CSCI-1680
Web Performance and Content Distribution

Rodrigo Fonseca

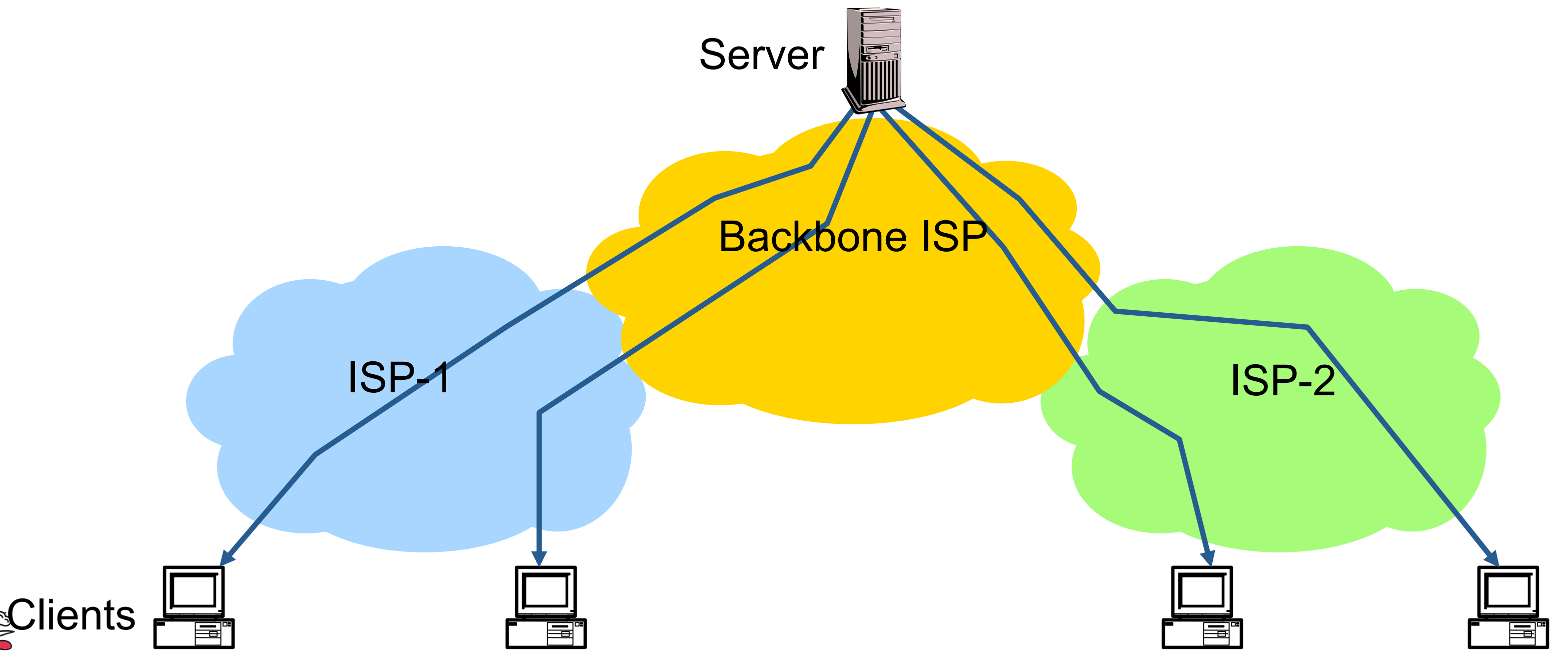
Based partly on lecture notes by Scott Shenker and John Jannotti

Last time

- **HTTP and the WWW**
- **Some performance issues**
  - Persistent Connections, Pipeline, Multiple Connections
  - Caching
- **Today**
  - More on Caching
  - Content Distribution Networks

