Multimedia synchronization

M. Haindl

Computer Science/Department of Interactive Systems

CS-R9538 1995

Report CS-R9538
ISSN 0169-118X

CWI
P.O. Box 94079
1090 GB Amsterdam
The Netherlands

Multimedia Synchronization

Michal Haindl
CWI
P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

email mh@cwi.nl

Abstract

This paper presents a hierarchical synchronization model for the description of the time relations and regimes necessary for the presentation of multimedia or animated data which have either natural or implied time dependencies. The model generalizes some previous multimedia synchronization models by unifying time and event based synchronization concepts and offering a consistent framework in which to handle dynamic media presentation functionality.

AMS Subject Classification (1991): 68M20, 94B50
CR Subject Classification (1991): D.4.1, D.4.4, D.4.8, H.3.5, H.5.1
Keywords & Phrases: Synchronization, multimedia.
Note: This paper is accepted for the IEEE Journal on Selected Areas in Communications.

1. Introduction

Multimedia refers to the presentation of collections of both static and dynamic data (i.e. data with natural time dependencies e.g. audio) in a specified order and time. Therefore, their mutual synchronization must assure a proper temporal order of presentation events. Multimedia synchronization can be defined as a mutual assignment of data items and time instants. These time instants may be known in advance (e.g. standard consumer data players) or they can be also results of some unknown function of time (event driven synchronization) or known with some limited accuracy (e.g. random network delays). This concept of time allows for the merging of common multimedia and hypermedia notions [12].

Time is a crucial notion for synchronization. The Encyclopedia of Science & Technology [17] offers the following definition: "Time is the dimension of the physical universe, at a given place, orders sequence of events, also denotes a designated instant in this sequence." Our notion of time follows this definition and is similar to the notion of reference point introduced by Steinmetz [16]. Our time unit can be any time quanta (i.e. not necessarily seconds) of some periodic event (e.g. video frame presentation). The mapping between this notion of time and standard time units is not required to be an injection and there is in fact, no need to know this mapping for synchronization purposes at all. This is typically the case with event driven synchronization. A time unit that is smaller than the shortest presentation time in the system is a reasonable assumption to avoid presentation ambiguities.

Some synchronization models use different notions of time simultaneously, including local, virtual versus real, global, etc. However, synchronization creates one homogeneous system of data and time (further on just the term time is used), so different time notions are superfluous and the model presented here does not need them. All changes of presentation speed are just appropriate time mappings.

Data to be presented is not restricted to basic media data (e.g. image, audio). The notion of data here covers pure data itself (media data, parameters, look-up tables, etc.) as well as any processing (in some specified time) on them. One consequence of such a data definition is that there is no difference between animation and synchronization. Scheduling of both processes can be fully described by the synchronization model.

The target of a multimedia presentation is a human being with different sensitivity of his or her senses. For example, most humans are more sensitive to violation of audio or brightness synchronization and relatively insensitive to the violation of colour presentation accuracy. Data requiring different synchronization accuracy is typically presented in digital form by computer or a distributed system and thus the required accuracy cannot be guaranteed in all circumstances. Synchronization mechanisms have, therefore, to be combined with some sort of synchronization failure treatment.

Synchronization can be controlled inside an application by keeping track of time internally and delivering requests to one or more media servers at times specified by the time specification set. This provides no guarantee about the time at which requests are executed, however. Another possibility is a time/synchronization extension within the media server which guarantees request processing within certain specified time limits, if possible (e.g. there are no other requests or if it is combined with a resource reservation mechanism). Server controlled synchronization can be more efficient in transporting data over a network (e.g. batch files) and can reduce some delays introduced by the network using, for example, a time series prediction mechanism.

The next section discusses the hierarchical synchronization model based on a description of start and end data unit presentation points. Incomplete timing is discussed in subsections 2.1 and 2.2. Synchronization of several parallel data streams is described in section 3, while synchronization failure treatment is discussed in section 4. Finally some other synchronization models are compared with the proposed model in section 5.

2. Synchronization Model

The main purpose of the synchronization model is to predict the presentation time for each data unit. Knowledge of maximal possible network delay, database retrieval times, data decompression prefetching capacity, etc., can permit the file server to prepare data that is to be presented at an appropriate time. A synchronization model can be used by a central synchronizer to impose synchronization requests on single data objects or the model can be distributed so that data objects can mutually synchronize between themselves.

Our synchronization scheme is based on the start and end points of presentable data units. Both points are given by time values using the meaning of time discussed earlier. Let us have an ordered set of static data $P = \{p_1, \ldots, p_N\}$, in which some of the data items can be equal i.e. $p_j \equiv p_i$ for $i \neq j$. (The realization of data units that are equal is through distinct pointers referring to the same storage unit.) The set P can have a directed graph structure with one starting point if some of the data units have an event based time specification (see section 2.2). In this case, the data index in subsequent equations is a multindex which is updated in such a way to follow a selected path through this graph. Repetitions or loops of any length can easily be handled as well, they are formally handled as possibly infinite serial streams of data. All data units are assumed to be stored (or their existence is predictable) and may be derived from different media types (e.g. audio, text, image, graphics, etc.)[1]. Data units of the same media type may have different granularity. For example, it is possible to synchronize a data unit representing a one hour video sequence (played by some independent hardware and considered as a single data item) with another data item representing a single video frame.

This meaning of data has even wider consequences. An entire presentation can be arranged in a hierarchical way and the synchronization model can easily work at several levels of hierarchy.

[1]Note that we should also consider the synchronization with other "non-media" entities, such as colour table manipulation.

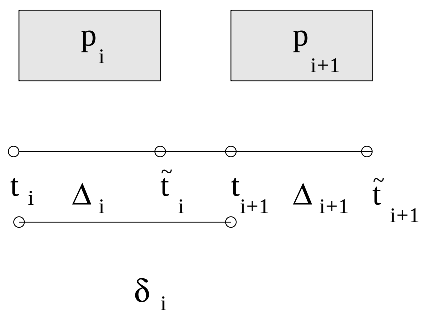

Figure 4.1: Time specification.

Let us reorganize the data set P into an ordered set of data subsets

$$P = \{p_1^{(h)}, \dots, p_{N_h}^{(h)}\}$$

where (h) denotes the h-th hierarchy level starting from the finest grain level $h = 1$, and single subsets are recursively defined as

$$p_i^{(h)} = \{p_{i,1}^{(h-1)}, \dots, p_{i,i_h}^{(h-1)}\} \ .$$

The finest (atomic) synchronization level data unit is denoted $p_i^{(1)} = p_i$.

The meaning of a single hierarchy level and any particular data clustering into higher level subsets depends on the application, e.g. $p_i^{(3)}$ can be a video film, $p_i^{(2)}$ a single scene from this video, with $p_i^{(1)}$ being a single video frame.

This hierarchical model not only supports top-down or bottom-up authoring, but also enables a consistent framework for a multimedia application running in different support environments. We can imagine a multimedia application running in the Internet HTML environment, with the possibility to synchronize only the start of video and audio streams $(h = 2)$, and the same application in a local network environment with mutual synchronization of every frame with the corresponding audio chunk $(h = 1)$.

The set ordering of presentable data units is given by the author and is therefore known in advance. Note that this a priori knowledge only concerns the mutual time constraints of data, and not the time values themselves. I.e. $t_i^{(h)} \leq t_j^{(h)} \ \forall i < j$ where $t_i^{(h)}$ is the start of the presentation of the i-th data item. The model does not require a priori knowledge of $t_i^{(h)}, t_j^{(h)}$ values.

