

Monsoon 2020

19 - Web Crawling

I n f o r m a t i o n

R e t r i e v a l

by

Dr. Rajendra Prasath



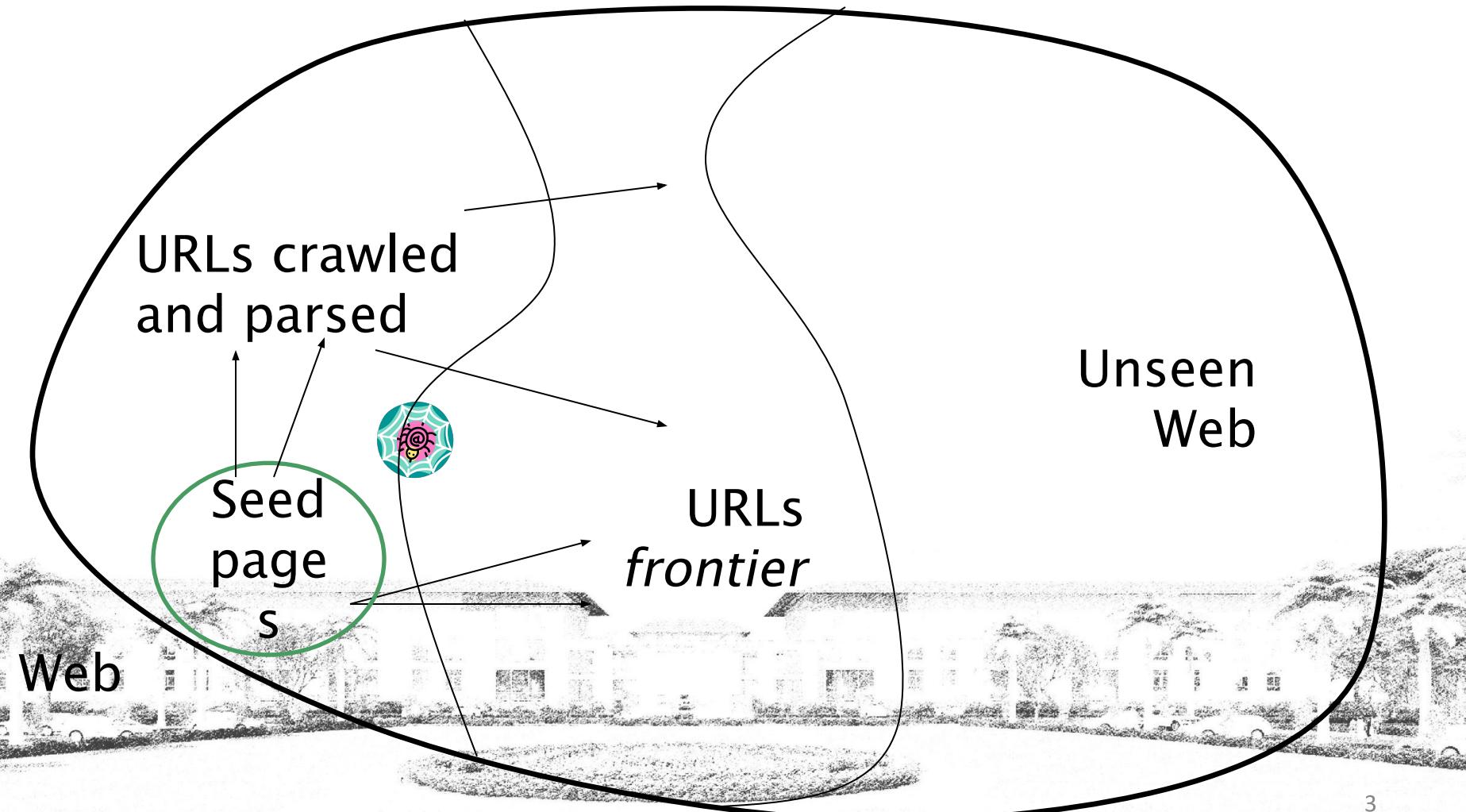
**Indian Institute of Information Technology, Sri City, Chittoor
Sri City – 517 646, Andhra Pradesh, India**