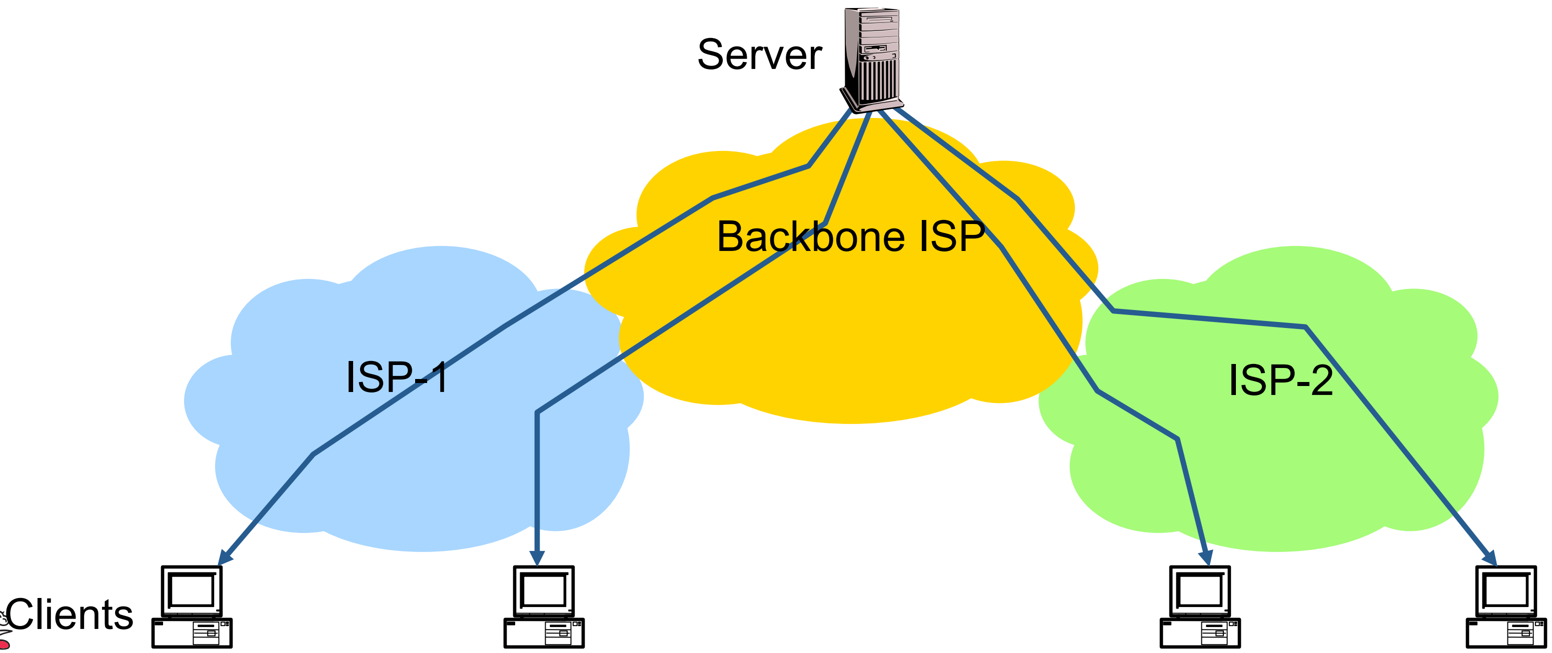
Caching

- **Why cache content?**
  - Client (browser): avoid extra network transfers
  - Server: reduce load on the server
  - Service Provider: reduce external traffic



Caching

- **Why caching works?**
  - Locality of reference:
    - Users tend to request the same object in succession
    - Some objects are popular: requested by many users



How well does caching work?

- **Very well, up to a point**
  - Large overlap in requested objects
  - Objects with one access place upper bound on hit ratio
- **Example: Wikipedia**
  - About 400 servers, 100 are HTTP Caches (Squid)
  - 85% Hit ratio for text, 98% for media

