Chapter 6

Proposals for Web Server Cooperation

As the number of publicly available Web pages increases, the problem of keeping a search engine index up-to-date with changes becomes increasingly more difficult [LG99], and it is common to find several pages out-dated in current search engines [SS04]. This makes things more difficult for the user who seeks information and affects the image of the search engine, but this also has costs for the Web sites that are misrepresented, if we consider the whole user experience.

If a user searches for a keyword on a search engine, and then chooses a page from the search results that no longer exists, or that contains material that is currently irrelevant for the user's information need, the user will be frustrated with both the search engine *and the Web site* for not finding this information. There is also an opportunity cost related to these visitors: maybe the information they want was moved to another page in the same website and the search engine was not aware of the change. In this case, it would be better for the Web site to inform the search engine of the update.

Web crawlers can use an important amount of network and processor resources from the Web server, especially if they do not follow existing rules of "good behavior" [Kos95]. Web crawlers tend to visit many more pages than humans, and they request them very fast, normally with 10 to 30 seconds between visits; so they are believed to account for at least 16% of the requests [MAR+00]. Many of the requests are to unmodified resources, and can be avoided in certain schemes if the server informs the crawler about which resources have not been modified since its last visit.

Hence, an Web site administrator has incentives to improve the representation of his Web site in the search engine's index and to prevent unnecessary visits from crawlers. The mechanism for accomplishing this is what we call a *cooperation* scheme between the Web server and the Web crawler.

In this chapter we show some existing techniques for cooperation, and we propose new ones; some of the techniques shown here were not designed for this specific purpose but they can be adapted. We also present a discussion about the relative merits of each technique. Finally, we implement one of them in the WIRE crawler.

The next section presents general aspects about cooperation schemes, Section 6.2 presents polling-based schemes, and Section 6.3 presents interruption-based schemes. Section 6.4 shows a relative comparison of costs. Section 6.5 describes how a cooperation scheme was implemented in the WIRE crawler. The last section presents the conclusions and future work.

Portions of this chapter were presented in [Cas03].

6.1 Cooperation schemes

The standard HTTP transaction follows the request-response paradigm: a client requests a page from a Web server, and the Web server responds with the page, as depicted in Figure 6.1. This transaction involves meta-data (information about the page) that is downloaded along with the actual page contents.

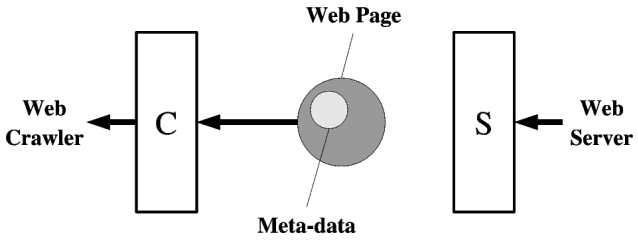

Figure 6.1: Schematic drawing of a normal transaction between a Web crawler ("C" on the left) and a Web server ("S" on the right). The large, dark circle in the middle represents a Web page and the small, light circle represents its meta-data. The arrow from the Web page to the Web crawler indicates that the Web crawler initiates the connection. We use this pictorial representation in the next two figures.

The cooperation schemes we study in this thesis can be divided into two groups: *polling* and *interrupt*.

Polling (or *pull*) schemes: the Web crawler periodically requests data from the Web server based on a scheduling policy. These requests check if a page have been changed, and then download the page.

Interrupt (or *push*) schemes: in this case the Web server begins a transaction with the search engine whenever there is an event. These events can happen when one or multiple pages are updated, based on server policies. The search engines must subscribe to the servers from which they want to receive events. This is similar to the relationship between the main processor and a hardware device (network card, scanner, etc.) in a modern computer.

Note that polling becomes equivalent to an interrupt when the polling period tends to zero; but the usage of resources at both ends of the polling line increase at the same time.

In this thesis, we study several cooperation schemes, which are summarized on Table 6.1.

Transfered data	Polling version	Interrupt version
Meta-data	Serve meta-data of content	Send meta-data of updates
Differences of site	Serve differences of content	Send differences of content
Pages	Serve pages only if modified	Send changed pages
Batches of pages	Serve many pages per request	Send batch update
Site	Serve entire site compressed	Send entire site
Mixed	Filtering interface	Remote agent

Table 6.1: List of cooperation schemes analyzed in this thesis. All of them have two versions: polling (poll) and interrupt (push).

