The End of DNS As We Know It

Lori MacVittie, 2009-28-08

DNS wasn't meant to handle hybrid cloud architectures and on-demand routing

When you start distributing services (workloads, applications) across multiple locations, a la cloud balancing, and those locations may change on a frequent basis you begin to run into problems with *finding* those services and scaling the rate of change effectively. DNS was designed to resolve host names, but never expected that the same host name might resolve to one of two, three, or four IP addresses all within the span of five minutes.

If we want to support a rapid rate of change, we'd also need to consider the strain on the existing DNS infrastructure as it would require that propagation rates be decreased such that changes would be discovered as needed rather than 2 or 3 hours (or days) later. That change, however, isn't specific to any particular technology and would affect all resolution requests. That invariably increases traffic and stress on the entire Internet infrastructure.

Too, DNS certainly isn't prepared to deal with the possibility that two different clients might need two completely different addresses for the same host. DNS was not designed to support contextual-based response to a query. It's host name in –> IP address out and – this is what makes it bad for cloud in the future – it's completely anonymous.

DNS DOESN'T TURN ON A DIME

It's not just a constant rate of change that's a potential problem for DNS. It's unlikely that organizations will see change in service/application locations that is so out of whack with conventional access patterns that it's necessary to support changing IP addresses on an hourly or daily basis. But the use of cloud-hosted applications as "backup" or "secondary" or "overflow" data centers is a very real one that introduces the need to be able to very quickly redirect requests from one location to another. DNS needs to be able to turn on a dime, as it were, and that's not something that DNS does well – at least not without introducing a lot of strain on the network and its infrastructure.

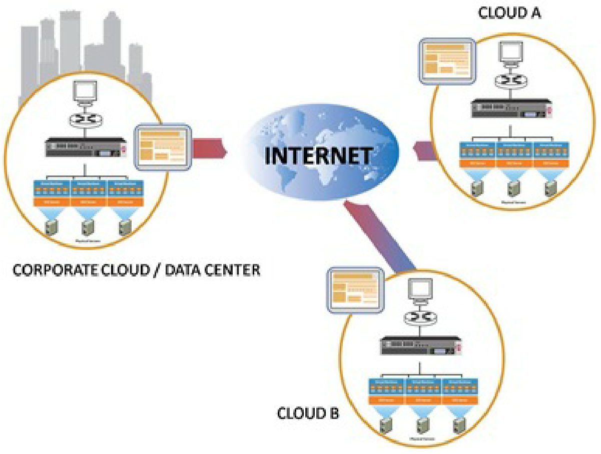
DNS would still need to propagate change rapidly "just in case" an overflow situation occurred in which some requests needed to be directed to other data centers. And the time-out value that forces resolution would need to be low enough to support a sudden redirection to another IP address.

But this brings up another problem: even though it is common enough to map multiple IP addresses to a single host/domain, these are not intelligently used by clients. They are more like successive retreating lines in a battle; after each failure the client falls back to the next, and the next, and the next, without any consideration for availability or performance. There's also no way to control the use of those IP addresses based on current conditions within the data center. The "failover" between one IP address and the next does not necessarily recognize a failure to connect because an application is over capacity as "failure"; it isn't a given that a client/customer will be redirected appropriately based on data center/application conditions.

DNS just wasn't designed to handle these kinds of scenarios, nor was it was meant to react that quickly to change.

GLOBAL LOAD BALANCING

Global server load balancing (GSLB), however, ***was*** designed to handle these types of scenarios. GSLB is one of the most misnamed technologies, in my opinion, because while the goal is to load balance requests globally (across multiple data centers and locations) the implementation is really via a flexible and intelligent system based on DNS. A GSLB implementation is designed with the understanding that any given request might need to be directed to some other location and does not maintain a one-to-one relationship between host/application and IP address. GSLB can assume both a high rate of change and on-demand resolution.



For example, GSLB is most often associated, today, with geographic-based load balancing or routing. The idea is based on the premise that the shortest distance between two points will be the fastest and therefore a request directed to a physically closer data center will achieve a much better response time than a request directed to a more remote data center. Now take ***that*** concept and use other variables to decide where to route the request – such as actual response time of the application, current capacity of applications at various data centers, the time of day, or even user-specific information. That's called ***context*** and it's what allows a GSLB solution to intelligently route requests to the data center best suited to respond based on all the variables known at that moment in time.

Large organizations with multiple data centers know this and have long been implemented global load balancing solutions to address this very scenario. Most medium and small businesses have never considered it before because they didn't have a secondary data center to which requests could be directed. They had not the staff nor the budgets to build out a second data center and thus global load balancing was never an option for them. But cloud computing makes it an option and it makes it a very likely (and attractive) option. One of the benefits of cloud computing is that the staff, the hardware, the facilities – everything physical – already exists and is completely managed by the provider. All that's required is the packaging up and deployment of the application into the cloud.

Cloud computing coupled with intercloud and cloud balancing concepts result in a requirement for GSLB for anyone considering a hybrid model of application deployment. Not just large organizations, but small ones, too, if they host applications on-premise ***and*** in the cloud.

DNS will continue to be the backbone of the Internet; without DNS the Internet would cease to exist as we know it. But DNS will surely begin to become one of the core technologies that is always implemented but rarely seen externally, like ARP. Or, perhaps, the DNS will merge with GSLB and become one fluid system; GDNS (Global DNS). GDNS would certainly address the core requirement and need for DNS, and the integration of GSLB would bring the context-awareness required of cloud to the table.

Either way, the concept of GSLB will eventually become the de facto standard for resolving external facing services and applications because without it there's not really an efficient way to handle the hybrid architectures that are predicted to come or the rapid rate of change inherent in cloud computing models.

Like all infrastructure, DNS needs to move toward its "2.0" version to support emerging data center models and that may mean merging with GSLB solutions in order to become as "dynamic" as the "D" in DNS.

- Governance: Service Catalogs and the Cloud
- We Don't Know What Cloud Is But What We're Doing It
- The Context-Aware Cloud
- Taking Down Twitter as easy as D.N.S.
- Business-Layer Load Balancing
- Cloud Balancing, Cloud Bursting, and Intercloud
- Getting Around That Pesky Speed of Light Limitation
- You are the new number 3ffe:1900:4545:3:200:f8ff:fe21:67cf