Proxy-Assisted Techniques for Delivering Continuous Multimedia Streams

Lixin Gao, Zhi-Li Zhang, and Don Towsley, *Fellow, IEEE*

***Abstract*—We present a proxy-assisted video delivery architecture that can simultaneously reduce the resources requirements at the central server and the service latency experienced by clients (i.e., end users). Under the proposed video delivery architecture, we develop and analyze two novel proxy-assisted video streaming techniques for on-demand delivery of video objects to a large number of clients. By taking advantage of the resources available at the proxy servers, these techniques not only significantly reduce the central server and network resource requirements, but are also capable of providing near-instantaneous service to a large number of clients. We optimize the performance of our video streaming architecture by carefully selecting video delivery techniques for videos of various popularity and intelligently allocating resources between proxy servers and the central server. Through empirical studies, we demonstrate the efficacy of the proposed proxy-assisted video streaming techniques.**

I. Introduction

THE past few years have seen a dramatic growth of multimedia applications which involve video streaming over the Internet. Server and network resources (in particular, server I/O bandwidth and network bandwidth) have proven to be a major limiting factor in the widespread usage of video streaming over the Internet. In order to support a large population of clients, techniques that efficiently utilize server and network resources are essential. In designing such techniques, another important factor that must be taken into consideration is the *service latency*, i.e., the time a client has to wait until the object he/she has requested is started to playback. The effectiveness of a video delivery technique must be evaluated in terms of both the server and network resources required for delivering a video object and the expected service latency experienced by clients. Clearly, the "popularity" or access pattern of video objects (i.e., how frequently a video object is accessed in a given time period) plays an important role in determining the effectiveness of a video delivery technique.

Manuscript received March 7, 2002; revised April 30, 2002; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor M. Ammar. The work of L. Gao was supported in part by the National Science Foundation under Grant ANI-9977555 and NSF CAREER Award Grant ANI-9875513. The work of Z.-L. Zhang was supported in part by the National Science Foundation under NSF CAREER Award Grant NCR-9734428 and NSF Grant ANI-9903228. The work of D. Towsley was supported in part by the National Science Foundation under Grant ANI-9805185.

L. Gao is with the Department of Electrical and Computer Engineering, University of Massachusetts, Amherst, MA 01060 USA (e-mail: lgao@ecs.umass.edu).

Z.-L. Zhang is with the Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN 55455 USA (e-mail: zhzhang@cs.umn.edu).

D. Towsley is with the Department of Computer Science, University of Massachusetts, Amherst, MA 01030 USA (e-mail: towsley@cs.umass.edu).

Digital Object Identifier 10.1109/TNET.2003.820423

In this paper, we propose a proxy-assisted video streaming architecture that takes advantage of the resources (processing and disk storage) available at proxy servers to significantly reduce the server and (backbone wide-area) network resource requirements, while at the same time providing near-instantaneous service to clients. The central server multicasts video objects periodically, using, for example [13], source-specific multicast. Proxy servers are strategically placed between, say, local access networks and the backbone wide-area network. A proxy server stores a fixed number of initial frames or a "prefix" of the multimedia stream [12] so as to serve the future requests: when a new request arrives, the client joins an on-going multicast group to retrieve the multicast stream from the central server and retrieves the missing initial frames from the proxy server. The missing portion of the prefix is delivered by the proxy using a unicast channel and played back immediately by the client. Hence, the proxy server reduces the service latency experienced by the user by unicasting a prefix of the multimedia stream.

This proxy-assisted video delivery environment has several advantages over the traditional stand-alone video server environment. First, since it requires only network resources between the proxy and the client, latency reduction is achieved without increasing the demand on backbone network resources. Second, unlike the proxy caching schemes proposed for conventional Web objects such as text and image objects, the proxy needs to store only prefixes of the multimedia streams. Thus it is feasible even with the large data volume typically associated with multimedia objects. Third, since the proxy server delivers only the prefixes and is only responsible for a limited number of clients, the I/O bandwidth requirement imposed on the proxy server is not significant.