Before we get into the details of each scheme, there are some issues we must mention that are almost independent of the scheme used:

Compression can be used for decreasing transmission cost at the expense of using more processing power on the server side. On the current Web, compression is used for transferring most images –because they are usually stored in a compressed format– but normally it is not applied for textual Web pages. The HTTP/1.0 protocol considers requests of compressed bodies using the `Accept-encoding` header, so compressing Web-pages can be used but it is usually left to the communication between the ISP and the final user. Compression can be used with complete Web pages, bundles of Web pages and their resources, or Web page differences [**?**] (more details in Section 6.2).

Privacy issues arise when the crawler has access to information in the server that was not meant to be public. This may sound strange, but in practice when using a Web crawler it is possible to download files that are linked by mistake or private directories that allow a virtual listing; we have even found complete plain-text password files!. Many Web site administrators mistakenly believe that by not publishing the URL of a Web page they can keep the page private. This is a common practice, so most Web site administrators are very reluctant to provide access for clients to list the contents of directories. Note that if users follow an external link from one of these "private" pages, their browser will inform the referrer to the next Web site, and this referrer could be logged and inspected so it is pointless to try to keep unknown URLs private.

Index update capabilities are very reduced in global-scale Web search engines: constraints in terms of disk space are the most important limitation, so it is not always possible to store a complete copy of the downloaded pages. This can lead to some difficulties; for instance, on a standard inverted index, removing a page or updating a paragraph without having the complete text can be impossible or very time-consuming. Also, in many cases updating batches of pages is preferred to updating single pages to reduce the overall processing cost.

Search engine "spamming" occurs whenever Web site administrators try to get undeserved high rat-

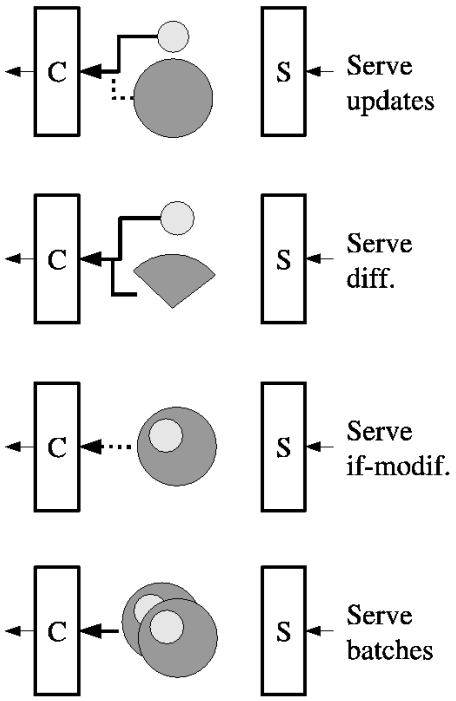

Figure 6.2: Diagrams of polling-based cooperation. The arrow between the Web crawler "C" and the Web server "S" represents who initiates the connection. The small, white circle represents meta-data and the large, gray circle represents the contents of the page.

ings in search engines. Data provided by Web servers cannot be trusted completely, for instance, in terms of self-asserted frequency of updates of the pages or local page importance. Web crawlers used by most search engines are interested only in some of the pages of a Web site, and the decision of which pages must be added to the index should be left to the search engine. In notification schemes, a degree of trust can be established, e.g.: if a Web site sends update notifications, but when pages are inspected by the Web crawler they have not changed, then the search engine can ignore further notifications from that Web site for a period of time.

Structure and HTML markup used in a Web site affects the visibility of its pages by search engine crawlers. Information that is accessible only through forms is, in general, difficult to gather for Web crawlers; this is called the "hidden Web" [RGM01]. Web sites could attract more visitors if they provide a crawler-friendly interface for this data to be indexed by search engines.

6.2 Polling-based cooperation

In all these cases, the Web crawler queries the Web server with certain periodicity; the cooperation schemes discussed in this section are depicted in Figure 6.2.

