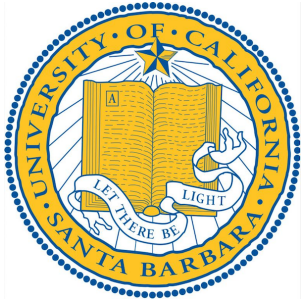
CS 290B

Scalable Internet Services

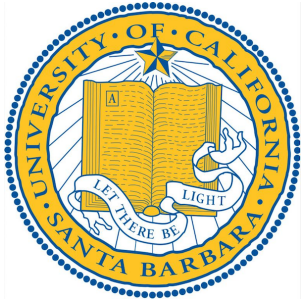
Andrew Mutz

October 16, 2014

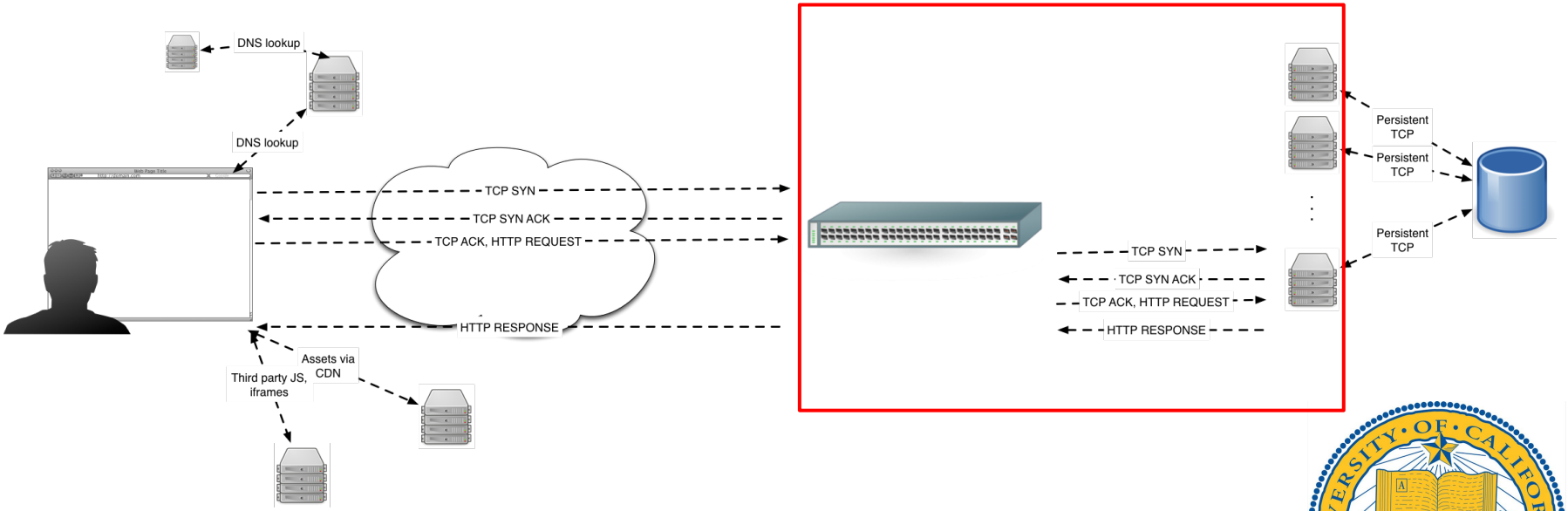


For Today

- Motivation
- Load Balancing on the Web
- High Availability Infrastructure
- For Next Time...



Motivation


DNS lookup
DNS lookup
TCP SYN
TCP SYN ACK
TCP ACK, HTTP REQUEST
HTTP RESPONSE
Assets via CDN
Third party JS, iframes
TCP SYN
TCP SYN ACK
TCP ACK, HTTP REQUEST
HTTP RESPONSE
Persistent TCP
Persistent TCP
Persistent TCP

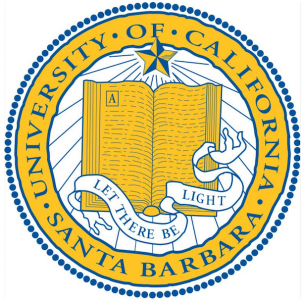
Motivation

Today we will be talking about two different problems with related solutions:

- Scaling
- High availability

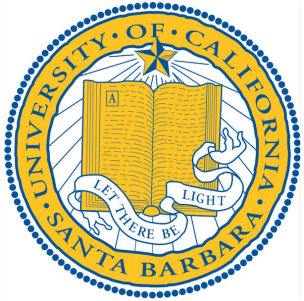
Let’s say we’ve got a web app running on a single EC2 instance and load keeps rising.

- What do we do?



Motivation

- Vertical Scaling
  - We buy a bigger and better instance
- Horizontal Scaling
  - We buy more instances
  - Use some technique to make them appear as a single web server
- Advantages of each?

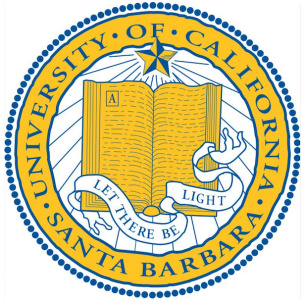


Motivation

- Vertical Scaling has its place, but horizontal scaling is generally preferable.

