

Quality-of-Service based assessment of synchronization algorithms

Report

Author(s):

Class, Christina

Publication date:

1999-08

Permanent link:

<https://doi.org/https://doi.org/10.3929/ethz-a-004287275>

Rights / license:

[In Copyright - Non-Commercial Use Permitted](#)

Originally published in:

TIK Report 74

Quality-of-Service Based Assessment of Synchronization Algorithms

Christina Class

Abstract— In this report we present a Quality-of-Service based methodology to assess synchronization algorithms. The methodology comprises two steps: Firstly, synchronization algorithms are analyzed based on a defined set of QoS parameters for synchronization algorithms. Secondly, the analysis results are evaluated. The assessment methodology is validated by applying it to two well-known algorithms. Analysis and evaluation results as well as the assessment of the two algorithms are equally presented in this report.

I. INTRODUCTION

MULTIMEDIA applications allow for the processing and presentation of diverse media such as voice, images, text, etc. Some of these media are time dependent, i.e., change their values as a function of time. To offer multimedia applications being accepted by users, special attention has to be paid to the quality of data presentation and processing. One key aspect related to the quality of time dependent data such as video and audio is the achievement of timely correct presentation. This requires the preservation of time relationships during processing and presentation. The term *synchronization* refers to mechanisms that fulfill the task of defining and restoring temporal relationships of data. Research in the field of synchronization has been very active since 1990 and numerous solutions to the problem have been presented. The solutions developed solve the synchronization problem mainly for specific applications or in specific environments.

The task of synchronization in multimedia applications is to retain data's temporal relationships relative to each other or to a clock. This includes two steps. Firstly, the temporal relationships within and between data streams have to be specified. There are numerous proposals to specify such relationships ranging from path expressions [1] and media clocks [2] to petri net based approaches [3], [4]. [5] defines a temporal reference framework consisting of five different models of time. The framework allows for the classification of synchronization specification methods. The second aspect of synchronization comprises the enforcement of specified temporal relationships by synchronization algorithms. While many ideas have equally been published in this field, a detailed assessment scheme is still lacking. Therefore, a methodology is required allowing for the assessment of synchronization algorithms.

In this paper we describe an assessment methodology of synchronization algorithms which is based on the idea of QoS for synchronization: We define synchronization as *Quality-of-Service (QoS)*. Synchronization algorithms can then be understood as units that provide a specific service which can be measured and described in terms of QoS parameters. At this point the question arises what QoS is

offered by specific synchronization algorithms and whether the synchronization provided by these algorithms differs in term of QoS parameters.

The concept of QoS requires the configuration, prediction and maintenance of QoS parameters. What does this mean for synchronization? When are QoS parameters of synchronization maintained? What values are required for these parameters? In this paper we propose to analyze synchronization algorithm with respect to the QoS parameter values that can be guaranteed by the algorithm. Therefore, we define the notion of guaranteed synchronization in Section II before introducing the four QoS parameters for synchronization.

In Section III we present a methodology to assess synchronization algorithms. This methodology is based on the defined QoS parameters for synchronization. First a notion of time is introduced which allows to specify asynchronies. Then the four assessment parameters asynchronies, buffer requirements, synchronization errors, and delays are described in detail. A short paragraphs explains why only the combination of these four assessment parameters is meaningful for an assessment. The assessment methodology is then introduced. The section concludes with a discussion of additional relevant factors.

In Section IV the assessment methodology is applied to two different synchronization algorithms which have been published in the literature. We present results from analysis and evaluation, discuss experiences and present the assessment resulting from our methodology.

In Section V related work in classification and assessment of synchronization algorithms is presented. The paper concludes with a summary in Section VI.

Overall, the findings presented in this papers include: (1) the definition of synchronization as QoS, (2) the application of QoS to assess synchronization algorithms, (3) the definition of an assessment methodology for synchronization algorithms, and (4) the evaluation of the methodology by applying it to existing algorithms.

II. SYNCHRONIZATION

In general, end-systems and networks provide *services* that allow for the retrieval, transmission and display of multimedia data. But these services generally are not sufficient to provide the required quality of multimedia applications to the user by simply playing out data. Transmission delay jitters and delay variances in data processing in the end-systems may lead to unsynchronized data and reduce application quality. The *requirements*, in contrast, for high-quality multimedia applications, especially when involving continuous media as audio and video, are

high with respect to Quality-of-Service (QoS). Specifically for time-dependent data a gap exists between user requirements and services offered by end-systems, and networks. To guarantee for the display of time dependent data only while it is valid, specific methods have to be applied that fill this gap. The term *synchronization* refers to the application of such methods. Therefore, synchronization fills the gap between services offered by network and end-systems and requirements towards display of data (see Figure 1).

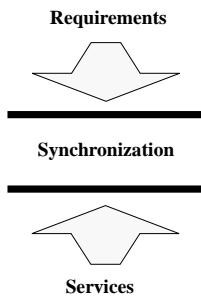


Fig. 1. Synchronization Gap between Services and Requirements

Two different tasks of synchronization can be distinguished: *Intra-stream* synchronization guarantees the correct display of data belonging to one single data stream. This implies that data has to be displayed in the correct order, which we refer to as *ordering aspect of intra-stream synchronization*, as well as at the time in which it is valid, which we refer to as *temporal aspect of intra-stream synchronization*. Of course, guaranteeing for the temporal aspect implies that the ordering aspect also is fulfilled. Guaranteeing first for an correctly ordered sequence of data units to be displayed, on the other hand, can facilitate synchronization as in a second step only the timely correct display of each first data unit has to be controlled.

The second aspect of synchronization is *inter-stream synchronization*. Its task consists in establishing the correct relations between data belonging to two or more different data streams. Time independent data units might be required to be displayed concurrently with data units of data streams. Inter-stream synchronization may be based on the assumption that all data streams are of the same relevance or may define one master stream, the other (slave) streams are synchronized to.

[6] differences between *event-driven synchronization* which is necessary for initiating an action within a distributed system, and *continuous synchronization*, when data presenting devices must be tied together so that they consume data at fixed rates. The algorithms that have been assessed with our methodology provide continuous synchronization.

Inter-stream synchronization requirements can be of different time scales. If a presentation is recorded, slides might be synchronized to the speaker's voice. Synchronization requirements for the audio stream and the slides are within milliseconds to seconds. In case of *lip-synchronization*, i.e., synchronization of a speaker's audio

with his video stream, synchronization requirements are more strict and in the lower milliseconds range. Even more stringent is the time requirement for synchronization of the two stereo channels of stereo audio being in the microseconds to milliseconds time range.

As data is presented and processed in computers in discrete values (bits and bytes), *continuous* data streams like sound, audio, and light, video, have to be digitized and quantized. One *presentation unit (PU)* is defined as the atomic information unit of a media stream that can be displayed independently from other presentation units. This is, e.g., an audio sample or a video frame. The duration of one PU in data streams is media dependent and can vary a lot. The *valid time interval* of PU i denotes the interval in which the presentation of PU i should be started correctly. The presentation time of a PU indicates the time at which it is presented in the receiver, whereas the reference time indicates the point in time at which the PU should be presented correctly in the receiver.

The term synchronization as used in this paper is based on the following, temporal definition:

Definition 1 The task of *synchronization* of multimedia data is to guarantee that all time dependent presentation units are only presented within their *valid time interval*. The valid time interval for each presentation unit is specified within the *synchronization specification* of multimedia data. ■

For all data we can define $T_{\text{opt}}(i) = [t_{\text{opt},\min}(i), t_{\text{opt},\max}(i)]$ as the valid time interval in the receiver for PU i . The interval in which the synchronization algorithm/application may start the presentation of PU i in the receiver is denoted as $T_{\text{real}}(i) = [t_{\text{real},\min}(i), t_{\text{real},\max}(i)]$. We can now define guaranteed synchronization.

Definition 2 We refer to synchronization as being *guaranteed*, iff

$$\exists B < \infty : \forall i : |t_{\text{opt},\min}(i) - t_{\text{real},\min}(i)| \leq B \quad \wedge \quad |t_{\text{opt},\max}(i) - t_{\text{real},\max}(i)| \leq B \quad (1)$$

Definition 2 is illustrated in figure 2. The two thick black lines in the bottom illustrate the values $b_1 = |t_{\text{opt},\min}(i) - t_{\text{real},\min}(i)|$ and $b_2 = |t_{\text{opt},\max}(i) - t_{\text{real},\max}(i)|$. If synchronization guarantees can be provided, there exists a B holding equation (1). In particular $b_1 \leq B$ and $b_2 \leq B$ hold.

The two grey triangles in Figure 2 denote points in time when the corresponding PU might be displayed. As the point in time marked by the third pale grey triangle is outside T_{real} , data will not be displayed even if it is within T_{opt} . We define a synchronization that guarantees for $T_{\text{real}} = T_{\text{opt}}$ to be *perfect*, i.e., synchronization that guarantees for $T_{\text{real}} \subset T_{\text{opt}}$ is not better than synchronization guaranteeing for $T_{\text{real}} \subseteq T_{\text{opt}}$.

