Synchronization in Joint-Viewing Environments

Kurt Rothermel, Gabriel Dermler

University of Stuttgart
Institute of Parallel and Distributed High-Performance Systems
7000 Stuttgart 80, Germany
E-mail: {kurt.rothermel, dermler}@informatik.uni-stuttgart.de

Abstract

Recent technology advances have made computer-based multimedia cooperation among distributed users feasible. Joint-Viewing (JV) is one such proposed cooperation concept. Correct interaction in it requires synchronization on various levels. In this paper we focus on synchronization aspects on the data stream level. We present several JV scenarios and state for each of these requirements concerning interstream synchronization. Then we introduce basic abstractions of a synchronization scheme, which is powerful enough to model a variety of JV configurations with their various synchronization needs. The presented scheme is based on the notion of a referencing system, which consists of a set of logical clocks, each defining a different time zone in the referencing system. Logical clocks provide timing information for input and output devices, which are used to produce and consume data in a synchronized fashion.

1 Introduction

Advances in communications and computer technology provide the technical foundation to integrate digital audio and video into today's distributed information processing systems. This is of great advantage for many application areas, especially for those, where users communicate and cooperate to achieve a common task. Joint Viewing (JV) systems are considered to be the basis for many applications in the field of computer-supported cooperative work [ElGi91], such as desktop conferencing [Cr90], joint-editing [FiKr88] and electronic classroom applications [ScCe87]. They provide the basic mechanisms for jointly viewing and manipulating information, user synchronization and user interaction, where the latter may be supported by appropriate video and audio channels.

Cooperative schemes go far beyond classical client/server schemes, where the clients are usually strictly isolated from each other. In cooperative environments, clients cooperate with each other in order to achieve a common task. This implies specific interaction patterns resulting in complex communication and synchronization requirements on various levels of abstraction.

In this paper, we focus on synchronization aspects on the data stream level [CoGa91]. Stream synchronization can be employed between data units within a continuous stream, called intrastream synchronization, and between data units of different streams, which is referred to as interstream synchronization. Various concepts and methods for intrastream synchronization have been proposed [WoMo91], [Fe91], [AlCa91], and most of them can be used for a wide range of applications. In contrast, the requirements imposed on interstream synchronization strongly depend on the corresponding application secenario. In JV scenarios, the sources and/or sinks of streams that have to be synchronized may reside on

This work was supported in part by the EC RACE-II research initiative, within the frame of the RACE project no. 2060 "Coordination, Implementation and Operation of Multimedia Services" (CIO). The views contained in this document are those of the authors, and should not be interpreted as representing official policies endorsed within the CIO project.

different sites, where streams may be of continuous or non-continuous nature. Moreover, streams to be synchronized may originate in different time zones of the same timing system.

The remainder of the paper is structured as follows. In Sec. 2, we introduce some terminology for expressing interstream synchronization requirements and apply this terminology when introducing a number of important JV scenarios. Then, in Sec. 3, we present the basic abstractions of a synchronization scheme, which is powerful enough to model a variety of JV configurations with their various synchronization needs. The presented scheme is based on the notion of a referencing system, which consists of a set of logical clocks, each defining a different time zone in the referencing system. Logical clocks provide timing information for input and output devices, which is used to produce and consume data in a synchronized fashion. Finally, we conclude with a brief summary and an outlook on future work.

2 Synchronization Scenarios in Joint-Viewing Environments

Joint Viewing was conceived as a computer-supported equivalent of a personal meeting, where people convene to jointly work on a work item by developing the work item and by exchanging visual and audible information among them. Joint Viewing allows these users to stay physically at their workstation and travel electronically instead.

A Joint Viewing scenario comprises users, applications and communication relationships among them. Users interacting with an application are presented with the same view of the application output. By introducing the notion of a shared window for the physical representation of the shared output we may alternatively say, the users view shared windows generated by applications.

Users can generate input to an application. The latter either can cope with multiple user input or requires external support of the Joint Viewing mechanism to select one of the users' input. Finally, the users are able to cooperate among each other by exchanging information about the jointly viewed shared windows. For this, they employ user-to-user communication means such as audio or video links or pointing tools, called telepointers, to point at objects in shared windows.

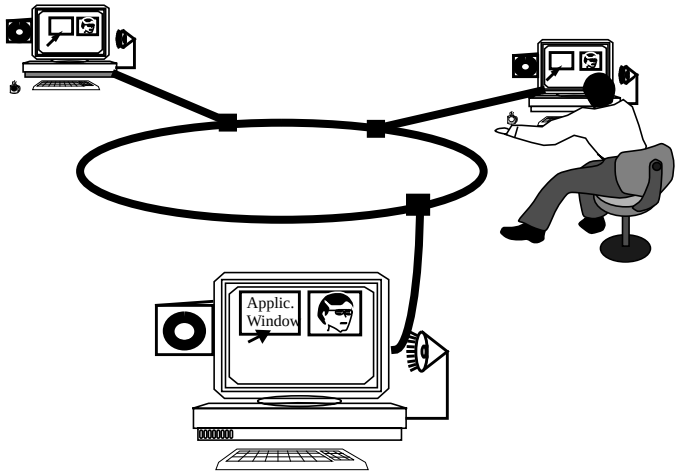

Figure 2.1: A Joint Viewing environment

A typical Joint Viewing scenario is shown in Figure 2.1. It involves joint viewing of an application window, the use of a pointing tool to point within the window and user-to-user

communication via audio and video links. The applications may run on either of the depicted machines or on a separate machine connected to the network.