HTTP Cache Control

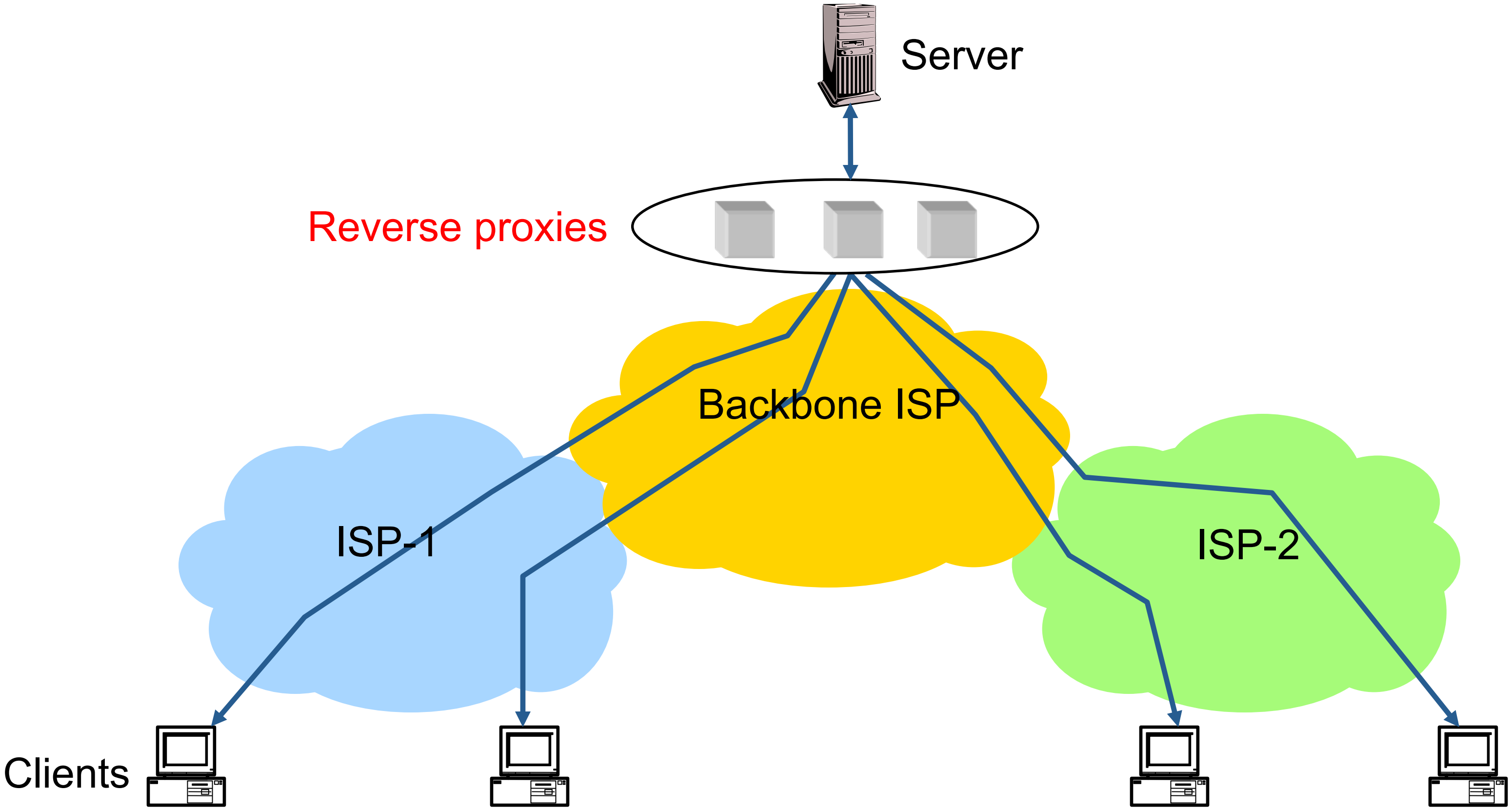
```
Cache-Control   = "Cache-Control" ":" 1#cache-directive
cache-directive = cache-request-directive
| cache-response-directive
cache-request-directive =
  "no-cache"                          ; Section 14.9.1
| "no-store"                          ; Section 14.9.2
| "max-age" "=" delta-seconds         ; Section 14.9.3, 14.9.4
| "max-stale" [ "=" delta-seconds ]   ; Section 14.9.3
| "min-fresh" "=" delta-seconds       ; Section 14.9.3
| "no-transform"                      ; Section 14.9.5
| "only-if-cached"                    ; Section 14.9.4
| cache-extension                     ; Section 14.9.6

cache-response-directive =
  "public"                               ; Section 14.9.1
| "private" [ "=" <"> 1#field-name <"> ] ; Section 14.9.1
| "no-cache" [ "=" <"> 1#field-name <"> ]; Section 14.9.1
| "no-store"                             ; Section 14.9.2
| "no-transform"                         ; Section 14.9.5
| "must-revalidate"                      ; Section 14.9.4
| "proxy-revalidate"                     ; Section 14.9.4
| "max-age" "=" delta-seconds            ; Section 14.9.3
| "s-maxage" "=" delta-seconds           ; Section 14.9.3
| cache-extension                        ; Section 14.9.6

cache-extension = token [ "=" ( token | quoted-string ) ]
```