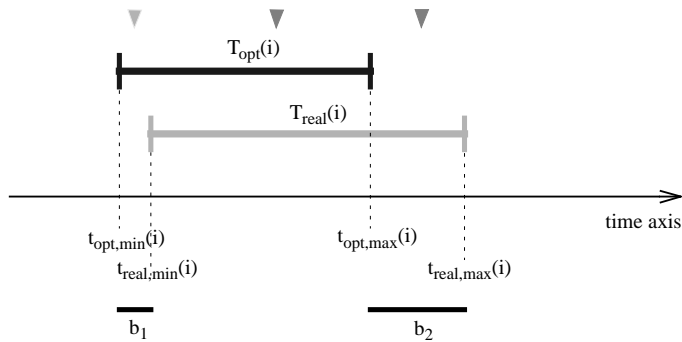


Fig. 2. Definition of Guaranteed Synchronization

Our definition of guaranteed synchronization does not require an asynchrony of zero. According to [7] asynchronies below a threshold cannot be perceived by humans. Therefore, we can allow for asynchronies and still guarantee synchronization as perceived by the user. As the values of perceptible asynchronies are media and application dependent [7], guaranteed synchronization is defined as synchronization guaranteeing for asynchrony bounds without specifying these bounds. The decision whether the bounds are acceptable for a specified application scenario depends on the media and the application and is taken in a second step. Therefore, synchronization for a specific application can be guaranteed, depending on the facts, whether guaranteed asynchrony bounds can be provided and whether these bounds are acceptable for the specific application.

Quality-of-Service (QoS) has been defined to describe the service required by a user or application or offered by a communication protocol. The task of a synchronization algorithm is the provision of synchronization, i.e., of a synchronization service. Therefore, we can apply the concept of QoS in order to describe synchronization algorithms. QoS characteristics of synchronization algorithms can then be applied to define a set of assessment parameters for those algorithms.

In [e] QoS is defined as “the set of those quantitative and qualitative characteristics of a distributed multimedia system necessary to achieve the required functionality of an application.” The concept of QoS involves multiple principles like the transparency principle, the integration principle, the separation principle, etc. [9] The integration principle requires that all architectural layers must allow for configuration, prediction and maintenance of QoS in order to meet end-to-end QoS. The provision of quality on an end-to-end level is necessary for high quality multimedia applications. Therefore, QoS parameters for synchronization must be maintained. The definition of guaranteed synchronization allows to define maintainable QoS values.

The basic idea of QoS of synchronization is depicted in Figure 3. The synchronization algorithm is understood as a part of a layer, i.e., as part of the application layer in

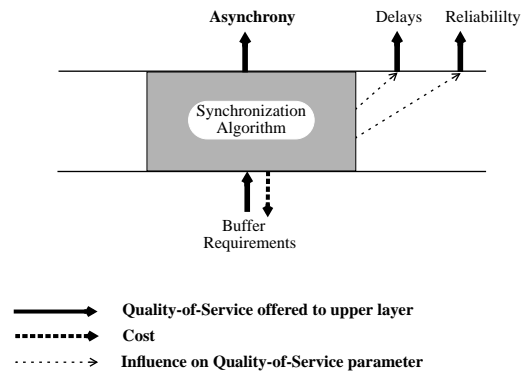


Fig. 3. Synchronization as Quality-of-Service

OSI terminology. It offers the provision of synchronization whose Quality-of-Service can be described in terms of asynchrony for intra-stream and inter-stream synchronization. In Figure 3 four QoS parameters are depicted. Asynchrony describes directly the QoS of synchronization, whereas the more general QoS parameters delays and reliability are influenced by the synchronization algorithm. Buffer requirements are required from the lower layer/the end-system and indicate costs of the synchronization algorithm. These four parameters are discussed in more in detail in the next section and form a set of assessment parameters for synchronization algorithms which will be applied in our assessment methodology equally introduced in the following section.

III. ASSESSMENT OF SYNCHRONIZATION ALGORITHMS

As defined in Section II, synchronization refers to the timely correct play-out of multimedia data. This incorporates that time has to be well defined between the systems being involved in a distributed multimedia application. For this reason, we require a notion of time that can be used for the definition of asynchrony. This time notion is described in this section. Then the set of assessment parameters is introduced and discussed in detail before the assessment of synchronization algorithms is described.

A. Design of a Time System

The term asynchrony refers to the deviation of the real presentation time from the optimal presentation time. Such deviations may have different reasons. One main reason lies in different local times in the involved end-systems: there do not exist two physical clocks that are identical [10]. For this reason different time systems are involved in a distributed multimedia application. A notion is required to describe these different time systems.

We first define a notion for distributed multimedia applications with one sender and one receiver, i.e., with a point-to-point association.

In the point-to-point case three different time systems are relevant which are depicted in Figure 4:

- The time system in the sender. All time points in the sender are denoted with t^s .

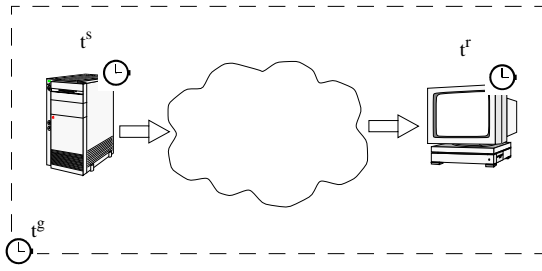


Fig. 4. The Different Time Systems

- The time system in the receiver. All time points in the receiver are denoted with t^r .
- The global time system. It serves as a reference time system and indicates the global, objective time¹. Time points in this time system are denoted with t^g . The global time system is introduced to allow for the definition of synchronized clocks.

The local time system of sender and receiver can be understood as the result of a mapping of the global reference time system. This mapping is done specifically for each computer system, as in the real world the local time systems of any two computers are not exactly the same. As time is monotonous in the real world, we can assume the mapping to be strictly monotonous as well. The assumption of a monotonous time mapping also in case of clock synchronization is especially reasonable as a non-monotonous time system may lead to severe problems in data bases and file systems. The mapping can be denoted as $M_r(t)$ in the receiver:

$$M_r(t) : t^g \rightarrow t^r \quad (2)$$

and $M_s(t)$ in the sender:

$$M_s(t) : t^g \rightarrow t^s \quad (3)$$

For every machine we can define the inverse time mapping that computes the global reference time for the local time. For the sender and receiver, this mapping can be denoted as $M_s^{-1}(t)$ and $M_r^{-1}(t)$. Figure 5 depicts the different mappings of these three time systems.

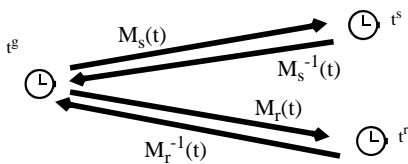


Fig. 5. Mapping Between the Different Time Systems

Let t_i^r and t_i^s denote any points in time in the local time systems of the receiver and the sender respectively. If both clocks of sender and receiver are perfectly synchronized, we determine, under the assumption of $t_i^r = t_i^s$:

$$\forall i : M_r^{-1}(t_i^r) = M_s^{-1}(t_i^s) \quad (4)$$

¹We assume the existence of such a time system whereas the knowledge of the global time is not required for analysis of algorithms.

or

$$\forall i : M_r(t_i^g) = M_s(t_i^g) \quad (5)$$

i.e., the time in both time systems is always identical.

Real-world clocks are, in general, not synchronized perfectly, so the time in the related two time systems is different. This difference involves two aspects: Firstly, the time indicated at one specific time point may have different values. If the difference is constant, we can easily map from one time system to the other. Secondly, the clocks may drift. Drifting clocks advance at different speeds. This *clock drift* may have a constant value which results in one clock that is advancing the other clock increasingly as time passes by. It may also vary over time. In general, clocks are drifting, but as they are regularly re-synchronized to a global time, the *time drift* is bounded by a minimum value and a maximum value. These bounds can be expressed as:

$$\forall i : b_{min} \leq |M_r^{-1}(t_i^r) - M_s^{-1}(t_i^s)| \leq b_{max}. \quad (6)$$

with $t_i^r = t_i^s$. Often b_{min} is assumed to equal 0.

To perform synchronization or any other task in the receiver being based on some time information specified by the sender, a mapping between the time system of the sender and the time system of the receiver has to be performed:

$$M_{s,r} : t^s \rightarrow t^r. \quad (7)$$

The easiest mapping is $M_{s,r}(t_i^s) = t_i^r$, i.e., the receiver assumes that the time in the sender corresponds to his time. The error of this mapping is bounded to the time drift b_{max} if the time drift itself is bounded.

Of course, there might be also an explicit mapping between the two time systems. In this case the mapping requires knowledge of the two clocks to perform better than a direct mapping. In case of synchronized clocks we can, in general, assume, that the time is directly mapped, i.e., the error of the mapping is bounded by the time drift. Even if a different mapping is performed, an upper bound for the error is specified by the time drift.

If multiple machines are involved, i.e., in case of concast or multicast, each sender and receiver has its own local time system. Mappings from one time system to the other are always specific for this pair of time systems. For this reason the time drift of all involved machines m_i can be defined as

$$\forall m_1, m_2 \quad \text{with} \quad t_i^{m_1} = t_i^{m_2} \quad \forall i : \\ b_{min} \leq |M_{m_1}^{-1}(t_i^{m_1}) - M_{m_2}^{-1}(t_i^{m_2})| \leq b_{max} \quad (8)$$

B. Assessment Parameters

In order to assess synchronization algorithms, we need to define the relevant parameters. The four QoS parameters for synchronization asynchronies, delays, buffer requirements, and reliability (synchronization errors) are base of the assessment methodology presented in this paper.

B.1 Asynchrony

The asynchrony measures the allowed deviation of the time interval between two subsequent presentations of PUs

from the time interval in case of optimal presentation of these two PUs.² The time points given in the local time systems of sender and receiver are mapped to the global time by the inverse mapping function.