The concept of Joint-Viewing as presented here, was defined within the “Joint-Viewing and Teleoperation Services” (JVTOS) working group of the CIO project. We refer to [DeFr92] for a detailed description of the JVTOS reference model.

In the remainder of this section, we combine the communication elements introduced above (shared windows, user-to-user links) to a set of scenarios with different requirements. We omit considering user input, since it is independent of data stream synchronization. We state synchronization requirements for both continuous and discrete streams.

Continuous streams, such as audio and video streams, comprise a sequence of periodic data units. We do not supply a final definition (or classification) of discrete streams. For this paper, we take the “usual” (text/graphics) output of an application as a typical representant of discrete streams comprising data units occurring at more or less arbitrary time intervals.

2.1 Synchronization Parameters for Joint-Viewing Scenarios

A data stream is generated by an input device (I) and consumed by an output device (O). We refer to a data stream by referring to the corresponding (I,O) tupel. (I,O) is unique since only one stream is attached to an output device[1]. Streams comprise data units. The temporal behavior of these can be observed both at the input and output device. We use I to refer to the input device I as the observation point of the temporal behavior of the generated data and O to refer to the output device as the observation point of consumed data. Temporal behavior is specified by referring to observation points by means of synchronization parameters.

For intrastream synchronization, requirements are expressed in terms of two parameters: delay and jitter. The first one refers to the maximal delay of a data unit of a stream (I,O) between its input and output at the corresponding devices. The second denotes the variance of delay allowed (maximum delay - minimum delay)[2]. Requirements have to state how these values are to be bounded.

The selection of delay bounds considered convenient by users depends on the application used and the user involved. A user requesting fast interactivity with an application, will request low delay bounds for the application output. Similarly, an application requiring frequent input, will be expected to exhibit low output delay. For continuous streams tight jitter bounds are required. Experimental values for audio and video are compiled in [HeSa90]. For discrete streams usually no explicit jitter bounds are specified, thus bounding jitter only to the selected delay bound. Yet, in some cases, as for instance for computer animation sequences, it may be necessary to specify tighter jitter bounds.

For specifying interstream synchronization requirements the skew parameter is used. In order to use it in cooperative environments, we define the following construct:

Skew[(P_1,P_3) -> (P_2,P_4)] < skew_bound.

---

1 We use indices to distinguish between different input and output devices. Input and output device of a stream have the same primary index. Output devices fed by the same source are distinguished by a secondary index.

2 Of course, the values may be defined based on a probabilistic approach.

P_1, P_2, P_3 and P_4 denote observation points of data streams (i.e. input/output devices). The construct above implies that the same data units are observed at P_1 and P_2. The data units can be physically the same, for instance if P_1 and P_2 denote the source and sink device of a data stream. However, it is only required that the data unit content and sequence number is to be the same, as is the case if at P_2 copies of units are visible which were observed at P_1. An analogous relationship is implied for P_3 and P_4.

Let D_1 denote an arbitrary data unit observed at P_1 at time t_1 and D_3 a data unit observed at P_3 at time t_3 (see Fig. 2.2). The same two data units are observed at P_2 and P_4 respectively, at times t_2 and t_4. The skew expression from above simply states, that if between D_1 and D_3 a certain distance in time (d1) was observed at points P_1 and P_3, the distance (d2) between D1 and D_3 as observed at points P_2 and P_4 does not differ from d1 by more than skew_bound. More formally:

Skew[(P_1,P_3) -> (P_2,P_4)] < skew_bound
<=> for all D_1, D_3:
$| (t(P_3,D_3) - t(P_1,D_1)) - (t(P_4,D_3) - t(P_2,D_1)) | <$ skew_bound,
if $t(P_3,D_3) - t(P_1,D_1) \geq 0$ and
$| (t(P_1,D_1) - t(P_3,D_3)) - (t(P_2,D_1) - t(P_4,D_3)) | <$ skew_bound,
if $t(P_3,D_3) - t(P_1,D_1) < 0$.

