HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

Internet Content Distribution

Chapter 4: Content Distribution Networks

Jussi Kangasharju

Chapter Outline

- Basics of content distribution networks (CDN)
  - Why CDN?
  - How do they work?
- Client redirection
  - DNS redirection
  - Full (or total) redirection
  - Selective redirection
- Performance issues of CDNs
- CoralCDN example

What Are CDNs?

- CDN is an architecture for efficient delivery of (web) content to a large number of clients
- CDNs are operated by companies which charge content providers for the delivery services
- CDNs are mostly transparent to the end-user
  - Meaning: You can see CDNs being used only if you look at actual DNS requests or read HTML-source of a page
- Commercial CDNs for actual content delivery:
  - Akamai, Panther Express, SAVVIS, VitalStream
- Academic CDNs for research on content delivery:
  - CoDeeN, CoralCDN, Globule

Why CDN?

- Why is a caching hierarchy not enough?
- What is missing in a caching hierarchy?
- Answer: Nothing, in principle :-)
- In practice, client-side caching has some problems
- Main problem is that content provider has no control over how her content is cached
  - HTTP does define Cache-Control headers, expiry, aging models
  - But: Nothing can force the cache operator to follow them
- Experience has shown cache operators do not always follow “the rules”
  - Result was cache busting (see earlier)

So, Why CDN?

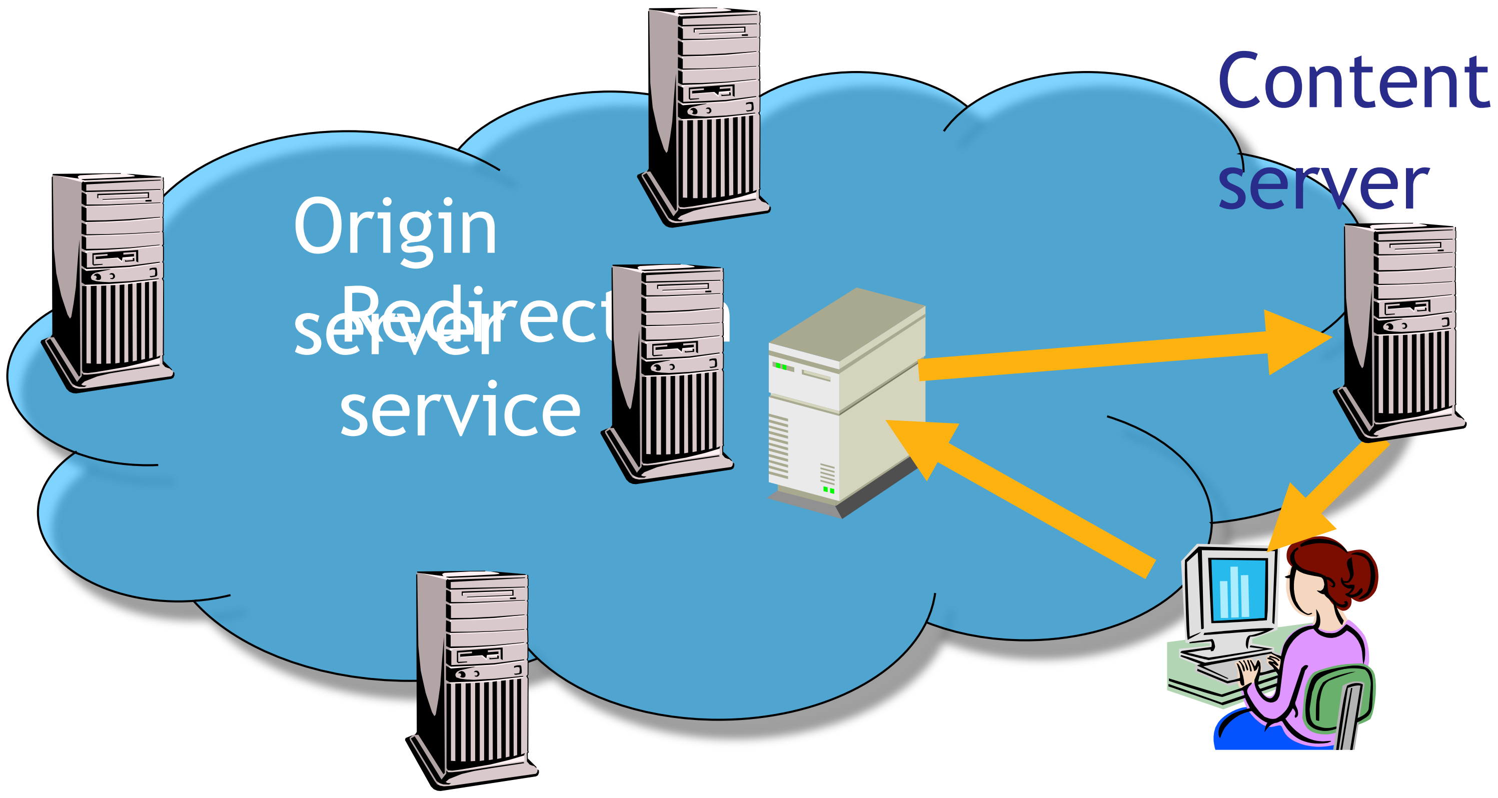
- One of the main goals of CDNs is to put content provider in control over how her content is cached
- Content provider signs a contract with CDN
  - Contract specifies how content can be cached
- Contract also means CDN will follow what content provider wants
- CDNs typically charge per-byte of traffic served
- CDNs can be used for any kind of content
  - Typically main use is for web content
  - Streaming media has also been delivered over CDNs

