NMFS: Network Multimedia File System Protocol

Sameer Patel, Ghaleb Abdulla,
Marc Abrams, and Edward Fox

TR 92-54

Department of Computer Science
Virginia Polytechnic Institute and State University
Blacksburg, Virginia 24061

November 3, 1992

NMFS: Network Multimedia File System Protocol

Sameer Patel, Ghaleb Abdulla, Marc Abrams, Edward A. Fox

Computer Science Department, Virginia Polytechnic Institute and State University
Blacksburg, VA 24061-0106 U.S.A.

Abstract. We describe an on-going project to develop a Network Multimedia File System (NMFS) protocol. The protocol allows "transparent access of shared files across networks" as Sun's NFS protocol does, but attempts to meet a real-time delivery schedule. NMFS is designed to provide ubiquitous service over networks both designed and not designed to carry multimedia traffic.

1 Introduction

This paper describes a protocol that allows a multimedia application running on a client machine to use multimedia files stored on a server reached through *any* type of network. For simplicity, we view the world as containing two categories of networks: *data networks*, which include existing local area networks (e.g., IEEE 802.3, 802.5, and FDDI) and the TCP/IP-based Internet, and *multimedia networks*, which are networks specifically designed to carry constant and variable bit rate traffic (e.g., ATM networks, and Synernetics and Starlight modified 10baseT Ethernet hubs). Therefore, our goal is remote access to multimedia files through both data and multimedia networks, whereas most research on distributed multimedia applications considers only multimedia networks.

Why consider delivery of multimedia documents over data networks? For a number of years into the future, the Internet will probably interconnect a combination of older data networks and newer multimedia networks. This must be the case, unless all networking infrastructure in the Internet is simultaneously upgraded to multimedia networks. A desirable goal is that of *universal access* by any host or client on the Internet to any multimedia server, even if the connection between client and host contains data network links that do not provide constant or variable bit rate traffic. The quality of presentation on the client workstation may suffer due to data network links, but this is a tradeoff caused by the economic decision not to replace all data networks by multimedia networks.

The paradigm that we use to allow multimedia applications to access media files on network-attached servers is that of Sun's Network File System (NFS) protocol. NFS "provides transparent access to shared files across networks"[1]. NFS provides asynchronous access to remote file systems with no guarantees on the latency or throughput of file access. This is adequate for many types of text and binary data and program files, but inadequate for multimedia.

This paper proposes the Network Multimedia File System (NMFS) protocol, which similarly "provides transparent access to shared files across networks," but also meets real-time requirements of delivery schedules with a high probability. To achieve synchronization, the protocol should deliver all data segments belonging to an interval within a certain real time delay as specified by the application. NMFS is suitable for multimedia files, containing audio, video, and other formats, in addition to text and binary files. The NMFS protocol, though similar in function to NFS, is not a modification of NFS but a new protocol. Like NFS, NMFS is an application layer protocol. NMFS can be ported to any client or server providing UDP and RPC. NMFS is intended to operate over any type or size of underlying network or internet, in conjunction with other traffic. In particular, NMFS can function with the minimum assumption that the underlying network offers datagram delivery with no bounds on latency and throughput.

2 File System Model

In our server file system model, a particular multimedia material is stored in a set of one or more files referred to as a *file group*, each of which can contain one or more *tracks*. Each track is divided into *blocks*, where a block is defined to be a contiguous portion of a track with constant quality of service (QOS) parameters. Blocks are divided into Application Data Units (ADU), as defined by Clark and Tennenhouse [4]. An ADU is the unit of error recovery. A set of ADU's in one or more tracks are logically grouped into a *frame*. Frames cannot cross block boundaries. The client specifies read calls in terms of frames. In addition, frames are used in specifying QOS parameters and in synchronizing presentation of tracks at the client. The relation of tracks, blocks, ADU's, and frames is illustrated in Fig. 1. *QOS* parameters specify the divergence vector (DV), inter-glitch spacing (IGS) and inter-frame pause (IFP), as proposed by Ravindran and Bansal [5].

Synchronization between tracks is specified by providing in an auxiliary file a *virtual time stamp* for each ADU in each track. Tracks are delivered to an application using a *virtual to real time stamp mapping or sequence of mappings*. The default mapping is 1:1, however an application can specify to the NMFS protocol that another mapping be used to play tracks at different or perhaps even variable rates relative to one another. Virtual time stamps are used as the fundamental synchronization mechanism because they can be used to synchronize different types of media.

For NMFS to function over data networks, with their inherent variances in latencies, NMFS tries to presend blocks that it anticipates are likely to be used by the application in the near future. Therefore, for each set of multimedia material, either the client or the server or both stores in an auxiliary file an *anticipated delivery schedule* (ADS). The ADS is an N by N matrix, where N is the total number of blocks in the multimedia material. Element i, j of the matrix contains several pieces of information, including an estimate of the probability

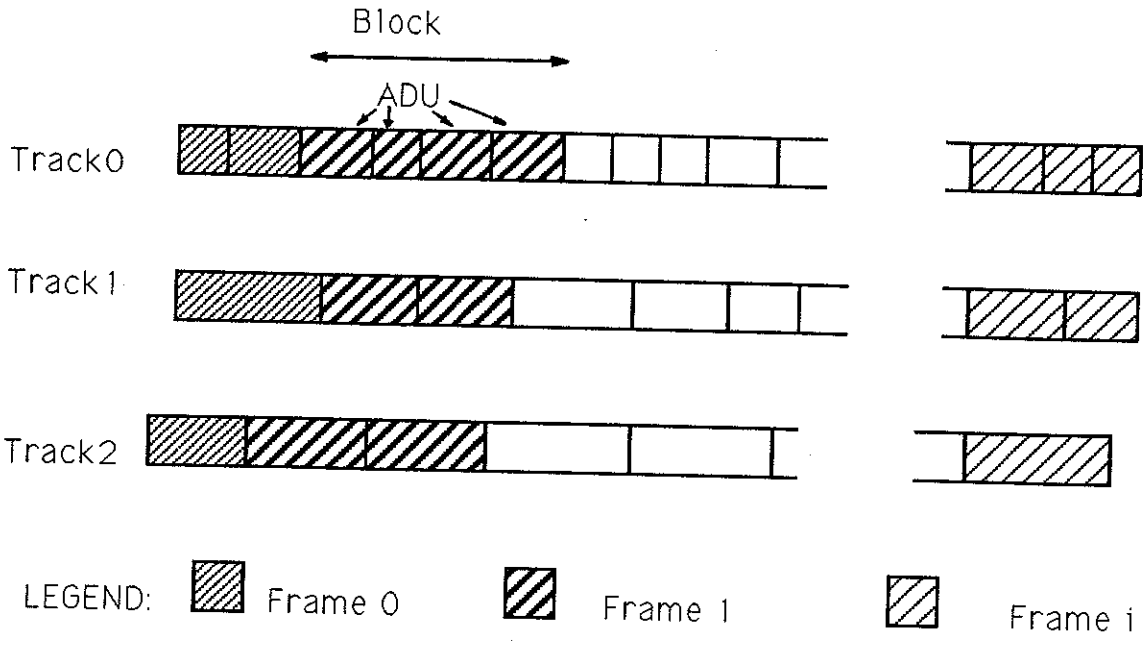

Fig. 1. The relation between tracks, blocks, ADU's and frames

that the application will next request block j, given that it last requested block i. The ADS maybe stored on the client, in which case the probabilities are based on a per-client basis. A default ADS is always stored on the server and is used by clients that do not have their own ADS and the probabilities represent the aggregate behavior of all network clients. The probabilities in the ADS are derived in one of two ways: from an authoring system [2] or by measurement. In the first case the authoring system predicts client behaviour. In the second case a log file is written that records the sequence of blocks accessed by a client. When the client terminates the ADS is updated based on the log file.

3 NMFS Protocol Overview

3.1 Restrictions in Initial Version of NMFS

The initial version of NMFS contains two restrictions. First, all files are read-only; a client cannot write to the server. Second the protocol does not provide synchronization between traffic types coming from different servers. Therefore, all the tracks that have to be synchronized must be on the same server.

3.2 Service

Like the NFS protocol, the service interface for NMFS is a set of remote procedure calls. In fact, the NMFS calls are only slightly modified versions of the

NFS calls. While an NFS file is byte addressable, an NMFS *file group* is frame addressible. In particular, in particular the NMFSPROC-READ call specifies an offset in unit of frames, and randomly accesses any frames in the *file group*.

Buffering: The protocol introduces the idea of buffering at the client's end a certain amount of data, that is likely to be referenced in the near future, in a *frame cache*. The cache size depends on the hardware resources of the client machine. Generally it is desired that the client has enough buffer space to buffer 1 or 2 seconds of presentation.

The frame cache is organized as a set of variable size frames. The frame cache is addressed with a frame number and an offset within a frame. The frame cache is used for two purposes. First, it is used to convert a network with variable latency into a network with constant latency. Second, it is used to hold frames which are anticipated to be used in the near future to reduce the network traffic. The frame cache is subject to a policy for replacing frames when the client reads a frame not in the cache; in this way the cache is conventional. However, the cache is unconventional in that the replacement policy will use the ADS to anticipate future frames that will be read, and notifies the server to presend these frames before the client reads them.

Flow Control and Error Recovery: Flow control is based on a model of the client buffer as a bucket with a spout at the bottom that delivers water to the client. This differs from a leaky bucket scheme [6] in that the buffer occupancy is used to set the server's transmission rate. The entry of water (e.g., ADU's) is sporadic. (For example, the delivery of ADU's may be slowed by a sudden burst in other network traffic, or interrupted entirely such as by a sequence of collisions in an Ethernet.) However, if the bucket never empties then the exit of water from the spout occurs at constant rate. In NMFS, the bucket corresponds to the set of frames within the cache that are currently being presented at the client. There are low and high water marks associated with the bucket. If the number of buffered ADU's reaches the low water mark, then the client side tells the server side to send ADU's at faster than real time. At the high water mark, the client tells the server to return to sending ADU's at real time (e.g., at the same rate at which the client presents them). If the client buffer falls below the low water mark, the NMFS client will modify the virtual-to-real time stamp mapping to slow down the presentation to avoid emptying the buffer.

For error recovery, NMFS allows an application to specify at the time of opening a file what user-defined algorithm should be used when reception of one or more ADU's occurs outside the QOS parameters. NMFS has default error recover for applications without error recovery algorithms. As proposed by Ravindran and Bansal [5], connections are broken and reestablished with tighter QOS parameters if continual errors and QOS violations occur.

3.3 Protocol Rules

Open Requests: A client begins by opening multimedia material on a server. In the open request, the NMFS client side requests a copy of the ADS from the

server if the client does not have an ADS for the currently running application.

The open request also contains an estimate of the maximum and average QOS parameters that will be used in subsequent reads. The server will decide whether or not to accept the open request based on its estimate of wither the networks between the client and server can satisfy the estimated QOS. To do this, the server uses statistics collected in the past if available about the mean and variance of network load. For example, for a server on a LAN which has frequent interactions with LAN clients, the server can add the aggregate bandwidths of existing NMFS open multimedia material, add the mean background traffic, and use the variance observed in the recent past to estimate whether the LAN can sustain additional open multimedia material.

After sending the ADS if requested, the server responds by associating each track in the material with one or more *NMFS connections*. Each NMFS connection is mapped to an appropriate entity of the underlying network. For example, this is a UDP port if the underlying network is UDP/IP, or a virtual circuit in an ATM network. The reason that a track is delivered over possibly more than one connection is that portions of a track may have different quality of service parameters. For example, in a video track, reference frames are sent over one connection with the QOS parameter IGS set to a large number to prohibit losses, while remaining frames are sent over another connection with a small IGS value to allow a much higher loss rate.

Read Calls: The client then will perform NMFS read calls. The client side of NMFS will first search the client cache for the frame to satisfy the read. If the frame is not cached, the client uses its copy of the ADS to select m client caches to free, and then sends the NMFS server m read requests to fill the caches. If $m = 1$, the client is requesting only the frame specified in the NMFS read call. If $m > 1$, then the NMFS client is also requesting the server to presend anticipated future frame requests.

Each read call specifies a start frame number and the number of subsequent frames to read in either the forward or reverse direction. If the frame is in a different block than the preceeding frame requested from the server, then the client and server must negotiate the QOS parameters used for delivery in the new blocks. Afterwards the server begins transmitting the ADU's constituting the frame. Note that ADU's travel over 1 or more NMFS connections, each with its own QOS parameters.

4 Project Status

We are currently (October 1992) designing the protocol and expect to implement version 1 by early 1993. A model for providing real-time support of continuous media is being developed. Our environment consists of servers at the Computing Center at Virginia Tech, clients that could be DVI machines, workstations or PC's located throughout the campus, and the interconnecting network which uses Ethernet and FDDI. In our final version of the protocol, we plan to relax constraints imposed by the initial version.

Acknowledgements: The authors wish to thank Scott Midkiff for his useful discussions. This work was sponsored in part by National Science Foundation grant NSF-CDA-9121999.

References

1. Norwicki,B.: NFS: Network File System Protocol Specification, RFC 1094.
2. Loeb, S.: Delivering Interactive Multimedia Documents Over Networks, IEEE Commun. Magazine **30** (1992) 52–59
3. Little T.D.C. and Ghafoor A.: Multimedia Synchronization Protocols for Broadband Integrated Services, IEEE J. on Sel. Areas in Commun. **9** (1991) 1368–1382
4. Clarck D.D. and Tennenhouse D.L.: Architectural Considerations for a New Generation of Protocols. Proc. SigComm '90 **20** (1990) 200–208
5. Ravindran K. and Bansal V.: Delay Compensation Protocols for Synchronization of Multimedia Data Streams, Technical Report, Dep. of Computing and Information Science, Kansas State University, Manhattan, KS
6. Bertsekas, D. and Gallager, R.: Data Networks, Prentice Hall (1992) 510–513