An Cross Layer Collaborating Cache Scheme to Improve Performance of HTTP Clients in MANETs

Jin Liu[1], Hongmin Ren[1], Jun Wang[2], Jin Wang[2]

1 College of Information Engineering, Shanghai Maritime University, Shanghai 200135, China

2 Computer and Software School, Nanjing University of Information Science & Technology, Nanjing 210044, China

{jinliu, hmren}@shmtu.edu.cn; {wangjun, wangjin}@nuist.edu.cn

Abstract. This paper proposes a novel cross layer collaborating cache scheme for HTTP clients. It is designed to accelerate the process that an HTTP client retrieves Web content in mobile ad hoc network environment. Unlike other techniques to exploit Web proxy caching or to build P2P protocol overlay to improve HTTP, our scheme incorporates collaboratively working capability into the HTTP client response cache, and thus there is no impact on the compatibility of upper level applications. In addition, as the sharing file information is diffused with routing information, there is no redundant traffic to maintain the cooperating cache mechanism. Using this scheme, cacheable Web content can be distributed among local peers and retrieved later by a client if needed. In our simulation results, this novel cache scheme significantly decreases the latency experienced by a client to download Web content.

Keywords: HTTP Client,, Ad Hoc Network, Cache, Latency

1 Introduction

The Hypertext Transfer Protocol (HTTP) is an application-level transactional protocol that uses the Transmission Control Protocol (TCP) for ordered guaranteed delivery of web objects between Web agents. The overall performance of a HTTP transaction depends on the performance of the HTTP clients, the HTTP server, the HTTP agents in between and the lower layer communication network. Traditional optimizations to Internet applications performance usually focus on link, network and transport layer protocols for they are the basis for all the different categories of upper layer Internet applications including HTTP. However, HTTP itself comprises of a complex set of mechanisms, optimizations can be done in different aspects. Persistent network connections and pipelining requests are used to improve HTTP's network connection management [1]. By applying publicly available compression algorithms such as gzip, compress and deflate, content encoding can save 75% off of text files (HTML, CSS, and JavaScript) and 37% overall on average [2]. Caching had been recognized as one of the most important techniques to reduce bandwidth consumption [3], much research work has been reported. There are mainly two categories of schemes to

improve HTTP cache mechanism, deploying cooperative proxies to construct Web cache, or building a P2P protocol overlay above HTTP [4-5]. However, the former cannot take immediate effect to HTTP clients in a local network environment where proxy is not deployed; while the latter adds an extra layer on top of HTTP which may not be compatible with the applications built directly on HTTP. Moreover, these techniques assume the environment of networks with stable transmission layer connections. In wireless ad hoc networks where links condition is not stable, to the best of our knowledge, there is no applicable mechanism reported in the literature.

Unlike previous work, this paper proposes a novel collaborating cache scheme, HTTP-CCC, which embeds into HTTP clients' own cache mechanism and the network layer's routing method. The basic idea is to let application layer behaves collaboratively with network layer to optimize the Web data transmission. In HTTP-CCC, when client makes the broadcast of routing message, it's up to date HTTP cache information will also be included. Therefore, with the propagation of routing messages, every client's latest cache information will be spread in the network. When a client initiates a request for a Web resource, it first search in its own cache, if not found it can check whether the resource is in his peers cache; then if the resource is found in local network, it can save the latency to retrieve data from origin server.

The rest of the paper is organized as follows. In Section 2, we discuss the mechanism and key data structures of HTTP-CCC. The performance in terms of cache hit ratio and average requests latency are discussed in Section 3. Section 4 concludes this paper and gives future work.

2 The HTTP-CCC Scheme

2.1 Composition of HTTP-CCC Client

We embedded a module, ShareAgent, in the HTTP layer platform of a terminal dedicated for sharing files among peers in local network. ShareAgent is responsible for maintaining shared file information, querying whether partner buffer contains needed document, querying whether the files in partner sharing buffer are out of date, responding request from partner terminal, managing the transport layer connection between file sharing partners, issuing and updating sharing files to the network layer module. The client's hierarchy containing ShareAgent is shown in Fig. 1.

When the upper layer application makes a request to obtain a web resource, after HTTP-CCC client obtains the resource URI, it first queries the resource table shown in Figure5, check whether this resource is in its own cache, and whether it is still usable. If both answers are true, it provides the resource to upper layer application. Otherwise, it checks whether that resource is in peers' cache. If so, it establishes a transport layer connection with the peer which has the resource, and requests it. After successfully receiving resource from the peer, client confirms it is valid and provides it to upper layer app. If the resource is not in the cache of any node in local network, the client initiates an HTTP request to origin server to get it.

