Crawling

T. Yang, UCSB 293S

Some of slides from Crofter/Metzler/Strohman’s textbook

Where are we?

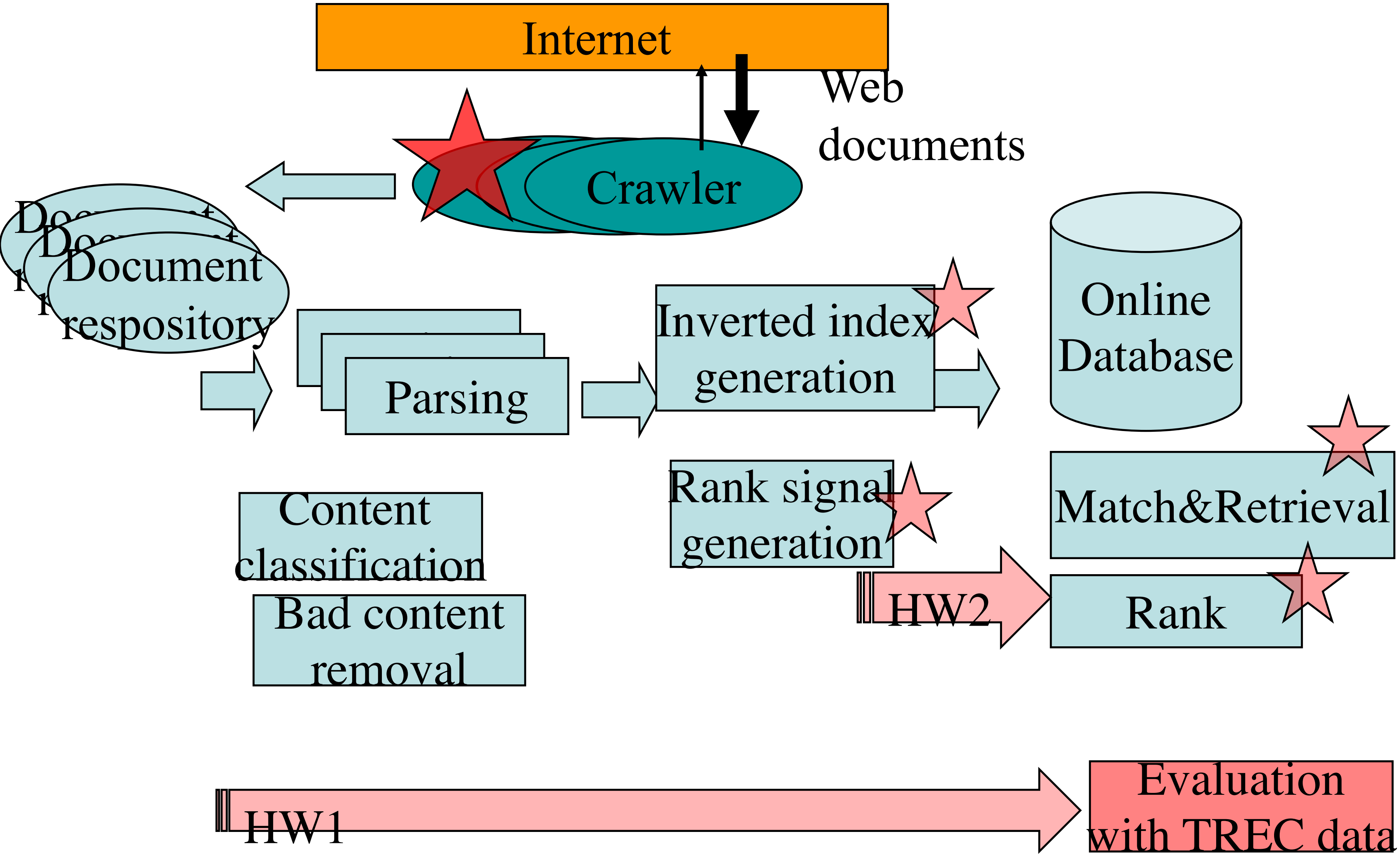
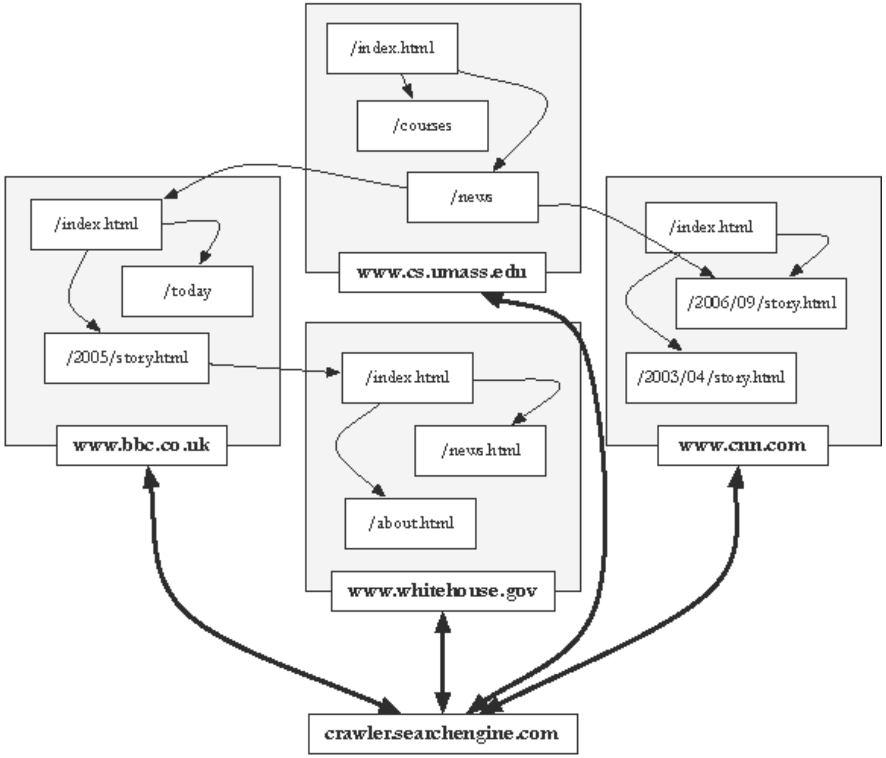

Internet
Web documents
Crawler
Document respository
Parsing
Inverted index generation
Online Database
Rank signal generation
Match&Retrieval
Content classification
Bad content removal
HW2
Rank
HW1
Evaluation with TREC data

Table of Content

- **Basic crawling architecture and flow**
  - Distributed crawling
- **Scheduling: Where to crawl**
  - Crawling control with robots.txt
  - Freshness
  - Focused crawling
- **URL discovery**
  - **Deep web, Sitemaps, & Data feeds**
- **Data representation and store**

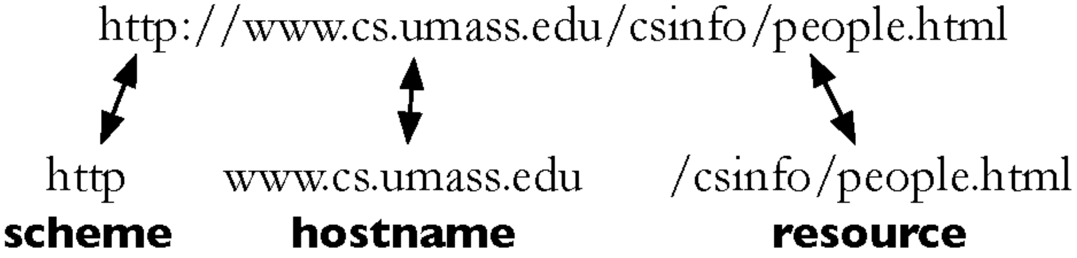
Web Crawler

- **Collecting data is critical for web applications**
  - Find and download web pages automatically