Under the proposed proxy-assisted video delivery architecture, we develop a novel video delivery technique called *proxy-assisted catching*, which can efficiently utilize server and proxy resources while providing near instantaneous service to clients. This technique is particularly suitable for "hot" (i.e., frequently access) video objects. The effectiveness of the technique is achieved through the intelligent integration of the "server-push" and "client-pull" video delivery paradigms. Using this technique, the server periodically "broadcasts" a video object via a number of *dedicated multicast channels*. A client who wishes to watch the video immediately joins an appropriate multicast channel without waiting for the beginning of the next broadcast period. At the same time, the client sends a request to a proxy server to retrieve the missing prefix of the video object. Using a smart broadcast scheme such as the *Greedy Disk-conserving Broadcast* (GDB) scheme [7], we present an analytical framework to determine design

parameters such as the size of the prefix stored in the proxy server. Furthermore, we show that the total resource required by the central server and proxy servers combined is close to the minimum achievable by any broadcasting scheme that supplies near-instantaneous service.

In order to account for the diverse access patterns for a collection of video objects in a video server, we design an efficient video delivery scheme called *proxy-assisted selective catching* which combines proxy-assisted catching with another video delivery technique—*controlled multicast* [8]. Controlled multicast is a "client-pull" technique which is most effective in delivering "cold" video objects. Based on video access patterns, we introduce a simple policy for classifying "hot" and "cold" video objects and apply catching and controlled multicast accordingly to deliver the video objects to clients. Through empirical studies, we demonstrate that, in terms of both network resource requirements and service latency, *proxy-assisted selective catching* outperforms either proxy-assisted catching or controlled multicast applied alone.

The remainder of this paper is organized as follows. The related work is briefly surveyed in Section I-A. Section II presents the proxy-assisted video delivery architecture. In Section III, we describe a specific proxy-assisted video delivery technique called proxy-assisted catching. Section IV introduces proxy-assisted selective catching and evaluates the scheme via empirical studies. The paper is concluded in Section V.

A. Related Work

In recent years, a variety of multicast techniques for video delivery have been proposed (see, e.g., [1], [3], [4], [6], [7], and [18]). These techniques can be broadly classified into either "client-pull" or "server-push." The simplest "client-pull" technique is to deliver a separate video stream upon each client request. This technique, while providing minimal service latency to a client, is obviously not efficient in terms of server and network resource utilization. Clever "client-pull" techniques such as *batching* [1], [6] and *patching* [5], [8], [15], [16] have been proposed that take advantage of the underlying network multicasting capabilities to reduce server and network resource requirements. In the case of batching, this reduction in server and network resource requirements is achieved through increased service latency, as it delays earlier requests for a video object until a certain number of requests for the same object arrive before the video object is scheduled to be delivered. Hence, batching is less effective for "cold" video objects. On the other hand, "patching," which allows multiple clients to share a multicast channel whenever possible, is most effective in reducing the server and network resource requirements for "cold" video objects without introducing service latency. A similar technique—the split and merge (SAM) protocol—is proposed in [14] for *interactive* VOD systems, where a unicast stream is scheduled on demand upon a client's request.

"Server-push" techniques [3], [4], [7], [17]–[19] are typically designed for "hot" video objects. They employ a fixed number of multicast channels to periodically broadcast video objects to a group of subscribers. The difference between various "server-push" techniques lies in the broadcast schemes used. These broadcast schemes determine the server and network resources required for broadcasting a video object. "Server-push" techniques have the advantage that they utilize server and network resources more efficiently. But this efficiency is achieved through increased service latency, as a client can only start receiving a video object at the beginning of next broadcast period.

The problem of delivering continuous media streams using proxy servers has been studied in a number of contexts. In [11], Wang *et al.* develop video staging techniques to store a pre-determined amount of video streams in strategically placed proxy servers to reduce the backbone network bandwidth requirement for delivering variable-bit-rate (VBR) video streams across a wide-area network. In [12], a prefix caching scheme is proposed to reduce the latency while delivering smoothed VBR continuous streams between the proxy and clients. Proxy-assisted video delivery is also proposed in the context of the dynamic skyscraper delivery scheme in [9].

