# Spores: a Push and Pull Peer-to-Peer File Sharing Approach

Thomas E. Portegys
*Illinois State University*
portegys@ilstu.edu

## Abstract

*Spores is a push and pull peer-to-peer method of file sharing and storage, making use of publicly available space on the network. A user stores a file by pushing it to a set of peers. The file then becomes visible and available to peers that search for it. Spores allows the exchange of folders as well as individual files. To ensure a file operation references a specific file or folder by content, an MD5 uniqueness code can be used to specify the search or download. If both code and file name are given, the target must match both. An analysis of the storage and retrieval performance is presented.*

**Keywords**: peer-to-peer, file sharing.

## 1. Introduction

In the last decade, numerous peer-to-peer file sharing systems have appeared on the scene sporting a variety of features. Some, such as KaZaA [4] and Morpheus [6], have gained great popularity by providing a convenient means of searching for and downloading music, video and software files. A positive feedback occurs for a file sharer: as more users share a greater number and variety of files, the system becomes ever more attractive to prospective users.

Spores is both a pull (download) and push (upload) file sharing system. A user may search for and download files that are made publicly available by other users. In addition, a user may upload files into public space on peer nodes. One purpose for this is to accelerate the sharing process. Peer-to-peer networks are typified by a high amount of peer transience. Pushing copies of a file increases the likelihood of finding it as nodes come and go. Moreover, in some political environments being able to push then delete a file may be beneficial. In addition, having users offer public storage provides a measure of absolution for the content stored there: if a file is found on one's machine, one may rightly disavow storing it there.

File sharing and storage systems are based on two main classifications of overlay networks that provide the underlying means of connecting peers. The first may be characterized as imposing a regularity on the overlay network, either by a static topology or by using a naming convention for files and nodes that allows an efficient algorithmic lookup capability. Publius [9] is an example of a static arrangement that features anonymous file storage through the use of encryption and disbursed partial keys. Oceanstore [5], Past [2], and CAN [7] are examples of file sharing and storage systems based on dynamic naming and lookup techniques such as Tapestry [10] and Chord [8]. For these, a file name is used to derive a key that is systematically routed toward nodes where the file might reside.

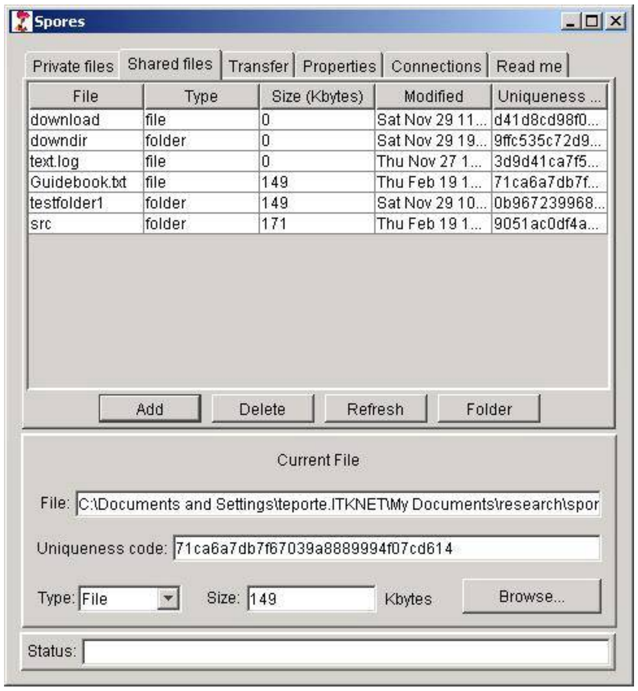
The second major type of overlay network may be classified as ad hoc and is typified by Gnutella [3]. Connecting to and communicating within this type of network is a simple process. In Gnutella, a peer initially obtains the addresses of a small number of peers from a known cache, and then proceeds to build up a neighborhood of more peers by exchanging and forwarding identifying messages. A peer's neighborhood forms a pool of searchable and downloadable files. Spores and Freenet [1] fall into this camp. Freenet also promotes the propagation of popular files by caching them along retrieval paths. In this way, popular files becomes more numerous and more readily available to peers that frequently request them. Users cannot intentionally push files into the network however, as can be done with Spores.