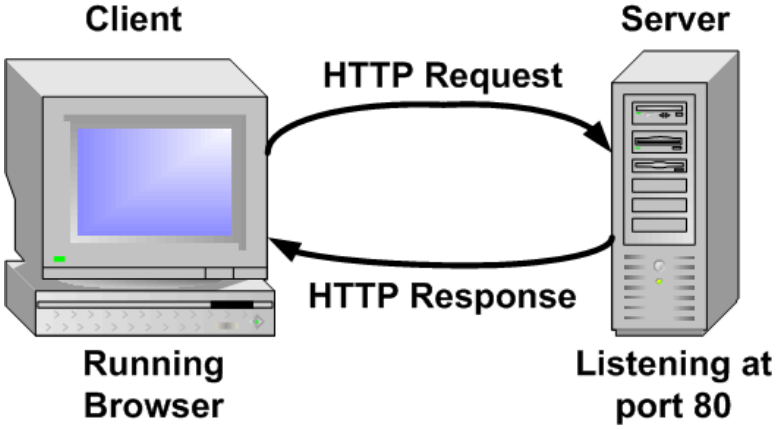


Downloading Web Pages

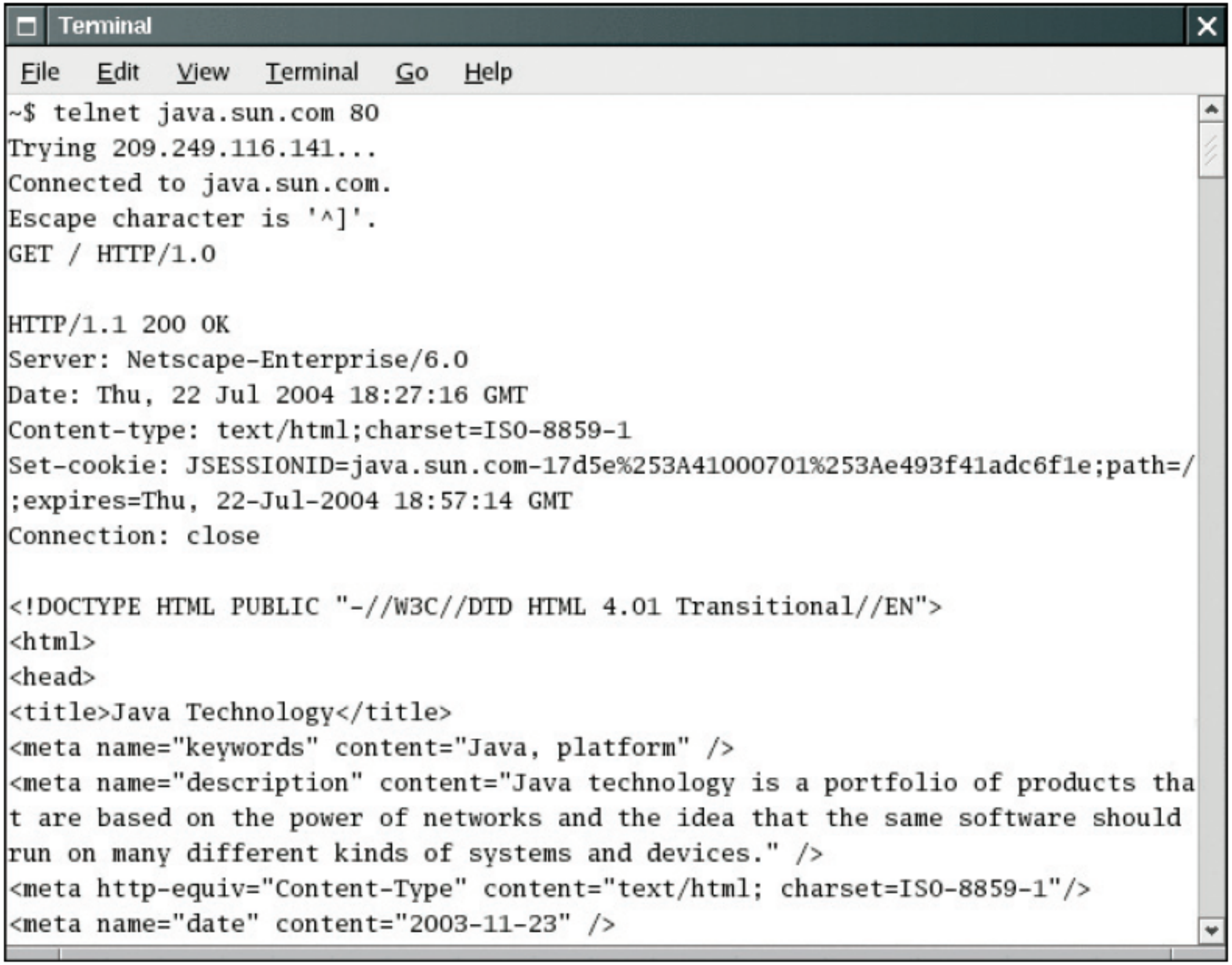
- Every page has a unique *uniform resource locator* (URL)



- Web pages are stored on web servers that use HTTP to exchange information with client software
  - HTTP /1.1



HTTP



```
~$ telnet java.sun.com 80
Trying 209.249.116.141...
Connected to java.sun.com.
Escape character is '^]'.
GET / HTTP/1.0

HTTP/1.1 200 OK
Server: Netscape-Enterprise/6.0
Date: Thu, 22 Jul 2004 18:27:16 GMT
Content-type: text/html;charset=ISO-8859-1
Set-cookie: JSESSIONID=java.sun.com-17d5e%253A41000701%253Ae493f41adc6f1e;path=/
;expires=Thu, 22-Jul-2004 18:57:14 GMT
Connection: close

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>Java Technology</title>
<meta name="keywords" content="Java, platform" />
<meta name="description" content="Java technology is a portfolio of products tha
t are based on the power of networks and the idea that the same software should
run on many different kinds of systems and devices." />
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1"/>
<meta name="date" content="2003-11-23" />
```

Figure 3 Using Telnet to Connect to a Web Server

Open-source crawler

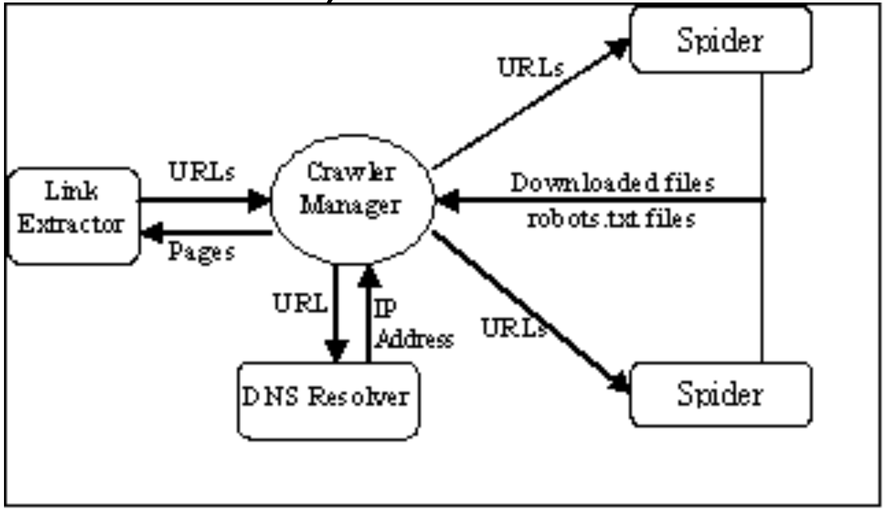
http://en.wikipedia.org/wiki/Web_crawler#Examples

- Apache Nutch. Java.
- Heritrix for Internet Archive. Java
- mnoGoSearch. C
- PHP-Crawler. PHP
- OpenSearchServer. Multi-platform.
- Seeks. C++
- Yacy. Cross-platform

