

CS514: Intermediate Course in Operating Systems

Professor Ken Birman
Vivek Vishnumurthy: TA

How do Web Services really work?

- Today:
 - WSDL: The Web Services Description Language
 - UDDI: The Universal Description, Discovery and Integration standard
 - Roles for brokers in Web Services systems
 - Challenges associated with naming, discovery and translation in large systems

Discovery

- This is the problem of finding the "right" service
 - In our example, we saw one way to do it – with a URL
 - Web Services community favors what they call a URN: Uniform Resource Name
- But the more general approach is to use an intermediary: a discovery service

Example of a repository

Name	Type	Publisher	Toolkit	Language	OS
Web Services Performance and Load Tester	Application	LavaWu		N/A	Cross-Platform
Temperature Service Client	Application	vsnk	Glib	Java	Cross-Platform
Weather Buddy	Application	csharp?24890	MS.NET	C#	Windows
DreamFactory Client	Application	bluemix.com	DreamFactory	Javascript	Cross-Platform
Temperature Perl Client	Example Source	glink13		Perl	Cross-Platform
Apache SOAP sample source	Example Source	xmlbooks.net	Apache SOAP	Java	Cross-Platform
ANSI	Example Source	TVG	SOAP4j	N/A	Cross-Platform
PodcastSOAP demo	Example Source	mimodell	PodcastSOAP	C++	Windows
easysoap temperature	Example Source	glib	EasySOAP++	C++	Windows
Weather Service Client with MS Visual Basic	Example Source	softmine	MS SOAP	Visual Basic	Windows
Temperature Client	Example Source	javalan	MS.NET	C#	Windows

Roles?

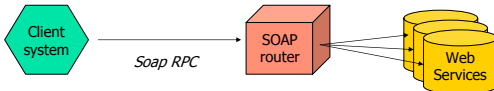
- UDDI is used to write down the information that became a "row" in the repository ("I have a temperature service...")
- WSDL documents the interfaces and data types used by the service
- But this isn't the whole story...

Discovery and naming

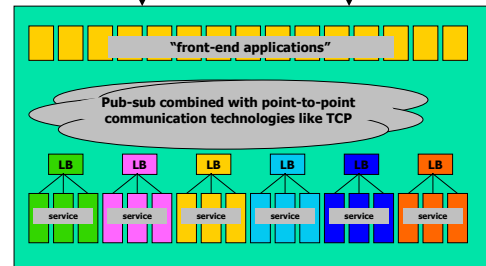
- The topic raises some tough questions
 - Many settings, like the big data centers run by large corporations, have rather standard structure. Can we automate discovery?
 - How to debug if applications might sometimes bind to the wrong service?
 - Delegation and migration are very tricky
 - Should a system automatically launch services on demand?

Client talks to eStuff.com

- One big issue: we're oversimplifying
- We think of remote method invocation and Web Services as a simple chain:



A glimpse inside eStuff.com



Discovery in eStuff.com

- Data centers are increasingly common
- And they raise hard questions!
 - How can a data center in California control decisions a client is making in Ithaca?
 - Services are clustered. How should client request be "routed" to the right member
 - Once you start talking to a server it may cache data for you. How can you be sure to get the right one next time?

CORBA approach

- CORBA had what are called
 - Ways to export specialized client stubs
 - The client stub could include server provided decision logic, like "which data center to connect with"
 - Gives data center a form of remote control
 - Factory services: manufacture certain kinds of objects as needed
 - Effect was that "discovery" can also be a "service creation" activity

CORBA is object oriented

- Seems obvious... and it is. CORBA is centered around the notion of an object
 - Objects can be passive (data)
 - ... active (programs)
 - ... persistent (data that gets saved)
 - ... volatile (state only while running)
- In CORBA the application that manages the object is inseparable from the object
 - And the stub on the client side is part of the application
 - The request per-se is an action by the object on itself and could even exploit various special protocols
 - We can't do this in Web Services

Will Web Services "help" with naming and discovery?

- Web Services tells us how
 - One client can...
 - ... find one server and
 - ... bind to that server and
 - ... send a request that will make sense
 - ... and make sense of the response
- So sure, WS will help

But Web Services won't...

