast (SSM), a restricted form of the more general IP Multicast service [38], provides one-to-many packet delivery between a designated source node and zero or more receivers [39]. As defined by RFC 3569 [40] and Holbrook [29, 41], source-specific multicast is implemented through the combined use of the Internet Gateway Multicast Protocol (IGMP) [42] and a reduced form of Sparse Mode PIM (Protocol Independent Multicast), denoted PIM-SSM. Through IGMP, a Designated Router (DR) on a local network tracks group membership on each of its network interfaces and participates in the wide-area multicast routing on behalf of the endhosts on its network. PIM-SSM is then used to construct a tree rooted at the source DR to all receivers' DRs. For simplicity, we use endhosts to mean their DRs and focus only on the mechanics of the wide-area routing.

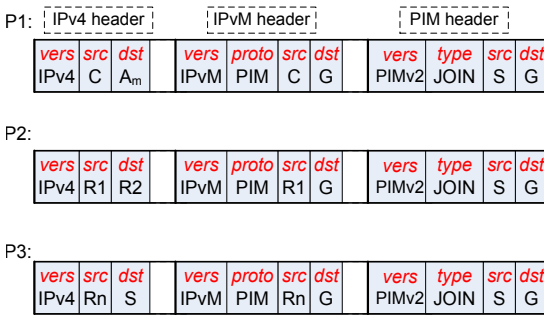

Figure 5: ***Deploying Source-Specific Multicast using IP Anycast encapsulation. Only relevant header fields are shown.***

In SSM, a multicast group, called a *channel*, is defined by the combination (S,G) of a multicast group address (G) and the unicast address (S) of the source. The SSM receiver interface supports joining and leaving a channel (`subscribe(S,G)`, `unsubscribe(S,G)`). A `subscribe` results in a JOIN message being routed toward S, setting up routing state for the new receiver at every point along the path until the JOIN messages hits a router on the distribution tree. `unsubscribe` operations trigger PRUNE messages that tear down routing state in a similar manner. To multicast to the group, packets from the source are forwarded down the distribution tree using reverse-path forwarding.

We now detail how SSM would operate using our framework from the previous section. We assume IPv4 is the ubiquitously deployed IP and use IPvM to denote a next generation IP that supports PIM-SSM. A_m denotes the anycast address allocated for the deployment of IPvM and vM-Bone the IPvM virtual topology. Note that a single vM-Bone is reused by all multicast groups.

- To use IPvM, a client C first checks whether its local ISP supports IPvM. If not, then all IPvM packets will be encapsulated in IPv4 packets as described below. Otherwise, IPvM packets are transmitted natively. In this example, we assume C's access provider does not support IPvM.
- To join channel (S,G) client C transmits a JOIN message shown as packet (P1) in Figure 5. Note that in practice, the source address denoted as *C* would be the IP address of C's Designated Router rather than C itself. Anycast routing delivers this PIM JOIN message to R1, the IPvM closest to C.
- R1 strips off the outer IPv4 header, and adds (G,S,*C*) to its multicast forwarding table. If this is R1's first multicast routing entry for (G,S) then R1 looks up the next hop, say R2, to destination *S* on the vM-Bone, encapsulates the packet as (P2) in Figure 5 and unicasts P2 to R2. Otherwise, the JOIN operation is terminated since R1 is already on the delivery tree for (G,S). Once again, if R1 and R2 are immediate neighbors then encapsulation is bypassed.
- The above is repeated until the packet hits an IPvM router already on the distribution tree for (G,S) or the egress IPvM router Rn in which case Rn unicasts the packet P3 from Figure 5 to S which sets up state (G,S,Rn). Again, in practice this state is stored at S's DR and not S itself. In fact, S would have no knowledge of the membership of G. Note that S's DR must know to decapsulate the packet. However as described in [RFC 2326], support for encapsulation is already required of DRs to handle packets tunneled to and from Rendezvous Points.
- Finally, to multicast to group G, S transmits a data packet to group G. The packet is picked up by S's Designated Router and forwarded through the (S,G) tree constructed as above.

The point to note in the above exercise is the manner in which our general framework enables SSM to be universally accessible (*i.e.*, to all endhosts) despite partial deployment of IPvM and support for PIM-SSM. At every step in the tree construction and multicast forwarding, IPvM and PIM-SSM can run "natively" if possible and, if not, the general techniques of tunneling and anycast forwarding bridge the gap to the next island of IPvM support.

