

Web mining

The World Wide Web, Web Crawling





The Internet

The World Wide Web

Web crawlers

Conclusion





<http://www.opte.org/>





A stack of communication protocols, on top of each other.

Application	HTTP, FTP, SMTP, DNS	
Transport	TCP, UDP, ICMP	(sessions, reliability. . .)
Network	IP (v4, v6)	(routing, addressing)
Link	Ethernet, 802.11 (ARP)	(addressing local machines)
Physical	Ethernet, 802.11 (physical)	



■ Addressing machines and routing over the Internet

■ Two versions of the IP protocol on the Internet: IPv4 (very well spread) and IPv6 (not that well-spread yet)

- IPv4: 4-byte addresses assigned to each computer, e.g., 137.194.2.24. Institutions are given ranges of such addresses, to assign as they will.
- Problem: only 2^{32} possible addresses (actually, a large number of them cannot be assigned to new hosts, for multiple reasons). This means many hosts connected to the Internet do not have an IPv4 address and some network address translation (NAT) occurs.
- IPv6: 16-byte addresses; much larger address space! Addresses look like 2001:660:330f:2::18 (meaning 2001:0660:0330f:0002:0000:0000:0000:0018). Other nice features (multicast, autoconfiguration, etc.).





- One of the two main transport protocols used on IP, with UDP (Use Datagram Protocol)

- Contrarily to UDP, provides **reliable** transmission of data (acknowledgments)
- Data is divided into small **datagrams** that are sent over the network, and possibly reordered at the end point
- Like UDP, each TCP transmission indicates a source and a destination **port number** (between 0 and 65535) to distinguish it from other traffic
- A client usually select a **random** port number for establishing a connection to a **fixed** port number on a server
- The port number on a server conventionally identifies an **application protocol** on top of TCP/IP: 22 for SSH, 25 for SMTP, 110 for POP3...





- IPv4 addresses are **hard to memorize**, and a given service (e.g., a Web site) may **change** IP addresses (e.g., new Internet service provider)
- Even more so for IPv6 addresses!
- DNS: a UDP/IP-based protocol for associating human-friendly names (e.g., `www.google.com`, `weather.yahoo.com`) to IP addresses
- Hierarchical domain names: **com** is a top-level domain (TLD), **yahoo.com** is a subdomain thereof, etc.
- Hierarchical domain name resolution: **root servers** with fixed IPs know who is in charge of TLDs, servers in charge of a domain know who is in charge of a subdomain, etc.
- Nothing magic with **www.google.com**: just a subdomain of `google.com`.





The World Wide Web

Introduction

The Web: a market

HTML

HTTP

Web crawlers

Conclusion





The World Wide Web

Introduction

The Web: a market

HTML

HTTP

Web crawlers

Conclusion





Internet: physical network of computers (or **hosts**)
World Wide Web, Web, WWW: logical collection of **hyperlinked** documents

- **static** and **dynamic**
- **public** Web and **private** Webs
- each document (or **Web page**, or **resource**) identified by a URL





- 1969 ARPANET (the ancestor of the Internet)
- 1974 TCP (Vinton G. Cerf & Robert E. Kahn, Turing award winners 2004)
- 1990 World Wide Web, HTTP, HTML (Tim Berners-Lee, Robert Cailliau)
- 1993 Mosaic (the first public successful graphical browser, ancestor of Netscape)
- 1994 Yahoo! (David Filo, Jerry Yang)
- 1994 Foundation of the W3C
- 1995 Amazon.com, Ebay
- 1995 Internet Explorer
- 1995 AltaVista (Louis Monier, Michael Burrows)
- 1998 Google (Larry Page, Sergey Brin)
- 2001 Wikipedia (Jimmy Wales)
- 2004 Mozilla Firefox
- 2005 YouTube

Sources: [Electronic Software Publishing Corporation, 2008], [BBC, 2006]





https :// www.example.com :443 / path/to/doc ?name=foo&town=bar #para

scheme hostname port path query string fragment

scheme: way the resource can be accessed; generally **http** or **https**

hostname: **domain name** of a host (cf. DNS); hostname of a website may start with **www.**, but not a rule.

port: **TCP port**; defaults: 80 for **http** and 443 for **https**

path: **logical path** of the document

query string: optional additional parameters (dynamic documents)

fragment: optional **subpart** of the document

Relative URLs with respect to a **context** (e.g., the URL above):

/titi **https://www.example.com/titi**

tata **https://www.example.com/path/to/tata**



■ For content: HTML/XHTML, but also PDF, Word documents, text files, XML (RSS, SVG, MathML, etc.)...

