

Anycast-Aware Transport for CDNs

Al-Qudah, Lee, Rabinovich,
Spatscheck, and Van der Merwe

Introduction

- CDNs use DNS to redirect clients to close servers
- Limitations:
 - Server selected based on proximity to client DNS server
 - CDN can't control caching of responses

Introduction

- Anycast is a potential alternative
- The same IP address is assigned to multiple servers
- IP routing chooses the closest server
 - Route is based on client, not client DNS
- Challenges
 - Load balancing
 - Disruption in connection-oriented downloads

Introduction

- Simple solution: server redirects to its own unicast IP at the beginning of a download
- Problems:
 - For long-running downloads, would like to adapt to changing network conditions
 - Does not allow redistribution of load after a flash crowd
 - Would like to redirect to new servers

Introduction

- Approach: when a session is disrupted, issue a request for remaining bytes to a new server
- Issues:
 - Ensuring the client finds out about disruption and issues request for new data
 - Impact on TCP state at old server
 - Impact of TCP slow start
 - Server performance

Background

- Figure 2 - routing change during anycast session
- Linux server will retry 15 times -- up to 30 minutes

Anycast-aware Transport

- Client-side
 - When the client detects a TCP connection failure, it issues an HTTP range request for remaining bytes
 - Suggested implementation as browser extension or download manager

Anycast-aware Transport

- Server-side
 - Functionally, there are no required changes on the server side
 - Issue: closing broken connections quickly
 - Solution: reduce number of times the server will retry

Performance Implications

- What is the implication of splitting a download into a sequence of range requests on the overall download throughput?
- What is the implication of closing dormant TCP connections quickly?
- What are the implications of our approach on server performance?