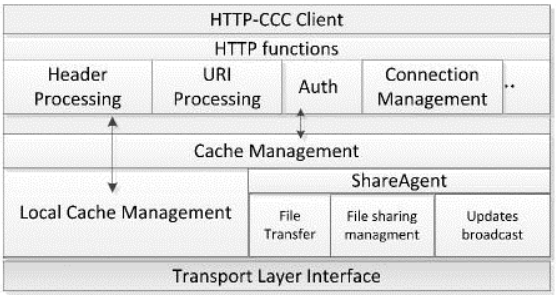

Fig. 1. HTTP-CCC hierarchy

When new resource is received or old resource is replaced by new one, cache needs to be updated. In the mean time, ShareAgent feeds key information of the resource, such as host name, resource name, the name of owner node, expiration time, etc, to the network layer module which updates routing information as shown in Fig. 1.

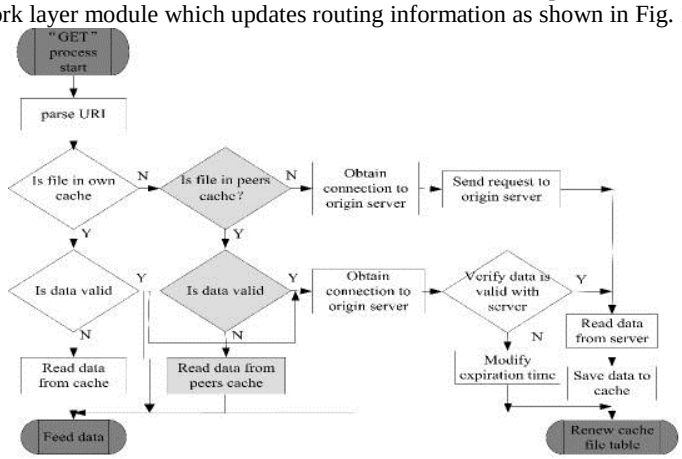

Fig. 2. Flow chart of HTTP-CCC's "GET" processing

If routing control module has not broadcasted the shared files information provided by ShareAgent last time, it will merge new information with previous one. That is, if there is new resource information, it will be directly added to the existing list. If there is newer version for an existing resource, the existing information will be updated to reflect the latest status. When routing control module initiates the routing broadcast, if the table of shared file information is modified, then this update will be released together with the routing information. When the subnet partners receive broadcast message containing shared file information, the routing control module will inform ShareAgent that there is update for shared files, then ShareAgent merges these updates with existing file table, to reflect the latest shared file information in local network. The simplified HTTP-CCC client's workflow is shown in Fig. 2.

2.2 Major Data Structures

We mainly added the following two data structures for HTTP-CCC client, i.e. the shared resources table and message of file table updates, as shown in Table 1 and 2. The former is for each client to manage their sharing resource, including the resource name, expiration date and file name. The latter is used to make broadcast of shared file updates conducted when making routing table broadcast.

Table 1. Sharing File Table.

No.	HOST	URI	Exp.Time	FileName
201	www.crshm.edu.cn	/lib/show.aspx	Mon, 1 Jun 2012, 01:00:00 GMT	./show.aspx
202	www.cdr.com	/nom/shd.html	Mon, 7 Jul 2012, 01:00:00 GMT	/usr/mh/buffer/ sshd.html
203	www.shmu.edu.cn	/hdg/hdd.jpeg	Mon, 9 Mar 2012 12:33:12 GMT	/usr/mh/buffer/ hdd.aspx

Table 2. Broadcast message for file table updating.

No.	HOST	URI	Containing Node	Exp.Time
301	www.crshm.edu.cn	/lib/show.aspx	MH10	Mon, 1 Jun 2012, 01:00:00 GMT
302	www.cdr.com	/nom/shd.html	MH10	Mon, 7 Jul 2012, 01:00:00 GMT

3 Performance Evaluation

The performance evaluation uses two parameters:average user request latency and cache hit rate. Average latency is defined as the average of the time span that begins at each client sends out GET request, and ends at the client successfully receiving the whole requested resource. Cache hit rate is defined as the ratio that compares clients' requests served by caches in local network to the total requests issued by clients.

3.1 Simulation Configurations

Experiments were conducted using a simulator built upon WSN Simulator1.1[6], which can simulate GET behavior of HTTP client in wireless ad hoc network, and the process that clients get shared file information while maintaining network's routing information. The simulated ad hoc network is 10x10 nodes mesh, except the boundary nodes, each node has four neighbors. We assume that nodes will not move, a gateway sits at the network boundary, and every node has to use gateway to access remote origin server. We also assume network bandwidth is always sufficient, the delay for a client to obtain a Web resource from its own cache is 5ms; to obtain a resource from cache of an adjacent node, the delay is 100ms; to obtain a resource from remote origin

server, and the delay is 2000ms. Based on statistical data presented in [7], the server randomly generates files of 100K different URLs, the average size is 50KB. 30% of the file less than 10KB, 60% of the file from 10KB to 100KB, and the remaining are 100KB to 1MB files. Assuming that each terminal every 10 seconds to launch a GET request, the request is consistently conform to the Zipf-like distribution [8], the access frequency for Web resource i is determined as the follows:

$$f=1\, i_{z} \cdot \sum_{mj} 1 \qquad . \qquad (1)_{-1}$$

where m is the number of the web documents in the system, and $0 \leqslant z \leqslant 1$ is the Zipf factor , and we set Zipf factor to 0.75.

