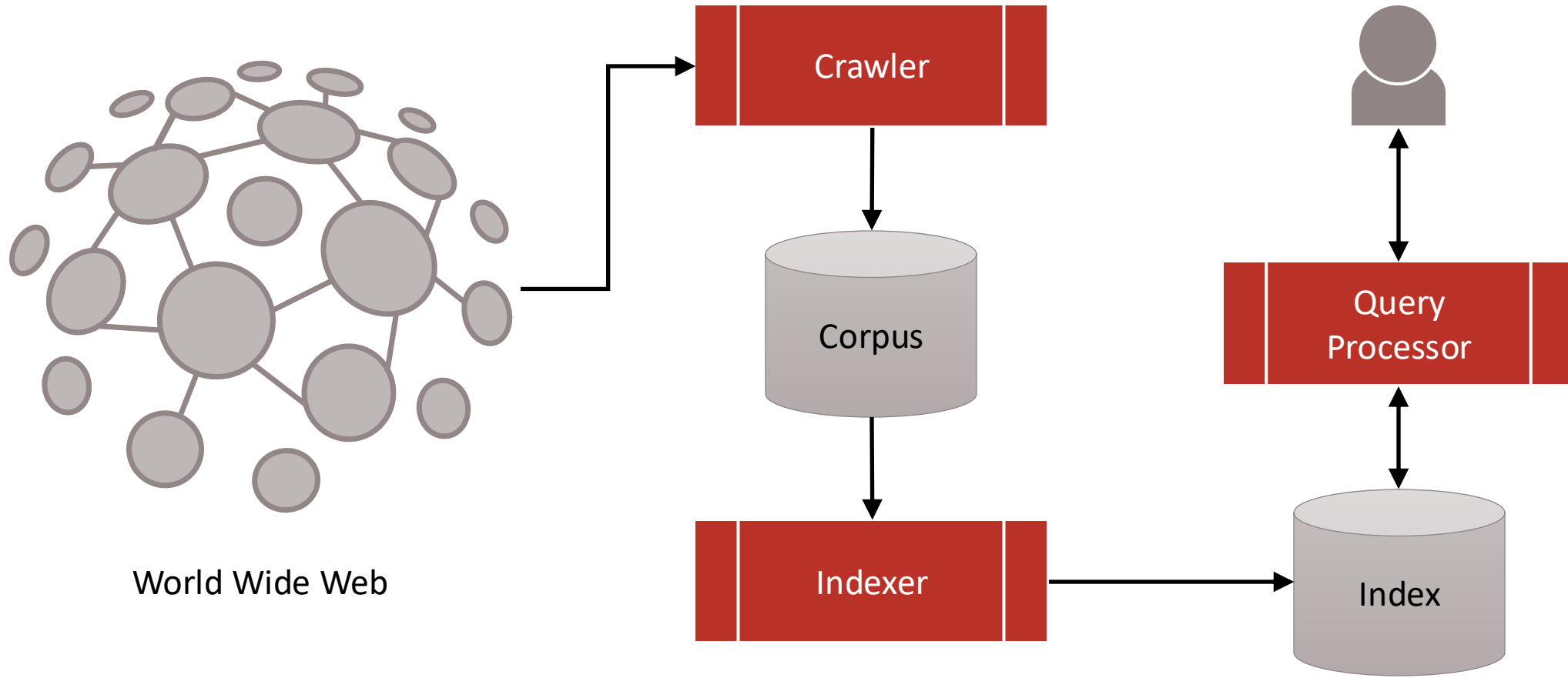


Information Retrieval

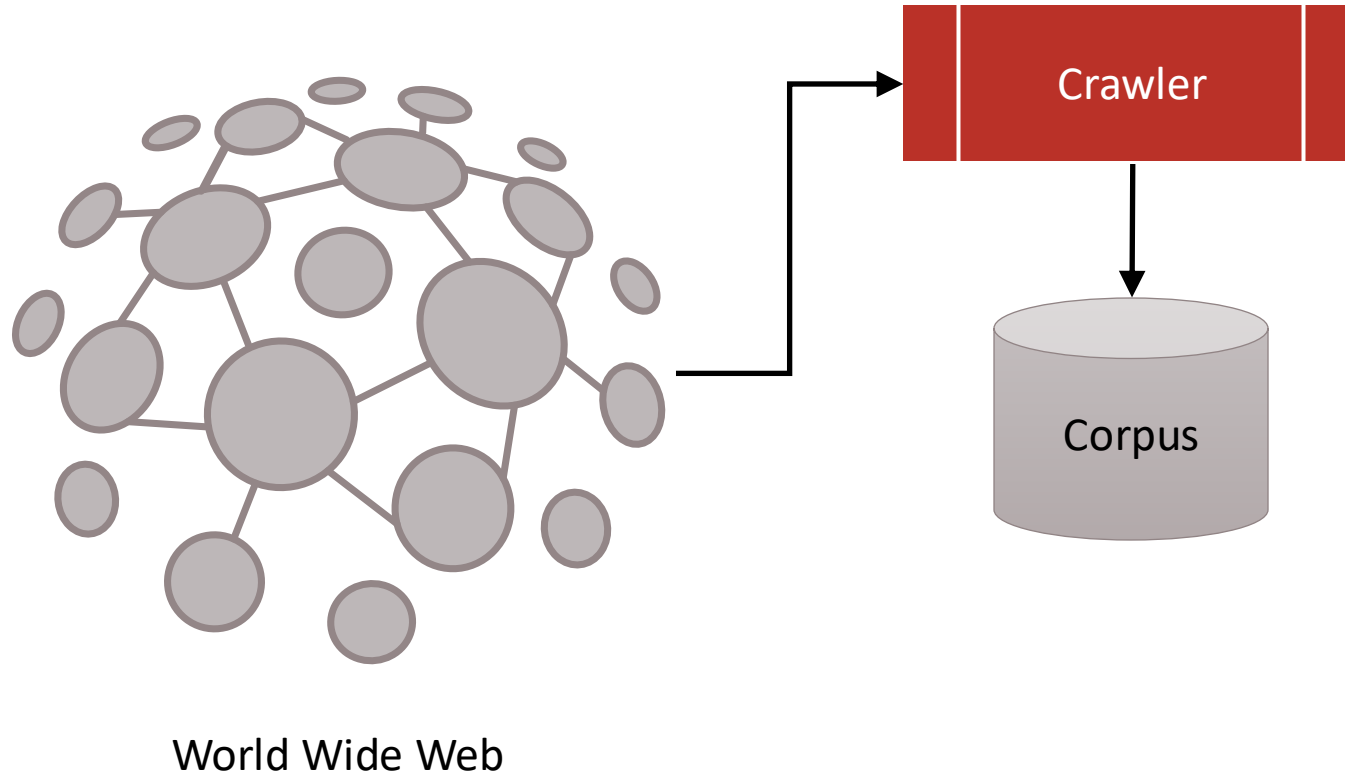
Web Crawling

Rodrygo L. T. Santos
rodrygo@dcc.ufmg.br

Search components



Search components



Web crawling

Web crawling is the process of locating, fetching, storing, and maintaining pages available in the Web

- Computer programs that perform this task are referred to as *crawlers, spiders, harvesters*

