



CS514: Intermediate Course in Computer Systems

Lecture 29: April 2, 2003

“Finding stuff in large systems”



CS514

- What is “stuff”?
- How large is large?



Different kinds of stuff

CS514

- IP address
- Server
- File
 - Web or file system
- Service
 - Printer, DNS, email sender,
- Event
 - Weather, stocks, alarm, etc.



Different ways to look for it

CS514

	Example Search Target	Example Query	Example System
Structured Key	IP address	foo.bar.com	DNS, LDAP
Key (flat)			Chord, Pastry
Keywords (Boolean)	Web document	Internet AND NAT	Google
Attribute-value	Web document, subscriber record	filetype:ppt, language:english	DNS, LDAP, Google
Prefix/suffix wildcard	Mpeg files	*.mpg	Napster, Gnutella
Regular expression		\$(Aa)lex.*	
Relational	Airline schedules	Flight search	orbitz.com



Over different types of systems

CS514

- Centralized-replicated versus distributed
- Hierarchical versus flat namespace
 - DNS versus DHT
- Infrastructure/peer-to-peer
- Publish/subscribe versus query/reply
- Peer-to-peer (even as infrastructure)



What we're going to look at

CS514

- DNS (again)
 - But this time as a general purpose distributed database
- Google
- Gnutella/Kazaa, publish/subscribe
 - Just a little for now
- Distributed Hash Tables (DHT)



DNS Revisited

CS514

- We have already seen how DNS is used:
 - To derive IP addresses from domain names (A records)
 - To determine the port number and server for applications (SRV records)
 - To load balance among servers



DNS as a general purpose distributed database?

CS514

- In theory, DNS can be used to lookup anything
- Required:
 - Agreed rules for how to create the “domain name” lookup “key”
 - Agreed rules for the record type to be looked up
 - Install the required DNS server hierarchy
- Example (not a real one), social security numbers:
 - 6.2.3.8.8.4.5.1.1.ssn



DNS services I didn't discuss

CS514

- Inverse lookup: learn domain name from IP address
- ENUM: learn services and addresses used to reach a device from a phone number
- DNSSEC: Distribute public keys and sign DNS records
 - Mainly useful for securing DNS itself



DNS services I didn't discuss

CS514

- DDDS (Dynamic Delegation Discovery System)
 - Broad use of NAPTR records to map lots of things into lots of things
 - URN to URL, for example
 - Includes ENUM
 - Not in widespread use (to my knowledge)

NAPTR RR (Naming Authority Pointer)

CS514

Order	Order in which records must be processed
Pref	Order in which records with same "order" may be processed
Flags	Application-specific flag says how to process record ('a': use to get A record)
Service	Services available ('sip', 'http', 'mailto', etc.)
RegEx	Regular expression used to transform lookup domain name into service name
Replace	Name with which to replace lookup name

ENUM Example ("E.164 Number")

CS514

- Phone number transformed into domain name like this:
 - +46-8-976-1234 becomes 4.3.2.1.6.7.9.8.6.4.e164.arpa.

```
$ORIGIN 4.3.2.1.6.7.9.8.6.4.e164.arpa.  
IN NAPTR 10 10 "u" "sip+E2U" "!.*$.!sip:paf@swip.net!" .  
IN NAPTR 102 10 "u" "mailto+E2U" "!.*$.!mailto:paf@swip.net!" .  
IN NAPTR 102 10 "u" "tel+E2U" "!.*$.!tel:+4689761234!" .
```

* E2U – Enum to URI



URN Example

CS514

urn:cid:199606121851.1@bar.example.com

Rule on thing between colons says to make a new name and lookup NAPTR record

cid.urn.arpa.

```
;;      order pref flags service      regexp      replacement
IN NAPTR 100 10 "" "" "!^urn:cid:.[^\\.]+\\.)(.*)$!\\2!i" .
```

This NAPTR says to replace the original URN according to the result of the regular expression, and do another NAPTR lookup with resulting name

example.com.

```
;;      order pref flags service      regexp      replacement
IN NAPTR 100 50 "a" "z3950+N2L+N2C" "" cidserver.example.com.
IN NAPTR 100 50 "a" "rcds+N2C" "" cidserver.example.com.
IN NAPTR 100 50 "s" "http+N2L+N2C+N2R" "" www.example.com.
```

This NAPTR shows various ways to obtain information about the URN ("a" = A RR, "s" = service)



Various types of discovery

CS514

N2L - Given a URN, return a URL

N2Ls - Given a URN, return a set of URLs

N2R - Given a URN, return an instance of the resource.

