Isochronous Distributed Multimedia Synchronization

Zhonghua Yang
Nanyang Technological University, Singapore and Southern Yangtze University, China

Yanyan Yang
University of California, Davis, USA

Yaolin Gu
Southern Yangtze University, China

Robert Gay
Nanyang Technological University, Singapore

INTRODUCTION

A multimedia system is characterized by the integrated computer-controlled generation, manipulation, presentation, storage, and communication of independent discrete and continuous media data. The presentation of any data and the synchronization between various kinds of media data are the key issues for this integration (Georganas, Steinmetz, & Nakagawa, 1996). Clearly, multimedia systems have to precisely coordinate the relationships among all media that include temporal and spatial relationships. Temporal relationships are the presentation schedule of media, and spatial relationships are the location arrangements of media. Multimedia synchronization is a process of maintaining these relationships by employing appropriate synchronization mechanisms and algorithms. Multimedia synchronization is traditionally challenging, especially in distributed environments.

Three types of multimedia synchronization can be distinguished: intrastream synchronization, interstream synchronization, and intermedia synchronization (Crowcroft, Handley, & Wakeman, 1999). The approaches used for interstream synchronization can also be used for intermedia synchronization.

The word *synchronization* refers to time. The easiest way of synchronizing between streams at different sites is to use a single time reference. There are several ways to provide this time reference.

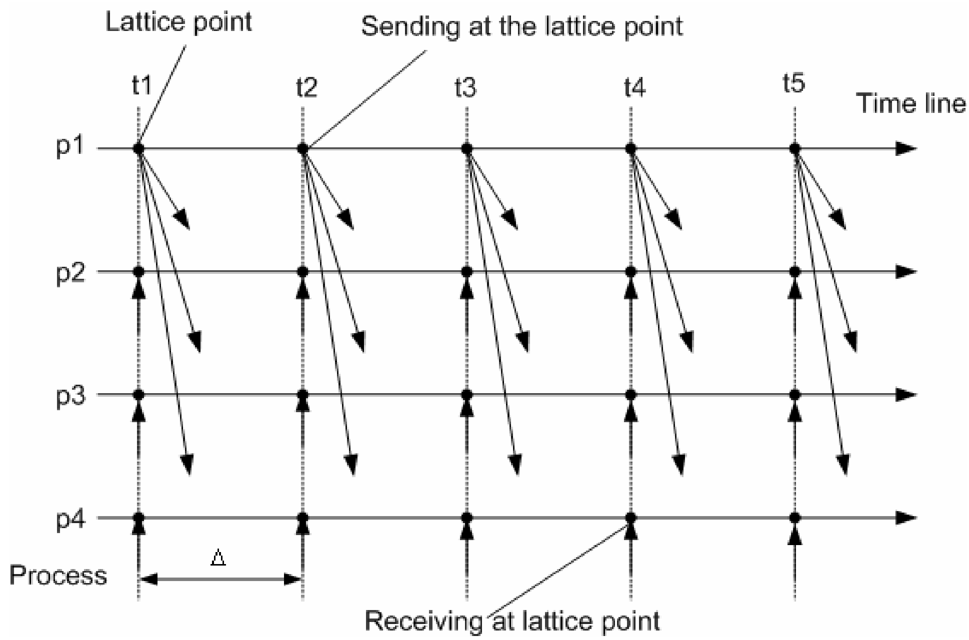
- The network will have a clock serve as a single reference. This approach is used in H.261/ISDN- (integrated services digital network) based systems. A single clock time is propagated around a set of codecs and multipoint control units (MCSs).
- The network deploys a clock-synchronization protocol, such as NTP (the network time protocol; Mills, 1993). The time stamps of media packets will be derived from the globally synchronized clocks. The isochronous synchronization approach as described in this article heavily relies on this time reference.

AN ISOCHRONOUS SYNCHRONIZATION APPROACH

The isochronous synchronization approach employs a clock-driven protocol for achieving multimedia synchronization (any one of three types of synchronization; Yang, Gay, Sun, Siew, & Sattar, 2002). This approach is particularly suitable for distributed collaborative multimedia environments where many-to-many multimedia communication is the basic interaction pattern. In this approach, multimedia synchronization is based on the use of synchronized physical clock time instead of any form of logical clock or sequence numbers, and thus clock synchronization across the distributed system is assumed. A real-time (synchronized) clock is incorporated in the system as a mechanism used for initiating significant events (actions) as a function of real time.

With globally synchronized clocks that satisfy the granularity condition, we can construct an *action lattice* (or *event lattice*; Kopetz, 1992). One dimension of this lattice represents the progression of time, the other dimension is the processes in the system (Figure 1). Processes in the system are designed to execute a simple *clock-driven protocol*, which requires that the events of sending and receiving messages are restricted to only

Figure 1. Lattice structure



occur at the lattice point of the globally synchronized space-time lattice (Figure 1). Thus, whenever an action has to be taken, it has to be delayed until the next lattice point of the event lattice.

This lattice structure greatly simplifies multimedia synchronization and readily maintains the temporal and causal relationship among the media.

The idea behind the clock-driven, isochronous synchronization is very simple and intuitive in that the easiest way to synchronize processes is to get them all to do the same thing at the same time. Using the simple mechanism based on the synchronized clock without requiring complex algorithms, the approach can equally well be applied to various multimedia applications in distributed environments, including live multimedia applications (live teleconferencing and CSCW) and stored media applications.

THE ORDERING PROPERTIES OF THE SYNCHRONIZATION PROTOCOL

In essence, what is really required for distributed multimedia synchronization is *order*; that is, a synchronization protocol must ensure that multimedia messages or streams are sent, delivered, and presented in an order that is consistent with the expected behavior of the distributed multimedia system as a whole. Clearly, multimedia systems have to precisely coordinate the relationships among all media. These relationships include temporal and spatial relationships. Temporal relationships are the presentation schedule of media, and spatial relationships are the location arrangements of media.

There are two specific cases concerning temporal order: *causal order* and *Δ-causal order*. These ordering concepts are derived from a happens-before relation, which is a more fundamental notion in distributed computing. The expression $a \rightarrow b$ is read as "*a* happens before *b*" and means that all processes agree that first, event *a* occurs, then afterward, event b occurs. The happens-before relation can be observed directly in two situations in a distributed environment: (a) If *a* and *b* are events in the same process and *a* occurs before *b*, then $a \rightarrow b$ is true, and (b) if *a* is the event of a message being sent by one process and *b* is the event of the message being received by another process, then $a \rightarrow b$ is also true. Obviously, a message cannot be received before it is sent or even at the same time it is sent since it takes a finite, nonzero amount of time to arrive. Note that happens-before is a transitive relation, so if $a \rightarrow b$ and $b \rightarrow c$, then $a \rightarrow c$.

The notion of causal order, as introduced by Birman and Joseph (1994), states that for any process, the order in which it delivers messages must respect the happens-before relation of the corresponding sending of the messages. More formally, a distributed computation *E*, all of whose messages is denoted as a set $M(E)$, respects causal order if for any two messages $m1$ and $m2$ and corresponding message-sending events $send(m1)$ and $send(m\_2)$, $send(m1) \rightarrow send(m2)$. If $m1$ and $m2$ have the same destina-