Web crawler repositories as a web cache

- Help speed up the indexing process

Success measures

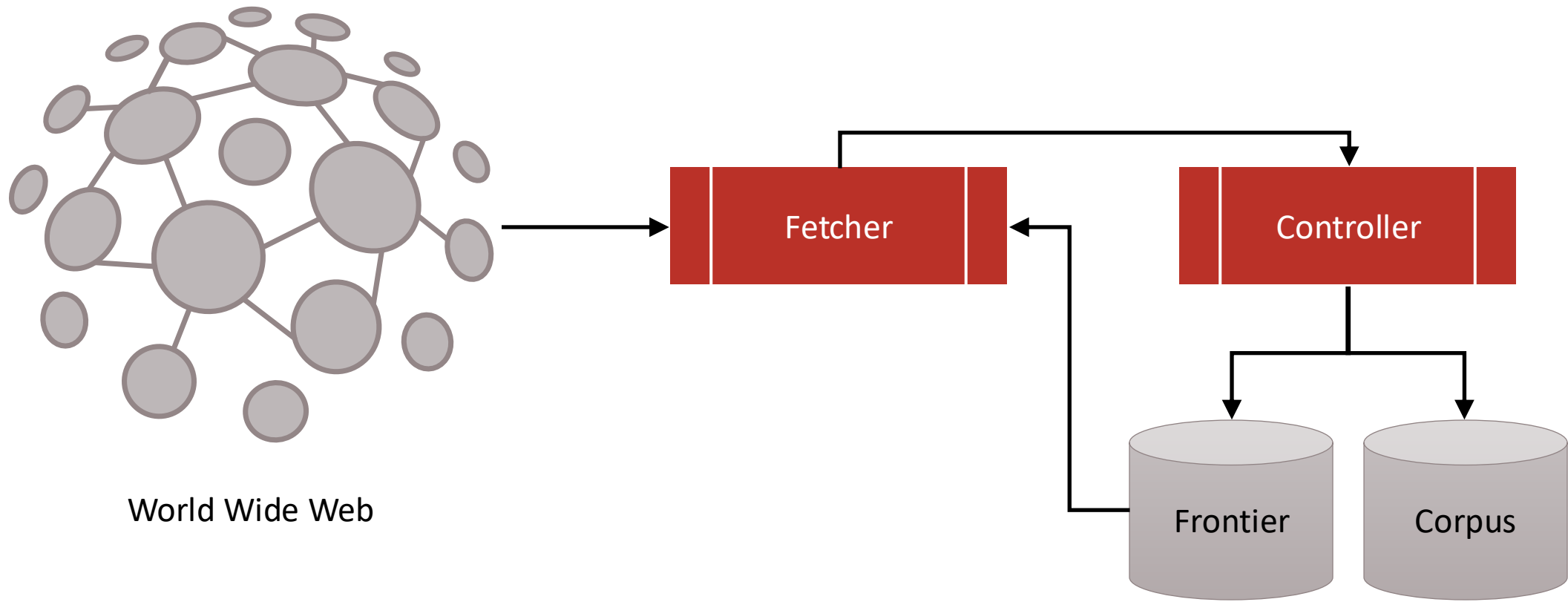
Coverage: fraction of the Web in the local corpus

Freshness: updatedness of local corpus vs. the Web

Utility: fraction of useful pages in the local corpus

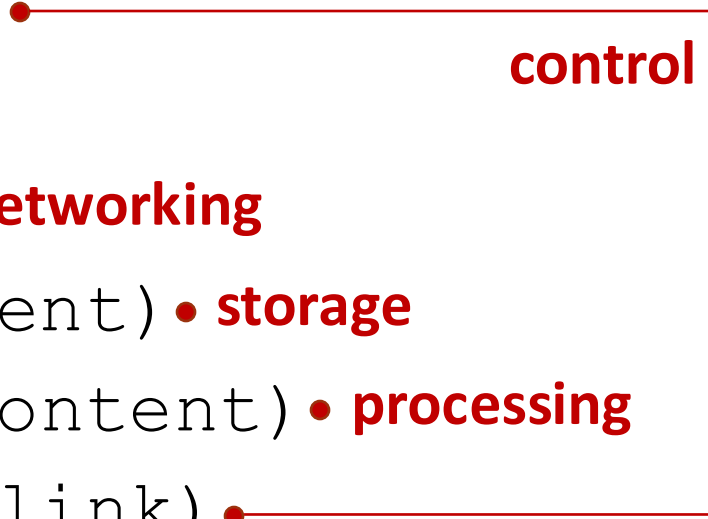
Efficiency: bytes downloaded per unit of time

