

JOE ABLEY

fear and loathing in the routing system



Joe Abley is the Director of Operations at Afiliias Canada, a DNS registry company, and a technical volunteer at Internet Systems Consortium. He likes his coffee short, strong, and black and is profoundly wary of outdoor temperatures that exceed 20°C.

jabley@ca.afiliias.info

ANYCAST IS A STRANGE ANIMAL. IN some circles the merest mention of the word can leave you drenched in bile; in others it's an overused buzzword which triggers involuntary rolling of the eyes. It's a technique, or perhaps a tool, or maybe a revolting subversion of all that is good in the world. It is "here to stay." It is by turns "useful" and "harmful"; it "improves service stability," "protects against denial-of-service attacks," and "is fundamentally incompatible with any service that uses TCP."

That a dry and, frankly, relatively trivial routing trick could engender this degree of emotional outpouring will be unsurprising to those who have worked in systems or network engineering roles for longer than about six minutes. The violently divergent opinions are an indication that context matters with anycast more than might be immediately apparent, and since anycast presents a very general solution to a large and varied set of potential problems, this is perhaps to be expected.

The trick to understanding anycast is to concentrate less on the "how" and far more on the "why" and "when." But before we get to that, let's start with a brief primer. Those who are feeling a need to roll their eyes already can go and wait outside. I'll call you when this bit is done.

Nuts and Bolts

Think of a network service which is bound to a greater extent than you'd quite like to an IP address rather than a name. DNS and NTP servers are good examples, if you're struggling to paint the mental image. Renumbering servers is an irritating process at the best of times, but if your clients almost always make reference to those servers using hard-coded IP addresses instead of names, the pain is far greater.

Before the average administrator has acquired even a small handful of battle scars from dealing with such services, it's fairly common for the services to be detached from the physical servers that house them. If you can point NTP traffic for 204.152.184.72 at any server you feel like, moving the corresponding service around as individual servers come and go becomes trivially easy. The IP address in this case becomes an identifier, like a DNS name, detached from the address of the server

that happens to be running the server processes on this particular afternoon. With this separation between service address and server address, a smooth transition of this NTP service from server A to server B within the same network is possible with minimal downtime to clients. The steps are:

1. Make sure the service running on both servers is identical. In the case of an NTP service, that means that both machines are running appropriate NTP software and that their clocks are properly synchronized.
2. Add a route to send traffic with destination address 204.152.184.72 toward server B.
3. Remove the route that is sending traffic toward server A.

Ta-da! Transition complete. Clients didn't notice. No need for a maintenance window. Knowing smiles and thoughtful nodding all round.

To understand how this has any relevance to the subject at hand, let's insert another step into this process:

- 2.5. Become distracted by a particularly inflammatory slashdot comment, spend the rest of the day grumbling about the lamentable state of the server budget for Q4, and leave the office at 11 p.m. as usual, forgetting all about step 3.

The curious result here is that the end result might very well be the same: Clients didn't notice. There is no real need for a maintenance window. What's more, we can now remove either one of those static routes and turn off the corresponding server, and clients *still* won't notice. We have distributed the NTP service across two origin servers using anycast. And we didn't even break a sweat!

Why does this work? Well, a query packet sent to a destination address arrives at a server which is configured to accept and process that query, and the server answers. Each server is configured to reply, and the source address used each time is the service address. The fact that there is more than one server available doesn't actually matter. To the client (and, in fact, to each server), it looks like there is only one server. The query-response behavior is exactly as it was without anycast on the client and on the server. The only difference is that the routing system has more than one choice about toward which server to send the request packet.

(To those in the audience who are getting a little agitated about my use of a stateless, single-packet exchange as an example here, there is no need to fret. I'll be pointing out the flies in the ointment very soon.)

The ability to remove a dependency on a single server for a service is very attractive to most system administrators, since once the coupling between service and server has been loosened, intrusive server maintenance without notice (and within normal working hours) suddenly becomes a distinct possibility. Adding extra server capacity during times of high service traffic without downtime is a useful capability, as is the ability to add additional servers.

