AWS Architecture Blog

A Case Study in Global Fault Isolation

by Lee-Ming Zen | on 21 MAY 2014 | in Amazon Route 53, Architecture | Permalink | Share

In a previous blog post, we talked about using shuffle sharding to get magical fault isolation. Today, we'll examine a specific use case that Route 53 employs and one of the interesting tradeoffs we decided to make as part of our sharding. Then, we'll discuss how you can employ some of these concepts in your own applications.

Overview of Anycast DNS

One of our goals at Amazon Route 53 is to provide low-latency DNS resolution to customers. We do this, in part, by announcing our IP addresses using "anycast" from over 50 edge locations around the globe. Anycast works by routing packets to the closest (network-wise) location that is "advertising" a particular address. In the image below, we can see that there are three locations, all of which can receive traffic for the 205.251.194.72 address.

(Blue circles represent edge locations; orange circles represent AWS regions)

For example, if a customer has ns-584.awsdns-09.net assigned as a nameserver, issuing a query to that nameserver could result in that query landing at any one of multiple locations responsible for advertising the underlying IP address. Where the query lands depends on the anycast routing of the Internet, but it is generally going to be the closest network-wise (and hence, low latency) location to the end user.

Behind the scenes, we have thousands of nameserver names (e.g. ns-584.awsdns-09.net) hosted across four top-level domains (.com, .net, .co.uk, and .org). We refer to all the nameservers in one top-level domain as a 'stripe;' thus, we have a .com stripe, a .net stripe, a .co.uk stripe, and a .org stripe. This is where shuffle sharding comes in: each Route 53 domain (hosted zone) receives four nameserver names one from each of stripe. As a result, it is unlikely that two zones will overlap completely across all four nameservers. In fact, we enforce a rule during nameserver assignment that no hosted zone can overlap by more than two nameservers with any previously created hosted zone.

DNS Resolution

Before continuing, it's worth quickly explaining how DNS resolution works. Typically, a client, such as your laptop or desktop has a "stub resolver." The stub resolver simply contacts a recursive nameserver (resolver), which in turn queries authoritative nameservers, on the Internet to find the answers to a DNS query. Typically, resolvers are

provided by your ISP or corporate network infrastructure, or you may rely on an open resolver such as Google DNS. Route 53 is an authoritative nameserver, responsible for replying to resolvers on behalf of customers. For example, when a client program attempts to look up amazonaws.com, the stub resolver on the machine will query the resolver. If the resolver has the data in cache and the value hasn't expired, it will use the cached value. Otherwise, the resolver will query authoritative nameservers to find the answer.

(Every location advertises one or more stripes, but we only show Sydney, Singapore, and Hong Kong in the above diagram for clarity.)

Each Route 53 edge location is responsible for serving the traffic for one or more stripes. For example, our edge location in Sydney, Australia could serve both the .com and .net, while Singapore could serve just the .org stripe. Any given location can serve the same stripe as other locations. Hong Kong could serve the .net stripe, too. This means that if a resolver in Australia attempts to resolve a query against a nameserver in the .org stripe, which isn't provided in Australia, the query will go to the closest location that provides the .org stripe (which is likely Singapore). A resolver in Singapore attempting to query against a nameserver in the .net stripe may go to Hong Kong or Sydney depending on the potential Internet routes from that resolver's particular network. This is shown in the diagram above.

For any given domain, in general, resolvers learn the lowest latency nameserver based upon the round trip time of the query (this technique is often called SRTT or smooth round-trip time). Over a few queries, a resolver in Australia would gravitate toward using the nameservers on the .net and .com stripes for Route 53 customers' domains.

Not all resolvers do this. Some choose randomly amongst the nameservers. Others may end up choosing the slowest one, but our experiments show that about 80% of resolvers use the lowest RTT nameserver. For additional information, this presentation presents information on how various resolvers choose which nameserver they utilize. Additionally, many other resolvers (such as Google Public DNS) use pre-fetching, or have very short timeouts if a resolver fails to resolve against a particular nameserver.