5. RELATED WORK

The literature abounds with proposals for architectural fixes and/or altogether new architectures. These identify, and offer solutions to, a number of problems that plague the current architecture – security [2, 3, 17], routing [43], mobility [7, 35], intermittent connectivity [16], quality-of-service [6], congestion control [8, 44, 45], naming [36, 46, 47], addressing [13, 37], robustness [12] – to name a few. Like us, all the above proposals choose to work within the framework of improving current ISP infrastructure. However, the question of evolvability, in the sense of repeated architectural transitions and the economic and technical issues it raises, has typically not been the focus of these proposals. While many discuss the incremental deployment of their solutions, they address the technical but not economic issues of partial deployment. Moreover, they do not address how one might eventually (and generally) move beyond it to the next architecture. This paper, by contrast, does not address the larger question of what an ideal next generation architecture might look like, focusing instead only on how one might transition between architectures. At the same time, our exploration is needed because the lack of evolvability is precisely what has traditionally hampered the adoption of proposals such as the above.

As mentioned in Section 1, many have used overlays to tackle common architectural problems [18–20, 22, 48–51]. These offer tractable application-layer solutions to pressing problems but are limited to effecting change at the endpoints of an Internet path and cannot evolve IP itself. We note that one might debate what functions merit IP-level support and the extent to which these can be approximated by application-level solutions. This is a useful and necessary debate but, to extrapolate from this debate to the conclusion that one should abandon, for all time, any attempt at innovation in the IP infrastructure seems premature. Such innovation is what this paper seeks to enable.

Peterson *et al.* [1], in recent work, are probably the first to explicitly call out evolvability as a major architectural challenge and put forth a concrete proposal (based on network virtualization) to address this challenge. As described in Section 1, our proposals differ on the fundamentals of how we envision evolution will occur. While they view evolution as occurring through a succession of "revolutions" in which a new generation of service providers undermine, and eventually replace, incumbent providers, we see evolution as a gradual process led by incumbent ISPs incented to evolve. We believe both approaches merit exploration. Finally, this paper works through many of the lower-level details (*e.g.*, topology construction and routing on a virtual network, who implements redirection and how, *etc.*) not addressed in [1].

Many of the architectural themes that run through this paper can be traced back to the general discussions on Internet architecture

by Clark *et al.*. For example, our discussion in Section 2 on the conflict between user choice and ISP control, is reminiscent of the discussion of "tussles" in cyberspace [52], while [53] articulates the need to "take explicit, architected action to preserve the ability to change, evolve and advance (network) technology". Of course, whether our proposal represents the right embodiment of the discussions in [11, 52, 53] is entirely debatable! We look forward to such debates in future work and instead present this paper as, at the very least, offering a concrete on-the-table proposal as a starting point for future discussions.

The technical framework for evolvability described in Section 3.1 has two main components, the use of anycast-based redirection and the construction of, and routing over, multi-provider virtual networks. RFC 1546 [28] first proposed the concept of an anycast service but does not address its implementation. As mentioned, some of our proposals for anycast routing are inspired by, but somewhat different from, GIA [31], a proposal for scalable IP anycast by Katabi *et al*. In recent work, Ballani *et al*. [54] propose PIAS, a proxy-based approach to deploying a global anycast service. PIAS proposes using a hybrid of native IP anycast (to reach the proxy service) and overlay routing (to implement anycast between proxies). While using a service such as PIAS is certainly a possibility that we intend to explore in the future, we chose not to for the immediate future due to the current lack of deployment of a PIAS service and also as it isn't readily apparent how ISPs might control the operation of an infrastructure such as PIAS.

Most relevant to our discussion on vN-Bones, is the work on testbeds such as the MBone and XBone [25, 26, 55]. Our proposal for vN-Bones, is identical in spirit and differs only in some of our proposals to automate much of the topology and route construction. Also relevant here is the large body of work on overlay networks [18, 20, 21, 56, 57]. While we could (and do) leverage many of their ideas, vN-Bones differ from the majority of overlays in two important aspects: (1) vN-Bones are deployed and operated by multiple ISPs acting in concert and (2) vN-Bones operate at the router-level path between two endhosts in a manner that is completely transparent to the endhosts. These two differences lead to somewhat different needs and solutions to the problem of routing on the virtual network.

Not surprisingly, mechanisms similar to some discussed in this paper can be found in the vast body of work on assisting the deployment of IPv6 [34, 58–60]. Our contribution lies in relating such mechanisms to the question of incentives (specifically universal access) and unifying them into a cohesive and general plan for the deployment of future IPvN.

