

Web Crawling

David Kauchak

cs160

Fall 2009

adapted from:

<http://www.stanford.edu/class/cs276/handouts/lecture14-Crawling.ppt>

Administrative

- Midterm
- Collaboration on homeworks

Possible topics with equations for midterm

- Note this is not necessarily a complete list, but are ones that are relevant
 - Heaps and Zipf's law (you don't need to know the exact equations, but you should understand the relationships they describe)
 - Posting file compression
 - Tf-Idf
 - log and boolean term normalization
 - idf term weighting
 - cosign length normalization
 - Evaluation
 - Precision/recall
 - F1
 - MAP

MAP

System 1



System 2

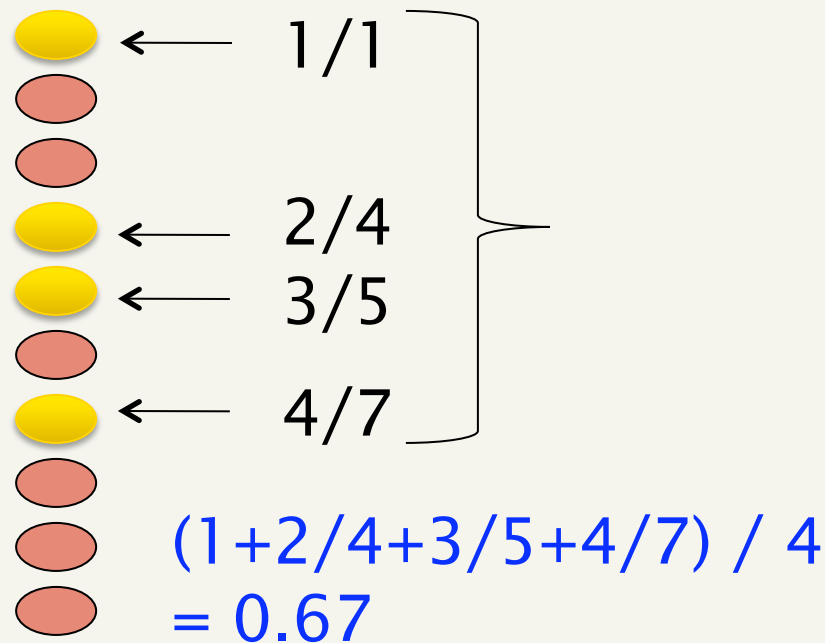


 = relevant

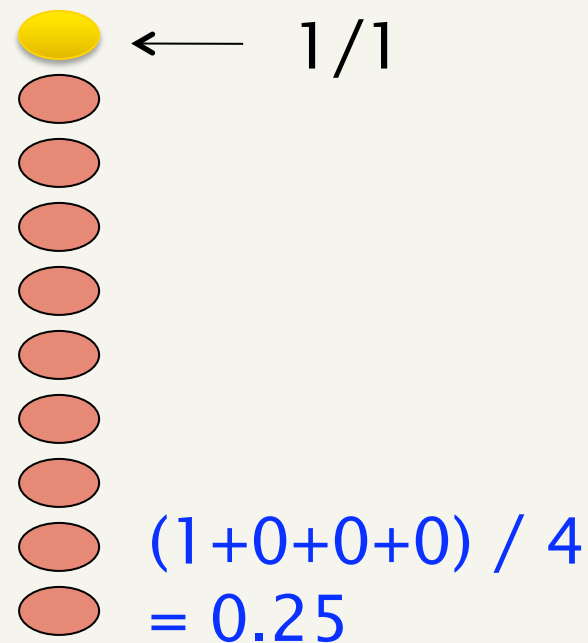
Which is better?

MAP

System 1



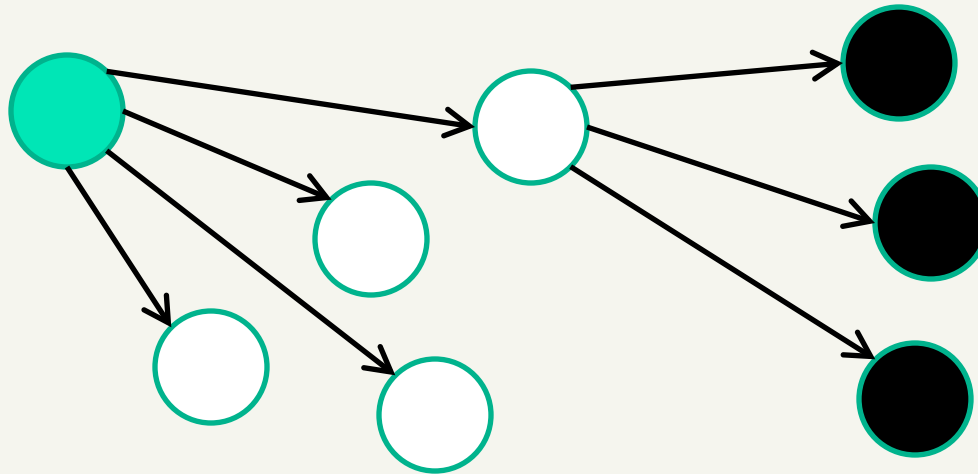
System 2



- MAP is calculate for ALL relevant documents
- If a relevant document is not in the result set it is given a precision of 0

