Should a load-balancer choose the path as well as the server?

Nikhil Handigol
Stanford University
Joint work with Nick McKeown and Ramesh Johari
Can't choose path :’( 
Outline and goals

- A new architecture for distributed load-balancing
  - joint (server, path) selection
- Demonstrate a nation-wide prototype
- Interesting preliminary results
OpenFlow Protocol (SSL)

Control Path

Data Path (Hardware)
Software Defined Networking

Network OS

Feature

Feature

OS

Custom Hardware

Feature

Feature

OS

Custom Hardware

Feature

Feature

OS

Custom Hardware

Feature

Feature

OS

Custom Hardware

Feature

Feature

OS

Custom Hardware

Feature

Feature

OS

Custom Hardware

Feature

Feature

OS

Custom Hardware

Feature

Feature

OS

Custom Hardware
Load Balancing is just Smart Routing
Load-balancing as a network primitive

Network OS

Load-balancing decision
Load-balancing logic
Load-balancing
Load-balancing

Custom Hardware
Custom Hardware
Custom Hardware
Custom Hardware
Aster*x Controller
Joint selection of Server and Network Path

http://www.openflow.org/videos
So far…

- A new architecture for distributed load-balancing
  - joint (server, path) selection
- Aster\(\textsuperscript{x}\) – a nation-wide prototype
- Promising results that joint (server, path) selection might have great benefits
What next?
How big is the pie?

Characterizing and quantifying the performance of joint (server, path) selection
Load-balancing Controller
Parameters

Topology

- Intra-AS topologies
  - BRITE (2000 topologies)
  - CAIDA (1000 topologies)
  - Rocketfuel (~100 topologies)
- 20-50 nodes
- Uniform link capacity
Parameters

Servers
- 5-10 servers
- Random placement

Service
- Simple HTTP service
- Serving 1 MB file
- Additional server-side computation
Parameters

Clients
- 3-5 client locations
- Random placement

Request pattern
- Poisson process
- Mean rate: 5-10 req/sec
Load-balancing strategies?
Design space

Simple but suboptimal

Complex but optimal

Disjoint-Shortest-Path

Disjoint-Traffic-Engineering

Joint
Anatomy of a request-response

Client

Load-Balancer

Server

Response Time

Request

Choose

Response 1st byte

Response last byte

Last byte ack

Retrieve

Deliver
Disjoint-Shortest-Path

- CDN selects the least loaded server
  - \( \text{Load} = \text{retrieve} + \text{deliver} \)
- ISP independently selects the shortest path
Disjoint-Traffic-Engineering

- CDN selects the least loaded server
  - $\text{Load} = \text{retrieve} + \text{deliver}$
- ISP independently selects path to minimize max load
  - Max bandwidth headroom
Joint

- Single controller jointly selects the best (server, path) pair

Total latency = retrieve + estimated deliver
Disjoint-Shortest-Path performs ~2x worse than Joint
Disjoint-Traffic-Engg. vs Joint

Disjoint-Traffic-Engineering performs almost as well as Joint
Is *Disjoint* truly disjoint?

Server response time contains network information
The bottleneck effect

A single bottleneck resource along the path determines the performance.
The CDN-ISP game
The CDN-ISP game

- System load monotonically decreases
- Both push system in the same direction
Summary of observations

- Disjoint-SP is ~2x worse than Joint
- Disjoint-TE performs almost as well as Joint
  (despite decoupling of server selection and traffic engineering)
- Game theoretic analysis supports the empirical observation
Conclusion

- A new architecture for distributed load-balancing
  - joint (server, path) selection
- Aster*x - a nation-wide prototype
- Interesting preliminary results
- Future – application to other contexts and applications
Let’s chat more!
Extra slides...
Sample topologies

[Graph of BRITE topology]

[Graph of CAIDA topology]