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 – 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 Robust: Be immune to spider traps and other malicious behavior from web servers
- Be Polite: Respect implicit and explicit politeness considerations



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/robotstxt.html
- Website announces its request on what can(not) be crawled
 - For a server, create a file / 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:

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

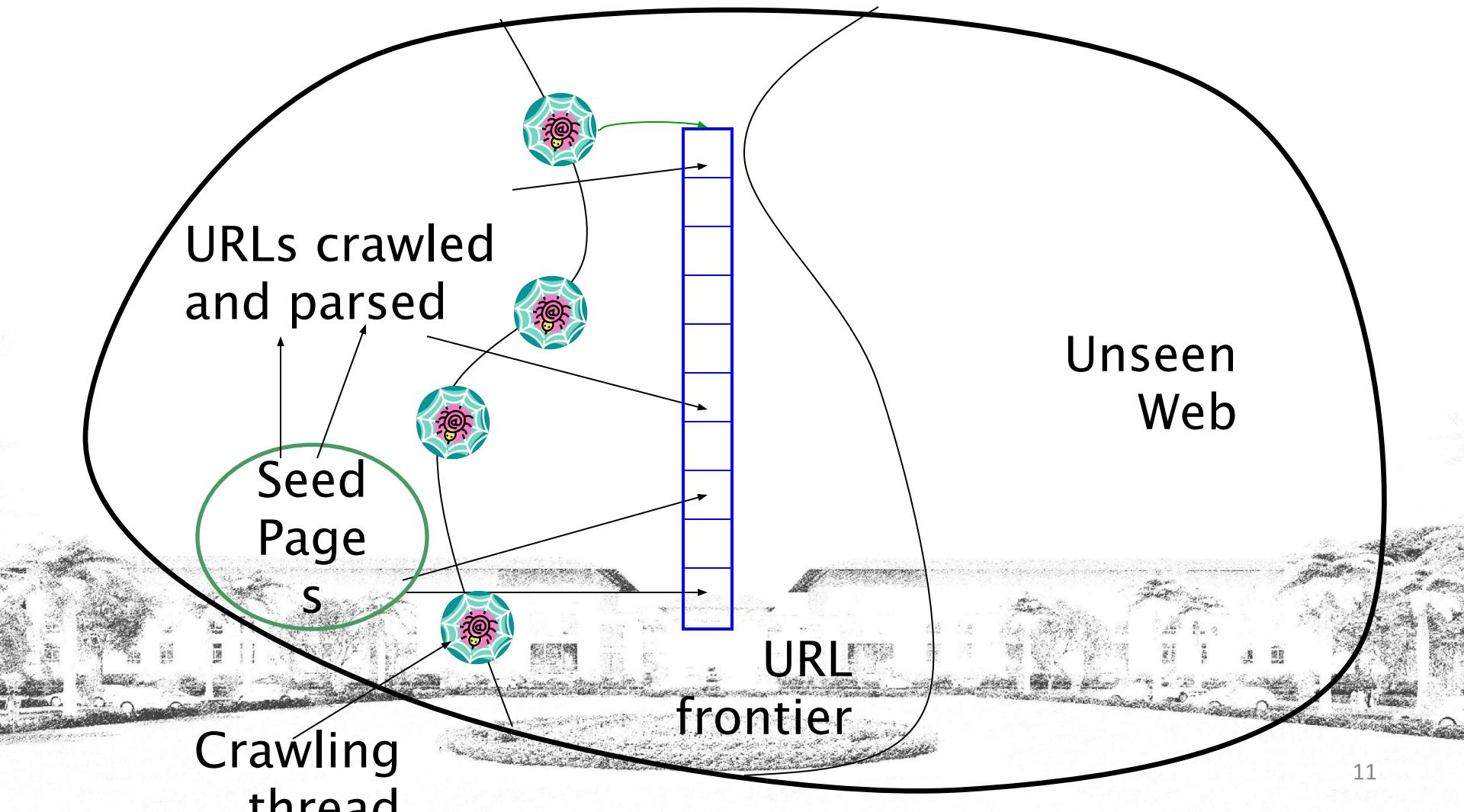


What any crawler should do

- 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



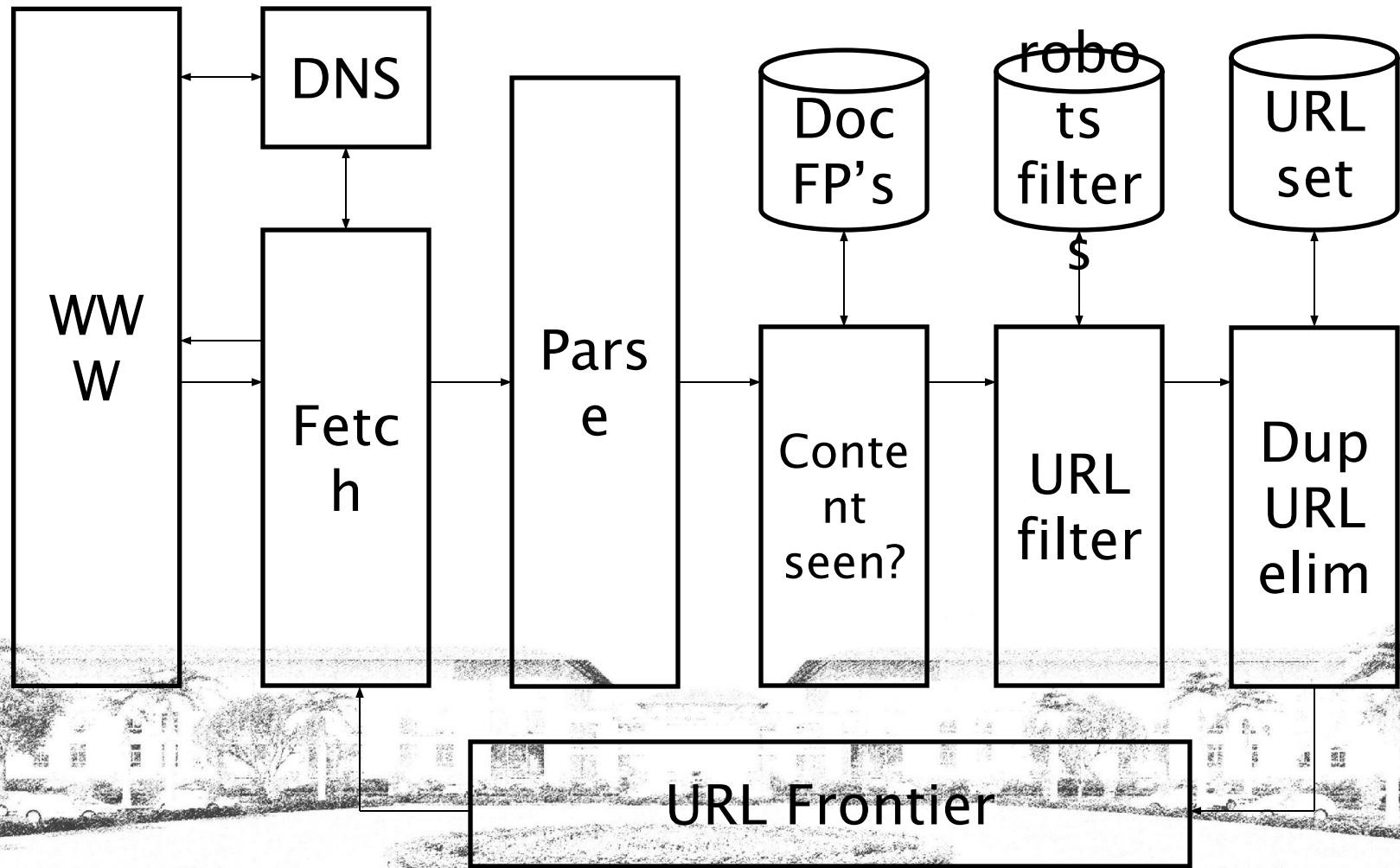
Processing steps in crawling

- 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., http://en.wikipedia.org/wiki/Main_Page has 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
 - Second part of this lecture

Filters and robots.txt

- Filters – regular expressions for URLs 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



