Peer-to-Peer Support for File Transfer and Caching Mechanism

Jenq-Haur Wang and Tzao-Lin Lee

Department of Computer Science and Information Engineering,
National Taiwan University,
Taipei, Taiwan.
{jhwang,tl_lee}@csie.ntu.edu.tw

Abstract. In existing Internet file transfer mechanism, proxy servers play a major role in load balancing and reducing duplicate file access requests for services like FTP and WWW. However, proxy servers are usually unaware of the availability of cached contents on other peer proxy servers. This is a waste of time since duplicate requests are needed. Unnecessary traffic can be reduced if cooperation and coordination among peer proxies can be utilized. In this paper, peer-to-peer support was incorporated in ordinary file transfer and caching mechanism to reduce unnecessary processing time and storage. Through the location service, hosts requesting file services can dynamically determine if a copy is available and its current location. Work load for file servers will be greatly reduced, and personalization of file transfer configuration can be fully supported.

1 Introduction

With the tremendous growth of the Internet, numerous networking applications such as WWW (World-Wide Web), E-mail, and FTP (File Transfer Protocol) [1] have been widely used. Specifically, file access applications like WWW and FTP have become ubiquitous and central to many people's daily lives. However, in current file transfer mechanism, FTP and Web servers play a critical role since all file access requests require the intervention of these servers. This could result in overloaded server and no service could be provided. Therefore proxy servers have been widely deployed to eliminate unnecessary transfers for file objects already retrieved by other clients.

When a user browses a web page through a proxy server, the URL (Uniform Resource Locator) of requested web page will be checked if a copy is available on proxy server. If so, no further outbound connections are needed since the page is already fetched. If not, the proxy server will act like an agent for the client and make HTTP (HyperText Transfer Protocol) [2] requests to the real web server as indicated in the URL on behalf of the client.

However, communication and coordination among peer proxy servers are still not much used. Proxy servers are usually configured in a hierarchical way where *parent* and *sibling* proxies are manually organized. When a proxy doesn't contain the requested file object (a *cache miss*), it may make Internet Cache Protocol (ICP) [3] requests to see if any of its neighbor proxies has the object. "Neighbor hits" where neighbor proxy has the object may be fetched from either *parent* or *sibling* proxy, but "neighbor misses" must be forwarded only to *parent* proxy. Since parent and sibling relationships must be manually configured in existing implementations like *squid* [4], reutilization of existing cached contents on peer proxy servers can be very difficult. Duplicate file replications among different proxies are still possible and caching efficiency may be further improved.

With the rapid development of various mobile devices, wireless LANs (WLANs) [5] have become more popular as an alternative network access method. In infrastructure mode, mobile nodes can connect to the wired network via access points (APs) as if they have been directly attached. However, since APs are limited in range, mobile nodes may roam into the ranges of different APs. IP roaming problem occurs if different APs are located on different subnets. Mobile IP scheme [6] is one of the most common ways to solve the IP roaming problem.

On the other hand, peer-to-peer technology has been widely deployed in various applications, for example, file sharing software like Napster [7] and ezPeer, instant messaging software like ICQ and MSN, and open source protocol like Jabber [8] and GnuTella [9]. Moreover, the distinction between centralized and decentralized applications has become blurred to leverage the advantages of both. Therefore, an infrastructure for integrating current Internet client-server file transfer mechanism and peer-to-peer file sharing applications was proposed to provide better integrated services. In a mobile environment, each mobile node may act as a peer proxy in which the cached content could be utilized by other nodes. Therefore, our focus is on better utilizing existing proxy caching mechanism and web cache communication and coordination protocols in peer-to-peer applications.

2 Motivation

In this section, the current architecture for file transfer and caching and its shortcomings will be briefly reviewed, and our infrastructure will be proposed as a feasible solution.

In current Internet file transfer mechanism, several protocols are used: HTTP [2] for transferring web pages, FTP [1] for transferring files, and ICP [3] for inter-proxy communication. As shown in Fig. 1, a typical scenario for current file transfer mechanism is illustrated.