Duplicate detection

Basic crawler



- Begin with “seed” URLs in the queue
 - Get a URL from the queue
 - Fetch the page
 - Parse the page and extract URLs it points to
 - Place the extracted URLs on a queue

Web crawlers

- Crawling is similar at a high-level to traditional graph search
- How is it different?
 - Latency/bandwidth issues (we have to actually fetch each node)
 - Malicious pages
 - Spam
 - Spider traps
 - Politeness concerns – don't hit the same server too frequently
 - Duplicate pages
 - Web is not fully connected

Fetching web pages

- Given a URL, we first need to fetch the actual web page

`www.cs.pomona.edu/classes/cs160/index.html`

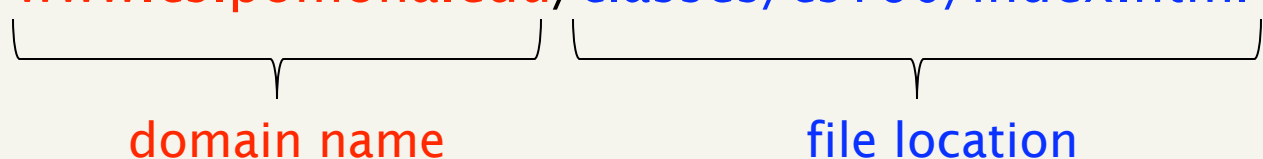
domain name file location

- What steps need to happen?
 - Find the web server
 - similar to “call Dave Kauchak” – we need to know how to contact the web server
 - Computers on the web are specified by IP addresses
 - DNS (domain name service) offers a directory lookup from domain to IP address
 - DNS lookup is distributed and can be slow

Fetching web pages

- Given a URL, we first need to fetch the actual web page

`www.cs.pomona.edu/classes/cs160/index.html`



The diagram shows the URL `www.cs.pomona.edu/classes/cs160/index.html` with two brackets underneath. The first bracket is under `www.cs.pomona.edu` and is labeled "domain name" in red text. The second bracket is under `/classes/cs160/index.html` and is labeled "file location" in blue text.

- What steps need to happen?
 - Contact the web server and download the file
 - A web server is just a computer connected to the internet listening on port 80 (or sometimes 8080) for HTTP requests
 - Connect to the server and request the particular page

```
GET /index.html HTTP/1.1
Host: www.yahoo.com
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1)
```

Parse web page and extract URLs

- Parsing the web page
 - Deal with many of the issues we talked about previously, like encoding, etc.
 - Full HTML parsing can be a pain since web browsers are fault tolerant
- Extract URLs
 - Handle “relative” URLs, e.g. “home.html”
 - Remove duplicate URLs
- Besides extracting the URLs/links for crawling purposes, is there anything else we need them for?

Connectivity Server

[CS1: Bhar98b, CS2 & 3: Rand01]

- Support for fast queries on the web graph
 - Which URLs point to a given URL?
 - Which URLs does a given URL point to?

Stores the mappings in memory

- Applications
 - Crawl control
 - Web graph analysis
 - Connectivity, crawl optimization
 - Link analysis

Polite web crawlers

- A web crawler has few constraints on which pages it can visit, but it *must* adhere to the to politeness policies
 - Never hit the same web server (generally IP) more frequently than once a second
 - Only one connection open to a giver web server at a time
 - robots.txt

Robots.txt

- Protocol for giving spiders (“robots”) limited access to a website, originally from 1994
 - www.robotstxt.org/wc/norobots.html
- Website announces its request on what can(not) be crawled
 - For a domain, create a file `Domain/robots.txt`
 - This file specifies access restrictions

Robots.txt example

- No robot should visit any URL starting with "/yoursite/temp/", except the robot called "searchengine":

```
User-agent: *
```