This data ordering can be complete or it can be given as a relative ordering between pairs of start/end times of the data units, e.g. $t_a^{(h)} < t_b^{(h)}, t_b^{(h)} = t_c^{(h)}, t_c^{(h)} \leq t_d^{(h)}$ etc. In the later case, it is necessary to check if such a system has a solution (it can be contradictory, e.g. $t_a^{(h)} < t_b^{(h)}, t_b^{(h)} < t_c^{(h)}, t_c^{(h)} < t_a^{(h)}$) and then to find this solution (a complete data ordering). This can be accomplished in the graph representation of time constraints using graph algorithms for detecting infeasible loops, for example.

Let t be the time modelled, for example, by a processor timer, and $\Delta_i^{(h)}$ be the standard presentation interval for the i-th data unit $p_i^{(h)}$ (Fig. 4.1). Then:

$$\Delta_i^{(h)} = \tilde{t}_i^{(h)} - t_i^{(h)} \tag{1}$$

where $\tilde{t}_i^{(h)} \geq t_i^{(h)}$ is the finishing time for the presentation ot the i-th data unit. Other synchronization points (other than start and end) can be specified in a data unit presentation interval, but such systems have a straightforward one to one mapping to the system used.

The time interval between start points of two subsequent data units is given by:

$$\delta_i^{(h)} = t_{i+1}^{(h)} - t_i^{(h)} \tag{2}$$

where $\delta_i^{(h)} \geq 0$ and $\delta_{N_h}^{(h)} = 0$.

Presentation times on different hierarchy levels are related through:

$$\Delta_i^{(h)} = \tilde{t}_{i,i_h}^{(h-1)} - t_{i,1}^{(h-1)} \tag{3a}$$

and start point intervals are related:

$$\delta_i^{(h)} = \sum_{j=1}^{i_h} \delta_{i,j}^{(h-1)}. \tag{4a}$$

If a $p_i^{(h)}$ subset contains elements with equal time specifications (e.g. a video sequence) then (3a),(4a) can be simplified into

$$\Delta_i^{(h)} = (i_h - 1)\delta_i^{(h-1)} + \Delta_i^{(h-1)} \tag{3b}$$

and

$$\delta_i^{(h)} = i_h \delta_i^{(h-1)}. \tag{4b}$$

In this case, an author specifies only the (h)-th level time specification set which is then automatically projected into the $(h-1)$-th level one.

Note that there is always mapping from the (h)-th level time point to all corresponding lower level $(h-1, \ldots, 1)$ time points. Time mapping in the opposite direction does not exist for every lower level time point. Obviously $t_1^{(h)} = t_1$ and $\tilde{t}_{N_h}^{(h)} = \tilde{t}_N$ for all h.

Synchronization starts at time $t_s^{(h)}$ and from data unit $p_s^{(h)}$.

a) Forward presentation - the synchronization impulse starting presentation of the $(s+n)$-th data unit in the direction of the natural presentation time flow $(p_i^{(h)}, p_{i+1}^{(h)}, \ldots)$, $(N - s \geq n \geq 0$ and $\sum_{i=j}^{l} \delta_i^{(h)} \overset{\text{def}}{=} 0$ for $j > l)$ is issued at time:

$$t_{s+n}^{(h)} = t_s^{(h)} + k \sum_{i=s}^{s+n-1} \delta_i^{(h)} \tag{5}$$

where k is a speed control constant, and the end of its presentation is at time:

$$\tilde{t}_{s+n}^{(h)} = t_{s+n}^{(h)} + k\Delta_{s+n}^{(h)} = t_s^{(h)} + k(\Delta_{s+n}^{(h)} + \sum_{i=s}^{s+n-1} \delta_i^{(h)}). \tag{6}$$

b) Similarly, for reverse presentation $(n \geq 1,\ n \leq s-1)$:

$$t_{s-n}^{(h)} = t_s^{(h)} + k(\sum_{i=s-n}^{s-1} \delta_i^{(h)} - \Delta_{s-n}^{(h)}) \tag{7}$$

and:

$$\tilde{t}_{s-n}^{(h)} = t_s^{(h)} + k \sum_{i=s-n}^{s-1} \delta_i^{(h)} \tag{8}$$

Note that reverse intervals have identical durations with their forward counterparts. If more data units $p_{i_j}^{(h)}$ start at time $t_s^{(h)}$ then $p_s^{(h)}$ is such that $s = \min_j\{i_j\}$ for forward presentation.

A presentation (forward or reverse) occurs at a standard speed if the constant $k = 1$. Fast-forward or fast-reverse presentation occurs if $0 < k < 1$ and slowed down presentation occurs if $k > 1$. Random access (i.e. a forward jump to a specific presentation item) is given by: $p_s^{(h)} \rightarrow p_{s+n}^{(h)}$ thus $\delta_i^{(h)} = \Delta_i^{(h)} = 0$ for $i = s, s+1, \ldots, s+n-1$ and a reverse jump by: $p_s^{(h)} \rightarrow p_{s-n}^{(h)}$ is for $\delta_i^{(h)} = \Delta_i^{(h)} = 0$ for $i = s-1, s-2, \ldots, s-n+1$ and $\delta_{s-n}^{(h)} = \Delta_{s-n}^{(h)}$.

Changing the synchronization level (accuracy) towards a more precise synchronization means replacing the (h)-th level terms in (5)-(8) equations with their lower level counterparts from (3),(4). The resulting finer granularity of data introduces several additional synchronization points for each previous point.

A loop synchronization of length l (forward or reverse) follows model (5)-(8) with

$$\left.\begin{aligned} p_i^{(h)} &\equiv p_{i\pm jl}^{(h)} \\ \Delta_i^{(h)} &= \Delta_{i\pm jl}^{(h)} \\ \delta_i^{(h)} &= \delta_{i\pm jl}^{(h)} \end{aligned}\right\} j = 1, 2, \ldots$$

where $p_i^{(h)}$ are data that are repeatedly presented in the loop and j is the number of loop repetitions.

There are five alternative ways (Cartesian products mutually convertible to each other) of specifying synchronization time in this model. It can be the time set:

$$\Theta = \{(t_i^{(h)}, \tilde{t}_i^{(h)}) : i = 1, \ldots, N_h\} \tag{9}$$

where $\delta_i^{(h)}, \Delta_i^{(h)}$ are substituted in (5)-(8) with (1),(2).

It can be the time shift Cartesian product:

$$\Theta = \{(\delta_i^{(h)}, \Delta_i^{(h)}) : i = 1, \ldots, N_h\}. \tag{10}$$

The time specification set (9) has the advantage of providing mutual data independence (see (5)-(8)). Any data unit or subset of them (e.g. an audio track) can be removed without influencing the others. This data independence can be seen especially in a distributed environment with unpredictable data arrival as an undesired feature. Note that because of the known complete data ordering it is not required to know all time reference points in advance. They can be supplied at runtime preceding each data unit activity with the time necessary to retrieve this data unit only. Relative time shifts (10) can be easily changed at run time and mutual data time relations are still maintained. If a data unit $p_i^{(h)}$ is removed from data set P, then it is necessary to remove $(\delta_i^{(h)}, \Delta_i^{(h)})$ and possibly to change $\delta_{i-1}^{(h)}$ in $(\delta_{i-1}^{(h)}, \Delta_{i-1}^{(h)})$ (similarly for inserting of a new data unit).

The remaining three Θ options are $(t_i^{(h)}, \Delta_i^{(h)})$, $(\tilde{t}_i^{(h)}, \delta_i^{(h)})$, and $(\tilde{t}_i^{(h)}, \Delta_i^{(h)})$. The time specification set can also be a unity of several partial time specification subsets, each using its own time specification option, if it is for any reason useful.

The static set of data P taken together with any of these optional Cartesian products creates the *dynamic data set* $\mathcal{P}$ (e.g. $\mathcal{P} = \{(p_i^{(h)}, \delta_i^{(h)}, \Delta_i^{(h)}) : i = 1, \ldots, N_h\}$). The dynamic data set $\mathcal{P}$ represents the synchronization part of a multimedia document.