Crawling overview



Crawling overview

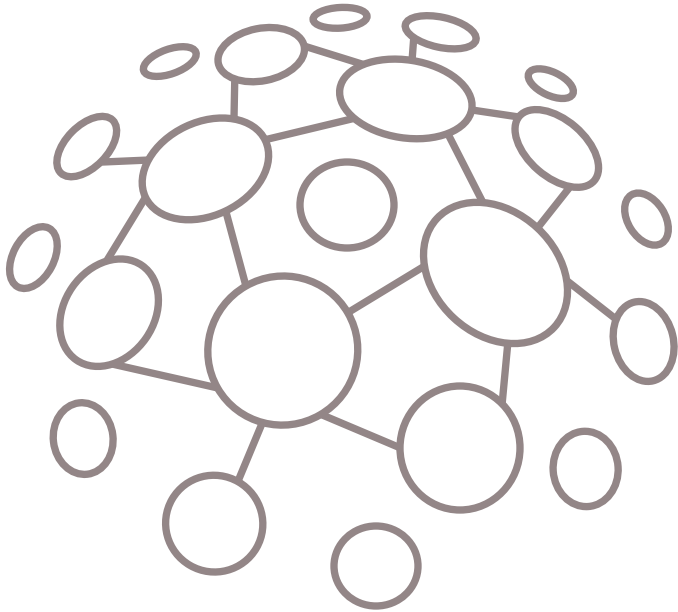
```
1: procedure crawler(frontier, corpus)
2:   while not frontier.empty()
3:     url ← frontier.dequeue() • control
4:     if crawlable(url) • politeness
5:       content ← fetch(url) • networking
6:       corpus.store(url, content) • storage
7:       for outlink in parse(content) • processing
8:         frontier.enqueue(outlink) • control
```

A red rectangular box highlights the control flow of the crawler procedure. It starts at line 3 with a red dot, extends to the right, then down to line 8, and finally left to a red dot at the end of line 8. The word "control" is written in red text at the top right of the box.

Crawl seeding

Start crawling from one or more *seed* pages

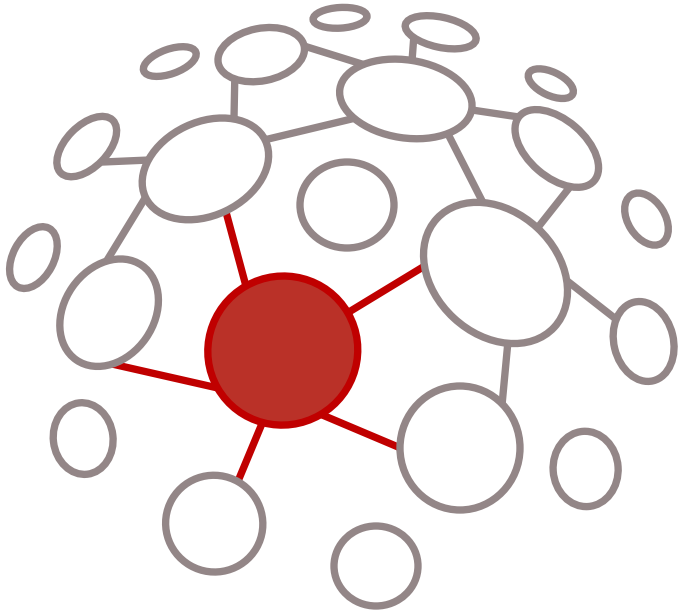
- Typically, hubs linking to many other pages



Traversing the Web

Fetch page, enqueue its outlinks

- Separate queues for coverage and freshness



Discovery queue

- URLs pointed by downloaded pages
- Goal is to increase coverage

Refreshing queue

- URLs of already downloaded pages
- Goal is to increase freshness

Enqueuing policy

Must keep track of seen URLs

- Route between discovery and refreshing queues

Hard space and time requirements

- Trillions of URLs (and counting), instant lookups

Stored as a distributed hashtable on disk

- Caching for frequent / recent URLs

URL keying

Multiple URLs map to the same page

- <http://www.CNN.com:80/x/./index.html> → cnn.com

Normalization ensures unique keying

- Needed for seen checks
- Needed for crawling distribution

Normalizing URLs

Semantic-preserving operations

- Lowercasing scheme (“http”) and host (“cnn.com”)
- Removing the default port (80)

Non-semantic-preserving operations

- Removing fragments (#section)
- Removing / adding leading “www”

Dequeuing policy

Goal is to reach important pages quickly

- Either for discovery or revisitation

Discovery policies

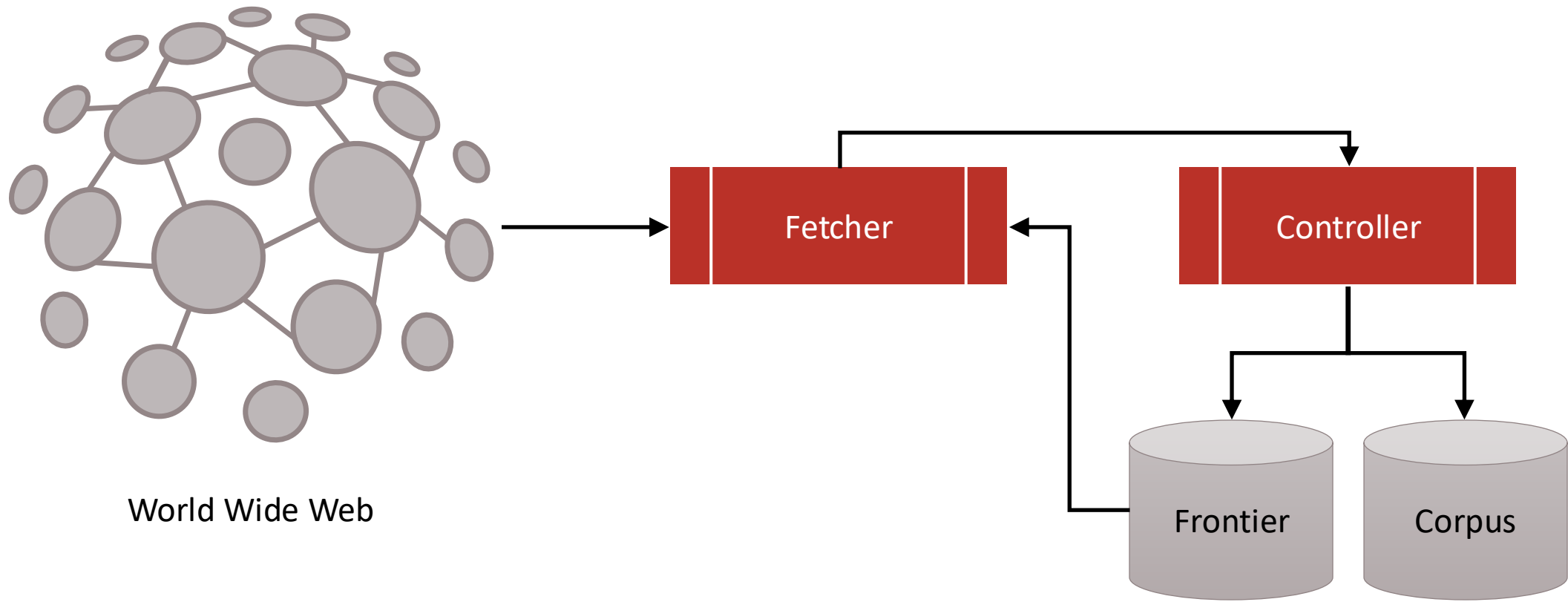
- Random
- Ordered (breadth-first)
- Centrality (indegree, PageRank)