What Is a CDN?

- CDN operates *content servers*
- Content servers are placed close to users
  - In terms of network distance
- Some or all of the content from the content provider is replicated on the content servers
  - Different content servers might have different content
- Users access content from the “nearest” content server
- Challenges:

1. How to replicate content?
   - Usually happens over a private network
   - Can optimize according to many criteria
2. How to redirect clients?
   - Our main focus here

How Does a CDN Work?



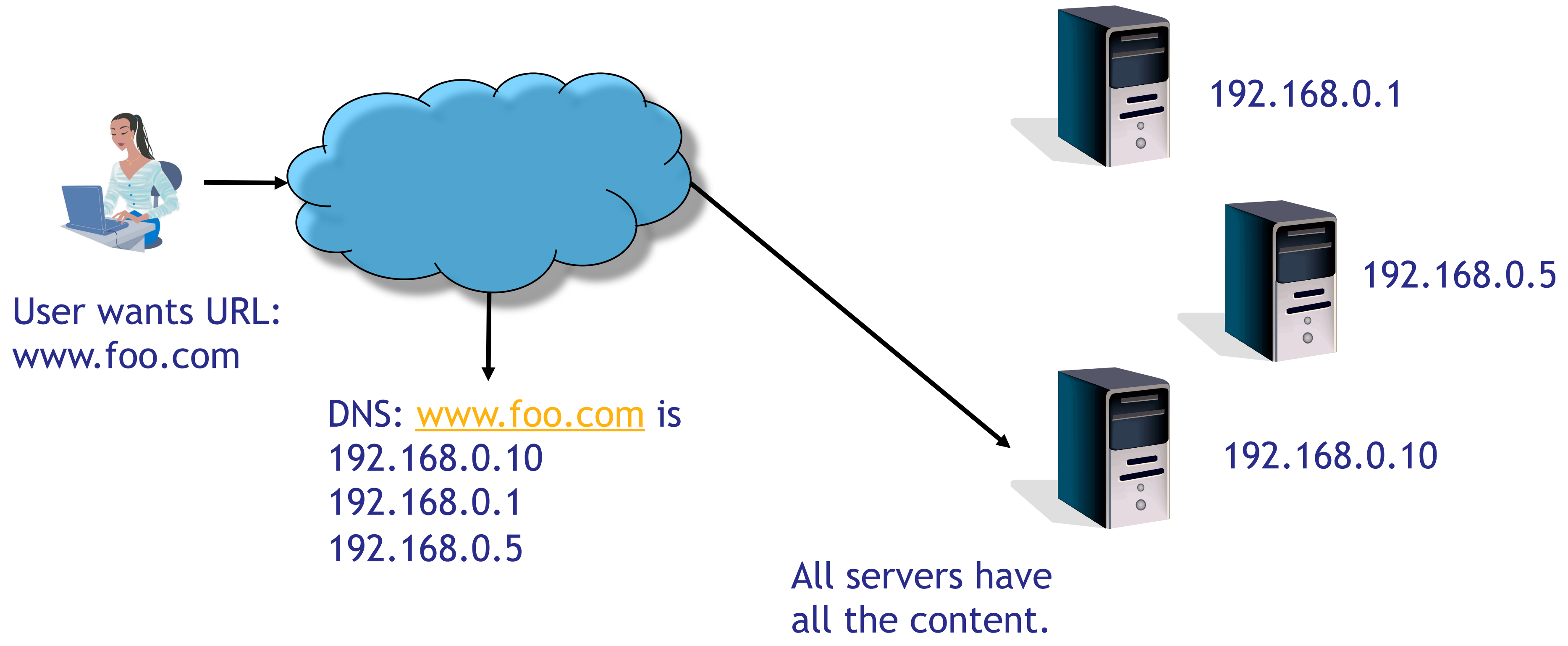
- User sends request to origin server
- Request somehow intercepted by redirection service
- Redirection service forwards user’s request to the “best” content server
- Content served from the content server

Redirection Service

- Key feature of a CDN is *redirection service*
- Practical constraints: (recall)
  - Not allowed to touch client software
  - Redirection should be efficient
- Practical answer: DNS redirection
  - DNS is mandatory to resolve URLs, hence it's always available
  - Works reasonably well
- Two kinds of redirection:
  - Total redirection
  - Selective redirection
- First, let's see how DNS redirection works

Recall: DNS Load Balancing

- DNS load balancing uses DNS to send clients to different content servers
- Reply to DNS query for server name results in several IP addresses
- Client picks one of them and sends request to that server



DNS Redirection vs. DNS Load Balancing

- No real difference between DNS redirection and DNS load balancing in practice
- Both take advantage of the DNS lookups to send traffic to different servers
- Main difference:
  - DNS load balancing typically works with clusters
  - DNS redirection is meant for CDNs
- DNS redirection must take into account:
  - Where is the client located in the network?
  - Where is the closest content server for that client?
- DNS redirection requires a larger supporting infrastructure than load balancing

DNS Redirection Infrastructure

- Client's DNS request comes to CDN's nameserver
  - Somehow, see below for two possibilities
- Typically the request has to go through some steps through the CDN's DNS hierarchy
- Each step redirects the client to a nearby nameserver
- Finally, last nameserver returns the address of a nearby content server
- For the infrastructure, CDN needs to measure the state of the network
  - Needed to determine which servers are the closest
  - Network measurements to determine current state