The synchronization of all common data players or recording devices (e.g. CD, film projector, gramophone, video recorder, tape-recorder, camera, etc.) can be described by the simplest single level version of the hierarchical synchronization model (11). These devices present data units with a

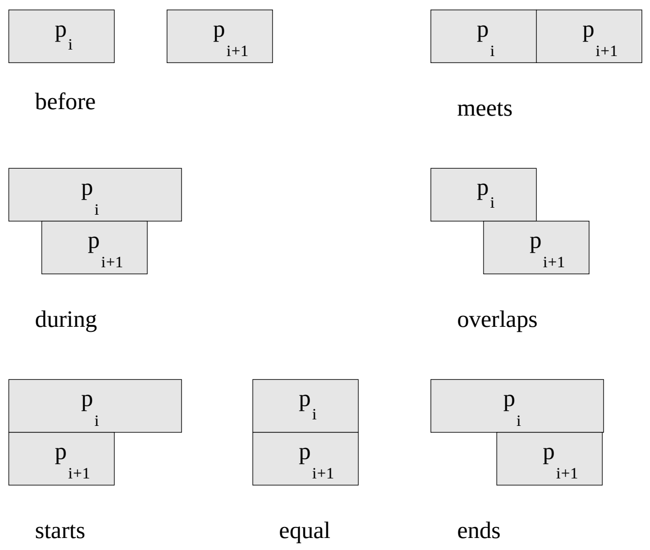

Figure 4.2: The complete set of two interval temporal relations.

constant presentation/recording interval $\Delta_i = \Delta\ \forall i = 1, \ldots, N$ and a constant time interval between subsequent data units $\delta_i = \delta\ \forall i = 1, \ldots, N$. These constant data intervals simplify the model equations (5)-(8) to:

$$t_{s+n} + c_3 = \tilde{t}_{s+n} + c_4 = t_{s-n} + c_5 = \tilde{t}_{s-n} + c_6 = kn\delta \tag{11}$$

where c_i are the corresponding constants.

The complete set of temporal relations between two time intervals consists of 13 possibilities [6] (before, meets, during, overlaps, starts, ends, equal and their inversions). All these time relations (see Fig. 4.2) as well as their modification as discussed in [8] are straightforwardly included in our synchronization model. I.e. before $\delta_i^{(h)} > \Delta_i^{(h)}$, meets $\delta_i^{(h)} = \Delta_i^{(h)}$, during $\delta_i^{(h)} < \Delta_i^{(h)} \wedge \tilde{t}_{i+1}^{(h)} < \tilde{t}_i^{(h)}$, overlaps $\delta_i^{(h)} < \Delta_i^{(h)} \wedge \tilde{t}_{i+1}^{(h)} > \tilde{t}_i^{(h)}$, starts $\delta_i^{(h)} = 0$, ends $\tilde{t}_i^{(h)} = \tilde{t}_{i+1}^{(h)}$, and equal $\delta_i^{(h)} = 0 \wedge \Delta_i^{(h)} = \Delta_{i+1}^{(h)}$.

We will simplify the notation in the following sections by dropping the hierarchy index, which is the same in all subsequent equations if not stated otherwise.

2.1 Incomplete Timing

Real synchronization systems cannot guarantee the precise timing required by the synchronization model under all situations and very often it is not even necessary. It is therefore advantageous to introduce the notion of incomplete timing to help the time/synchronization system whenever it is possible. A presentation of synthetic temporal relations, like an animation of still images, graphics, text, etc. usually does not require keeping strict temporal relationships. In contrast, each synchronization failure in an audio stream and to a less extent in video sequence is clearly distinguishable. Another example, dynamic data mixed with purely static parts, e.g. a video sequence interrupted with a text page where the video's resumption depends on some reader confirmation, is discussed in the following subsection.

Let us assume that the true presentation time $(t_i, \tilde{t}_i)$ differs from the specified time $(\bar{t}_i, \bar{\tilde{t}}_i)$ due to errors caused by an imprecise system clock, inaccurate data delivery, request servicing, etc., then the presentation can be specified:

$$t_i = \bar{t}_i \odot \tau_i \tag{12}$$

where the operator $\odot$ is equivalent to the operator $\pm$ or $+$ and τ_i is the error estimation of the $\bar{t}_i$, or a random variable for the stochastic version of the model (see later).

Algebraic Model Let us assume equal time errors:

$$\tau_i = \tau \quad \forall i \tag{13}$$

and the operator $\odot$ to be $\pm$ then:

$$\Delta_i = \bar{\Delta}_i \odot 2\tau \tag{14}$$

and:

$$\delta_i = \bar{\delta}_i \odot 2\tau. \tag{15}$$

If we further assume the start impulse is without an error, the equations (5)-(8) now become for the algebraic model:

$$t_{s+n} = t_s + k \sum_{i=s}^{s+n-1} \bar{\delta}_i \odot e_{s+n} \tag{16}$$

$$\tilde{t}_{s+n} = t_s + k(\bar{\Delta}_{s+n} + \sum_{i=s}^{s+n-1} \bar{\delta}_i) \odot \tilde{e}_{s+n} \tag{17}$$

$$t_{s-n} = t_s + k(\sum_{i=s-n}^{s-1} \bar{\delta}_i - \bar{\Delta}_{s-n}) \odot e_{s-n} \tag{18}$$

$$\tilde{t}_{s-n} = t_s + k \sum_{i=s-n}^{s-1} \bar{\delta}_i \odot \tilde{e}_{s-n} \tag{19}$$

where

$$e_{s+n} = \tilde{e}_{s-n} = 2n\tau k \tag{20}$$

$$\tilde{e}_{s+n} = e_{s-n} = 2(n+1)\tau k \tag{21}$$

If the assumption (13) does not hold, then (20),(21) become sums of individual errors for each of $e_{s+n}, \tilde{e}_{s+n}, e_{s-n}, \tilde{e}_{s-n}$.

The equations (16)-(21) can be used to trigger synchronization failure treatment. τ (τ_i) is then interpreted as the maximum acceptable error and the failure condition occurs if the real t_{s+n} is out of the specified range (16), e.g. p_{s+1} data unit is presented only if its presentation can start in time:

$$t_{s+1} \in \langle t_s + k\bar{\delta}_s - 2\tau k; t_s + k\bar{\delta}_s + 2\tau k\rangle$$

otherwise, a synchronization failure treatment is performed.

Let us now assume that the start times can only be delayed (e.g. late data arrival or request serving) but the presentation time Δ_i is always maintained (expansion method). Again assuming (13), the $\odot$ operator in (12) now become $+$ and (14) is replaced by equation (22):

$$\Delta_i = \bar{\Delta}_i. \tag{22}$$

The equations (16)-(19) hold as before, (15),(20) hold if the constant 2 is removed, but (21) now becomes:

$$\tilde{e}_{s+n} = e_{s-n} = n\tau k. \tag{23}$$

Instead of specifying δ_i, Δ_i (or $t_i, \tilde{t}_i$) as constants, they can be defined using inequalities (24)-(25) with fixed limits.

$$\Delta_{\min_i} \leq \Delta_i \leq \Delta_{\max_i} \tag{24}$$

$$\delta_{\min_i} \leq \delta_i \leq \delta_{\max_i} \tag{25}$$

Time specifications (24),(25) help to recover from synchronization failures and should be used whenever possible, because they decrease the frequency of failure treatment invocations (described in section 4). For example, in the case of late data unit arrival $(\delta_i \rightarrow \delta_{\max_i})$ we can prolong presentation of the previous unit $(\Delta_i \rightarrow \Delta_{\max_i})$ and shorten the gap between subsequent data units $(\delta_{i+1} \rightarrow \delta_{\min_{i+1}})$. This time tolerance should be exploited in such a way that subsequent data can again operate from the middle of their tolerance intervals and so decrease the chance for a subsequent synchronization failure treatment invocation (16),(18).

