Introduction to

# Information Retrieval

Lecture 17: Crawling and web indexes

# Today’s lecture

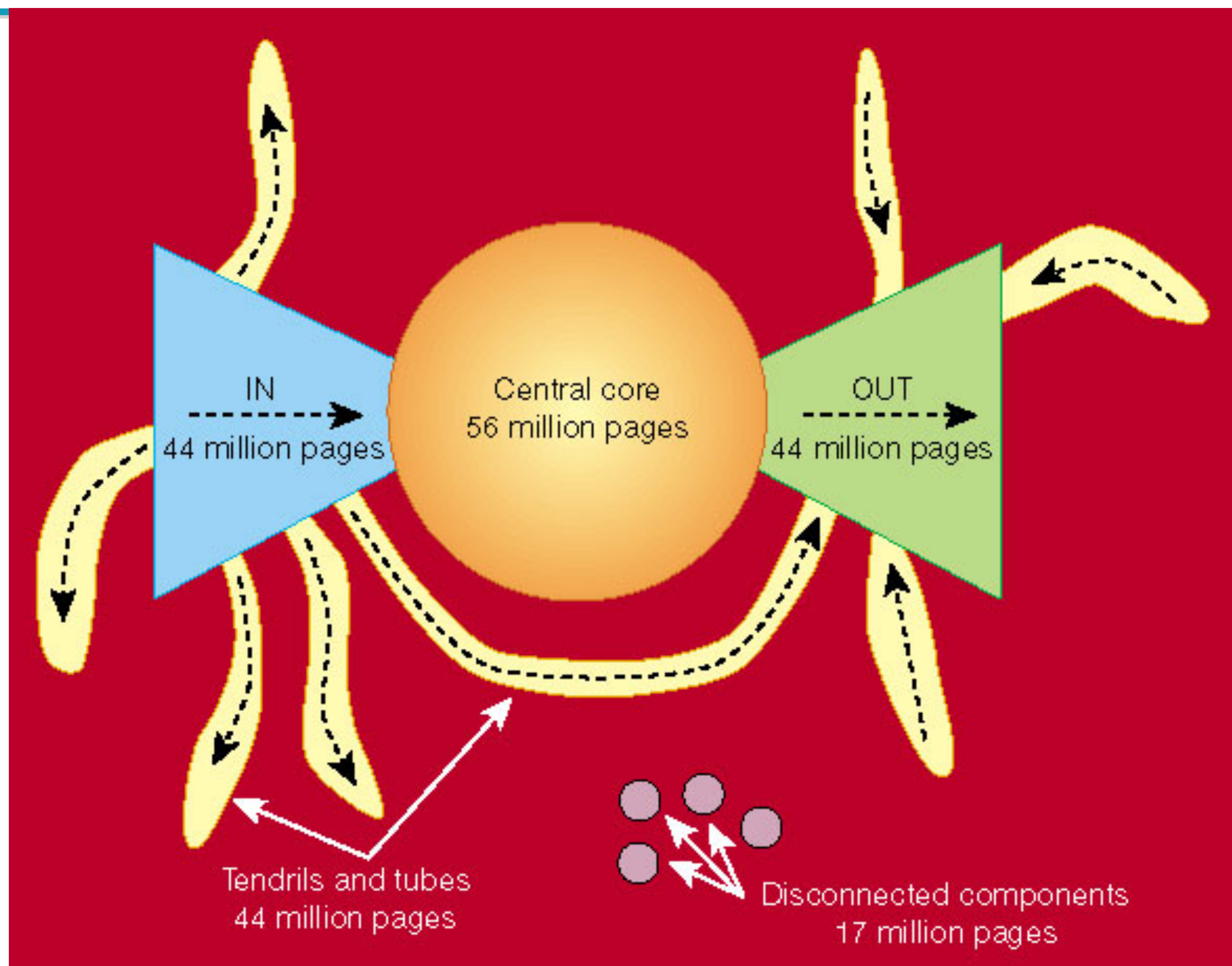
- Crawling

# Basic crawler operation

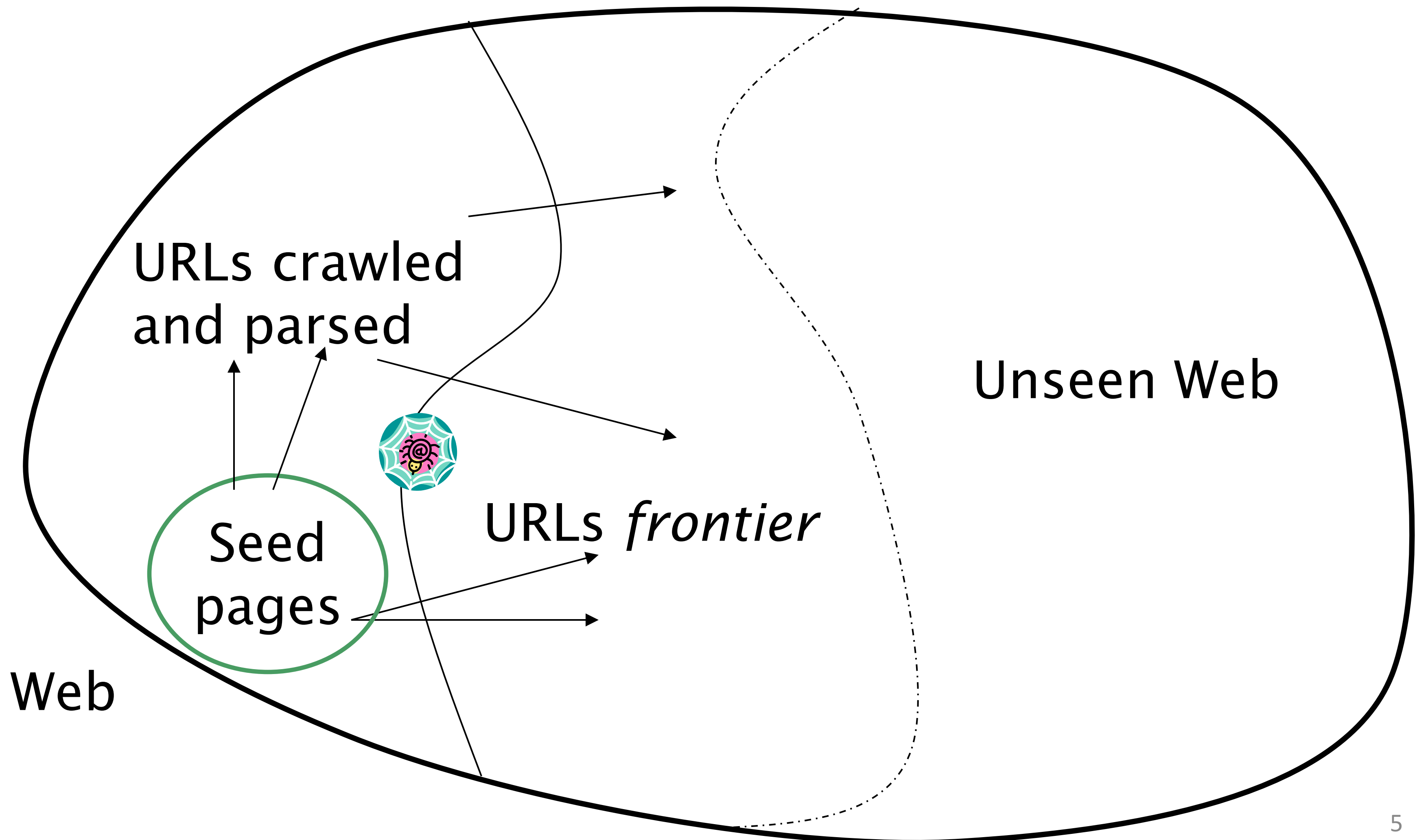
- Begin with known "seed" URLs
- Fetch and parse them
  - Extract URLs they point to
  - Place the extracted URLs on a queue
- Fetch each URL on the queue and repeat