N2Rs - Given a URN, return multiple instances of the resource, typically encoded using multipart/alternative.

N2C - Given a URN, return a collection of meta- information on the named resource. The format of this response is the subject of another document.


N2Ns - Given a URN, return all URNs that are also identifiers for the resource.

L2R - Given a URL, return the resource.

L2Ns - Given a URL, return all the URNs that are identifiers for the resource.

L2Ls - Given a URL, return all the URLs for instances of of the same resource.


L2C - Given a URL, return a description of the resource.



Are the additional DNS functions useful?

CS514

- Inverse lookup of IP addr useful
 - Commonly used as a security/logging mechanism
 - Used for network measurements, etc.
- DNSSEC required for securing DNS
 - I don't know how useful it might be as a general PKI (Public Key Infrastructure)
 - Means ultimately we need to trust the root DNS servers???



Are the additional DNS functions useful?

CS514

- ENUM potentially very useful
 - Critical piece for gluing phone network to internet
- URN stuff
 - Not clear to me if this is real useful
 - Many ways to identify and find stuff in this world
 - LDAP distributed directory has richer search and attribute semantics



Quick diversion from CDN lecture (19)

CS514



Remember this? ARL: Akamai Resource Locator

CS514

<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/

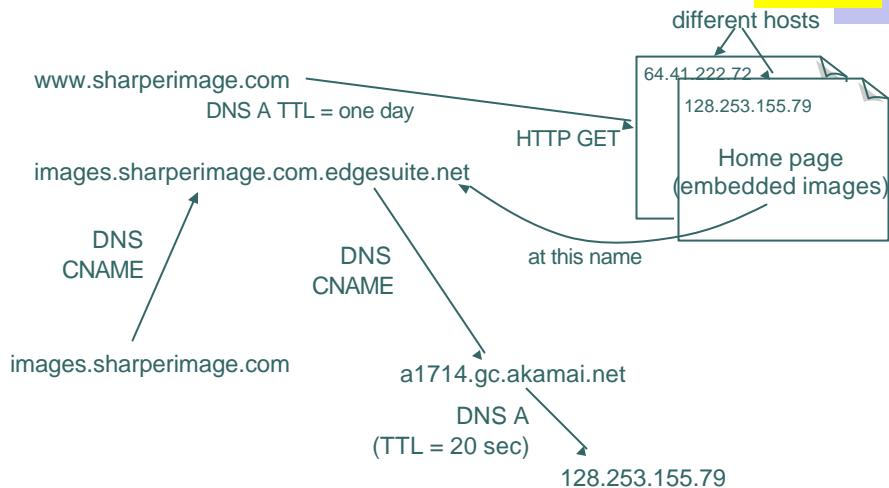
a620.g.akamai.net/

/www.cnn.com/i/22.gif

