Anycast-Aware Transport for Content Delivery Networks

This paper offers an anycast solution to the static binding problem in content delivery networks. In particular the authors are solving the connection disruption in connection-orientated download problem. The authors have proposed a solution in another paper that partially mitigates this problem by performing more efficient binding but the authors argue in this paper that the solution must be taken further in order to accommodate the need for server re-assignment caused by rapidly changing conditions. The authors motivate the need for re-assignment in the following ways: 1) large file downloads may take hours and may need to be re-assigned during the download. 2) Flash crowds may form causing a server to admit more clients than it can handle and some of the work needs to be offloaded to other servers. 3). Increasing demand may make it desirable to add servers. 4) When a flash crowd disperses re-assigning work from servers will allow the CDN to take down some servers sooner.

The problem with changing routes while a TCP connection is active is that the TCPs are unaware of the change. The TCP at the client will send an ACK which will arrive at the TCP on the new server. The new server will respond with a RST and the connection closes. The original server will not receive the ACK it was expecting and will wait an RTO and resend. The RTOs increase exponentially and are retried up to 15 times for Linux.

I think that the problem is significant and that the authors did a good job of explaining and motivating it.

The authors promise to answer the following solution. If the connection is closed the client will issue a new HTTP request asking for a byte range containing the missing data. If the connection close occurs during a handshake the download is aborted. The author's make the following five research contributions:

1) Ensuring that the client TCP will always be aware of the disconnection and react to it.
2) Understanding the implication of the dormant TCP state at the old servers.
3) Understanding the impact of multiple range requests on overall throughput (slowstart etc.)
4) Understanding the impact on server performance. What happens if many new requests reach a server simultaneously?
5) What are the security implications?

On the server side the RTO timer is reduced to 200ms and the socket implementation is extended allowing the application to specify the number of retries in order to mitigate the problem described in 2. Addressing problem 1, in order to detect connection disruption the server will send an explicit RST after retries (which happens quickly due to the above modifications). The new connection is on a different socket thus is not affected by RSTs and ACKs on the old socket.

The solution is simple, however the exploration of the implications has good research value.

Problem 5 is briefly addressed in the paper. Since connection assignment is changed quickly the effects of DOS attacks is mitigated. In addition, since the clients do not control which server they are assigned to as they do in a static DNS binding CDN the task of the DOS attacker is made more difficult.

This seems reasonable. However, there are no experiments to show that it is true.

Problem 3 is addressed with the following experiment. A client is downloading a 50 mb from a server through a dummynet connection that shapes the link to 10mbs. The connection is repeatedly interrupted causing the 3 way handshake and slow start to be repeated.

The data was averaged across 10 seeds. It would have been better if they had given us confidence bars. The graphs show that for switching occurring at a frequency of 30 seconds or more there is little loss of throughput. It is probably not important but it would have been nice if the authors explained why the throughput when there are connection disruptions is sometimes higher than the baseline with no interruptions. Also I would like to know why in figure 5a that the throughput takes a sudden drop with 1 second interruptions and 0.01% loss. This makes me suspicious of the graph.

The final experiment addresses how low the retry value can be set.

I think this is meant to address problem 4 but it is not clear.

The experiment uses two servers side by side with 59 clients in a cloud connecting to them once every 15 minutes. The first server uses a vanilla TCP with 15 retries, on the second server the retry value is varied from 0 to 3. A 2Mb file is downloaded by each client once every 15 minutes according to some distribution that they did not specify. The experiment shows that increasing the retry value from 0 to 1 reduces the problem of dropped connections to an insignificant amount. This shows that the open connection problem can be mitigated successfully without introducing unacceptable amounts of disconnections.

Finally the authors address the problem of synchronization among connections. They anticipate that this may be caused by a large number of clients being offloaded from an overloaded server to a new server. This experiment uses a single client to generate a large number of connections as fast as possible to a single server. If there is a synchronization problem then we will see an oscillation effect in the queue. The queue lengths do not exhibit any oscillation indicating that synchronization is not a problem.