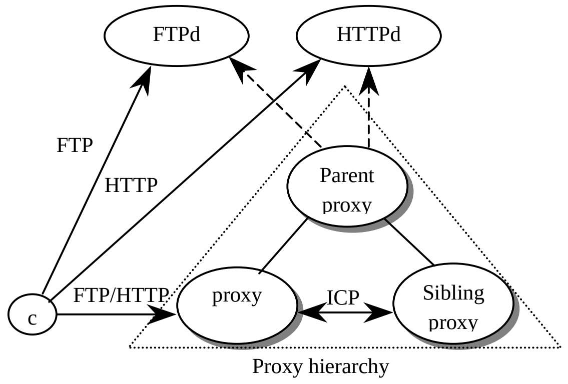

Fig. 1. Shows a typical scenario for file transfer and caching, where a proxy hierarchy is deployed.

As shown in Fig.1, users can browse a web page or access a file with a specific *URL* (*uniform resource locator*) via browser like Microsoft Internet Explorer or Netscape. File access requests are made via either HTTP or FTP directly or through a proxy server. Since common files on the same web site may be requested by different users, proxy server is usually deployed as a cache of similar requests for domain users.

In order to get better performance, more than one proxy servers may be deployed in a hierarchical way. There are many possible deployment schemes for proxy servers with regards to the relative place of a cache between client and server. *Proxy caching* as described above is the most common one. The other possible schemes include *personal proxy server* where cache is on each individual client, *transparent proxy caching* where proxy setting is transparent to clients, *reverse cache* where the focus is on server not clients, and *active caching* where applets are used for caching dynamic documents [10]. In fact, these schemes may be deployed simultaneously with better overall performance.

Proxy server configuration in a browser can be done automatically by protocols like WPAD (Web Proxy Automatic Discovery) [11], through a PAC file (Proxy Auto-Config File Format) [12], or manually configured.

The web caching mechanism works fine, but there are several problems that affect the performance of file retrieval. Firstly, the load on a proxy server is heavy in terms of file storage and time for HTTP/FTP processing. Each proxy server has to deal with every file access request from domain clients. Usually cache hits in proxy server will result in better performance for retrieving file objects. However, in the case of busy proxy server or even server failure, the performance would be worse than without proxy.

Secondly, file objects may have been cached by other peer proxies or personal proxy servers which are unknown to our proxy server. Although inter-cache communication protocols such as ICP [3], WCCP [13], HTCP [14], CARP [15], and Cache Digest [16] have been proposed, they are not widely implemented. Moreover, inter-proxy communication relationships are usually manually configured and dynamic addition and removal of peer proxy can be difficult.

Thirdly, proxy configuration in a browser is usually not versatile enough. In the case of busy server or server failure, no fallback mechanism for bypassing overloaded server is provided. This could result in worse performance than direct connection without proxy.

Fourthly, personalization cannot be done very efficiently in proxy server. For example, it's difficult to configure a content filter for each individual domain user. That would be time-consuming and impractical. It's common to configure on firewall or proxy server a content filter for the whole domain. But for each individual domain user, a finer-grain control of configuration is needed, for instance, a content filter for each user, which is more reasonable since each user may want to filter content from different sources.

In order to offload proxy server and to provide complete customization in file retrieval, a peer-to-peer infrastructure for file transfer and caching was proposed. Specifically, we want to bypass a busy or overloaded proxy server if there are other replications for requested objects. Cached content on peer proxy servers can be utilized for improving cache utilization. Besides, users can have their own configurations for file processing like content filtering.

3 Infrastructure

As shown in Fig.2, an infrastructure for peer-to-peer file transfer is illustrated.

When clients issue HTTP/FTP requests to proxy server, it will first query the location server for possible replications of the given URL. Since peer proxy has registered to location server, its presence and content will be known to location server. After receiving reply from location server, proxy server will issue *redirect* messages to client which will then try to access from the peer proxy.