Finally, this section would be incomplete without paying tribute to the vision of active networks [15] whose authors were probably the first to tackle the need to enable innovation within the network infrastructure. However, as should be clear, our approaches are diametrically opposite. Active networks advocates new services being loaded into the infrastructure on demand and allows endhosts to define these new services. Instead, we rely on ISPs to deploy new IP functionality and limit endhosts to choosing between ISP offerings – a much more limited and hence more tractable approach to supporting innovation in the network infrastructure.

6. DISCUSSION

The inability to evolve IP, which lies at the very core of the Internet architecture, has long vexed the research community. Overlays, by either circumventing or undermining the control of ISPs, offer one solution to evolving today's architecture. This paper explored an alternate approach. Rather than achieve evolution by overhauling the administrative and operational structure of today's Internet, we look for what is missing from today's architecture that would make evolution by its *incumbent* operators economically desirable and technically feasible. A difference, in some sense, between evolving a network and architecting a network for evolvability.

We set out on our exploration fully expecting to discover that achieving evolvability would require a dramatic re-architecting of today's network, rendering our study into a mostly academic thought exercise. Instead, we found that our current architecture is largely evolvable as is. (A testament to the wisdom of the current architecture!). The one missing piece in the puzzle is widespread support for a global IP anycast service. This is good news because global support for anycast need not be a pipe dream; on the contrary deployment of anycast is well within reach even today.

Thus, we leave this paper with one concrete proposal for action – that it is worth resurrecting the case for a global IP anycast service.

While our proposed plan for evolvability does not strictly require global deployment of anycast, it would certainly be assisted by such deployment. Anycast has been proposed and discussed in the past but has never achieved significant momentum in terms of global deployment or wide-area evaluation. This was in part because its target usage was fairly narrow (DNS configuration, server selection, RP location) and partly because, often, similar functionality could be achieved through equivalent application-layer solutions (*e.g.*, Akamai redirection for server selection). This paper has argued that instead, IP anycast could be just the mechanism needed to evolve the Internet.

Finally, we point out that even though our discussion in this paper caters to ISPs and their retaining control over the Internet, it also lays the seeds for new entrants to enter the field as proposed in [1]. In particular, ISPs might choose to sell their willingness to route anycast to the third-party Next Generation Service Providers (NGSPs) proposed in [1]. NGSPs can then deploy their boxes in various ISP domains and use our framework for evolvability to stitch these together into a single global provider of a next-generation service. In fact, such an architecture would free NGSPs from having to deploy the client-side proxies that are otherwise needed to connect endhosts to the NGSP network.

7. REFERENCES

[1] Larry Peterson, Scott Shenker, and Jonathan Turner. Overcoming the Internet Impasse through Virtualization. In *Third Workshop on Hot Topics in Networks (HotNets-III)*, November 2004.

[2] T. Anderson, T. Roscoe, and D. Wetherall. Preventing Internet Denial-of-Service with Capabilities. In *2nd ACM Workshop on Hot Topics in Networks*, Cambridge, MA, November 2003.

[3] Adrian Perrig, Avi Yaar and Dawn Song. SIFF: A Stateless Internet Flow Filter to mitigate DDoS Flooding Attacks. In *Proceedings of the 2004 IEEE Symposium on Security and Privacy*. IEEE, 2004.

[4] Stefan Savage, David Wetherall, Anna Karlin, and Tom Anderson. Practical network support for ip traceback. In *Proceedings of SIGCOMM*, August 2000.

[5] Alex Snoeren *et al*. Hash-based IP Traceback. In *Proceedings of SIGCOMM*, August 2001.

[6] Ion Stoica, Scott Shenker, and Hui Zhang. Core-stateless fair queueing: Achieving approximately fair allocations in high speed networks. In *Proceedings of SIGCOMM*, September 1998.

[7] Alex Snoeren, Hari Balakrishnan, and Frans Kaashoek. Reconsidering internet mobility. In *Proceedings of the 8th IEEE Workshop on Hot Topics in Operating Systems (HotOS-VIII)*, Elmau/Oberbayern, Germany, May 2001.

[8] Dina Katabi, Mark Handley, and Charles Rohrs. Internet Congestion Control for High Bandwidth-Delay Product Networks. In *ACM SIGCOMM*, Pittsburgh, PA, August 2002.