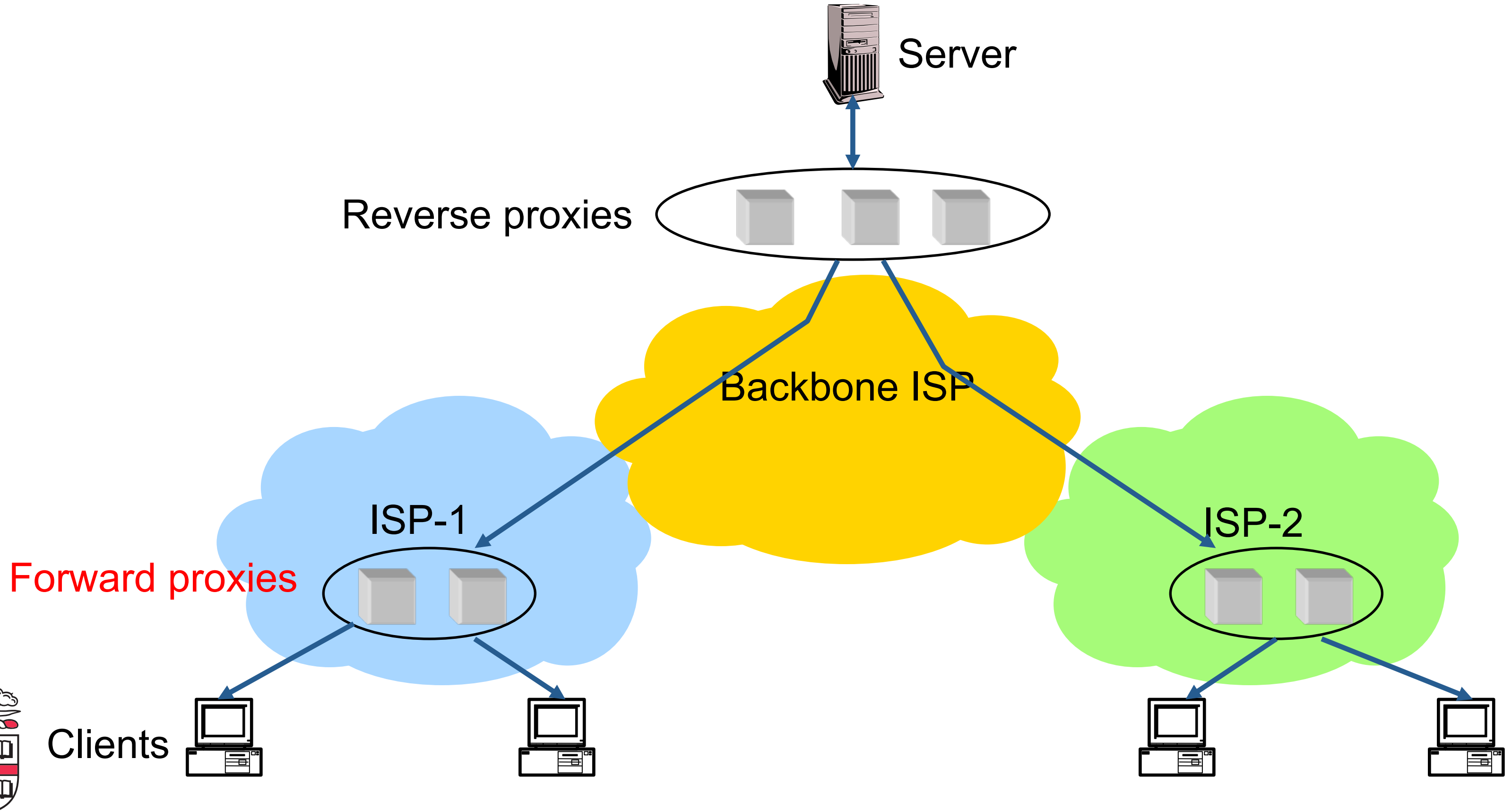
Reverse Proxies

- **Close to the server**
  - Also called Accelerators
  - Only work for static content



Forward Proxies

