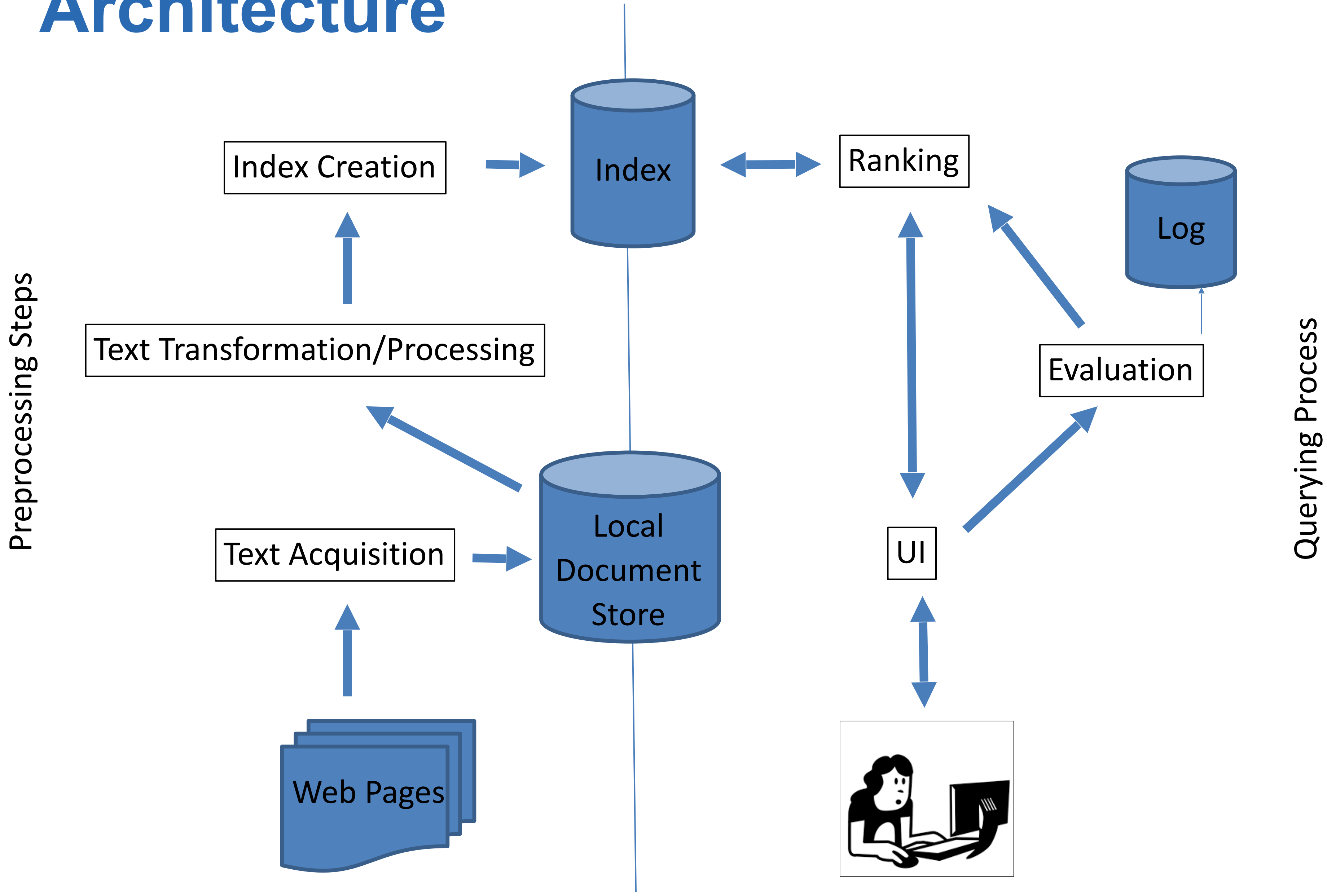
Informatics 141
Computer Science 121

Information Retrieval
Lecture 3.1

Professor Iftekhar Ahmed
Department of Informatics
https://www.ics.uci.edu/~iftekha/

Architecture



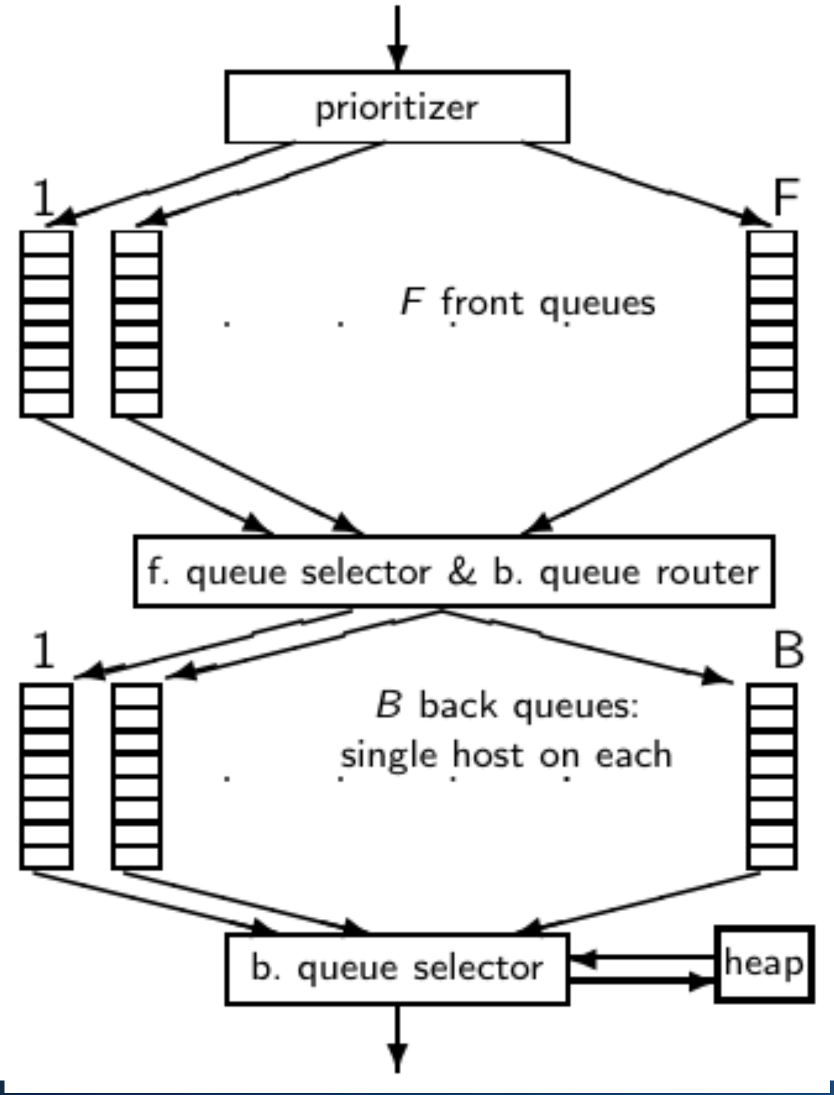
Pseudo Code

```
procedure CrawlerThread(frontier)
    while not frontier.done() do
        website ← frontier.nextSite()
        url ← website.nextURL()
        if website.permitsCrawl(url) then
            text ← retrieveURL(url)
            storeDocument(url, text)
            for each url in parse(text) do
                frontier.addURL(url)
            end for
        end if
        frontier.releaseSite(website)
    end while
end procedure
```

“Basic algorithm” is...

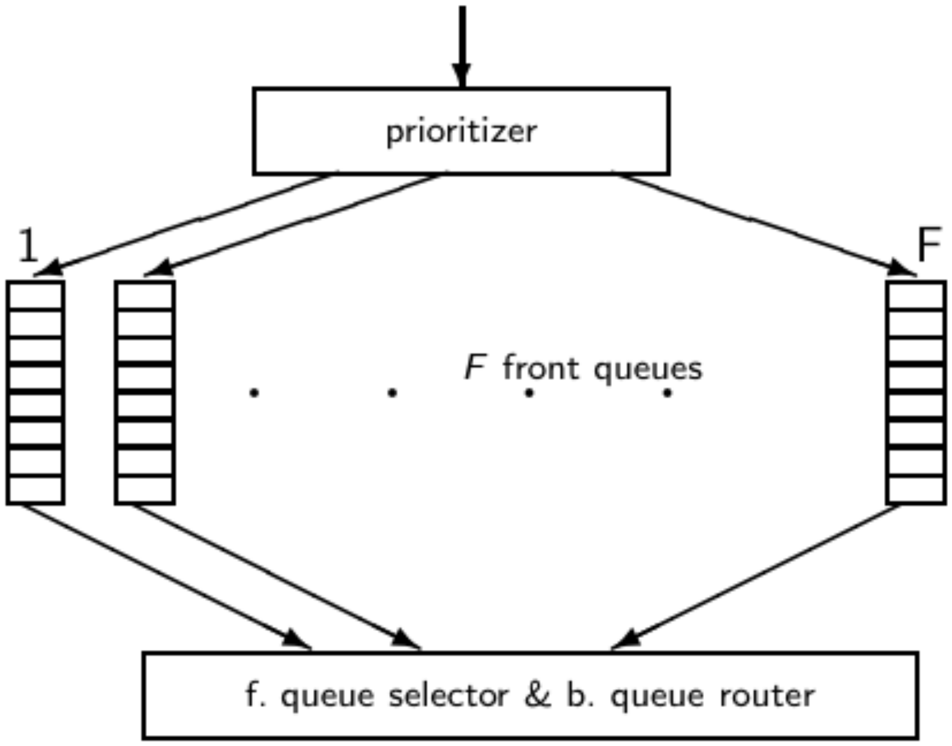
- Theoretically correct
- Seriously lacking to use in practice
  1. Will upset web admins (impolite)
     - It’s abusing the web servers
  2. Very slow
     - 1 page at a time
  3. Will get caught in traps and infinite sequences
  4. Will fetch duplicates without noticing
  5. Will bring in data noise
  6. Will miss content due to client-side scripting