## 2. Description

The Spores GUI is shown in Figure 1. Top-level tabs allow access to private and shared files, transfer operations, properties, and initial connection information. The open source Java code is available at www.ilstu.edu/faculty/portegys/research.html.

### 2.1. Connecting

When Spores is started, it obtains initial peer addresses from a file of static addresses, as well as from a customizable list of web caches (see www.gnucleus.com/gwebcache for details). A web cache allows a peer to both obtain addresses as well as store its own address to be made available to other connecting peers. In addition, a web cache can store the location of other web caches, allowing new ones to be discovered for subsequent initializations. Using the initial address list from the caches, a persistent thread attempts to fill and maintain a list of active peers. It does this by continuously cycling through the list and requesting more addresses from neighbors when the number of active (responding) peers falls below a parameterized value. If a peer fails to respond to a periodic heartbeat message within a specified time, it is removed from the list. If the list becomes empty, for example when a modem connection drops and remains disconnected for a time, a periodic attempt to revisit the web caches is made.



| File | Type | Size (Kbytes) | Modified | Uniqueness ... |
|---|---|---|---|---|
| download | file | 0 | Sat Nov 29 11... | d41d8cd98f0... |
| downdir | folder | 0 | Sat Nov 29 19... | 9ffc535c72d9... |
| text.log | file | 0 | Thu Nov 27 1... | 3d9d41ca7f5... |
| Guidebook.txt | file | 149 | Thu Feb 19 1... | 71ca6a7db7f... |
| testfolder1 | folder | 149 | Sat Nov 29 10... | 0b967239968... |
| src | folder | 171 | Thu Feb 19 1... | 9051ac0df4a... |

Current File

File: C:\Documents and Settings\teporte.ITKNET\My Documents\research\spor

Uniqueness code: 71ca6a7db7f67039a8889994f07cd614

Type: File Size: 149 Kbytes

Status:

Figure 1. Spores GUI

## 2.2. Files and folders

Users specify and manage a private and shared file space. Both files and folders (directories) are allowed. Files/folders in the shared space are visible to and downloadable by other peers. An MD5 uniqueness code is derived from the content of files and can be used to specify or augment a search or download, thus allowing a file to be referenced by content in addition to or instead of a file name. Specifying a code prevents counterfeit files from satisfying requests. A code, of course, must be known beforehand in order to use it. For a folder, the code is an MD5 calculation over a recursive sequence of file and folder codes contained within the folder.

A user has control over the total size and number of files in the shared space. A user can also specify shared file extensions that will be accepted or refused from prospective uploaders. For example, ".mpeg3,.wav" allows only sound files of these types. Conversely, if a user wants to accept all files except jpg and mpeg files, "-.jpg,-.mpeg" implements this restriction.

## 2.3. File searching and transfers

File searching in Spores is done in parallel. A peer sends search requests to its immediate neighbors who in turn forward the request to their immediate neighbors, etc., for a specified depth into the network.