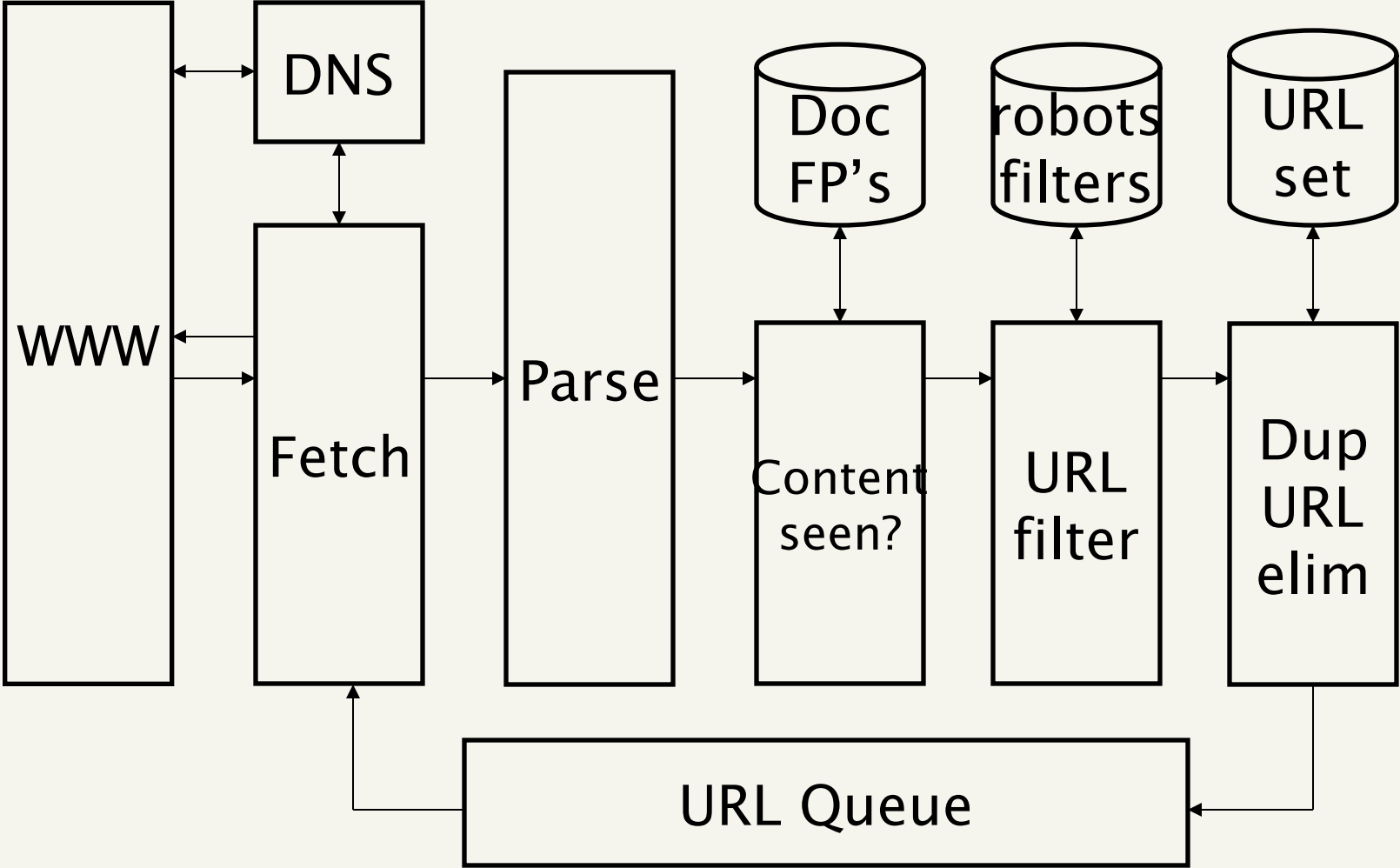
```
Disallow: /yoursite/temp/
```

```
User-agent: searchengine
```

```
Disallow:
```