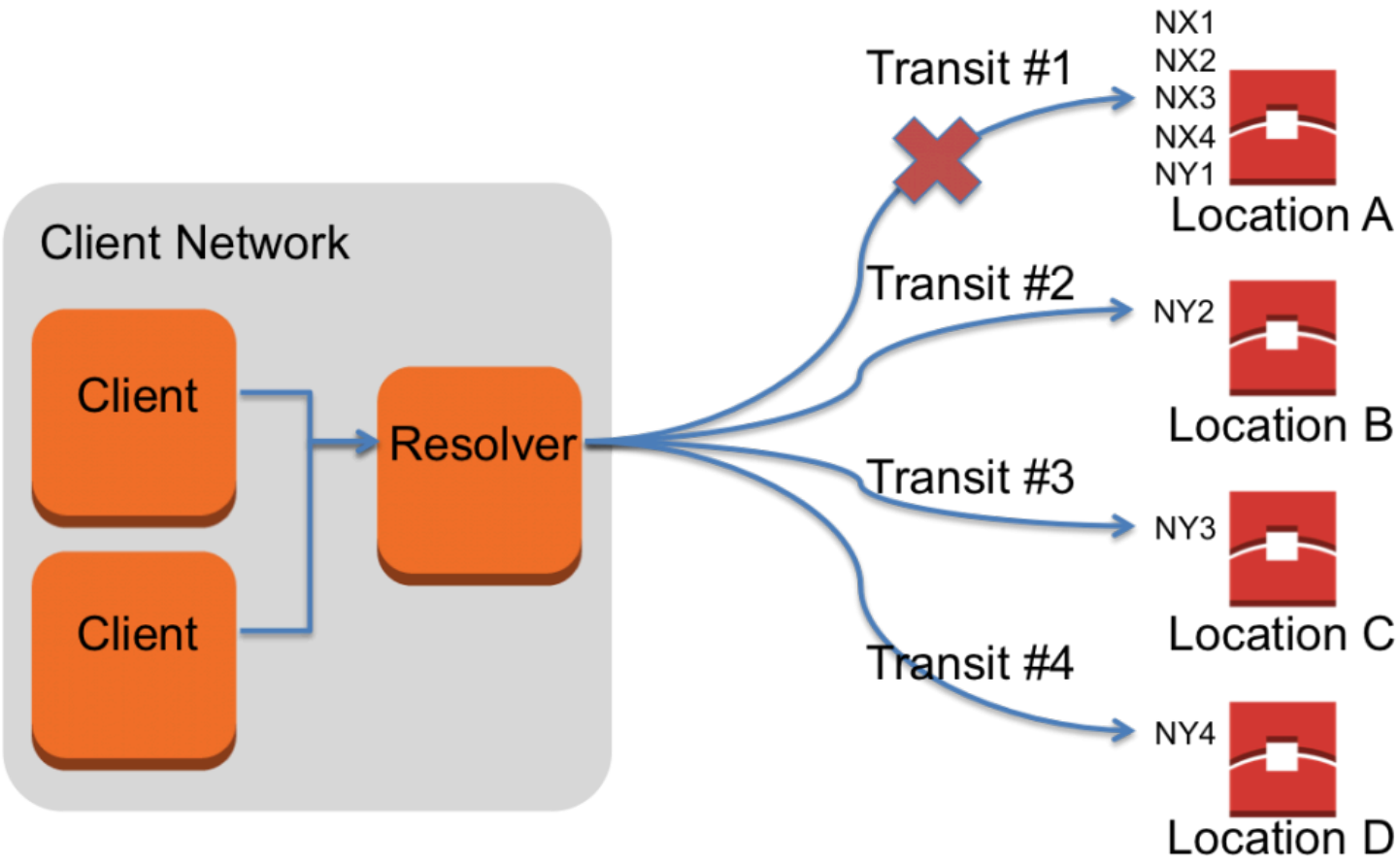
The Latency-Availability Decision

Given the above resolver behavior, one option, for a DNS provider like Route 53, might be to advertise all four stripes from every edge location. This would mean that no matter which nameserver a resolver choses, it will always go to the closest network location. However, we believe this provides a poor availability model.

