A Tcl/Tk Continuous Media Player†

Brian C. Smith, Lawrence A. Rowe, Stephen C. Yen

Computer Science Division
University of California
Berkeley, California 94720

1. Introduction

The Continuous Media (CM) Player is a tcl/tk application that supports playback of live digital audio and video on a Unix workstation. The current implementation uses a Parallax Xvideo framebuffer [1] with JPEG image decompression hardware [2]. The recorded video and audio (CM data) is stored on a file server and delivered to the application using a protocol based on UDP [3]. In keeping with the tcl/tk philosophy, the system takes a "toolkit" approach: it is designed to allow other media types, such as animation and scientific visualization, to be added into the system. This abstract describes the CM Player and the modifications we made to the Tcl/tk toolkit to implement it.

2. Data Model

The CM Player plays live digital and audio stored on a file server. Source material is stored as a contiguous sequence of frames in conventional Unix files, called *clip files*. Sections of clip files are called *clips*. For example, one clip might consist of five seconds of source material in a ten minute video file and another might be the whole ten minute video. Clips from multiple file servers can be assembled into *streams* of CM data by specifying the playback time of each clip along a time axis, called the *logical time system*. This collection of streams is called a *script*. Figure 1 shows the relationships of clips, streams, scripts, and the logical time system.

Notice that source material can be shared among multiple clips without copying the CM data. In systems where the assembly and synchronization information for streams is stored by interleaving streams (e.g., MPEG [5] or Microsoft video for windows [7]), the CM data must be copied to create new streams. Since 1 second of compressed video can take between 200 and 500 KBytes, this process is expensive both in storage and in time. Assembling a 5 minute video sequence will take between 50 and 200 MBytes and 30 to 240 seconds.

Another advantage of this representation is that source material may be distributed over multiple file servers. Since video occupies relatively large amounts of disk space (e.g., 1 hour of NTSC quality video compressed at 25:1 uses about 1 GB), storing source material on distributed file servers has distinct advantages. For example, source material need only be stored once, on one file server, even though it is used in many different scripts.

3. Process Architecture

The CM Player consists of four communicating processes: the *CM Source*, the *CM Server*, the *Application*, and the X Server. Figure 2 shows the relationship between the processes. The CM Source, CM Server and Application all use Tcl-DP [4] for communication.

The application creates the user interface and establishes a connection with the CM server, identifying the X window id of the video window. The video window is the window in which video will is played. The application also reads the script data from a user selected file and sends a request to the CM Server to establish connections to the appropriate CM Source processes.

The CM Source processes read CM data from a local disk and sends the data over a UDP connection to the CM Server. The CM Server receives the data, computes the Unix system clock time for playback and queues the data for playback at that time using at-events, a type of time event described in the next section. The at-event causes the data to be sent to the X server at the appropriate time.

†This material is based in part upon work supported by the National Science Foundation under Infrastructure Grant No. CDA-8722788. Additional support was provided by Fujitsu and Hewlett-Packard.

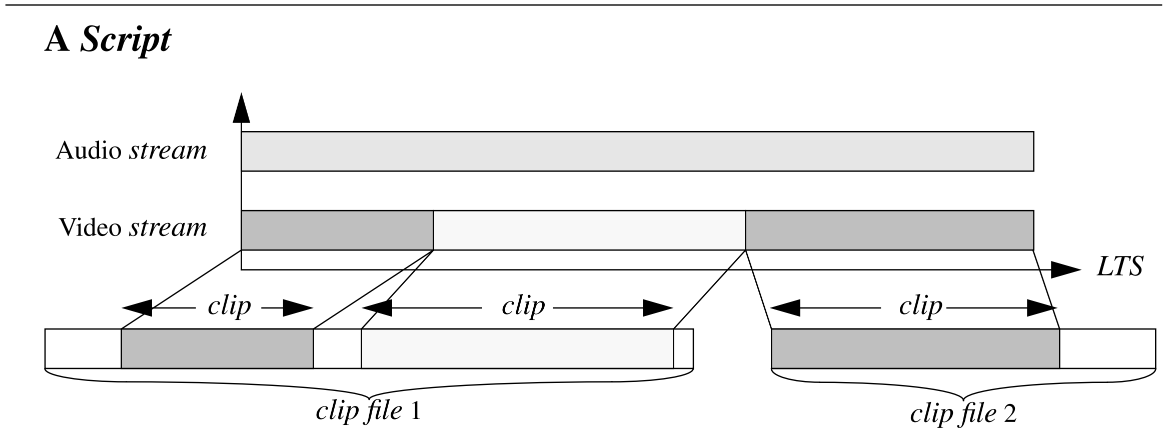

Figure 1: Scripts ,streams, clipfiles, clips and the LTS

4. Tk Modifications

Tcl/tk supports events from 4 sources. Events can be X-events, file-events, timer-events or idle-events. File-events call a function when a file becomes readable, writable, or an exception occurs on the file. Timer-events arrange for a function to be executed after a specified time has elapsed. Idle-events arrange for a function to be called when there is no more useful work to do (i.e., just before the process is put to sleep on a select call). Events are processed in the following order: X-events are processed first, then file-events, followed by timer-events and idle-events.

We modified the event loop to add a new event type, *at-events*. At-events arrange for a function to be called when the Unix system clock reaches a specified value. At-events are distinguished from timer-events in three ways. First, at-events are processed with the highest priority. That is, they are processes before X-events, file-events, timer-events, or idle-events. Since at-events are used to schedule the playback of CM data, close synchronization can be maintained only when they have high priority. Second, at-events are specified in absolute (i.e., Unix system clock) time, not relative time. This is significant only in the functional interface, since timer-events are stored internally in absolute time. Finally, at-events use a heap structure to store the events, whereas timer-events are stored in a linked list in the current Tcl/tk implementation. Heaps provide efficient insertations and deletions when large number of at-events are in the queue, a situation that commonly arises in our player.

5. Networking

The CM Source sends data to the CM Server using a protocol built on top of UDP. Tcl-DP provides a *connect* tcl command used to create the UDP socket, and a filehandler is created that calls a receiving function when data arrives on the UDP socket. This receiving function performs many functions. It strips protocol headers, assembles the (possibly fragmented) frame, schedules their playback using at-events, and requests retransmission of lost data. If it detects many lost packets, it sends feedback to the source using the Tcl-DP RDO facility (a type of non-blocking RPC) to request that the source reduce it's transmission rate. This adaptive feedback technique dramatically improves the perceived quality of video at the destination in the presence of changing network conditions.

The CM Source sends data to the CM Server a small amount of time (typically 0.5 seconds) before it is needed for playback. This delay reduces the effect of jitter in the network and allows the CM Server to request retransmission of lost packets. The CM Source determines which frames to send based on the value of the *Logical Time System (LTS)*.

The LTS specifies a linear mapping from the value of the Unix system clock to logical time. That is

$$LogicalTime = Speed \times SystemTime + Offset$$

Speed is a real number that determines the rate at which logical time advances relative to system time. A speed of 1.0 corresponds to normal playback, a speed of 2.0 corresponds fast forward, and a speed of -1.0 corresponds to reverse play. *Offset* is used to provide random access to various sections of the script. By appropriately setting offset, any part of the video may be accessed.

The speed and offset values are encapsulated in a Tcl-DP distributed object that is shared among the CM Source, CM Server, and Application processes.

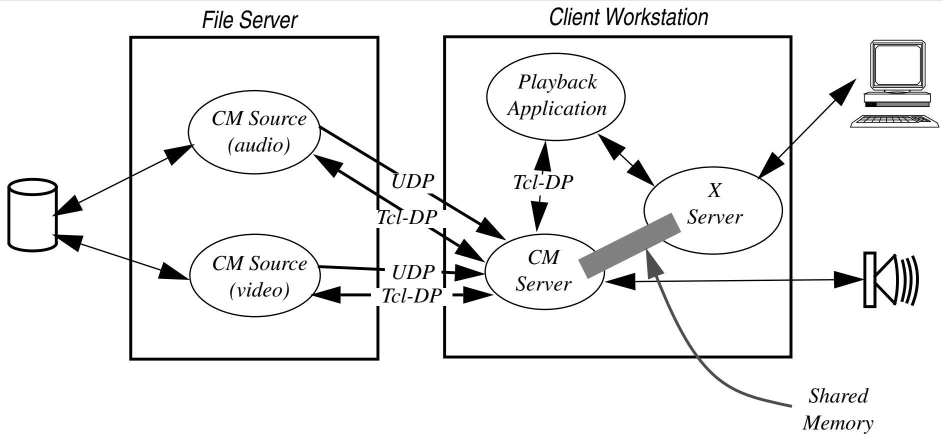

Figure 2: The CM Player Processes

The Unix system clocks on all machines are synchronized using the Network Time Protocol [6]. When the Application changes the speed or offset of the lts (e.g., by pressing the “play” button), the changes of passed on to the CM Server and CM Source processes. The CM Source has triggers that detect the changes in the lts and start the flow of CM data.

Notice that this method of transmission avoids explicit requests by the CM Server for CM data. Early experiments with request/response protocols indicated that the latency introduced by the protocols was intolerable. Moreover, request response protocols do not scale well to high latency networks. By using the lts to implicitly request data, we expect our solution to scale well to high latency networks.

6. Conclusion

In this abstract, we described the design of a CM player based on Tcl/tk. The system has several novel features, including the ability to store CM data on multiple file servers, the use of the logical time system and Tcl-DP to passively determine the CM data to send, and the use of feedback to adjust the playback rate in the presence of variable network load.

The performance of the system is quite good. On our departmental 10 MBit ethernet, we typically achieve playback rate of 24 frames per second on 320 by 240 full color video. This corresponds to a throughput of approximately 2.5 MBits/sec. When higher data rates are required (e.g., for 640 by 480 video), the adaptive feedback algorithm that slows the transmission at the source maintains high quality video.

References

[1] *X Video Users Guide*, Parallax Graphics, Santa Clara, CA, 1991

[2] Gregory K. Wallace, *The JPEG Still Picture Compression Standard*, CACM, Volume 34, No 4, pp 30-44, April 1991.

[3] Stephen G. Kochman, et. al, *Unix Networking*, Hayden Books, Carmel, IN,

[4] Brian C. Smith, et. al, *Tcl Distributed Programming*, Internet Software Distribution, University of California, Berkeley, CA, 1993

[5] Didier Le Gall, *MPEG: A Video Compression Standard for Multimedia Applications*, CACM, Volume 34, No 4, pp 46-58, April 1991.

[6] Mills, D., *Measured performance of the network time protocol in the internet system*, Network Working Group, RFC 1128 (October 1988).

[7] Tom Yager, *Inside Video for Windows*, BYTE Magazine, Special Issue, Spring 1993, pp 57-60