For these kinds of transitions to be automatic, the interaction between the routing system and the servers needs to be dynamic: that is, a server needs to be able to tell the routing system when it is ready to receive traffic destined for a particular service, and correspondingly it also needs to be able to tell the routing system when that traffic should stop. This signaling can be made to work directly between a server and a router using standard routing protocols, as described in ISC-TN-2004-1 [1] (also presented at USENIX '04 [2]). This approach can also be combined with load balancers (sometimes

called “layer-4 switches”) if the idea of servers participating in routing protocols directly is distasteful for local policy reasons.

This technique can be used to build a cluster of servers in a single location to provide a particular service, or to distribute a service across servers that are widely distributed throughout your network, or both. With a little extra attention paid to addressing, it can also be used to distribute a single service around the Internet, as described in ISC-TN-2003-1 [3].

Anycast Marketing

Some of the benefits to the system administrator of distributing a service using anycast have already been mentioned. However, making the lives of system administrators easier rarely tops anybody’s quarterly objectives, much as you might wish otherwise. If anycast doesn’t make the service better in some way, there’s little opportunity to balance the cost of doing it.

So what are the tangible synergies? What benefits can we whiteboard proactively, moving forward? Where are the bullet points? Do you like my tie? It’s new!

Distributing a service around a network has the potential to improve service availability, since the redundancy inherent in using multiple origin servers affords some protection from server failure. For a service that has bad failure characteristics (e.g., a service that many other systems depend on) this might be justification enough to get things moving.

Moving the origin server closer to the community of clients that use it has the potential to improve response times and to keep traffic off expensive wide-area links. There might also be opportunities to keep a service running in a part of your network that is afflicted by failures in wide-area links in a way that wouldn’t otherwise be possible.

For services deployed over the Internet, as well as nobody knowing whether you’re a dog, there’s the additional annoyance and cost of receiving all kinds of junk traffic that you didn’t ask for. Depending on how big a target you have painted on your forehead, the unwanted packets might be a constant drone of backscatter, or they might be a searing beam of laser-like pain that makes you cry like a baby. Either way, it’s traffic that you’d ideally like to sink as close to the source as possible, ideally over paths that are as cheap as possible. Anycast might well be your friend.

Flies in the Ointment

The architectural problem with anycast for use as a general-purpose service distribution mechanism results from the flagrant abuse of packet delivery semantics and addressing that the technique involves. It’s a hack, and as with any hack, it’s important to understand where the boundaries of normal operation are being stretched.

Most protocol exchanges between clients and servers on the Internet involve more than one packet being sent in each direction, and most also involve state being retained between subsequent packets on the server side. Take a normal TCP session establishment handshake, for example:

1. Client sends a SYN to a server.
2. Server receives the SYN and replies with a SYN-ACK.
3. Client receives the SYN-ACK and replies with an ACK.
4. Server receives the ACK, and the TCP session state on both client and server is “ESTABLISHED.”

This exchange relies on the fact that “server” is the same host throughout the exchange. If that assumption turns out to be wrong, then this happens:

1. Client sends a SYN to server A.
2. Server A receives the SYN and replies with a SYN-ACK.
3. Client receives the SYN-ACK and replies to the service address with an ACK.
4. Server B receives the ACK and discards it, because it has no corresponding session in “SYN-RECEIVED.”

At the end of this exchange, the client is stuck in “SYN-SENT,” server A is stuck in “SYN-RECEIVED,” and server B has no session state at all. Clearly this does not satisfy the original goal of making things more robust; in fact, under even modest query load from perfectly legitimate clients, the view from the servers is remarkably similar to that of an incoming SYN flood.

It’s reasonable to wonder what would cause packets to be split between servers in this way, because if that behavior can be prevented perhaps the original benefits of distributed services that gave us all those warm fuzzies can be realized without inadvertently causing our own clients to attack us. The answer lies in the slightly mysterious realm of routing.

