1995

Building High Performance Communication Services for Digital Libraries

Shunge Li

Jin Huai

Bharat Bhargava
Purdue University, bb@cs.purdue.edu

Report Number:
95-036

BUILDING HIGH PERFORMANCE COMMUNICATIONS SERVICES FOR DIGITAL LIBRARIES

Shunge Li
Jin Huai
Bharat Bhargava

Department of Computer Sciences
Purdue University
West Lafayette, IN 47907

CSD-TR-95-036
May 1995

Building High Performance Communication Services for Digital Libraries

Shunge Li, Jin Huai, Bharat Bhargava
Department of Computer Sciences
Purdue University
West Lafayette, IN 47907, U.S.A
{lis,huai,bb}@cs.purdue.edu

Abstract

The criteria for evaluating the performance of communication services for digital libraries and the requirements for building such services based on these criteria are summarized in this paper. Several major issues, such as transport protocols and media-specific protocols, adaptability and flow control, buffer management, and client-server interactions, are discussed. About 10% performance improvement has been achieved by employing some of the techniques proposed in this paper.

1 Introduction

A digital library, featuring supply of a huge amount of on-line real-time information services, involves such fundamental issues as representation, retrieval, storage, transmission, and presentation of information of various types, including video, audio, images, text, and structured data. These issues become even more critical when the volume and complexity of data scales up. Of these issues, communication services play an important role in making a digital library a reality in a physically distributed environment.

The communication services are critical in digital libraries since the database is inherently distributed. First, users are distributed, and data should be shared by as many users as possible. Second, data is distributed since the size of data becomes so large that it is not necessary and sometimes difficult to store all the data at one site. Third, information processing is distributed, and particularly, the user's queries can be so complicated that the process of information retrieval requires multiple rounds of interactions among users and various components of information system. Likewise, locating a particular information and maintaining consistency among information servers are also communication intensive.

However, what really makes the communications in digital libraries different from those in traditional distributed systems lies in the observation that digital libraries involve multiple types of data, each of which has its own unique media-specific characteristics and communication requirements that need special treatment and proper integration.

In the next section, we briefly summarize the criteria for evaluating the performance of communication services for digital libraries and the requirements for building such services based on these criteria. In the main part of this paper, we discuss several major issues involved in building high performance communication services for digital libraries, present techniques for achieving this goal, and report results of experiments that employed some of these techniques. Finally, we conclude the paper and summarize future work.

2 Criteria and Requirements

The criteria for evaluating the performance of communication services for digital libraries are summarized as follows:

- Reliability: For continuous media data such as audio and video, a certain amount of data loss in communications can be tolerated in applications, whereas for other data type such as text, reliable data transmission is required.
- Throughput and Response Time: The system should respond as quickly as possible even for very large data transmission. Suppose a very large document is to be sent to a user across the continent. The service intended to minimize the response time should deliver the very first page of the document quickly to the users. During the display of the first page of the document, the system can then transfer another parts of the document. Sometimes, the continuous media applications require that the data be transferred *continuously* in real-time. For example, in real-time video applications, the video data need be delivered at a rate of at least 20 frames per second (fps) to guarantee smooth presentation.
- Interoperability: Applications of digital libraries running in heterogeneous environments, including different machines, operating systems, and underlying networks, should be able to interoperate with each other. Moreover, documents represented in different formats such as in Latex, Postsripts, plain text, and hypertext, or video streams encoded in different coding schemes, such as MPEG, motion JPEG, and H.261, should be inter-transferable.
- Adaptability: The system should have a communication mechanism to adapt to environmental changes, such as network congestion, without significant degradation of performance. When the network bandwidths are overloaded, the system may, for example, slow down the transmission of video, reduce the size of each video frame, or dither the image to accommodate the external conditions. This applies to hypermedia documents, too.

- Scalability: The system should perform reasonably well when the number of users, the number of sites, and the number of machines per site, become very large. Group communications (one-to-many or many-to-many) are essential in some applications. When the number of participants in a group or the number of groups increases, the system performance will go down, but should not be decreased dramatically.