- For web servers, we do this through load balancing
  - Making many servers appear as one.
  - Users experience is the same, regardless of which machine ultimately serves the request.

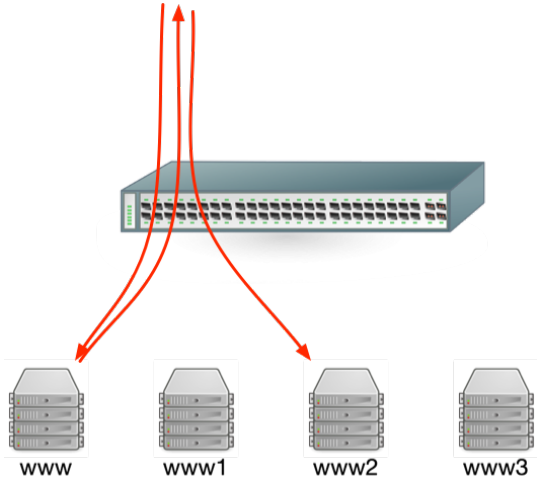
- If we can make many servers appear as one, we get some other advantages:
  - High availability
  - Rolling upgrades



Load Balancing on the Web

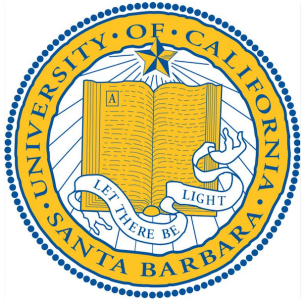
Idea #1: HTTP Redirects

- Discussed in section 6.1 of the reading
- Implemented using multiple web servers with different domain names
- www.domain.com uses http status 301 or 302 to redirect users to a pool of possible hosts.



Load Balancing on the Web

```
% nc www.domain.com 80
GET / HTTP/1.1
host: www.domain.com

HTTP/1.1 301 Moved Permanently
Date: Wed, 15 Oct 2014 21:08:22 GMT
Server: Apache/2.2.22 (Ubuntu)
Location: http://www2.domain.com/
Content-Type: text/html; charset=iso-8859-1
...
```



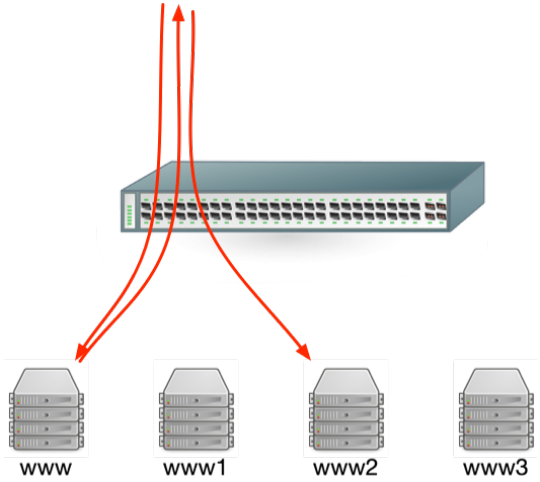
Load Balancing on the Web

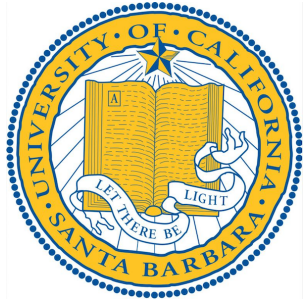
Strengths:

- Simple
- Complete control over load balancing algorithm.
- Location independent.

Weaknesses:

- Visible to user: URL bar, bookmarks, etc.
- High availability isn't really helped much since we rely on www always being up to server redirects

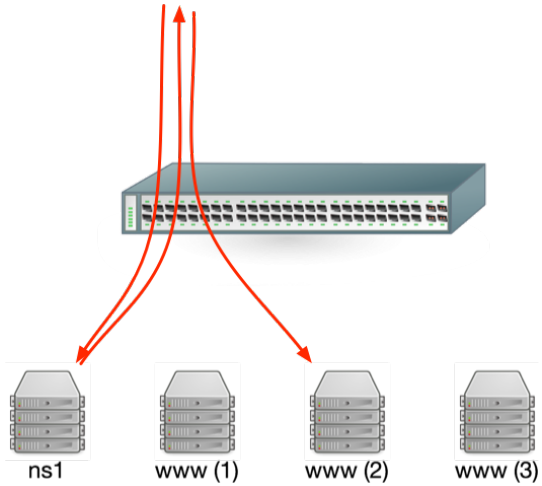




Load Balancing on the Web

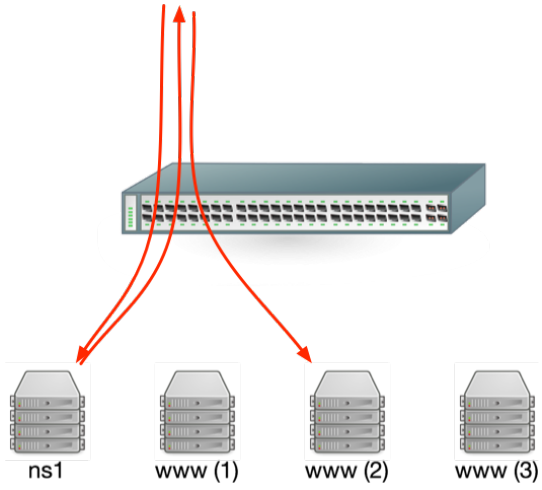
Idea #2: Round Robin DNS

- Discussed in section 4.1 of the reading
- When user's browser queries DNS for www.example.com, a list of IPs is returned.
- User's browser chooses which IP to connect to (generally the first listed)