Basic Process of Crawling

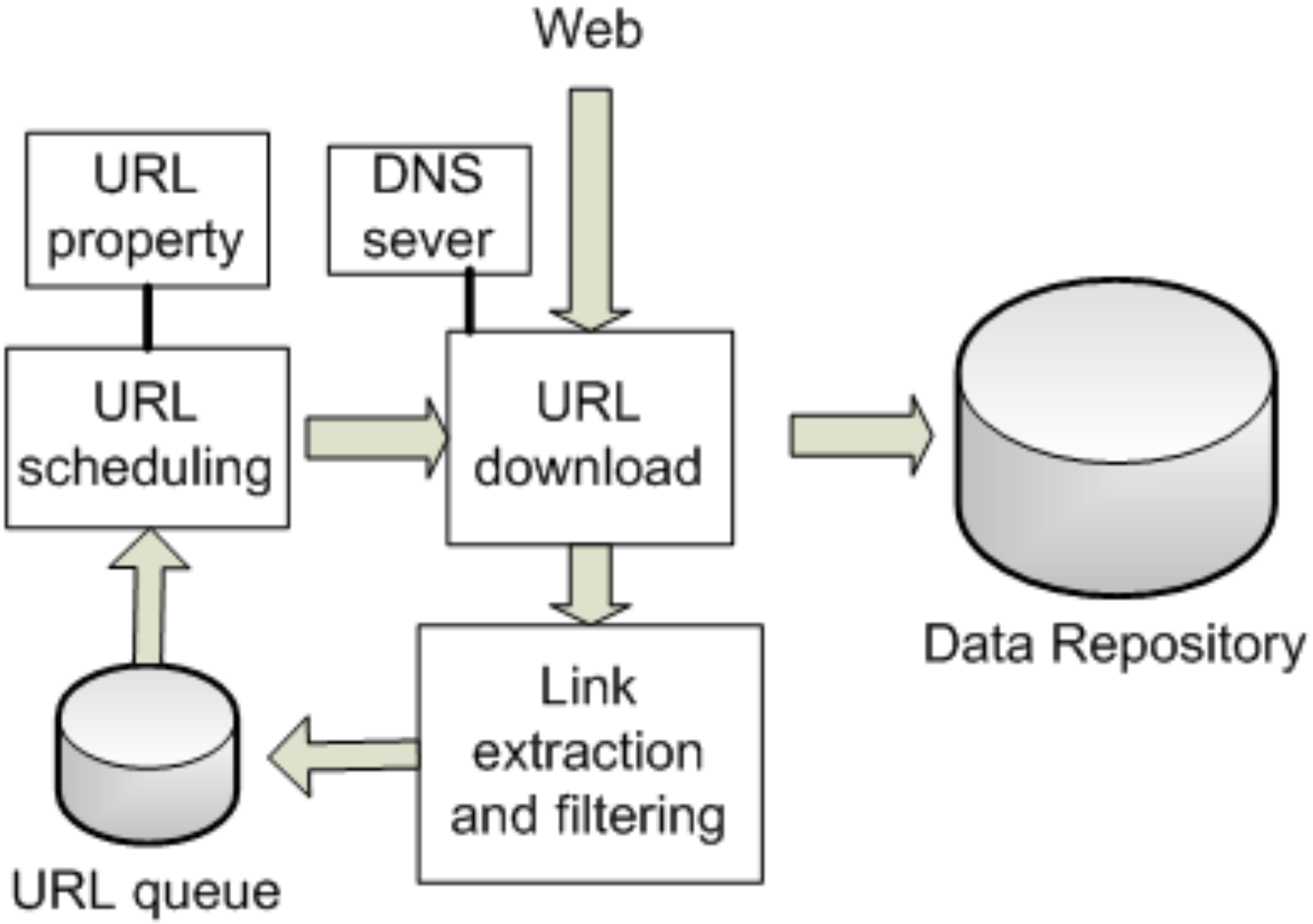
- Need a scalable *domain name system* (DNS) server (hostname to IP address translation)

- Crawler attempts to connect to server host using specific *port*



- After connection, crawler sends an HTTP request to the web server to request a page
  - usually a GET request

A Crawler Architecture at Ask.com



Web Crawling: Detailed Steps

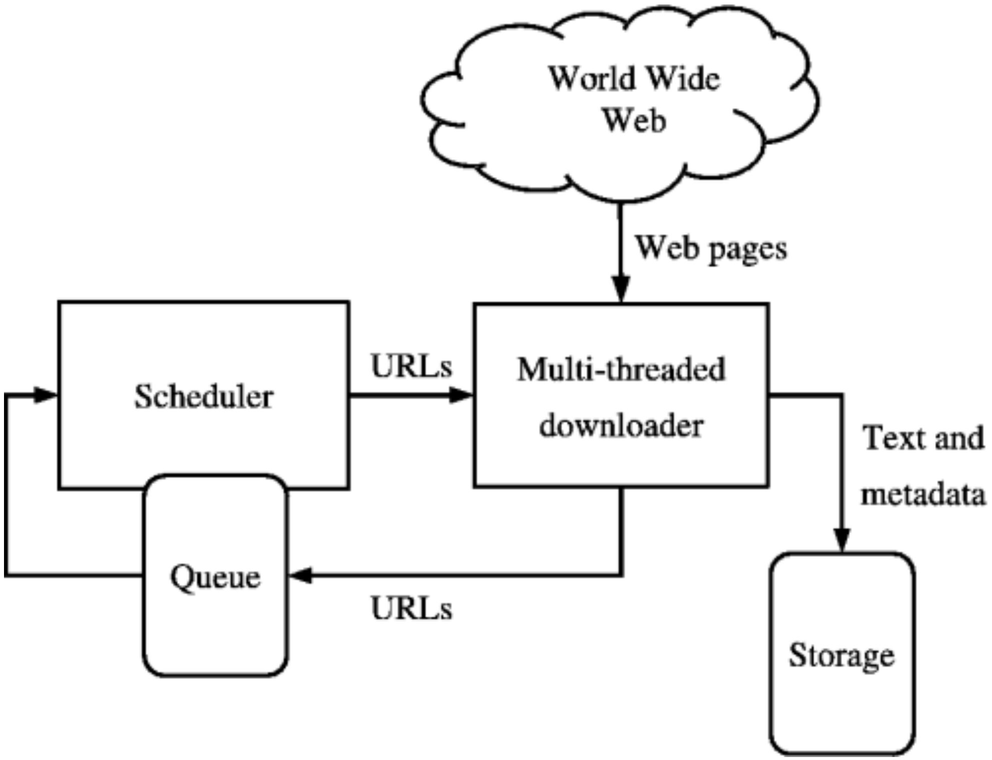
- Starts with a set of *seeds*
    - Seeds are added to a URL request queue
- Crawler starts fetching pages from the request queue
- Downloaded pages are parsed to find link tags that might contain other useful URLs to fetch
- New URLs added to the crawler's request queue, or *frontier*
- Scheduler prioritizes to discover new or refresh the existing URLs
- Repeat the above process

Multithreading in crawling

- **Web crawlers spend a lot of time waiting for responses to requests**
  - Multi-threaded for concurrency
  - Tolerate slowness

  of some sites
- **Few hundreds**

of threads/machine

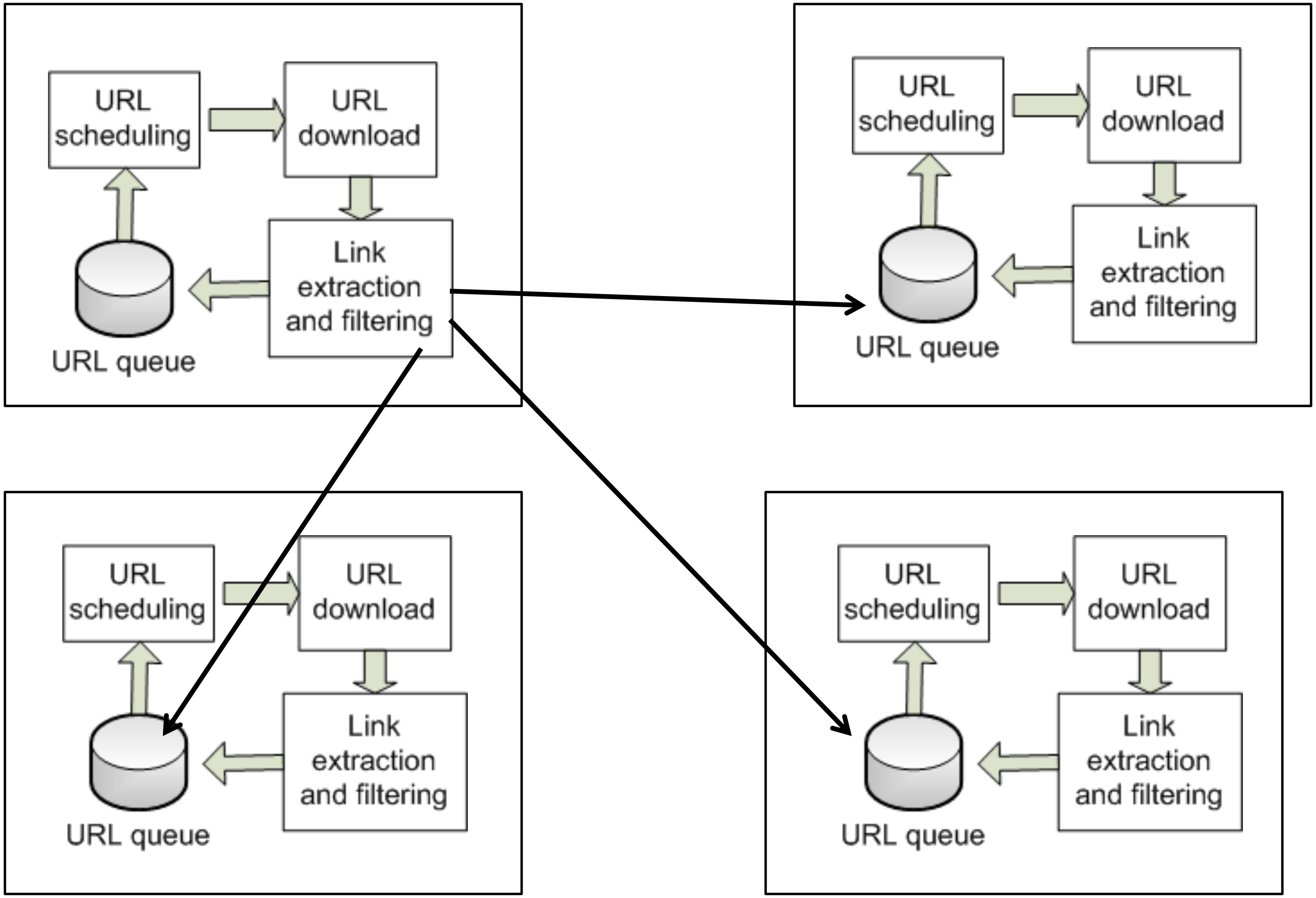


Distributed Crawling: Parallel Execution

- **Crawlers may be running in diverse geographies – USA, Europe, Asia, etc.**
  - Periodically update a master index
  - Incremental update so this is “cheap”
- **Three reasons to use multiple computers**
  - Helps to put the crawler closer to the sites it crawls
  - Reduces the number of sites the crawler has to remember
  - More computing resources

A Distributed Crawler Architecture

What to communicate among machines?



Variations of Distributed Crawlers

- **Crawlers are <u>independent</u>**
  - Fetch pages oblivious to each other.
- **<u>Static</u> assignment**
  - Distributed crawler uses a hash function to assign URLs to crawling computers
  - hash function can be computed on the host part of each URL
- **<u>Dynamic</u> assignment**
  - Master-slaves
  - Central coordinator splits URLs among crawlers

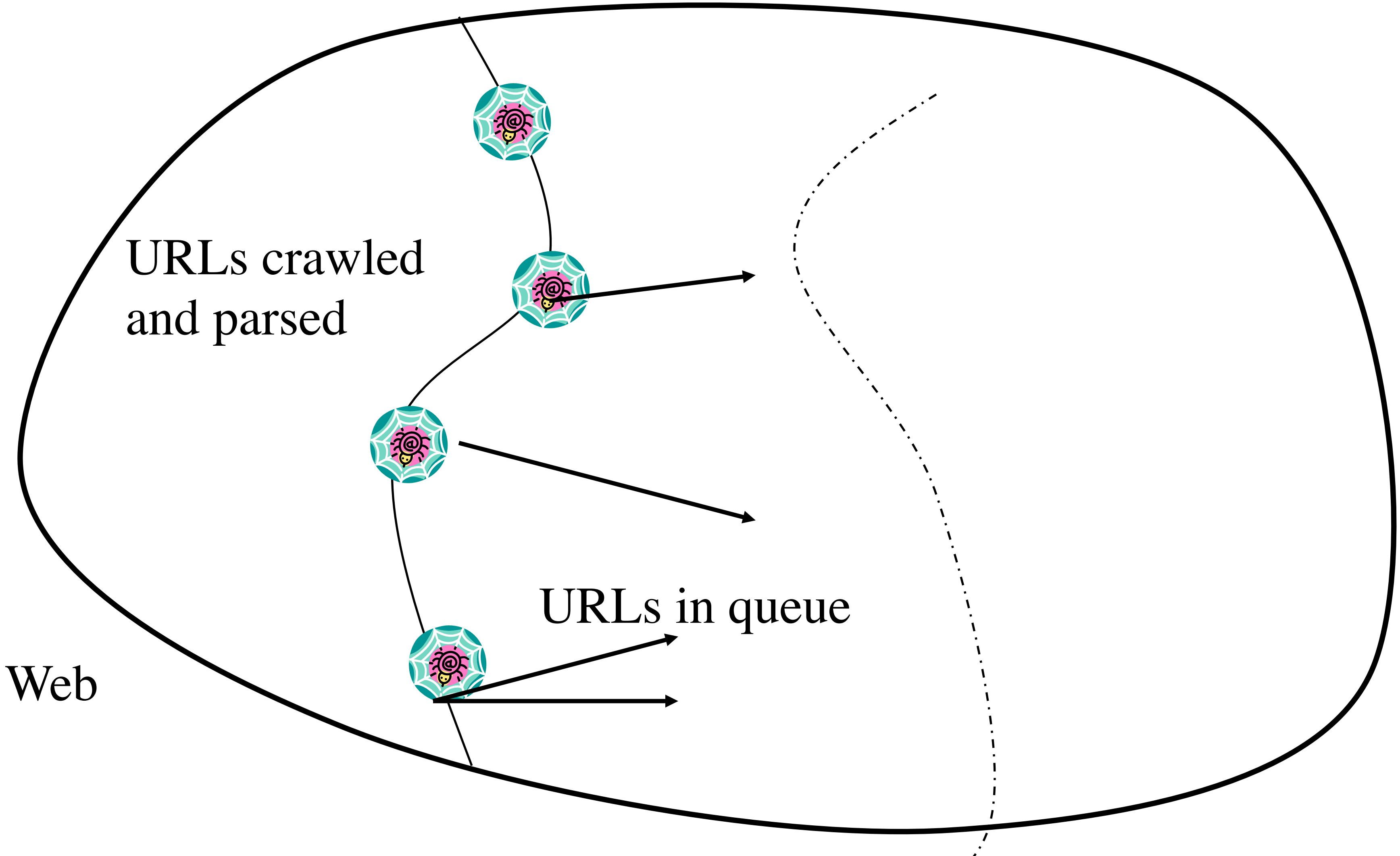
Comparison of Distributed Crawlers

	Advantages	Disadvantages
Independent	Fault tolerance Easier management	Load imbalance Redundant crawling
Hash-based URL distribution	Improved load imbalance Non-duplicated crawling	Inter-machine communication Load imbalance/slow machine handling
Master-slave	Load balanced Tolerate slow/failed slaves Non-duplication	Master bottleneck Master-slave comm.

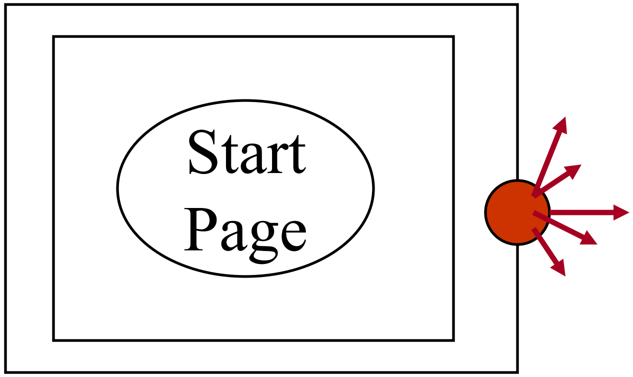
Table of Content

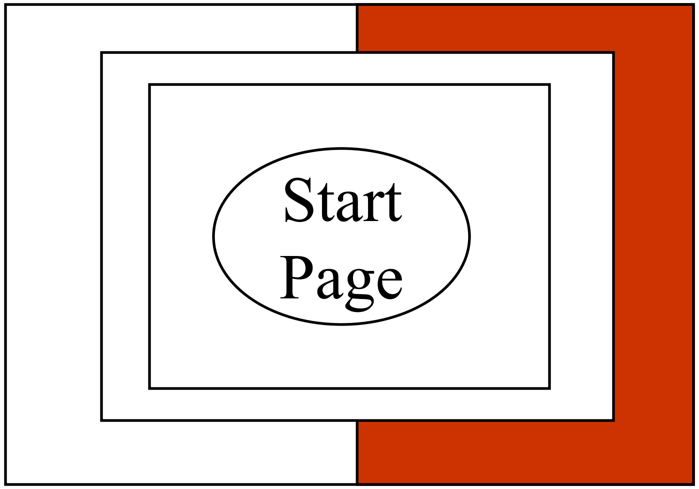
- **Crawling architecture and flow**
- **Schedule: Where to crawl**
  - Crawling control with robots.txt
  - Freshness
  - Focused crawling
- **URL discovery:**
  - **Deep web, Sitemaps, & Data feeds**
- **Data representation and store**

Where do we spider next?



How fast can spam URLs contaminate a queue?





BFS depth = 2

Normal avg outdegree = 10

100 URLs on the queue
including a spam page.

Assume the spammer is able to
generate dynamic pages with
1000 outlinks

BFS depth = 3
2000 URLs on the queue
50% belong to the spammer

BFS depth = 4
1.01 million URLs on the queue
99% belong to the spammer

Scheduling Issues: Where do we spider next?

- **Keep all spiders busy (load balanced)**
  - Avoid fetching duplicates repeatedly
- **Respect politeness and robots.txt**
  - Crawlers could potentially flood sites with requests for pages
  - use *politeness policies:* e.g., delay between requests to same web server
- **Handle crawling abnormality:**
  - Avoid getting stuck in traps
  - Tolerate faults with retry

More URL Scheduling Issues

- **Conflicting goals**
  - Big sites are crawled completely;
  - Discover and recrawl URLs frequently
    - Important URLs need to have high priority
      - What's best? Quality, fresh, topic coverage
    - Avoid/Minimize duplicate and spam
  - Revisiting for recently crawled URLs should be excluded to avoid the endless of revisiting of the same URLs.
- **Access properties of URLs to make a scheduling decision.**

/robots.txt

- **Protocol for giving spiders ("robots") limited access to a website**
  - www.robotstxt.org/
- **Website announces its request on what can(not) be crawled**
  - For a URL, create a file `robots.txt`
  - This file specifies access restrictions
  - Place in the top directory of web server.
    - E.g. www.cs.ucsb.edu/robots.txt
    - www.ucsb.edu/robots.txt

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:
```

More Robots.txt example

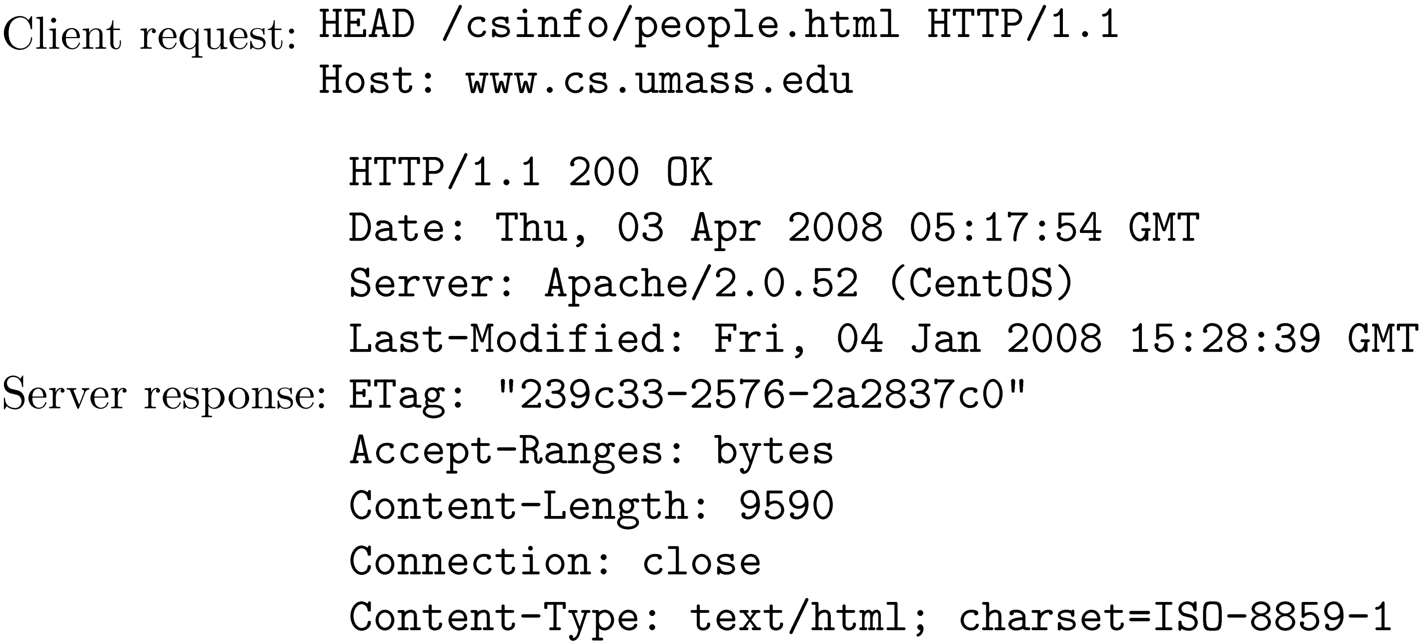
```
User-agent: *
Disallow: /private/
Disallow: /confidential/
Disallow: /other/
Allow: /other/public/

User-agent: FavoredCrawler
Disallow:

Sitemap: http://mysite.com/sitemap.xml.gz
```

Freshness

- **Web pages are constantly being added, deleted, and modified**
- **Web crawler must continually revisit pages it has already crawled to see if they have changed in order to maintain the *freshness* of the document collection**
- **Not possible to constantly check all pages**
  - Need to check important pages and pages that change frequently

Freshness

- **HTTP protocol has a special request type called HEAD that makes it easy to check for page changes**
  - returns information about page, not page itself
  - Information is not reliable. (e.g ~40+% incorrect)



```
Client request: HEAD /csinfo/people.html HTTP/1.1
                Host: www.cs.umass.edu