Mercator URL frontier



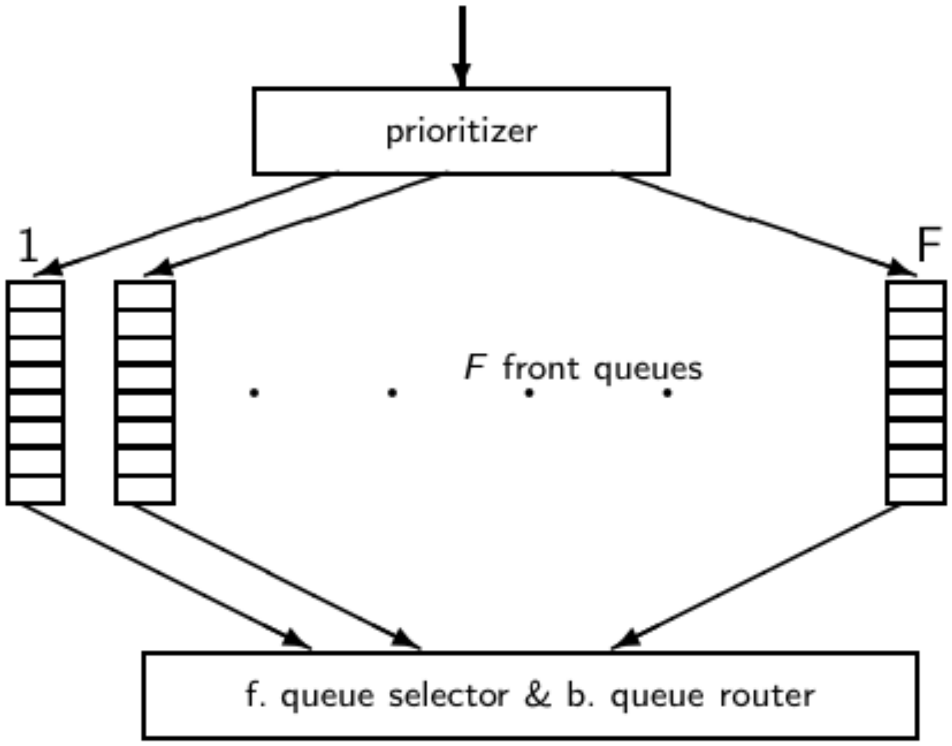
- URLs flow in from the top into the frontier.
- Front queues manage prioritization.
- Back queues enforce politeness.

Mercator URL frontier



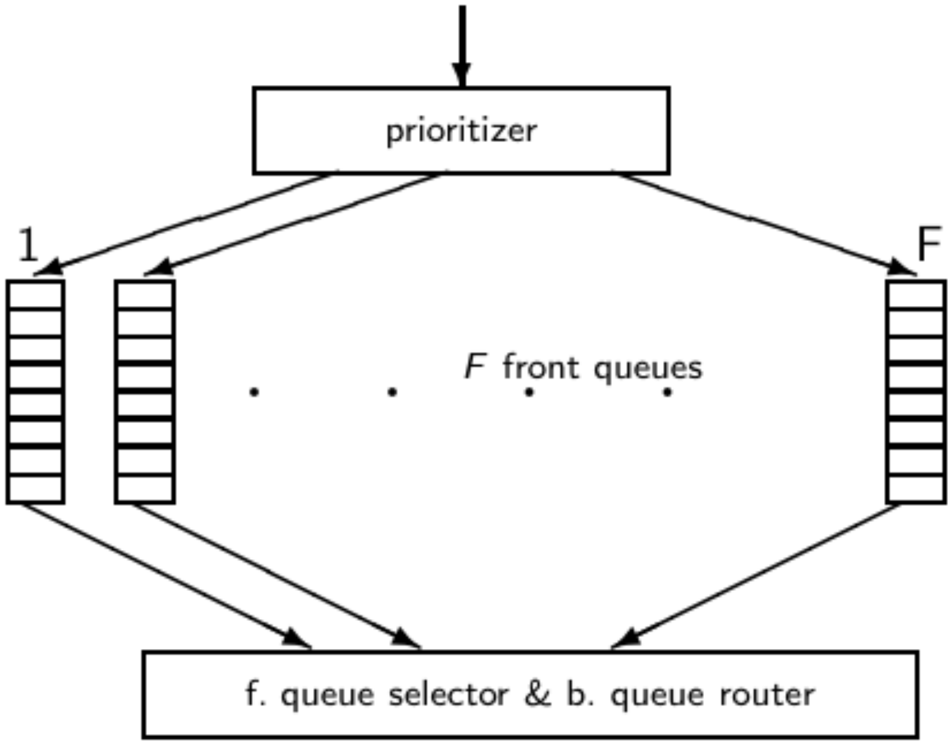
- Prioritizer assigns to URL an integer priority between 1 and *F*.

Mercator URL frontier



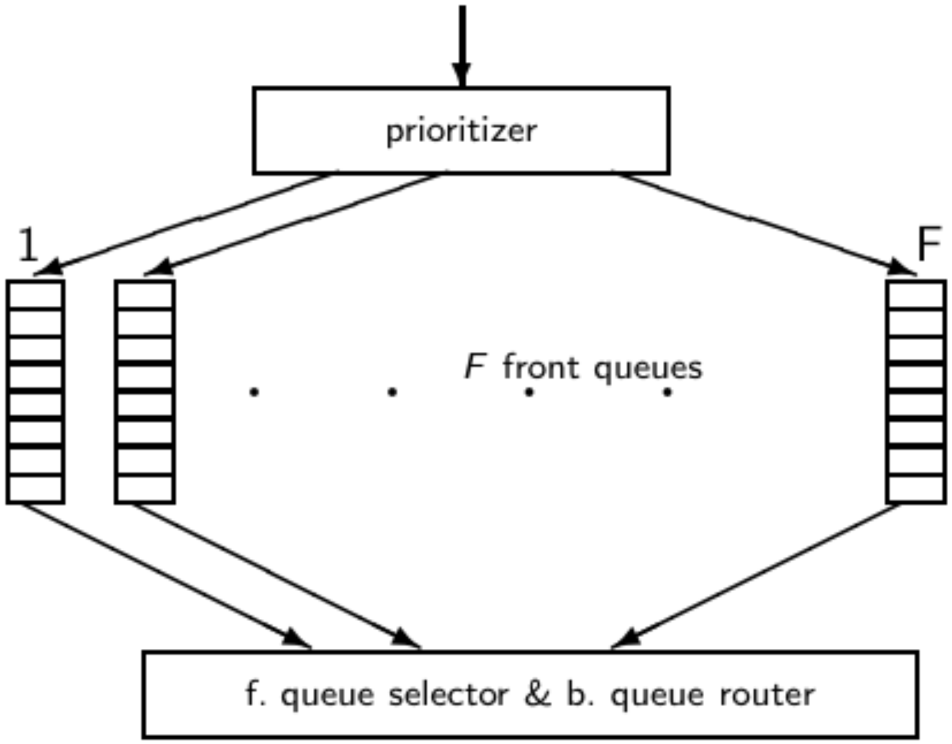
- Prioritizer assigns to URL an integer priority between 1 and *F*.
- Then appends URL to corresponding queue

Mercator URL frontier



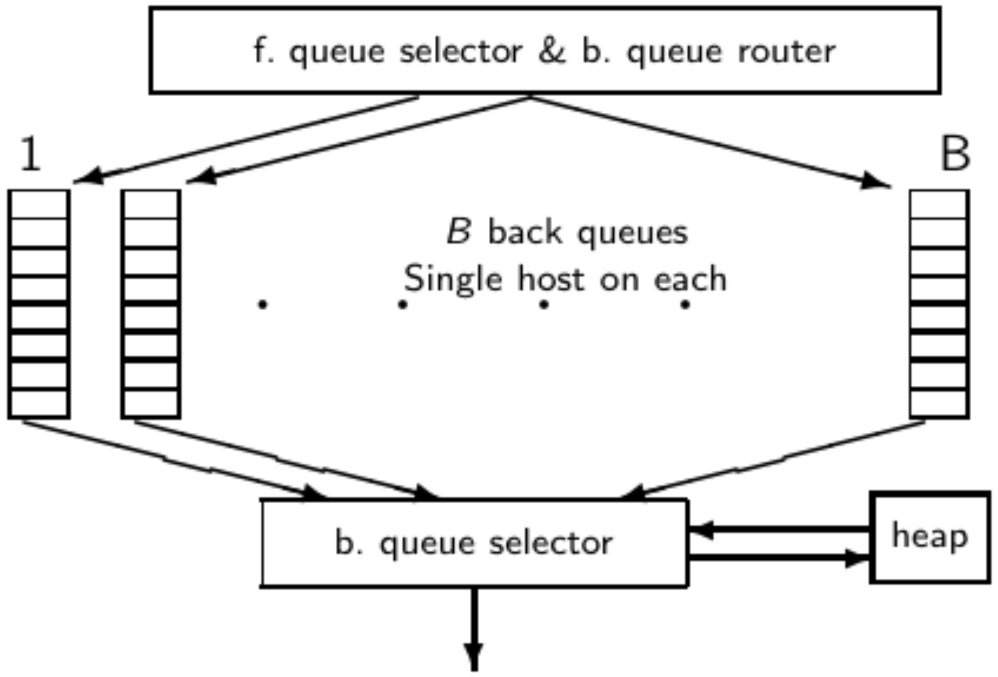
- Prioritizer assigns to URL an integer priority between 1 and *F*.
- Then appends URL to corresponding queue
- Heuristics for assigning priority: refresh rate, PageRank etc

Mercator URL frontier



- Selection from front queues is initiated by back queues
- Pick a front queue from which to select next URL: Round robin, randomly, or more sophisticated variant
- But with a bias in favor of high-priority front queues

Mercator URL frontier



Politeness

- Avoid hitting any site too often
  - Sites are for people, not for bots
- Ignore politeness ➔ Denial of service (DOS) attack
- Be polite ➔ Use artificial delays

Explicit and implicit politeness

- Explicit politeness: specifications from webmasters on what portions of site can be crawled
  - robots.txt
- Implicit politeness: even with no specification, avoid hitting any site too often
- Even if we restrict only one thread to fetchfrom a host, can hit it repeatedly
- Common heuristic: insert time gap betweensuccessive requests to a host that is >> timefor most recent fetch from that host

What any crawler *should* do

- Fetch pages of "higher quality" first
- Continuous operation: Continue fetching fresh copies of a previously fetched page
- Extensible: Adapt to new data formats, protocols

2. Performance (I)

- Back of the envelope calculation:
  - 1 page fetch = 500ms
  - How much time to crawl 1 million pages?
    - (it’s worse than that... Unresponsive servers)
- Most of the time, the crawler thread is waiting for the network data
- Solution: multi-threaded or distributed crawling
  - Politeness is harder to control, but it is possible (e.g. different servers)