If (12),(13) is interpreted as our acceptable constant time tolerance, then (14)-(21) hold, $\Delta_{\min_i} = \bar{\Delta}_i - 2\tau$, $\Delta_{\max_i} = \bar{\Delta}_i + 2\tau$, $\delta_{\min_i} = \bar{\delta}_i - 2\tau$, $\delta_{\max_i} = \bar{\delta}_i + 2\tau$, and they describe the predicted presentation time of the $s+n$ $(s-n)$ data unit. If we cannot assume equal tolerance intervals for all data then (20),(21) do not hold anymore and must be replaced by sums of individual tolerances. A similar possibility is to relate these tolerance intervals to some data unit requiring higher synchronization precision (e.g. an image presentation can be directed by its audio commentary $\Delta_{image_i} \leq \Delta_{audio_i}$).

Stochastic Model Let us assume time to follow the equation (12) with the operator $\odot$ being $+$ and random variable (random noise) τ_i. Random variables τ_i are assumed to be uncorrelated:

$$E\{\tau_i \tau_j\} = 0 \quad \forall i \neq j \tag{26}$$

$$E\{\tilde{\tau}_i \tau_j\} = 0 \quad \forall i, j \tag{27}$$

with constant mean and variance:

$$E\{\tau_i\} = \mu \quad \forall i \tag{28}$$

$$var\{\tau_i\} = \sigma^2 \quad \forall i \tag{29}$$

(e.g. Gaussian noise). The characteristics μ, σ^2 do not need to be known, they can easily be estimated.

Lemma 1 *Assume (12),(26)-(29) hold, then the time interval's stochastic moments are (30)-(34).*

$$E\{\Delta_i\} = \bar{\Delta}_i \tag{30}$$

$$E\{\delta_i\} = \bar{\delta}_i \tag{31}$$

$$cov\{\Delta_i, \Delta_j\} = \begin{cases} 2(\sigma^2+\mu^2) & for \;\; i=j \\ 0 & otherwise \end{cases} \tag{32}$$

$$cov\{\delta_i, \delta_j\} = \begin{cases} -\sigma^2-\mu^2 & for \;\; i=j-1 \vee i=j+1 \\ 2(\sigma^2+\mu^2) & for \;\; i=j \\ 0 & otherwise \end{cases} \tag{33}$$

$$cov\{\delta_i, \Delta_j\} = \begin{cases} \sigma^2+\mu^2 & for \;\; i=j \\ -\sigma^2-\mu^2 & for \;\; i=j-1 \\ 0 & otherwise \end{cases} \tag{34}$$

We chose the predictor to be the mean value. The following lemma specifies the stochastic model equations.

Lemma 2 *Under assumptions of Lemma 1, the predicted time stochastic moments are (35)-(40).*

$$E\{t_{s+n}\} = t_s + k\sum_{i=s}^{s+n-1}\bar{\delta}_i \tag{35}$$

$$E\{\tilde{t}_{s+n}\} = t_s + k(\bar{\Delta}_{s+n} + \sum_{i=s}^{s+n-1}\bar{\delta}_i) \tag{36}$$

$$E\{t_{s-n}\} = t_s + k(\sum_{i=s-n}^{s-1}\bar{\delta}_i - \bar{\Delta}_{s-n}) \tag{37}$$

$$E\{\tilde{t}_{s-n}\} = t_s + k\sum_{i=s-n}^{s-1}\bar{\delta}_i \tag{38}$$

and

$$var\{t_{s+n}\} = var\{\tilde{t}_{s-n}\} = 2k^2(\sigma^2+\mu^2) \tag{39}$$

$$var\{\tilde{t}_{s+n}\} = var\{t_{s-n}\} = 2k^2(\sigma^2+\mu^2). \tag{40}$$

Proof The proof of the first lemma uses basic stochastic moment properties. The second lemma proof uses the results of Lemma 1.

Synchronization failure treatment for the stochastic model can be based on a statistical hypothesis test (e.g. $H_0 : \tau_i \in \mathcal{N}(\mu, \sigma^2)$) or a similar strategy to the algebraic model can be used.

Finally this model can easily be generalized for $t_i = \bar{t}_i + a\tau_i$ where a is an unknown parameter.

2.2 Event Based Synchronization

Synchronization can also be dependent on an event, such as, a user interaction (start/stop buttons, hyperlink traversal, etc.).

We define an event based presentation to be a data unit presentation in which at least one component of a time specification pair ((9),(10)) is unpredictable, i.e. some of $\Delta_{\min_i}, \Delta_{\max_i}, \delta_{\min_i}, \delta_{\max_i}$ (24),(25) can be 0 or ∞, i.e. $\Delta_{\min_i} = 0$ a data unit from a sequence is possibly not presented, $\delta_{\max_i} = \infty$. Nevertheless, to be able to synchronize an event based presentation with the rest of the data

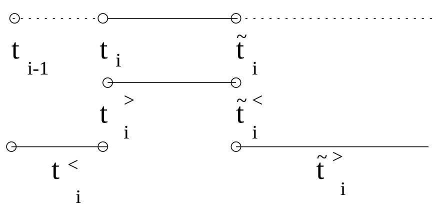

Figure 4.3: Conditional time specification range.

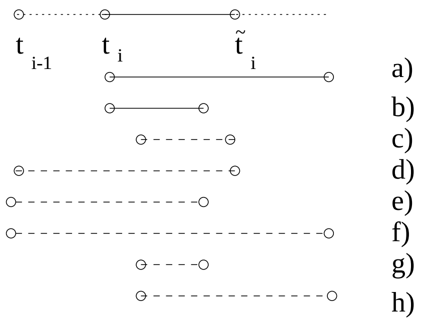

Figure 4.4: Event based time specifications (dashed line denotes a conditional presentation, a solid line presentation with conditional ending).

presentation, some relational time information (other than 0 or ∞) of this event must be known, otherwise the presentation will consist of two mutually independent (unsynchronized) parts.

Let us introduce the notation for the conditional activity $t_i^>$ (starts after), $t_i^<$ (starts before), $\tilde{t}_i^>$ (ends after), $\tilde{t}_i^<$ (ends before) with the following meaning (Fig.4.3):

$$t_{i+1} \geq t_i^> \geq t_i \;\wedge\; \tilde{t}_i \geq t_i^>$$

$$t_{i-1} \leq t_i^< \leq t_i$$

$$\tilde{t}_i^> \geq \tilde{t}_i$$

$$t_i \leq \tilde{t}_i^< \leq \tilde{t}_i$$

i.e. the p_i presentation can start after $(t_i^>)$ or before $(t_i^<)$ the specified time t_i. Similarly $\tilde{t}_i^>$, $\tilde{t}_i^<$ stands for conditional presentation finishing times.