Why? Because edge locations can sometimes fail to provide resolution for a variety of reasons that are very hard to control: the edge location may lose power or Internet connectivity, the resolver may lose connectivity to the edge location, or an intermediary transit provider may lose connectivity. Our experiments have shown that these types of events can cause about 5 minutes of disruption as the Internet updates routing tables. In recent years another serious risk has arisen: large-scale transit network congestion due to DDOS attacks. Our colleague, Nathan Dye, has a talk from AWS re:Invent that provides more details: www.youtube.com/watch?v=V7vTPlV8P3U.

In all of these failure scenarios, advertising every nameserver from every location may result in resolvers having no fallback location. All nameservers would route to the same location and resolvers would fail to resolve DNS queries, resulting in an outage for customers.

In the diagram below, we show the difference for a resolver querying domain X, whose nameservers (NX1, NX2, NX3, NX4) are advertised from all locations and domain Y, whose nameservers (NY1, NY2, NY3, NY4) are advertised in a subset of the locations.



When the path from the resolver to location A is impaired, all queries to the nameservers for domain X will fail. In comparison, even if the path from the resolver to location A is impaired, there are other transit paths to reach nameservers at locations B, C, and D in order to resolve the DNS for domain Y.

Route 53 typically advertises only one stripe per edge location. As a result, if anything goes wrong with a resolver being able to reach an edge location, that resolver has three other nameservers in three other locations to which it

can fall back. For example, if we deploy bad software that causes the edge location to stop responding, the resolver can still retry elsewhere. This is why we organize our deployments in "stripe order"; Nick Trebon provides a great overview of our deployment strategies in the previous blog post. It also means that queries to Route 53 gain a lot of Internet path diversity, which helps resolvers route around congestion and other intermediary problems on their path to reaching Route 53.

Route 53's foremost goal is to always meet our promise of a 100% SLA for DNS queries – that all of our customers' DNS names should resolve all the time. Our customers also tell us that latency is next most important feature of a DNS service provider. Maximizing Internet path and edge location diversity for availibility necessarily means that some nameservers will respond from farther-away edge locations. For most resolvers, our method has no impact on the minimum RTT, or fastest nameserver, and how quickly it can respond. As resolvers generally use the fastest nameserver, we're confident that any compromise in resolution times is small and that this is a good balance between the goals of low latency and high availability.

On top of our striping across locations, you may have noticed that the four stripes use different top-level domains. We use multiple top-levels domains in case one of the three TLD providers (.com and .net are both operated by Verisign) has any sort of DNS outage. While this rarely happens, it means that as a customer, you'll have increased protection during a TLD's DNS outage because at least two of your four nameservers will continue to work.

Applications

You, too, can apply the same techniques in your own systems and applications. If your system isn't end-user facing, you could also consider utilizing multiple TLDs for resilience as well. Especially in the case where you control your own API and clients calling the API, there's no reason to place all your eggs in one TLD basket.

Another application of what we've discussed is minimizing downtime during failovers. For high availability applications, we recommend customers utilize Route 53 DNS Failover. With failover configured, Route 53 will only return answers for healthy endpoints. In order to determine endpoint health, Route 53 issues health checks against your endpoint. As a result, there is a minimum of 10 seconds (assuming you configured fast health checks with a single failover interval) where the application could be down, but failover has not triggered yet. On top of that, there is the additional time incurred for resolvers to expire the DNS entry from their cache based upon the record's TTL. To minimize this failover time, you could write your clients to behave similar to the resolver behavior described earlier. And, while you may not employ an anycast system, you can host your endpoints in multiple locations (e.g. different availability zones and perhaps even different regions). Your clients would learn the SRTT of the multiple endpoints over time and only issue queries to the fastest endpoint, but fallback to the other endpoints if the fastest is unavailable. And, of course, you could shuffle shard your endpoints to achieve increased fault isolation while doing all of the above.

– Lee-Ming Zen
TAGS: Fault Isolation