Load Balancing on the Web

```
% host www.google.com
www.google.com has address 74.125.224. 48
www.google.com has address 74.125.224. 51
www.google.com has address 74.125.224. 52
www.google.com has address 74.125.224. 49
www.google.com has address 74.125.224. 50

% host www.google.com
www.google.com has address 74.125.224. 49
www.google.com has address 74.125.224. 52
www.google.com has address 74.125.224. 51
www.google.com has address 74.125.224. 50
www.google.com has address 74.125.224. 48
```



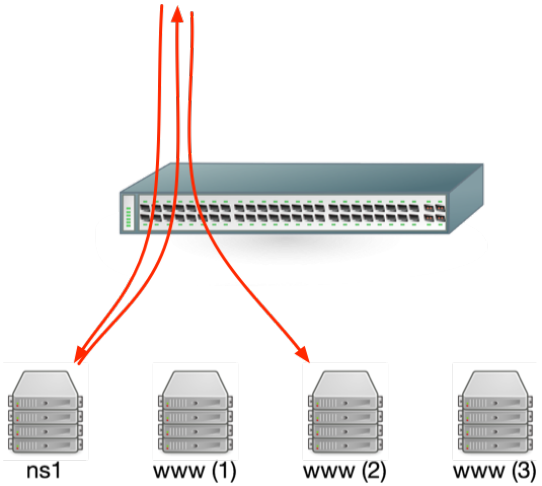
Load Balancing on the Web

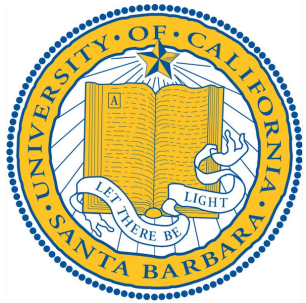
Strengths:

- Easy
- Cheap
- Simple

Weaknesses:

- Less control over balancing
- Modifying the list is inhibited by caching in the browser and proxies

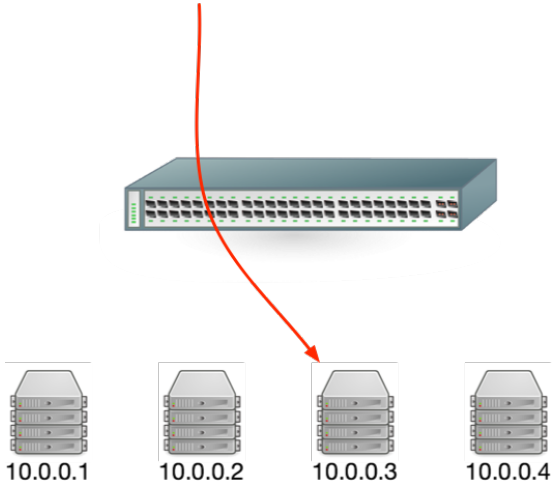


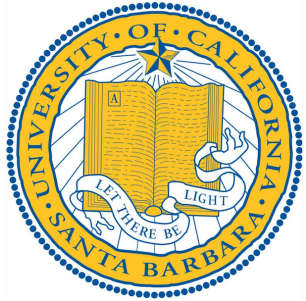


Load Balancing on the Web

Idea #3: Load Balancing Switch

- Discussed in section 4.1 of the reading
- AKA: “TCP Load Balancing”
- The idea: rewrite TCP packets to send them to the correct server
  - Addresses, sequence numbers and checksums need to be rewritten on the fly
  - Constructed using hardware (ASICs) to perform this fast
- Commercial products available:
  - Cisco Content Services Switch
  - Citrix Netscaler
  - F5 Big IP





Load Balancing on the Web

```
__ client -> switch : 216.64.159.149 -> 208.50.157.136
IP D=208.50.157.136 S=216.64.159.149 LEN=60, ID=48397 2.94950 216.64.159.149 -> 208.50.157.136
TCP D=80 S=1421 Syn Seq=899863543 Len=0 Win=32120 …

__ switch -> client : 208.50.157.136 -> 216.64.159.149
IP D=216.64.159.149 S=208.50.157.136 LEN=48, ID=26291 2.95125 208.50.157.136 -> 216.64.159.149
TCP D=1421 S=80 Syn Ack=899863544 Seq=1908949446 Len=0 ...

__ client -> switch : 216.64.159.149 -> 208.50.157.136
IP D=208.50.157.136 S=216.64.159.149 LEN=40, ID=48400 2.98324 216.64.159.149 -> 208.50.157.136
TCP D=80 S=1421 Ack=1908949447 Seq=899863544 Len=0 ...

__ client -> switch : 216.64.159.149 -> 208.50.157.136
IP D=208.50.157.136 S=216.64.159.149 LEN=154, ID=48401 2.98395 216.64.159.149 -> 208.50.157.136
TCP D=80 S=1421 Ack=1908949447 Seq=899863544 Len=114 ... 2.98395 216.64.159.149 -> 208.50.157.136
HTTP GET /eb/images/ec_home_logo_tag.gif HTTP/1.0
```