Dequeuing policy

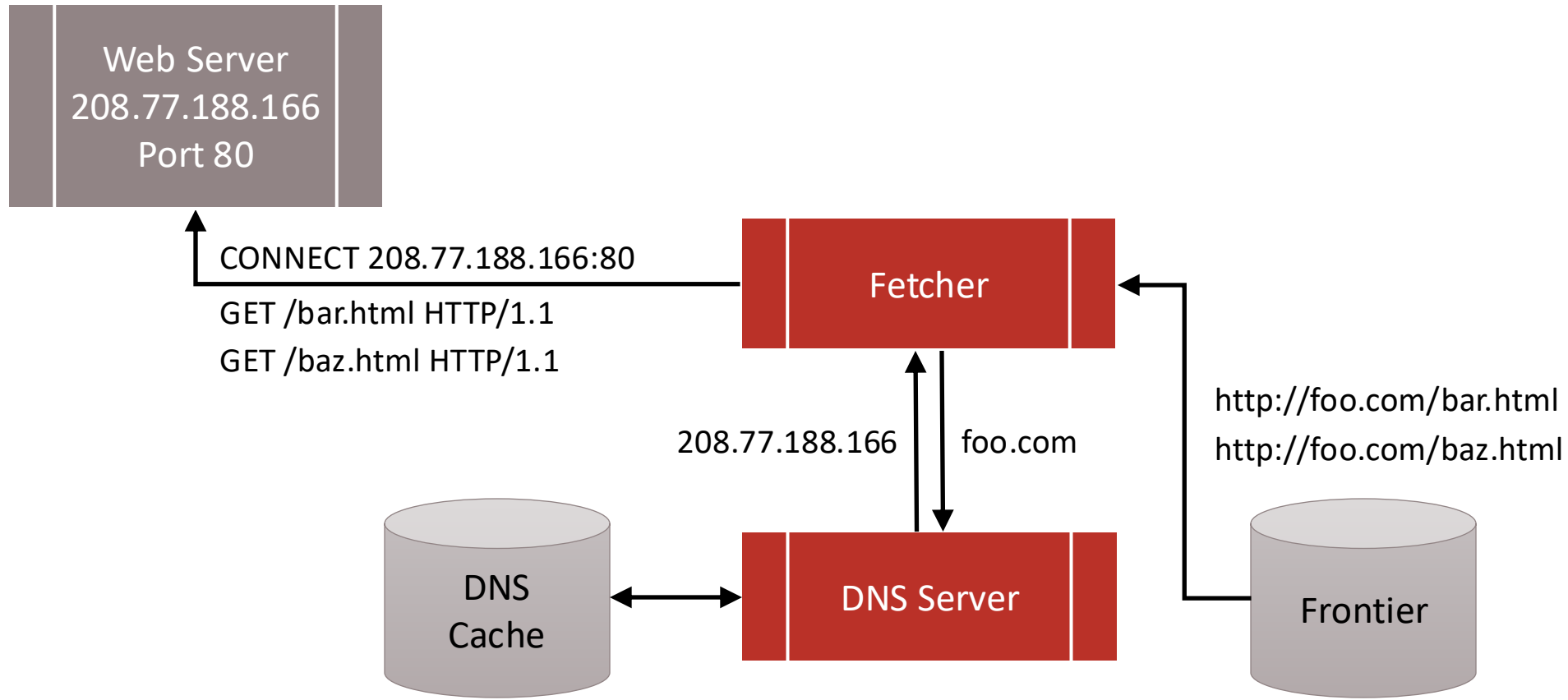
Revisitation policies

- Random
- Centrality (indegree, PageRank)
- Impact (clicks, browsing, likes, shares)
- Age (time since last crawl)
- Longevity (update frequency)

Page fetching



Page fetching



Multi-threading

Crawling is a network-bound task

- Crawlers employ multiple threads to crawl different web pages simultaneously, increasing throughput

A single node can run up to 100 crawling threads

- CPU context switching overhead makes it infeasible to increase multi-threading much further

Politeness

Multi-threading may lead to politeness issues

- Uncoordinated simultaneous requests may overload web servers and even entire sub-networks

A polite crawler

- Keeps at most one TCP-IP connection open per server
- Adds a delay between consecutive requests (~20-60s)

Robots exclusion protocol

A standard from the early days of the World Wide Web

- **robots.txt** provided by a web site to guide web crawlers
- Signals which parts of the site should not be crawled
- Crawlers often cache robots.txt files for efficiency purposes

robots.txt file

User-agent: googlebot	# all services
Disallow: /private/	# disallow this directory