[9] Ion Stoica, Dan Adkins, Shelley Zhuang, Scott Shenker, and Sonesh Surana. Internet indirection infrastructure. In *Proceedings of SIGCOMM*, August 2002.

[10] Michael Walfish, Jeremy Stribling, Maxwell Krohn, Hari Balakrishnan, Robert Morris, and Scott Shenker. Middleboxes no longer considered harmful. In *Proceedings of the USENIX Symposium on Operating System Design and Implementation*, December 2004.

[11] David Clark, Robert Braden, Aaron Falk, and Venkata Pingali. FARA: Reorganizing the addressing architecture. In *ACM SIGCOMM Workshop on Future Directions in Network Architecture*, Karlsruhe, Germany, August 2003.

[12] Mark Gritter and David R. Cheriton. TRIAD: A new next-generation Internet architecture. `http://www-dsg.stanford.edu/triad/`, July 2000.

[13] Paul Francis and Ramakrishna Gummadi. IPNL: A NAT-extended Internet architecture. In *ACM SIGCOMM*, San Diego, CA, August 2001.

[14] C. Tschudin and R. Gold. Network Pointers. In *1st ACM Workshop on Hot Topics in Networks*, Princeton, NJ, October 2002.

[15] David L. Tennenhouse and David J. Wetherall. Towards an active network architecture. *Computer Communication Review*, 26(2):5–18, April 1996.

[16] Kevin Fall. A delay tolerant networking architecture for challenged internets. In *Proceedings of SIGCOMM*, August 2003.

[17] Mark Handley and Adam Greenhalgh. Steps towards a DoS-Resistant Internet Architecture. In *ACM SIGCOMM Workshop on Future Directions in Network Architecture*, August 2004.

[18] Yang hua Chu, Sanjay Rao, and Hui Zhang. A case for end system multicast. In *Proceedings of SIGMETRICS*, CA, June 2000.

[19] L. Subramanian, Ion Stoica, Hari Balakrishnan, and Randy Katz. OverQoS: An Overlay Based Architecture for Enhancing Internet QoS. In *1st USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI '04)*, San Francisco, CA, March 2004.

[20] David Andersen, Hari Balakrishnan, Frans Kaashoek, and Robert Morris. Resilient overlay networks. In *Proceedings of the Symposium on Operating Systems Principles*, New York, NY, 2001. ACM.

[21] John Jannotti, David Gifford, Kirk Johnson, Frans Kaashoek, and James O'Toole. Overcast: Reliable multicasting with an overlay network. In *Proceedings of the Fourth Symposium on Operating Systems Design and Implementation*, San Diego, CA, October 2000.

[22] Limin Wang, Vivek Pai, and Larry Peterson. The effect of request redirection on cdn robustness. In *Proceedings of the Fifth Symposium on Operating Systems Design and Implementation*, December 2002.

[23] Larry Peterson, Tom Anderson, David Culler, and Timothy Roscoe. A blueprint for introducing disruptive technology into the internet. In *First Workshop on Hot Topics in Networks*, October 2002.

[24] Akamai. http://www.akamai.com.

[25] H. Eriksson. Mbone: The multicast backbone. *Communications of the ACM*, 37(8):54–60, 1994.

[26] Joe Touch and Steve Hotz. The X-Bone.

[27] D. Joseph, J. Kannan, A. Kubota, K. Lakshminarayanan, I.Stoica, and K. Wehrle. OCALA: An Architecture for Supporting Legacy Applications over Overlays. Technical Report UCB/CSD-05-1397, 2005.

[28] C. Partridge, T. Mendez, and W. Milliken. *Host Anycasting Service*. Requests for Comment, November 1993. RFC-1546.

[29] Bill Fenner, Mark Handley, Hugh Holbrook, and Isidor Kouvelas. Protocol internet multicast – sparse mode (pim-sm): Protocol specification, October 2003. Internet Draft (work in progress).

[30] T. Hardie. *Distributing Authoritative Name Servers via Shared Unicast Addresses*. Requests for Comment, April 2002. RFC-3258.

[31] Dina Katabi and John Wroclawski. A framework for scalable global IP Anycast. In *Proceedings of SIGCOMM*, Sweden, 2000.

[32] G. Malkin. *Routing Information Protocol RIP version 2*. Internet Engineering Task Force, November 1998. RFC-2453

[33] Steven McCanne and William Destein. Proximity-based redirection system for robust and scalable service-node location in an internetwork, December 1999. United States Patent.

[34] B. Carpenter and K. Moore. *Connection of IPv6 Domains via IPv4 Clouds*. Requests for Comment, February 2001. RFC-3056.