Load Balancing on the Web

```
__ switch -> server : 216.64.159.149 -> 10.16.100.121
IP D=10.16.100.121 S=216.64.159.149 LEN=48, ID=26292 0.00000 216.64.159.149 -> 10.16.100.121
TCP D=80 S=1421 Syn Seq=899863543 Len=0 Win=32120 Options...


__ server -> switch : 10.16.100.121 -> 216.64.159.149
IP D=216.64.159.149 S=10.16.100.121 LEN=44, ID=22235 0.00001 10.16.100.121 -> 216.64.159.149
TCP D=1421 S=80 Syn Ack=899863544 Seq=2156657894 Len=0 ...


__ switch -> server : 216.64.159.149 -> 10.16.100.121
IP D=10.16.100.121 S=216.64.159.149 LEN=154, ID=48401 0.00131 216.64.159.149 -> 10.16.100.121
TCP D=80 S=1421 Ack=2156657895 Seq=899863544 Len=114 ... 0.00131 216.64.159.149 -> 10.16.100.121
HTTP GET /eb/images/ec_home_logo_tag.gif HTTP/1.0
```

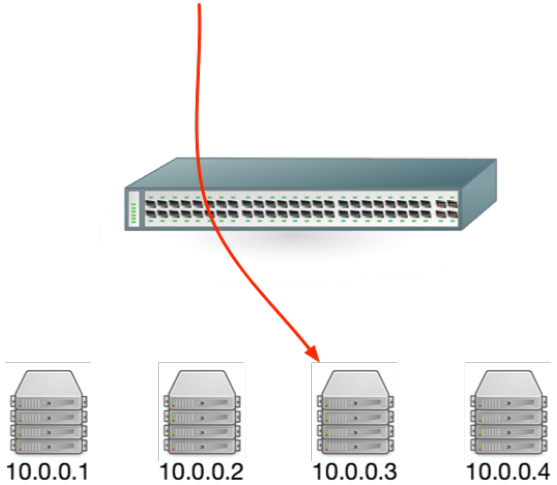
Load Balancing on the Web

```
__ server -> switch : 10.16.100.121 -> 216.64.159.149
IP D=216.64.159.149 S=10.16.100.121 LEN=40, ID=22236 0.00134 10.16.100.121 -> 216.64.159.149
TCP D=1421 S=80 Ack=899863658 Seq=2156657895 Len=0 …

__ switch -> client : 208.50.157.136 -> 216.64.159.149
IP D=216.64.159.149 S=208.50.157.136 LEN=40, ID=22236 2.98619 208.50.157.136 -> 216.64.159.149
TCP D=1421 S=80 Ack=899863658 Seq=1908949447 Len=0 ...

__ server -> switch : 10.16.100.121 -> 216.64.159.149
IP D=216.64.159.149 S=10.16.100.121 LEN=1500, ID=22237 0.00298 10.16.100.121 -> 216.64.159.149
TCP D=1421 S=80 Ack=899863658 Seq=2156657895 Len=1460 ... 0.00298 10.16.100.121 -> 216.64.159.149
HTTP HTTP/1.1 200 OK

__ switch -> client : 208.50.157.136 -> 216.64.159.149
IP D=216.64.159.149 S=208.50.157.136 LEN=1500, ID=22237 2.98828 208.50.157.136 -> 216.64.159.149
TCP D=1421 S=80 Ack=899863658 Seq=1908949447 Len=1460 ... 2.98828 208.50.157.136 -> 216.64.159.149
HTTP HTTP/1.1 200 OK
```

Load Balancing on the Web

Strengths:

- More control over which requests go to which servers
- Works fine with HTTPS



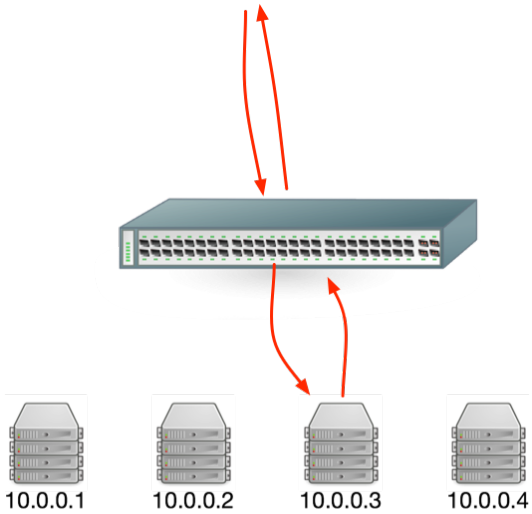
Weaknesses:

- Constraints on where the servers are
- Complicated

Load Balancing on the Web

Idea #4: Load Balancing Proxy

- Also known as “Layer 7 load balancing”
- Terminate HTTP requests: act like a web server
- Issue “back-end” HTTP requests to real web servers to get responses.
- Many hardware products available:
  - Citrix Netscaler
  - BigIP F5
- Most HTTP servers have modules to do this as well.