Alternatively, for the time shift specification (10) the notations $\delta_i^<$, $\delta_i^>$, $\Delta_i^{x>}$, $\Delta_i^{<x}$ have the meaning:

$$0 \leq \delta_i^< \leq \delta_i$$

$$t_{i+1} - t_{i-1} \geq \delta_i^> \geq \delta_i$$

$$x \geq \Delta_i^{x>} \geq \Delta_i$$

$$0 \leq \Delta_i^{<x} \leq x.$$

Using this notation there can be 8 different (relative to a time interval) or 7 (for the specification type (10)) event based time specifications (see Fig.4.4):

$$(t_i, \tilde{t}_i^>) \equiv (\delta_i, \Delta_i^{\infty>}), \qquad (a)$$

$$(t_i, \tilde{t}_i^<) \equiv (\delta_i, \Delta_i^{<\Delta_i}), \qquad (b)$$

$$(t_i^{>}, \tilde{t}_i) \equiv (\delta_i^{<}, \Delta_i^{<\Delta_i}), \qquad (c)$$

$$(t_i^{<}, \tilde{t}_i) \equiv (\delta_i^{>}, \Delta_i^{\tilde{t}_i - t_{i-1}>}), \qquad (d)$$

$$(t_i^{<}, \tilde{t}_i^{<}) \equiv (\delta_i^{>}, \Delta_i^{<\tilde{t}_i - t_{i-1}}), \qquad (e)$$

$$(t_i^{<}, \tilde{t}_i{}^{>}) \equiv (\delta_i^{>}, \Delta_i^{\infty>}), \qquad (f)$$

$$(t_i^{>}, \tilde{t}_i^{<}) \equiv (\delta_i^{<}, \Delta_i^{<\Delta_i}), \qquad (g)$$

$$(t_i^{>}, \tilde{t}_i{}^{>}) \equiv (\delta_i^{<}, \Delta_i^{<\infty}). \qquad (h)$$

The time interval between start points of two subsequent data units (2) is for (a),(b) unchanged, for (c),(g),(h) shorter, and for (d),(e),(f) longer than the specification limits (10). Similarly the presentation time (1) is possibly longer for (a),(d),(f), shorter for (b),(c),(g) and can be both for (e),(h).

Two subsequent data specifications $\theta_i, \theta_{i+1} \in \Theta$ can both have relative time specifications ((a)-(h)).

Definition 1 *An* acceptable specification *is any couple of* $\theta_i, \theta_{i+1} \in \Theta$ *such that* $\delta_i^{<} \notin \theta_i \wedge \delta_{i+1}^{>} \notin \theta_{i+1}$ *(because these six combinations cannot be unambiguously resolved).*

A complete specification *is a couple of acceptable* $\theta_i, \theta_{i+1} \in \Theta$ *which generates the complete set of presentation intervals* $i, i+1$ *(see Fig. 4.2).*

A unique specification *is a couple of acceptable* $\theta_i, \theta_{i+1} \in \Theta$ *which cannot be replaced by any other acceptable couple* $\tilde{\theta}_i \neq \theta_i, \tilde{\theta}_{i+1} \neq \theta_{i+1}$ *in such a way that both couples describe the same subset of the complete set of two temporal intervals (Fig. 4.2).*

A limit specification *is a couple of* $\bar{\theta}_i, \bar{\theta}_{i+1}$ *which is created from an acceptable specification* θ_i, θ_{i+1} *replacing all relative specifications* $(t^{>}, t^{<})$ *with their corresponding limits.*

Lemma 3 *Every limit specification type, except "overlaps", has a complete specification.*

Proof These complete specifications are for "before" $[,(e)]$,$[(h),]$ for "meets" $[(b),(e)]$,$[(e),(e)]$, $[(h),(c)]$,$[(h),(h)]$ for "during" $[(e),(f)]$ for "starts" $[(e),]$,$[,(h)]$ for "equal" $[(f),(h)]$ and finally for "ends" $[(b),(e)]$,$[(e),(e)]$, $[(h),(h)]$.

It is obviously possible to specify the complete set of two interval temporal relations using acceptable specifications, but not every one of its 127 possible subsets has an acceptable specification. For example, there are no acceptable specifications for { meets,ends},{ during,overlaps,ends}, etc.

Lemma 4 *Every limit specification has at least two unique specifications. The number of unique specifications is at most 22 (less than half of the number of all acceptable specifications).*

Such unique specifications are for "before" $[(a),(d)]$,$[(f),(d)]$, for "meets" $[(a),(c)]$,$[(a),(d)]$,..., etc.

The algebraic model equations (16)-(19) hold for a time specification set Θ with some event based time specifications as well. The operator $\odot$ is the $+$ operator and assuming time specifications of $s+n, s-n$ items not event dependent:

$$e_{s+n} = \tilde{e}_{s+n} = k(\sum_{i=s}^{s+n-1} \delta_i^{e} - \delta_i) \qquad (41)$$

$$e_{s-n} = \tilde{e}_{s-n} = k(\sum_{i=s-n}^{s-1} \delta_i^\epsilon - \delta_i) \tag{42}$$

where δ_i^ϵ is defined:

$$\delta_i^\epsilon = \begin{cases} \delta_i & \text{for (a),(b)} \\ \delta_i^> & \text{for (d)-(f)} \\ \delta_i^< & \text{for (c),(g),(h)} \end{cases} . \tag{43}$$

Note that we are assuming ((41),(42)) the occurrence of all conditional presentations between s and $s+n$ $(s-n)$, otherwise all data dependent on a data unit specified by (c)-(h) cannot be presented. In the case of event dependent $s+n, s-n$ units, $\tilde{e}_{s+n}$ and e_{s-n} have to be updated also with their corresponding $\Delta_i^\epsilon - \Delta_i$ parts.

An event ϵ_i can be seen as a logical function of time dependent arguments ($\epsilon_i = f_i(\kappa_1(t), \ldots, \kappa_n(t))$) or alternatively, as a time unknown to us (in contradistinction to the $t_i, \tilde{t}_i$ notion of time).

This notion of "time unknown in advance" (or an unknown logical function of time) is useful for unifying hypertext link traversal with multimedia in the accepted meaning of this word. From the model's point of view, there is no difference between multimedia, hypertext or hypermedia presentation time prediction.

Let us assume that a hyperlink (p_i) can be followed only between starting time t_i and ending time $\tilde{t}_i$ if a corresponding event $\epsilon_i = 1$ (mouse button click) is issued. This situation is specified in (g).

$$0 \le \Delta_i^\epsilon \le \tilde{t}_i - t_i \tag{44}$$

Δ_i^ϵ is interpreted as the possible hot spot (hyperlink entry) activity time and:

$$0 \le \delta_i^\epsilon \le t_{i+1} - t_i \tag{45}$$

is the possible time delay between hyperlink activation and the start of the conditional presentation of a data unit pointed to by the hyperlink (if $\epsilon_i = 1$).

The specification (h) can be used for a user defined pause between data presentation, if we interpret a constant presentation (47) as a button click interval starting a suspended data stream presentation after a pause shorter (46) than the specified maximum.

$$\delta_i^\epsilon = t_{i+1} - t_i{}^> \le \delta_i \tag{46}$$

$$\Delta_i^\epsilon = \Delta_i \tag{47}$$

If the pause is longer than allowed (e.g. the user cannot select the correct answer button in a driving-licence test in time), the whole presentation ends. Similarly, other scenarios can be easily designed using event based specifications (a)-(h).

Incomplete timing and event-based synchronization differ only in their presentation path structures, the latter can describe a dynamic acyclic graph structure of any complexity, while the former describes a single path presentation only. In the case of the event-based synchronization, data must be prepared for all optional paths with a minimal retrieval time.