http://www.nature.com/nature/journal/v405/n6783/pdf/405112a0.pdf

# Structure of the Web (circa 2000)



# Crawling picture



# Simple Crawler Thread

```
1  procedure CrawlerThread(frontier)
2      while not frontier.done() do
3          website ← frontier.nextSite()
4          url ← website.nextURL()
5          if website.permitsCrawl(url) then
6              text ← retrieveURL(url)
7              storeDocument(url, text)
8              for each url in parse(text) do
9                  frontier.addURL(url)
10             end for
11         end if
12         frontier.releaseSite(website)
13     end while
14 end procedure
```

# Simple picture – complications

- Web crawling isn't feasible with one machine
  - All of the above steps distributed
- Malicious pages
  - Spam pages
  - Spider traps – incl dynamically generated
- Even non-malicious pages pose challenges
  - Latency/bandwidth to remote servers vary
  - Webmasters' stipulations
    - How "deep" should you crawl a site's URL hierarchy?
  - Site mirrors and duplicate pages
- Politeness – don't hit a server too often

# What any crawler *must* do

- Be Polite: Respect implicit and explicit politeness considerations
  - Only crawl allowed pages
  - Respect *robots.txt* (more on this shortly)
- Be Robust: Be immune to spider traps and other malicious behavior from web servers

# What any crawler *should* do

- Be capable of distributed operation: designed to run on multiple distributed machines
- Be scalable: designed to increase the crawl rate by adding more machines
- Performance/efficiency: permit full use of available processing and network resources