Load Balancing on the Web

Strengths:

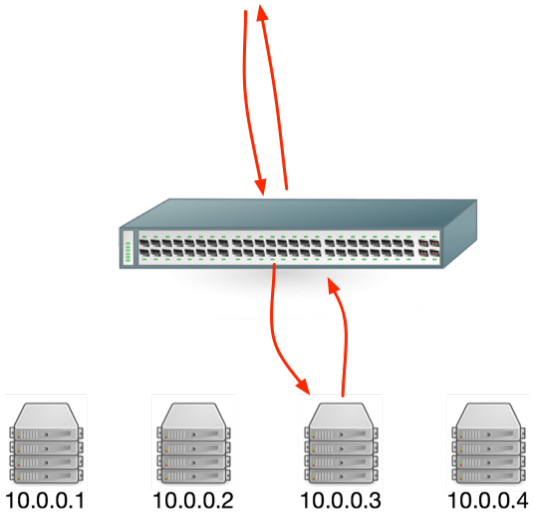
- Cleaner implementation than packet rewriting

Weaknesses:

- Load balancer needs to be doing more work
- SSL is more complicated

This is the most popular technique in use today.

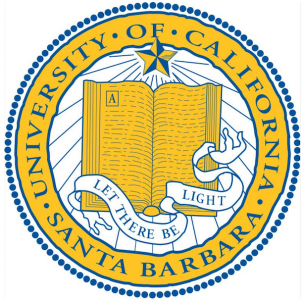
- Technique #3 also used as well



Load Balancing on the Web

Let's say you are designing a load balancing proxy…

- You see every request go in and out.
- What algorithm would you choose for balancing between machines?

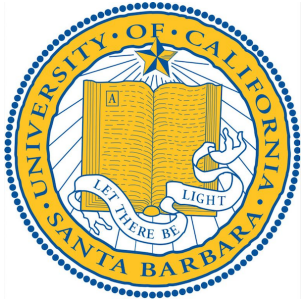


Load Balancing on the Web

Let's say you are designing a load balancing proxy...

- You see every request go in and out.
- What algorithm would you choose for balancing between machines?
  - Random
  - Round robin
  - Least number of connections
  - Fastest response time
  - Bandwidth per Server
  - Based on URI (e.g. /images, or /cgi-bin)

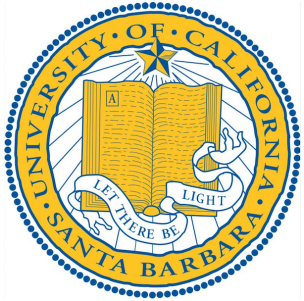
- What are strengths of each of these?



Load Balancing on the Web

Some interesting challenges come up when designing a load balancing proxy…

- Detecting Server Failures
- Session persistence & affinity
- Connection pooling

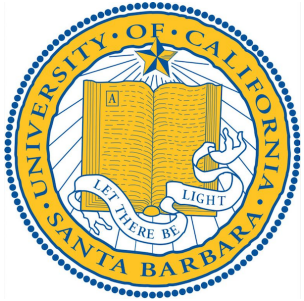


Load Balancing on the Web

Detecting Server Failures

How do we know when a member of our pool has died?

- We can observe traffic: are requests being serviced? Some requests just take a long time.
- We can probe the server
  - Various protocols
    - ICMP ping: test network and kernel
    - TCP connection set up: process is running
    - HTTP HEAD: it's serving pages
    - SNMP: server load



Load Balancing on the Web

Session persistence & affinity

Can we redirect users back to the same web server they used before?

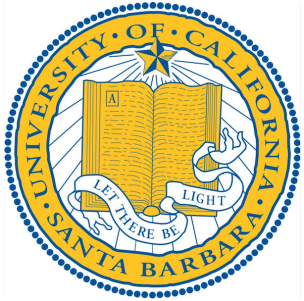
- This can get us caching improvements
- It's difficult to do well
  - Affinity based on client IP address
    - Client IPs change & IPs can be shared
  - HTTP Cookie
    - Works, but needs proxy configuration
  - Session ID in URL
    - Hard to parse out in a load balancer (http://..../.../.../...?...&...&SID=01234&...)
    - Your sessions are in your URL??

Load Balancing on the Web

Connection Pooling

Can we multiplex many separate web requests on to fewer persistent requests?

- Saves on repeated TCP setup
- Reduce idle waiting on server for reads and writes



Load Balancing on the Web

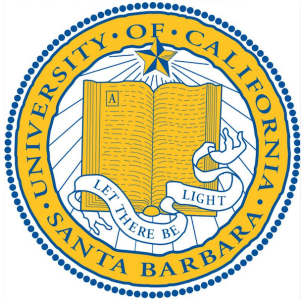
Common Load Balancers

Software

- Apache
- HAProxy
- Varnish
- Pound
- Squid
- Nginx
- ....