II. Proxy-Assisted Video Delivery Architecture

In this section we propose a proxy-assisted video delivery architecture that employs a central-server-based periodic broadcast scheme to efficiently utilize central server and network resources, while in the same time exploiting proxy servers to significantly reduce service latency experienced by clients. Under the proposed proxy-assisted video delivery architecture, we develop two novel video streaming techniques—*proxy-assisted catching* and *proxy-assisted selective catching*. The proxy-assisted catching technique eliminates the shortcoming associated with periodic-broadcast-based "server push" techniques, namely, the increased service latency, by taking advantage of the resources available at the proxy servers. The proxy-assisted selective catching technique further improves the overall performance by combining proxy-assisted catching and controlled multicast to account for diverse user access patterns. In addition, unlike [9], our video streaming techniques can handle variable size video objects, and is based on formal analysis of multicast scheduling policies. From this analysis, the design parameters can be derived in a straightforward manner. As a result, our solution can be optimized accordingly. In the following we describe the proposed proxy-assisted video delivery architecture and introduce the necessary notation and terminology. The proxy-assisted catching and proxy-assisted selective catching techniques are presented and studied in Sections III and IV, respectively.

Fig. 1 shows a simple example of a proxy-assisted video delivery system. A central video server delivers video streams from a video object repository to a large number of clients across an inter-network (e.g., the Internet). A number of proxy servers are strategically placed between the wide-area backbone network and the local access networks where clients reside. The central server organizes the central server and network resources required to deliver a video stream[1] into a *data channel*. The server uses a multicast channel to deliver a video stream *periodically* to a group of clients (this group of clients is referred to

[1]In this paper we use the term *video stream* to denote a continuous flow or "stream" of video data (belonging to a certain video object) delivered from the server to a client or a group of clients. As will be clear later, a single video object can be partitioned into segments and delivered using multiple video streams via several delivery channels.

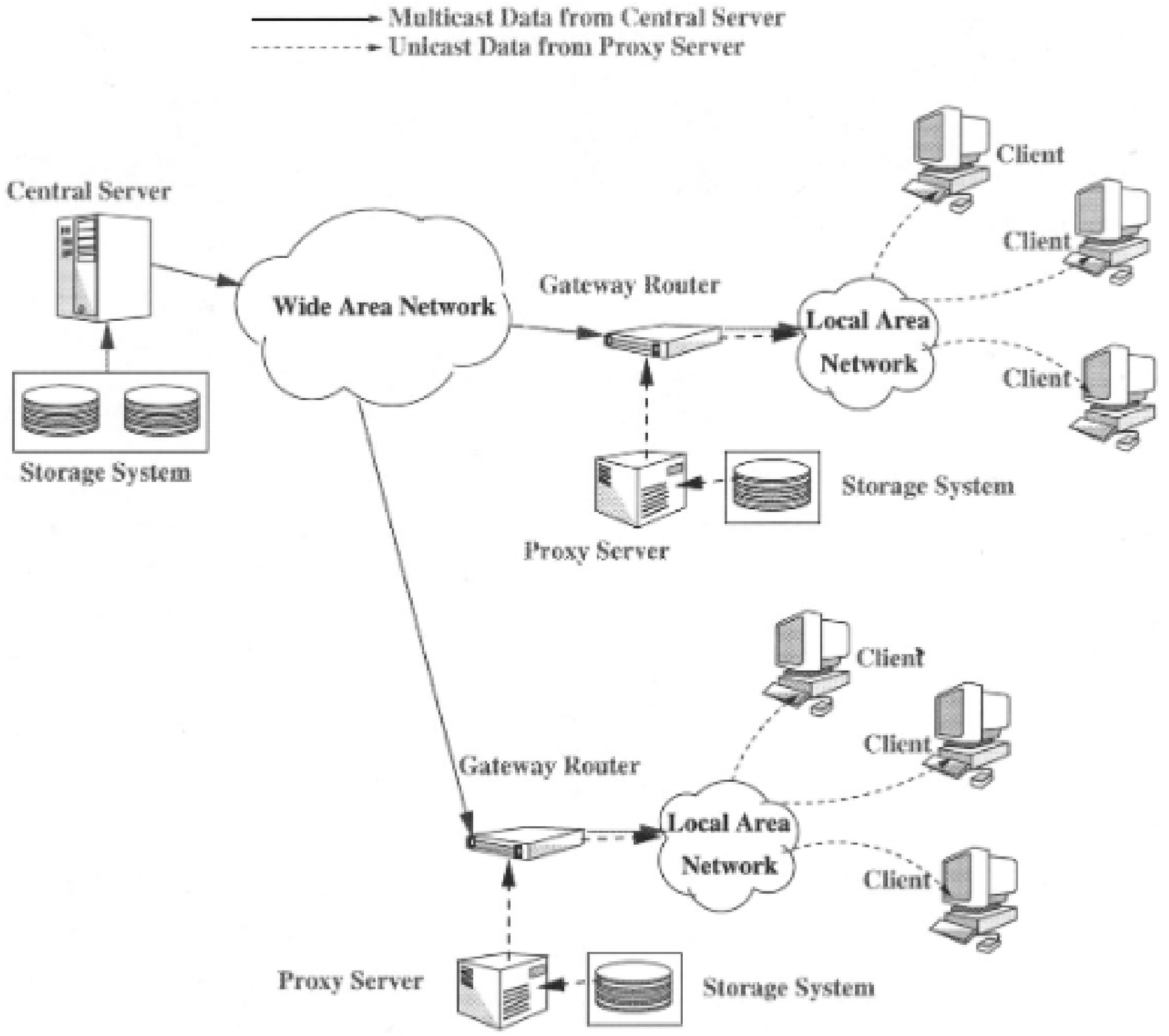

Fig. 1. An overview of proxy-assisted video delivery architecture.

as a *multicast group*). In addition to the logical channels used for delivering video streams (i.e., the data channels), we also assume that there are *control* channels to deliver signaling messages to a client or a group of clients and vice versa for control purposes (e.g., which video object is requested by a client, which data channels a client should tune in to, when to start video play-back, etc.). The video server has a scheduler which receives client requests for video objects via control channels, processes them and determines when and which video delivery channels to deliver requested video objects to clients. Since the bandwidth required by control channels is negligible comparing to that required by data channels, we concern ourselves only with the bandwidth required by the data channels throughout this paper.

The proxy servers *stage* (i.e., pre-store) a fixed number of initial frames of (some) video objects. When a client requests for a video object, it tunes to the central server to fetch its desired video data. However, to ensure near-instantaneous playback, the central server directs client to immediately fetch the initial frames of the video object that is staged at a proxy server that is "closest" to the client.[2] These initial frames are delivered from the proxy server by initiating a unicast channel. By staging a small amount of video data at the proxy, we see that proxy servers can effectively reduce the service latency experienced by the client without increasing the server network bandwidth requirement. In other words, the proxy-assisted video delivery architecture leverages the strategical location of the proxy servers and their storage and processing capacity by appropriately distributing the responsibility of video delivery between the central video server and the proxy servers.

In our work, we assume that each client contains a disk and a video display monitor. A client selects one or more network channels to receive a requested video object according to the instructions from the server. The received video data are either sent to the display monitor for immediate playback, or temporarily stored on the disk which is retrieved and later played back on the display monitor. The *client storage space* is the maximum disk space required throughout the client playback period. For ease of exposition, we assume that the client disk space is sufficiently large to store at least half a video.[3] The *client network bandwidth* is the maximum client network bandwidth required to receive video data from the network throughout the client playback period. We also assume that a client has the ca-

[2]The issue of how to locate the "closest" proxy server is outside the scope of this paper. Such issue has been studied by a number of researchers, e.g., in the context of replicated servers [20]. Other related issues such as proxy server placement, i.e., the number of proxy servers required and where to place them over the Internet are also important to the deployment of the proposed proxy-assisted video delivery architecture; likewise, they are beyond the scope of this paper. Some proposals can be found in, e.g., [21]–[23].

[3]This assumption is not essential, since our proposed schemes can be easily extended to a general case where clients have any given amount of disk storage space, as we will point out in Section III. In all of our empirical studies, the amount of client disk storage space used is actually only at most one third of a video object.