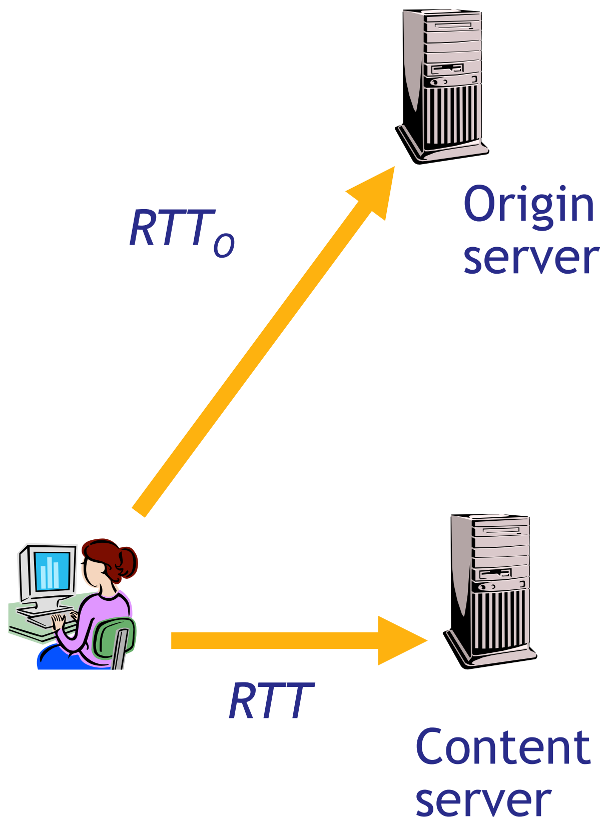
Two Types of Redirection

1. Total redirection
   - Any request for origin server is redirected to CDN
   - Basically, CDN takes control of content provider’s DNS zone
   - Benefit: All requests are automatically redirected
   - Disadvantage: May send lots of traffic to CDN, hence expensive for the content provider
2. Selective redirection
   - Content provider marks which objects are to be served from CDN
   - Typically, larger objects like images are selected
   - Refer to images as: <img src=http://cdn.com/foo/bar/img.gif>
   - When client wants to retrieve image, DNS request for cdn.com gets resolved by CDN and image is fetched from the selected content server
   - Pro: Fine-grained control over what gets delivered
   - Con: Have to (manually) mark content for CDN

Performance of Redirection Schemes

1. Total redirection
   - All requests redirected to content servers
2. Selective redirection
   - Get HTML page from origin server, images from content server
   - Need to open new TCP connection for images

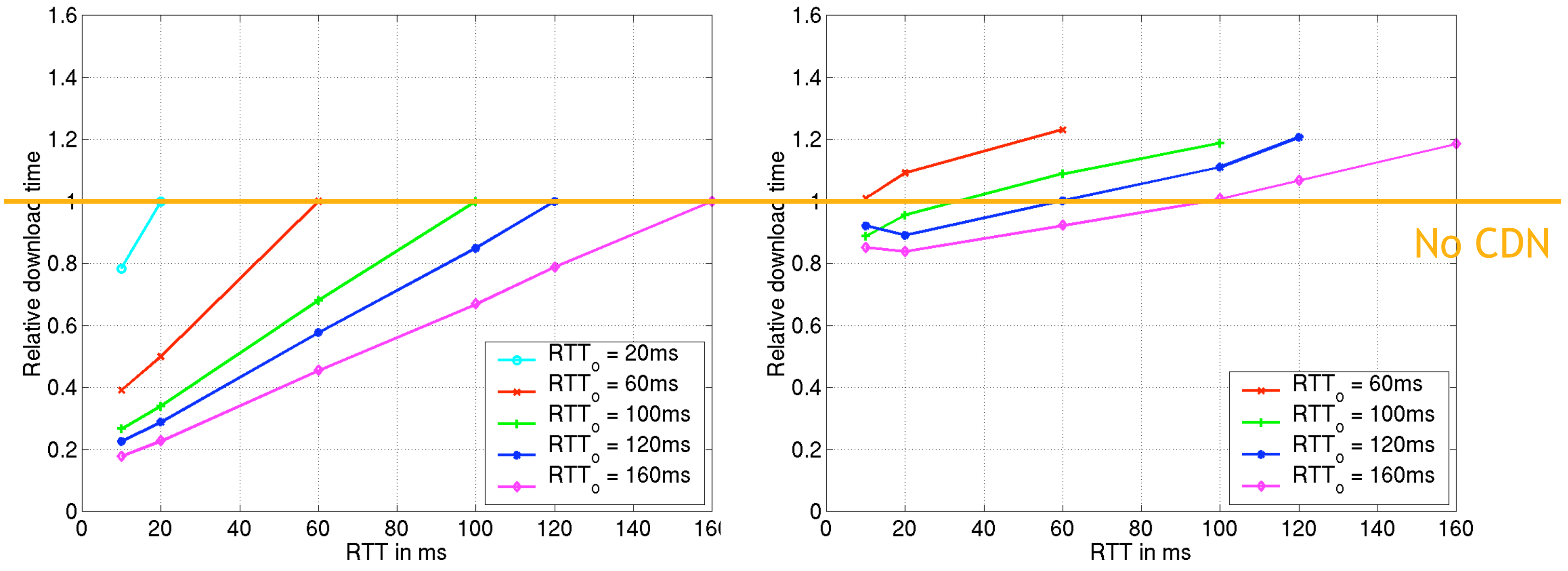
- Question: Which gives better performance to the user?