- Allow the data center to control decisions the client makes
- Assist us in implementing naming and discovery in scalable cluster-style services
 - How to load balance? How to replicate data? What precisely happens if a node crashes or one is launched while the service is up?
 - Help with dynamics. For example, best server for a given client can be a function of load but also affinity, recent tasks, etc

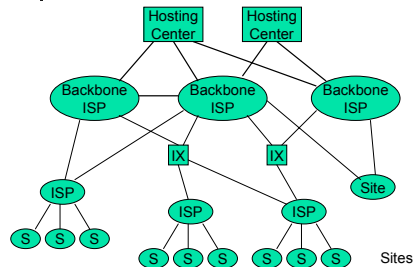
How we do it now

- Client queries directory to find the service
- Server has several options:
 - Web pages with dynamically created URLs
 - Server can point to different places, by changing host names
 - Content hosting companies remap URLs on the fly. E.g. <http://www.akamai.com/www.cs.cornell.edu> (reroutes requests for www.cs.cornell.edu to Akamai)
 - Server can control mapping from host to IP addr.
 - Must use short-lived DNS records; overheads are very high!
 - Can also intercept incoming requests and redirect on the fly

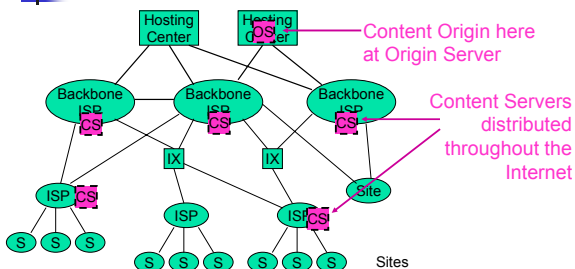
Why this isn't good enough

- The mechanisms aren't standard and are hard to implement
 - Akamai, for example, does content hosting using all sorts of proprietary tricks
- And they are costly
 - The DNS control mechanisms force DNS cache misses and hence many requests do RPC to the data center
- We lack a standard, well supported, solution!

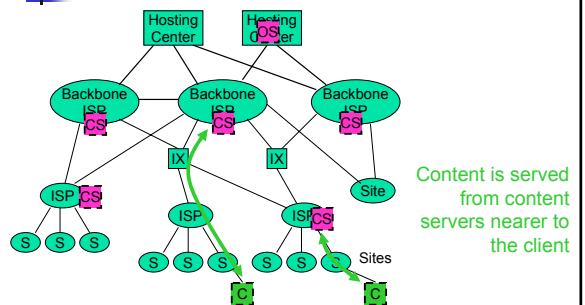
Content Routing Principle (a.k.a. Content Distribution Network)



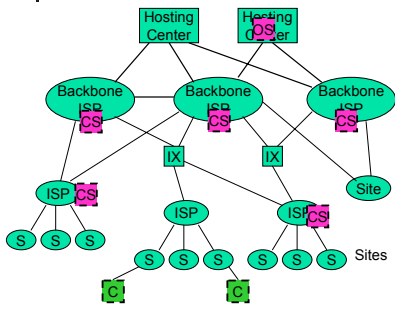
Content Routing Principle (a.k.a. Content Distribution Network)



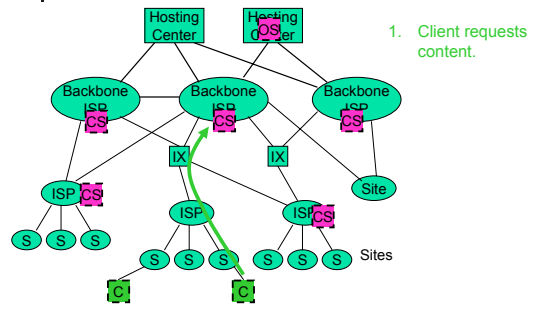
Content Routing Principle (a.k.a. Content Distribution Network)



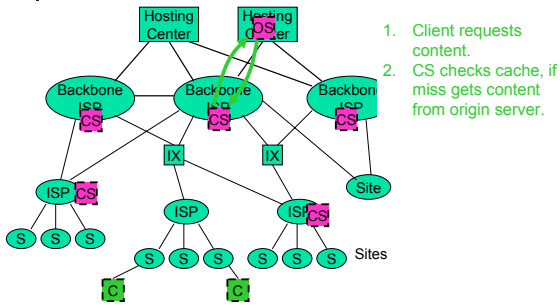
Two basic types of CDN: cached and pushed



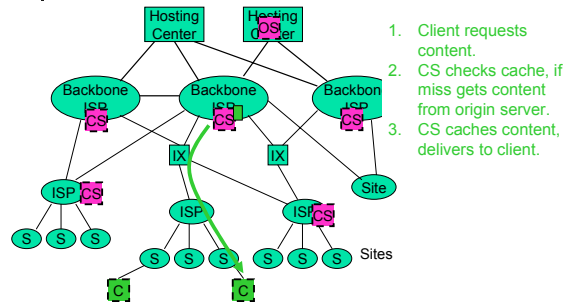
Cached CDN



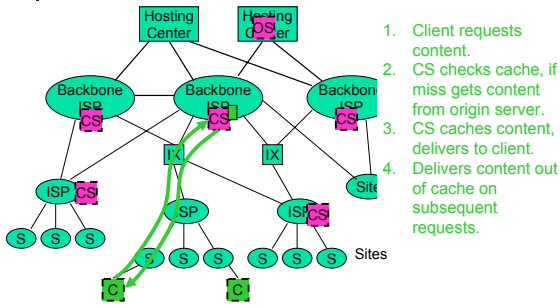
Cached CDN



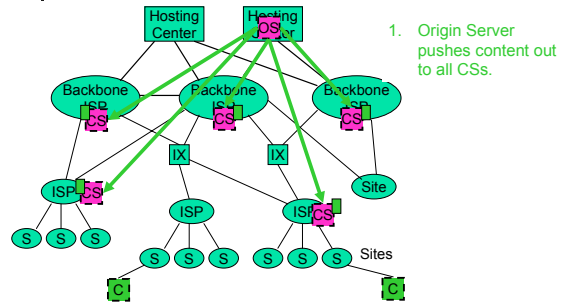
Cached CDN



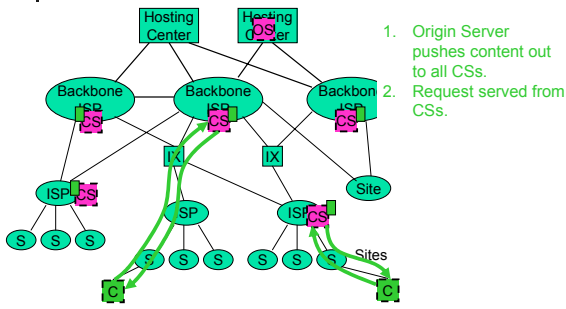
Cached CDN



Pushed CDN



Pushed CDN



CDN benefits

- Content served closer to client
 - Less latency, better performance
- Load spread over multiple distributed CSs
 - More robust (to ISP failure as well as other failures)
 - Handle flashes better (load spread over ISPs)
 - *But well-connected, replicated Hosting Centers can do this too*

CDN costs and limitations

- Cached CDNs can't deal with dynamic/personalized content
 - More and more content is dynamic
 - "Classic" CDNs limited to images
- Managing content distribution is non-trivial
 - Tension between content lifetimes and cache performance
 - Dynamic cache invalidation
 - Keeping pushed content synchronized and current

CDN example: Akamai

- Won huge market share of CDN business late 90's
- Cached approach
- Now offers full web hosting services in addition to caching services
 - Called edgesuite

Akamai caching services

ARL: Akamai Resource Locator

<http://a620.g.akamai.net/7/620/16/259fdbf4ed29de/www.cnn.com/i/22.gif>

Host Part Akamai Control Part Content URL

[/7/620/16/259fdbf4ed29de/](http://a620.g.akamai.net/7/620/16/259fdbf4ed29de/)

a620.g.akamai.net/

[/www.cnn.com/i/22.gif](http://www.cnn.com/i/22.gif)

Thanks to ratul@cs.washington.edu, "How Akamai Works"

ARL: Akamai Resource Locator

<http://a620.g.akamai.net/7/620/16/259fdbf4ed29de/www.cnn.com/i/22.gif>

Content Provider (CP) selects which content will be hosted by Akamai. Akamai provides a tool that transforms *this CP URL* into *this ARL*

a620.g.akamai.net/

[/www.cnn.com/i/22.gif](http://www.cnn.com/i/22.gif)

ARL: Akamai Resource Locator

<http://a620.g.akamai.net/7/620/16/259fdbf4ed29de/www.cnn.com/i/22.gif>

This in turn causes the client to access Akamai's content server instead of the origin server.

a620.g.akamai.net/

[/www.cnn.com/i/22.gif](http://www.cnn.com/i/22.gif)

ARL: Akamai Resource Locator

<http://a620.g.akamai.net/7/620/16/259fdbf4ed29de/www.cnn.com/i/22.gif>

If Akamai's content server doesn't have the content in its cache, it retrieves it using this URL.

a620.g.akamai.net/

[/www.cnn.com/i/22.gif](http://www.cnn.com/i/22.gif)

ARL Control Part

Type Code (different types will have different contents) → **7**
Customer Number (i.e. CNN, Yahoo...) → **620**
Content Checksum (May be used for identifying changed content. May also validate content???) → **16/259fdbf4ed29de/**

7/620/16/259fdbf4ed29de/

a620.g.akamai.net/

[/www.cnn.com/i/22.gif](http://www.cnn.com/i/22.gif)

<http://a620.g.akamai.net/7/620/16/259fdbf4ed29de/www.cnn.com/i/22.gif>

ARL Host Part

But why such a complex domain name????

7/620/16/259fdbf4ed29de/

a620.g.akamai.net/

[/www.cnn.com/i/22.gif](http://www.cnn.com/i/22.gif)

<http://a620.g.akamai.net/7/620/16/259fdbf4ed29de/www.cnn.com/i/22.gif>

ARL Host Part

.net gTLD

akamai.net

g.akamai.net

a620.g.akamai.net

Points to ~8 akamai.net DNS servers (random ordering, TTL order hours to days)

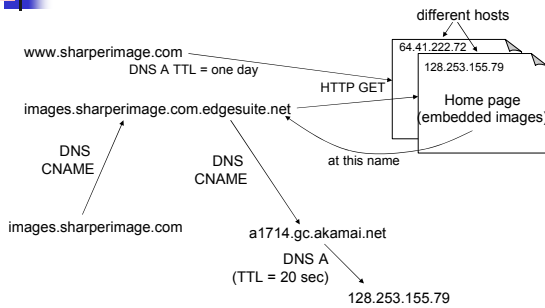
Attempts to select ~8 g.akamai.net DNS servers near client. (Using BGP? TTL order 30 min – 1 hour)

Makes a very fine-grained load-balancing decision among local content servers. TTL order 30 sec – 1 min.

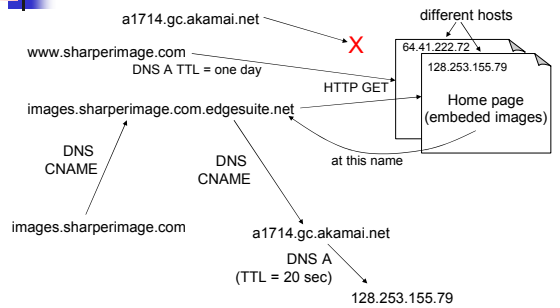
Akamai Edgesuite

- Appears that both DNS and web service handled by akamai
- Also may be that content may be pushed out to edge servers---no caching!

Sharper Image and Edgesuite



Sharper Image and Edgesuite



What may be happening...

- images.sharperimage.com.edgesuite.net returns same pages as www.sharperimage.com
 - But the shopping basket doesn't work!!
- Perhaps akamai cache blindly maps foo.bar.com.edgesuite.net into bar.com to retrieve web page
 - No more sophisticated akamaization
 - Easier to maintain origin web server??
 - Simpler akamai web caches??

Other content routing mechanisms

- Dynamic HTML URL re-writing
 - URLs in HTML pages re-written to point at nearby and non-overloaded content server
 - In theory, finer-grained proximity decision
 - Because know true client, not clients DNS resolver
 - In practice very hard to be fine-grained
 - Clearway and Fasttide did this
 - Could in theory put IP address in re-written URL, save a DNS lookup
 - But problem if user bookmarks page

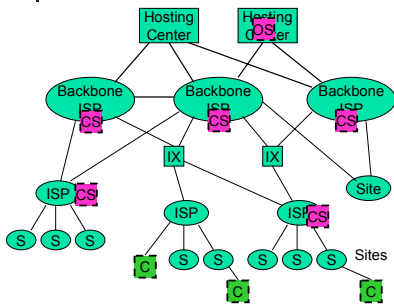
Other content routing mechanisms

- Dynamic .smil file modification
 - .smil used for multi-media applications (Synchronized Multimedia Integration Language)
 - Contains URLs pointing to media
 - Different tradeoffs from HTML URL re-writing
 - Proximity not as important
 - DNS lookup amortized over larger downloads
 - Also works for Real (.rm), Apple QuickTime (.qt), and Windows Media (.asf) descriptor files

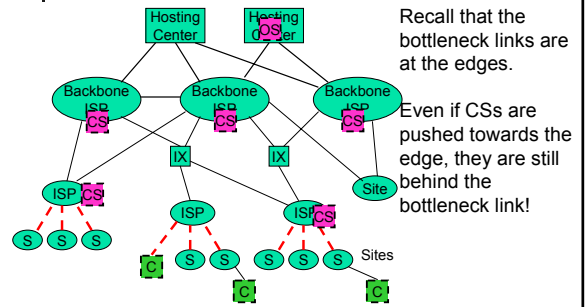
Other content routing mechanisms

- HTTP 302 Redirect
 - Directs client to another (closer, load balanced) server
 - For instance, redirect image requests to distributed server, but handle dynamic home page from origin server
 - See *draft-cain-known-request-routing-00.txt* for good description of these issues
 - But expired, so use Google to find archived copy

How well do CDNs work?



How well do CDNs work?



Reduced latency can improve TCP performance

- DNS round trip
- TCP handshake (2 round trips)
- Slow-start
 - ~8 round trips to fill DSL pipe
 - total 128K bytes
 - Compare to 56 Kbytes for cnn.com home page
 - Download finished before slow-start completes
- Total 11 round trips
- Coast-to-coast propagation delay is about 15 ms
 - Measured RTT last night was 50ms
 - No difference between west coast and Cornell!
- 30 ms improvement in RTT means 330 ms total improvement
 - Certainly noticeable

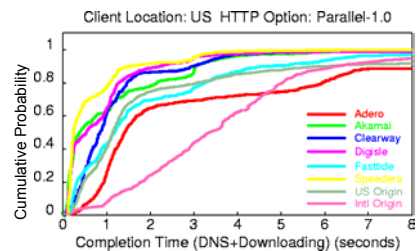
Lets look at a study

- Zhang, Krishnamurthy and Wills
 - AT&T Labs
- Traces taken in Sept. 2000 and Jan. 2001
- Compared CDNs with each other
- Compared CDNs against non-CDN

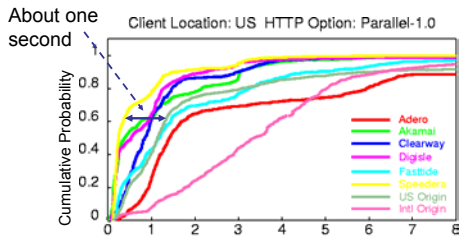
Methodology

- Selected a bunch of CDNs
 - Akamai, Speedera, Digital Island
 - Note, most of these gone now!
- Selected a number of non-CDN sites for which good performance could be expected
 - U.S. and international origin
 - U.S.: Amazon, Bloomberg, CNN, ESPN, MTV, NASA, Playboy, Sony, Yahoo
- Selected a set of images of comparable size for each CDN and non-CDN site
 - Compare apples to apples
- Downloaded images from 24 NIMI machines

Response Time Results (II) Including DNS Lookup Time



Response Time Results (II) Including DNS Lookup Time

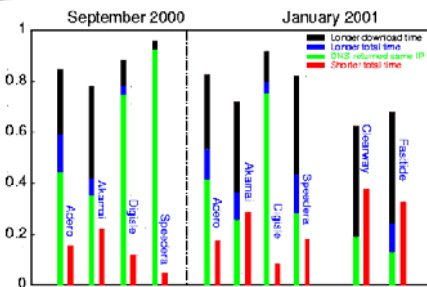


Author conclusion: CDNs generally provide much shorter download time.

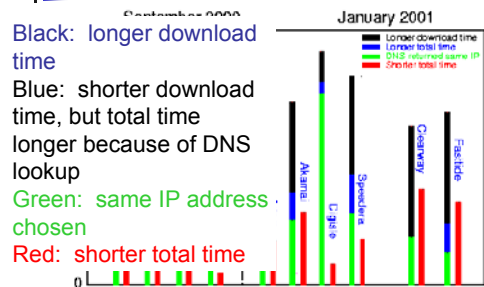
CDNs out-performed non-CDNs

- Why is this?
- Lets consider ability to pick good content servers...
- They compared time to download with a fixed IP address versus the IP address dynamically selected by the CDN for each download
 - Recall: short DNS TTLs

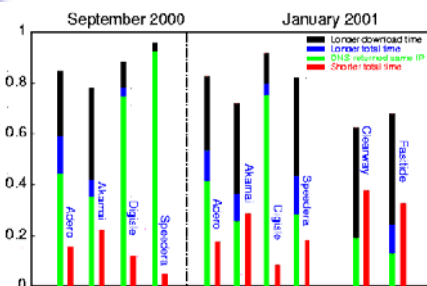
Effectiveness of DNS load balancing



Effectiveness of DNS load balancing



DNS load balancing not very effective



Other findings of study

- Each CDN performed best for at least one (NIMI) client
 - Why? Because of proximity?
- The best origin sites were better than the worst CDNs
- CDNs with more servers don't necessarily perform better
 - Note that they don't know load on servers...
- HTTP 1.1 improvements (parallel download, pipelined download) help a lot
 - Even more so for origin (non-CDN) cases
 - Note not all origin sites implement pipelining

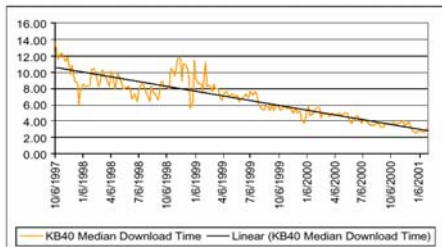
Ultimately a frustrating study

- Never actually says *why* CDNs perform better, only that they do
- For all we know, maybe it is because CDNs threw more money at the problem
 - More server capacity and bandwidth relative to load

Another study

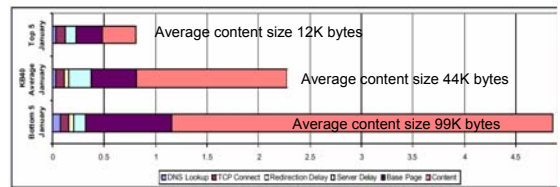
- Keynote Systems
 - "A Performance Analysis of 40 e-Business Web Sites"
- Doing measurements since 1997
 - (All from one location, near as I can tell)
- Latest measurement January 2001

Historical trend: Clear improvement

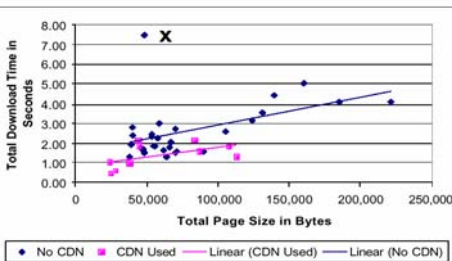


Performance breakdown

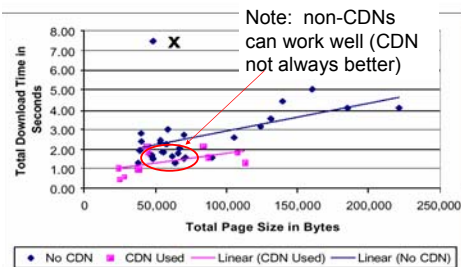
Basically says that smaller content leads to shorter download times (duh!)



Effect of CDN: Positive (but again, we don't know why)



Most web sites not using CDN (4-1)





To wrap things up

- As late as 2001, CDNs still used and still performing well
 - On a par or better than best non-CDN web sites
- CDN usage not a huge difference
- We don't know why CDNs perform well
 - But could very well simply be server capacity
- Knowledge of client location valuable more for customized advertising than for latency
 - Advertisements in right language



Layered Naming

- Recent proposal for discovery: naming requires four distinct layers:
 1. User-level descriptor (ULD) lookup (e.g. email address, search string, etc)
 2. Service-ID descriptor (SID): a sort of index naming the service and valid over the duration of this interaction
 3. SID to Endpoint-ID (EID) mapping: client-side protocol (e.g. HTTP) maps from SID to EID
 4. EID to IP address "routing": server side control over the decision of which "delegate" will handle the request
- Today we tend to blur the middle two layers and lack standards for this process, forcing developers to innovate
- See: "A Layered Naming Infrastructure for the Internet", Balakrishnan *et. al.*, ACM SIGCOMM Aug. 2004, Portland.



Research challenges

- Naming and discovery are examples of research challenges we're now facing in the Web Services arena
- There are many others, we'll see them as we get more technical in the coming lectures
- CS514 won't tackle naming but we will look hard at issues bearing on "trust"



Homework (not to hand in)

- Continue to read Parts I and II of the book
- Visit the semantic web repository at www.w3.org
- What does that community consider to be a potential "home run" for the semantic web?