Let the dynamic presentation set $\mathcal{P}$ consist of n alternative branches, i.e. $\mathcal{P} = \bigcup_{j=1}^n \mathcal{P}_\mathrm{j}$, $\mathcal{P}_\mathrm{i} \not\equiv \mathcal{P}_\mathrm{j}$ $\forall i \neq j$, where $\mathcal{P}_\mathrm{i} \cap \mathcal{P}_\mathrm{j}$ can be non-empty. The presentation of the j-th branch depends on the approval event ϵ_j ($\epsilon_j = 1$ at the j-th branch starting time). Single events corresponding to a common graph node must be mutually exclusive:

$$\sum_{i=1}^n \epsilon_i^j = 1 \tag{48}$$

where $\epsilon_1^j, \ldots, \epsilon_n^j$ are approval events for single graph branches starting at the j-th node. An example, might be the famous Czech Cinema-automate where spectators vote in several places for one of a member of possible film continuations. If $p_i \in P_j$ is the arrival of a majority vote for the j-th presentation branch (usually one of two options) then $\Delta_i^\epsilon = \tilde{t}_i^< - t_i^>$. ((g)) is the possible voting time for starting the j-th presentation branch.

$$t_i^> = \begin{cases} t_i & \text{if } \epsilon_j(t_i) = 1 \\ \tilde{t}_i & \text{otherwise} \end{cases} \tag{49}$$

Where $t_i^> = \tilde{t}_i$ is interpreted as the j-th option's rejection.

Finally there can be any combination of event driven intervals and their upper and lower finite limits given by (24),(25). A presentation can wait for an event, but only for a specified time interval ((c),(d)). If the event does not appear in that time, the corresponding data unit is discarded. Incomplete timing (event driven) specifications can be mixed together with constant time specifications. For example, all still images can be specified using incomplete timing to give a synchronizer scheduler some manoeuvering space for solving synchronization failures, and they can be organized in several optional branches using event based specifications.

Let the i-th data unit be specified by incomplete timing and its neighbours have constant time specifications, then the rest of a dynamic stream can be dependent on this incomplete timing ((15),(c)-(h)) or independent ((a),(b)). In the case of subsequent data units being dependent on incomplete timing specifications, model equations (16)-(19) offer a wider or closer range of predicted time points. These time estimations can serve as feedback for an author to select appropriate time limits to achieve the required presentation behaviour.

It is possible to combine several hierarchical levels of event based time specifications as well. Taking again our Cinema-automate example, $p_i^{(h)}$ can be a film scene with a voting part and several optional endings, with $p_{i,j}^{(h-1)}$ being single film frames, audio chunks or voting parts. Remember that several hierarchic levels can also share the same data, i.e.

$$p_i^{(h)} \equiv p_i^{(h-1)} \equiv \ldots \equiv p_i^{(h-l)}$$

for some i, l with corresponding relations valid for $\delta_i^{(h)}, \Delta_i^{(h)}$. This solution is the most simple one because there is one to one mapping between different hierarchical levels of incomplete time specifications. Otherwise, if there is an incompletely specified data unit $p_i^{(1)}$ (except (b) not being the last item), then all higher level data units $p_j^{(h)} \cap p_i^{(1)} \neq \emptyset$ are also incompletely specified. Their incomplete specification results from the appropriate mapping of all of their incompletely specified sub-sets using (3),(4). E.g. (a) with (c)-(h) maps into a higher level (a) but (c)(a) maps into (h), etc. These possible specification combinations are not commutative, i.e. their result depends on their order.

An author is free to create any data grouping to define his hierarchy levels. He only has to keep one rule: Every $p_i^{(h)}$ data unit is not allowed to have more than one predecessor and more than one successor (Fig.4.5).

3. Parallel Synchronization

In this section we discuss the mutual synchronization of several parallel static or dynamic data streams from the point of view of the proposed synchronization model. We assume here that all synchronized parallel static or dynamic data sets have the same directed graph structure, or slave (or static) data streams have a subgraph structure of the master stream (see Fig.4.6).

Let us have some ordered set of static data $R = \{r_1, \ldots, r_R\}$. Synchronizing set R with dynamic data set $\mathcal{P}$ (for example, a sequence of still images with a musical composition) means creating the

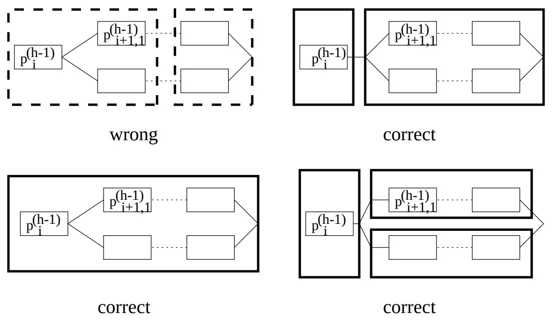

Figure 4.5: A hierarchical grouping example.

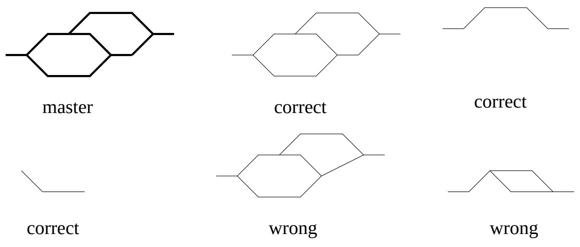

Figure 4.6: Examples of parallel data structures.

union set $P' = P \cup R$ with $card(P') = card(P) + card(R)$ and the corresponding Θ set, specifying its synchronization $\Theta_{P'} = \Theta_P \cup \Theta_R$. Data P keep their former time specification set Θ_P, while Θ_R can be its subset $\Theta_R \subset \Theta_P$ or can be some other set derived from Θ_P denoted by the long arrow symbol: $\Theta_R \longrightarrow \Theta_P$. The new dynamic data set $\mathcal{P}'$ again follows our synchronization model. If we need to synchronize data unit r_i with a part of data unit $p_j^{(h)}$ then we only have to change the hierarchy level into a finer grain level, i.e. $p_j^{(h)} = \bigcup_i p_{j,i}^{(h-l)}$ with corresponding lower level Θ_P.

Let us have two dynamic data sets $\mathcal{P}, \mathcal{R}$ which should be mutually synchronized, e.g. video and audio data. In this case one dynamic data stream must be the master synchronizer (e.g. $\mathcal{P}$) while the other $(\mathcal{R})$ must be a dependent data stream. Synchronizing both data sets means eliminate the dependent data Θ_R set, and then to follow the previous example, i.e. $P' = P \cup R$ and $\Theta_{P'} = \Theta_P \cup \Theta_{\hat{R}}$ with some $\Theta_{\hat{R}} \longrightarrow \Theta_P$.

A $\Theta_{\hat{R}}$ set can be created in two stages. First an automatic approximation of $\Theta_{\hat{R}}$ is computed given Θ_R, Θ_P and an acceptable tolerance for moving $\mathcal{R}$ time specification points. The second stage is the interactive correction of $\Theta_{\hat{R}}$ in a graph editor. This stage is optional and in simple cases (e.g. digital version of consumer data players) can be completely avoided.

The master synchronizer does not need to be from the same dynamic set, e.g., the first period the master synchronizer is $\mathcal{P}$, then $\mathcal{R}$ and finally again $\mathcal{P}$. One possible example is two video streams, each having parts with static scenes. Similarly, still image sequence with commentary in several languages will have the master audio channel combined from the longest single language audio comment corresponding to each still image. The synchronization solution is the same with previous one always the dependent part of data stream (the part with the least synchronization accuracy requirement) has to be made static and merged with the master dynamic data stream.

Synchronization has the transitivity property. If there should n be mutually synchronized data streams $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n$, they can be ordered according to their preference from left to right ($\mathcal{A}_1$ is master stream with the highest priority).

$$A' = \bigcup_{i=1}^{n} A_i \tag{50}$$

$$\Theta_{A'} = \Theta_{A_1} \cup \bigcup_{i=2}^{n} \Theta_{\hat{A}_i} \tag{51}$$

The associativity property is valid for data sets A_i but is not valid for Θ_i time specification sets $(\Theta_{\hat{A}_i} \longrightarrow \Theta_{A_{i-1}}$ for $i = 2, \ldots, n)$ and only data from A_1 keep their former Θ_{A_1} specification.

Several neighbouring data streams in the stream ordering can share an equal priority (e.g. $\mathcal{A}_{\mathrm{i}}, \mathcal{A}_{\mathrm{i}+1}, \mathcal{A}_{\mathrm{i}+2}$), and then they derive their time specification sets from the same higher priority time specification set $(\Theta_{\hat{A}_j} \longrightarrow \Theta_{A_{i-1}}$ for $j = i, i+1, i+2)$. An example is a multimedia presentation on a matrix of computer screens each showing a subimage from a mosaic synchronized with a master audio channel $(\mathcal{A}_{\mathrm{i}-1})$ to obtain the required data resolution.

Priority cycles $(\mathcal{A}_{\mathrm{i}}, \mathcal{A}_{\mathrm{i}+1}, \mathcal{A}_{\mathrm{i}+2}, \mathcal{A}_{\mathrm{i}})$

$$\begin{tikzcd}
& \Theta_{A_{i+1}} \arrow[dl] \\
\Theta_{A_i} \arrow[dr] & \\
& \Theta_{A_{i+2}} \arrow[uu]
\end{tikzcd}$$

are illegal, because they cannot be unambiguously resolved.

One of $\mathcal{P}$,$\mathcal{R}$ or both can be a loop. If the master set creates a loop then there are two possible scenarios - to run until all slave data are exploited, or to run permanently even without any slave data to be synchronized. If the slave set creates a loop, then it is reasonable to expect it to stop after all master data has run out. Finally, two loops can create an infinite presentation with possible synchronization shifts, if both loops do not have the same length (e.g. cineloops commonly used for medical ultrasound presentations).

The presented synchronization model (16)-(19) can be used to select between several parallel or hierarchical streams. For example, we can have several versions of a subsampled audio stream (DAT, CD, FM quality) and based on the prediction (16) and known network delay $(\tau^{(h)} = \tau)$ audio quality can be lowered if necessary to meet the required time constraints.

4. Synchronization Failure Treatment

In a real synchronization system without special reserve resources it is not possible to avoid data delays (e.g. slow data retrieval from a database, limited capacity of a communication channel, higher priority interrupts, etc.), which consequently cause synchronization anomalies. Failure treatment situations are usually solved using one of the following three strategies [11]:

Expansion (pausing, waiting) method: Each data unit is presented even if late. The result is the delay of all succeeding packets and the delay accumulation.

Ignoring (skipping) method: The missing data unit is ignored and presentation continues with first subsequent unit available in the required time.

Gap-compensation method: The missing data unit from a dynamic stream of data is reconstructed (i.e. prediction, alternation).

Each of these strategies can be described using the incomplete timing of section 2.1. The expansion method has unlimited intervals between data units:

$$\delta_{\min_i} \le \delta_i \ \forall i \tag{52}$$

and equations (13)-(15),(20),(21),(23) do not hold. The operator $\odot$ becomes the $+$ operator, $\delta_i = \bar{\delta}_i + 2\tau_i$ and the model equations (16)-(19) are completed with:

$$e_{s+n} = \tilde{e}_{s+n} = 2k \sum_{i=s}^{s+n-1} \tau_i \tag{53}$$

$$e_{s-n} = \tilde{e}_{s-n} = 2k \sum_{i=s-n}^{s-1} \tau_i . \tag{54}$$

The skipping strategy is specified by combining (25) possibly with $\delta_{\min_i} = \delta_{\min}, \delta_{\max_i} = \delta_{\max}$ and event specification (e), where event ϵ_{i+1} is arrival of the p_{i+1} data unit. If a data unit was skipped $(\delta_{i+1} < \delta_{\min_{i+1}} \vee \delta_{i+1} > \delta_{\max_{i+1}})$ and the relative timing (10) is used, then the following unit start of presentation should be recalculated to refer to the last properly presented data unit.

The gap-compensation strategy does not create a new situation for the synchronization model. The difference from the skipping strategy is that alternative data are presented, instead. Such an alternative can be the last data unit presented, data predicted from several last data units, standard data (blank image, icon), etc.

The ideal situation occurs if all presentable data are stored as frequency spectra then we propose to use the *Quality reduction method*: Data units are transported in frequency bands, starting from the lowest frequencies. Data items are presented in their requested time even if high frequency components are still missing, in the case of synchronization failure. This may result in a still image sharpening during its presentation, lower resolution video frames, lower quality audio, or missing textual parts.

The skipping, gap-compensation and our new proposed quality reduction methods require a mechanism to detect synchronization failures.

Several triggering strategies are possible depending on the application. It is possible to require the interval between finishing the presentation of the p_i data unit and the beginning of the subsequent unit p_{i+1} to be within a tolerance limit η_i (admissible rest), i.e.:

$$|\delta_i - \Delta_i - \bar{\delta}_i + \bar{\Delta}_i| \leq \eta_{a,i}, \tag{55}$$

the start point interval to meet the required precision (25),(56) (admissible fire):

$$|\delta_i - \bar{\delta}_i| \leq \eta_{b,i}, \tag{56}$$

admissible duration (24),(57):

$$|\Delta_i - \bar{\Delta}_i| \leq \eta_{c,i}, \tag{57}$$

admissible end:

$$|\tilde{t}_{i+1} - \tilde{t}_i - \bar{\tilde{t}}_{i+1} + \bar{\tilde{t}}_i| \leq \eta_{d,i} \tag{58}$$

or some of their combinations. A failure treatment is triggered for the admissible rest (55) strategy if:

$$|\tau_{i+1} + \tilde{\tau}_i| > \eta_{a,i}, \tag{59}$$

for (56) if:

$$|\tau_{i+1} + \tau_i| > \eta_{b,i}, \tag{60}$$

for (57) if:

$$|\tau_i + \tilde{\tau}_i| > \eta_{c,i}, \tag{61}$$

and finally for (58) if:

$$|\tilde{\tau}_{i+1} + \tilde{\tau}_i| > \eta_{d,i}. \tag{62}$$

If conditions (12),(13) hold, $\eta_{x,i} = \eta_x = \eta \ \ \forall i, x$ and $\tau < 0.5\eta$, then the correct synchronization is guaranteed for all presented data units and all strategies (55)-(58) and their possible combinations.

The trigger strategy conditions (59)-(62) can be used also in the form of statistical hypothesis tests for the stochastic model (35)-(40).

5. Related Work

In this section we discuss the relation of the synchronization model with some other previously published models.

Hierarchical synchronization [2],[11] or chained synchronization: Data objects are regarded as a tree consisting of nodes that denote serial or parallel presentation of the outgoing subtrees. Hierarchical structures can be synchronized at their beginning or end. Synchronization in [11],[10] is based on a Petri-net model and does not allow incomplete timing specification. This model has neither facility for describing incomplete timing specifications nor a facility to describe event based timing specifications (e.g. user interactions). A modification called Time Petri Net [13] extends Petri nets model with a potential firing interval that allows for an incomplete timing specification [4], but does not support event based synchronization.

No Petri net-based model can describe cycles of odd length (basic property of bipartite graphs like Petri nets).

Any such hierarchical tree or Petri net-based model can be easily mapped into our model, but the contrary does not hold. Note that our model does not prevent simultaneous (parallel) presentation of any number of data units.

Synchronization on a time axis [2] or continuous time model or cyclic synchronization: Single-media objects are attached to a time axis that represents an abstraction of time. Removing one object does not effect the synchronization of the other objects. This model has difficulties to describe event based synchronization as well as loops or complicated data structures. Such a synchronization can be achieved using our model with the time specification set (9). A variant of this model with time relations specified relative to the end of the previous data unit is called in [12] sequential synchronization or in the case of the "meets" type of presentation, intervals the serial synchronization. Parallel synchronization [12] describes the reference of several objects to a common time (activation time).