- For presenting this content: CSS, XSLT
- For animating this content: JavaScript, AJAX, VBScript. . .
- For interaction-rich content: Flash, Java, Silverlight, ActiveX, `<canvas>` API. . .
- Multimedia content: images, sounds, videos. . .
- And on the server side: any programming language and database technology to serve this content, e.g., PHP, JSP, Java servlets, ASP, ColdFusion, etc.

Quite **complex to manage**! Being a Web developer nowadays requires mastering a lot of different technologies; designing a Web client requires being able to handle a lot of different technologies!



The World Wide Web

Introduction

The Web: a market

HTML

HTTP

Web crawlers

Conclusion





- Graphical browsers (cf. next slide)
- Text browsers: w3m, lynx, links (free software, Windows, Mac OS, Linux, Unix); rarely used nowadays
- Other browsers: audio browsers, etc.
- But also: spiders for siphoning a Web site, search engine crawlers (see later on), machine translation software. . .

A very large **variety** of clients! Web standards (mainly, HTML, CSS, HTTP) are supposed to describe what their interpretation of a Web page should be. In reality, more complex (tag soup).





	Engine	Share	Distribution
Chrome+Android	WebKit	32%	Windows, MacOS, Linux FS
Internet Explorer	Trident	30%	with Windows
Firefox	Gecko	20%	Windows, MacOS, Unix FS
Safari, inc. iOS	WebKit	10%	MacOS, Windows FC
Opera	Presto	4%	Windows, MacOS, Unix, mobiles FC

FC: free of charge (free as a beer)

FS: free software (free as a man)

Market shares: various sources, precise numbers hard to obtain. IE continually decreasing over the last years.

Trident remains the worst standard-compliant rendering engine.





- Google Chrome has known impressive success (only 4 years since its initial release)
- Versions of Internet Explorer 6 to 9 still all commonly used (especially in the enterprise world); IE6 is the browser coming with initial releases of Windows XP browser.
- Versions of Internet Explorer tied with versions of Windows (IE10 upcoming with Windows 8).
- Other browsers tend to have recent versions installed, but not always (esp., mobile browsers).



Server	Share	Distribution
Apache	65%	Windows, Mac OS, Linux, Unix FS
Microsoft IIS	15%	with some versions of Windows
nginx	10%	Windows, Mac OS, Linux, Unix FS
lighttpd	1%	Windows, Mac OS, Linux, Unix FS

- **Market share:** according to various studies, precise numbers do not really mean anything.
- Many large software companies have either their own Web server or their own modified version of Apache (notably, GFE/GWS for Google).
- nginx and lighttpd are lighter (i.e., less feature-rich, but faster in some contexts) than Apache.
- The versions of Microsoft IIS released with consumer versions of Windows are very limited.





- A large number of different search engines, with market shares **varying a lot** from country to country.
- At the world level:
 - **Google** vastly dominating (around 80% of the market; more than 90% market share in France!)
 - **Yahoo!+Bing** still resists to its main competitor (perhaps 10% of the market)
- In some countries, local search engines dominate the market (Baidu with 75% in China, Naver in Korea, Yahoo! Japan in Japan)





In July 2009, Microsoft and Yahoo! announced a major agreement:

- Yahoo! stops developing its own search engine (launched in 2003, after the buyouts of Inktomi and Altavista) and will use Bing instead;
- Yahoo! will provide the advertisement services used in Bing.

Operational, but does not concern Yahoo! Japan, which on the contrary uses Google as engine.





The World Wide Web

Introduction

The Web: a market

HTML

HTTP

Web crawlers

Conclusion

1 October 2012





Standardized by the W3C (World Wide Web Consortium) formed of industrials (Microsoft, Google, Apple. . .) and academic institutions (ERCIM, MIT, etc.)