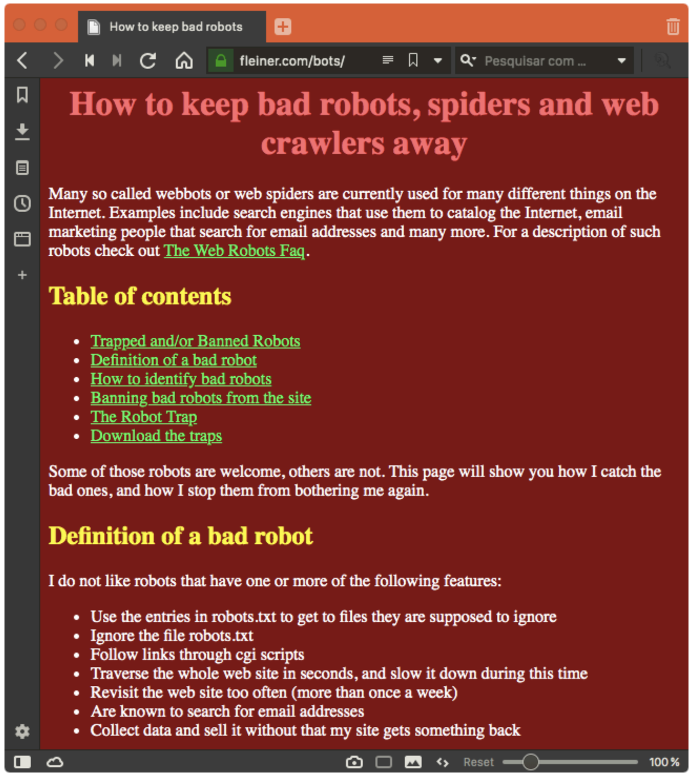
2. Performance (II)

- Domain Name lookups
  - Given a domain name, retrieve its IP address
    - www.ics.uci.edu -> 128.195.1.83
- Distributed set of servers
  - Latency can be high (2 secs is not unusual)
- Common implementations are blocking
  - One request at a time
  - Result is cached
- Back of the envelope calculation:
  - 1 DNS lookup ➔ 800ms
  - How much time to lookup the entire Web?

3. Crawler traps

- May trap the crawler on the site forever
  - Web server responds with ever changing URLs and content
    - Dynamic pages
  - Can be intentional or unintentional
    - E.g. the ICS calendar is a crawler trap
    - Some webadmins can create traps
      to penalize impolite crawlers

- See http://www.fleiner.com/bots/
    - E.g. very large documents, disallowed
      in robots.txt, created to consume
      crawler resources or event to break poorly designed parsers of
      crawlers that ignore robots.txt



How to keep bad robots, spiders and web crawlers away

Many so called webbots or web spiders are currently used for many different things on the Internet. Examples include search engines that use them to catalog the Internet, email marketing people that search for email addresses and many more. For a description of such robots check out The Web Robots Faq.

Table of contents

- Trapped and/or Banned Robots
- Definition of a bad robot
- How to identify bad robots
- Banning bad robots from the site
- The Robot Trap
- Download the traps

Some of those robots are welcome, others are not. This page will show you how I catch the bad ones, and how I stop them from bothering me again.

Definition of a bad robot

I do not like robots that have one or more of the following features:

- Use the entries in robots.txt to get to files they are supposed to ignore
- Ignore the file robots.txt
- Follow links through cgi scripts
- Traverse the whole web site in seconds, and slow it down during this time
- Revisit the web site too often (more than once a week)
- Are known to search for email addresses
- Collect data and sell it without that my site gets something back

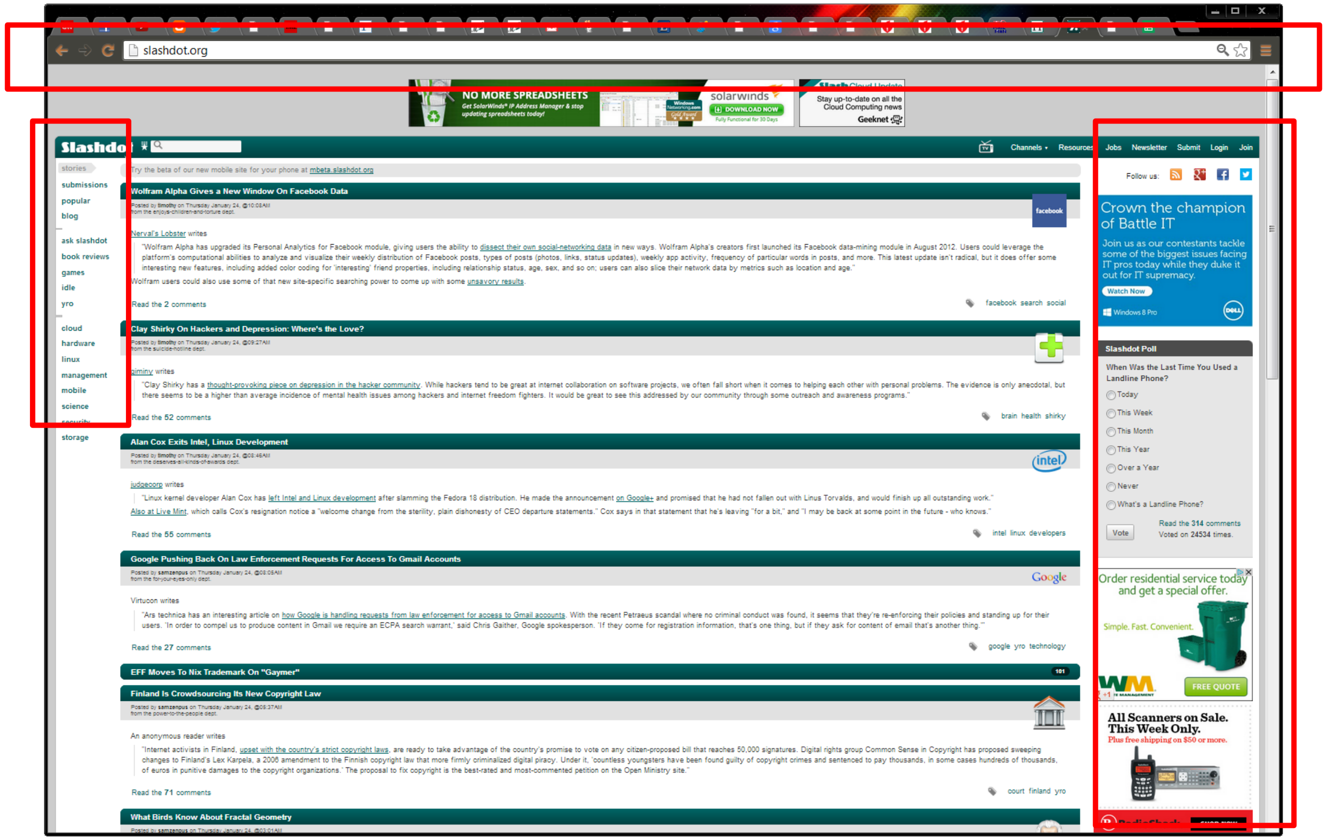
4. Duplicate Detection

- **Exact and near duplication are widespread**
  - Copies, mirror sites, versions, spam, plagiarism...
  - Studies: 30% of Web pages are [near-]duplicates of the other 70%
  - Little or no value (noise to the search engine and the user; you can show only one to the user and perhaps a “show similar” link)

- Detection
  - Detection of exact duplication is easy, but *exact duplication is rare*
    - Hashes, checksums
  - Detection of *near-duplicates* is harder
    - Page fingerprints

5. Data Noise

- **Web pages have content not directly related to the page**
  - **Ads, templates, etc**
  - **Noise negatively impacts information retrieval**



slashdot.org

NO MORE SPREADSHEETS
Get SolarWinds® IP Address Manager & stop updating spreadsheets today!
solarwinds DOWNLOAD NOW Fully Functional for 30 Days

Stay up-to-date on all the Cloud Computing news
Geeknet

Slashdot

Channels ▾ Resources Jobs Newsletter Submit Login Join

stories
submissions
popular
blog
ask slashdot
book reviews
games
idle
yro
cloud
hardware
linux
management
mobile
science
security
storage

Try the beta of our new mobile site for your phone at mbeta.slashdot.org

Wolfram Alpha Gives a New Window On Facebook Data
Posted by timothy on Thursday January 24, @10:08AM
from the enjoys-children-and-torture dept.

Nerval's Lobster writes
"Wolfram Alpha has upgraded its Personal Analytics for Facebook module, giving users the ability to dissect their own social-networking data in new ways. Wolfram Alpha's creators first launched its Facebook data-mining module in August 2012. Users could leverage the platform's computational abilities to analyze and visualize their weekly distribution of Facebook posts, types of posts (photos, links, status updates), weekly app activity, frequency of particular words in posts, and more. This latest update isn't radical, but it does offer some interesting new features, including added color coding for 'interesting' friend properties, including relationship status, age, sex, and so on; users can also slice their network data by metrics such as location and age."
Wolfram users could also use some of that new site-specific searching power to come up with some unsavory results.

Read the 2 comments — facebook search social

Clay Shirky On Hackers and Depression: Where's the Love?
Posted by timothy on Thursday January 24, @09:27AM
from the suicide-hotline dept.

giminy writes
"Clay Shirky has a thought-provoking piece on depression in the hacker community. While hackers tend to be great at internet collaboration on software projects, we often fall short when it comes to helping each other with personal problems. The evidence is only anecdotal, but there seems to be a higher than average incidence of mental health issues among hackers and internet freedom fighters. It would be great to see this addressed by our community through some outreach and awareness programs."

Read the 52 comments — brain health shirky

Alan Cox Exits Intel, Linux Development
Posted by timothy on Thursday January 24, @08:46AM
from the deserves-all-kinds-of-awards dept.

judgecorp writes
"Linux kernel developer Alan Cox has left Intel and Linux development after slamming the Fedora 18 distribution. He made the announcement on Google+ and promised that he had not fallen out with Linus Torvalds, and would finish up all outstanding work."
Also at Live Mint, which calls Cox's resignation notice a "welcome change from the sterility, plain dishonesty of CEO departure statements." Cox says in that statement that he's leaving "for a bit," and "I may be back at some point in the future - who knows."