The requirements for building high performance communication services for digital libraries, based on the criteria above, are listed as follows:

- To achieve high throughput and low response time, high speed networks such as gigabits networks are essential. For some applications, networks with multicasting ability are required. Currently, multicasting is not widely available. Only a small community has access to the Internet MBONE service. However, in the future, the multicasting will definitely become an indispensable feature of the communication systems.
- To handle reliability and adaptability, transport protocols have to be properly designed. Since the reliability requirements vary with different media types, the transport protocols should be developed on a per-medium type basis. These media-specific transport protocols must be connection-oriented to provide flow control mechanism, based on which adaptability can be achieved, and optionally reliable, depending on whether or not that media type can tolerate a certain amount of data loss in communications. Reliable transport protocols can be built on top of any unreliable (best effort) transport protocols with additional mechanism dealing with error handling. Integration of the media-specific protocols is needed at application layer and should be transparent to users.
- Most distributed systems nowadays are based on client/server model. To deal with the issue of scalability, structure has to be introduced into the server space. In particular, hierarchical organization of servers is desirable because it permits information sharing among geographically nearby servers, facilitates their management, and effectively bounds WAN traffic.

3 Design Issues

The following subsections discuss several major issues involved in the design of high performance communication services for digital libraries.

3.1 Transport Protocols and Media-Specific Protocols

There are two types of transport protocols: connectionless and connection-oriented. Connectionless protocols treat each packet or datagram separately. They ignore the logical ordering among packets or datagrams. On the other hand, connection-oriented protocols

provide more services such as flow control and error recovery. Of course, they are more expensive.

In digital libraries, both documents and continuous media data require stream transmissions. Pure connectionless protocols are not suitable for digital libraries. However, existing connection-oriented protocols may not be suitable for digital libraries either, because such built-in services in these protocols as error-recovery and flow-control were not designed for digital library applications. For example, TCP's sliding window-based error recovery mechanism is not well suitable for video applications since TCP streams may be unnecessarily retransmitted even if some of them have been delivered reliably. One way to deal with this dilemma is to design new connection-oriented protocols to meet specific application requirements. A connection-oriented protocol can be developed by augmenting an existing connectionless protocol in such aspects as data sequencing, error handling, and flow control. This approach is effective because it takes advantage of the basic services provided by connectionless protocols. In fact, many protocols were designed in this way. They include: TCP, RTP [SCFJ94], cyclic-UDP [Smi94], and the protocol used in Raid distributed database system [BR89].

Reliability is another issue in choosing or designing a transport protocol for digital libraries. Two causes contribute to the packet loss and error: (1) unreliability of underlying transmission media, and (2) network congestion or resource exhaustion (e.g., buffers overflow) to cause intermediate gateways to drop packets. Several continuous media applications, such as video conferencing, can tolerate some sort of data loss or error, whereas electronic documents and structured data require reliable transmissions, which are usually provided either by reserving network resources ([Fer92], [Fer90]) or by adding more services (e.g., flow control and error recovery) on top of some best-effort protocols. Due to the expensiveness of the resource reservation schemes, most reliable transport protocols choose the other approach. TCP is such an example that has been shown in practice to be effective in transmitting documents and structured data. For continuous media applications, however, totally different strategy can be taken in providing reliability. For example, if the packet loss rate and bit errors rate are small and acceptable by the applications, the errors can be simply ignored and no retransmission is necessary. If the error or loss rate is not negligible, retransmissions are needed. However, unlike TCP where everything lost is required to be retransmitted, here only significant data (e.g., the I-frames in MPEG schemes) is selected for retransmission.

There are other requirements for continuous media applications in digital library [DTH92], [SHGC92], [DDK+90]. For example, the real-time Quality of Services (QOSs) guarantee is such a requirement. Best-effort protocols by nature are not good at meeting real-time requirements, such as time constraints, even if time is taken into consideration in protocol design. A safer but more expensive way is the resource reservation mechanism, which can *guarantee* services. However, in not so-critical real-time applications, best-effort protocols can still achieve relatively high performance.

It is now clear that no single transport protocol is optimal for applications of all media.

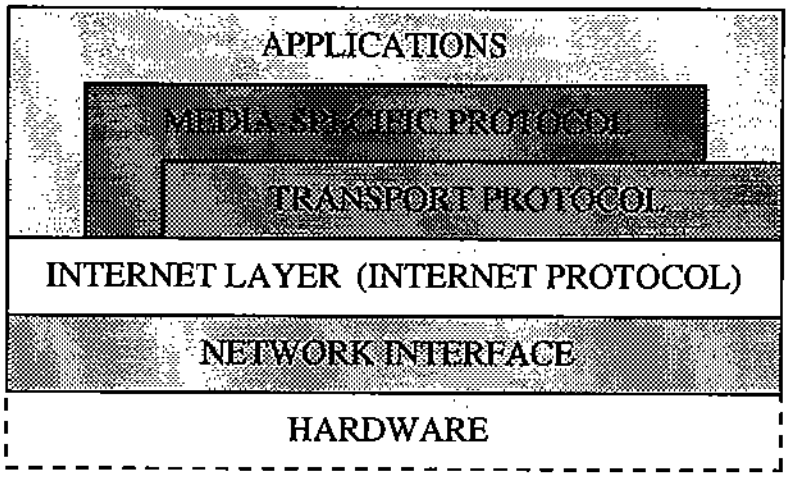

Figure 1: Protocol Layering

With media-specific protocols, it is possible to optimize the performance for every particular media applications. The optimal performance of the overall system can thus be achieved by proper integration of all the media-specific protocols. Figure 1 illustrates the idea of the protocol layering principle.