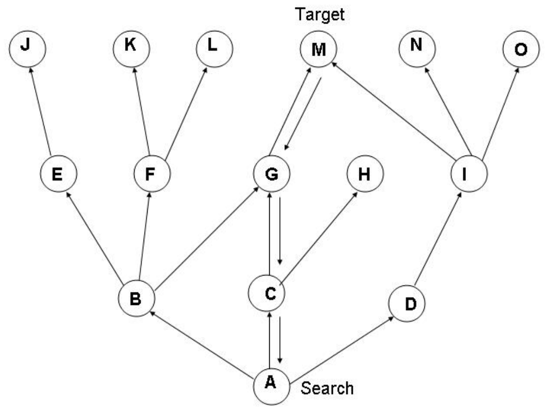


Figure 2. Parallel file search

A search request specifies a file name, uniqueness code, type (file or folder), and search id. Either a file name or uniqueness code (or both) must be given. A search id is derived from the other components of the request plus the originating peer address. It is cached at peers receiving the request and used to prevent looping and redundant messages. Figure 2 shows peer A searching for a file that is stored in peer M. When a search request reaches M, the file name and/or uniqueness code are matched against files in M's shared space. The return arrows show the path of the search success messages that contain the address of M. In Spores it is useful to know how many copies of a file exist in the surrounding network to estimate the file saturation level. For this reason, every outgoing search message results in a reply indicating the number of copies found. In the example, peer A will receive three replies containing accumulated copy counts. A successful search results in the target peer's address being stored in the searching peer's connected list in the anticipation that it will later be downloaded from.

A download consists of a preliminary search that will obtain a list of target peers containing the desired file. A download then consists of transmitting the contents of the file from a target peer to a temporary space within the requesting peer. A maximum downloadable file size prevents overloading the transmission. Folders are transferred in a recursive manner. If a uniqueness code is specified, the file is checked to prevent an erroneous or malicious peer from responding with the wrong file. If only a uniqueness code is specified, the downloaded file name is checked against files residing in the shared space. If a file with the same name is found, it is assumed that the user does not want the file to be overwritten and the file is not accepted. Next the file extension is checked against the list of accepted and refused extensions. Finally, the user-specified number of shared files and total shared file size is checked for possible violations. If all constraints are satisfied, the file is copied to the shared space. Otherwise, the file is downloaded from the next target peer.

To upload a file, the user specifies the file by name and the number of copies to push into the network. The process is centralized at the uploading peer and proceeds in a tier by tier fashion as shown in Figure 3. Here peer A will first request an upload to its immediate neighbors B, C, and D, which form tier 0. The upload operation is terminated when the desired number of copies is pushed or when peer addresses are exhausted. If after tier 0 there are more copies to push, B, C, and D are asked for lists of their immediately connected neighbors. This list forms tier 1 and consists of peers E, F, G, H, and I. A peer will accept a file after checking the various constraints discussed in the section on downloading.

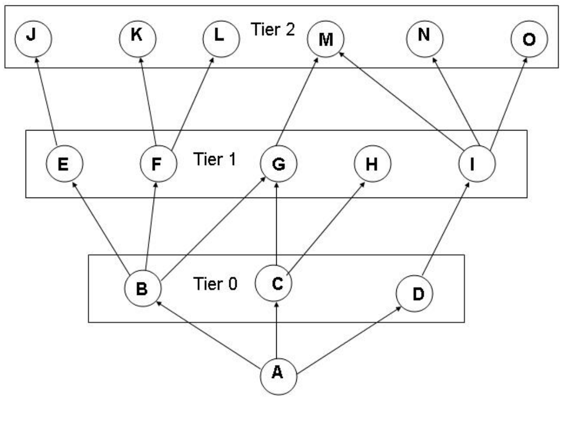


**Figure 3. File uploading**

Spores currently can be characterized as a robust prototype; it is missing a number of features of more mature products. For example, it does not preferentially transfer files from peers connected by high bandwidth paths. It also does not do “swarming”, i.e., simultaneously transferring portions of a file from different peers. Along the same lines, it cannot restart a broken transfer from the point of failure.