                  HTTP/1.1 200 OK
                  Date: Thu, 03 Apr 2008 05:17:54 GMT
                  Server: Apache/2.0.52 (CentOS)
                  Last-Modified: Fri, 04 Jan 2008 15:28:39 GMT
Server response:  ETag: "239c33-2576-2a2837c0"
                  Accept-Ranges: bytes
                  Content-Length: 9590
                  Connection: close
                  Content-Type: text/html; charset=ISO-8859-1
```

Focused Crawling

- **Attempts to download only those pages that are about a particular topic**
  - used by *vertical search* applications
  - E.g. crawl and collect technical reports and papers appeared in all computer science dept. websites
- **Rely on the fact that pages about a topic tend to have links to other pages on the same topic**
  - popular pages for a topic are typically used as seeds
- **Crawler uses *text classifier* to decide whether a page is on topic**

Where/what to modify in this architecture for a focused crawler?

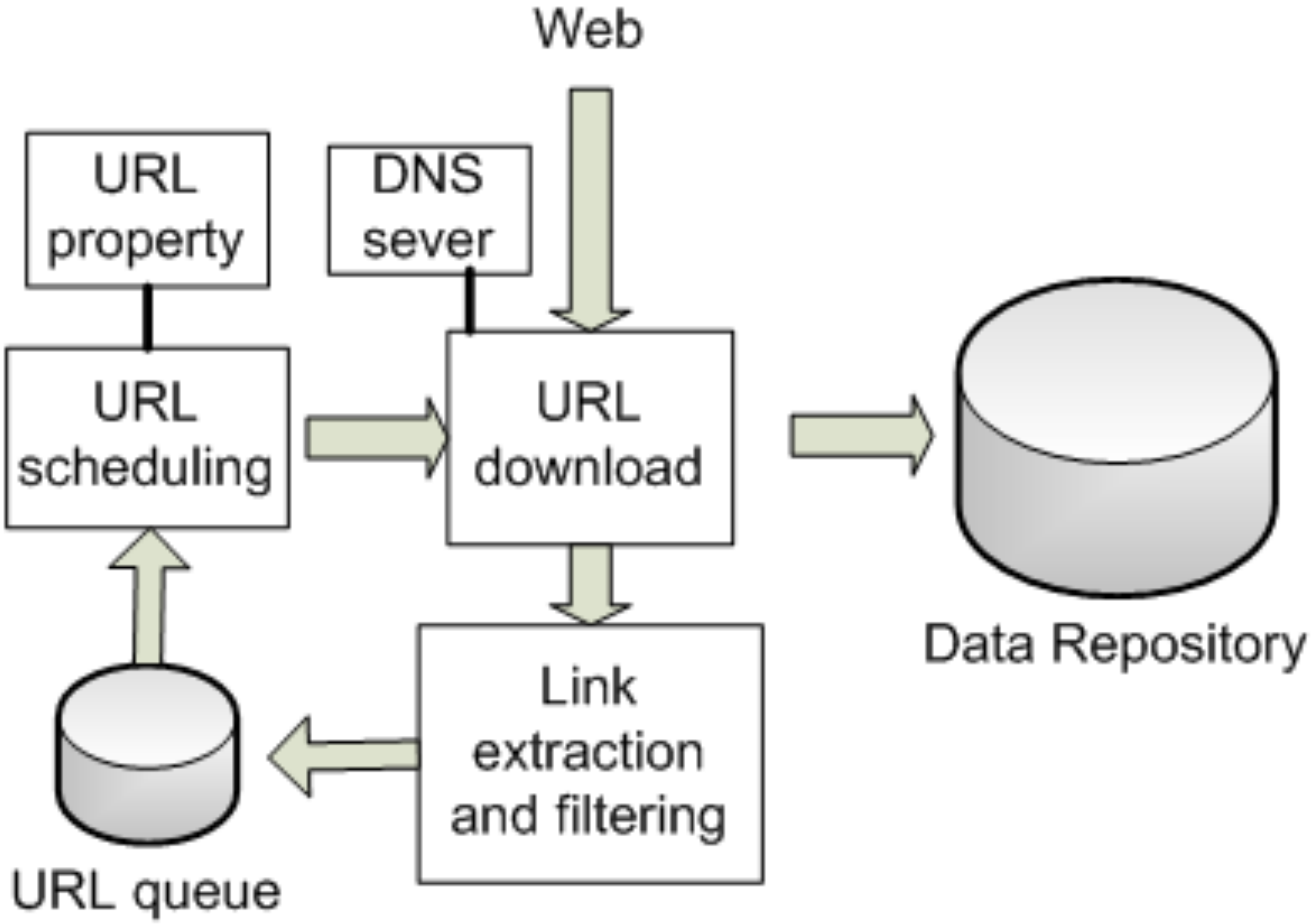


Table of Content

- **Basic crawling architecture and flow**
- **Schedule: Where to crawl**
  - Crawling control with robots.txt
  - Freshness
  - Focused crawling
- **Discover new URLs**
  - **Deep web, Sitemaps, & Data feeds**
- **Data representation and store**

Discover new URLs & Deepweb

- **Challenges to discover new URLs**
  - Bandwidth/politeness prevent the crawler from covering large sites fully.
  - Deepweb
- **Strategies**
  - Mining new topics/related URLs from news, blogs, facebook/twitters.
  - Idendify sites that tend to deliver more new URLs.
  - Deepweb handling/sitemaps
  - RSS feeds

Deep Web

- **Sites that are difficult for a crawler to find are collectively referred to as the *deep* (or *hidden*) *Web***
  - much larger than conventional Web
- **Three broad categories:**
  - private sites
    - no incoming links, or may require log in with a valid account
  - form results
    - sites that can be reached only after entering some data into a form
  - scripted pages
    - pages that use JavaScript, Flash, or another client-side language to generate links

Sitemaps

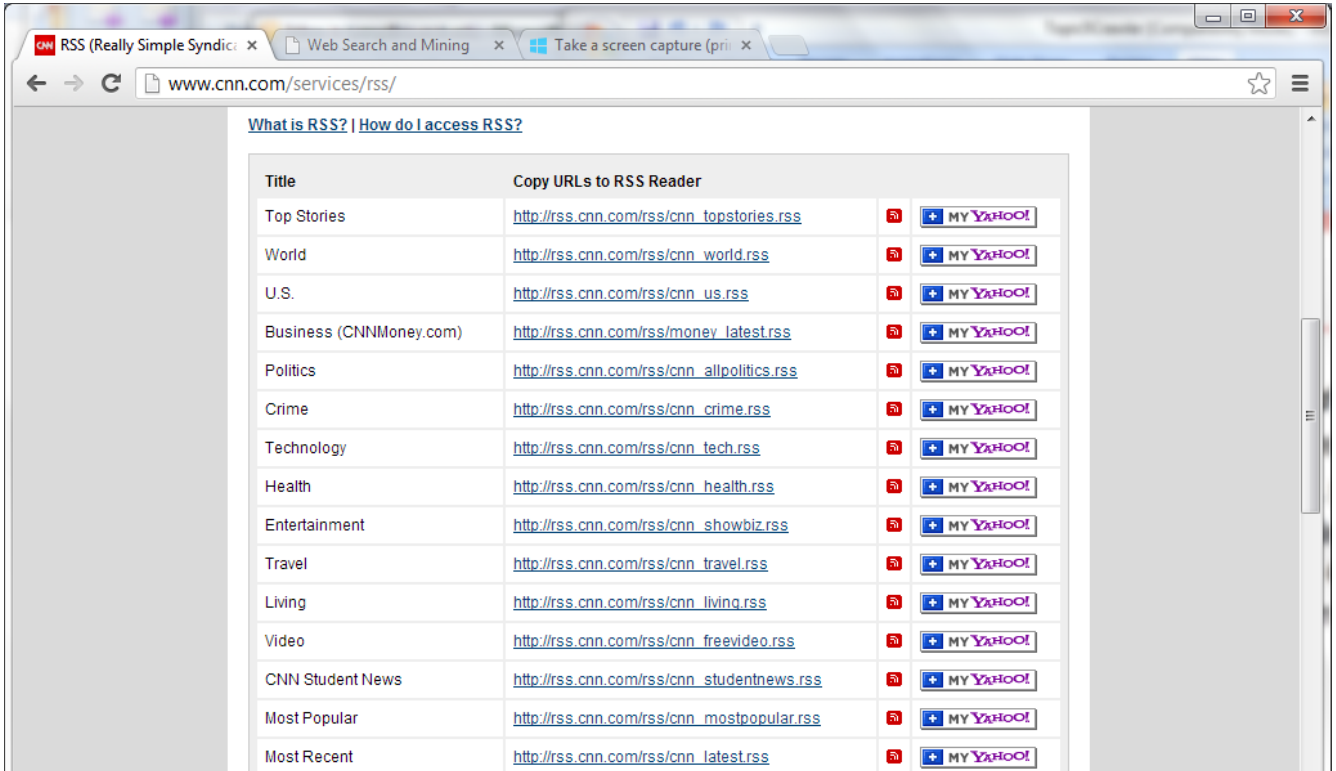
- Placed at the root directory of an HTML server.
  - For example, http://example.com/sitemap.xml.
- Sitemaps contain lists of URLs and data about those URLs, such as modification time and modification frequency
- Generated by web server administrators
- Tells crawler about pages it might not otherwise find
- Gives crawler a hint about when to check a page for changes

Sitemap Example

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
  <url>
    <loc>http://www.company.com/</loc>
    <lastmod>2008-01-15</lastmod>
    <changefreq>monthly</changefreq>
    <priority>0.7</priority>
  </url>
  <url>
    <loc>http://www.company.com/items?item=truck</loc>
    <changefreq>weekly</changefreq>
  </url>
  <url>
    <loc>http://www.company.com/items?item=bicycle</loc>
    <changefreq>daily</changefreq>
  </url>
</urlset>
```

Document Feeds

- **Many documents are *published on the web***
  - created at a fixed time and rarely updated again
  - e.g., news articles, blog posts, press releases, email
  - new documents found by examining the end of the feed


RSS (Really Simple Syndica
Web Search and Mining
Take a screen capture (pri
www.cnn.com/services/rss/

What is RSS? | How do I access RSS?

Title	Copy URLs to RSS Reader		
Top Stories	http://rss.cnn.com/rss/cnn_topstories.rss		MY YAHOO!
World	http://rss.cnn.com/rss/cnn_world.rss		MY YAHOO!
U.S.	http://rss.cnn.com/rss/cnn_us.rss		MY YAHOO!
Business (CNNMoney.com)	http://rss.cnn.com/rss/money_latest.rss		MY YAHOO!
Politics	http://rss.cnn.com/rss/cnn_allpolitics.rss		MY YAHOO!
Crime	http://rss.cnn.com/rss/cnn_crime.rss		MY YAHOO!
Technology	http://rss.cnn.com/rss/cnn_tech.rss		MY YAHOO!
Health	http://rss.cnn.com/rss/cnn_health.rss		MY YAHOO!
Entertainment	http://rss.cnn.com/rss/cnn_showbiz.rss		MY YAHOO!
Travel	http://rss.cnn.com/rss/cnn_travel.rss		MY YAHOO!
Living	http://rss.cnn.com/rss/cnn_living.rss		MY YAHOO!
Video	http://rss.cnn.com/rss/cnn_freevideo.rss		MY YAHOO!
CNN Student News	http://rss.cnn.com/rss/cnn_studentnews.rss		MY YAHOO!
Most Popular	http://rss.cnn.com/rss/cnn_mostpopular.rss		MY YAHOO!
Most Recent	http://rss.cnn.com/rss/cnn_latest.rss		MY YAHOO!

