BANDWIDTH MANAGEMENT IN A CENTRALIZED LARGE SCALE DISSEMINATION NETWORK – A SIMULATION STUDY

KONSTANTINOS G. ZERFIRIDIS

Department of Informatics
Aristotle University of Thessaloniki, 54124, Greece
zerf@csd.auth.gr

HELEN D. KARATZA

Department of Informatics
Aristotle University of Thessaloniki, 54124, Greece
karatza@csd.auth.gr

Abstract: The evolution of the Internet gave rise to new applications. The need to disseminate high volumes of data to numerous users along with the evolution of Peer-to-Peer systems, introduced a new alternative to the traditional client-server paradigm. File sharing networks became the platform for thousands of users to share content. Users often turn to these networks to find highly anticipated, newly released software or video files which sometimes are of considerable size. However, increased mean response time or even network failures can be observed in such P2P systems, often caused because of uneven flow of data and intersperse congestion points. In this paper, the structure of Peercast, an agent based dissemination network, is presented. Several simulation experiments were conducted and their results are examined in order to determine how the network's bandwidth can be best utilized during the dissemination process.

keywords: peer-to-peer, network modeling, middleware, grid computing

1. INTRODUCTION

As bandwidth availability is increasing, users' demands change constantly. Today the internet is used to download music, software, video clips and other files of considerable size. This can saturate the network quickly, clogging the host computer. Such is the case for example when any highly anticipated software is released and several people are trying to download it at the same time. This became known as the middle night madness problem [Schooler and Gemmell, 1997]. Conventional FTP servers can no longer serve as a way of distributing large amounts of data. For example, modern Linux distributions can span more than one CD. Assuming that the server's bandwidth is 1 MBit/sec and the requested software is distributed in 2 ISO CD images, the server could only serve about 50 clients in a period of one week, even in the theoretical case that no errors occur. Mirroring the required content on several dispersed servers, cannot always compensate for the rapid traffic increase.

In such cases, traditional ways of making data available to the masses do not apply to modern demands. The main architecture used for casting data through the Internet is IP multicast, which mainly targets real-time non-reliable applications. It extends the IP architecture so that packets travel only once on the same parts of a network to reach multiple receivers. A transmitted packet is replicated only if it needs to, on network routers along the way to the receivers. Although it has been considered as the foundation for Internet distribution and it is available in most routers and on most operating systems, IP multicast has not so far lived up to early expectations. Its fundamental problem is that it requires that all recipients receive the content at the same time. The most popular solution to this problem was to multicast the content multiple times until all of the recipients obtain it. Some of the other drawbacks of IP multicast include small address space (26-bit), need of large routing tables and lack of congestion control and reliable transfer control.

Several algorithms arise for membership management and packet replication to solve problems such as server implosion from client side NACKs (negative acknowledgments), server explosion from maintaining status of the download process for each client and managing downloads requests by users connected with different bandwidths. Forward Error Correction (FEC) has long been used for the dissemination of static data as it provides graceful degradation of performance in the presence of packet losses. Its greatest disadvantage is that it is very demanding on CPU and memory [Rizzo, 1997].

Although IP multicast might be considered ideal for applications that require relatively high and constant throughput but not much delay, it is not suitable for applications that may tolerate significant delays but no losses. This is the case with file distribution. These days, a new way of disseminating files emerged. File sharing networks [Parameswaran et al,

 ISBN 3-936150-25-7

2001] are perhaps the most commonly used Peer-To-Peer applications. P2P systems existed since the birth of the Internet, but as bandwidth, computational power and great storage capacity became available, their popularity increased. Such systems have been used for diverse applications: combining the computational power of thousands of computers, forming collaborative communities, instant messaging, etc.

P2P file sharing networks' main purpose is to create a common pool of files where everybody can search and retrieve any shared files. Depending on the algorithm used, these sharing networks can be divided in two groups. Networks that maintain a single database of peers and their content references are known as centralized. Such file sharing networks [Shirky, 2001] have several advantages, such as easy control and maintenance, and some disadvantages as, for example, server overload. On the other hand, dynamically reorganizing networks such as Gnutella [Ripeanu, 2001], have a rather more elaborate service discovery mechanism, avoiding this way the use of a centralized server. Those kinds of networks are known as decentralized, and their main advantage is the absence of a single point of failure. However, the lack of a coordinating server may lead to inefficient use of the network's resources.

Along with the widespread use of those networks, several problems emerged. A study conducted at the Xerox Palo Alto Research Center showed that 70% of Gnutella users provided no files or resources to the system and that 1% of the users were providing half of the total system resources [Adar and Huberman, 2000]. This created network bottlenecks causing further inter-domain jamming. File sharing networks had never been designed for file dissemination. Nevertheless people turn to them to find highly anticipated files, when the official server stops responding due to high demand. Extensive research has been done about how existing P2P networks operate over time and how they can be optimized [Markatos, 2002; Ripeanu et al, 2002]. However, the dissemination process of highly anticipated files on P2P networks over unreliable network connections remains unexplored. Peercast, a P2P network first presented in [Zerfiridis and Karatza, 2003], is designed to assist the dissemination of a file in a heterogeneous network of clients. The purpose of this paper is to show how the Peercast performs under different bandwidth utilization scenarios using a simulated model of the network. The drown conclusions can be used to optimize other P2P file sharing networks as well.

The structure of this paper is as follows. In section 2 PeerCaster, the agent based infrastructure used, is presented. Section 3 shows Peercast's structure, along with its latest extensions. Section 4 elaborates on the network's simulation model and in section 5 the results and drawn conclusions are summarized. Finally, section 6 presents plans for further research.

2. THE INFRASTRUCTURE

Software agents are programs that act on behalf of clients. They are able to perform predefined tasks that are assigned to them. This is done either with or without the supervision of the user, depending on the given job. Mobile agents have an additional property [Chess et al, 1995]. The ability to transport themselves on different systems after being executed, carrying with them their program code, current state of execution and any data which was obtained. This gives them the unique capacity of living on a distributed network rather than on a distant stationary system, and to take advantage of the services that each host has to offer locally. Furthermore, mobile agents allow proprietary code to be used on the hosts, allowing complete customization of the retrieved results.

The unique properties of the mobile agents give them the edge in comparison to the traditional client-server paradigm. They have been used in the past instead of protocols [Joy, 2000], for file transfer [Spalink et al, 1999] and as a dynamic system for information discovery and retrieval. There are many applications that would benefit from the use of mobile agents as a vehicle for getting around bottlenecks. PeerCaster [Zerfiridis and Karatza 2002] is a platform implemented in Java that uses mobile agents as a vehicle delivering great amount of static data to users on a heterogeneous network. This is done by splitting the data into small packets, loading them onto mobile agents and releasing them to the peers where the payload is delivered and continue according to their itinerary. The coordination and communication overhead is acceptable considering the scalability that can be gained by the dynamic nature of the agents. As they can operate asynchronously and independently of the process that created them, they do not need to report back to the server. In this paper, PeerCaster was used as a mean of distributing high-demand files without clogging the host computer. This system could be integrated as part of a P2P file transfer network, or it could be used as an alternative to multicast for large files with great demand, such as the release of a new version of popular software as depicted in [Schooler and Gemmell, 1997].

3. THE NETWORK

When a file needs to be downloaded by more clients than the server can handle, alternative algorithms have to be utilized. The naive way of avoiding retransmissions is to pipeline the file through all the clients. But this is not a viable solution because clients might have to indefinitely wait to be served.

The proposed algorithm uses centralized approach in order to avoid uneven flow of data and intersperse congestion points which can compromise inter-domain quality of service. The server can upload the file to a certain number of clients simultaneously. When the server successfully uploads a file to a client, it keeps a reference of this client to a list. The server has the responsibility of maintaining a complete list of served clients that are currently on-line.

Although the server has a queue, most of the clients are expected to find this queue full. This is the case especially at the beginning of the dissemination process, as clients arrive more rapidly than the server can handle. When this happens, the server sends to the client a short (up to 100 entries) list of randomly selected peers that downloaded successfully the file, and are known to be on-line. This way, the new client can download the file from a peer that was already served, removing the congestion from the server. If the client cannot be served by any of those peers it requests another list of clients in order to continue searching for service. If the server is contacted more than 10 times, or the returned list is less than 100 entries long, the client waits for a certain period of time before it contacts the server again. If a client cannot contact a peer either because it is off-line or because it is unreachable due to network failure, it sends to the server a short message so that the server can update its database.

As it was mentioned earlier, when a client finishes the download it acts as a server for other clients. Similarly to the server, the clients have a short queue. If a client *A* requests the file from a client *B* that has it, and client *B* can not serve client *A* immediately, *A* is queued. If the queue is full, client *B* dismisses client *A*. When a client finishes the download, it sends a short report message to the server in order to include it in its list.

When a peer leaves the network, the list maintained at the server is left in an inconsistent state. In order to compensate for this, clients that are not able to contact other peers, report to the server that this peer is no longer reachable. If the server receives several such reports for the same peer, it removes its reference from the list.

In order to utilize all the available upload bandwidth, a single peer can serve several clients concurrently. Additionally, each client can initiate multiple concurrent download connections in order to utilize all the available download bandwidth. At the end of the transfer, the downloading client sends a message to the server in order to be included in the list.

Several issues arise about the performance of this algorithm under different network conditions in a heterogeneous network of clients. For example, what is the benefit of allowing several clients to download from a single peer? It will reduce the average waiting time, but what consequences will it have on the downloading speed and in the long run on the total number of served clients? On the other hand, if the clients are able to download from multiple peers simultaneously, how will it affect the system's dissemination process? This can in theory utilize all the download bandwidth of client and therefore, reducing the mean response time.

4. SIMULATION MODEL

In this section details are presented about the simulation model for the proposed network, and show how different strategies might affect the dissemination process. An object-oriented model of the network was used for the simulation. The programming language used was Java. The system was populated with clients arriving according to the exponential distribution. The simulation period was set to be 2 weeks (1209600 seconds). During the first week the mean interarrival time was incremented linearly from 5 to 20 sec in order to simulate demand on a highly anticipated file. For the second week the exponential distribution was used with 20 sec mean interarrival time. The file size was set to be 650MB (the size of a full CD).

All the clients that populated the system were set to have broadband connections to the Internet, resembling cable modems and DSL. This is done in order to use a realistic model. As in many cases, such connections have different download and upload speeds. Four different categories of users were used. The first category (10% of the clients) had download and upload speed of 256 Kbps, the second (40% of the clients) had 384 Kbps and 128 Kbps respectively, the third (20% of the clients) had 384 Kbps download and 384 Kbps upload speed, and the fourth (30% of the clients) had 1.5 Mbps and 384 Kbps respectively. This configuration is a theoretical model, and is used to compare how the same network performs under different conditions.

These kinds of clients are always on-line. However, they are not expected to share the file for ever. Therefore they were set to leave the dissemination network with exponential distribution and mean time of four days. The server was set to have 1.5 Mbps download / 384 Kbps upload connection (resembling a DSL user) to the net and never to go off-line. As the server is only uploading files, the simulation would have given the same results if the server had 384/384 connection to the net (third category). An additional difference between the server and the clients is that the server keeps a list of all the served

Table 1. Mean response time

	Concurrent upload streams			
Concurrent download streams	*1 slot*	*2 slots*	*4 slots*	*8 slots*
1 slot	134785	164878	193065	283042
2 slots	118351	127631	157199	251143
4 slots	117844	122076	149080	241583
8 slots	122760	123780	145910	239038

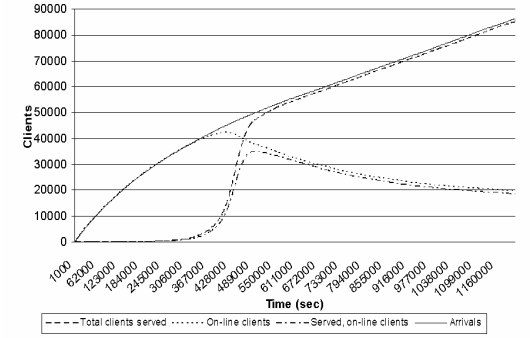


Figure 1. Network's status over time, 4 upload and 4 download streams

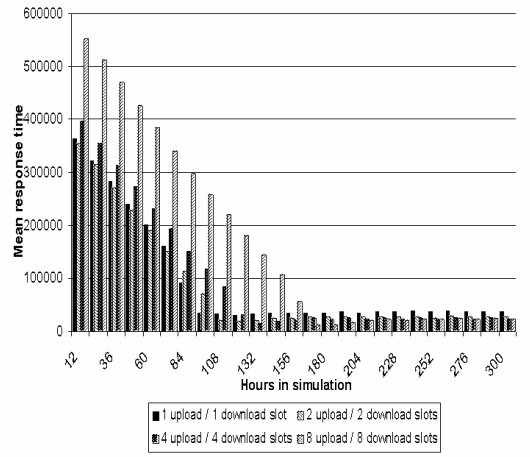


Figure 2. Mean response time in 12–hour intervals according to each client's arrival

clients that are currently on-line. This list is constantly updated.

The actual connection speed between two clients is calculated at the beginning of each session, taking into consideration the theoretical maximum speed they could achieve and an exponentially distributed surcharge, in order to simulate additional network traffic and sparse bottlenecks. If a new client cannot be served or queued immediately, it waits for 600 seconds and retries. In order to simulate peers that are not willing to assist in the dissemination process, 10% of the clients were set to go off-line immediately after they finish downloading the file. This is expected to significantly decrease the performance of the dissemination process. Nevertheless it is a behavior that can be expected.

If a client cannot contact another peer, it sends a message to the server that this peer is unreachable. When the server receives three such messages from different clients for the same peer, that peer is removed from the list. This is done to avoid removing a client from the list just because one connection could not be established. However, if a client that is participating in the dissemination process is not requested to serve another peer for over 1200 seconds, it contacts the server to verify that it is still included in the server's list. This is done as a countermeasure to accidental removals from the list.

As it was mentioned earlier, the behavior of this network can change significantly under certain conditions. The system's performance is investigated at the beginning (2 weeks) of the dissemination, under different conditions. Our focus is on how the system behaves under different bandwidth loads. More specifically, the simulations tested the system's performance when 1, 2, 4 and 8 concurrent upload streams were used. In each case, a serving-client was able to serve one or multiple peers at the same time by sharing the client's bandwidth. By sharing the bandwidth to multiple peers the full bandwidth is utilized, but the connection speed decreases. Additionally, the system's performance was tested with clients that were able to download from 1, 2, 4 and 8 serving-clients simultaneously. If a client can not use all its available download streams it retries to find an available serving-client after 600 seconds. With this approach, the client's download bandwidth can be utilized to the maximum. On the other hand, several serving-clients are occupied by serving one client, diminishing this way overall network performance.

5. SIMULATION RESULTS AND CONCLUSION

In total 16 simulations were done. Table 1 reveals significant differences between the tested scenarios. The increased mean response time in all cases can be explained as the clients that arrive early on the dissemination process have to wait for a long period of time to be served. When the rate of arrivals balances with the rate of clients being served, the mean response time stabilizes to lower levels. This balance occurs when a critical mass of serving-clients has been built. The critical mass is reached when the number of served clients in the system starts to decline (figure 1). Therefore, clients arriving later in the system benefit from a faster service. This is depicted in figure 2 where mean response time is shown in 12 hour intervals according to each client's arrival in the system.

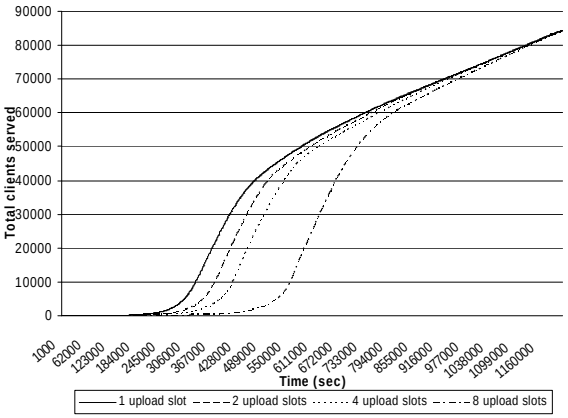


Figure 3. Total clients served over time (1 download stream)

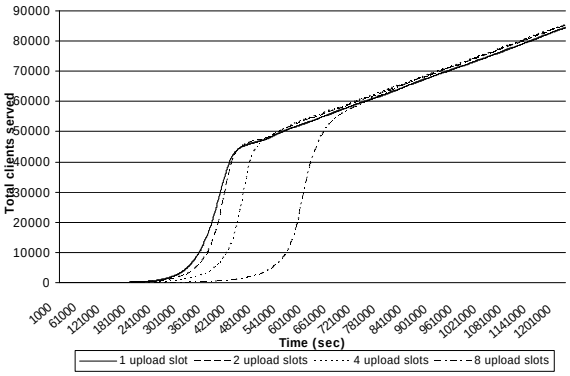


Figure 4. Total clients served over time (8 download streams)

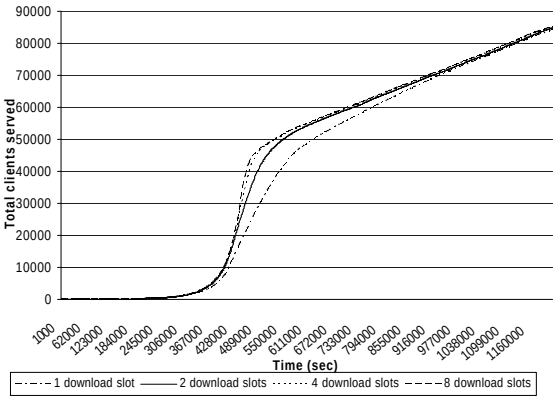


Figure 5. Total clients served over time (4 upload streams)

Table 1 shows that using 8 concurrent upload streams for each client increased dramatically the mean response time in all cases. Additionally, figure 2 shows that although the mean response time for the 8/8 case is increase at the beginning of the dissemination process, after the critical mass is reached it decreased dramatically, even to lower levels than those of the other cases,. The explanation for this is that when the critical mass has been built, there are enough serving-clients in the system to accommodate the rest of the peers and the new clients as they arrive. Therefore, the multiple upload streams utilize the client's upload bandwidth to the maximum and assist the peers in finding service immediately as they arrive in the system.

On the other hand, multiple upload streams have the opposite affect at the beginning of the dissemination process. Sharing the serving-client's bandwidth to multiple peers reduces the downloading speed. This increases the response time, and therefore the critical mass is built much later in the dissemination process. This is shown in figures 3 and 4, where as the upload streams increase, the time period in which the system reaches the critical mass increases also. Comparing these two figures reveals also that the use of multiple download streams accelerates the build of the critical mass.

Figure 5 reveals that for the 4 upload streams case, the more download streams used, the sooner the critical mass is built. This can be seen in table 1 as well, where shorter mean response time is observed as the download streams increase. This is also the case for the 8 upload streams scenario. However, table 1 shows that this is not valid for the other two sets of tests. For example, the 1 download / 4 upload streams test produced reduced mean service time in comparison with the 1 download / 8 upload streams case. This shows that although multiple download streams have a positive affect on the utilization of the given bandwidth, they can also be accountable for the depletion of network resources.

Overall, the system's behavior can change dramatically by using different bandwidth utilization scenarios. Increased number of download streams helped in all the cases to the faster build of the critical mass. However in some cases this was the reason for an increase in the mean response time. On the other hand the use of multiple upload streams increased the mean response time before the build of the critical mass, but afterwards it decreased the mean response time. We propose the use of a dynamically changing number of upload and download streams as the dissemination process develops. The server can estimate when the critical mass is reached by the size of the list of serving-clients that it maintains. Before that point, using 2 upload and 4 download streams can speed up the build of the critical mass. After that point, by gradually increasing the upload and download streams to 8, is expected to decrease the mean response time. Simulation results of this scenario are under way.

6. FUTURE WORK

The use of a decentralized approach, as described in [Zerfiridis and Karatza, 2003], is also investigated in order to determine the best upload/download bandwidth utilization scenario in that case. Additional simulation experiments are under way, using distributions varying with time for more

realistic long-run simulations, as depicted in [Karatza, 2002]. Peercast is an evolving platform. For the current P2P network implementation we used a monolithic approach: all the data has to be sent to a client, before this client starts sending it to another peer. A new version that replicates groups of 256KB packets, to adjacent peers as they arrive, is under way. This is expected to alleviate the problems that are caused from peers that go off-line immediately or soon after they finish downloading the requested file. The synchronization between the peers is done in predetermined time intervals, called epochs [Karatza and Hilzer, 2001]. The peers are segmented in virtual groups according to their bandwidth and the epoch size depends on an estimation of the minimum bandwidth between the peers that form each dissemination group. Simulation results from this network are expected to show alleviation of several issues raised in this paper such as the increased mean response time at the beginning of the dissemination. An alternative way which we also investigate is to use prior knowledge of a peer's content to push newly arrived packets.

REFERENCES

Adar E. and Huberman B.A. 2000, "Free Riding on Gnutella", *Technical report*, Xerox Palo Alto Research Center.

Chess D.M., Grosof B., Harrison C.G., Levine D., Parris C. and Tsudik G. 1995, "Itinerant Agents for Mobile Computing", *Journal of Personal Communications*, IEEE Computer Society, Vol. 2 (5). Pp34-49.

Joy B. 2000, "Shift from Protocols to Agents", *Internet Computing*, IEEE Computer Society, Vol. 4 (1). Pp63-64.

Karatza H.D. 2002, "Task Scheduling Performance in Distributed Systems with Time Varying Workload", *Neural, Parallel & Scientific Computations*, Dynamic Publishers, Atlanta, Vol. 10. Pp325-338.

Karatza H.D. and Hilzer R.C. 2001, "Epoch Load Sharing in a Network of Workstations", *In Proc. 34th Annual Simulation Symposium*, IEEE Computer Society Press, SCS, Seattle, Washington. Pp36-42.

Markatos E.P. 2002, "Tracing a large-scale Peer to Peer System: an hour in the life of Gnutella", *In Proc. CCGrid 2002*, Second IEEE/ACM International Symposium on Cluster Computing and the Grid. Pp65-74.

Parameswaran M., Susarla A. and Whinston A.B. 2001, "P2P Networking: An Information Sharing Alternative", *Computer Journal*, IEEE Computer Society, Vol. 34. Pp31-38.

Ripeanu M., Foster I. and Iamnitchi A. 2002, "Mapping the Gnutella Network: Properties of large scale peer-to-peer systems and implications for system design", *Internet Computing Journal*, IEEE Computer Society. Pp50-57

Rizzo L. 1997, "On the feasibility of software FEC", *Technical report*, Univ. di Pisa, Italy.

Schooler E. and Gemmell J. 1997, "Using Multicast FEC to solve the Midnight Madness Problem", *Technical Report*, Microsoft research.

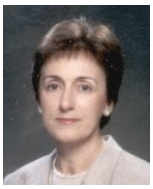
Shirky C. 2001, *Peer-to-Peer: Harnessing the Benefits of a Disruptive Technology / Listening to Napster*, ed. I.A. Oram, O'Reilly & Associates.

Spalink T., Hartman J.H. and Gibson G. 1999, "The Effects of a Mobile Agent on File Service", In *Proc. First International Symposium on Agent Systems and Applications*, Third International Symposium on Mobile Agents (ASA/MA '99), Palm Springs, California, IEEE Computer Society. Pp42-49.

Zerfiridis K.G. and Karatza H.D. 2002, "Mobile Agents as a Middleware for Data Dissemination", *Neural, Parallel & Scientific Computations*, Dynamic Publishers, Atlanta, Vol. 10. Pp313-323.

Zerfiridis K.G. and Karatza H.D. 2003, "Large Scale Dissemination using a Peer-to-Peer Network". To appear in the *Proceedings of the 3rd International Workshop on Global and Peer-to-Peer Computing on Large Scale Distributed Systems*, IEEE/ACM International Symposium on Cluster Computing and the Grid 2003, Tokyo.

KONSTANTINOS G. ZERFIRIDIS received his Diploma degree in Mathematics in June 1998 at the Aristotle University of Thessaloniki. In 1999 he received his M.Sc. degree in computer science from the University of Edinburgh. He is currently a researcher and working towards a Ph.D. at the Aristotle University of Thessaloniki. His research interests are mobile computing, mobile agents, distributed and Peer-to-Peer systems.



HELEN D. KARATZA is an Associate Professor in the Department of Informatics at the Aristotle University of Thessaloniki, Greece. Her research interests mainly include Performance Evaluation of Parallel and Distributed Systems, Multiprocessor Scheduling, Mobile Agents, Mobile Computing, and Simulation. Dr. Karatza is a member of the Editorial Board of the International Journal of Simulation: Systems, Science & Technology (the UK Simulation Society), Associate Editor of the Journal Simulation: Transactions of the Society for Modeling and Simulation International and area Editor for computer systems of the Journal of Systems and Software (Elsevier).