The IP routing tables most familiar to system administrators are likely to be relatively brief and happily uncontaminated with complication. A single default route might well suffice for many hosts, for example; the minimal size of that routing table is a reflection of the trivial network topology in which the server is directly involved. If there’s only one option for where to send a packet, that’s the option you take. Easy.

Routers, however, are frequently deployed in much more complicated networks, and the decision about where to send any particular packet is correspondingly more involved. In particular, a router might find itself in a part of the network where there is more than one viable next hop toward which to send a packet; even with additional attributes attached to individual routes, allowing routers to prioritize one routing table entry over another, there remains the distinct possibility that a destination address might be reached equally well by following any one of several candidate routes. This situation calls for Equal-Cost Multi-Path (ECMP) routing.

Without anycast in the picture, so long as the packets ultimately arrive at the same destination, ECMP is probably no cause for lost sleep. If the destination address is anycast, however, there’s the possibility that different candidate routes will lead to different servers, and therein lies the rub.

Horses for Courses

So, is anycast a suitable approach to making services more reliable? Well, yes and no. Maybe. Maybe not, too. Oh, it’s all so vague! I crave certainty! And caffeine-rich beverages!

The core difficulty that leads to all this weak hand-waving is that it’s very difficult to offer a general answer when the topology of even your own network depends on the perspective from which it is viewed. When you start considering internetworks such as, well, the Internet, the problem of formulating a useful general answer stops being simply hard and instead becomes intractable.

From an architectural perspective, the general answer is that for general-purpose services and protocols, anycast doesn’t work. Although this is mathematically correct (in the sense that the general case must apply to all possible scenarios), it flies in the face of practical observations and hence

doesn't really get us anywhere. Anycast is used today in applications ranging from the single-packet exchanges of the DNS protocol to multi-hour, streaming audio and video. So it does work, even though in the general case it can't possibly.

The fast path to sanity is to forget about neat, simple answers to general questions and concentrate instead on specifics. Just because anycast cannot claim to be generally applicable doesn't mean it doesn't have valid applications.

First, consider the low-hanging fruit. A service that involves a single-packet, stateless transaction is most likely ideally suited to distribution using anycast. Any amount of oscillation in the routing system between origin servers is irrelevant, because the protocol simply doesn't care which server processes each request, so long as it can get an answer.

The most straightforward example of a service that fits these criteria is DNS service using UDP transport. Since the overwhelming majority of DNS traffic on the Internet is carried over UDP, it's perhaps unsurprising to see anycast widely used by so many DNS server administrators.

As we move on to consider more complicated protocols—in particular, protocols that require state to be kept between successive packets—let's make our lives easy and restrict our imaginings to very simple networks whose behavior is well understood. If our goal is to ensure that successive packets within the same client-server exchange are carried between the same client and the same origin server for the duration of the transaction, there are some tools we can employ.

We can arrange for our network topology to be simple, such that multiple candidate paths to the same destination don't exist. The extent to which this is possible might well depend on more services than just yours, but then the topology also depends to a large extent on the angle you view it from. It's time to spend some time under the table, squinting at the wiring closet. (But perhaps wait until everybody else has gone home, first.)

We can choose ECMP algorithms on routers that have behavior consistent with what we're looking for. Cisco routers, for example, with CEF (Cisco Express Forwarding) turned on, will hash pertinent details of a packet's header and divide the answer space by the number of candidate routes available. Other vendors' routers have similar capabilities. If the computed hash is in the first half of the space, you choose the left-hand route; if the answer is in the other half, you choose the right-hand route. So long as the hash is computed over enough header variables (e.g., source address and port, destination address and port) the route chosen ought to be consistent for any particular conversation ("flow," in router-ese).

When it comes to deploying services using anycast across other people's networks (e.g., between far-flung corners of the Internet), there is little certainty in architecture, topology, or network design and we need instead to concentrate our thinking in terms of probability: We need to assess benefit in the context of risk.

Internet, n: "the largest equivalence class in the reflexive transitive symmetric closure of the relationship 'can be reached by an IP packet from'" (Seth Breidbart).

