BS-11-1

Performance comparison of request models for a P2P based streaming Content Distribution Network

Karl-André Skevik
Department of Informatics,
University of Oslo
{karlas@ifi.uio.no}

Jiro Katto
Graduate School of Science and Engineering,
Waseda University
{katto@katto.comm.waseda.ac.jp}

Vera Goebel and Thomas Plagemann
Department of Informatics,
University of Oslo
{goebel,plageman@ifi.uio.no}

I. Introduction

Streaming of high-quality video over the Internet is a research field with a long history. One of the more recent developments is in the adaption of techniques from Peer-to-Peer (P2P) based file distribution networks, to reduce resource requirements. We have examined different design alternatives for such a system and show through simulations that a streaming system can perform well even with a very simple infrastructure.

The system allows content providers to offer content from their own servers, in a way similar to the WWW. As a result there can be ongoing downloads of a large number of different files, in addition to users viewing the same file, but at different points in the file.

Files are divided into blocks for which clients make requests, either directly to the server or to hosts which has previously downloaded them. Each client application is responsible for making requests in a timely manner to ensure that the data is presented to the user without delays or interruptions.

The basic design goals of the system are to reduce the cost of serving for the content providers and to reduce traffic at sites with many users, or ISPs. The system allows for the use of proxy caches at ISPs and large sites, to reduced resource requirements through caching. These proxies also reduce the negative effects on the network when a large number of clients are unable to accept incoming connections, typically due to firewalls[1].

II. Request models

In our system, clients make requests in a way comparable to how a browser works in the WWW; either directly to a server, or alternatively, for users at large sites or ISPs, via a content cache. However, instead of always returning the requested data, the client might be redirected to a peer which has previously downloaded the requested content. We have examined different ways of managing the information in these redirection lists.

The simplest approach adds entries to this list in a FIFO manner, after a client has successfully downloaded a file block. A problem with this approach is the quality of the entries; if old peers are removed when new ones are added, there will likely be hosts in the network with the content available, but these will never be used after they have been removed. Hosts which are on the list might also have left the network. An improved version of this model uses techniques such as adding peers which upload data to ensure that long-lived hosts are kept on the list, and not adding hosts which are behind a firewall. The effects of varying the size of the list were also examined.

We compared these simple variations to a more comprehensive system where knowledge about the network is perfect, and downloads can be made from any available host with the required block. Requests are still done in the same way, but the redirection list returned is based on complete and updated knowledge of the network. The infrastructure cost for maintaining this information would likely be much higher, and the infrastructure more complicated. In the real world it would not be possible to have a perfect overview of the network all the time, but we use this model to simulate the best case performance of this approach.

III. Simulation results and discussion

We simulated the performance of the request models under different conditions and measured resource usage. The simulations used a request pattern spanning seven days, for a streaming server containing 1000 different files, generated with *MediSyn*[2]. The requests were distributed randomly among roughly 9000 clients in a network created with *Inet*[3]. The network had 27 subnetworks with a large number of client hosts, which we defined as ISPs, and where we placed caching proxies. We evaluate the efficiency of the request models by the reduction in resource requirements. The load on the server should be reduced by having clients upload previously downloaded content. A consequence of of moving the load of serving to clients is an increased load on ISPs, but it should be reduced when possible through caching.

The final results are presented in Figure 1. Of the request models we examined, the most interesting are shown in the figure; the unimproved simple request model, the improved simple model (Simple), and the ideal design (Ideal). The first column of numbers shows the basis for comparison; the total amount of data transmitted for the simple, unimproved request model. The other columns show the changes compared to these values. The other columns shows the percentage of reduced bandwidth requirements. The bandwidth requirements are shown for four different points in the network; data downloaded from the server (Server), data entering the ISPs

due to downloads by internal clients (ISP - down), data leaving the ISP networks due to requests made by external clients (ISP - up), and the total amount of data uploaded by all clients in the network (Clients - up). The load on the ISP caches is also shown (Cache load). One of the more significant parameters is the time clients stay connected. The figure shows the results when clients leave immediately after completing a download, when they stay for an additional 90 minutes, and when they stay for an extra 6 hours. The total amount of data received by clients is $16.96TB/week$.

	+0	+90 min		+6h	
	TB/week	Simple	Ideal	Simple	Ideal
Server	1.86	0.54	0.53	0.52	0.52
ISP - down	2.88	0.69	0.66	0.46	0.38
ISP - up	6.70	0.67	0.70	0.36	0.31
Clients - up	5.26	1.78	2.13	2.31	2.64
Cache load	6.70	0.67	0.63	0.36	0.24

Fig. 1. Simulation results

There is no big change in the server load; both designs roughly halve this value. For the load on the ISPs there is a slightly difference, especially when clients stay for a long time. The variation is not so big however, and as can be seen in one case, the ISP load is not necessarily lower even with better information about the network. The clients ultimately select which peers to connect to, and this choice affects the load on the ISPs. The most significant difference is in the load on the clients; there is a bigger potential for increasing the amount of sharing done by the clients when the location of all clients is known. The load on the caches can also be significantly reduced by having clients bypass the ISP cache. This will however not be possible if these hosts are behind a firewall, and would also make it more difficult to control bandwidth usage.

The simulations also show that the factor which has largest influence on the resource requirements is the time clients stay in the network. The benefits of increasing this time by an extra 90 minutes after completing a download are larger than the effects of any of the different design variations we examined. The size of the peer list was also found to have a significant influence on the performance of the system in the simple request model.

A. Alternative download locations

We have so far mainly been looking at resource requirements. There are other factors which also are important, especially the service quality experienced by users. Variations in connection quality means that having multiple download locations for a block would be beneficial. If performance from one is too low, another can be used.

To see how well the two alternatives would perform in this regard, the number of possible download locations for each request was examined. With a complete overview of the network, we find that the median value of available download locations lies between 10 and 15 peers. The simple approach has information on around 4 of these. Network conditions and peer quality would determine the ideal number of peers. The size of the peer list could be adjusted as appropriate.

B. Firewalls

In the results above we used a network where 40% of the hosts were behind a firewall[4]. A higher number of hosts using firewalls reduces the potential for resource sharing. Fewer hosts can contribute resources and the difference between the design alternatives examined here also diminishes. When all hosts are behind firewalls, only the ISP caches remain.

With fewer hosts behind a firewall, there is increased resource sharing from the P2P techniques. Our design addresses this problem by allowing external hosts to connect to the cache, even when internal hosts are unavailable; the number of hosts using firewalls is only likely to increase in the future.

IV. Caveats

The behavior of clients is difficult to model, since variations in implementations, network conditions, and user actions are possible. This does not affect the server load, but the traffic in and out of ISP networks. The values returned in the redirection list does however provide a limited means of influencing this, by for example favoring new, rather than long-lived hosts in the list of hosts which is returned to clients.

V. Conclusions

We have examined different design alternatives for a streaming service. Our results show that even with a simple infrastructure, it is possible to significantly reduce resource requirements, almost to the level of an approach requiring a more complicated infrastructure.

The size of the list used to keep track over available peers was found to be an efficient way of tuning the performance of the simple model. Allowing for the cache to be bypassed was also identified as a way to reduce server load on caches, should they prove to be a bottleneck in the system. However, the most important factor for reduction of resource requirements is the time clients stay in the network. Any design should allow for freeloader prevention techniques to be used, but it is difficult to measure the efficiency of these in simulations, since they depend upon human behavior. We are now working on implementation, to better measure factors like these in the real world.

References

[1] K.-A. Skevik, V. Goebel, and T. Plagemann, “Design of a hybrid cdn,” in *Interactive Multimedia and Next Generation Networks: Second International Workshop on Multimedia Interactive Protocols and Systems, MIPS 2004*, November 2004, pp. 206–217.

[2] W. Tang, Y. Fu, L. Cherkasova, and A. Vahdat, “Medisyn: a synthetic streaming media service workload generator,” in *Proceedings of the 13th international workshop on NOSSDAV*. ACM Press., 2003.

[3] J. Winick and S. Jamin, “Inet-3.0: Internet topology generator,” University of Michigan., Tech. Rep. CSE-TR-456-02, 2002.

[4] K.-A. Skevik, V. Goebel, and T. Plagemann, “Analysis of bittorrent and its use for the design of a p2p based streaming protocol for a hybrid cdn,” Department of Informatics, University of Oslo, http://www.ifi.uio.no/dmms/papers/129.pdf, Tech. Rep. 310, 2004.