# 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

# 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
```

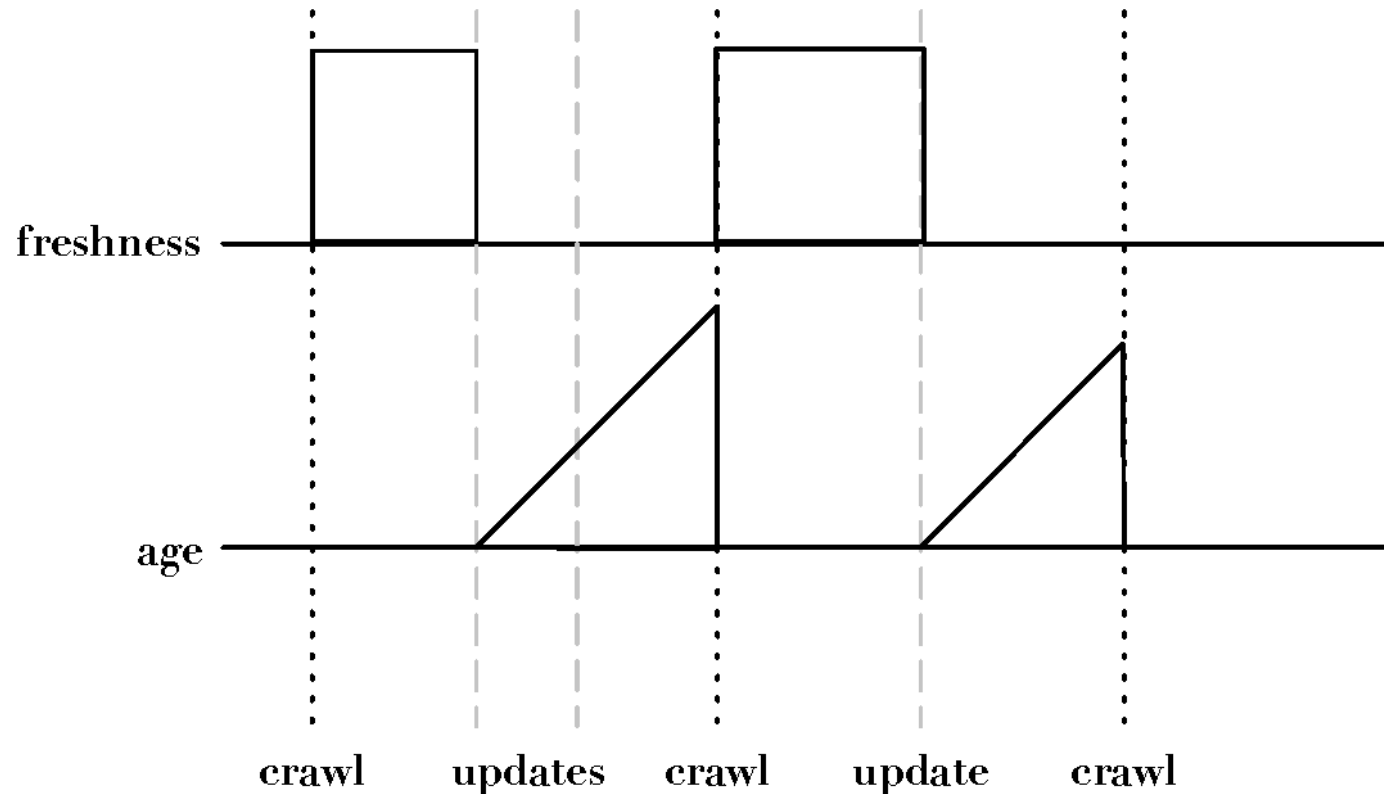
# Freshness

- Web pages are constantly being added, deleted, and modified
- Web crawler must continually revisit pages it has already crawled to see if they have changed in order to maintain the *freshness* of the document collection
  - *stale* copies no longer reflect the real contents of the web pages

# 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
- Optimizing for this metric can lead to bad decisions, such as not crawling popular sites
- *Age* is a better metric

# Freshness vs. Age



# Age

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

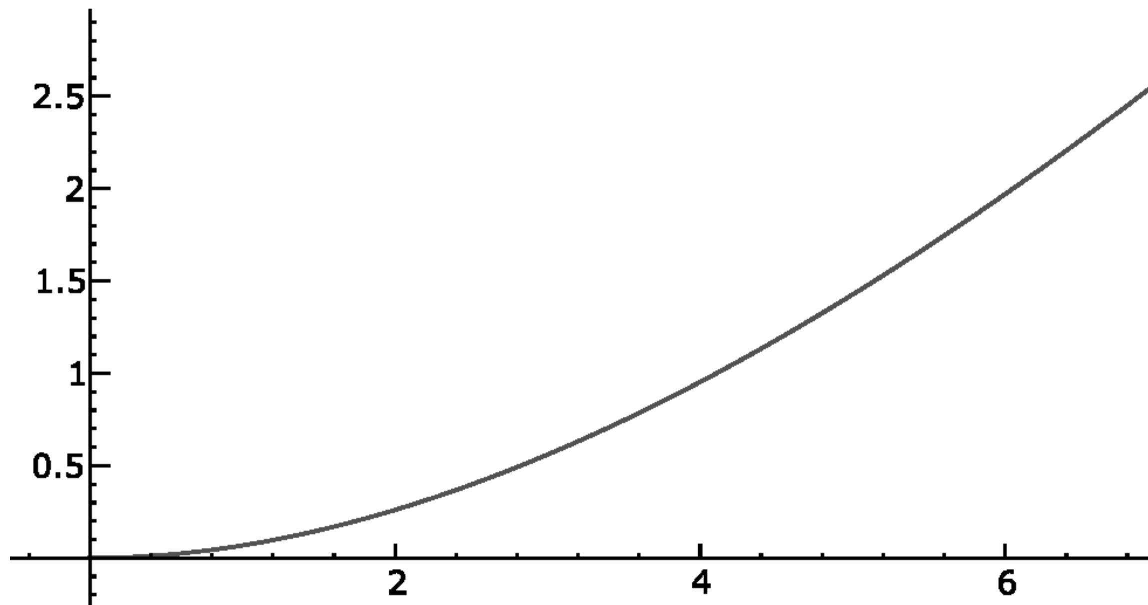
$$\text{Age}(\lambda, t) = \int_0^t P(\text{page changed at time } x)(t - x)dx$$

- Web page updates follow the Poisson distribution on average
  - time until the next update is governed by an exponential distribution

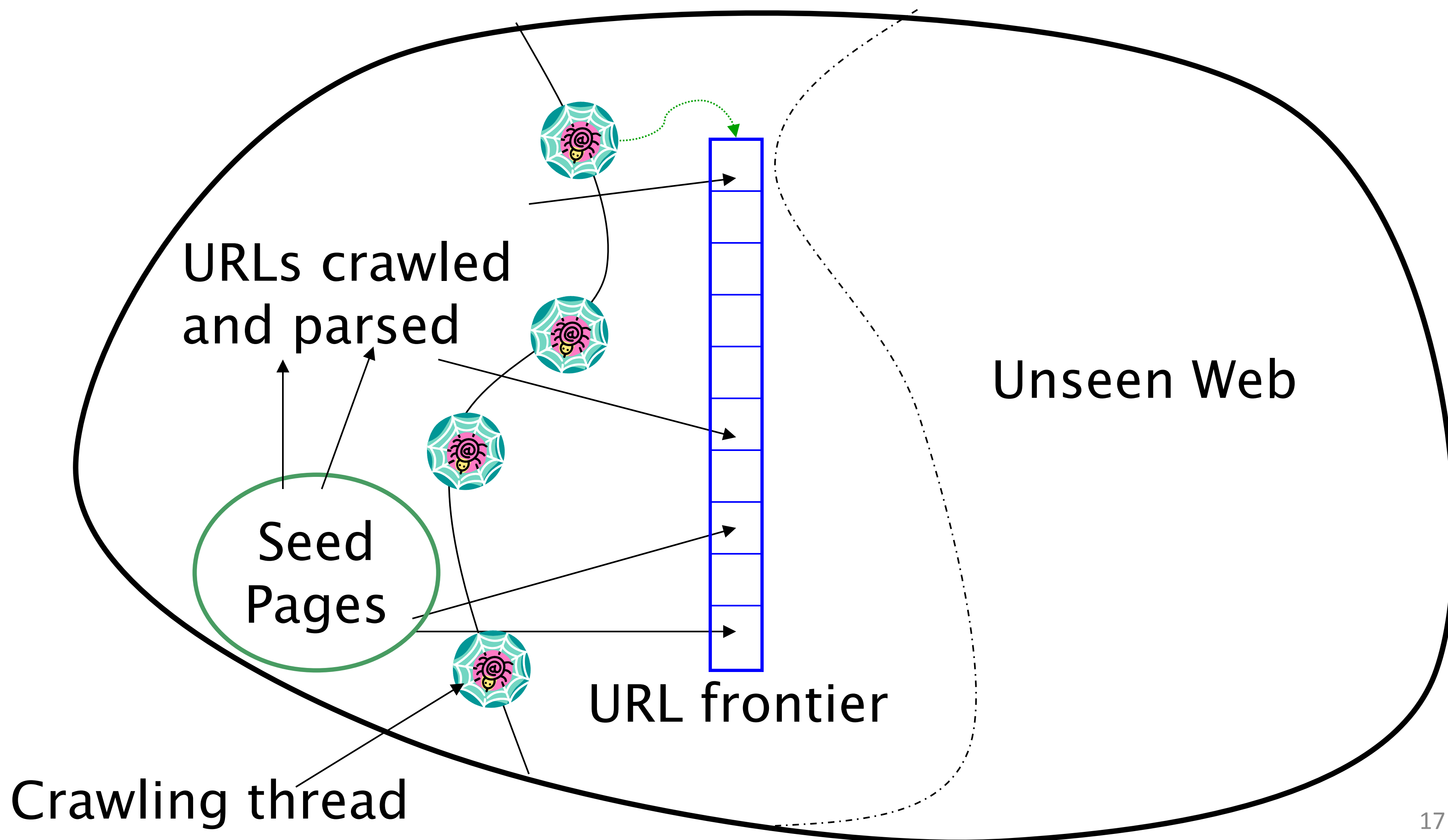
$$\text{Age}(\lambda, t) = \int_0^t \lambda e^{-\lambda x}(t - x)dx$$

# Age

- The 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)



# Updated crawling picture



# Simple Crawler Thread

```
1   procedure CrawlerThread(frontier)
2       while not frontier.done() do
3           website ← frontier.nextSite()
4           url ← website.nextURL()
5           if website.permitsCrawl(url) then
6               text ← retrieveURL(url)
7               storeDocument(url, text)
8               for each url in parse(text) do
9                   frontier.addURL(url)
10              end for
11          end if
12          frontier.releaseSite(website)
13      end while
14  end procedure
```

# URL frontier

- Can include multiple pages from the same host
- Must avoid trying to fetch them all at the same time
- Must try to keep all crawling threads busy

# 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

# 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 URL, create a file `URL/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:
```

try http://www.taobao.com/robots.txt

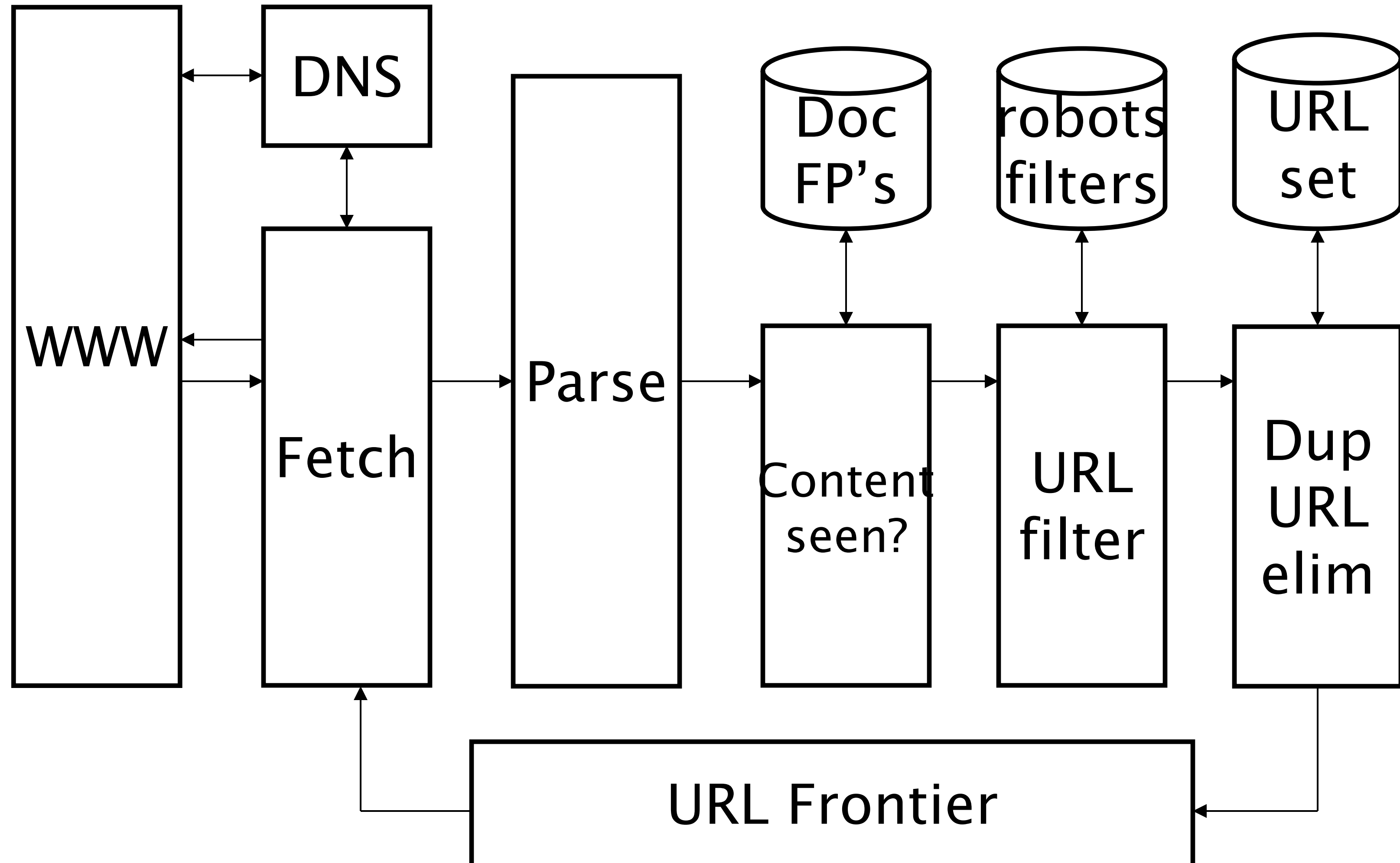
# Processing steps in crawling

- Pick a URL from the frontier
- Fetch the document at the URL
- Parse the URL
  - Extract links from it to other docs (URLs)
- Check if URL has content already seen
  - If not, add to indexes
- For each extracted URL
  - Ensure it passes certain URL filter tests
  - Check if it is already in the frontier (duplicate URL elimination)

Which one?

E.g., only crawl .edu, obey robots.txt, etc.

# Basic crawl architecture



# DNS (Domain Name Server)

- A lookup service on the internet
  - Given a URL, retrieve its IP address
  - Service provided by a distributed set of servers – thus, lookup latencies can be high (even seconds)
- Common OS implementations of DNS lookup are *blocking*: only one outstanding request at a time
- Solutions
  - DNS caching
  - Batch DNS resolver – collects requests and sends them out together

# Parsing: URL normalization

- When a fetched document is parsed, some of the extracted links are *relative* URLs
- E.g., at http://en.wikipedia.org/wiki/Main_Page

we have a relative link to /wiki/Wikipedia:General_disclaimer which is the same as the absolute URL http://en.wikipedia.org/wiki/Wikipedia:General_disclaimer

- During parsing, must normalize (expand) such relative URLs

# Content seen?

- Duplication is widespread on the web
- If the page just fetched is already in the index, do not further process it
- This is verified using document fingerprints or shingles

# Removing Noise

- Many web pages contain text, links, and pictures that are not directly related to the main content of the page
- This additional material is mostly *noise* that could negatively affect the ranking of the page
- Techniques have been developed to detect the content blocks in a web page
  - Non-content material is either ignored or reduced in importance in the indexing process

# Noise Example

CNN.com Member Center: Sign In | Register International Edition

SEARCH THE WEB CNN.COM Search

Home Page
World
U.S.
Weather
Business
Sports
Analysis
Politics
Law
Technology
Science & Space
Health
Entertainment
Offbeat
Travel
Education
Special Reports
Video
Autos
I-Reports

SERVICES
E-mails
RSS
Podcasts
Mobile
CNN Pipeline

SEARCH WEB CNN.COM Search

## SCIENCE & SPACE

### Aquarium plays whale shark matchmaker

**Two females flown 8,000 miles for double date in Atlanta**

Monday, June 5, 2006; Posted: 5:28 p.m. EDT (21:28 GMT)

ATLANTA, Georgia (CNN) -- Ralph and Norton, meet Alice and Trixie.

The Georgia Aquarium's two male whale sharks got some female companionship on Saturday, when they were joined by two females transported to Atlanta from Taipei, Taiwan.

Researchers are hoping the sharks will mate.

The females -- 11 feet and 14 feet long -- were flown more than 8,000 miles by UPS, which reconfigured a company B-747 freighter with advanced marine life support systems to carry them. ( Watch what it took to get the sharks together -- 1:55)

The pilot said they treated the massive fish like first-class passengers.

"As we were doing the descent, we asked to start down a little sooner to make a nice shallow descent, to not make things too uncomfortable back there for the whale sharks," UPS pilot Capt. Bob Crum said.

The plane's center of balance was carefully planned, according to a statement from the aquarium, and veterinarians accompanied the sharks.

The delivery company also brought the two males to Atlanta, where researchers can study the whale sharks' behavior, breeding and development.

The whale sharks -- named after the main characters in the 1950s sitcom "The Honeymooners" -- were delivered to the aquarium in special transportation containers.

The Georgia Aquarium, which opened in November, is the world's largest aquarium. It was a $250 million gift to Georgia from Bernie Marcus, co-founder of The Home Depot and his wife, Billi, through the Marcus Foundation.

It is the only aquarium outside of Asia to showcase whale sharks, which are the largest fish on Earth.

The aquarium's 6.2-million gallon "Ocean Voyager" tank can hold up to six whale sharks...

Alice the whale shark swims into the Ocean Voyager tank at the Georgia Aquarium for the first time.

YOUR E-MAIL ALERTS
Atlanta (Georgia)
Taiwan
ACTIVATE or Create Your Own
Manage Alerts | What Is This?

Search Jobs
Enter Keywords
Enter City ALL
careerbuilder.com

Save up to 75% on Last-Minute Cruises.
VacationsToGo.com
Best Price Guarantee
GO!

Story Tools

Subscribe to Time for $1.99

SPACE
Section Page | Video

Astronauts prepare for third spacewalk
• Astronomers vie to make biggest telescope
• NASA to beam Beatles song to North Star
• U.S. plans for falling satellite

TOP STORIES
Home Page | Video | Most Popular

Russians choose Putin's successor
• Iran's president makes landmark visit to Iraq
• Israel PM: Attacks on militants go on
• Cabbie arrested in abandoned baby case

International Edition | Languages | CNN TV | CNN International | Headline News | Transcripts | Advertise with Us | About Us

SEARCH THE WEB CNN.COM Search

© 2007 Cable News Network.
A Time Warner Company. All Rights Reserved.
Terms under which this service is provided to you.
Read our privacy guidelines. Contact us. Site Map.

External sites open in new window; not endorsed by CNN.com
Pay service with live and archived video. Learn more
Download audio news | Add RSS headlines

Content block

# Filters and robots.txt

- Filters – regular expressions for URL's to be crawled/not
- Once a robots.txt file is fetched from a site, need not fetch it repeatedly
  - Doing so burns bandwidth, hits web server
- Cache robots.txt files

# Duplicate URL elimination

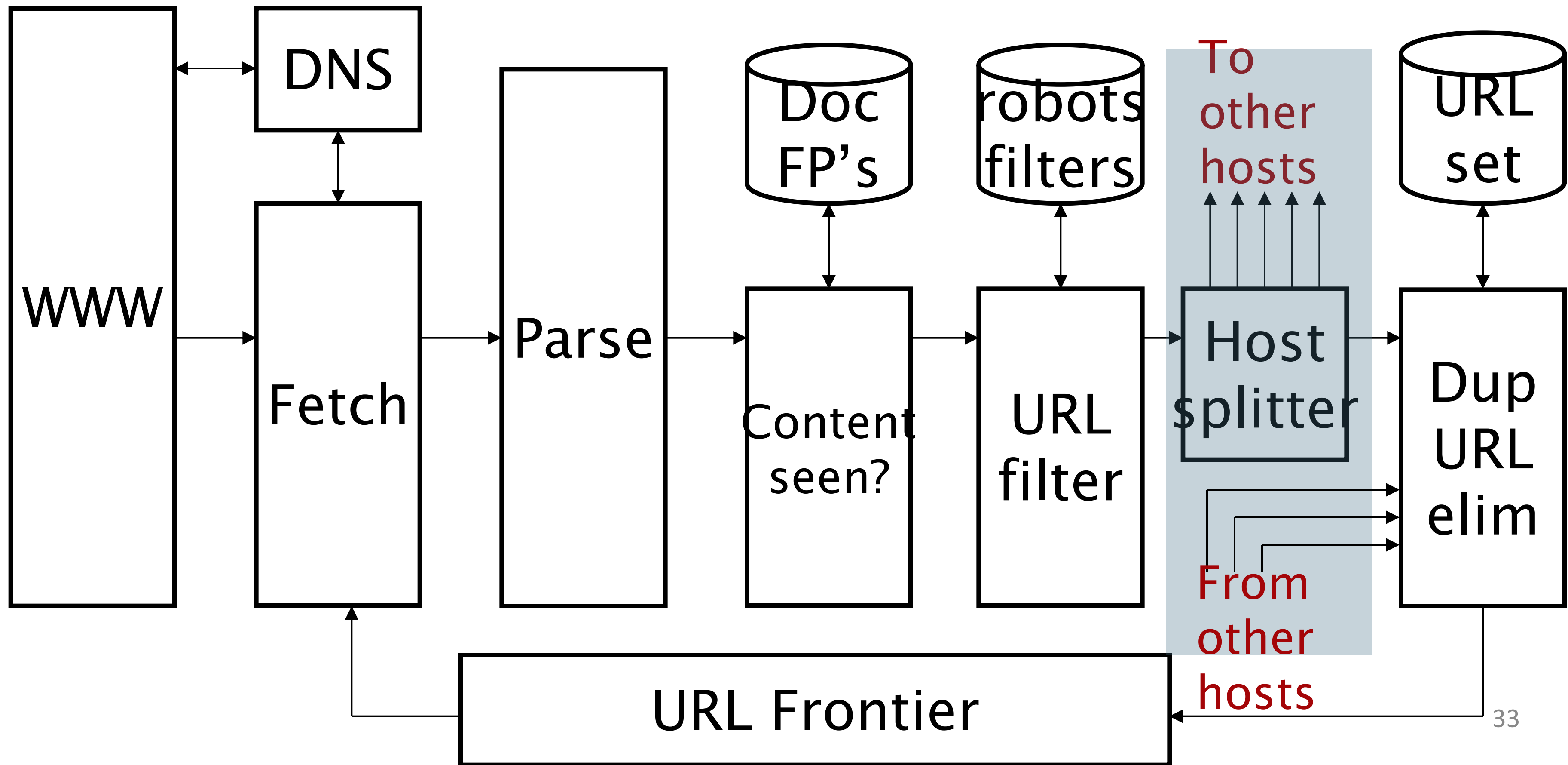
- For a non-continuous (one-shot) crawl, test to see if an extracted+filtered URL has already been passed to the frontier
- For a continuous crawl – see details of frontier implementation

# Distributing the crawler

- Run multiple crawl threads, under different processes – potentially at different nodes
  - Geographically distributed nodes
- Partition hosts being crawled into nodes
  - Hash used for partition
- How do these nodes communicate?

# Communication between nodes

- The output of the URL filter at each node is sent to the Duplicate URL Eliminator at all nodes



# URL frontier: two main considerations

- Politeness: do not hit a web server too frequently
- Freshness: crawl some pages more often than others
  - E.g., pages (such as News sites) whose content changes often

