To Use or Avoid Global Clocks?

Cormac J. Sreenan

AT&T Bell Laboratories
Murray Hill, NJ 07974
cjs@research.att.com

1 Introduction

The subject of multimedia synchronization in distributed systems has been a focus of attention for several years now. Broadly speaking, individual research efforts have tended to concentrate either on tools and abstractions for specifying presentation ordering, or techniques for performing synchronization at run time. A specification combines a set of related data elements with instructions on how they should be presented. Thus, one may wish to indicate sequencing, concurrency, responses to user interaction, etc. Most specification schemes involve some kind of timelines and/or scripts, often stored in conjunction with associated multimedia documents. The use of formal methods for dealing with potentially complex requirements has also been explored. Research into the design of systems to provide run time synchronization has been primarily for streams of continuous media, such as audio and video. The temporal characteristics of continuous media present unique difficulties, and hence the issue has attracted considerable interest. Several schemes can be found in the literature, a few of which have been used experimentally. The focus has been on the problems arising in geographically distributed multimedia systems based on packet networks and involving multipoint as well as point to point scenarios. The question here is how to synchronize presentation across these nodes in the face of differing and varying network delays. This must be done to satisfy constraints imposed by any specified ordering as well as stream QOS parameters such as end to end delay. A key issue in the design of these systems is how to deal with time, and more specifically, whether to use a global clock system, a question which has been divisive for traditional distributed system design. This position paper discusses the use of global clocks from the perspective of multimedia synchronization, seeking to clarify the issues involved.

2 Requirements

The need to be able to determine when a presentation action should be done is a common feature of synchronization algorithms. When executing any form of timeline or script, the order of presentation defines the relationship between the data elements, and in particular, will require support for simultaneous and sequential actions, in addition to timed delays, overlaps, etc. Ensuring that the elements are synchronized at transmission time is not sufficient if

different network delays can occur, a prospect of particular concern when multiple receiver nodes (and perhaps routes) are involved. Similarly, if using multiple sender nodes, achieving synchronized transmission is not straightforward. An example of this n-to-1 scenario is found in [3], where a workstation accesses media-dependent servers for audio, video, etc. In [6] a single server distributes streams to audiophone nodes and videophone nodes, representing the 1-to-n scenario.

In the case of a single stream there is typically a need for the receiver to operate a buffer for the purpose of smoothing inter-arrival variations, in effect, ensuring continuity by synchronizing the stream data with time. A crucial parameter in designing and operating this is its length, which dictates the amount of variation that can be tolerated before packets have to be deemed lost because they are too late. The time a packet spends in the buffer adds to the total end to end delay between its generation and presentation. Keeping this delay low and bounded is of special importance for applications which involve a degree of interactivity such as conferencing. To accurately introduce such a delay requires that a suitable duration be determined.

When dealing with multiple streams, there may be a requirement to maintain some relative synchronization, commonly expressed in terms of bounding inter-stream skew. The typical example is to maintain lip-sync in the 100-150 millisecond range. A simple, though rather limited, approach is to interleave the streams, thereby maintaining their temporal ordering. If the streams are sent separately, as is more common, the receiver must be able to detect unwanted relative delay differences and provide buffering so that these can be normalized if necessary. Extending skew bounds across multiple senders and/or receivers is also a requirement for many applications. Of course, once synchronization has been attained, issues of rate mismatches may also need to be considered. However for many sets of streams this may not be a problem, because their lifetime is such that typical clock rate differences (see next section) are not large enough[1] to cause noticeable skew.

Another form of synchronization is required for many cooperative applications such as shared whiteboards, etc. In these situations, the actions of different geographically separated users need to be coordinated so as to avoid problems caused by differences in delays during the issue and execution of their requests.

3 Global Clock Solutions

The existence of a time system shared between senders and receivers allows relatively straightforward solutions to the synchronization requirements just described. Using sender timestamps and receiver comparisons the presence of delay differences are easily detected, and can be compensated for by receiver buffering [2, 9]. But using conventional system clocks for this purpose is unsuitable. This is because each system clock is driven by a physical clock device, the frequency of which can vary, influenced mainly by temperature changes. Thus, system clock values tend to drift slowly over time. The task of initializing a system clock is often left to a human operator, and is also prone to inaccuracies. It is fairly routine to find that system clocks may differ by many seconds or minutes.

[1] The frequency offset of a clock with regard to real time can be as bad as 1 part in 10^3, although 1 part in 10^6 is more common.