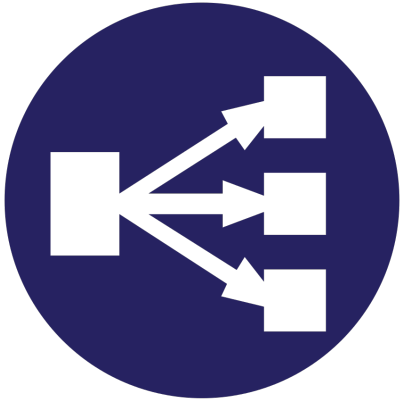
Hardware

- F5 Big IP
- Citrix Netscaler
- Cisco



Load Balancing on the Web

Amazon Elastic Load Balancer

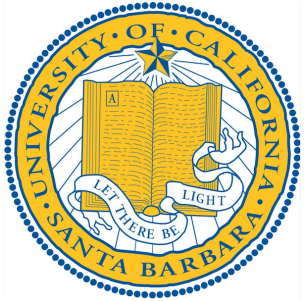


Load balancing as a service

- $0.025 per hour plus $0.008 per GB processed
- Works with EC2 autoscaling
- Supports SSL termination

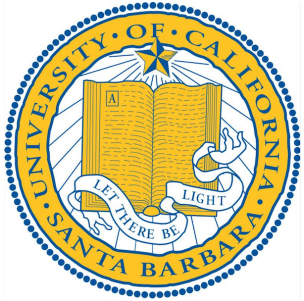
Can load balance between Availability Zones within a Region

Can’t load balance between regions, rely on Route 53 DNS for region failover.



For Today

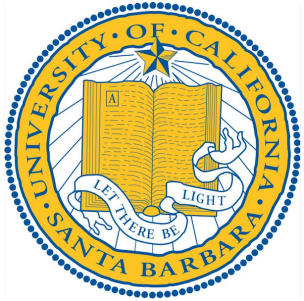
- Motivation
- Load Balancing on the Web
- **High Availability Infrastructure**
- For Next Time...



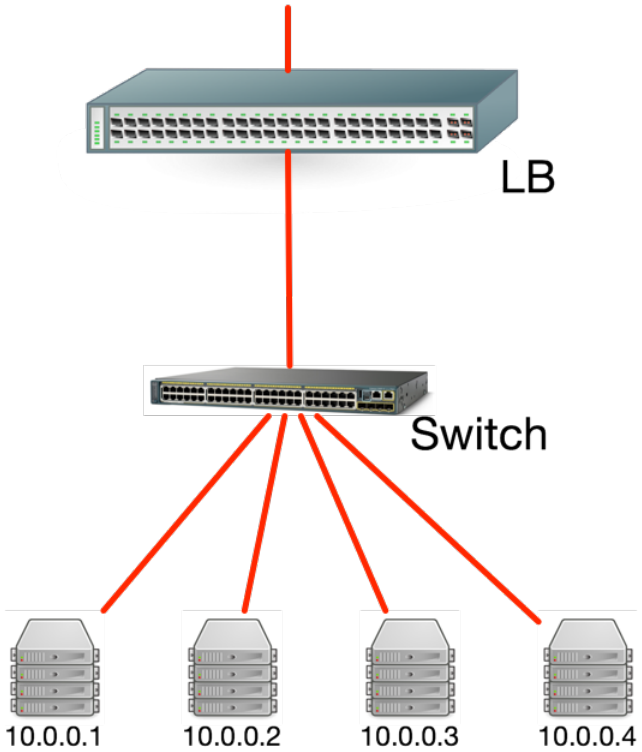
High Availability

Modern web applications power some very important parts of our lives

- Banking, Medical, Telephony (increasingly), etc.
- High availability is increasingly important.
- A common phrase targeted by businesses is “X nines”
  - Three nines = 99.9% uptime =~ 45 minutes a month down
  - Four nines = 99.99% uptime =~ 5 minutes a month down
    - Business applications
  - Five nines = 99.999% uptime =~ five minutes a year down
    - Communications companies



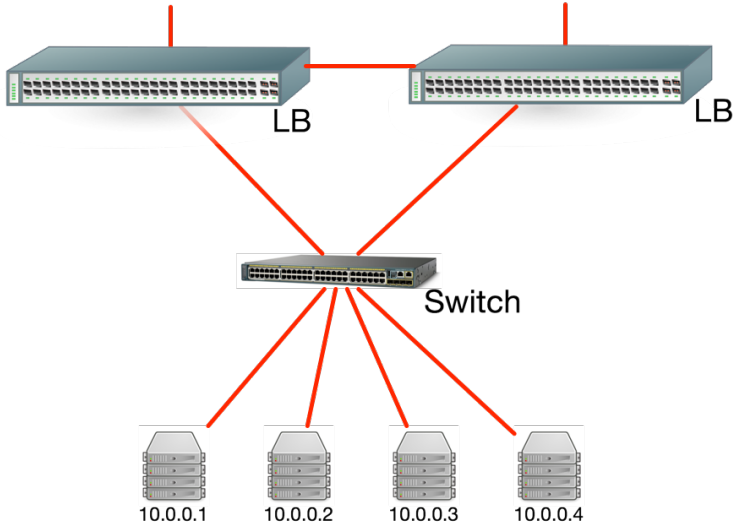
High Availability



What are possible causes of failures?

- Server fails?
  - Have we already handled this?
- Load balancer fails?
- Switch fails?
- Internet fails?
- Entire datacenter fails?

High Availability



Load balancer fails?