Flow control is also an important topic in designing media-specific protocols. We will discuss it in a separate subsection.

3.2 Clients, Servers, and their Interactions

In client/server paradigm, usually users (or clients) do not know what information is stored in digital libraries and where they are stored. Clients rely entirely on servers to find out the information they need by querying servers. It is not uncommon in digital libraries that clients are unable to specify their queries very clearly until they find something interesting and useful. Sometimes, clients have only partial knowledge of what they want. So at first, they give queries at very raw level. As soon as they get some feedbacks from servers, they can refine their queries further. Likewise, servers cannot fully answer clients' queries until they obtain additional information from clients. This highly interactive process may last multiple rounds and thus is communication intensive [BAG+94].

Since each server has limited resources and can store only a portion of the digital libraries, it is necessary for each server to maintain a catalog that contains information about what it has and what other servers have. If a server finds that the information requested by clients is not stored locally, it must contact other servers for help.

The way in which a server contacts and exchanges information with other clients and servers depends largely on the structures of the distribution of servers and clients, namely, hierarchical or interconnected. It also determines the performance of overall networked digital libraries.

The advantage of the hierarchical server structure is that it helps in clustering geographically nearby servers. Usually, these servers tend to interact more with each other than with other servers outside this cluster because they are closer to each other than to any other servers.

For example, suppose the default server for client X is A. Suppose further that X queries A for some information. A first looks up its local catalog, and if there is an entry corresponding to that information, A can simply answer X. Otherwise, A has to find the information for X. If A's catalog maintains the information about which servers contain what, A should be able to tell X if any other servers contain the requested information and if so which servers. But with this scheme, the catalog maintained by each server must contain entries not only for its locally stored information but also for all the information stored in *every* server. Obviously, this scheme does not scale well because as the number of servers increases, the size of catalog on each server increases dramatically. Moreover, whenever an update occurs to any catalog, all other catalogs must be updated accordingly to maintain their inter-consistency. Note that there are a large number of redundant information (replicated information) among catalogs of servers within a cluster. With hierarchical organization of servers, these redundant information can be reduced or eliminated by creating a per-cluster catalog that contains common entries in catalogs of servers under this cluster and by letting the servers share this catalog.

Once a server finds the set of servers containing the requested information, it can either pick one server for the client and let the client contact it, or return all the servers and let the client choose one itself. It can also fetch the information from one of those servers and deliver it to the client, meanwhile the fetched information can be cached in the server for later resolution of the same or related queries. Of course, the choice of servers is not random, it should be based on such criteria as communication costs (which one is the closest server, or which one has the lowest current load, etc) and network bandwidths.

Let's continue the above example. Suppose finally A finds and tells X that server B and server C contain the information requested by X. There are two options for X to get this information. First, X can directly contact either B or C; second, A can act as a proxy of either B or C, meaning that A gets this information from either B or C, then delivers it to X. There are tradeoffs between these two approaches: the former seems more efficient than the latter since it needs only one connection between X and B (or C), whereas the latter requires two connections: one between X and A and the other between X and B (or C). However, the former requires explicit user involvement whereas in the latter the information exploration is transparent to the users. With caching mechanism, the latter may reduce its communication cost if some information tends to be "hot spot", which may not be uncommon in digital libraries.

The question now is how to maintain the catalog of each server, and when to cache the information, if necessary, and how. This is quite similar to the problem RIP ([Hed88]) tries to solve, where the consistency of routing tables of gateways must be maintained. Unlike RIP, which is good for a rapidly changing environment, the servers here do not have to

periodically exchange their catalog information among clusters. All they have to do is to guarantee that all updates be propagated properly. That is, whenever there is a change or update to a catalog belonging to one server, this server must inform other servers of this change (update) by propagating it through all the clusters in the hierarchy.

