BLG 540E
TEXT RETRIEVAL SYSTEMS

Web Search and Crawling

Arzucan Özgür

Faculty of Computer and Informatics, İstanbul Techical University

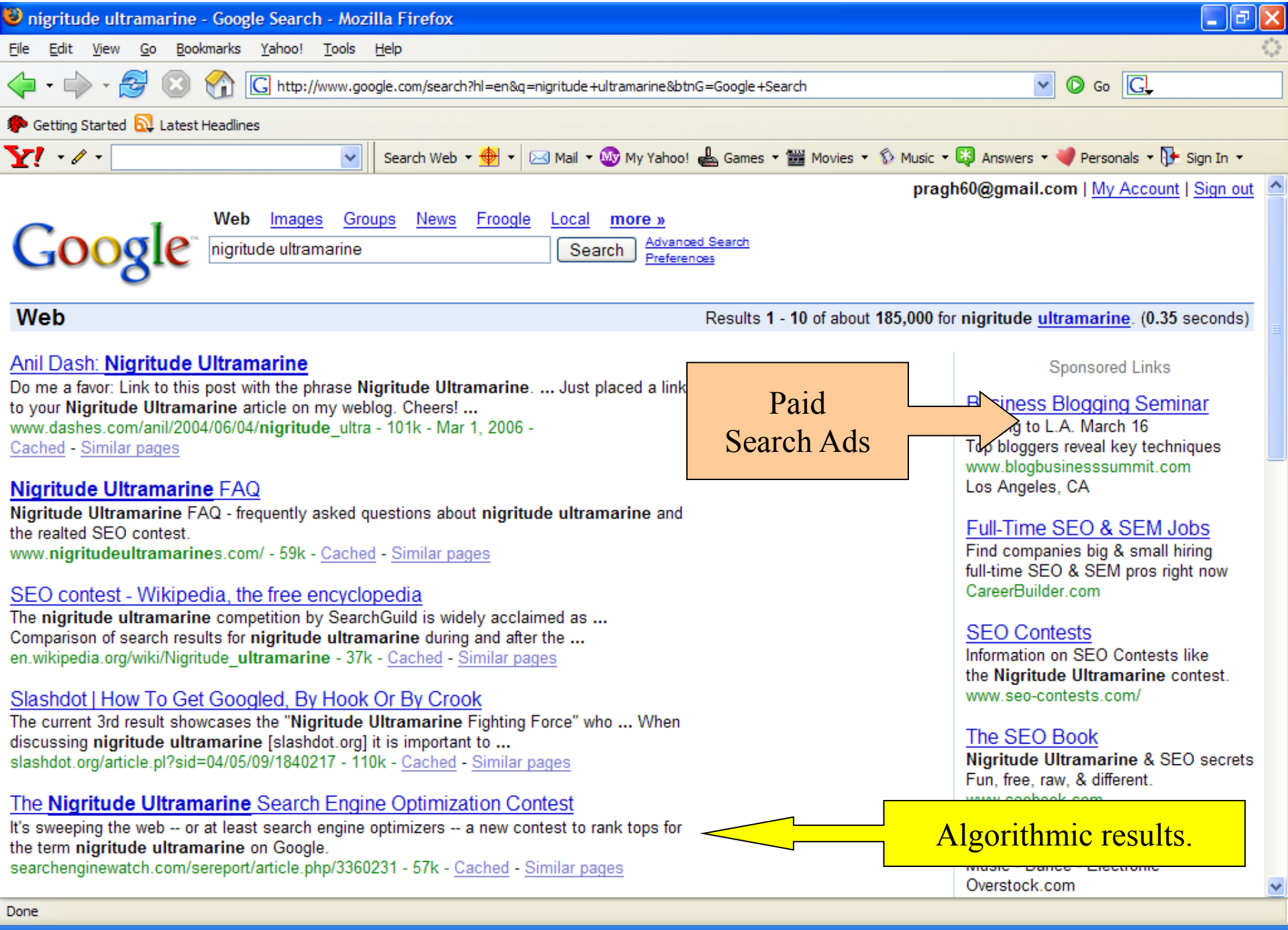
April 22, 2011

Brief (non-technical) history

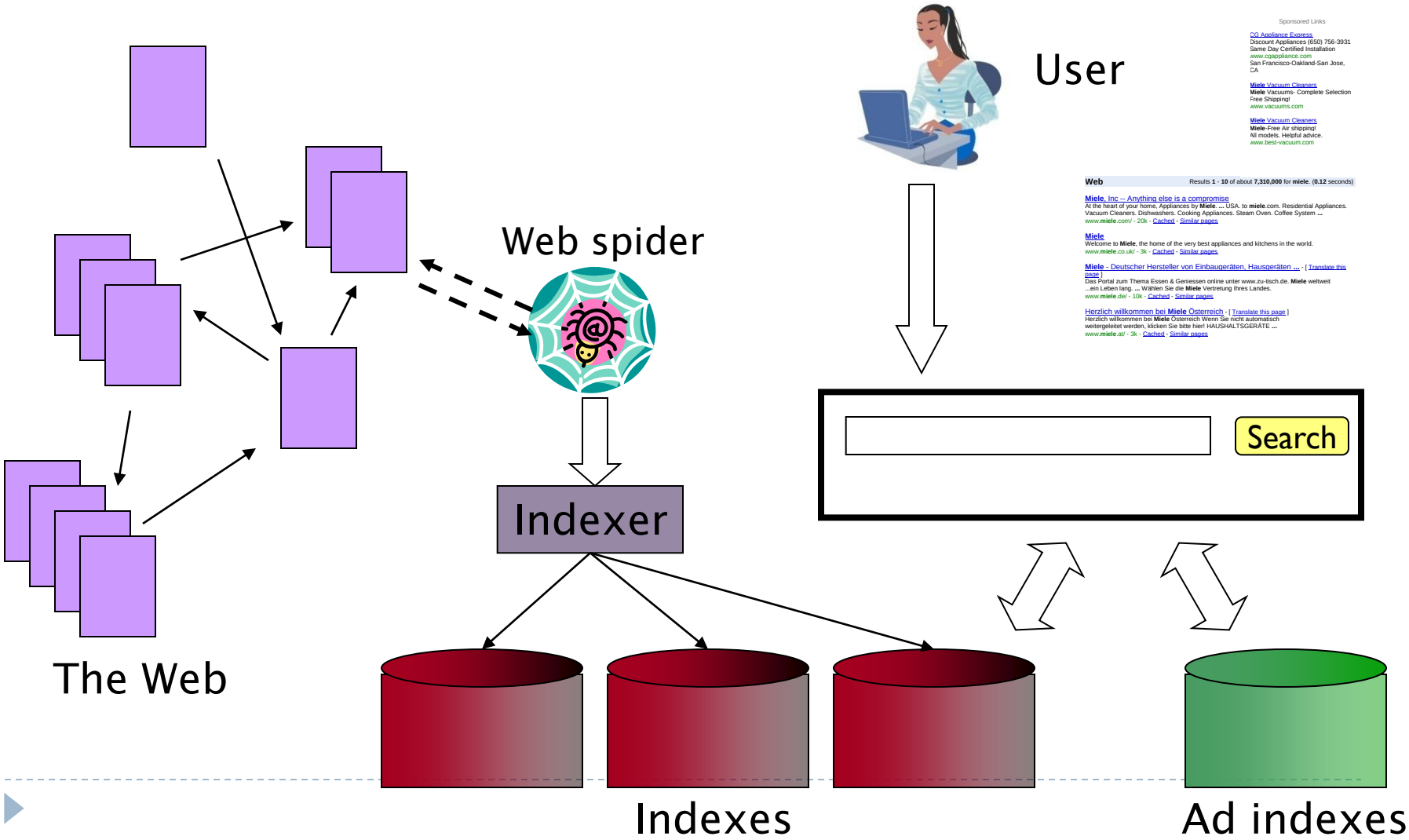
- Early keyword-based engines ca. 1995-1997
  - Altavista, Excite, Infoseek, Inktomi, Lycos
- Paid search ranking: Goto (morphed into Overture.com → Yahoo!)
  - Your search ranking depended on how much you paid

Brief (non-technical) history

- 1998+: Link-based ranking pioneered by Google
  - Blew away all early engines save Inktomi
  - Meanwhile Goto/Overture's annual revenues were nearing $1 billion
- Result: Google added paid search "ads" to the side, independent of search results
  - Yahoo followed suit, acquiring Overture (for paid placement) and Inktomi (for search)


nigritude ultramarine - Google Search - Mozilla Firefox
File Edit View Go Bookmarks Yahoo! Tools Help
http://www.google.com/search?hl=en&q=nigritude+ultramarine&btnG=Google+Search
Go
Getting Started
Latest Headlines
Search Web
Mail
My Yahoo!
Games
Movies
Music
Answers
Personals
Sign In
pragh60@gmail.com | My Account | Sign out
Google
Web Images Groups News Froogle Local more »
nigritude ultramarine
Search
Advanced Search
Preferences
Web
Results 1 - 10 of about 185,000 for nigritude ultramarine. (0.35 seconds)
Anil Dash: Nigritude Ultramarine
Do me a favor: Link to this post with the phrase Nigritude Ultramarine. ... Just placed a link to your Nigritude Ultramarine article on my weblog. Cheers! ...
www.dashes.com/anil/2004/06/04/nigritude_ultra - 101k - Mar 1, 2006 -
Cached - Similar pages
Nigritude Ultramarine FAQ
Nigritude Ultramarine FAQ - frequently asked questions about nigritude ultramarine and the realted SEO contest.