Synchronization at reference points [2],[16]: Single-media presentations are composed of subunits presented at periodic times. A position of a subunit in an object is called a reference point. Synchronization between objects is defined by connections between subunits of different objects that have to be presented at the same time (without explicitly referencing time). The advantage of the reference point concept is in possibility to easily change the presentation speed or to ignore presentation irregularities. Using only reference points, it is impossible to synchronize purely event driven parts or to model delays, speed changes, etc. This type of synchronization can be described as parallel synchronization with our periodic model (11).

Intra-object synchronization [2]: Synchronization of semantically related data units of the same type (e.g. video frames). Synchronization is again described by the periodic model (11) with a data set P containing only single media data units.

Inter-object synchronization [2]: Synchronization between the presentation of basic objects from different media. Media objects are synchronized using reference points. Synchronization is specified as a list of synchronization points (reference point with two or more basic objects attached to it).

Conditional synchronization: Is defined [12] as the activation of the presentation of objects to be dependent upon the state of a mark state variable. This variable is a logical expression and if its value is true, then the presentation of the data unit is activated. This type of synchronization is discussed in section 2.2, where single events are logical functions with time dependent arguments.

Recently two multimedia standards were proposed - the MHEG (Coded Representation of Multimedia and Hypermedia Information) and HyTime (Hypermedia Interchange Standard) [12]. Both standards use the virtual time concept together with time axis synchronization. HyTime also enables event based synchronization, while MHEG uses conditional presentation functionality together with hierarchical grouping (chained synchronization) of component objects. Both standards lack the incomplete timing functionality.

An algorithm for the recovery from loss of synchronization between interrupt-driven single-site media input/output devices is described in [1].

Synchronization between media streams during file storage and retrieval on multimedia networks is discussed in [15].

A common problem for all the models mentioned is their difficulty to handle general event based synchronization, some cannot handle incomplete timing specifications either.

6. Conclusion

This paper has presented a hierarchical model that offers a consistent framework for data synchronization research and understanding, as well as basis for a powerful synchronization system suitable for multimedia, hypertext, animation or any mixture of these.

The model represents major extension and generalization of the previously published multimedia synchronization models, briefly discussed in section 5. While the model is able to cover all of their functionalities, it does not suffer from some of their restrictions (event based synchronization, incomplete timing, hierarchical synchronization, complex graph type of presentation structure with optional paths, presentation time prediction). The synchronization model enables the mutual synchronization of any number of dynamic data streams, possibly described as a general acyclic graph together with static data. It is possible to freely mix several master synchronization streams, provided that at any time there is specified stream preference with one master synchronization channel. It is also possible to mix several sorts of time specifications (complete or incomplete) to help solve time presentation problems.

The model can be realized inside multimedia application or using special time extensions of a server. Realization inside an application requires a central synchronization object controlling data presentation or alternatively single data units can be active objects which synchronize themselves [16].

The model has to be supported by a sophisticated synchronization editor which has to be able, not only to fulfill standard tasks like layout and time data parameter input, but also supply a time constraint system solver, a mapping system between hierarchy levels, an ability to manipulate and fill with abstract data structures, and communication with a database. Such a synchronization system is completed with a presentation predictor and failure treatment subsystem.

The model does not suffer from any binding with a special graphical representation (e.g. Petri net), although a synchronization editor can profit from using a graph representation for a user friendly UI.

The model can easily be generalized from the linear modification of presentation time to a time dependent modification, if k becomes a time function and the sums in the model are replaced with corresponding integrals. The model can then support synchronization with smooth speed changes (e.g. uniform fast-forward or fast-reverse acceleration) or stepwise speed changes.

Finally, the model can be used to investigate mutual relations between two independent dynamic data streams $(\mathcal{P}, \mathcal{R})$ with known time specification sets Θ_P, Θ_R. This can be useful for improving multimedia document, multimedia document user performance evaluation, on-line control of several multimedia workstations, etc.

The presented model offers a large set of possible synchronization strategies and it can serve as the core of a powerful multimedia authoring/presentation system.

Acknowledgement
The model presented in this paper was developed by the author during his involvement in an EEC funded Esprit III project, MADE (no.6307) in Centre for Mathematics and Computer Science, Amsterdam, The Netherlands. The author thanks his colleague Graham Reynolds for reviewing this paper.

References

1. D.P. Anderson and G. Homsy, "A Continuous Media I/O Server and Its Synchronization Mechanism," *Computer*, vol. 24, no. 10, pp. 51-57, 1991.
2. G. Blakowski, J. Huebel, and U. Langrehr, " Tools for Specifying and Executing Synchronized Multimedia Presentations," In: R. G. Herrtwich (ed.) *Network and Operating System Support for Digital Audio and Video, Second International Workshop Heidelberg*, Lecture Notes in Computer Science No. 614, pp. 271-282, Springer, Berlin.
3. D.C.A. Bulteman, G. Rossum, and D. Winter, "Multimedia Synchronization and UNIX or If Multimedia Support is the Problem, Is UNIX the Solution?," In: *Proc. of the EurOpen Autumn 1991 Conf.*, 1991.
4. M. Diaz and P. Senac, "Time Stream Petri Nets a Model for Multimedia Stream Synchronization," In: *Proc. of Multimedia Modelling'93*, Singapore, 1993.
5. S. Gibbs, L. Dami, and D. Tsichritzis, " An Object-Oriented Framework for Multimedia Composition and Synchronization," In: *Eurographics Multimedia Workshop*, Stockholm, 1991.
6. C.L. Hamblin, " Instants and Intervals," In: J.T. Fraser (ed.) *Proc. of the 1st Conf. of the Intl. Society for the Study of Time*, Springer, New York, pp. 324-331, 1972.
7. M.E. Hodges, R.M. Sasnett, and M.S. Ackerman, " A Construction Set for Multimedia Applications," *IEEE Software* , no. 1, pp. 37-43, 1989.
8. P. Hoepner, "Synchronizing the Presentation of Multimedia Objects - ODA Extension," *SIGOIS Bulletin* vol. 12, no. 1, pp. 19-32, 1991.
9. M. Kummer and W. Kuhn, " ASE - Audio and Synchronization Extension of Compound Documents, " In: L. Kjelldahl (ed.) *Multimedia - Systems Interaction and Applications* ,
10. T.D.C. Little and A. Ghafoor, "Synchronization and Storage Models for Multimedia Objects," *IEEE J. on Selected Areas of Communication*, vol. 8, no. 3, pp. 413-427, 1990.
11. T.D.C. Little, A. Ghafoor, C.Y.R. Chen, C.S. Chang, and P.B. Berra, "Multimedia Synchronization," *Data Engineering*, vol. 14, no. 3, pp. 26-35, 1991.
12. B.D. Markey, "Emerging Hypermedia Standards - Hypermedia Marketplace Prepares for HyTime and MHEG," In: *Proc. of the Summer 1991 USENIX Conf. - Multimedia for Now and the Future*, Nashville, 1991.
13. P.Merlin, "A Study of the Recoverability of Computer Systems," Thesis, Comp. Science Dept., University of California, Irvine, 1974.
14. C. Nicolaou, " An Architecture for Real-Time Multimedia Communication Systems," *IEEE Journal on Selected Areas in Communications*,vol. 8, no. 3, pp. 391-400, 1990.

15. P.V. Rangan, S. Ramanathan, H.M. Vin, and T. Kaepner, " Techniques for Multimedia Synchronization in Network File Systems," *Computer Communication Journal*, no. 3, 1993.

16. R. Steinmetz, " Synchronization Properties in Multimedia Systems," *IEEE Journal on Selected Areas in Communications*,vol. 8, no. 3, pp. 401-411, 1990.

17. *Encyclopedia of Science & Technology*, McGraw-Hill, New York, 1977.