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

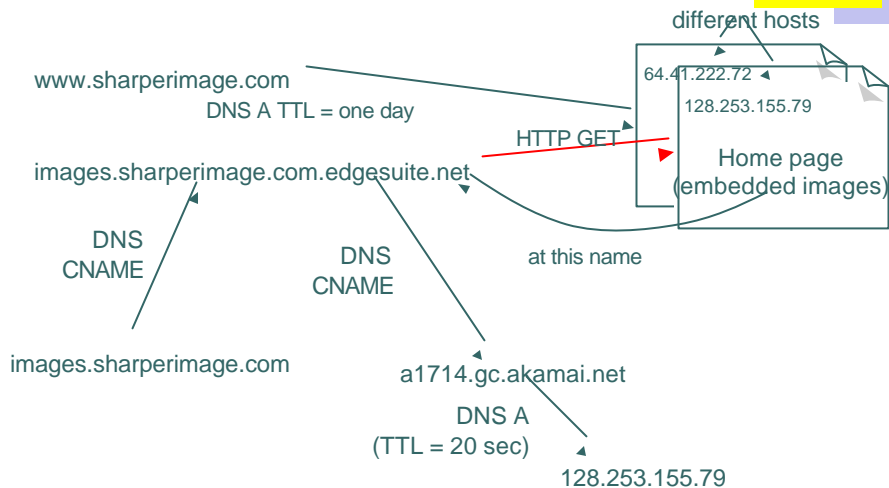
New Akamai CDN technique: Edgesuite

CS514



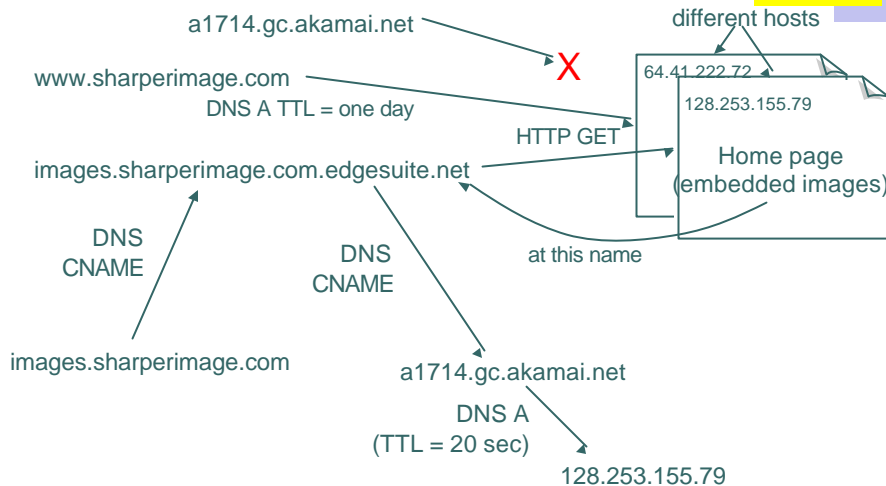
But the “images” name can retrieve the home page!

CS514



The “akamai.net” name cannot

CS514



What may be happening...

CS514

- **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??



Google

CS514

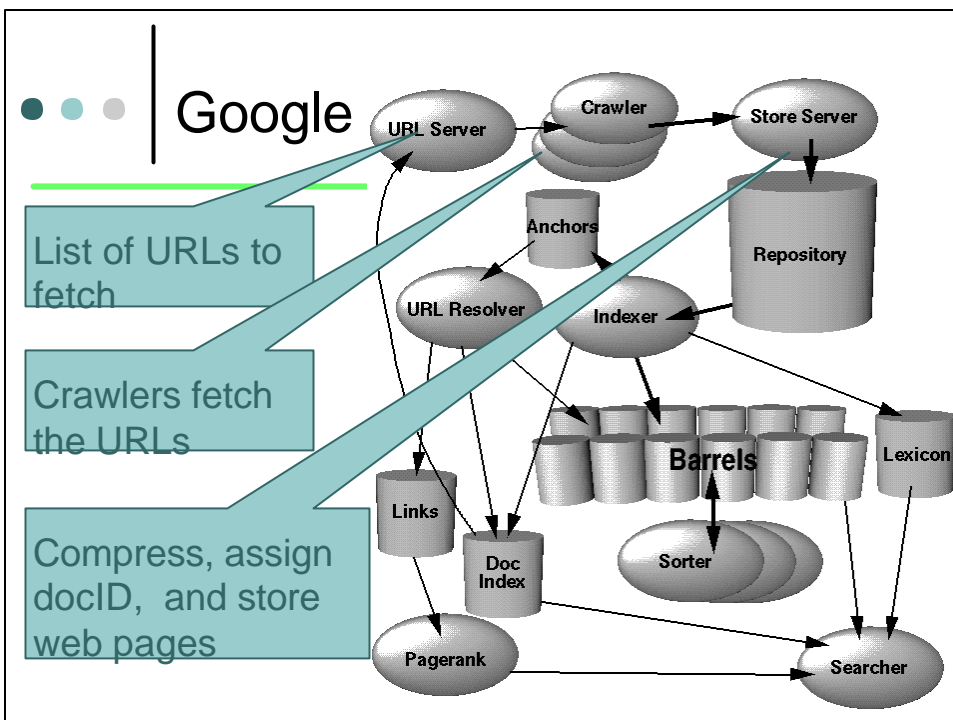
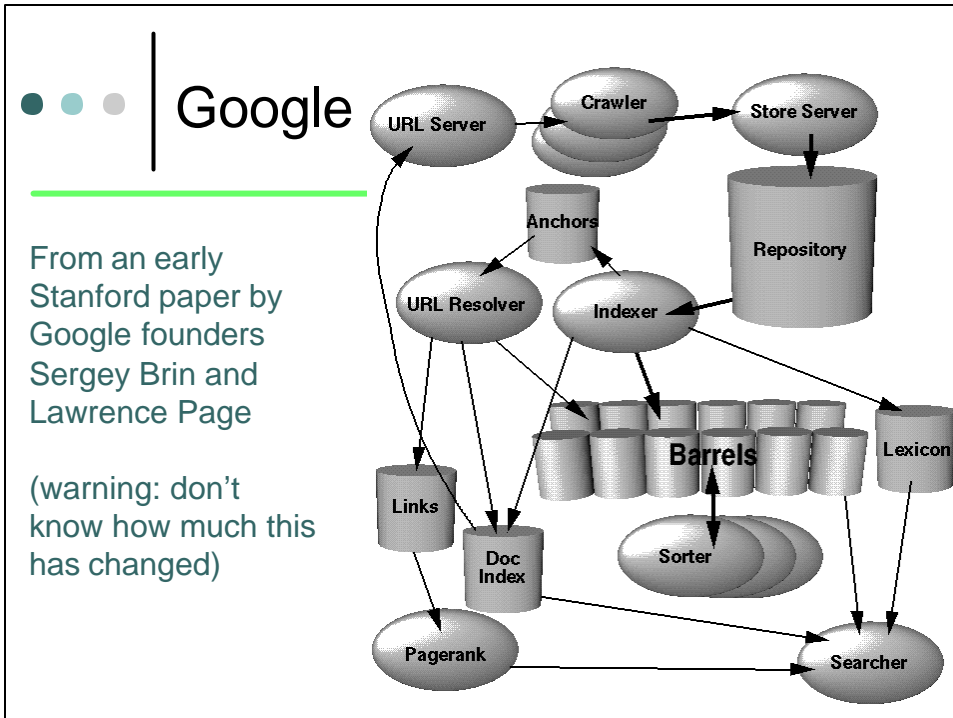
- Stanford project of Sergey Brin and Lawrence Page
- Key idea was to use hypertext links as a key component of page rank
 - Rather than frequency of word occurrence
- Also, use anchor text as well as document text
 - Anchor text often more descriptive than actual text
 - Can be used to search non-text files (images)

