**Crawlers and Spiders**

**Simple picture: Crawler operation**
- Create queue with “seed” pages
- Repeat
  - Fetch each URL on the queue
  - Parse fetched pages
  - Extract URLs they point to
  - Place the extracted URLs on a queue
- Until empty queue or out of time

**Q:** What type of search is this? DFS? BFS? Other?
**What benefits does the search method provide?**

**Simple picture - Complications**
- Web crawling isn’t feasible with one machine
- Malicious pages
- Spider traps – including 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
  - Duplicate and near-duplicate pages
- Politeness – don’t hit a server too often
What any crawler **must** do

- **Be Polite:**
  - Respect implicit and explicit politeness considerations for a website
  - Only crawl pages you’re allowed to
  - 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
- **Be efficient:** permit full use of available processing and network resources
- **Fetch** pages of “higher quality” first
- **Have Continuous** operation: Continue fetching fresh copies of a previously fetched page
- **Be Extensible:** Adapt to new data formats, protocols

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, [http://www.robotstxt.org/orig.html](http://www.robotstxt.org/orig.html)
- Website announces its request on what can(not) be crawled in a top-level URL, create a file `robots.txt`
- No robot should visit any URL starting with "/yoursite/temp/", except the robot called "searchengine"

```
User-agent: *
Disallow: /yoursite/temp/
User-agent: searchengine
Disallow: 
```

**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)
  - Normalize links (into absolute 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)
- Protect your work
  - Every so often, do a "checkpoint" = commit your data to disk

**Basic Crawl Architecture**

![Diagram of the crawl architecture]

- The Web
- DNS
- Fetch
- Parse
- Content seen?
- URL filter
- Robots filters
- URL set
- URL Frontier
- Dupl. URL Elim.
- Doc FP’s

**Why you need a DNS**

- Every server is enumerated in an IP address
- IPv4: 32 bits written as 4 decimal numerals up to 256, e.g. 149.130.12.213 ([Wellesley College](http://www.wellesley.edu))
  - How many addresses can it represent?
  - IPv6: 128 bits written as 8 blocks of 4 hex digits each, e.g. AF43:23BC:CAA1:0045:582:90AC:FFEE:8080
  - How many addresses are in IPv6?
- Client translates URLs to IP addresses, e.g. cs.wellesley.edu ➔ 149.130.136.19
  - Uses authoritative sites for address translation a.k.a. "Domain Name Server" (DNS)
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 DNS lookup implementations are synchronous: client blocked from issuing > one outstanding request @ a time

Solutions
- DNS caching – keep copies
- 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 (Doc FP’s) or shingles

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