Global clock systems consist of a network of time servers which operate to minimize differences between individual system clocks. They do this by making comparisons over long time periods and automatically making clock adjustments if deemed necessary. Many problems with these systems, primarily regarding accuracy, but also fault tolerance, cost and administration, have been used to argue against their use. But techniques have been steadily improving, to the point where the well known Network Time Protocol (NTP) boasts precision to within milliseconds when operating across the Internet, even in the face of failures and disruption [4]. The falling cost and wider acceptance of Global Positioning Systems (GPS), which provide a source of highly accurate time, may also help to increase the precision and deployment of global clock systems.

4 Alternative Solutions

If the ability to make direct comparisons between clocks is not available, other perhaps less accurate approaches can be used. In essence, these say that if the network delay that a packet encountered can be determined, then the receiver can tell when it originated at the sender. Several ways of doing this have been proposed. One can work out the average round trip delay by "pinging" the destination, yielding a one way estimate by dividing in two. This assumes that the delay is symmetrical and that the calculated value is likely to stay the same in the future when actual stream transmission occurs. Another approach assumes that network relays, such as packet routers, can record queueing delays as part of each packet's header [5]. This expects a lot in terms of relay support and is particularly unsuitable for high-speed cell-based technologies like ATM.

By making an assumption about the delay of the first packet to arrive on a given stream, one can observe later arrivals and improve the accuracy [1]. This can be optimized if a delay bounds guarantee can be obtained for the first packet, but conventional packet networks do not support this type of behavior. However networks based on ATM do hold out the promise of being able to offer guarantees on delay. This has prompted some groups to go a step further in suggesting the use of QOS to ensure that synchronization is maintained or performed implicitly by the network, rather than relying solely on receiver buffering [7]. The real or virtual costs in terms of network resources may have important repercussions for such proposals.

When dealing with streams being captured in real time, knowledge of the network delays allows any original difference in start times to be reproduced. For pre-recorded streams, such information is only useful if the actual start of transmission times can be synchronized. Otherwise, the receivers will end up simply reproducing an initial skew. For a single source this is not a serious problem, but synchronizing multiple sources is a different matter. In the absence of a shared time system, which could be used to pre-schedule the playback operations across each sender, an interesting alternative is to use a "prepare and act" arrangement as in [8]. The idea is that sources are asked to prepare in advance of the start time, allowing resources to be allocated, and with the aim of reducing the latency of the subsequent act command which initiates playback. The existence of a real time message passing or RPC protocol would be useful in ensuring the accuracy of this mechanism. Alternatively, implementing this type of function across the destination nodes is sufficient to allow synchronization by the judicious use of buffering in combination with knowledge of the playback constraints. For example, if it is

known to the receivers that a set of streams should commence simultaneously, they can wait until every stream is arriving before starting playback.

5 Conclusion

Most synchronization problems can be solved by either using a global clock system, or relying on assumptions or guarantees about network performance. An important issue when dealing with CM streams is that of exactly how much accuracy is required, unlike with many of the traditional applications of global clocks where approximate methods are generally to be avoided. There is a pressing need for more studies on how synchronization affects the quality of audio-visual presentation. In particular, using typical values quoted as acceptable as a lip-sync skew bound, it may be possible to simply rely on the low latency in LAN environments, coupled with some form of loose guarantees obtained using network QOS support in the wider area.

References

[1] F. Alvarez-Cuevas et al. *Voice Synchronization in Packet Switching Networks.* IEEE Network, pages 20–25, September 1993.

[2] J. Escobar et al. *Flow Synchronization Protocol.* In Proceedings of GlobeCom '92, pages 1381–1387, 1992.

[3] L. Lamont and N. D. Georganas. *Synchronization Architecture and Protocols for a Multimedia News Service Application.* In Proceedings of the 1st International Conference on Multimedia Computing and Systems (ICMCS '94), pages 3–8, May 1994.

[4] D. Mills. *Internet Time Synchronization: The Network Protocol.* IEEE Transactions on Communications, COM-39(10):1482–1493, October 1991.

[5] W. A. Montgomery. *Techniques for Packet Voice Synchronisation.* IEEE Journal on Selected Areas in Communications, 1(6):1022–1028, December 1983.

[6] P. V. Rangan et al. *Designing an On-Demand Multimedia Service.* IEEE Communications Magazine, pages 56–64, July 1992.

[7] K. Ravindran and V. Bansal. *Delay Compensation Protocols for Synchronization of Multimedia Data Streams.* IEEE Transactions on Knowledge and Data Engineering, 5(4):574–589, August 1993.

[8] C. Schmandt and M. A. McKenna. *An Audio and Telephone Server for Multi-Media Workstations.* In Proceedings of the 2nd IEEE Conference on Computer Workstations, Santa Clara, California, 1988.

[9] R. Yavatkar. *MCP: A Protocol for Coordination and Temporal Synchronization in Multimedia Collaborative Applications.* In Proceedings of IEEE International Conference on Distributed Computing Systems, pages 606–613, June 1992.