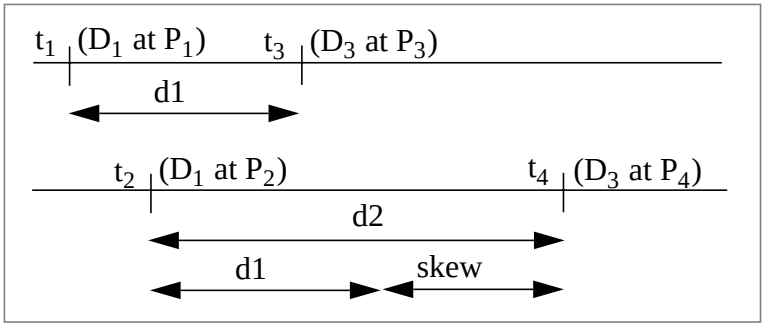

Figure 2.2: definition of skew

As will be shown in section 3, time can be related to different referencing systems. For instance, if time denotes the sequence number of data units of a specific reference stream, all data units of a second stream carry as timestamp the sequence number of the last data unit observed in the reference stream. In such cases, the skew construct is to be augmented by stating which of the observation points observes the reference data units.

The definition given above applies both to live and storage data sources. In the latter case, not creation time is observed initially but the specification of presentation time, as stored together with the data units, for instance as timestamps.

The defined skew construct proves powerful in specifying synchronization requirements between distributed observation points in cooperative environments. For instance, consider an input device generating data units for two users watching their output devices $O_{1,1}$ and $O_{1,2}$. We require that the consumption of the same data units occur at approximately the same time (not more than skew_bound apart), i.e.:

Skew[(I,I) -> $(O_{1,1}, O_{1,2})$] < skew_bound.

It should be remarked that specifying delay and jitter bounds for single streams may automatically bound the skew between them as well. However, for interstream synchronization the primary requirement remains that a skew bound is to be ensured, regardless of how it is

done. It is up to the implementation to decide (under consideration of constraints concerning resources, location of sources and sinks, etc.), what requirements are to be derived ultimately.

2.2 Synchronization Scenarios

Scenario 1

In this most simple case shown in Fig. 2.3a, two users view the output of an application through one shared window. There is no communication (i.e no cooperation) between the users. Therefore, synchronizing the contents of the two shared windows is not necessary, i.e.:

Skew[(I_1,I_1) -> ($O_{1,1}$,$O_{1,2}$)] = arbitrary.

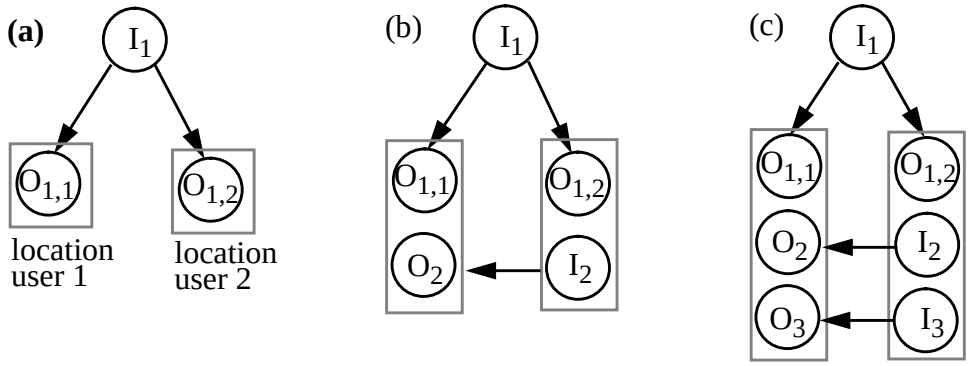

Figure 2.3: scenarios 1 and 2

Scenario 2

The previous scenario is augmented by one-way communication links which are employed by user 2 to refer to the shared window content (see Fig. 2.3 b and c). He may use a telepointer to point at objects in the window, an audio link to comment on them and perhaps a video link to show real objects relating to the shared window content or to support audio communication. The scenario reflects a master-slave relationship where the master gives explanations about the objects jointly viewed. The users may switch their roles, thus making information exchange in both directions possible. However, at any time only one-way communication is possible.

In a first step (Fig. 2.3b), we assume that one communication link exists. One new requirement has to be met: data units generated by user 2 relating to a shared window content are to be received by user 1 while viewing the same shared window content. For example, for a telepointer link, this implies that its movements are visible in front of the same window background at both user sites. Formally, the new requirement is:

(1) Skew[($O_{1,2}$,I_2) -> ($O_{1,1}$,O_2)] < skew_bound.

The bound value depends on the characteristics of objects referred to. If they change rapidly, the bound will have to be correspondingly low. At the other extreme, static objects may not require any skew bound at all.

Consider now the case that both an audio and a telepointer stream are employed by user 2 (Fig. 2.3c). The relation between each of these and the shared window content is as required in (1) with possibly different bound values. However, since the two streams may

be related to each other as well (e.g. user 2 saying: “I mean the object I point to”), we state a requirement between these streams as well. The same set of requirements is necessary, if user 2 employs audio and video communication. Thus, we require:

(1) Skew[($O_{1,2}$,I_2) -> ($O_{1,1}$,O_2)] < skew_bound
(2) Skew[(I_2,I_3) -> (O_2,O_3)] < skew_bound.