3.2 Simulation Results

In simulation experiments, we evaluate the performance of HTTP-CCC by comparing it with normal HTTP where no cache sharing at all and a shared cache scheme HTTP-PRX that employs a proxy at gateway which is a common configuration in real world Internet autonomous domain. The cache replacement algorithm used by each scheme is LRU. Clients initiate 5M requests by using each of these three cache schemes, and we obtain the performance data by varying the size of clients' caches.

The simulation results are depicted in Fig. 3 and 4. As we expected, increasing the cache size increases the chance of a web page being cached in the client and proxy, hence increasing the cache hit ratio and decreasing the average user request latency. Compared to normal HTTP with non-cooperative cache, the scheme with one proxy improves the overall performance of clients in local network, while HTTP-CCC brings significant improvement to clients' performance and does not introduce excessive traffic to realize the file sharing.

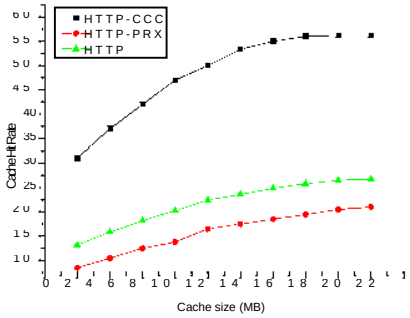

Fig. 3. Comparison on cache hit rates obtained on various cache size

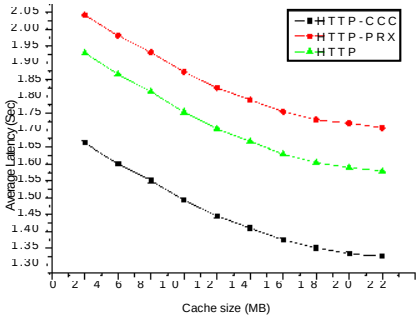

Fig. 4. Comparison on average request latency obtained on various cache size

4 Conclusions and Future Work

In this paper, we proposed a novel sharing cache scheme to improve HTTP client performance in mobile ad hoc network. This scheme let HTTP layer cache cooperate with routing control mechanism at network layer. The updated sharing file information will be broadcasted with routing table, thus no excessive communication overhead is incurred. When a client's request is served by its peer's cache, data transfer latency is significantly reduced.

Although we presented the performance results of HTTP-CCC transferring Web contents, in the future work, we will conduct research from energy perspective to evaluate HTTP-CCC's performance, as the client's energy consumption is also very important in wireless ad hoc network,. In addition, reactive routing such as AODV is also widely used in ad hoc network; we'll also conduct research to see how to let HTTP-CCC cooperate with such routing methods.

Acknowledgements

This work was supported by Shanghai Municipal Baiyulan Sci.&Tech Talent Fund (11BA1404700), and Shanghai Municipal Education Commission Sci&Tech Innovation Project (12ZZ157). This work was also supported by a project funded by the Priority Academic Program Development of Jiangsu Higher Education Institutions.

References

1. Venkata N. Padmanabhan; Jeffrey C. Mogul; Improving HTTP latency; Computer Networks and ISDN Systems, 28(1/2):25-35, December 1995
2. Xiaohui Chen, Weidong Wang, Guo Wei, Impact of HTTP compression on web response time in asymmetrical wireless network, In Proceedings of International Conference on Networks Security, Wireless Communications and Trusted Computing, NSWCTC 2009, v 1, pp 679-682, 2009
3 V. Jacobson. How to kill the internet, presented at SIGCOMM'95 Middleware Workshop, Aug. 1995. Available: ftp://ftp.ee.lbl.gov/talks/vj-webflame.pdf
4. A. DATTA, "Proxy-Based Acceleration of Dynamically Generated Content on the World Wide Web: An Approach and Implementation," ACM Transactions on Database Systems, vol.29, pp.403 - 443, 2004.
5. J. Wang, "A Survey of Web Caching Schemes for the Internet," ACM SIGCOMM Computer Communication Review, vol. 29, pp. 36 - 46, 1999
6. Available online: http://www.djstein.com/projects/WirelessSensorNetworkSimulator.html
7. Available online: http://theband.hiof.no/httpd-stat.html
8. L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and Zipf-like distributions: Evidence and implications," in Proc. IEEE INFOCOM, Mar. 1999, pp. 126–134