3.3 Flow Control and Adaptability

Users experience that patience is needed when they want to transfer a very large video file (or home page) via World-Wide-Web (WWW) and display it locally, since the HTTP was designed such that users cannot watch the video until the entire video file has been transferred even though part of video is available. For example, a two-hour full color video with resolution of 320 x 240, compressed in MPEG (with compression rate being 100:1), has size as large as 480MBytes. If a user wants to transfer this particular video file over a 10Mb/s ethernet (LAN), he has to wait for nearly 6.5 minutes before she can watch the first frame of the video! If the resolution of the video becomes 640 x 480, he has to wait for at least 25 minutes before she can start watching! The situation will become worse in a WAN environment. Also, it is unreal to consume large disk space for only one application.

From this example, we can see that HTTP and the like are not a good choice for building high performance, even real-time communication services for digital libraries. In fact, there are more reasons why we need a better scheme or protocols for digital libraries: (1) Reduce the response time of the client queries; (2) Reduce disk space requirement on client sides; (3) Dynamically adapt data transmissions to network changes (e.g., bandwidth available, loss rate, etc.); (4) Gain full control of data transmissions to allow better buffering on both sender and receiver sides; (5) Facilitate dynamic prioritization of some transmissions over others; and (6) Embed security mechanism, such as copyright protection and encryption, into data transmissions. Some of them are beyond the scope of this paper and thus are not discussed any further here.

There are two approaches to deal with flow control. In the first approach, senders control the transmissions based on their knowledge about current network situations. This requires that senders monitor network behaviors and collect both statistical and instantaneous information about network throughput, reliability measurement, and bandwidth available. Once senders have this information, they can decide whether it is necessary to adjust current transmissions, and if so, how much. In the second approach, transmission flow is controlled based on feedback messages from receivers, including the receivers' current buffer sizes, whether they are congested, how many packets have been received, and whether some packets need to be retransmitted. This feedback information is conveyed through acknowledgement mechanism, which at the same time confirms the previous data transmissions. For example, the window-based flow control mechanism used in TCP allows the receiver to advertise its window size (through acknowledgement) dynamically, but this scheme fails to avoid unnecessary retransmission.

Both approaches have their own advantages and disadvantages. On one hand, the

first approach is superior to the second one because it can trace the network changes and therefore uses the network more efficiently. Even though the second scheme fails to do so, it surpasses the first in that the second can keep track of receivers' situations and respond to users' actions from receiver sides. Nevertheless, it is possible to invent a new scheme that combines the features of the two approaches. We are now investigating such a scheme that allows flow-control based on both current network behaviors and receivers' situations.

4 Experiments

To attack some of the issues discussed above in practice, we modified a software originally developed at University of California at Berkeley to study the performance and implementation issues. The software is called MPEG_PLAY [PSR93], which displays a given video file on a local machine. If the video file is on a remote machine, it is still possible to transfer it over the network through UNIX *pipe* and *rsh* mechanisms and display it locally. However, this approach has many restrictions. First, it requires that users have accounts on both local machines and remote machines. Second, it requires users know the location of the video file beforehand. Third, the software itself provides little or no flow-control mechanism. Instead, it employs the flow-control mechanism provided by the underlying network transport protocols. Fourth, the buffer size associated with a pipe is small and fixed, causing inflexibility – it is unable to adapt to external environmental changes as well as to respond to actions from users (such as pause and fast forwarding).

To overcome these restrictions, we modified the software in the following aspects: (1) Separate (decouple) the data transmission from presentation; (2) Support client/server model; (3) Parameterize the buffer size to make buffering easier; and (4) Use request-on-demand technique to handle flow-control.

The idea behind the so-called request-on-demand technique is as follows: we always request the amount of data as needed. At first, we prefetch a certain amount of data that is sufficient for the first presentation to last enough time which is at least equal to the latency of fetching the second piece of data. During the period of the current presentation, we start requesting and fetching another piece of data that is large enough for the next period of presentation. There are several benefits in this scheme. First, we only need to maintain a reasonably large buffer space, but not as large as the entire data file. Second, it is easier to control the frame-rate of presentation by controling the amount of data requested. Third, by monitoring run-time activities of underlying networks through surveillance mechanism [Zha94], it is also possible to adapt to any changes of the networks by dynamically adjusting the buffer size at client side.

In order to get smooth presentation of a video file, the time taken for requesting and fetching data must be equal to that spent in decoding the previous piece of video and in displaying them. If, for some reason, the time spent on data transmission increases, the process for decoding and displaying video data has to wait for new piece of video data to come before it can go on. The effect is that the presenation is not smooth. One way

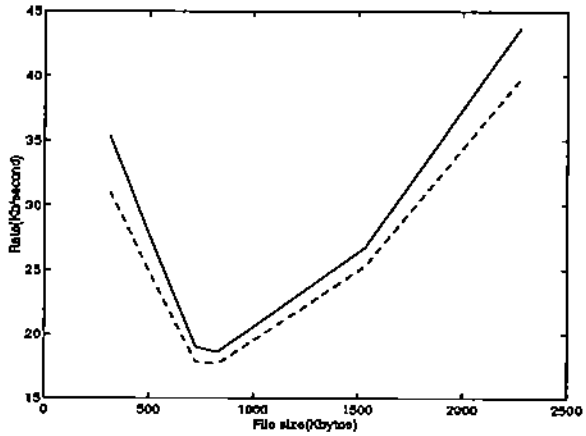

Figure 2: Presentation Rates of Original and Modified MPEG_Player

to overcome this problem is to enlarge the buffer size and to slow down the decoding and displaying process so that more data can be fetched and stored, and longer time will be taken to decode and display video piece. Although the frame-rate of presentation decreases, the smoothness is maintained. Similarly, if the network becomes faster, the buffer size should be reduced to slow down the network by blocking the network transmission for a certain amount of time after filling up the smaller buffer. Again, the smoothness is maintained. The basic principle here is that "smoothness of presentation is more important than speed of presentation".

By using request-on-demand technique, along with proper buffering mechanism, we get better performance. Figure 2 shows the performance improvement when the original and modified softwares are running under the same environment at the same time. The dashed lines represent the presentation rate (in Kbytes/second) using Berkeley MPEG_PLAY; the solid lines represent the same rate using modified MPEG_PLAY.

5 Conclusions

In this paper, we have addressed several issues in building high performance communication services for digital libraries and suggested the development of new transport protocols and media-specific protocols. From the experiments, we showed that using some of the techniques proposed here increases the frame-rate of presentation by about 10%, thus improves the system performance.

We are now working on the design of a media-specific protocol on top of UDP for real-time video applications. We are conducting research on how the video transmissions can be adapted to network dynamic behaviors so that the presentation of video streams will maintain acceptable quality. For this purpose, we are investigating efficient implementations of general surveillance mechanism.

6 Acknowledgement

The authors would like to thank the participants in the research seminars on Parallel and Distributed Systems for their questions and insights to this topic.

References

[BAG+94] Bharat Bhargava, Melliyal Annamalai, Shalab Goel, Shunge Li, Evaggelia Pitoura, Aidong Zhang, and Yongguang Zhang. Communication issues in wide area networks for digital libraries. In *Workshop on Digital Libraries: Current Issues*, Newark, NJ, May 1994.

[BR89] Bharat Bhargava and John Riedl. The Raid distributed database system. *IEEE Transaction on Software Engineering*, 15(6), June 1989.

[DDK+90] Willibald A. Doeringer, Doug Dykeman, Matthias Kaiserswerth, Bernd Werner Meister, and Harry Rudin. A survey of light-weight transport protocols for high-speed network. *IEEE Transactions on Communications*, 38(11):2025–2039, 1990.

[DTH92] Sylvie Dupuy, Wassim Tawbi, and Eric Horlait. Protocols for high speed multimedia communications networks. *Computer Communications*, 15(6):349–358, 1992.

[Fer90] D. Ferrari. A scheme for real-time channel establishment in wide-area networks. *IEEE Journal on Selected Areas in Communications*, 8(4):368–379, 1990.

[Fer92] D. Ferrari. Real-time communication in an internetwork. *Journal of High-Speed Networks*, 1(1):79–103, 1992.

[Hed88] C. Hedrick. Routing information protocol. Technical Report RFC 1058, Network Working Group, 1988.

[PSR93] K. Patel, B. C. Smith, and L. A. Rowe. Performance of a Software MPEG Video Decoder. In *Proceedings of the ACM Multimedia 93, Anaheim, CA*, pages 75–82. ACM, 1993.

[SCFJ94] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP, A Transport Protocol for Real-Time Applications. Internet Draft ieft-avt-rtp-06, Nov., 1994.

[SHGC92] Doug Shepherd, David Hutchinson, Francisco Garcia, and Geoff Coulson. Protocol support for distributed multimedia applications. *Computer Communications*, 15(6):359–366, 1992.

[Smi94] Brian C. Smith. *Implementation Techniques for Continuous Media Systems and Applications.* PhD thesis, University of California at Berkeley, 1994.

[Zha94] Yongguang Zhang. *Communication Experiments for Distributed Transaction Processing – From LAN to WAN.* PhD thesis, Purdue University, 1994.