User-agent: googlebot-news	# only the news service
Disallow: /	# on everything

User-agent: *	# all robots
Disallow: /something/	# on this directory

User-agent: *	# all robots
Crawl-delay: 10	# wait at least 10 seconds

Disallow: /dir1/	# disallow this directory
Allow: /dir1/myfile.html	# allow a subdirectory

Host: www.example.com	# use this mirror
-----------------------	-------------------

Mirror sites

A mirror site is a replica of an existing site

- Reduced traffic and improved availability for the site

Mirror sites lead to redundant crawling

- Reduced discovery rate and coverage for the crawler

Mirror sites can be detected by analyzing

- URL similarity, link structure, content similarity

Sitemap protocol

A standard that complements the robots exclusion protocol

- **sitemap.xml** generated by web server administrators
- Tells crawler about pages it might not otherwise find
- Gives crawler a hint about when to check for updates

```
<!-- sitemap.xml file -->

<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/
  sitemap/0.9">

  <url>
    <loc>http://www.test.com/</loc>
    <lastmod>2015-04-03</lastmod>
    <changefreq>weekly</changefreq>
    <priority>0.65</priority>
  </url>

  <url>
    ...
  </url>

</urlset>
```

Crawling architectures

Single node (not scalable)

- CPU, RAM, and disk becomes a bottleneck

Multiple nodes (scalable)

- Distributed crawling in a single data center

Distributed web crawling

Web partitioned by URL hashing

- May lead to impolite crawling

Web partitioned by host hashing

- Politeness controlled per host

Uncoordinated crawling leads to duplicate pages

- Same URL downloaded by different nodes

Crawling architectures

Single node (not scalable)

- CPU, RAM, and disk becomes a bottleneck

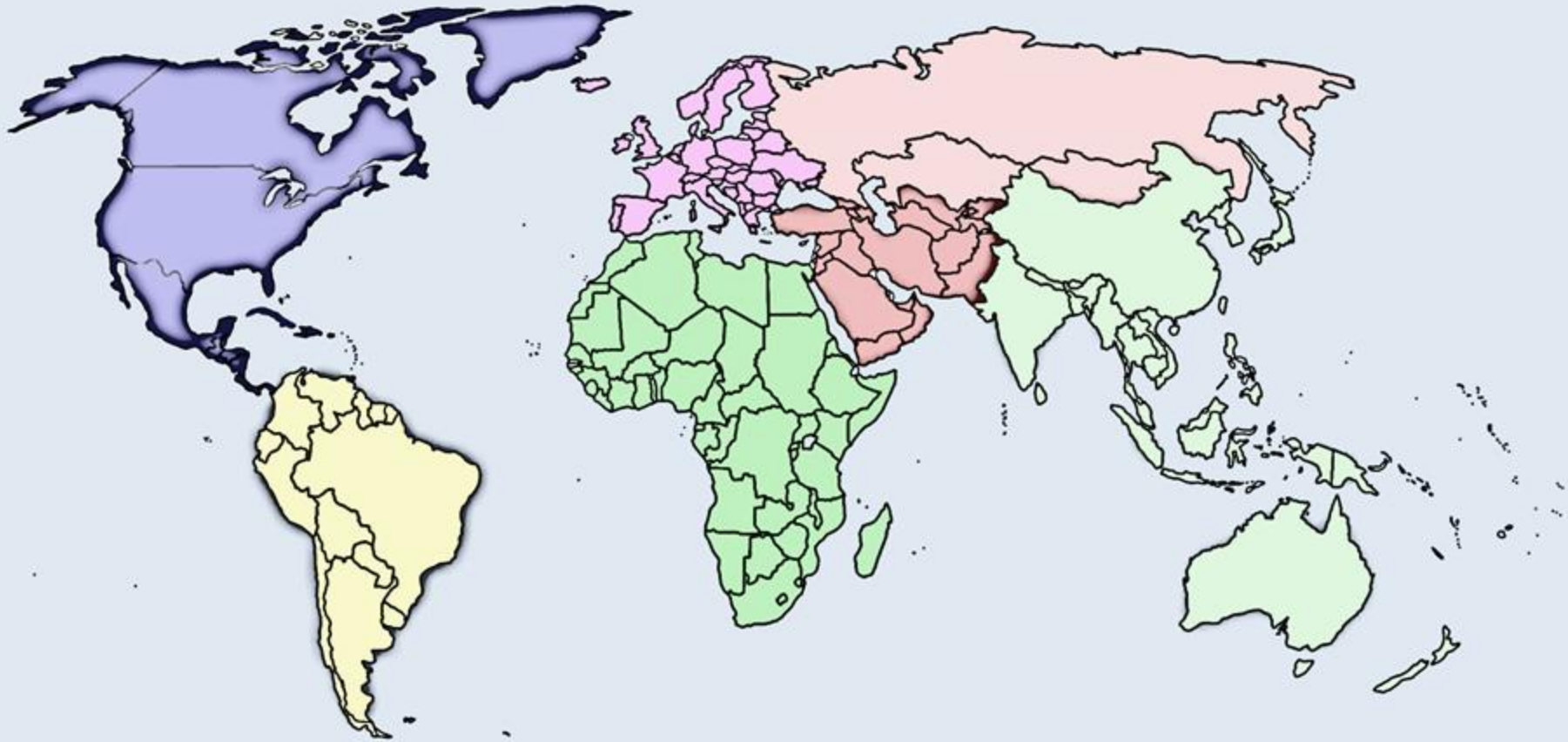
Multiple nodes (scalable)

