Anycast-Aware Transport for Content Delivery Networks Review

This paper was published in the World Wide Web Conference in 2009, located in Madrid, Spain. This international conference has been ongoing for nearly two decades, as this was the 18th. It is sponsored by large internet based companies such as Google and Yahoo, as well as many other smaller companies.

This is a very recent paper about CDN's, which currently are a hot topic in networking. They propose to address the problem of severed TCP sessions due to routing changes with a simple but efficient solution.

The abstract is very well written—it describes the problem, their solution, and even states that they have results showing they can in fact help server performance without any negative impact. This makes the paper seem like a very worthwhile read.

The introduction begins explaining the basics of CDN's for those unfamiliar. It than raises a few more specific problems than expressed in the abstract—CDN's select the edge server based on the location of the DNS server, not the client, which can be inaccurate in many cases. Also, load balancing is a problem due to clients caching DNS servers for long periods of time regardless of current load conditions, and is made worse by clients who ignore the TTL returned by the DNS response using that cached response for even longer. In addition, malicious users can use a specific server by bypassing the DNS responses, which can have a negative impact on server performance. Finally, there is a big difference in workload between different DNS servers based on their location and client base.

The authors propose IP anycast as a solution. It assigns the same IP address to multiple edge servers, and uses IP routing to discover the closest server to a client. This solves the problem of proximity based upon DNS location rather than actual client location, but raises two problems. The first problem, the authors claim to have solved in a previous paper (server load imbalance). The second problem, is that of terminated TCP connections due to routing changes during a (typically large file size) transfer, and is what this paper focuses on solving.

Their solution is simple—if a TCP session is terminated, the browser should send a new HTTP range request for the remaining bytes of the file, and download it from a new near by server. The authors believe this will work equally well for large files as for streaming multimedia, as long as the download speed is higher than playback speed.

The rest of the paper addresses 5 possible issues for implementing their proposed solution. In short, these are how will the client notice the connection disruption, how the server will deal with dormant TCP connections, what is the impact of multiple range requests for a single file, how does this solution affect server performance, and finally what are the security issues with this solution.

Next the authors give a more detailed background of anycast, and reiterate the two current problems with it in a bit more detail.

In the Related Work section, they site their previous work and give a more in depth summary of how it solves the first of the two main problems (server load imbalance), and how this paper builds upon it to solve the second problem (terminated TCP connections during file transfers).

The necessary client and server implementations are then described: clients would need a browser extension or stand alone download manager to support anycast at the application layer that distinguishes between a redirection and true network outage. It would accomplish this by simply aborting the download when a connection failure occurs during the TCP handshake, and sending a new HTTP range request. Servers would need to set the retry to a low value like 1 to avoid resource hogging dormant connections. While this may cause multiple redirects, the authors claim that their client handled the new range requests gracefully in their live Internet experiments. This solves the first three previously mentioned possible issues with their implementation.

As far as performance issues, the initial worries were that with multiple redirections there would be many new three-way handshakes and TCP slow starts. The authors originally implemented a solution to this in their design, until they realized that the actual effect of these redirects on performance was miniscule with the rate of redirections expected.

As for security, they say CDN's have recently been vulnerable to DoS attacks. Their solution solves this problem by “absorbing the attack”-- it can redistribute some of the ongoing downloads from overloaded servers to other near by edge servers. On top of this, by shortening the connection timeout the CDN's become more resilient to DoS attacks by malicious user requesting many downloads from a botnet and than disappearing, causing many open TCP connections to linger until a timeout occurs (up to 30 minutes in some cases), hogging up resources. Finally, normal CDN's are susceptible to users flooding a specific server by ignoring the DNS response as mentioned before, while with anycast the user has no choice over which server will serve their request.

Next the authors examine performance results based upon their FreeBSD dummynet. They show through their results why a solution to the three-way handshake and TCP slow start for every redirection is not necessary. While this experiment setup is small scale, it seems to provide legitimate results since they are only testing the quality of a single client with varying frequencies of redirections.

Next they analyze performance outcome of quick state purging (lowering the number of server connection retries (RTO)). They setup two servers, one with retries set to 15, and the other set with 1 and 3 RTOs. Next they had 59 Keynote clients connect to both servers and download a 2MB file every 15 minutes for over a week and collected over 33,000 connections to each of the two servers. The authors attribute some error in their results to Keynote clients going offline, or having connection issues and offsetting the averages. This makes the overall results a bit sketchy, but they still gather some useful information from it. They claim that due to the extremely low drop ratio even with Keynote client issues, that setting the RTO to 3 will help clean up dormant connections and protect against DoS attacks without impacting performance.

A final performance concern is a burst of TCP connections attempting range requests resulting in synchronization problems in which many connections ramp up and back off at roughly the same time. To test this, they setup an experiment and tested the worst case scenario. Taking into account some randomization due to their experiment setup, they conclude that it is minimal and randomization in the real world would even exceed theirs. Their results show that even with such a small amount of randomization due to host processing, it is enough to avoid synchronization.

In the conclusions they mention that policy-based routing may interfere with the benefits of anycast, and that a careful study of its effects is a topic for future work.

This paper showed many problems that anycast provides a very simple solution to, without the

necessity for additional work. They also backed up these claims with believable experiment data.

This paper is especially useful because due to the simplicity of their solution, it has a real chance at being implemented in the real world (as mentioned in the conclusion), which is often not the case with protocols and solutions created in scholarly papers.

A weakness is that while they had decent experiments to prove certain points, they never conducted an experiment simulating the system as a whole—multiple CDN's running anycast with many clients downloading various files at various speeds and times. So while it sounds fairly convincing that it is better performing than current CDN's, there is no data to prove it which would be a nice addition to the paper.

The organization and readability of the paper was very well done. Coming from outside of the field, it was still easy to follow along and understand. The abstract was well done, it described exactly what the paper was about, and made claims worthy of reading. The conclusion was a bit offsetting due to the part about policy-based routing possibly interfering with the benefits of anycast, and that they had done no research into this topic. On the other hand, they seem to plan on doing said research and possibly even another paper on this topic, which would be interesting to read. They end with the point that due to the simplicity of their solution it has a real chance at being implemented in the real world, which is a very strong point.

Overall this is a great contribution to it's field, well written, with reasonable data, and solves a great deal of problems with CDN's today.