Definition 3 Let $t_{r,i}^s$ denote the reference time of PU i in the sender's time system, and $t_{p,i}^r$ denote the presentation time of PU i in the time system of the receiver. Δ_{ia} is the *maximum asynchrony in case of intra-stream synchronization*, if

$$\begin{aligned} \forall i & : \max(0, M_s^{-1}(t_{r,i+1}^s) - M_s^{-1}(t_{r,i}^s) - \Delta_{ia}) \\ & \leq M_r^{-1}(t_{p,i+1}^r) - M_r^{-1}(t_{p,i}^r) \\ & \leq M_s^{-1}(t_{r,i+1}^s) - M_s^{-1}(t_{r,i}^s) + \Delta_{ia}. \end{aligned} \quad (9)$$

■

The term $\max(0, M_s^{-1}(t_{r,i+1}^s) - M_s^{-1}(t_{r,i}^s) - \Delta_{ia})$ avoids that a large term Δ_{ia} allows for the violation of the ordering aspect of intra-stream synchronization.

The intra-stream asynchrony is depicted in figure 6. The optimal presentation interval is depicted as the black, thick line. The minimum interval is depicted in the dark grey color and the maximum interval in the light grey color. The intra-stream asynchrony is defined as the maximum deviation of the play-out interval from the interval in case of optimal presentation of the PUs. These deviations are depicted by a black line and indicated by a Δ_{ia} within figure 6.

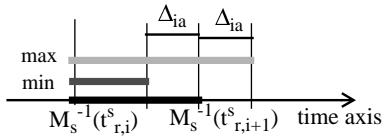


Fig. 6. Intra-stream Asynchrony

The following definition of inter-stream asynchrony is a general one, as it allows for any two streams to be sent from two different senders.

Definition 4 Let $t_{r,(k,i)}^{s_1}$ denote the reference time of PU i in flow k in sender s_1 and $t_{p,(k,i)}^r$ denote the presentation time of PU i in flow k in the receiver. Δ_{ie} is the *maximum asynchrony in case of inter-stream synchronization* for the two flows k and l , if

$$\begin{aligned} \forall i & : M_{s_1}^{-1}(t_{r,(k,i)}^{s_1}) - M_{s_2}^{-1}(t_{r,(l,i)}^{s_2}) - \Delta_{ie} \\ & \leq M_r^{-1}(t_{p,(k,i)}^r) - M_r^{-1}(t_{p,(l,i)}^r) \\ & \leq M_{s_1}^{-1}(t_{r,(k,i)}^{s_1}) - M_{s_2}^{-1}(t_{r,(l,i)}^{s_2}) + \Delta_{ie}. \end{aligned} \quad (10)$$

■

In interactive multi-party applications we can observe a third kind of asynchrony, the intra-group asynchrony. The

²Note that a time interval can be of length zero to cover inter-stream asynchrony.

idea of intra-group asynchrony can be illustrated by the scenario of a video-conference. Given one person A that tells a joke, the participants in the conference, e.g., B and C, only hear the end of the joke after a delay has elapsed that is specific for the synchronization scheme as well as for the specific transmission link between the parties. Thus, each of the participants B and C reacts to the joke with a specific delay. This delay elapses once more until A sees the reactions of his participants to the joke that has been told. As the delays between A and B on one hand and between A and C on the other hand may differ considerably from each other, it is possible that A has problems in relating the observed reactions to each other and to the end of the joke he told. The time deviation in the reactions as seen by A, even if B and C react instantaneously after the end of the joke been displayed locally, is termed *intra-group asynchrony*. This intra-group asynchrony may be especially disturbing for the problem of turn-taking. Turn-taking refers to the action of agreeing upon the next speaker in a discussion. Normally this problem is solved by non-verbal means. These means are only restrictedly available in video-conferences where the problem of turn-taking has been observed and reported. [11]

In the literature the maximum asynchrony of intra-stream synchronization is often referred to as jitter, whereas the maximum asynchrony of inter-stream synchronization is often referred to a skew. [7] refers to skew as the essential QoS parameter for synchronization and discusses user requirements. As shown in [7] low asynchronies are essential for high user-perceptible quality and, therefore, also for the assessment of synchronization algorithms.

B.2 Synchronization Delays

In conferencing applications like video conferences, the *end-to-end delay* is another critical QoS parameter. In applications like Video on Demand that are interactive and based on stored data the response time depends on the delay of the control command as well as on processing times in senders. Response times to commands should generally be kept low. Delay, therefore, has been identified as an important criterion for multimedia applications.

Synchronization algorithms in general increase the delay of PUs in order to smooth out transmission jitter in the receiver. Therefore, the user-perceptible delay is composed of two parts. The first part is the end-to-end delay determined by the network and end-system also including processing times of, e.g., commands. The second part of the delay is the synchronization delay used to smooth out jitter. This part of the delay depends on the specific synchronization algorithm applied and is, therefore, an assessment parameter for synchronization algorithms.

B.3 Buffer Requirements

In distributed multimedia applications transmission jitters must be smoothed out in order to provide multimedia synchronization. Smoothing out jitter requires buffer in the receiver. The *buffer requirements* are an important characteristic of synchronization algorithms as synchronization

algorithms can completely smooth out jitter or adapt the end-to-end delay of PUs to the current transmission delay. Thus, the buffer requirements describe the buffers available to the synchronization algorithm to smooth out delay jitter. If the effective jitter is higher than the one smoothed out by buffers, the algorithm might choose to increase the delay or synchronization errors occur. Different classes of synchronization schemes like rigid and adaptive ones imply different buffer requirements.

B.4 Synchronization Errors (Reliability)

Synchronization guarantees the play-out of PUs within their valid time intervals. The valid time intervals in the receiver are defined by the media schedule as well as by the synchronization algorithm determining the end-to-end delay of PUs. For this reason, the number of PUs that can not be displayed within their valid time interval is a further characteristic of synchronization schemes. With *synchronization error* we refer to the rate of PUs that cannot be displayed in time. In [12] different kinds of synchronization errors and their reasons are discussed.

It is not easy to provide a general definition of a synchronization error for multiple streams that have to be synchronized as there exist multiple possibilities. The interpretation of the synchronization error depends on the aim of the analysis and the error must be clearly defined in every analysis. Some of these possible definitions of synchronization errors for multiple streams are:

- Following the master/slave concept: a synchronization error is given, if the master PU cannot be displayed in time.
- Following the master/slave concept: a synchronization error is given, if the master or more than a specified percentage of the slave PUs cannot be presented in time.
- A synchronization error is present, if any of the PUs cannot be presented in time.
- A synchronization error is given, if a specified percentage or more of the PUs cannot be displayed in time.

B.5 Selecting Four Assessment Parameters

The defined set of assessment parameters describes four assessment parameters that are not orthogonal. Delays, buffer requirements and synchronization errors are highly interdependent as are also asynchronies and buffer requirements. We propose, nevertheless, to include all four parameters into the set of assessment parameters, as we believe, that it is important to provide a set of meaningful parameters to people who want to take a decision in favor or disfavor of a given synchronization algorithm based on parameters. We have identified the four QoS parameters asynchrony, buffer requirements, delays and synchronization errors to be meaningful and relevant for such decisions.

Somebody wanting to apply a parameter based assessment in order to decide for or against algorithms is interested in a set of parameter values that shows the advantages and disadvantages of the algorithms as well as the trade-offs between, e.g., buffer requirements and asynchronies. These trade-offs can be studied during an anal-

ysis and formulae can be derived. (This happens in analogy to the analyses described in [12], [13].) Selecting only one or two parameters and deriving mathematical formulae describing the mapping to other parameters will decrease usability. These formulae are different for each algorithm as they depend directly on the algorithm's characteristic. Therefore, users are required to either compare the formulae and their impact on interesting parameters or to calculate remaining parameter values based on these formulae. For this reason we apply the set of all four interdependent parameters as base for assessment of synchronization algorithms.

The following example illustrates the relevance of the four assessment parameters introduced above:

We consider a case with a given synchronization mechanism based on global clocks and a known delay distribution for the communication as well as an error-free, loss-free data transmission over the network. In scenario A the receiver displays all data exactly at $t_{s,i} + \Delta_{\max}$, (Δ_{\max} known maximum delay). In scenario B the receiver displays data at $t_{s,i} + \Delta_{\min}$ (Δ_{\min} known minimum delay) and drops all PUs arriving too late. In both cases the achievable asynchrony is 0, as all displayed PUs are displayed in time. Scenarios A and B describe extreme situations, but table I indicates that one single synchronization quality parameter alone cannot serve as a valid indicator to describe the overall quality. In scenario C all data is displayed directly after arrival. Comparing the parameter values in table I we see that under the given conditions all three scenario might produce synchronization well enough to the user and that the base for a decision between A, B, or C is an optimization problem on all four parameters and requires the definition of an utility function specific to the application.

TABLE I
ASSESSMENT PARAMETER VALUES FOR SAMPLE SCENARIOS

	Asynchrony	Delay	Buffer	Errors
A	0	max	max	0
B	0	min	min	max
C	max	min ... max	min	0

C. Assessment of Synchronization Algorithms

The goal of analyzing synchronization algorithms with respect to a set of assessment parameters is the description of characteristics being inherent to the algorithms and possibly being the base for a decision in favor or disfavor of a specific algorithm. In order to do so, we distinguish five different components being involved in or influencing synchronization. These components are depicted in figure 7 and consist of: (1) User, (2) Application, (3) Synchronization algorithm, (4) End-system, and (5) Communication System or Network.