Serve meta-data of updates. A file containing last-modification data (and probably file size and path) is served. This file can contain a description of many pages on the Web site. In the case of single files, the HTTP protocol provides `HEAD` requests that are responded with meta-data about the requested object. Multiple `HEAD` requests can be pipelined, but this is not as efficient as serving a concise file. Examples: the Distribution and Replication Protocol (DRP) [vHGH+97], the proposal by Brandman *et al.* [BCGMS00] in which files containing information about changes are served, and the proposal by Buzzi [Buz03] that includes information obtained from the access log files. RDF [LS99] also includes the possibility of informing time-related data about the resources.

Finally, the HTTP `Expires:` header presents a way of informing the crawler of the next change in a Web page, but this involves prediction and therefore is not widely used.

Serve differences of content. The Web server provides a series of differences between a base version and newer versions. In the most simple case, the difference is only between the last and the current version. Examples: the HTTP Delta responses proposed by Mogul *et al.* [MDFK97] that use the `Content-encoding` field of HTTP responses, however, a disadvantage is that servers must retain potentially many different versions of their Web pages and that it can only be used for Web pages that have already been visited. Another disadvantage is that re-visits account for a small fraction of the total downloads, so this cannot be used for all pages.

Savant and Suel [**?**] study the possibility of delta-encoding Web pages with respect to other similar Web pages in the same server that are already in a client's cache, which gives a lower compression ratio but imposes less workload on the Web server and can be used for a larger fraction of the accesses. In their approach these differences are also compressed. See the survey by Suel and Memon [**?**] for a summary of techniques for delta compression for remote synchronization of files.

An example of delta compression being used in practice is that source code for popular free software can be updated using the `patch` [WEDM00] program, with servers providing differences between the original version and the new version. For Web sites, differences in terms of structural changes in the links, can be encoded using tables as in *WHOWEDA* [BMN03].

Serve pages only if modified. The Web crawler can avoid downloading a file if the file has not been modified. Examples: in HTTP/1.0 this is done using a date the crawler provides (usually the last visit to the same page) in an `If-Modified-Since` header; these headers are used only by a minority of crawlers [BCGMS00], although they are supported by most Web servers. In HTTP/1.1, an *entity-tag* (E-Tag) can be provided: this is a hash function of the text of the document.

Serve multiple pages on one request. The overhead arising from multiple TCP connections can be avoided by requesting a series of pages in the same connection. Example: this is usual for modern Web browsers,

and is implemented using the `Connection` header with the `keep-alive` value in HTTP/1.1 [FGM+99]; in this case, pipelining of the requests can also be used. With pipelining, the user agent requests multiple pages without waiting for a response, and then receives multiple responses in the same order as they were requested.

Serve entire site in a large file. This is suitable only if the Web changes occur in many pages, or if the Web site is composed of many small files. Example: typically, Linux distributions are distributed in whole CD- or DVD-sized disk images and not on a file-by-file basis.

Filtering interfaces. This is a standard method for answering queries from the crawler. The typical query a Web crawler could ask is "give me all the pages that have changed since this date". A more powerful filtering interface could also include requests for differences, or querying about other characteristics such as page sizes or local importance. Examples: DASL [RRDB02] for searching Web servers, RSYNC [TP03] for mirroring content, and the Common Index Protocol (CIP) [AM99]. In WebDAV [web04], the `PROPFIND` method allows to query for properties of a document, and a proposed extension `BPROPFIND` for querying about groups of documents. A generic filtering interface could also be implemented using a Web Service [CCMW01].

6.3 Interruption-based cooperation

In all these cases, the Web server sends data to the Web server whenever there is an update (page change, deletion or new page). The Web crawlers must subscribe with the Web server to start receiving notifications, and when they receive them, they can choose to process, enqueue or ignore them.

The following cooperation schemes are depicted in Figure 6.3:

Send meta-data of updates. The notification includes only meta-data about the update, as a minimum, the URL of the resource and a timestamp of the event would be required. Examples: the Keryx [BK97] notification service, developed during the apogee of push-based content delivery, and the Fresh Flow proposal for Web server cooperation [GC01].

