Journal of Theoretical and Applied Information Technology
20th November 2014. Vol. 69 No.2
© 2005 - 2014 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

DESIGN OF EFFICIENT CACHING ALGORITHMS FOR PEER TO PEER NETWORKS TO REDUCE TRAFFIC IN INTERNET

[1]S.SIVAKUMAR and [2]Dr.S.ANBU

[1]Research Scholar, Department of CSE, St.Peter's University, Avadi, Chennai-54
[2]Professor, Department of CSE, St.Peter's College of Engineering and Technology, Avadi, Cehnnai-54

ABSTRACT

Peer-to-peer (P2P) file sharing systems generate a major portion of the Internet traffic, and this portion is expected to increase in the future. We explore the potential of deploying proxy caches in different autonomous systems with the goal of reducing the cost incurred by Internet service providers and alleviating the load on the Internet backbone. P2P traffic is more complex than caching other internet traffic and needs several new algorithms and storage systems. The paper presents the design and evaluation of a entire, running, proxy cache for P2P traffic, referred to as PrCache. PrCache transparently intercepts and serves traffic from different P2P systems. A replacement storage system is projected and enforced in PrCache. Storage system is optimized for storing P2P traffic, and shown to outperform other storage systems. A new algorithm to infer the information required to store and serve P2P traffic by the cache is proposed. Extensive experiments to evaluate all aspects of PrCache using actual implementation and real P2P traffic are presented. Based on the analysis, proposed a new algorithm, called Optimized Least Grade Replacement (OLGR), takes recency, frequency, perfect-history, and document size under consideration for Web cache optimization.

Keywords : *Caching, LRU,LFU,Traffic,P2P,Network,Security*

1. INTRODUCTION

The increase in popularity of the World Wide Web (WWW) has introduced new issues such as Internet traffic and bandwidth consumption. Recently, much research has focused on improving Web performance by reducing the bandwidth consumption and WWW traffic. Most of the approaches have presented Web caching as the most beneficial solution for Web performance improvement. Web caching systems can lead to significant bandwidth savings, higher content availability, reduction of client latency and increase in server scalability and availability. Peer-to-Peer (P2P) file-sharing systems have gained tremendous popularity in the past few years. More users are continually joining such systems and more objects are being made available, enticing even more users to join. Currently, traffic generated by P2P systems accounts for a major fraction of the Internet traffic, and it is expected to increase. The sheer volume and expected high growth of P2P traffic have negative consequences, including: (i) significantly increased load on the Internet backbone, hence, higher chances of congestion; and (ii) increased cost on Internet Service Providers (ISPs), hence, higher service charges for all Internet users. A potential solution for alleviating those negative impacts is to cache a fraction of the P2P traffic such that future requests for the same objects could be served from a cache in the requester's autonomous system. Caching in the Internet has mainly been considered for web and video streaming traffic, with little attention to the P2P traffic. Many caching algorithms for web traffic and for video streaming systems have been proposed and analyzed. Directly applying such algorithms to cache P2P traffic may not yield the best cache performance, because of the different traffic characteristics and caching objectives. For instance, reducing user-perceived access latency is a key objective for web caches. Consequently, web caching algorithms often incorporate information about the cost of a cache miss when deciding which object to cache/evict. Although latency is important to P2P users, the goal of a P2P cache is often focused on the ISP's primary concern; namely, the amount of bandwidth consumed by large P2P transfers. Consequently, the byte hit rate, i.e., the number of bytes served from the cache to the total number of transferred bytes, is more important than latency. Moreover, P2P objects tend to be larger than web objects reducing the number of complete objects that can be held in a cache.

2. FEATURES OF WEB CACHING

There are three features of Web caching that make it attractive to all Web participants including users, network managers, and content creators :

- Caching reduces network bandwidth usage.
- Caching reduces user-perceived delays.
- Caching reduces loads on the origin server.

The Web caches are used due to the following two main reasons:

Reduces latency : The request is satisfied from the cache (which is closer to the client) instead of the origin server, it takes less time for the client to get the object and display it. Such Web sites appear more responsive.

Reduces traffic : Each object is got from the server once, and this reduces the amount of bandwidth used by a client. This results in saving of money if the client is paying for traffic, and keeps their bandwidth requirements lower and more manageable.