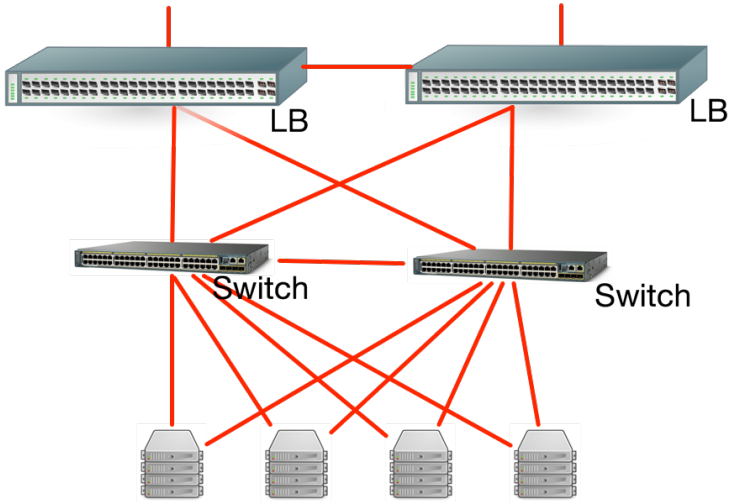
- Lets buy two: primary & failover
- Load balancers use heartbeats to determine health
- During failover, what happens to
  - Established flows?
  - IP address?

High Availability

HSRP: Hot Standby Router Protocol

- Proprietary Cisco protocol, but specified in RFC 2281
- Two devices each have their own IP, but also a shared IP
- During normal operation, the primary responds to ARP requests for the shared IP with a specific MAC address
- When failure is detected, the secondary immediately starts responding to ARP requests for the shared IP with the same MAC address.

High Availability



Switch fails?

- Same answer!
- Lets buy two: primary & failover
- Switches use heartbeats to determine health
- During failover, same issues regarding IP change, but no sessions to fail

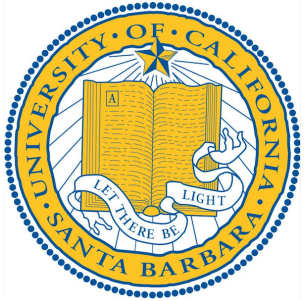
High Availability

Internet fails?

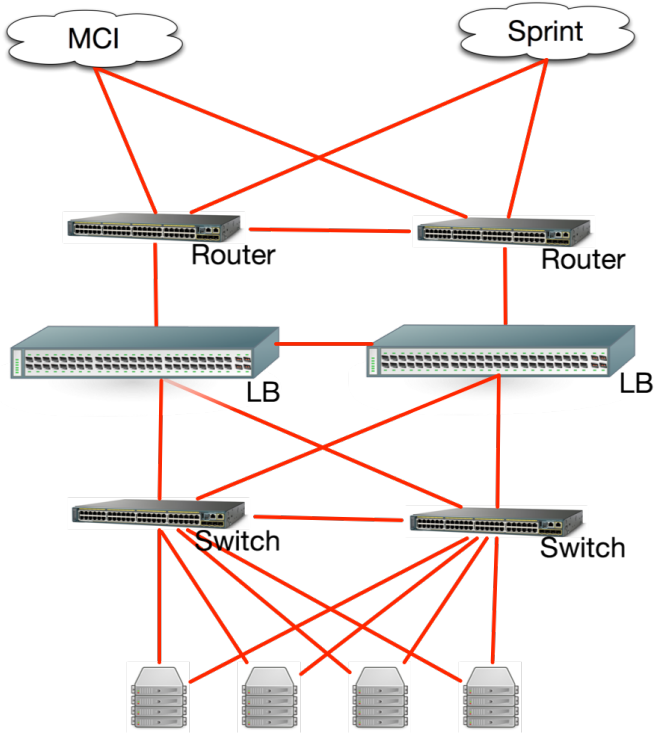
- Lets have two ISPs.
  - One link to, say, Sprint and another to MCI.

How do we handle routing when we have two ISPs?

- Outgoing traffic is easy, since we control these decisions.
  - Pick the cheapest or most reliable link
  - Pick the “closer” link
- Incoming traffic is hard
  - We can’t directly tell clients how to reach our website
  - We need to use BGP to persuade clients
    - Prepending, community strings

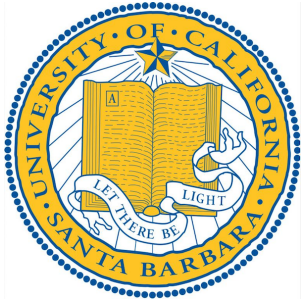


High Availability



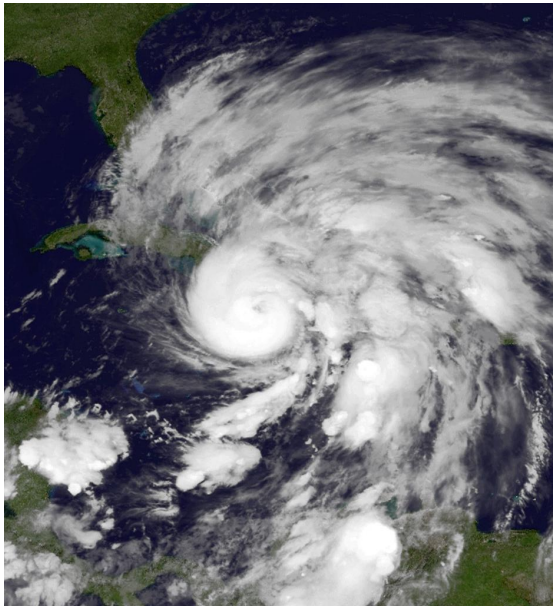
High Availability

Ok so we're good right? Nothing can go wrong?



