Multimedia Synchronization Techniques: Experiences Based on Different System Structures

Ralf Steinmetz[1] , *Thomas Meyer*[2]

Introduction

"Multimedia synchronization" is needed to ensure a temporal ordering of events in multimedia systems. There are two parts to this problem: the **definition** of temporal relationships among audio, video, and other data; and the **delivery** of the data in accordance with these relationships. It is important to distinguish between **live-** and **synthetic-synchronization**. **Live-synch.** deals with the presentation of information in the same order as it was originally collected. For **synthetic-synch.** various pieces of information must be properly ordered and synchronized in time. Based on the experiences in DiME and HeiTS, we document the implications of using a *hybrid* or a *unified* system structures on the required techniques for implementing synchronization.

Synchronization in Hybrid Distributed Multimedia Systems

The DiME (Distributed Multimedia Environment) project, carried out at the IBM European Networking Center in Heidelberg, was based on a *hybrid* system structure. Continuous media (i.e., audio and video) were routed over separate, dedicated channels using off-the-shelf hardware technology. Continuous and discrete media were integrated by connecting the CM equipment (e.g. VCRs) to a computer via an RS-232C interface. Most real-time processing is kept out of the main CPU and operating system.

Live-synch. between various CM streams is directly performed by the dedicated processing devices. Correlation between various DM streams, objects or information units in the live-synch. mode occurs very rarely in such an environment. A rough implementation can be achieved by time-stamping DM objects and delivering them on cue from timer events at the remote workstation. In hybrid structures, it is very difficult to achieve tight synchronization between DM and CM. DM and CM data are transmitted over different networks, through processing nodes having different end-to-end delay characteristics. End-to-end delay over CM paths is typically lower than that for DM. As it turns out, it is difficult and expensive to delay CM data delivered from devices like cameras or microphones. If DM is faster than CM, buffering and time stamping as described above could be used to slow down the data stream.
In synthetic-synch., where data is retrieved from external storage devices, one must contend with another type of delay, that of the control signals to the storage devices. Between the issuing of a "start" command by a workstation application until the commencement of physical delivery of a video sequence, we experienced a maximal delay of about 500 msec (if positioning must be done by, say a VCR, this could take considerably longer). The main reasons for this delay are:

- The system software is in principle not designed to cope with the real-time demands of physical interfaces such as the RS-232.
- The same device driver is often used for controlling many devices at the same time. A shared RS-232 C interface may introduce access conflicts between the various devices.
- Most of the external devices process queued work requests in an "as fast as possible" mode. The control interfaces do not include options to specify when information is to be presented.

Decoupling the seek time and the playback delay allows for a nearly deterministic behavior of the whole system. More interesting is the decomposition of the end-to-end delay d_{tot} into a fixed component d_{fix} and a variable part d_{var}: $d_{tot} = d_{fix} + d_{var}$. The variable part with its distribution comes from the above mentioned phenomena and/or the underlying API. Fortunately, the variance is not considerable and we can assume system- and device-specific values for d_{fix} to derive d_{tot}.

---

[1] IBM European Networking Center, Tiergartenstr. 8, 6900 Heidelberg, P.O.BOX 10 30 68, Germany
e-mail: STEINMET or MEY at DHDIBM1.BITNET

[2] University of Mannheim, Lehrstuhl für Praktische Informatik IV, Seminargeb. A5, 6800 Mannheim, Germany

Synchronization in Unified Digital Distributed Multimedia Systems

Achieving live-synch. between CM streams is generally more challenging in a "unified" system structure like in our follow-on project, the Heidelberg High-Speed Transport System (HeiTS), where all discrete as well as continuous media are routed through the workstation. Scheduling can either be done using dedicated hardware, such as the ActionMedia II (DVI), or exclusively in software. Either solution requires real-time scheduling techniques in a time-sharing environment. In both DM and CM processing environments scheduling and reservation can be performed by an operating system that provides a RTE (real-time environment). Processes running in such an environment are, in general scheduled according to real-time scheduling techniques. With this approach we can hide time constraints from the user.

In a distributed multimedia system where multiple, related streams originate in the same workstation, a very common and straight-forward approach is to interleave different streams into one composite stream. Such a stream consists of a sequence of logical data units (LDU), each time-stamped at the source and separated at the sink for presentation according to the time stamp. This technique is easy to implement and is used to guarantee lip synchronization in our current HeiTS prototype, which was demonstrated at the CeBIT'92 fair in Hannover, Germany.
We found that any audio glitch is perceived immediately and can not be tolerated, whereas a small variance in the video rate leading to, e.g., display of same picture twice, is difficult to perceive. In general, audio imposes more stringent requirements than video does. In the design of HeiTS and subsequent prototypes we take this into account as follows:

1. We define different quality of service (QOS) at connection set-up for audio, video and other CM streams. and choose the QOS parameters to ensure that there are no audio faults and either no or very few video glitches. We can thus assure that whenever a shortage of resources occurs, it will first affect the video connections.
2. We use resource management during call establishment to help prevent glitches.

The evolving HeiTS prototype has the ability to either (1) combine audio and video, or (2) use separate connections for different CM (or DM) streams. By imposing the same end-to-end delay on related CM streams (by choosing an absolute end-to-end delay and limiting the jitter of the LDUs to 0 msec), live synchronization can be guaranteed. In practice, it is neither possible nor necessary to guarantee service with such tight tolerances. Audio can be played ahead of video for about 120 msec, and video can be displayed ahead of audio for about 240 msec.
Another implementation of live-synch. allows these effects to be used to advantage and even guarantees synchronization between different sources: A logical time system (LTS) is introduced. Presentation of the data is performed based on a comparison of the LTS with the real-time clock of the sink workstation. Sinks and sources interacting in a logical group are tied to the same LTS. The RTE provides for the presentation of LDUs according to the LTS and the current real time. Scheduling techniques are capable of providing this in-time playback. For synthetic-synch., the application defines an interaction time and an event to happen. From the implementation point of view, application code can be linked into the RTE, or time-sharing code can be called from the RTE (upcall).
Although the goal is to have isochronous data streams at the sink devices, it is not necessary to have isochronous communication in all of the components in a communication path. For instance, it is sufficient to have upper bounds on delay for each individual component. With this assumption, buffering can be used to achieve isochronism, but this can lead to a waste of storage, especially for video. Restricting jitter ("jitter control") at intermediate gateways drastically reduces total buffer requirements. Ferrari's approach can also be applied to system components in the end-systems, leading to what we call "weak isochronous" behavior

Conclusion

Our work on synchronization is being performed as part of the HeiTS project. All the experience to date indicates that synchronization requirements should not be viewed as an isolated issue. The system structure, the hardware capabilities, the operating system capabilities, the communication subsystem and its protocols, the kind of media, the coding techniques, and even the envisaged types of applications, all influence the best synchronization techniques to use; a system-wide solution should be attempted.