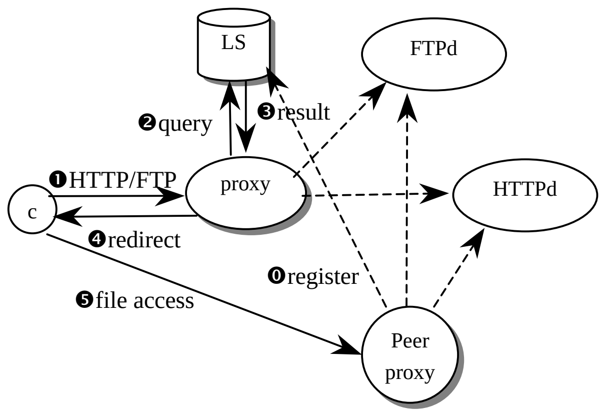

Fig. 2. Shows peer-to-peer support (*redirect* mode) for file transfer and caching.

In this infrastructure, searching for proxy servers can be transparent to users since location service lookups can be done automatically by proxy servers. Besides, the availability of peer proxy can be used as a way of load balancing between servers. Since the latest information for each cache in a domain can be obtained, the most suitable proxy server can be reached and load balancing can be achieved. Fault tolerance mechanism can also be provided in the case of proxy failure. As shown in Fig. 2, key components in the infrastructure include: location servers (LS), proxy servers, and FTP/HTTP servers. The functional description of each component is provided as follows.

3.1 Location Server

Location server is responsible for storing the current location and content index for each peer proxy. These include hostname, current IP address, URLs for cached content, and resource profiles (for example, access control list). Since the peer proxy server may be changing its location or contents frequently, the amount of data update may be quite large. Therefore, Resource Location Records (RLRs) can be stored in a distributed way, for example, one location server for each domain (like DNS server). RLRs for cached URLs on each proxy server are stored in location server of its home domain.

Most of the relevant works in location service are related to geographical positioning of mobile nodes in a wireless network, the location of servers, and location-based services. They mainly focused on the physical positioning of mobile nodes or servers, not the current way of accessing a particular resource, for example, the IP address of currently available peer proxy with the requested data.

As shown in Fig. 3, there are two possible operations for a location server: *update* and *query*. Proxy servers *update* their current location (IP address), URLs and resource profiles for cached content to location server when they are first added, changed, or removed from the domain. On the other hand, peer proxy servers *query* the location server for available replication of a particular URL in order to retrieve resource from it. In other words, location server has to be coupled with the management of resource addition/removal. Proxy servers must do registration/de-registration when being added or removed.

However, when mobile node is roaming into a foreign network, it must register to its home location server for location update. This can be done directly or through the help of location server in foreign network (Indirect Update). For a mobile node to detect it has left its home network, the advertisement based mechanism used in Mobile IP [6] or hint based move detection method [17] can be deployed.

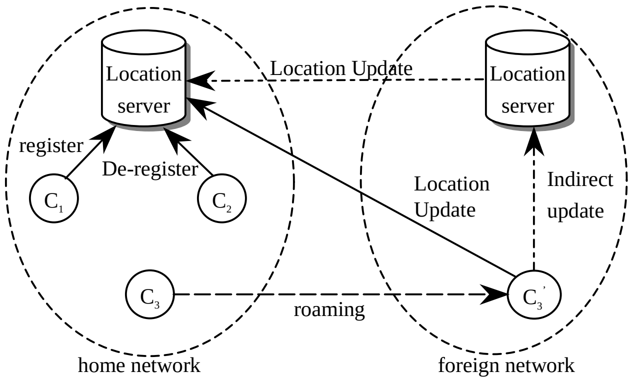

Fig. 3. Shows the operations of location servers.