- **open** format: possible processing by a wide variety of software and hardware
- **text** files with **tags**
- describes the **structure** and **content** of a document, focus on **accessibility**
- (theoretically) no presentation information (this is the role of CSS)
- no description of dynamic behaviors (this is the role of server-side languages, JavaScript, etc.)





- HTML is a language alternating text and tags (`<blabla>` or `</blabla>`)
 - Tags allow structuring each part of a document, and are used for instance by a browser to lay out the document.
- HTML files
 - are structured in two main parts: the header `<head> . . . </head>`) and the body `<body> . . . </body>`)
- In HTML, blanks (spaces, tabs, carriage returns) are generally equivalent and only serve to delimit words, tags, etc. The number of blanks does not matter.





Syntax: (opening and closing tag)

```
<tag attributes>content</tag>
```

or (element with no content)

```
<tag attributes>
```

- tag** keyword referring to some particular HTML **element**
- content** may contain text and other tags
- attributes** represent the various parameters associated with the element, as a list of `name="value"` or `name='value'`, separated by spaces (quotes are not always mandatory, but they become mandatory if `value` has “exotic” characters)






- Names of elements and attributes are usually written in lowercase, but `<head>` and `<HeAd>` are equivalent.
- Tags are opened and closed in the right order (`<i></i>` and not `<i></i>`).
- Strict rules specify which tags can be used inside which.
- Under some conditions, a tag can be implicitly closed, but these conditions are complex to describe.
- `<!--foobar-->` denotes a comment, which is not to be interpreted by a Web client.



```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html lang="en">
  <head>
    <!-- Header of the document -->
  </head>
  <body>
    <!-- Body of the document -->
  </body>
</html>
```

- The doctype declaration `<!DOCTYPE ...>` specify which HTML version is used.
- The language of the document is specified with the `lang` attribute of the main `<html>` tag.



 The **header** of a document is delimited by the tags

```
<head> ... </head> .
```

- The header contains **meta-informations** about the document, such as its title, encoding, associated files, etc. The two most important items are:
 - The character set of the page, usually at the **very beginning** of the header

```
<meta http-equiv="Content-Type"
      content="text/html; charset=utf-8">
```

- The title of the page (the only required item inside the header). This is the information displayed in the title bar of Web browsers.

```
<title>My great website</title>
```



character repertoire, assigning to each character, whatever its script or language, an integer number.

Examples

A	→	65		ε	→	949
é	→	233		ℵ	→	1488

Character set: concrete method for representing a Unicode character.

Examples (é)

iso-8859-1	11101001		only for some characters
utf-8	11000011	10101001	
utf-16	11101001	00000000	

utf-8 has the advantage of being able to represent all Unicode characters, in a way compatible with the legacy **ASCII** encoding.



■ `<body> ... </body>` tags delimit the **body** of a document.
The **body** is **structured** into sections, paragraphs, lists, etc.

■ 6 tags describe **sections**, by decreasing order of importance:

- `<h1>Title of the page</h1>`
- `<h2>Title of a main section</h2>`
- `<h3>Title of a subsection</h3>`
- `<h3>Title of a subsubsection</h3>`
- ...

■ `<p> ... </p>` tags delimit **paragraphs** of text. All text paragraphs should be delimited thusly.

■ Directly inside `<body> ... </body>` can only appear **block** elements: `<p>`, `<h1>`, `<form>`, `<hr>`, ``, `<table>` ... in addition to the `<div>` tag which denotes a block without precise semantics.



- What differentiates Web pages (hypertext pages) from normal documents: **links!**
- Introduced with `<a> ... `
- Navigating a link can bring to:
 - a resource on another server or another file of the same server
 - another part of the same document





Links are made using the `href` attribute of the `<a>` tag, whose content will be the link:

```
<a href="http://www.cnrs.fr/">  
    
</a>  
  
<a href="bio/indexbioinfo.html">Bioinformatics</a>
```





■ **Anchors** serve to reach a precise point in the document.

- They are defined, either on an existing tag by using the `id` attribute, or with an `` :

```
<h3 id="tutorials">Tutorials</h3>
<a id="tutorials">
```

- Then, one can link to this anchor:

```
<a href="#tutorials">tutorials</a>
<a href="http://www.w3.org/#tutorials">tutorials</a>
```

- Commonly, the old `` syntax is used.