Send differences of content. Whenever an event happens, the Web server sends a file containing the difference between the last version and the current one (if the changes are major, the Web server may send the entire file). This exploits the fact that most page changes are minor, e.g.: after one year, 50% of changed pages have changed less than 5% [NCO04]. Example: CTM [Kam03]: in this case, differences on a repository of source code are sent to interested users by electronic mail and the receivers automatically execute

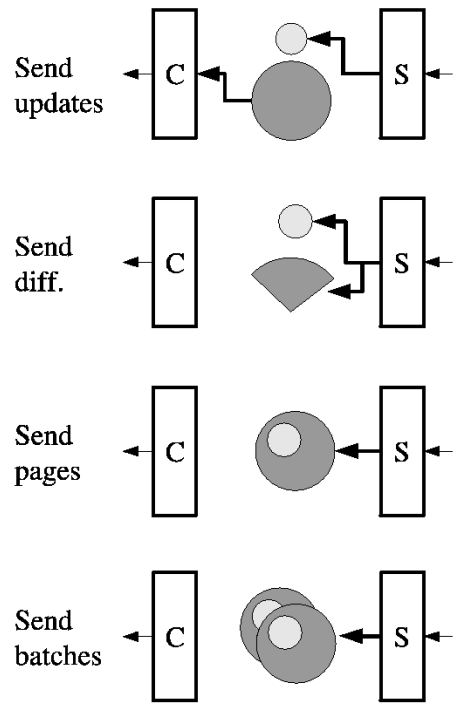

Figure 6.3: Diagrams of interruption-based schemes of cooperation. Connections are initiated by the server-side (right) and handled by the crawler (left).

the `patch` [WEDM00] program to apply the update to the local copy. In CTM, every 50 relative "deltas", a complete base set is sent.

Send changed pages. The Web server sends the complete text of each updated or new page when the modifications are made. Examples: this was typical in push technologies [KNL98], and was implemented in services like "Marimba Castanet" and "Pointcast" in the early days of the Web. Currently, it is being used in wireless devices [Cha02].

Send multi-pages updates. The Web server sends batches of modified pages according to some schedule. This can be useful if the updates are regular and involve several pages; for example, in the Web site of a daily or weekly newspaper. This is the idea behind the protocol used for keeping pages updated in mobile devices used by AvantGO [ava04], in which a single device receives a compressed bundle of pages from several Web sites.

Send entire site. The Web server sends the entire site. This is useful, for instance, for uploading an entire Web site when the site is publicly available for the first time, or if there is a major modification involving most of the pages, as an extension of the previous scheme.

Strategy	Network cost	Processing (server)	Processing (crawler)	Freshness improvement
Send meta-data of updates	+	+		High
Send differences of content	– –	+ +	+	High
Send changed pages	–	+		High
Send batch update	+	+		High
Send entire site	+ +	+		High
Remote agent	– –	+ +	–	High
Serve meta–data of content	+	+		Normal
Serve differences of content	– –	+ +	+	Normal
Serve pages only if modified	–			Normal
Serve many pages in one request	–			Normal
Serve entire site compressed	+ +	+		Normal
Filtering interface	– –	+ +	–	High

Table 6.2: Relative costs of server cooperation schemes discussed, against the base case where no cooperation exists: "+" means more cost, "−" means less cost. The last column is the expected improvement in freshness for the search engine's collection

Remote agent. The Web server executes software provided by the search engine; this software includes instructions to identify important pages and to detect changes in the pages that are relevant for the search engine. Important pages can be identified based on local connectivity, textual information, or log file analysis. Changes can be detected using a custom algorithm that varies depending on the search engine's characteristics. When there is a change, the agent sends back some data to the search engine. This can be meta-data, complete pages, or differences. This is a typical application for a mobile agent [LO99], and the cooperation can be taken one step further, as in some cases the agent could help the search engine by fetching data from "near" servers, as proposed by Theilmann and Rothermel [TR99].

6.4 Cost analysis

6.4.1 Costs for the Web server

We will consider unitary (per-page) costs and benefits:

- b: Benefit from a user viewing one page, from advertising revenues or from other sources.
- c_n: Network cost for serving one page, i.e.: bandwidth cost.

- c_p: Processing cost for serving one page, i.e.: servers cost.

A simple observation is that we should have (in theory) $b \geq c_n + c_p$, otherwise, the Web site would not be able to pay the operation costs. However, we should note that some Web sites can be financed by revenues from other sources. Another observation is that in general processing capacity is cheaper than network connectivity, so in general $c_n > c_p$.

Estimates: We cannot measure these quantities, but we can make some estimates: as of 2004, the cost per page-view of an advertising campaign on the Web is about US$ 0.05, so it is likely that $b \geq 0.05$. On the other end, having a Web server costs about US$ 10 for 5 gigabits of traffic, or 625Mb; if each page including images is 40Kb on average, this is enough for 15,000 page-views; notice that network bandwidth is usually "overbooked" in popular virtual servers, probably by a factor or 2 or 3, so an estimate of the cost is: $c_n + c_p \leq 0.002$. Serving pages to a Web crawler is cheaper because the Web crawler does not download the images.

This is a very rough estimate, but it reveals something interesting: if we only account for Web server usage, serving a Web page costs at most $1/25$ of the benefit, and this is probably the biggest cause of the huge success of the World Wide Web as a platform for publishing information. The main source of cost when keeping a large Web site is not the Web hosting, but rather the cost of producing and maintaining its contents.

In Table 6.2 we provide a rough estimation of relative costs associated with these cooperation schemes. Network bandwidth savings are the product of not dealing with unnecessary requests from the crawlers, and costs, from sending more than is necessary. Processing costs involve keeping meta-data, calculating differences, or more complex processing. Benefits arise from increased freshness on the Web search engine, and are higher if an interruption (*push*) is involved.

Which is the best strategy for the Web server? This will depend on the price the Web server is willing to pay; if this is minor, then using server software that correctly implements HTTP/1.1 is the best option. If the price is moderate, then serving and sending meta-data of updates is the best option. If the server wishes to invest more resources, it can benefit from providing content differences and/or a filtering interface for the crawlers.

6.4.2 Costs for the crawler

The main costs for the crawler for each page are:

- Polling or receiving an interrupt. We will consider that in terms of network and processing, generating a connection or handling an interrupt are the same.

- Downloading the page.
- Processing the downloaded page.

An estimation of these costs is due to Craswell *et al.* [CCHM04], and it is close to US $1.5 Million for an entire crawl of the Web, or about US$ 0.002 per page. Remarkably, this is exactly our estimation of the costs for the Web server, and both figures were obtained independently.

The freshness of the repository is higher in interrupt-based strategies, as there is no significant delay between the server update and the search engine syncing of the page. The costs for the crawler are summarized in Table 6.2. Network cost for the crawler is the same as for the server, as each transfer in these schemes involves one crawler and one server.

However, interrupt-based strategies have to be implemented carefully, because if too many Web servers are sending interrupts to the Web crawler at whatever time they choose, then the search engine risks being overloaded by these requests, loosing control over the crawling process. It is likely that interrupt-based strategies can only be deployed for a small group of Web sites.

Which is the best strategy for the crawler? A remote agent or filtering interface can help to distribute the workload of the search engine, especially if servers cooperate in pre-processing documents or in generating partial indexes. The remote agent can be used for the more important Web sites (such as news sources) if the crawler can process interrupts as they arrive, probably by keeping a small index for the most changing data.

An extreme case of using an agent could be when the Web server generates the (partial) inverted index and then sends it to the search engine, which only needs to perform a merge. In this case, the crawling problem is simplified, and is transformed into polling or pushing of indexes.

6.4.3 Overall cost

It is very important to consider that not all Web servers are equal, and the distribution of "quality" on the Web is, by all measures, very skewed: most of the important Web pages are on a few Web servers, as shown in Figure **??** (page **??**). Those servers are not necessarily the larger ones, in Figure 6.4 we compare average Pagerank with site size and find no correlation.

By inspecting Table 6.2, a noticeable fact is that the schemes that do not require extra cost for the Web server are already implemented in HTTP (`keep-alive` and `if-modified-since` features).

It is clear that if the request–response paradigm is enforced strictly, the scheme that can provide the best benefits in terms of freshness is a filtering interface. Pushing or pulling differences of content are probably the most balanced schemes, because the server gains in less bandwidth usage. These schemes are more useful if many clients can benefit from differences: not only the Web crawlers of search engines, but also the general public using enabled browsers or cache services.

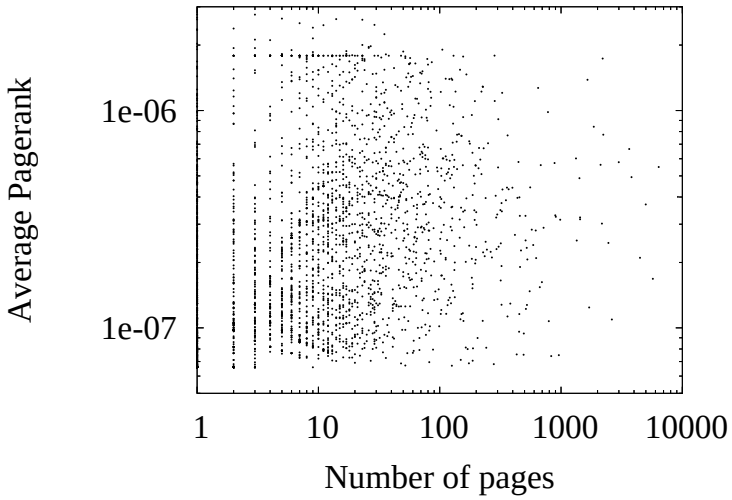

Figure 6.4: Average Pagerank versus number of pages, for a sample of 25,000 Web sites in the Chilean Web. The size of a Web site does not seems to be correlated with the quality of its pages according to this metric.

6.5 Implementation of a cooperation scheme in the WIRE crawler

The WIRE crawler supports a cooperation scheme based on serving meta-data of updates. The Web server provides an XML file containing a description of the documents provided by the Web server.

We wanted to use publicly-available XML name spaces to conform to existent definitions. We used the following XML applications (languages):

RSS RDF Site Summary, also called “Rich Site Summary” or “Really Simple Syndication” is an extension of RDF. It was conceived as a simplification of RDF to be able to aggregate multiple Web sites in a single interface for the “My Netscape” service [Lib00]. Nowadays, it is widely used by news sources to provide a short list of the latest news histories to be used by news aggregators.

DC The Dublin Core is a simple set of metadata elements to describe electronic documents. It is designed to provide a minimal set of descriptive elements for Web pages [dc04], including date, type, format, copyright status, etc.

The Web server periodically generates a file `robots.rdf`, located at the root of the Web site, containing the last-modification time of all the URLs in its public space. An example file is shown in Figure 6.5.

Currently the file contains only the URL and the last-modification time, which is the information the WIRE crawler can use, but in the future it could include more information such as page size, format, number of accesses, etc.

```
<?xml version="1.0"?>
<rdf:rdf
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns"
 xmlns:dc="http://purl.org/dc/elements/1.1/"
 xmlns="http://purl.org/rss/1.0/">
   <channel rdf:about="http://www.example.com/">
     <items><rdf:Seq>
       <rdf:li rdf:resource="http://www.example.com/one.html"/>
       <rdf:li rdf:resource="http://www.example.com/two.html"/>
     </rdf:Seq></items>
   </channel>
  <item rdf:about="http://www.example.com/one.html">
    <dc:modified>2004-10-01T12:05+02:00</dc:modified>
  </item>
  <item rdf:about="http://www.example.com/one.html">
    <dc:modified>2004-11-21T09:01+02:00</dc:modified>
  </item>
</rdf:rdf>
```

Figure 6.5: Example of a `robots.rdf` file.

The implementation of this scheme has two parts: a server-side script that generates the file, and an interpreter in the Web crawler.

6.5.1 Web Server implementation

On the server-side, a Perl script is provided for generating the RSS file. This script requires two parameters:

- The root directory of pages in the Web site.
- The base URL of the home page of the Web site.

Optional parameters are:

- Patterns to include pages. The default is to include all pages that include the substring "`.htm`".
- Patterns to reject pages, to exclude private directories.
- The maximum number of pages to include in the file.

A typical example of usage is:

```
% wire-rss-generate --docroot /home/httpd/html --base http://www.example.com/
    > /home/httpd/html/robots.rdf
```

This program is executed periodically using `crontab`. The frequency of updates should be related to the frequency of update of the Web site, but generating the file on a daily basis seems acceptable.

The `.rdf` extension was chosen because it is usually a registered file type in the Apache Web servers, and therefore the response included the corresponding `application/xml+rdf` content-type.

6.5.2 Web Crawler implementation

The WIRE crawler handles the download of this file similarly to the `robots.txt` file [Kos95]. A setting in the crawler configuration file controls the frequency at which this file is checked for changes.