Read the 55 comments — intel linux developers

Google Pushing Back On Law Enforcement Requests For Access To Gmail Accounts
Posted by samzenpus on Thursday January 24, @08:05AM
from the for-your-eyes-only dept.

Virtucon writes
"Ars technica has an interesting article on how Google is handling requests from law enforcement for access to Gmail accounts. With the recent Petraeus scandal where no criminal conduct was found, it seems that they're re-enforcing their policies and standing up for their users. 'In order to compel us to produce content in Gmail we require an ECPA search warrant,' said Chris Gaither, Google spokesperson. 'If they come for registration information, that's one thing, but if they ask for content of email that's another thing.'"

Read the 27 comments — google yro technology

EFF Moves To Nix Trademark On "Gaymer" 101

Finland Is Crowdsourcing Its New Copyright Law
Posted by samzenpus on Thursday January 24, @05:37AM
from the power-to-the-people dept.

An anonymous reader writes
"Internet activists in Finland, upset with the country's strict copyright laws, are ready to take advantage of the country's promise to vote on any citizen-proposed bill that reaches 50,000 signatures. Digital rights group Common Sense in Copyright has proposed sweeping changes to Finland's Lex Karpela, a 2006 amendment to the Finnish copyright law that more firmly criminalized digital piracy. Under it, 'countless youngsters have been found guilty of copyright crimes and sentenced to pay thousands, in some cases hundreds of thousands, of euros in punitive damages to the copyright organizations.' The proposal to fix copyright is the best-rated and most-commented petition on the Open Ministry site."

Read the 71 comments — court finland yro

What Birds Know About Fractal Geometry
Posted by samzenpus on Thursday January 24, @03:01AM

Follow us:

Crown the champion of Battle IT
Join us as our contestants tackle some of the biggest issues facing IT pros today while they duke it out for IT supremacy.
Watch Now
Windows 8 Pro

Slashdot Poll
When Was the Last Time You Used a Landline Phone?
- ◯ Today
- ◯ This Week
- ◯ This Month
- ◯ This Year
- ◯ Over a Year
- ◯ Never
- ◯ What's a Landline Phone?

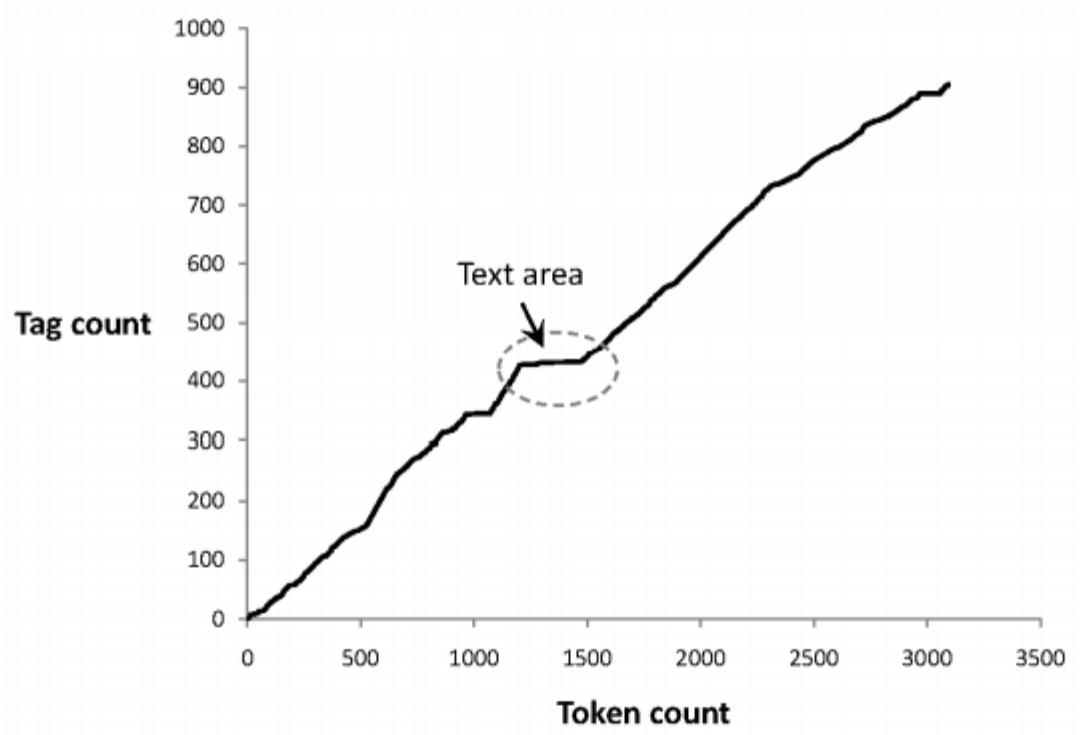
Vote — Read the 314 comments — Voted on 24534 times.

Order residential service today and get a special offer.
Simple. Fast. Convenient.
FREE QUOTE

All Scanners on Sale. This Week Only.
Plus free shipping on $50 or more.

5. Data Noise : Finding Content Blocks

- **Technique 1: Cumulative distribution of tags**
  - **Document slope curve** (e.g. Finn, Kushmerick & Smyth, 2001)



- **Other techniques in literature**

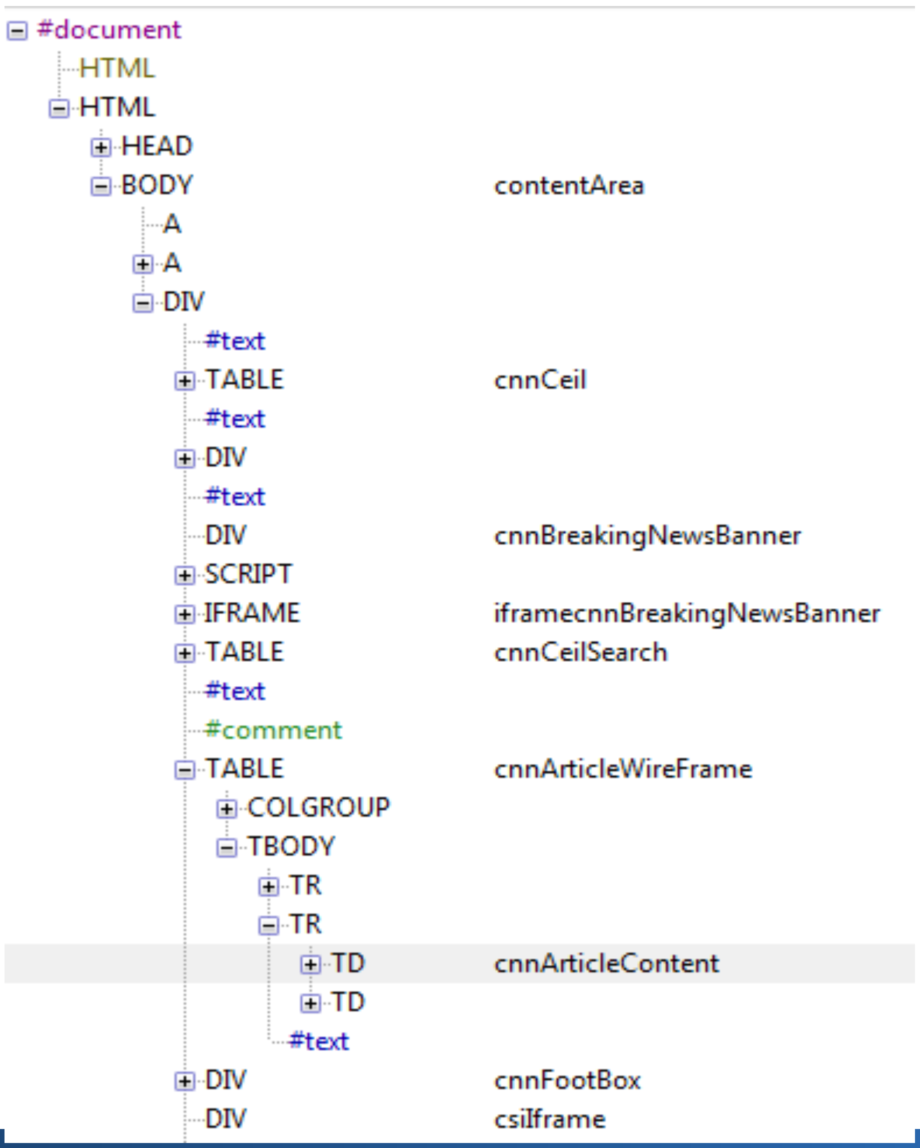
Finding Content Blocks

- Represent a web page as a sequence of bits, where $b_n = 1$ indicates that the *n*th token is a tag
- Optimization problem where we find values of *i* and *j* to maximize both the number of tags below *i* and above *j* and the number of non-tag tokens between *i* and *j*
- i.e., maximize

$$\sum_{n=0}^{i-1} b_n + \sum_{n=i}^{j} (1 - b_n) + \sum_{n=j+1}^{N-1} b_n$$

Finding Content Blocks

- Other approaches use DOM structure and visual (layout) features



```
#document
  HTML
  HTML
    HEAD
    BODY                      contentArea
      A
      A
      DIV
        #text
        TABLE                 cnnCeil
        #text
        DIV
        #text
        DIV                   cnnBreakingNewsBanner
        SCRIPT
        IFRAME                iframecnnBreakingNewsBanner
        TABLE                 cnnCeilSearch
        #text
        #comment
        TABLE                 cnnArticleWireFrame
          COLGROUP
          TBODY
            TR
            TR
              TD              cnnArticleContent
              TD
            #text
        DIV                   cnnFootBox
        DIV                   csiIframe
```

The Deep Web

- Places where crawlers rarely go...
  - Content behind login forms
  - Content behind JavaScript/TypeScript
  - Sites that aren’t linked from anywhere

- It is estimated that the deep web is 400-500x larger than the shallow web [http://dx.doi.org/10.3998/3336451.0007.104]

Main point

Just because you can, doesn’t mean you should.

Ways of Acquiring Web Data

- Data dumps
- Web APIs
- Targeted downloads

- Web crawling ← last option

Guidelines for bot writers

- An old (1993 !) but still very relevant text
  - http://www.robotstxt.org/guidelines.html

- Summary
  - **Reconsider** – do you really need a bot?
  - **Be Accountable** – if your actions cause problems, be available to take prompt action in response;
  - **Test Locally** – expand the scope gradually before you unleash your crawler on others;
  - **Don't hog resources** – web servers are for people primarily. Walk, don't run.
  - **Stay with it** – "it's vital to know what your robot is doing, and that it remains under control".

Guidelines for bot writers

- **Find out the sites' crawling policies**
  - GitHub (https://help.github.com/en/github/site-policy/github-acceptable-use-policies#5-scraping-and-api-usage-restrictions):

5. Scraping and API Usage Restrictions

Scraping refers to extracting data from our Service via an automated process, such as a bot or webcrawler. It does not refer to the collection of information through our API. Please see Section H of our Terms of Service and Corporate Terms of Service for our API Terms. You may scrape the website for the following reasons:

- *Researchers may scrape public, non-personal information from the Service for research purposes, only if any publications resulting from that research are open access.*

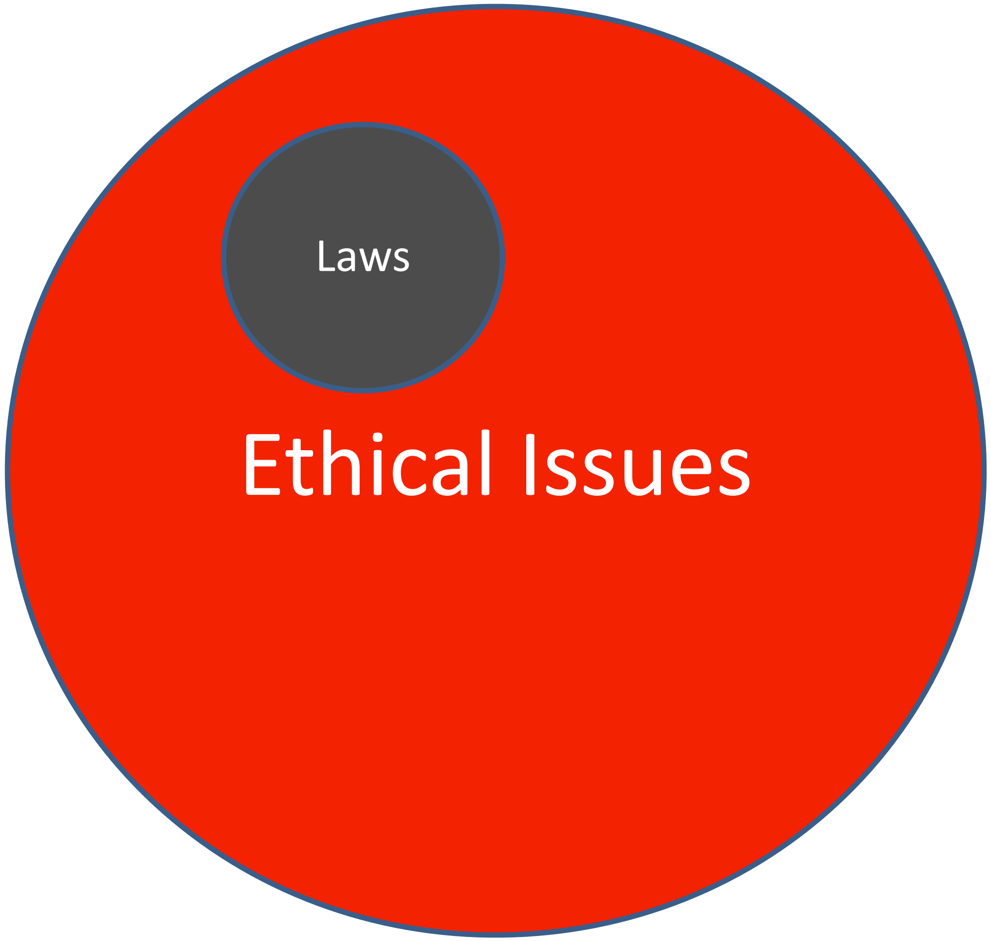
- *Archivists may scrape the Service for public data for archival purposes.*

You may not scrape the Service for spamming purposes, including for the purposes of selling User Personal Information (as defined in the GitHub Privacy Statement), such as to recruiters, headhunters, and job boards.

All use of data gathered through scraping must comply with the GitHub Privacy Statement.

- **Contact the Web admins**

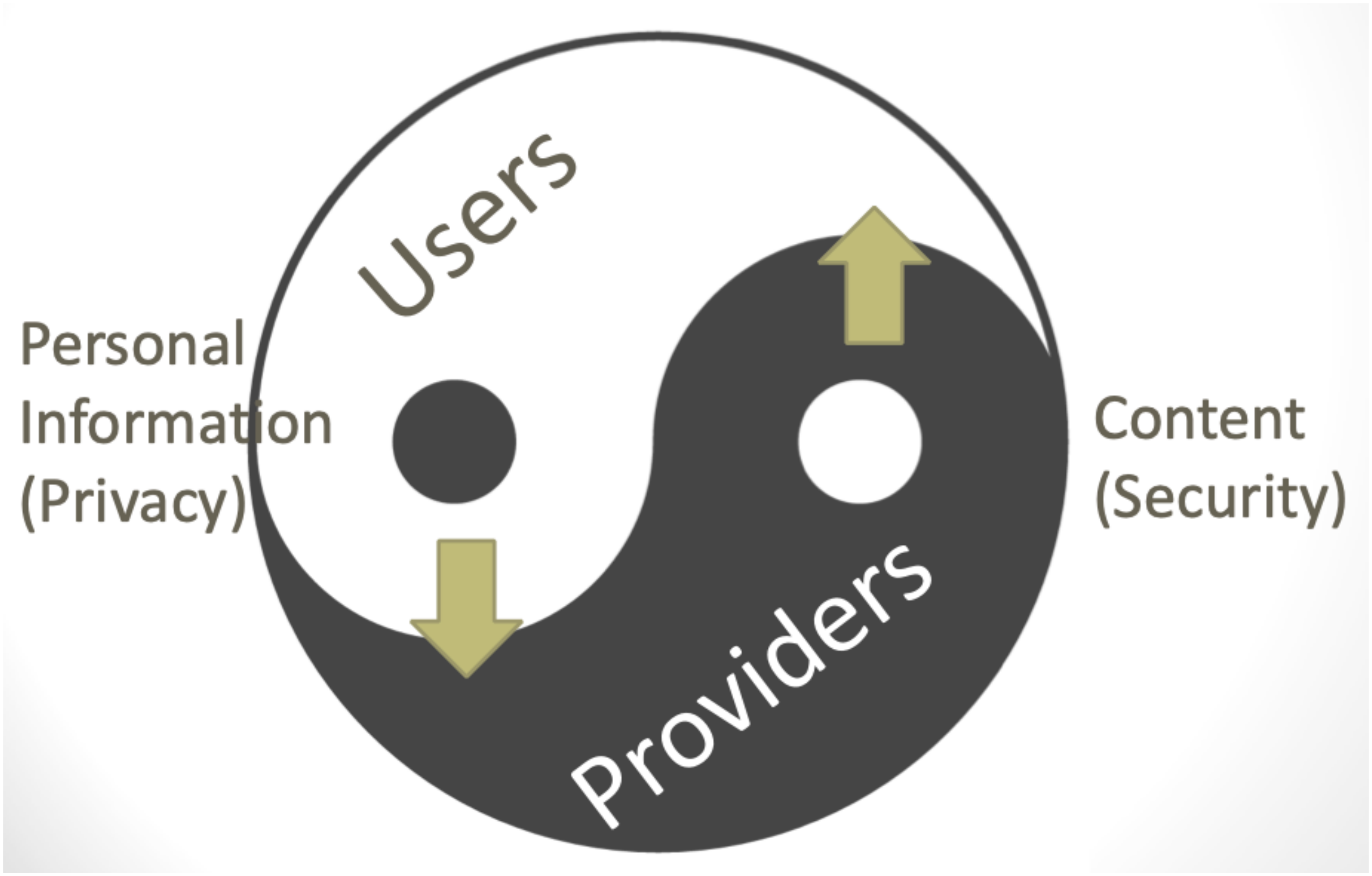
Ethical issues and law



The Web and the Law

- **Web: ~30 years old only**
- Wild Wild West, for the most part
- Very few laws
- Existing laws either untested or outdated
- **Our own judgments/actions matter**

Who can offend who?



Privacy Statements

- Web sites may have privacy statements:
  - What info they collect, how it’s used

- E.g. http://uci.edu/privacy.php

Privacy : Apache logs

- 66.249.66.18 - - [25/Jan/2012:00:18:27 -0800] "GET /xwiki/bin/export/Stats/CurrentYearActivity?format=rtf HTTP/1.1" 404 335 "-" "Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)”
- 180.76.6.28 - - [25/Jan/2012:00:18:31 -0800] "GET / HTTP/1.1" 200 45 "-" "Mozilla/5.0 (compatible; Baiduspider/2.0; +http://www.baidu.com/search/spider.html)”
- 66.249.66.18 - - [25/Jan/2012:00:19:02 -0800] "GET /calswim/coverage-s1.png HTTP/1.1" 200 961543 "-" "Googlebot-Image/1.0”
- 173.192.98.90 - - [25/Jan/2012:00:33:00 -0800] "GET / HTTP/1.1" 404 290 "-" "Python-urllib/2.4”
- 80.165.186.186 - - [25/Jan/2012:01:21:24 -0800] "GET /events-dataset-api/ HTTP/1.1" 200 3214 "http://www.kdnuggets.com/datasets/index.html" "Mozilla/5.0 (Windows NT 5.1; rv:9.0.1) Gecko/20100101 Firefox/9.0.1”
- 180.76.5.65 - - [25/Jan/2012:02:18:20 -0800] "GET / HTTP/1.1" 200 45 "-" "Mozilla/5.0 (compatible; Baiduspider/2.0;

Copyrights

- Products of intellect are ruled by “Copyright Law”
- Web content, source code, etc. are products of intellect
- See : http://www.copyright.gov
  - Next slides, from this site!

What is copyright?

- "Copyright is a form of protection grounded in the U.S. Constitution and granted by law for original works of authorship fixed in a tangible medium of expression. Copyright covers both published and unpublished works."

- *But then, can I use web material?*

https://www.copyright.gov/help/faq/faq-general.html

Fair use

- Various purposes for which the reproduction of a particular work may be considered fair:
  - Criticism
  - Comment
  - News
  - Teaching
  - Scholarship
  - Research

- It’s a very grey area (see https://www.copyright.gov/fls/fl102.html)

Fair use

- Four factors to be considered in determining whether or not a particular use is fair:
  - The purpose and character of the use, including whether such use is of commercial nature or is for nonprofit educational purposes
  - The nature of the copyrighted work
  - **The amount and substantiality of the portion used in relation to the copyrighted work as a whole**
  - The effect of the use upon the potential market for, or value of, the copyrighted work

Legal cases

- American Airlines vs. FareChase
  - March 10, 2003
  - American Airlines stops crawlers from Farechase for scraping AA.com website for "web fares" information (online fare comparison)
  - “trespass to chattels”
  - Jury (https://www.eff.org/document/preliminary-injunction-american-airlines-v-farechase-inc)
    - *Farechase’s actions are intentional and without American’s consent*
    - *Farechase’s conduct interferes with American’s computer network and system (…) such actions adversely affect and harm American and the condition, quality and value of American’s property.*
  - Settled suit later during that year.

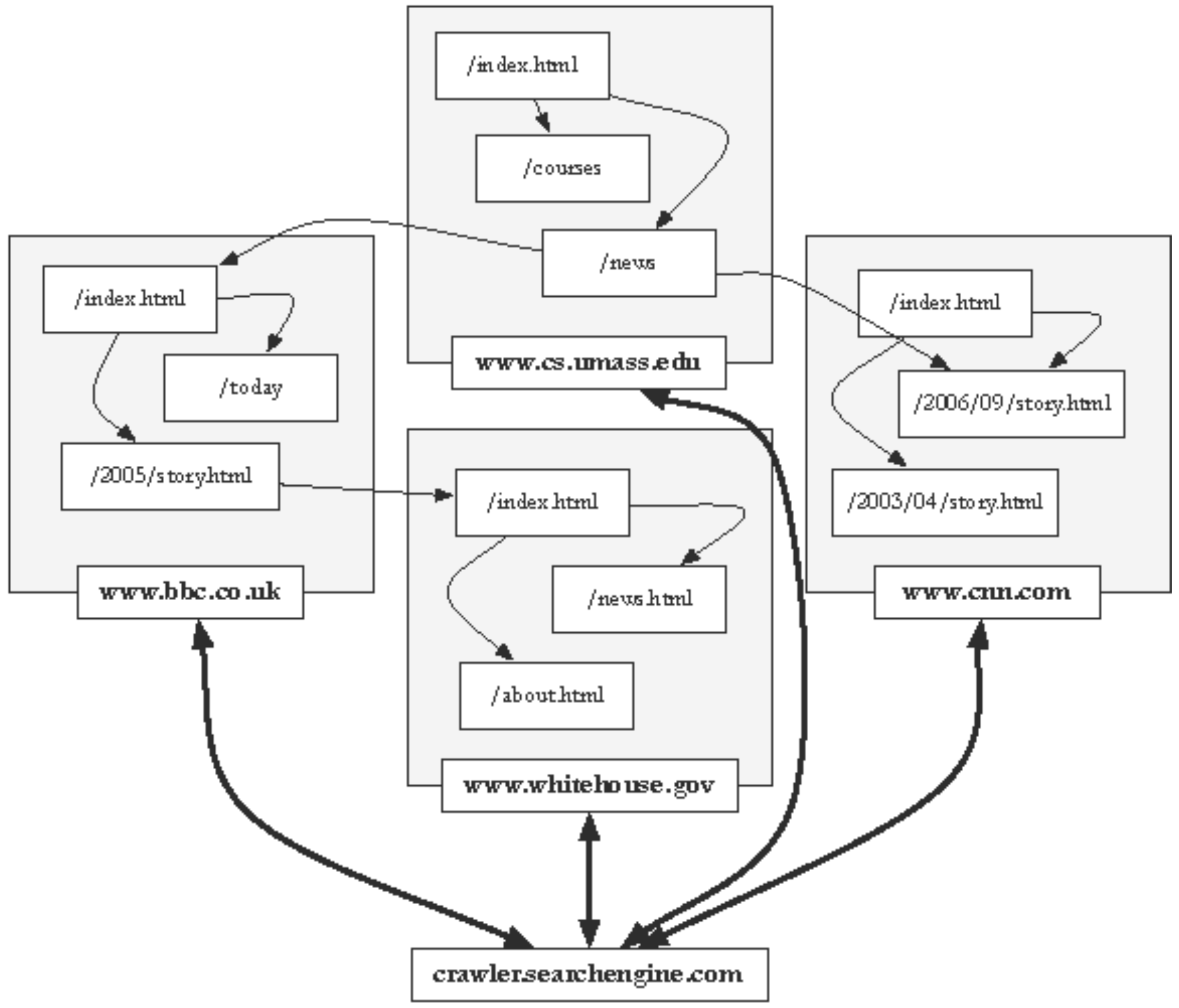
Back to crawling

Pseudo Code

```
procedure CrawlerThread(frontier)
    while not frontier.done() do
        website ← frontier.nextSite()
        url ← website.nextURL()
        if website.permitsCrawl(url) then
            text ← retrieveURL(url)
            storeDocument(url, text)
            for each url in parse(text) do
                frontier.addURL(url)
            end for
        end if
        frontier.releaseSite(website)
    end while
end procedure
```

Two major jobs : download webpages if authorized, and discover new URLs.

How the crawler sees the web



```
procedure CrawlerThread(frontier)
    while not frontier.done() do
        website ← frontier.nextSite()
        url ← website.nextURL()
        if website.permitsCrawl(url) then
            text ← retrieveURL(url)
            storeDocument(url, text)
            for each url in parse(text) do
                frontier.addURL(url)
            end for
        end if
        frontier.releaseSite(website)
    end while
end procedure
```

adapted from Addison Wesley, 2008 39

Retrieving Web Pages

- Web crawler client program connects to a *domain name system* (DNS) server

- DNS server translates the hostname into an *internet protocol* (IP) address

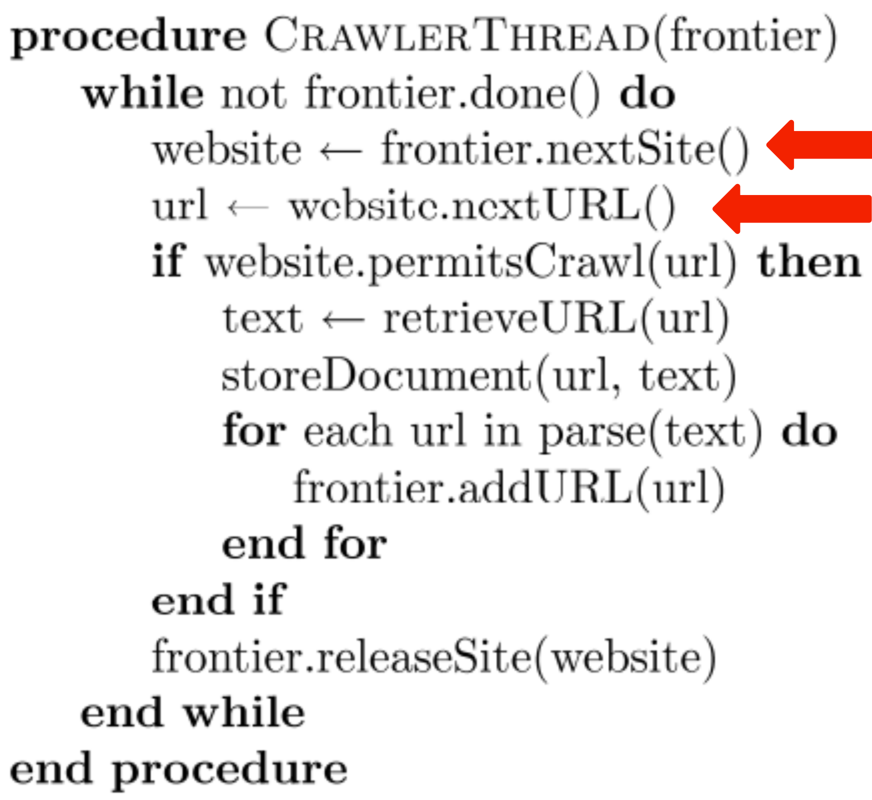
- Crawler then attempts to connect to server host using specific *port*

- After connection, crawler sends an HTTP request to the web server to request a page
  - usually a GET request

Web Crawling

- Problem: Web crawlers spend a lot of time waiting for responses to requests
- Solution: To reduce this inefficiency, web crawlers use threads and fetch hundreds of pages at once
- Drawback: Crawlers could potentially flood sites with requests for pages
- Mitigation: To avoid this problem, web crawlers use *politeness policies*
  - e.g., distribute requests between different servers, delay between requests to same web server, …

Pseudo Code : where to enforce politeness?



```
procedure CrawlerThread(frontier)
    while not frontier.done() do
        website ← frontier.nextSite()
        url ← website.nextURL()
        if website.permitsCrawl(url) then
            text ← retrieveURL(url)
            storeDocument(url, text)
            for each url in parse(text) do
                frontier.addURL(url)
            end for
        end if
        frontier.releaseSite(website)
    end while
end procedure
```

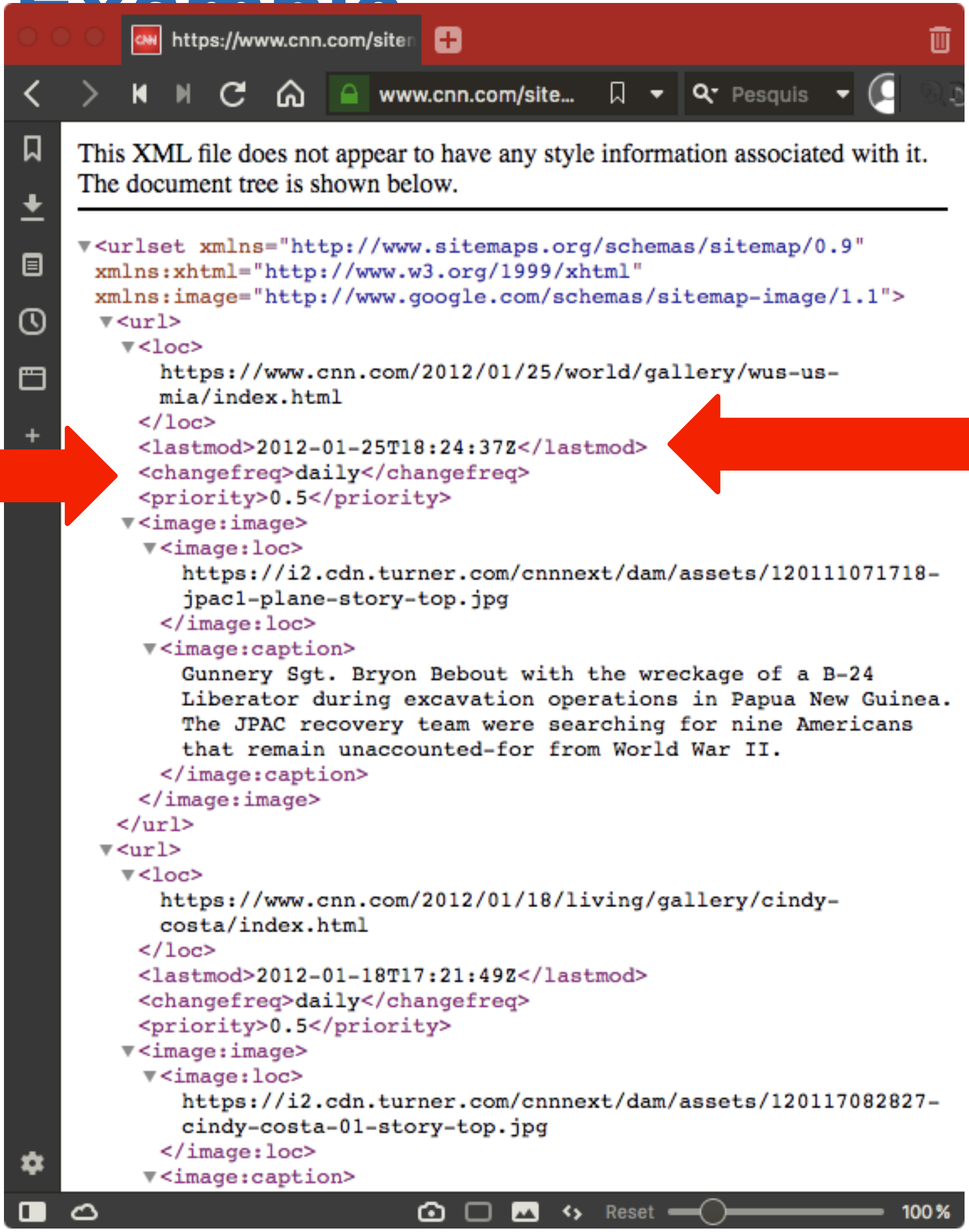
Can be here

And here

Sitemaps

- Sitemaps contain lists of URLs and data about those URLs, such as modification time and modification frequency
- Generated by web server administrators (usually by scripts)
- Can inform the crawler about pages it might not find
- Gives crawler a hint about when to check a page for changes

Sitemap Example



This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9"
 xmlns:xhtml="http://www.w3.org/1999/xhtml"
 xmlns:image="http://www.google.com/schemas/sitemap-image/1.1">
  ▼<url>
    ▼<loc>
       https://www.cnn.com/2012/01/25/world/gallery/wus-us-
       mia/index.html
     </loc>
     <lastmod>2012-01-25T18:24:37Z</lastmod>
     <changefreq>daily</changefreq>
     <priority>0.5</priority>
    ▼<image:image>
      ▼<image:loc>
         https://i2.cdn.turner.com/cnnnext/dam/assets/120111071718-
         jpac1-plane-story-top.jpg
       </image:loc>
      ▼<image:caption>
         Gunnery Sgt. Bryon Bebout with the wreckage of a B-24
         Liberator during excavation operations in Papua New Guinea.
         The JPAC recovery team were searching for nine Americans
         that remain unaccounted-for from World War II.
       </image:caption>
     </image:image>
   </url>
  ▼<url>
    ▼<loc>
       https://www.cnn.com/2012/01/18/living/gallery/cindy-
       costa/index.html
     </loc>
     <lastmod>2012-01-18T17:21:49Z</lastmod>
     <changefreq>daily</changefreq>
     <priority>0.5</priority>
    ▼<image:image>
      ▼<image:loc>
         https://i2.cdn.turner.com/cnnnext/dam/assets/120117082827-
         cindy-costa-01-story-top.jpg
       </image:loc>
      ▼<image:caption>
```

Freshness

- In a website, webpages are constantly: added, deleted, and modified

- *Stale* copies (*or their representation in our index*) no longer reflect the real contents of the web pages

- Web crawler must continually revisit pages
  - Check for modifications and maintain the freshness of the collection

Freshness

- HTTP protocol has a special request type called HEAD that makes it easy to check for page changes

Client request:
```
HEAD /csinfo/people.html HTTP/1.1
Host: www.cs.umass.edu
```

Server response:
```
HTTP/1.1 200 OK
Date: Thu, 03 Apr 2008 05:17:54 GMT
Server: Apache/2.0.52 (CentOS)
Last-Modified: Fri, 04 Jan 2008 15:28:39 GMT
ETag: "239c33-2576-2a2837c0"
Accept-Ranges: bytes
Content-Length: 9590
Connection: close
Content-Type: text/html; charset=ISO-8859-1
```

Freshness

- HTTP protocol has a special request type called HEAD that makes it easy to check for page changes
  - returns information about page, not page itself

Client request:

```
HEAD /csinfo/people.html HTTP/1.1
Host: www.cs.umass.edu
```

Server response:

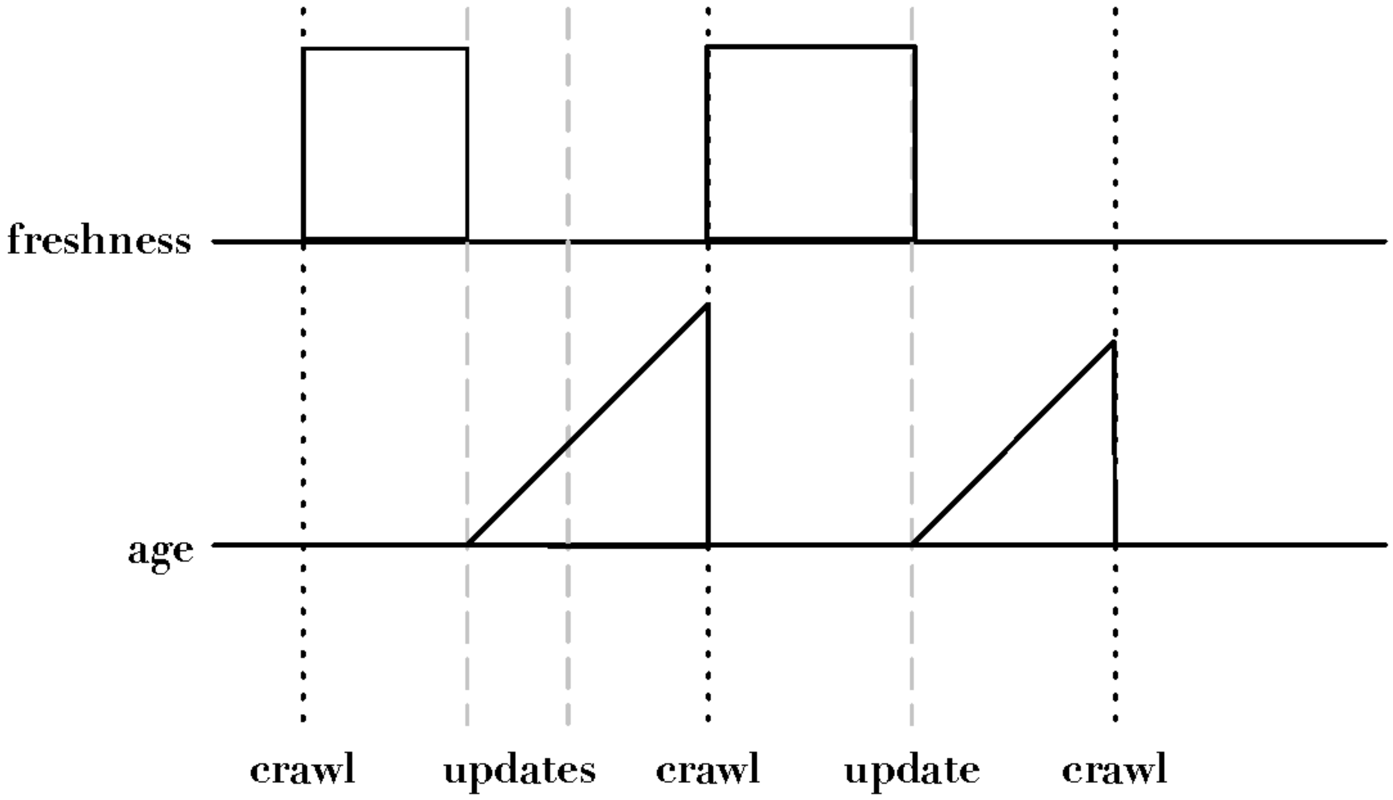
```
HTTP/1.1 200 OK
Date: Thu, 03 Apr 2008 05:17:54 GMT
Server: Apache/2.0.52 (CentOS)
Last-Modified: Fri, 04 Jan 2008 15:28:39 GMT
ETag: "239c33-2576-2a2837c0"
Accept-Ranges: bytes
Content-Length: 9590
Connection: close
Content-Type: text/html; charset=ISO-8859-1
```

Try, for instance using “curl --head https://www.ics.uci.edu/~algol/index.html”

Freshness

- Not possible to constantly check all pages
  - must check important pages and pages that change frequently
- Freshness is the proportion of pages that are fresh in the collection
- Optimizing for this metric is dangerous:
  - E.g. If you have a website that is always updated, optimizing your search engine for freshness would lead you to not crawl that site.
- Another metric: *Age*

Freshness vs. Age



Age

- Expected age of a page *t* days after it was last crawled:

$$\text{Age}(\lambda, t) = \int_0^t P(\text{page changed at time } x)(t - x)dx$$

Age

- Expected age of a page *t* days after it was last crawled:

$$\mathrm{Age}(\lambda, t) = \int_0^t \boxed{P(\text{page changed at time } x)}\boxed{(t-x)}\,dx$$

Age metric

The probability for the page to change at time x

some time difference between the crawling and the page change

Age

- Expected age of a page *t* days after it was last crawled:

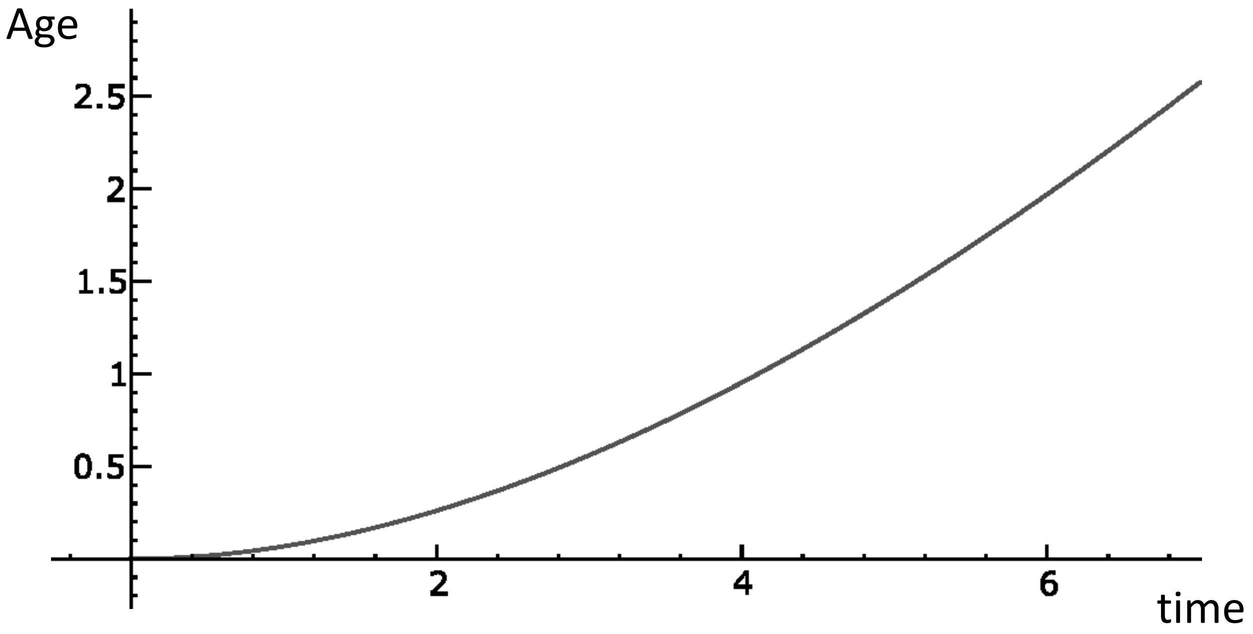
$$\text{Age}(\lambda, t) = \int_0^t P(\text{page changed at time } x)(t - x)dx$$

- Web page updates follow the Poisson distribution on average
  (Cho & Garcia-Molina, 2003)
  - time until the next update is governed by an exponential distribution

$$\text{Age}(\lambda, t) = \int_0^t \lambda e^{-\lambda x}(t - x)dx$$

Age

- Older a page gets, the more it costs not to crawl it
  - e.g., expected age with mean change frequency $\lambda = 1/7$ (one change per week)



Distributed Crawling

- Three reasons to use multiple computers for crawling
  - Helps to put the crawler closer to the sites it crawls
    - Lower requirements in the network
  - Reduces the number of sites each crawler has to remember
    - Lower requirements on local memory
  - Reduces the locally required resources

- Distributed crawlers may use a hash function to assign URLs to crawling computers/threads
  - Some hash function should be computed on the host part of each URL

Pseudo Code