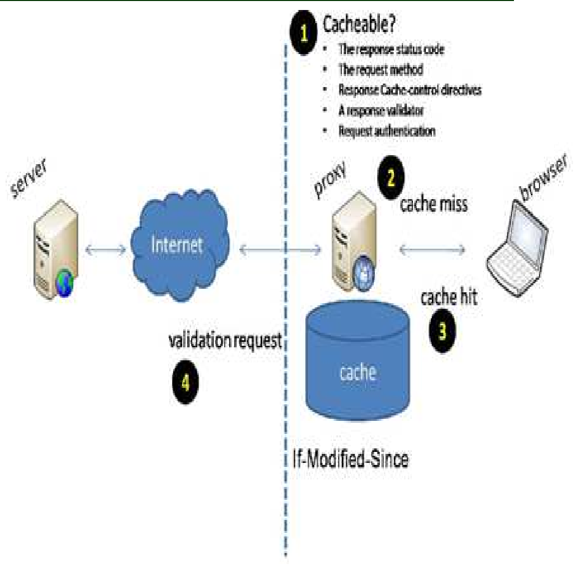

Figure 1 Basic Architecture Of Web Caching

Figure 1 shows the basic architecture of Web caching. The functionality of this architecture is described using four steps. Step 1 describes whether the data is cacheable or not. It provides the response status code, requested method, response validator and authentication methods. Step 2 checks whether any cache miss has occurred. A cache miss refers to a failed attempt to read or write a piece of data in the cache, which results in a main memory access with much longer latency. There are three kinds of cache misses: instruction read miss, data read miss, and data write miss. Step 3 performs a cache hit. Whenever the CPU requests for any data, then it first checks the cache whether the data is present or not. If it is present, the data is being taken from the cache memory itself and this is referred as cache hit. Step 4 performs the validation process; it can be used to check whether a cached response is still good after becoming stale. For example, if the response has a Last-Modified header, a cache can make a conditional request using the If-Modified-Since header to see if it has changed. The entity tag mechanism also allows for both strong and weak validation

3. PROXY CACHING ARCHITECTURE

A proxy cache server receives HTTP requests from clients for a Web object and if it finds the requested object in its cache, it returns the object to the user without disturbing the upstream network connection or destination server. If it is not available in the cache, the proxy attempts to fetch the object directly from the object's home server. Finally, the originating server, which has the object, gets it, possibly deposits it and returns the object to the user. The benefits of proxy caching are supposed to reduce network traffic and reduce average latency.

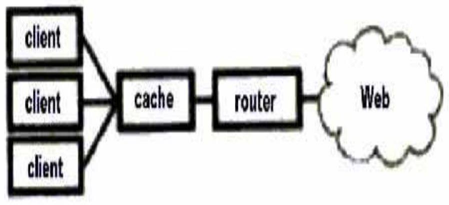

Figure 2 Standalone Proxy Configuration

A standalone proxy configuration is shown in Figure 2. One disadvantage of this design is that the cache represents a single point of failure in the network. When it is unavailable, the network also appears unavailable to users. Furthermore, another drawback is that all user Web browsers are manually configured to use the appropriate proxy cache. So, if the server is unavailable, all the users must reconfigure their browsers in order to use a different cache. A final issue related to the standalone approach is that there is no way to dynamically add more caches when needed.

4. DEFINITION OF P2P

A P2P network is a network that relies on computing power of it's clients rather than in the network itself. This means the clients (peers) will do the necessary operations to keep the network going rather than a central server. Of course, there are different levels of peer-to-peer networking:

Hybrid P2P: There is a central server which keeps information about the network. The peers are responsible for storing the information. If they want to contact another peer, they query the server for the address.

Pure P2P: There is absolutely no central server or router. Each peer acts as client and server at the same time. This is also sometimes referred to as "serverless" P2P.

Mixed P2P: Between "hybrid" and "pure" P2P networks. An example of such a network is Gnutella which has no central server but clusters its nodes around so-called "supernodes".