www.nigritudeultramarines.com/ - 59k - Cached - Similar pages
SEO contest - Wikipedia, the free encyclopedia
The nigritude ultramarine competition by SearchGuild is widely acclaimed as ...
Comparison of search results for nigritude ultramarine during and after the ...
en.wikipedia.org/wiki/Nigritude_ultramarine - 37k - Cached - Similar pages
Slashdot | How To Get Googled, By Hook Or By Crook
The current 3rd result showcases the "Nigritude Ultramarine Fighting Force" who ... When discussing nigritude ultramarine [slashdot.org] it is important to ...
slashdot.org/article.pl?sid=04/05/09/1840217 - 110k - Cached - Similar pages
The Nigritude Ultramarine Search Engine Optimization Contest
It's sweeping the web -- or at least search engine optimizers -- a new contest to rank tops for the term nigritude ultramarine on Google.
searchenginewatch.com/sereport/article.php/3360231 - 57k - Cached - Similar pages
Paid Search Ads
Sponsored Links
Business Blogging Seminar
g to L.A. March 16
Top bloggers reveal key techniques
www.blogbusinesssummit.com
Los Angeles, CA
Full-Time SEO & SEM Jobs
Find companies big & small hiring full-time SEO & SEM pros right now
CareerBuilder.com
SEO Contests
Information on SEO Contests like the Nigritude Ultramarine contest.
www.seo-contests.com/
The SEO Book
Nigritude Ultramarine & SEO secrets
Fun, free, raw, & different.
Overstock.com
Algorithmic results.
Done

Web search basics



User Needs

- Need [Brod02, RL04]
  - **Informational** – want to learn about something (~40% / 65%)

    `Low hemoglobin`
  - **Navigational** – want to go to that page (~25% / 15%)

    `United Airlines`
  - **Transactional** – want to do something (web-mediated) (~35% / 20%)
    - Access a service

      `Seattle weather`
    - Downloads

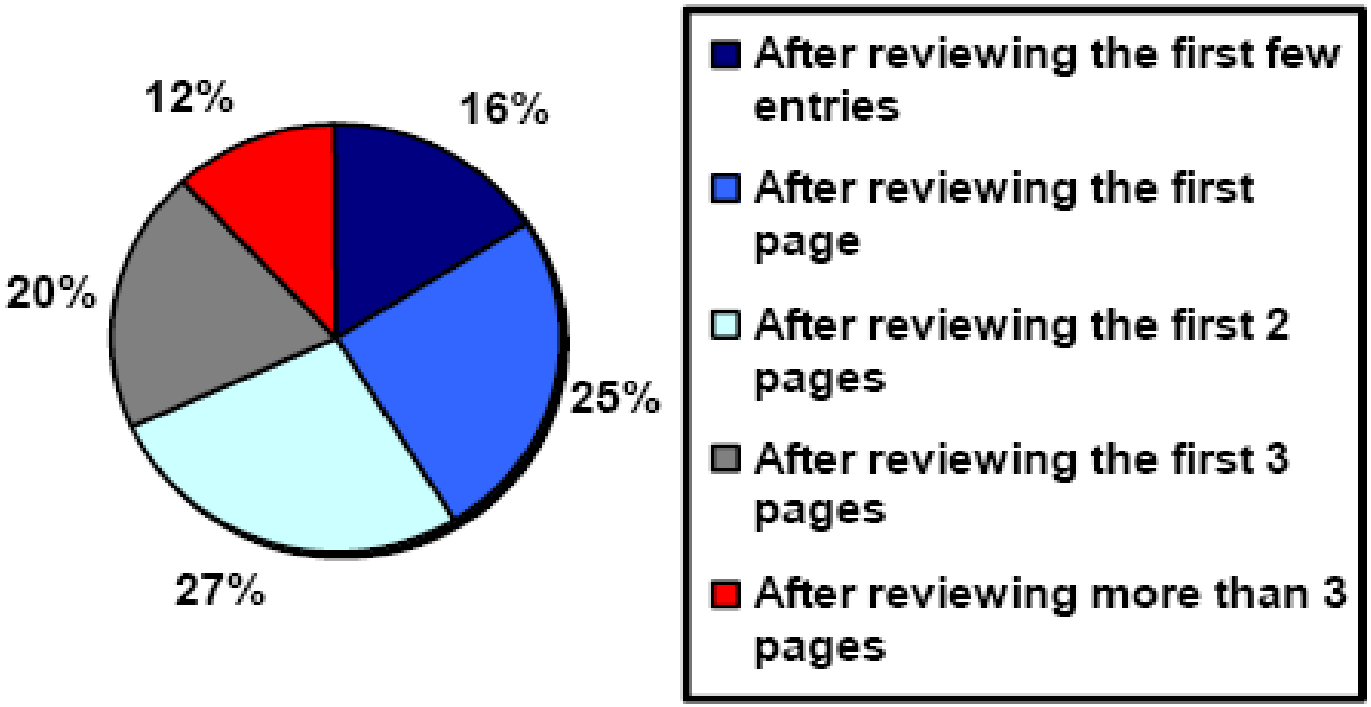
      `Mars surface images`
    - Shop

      `Canon S410`
  - **Gray areas**

    `Car rental Brasil`
    - Find a good hub
    - Exploratory search "see what's there"

How far do people look for results?

"When you perform a search on a search engine and don't find what you are looking for, at what point do you typically either revise your search, or move on to another search engine? (Select one)"



(Source: iprospect.com WhitePaper_2006_SearchEngineUserBehavior.pdf)

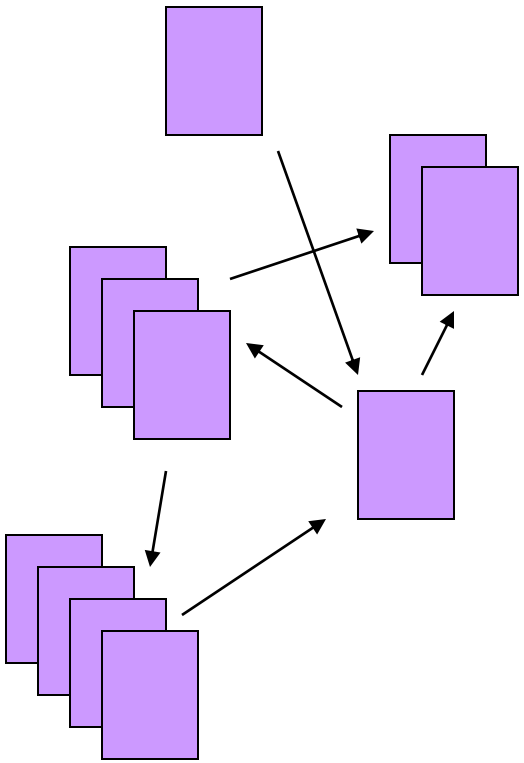
Users' empirical evaluation of results

- Quality of pages varies widely
  - Relevance is not enough
  - Other desirable qualities (non IR!!)
    - Content: Trustworthy, diverse, non-duplicated, well maintained
    - Web readability: display correctly & fast
    - No annoyances: pop-ups, etc
- Precision vs. recall
  - On the web, recall seldom matters
- What matters
  - Precision at 1? Precision above the fold?
  - Comprehensiveness – must be able to deal with obscure queries
    - Recall matters when the number of matches is very small

Users' empirical evaluation of engines

- Relevance and validity of results
- UI – Simple, no clutter, error tolerant
- Trust – Results are objective
- Coverage of topics for polysemic queries
- Pre/Post process tools provided
  - Mitigate user errors (auto spell check, search assist,…)
  - Explicit: Search within results, more like this, refine ...
  - Anticipative: related searches
- Deal with idiosyncrasies
  - Web specific vocabulary
    - Impact on stemming, spell-check, etc
  - Web addresses typed in the search box

The Web document collection



- No design/co-ordination
- Distributed content creation, linking, democratization of publishing
- Content includes truth, lies, obsolete information, contradictions …
- Unstructured (text, html, …), semi-structured (XML, annotated photos), structured (Databases)…
- Scale much larger than previous text collections …
- Growth – slowed down from initial "volume doubling every few months" but still expanding
- Content can be *dynamically generated*

Spam

- (Search Engine Optimization)

The trouble with paid search ads …

- It costs money. What's the alternative?
- *Search Engine Optimization:*
  - "Tuning" your web page to rank highly in the algorithmic search results for select keywords
  - Alternative to paying for placement
  - Thus, intrinsically a marketing function
- Performed by companies, webmasters and consultants ("Search engine optimizers") for their clients
- Some perfectly legitimate, some very shady

Simplest forms

- First generation engines relied heavily on *tf/idf*
  - The top-ranked pages for the query **`maui resort`** were the ones containing the most **`maui`**'s and **`resort`**'s
- SEOs responded with dense repetitions of chosen terms
  - e.g., **`maui resort maui resort maui resort`**
  - Often, the repetitions would be in the same color as the background of the web page
    - Repeated terms got indexed by crawlers
    - But not visible to humans on browsers

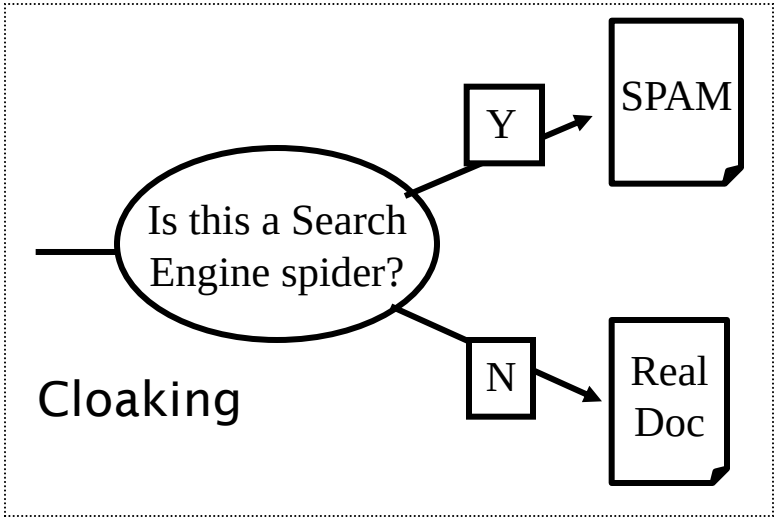
Variants of keyword stuffing

- Misleading meta-tags, excessive repetition
- Hidden text with colors, style sheet tricks, etc.

Meta-Tags =
"… London hotels, hotel, holiday inn, hilton, discount, booking, reservation, sex, mp3, britney spears, viagra, …"

Cloaking

- Serve fake content to search engine spider



More spam techniques

- **Doorway pages**
  - Pages optimized for a single keyword that re-direct to the real target page
- **Link spamming**
  - Mutual admiration societies, hidden links, link farms
  - *Domain flooding:* numerous domains that point or re-direct to a target page

The war against spam

- Quality signals - Prefer authoritative pages based on:
  - Votes from authors (linkage signals)
  - Votes from users (usage signals)
- Policing of URL submissions
  - Anti robot test
- Limits on meta-keywords
- Robust link analysis
  - Ignore statistically implausible linkage (or text)
  - Use link analysis to detect spammers (guilt by association)
- Spam recognition by machine learning
  - Training set based on known spam
- Editorial intervention
  - Blacklists
  - Top queries audited
  - Complaints addressed
  - Suspect pattern detection

More on spam

- Adversarial IR: the unending (technical) battle between SEO's and web search engines
- Research http://airweb.cse.lehigh.edu/

Size of the Web

- The Web is the largest repository of data and it grows exponentially.
  - 320 Million Web pages [Lawrence & Giles 1998]
  - 800 Million Web pages, 15 TB [Lawrence & Giles 1999]
  - 20 Billion Web pages indexed [now]
- Amount of data
  - roughly 200 TB [Lyman et al. 2003]

Size of the web

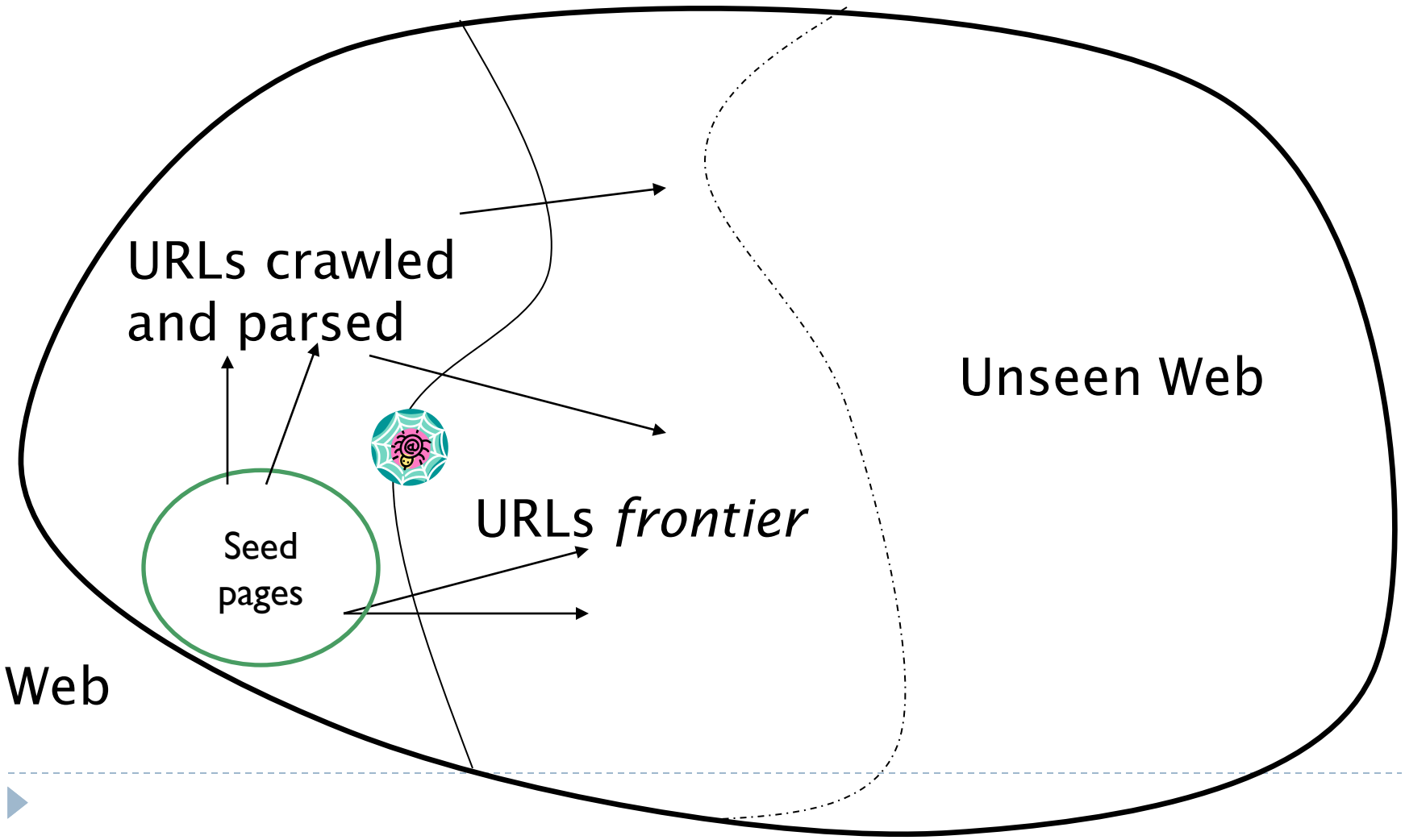
- Issues
  - The web is really infinite
    - Dynamic content, e.g., calendar
    - Soft 404: www.yahoo.com/<anything> is a valid page,
    - Infinite sized – size is whatever can be indexed!
  - Static web contains syntactic duplication, mostly due to mirroring (~30%)