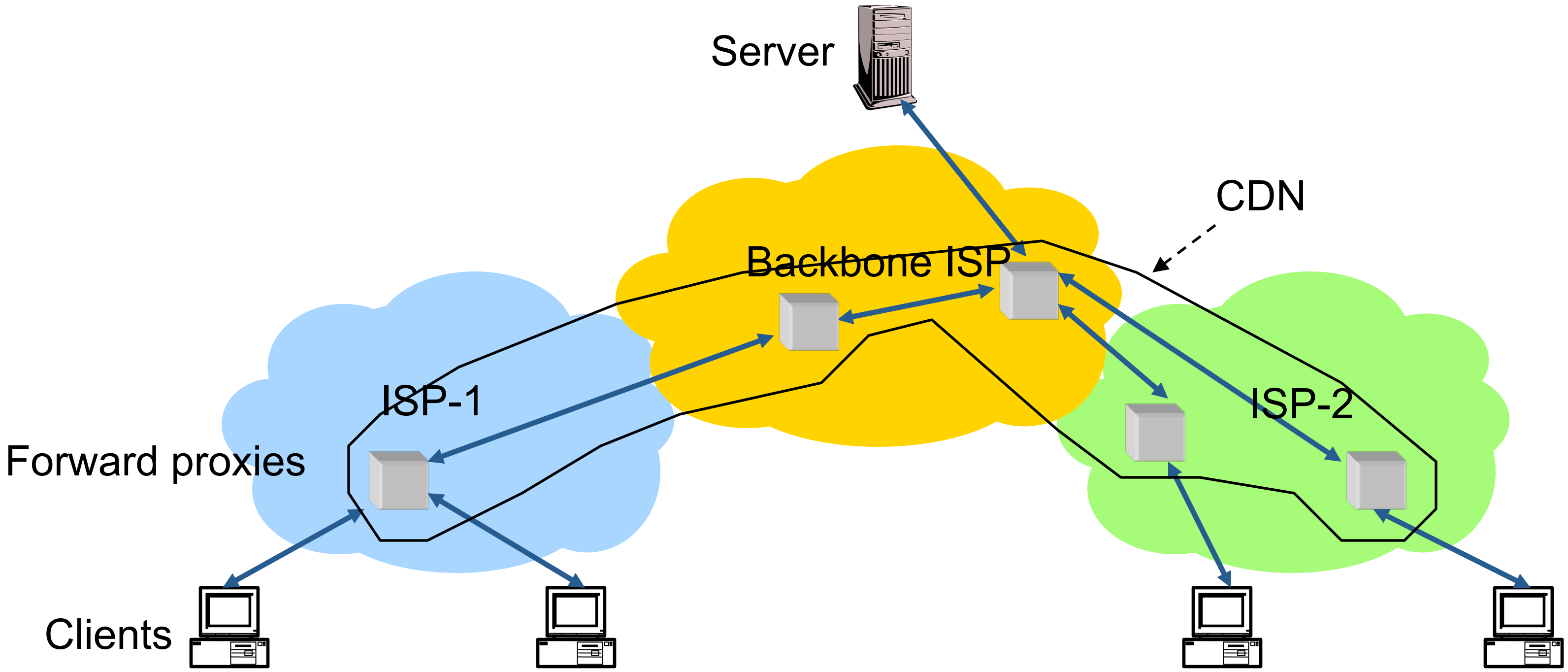
- **Typically done by ISPs or Enterprises**
  - Reduce network traffic and decrease latency
  - May be transparent or configured