5.RELATED WORK

Partial and popularity-based caching schemes for web caching, and video streaming, have been proposed before proposes a popularity-aware Greedy-Dual-Size algorithm for caching web traffic. Because the algorithm focuses on web objects, it does not consider partial caching, which is critical for P2P caching due to large sizes of objects. Jin *et al.* consider partial caching based on object popularity, encoding bit rate, and available bandwidth between clients and servers. Their objectives are to minimize average start-up delays and to enhance stream quality. In contrast, our partial caching approach is based on the number of bytes served from each object normalized by its cached size. This achieves our objective of maximizing the byte hit rate without paying much attention to latency. A partial caching algorithm for video-on-demand systems is proposed in, where the cached fraction of a stream is proportional to the number of bytes played back by all clients from that stream in a time slot. Unlike our algorithm, the algorithm in periodically updates the fractions of all cached streams, which adds significant overhead on the cache. The algorithms in are not usable in caching P2P traffic because they require several inputs that are not available in P2P systems. Finally, our caching algorithm is designed for P2P systems, which contain multiple workloads corresponding to various types of objects. This is in contrast to the previous web and streaming caching algorithms which are typically optimized for only one workload.

P2P is a representative framework towards ISP-driven biased neighbor selection. In P2P, ISPs provide an interface to advertise preferred paths to P2P applications. This allows them to explicitly and efficiently implement traffic control between applications and network providers. Our work in this paper differs from these related works in important ways. First, our work adopts ISP caching as the basic scheme rather than biased neighbor selection, which potentially impairs the robustness of P2P overlay. Second, a collaborative caching scheme is transparent to end users while a locality-aware peer selection scheme explicitly interferes with P2P overlay construction. Finally, our work can build upon locality-aware P2P systems to further eliminate the inter-ISP traffic. The caching strategy is traditionally applied to the web for reducing user access latencies to web pages. With a remarkable increase of P2P traffic over the Internet backbone, several commercial cache products specifically designed for P2P traffic have appeared in the market, such as Cache Logic. In the view of caching algorithms, Hefeeda *et al.* have proposed a proportional partial caching scheme targeting on general P2P systems. However, these works mainly focus on independent server caching with the primary concern of improving the byte hit ratio. Neither the collaboration between ISPs nor cache server bandwidth constraints are included in the consideration. Our work not only provides a theoretic framework with respects to both storage and bandwidth utilization, but also incorporates concerns about ISP peering relation.

Dan introduced a collaborative caching scheme with awareness of peering agreements among ISPs to reduce the volume of transit traffic crossing ISPs. However, this model simplifies the scenario as a rate allocation scheme among multiple cache servers, which lacks investigations into the fundamental properties of inter-ISP traffic and other practical constraints in real-world P2P applications. Compared to this work, our study first develops an inter-ISP traffic model based on practical P2P application properties, and then elaborates on the design of a new collaborative caching scheme with respect to server storage and bandwidth constraints, peer selection strategies and ISP peering agreements. This paper provides a general and unified framework with practical constraints in mind.

6. CACHING ALGORITHMS

6.1 Least Recently-Used (LRU)

In the standard least recently used (LRU) caching algorithm for equal sized objects we maintain a list of the objects in the cache, which is ordered, based on the time of last access. In particular, the most recently accessed object is at the top of the list, while the least recently accessed object is at the bottom. When a new object comes in and the cache is full, one object in the cache must be pruned in order to make room for the newly accessed object. The object chosen is the one which was least recently used. Clearly the LRU policy needs to be extended to handle objects of varying sizes. LRU treats all documents equally, without considering the document size, type or network distance. It also ignores frequency information, thus an often-requested document will not be kept if it is not requested for a short period so LRU does not perform well in web caches. A scan, stream of multiple documents accessed only once, can force all the popular documents out of an LRU-based cache.

6.2 Least-Frequently-Used (LFU)

In least-frequently used method evicts the document, which is accessed least frequently. Least Frequently Used also has some disadvantages; the important problem with this method is cache pollution which means that a document that was formerly popular, but no longer is, will stay in the cache until new documents become more popular than the old one which was used. This can take a long time, and during this time, part of the cache is wasted. It assumes that probabilities are constant, but in practical it is not so. it also includes problem with implementation. Ideally, an implementation of the algorithm would keep a frequency counter for documents not in the cache as well as those present. On the scale of the web, this is prohibitive. However, if this is not done, the performance of the algorithm diminishes. Even if only the counters for the documents in the cache are kept, counters have to be updated continuously, even when no document needs to be replaced, incurring considerable overhead. It is also necessary to keep the documents sorted by frequency to be able to make rapid decisions at replacement time. This method has no notion of document size or cost of retrieval.