The crawler parses the `robots.rdf` file and for each item found, it checks the last modification time of the file. This timestamp is entered into the equation for estimating the probability of the object being outdated, as shown in Section **??** (page **??**).

6.5.3 Testing

We tested our implementation to gather insights about how it works in practice. This is a first step that is necessary to learn about the system before a large-scale studied is carried.

We tested our implementation over a month with a Web site containing medical information; this Web site has 249 pages. Issuing a `HEAD` request for each page, just to check for the last-modification timestamp, generates 108,777 bytes of traffic, with an average of 434 bytes per page. It takes about 5 minutes to sequentially make all of these requests, even if we do not wait between them.

When using cooperation, the generated `robots.rdf` file is about 61,504 bytes, with an average is 247 bytes per page; this is more than 40% of savings in bandwidth, and with an important advantage: everything is done in just one request in less than 5 seconds.

An unexpected benefit of this implementation is that Web pages are usually discovered slowly, level by level as the crawler must parse the Web pages to find links. When the page listing is found on a single file, the Web crawler can download the entire site in just one session, without having to parse data to discover pages.

We learned that if the updates involve only just one page, then if the `robots.rdf` file is too large this scheme can waste network resources because the complete file with the metadata is transfered each time. The `robots.rdf` could be divided into several parts for very large Web sites, and these parts could be chosen in

such a way that the most important pages are found in a small part –for instance, by dividing the Web site by levels.

We also learned that for a good scheduling using a file with meta-data, the important search engine parameter is not the minimum re-visiting period, but the maximum acceptable outdated probability; otherwise, bandwidth can be wasted by requesting the meta-data file more often that it is necessary, especially if only a few Web sites are involved and they do not change too often.

6.6 Conclusions

How probable is the wide adoption of a cooperation strategy? The basic HTTP protocol is not completely implemented in the same way across different servers, and the minimum-common-denominator is quite poor in terms of functionalities, as explained in Appendix **??**.

On the other hand, Web site administrators can cooperate if it is not too costly and means an important benefit. This benefit should come mostly in terms of being better represented on the Web search engines. We consider that the reductions on load for the Web server are probably not enough by themselves to justify the adoption of a cooperation strategy.

There are also some specific applications that can use a cooperation strategy: most general search engines offer specialized (paid) search services for specific Web sites. These search services could be improved if software for cooperating with the search service is installed in the server-side.

With the emergence of Web services, filtering strategies could be an interesting possibility for the near future, as they can help crawlers and other autonomous agents to interact with Web servers at a more meaningful level.

Bibliography

[AM99] J. Allen and M. Mealling. RFC 2651: The architecture of the Common Indexing Protocol. http://www.ietf.org/rfc/rfc2651.txt, 1999.

[ava04] AvantGO. http://www.avantgo.com/, 2004.

[BCGMS00] Onn Brandman, Junghoo Cho, Hector Garcia-Molina, and Narayanan Shivakumar. Crawler-friendly web servers. In *Proceedings of the Workshop on Performance and Architecture of Web Servers (PAWS)*, Santa Clara, California, USA, June 2000.

[BK97] S. Brandt and A. Kristensen. Web push as an Internet Notification Service. In *W3C workshop on push technology*, Boston, MA, USA, 1997.

[BMN03] Sourav Bhowmick, Sanjay Kumar Madria, and Wee Keong Ng. Detecting and representing relevant web deltas in WHOWEDA. *IEEE Transactions on Knowledge and Data Engineering*, (2):423–441, 2003.

[Buz03] Marina Buzzi. Cooperative crawling. In *Proceedings of Latin American Conference on World Wide Web (LA-WEB)*, pages 209–211. IEEE Cs. Press, 2003.

[Cas03] Carlos Castillo. Cooperation schemes between a web server and a web search engine. In *Proceedings of Latin American Conference on World Wide Web (LA-WEB)*, pages 212–213, Santiago, Chile, 2003. IEEE Cs. Press.

[CCHM04] Nick Craswell, Francis Crimmins, David Hawking, and Alistair Moffat. Performance and cost tradeoffs in web search. In *Proceedings of the 15th Australasian Database Conference*, pages 161–169, Dunedin, New Zealand, January 2004.

[CCMW01] Erik Christensen, Francisco Curbera, Greg Meredith, and Sanjiva Weerawarana. WSDL: Web services description language. http://www.w3.org/TR/wsdl, 2001.