The world contains many hosts that consider themselves connected to the Internet. However, that "Internet" is different, in general, for every host—it's a simple truism that not all the nodes in the world that believe themselves to be part of "the" Internet can exchange packets with each other, and that's even without our considering the impact of packet filters and network ad-

dress translation. The Internet is a giant, seething ball of misconfigured packet filters, routing loops, and black holes, and it's important to acknowledge this so that the risks of service deployment using anycast can be put into appropriate context.

A service that involves stateful, multi-packet exchanges between clients and servers on the Internet, deployed in a single location without anycast, will be unavailable for a certain proportion of hosts at any time. You can sometimes see signs of this in Web server and mail logs in the case of asymmetric failures (e.g., sessions that are initiated but never get established); other failure modes might relate to control failures (e.g., the unwise blanket denial of ICMP packets in firewalls which so often breaks Path MTU Discovery). In other cases the unavailability might have less mysterious origins, such as a failed circuit to a transit provider which leaves an ISP's clients only able to reach resources via peer networks.

Distributing the same service using anycast can eliminate or mitigate some of these problems, while introducing others. Access to a local anycast node via a peer might allow service to be maintained to an ISP with a transit failure, for example, but might also make the service vulnerable to rapid changes in the global routing system, which results in packets from a single client switching nodes, with corresponding loss of server-side state. At layer-9, anycast deployment of service might increase costs in server management, data center rental, shipping, and service monitoring, but it might also dramatically reduce Internet access charges by shifting the content closer to the consumer. As with most real-life decisions, everything is a little grey, and one size does not fit all.

Go West, Young Man

So, suppose you're the administrator of a service on the Internet. Your technical staff have decided that anycast could make their lives easier, or perhaps the pointy-haired guy on the ninth floor heard on the golf course that anycast is new and good and wants to know when it will be rolled out so he can enjoy his own puffery the next time he's struggling to maintain par on the eighth hole. What to do?

First, there's some guidance that was produced in the IETF by a group of contributors who have real experience in running anycast services. That the text of RFC 4786 [4] made it through the slings and arrows of outrageous run-on threads and appeals through the IETF process ought to count for something, in my opinion (although as a co-author my opinion is certainly biased).

Second, run a trial. No amount of theorizing can compete with real-world experience. If you want to know whether a Web server hosting images can be safely distributed around a particular network, try it out and see what happens. Find some poor victim of the slashdot effect and offer to host her page on your server, and watch your logs. Grep your netstat -an and look for stalled TCP sessions that might indicate a problem.

Third, think about what problems anycast could introduce, and consider ways to minimize the impact on the service or to provide a fall-back to allow the problems to be worked around. If your service involves HTTP, consider using a redirect on the anycast-distributed server that directs clients at a non-anycast URL at a specific node. Similar options exist with some streaming media servers. If you can make the transaction between clients and the anycast service as brief as possible, you might insulate against periodic routing instability that would be more likely to interrupt longer sessions.

Fourth, consider that there are some combinations of service, protocol, and network topology that will never be good environments for anycast to work. Anycast is no magic wand; to paraphrase the WOPR [5], sometimes the only way to win is not to play.

REFERENCES

- [1] J. Abley, "A Software Approach to Distributing Request for DNS Service Using GNU Zebra, ISC BIND9 and FreeBSD," ISC-TN-2004-1, March 2004: <http://www.isc.org/pubs/tn/isc-tn-2004-1.html>.
- [2] USENIX Annual Technical Conference (USENIX '04) report, *;login:*, October 2004, page 52.
- [3] J. Abley, "Hierarchical Anycast for Global Service Distribution," ISC-TN-2003-1, March 2003: <http://www.isc.org/pubs/tn/isc-tn-2003-1.html>.
- [4] J. Abley and K. Lindqvist, "Operation of Anycast Services," RFC 4786, December 2006.
- [5] "War Operation Plan Response": <http://en.wikipedia.org/wiki/WarGames>.