The skew bound for (2) may be different depending on the involved stream types. We expect that it will be in the range of the jitter bounds of the streams. If the jitter bounds differ, we expect that specifying the skew bound in the range of the largest is a reasonable approach. Own experiments with a telepointer and an audio link have shown, that values up to 0.5 sec are acceptable, if the telepointer is moved at usual mouse pointing speed.

A third case allows communication via telepointer, audio and video at the same time. The specification of the requirements is developed like in the previous case.

Scenario 3

The scenario depicted in Fig. 2.4a differs from the last one in that two-way communication between the users is allowed like in a conference situation. It requires that things occurring at one user location are perceived as occurring at virtually the same time at the location of the other user. For instance, for audio the requirements would have to ensure that the users are able to discuss like in a face-to-face meeting. In the following we only consider communication between users via one medium. Extending the scenario to simultaneous communication via multiple media (like in the previous scenario) leads to analogous requirements. Formally, we require:

(1) Skew[(I_1,I_1) -> ($O_{1,1}$,$O_{1,2}$)] < skew_bound
(2) Delay[I_2, O_2] < nnd.
(3) Delay[I_3, O_3] < nnd.

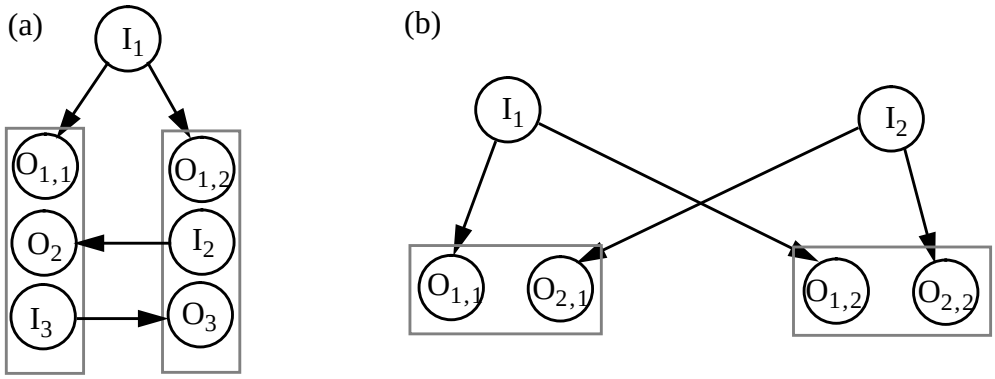

Figure 2.4: scenarios 3 and 4

Requirement (1) states that the content of the shared windows should not diverge more than specified by the skew bound, while (2) and (3) are requirements which are the direct consequence of the conferencing requirement. The delay is restricted in both directions to a value which is not noticeable to humans. For audio and video the jitter bound values of [HeSa90] can be applied. For telepointing a value below 0.4 sec was found satisfactory.

Scenario 4

Joint-Viewing is not restricted to one shared window. An application can generate content for many windows, which may be required to be presented in a synchronized fashion. This

may be the case, even if the windows are generated by different possibly remote applications. A good example for this is provided by scientific visualization, where experimental results may be viewed on line on the computer display, and compared with simulation results generated by a computer program and presented on the computer display.

The requirements for such a situation with two users (see Fig. 2.4b) are expressed as:

(1) Skew[(I_1,I_2) -> $(O_{1,1},O_{2,1})$] < skew_bound
(2) Skew[(I_1, I_2) -> $(O_{1,2},O_{2,2})$] < skew_bound.

These requirements arise from presenting different shared windows. The scenario can be augmented by communication links, as it was done for scenarios with one shared window. The resulting requirements are identical to those presented in the context of these scenarios.

The skew bounds depend on the applications involved. For instance, for many cases of scientific visualization a very low bound can be expected, since matching simulation and experimental results correctly is of essence. In contrast, for a travel agency, where a still picture of a hotel is presented in one window and text information about cost appears in another, the skew bound can obviously be less strict.

3 Abstractions for Data Stream Synchronization

In the following, we will describe a very simple but powerful model for stream-level synchronization in distributed multimedia systems. This model will serve as a foundation for the programming abstractions our prototype will be based upon. A more comprehensive description of this model will be provided in [DeRo93].

Some of the basic abstractions are very similar to that introduced in [AnHo90]. We also define *logical devices*, such as virtual microphones, speakers, video windows, telepointer devices, that can be mapped to *physical devices*. We also distinguish between *input devices*, which produce data units, and *output devices*, which consume data units. Since physical devices play no role in the remainder of this section, we always mean logical devices when using the notion device.

Information can be conveyed from an input device to one or more output devices by means of *data streams*. A data stream is a unidirectional communication channel that transfers data units. Each data unit may have one or more references (or timestamps), that temporally relate this data unit to other data units of the same and other streams. Our model covers continuous as well as discrete data streams.

Basically, synchronization of data streams is performed on the basis of references that correlate data units of different streams with regard to their relative display time. References can be realized in many different ways, such as implicit (e.g. sequence numbering) and explicit timestamping [AnHo90], or interleaving of data streams [LeBa90]. In our model, we provide the concept of a *referencing system (RS)*, to group a number of data streams that may be synchronized on the basis of a certain referencing method. Consequently, an RS also defines the semantics of skew bounds required between data streams. Of course, multiple referencing systems of different type may coexist in a given MM-application.

More precisely, a reference system consists of a number of data streams and a set of *logical clocks*, which are bound to input and output devices of these streams. A logical clock

determines the current time in a certain “time zone” of an RS. Since there may be different time zones in an RS, multiple clocks may be needed, one for each zone.

As already mentioned above, logical clocks may be bound to input and output devices. Conceptually, all devices bound to the same clock operate in the same time zone. In an RS, multiple input and/or output devices may be bound to one logical clock, whereas each device may be bound to at most one clock. In other words, for a given RS a device may operate in at most one time zone.

In the simple scenario depicted in Fig. 3.1, referencing system RS1 comprises a (continuous) audio stream (A) and a (discrete) telepointer stream (T). It contains two clocks, LC1 and LC2, where LC1 is bound to the both input devices and LC2 is bound to the two output devices. That is, both input devices operate in one time zone, the one defined by LC1, and the output devices operate in another time zone, the one defined by LC2.

When describing the semantics of binding a device to a logical clock, we must distinguish between input and output devices. Let’s consider input devices first and assume that input device ID is bound to logical clock LC. Whenever ID produces a data unit, the current value of LC is read and assigned to the produced data unit. Therefore, if two devices bound to the same clock produce a data unit at the same time, then both data units carry equivalent timestamps. Now let’s consider output devices and assume that output device OD is bound to clock LC. The display of data units at OD is controlled by LC in the sense that a data unit with timestamp t is displayed at OD only when the value of LC does not differ from t by more than a specified skew bound. Consequently, two data units with equivalent timestamps are displayed at (almost) the same time if the corresponding output devices are bound to the same logical clock.

By binding input devices to clocks it can be specified which data streams can potentially be synchronized. The streams originating at input devices connected to a common logical clock carry (implicit or explicit) time information that can be used for synchronization purposes. Which streams are to be synchronized has to be specified in a second step by binding output devices to logical clocks. All data streams that are connected via their output devices to a common clock are synchronized on the basis of this clock and the timing information included in these streams.

In the example illustrated in Fig. 3.1, binding both input devices to LC1 only ensures that the (explicit or implicit) timestamps associated with the data units of the audio and telepointer stream are based on a common logical clock. This specification step says nothing about where and how this time information is exploited for synchronization purposes. This is done in a second step, which binds the both output devices to LC2. As will be seen in more complex examples below, we gain a lot of flexibility by separating specification into these two steps.

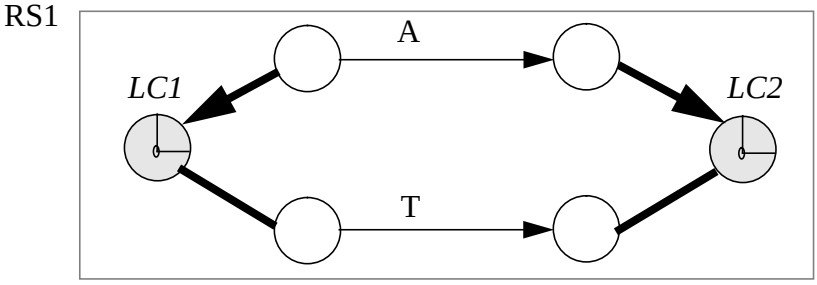

Figure 3.1

In our model, the advance of logical clocks can be driven in different ways. In particular, it can be driven by input devices, output devices or some external devices, such as real time clocks. To be able to capture this fact in specifications, we distinguish between *active and passive bindings*. A device is said to be actively bound to a logical clock if it triggers the advance of this clock, otherwise it is called passively bound. Now, we will consider when and how the different devices cause actively bound clocks to advance:

- *Input devices:* A clock actively bound to an input device "ticks" whenever this device produces a data unit. How far a clock advances when it ticks depends on the type of the RS as well as on the type of the device. For example, if the device produces continuous data, ticking is determined by the (natural) data rate of this device. If an RS implements Lamport clocks [La78] for including (explicit) timestamps into discrete data streams, the clock value advances by one whenever a data unit is produced.

- *Output devices:* Each time an output device that is actively bound to a logical clock consumes a data unit, the value of the bound clock is set to the timestamp of this data unit. Note that no assumptions are made about the type of the device. In the case of a device consuming continuous data, an interrupt of a device-internal clock may cause the logical clock to advance. (This corresponds to Anderson's logical timing systems driven by a master device [AnHo90]). In the case of an output device consuming discrete data, the data arrival on a communication channel may drive the clock advance.

- *External devices:* The advance of a clock can be also triggered by an external device, such as a real time clock. This does not say that the logical clock has the value of the driving real time clock but only means that the logical clock advances with the same rate as the real time clock.

So far, nothing has been said about how and when logical clocks are started. This is subject to start-up synchronization [AnHo90], [LiGh91], which may be a complex procedure, especially in more complicated settings (for example see Fig. 3.2 to 3.4).

In the simple scenario depicted in Fig. 3.1, the audio input device is actively bound to LC1, while the audio output device is actively bound to LC2. That is, LC1 and LC2 advance in the natural data rate of the audio input device and audio output device, respectively. At the input side, this setting enables the audio stream to carry implicit timestamps only. Data units of the telepointer stream have to be associated with timestamps, which - for instance - may be encoded as sequence numbers of audio data units. At the output side, this setting ensures that the audio device displays data units at its natural rate. Display of telepointer data units is synchronized according to LC2, which advances at this rate.

The notion of time may differ from RS to RS. For example, one RS may implement clocks as counters which are incremented whenever an actively bound device produces or consumes a data unit. Another RS, may implement clocks that advance at (approximately) the same rate as real time. Clearly, to make synchronization possible all logical clocks in a given RS have to be based on the same notion of time. RS may also differ in the way they encode timestamps.

To show the flexibility of our model, we will finally discuss some other scenarios that are more complex than the one introduced in Fig. 3.1. Reference system RS2 in Fig. 3.2 models a Joint Viewing scenario similar to the one illustrated in Fig. 2.3b. RS2 comprises two video streams (V1 and V2), originating at the same device, and one telepointer stream (T). Streams V1 and T have to be synchronized in the following sense: if V2's output device and T's input device consume resp. produce two data units at the same time, then T's and

V1's output device must consume these two data units also at (approximately) the same time.

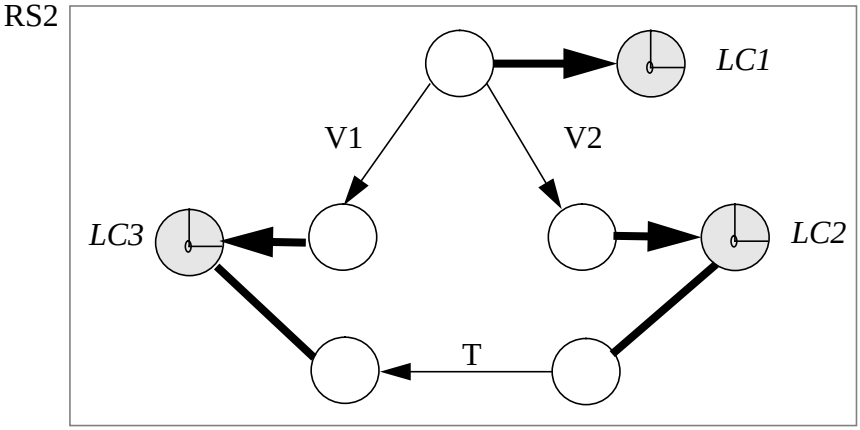

Figure 3.2

RS2 contains logical clocks LC1, LC2 and LC3. The video input device is actively bound to LC1, and V2's output device is actively bound to LC2. In other words, the value of LC2 corresponds to the timestamp of the latest data unit consumed (i.e. displayed) by V2's output device. Since T's input device is (passively) bound to LC2, each telepointer data unit gets associated with a timestamp, which correlates it with the video data unit that has been displayed at the time it has been produced. This timestamp information is used to synchronize streams T and V1. LC3 is actively bound to V1's output device and passively bound to T's output device. That is, the video data units are displayed at the natural rate of the video output device. The telepointer data units are displayed according to the advance of LC3. Remember that the starting of logical clocks is subject to a start-up procedure. LC3 may only be started when enough video and telepointer information is available at both output devices to ensure a synchronized consumption of both streams.

The Joint Viewing scenario illustrated in Fig. 3.3 is modelled by means of the two reference systems RS4 and RS5. In this scenario, two users share a window, in which discrete data are displayed. This is modelled by the two discrete data streams X1 and X2, which originate at the same input device. One user communicates with the other via a telepointer channel and an audio channel. This one-way communication is reflected in our model by audio stream (A) and telepointer stream (T). Two synchronization requirements can be indicated in this scenario: (1) Streams A and T have to be synchronized with each other and (2) stream A has to be synchronized with stream X1 in the following sense: if X2's output device and A's input device consumes resp. produce a data unit at the same time, then A's and X1's output device must consume these two data units also at the same time. RS5 ensures the first synchronization requirement while the latter one is fulfilled by RS4.

The input device of X1 and X2 is actively bound to logical clock LC1. This clock may be realized by a counter for instance, which is incremented whenever a data unit is produced in the bounded input device. Since this input device produces discrete data, the value of such a logical clock has no relationship to real time. Since LC2 and LC4 are in the same RS as LC1, they are based on the same notion of time. X2's output device is actively bound to LC2, which causes the value of this clock to correspond to the timestamp (e.g. sequence number) of the latest data unit displayed by X2's output device. By passively

binding A's input device to LC2, the produced audio data units get temporally related to the displayed X2 data units.

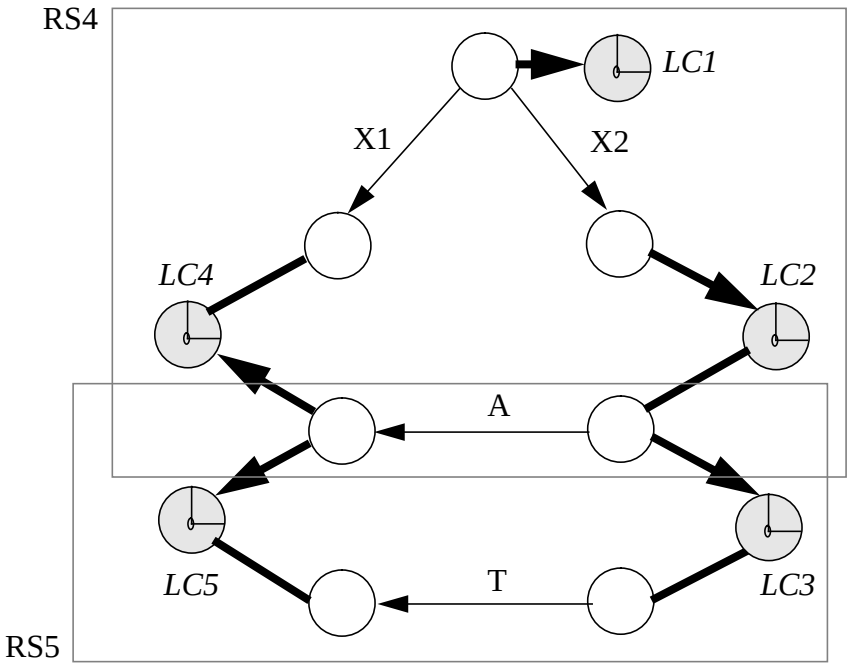

Figure 3.3

RS5 is equivalent to RS1 depicted in Fig. 3.1. RS5 and RS4 are connected by stream A, whose data units are associated with two timestamps, one for each RS. A's output device is actively bound to LC4 and LC5, which reside in different RS. As the advance of these two clocks is driven by A's output device, audio data units can be consumed at the natural rate of this device. The output devices of X1 and T consume their data units in accordance to the advance of LC4 and LC5, respectively. This ensures that streams X1, A and T are synchronized with each other.

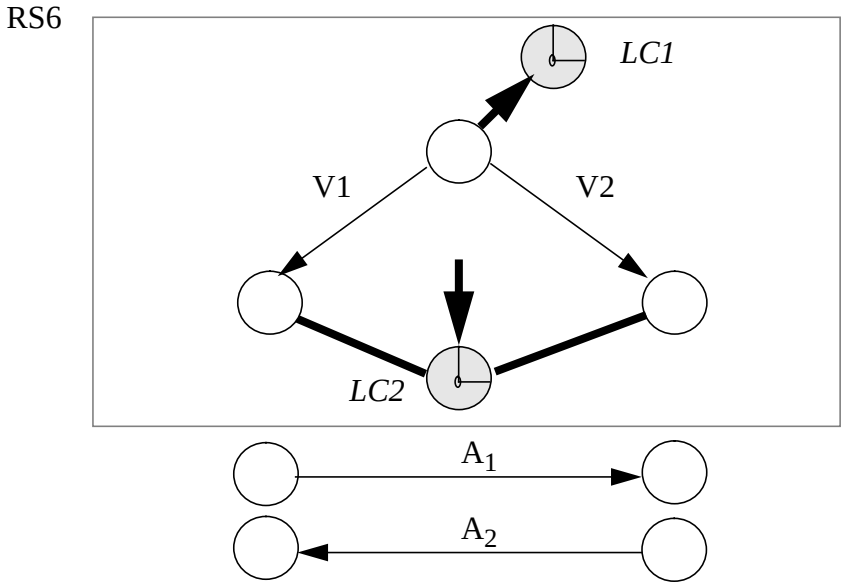

Figure 3.4

The setting depicted in 3.4 corresponds to scenario 3 described in the previous section. Two users share a video window which is modeled by the output devices of streams V1 and V2. Both users communicate with each other via audio channels A1 and A2, which are assumed to have a non-noticeable delay. As discussed in the previous section this scenario requires V1 and V2 to be synchronized.