HTML 4.01 (1999) strict (as described earlier) and transitional

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"  
"http://www.w3.org/TR/html4/strict.dtd">
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"  
"http://www.w3.org/TR/html4/loose.dtd">
```

- XHTML 1.0 (2000) strict and transitional
- XHTML 1.1 and XHTML 2.0: mostly a failure, unusable and unused in today's Web
- HTML 5: future standard, in development, partly implemented, continuously updated

```
<!DOCTYPE html>
```





- A lot of HTML documents on the Web date back from before HTML 4.01
- In practice: many Web pages do not respect any standards at all (with or without doctype declarations) \Rightarrow browsers do not respect these standards \Rightarrow tag soup!
- When dealing with pages from the real Web, necessary to use all sorts of heuristics to interpret a Web page.



XHTML: an XML format

Tags without content ``, are written `` in XHTML.

- Some elements can be left unclosed in HTML (` one two `), but closing is mandatory in XHTML.
- Attribute values can be written without quotes (``) in HTML, quotes are required in XHTML.
- Element and attribute names are not case-sensitive in HTML (`<HTML lang=fr>`), but are in XHTML (everything must be in lowercase).
- Attributes `xmlns` and `xml:lang` on the `<html>` tag in XHTML.
- And some other small subtleties...





XHTML 2.0: initiative of the W3C, incompatible with HTML 4.01/XHTML 1.0, major changes

- HTML 5: initiative of browser developers, compatible with HTML 4.01/XHTML 1.0, incremental but numerous changes
- XHTML 2.0 abandoned in July 2009
- HTML 5 features have appeared in recent browsers (Internet Explorer 9 included)
- HTML 5 offers the choice between syntactic conventions inherited from both HTML 4.01 and XHTML
- New features: 2D drawing (`<canvas>`), multimedia (`<audio>` , `<video>`), better structuring elements (`<section>` , `footer`), etc.





The World Wide Web

Introduction

The Web: a market

HTML

HTTP

Web crawlers

Conclusion

1 October 2012





Application protocol at the basis of the World Wide Web

- Latest and most widely used version: HTTP/1.1

- Client **request**:

```
GET /MarkUp/ HTTP/1.1  
Host: www.w3.org
```

- Server **response**:

```
HTTP/1.1 200 OK  
...  
Content-Type: text/html; charset=utf-8  
  
<!DOCTYPE html ...> ...
```

- Two main HTTP **methods**: GET and POST (HEAD is also used in place of GET, to retrieve meta-information only).
- Additional headers, in the request and the response
- Possible to send parameters in the request (key/value pairs).



■ Simplest type of request.

■ Possible parameter are sent at the end of a URL, after a ‘?’

- Not applicable when there are too many parameters, or when their values are too long.
- Method used when a URL is directly accessed in a browser, when a link is followed, and for some forms.

Example (Google query)

URL: `http://www.google.com/search?q=hello`

Corresponding HTTP GET request:

```
GET /search?q=hello HTTP/1.1
```

```
Host: www.google.com
```





- Method only used for submitting forms.

Example

```
POST /php/test.php HTTP/1.1
```

```
Host: www.w3.org
```

```
Content-Type: application/x-www-form-urlencoded
```

```
Content-Length: 100
```

```
type=search&title=The+Dictator&format=long&country=US
```





- By default, parameters are sent (with GET or POST) in the form: `name1=value1&name2=value2`, and special characters (accented characters, spaces...) are replaced by codes such as `+`, `%20`. This way of sending parameters is called `application/x-www-form-urlencoded`.
- For the POST method, another heavier encoding can be used (several lines per parameter), similar to the way emails are built: mostly useful for sending large quantity of information. Encoding named `multipart/form-data`.





- The HTTP response always starts with a **status code** with three digits, followed by a human-readable message (e.g., 200 OK).
- The first digit indicates the class of the response:
 - 1 Information
 - 2 Success
 - 3 Redirection
 - 4 Client-side error
 - 5 Server-side error





200	OK
301	Permanent redirection
302	Temporary redirection
304	No modification
400	Invalid request
401	Unauthorized
403	Forbidden
404	Not found
500	Server error



■ Different **domain names** can refer to the same IP address, i.e., the same physical machine (e.g., `www.google.fr` and `www.google.com`)

- When a machine is contacted by TCP/IP, it is through its **IP address**
- No *a priori* way to know which precise domain name to contact
- In order to serve different content according to the domain name (**virtual host**): header `Host:` in the request (only header really required)

Example