Content Distribution Networks

- **Integrate forward and reverse caching**
  - One network generally administered by one entity
  - E.g. Akamai
- **Provide document caching**
  - Pull: result from client requests
  - Push: expectation of high access rates to some objects
- **Can also do some processing**
  - Deploy code to handle some dynamic requests
  - Can do other things, such as transcoding

Example CDN



How Akamai works

- **Akamai has cache servers deployed close to clients**
  - Co-located with many ISPs
- **Challenge: make same domain name resolve to a proxy close to the client**
- **Lots of DNS tricks. BestBuy is a customer**
  - Delegate name resolution to Akamai (via a CNAME)
- **From Brown:**

```
dig www.bestbuy.com
;; ANSWER SECTION:
www.bestbuy.com. 3600     IN   CNAME    www.bestbuy.com.edgesuite.net.
www.bestbuy.com.edgesuite.net. 21600 IN    CNAME    a1105.b.akamai.net.
a1105.b.akamai.net.  20   IN   A   198.7.236.235
a1105.b.akamai.net.  20   IN   A   198.7.236.240
```

  - Ping time: 2.53ms
- **From Berkeley, CA:**

```
a1105.b.akamai.net.  20   IN   A   198.189.255.200
a1105.b.akamai.net.  20   IN   A   198.189.255.207
```

  - Pint time: 3.20ms

DNS Resolution

```
dig www.bestbuy.com
;; ANSWER SECTION:
www.bestbuy.com. 3600     IN   CNAME    www.bestbuy.com.edgesuite.net.
www.bestbuy.com.edgesuite.net. 21600 IN    CNAME    a1105.b.akamai.net.
a1105.b.akamai.net.   20   IN   A    198.7.236.235
a1105.b.akamai.net.   20   IN   A    198.7.236.240
;; AUTHORITY SECTION:
b.akamai.net.    1101 IN   NS   n1b.akamai.net.
b.akamai.net.    1101 IN   NS   n0b.akamai.net.
;; ADDITIONAL SECTION:
n0b.akamai.net.  1267 IN   A    24.143.194.45
n1b.akamai.net.  2196 IN   A    198.7.236.236
```

- **n1b.akamai.net finds an edge server close to the client's local resolver**
  - Uses knowledge of network: BGP feeds, traceroutes. *Their secret sauce…*

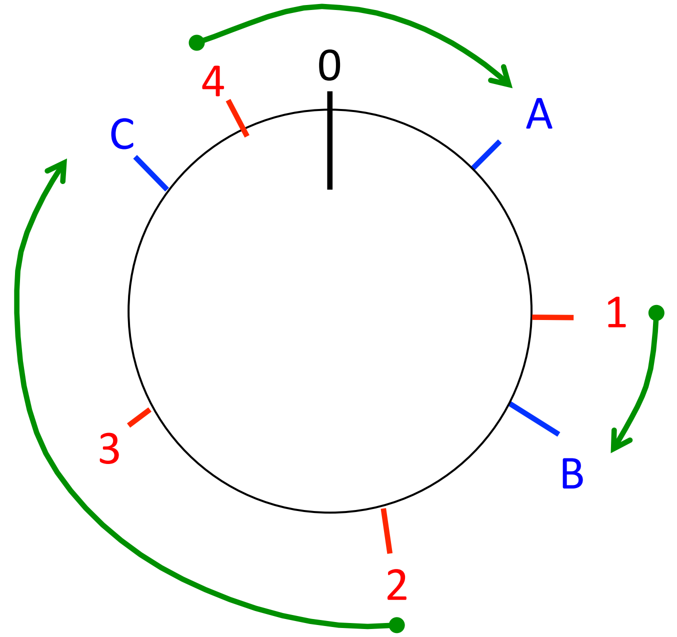
What about the content?