[35] Hari Balakrishnan, Srinivasan Seshan, Elan Amir, and Randy H. Katz. Improving TCP/IP performance over wireless networks. In *Proceedings of 1st ACM Conf. on Mobile Computing and Networking (MOBICOM)*, Berkeley, CA, Nov 1995. ACM.

[36] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6), December 1998. RFC 2460.

[37] R. Moskowitz and P. Nikander. Host identity protocol architecture, Sep 2003. IETF draft (work in progress).

[38] Stephen E. Deering. *Multicast Routing in a Datagram Internetwork*. PhD thesis, Stanford University, December 1991.

[39] Hugh Holbrook and David Cheriton. Ip multicast channels: Express support for single-source multicast applications. In *Proceedings of SIGCOMM '99*, Cambridge, MA, September 1999.

[40] S. Bhattacharyya. *An Overview of Source-Specifi c Mulitcast (SSM)*. Requests for Comment, July 2003. RFC-3569.

[41] H. Holbrook and B. Cain. Source specific multicast for ip. Internet Draft. (work in progress).

[42] W. Fenner. *Internet Group Management Protocol, Version 2*. Internet Engineering Task Force, Inter-Domain Multicast Routing Working Group, February 1996. Internet Draft (work in progress).

[43] Xiaowei Yang. NIRA: A new Internet routing architecture. In *ACM SIGCOMM Workshop on Future Directions in Network Architecture*, Germany, August 2003.

[44] Amit Jain and Sally Floyd. Quick-Start for tcp and ip, September 2004. Internet Draft, (work in progress).

[45] Sally Floyd and Van Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, August 1993.

[46] G. Ballintijn, M. van Steen, and A. S. Tanenbaum. Scalable user-friendly resource names. *IEEE Internet Computing*, 2001.

[47] Hari Balakrishnan, Karthik Lakshminarayanan, Sylvia Ratnasamy, Scott Shenker, Ion Stoica, and Michael Walfish. A Layered Naming Architecture for the Internet. In *Proceedings of SIGCOMM*, Portland, OR, September 2004.

[48] David G. Andersen. Mayday: Distributed filtering for Internet Services. In *4rd USENIX Symposium on Internet Technologies and Systems (USITS '03)*, Seattle, WA, March 2003.

[49] A. D. Keromytis, V. Misra, and D. Rubenstein. SOS: Secure overlay services. In *ACM SIGCOMM*, Pittsburgh, PA, August 2002.

[50] M. Walfish, H. Balakrishnan, and S. Shenker. Untangling the Web from DNS. In *1st USENIX/ACM Symposium on Networked Systems Design and Implementation*, San Francisco, CA, March 2004.

[51] Daniel Adkins, Karthik Lakshminarayanan, Adrian Perrig, and Ion Stoica. Taming IP packet flooding attacks. In *2nd ACM Workshop on Hot Topics in Networks*, Cambridge, MA, November 2003.

[52] David D. Clark, John Wroclawski, Karen R. Sollins, and Robert Braden. Tussle in cyberspace: defining tomorrow's Internet. In *ACM SIGCOMM*, Pittsburgh, PA, August 2002.

[53] David Clark, Karen Sollins, John Wroclawski, and Ted Faber. Addressing reality: An architectural response to demands on the evolving Internet. In *ACM SIGCOMM Workshop on Future Directions in Network Architecture*, Germany, August 2003.

[54] Hitesh Ballani and Paul Francis. Towards a deployable ip anycast service. In *First Workshop on Real Large Distributed Systems (WORLDS)*, December 2004.

[55] Joe Touch, Y. Wang, L. Eggert, and G. Finn. Virtual Internet Architecture. In *ACM SIGCOMM Workshop on Future Directions in Network Architecture*, Karlsruhe, Germany, August 2003.

[56] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A Scalable Content-Addressable Network. In *Proceedings of SIGCOMM*, August 2001.

[57] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of SIGCOMM*, August 2001.

[58] A. Durand, P. Fasano, I. Guardini, and D. Lento. *IPv6 Tunnel Broker*. Requests for Comment, January 2001. RFC-3053.

[59] C. Huitema. *An Anycast Prefi x for 6to4 Relay Routers*. Requests for Comment, June 2001. RFC-3068.

[60] R. Gilligan and E. Nordmark. *Transition Mechanisms for IPv6 Hosts and Routers*. Requests for Comment, April 1996. RFC-1933.