3.2 Proxy Server

In our infrastructure, each proxy server has to register to its domain location server when the cached contents are added, changed, or removed. The current IP address and the cached contents are indexed by the location server. When a peer proxy needs to search for the availability of a specific URL, a location service query will be issued and the result will be checked to see if redirect is needed.

There are several deployment alternatives for peer-to-peer file transfer and caching. Besides the *redirect* mode depicted in Fig.2, two other schemes are possible, *proxy* mode and *server-to-server copy* mode, which are illustrated as follows.

In *proxy* mode, file access requests from clients are repeated on local proxy where file objects fetched from peer proxy are cached. This “greedy” caching mechanism would require more storage requirement but less penalties for a cache miss will be experienced since as much content as possible will be cached. But it’s not suitable for proxy server load balancing since the load of proxy server is heavy.

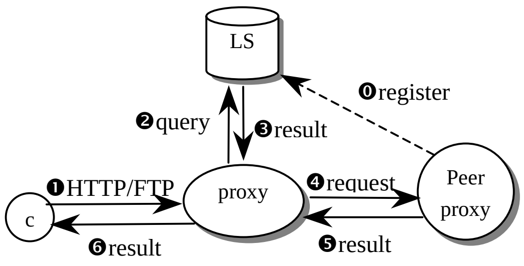

Fig. 4. Shows an alternative deployment scheme (*proxy* mode) for peer-to-peer file transfer and caching.

On the other hand, in *server-to-server copy* mode, file access requests for clients are not redirected to peer proxies. Instead, notifications to both client and peer proxy are issued by local proxy and the real file transmission takes place without the intervention of local proxy. This is illustrated in Fig. 5.

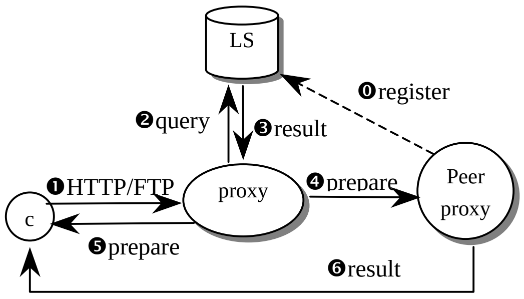

Fig. 5. Shows *server-to-server copy* mode for peer-to-peer file transfer and caching.

This caching mechanism has the advantage of load balancing for *redirect* mode, without much intervention of local proxy.

Note that existing inter-cache communication protocols can still be used in different conditions. For example, ICP [3] can be used for inter-proxy communication protocol, but modifications to ICP are required for supporting mechanisms such as server-to-server copy. On the other hand, cache digests [16] can be used in which full index doesn’t have to be built. Only the cache digests for each peer proxy are needed.

Among these alternatives, *redirect* mode is better for load balancing, while *proxy* mode has the advantage of “greedy” caching in local proxy. *Server-to-server copy* mode has the advantage of *redirect* mode without much intervention for local proxy server if inter-proxy communication protocol support is available.

4 Advantages

In our architecture, there are several advantages over current file transfer mechanism. Firstly, file (WWW, FTP, proxy) servers can be offloaded since replication can be found via location service lookups. Load balancing can thus be achieved. Secondly, peer-to-peer support for file transfer can be achieved, and integration of existing file transfer protocols with peer-to-peer applications can be done. Thirdly, personal configuration for file server, for example, content filtering, such as ACL: allow/deny <source URL>, can be fully supported.

5 Security Concerns

When one mobile node is roaming into a foreign network, authentication and authorization is required before it's granted network access. For example, IEEE 802.1x [18] can be used as the network access control mechanism as shown in Fig. 6.