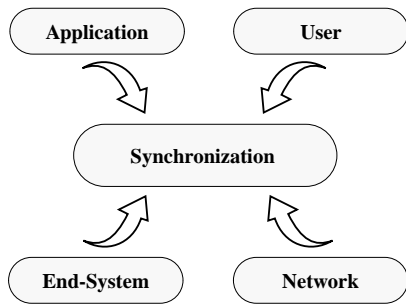


Fig. 7. Components of the Synchronization Analysis

Applications offer *data*, where at least a part of them is time dependent and, thus, requires mechanisms to ensure a synchronized play-out. Data shows characteristics being relevant to synchronization, e.g., the size of one presentation unit, the play-out time of one presentation unit, and the periodicity. The user of an application has *requirements* with respect to the entire application as well as to the synchronization itself. Possible requirements for synchronization, as the maximum accepted asynchronies, may be found in [7]. Application requirements that are relevant to the synchronization algorithm include, e.g., the data format and cost requirements in form of buffer requirements.

The *end-system* has characteristics influencing the synchronization. One important characteristic is the local time system of the receiver and how well it is synchronized to the local time system of the sender. Another important criterion is an end-system's support of real-time requests. The synchronization algorithm initiates the synchronous play-out of data. Synchronization cannot be guaranteed, if the end-system does not play out the data instantaneously or with a specified delay that has to be taken into account by the synchronization mechanism. The *communication system* is specified by the characteristics of the underlying *network*. Data that is to be played out, but has not (yet) reached the end-system, leads to a synchronization error. The delay as well as the packet loss rate of the underlying network plays an important role for play-out quality after synchronization.

To be assessed or classified synchronization algorithms have to be analyzed with respect to the assessment parameters. After synchronization algorithms have been analyzed, they can either be assessed for a specific environment or classified. A sample assessment is based on the values of the four assessment parameters and will be described later in this report. The assessment scheme is depicted in Figure 8. First, a synchronization algorithm is analyzed obtaining analysis results (A). These results consists of expressions of variables describing end-system, network and data characteristics. In order to evaluate these expressions, variable values for the concrete, given situation have to be specified. These values, the concrete parameters (B), are filled into the analysis results (evaluation). The resulting assessment parameter values are compared with specified user requirements in order to assess whether the synchronization algorithm is suitable for providing the required

synchronization in the given situation (C).

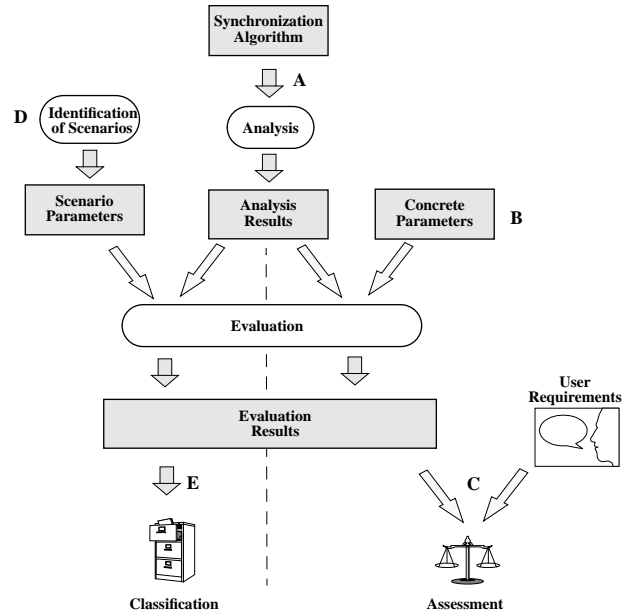


Fig. 8. Assessment of Synchronization Algorithms

A similar process can be applied in order to classify the numerous synchronization algorithms. Typical real-world scenarios have to be identified and parameter values describing these scenarios have to be specified (D). These parameter values have to be filled in the variables of the expressions resulting from the analysis. Based on the assessment parameter values resulting from this process classifications can be undertaken that allow to find algorithms best supporting classes of application scenarios like video-conferences in local environments with guaranteed maximum delays (E).

Both, classification and assessment of synchronization algorithm are based on two different steps:

1. the *analysis* of synchronization algorithms. During this process relevant characteristics of data, environment, and network are parameterized and included in the analysis results.

2. the *evaluation* of synchronization algorithms. During this process concrete values are assigned to the assessment parameters. This assignment is specific for characteristics of data, environment, and network.

Our assessment methodology requires assessment parameters that can be analyzed and evaluated independently from any specific environment and/or implementation. The assessment is of general nature and can determine basic recommendations of situations favorable for the algorithms. In order to decide for or against *specific* algorithms for synchronization support in *specific* applications and environments, other factors than our four assessment parameters have to be equally taken into account.

Such factor are, of course, the costs of the specific algorithm in terms of, e.g., CPU utilization. In contrast to the four defined assessment parameters, CPU utilization

cannot be analyzed mathematically without concrete assumptions on environment and machines. Therefore, CPU utilization has only been discussed on a very abstract level in our work, based on pseudo-code in order to conclude which algorithm requires more CPU [12].

A second factor being dependent on the algorithm as well as on its implementation is the robustness of an algorithm and its ability to recover from errors. There are synchronization algorithms that allow for the display of more PUs than others in case of large delay jitters. These algorithms are therefore very robust with respect to delay jitters. This robustness is expressed within the assessment parameter synchronization errors. Robustness and recovery after erroneous situations depend directly from the current implementation of the algorithms. Therefore, these parameters are not independent from environment and implementation and do not fit into our assessment methodology.

Qualitative factors have to be equally taken into account. These factors can be summarized by the *environmental* and *application oriented requirements* of synchronization algorithms. Synchronization algorithms that have been developed and published are written for a specific application and environment. Different computing environments have different characteristics, e.g., with respect to the architecture that is required. There exist algorithms that require a central synchronization controller [14] or group agent [15] or that can perform synchronization without a central unit like [16]. The architecture and the system, being a communication middleware, a multimedia application program or an application framework, can facilitate algorithms requiring a specific synchronization architecture or even prevent algorithms from being implemented within the given environment. Other environmental factors are the requirements with respect to global time. There exist synchronization algorithms that require global time [17] whereas others can work pretty well without a specified global time [18], [19], [20]. Assumptions on the transmission link also need to be taken into account. Whereas, e.g., Concord [16] assumes a well-known maximum delay, ASP [14] can equally work for transmission links that do not provide bounded delays.

Application oriented requirements include all characteristics of data and synchronization specification that can be relevant for synchronization. Such characteristics include the use of time stamps, PU numbers or an explicit media schedule, the periodicity of data and the relationships between data streams (master/slave concept versus equally important streams). In order to decide pro or contra a given synchronization algorithm for a given synchronization task in a given environment, all these factors have to be taken into account.

By compiling a list of environmental parameters that have been identified during analysis of synchronization algorithms, algorithms can be characterized based on influence factors. In combination with qualitative parameters the algorithms depend on, a template can be created allowing for the qualitative characterization and comparison of algorithms.

IV. APPLICATION OF THE ASSESSMENT METHODOLOGY

A. Analysis

For analysis purposes, we assume that PUs are sent according to the media schedule. The accepted delay of a synchronization method determines PUs that have to be stored due to early arrival as well as PUs that have to be thrown away due to late arrival. This delay influences the probability of a presentation gap and, thus, the synchronization error probability.

We assume for analysis purpose that content of PUs is *independent*, i.e., that the PUs can be processed independently. Video compression methods such as MPEG [21] achieve compression by only coding the differences between given video frames or by using a frame prediction based on, e.g., motion vectors and coding the difference between predicted and real frames, only. These schemes introduce dependencies between different coded PUs to reduce the amount of data. After decompression the PUs are again independent from each other. There are different possibilities to fulfill our assumption of independent PUs:

1. Compression schemes can be used that do not create dependencies between video frames, e.g., Motion JPEG [22]. In general, these schemes have lower compression rates and are less performant.
2. The synchronization relevant tasks are performed before and/or after the data is compressed/decompressed. This results in higher buffer requirements for PUs that have to be stored in a play-out buffer.
3. Independent PUs can be modeled by adapting the PU loss rate of the transmission link. If one PU gets loss, it must be taken into account that other PUs might not be correctly derived due to this loss. Therefore, the PU loss rate has to be increased.

Characteristics of the underlying network and end-systems are parameterized during the analysis. Therefore, the analysis results remain independent from those characteristics. Data characteristics in terms of parameters are included into the analysis as well. The results of the analysis are expressions of parameters describing the environment³. These expressions are evaluated by replacing the parameters with concrete values for end-system and network characteristics. This evaluation results in the synchronization quality characteristics of the analyzed algorithm within the specified environment. In addition, from the analysis a list of parameters describing characteristics of the environment and influencing the synchronization quality can be derived. These results allow for a deeper, analytical understanding of the synchronization service provided by the studied algorithm as well as for an analytical assessment of the algorithm by evaluating parameter expressions.

For the analyses carried out, two synchronization methods belonging to two different classes of synchronization have been chosen. The *Concord* algorithm [16] belongs to the class of rigid synchronization algorithms, whereas the

³In this context environment refers to the environment of the synchronization algorithm, i.e., the communication system, end-system, data, and application.

