# Topic Review: Content Delivery Networks

## Review: Anycast-Aware Transport for Content Delivery Networks

Published at the most recent WWW conference (WWW '09), "Anycast-Aware Transport for Content Delivery Networks" attempts to address several challenges in utilizing IP anycast as a redirection mechanism for content delivery networks (CDNs). Typically, DNS is used by CDNs to automatically distribute load to the most appropriate edge-server and to redirect traffic away from particular edge-servers as needed (for example, to take the server out of service for maintenance). In the authors' prior work (see reference [6] cited in the paper), they proposed utilizing IP anycast as an alternative to DNS to overcome several of its inherent limitations:

1. DNS servers are sometimes topologically far from the users they serve. Mapping DNSs to end-server IPs, therefore, may not always yield the most efficient route for end-users.

2. DNS can be slow to react to IP address and hostname mapping changes. This inhibits the CDNs ability to react to sudden changes in load patterns, and can be burdensome when needing to take servers out of service for maintenance.

The authors claim IP anycast has the potential to solve these problems because it utilizes existing routing mechanisms that a) achieve optimal (i.e. shortest) routes for end-users; and b) are able to quickly and dynamically adjust routes to changing network conditions.

The authors point out, however, that IP anycast has several characteristics that have prevented from being used as a redirection mechanism for CDNs:

1. Existing routing mechanisms are not load aware. Therefore a typical IP anycast solution could potentially overload a server that happened to have many "nearby" users.

2. IP anycast is not session aware and can cause significant problems for session based protocols, such as TCP. Since a route change may happen at any point in time, interruption of in-flight sessions can occur, resulting in potentially excessive amounts of data retransmission.

Having addressed the first problem in their earlier work, the authors focus this paper primarily on addressing the second problem.

This paper proposes modifications to application and transport protocols to make them anycast aware. More specifically, the authors introduce strategic changes CDN client applications that allow them to a) detect a connection disruption due to an anycast change, and b) gracefully recover from these disruptions. Additionally, changes to server TCP stacks are proposed to improve their performance and resiliency in an anycast network.

Conceptually, the strategy proposed in the paper is rather straightforward. When a client detects a connection failure while downloading a resource, it recovers by issuing an HTTP range request for the portion of the resource that has yet to be transmitted. The disruption detection mechanism offered in the paper utilizes existing error information made available by the TCP layer; therefore, it can be implemented entirely in the application layer and requires no modifications to the TCP implementation on client machines. This is a distinct advantage of the proposed solution and adds significantly to the weight of the paper's contribution.

The authors identify a number of research questions raised by their solution regarding is implications for client and server performance and security. Specifically, they consider the following implications of solution they propose:

1. The effect of the orphaned TCP state on servers whose traffic was redirected
2. The resiliency of a system that implements the proposed mechanisms to common attacks, such as DoS.
3. The effect on average throughput of restarting TCP sessions (potentially many times) during the course of a download
4. The effect of spikes in new HTTP range requests on CDN servers that are targets of route redirection

The authors' treatments of these considerations are very thorough and establish the viability of their solution. To address the first concern regarding orphaned TCP resources on old servers whose connections were moved to new servers, the authors propose modifications to server-side TCP configurations that limit the number of attempts the server makes to retransmit unacknowledged segments. This allows the server to much more quickly recover resources held by dormant TCP connections. While the repeated retries and long timeouts in the typical TCP protocol are in place to protect reliability and minimize dropped connections, the authors conduct experiments that show reducing the retry count from the default of 15 to 1 yields connection drop rates that are still within an acceptable range. This observation alone is a distinct contribution of the paper and could have implications beyond the scope of anycast enabled CDNs.

The authors argue that the proposed server-side changes to the TCP stack have an additional benefit of making servers more resistant to denial of service attacks. This claim is unsubstantiated by empirical evidence, and would have been strengthened if they had conducted experiments to demonstrate improved DoS resiliency. Another weakness of this proposal is that even minor changes to TCP stack can have significant side effects. A deeper investigation of the impact of the proposed changes in a broad range of TCP scenarios would likely be needed before they could be adopted in practice. While such an investigation is surely beyond the scope of this paper, the authors fail to acknowledge this as a concern that should be addressed in future work.

The authors conduct further experimentation which shows that even a relatively high rate of connection disruptions (and therefore TCP session restarts) has minimal effect on data throughput. This conclusion

supports the paper's claim that its proposed solution is viable without modifications to client-side TCP stacks.

To address the final concern regarding the effect of possible sudden and dramatic increases in HTTP range requests on a single server, the authors designed an experiment that simulated a large number of users simultaneously requesting to download the same 5 MB file. This experiment was designed to detect TCP synchronization that could result in poor user performance and underutilization of the server. The experiment did show that synchronization did not occur in this scenario. However, synchronization is only one of several potential performance problems that could occur from a sudden surge in user load due to an IP anycast change. For example, the experiment assumed the server was cold at the time of the load increase and does not consider the possible implications of increased load to a server that was already servicing other connections. The authors also do not consider the specific effects of spikes in HTTP range requests as opposed to typical HTTP requests. For example, particular server or proxy implementations may react poorly to range requests.

Overall, the paper is well structured and clearly written. It effectively communicates the benefits of the proposed solution to enable anycast as a redirection mechanism for CDNs, and addresses the primary difficulties in implementing such a system. The authors could have done more to reinforce the fact that their solution requires no changes to the client-side TCP stack – perhaps the most significant advantage of their proposal. They do make this point in section 3, but the title of the paper creates the impression that the authors are proposing changes primarily at transport layer. In addition, discussion in subsequent sections fails to adequately distinguish the suggested changes to server-side TCP implementations from the client-side changes that are required at the application layer only. This could cause confusion and some readers to miss this significant feature of the proposed solution.