High Availability

Ok so we're good right? Nothing can go wrong?



High Availability

Hurricane Sandy takes data centers offline with flooding, power outages

Hosting customers stranded as generators in NY data centers run out of fuel.

by **Jon Brodkin** - Oct 30 2012, 9:25am PDT

Share Tweet 100

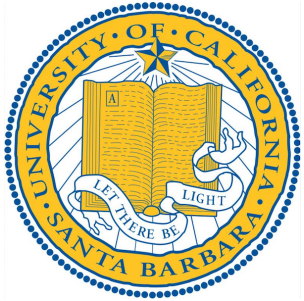
High Availability

Availability Axiom (Pete Tenereillo):

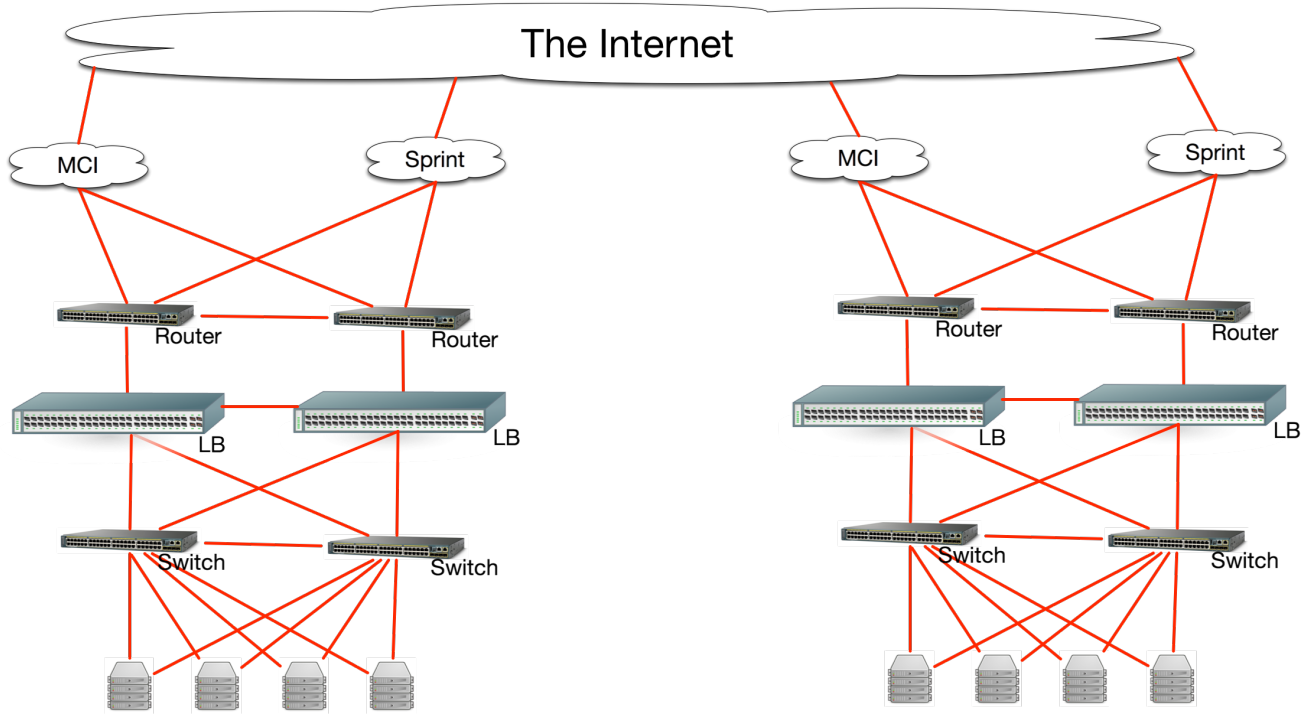
- The only way to achieve high-availability for browser based clients is the include the use of multiple A-records.

Result:

- For performance, we want to send the browser to one datacenter.
- For availability, we want to send the browser multiple A records.
- We end up having to make a choice between performance and availability.



High Availability



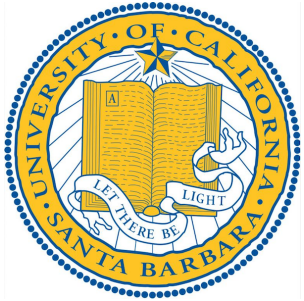
High Availability

Availability Axiom (Pete Tenereillo):

- The only way to achieve high-availability for browser based clients is the include the use of multiple A-records.

Result:

- For performance, we want to send the browser to one datacenter.
- For availability, we want to send the browser multiple A records.
- We end up having to make a choice between performance and availability.



For Next Time...

By next Wednesday, have a group and a project and be prepared to write down some initial features.

Read chapters 9 through 17 in AWDR.