Adaptive Synchronization Protocol (ASP) [14] defines an adaptive synchronization algorithm. We analyze the algorithms aiming to obtain synchronization characteristics that can be guaranteed by these algorithms. For this reason, the analyses have to take into account the worst case.

The analyses have been carried out in detail for the unicast case and periodic live data or data that is sent according to a media schedule, respectively. The assessment parameters are visible in the receiver, the asynchronies, delays and synchronization errors are visible to the user and buffers have to be provided in the receiver. The analysis results can be transferred directly to multi-party applications. The receiver specific assessment parameter values can be obtained based on the characteristics of the specific associations between receiver and senders. For synchronization groups the delays depend on the maximum delay of all flows. Special focus has been put on asynchrony, whereas synchronization errors and partly buffer requirements and delays depend to a high degree on the communication system's characteristics. The transmission delay is analyzed, but the start-up delay is not considered, as this occurs only once during the beginning of a presentation and does not have a great impact on the quality.⁴ An excerpt of analysis results is presented in [13] and in [23] together with a description of the assessment methodology.

The synchronization errors *loss of synchronization* may occur due to synchronization information being modified by an unperceived bit error, errors in the synchronization function, or additional delays in the end-system. If the synchronization method is implemented correctly, these errors only occur due to problems in the end-system or the network and are, therefore, not included in the analysis. For the same reason, an incorrect time mapping resulting in loss of a common time base of synchronization errors is not included in the analysis of synchronization quality. The parameter *synchronization error probability* is, thus, for analysis purposes restricted to the case in which a PU is not presented.

To present a complete application of the assessment methodology, we present central resulting expressions of the performed analyses from [13] while the detailed results can be found in [12].

A.1 Analyzing the Concord Algorithm

The *Concord algorithm* [16] is an easy-to-understand and easy-to-implement algorithm as it achieves synchronization by guaranteeing for a constant delay of packets for each data stream. As data is sent periodically, synchronization is simplified. Unknown network delays only play a role with respect to the first data packet.

The algorithm computes the delay each packet has to suffer and operates a synchronization buffer that is implemented as a shift register. A *total end-to-end delay* (Δ) is calculated, which is constant for every packet. This delay takes into account a specified maximum acceptable delay

⁴This may be different, if schemes for stored data are analyzed that allow for in advance transmission of data to minimize synchronization errors.

and a specified maximum packet loss rate. Three policies can be chosen to set the Δ of packets: to minimize the packet loss rate, to minimize Δ , or to minimize some sort of hybrid costs.

[16] assumes that packets are sent periodically with the period T_r which is specified in the media schedule. For analysis purposes, it is assumed that one packet refers to one PU. When a PU arrives, it is placed in the synchronization buffer to delay its play-out until the PU suffered the specified total end-to-end delay. If the delay of the first packet is not known, the total end-to-end delay cannot be calculated exactly and, depending on the error of the network delay, several packets need to be dropped. In case of different rates in the sender and the receiver, which result from clocks tuning at different speeds, PUs may need to be dropped or the play-out of PUs may need to be delayed.

Following parameters are required for the analysis:

λ	error in estimating the network delay of first packet
ϵ	bound on the clock asynchrony
Δ_{min}	minimum network delay
Δ_{opt}	total end-to-end delay specified by the algorithm
T_r	play-out period of the receiver (local time system)
τ_r	play-out period of the receiver (global time system)
τ_s	sending period of the sender (global time system)

In case of the basic Concord algorithm (cf. Table II), synchronization can be retained by the algorithm and the total end-to-end delay is specified by Δ_{opt} which is optimal. The synchronization error is specified by the packet loss rate that was handed in by the user to determine Δ_{opt} . Buffer requirements per stream are defined by a buffer for $\left\lceil \frac{\Delta_{opt} - \Delta_{min}}{T_r} \right\rceil$ presentation units (for the basic algorithm).

TABLE II
CHARACTERISTICS OF CONCORD FOR ONE STREAM

asynchrony	0
delay	Δ_{opt} (depends on the delay distribution of the network)
synchronization error	specified packet loss rate
buffer requirements	$\frac{\Delta_{opt} - \Delta_{min}}{T_r}$ packets

In case the network delay of the first packet is not known, the total end-to-end delay is increased by the uncertainty in assuming the delay of the first packet. If this uncertainty is less than the play-out rate, other parameters are not influenced. In case the uncertainty is higher than the play-out rate, a buffer overflow may occur and in total $\left\lceil \frac{\lambda}{T_r} \right\rceil$ additional packets need to be thrown away, resulting in a momentary asynchrony. After these packets have been discarded, an asynchrony of 0 can be guaranteed by the algorithm.

Drifting clocks lead to an additional asynchrony of the clock drift's bound. They increase buffer requirements by

the number of PUs that may be displayed during the time of the clock drift.

Different sending and play-out rates ($\tau_r \neq \tau_s$) require that every $\frac{\tau_r}{|\tau_r - \tau_s|}$ -th packet has to be discarded or duplicated which adds to the asynchrony. In case of a faster receiver, the delay is slightly increased, but buffer requirements are not influenced.

In an environment that has, e.g., unknown network delay and different production and service rates, values for QoS parameters have to be combined to assess the algorithm for this specific environment.

The asynchrony of a synchronization group results in the maximum of all stream asynchronies plus ϵ in case of bounded clocks. The delay is specified to be the maximum delay of streams belonging to this synchronization group. The buffer requirement for the whole group is simply the sum of buffer requirements of all streams. Hereby it must be taken into account that buffer requirements per stream must be calculated with respect to the delay of the synchronization group. The buffer requirement for one isolated stream may be smaller, if its maximum network delay is smaller than the total end-to-end delay for the group.

A.2 Analyzing ASP

The *Adaptive Synchronization Protocol (ASP)* [14] is an adaptive synchronization algorithm with centralized control. It monitors the underlying network and adapts to changing network conditions. The control mechanism of this protocol is based on the buffer level. One synchronization group consists of one controller for the whole group and one agent per stream. An agent is a software entity controlling an individual stream. The packet sequence must not be changed by the underlying network.

ASP distinguished between one master stream and slave streams. A master stream defines a stream, where slave streams are synchronized to.

ASP is based on stream rates, i.e., data is periodic. The controller sends a start message to source agents, and sink agents start the play-out of data after an initial delay of Δ . Δ is the maximum delay of all data transmissions plus the delay that is needed to fill the buffers to a minimum level.

ASP consists of four different protocols. The goal of the algorithm is to minimize the end-to-end delay or to minimize the data loss, respectively, depending on the synchronization policy chosen and the actual network state. To achieve this goal, a buffer area is defined in each play-out buffer. The buffer fill must be kept within this buffer area. Buffer regions below or above this area are critical. In the master stream a second buffer area is defined, a target buffer area. This area is within the buffer area of the master. The stream agent of the master tries to keep the buffer fill within this target area and adapts the play-out rate while the sending rate remains unchanged, whenever the buffer fill is outside the target area. The adaption is performed by sending an *Adapt* message to slave agents. The message contains the end time for the adaption phase as well as the media time that has to be valid at the end time. Master and slaves adapt their play-out rate, thus,

that the specified media time is valid at the end of the adaption phase. In case a slave buffer fill enters a critical buffer region, the slave triggers an adaption phase and becomes a tentative master.

There are two different kinds of adaption phases. The goal of a *slowing down adaption* phase is to slow down the play-out rate for a given time interval to increase the buffer fill. If the buffer fill has to be decreased, the *speeding up adaption* increases the play-out rate during the adaption phase.

Following parameters are required in the analysis:

Δ_{min}	minimum network delay in the master stream ^a
Δ_{max}	maximum network delay in the master stream ^a
c	width of the target buffer area (1 in case of a tentative master)
j_M	accepted jitter of the master stream
j_S	accepted jitter of the slave stream
L_M	length of the adaption phase as defined by the master
l	time to rest in the slave stream for carrying out an adaption phase
$T_{r,M}$	play-out period of the master stream (in the local time system)
$T_{r,S}$	play-out period of the slave stream (in the local time system)
ϵ	bound for the clock asynchrony

^aDuring a time interval.

If no adaption has to be performed, the intra-stream asynchrony achieved by ASP equals 0 or ϵ in case of bounded clocks. This is a similar result as for the rigid Concord algorithm. But the delay of the stream is smaller than the delay for the Concord algorithm. It is determined by $\Delta_{min} + \frac{1}{2} \cdot c \cdot j_M$ or $\Delta_{max} - \frac{1}{2} \cdot c \cdot j_M$, and it is smaller than the maximum total end-to-end delay, that can guarantee for a given data loss probability as, e.g., the delay in the Concord algorithm. Δ_{min} and Δ_{max} are not fixed, but can vary over time to adapt to the current network delay. The delay in all streams is determined by the delay of the master stream. The delay during an adaption phase and its changes depend highly on the current situation in which the adaption phase takes place.

The asynchrony can be calculated for different adaption phases. The value of the asynchrony depends on the length of the adaption phase in the master and also on the length of the adaption phase in the slave which may be shorter due to propagation delays of the *Adapt* message in the network. This can lead to a significantly higher asynchrony. The asynchrony also depends on the width of the target buffer or the buffer region in a tentative master. Tables III and IV depict asynchrony and delay parameter values for master and slave streams.