Referencing system RS6 includes two logical clocks, LC1 and LC2. The former is actively bound to the video input device, while the latter is passively bound to the video output devices. LC2 is actively bound to an external device, such as a global time generator.

Of course, the above examples cannot cover all aspects of synchronization in Joint Viewing environments. A more comprehensive discussion of the concept of referencing systems in the context of JVTOS environments will be given in [DeRo93].

4 Summary and Future Research

In this paper we have first presented several JV scenarios along with their synchronization requirements. In a second part, we have defined a set of abstractions, which combined appropriately, prove powerful in modelling JV configurations with their synchronization schemes. The building blocks of such schemes, were introduced as referencing systems, logical clocks and input/output devices. Examples of several synchronization schemes were provided.

The ultimate goal of the scheme developed is to provide a programming paradigm allowing a programmer to specify all information required for correct synchronization in cooperative environments. So, we will have to look into which information has to be specified by the programmer itself and which information can be derived from this specification by the underlying system. This depends very much on whether synchronization mechanisms prove to be general enough with respect to different application scenarios involving data streams of different characteristics. This applies in particular to the start-up phase of synchronization in a distributed environment.

We will also reconsider the definition of skew, which currently proves to be too rigid, especially with respect to synchronization between discrete data streams. Another open issue is how a global clock can be realized in a distributed environment without a more or less inherently synchronized behavior. Finally, we will investigate what other types of logical clocks and referencing systems are useful in specifying distributed synchronization schemes.

References:

[AlCa91] Almeida N., Cabral J., Alves A.: “End-to-End Synchronization in Packet-Switched Networks”, Proc. of the “Second Intl. Workshop on Network and Operating System Support for Digital Audio and Video”, Nov 1991, to be published by Springer-Verlag

[AnHo90] Anderson P., Homsy G., Govindan R.: “Abstractions for continuous media in a network window system”, Report No. UCB/CSD 90/596, Computer Science Division, University of California, Berkeley, Sep 1990

[CoGa91] Coulson G., Garcia F., Hutchinson D., Shepherd D.: “Protocol Support for Distributed Multimedia Applications”, Proc. of the “Second Intl. Workshop on Network and Operating System Support for Digital Audio and Video”, Nov 1991, to be published by Springer-Verlag

[Cr90] Crowley T. et al.: “MMConf: An Infrastructure for Building Shared Multimedia Applications”, Proc. of the “Third Conference on Computer-Supported Cooperative Work”, ACM, New York, 1988

[DeFr92] Dermler G., Froitzheim K.: “Joint Viewing and Teleoperation Services: A Reference Model for a New Multimedia Service”, to be presented at the “4th IFIP Conference on High-Performance Networking”, Liege, Dec. 1992

[DeRo93] Dermler G., Rothermel K.: “Referencing Systems: A Programming Paradigm for Stream Synchronization”, in preparation

[ElGi91] Ellis C., Gibbs S., Rein G.: “Goupware - some Issues and Experiences”, Communications of the ACM, Vol.34, No.1, Jan 1991

[Fe91} Ferrari D.: “Design and Applications of a Delay Jitter Control Scheme for Packet-Switching Internetworks”, Proc. of the “Second Intl. Workshop on Network and Operating System Support for Digital Audio and Video”, Nov 1991, to be published by Springer-Verlag

[FiKr88] Fish R., Kraut R., Leland M., Cohen M.: “Quilt: A Collaborative Tool for Cooperative Writing”, Proc. of the “Conference on Office Information Systems”, Mar 1988, ACM 1988

[HeSa90] Hehmann D., Salmony M., Stuttgen H.: “Transport Services for Multimedia Applications on Broadband Networks”, Computer Communications, Vol. 13, No. 4, 1990

[La78] Lamport L.: “Time, Clocks, and the Ordering of Events in Distributed Systems”, Communications of the ACM, vol. 21, July 1978

[LeBa90] Leung W., Baumgartner T., Hwang Y., Morgan M., Tu S.: “A Software Architecture for Workstations Supporting Multimedia Conferencing in Packet-Switched Networks”, IEEE Journal on Selected Areas in Communications, Vol. 8, No. 3, April 1990

[LiGh91] Little T., Ghafoor A.: “Scheduling of Bandwidth-Constrained Multimedia Traffic”, Proc. of the “Second Intl. Workshop on Network and Operating System Support for Digital Audio and Video”, Nov 1991, to be published by Springer-Verlag

[ScCe87] Scigliano J., Centini B., Joslyn D.: “A Real-Time Unix-based Electronic Classroom”, Proc. of the “IEEE Southeastcon ‘87”, IEEE 1987

[WoMo91] Wolfinger B., Moran M.: “A Continuous Media Data Transport Service and Protocol for Real-Time Communication in High Speed Networks”, Proc. of the “Second Intl. Workshop on Network and Operating System Support for Digital Audio and Video”, Nov 1991, to be published by Springer-Verlag