Redirection Comparison

- Compare total and selective redirection
- Place origin server at a given distance RTT_O
- Place content server at distance RTT
- Insert some HTML and images on both servers
- Vary RTT and RTT_O
- Calculate how long it takes to download full page
  - Results from: J. Kangasharju, K. W. Ross, J. W. Roberts, “Performance Evaluation of Redirection Schemes in Content Distribution Networks”, Computer Communications, Feb. 2001

Redirection Performance



Total redirection

Selective redirection

- Total redirection has superior performance
- Cost of new TCP connections high (slow-start, lost SYN)
- Caveat: Does not include server load or DNS lookups

Conclusion of Redirection Performance

- Total redirection has clearly superior performance
- Selective redirection is typically slower than downloading everything from the origin server
  - But origin server might be loaded…
- Results do not include DNS lookup delays
  - DNS lookups typically fast, but can take very long in worst case
- Which redirection is more used?
- Initially, selective redirection was used
- These days, mainly total redirection

DNS Redirection: Other Issues

- DNS redirection has one (big) problem
- Because redirection is based on DNS queries, the content server is chosen based on who sent that query
- DNS queries do not come from clients, but from the DNS servers used by the clients
- *Why is this a problem?*
- In many cases it’s not a problem
  - For example, clients in a university use university’s nameserver
- In many cases, it’s a big problem
  - Larger ISPs might run only a few nameservers
  - Especially in US for dial-up users, DNS lookups are concentrated
  - This means the content server is optimized for the nameserver, not the actual client
  - The difference can sometimes be very large

CoralCDN

- Let’s take a closer look at a real CDN
- CoralCDN is a peer-to-peer content distribution network
- Mainly aimed at people running smaller websites
- CoralCDN offers such sites high performance and allows them to meet high demand
  - So called Slashdot-effect
- Goals of CoralCDN:

1. Make publishing easy
   - Anybody can participate
2. Avoid hot-spots
   - Volunteer participation might be affected if load gets too high

Why CoralCDN?

- Why we need yet another CDN?
- Commercial CDNs already do their job well
  - In many ways better than CoralCDN
- But:

1. Commercial CDNs are aimed at content providers with lots of money and resources
   - Not cheap to serve content through a CDN
2. Smaller web sites cannot afford a CDN
   - Should need to install bigger servers themselves
   - Again, not affordable

- Hence, smaller content providers are limited in what they can afford to offer
  - Again, Slashdot-effect among other things

CoralCDN: Details

- CoralCDN is based on mirroring
- Volunteers offer their computers as proxies which mirror data from origin servers
- Large number of volunteers provides a large amount of bandwidth for distributing content
  - Every content provider gets good service
  - Level of popularity affects content replication
- CoralCDN is based on DNS redirection
- Cooperation between content servers (caches, proxies) done over Coral
- Coral is a DHT-like indexing infrastructure
  - Not a normal DHT, but similar

CoralCDN: DNS Redirection

- CoralCDN has a simple approach to DNS redirection
- Suppose you want to publish URL for www.foo.com
- You “coralize” the URL, by adding nyud.net
  - Older versions used specific ports as well (8080 and 8090)
- For example: http://www.foo.com/image.jpeg becomes http://www.foo.com.nyud.net/image.jpeg
- The URL has to be changed manually by *someone*
- Someone can be:
  - Content provider
  - Individual user (if a site is too slow)
  - Any third party (e.g., before posting a URL on Slashdot, you coralize it)

CoralCDN: How Does It Work?