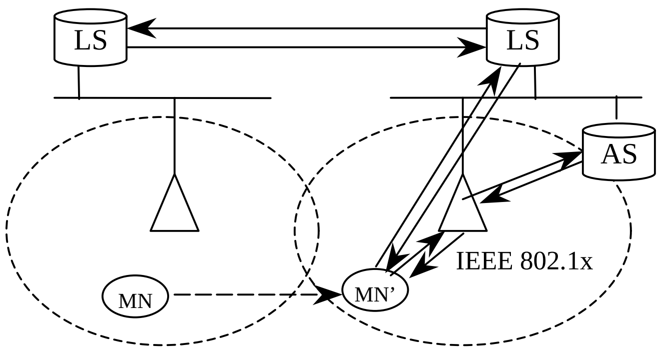

Fig. 6. Shows the authentication and authorization for mobile nodes, where AS is the Authentication Server, and LS is the Location Sever.

On the other hand, for each operation of update and query, authentication and authorization are required to ensure the correctness of each record in location server.

6 Future Work

Most importantly, file authenticity is the most difficult problem. We have to make sure that the file objects registered by peer proxies are indeed the objects as they claim. The authenticity and non-repudiation principle is most important. Besides, conditions for users behind firewall and users inside private network have to be dealt with.

7 Conclusion

In this paper, a peer-to-peer support for file transfer and caching mechanism was proposed. Through the sharing of cached contents of peer proxies in the same domain, we could further improve the cache utilization and reduce unnecessary duplicate file access requests. In addition, load balancing for overloaded proxy servers can be achieved by means of proxy redirecting and server-to-server copy operations incorporated in our scheme.

References

1. J. Postel and J. Reynolds, "File Transfer Protocol (FTP)," *STD 9, RFC 959*, IETF, October 1985.
2. R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext Transfer Protocol – HTTP/1.1," *RFC 2616*, IETF, June 1999.
3. D. Wessels and K. Claffy, "Internet Cache Protocol (ICP), version 2," *RFC 2186*, IETF, September 1997.
4. Squid Web Proxy Cache, http://www.squid-cache.org/.
5. Information Technology – Telecommunications and Information Exchange between System – Local and Metropolitan Area Networks – Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, *IEEE Std. 802.11-1999*, 1999.
6. C. Perkins, "IP Mobility Support for IPv4," *RFC 3220*, IETF, Jan. 2002.
7. Napster, http://www.napster.com/.
8. Jabber, http://www.jabber.org/.
9. The GnuTella Protocol Specification v0.4, http://www.clip2.com/GnutellaProtocol04.pdf.
10. G. Barish and K. Obraczka, "World Wide Web Caching: Trends and Techniques," *IEEE Communication Magazine, vol. 38, issue 5*, pp. 178-185, May 2000.
11. P. Gauthier, J. Cohen, M. Dunsmuir, and C. Perkins, "Web Proxy Auto-Discovery Protocol (WPAD)," *Internet Draft*, Internet article at: http://www.web-cache.com/Writings/Internet-Drafts/draft-ietf-wrec-wpad-01.txt, IETF, July 1999.
12. Netscape, "Navigator Proxy Auto-Config File Format," Internet article at: http://wp.netscape.com/eng/mozilla/2.0/relnotes/demo/proxy-live.html, March 1996.
13. M. Cieslak, D. Forster, G. Tiwana, and R. Wilson, "Web Cache Communication Protocol V2.0," *Internet Draft*, IETF, April 2001.
14. P. Vixie and D. Wessels, "Hyper Text Caching Protocol (HTCP/0.0)," *RFC 2756*, IETF, January 2001.
15. V. Valloppillil and K.W.Ross, "Cache Array Routing Protocol v1.0," *Internet Draft*, IETF, February 1998.
16. A. Russkov and D. Wessels, "Cache Digests," *Proceedings of 3rd International WWW Caching Workshop*, April 1998.
17. N. A. Fikouras and C. Goerg, "Performance Comparison of Hinted and Advertisement Based Movement Detection Methods for Mobile IP Hand-offs," *Proceedings of the European Wireless 2000*, Dresden, Germany, September 2000.
18. IEEE Standards for Local and Metropolitan Area Networks: Port based Network Access Control, *IEEE Std. 802.1X-2001*, June 2001.