Document Feeds

- **Two types:**
  - A *push feed* alerts the subscriber to new documents
  - A *pull feed* requires the subscriber to check periodically for new documents
- **Most common format for pull feeds is called *RSS***
  - Really Simple Syndication, RDF Site Summary, Rich Site Summary, or ...
- **Examples**
  - CNN RSS newsfeed under different categories
  - Amazon RSS popular product feeds under different tags

RSS Example

```
<?xml version="1.0"?>
<rss version="2.0">
  <channel>
    <title>Search Engine News</title>
    <link>http://www.search-engine-news.org/</link>
    <description>News about search engines.</description>
    <language>en-us</language>
    <pubDate>Tue, 19 Jun 2008 05:17:00 GMT</pubDate>
    <ttl>60</ttl>

    <item>
      <title>Upcoming SIGIR Conference</title>
      <link>http://www.sigir.org/conference</link>
      <description>The annual SIGIR conference is coming!
        Mark your calendars and check for cheap
        flights.</description>
      <pubDate>Tue, 05 Jun 2008 09:50:11 GMT</pubDate>
      <guid>http://search-engine-news.org#500</guid>
    </item>
```

RSS Example

```
...
   <item>
      <title>New Search Engine Textbook</title>
      <link>http://www.cs.umass.edu/search-book</link>
      <description>A new textbook about search engines
        will be published soon.</description>
      <pubDate>Tue, 05 Jun 2008 09:33:01 GMT</pubDate>
      <guid>http://search-engine-news.org#499</guid>
    </item>
  </channel>
</rss>
```

RSS

- **A number of channel elements:**
  - Title
  - Link
  - description
  - ttl tag (time to live)
    - amount of time (in minutes) contents should be cached
- **RSS feeds are accessed like web pages**
  - using HTTP GET requests to web servers that host them
- **Easy for crawlers to parse**
- **Easy to find new information**

Table of Content

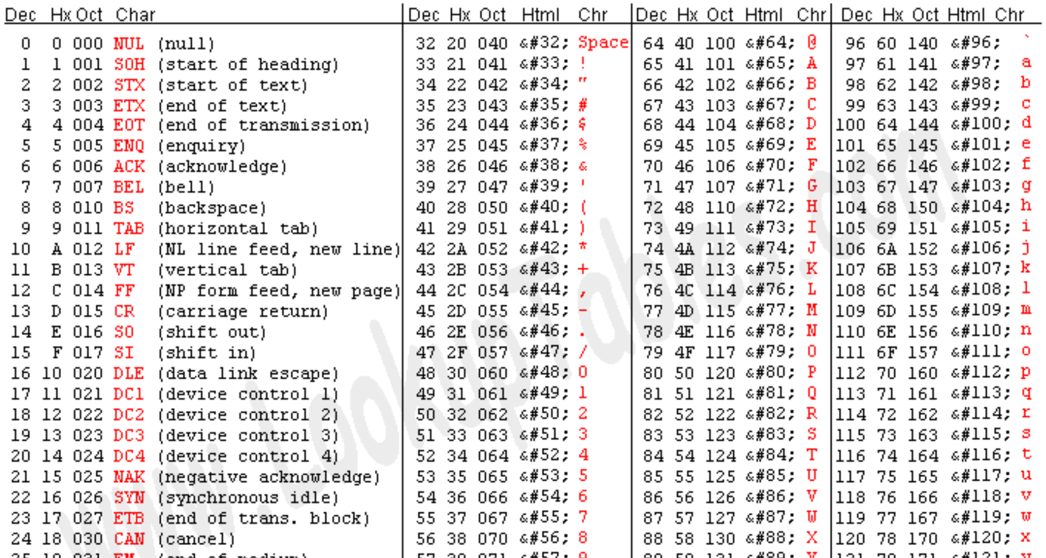
- **Crawling architecture and flow**
- **Scheduling: Where to crawl**
  - Crawling control with robots.txt
  - Freshness
  - Focused crawling
- **URL discovery**
  - **Deep web, Sitemaps, & Data feeds**
- **Data representation and store**

Conversion

- **Text is stored in hundreds of incompatible file formats**
  - e.g., raw text, RTF, HTML, XML, Microsoft Word, ODF, PDF
- **Other types of files also important**
  - e.g., PowerPoint, Excel
- **Typically use a conversion tool**
  - converts the document content into a tagged text format such as HTML or XML
  - retains some of the important formatting information

Character Encoding

- **A character encoding is a mapping between bits and glyphs**
  - Mapping from bits to characters on a screen
- **ASCII is basic character encoding scheme for English**
  - encodes 128 letters, numbers, special characters, and control characters in 7 bits



Dec	Hx	Oct	Char		Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL	(null)	32	20	040	&#32;	Space	64	40	100	&#64;	@	96	60	140	&#96;	`
1	1	001	SOH	(start of heading)	33	21	041	&#33;	!	65	41	101	&#65;	A	97	61	141	&#97;	a
2	2	002	STX	(start of text)	34	22	042	&#34;	"	66	42	102	&#66;	B	98	62	142	&#98;	b
3	3	003	ETX	(end of text)	35	23	043	&#35;	#	67	43	103	&#67;	C	99	63	143	&#99;	c
4	4	004	EOT	(end of transmission)	36	24	044	&#36;	$	68	44	104	&#68;	D	100	64	144	&#100;	d
5	5	005	ENQ	(enquiry)	37	25	045	&#37;	%	69	45	105	&#69;	E	101	65	145	&#101;	e
6	6	006	ACK	(acknowledge)	38	26	046	&#38;	&	70	46	106	&#70;	F	102	66	146	&#102;	f
7	7	007	BEL	(bell)	39	27	047	&#39;	'	71	47	107	&#71;	G	103	67	147	&#103;	g
8	8	010	BS	(backspace)	40	28	050	&#40;	(	72	48	110	&#72;	H	104	68	150	&#104;	h
9	9	011	TAB	(horizontal tab)	41	29	051	&#41;	)	73	49	111	&#73;	I	105	69	151	&#105;	i
10	A	012	LF	(NL line feed, new line)	42	2A	052	&#42;	*	74	4A	112	&#74;	J	106	6A	152	&#106;	j
11	B	013	VT	(vertical tab)	43	2B	053	&#43;	+	75	4B	113	&#75;	K	107	6B	153	&#107;	k
12	C	014	FF	(NP form feed, new page)	44	2C	054	&#44;	,	76	4C	114	&#76;	L	108	6C	154	&#108;	l
13	D	015	CR	(carriage return)	45	2D	055	&#45;	-	77	4D	115	&#77;	M	109	6D	155	&#109;	m
14	E	016	SO	(shift out)	46	2E	056	&#46;	.	78	4E	116	&#78;	N	110	6E	156	&#110;	n
15	F	017	SI	(shift in)	47	2F	057	&#47;	/	79	4F	117	&#79;	O	111	6F	157	&#111;	o
16	10	020	DLE	(data link escape)	48	30	060	&#48;	0	80	50	120	&#80;	P	112	70	160	&#112;	p
17	11	021	DC1	(device control 1)	49	31	061	&#49;	1	81	51	121	&#81;	Q	113	71	161	&#113;	q
18	12	022	DC2	(device control 2)	50	32	062	&#50;	2	82	52	122	&#82;	R	114	72	162	&#114;	r
19	13	023	DC3	(device control 3)	51	33	063	&#51;	3	83	53	123	&#83;	S	115	73	163	&#115;	s
20	14	024	DC4	(device control 4)	52	34	064	&#52;	4	84	54	124	&#84;	T	116	74	164	&#116;	t
21	15	025	NAK	(negative acknowledge)	53	35	065	&#53;	5	85	55	125	&#85;	U	117	75	165	&#117;	u
22	16	026	SYN	(synchronous idle)	54	36	066	&#54;	6	86	56	126	&#86;	V	118	76	166	&#118;	v
23	17	027	ETB	(end of trans. block)	55	37	067	&#55;	7	87	57	127	&#87;	W	119	77	167	&#119;	w
24	18	030	CAN	(cancel)	56	38	070	&#56;	8	88	58	130	&#88;	X	120	78	170	&#120;	x
25	19	031	EM	(end of medium)	57	39	071	&#57;	9	89	59	131	&#89;	Y	121	79	171	&#121;	y

Character Encoding

- **Major source of incompatibility**
- **Other languages can have many more glyphs**
  - e.g., Chinese has more than 40,000 characters, with over 3,000 in common use
- **Many languages have multiple encoding schemes**
  - e.g., CJK (Chinese-Japanese-Korean) family of East Asian languages, Hindi, Arabic
  - can’t have multiple languages in one file
- **Unicode developed to address encoding problems**

Unicode

- **Single mapping from numbers to glyphs**
  - attempts to include all glyphs in common use in all known languages
  - e.g., UTF-8, UTF-16, UTF-32

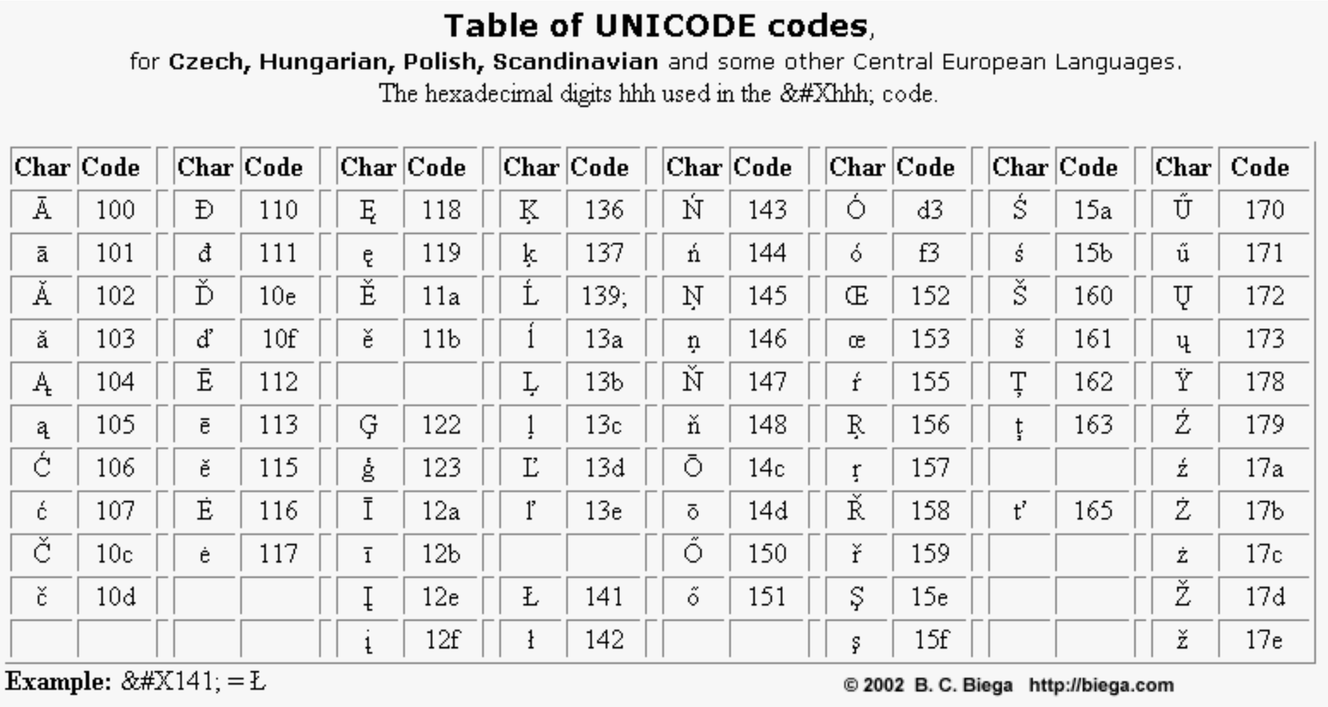


Table of UNICODE codes,
for **Czech, Hungarian, Polish, Scandinavian** and some other Central European Languages.
The hexadecimal digits hhh used in the &#Xhhh; code.

Char	Code	Char	Code	Char	Code	Char	Code	Char	Code	Char	Code	Char	Code	Char	Code
Ā	100	Đ	110	Ę	118	Ķ	136	Ń	143	Ó	d3	Ś	15a	Ű	170
ā	101	đ	111	ę	119	ķ	137	ń	144	ó	f3	ś	15b	ű	171
Ă	102	Ď	10e	Ě	11a	Ĺ	139;	Ņ	145	Œ	152	Š	160	Ų	172
ă	103	ď	10f	ě	11b	ĺ	13a	ņ	146	œ	153	š	161	ų	173
Ą	104	Ē	112			Ļ	13b	Ň	147	ŕ	155	Ţ	162	Ÿ	178
ą	105	ē	113	Ģ	122	ļ	13c	ň	148	Ŗ	156	ţ	163	Ź	179
Ć	106	ĕ	115	ģ	123	Ľ	13d	Ō	14c	ŗ	157			ź	17a
ć	107	Ė	116	Ī	12a	ľ	13e	ō	14d	Ř	158	ť	165	Ż	17b
Č	10c	ė	117	ī	12b			Ő	150	ř	159			ż	17c
č	10d			Į	12e	Ł	141	ő	151	Ş	15e			Ž	17d
				į	12f	ł	142			ş	15f			ž	17e

Example: &#X141; = Ł

© 2002 B. C. Biega http://biega.com

Software Internationalization with Unicode

- **Search software needs to be able to run for serving different international content**
  - compatibility & space saving
  - UTF-8 uses one byte for English (ASCII), as many as 4 bytes for some traditional Chinese characters
  - UTF-32 uses 4 bytes for every character
- **Many applications use UTF-32 for internal text encoding (fast random lookup) and UTF-8 for disk storage (less space)**

Example of Unicode

Decimal	Hexadecimal	Encoding			
0–127	0–7F	0xxxxxxx			
128–2047	80–7FF	110xxxxx	10xxxxxx		
2048–55295	800–D7FF	1110xxxx	10xxxxxx	10xxxxxx	
55296–57343	D800–DFFF	Undefined			
57344–65535	E000–FFFF	1110xxxx	10xxxxxx	10xxxxxx	
65536–1114111	10000–10FFFF	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

- Greek letter pi (π) is Unicode symbol number 960
  - In binary, 00000011 11000000 (3C0 in hexadecimal)
  - Final encoding is **110**01111 **10**000000 (CF80 in hexadecimal)

Storing the Documents

- **Many reasons to store converted document text**
  - saves crawling time when page is not updated
  - provides efficient access to text for snippet generation, information extraction, etc.
- **Data stores used for page repository**
  - Store many documents in large files, rather than each document in a file
    - avoids overhead in opening and closing files
    - reduces seek time relative to read time
- **Compound documents formats**
  - used to store multiple documents in a file
  - e.g., TREC Web

TREC Web Format

```
<DOC>
<DOCNO>WTX001-B01-10</DOCNO>
<DOCHDR>
http://www.example.com/test.html 204.244.59.33 19970101013145 text/html 440
HTTP/1.0 200 OK
Date: Wed, 01 Jan 1997 01:21:13 GMT
Server: Apache/1.0.3
Content-type: text/html
Content-length: 270
Last-modified: Mon, 25 Nov 1996 05:31:24 GMT
</DOCHDR>
<HTML>
<TITLE>Tropical Fish Store</TITLE>
Coming soon!
</HTML>
</DOC>
<DOC>
<DOCNO>WTX001-B01-109</DOCNO>
<DOCHDR>
http://www.example.com/fish.html 204.244.59.33 19970101013149 text/html 440
HTTP/1.0 200 OK
Date: Wed, 01 Jan 1997 01:21:19 GMT
Server: Apache/1.0.3
Content-type: text/html
Content-length: 270
Last-modified: Mon, 25 Nov 1996 05:31:24 GMT
</DOCHDR>
<HTML>
<TITLE>Fish Information</TITLE>
This page will soon contain interesting
information about tropical fish.
</HTML>
</DOC>
```

Text Compression

- **Text is highly redundant (or predictable)**
- **Compression techniques exploit this redundancy to make files smaller without losing any of the content**
- **Compression of indexes: a separate topic**
- **Popular algorithms can compress HTML and XML text by 80%**
  - e.g., DEFLATE (zip, gzip) and LZW (UNIX compress, PDF)
  - may compress large files in blocks to make access faster