The buffer fill smoothing function also influences the adaption phase, as this function determines, when an adaption phase is entered and how aggressive the stream agent reacts to delay differences in the network. The synchronization error probability depends on the buffer width in the stream, the above mentioned smoothing function, and the number of adaption phases, the delay of the network

TABLE III
CHARACTERISTICS OF ASP FOR MASTER STREAM

	Minimum Delay Policy	Minimum Loss Policy
no adaption in progress		
asynchrony	0	
delay	$\Delta_{min} + \frac{1}{2} \cdot c \cdot j_M$	$\Delta_{max} - \frac{1}{2} \cdot c \cdot j_M$
slowing down adaption		
asynchrony	$\frac{\frac{1}{2} \cdot c \cdot j_M}{L_M}$	
delay	increasing	
speeding up adaption		
asynchrony	$T_{r,M} - \frac{T_{r,M} \cdot L_M}{\frac{\frac{1}{2} \cdot c \cdot j_M}{T_{r,M}} + L_M}$	
delay	decreasing	

and the bit rate. If the adaption phase is not delayed until all slave agents received the *Adapt* message, the synchronization error probability is increased.

TABLE IV
CHARACTERISTICS OF ASP FOR ONE SLAVE STREAM

	Minimum Delay Policy	Minimum Loss Policy
no adaption in progress		
asynchrony	0	
delay	$\Delta_{min} + \frac{1}{2} \cdot c \cdot j_M$	$\Delta_{max} - \frac{1}{2} \cdot c \cdot j_M$
slowing down adaption		
asynchrony	$\frac{\frac{1}{2} \cdot c \cdot j_M}{\frac{l}{T_{r,S}}}$	
delay	increasing	
speeding up adaption		
asynchrony	$T_{r,S} - \frac{T_{r,S} \cdot \frac{l}{T_{r,S}}}{\frac{\frac{1}{2} \cdot c \cdot j_M}{T_{r,S}} + \frac{l}{T_{r,S}}}$	
delay	decreasing	

Concerning buffer requirements, $\left\lceil \frac{j_M}{T_{r,M}} \right\rceil$ and $\left\lceil \frac{j_S}{T_{r,S}} \right\rceil$ denote the buffer size in PUs for the buffer region in the master and the slave. Real buffer requirements are higher, as there exists also a critical buffer region below and above the buffer region which ASP takes into account.

The inter-stream asynchrony provided by ASP equals 0 or ϵ , in case of bounded clocks, if there is no adaption phase in progress. The inter-stream asynchrony in adaption phases is limited to $\frac{1}{2} \cdot c \cdot j_M$ which describes the value by which the media time has to be corrected during the adaption phase. This asynchrony may occur in case that one slave agent receives the *Adapt* message after or at the end of the adaption phase. If the adaption phase is delayed until all streams receive the *Adapt* message, the inter-stream asynchrony remains 0. In case of master switching, when a master and a tentative master may send contradicting *Adapt* messages, this inter-stream

asynchrony is determined by the sum of the value by which both streams want to change the media time. The delay of the synchronization group is specified by the delay of the master stream. As for the Concord algorithm, buffer requirements for single streams are added to obtain buffer requirements for the whole synchronization group.

A.3 Discussion of Analysis Results

The results of the analysis of the *Concord* algorithm concur with the values for buffer size and delay as discussed in [16], validating our results. As asynchrony has been partly discussed in [16], parts of our results concerning asynchrony as well as the discussion on synchronization errors provided in [24] and mentioned above determine new results. These results corroborated our assumptions on the quality and performance of the algorithm. Based on the simulation in the CINEMA environment [25], more detailed discussions on buffer requirements and stability issues of ASP are included in [26] and [14]. In contrast, the provided analytical results above, especially concern asynchrony. The obtained results illuminate the concrete effect of adaptation phases and illustrate that it is possible to remain largely within the asynchrony limits defined in [7] even during adaption phases. This was shown by simulation in [26] and [14].

Based on the analysis, a list of influencing factors for the QoS of synchronization for both algorithms has been derived. Besides providing more insight in the functioning of the algorithm, these results can be incorporated into templates characterizing qualitatively synchronization algorithms and serving for comparison of synchronization algorithms. The influencing factors as well as analysis results are specifically helpful to tune environmental parameters to increase user-perceptible quality of distributed multimedia applications.

Characteristics of the Concord algorithm have been directly derived from the description of the algorithm. In case of ASP this was not straight-forward, as there are many interdependent and changing parameters, influencing the behavior of the algorithm. Analysis is a time consuming task, but offers also a lot of insight to the algorithm and can be base of assessment, and qualitative characterization of algorithms as well as of decisions on how to improve the QoS parameter values of synchronization by changing the environment (network and end-system characteristics, e.g.).

B. Evaluation

B.1 Evaluation Method

As already discussed in this paper parameter expressions resulting from analyses have to be evaluated by using concrete parameter values that describe the environment like delays and error rates.

Evaluations of synchronization algorithms may have different goals:

- based on the evaluation one or more algorithms that are best/reasonably well suited to solve a given synchronization problem are to be chosen. Thus, the goal of the evaluation consists in a *comparison* of different algorithms for

a given problem. In this case the algorithms should consequently be evaluated with parameters describing the given synchronization problem.

- based on the evaluation a *classification* of several algorithms with respect to some specified synchronization problems may be obtained. Thus, the evaluation should be based on parameters that describe best the different synchronization problems used as base for the classification scheme.
- based on the evaluation *conclusions* on the behavior and the advantages of the algorithms shall be drawn as base for an assessment. To do so we need to cover a parameter space that allows to see the different behaviors of the algorithms. The goal of the evaluation presented lies in conclusions of this kind.

For the evaluations we need, in consequence, parameter values that cover a wide parameter space. To obtain parameter values used for evaluation, three different methods can be applied:

1. *real-world measurements*
2. *simulation measurements*
3. *theoretically constructed scenarios*

These ways of obtaining parameter values are evaluated based on their ability to provide parameters that cover the required parameter space:

1. *real-world measurements*: If real-world measurements are used to obtain the parameter values that are the basis for the evaluation, we require either to measure all possible environments in order to cover the parameter space or to choose some environments that can span the parameter space. Whereas the first possibility results in too many environments that are to be measured as to be realistic, the second possibility requires specific, in advance knowledge of the parameter space that is relevant for the evaluated algorithms as well as of the environments that can span the required parameter space.
2. *simulation measurements*: The same problem holds for measurements that are obtained during simulations. Either all possible cases have to be simulated to allow for spanning of the parameter space or theoretical knowledge of the algorithms is required to correctly choose simulations for measurement.
3. *theoretically constructed scenarios*: The same theoretical knowledge of the behavior of the algorithms as required for above methods can be used to directly construct different scenarios and to use the parameters describing these scenarios as base for the evaluation. The use of theoretically constructed scenarios allows for the exploration of the parameter space while using a data set of a reasonable size.

A theoretical in advance knowledge of some characteristics of the algorithms is required in all three cases. This knowledge can be acquired during the process of analysis of algorithms. We have chosen to base the evaluation presented here on theoretically constructed scenarios.

Our goal is the evaluation of two different synchronization algorithms, Concord and ASP. Whereas Concord is a rigid synchronization method and base on a constant delay for all data units, ASP adapts the end-to-end delay of data

units to the current network delay.

For this reason, the algorithms should behave differently when the transmission delay varies. Therefore, the evaluation of the analyzed algorithms is based on the following basic scenarios:

1. constant known delay
2. variable delay with small jitter
3. variable delay with large jitter but slowly changing delay
4. variable delay with large jitter and fast changing delay

Based on the knowledge of the algorithms that has been gained during analysis, we suppose that in the first case which is based on a constant delay, the delay of the rigid and the adaptive algorithms is the same and that the differences in other fields become clear. In case of a small delay variance, the delay gains of the adaptive algorithm will probably be not very important compared to the rigid algorithms. This will probably be different in the third case where the delay variances are quite high. The fact that the delay changes vary slowly should make this case a preferred scenario for the adaptive algorithm, thus showing the advantages over the rigid ones. The abrupt delay changes in case 4, supporting a high delay variance, might lead to unsteady behavior of the adaptive algorithm which requires numerous adaptations.

Due to the fact that Concord as well as ASP are only suited for the case of periodic data, the evaluation will not take into account non periodic data.

As the parameter space to cover is wide, we exclude bit errors leading to synchronization errors in the presented evaluation. Standard video compression techniques like JPEG [27] and MPEG [21] reduce the probability of losing a whole image of a video sequence in cable networks to almost zero. Even for mobile environments with their very elevated error rates in data transmission compression schemes exist, that rarely cannot present a video image at all (e.g., WaveVideo [28]). As disturbances in audio presentation are remarked very easily and as the number of audio data that is to be transmitted generally is not so high compared to video, schemes for FEC as well as data replication should be applied in order to prevent data loss.

In table V we define four basic evaluation scenarios A to D based on the different delays.

TABLE V
THE FOUR BASIC EVALUATION SCENARIOS

A	the delay is known and constant
B	the delay varies. The jitter is small
C	the delay varies. The jitter is large, but delay changes slowly (we use the TES [29] method to compute the delays, and define the interval length for TES as 10 % of the jitter and to be equally around the last delay)
D	the delay varies. The jitter is large and delay changes may occur abruptly

For these basic scenarios we have defined different evaluation scenarios based on delay/jitter values.

In [12] the detailed evaluation of all four evaluation scenarios is described.

First, scenario A has been evaluated. As the delay remains constant the scenario's parameter values are inserted in the formulae derived from the analyses to directly obtain evaluation results.