## 3. Analysis and results

As previously noted, one of Spores’ goals is to accelerate the sharing process in a network of transient peers by allowing files to be pushed to other peers for later discovery and transfer. Toward this end, the desired outcome is to be able to retrieve a file once it is stored on the network. In this model, there are three factors determining the retrieval rate: N, the size of the network; K, the number of active (connected) peers, and M, the number of files stored in the network. The retrieval rate is then the probability of choosing K peers from N, C(N,K), and finding one or more of the M files in the set of K. This is expressed by:

$$\text{Retrieval rate} = 1 - C(N-M,K)/C(N,K)$$

Figure 4 shows the retrieval performance for a network of 1,000,000 peers. The rate is plotted against the number of stored files, and is shown for 1000, 5000, and 10,000 active peers. For 1000 active peers, the rate climbs slowly as the number of files increases. Encouragingly, for 5000 and 10,000 peers the retrieval rate approaches certainty fairly rapidly. So for example, only 300 files are required to be stored to achieve a 96% rate when 10,000 (1%) of the peers are active. Moreover, this performance pertains to a single retrieval attempt; more attempts increase the chances of retrieval.

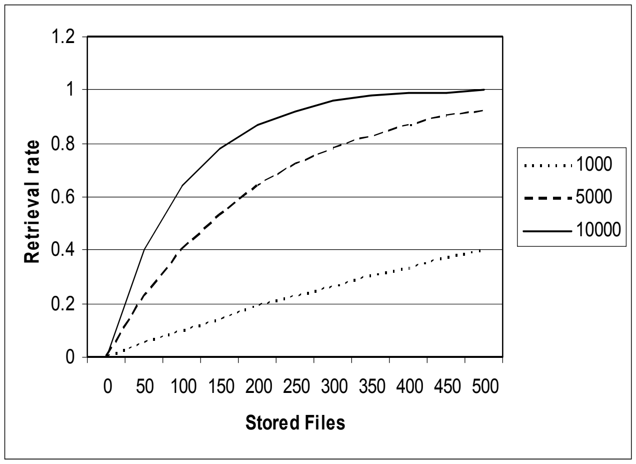


**Figure 4. Retrieval performance, N=1,000,000 peers**

Spores has been load tested on a very limited basis (10s of peers) in an educational facility equipped with high bandwidth connectivity as part of a classroom assignment for a course in parallel processing. Tests involving bursts of file transfers between small numbers of workstations have also been successful.

## 4. Conclusion

Spores was inspired, as Freenet was, by a desire to promote the free exchange of information on the internet. It is hoped that it or the techniques employed by it contribute something of value in the realm of peer-to-peer file services.

## 5. References

[1] Clarke, I., Wiley, B., Sanberg, O., Hong, T., "Freenet: A Distributed Anonymous Information Storage and Retrieval System," in *Designing Privacy Enhancing Technologies: International Workshop on Design Issues in Anonymity and Unobservability*, Springer-Verlag LNCS 2009, ed. by H. Federrat, Springer: New York 2001.

[2] Druschel, P., and Rowstron, A., "PAST: A large-scale, persistent peer-to-peer storage utility", *HotOS VIII*, Schoss Elmau, Germany, May 2001

[3] Gnutella, http://www.gnutella.com/, 2004.

[4] KaZaA, http://www.kazaa.com/, 2004.

[5] Kubiatowicz, J.,et al., "OceanStore: An Architecture for Global-Scale Persistent Storage", in *Proceedings of the Ninth international Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2000)*, November 2000.

[6] Morpheus, http://www.morpheus.com/, 2004.

[7] Ratnasamy, S., Francis, P., Handley, M., Karp, R., Shenker, S., "A Scalable Content-Addressable Network" *ACM Sigcomm* 2001,

[8] Stoica, I., Morris, R., Karger, D., Kaashoek, F., Balakrishnan, H., "Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications," *ACM Sigcomm 2001*,

[9] Waldman, M., Rubin, A.D., and Cranor, L.F., "Publius: a robust, tamper-evident, censorship-resistant, web publishing system", in *Proceedings of the Ninth USENIX Security Symposium*, Denver, CO, USA, 2000.

[10] Zhou, B., Joseph, D.A., Kubiatowicz, J., "Tapestry: a fault tolerant wide area network infrastructure," *Sigcomm* 2001 poster and UC Berkeley Tech. Report UCB/CSD-01-1141.