- Distributed crawling in a single data center

Geographically distributed (scalable, reduced latency)

- Distributed crawling in multiple data centers

Geographically distributed web crawling



Region-based partitioning

Benefits

Higher crawling throughput

- Geographical proximity, lower crawling latency

Improved network politeness

- Lower overhead on routers

Resilience to network partitions

- Better coverage, increased availability

Content parsing

Text is stored in hundreds of incompatible file formats

- Raw text, RTF, HTML, XML, MS Word, ODF, PDF, ...

Typically use a conversion tool

- Converts the document content into a marked down text format such as HTML or XML
- Retains some of the important formatting information

Character encoding

Mapping between bits and glyphs

- Getting from bits in a file to characters on a screen
- Can be a major source of incompatibility

ASCII is basic character encoding scheme for English

- Unicode as a single mapping that attempts to include all glyphs in common use in all known languages

Removing noise

Many web pages contain text, links, and pictures not directly related to the main content of the page

- Noisy content can negatively impact ranking

Many techniques to detect content blocks

- Page content vs. advertisements
- Non-content material ignored or flagged

Content storage

Store converted document text

- Efficient information extraction (e.g. snippets)

Requirements for document storage system

- Random access (keyed by hashed URL)
- Fast updates (e.g. modifications, anchor text)
- Compressed representation

Compression

Text is highly redundant (or predictable)

- Compression exploits this redundancy to make files smaller without losing any of the content

Popular algorithms can compress HTML by 80%

- DEFLATE (zip, gzip) and LZW (UNIX compress, PDF)
- Compress large files in blocks to make access faster

Focused web crawling

Goal is to fetch thematic pages

- Topic (nuclear energy)
- Genre (music)
- Type (forums)
- Demographics (kids)

Feature-based (URL patterns, referrer, graph structure)

Deep web crawling

Crawling traditionally focused on the surface Web

- Web pages accessible by following links

Hidden content potentially useful

- Unlinked pages
- Private sites
- Scripted content

Summary

Crawled corpus as a cache of the Web

- Enables fast indexing and retrieval

Web as a huge, dynamic environment

- Must aim for coverage and freshness
- Must respect politeness constraints

Several open challenges

References

[Search Engines: Information Retrieval in Practice](#), Ch. 3

Croft et al., 2009

[Scalability Challenges in Web Search Engines](#), Ch. 2

Cambazoglu and Baeza-Yates, 2015



UNIVERSIDADE*FEDERAL
DE*MINAS*GERAIS

Coming next...

Document Understanding

Rodrygo L. T. Santos
rodrygo@dcc.ufmg.br