- **Say you are Akamai**
  - Clusters of machines close to clients
  - Caching data from many customers
  - Proxy fetches data from *origin* server first time it sees a URL
- **Choose cluster based on client network location**
- **How to choose server within a cluster?**
- **If you choose based on client**
  - Low hit rate: N servers in cluster means N cache misses per URL

Straw man: modulo hashing

- **Say you have N servers**
- **Map requests to proxies as follows:**
  - Number servers 0 to N-1
  - Compute hash of URL: h = hash (URL)
  - Redirect client to server #p = h mod N
- **Keep track of load in each proxy**
  - If load on proxy #p is too high, try again with a different hash function (or "salt")
- **Problem: most caches will be useless if you add or remove proxies, change value of N**

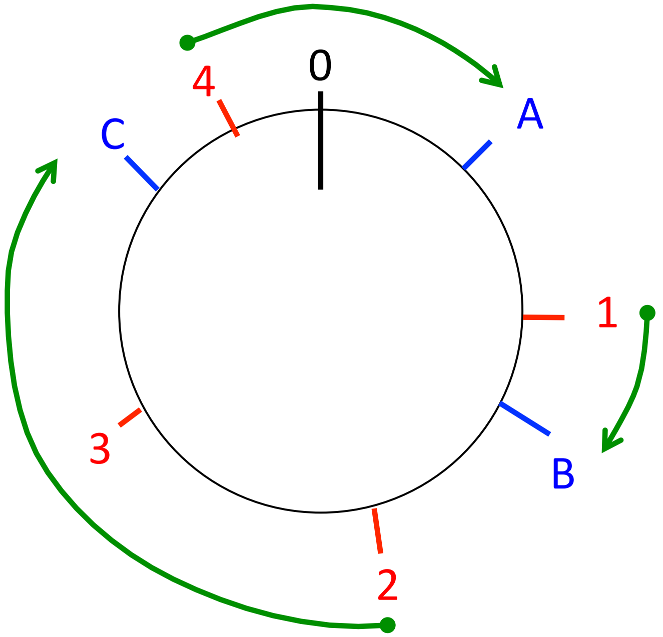
Consistent Hashing [Karger et al., 99]



Object	Cache
1	B
2	C
3	C
4	A

- **URLs and Caches are mapped to points on a circle using a hash function**
- **A URL is assigned to the closest cache clockwise**
- **Minimizes data movement on change!**
  - When a cache is added, only the items in the preceding segment are moved
  - When a cache is removed, only the next cache is affected

Consistent Hashing [Karger et al., 99]



Object	Cache
1	B
2	C
3	C
4	A

- **Minimizes data movement**
  - If 100 caches, add/remove a proxy invalidates ~1% of objects
  - When proxy overloaded, spill to successor
- **Can also handle servers with different capacities. How?**
  - Give bigger proxies more random points on the ring

CoralCDN

- **What if a content provider can't pay a CDN?**
  - Slashdotted servers
- **CoralCDN is a clever response to that**
- **Say you want to access**

  http://www.cs.brown.edu/courses/cs168
- **Instead, try to access**

  http://www.cs.brown.edu.**nyud.net**/courses/cs168
- **What does this accomplish?**

CoralCDN

http://www.cs.brown.edu.**nyud.net**/courses/cs168

- – Resolution controlled by the owner of nyud.net
- – CoralCDN runs a set of DNS servers and a set of HTTP proxies
- – DNS servers return an HTTP proxy close to the client

- **The HTTP proxies form a Distributed Hash Table, mapping (url -> {proxies})**
  - – The mapping for a URL is stored in the server found by a technique similar to consistent hashing
- **The HTTP proxy can:**
  1. Return the object if stored locally
  2. Fetch it from another CoralCDN proxy if stored there
  3. Fetch it from the origin server
  4. In case of 3 or 4, store the object locally

Summary

- **HTTP Caching can greatly help performance**
  - Client, ISP, and Server-side caching
- **CDNs make it more effective**
  - Incentives, push/pull, well provisioned
  - DNS and Anycast tricks for finding close servers
  - Consistent Hashing for smartly distributing load

Next time

- Justin DeBrabant will talk about Application Layer Data, or *how to write your own application layer protocol…*