[Cha02] Ben Charny. CNET news: Wireless Web embraces “push”. http://news.com/2100-1033-958522.html, 2002.

[dc04] Dublin Core Metadata Initiative. http://dublincore.org/, 2004.

[FGM+99] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and Tim Berners-Lee. RFC 2616 - HTTP/1.1, the hypertext transfer protocol. http://w3.org/Protocols/rfc2616/-rfc2616.html, 1999.

[GC01] Vijay Gupta and Roy H. Campbell. Internet search engine freshness by web server help. In *Proceedings of the Symposium on Internet Applications (SAINT)*, pages 113–119, San Diego, California, USA, 2001.

[Kam03] Poul-Henning Kamp. OpenBSD CTM. http://www.openbsd.org/ctm.html, 2003.

[KNL98] Tuula Kapyla, Isto Niemi, and Aarno Lehtola. Towards an accessible web by applying push technology. In *Fourth ERCIM Workshop on "User Interfaces for All"*, Stockholm, Sweden, 1998.

[Kos95] Martijn Koster. Robots in the web: threat or treat ? *ConneXions*, 9(4), April 1995.

[LG99] Steve Lawrence and C. Lee Giles. Accessibility of information on the web. *Nature*, 400(6740):107–109, 1999.

[Lib00] Dan Libby. History of RSS. http://groups.yahoo.com/group/syndication/message/586, 2000.

[LO99] Danny B. Lange and Mitsuru Oshima. Seven good reasons for mobile agents. *Communications of the ACM*, 42(3):88–89, 1999.

[LS99] Ora Lassila and Ralph Swick. World Wide Web Consortium - RDF. http://www.w3.org/TR/REC-rdf-syntax, 1999.

[MAR+00] D. Menasce, V. Almeida, R. Riedi, F. Pelegrinelli, R. Fonseca, and W. Meira Jr. In search of invariants for e-business workloads. In *Proceedings of the second ACM Conference on Electronic Commerce*, Minneapolis, October 2000.

[MDFK97] Jeffrey C. Mogul, Fred Douglis, Anja Feldmann, and Balachander Krishnamurthy. Potential benefits of delta encoding and data compression for HTTP. In *Proceedings of ACM conference of Applications, Technologies, Architectures and Protocols for Computer Communication (SIGCOMM)*, pages 181–194, Cannes, France, 1997.

[NCO04] Alexandros Ntoulas, Junghoo Cho, and Christopher Olston. What's new on the web?: the evolution of the web from a search engine perspective. In *Proceedings of the 13th conference on World Wide Web*, pages 1 – 12, New York, NY, USA, May 2004. ACM Press.

[RGM01] Sriram Raghavan and Hector Garcia-Molina. Crawling the hidden web. In *Proceedings of the Twenty-seventh International Conference on Very Large Databases (VLDB)*, pages 129–138, Rome, Italy, 2001. Morgan Kaufmann.

[RRDB02] J.F. Reschke, S. Reddy, J. Davis, and A. Babich. DASL - DAV searching and locating protocol. http://www.webdav.org/dasl/, 2002.

[SS04] Danny Sullivan and Chris Sherman. Search Engine Watch reports. http://www.searchengine-watch.com/reports/, 2004.

[TP03] Andrew Tridgell and Martin Pool. RSYNC: fast incremental file transfer. http://samba.anu.edu.au/rsync/, 2003.

[TR99] W. Theilmann and K. Rothermel. Maintaining specialized search engines through mobile filter agents. In M. Klusch, O. Shehory, and G. Weiß, editors, *Proc. 3rd International Workshop on Cooperative Information Agents (CIA'99)*, pages 197–208, Uppsala, Sweden, 1999. Springer-Verlag: Heidelberg, Germany.

[vHGH+97] Arthur van Hoff, John Giannandrea, Mark Hapner, Steve Carter, and Milo Medin. DRP - distribution and replication protocol. http://www.w3.org/TR/NOTE-drp, 1997.

[web04] WebDAV resources. http://www.webdav.org/, 2004.

[WEDM00] Larry Wall, Paul Eggert, Wayne Davison, and David MacKenzie. GNU patch. http://www.gnu.org/software/patch/patch.html, 2000.