As the transmission delays are constant in scenario A, this allows for the study of the effect of drifting production and service rates within scenario A. Effects that can be observed within this study are exclusively related to the drifting rates. For this study Mathematica™ simulations for both, Concord and ASP, have been carried out. In case of Concord the effects have been studied for a faster and a slower receiver. In case of ASP a choice of ASP internal parameters, e.g., the length of the adaption phase, is required for simulations and the calculation of asynchronies during adaption phases. In order to derive results that are valid for ASP independently from the choice of ASP internal parameters, the length of the adaption phases as well as the target buffer area have been varied during the evaluation process.

The evaluation of scenario B has first been done based on the insertion of scenario parameter values in the analysis results. As equal production and service rates are assumed for evaluation purpose, Concord does not skip or duplicate PUs. Therefore, no Mathematica™ simulations have been carried out for Concord. Simulations have been carried out for ASP in order to evaluate its behavior in case of scenario B. The target buffer area, which influences the necessity of adaption phases, and the length of adaption phases were changed during evaluation.

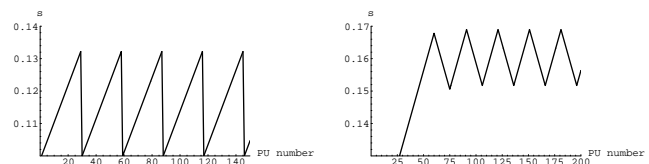
The evaluation of scenarios C and D is described and discussed together in [12]. After inserting the scenario parameter values into the formulae and discussing the results, one scenario has been simulated in detail with different adaption phases and different transmission delay curves. Two other scenarios have also been simulated. The discussion of evaluation results in [12] presents only results of the evaluation of one single scenario D. Based on the discussed observations during other evaluations enough data is presented for the assessment as described in this paper.

As the evaluation of inter-stream synchronization is based mainly on the evaluation of intra-stream synchronization, and as advantages of algorithms with respect to assessment parameter values are visible in inter-stream synchronization similarly to intra-stream synchronization, we have been concentrating mainly on the discussion of intra-stream synchronization assessment values.

The evaluation process revealed many detailed information about the QoS parameter values for synchronization, specifically delays and asynchronies, as well as buffer requirements. The detailed results are presented and discussed in [12]. In the following paragraphs, two results are shortly discussed to demonstrate the kind of results obtained during evaluation.

As already mentioned, scenario A is based on the assumption of a constant and well-known end-to-end delay. This scenario allows for the isolated study of effects of drift-

ing service and production rates, which may result from unsynchronized clocks. The scenario which has been chosen to study these rates is not very realistic but visualizes clearly the effects. We assume a slower receiver with a service rate 29 images per seconds, whereas the sender sends 30 images per second. This results in a buffer overflow, if we assume a buffer of the length of one single image, every second. In Concord, one image is skipped, i.e., thrown away every second to cope with this buffer overflow. Delays are constantly increasing in case of Concord until an image is thrown away, and then decreased again to the value of the end-to-end delay (see figure 9 (a)). Assuming a constant end-to-end delay of 0.1 s, the average delay for Concord is about 0.115 s.



(a) Concord

(b) ASP

Fig. 9. End-to-end Delay for a Slower Receiver

As is shown in figure 9 (b) the end-to-end delay with ASP is longer, its average is at 0.158 s. As ASP is an adaptive synchronization algorithm, buffer sizes play an important role. Above and below the target buffer area are critical buffer regions. We assume that such a region can at least buffer one PU, in case of video this means one image. The critical buffer region below the target buffer area has to be filled before play-out starts. This delays the play-out of the first PU by 0.033 s to 0.133 s. After one second one additional PU is in the buffer and only after the target buffer area is completely filled, adaption phases are started to empty the buffer. As the buffer is not completely emptied but only to within the target buffer area, the delay applying ASP is and remains for all PUs longer than applying Concord. Of course, the case that a maximum delay is well known and can be guaranteed which allows Concord to outwork ASP is not the most general case.

The second example illustrates how ASP adapts the end-to-end delay to the transmission delay. In scenario C we have chosen a transmission delay varying between 0.06 s and 0.1 s. We have assumed the delay in the end-system to be constant and to equal 0 s. A sample sequence of transmission delays is shown in figure 10 (a) whereas figure 10 (b) shows the end-to-end delay. We can observe that the end-to-end delay follows the shape of the transmission delay. The average transmission delay has a value of 0.0759 s whereas the average end-to-end delay has a value of 0.12632 s and is, thus, higher than the end-to-end delay of 0.1 s as achieved by Concord. The adaptive algorithm ASP can outwork Concord with respect to smaller delays

as soon as the delay jitter is higher than the target buffer fill expressed in time units.

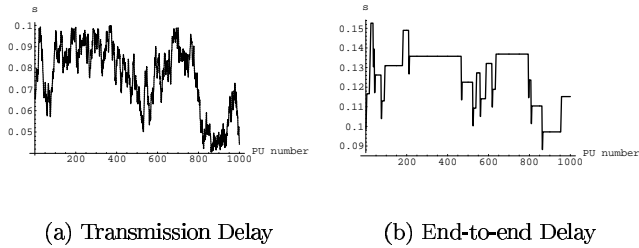


Fig. 10. Transmission and End-to-end Delay for ASP

C. Assessment

As the evaluation of inter-stream synchronization is based mainly on the evaluation of intra-stream synchronization, and as advantages of algorithms with respect to assessment parameter values are visible in inter-stream synchronization similarly to intra-stream synchronization, we concentrate on the discussion of intra-stream synchronization assessment values.

Given a constant delay Concord has better delay, buffer and asynchrony characteristics than ASP in case of *drifting production and service rates*. But ASP, in contrast, allows for the play-out of all PUs.

If the *jitter* is *small*, Concord also offers better delay, buffer and asynchrony characteristics. If the target area width in ASP is large enough, ASP also can synchronize data with an asynchrony of zero. The delay increase in ASP depends on the required buffer fill and plays the more a role the smaller the maximum transmission delay is.

If the *jitter* is *large*, ASP can achieve delay gains. But the delays may be higher than in Concord, as well. Asynchronies introduced by adaptations are not important, as adaptation phases can be chosen long enough so that asynchronies can no longer be perceived by the user. In case of large jitter, buffer gains in ASP can be significantly. If delays are not changing slowly, the delay gains in ASP are very small, if they can be achieved at all. In this case ASP only is advantageous with respect to buffer requirements.

If delays cannot be bounded, i.e., if we have an *unbounded delay*, only an adaptive scheme, i.e., ASP in our case, can guarantee for the play-out of all PUs.

TABLE VI
ASSESSMENT OF CONCORD AND ASP

	Asyn- chrony	Maxi- mum Delay	Average Delay	Buffer Re- quire- ments	Com- plete Play- out
Constant Delay, Drifting Production and Service Rates					
Concord	0	0	0	+	-
ASP	-	-	0/-	0	+
Constant Delay					
Concord	+	+	+	+	+
ASP	+	-	-	-	+
Small Jitter					
Concord	+	+	+	+	+
ASP	+	-	-	0	+
Large Jitter, Small Maximum Delay, Delay Changing Slowly					
Concord	+	+	0	-	+
ASP	0/+	-/0	-/0	+	+
Large Jitter, Large Maximum Delay, Delay Changing Slowly					
Concord	+	0	0	-	+
ASP	0/+	0	0/+	+	+
Large Jitter, Delay Changing Fast					
Concord	+	+	0	-	+
ASP	0/+	0	0	+	+
Unbounded Delay (maximum delay assumed)					
Concord	+	0	0	0/-	-
ASP	0/+	0	0	0/+	+

The results are summarized in table VI. (+) indicates that the algorithms behave very well with respect to the identified criterion, (0) indicates indifference or a behavior that is not so bad, whereas (-) indicates an undesirable criterion value. The results are always specified in order to compare the two algorithms and provide an assessment. The criterion *complete play-out* refers to synchronization error probabilities.

Asynchrony will be an important criterion for all high quality multimedia applications in which data must be well synchronized. As [7] points out, asynchronies are very important and stringent criteria. Such applications might be video movies, recordings of concerts, etc. The parameter *maximum delay* will probably play a vital role in many interactive applications especially in those that involve direct and regular interactions between humans. Such applications include video-conferencing. The problem of turn-taking, e.g., might be aggravated by longer delays. Intra-group asynchrony is vitally influenced by long delays. Next to maximum delays, the *average delay* also plays a vital role. In specific scenarios, especially if average and maximum delay really drift, a lower average delay might be preferable even for the price of temporally higher maximum delays. Also for this case we might think of video-conferencing, where problems of high delays might be de-

creased by smaller delays most of the time. *Buffer requirements* indicate some cost factors. On workstations and alike, they probably play no important role. But if we think of cheap stations where synchronized data is to be played, e.g., a cheap receiver for Video on Demand applications, buffer requirements especially for video data might become too expensive if they exceed a specific number of PUs. The *complete play-out* of PUs is vital for many high quality multimedia applications or data streams where single PUs are highly interdependent and must all be handled by the device.

In order to find the right scheme out of the rigid Concord and the adaptive ASP, for each criterion values have to be assigned to (+), (0), and (-) and then an overall value for each delay scenario can be calculated based on weights. These values allow for the choice of one scheme. The definition of such values equals the definition of a utility function. To define a utility function correctly detailed user studies are required. Such user studies are currently carried out in the TIK institute. First results have been described within a master's thesis [30].