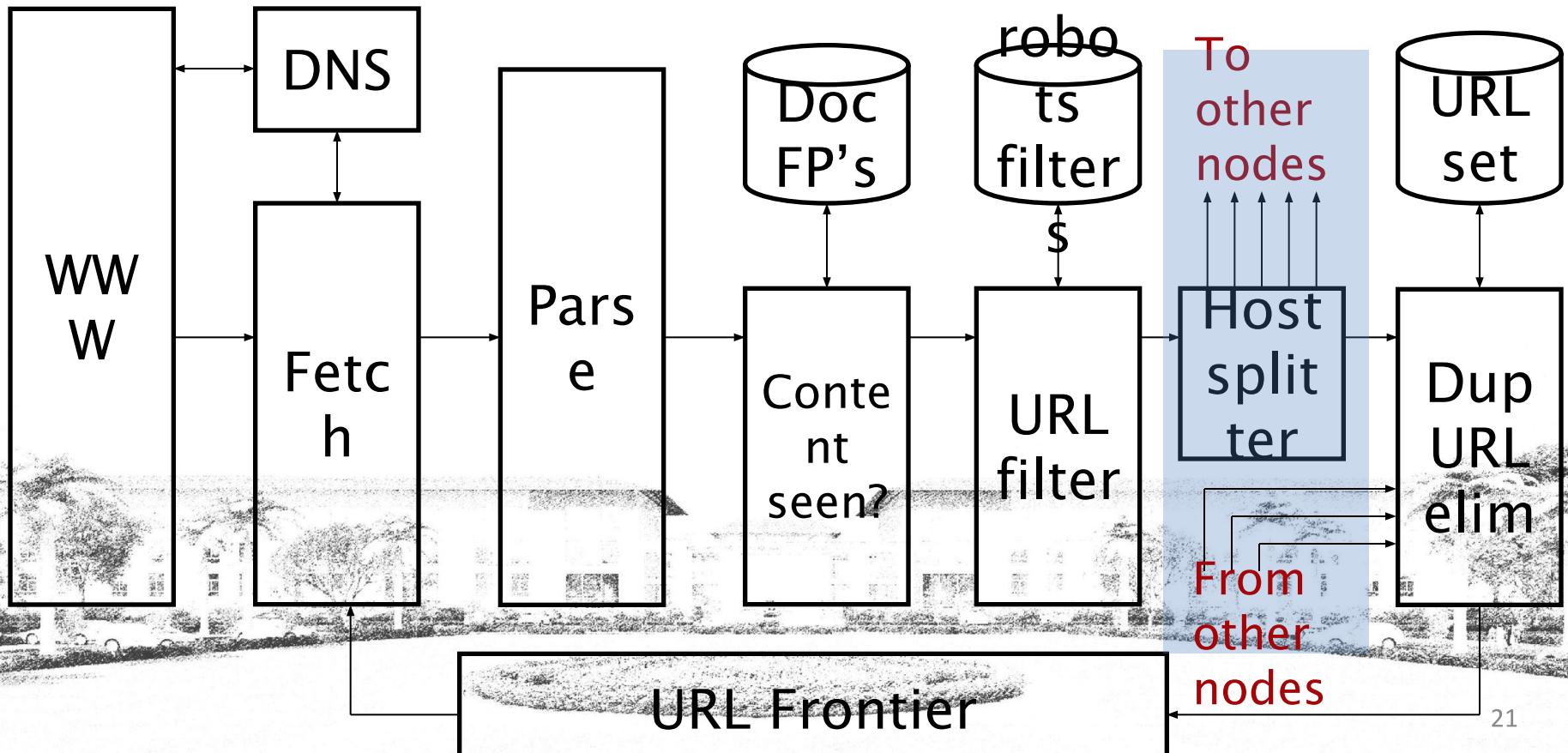
Distributing the crawler

- 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 and share URLs?



Communication between nodes

- Output of the URL filter at each node is sent to the Dup URL Eliminator of the appropriate node



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 with each other.

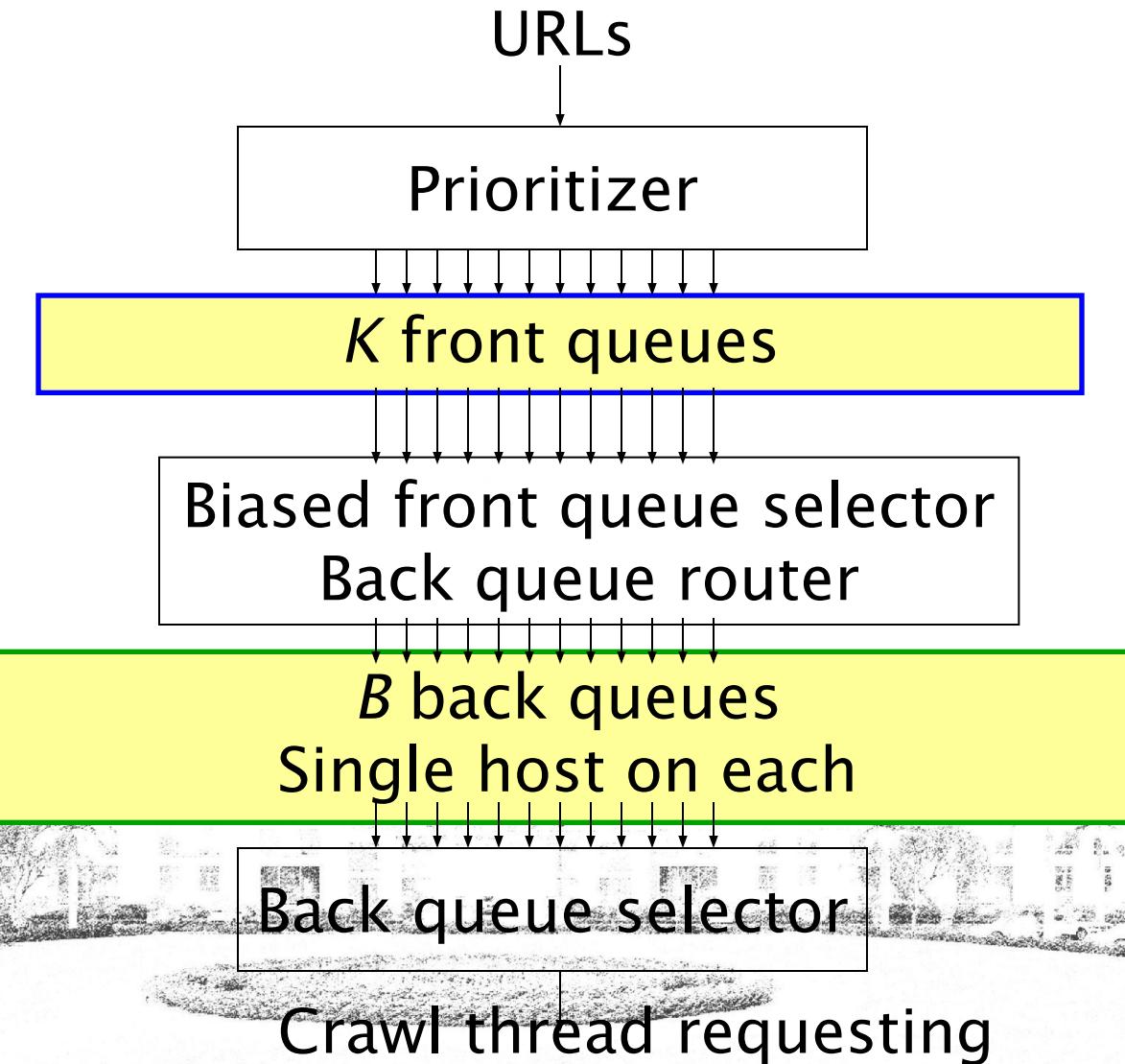
(E.g., simple priority queue fails – many links out of a page go to its own site, creating a burst of accesses to that site.)

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 \gg time for most recent fetch from that host



URL frontier: Mercator scheme

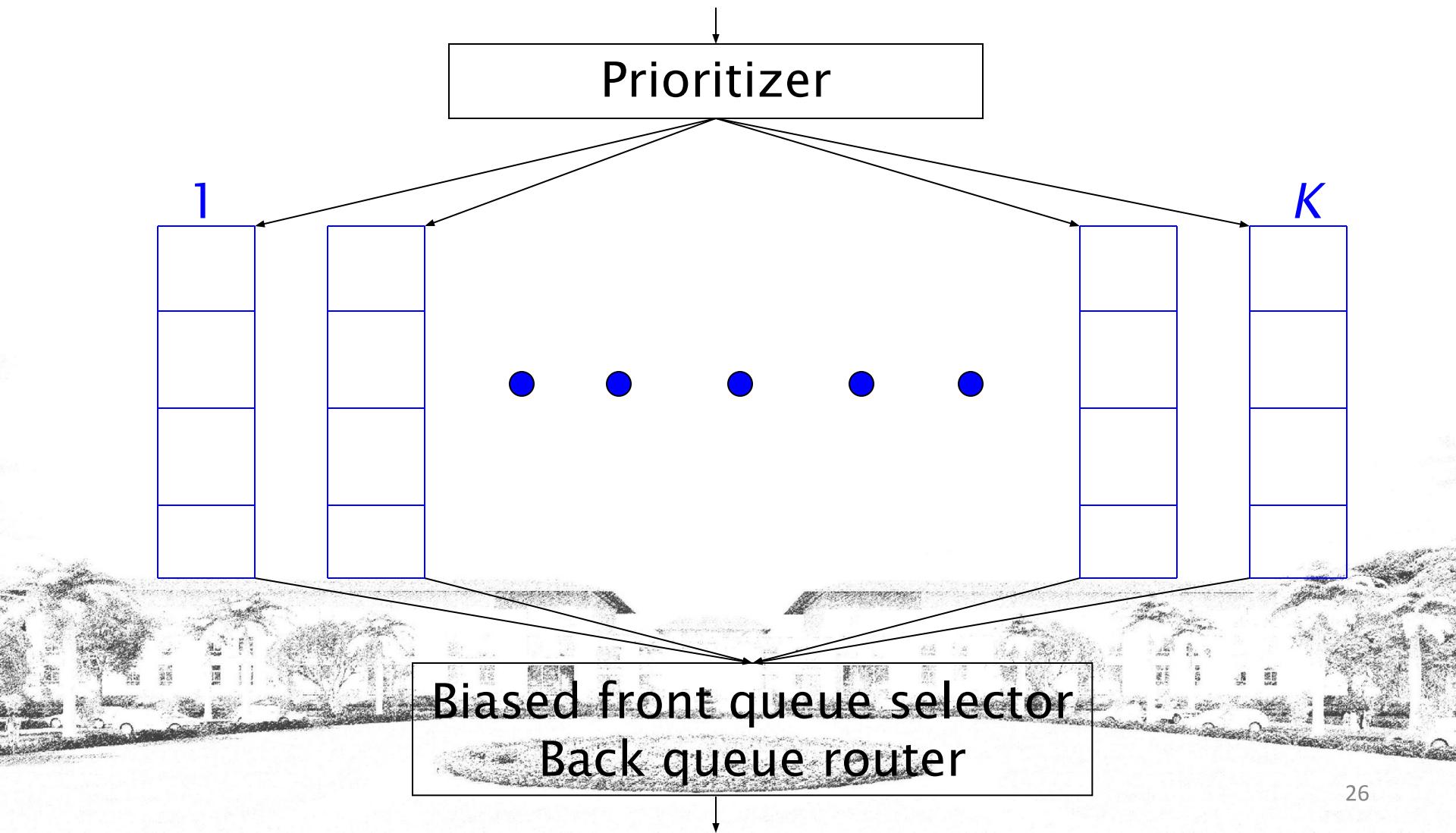


Mercator URL frontier

- 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”)

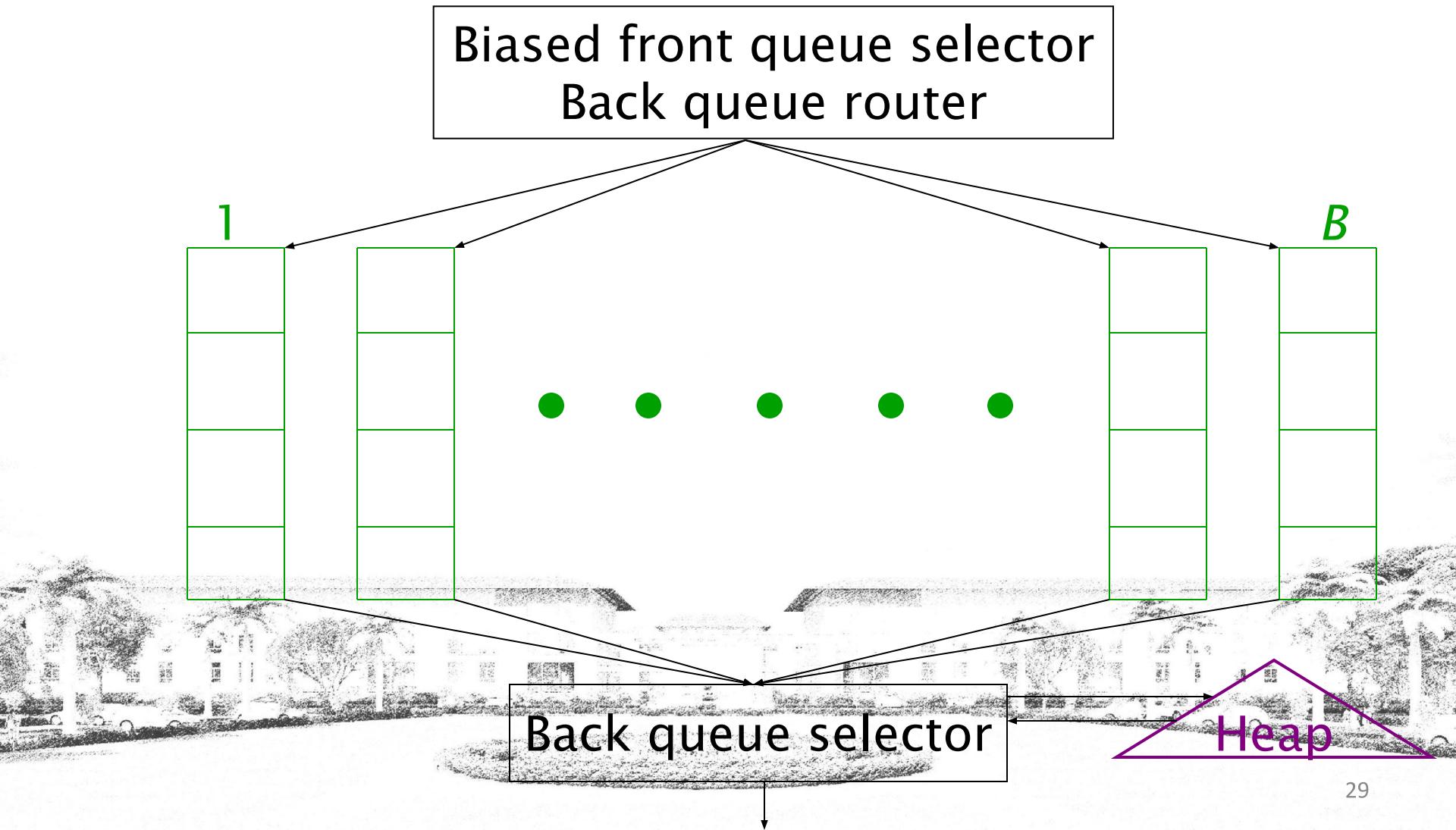


Biased front queue selector

- 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 it 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



Summary

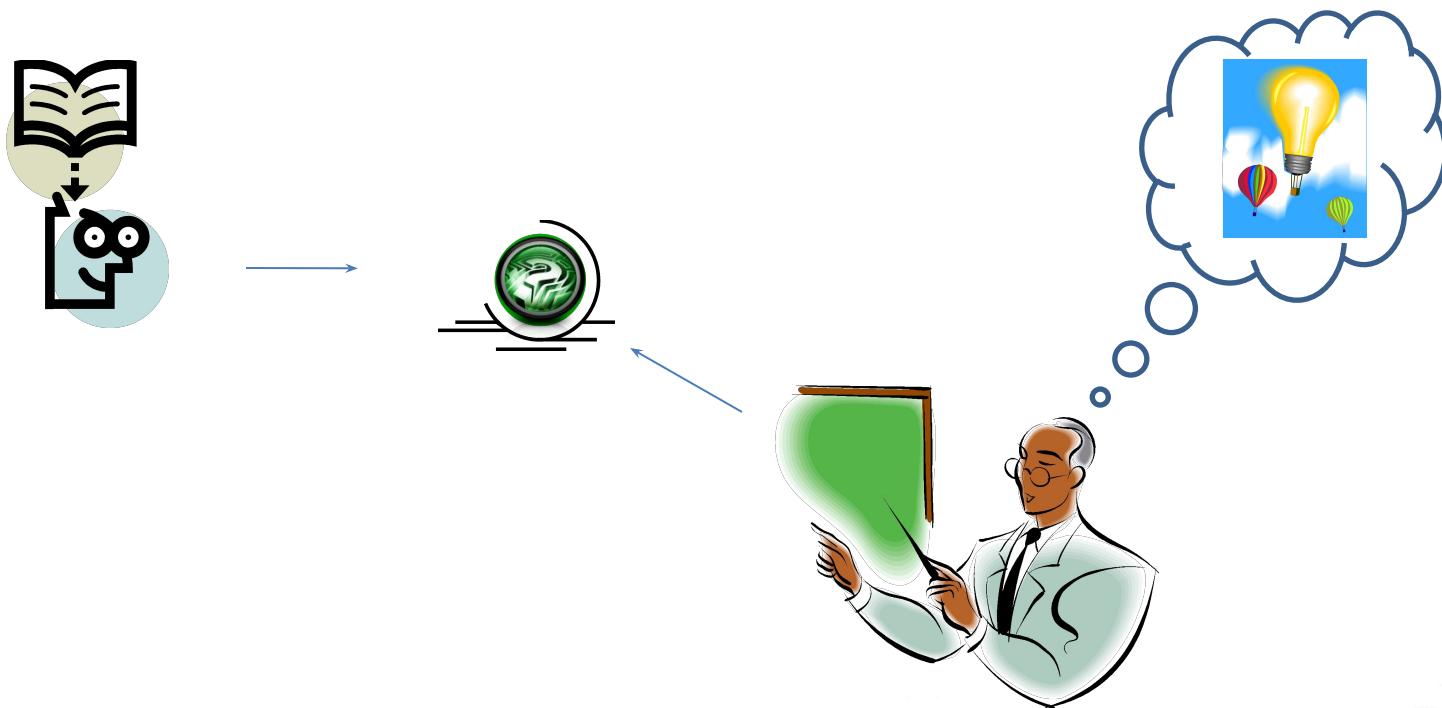
- ❖ Web Crawling
 - ❖ Crawling Architecture
 - ❖ Fetching
 - ❖ Distributed Crawler
 - ❖ Parsing
 - ❖ Indexing
 - ❖ Ranking

Acknowledgements

Thanks to all IR Researchers:

1. Introduction to Information Retrieval Manning, Raghavan and Schutze, Cambridge University Press, 2008.
2. Search Engines Information Retrieval in Practice W. Bruce Croft, D. Metzler, T. Strohman, Pearson, 2009.
3. Information Retrieval Implementing and Evaluating Search Engines Stefan Büttcher, Charles L. A. Clarke and Gordon V. Cormack, MIT Press, 2010.
4. Modern Information Retrieval Baeza-Yates and Ribeiro-Neto, Addison Wesley, 1999.
5. Many Authors who contributed to SIGIR / WWW / KDD / ECIR / CIKM / WSDM and other top tier conferences
6. Prof. Mandar Mitra, Indian Statistical Institute, Kolkata (<https://www.isical.ac.in/~mandar/>) for sharing the IR Evaluation Slides

Thanks ...



... Questions ???