7. PROBLEM DEFINITION

The Internet is rapidly growing in number of users, traffic levels, and topological quality. The same time it is increasingly driven by economic competition. These developments provide the characterization of network usage and workloads more difficult, and yet more critical. The existing system has lot of problems for p2p web cache. Current P2P web caching schemes can solve the problems of proxy partly drawbacks respectively. The building block of substrate of existed P2P web caching systems is the notion of peer-group, in a number of nodes are organized by Distributed Hash Table algorithms or physical locations. DHT organization is not appropriate for the P2P web caching application, since it assigns file storage based on a random hash function, not based on files users already possess, in P2P web caching network, users share cache have, and it would be impractical to imagine users storing files other than their own. In the other organization based on physical locations, to find a specific object, peer must search the whole P2P space, is time-consuming. More, the above methods cannot make use of semantic information among the sharing contents of peers. The proposed system solved all the above problems and increases the performance of p2p web cache.

8. PROPOSED WORK

Caching is a technique first used by memory management to reduce bus traffic and latency of data access. Net traffic has increased tremendously since the starting of the 1990s. With the numerous increase of net traffic, caching techniques are applied to web caching to decrease internetwork traffic, user-perceived latency, and server load by caching the documents in local proxies. The analyzed both advantages and disadvantages of some current Web cache replacement algorithms including lowest relative value rule, least weighted usage algorithm and Least Unified-Value (LUV) algorithm. Based on our analysis, proposed a new algorithm, called Optimized Least Grade Replacement (OLGR), takes recency, frequency, perfect history, and document size under consideration for net cache optimization. The optimum recency coefficients were resolute by using 2- and 4- way set associative caches. The cache size was different from 32k to 256k in the simulation. The simulation results showed the new algorithm (OLGR) is better than LRU and LFU in terms of hit ratio (BR) and Byte Hit Ratio (BHR).

8.1 Proxy Server Implementaion

A Proxy server, runs with mentioned features, inherently helps speeder browsing of web pages with use of least grade page replacement algorithms. Server is successfully implemented with a few numbers of clients but it could be implemented for more of them. It is more reliable, more advantageous than the existing one uses the old Data structures concept. It will add a bigger network and also maintains load balancing system application is executable under any platform and with any number of clients too.

8.2 Web Crawler

A web crawler is a program, given one or more seed URLs, downloads the web pages associated with these URLs, extracts any links having them, and recursively continues to transfer the web pages identified by these links. Web crawlers are an important component of web search engines, used to collect the corpus of web pages indexed by the search engine. used in many other applications process large numbers of web pages, as web data mining, comparison shopping engines. Despite their conceptual simplicity, implementing high-performance web crawlers poses major engineering challenges due to the scale of the net. In order to crawl a considerable fraction of the surface web in a reasonable amount of time, web crawlers must download thousands of pages per second, and are typically distributed over tens or hundreds of computers. Their two main data structures – the frontier set of yet-to-be-crawled URLs and the set of discovered URLs – typically do not fit into main memory, efficient disk-based representations need to be used. Finally, the need to be polite to content providers and not to overload any particular web server, and a desire to prioritize the crawl towards high-quality pages and to maintain corpus freshness impose additional engineering challenges.