- Because the URL ends in nyud.net, DNS queries are handled by the nameserver of nyud.net (Coral’s DNS)
- CoralCDN’s DNS servers figure out the nearest content server to the client
- Client request is then automatically sent to that server
- Content servers are actually proxies
  - Note: Client does not know it’s talking to a proxy!
- Proxies cache content locally or fetch it from other CoralCDN proxies
  - Worst case: Fetch from origin server, but this ideally happens only on the first request
- How to find content stored on other proxies?

CoralCDN: Indexing Infrastructure

- CoralCDN uses a so-called distributed sloppy hash table
- DSHT groups nodes into *clusters*
- Cluster is defined by *diameter*
  - Diameter is the maximum RTT between nodes in cluster
- Hierarchy of diameters builds *levels*
- CoralCDN implemented with 3 levels
  - Level 0 has RTT ∞
  - Level 1 has RTT 60 msec
  - Level 2 has RTT 20 msec
- Preference to nodes in higher levels (= nearby nodes)

Coral DNS Server

- DNS server is responsible for redirecting clients
  - Does it by giving answers to DNS queries
- Uses a DNS feature to map URLs to levels
  - DNS redirection, RFC 2672
  - Allows to map a subtree in DNS to another domain
- In practice:
  - www.foo.com.nyud.net =
    www.foo.com.http.l2.l1.l0.nyucd.net
- Answer contains also nameservers for:
  - l0.nyucd.net, l1.l0.nyucd.net, and l2.l1.l0.nyucd.net
  - Depends on the RTT to the client’s DNS resolver
- DNS resolver on client side will pick the one that matches the most levels

CoralCDN: Getting Content

- Goal is to minimize load on origin server
- Should avoid many proxies contacting origin server
- Hence, should fetch content from other proxies whenever it is possible
- DSHT determines which proxies have the file
- When a proxy starts downloading a file, it inserts itself into the DSHT with a short TTL
    - When download completed, TTL is increased
- Immediate insertion protects against flash-crowds
    - Proxies form a kind of a multicast tree in that case
- Details of indexing, see paper:
    - M. J. Freedman, E. Freudenthal, D. Mazieres, Democratizing content publication with Coral, NSDI 2004

CoralCDN: Performance

- Performance evaluation has shown:

1. CoralCDN helps solve flash crowd problem
2. Clustering provides large performance gains
3. Clusters are suitably formed
4. No hot spots in indexing system

- Evaluation done on PlanetLab
  - 166 machines as clients
  - 1 webserver on a 384 Kbit/s link with 12 41 KB files
  - Clients request files, pick 3 files from 12 (simulate web page)
  - Request rate ~ 100 reqs/sec, total data rate 32800 Kbit/s

CoralCDN: Performance

Load on origin server

- Origin server typically gets about 15 requests
- Should get only 12 requests, since there were 12 files
- Reason: Some requests overlap in the beginning

Clustering

- RTT thresholds divide clusters naturally
- Level 1 clusters with 60 msec RTT
  - Separates Europe from US
- Level 2 clusters with 20 msec RTT
  - East coast vs. West coast

Chapter Summary

- Basics of content distribution networks (CDN)
  - Why CDN?
  - How do they work?
- Client redirection
  - DNS redirection
  - Full (or total) redirection
  - Selective redirection
- Performance issues of CDNs
- CoralCDN example