Even Google has a robots.txt 😊

Basic crawl architecture



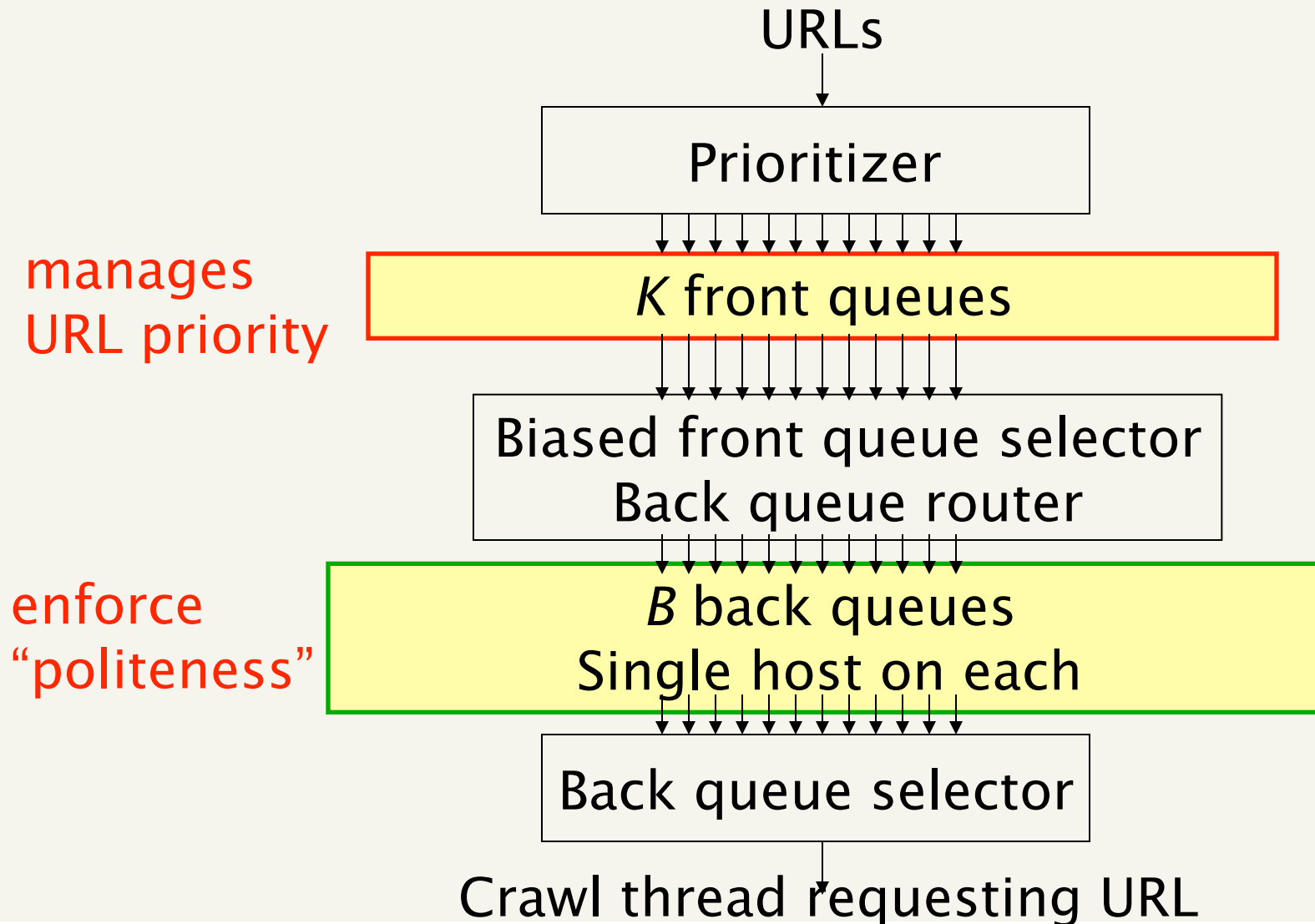
Web crawler scale

- The biggest challenges for web crawlers is dealing with the size of the web
- How many web pages per second would we need to download to obtain 1 billion web pages in a month?
 - $30 \text{ d} * 24 \text{ h} * 60 \text{ m} * 60 \text{ s} = 2,592,000$
 - $1,000,000,000 / 2,592,000 = 385 \text{ pages/sec}$
- Have to be multithreaded/multi-computer
- Logistics become trickier

Web crawler scale issues

- What complications does this create?
 - Can't hit same web server
 - Often pages point to pages on the same server
 - Can't just wait... need to keep servers busy
 - Cache robots.txt
 - Distributed computing
 - Duplicate URL detection
 - Keeping track of who's doing what
 - Cache DNS lookup since it's slow
- The URL queue becomes an important data structure to try and prioritize things appropriately
 - Can't just do a priority queue!

URL frontier: Mercator scheme



Priority

- Prioritizer assigns to URL an integer priority between 1 and K
 - Appends URL to corresponding queue
- Heuristics for assigning priority?
 - Refresh rate sampled from previous crawls
 - Importance
 - Application-specific (e.g., “crawl news sites more often”)

Resources

- IIR Chapter 20
- [Mercator: A scalable, extensible web crawler \(Heydon et al. 1999\)](#)
- [A standard for robot exclusion](#)
- [The WebGraph framework I: Compression techniques \(Boldi et al. 2004\)](#)

Cool search engines

- What do you think will be the most important feature(s) in next-generation search algorithms?
- Is it better to have a broad, general search engine or one that is tailored to your needs?
- What new markets can be explored using a search engine?
- Some of these search engines are niche-specific sites and others are search aggregators. Is web search diverging in the direction of many topic-specific sites or converging to one large find-everything site? Is one of these better? What should we be aiming for?
- What are the benefits of live updating searches (Collecta) vs. previously indexed content (Google)?
- How do you think Collecta is able to find results so quickly?
- The article mentions “inserting a human element into search.” What exactly does this mean? How can a web search include human power? Is that useful?