8.3 Web Cache Replacement Polices

According to the above conclusions, proposed a new replacement algorithm. The algorithm grades each in-cache document basing on its passed history, are recency, frequency, perfect-history and size. The set is full, the least grade document will be replaced, and then its grade will be stored in a Perfect History Grade Depository (PHGD) for future references. Due to the survey, can draw a conclusion the relatively most important factor is its recency, and frequency, perfect-history and size. Consider these four factors for grading because are relatively most important factors for real network traffic. The n-way set associative caches is employed. The least grade page replacement algorithm with optimized is given below.

```
ALGORITHM :: OptimizedReplaceLeastGradePage

/* In-cache document based on its passed history */
ICDR: In-Cache Documents Records
/* Discarded document in cache */
 PDDG: Previously Discarded Documents Grade
/* The de facto hit ratio beginning with 0 */
 F : Frequency of ICDR ;
R : Recency Set of ICDR;
/* R=< r1;r2;…;rn>*/
L : Length of document;
BG : Bonus Grade of document)
{ IF (document k in cache)
FOREACH doc in ICDR DO
WHILE (doc.F in F
& doc.R in R & Size(doc) in L)
doc ← newValue;
PDDG ←weightα , α ∈(0, 1);
ELSE
FetchDocFromOrigina();
DiscardInCachedDoc(); }
/*Fetch document k from original site*/
PROCEDURE: FetchDocFromOrigina()
{ IF(L is NOT NULL)
INSERT k into Cache;
UPDATE each ICDR;
k.BG = 0;
ELSE FOREACH g in grade DO
g =R 1δ + × F 2 δ + ×C 3 δ + × BG 4 δ ; }
 /*discard in-cached doc with the least grade*/
PROCEDURE: DiscardInCachedDoc()
{ PDDG ← PHGD.grade;
 INSERT k into Cache;
 IF ( PDDG of k in PHGD) k.BG←PDDG;
Delete its PDDG in PHGD;
ELSE k.BG=0;
Update each ICDR and PDDG; }
```

9. IMPLEMENTATION OF NEW SYSTEM

A principal activity of the development phase is coding and testing until the user requirement specification fulfilled by the each modules and component of the overall system. Other important activities include implementation planning, equipment acquisition and system testing. The development phase concludes with a development phase report and user review. A software application in generally implemented after navigating the complete life cycle method of a project. Various life cycle processes like requirement analysis, design module phase, verification, testing and at last followed by the implementation phase results in a successful project management. The software application is basically a web based application has been successfully implemented after passing various life cycle processes mentioned above. The software is to be implemented in a high standard industrial sector, various factors includes application environment, user management, security, reliability and finally performance are taken as key factors throughout the design part. These factors area unit analyzed step by step and the positive as well as negative outcomes are noted down before the final implementation. The applications validations area unit created, taken into account of the entry levels accessible in various modules. Possible restrictions like number data format, date formatting and confirmations for both save and update options ensures the correct data to be fed into the database. Therefore all the aspects are charted out and the complete project study is practically implemented successfully for the end users is shown in figure 3.

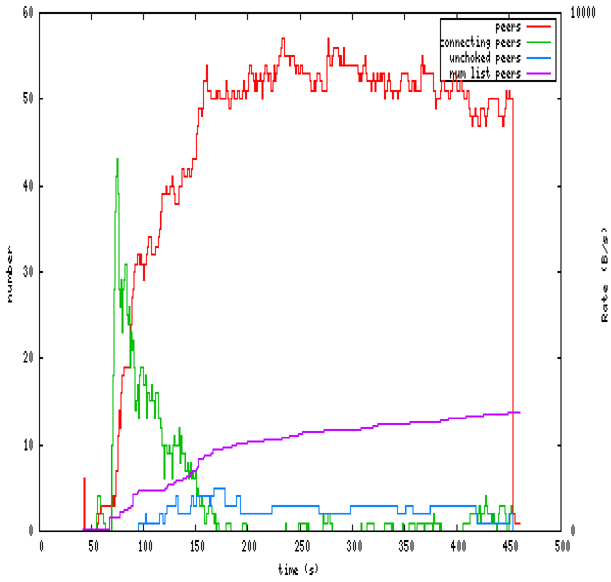

Figure 3. Cache Performance

10. CONCLUSION AND FUTURE SCOPE OF WORK

Using Web Caching in Internet reduces the latency, because the request is satisfied from the cache instead of the origin server. It takes less time for the client to get the object and display it and reduces the traffic. Each object is only gotten from the server once, it reduces the amount of bandwidth used by a client. This saves money if the client is paying by traffic, and keeps their bandwidth requirements lower and more manageable. Based on perfect-history LFU and LRU, proposed a new algorithm (OLGR) by considering recency, frequency, perfect-history and size in replacing policy. Experimental results show the proposed algorithm can reduce network traffic and latency of data access efficiently. The potential gain of cooperative caches are analyzed for P2P traffic. Proposed two models for cooperation are: 1) among caches deployed in different autonomous system and 2) among caches deployed within an oversized AS. In each models, caches collaborate to save bandwidth on expensive WAN links. The traces describe object requests would have been seen by many caches if were deployed in ASs operating in different geographical regions and have different number of clients. Many trace-based simulation experiments are designed to rigorously analyze various aspects of cooperative caching. Results show cooperative caching is viable for P2P traffic because it could improve the byte hit rate by up to 330 percent in some cases. Considering the large volume of the P2P traffic, even one percent improvement in computer memory byte hit rate accounts to saving in the order of terabytes of traffic on the expensive WAN links. In addition, proposed simple models for object replacement policies in cooperative caching systems.