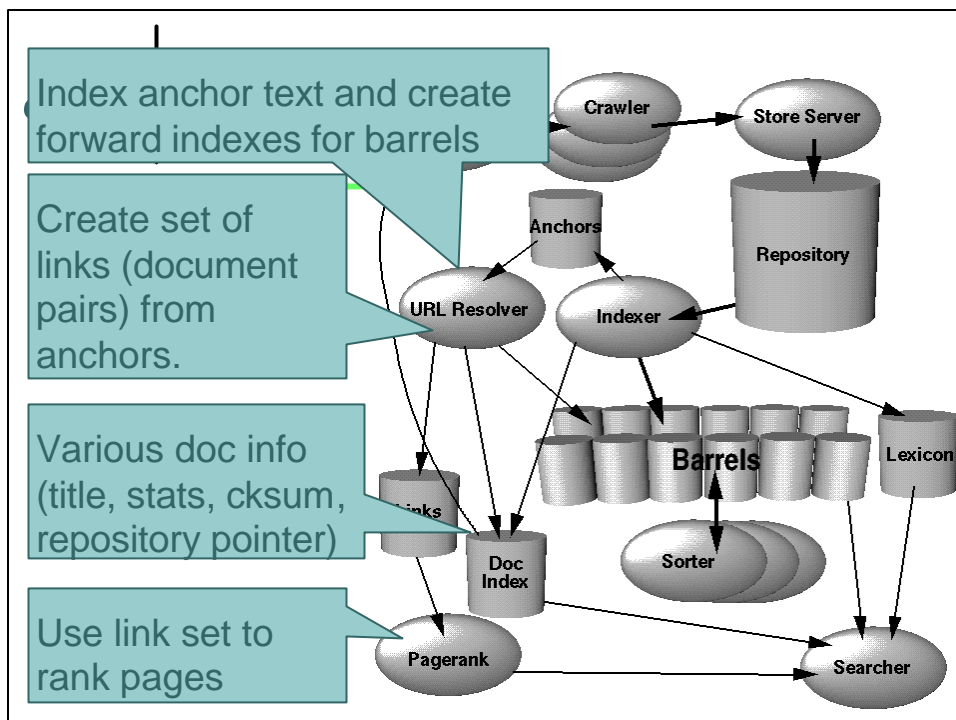
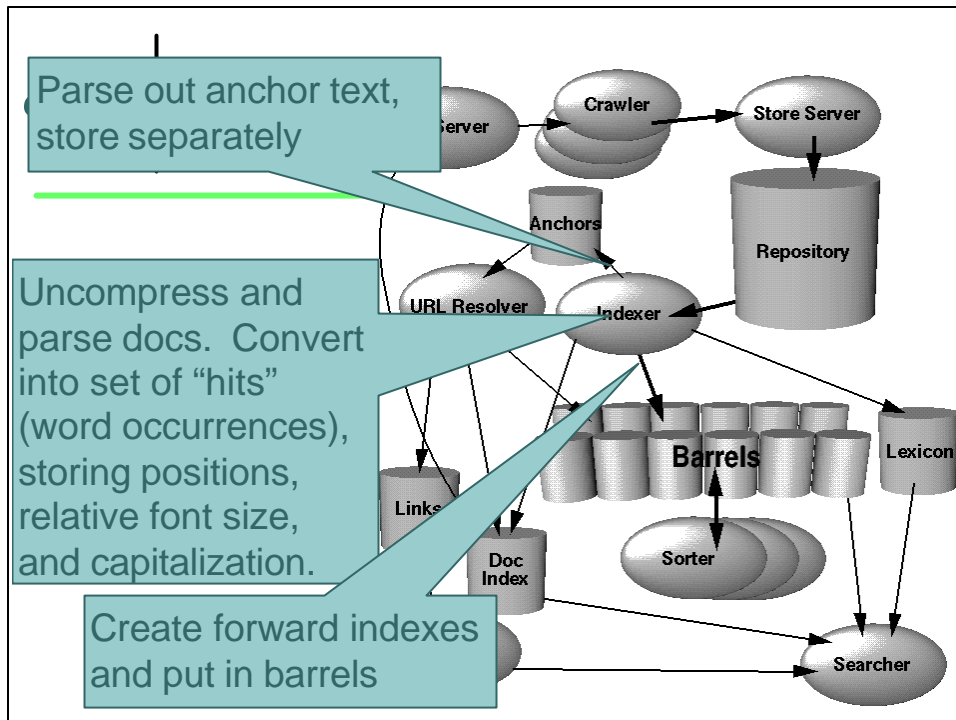


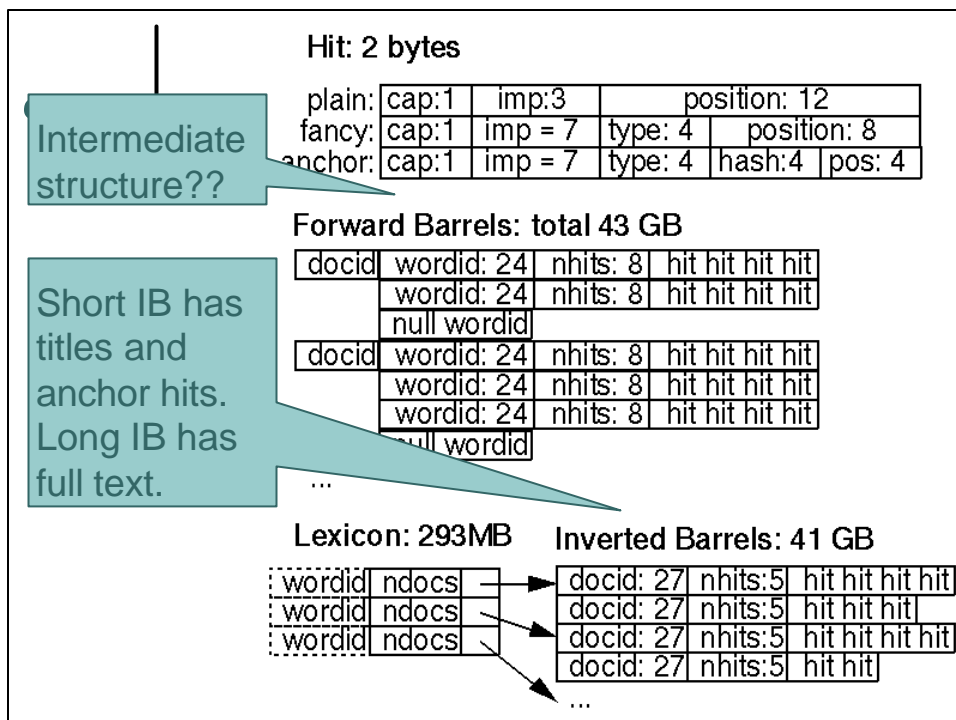
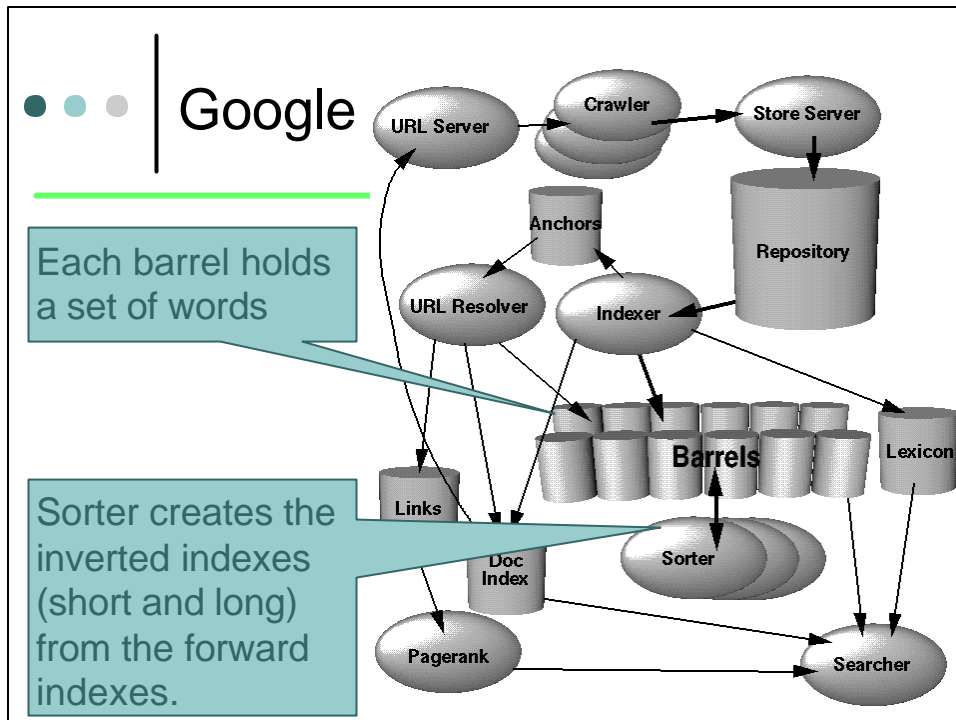
Google page rank

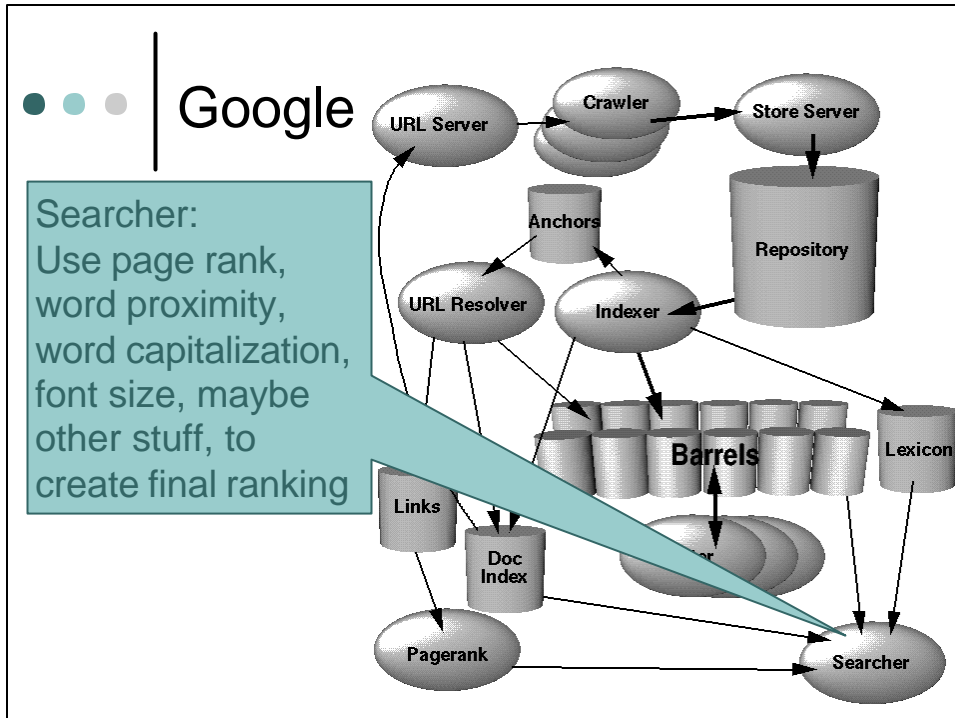
CS514

- $PR(A) = (1-d) + d (PR(T1)/C(T1) + \dots + PR(Tn)/C(Tn))$
 - $PR(A)$ = page rank of document A
 - $T1 \dots Tn$ are documents that point to A
 - $0 \leq d \leq 1$ is a damping factor (0.85)
- At time of Google, only $\frac{1}{4}$ of major search engines returned themselves then searched for them!
 - Remember Altavista searches?
 - +a +b -c -d -e -f -g ...









Some Google limitations

CS514

- Basic index is six to eight week old
 - News service obviously real time, but limited to hundreds or few thousands of news services
- Searched documents may be gone
 - Though the repository helps a lot
 - And can sometimes cause problems... something you want removed may still be searchable!



Distributed search

CS514

- Current distributed search is for peer-to-peer file sharing networks
 - Gnutella, Kazaa (FastTrack)
 - Napster was not a distributed search
 - The file download was peer-to-peer
- There is a history of pre-Lycos distributed search
 - Harvest, Centroid, WAIS, Ingrid, ...



Historic distributed search

CS514

- Various flavors
 - Typically “meta-search” engines that had knowledge of each topical search engine
 - Though Ingrid was more P2P
 - These were all blown away by the centralized approach!
- ~~Music theft~~ file sharing makes distributed search valid again



Original Gnutella approach

CS514

- Nodes form a huge mesh network
 - Richly connected---10's of neighbors
- Each node indexes its own files
- Keyword searches are broadcast through the network
 - Truncated by hop count
 - Identity of searcher is hidden---replies follow the reverse path
- Problem was that thinly connected nodes would be overwhelmed with queries



Modified Gnutella approach

CS514

- Also approach of FastTrack
- Well-connected and stable nodes become "super nodes"
 - Other nodes upload file catalogue to super nodes
 - Super nodes index the nodes' catalogues
 - Small 10's of nodes per super node
- Searches broadcast only among super nodes
 - This works a whole lot better



Publish/subscribe systems

CS514

- Publish/subscribe systems are a kind of search
- Clients “subscribe” to certain content (by topic, for instance) from servers
 - Kind of a standing search request
- When content matching subscription arrives, server multicasts it to all subscribed clients
 - Reverse of P2P search--search is unicast, content is multicast



Benefits of Publish/subscribe

CS514

- *Event driven*: content is pushed out to client as soon as it becomes available
- *Content addressing*: clients (subscriber) and servers (publishers) do not need to be explicitly aware of each other
 - Handled by middleware infrastructure
- Can have sophisticated subscription rules
 - Tell me when Akamai stock exceeds \$2.00
- TIBCO is a successful commercial example
 - Pointcast, Marimba Castanet, were not...

All these search approaches have scaling issues

CS514

- Google requires 30K* PCs and 6-8 weeks to make an index
- Gnutella broadcasts searches
- Publish/subscribe multicasts results
- Distributed Hash Table (DHT) has better scaling properties
 - Though limited search semantics
 - Are DHTs the future, or a dead-end?

* Not 100% sure about this number

Scalability summary

CS514

	Index	Search Query	Search Reply
Google, Napster	Centralized	Centralized	Unicast
Publish/Subscribe	Centralized	Centralized	Multicast
Gnutella, FastTrack	Semi-distributed	Truncated flood	Unicast
DHT (Chord, Pastry, RAN...)	Distributed	Unicast	Unicast

Next lecture.....all about DHTs