```
procedure CrawlerThread(frontier)
    while not frontier.done() do
        website ← frontier.nextSite()
        url ← website.nextURL()
        if website.permitsCrawl(url) then
            text ← retrieveURL(url)
            storeDocument(url, text)
            for each url in parse(text) do
                frontier.addURL(url)
            end for
        end if
        frontier.releaseSite(website)
    end while
end procedure
```

Pseudo Code

The frontier information should be shared

```
procedure CrawlerThread(frontier)
    while not frontier.done() do
        website ← frontier.nextSite()
        url ← website.nextURL()
        if website.permitsCrawl(url) then
            text ← retrieveURL(url)
            storeDocument(url, text)
            for each url in parse(text) do
                frontier.addURL(url)
            end for
        end if
        frontier.releaseSite(website)
    end while
end procedure
```

```
procedure CrawlerThread(frontier)
    while not frontier.done() do
        website ← frontier.nextSite()
        url ← website.nextURL()
        if website.permitsCrawl(url) then
            text ← retrieveURL(url)
            storeDocument(url, text)
            for each url in parse(text) do
                frontier.addURL(url)
            end for
        end if
        frontier.releaseSite(website)
    end while
end procedure
```

```
procedure CrawlerThread(frontier)
    while not frontier.done() do
        website ← frontier.nextSite()
        url ← website.nextURL()
        if website.permitsCrawl(url) then
            text ← retrieveURL(url)
            storeDocument(url, text)
            for each url in parse(text) do
                frontier.addURL(url)
            end for
        end if
        frontier.releaseSite(website)
    end while
end procedure
```

```
procedure CrawlerThread(frontier)
    while not frontier.done() do
        website ← frontier.nextSite()
        url ← website.nextURL()
        if website.permitsCrawl(url) then
            text ← retrieveURL(url)
            storeDocument(url, text)
            for each url in parse(text) do
                frontier.addURL(url)
            end for
        end if
        frontier.releaseSite(website)
    end while
end procedure
```

```
procedure CrawlerThread(frontier)
    while not frontier.done() do
        website ← frontier.nextSite()
        url ← website.nextURL()
        if website.permitsCrawl(url) then
            text ← retrieveURL(url)
            storeDocument(url, text)
            for each url in parse(text) do
                frontier.addURL(url)
            end for
        end if
        frontier.releaseSite(website)
    end while
end procedure
```

…

Other types of crawling

- Information Retrieval

Information adapted from Addison Wesley, 2008

Focused Crawling

- Attempts to download only those pages that are about a particular topic
  - used by vertical search applications
  - can also be used to improve a search engine in a specific topic
    - Or around specific events in time

- Rely on the fact that pages about a topic tend to have links to other pages on the same topic
  - popular pages for a topic are typically used as seeds
  - E.g. pages that many users are clicking after they search a keyword

- Crawler uses some *text classifier* to decide whether a page is on topic

Desktop Crawls

- Used for desktop search and enterprise search
- Differences to web crawling with respect to the web
    - Much easier to find the data (file system ~ sitemap)
    - Responding quickly to updates is more important
    - Must be conservative in terms of disk, memory and CPU usage
        - *It should be transparent to the user.*
    - Need to cope with many different document formats
    - Data privacy much more important than in the web context

Document Feeds Crawls

- In general on the web: documents are created and updated
- But many documents are also just *published*
  - created at a fixed time and rarely updated again
  - e.g., news articles, blog posts, press releases, email
- Document feed: Published documents from a single source can be ordered in a sequence
  - Crawlers can find new documents by examining the end of the feed

Document Feeds

- Two types of document feeds:
  - A push feed alerts the subscriber to new documents
    - E.g. news agencies
  - A pull feed requires the subscriber to check periodically for new documents

- A *push* mechanism involves a subscription:
  - E.g. WebSub (https://www.w3.org/TR/websub/)

- Most common format for (*pull*) feeds is called *RSS*
  - Really Simple Syndication, Resource Description Framework Site Summary, Rich Site Summary

RSS Example

```
<?xml version="1.0"?>
<rss version="2.0">
  <channel>
    <title>Search Engine News</title>
    <link>http://www.search-engine-news.org/</link>
    <description>News about search engines.</description>
    <language>en-us</language>
    <pubDate>Tue, 19 Jun 2008 05:17:00 GMT</pubDate>
    <ttl>60</ttl>

    <item>
      <title>Upcoming SIGIR Conference</title>
      <link>http://www.sigir.org/conference</link>
      <description>The annual SIGIR conference is coming!
        Mark your calendars and check for cheap
        flights.</description>
      <pubDate>Tue, 05 Jun 2008 09:50:11 GMT</pubDate>
      <guid>http://search-engine-news.org#500</guid>
    </item>
```

RSS Example

FEED CHANNEL INFO

```
<?xml version="1.0"?>
<rss version="2.0">
  <channel>
    <title>Search Engine News</title>
    <link>http://www.search-engine-news.org/</link>
    <description>News about search engines.</description>
    <language>en-us</language>
    <pubDate>Tue, 19 Jun 2008 05:17:00 GMT</pubDate>
    <ttl>60</ttl>
```

ITEM INFO

```
    <item>
      <title>Upcoming SIGIR Conference</title>
      <link>http://www.sigir.org/conference</link>
      <description>The annual SIGIR conference is coming!
        Mark your calendars and check for cheap
        flights.</description>
      <pubDate>Tue, 05 Jun 2008 09:50:11 GMT</pubDate>
      <guid>http://search-engine-news.org#500</guid>
    </item>
      …
```

RSS Example

```
<?xml version="1.0"?>
<rss version="2.0">
  <channel>
    <title>Search Engine News</title>
    <link>http://www.search-engine-news.org/</link>
    <description>News about search engines.</description>
    <language>en-us</language>
    <pubDate>Tue, 19 Jun 2008 05:17:00 GMT</pubDate>
    <ttl>60</ttl>

    <item>
      <title>Upcoming SIGIR Conference</title>
      <link>http://www.sigir.org/conference</link>
      <description>The annual SIGIR conference is coming!
        Mark your calendars and check for cheap
        flights.</description>
      <pubDate>Tue, 05 Jun 2008 09:50:11 GMT</pubDate>
      <guid>http://search-engine-news.org#500</guid>
    </item>
```

RSS Example

```
...
   <item>
      <title>New Search Engine Textbook</title>
      <link>http://www.cs.umass.edu/search-book</link>
      <description>A new textbook about search engines
        will be published soon.</description>
      <pubDate>Tue, 05 Jun 2008 09:33:01 GMT</pubDate>
      <guid>http://search-engine-news.org#499</guid>
    </item>
  </channel>
</rss>
```

Major characteristics of RSS to crawlers

- `ttl` tag (time to live)
  - amount of time (in minutes) contents should be cached
- RSS feeds are accessed like web pages
  - using HTTP GET requests to web servers that host them
  - You can use HTTP HEAD requests to check if the feed has changed
- Easy for crawlers to parse (it is just XML)
- Easy to find new information:
  - just GET, parse and if any, grab the new <item> … </item> s.