Web 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

Crawling picture



Simple picture – complications

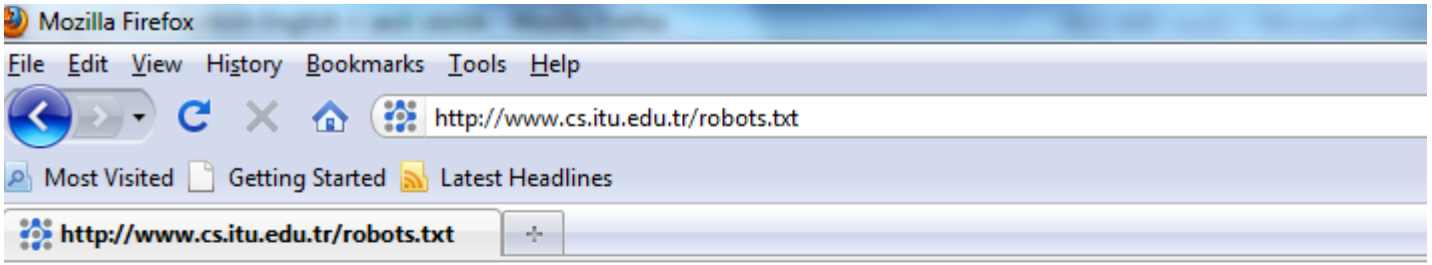
- Web crawling isn't feasible with one machine
  - All of the above steps distributed
- Malicious pages
  - Spam pages
  - Spider traps
  - Even non-malicious pages pose challenges
  - Latency/bandwidth to remote servers vary
  - Webmasters' stipulations
    - How "deep" should you crawl a site's URL 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

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




```
# Define access-restrictions for robots/spiders
# http://www.robotstxt.org/wc/norobots.html



# By default we allow robots to access all areas of our site
# already accessible to anonymous users

User-agent: *
Disallow:



# Add Googlebot-specific syntax extension to exclude forms
# that are repeated for each piece of content in the site
# the wildcard is only supported by Googlebot
# http://www.google.com/support/webmasters/bin/answer.py?answer=40367&ctx=sibling

User-Agent: Googlebot
Disallow: /*sendto_form$
Disallow: /*folder_factories$
```

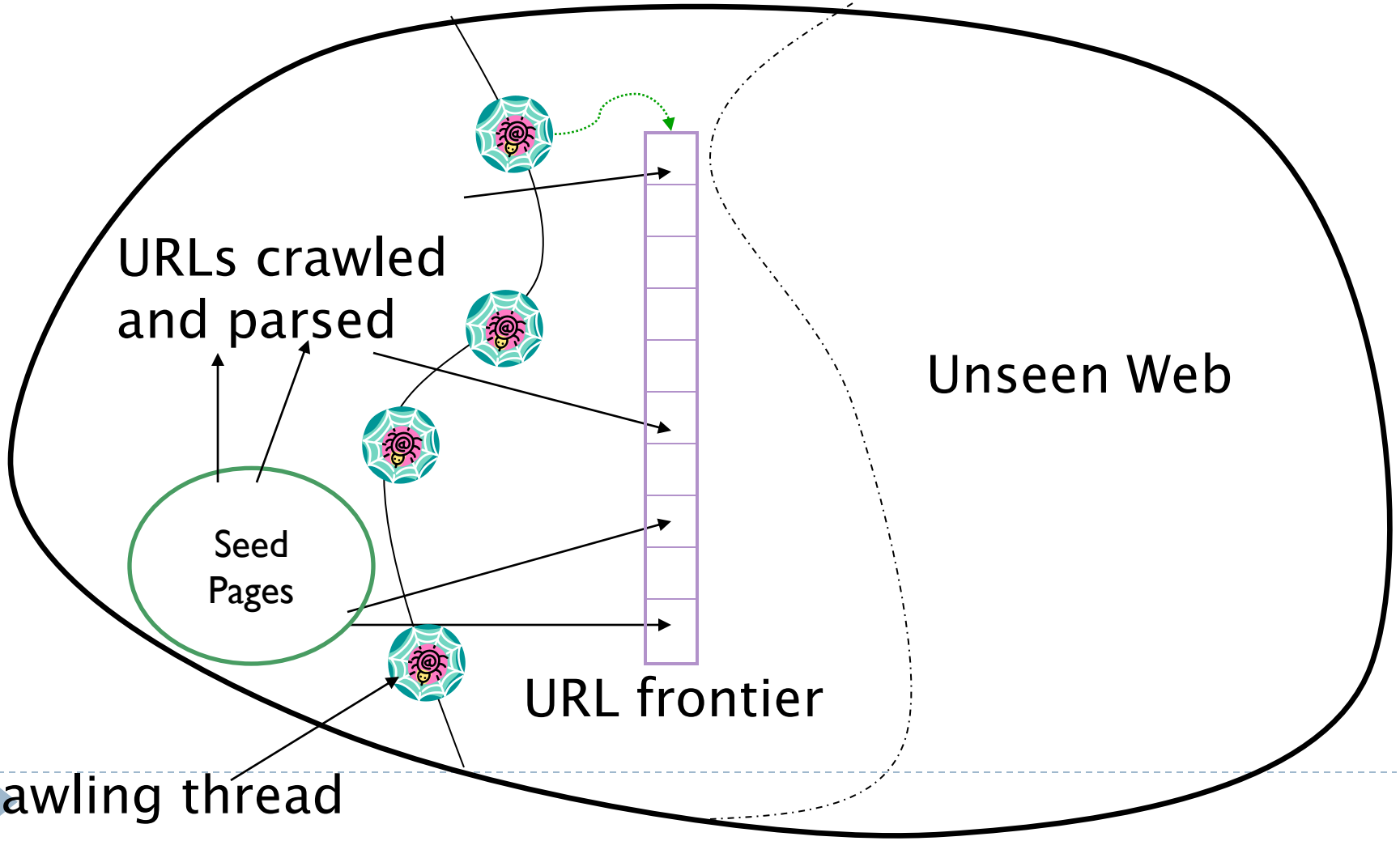
What any crawler *should* do

- 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

Updated crawling picture



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

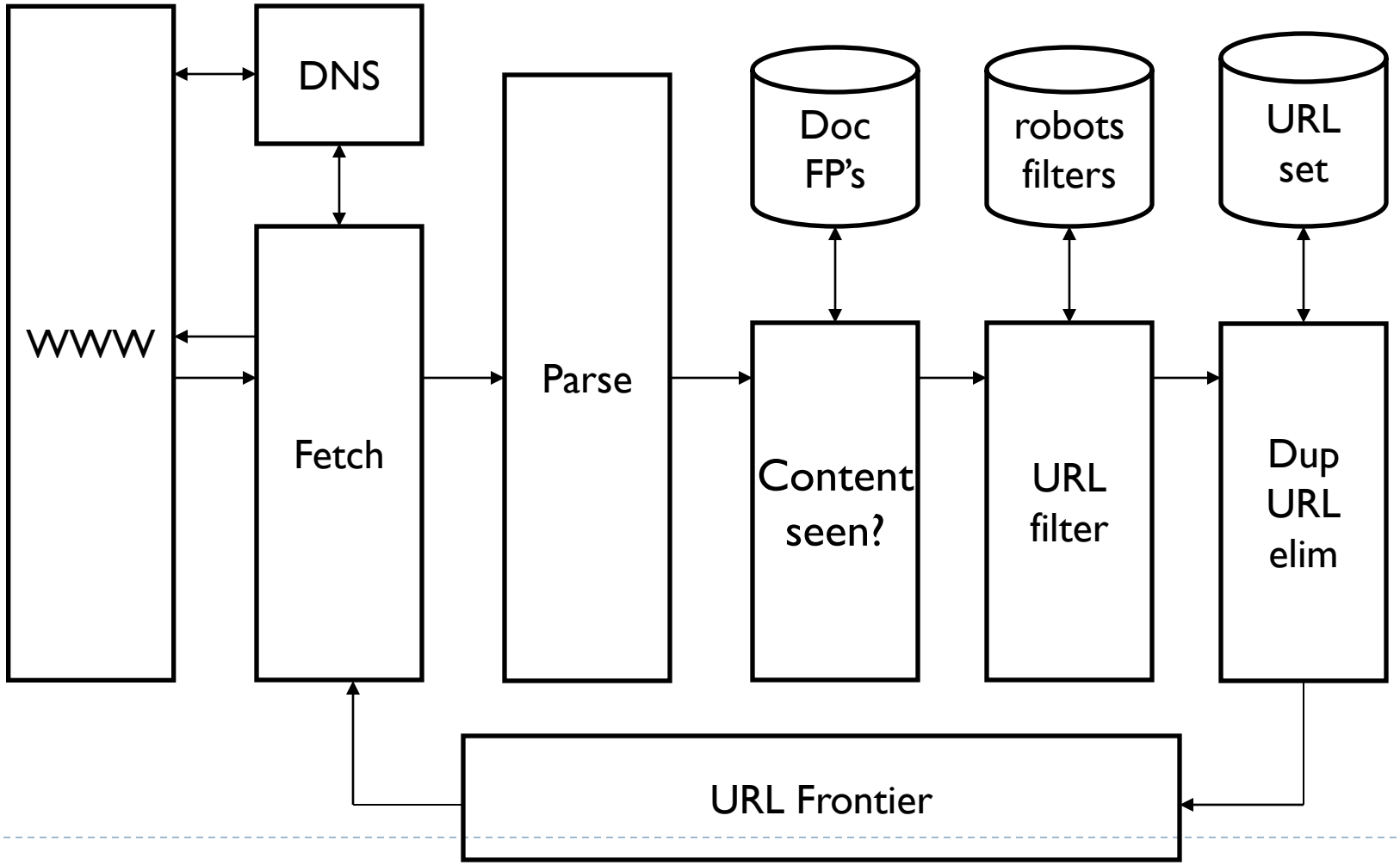
Processing steps in crawling

- 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



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
  - (see Serdar Bağış's presentation)

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

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

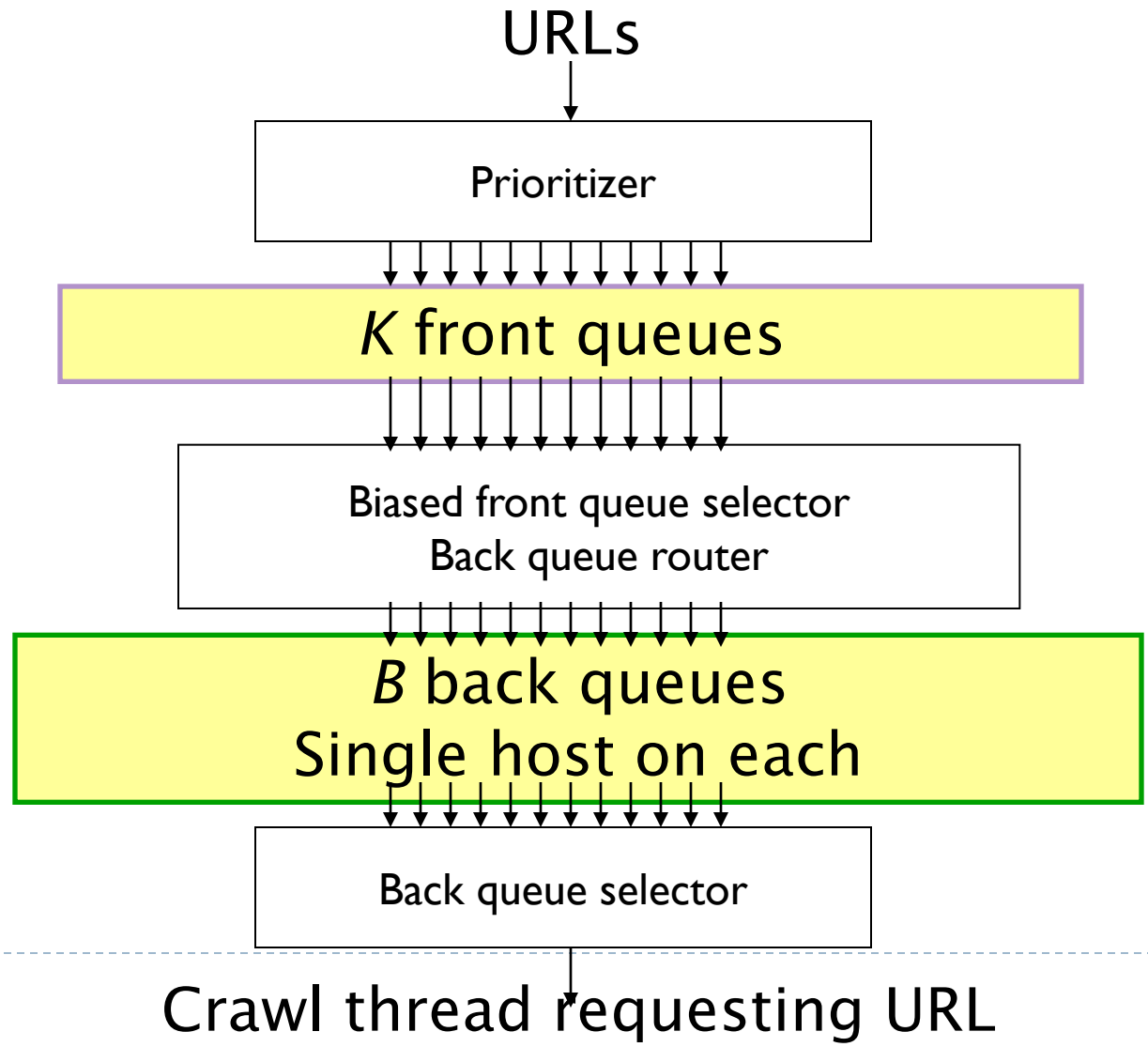
These goals may conflict each other.

(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

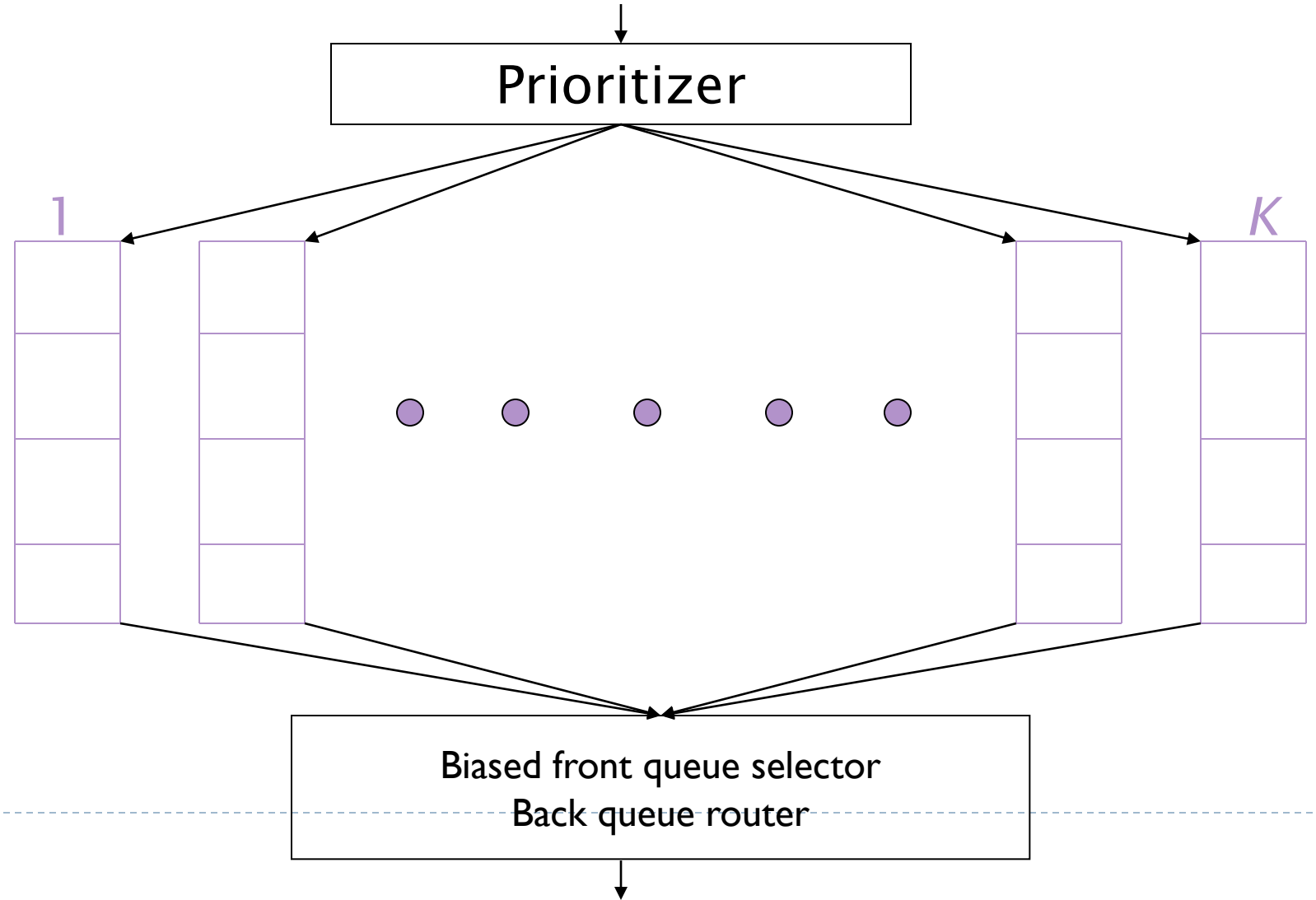
URL frontier: Mercator scheme



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



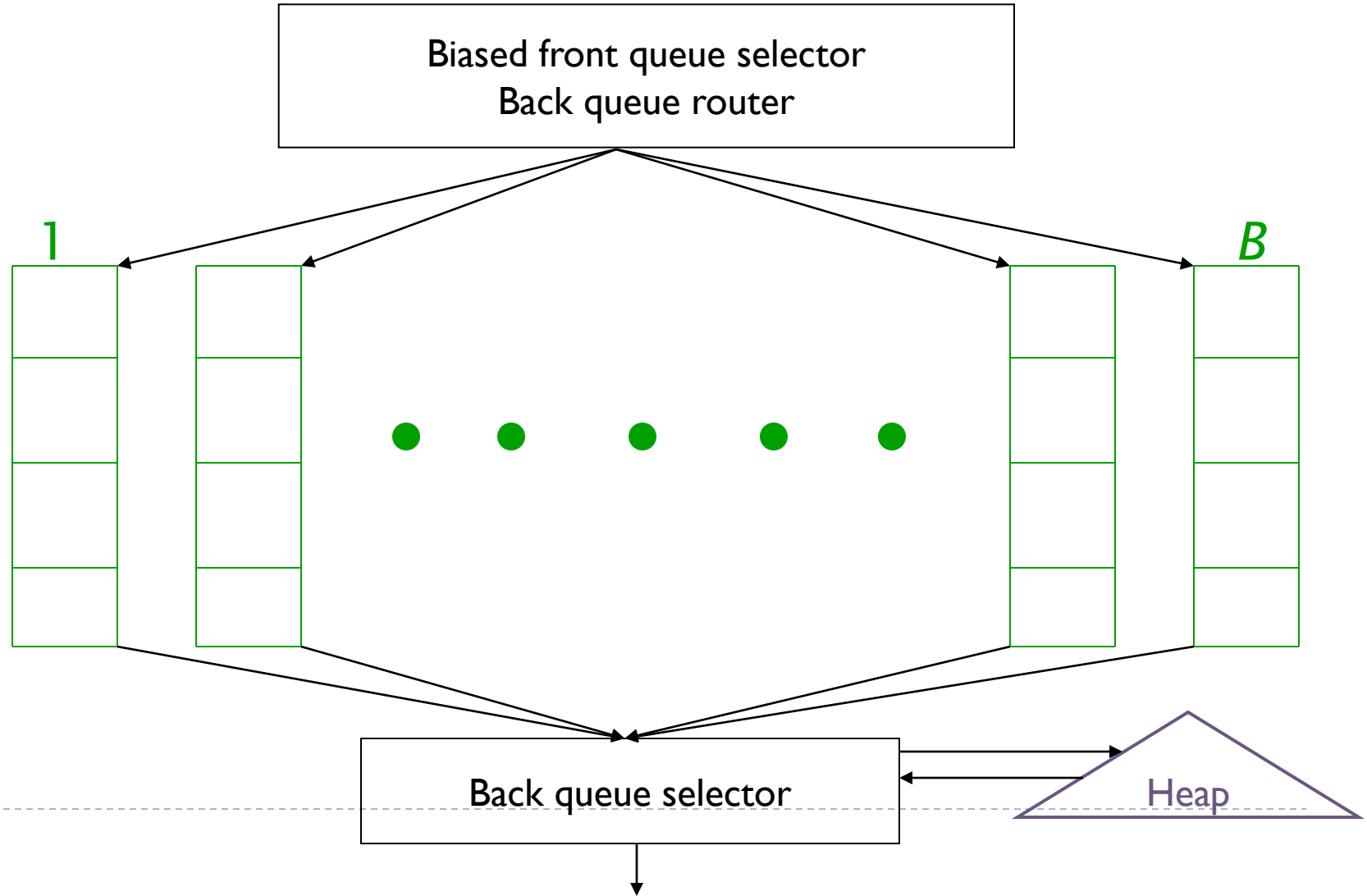
Front queues

- 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


Biased front queue selector
Back queue router
1
B
Back queue selector
Heap

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

Resources

- *Introduction to Information Retrieval*, chapters 19,20.
- Some slides were adapted from
  - Prof. Dragomir Radev's lectures at the University of Michigan:
    - http://clair.si.umich.edu/~radev/teaching.html
  - the book's companion website:
    - http://nlp.stanford.edu/IR-book/information-retrieval-book.html