```
GET /search?hl=fr&q=hello HTTP/1.1
Host: www.google.fr
```





- The browser behaves differently depending on the **content type** returned: display a Web page with the layout engine, display an image, load an external application, etc.
- **MIME** classification of content types (e.g., image/jpeg, text/plain, text/html, application/xhtml+xml, application/pdf etc.)
- For a HTML page, or for text, the browser must also know what **character set** is used (this has precedence over the information contained in the document itself)
- Also returned: the content length (can be used to display a progress bar)

Example

HTTP/1.1 200 OK

Content-Type: text/html; charset=UTF-8

Content-Length: 3046




- Web clients and servers can identify themselves with a character string
- Useful to serve **different content** to different browsers, detect robots...
- ... but any client can say it's any other client!
- Historical confusion on naming: all common browsers identify themselves as Mozilla!

Example

```
User-Agent: Mozilla/5.0 (X11; U; Linux x86_64; fr;
rv:1.9.0.3) Gecko/2008092510 Ubuntu/8.04 (hardy)
Firefox/3.0.3
```

```
Server: Apache/2.0.59 (Unix) mod_ssl/2.0.59 OpenSSL/0.9.8e
PHP/5.2.3
```



 HTTP allows for protecting access to a Web site by an **identifier** and a **password**

- Attention: (most of the time) the password goes through the network unencrypted (but for instance, just encoded in Base64, revertible encoding)
- **HTTPS** (variant of HTTP that includes encryption, cryptographic authentication, session tracking, etc.) can be used instead to transmit sensitive data

Example

GET ... HTTP/1.1

Authorization: Basic dG90bzip0aXRp



- A Web client can specify to the Web server:
 - the **content type** it can process (text, images, multimedia content), with preference indicators
 - the **languages** preferred by the user
- The Web server can thus propose different file formats, in different languages.
- In practice, content negotiation on the language works, and is used, but content negotiation on file types does not work because of bad default configuration of some browsers.

Example

```
Accept: text/html,application/xhtml+xml,application/xml;  
q=0.9,*/*;q=0.8
```

```
Accept-Language: fr,fr-fr;q=0.8,en-us;q=0.5,en;q=0.3
```

- Information, as key/value pairs, that a Web client keeps and retransmits with each HTTP request (for a given domain name).
- Can be used to keep information on a user as she is visiting a Web site, between visits, etc.: electronic cart, identifier, and so on.
- Practically speaking, most often only stores a **session identifier**, connected, on the server side, to all session information (connected or not, user name, data...)
- Simulates the notion of session, absent from HTTP itself

Example

```
Set-Cookie: session-token=RJYBsG//azkfZrRazQ3SPQhlo1FpkQka2;  
path=/; domain=.amazon.de;  
expires=Fri Oct 17 09:35:04 2008 GMT
```

```
Cookie: session-token=RJYBsG//azkfZrRazQ3SPQhlo1FpkQka2
```



A client can ask for downloading a page only if it has been modified since some given date.

- Most often not applicable, the server giving rarely a reliable last modification date (difficult to obtain for dynamically generated content!).

Example

```
If-Modified-Since: Wed, 15 Oct 2008 19:40:06 GMT
```

```
304 Not Modified
```

```
Last-Modified: Wed, 15 Oct 2008 19:20:00 GMT
```





- When a Web browser follows a link or submits a form, it transmits the originating URL to the destination Web server.
- Even if it is not on the same server!

Example

Referer: `http://www.google.fr/`



The World Wide Web

Web crawlers

- Discovering new URLs

- Identifying duplicates

- Crawling architecture

- Recrawling URLs?

Conclusion

The World Wide Web

Web crawlers

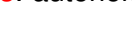
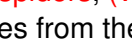
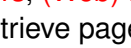
- Discovering new URLs

- Identifying duplicates

- Crawling architecture

- Recrawling URLs?

Conclusion



- **crawlers, (Web) spiders, (Web) robots**: autonomous user agents that retrieve pages from the Web

- **Basics of crawling:**

1. Start from a given URL or set of URLs
2. Retrieve and process the corresponding page
3. Discover new URLs (cf. next slide)
4. Repeat on each found URL

- No real termination condition (virtual unlimited number of Web pages!)

- **Graph-browsing** problem

deep-first: not very adapted, possibility of being lost in **robot traps**

breadth-first

combination of both: breadth-first with limited-depth deep-first on each discovered website





From HTML pages:

- hyperlinks `...`
 - media `` `<embed src="...">`
`<object data="...">`
 - frames `<frame src="...">` `<iframe src="...">`
 - JavaScript links `window.open("...")`
 - etc.
- Other hyperlinked content (e.g., PDF files)
 - Non-hyperlinked URLs that appear anywhere on the Web (in HTML text, text files, etc.): use regular expressions to extract them
 - Referrer URLs
 - Sitemaps [sitemaps.org, 2008]



- The Web is infinite! Avoid robot traps by putting depth or page number **limits** on each Web server
- Focus on **important** pages [Abiteboul et al., 2003] (cf. lecture on the Web graph)
- Web servers under a list of **DNS domains**: easy filtering of URLs
- A given topic: **focused crawling** techniques [Chakrabarti et al., 1999, Diligenti et al., 2000] based on classifiers of Web page content and predictors of the interest of a link.
- The national Web (cf. **public deposit**, national libraries): what is this? [Abiteboul et al., 2002]
- A given Web site: what is a Web site? [Senellart, 2005]



The World Wide Web

Web crawlers

Discovering new URLs

Identifying duplicates

Crawling architecture

Recrawling URLs?

Conclusion



A **hash function** is a deterministic mathematical function transforming objects (numbers, character strings, binary. . .) into fixed-size, seemingly random, numbers. The more random the transformation is, the better.

Example

Java hash function for the `String` class:

$$\sum_{i=0}^{n-1} s_i \times 31^{n-i-1} \bmod 2^{32}$$

where s_i is the (Unicode) code of character i of a string s .





Problem

Identifying duplicates or near-duplicates on the Web to prevent multiple indexing

trivial duplicates: same resource at the same **canonized** URL:

`http://example.com:80/toto`

`http://example.com/titi/../toto`

exact duplicates: identification by **hashing**

near-duplicates: (timestamps, tip of the day, etc.) more complex!





Edit distance. Count the **minimum number of basic modifications** (additions or deletions of characters or words, etc.) to obtain a document from another one. Good measure of similarity, and can be computed in $O(mn)$ where m and n are the size of the documents. But: **does not scale** to a large collection of documents (unreasonable to compute the edit distance for every pair!).

Shingles. Idea: two documents similar if they mostly share the same **succession of k -grams** (succession of tokens of length k).

Example

I like to watch the sun set with my friend.

My friend and I like to watch the sun set.

$S = \{i \text{ like, like to, my friend, set with, sun set, the sun, to watch, watch the, with my}\}$

$T = \{\text{and i, friend and, i like, like to, my friend, sun set, the sun, to watch, watch the}\}$





Similarity: **Jaccard coefficient** on the set of shingles:

$$J(S, T) = \frac{|S \cap T|}{|S \cup T|}$$

- Still **costly to compute!** But can be approximated as follows:
 1. Choose N **different hash functions**
 2. For each hash function h_i and each set of shingles $S_k = \{s_{k1} \dots s_{kn}\}$, store $\phi_{ik} = \min_j h_i(s_{kj})$
 3. Approximate $J(S_k, S_l)$ as the **proportion** of ϕ_{ik} and ϕ_{il} that are equal
- Possibly to repeat in a hierarchical way with **super-shingles** (we are only interested in **very** similar documents)



The World Wide Web

Web crawlers

Discovering new URLs

Identifying duplicates

Crawling architecture

Recrawling URLs?

Conclusion



Standard for robot exclusion: **robots.txt** at the root of a Web server [Koster, 1994].

```
User-agent: *  
Allow: /searchhistory/  
Disallow: /search
```

- Per-page exclusion (*de facto* standard).

```
<meta name="ROBOTS" content="NOINDEX,NOFOLLOW">
```

- Per-link exclusion (*de facto* standard).

```
<a href="toto.html" rel="nofollow">Toto</a>
```

- Avoid **Denial Of Service** (DOS), wait 100ms/1s between two repeated requests to the same Web server

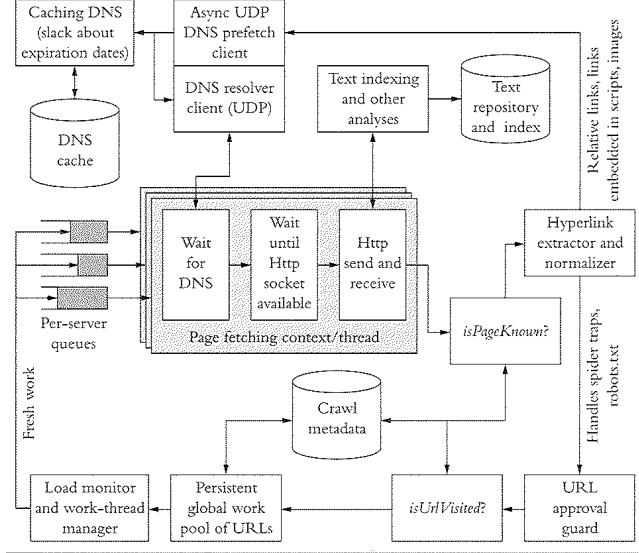




Network delays, waits between requests:

- **Per-server queue** of URLs
- Parallel processing of requests to different hosts:
 - **multi-threaded** programming
 - **asynchronous** inputs and outputs (`select`, classes from `java.util.concurrent`): less overhead
- Use of **keep-alive** to reduce connexion overheads





The World Wide Web

Web crawlers

Discovering new URLs

Identifying duplicates

Crawling architecture

Recrawling URLs?

Conclusion



- Content on the Web **changes**
- Different **change rates**:
 - online newspaper main page: every hour or so
 - published article: virtually no change
- **Continuous** crawling, and identification of change rates for **adaptive** crawling: how to know the **time of last modification** of a Web page?



Two timestamping mechanism in HTTP: **entity tags** and **modification**



ETag. Potentially provided with all requests:

ETag: "497bef-1fcb-47f20645"

Last-Modified: Tue, 01 Apr 2008 09:54:13 GMT

Etag: unique identifier for the provided document, should change if the document changes; can be used in requests with If-Match and If-None-Match.

Last-Modified: last modification time; can be used in requests with If-Modified-Since and If-Unmodified-Since.

- Information generally provided and very reliable for static content (often includes media files in CMS).
- Information hardly ever provided (or with dummy *now* date) for dynamic content, CMS, etc.





Two additional (redundant) items of **freshness information**, for caches and proxies:

```
Cache-Control: max-age=60, private  
Expires: Tue, 01 Apr 2008 13:25:55 GMT
```

max-age: maximum time in second a document remains fresh.

Expires: time when a document stops being fresh.

- Often provided. . .
- . . . but with 0 or very low expiration delay.
- \Rightarrow does not give any interesting information.





Very frequent in CMS:

- either as a **global** timestamps (*Last modified:*);
 - or on **individual** items: news stories, blogs, etc. (is the global timestamp the max of the individual ones?);
 - possibly also in meta-data on the Web page: comments, Dublin Core <meta> tags.
- Quite easy to identify and extract from the Web page (keywords, date recognition).
 - Informal: sometimes partial (no time indication), often without timezone.
 - Might not always be trustworthy.



Files of other types than HTML may have **semantic** timestamping

Mechanisms:

PDF, Office suite documents, etc.: both **creation** and **modification** date available in metadata. Quite reliable.

RSS feeds: reliable **semantic** timestamps.

Images, Sounds: **EXIF** (or similar) metadata. Not always reliable, and the capture date of a picture may have nothing in common with its publication date.

Semantic external content used for dating a HTML Web page:

- Possibility of mapping a **RSS feed** to Web page content
- **Sitemaps** provided by the Web site owner. Allows for providing both timestamps and change rate indications (*hourly, monthly...*), but these functionalities are not often used. A few CMS produce all of this: **ideal case!**





1. Check HTTP timestamp.
2. Check content timestamp.
3. Compare a hash of the page with a stored hash.
4. Non-significant differences (ads, fortunes, request timestamp):
 - only hash text content, or “useful” text content;
 - compare distribution of n -grams (shingling);
 - or even compute edit distance with previous version.

Adapting strategy to each different archived website?





The Internet

The World Wide Web

Web crawlers

Conclusion

1 October 2012





What you should remember

- The Web is **not the same thing** as Internet!
- **Variety** of protocols, languages, technologies used on the Web.
- Crawling as a **graph-browsing** problem.
- **Shingling** for identifying duplicates.
- Numerous **engineering issues** in building a Web-scale crawler.





- Wget, a simple yet effective Web spider (free software)
- Heritrix, a Web-scale highly configurable Web crawler, used by the Internet Archive (free software)
- HTML Parser, TagSoup: Java libraries for parsing real-world Web pages

To go further

- A good textbook [Chakrabarti, 2003]
- Main references:
 - HTML 4.01 recommendation [W3C, 1999]
 - HTTP/1.1 RFC [IETF, 1999]



Serge Abiteboul, Grégory Cobena, Julien Masanès, and Gerald Sedrati. A first experience in archiving the French Web. In *Proc. ECDL*, Roma, Italie, September 2002.

Serge Abiteboul, Mihai Preda, and Gregory Cobena. Adaptive on-line page importance computation. In *Proc. WWW*, May 2003.

BBC. Fifteen years of the web.

<http://news.bbc.co.uk/2/hi/technology/5243862.stm>, 2006.
Accessed March 2009.

Soumen Chakrabarti. *Mining the Web: Discovering Knowledge from Hypertext Data*. Morgan Kaufmann, San Fransisco, USA, 2003.

Soumen Chakrabarti, Martin van den Berg, and Byron Dom. Focused crawling: A new approach to topic-specific Web resource discovery. *Computer Networks*, 31(11–16):1623–1640, 1999.

Michelangelo Diligenti, Frans Coetzee, Steve Lawrence, C. Lee Giles, and Marco Gori. Focused crawling using context graphs. In *Proc. VLDB*, Cairo, Egypt, September 2000.

Electronic Software Publishing Corporation. Internet & World Wide Web history. http://www.elsop.com/wrc/h_web.htm, 2008. Accessed March 2009.

IETF. Request For Comments 2616. Hypertext transfer protocol—HTTP/1.1. <http://www.ietf.org/rfc/rfc2616.txt>, June 1999.

Martijn Koster. A standard for robot exclusion. <http://www.robotstxt.org/orig.html>, June 1994.

Pierre Senellart. Identifying Websites with flow simulation. In *Proc. ICWE*, pages 124–129, Sydney, Australia, July 2005.

sitemaps.org. Sitemaps XML format. <http://www.sitemaps.org/protocol.php>, February 2008.

W3C. HTML 4.01 specification, September 1999. <http://www.w3.org/TR/REC-html40/>.



Par le téléchargement ou la consultation de ce document, l'utilisateur accepte la licence d'utilisation qui y est attachée, telle que détaillée dans les dispositions suivantes, et s'engage à la respecter intégralement.

La licence confère à l'utilisateur un droit d'usage sur le document consulté ou téléchargé, totalement ou en partie, dans les conditions définies ci-après et à l'exclusion expresse de toute utilisation commerciale.

Le droit d'usage défini par la licence autorise un usage à destination de tout public qui comprend :

- le droit de reproduire tout ou partie du document sur support informatique ou papier,
- le droit de diffuser tout ou partie du document au public sur support papier ou informatique, y compris par la mise à la disposition du public sur un réseau numérique,
- le droit de modifier la forme ou la présentation du document,
- le droit d'intégrer tout ou partie du document dans un document composite et de le diffuser dans ce nouveau document, à condition que :
 - L'auteur soit informé.

Les mentions relatives à la source du document et/ou à son auteur doivent être conservées dans leur intégralité.

Le droit d'usage défini par la licence est personnel et non exclusif.

Tout autre usage que ceux prévus par la licence est soumis à autorisation préalable et expresse de l'auteur : sitopedago@telecom-paristech.fr