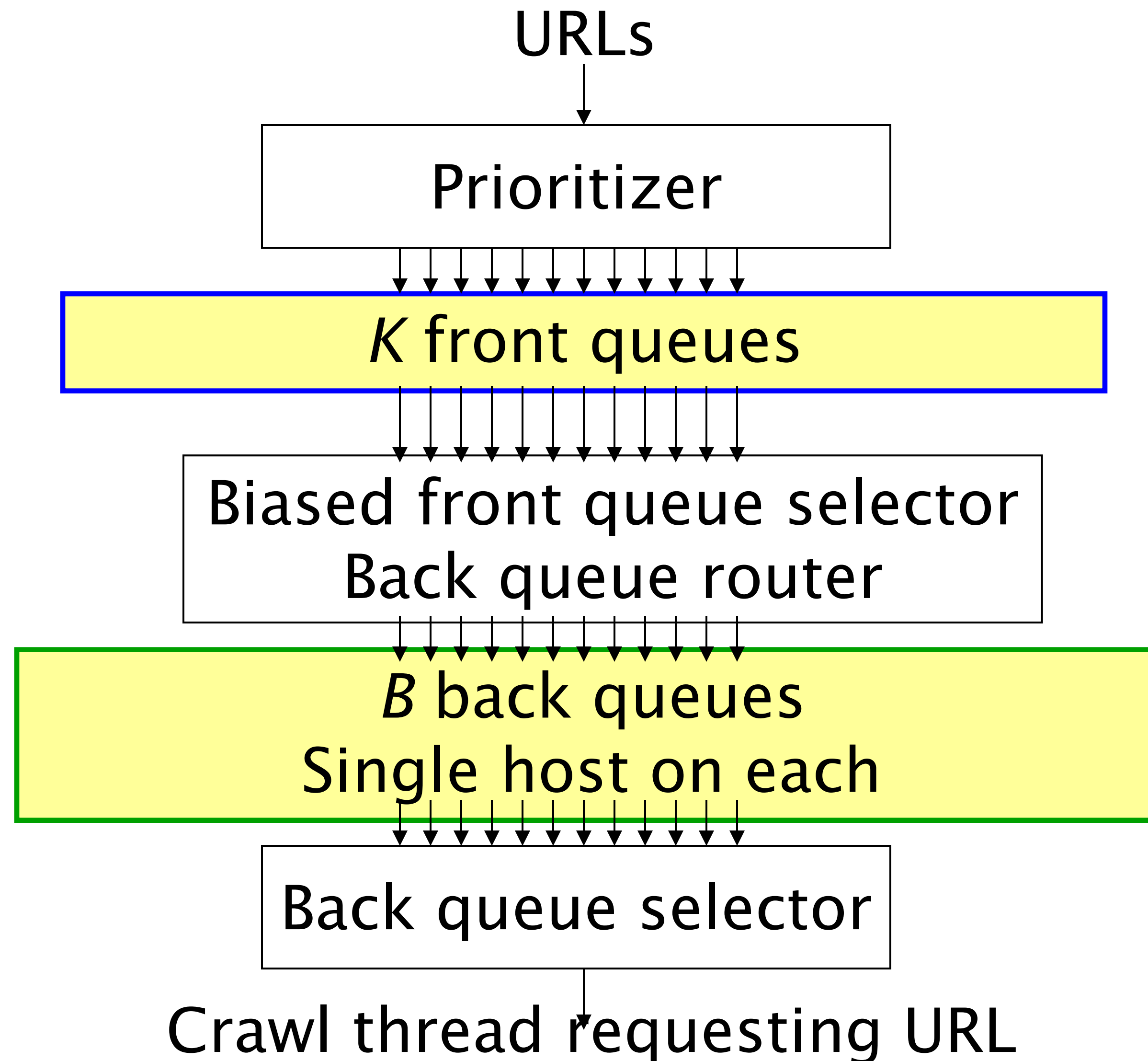
These goals may conflict each other.

(E.g., simple priority queue fails – many links out of a page go to its own site, creating a burst of accesses to that site.)

# Politeness – challenges

- Even if we restrict only one thread to fetch from a host, can hit it repeatedly
- Common heuristic: insert time gap between successive requests to a host that is >> time for most recent fetch from that host

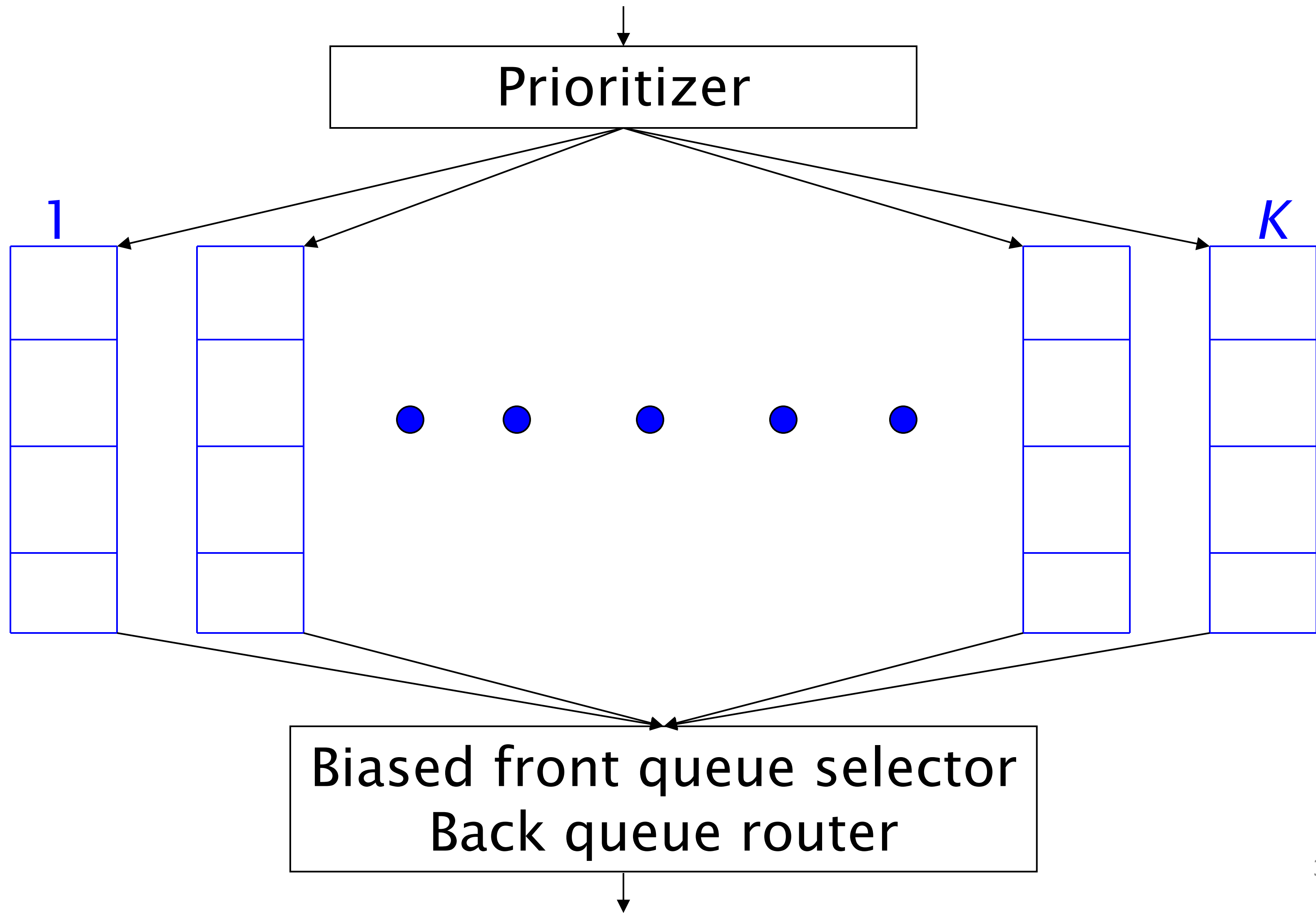
# URL frontier: Mercator scheme



# Mercator URL frontier

- URLs flow in from the top into the frontier
- Front queues manage prioritization
- Back queues enforce politeness
- Each queue is FIFO

# Front queues



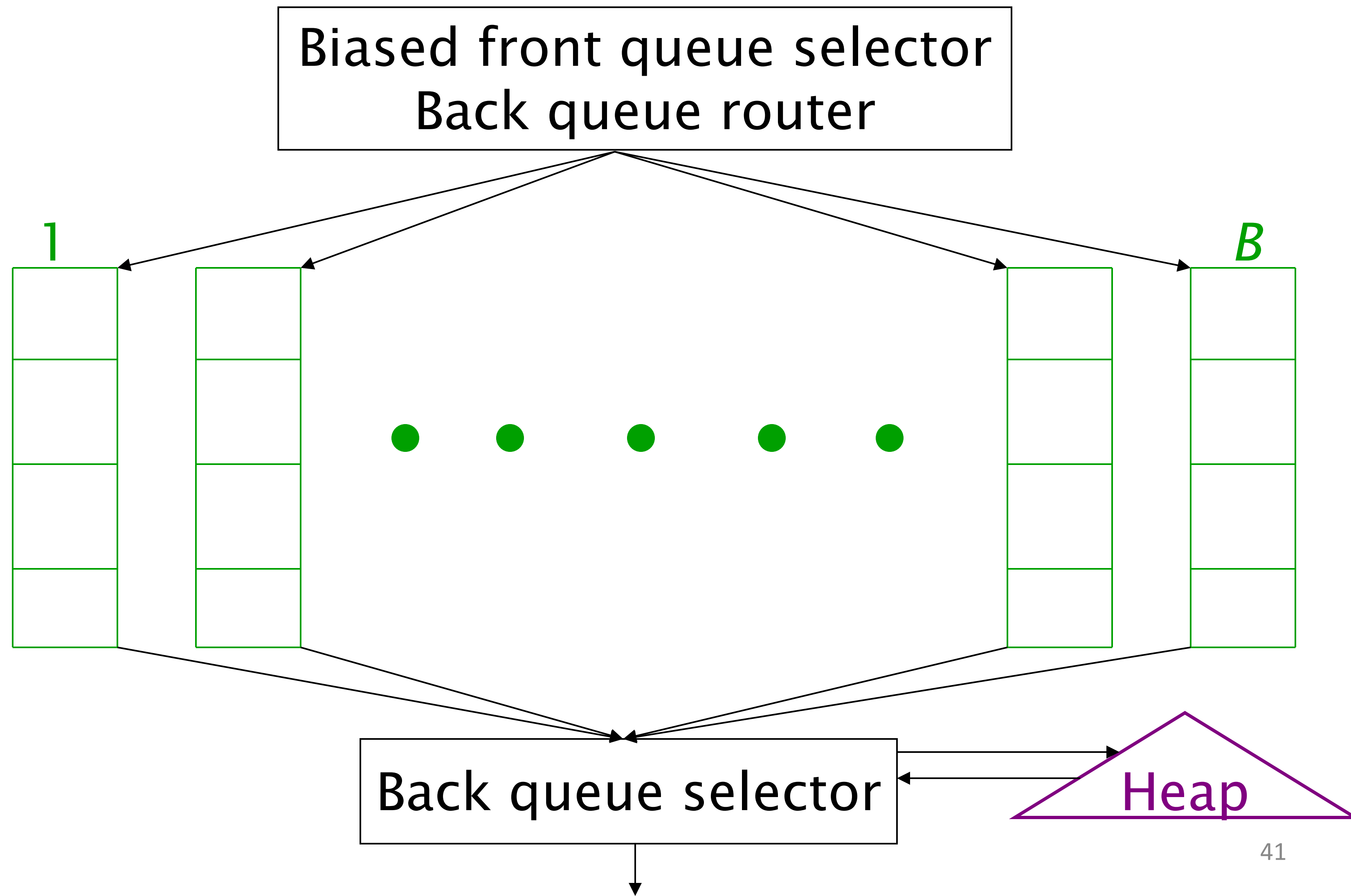
# Front queues

- 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
  - Application-specific (e.g., "crawl news sites more often")

# Biased front queue selector

- When a back queue requests a URL (in a sequence to be described): picks a front queue from which to pull a URL
- This choice can be round robin biased to queues of higher priority, or some more sophisticated variant
  - Can be randomized

# Back queues



# Back queue invariants

- Each back queue is kept non-empty while the crawl is in progress
- Each back queue only contains URLs from a single host
    - Maintain a table from hosts to back queues

| Host name | Back queue |
| --- | --- |
| … | 3 |
| | 1 |
| | *B* |

# Back queue heap

- One entry for each back queue
- The entry is the earliest time $t_e$ at which the host corresponding to the back queue can be hit again
- This earliest time is determined from
  - Last access to that host
  - Any time buffer heuristic we choose

# Back queue processing

- A crawler thread seeking a URL to crawl:
- Extracts the root of the heap
- Fetches URL at head of corresponding back queue *q* (look up from table)
- Checks if queue *q* is now empty – if so, pulls a URL *v* from front queues
  - If there's already a back queue for *v*'s host, append *v* to *q* and pull another URL from front queues, repeat
  - Else add *v* to *q*
- When *q* is non-empty, create heap entry for it

# Number of back queues *B*

- Keep all threads busy while respecting politeness
- Mercator recommendation: three times as many back queues as crawler threads

# Resources

- IIR Chapter 20
- Mercator: A scalable, extensible web crawler (Heydon et al. 1999)
- A standard for robot exclusion