In future, plan to fully implement our scheme, and further optimize the similarity measurement algorithms. Designing a cooperative P2P caching protocol, proxy caches in different cooperate to serve requests for each other in the hope of minimizing WAN traffic. Designing efficient cache zoning techniques different workloads occupy different zones in the cache with different cache spaces. The same cache could also be tuned to minimize the outgoing traffic by allocating a larger zone for large video objects. In all cases, different algorithms might work better for different zones and could be run in the same cache concurrently. Cache zones might also cooperate by evicting the least valuable object from any of them to optimize a common desired metric.

REFRENCES:

[1] K. Gummadi, R. Dunn, S. Saroiu, S. Gribble, H. Levy, and J. Zahorjan, "Measurement, modeling, and analysis of a peer-to-peer file-sharing workload," in Proc. of the 19th ACM Symposium on Operating Systems Principles (SOSP'03), Bolton Landing, NY, USA, Oct. 2003, pp. 314–329.

[2] T. Karagiannis, A. Broido, N. Brownlee, K. C. Claffy, and M. Faloutsos,"Is P2P dying or just hiding?" in Proc. of IEEE Global Telecommunications Conference (GLOBECOM'04), Dallas, TX, USA, Nov. 2004, pp.1532–1538.

[3] T. Karagiannis, P. Rodriguez, and K. Papagiannaki, "Should internet service providers fear peer-assisted content distribution?" in Proc. of the 5th ACM SIGCOMM Conference on Internet Measurement (IMC'05), Berkeley, CA, USA, Oct. 2005, pp. 63–76

[4] S. Zhao, D. Stutzbach, and R. Rejaie, "Characterizing files in the modern Gnutella network: A measurement study," in Proc. SPIE/ACM Multimedia Computing and Networking (MMCN'06), San Jose, CA, Jan. 2006.

[5] S. Saroiu, P. K. Gummadi, and S. D. Gribble, "Measuring and analyzing the characteristics of Napster and Gnutella hosts," ACM/Springer Multimedia Syst. J., vol. 9, no. 2, pp. 170–184, Aug. 2003.

[6] J. Pouwelse, P. Garbacki, D. Epema, and H. Sips, "The BitTorrent P2P file-sharing system: Measurements and analysis," in Proc. 4th Int.Workshop on Peer-To-Peer Systems (IPTPS'05), Ithaca, NY, Feb. 2005, pp. 205–216.

[7] Peicao, Snady Irani(1997), 'Cost-Aware WWW Proxy Caching Algorithms' in Proceedings of the USENIX Symposium on Internet Technologies and Systems.

[8] Thompson. K, Miller. G and Wilder. R(1998), 'Wilde-Area Internet Traffic Patterns and Characteristics', in Proceedings of third International Conference Web caching.

[9] Seda Cakiroglu, Erdal Arikan (2003), 'Replace Problem in Web Caching', in Proceedings of IEEE Symposium on Computers and Communications.

[10] S. Ren, E. Tan, T. Luo, S. Chen, L. Guo, and X. Z. M. Corporation, "TopBT: A topology-aware and infrastructure-independent BitTorrent client," in Proc. IEEE INFOCOM, Mar. 2010.

[11] M. Piatek, H. V. Madhyastha, J. P. John, A. Krishnamurthy, and T. Anderson, "Pitfalls for ISP-friendly P2P design," in Proc. ACM HotNets, Oct. 2009.

[12] Z. Liu, C. Wu, B. Li, and S. Zhao, "UUSee: Large-scale operational on-demand streaming with random network coding," in Proc. IEEE INFOCOM, Mar. 2010.

[13] Jason E. Bailes, Gary F. Templeton: Managing P2P Security

[14] Geetha Ramachandran, Delbert Hart: A P2P Intrusion Detection System based on Mobile Agents

[15] Salman A. Baset, Henning Schulzrinne, September 15, 2004 An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol