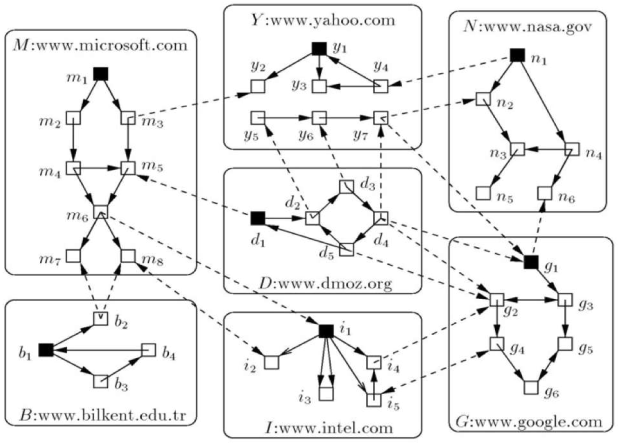
Web Crawling

- **Web crawling** is the process of locating, fetching, and storing the pages available in the Web
- Computer programs that perform this task are referred to as
  - Crawlers
  - Spiders
  - Harvesters
  - Robots
- Web crawler repositories
  - Cache the online content in the Web
  - Provide quick access to the physical copies of pages in the Web
  - Help to speed up the indexing process

3

Web Graph

- **Fundamental Assumption**: The web is well linked
  - Web crawlers exploit the hyperlink structure of the Web



4

(Very Basic) Web Crawling Process

- Initialise a URL **download queue** (**URL Frontier**) with some **seed** URLs
  - Good **seeds** will link to many other pages – e.g. for crawling the university, use the homepage as a seed
- Repeat the following steps
  - Fetch the content of a URL selected from the **download queue**
  - Store the fetched content in a repository
  - Extract the hyperlinks within the fetched content
  - Add the **new** extracted links into the **download queue** (URL Frontier)

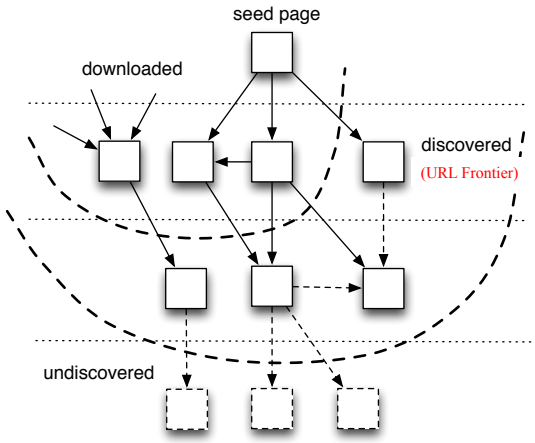
Explain why such a seemingly simple procedure is problematic?

Challenges

- To fetch 1,000,000,000 unique pages in one month:
  - We need to fetch almost 400 pages per second
  - Actually: many more since many of the pages we attempt to crawl will be duplicates, unreachable, spam etc.
- Building an industrial strength & **scalable** crawler is a (challenging) system **engineering problem**
  - We need many machines – how do we distribute?
  - Latency/bandwidth – how to make best use of available resources
  - Identifying duplicates/near duplicates
  - How often should we re-crawl sites (freshness, politeness)?
  - etc

Web Crawling

- Crawling divides the Web into three sets
  - downloaded
  - discovered
  - undiscovered



Why a crawler can't see all documents on the web?

7

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
  - Only crawl allowed pages and not too quickly
  - Respect robots.txt (more on this shortly)

Specify two must do requirements on crawler

8

Spider Traps & Malicious Intent

- Crawler needs to avoid crashing on
  - Ill-formed HTML
    - e.g.: page with 68 KB of null characters
  - Misleading & hostile sites
    - Indefinite number of pages dynamically generated by Web application scripts (e.g. CGI/Python/ASP)
    - Paths of arbitrary depth created using soft directory links and path remapping features in HTTP server
    - Spam sites (e.g. link farms)
    - etc.
- Why would a site do this?

What are spider traps? Give examples.

9

Spider Traps: Possible Solutions

- No automatic technique can be foolproof
- Check for URL length …
  - Avoid ill-formed URLs, CGI scripts etc.
- Trap guards
  - Preparing regular crawl statistics
  - Adding dominating sites to a guard module
  - Disable crawling active content such as CGI form queries
  - Eliminate URLs with non-textual data types

How to address spider traps?

10

Explicit and Implicit Politeness

- **Explicit politeness**: specifications from webmasters on what portions of a site can be crawled
  - robots.txt
  - Robot <META> tag

- Implicit politeness: even with no specification, avoid hitting any site too often
  - Even if we restrict only one thread to fetch from a host, it can hit it repeatedly
  - Common heuristic: insert a time gap between successive requests to a host

What impact politeness criteria have on the design of a crawler?

11

Robot Exclusion Protocol

Discuss 2 approaches to specifying explicit politeness.

- A standard from the early days of the Web (1994)
- A file (called **robots.txt**) in a web site advising web crawlers about which parts of the site are accessible
- Crawlers often cache robots.txt files for efficiency purposes

```
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: /directory1/         # disallow this directory
Allow: /directory1/myfile.html # allow a subdirectory

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

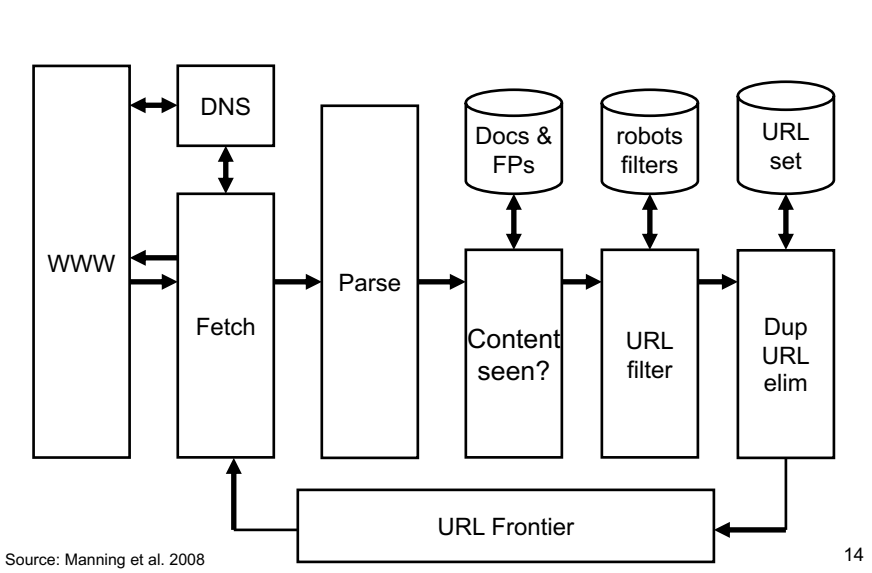
12

Duplicate Detection

- **Duplication** is wide spread on the web
  - If the page just fetched is already in the index, do not further process it to avoid wasting crawling resources and annoying users
- Exact duplicates: easy to eliminate: e.g., use hash/fingerprint
- Near-duplicates: Abundant on the web and more difficult to eliminate
  - Could be identified using document fingerprints or shingles
  - Index page using textual content as a key.

13

Basic Crawl Architecture



Source: Manning et al. 2008

14

Key Crawling Components

- URL Frontier: a *queue* data structure containing the URLs to be crawled
  - Can be sorted to give priority to some pages over others
- Seen URLs: a *set* data structure, permitting the crawler to know if it has crawled a URL before or not
- Fetcher: downloads an unseen URL & store it in the data repository
- Parser: extracts outgoing links from the page
- URL Filtering: eliminate URLs that appear to be images, or that are disallowed by the robots.txt files
- Content-seen filter: eliminate duplicate pages

What are the main components of a crawler?

15

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: permits full use of available processing and network resources
  - E.g., no idle threads!
- *Quality: biased towards useful pages first?*
- *Freshness: Crawler must operate in continuous mode*
- *Extensible: to add new data formats, new fetch protocols etc.*

Specify the crawler "should do" criteria

16

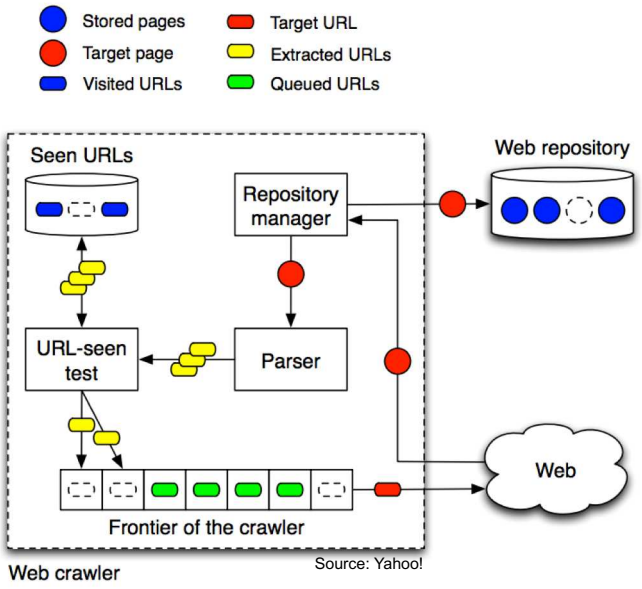
URL Prioritisation

- A state-of-the-art web crawler maintains two separate queues for prioritising the download of URLs:
  - **Discovery queue**
    - Downloads pages pointed to by already discovered links – *But in which order?* A **queuing strategy** is needed
    - Tries to increase the **coverage** of the crawler – i.e. you want to crawl pages that are likely going to be relevant to some search queries
  - **Refreshing queue**
    - Re-downloads already downloaded pages
    - Tries to increase the freshness of the repository

How to optimally allocate available crawling resources between page discovery and refreshing is still an ongoing research problem

17

Discovery Queue

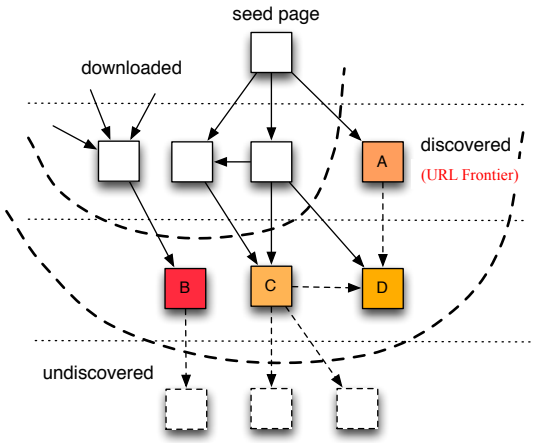


What are the main components of a crawler?

18

URL Prioritisation

- Random (A, B, C, D)
- Breadth-first (A)
- In-degree (C)
- PageRank (B)



(more intense red color indicates higher PageRank)
Source: Yahoo!

19

Crawling/Queuing Strategy (1)

- The way new links are added to the queue (URL Frontier) determine their priority
- Two (common) **crawling approaches**
  - **FIFO** (append to end of Q) gives breadth-first search
    - Gets the important pages earlier in the crawl
  - LIFO (add to front of Q) gives depth-first search

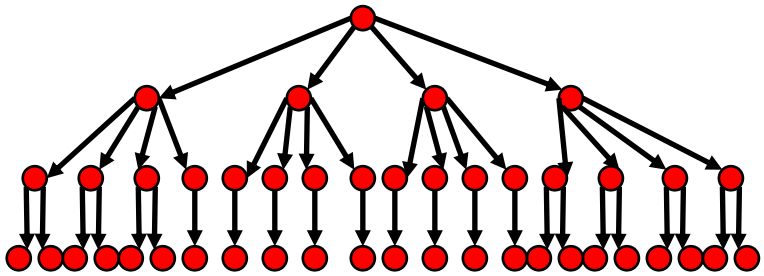
- Ordering the queue gives a crawler that directs its search towards more **useful** pages
  - Historically, PageRank has been used (but requires the whole link matrix)
  - **OPIC** (On-line Page Importance Computation) is more effective and continuously refines its estimate of page importance while the web graph is visited

Discuss the main crawling strategies in developing the frontier and their effect?

20

Crawling/Queuing Strategy (2)

Depth-first Queuing (LIFO)

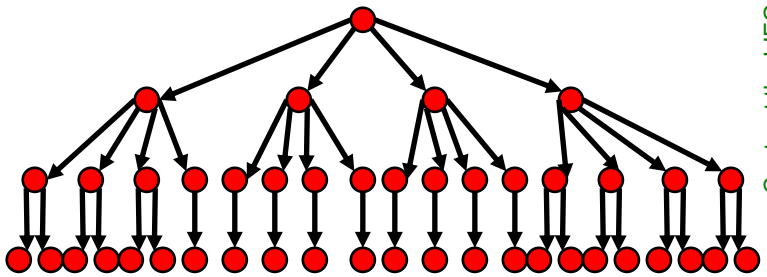


How does depth-first crawling affect politeness?

Depth-first requires memory of only depth times branching-factor (linear in depth) but gets "lost" pursuing a single thread.

Crawling/Queuing Strategy (3)

Breadth-first Queuing (FIFO)



Contrast the LIFO and FIFO queuing strategies on crawler design

Breadth-first explores uniformly outward from the root page but requires memory of all nodes on the previous level (exponential in depth)

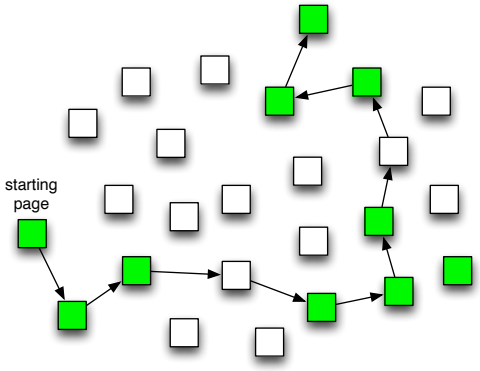
Standard *crawling* strategy

Focused Web Crawling

- The goal is to locate and download a large portion of web pages that match a given target theme as early as possible.

- Example themes
  - Topic (e.g. energy)
  - Media type (forums)
  - Demographics (kids)

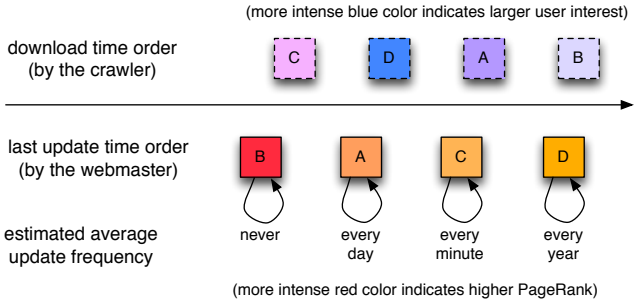
- Strategies
  - URL patterns
  - Referring page content
  - Graph structure (e.g. applying HITS)



23

URL Prioritisation (Refreshing)

- Random (A, B, C, D)
- PageRank (B)
- Age (C)
- User feedback (D)
- Longevity (A)



Source: Yahoo!

24

URL Frontier: Two Main Considerations

Why Politeness and Freshness are conflicting?

- 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 very 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.

25

Crawling Metrics

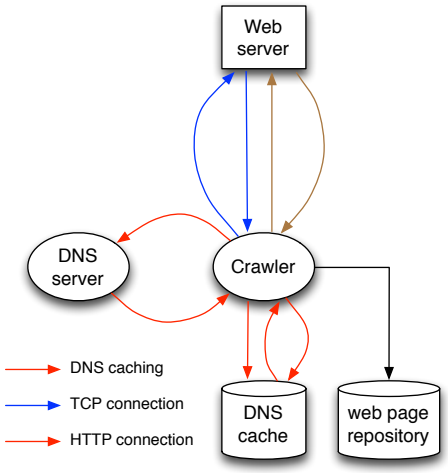
- **Quality metrics**
  - Coverage: percentage of the Web discovered or downloaded by the crawler
  - Freshness: measure of staleness of the local copy of a page relative to the page's original copy on the Web
  - Page importance: percentage of important (e.g., popular) pages in the repository

- **Performance metrics**
  - **Throughput**: content download rate in bytes per unit of time

26

DNS Caching

- Before a web page is crawled, the host name needs to be resolved to an IP address
- Since the same host name appears many times, DNS entries are locally cached by the crawler



Source: Yahoo!

27

Multi-threaded Crawling

- Multi-threaded crawling
  - Crawling is a network-bound task
  - Crawlers employ **multiple threads** to crawl different web pages simultaneously, increasing their **throughput** significantly
  - In practice, a single node can run up to around a hundred crawling threads
  - Multi-threading becomes infeasible when the number of threads is very large due to the overhead of **context switching**
- Multi-threading leads to politeness issues
  - The crawler may issue too many download requests at the same time, overloading a server and the entire sub-network
  - Importance of observing **politeness** (e.g. a delay of 20 seconds between 2 consecutive downloads from the same server and **closing** the established TCP-IP **connection** after each download)

28

Mirror Sites

- A **mirror site** is a replica of an existing site, used to reduce the network traffic or improve the **availability** of the original site
- Mirror sites lead to redundant crawling and, in turn, reduced discovery rate and coverage for the crawler
- Mirror sites can be detected by analysing
  - URL similarity
  - Link structure
  - Content similarity

29

Data Structures

- Good implementation of data structures is crucial for the **efficiency** of a web crawler
- The most critical data structure is the "**seen URL**" table
  - Stores all URLs discovered so far and continuously grows as new URLs are discovered
  - Consulted before each URL is added to the discovery queue
  - Has high space requirements (mostly stored on the disk)
    - URLs are stored as **MD5 hashes**
    - Frequent/recent URLs are cached in memory

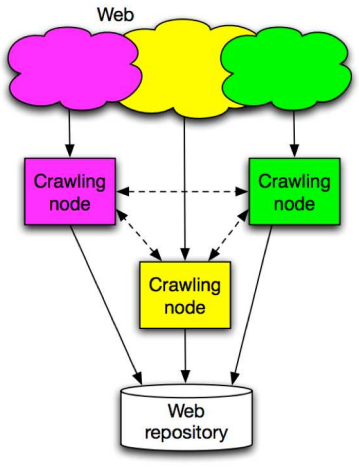
30

Crawling Architectures

- Single node
  - CPU, RAM, and disk becomes a bottleneck
  - Not scalable
- **Parallel** (multiple nodes)
  - Multiple computers, single data center
  - Scalable
- **Geographically distributed**
  - Multiple computers, multiple data centers
  - Scalable
  - Reduces the network latency

Parallel Crawling

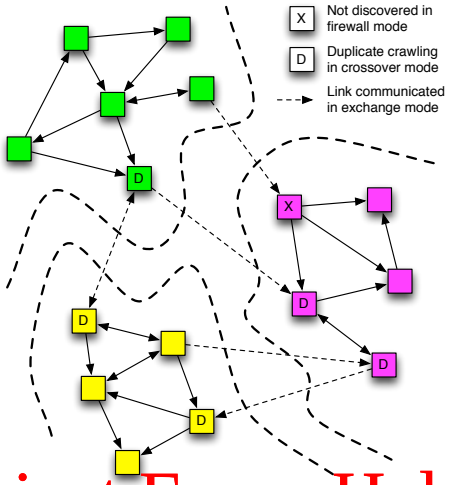
- Web partitioning
  - Typically based on the **MD5 hashes** of URLs or host names
  - Site-based partitioning is preferable
    - URL-based partitioning may lead to politeness issues if the crawling decisions given by individual nodes are not coordinated
- Fault tolerance
  - When a crawling node dies, its URLs are partitioned over the remaining nodes



Source: Yahoo!

Coordination Between Nodes

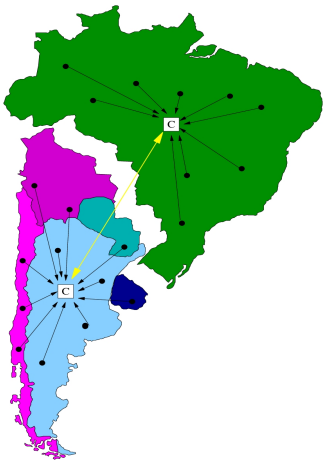
- Firewall mode
  - Lower coverage
- Crossover mode
  - Duplicate pages
- Exchange mode
  - Communication overhead



33

Geographically Distributed Web Crawling

- Benefits
  - Higher crawling throughput
    - Geographical proximity
    - Lower crawling latency
  - Improved network politeness
    - Less overhead on routers because of fewer hops
  - Resilience to network partitions
    - Better coverage
  - Increased **availability**
    - Continuity of business
  - Better coupling with distributed indexing/search
    - Reduced data migration



Optimum geographical placement is an ongoing research problem

34

Mishandling Queries on Emerging Topics

- **Guaranteed zero recall**: if no on-topic information exists in the index, the user can never find it no matter the reformulations (without prior knowledge)
- **Information need is urgent**: the user wants information immediately and is willing to switch providers in order to find it
- **High visibility failure**: like it or not, the media often uses breaking news queries to compare search engines

35

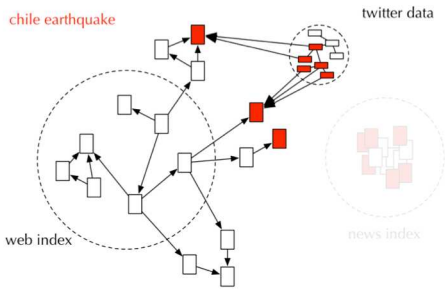
Sources of Real-time Data for New Pages Discovery

- **User monitoring**: toolbars, browsers, DNS requests
- **Interaction monitoring**: email, IM, SMS
- **Real-time *personal* publishing**: Facebook, Twitter, blogs, delicious, etc.

36

Using Twitter as a Crawling Sensor

- Tweets can include embedded **URLs**
- Tweets are generated by unique **users**
- Tweets include **text**



37

Other Uses to Twitter Data

- **Enriched document representation**: can add tweet text as a field.
- **Enriched popularity representation**: number of tweets as a surrogate for interest and for guiding the crawling process
- **Enriched user representation**: number of followers of a twitter user as a surrogate for user authority

38

Open Source Web Crawlers

- DataparkSearch: GNU General Public License
- GRUB: open source distributed crawler of Wikia Search
- **Heritrix**: Internet Archive's crawler
- ICDL Crawler: cross-platform web crawler
- Norconex HTTP Collector: licensed under GPL
- **Nutch**: Apache License
- Open Search Server: GPL license
- PHP-Crawler: BSD license
- Scrapy: BSD license
- Seeks: Affero general public license