In order to demonstrate the application of our assessment based on utility functions, we have defined two sample functions. The first might be used within a video-conference. The weights for the different criteria are: (1,3,3,2,1), i.e., asynchrony has the weight 1, maximum delay the weight 3, average delay also the weight 3, buffer requirements the weight 2 and the complete play-out the weight 1. This means that delays are specially important, i.e., main attention should be paid to minimize delays. Buffer requirements are also important, i.e., the algorithm should try to minimize buffer requirements in order to save costs in end-systems. The correct and complete play-out is less important during the application, people can interpolate some information that is missing. The utility function further assigns (-) = -2, (0) = 0 and (+) = 1. This assignment means that the algorithm should at least behave quite good, perfection is not so important, but a bad behavior in one criterion is judged to be very negative. The application of the utility function for the constant delay scenario with drifting service and production rates in case of the Concord algorithm ((0), (0), (0), (+), (-)) results in $1 \cdot 0 + 3 \cdot 0 + 3 \cdot 0 + 2 \cdot 1 + 1 \cdot (-2) = 0$. The utility function applied to ASP in the same scenario ((-), (-), (0/-), (0), (+)) results in $1 \cdot (-2) + 3 \cdot (-2) + 3 \cdot (\frac{1}{2} \cdot (0 + (-2))) + 2 \cdot 0 + 1 \cdot (1) = -10$. A possible utility function for CD quality audio on demand could be the following (4,1,1,1,4), i.e., small asynchrony as well as the complete play-out of data is extremely important. In this application it is important to fulfill criteria very well, therefore the following assignment is defined: (-) = -1, (0) = 0 and (+) = 2. The application of the utility function in case of the same scenario as above for the Concord algorithm results in: $4 \cdot 0 + 1 \cdot 0 + 1 \cdot 0 + 1 \cdot 2 + 4 \cdot (-1) = -2$. and for ASP in: $4 \cdot (-1) + 1 \cdot (-1) + 1 \cdot (\frac{1}{2} \cdot (0 + (-1))) + 1 \cdot 0 + 4 \cdot 2 = +2.5$.

As result of the application of utility functions we can state that in the calculated example, constant delay with drifting service and production rates, i.e., clock drift, ASP

is preferable in case of high quality CD audio and Concord is preferable in case of a video-conference. Applying the same utility functions to table VI we obtain table VII. In this table the utility function value of the recommended algorithm is marked bold.

TABLE VII
APPLICATION OF THE SAMPLE UTILITY FUNCTIONS TO THE
ASSESSMENT

	Concord	ASP
Constant Delay, Drifting Production and Service Rates		
video-conference	0	-10
CD quality audio	-2	2.5
Constant Delay		
video-conference	10	-14
CD quality audio	22	13
Small Jitter		
video-conference	10	-10
CD quality audio	22	14
Large Jitter, Small Maximum Delay, Delay Changing Slowly		
video-conference	1	-0.5
CD quality audio	17	12
Large Jitter, Large Maximum Delay, Delay Changing Slowly		
video-conference	-2	5
CD quality audio	15	14
Large Jitter, Delay Changing Fast		
video-conference	1	3.5
CD quality audio	17	14
Unbounded Delay		
video-conference	-3	2.5
CD quality audio	3	12.5

The utility functions that have been applied to derive table VII have been guessed. In order to derive correct results for real-world applications, utility functions have to be studied.

As Concord and ASP behave typically for receiver-based algorithms being adaptive or rigid, we believe that the results of the assessment with the (+,0,-) values can be generally applied to decide between these two classes of algorithms. This assumption, however, only can be validated by the analysis and evaluation of many more algorithms belonging to these two classes.

V. RELATED WORK

The assessment methodology described in this paper is not the first classification scheme that has been proposed in the literature. Due to the number of existing synchronization approaches and the relevance of synchronization for multimedia applications, different classification schemes have been proposed. We will discuss these schemes in short.

[31] defines a four layer synchronization model (the *Integrated Object Synchronization Model*) consisting of stream synchronization, object synchronization, window manager, and application layer.

1. *Transport Mechanism/Stream Synchronization:*

This layer can be mapped to layer one to four of the ISO/OSI reference model. It provides services to support stream synchronization as well as other network services. Functionality for the control of skew, jitter, utilization and reliability as well as of other QoS parameters is required. Multimedia objects are synchronized at the stream level, i.e., single audio packets and video frames are visible.

2. *Object Composition/Object Synchronization:*

This layer can be mapped to the session and presentation layer of the ISO/OSI reference model. It is responsible for synchronization at the level of objects. The applied algorithm should be suitable for real-time presentation of concurrent media. Synchronization is performed based on audio verbalizations and video segments.

3. *Window Manager and Application Layer:*

These layers belong to the application layer in the ISO/OSI reference model. They implement both the user interaction functionality. Synchronization on this level is very coarse. Synchronization points occur at invocations and terminations of application programs, object selection or query, pause operations, or any other user-object operation.

Based on this object synchronization model, both schemes, Concord and ASP belong to the same synchronization layer, i.e., to the stream or object layer depending on the granularity of PUs. For this reason, the model cannot be applied in order to compare and classify Concord and ASP. The classification is not fine-grained enough.

The *Synchronization Reference Model* [32] is described by the four layers media layer, stream layer, object layer, and specification layer.

Synchronization mechanisms are implemented on each layer and provided by appropriate interfaces, which can be used to specify or to enforce the temporal relationships. They define services that can be used by an application directly or by the next higher layer. Higher layers offer higher programming and Quality-of-Service abstractions.

An application operates at the *media layer* on a single continuous stream, seen as a sequence of logical data units. Using this layer, the application itself is responsible for intra-stream synchronization.

The *stream layer* operates on continuous media streams as well as on groups of media streams. In a group all streams are presented in parallel by using mechanisms for inter-stream synchronization. The offered abstraction is the notion of streams with timing parameters concerning the QoS for intra-stream and inter-stream synchronization. The streams are executed in a real-time environment whereas applications using the stream layer services are executed in a non real-time environment.

The *object layer* operates on all types of media and hides the differences between time independent and time dependent media. The layer takes a synchronization specification as input and is responsible for the correct schedule of the

overall presentation.

The *specification layer* is an open layer without explicit interface. It contains applications and tools allowing for the creation of synchronization specifications. [33] and [34] describe the same reference model.

The two algorithms Concord and ASP belong to the same layer, the stream layer, of the synchronization reference model. For this reason, this model is not suitable to compare the two algorithms. The classification based on the proposed model is too high level.

The *evaluation of multimedia synchronization schemes* by Ehley, Furht and Ilyas is described in [35]. A classification of synchronization algorithms is proposed based on four different data location models: local single source local, local multiple sources, distributed single source, distributed multiple sources.

Synchronization techniques have been classified based on a tree structure. The basic classification criterion is the location of the synchronization control. The local schemes that can control synchronization at different levels within one workstation or on servers and co-processors, are not suitable for distributed multimedia applications as aggravating factors of data transmission like delay jitters are not taken into account. According to [35] synchronization in distributed multimedia applications can either be protocol-based at sending and/or receiving processes, network-based with distributed synchronization functionality among all internetwork gateway nodes, as well as within servers or co-processors.

Based on the classification of Ehley et al. Concord as well as ASP provide distributed protocol-based synchronization and, thus, belong to the same class of algorithms.

In [36] several inter-stream synchronization schemes are compared based on a *specified criteria catalogue*. The authors have chosen the methods of multiplexing or interleaving, out-of band and time-stamping and evaluated them based on the criteria:

- *Prerequisites:* network wide clocks, availability of extra channels, alteration of data streams.
- *Type of applications:* stored data, real-time (live data), different sources.
- *Costs:* extra messages, additional channels, processing costs.
- *QoS requirements:* delay/jitter, error, order of delivery.

Concord and ASP are both synchronization schemes which are based on the time-stamping approach. The QoS criteria delay/jitter and error are also included in our set of assessment parameters, but the criteria catalogue as presented in [36] is not fine grained enough to assess Concord and ASP with respect to each other.

[37] describes an approach to build a *test environment for synchronization algorithms* that measures skew, drift, jitter and losses. These parameters are also incorporated in our set of assessment parameters. Based on this apparatus, arbitrary synchronization schemes are measured and results are discussed. The apparatus is designed in order to allow for the measurement of synchronization performance of any synchronization algorithm, but it has not been eval-

uated by applying the described method to synchronization algorithms that have already been published in the literature. With our assessment scheme as described in part II we propose a classification method that allows equally for the determination of the described parameters skew, drift, jitter and losses. Our methodology is not based on measurement results but on analytical results and has been, in addition, evaluated by applying it to existing synchronization algorithms.

VI. SUMMARY

In this report we have introduced the concept of QoS for synchronization algorithms. We have further applied this concept in an assessment scheme for synchronization algorithms and have presented assessment results for two specific synchronization algorithms.

QoS for synchronization algorithms can be defined based on the understanding of a synchronization algorithm as a layer that fills the gap between services provided by networks and end-systems and the requirements of user and application. Based on this concept four QoS parameters have been identified that allow for the description of the synchronization service. Asynchrony is the inherent QoS parameter for synchronization whereas the more general QoS parameters reliability and delays are influenced by synchronization algorithms. Buffer requirements are QoS parameters being offered by lower layers. The parameter asynchrony has been formally defined based on a time system that has equally been introduced. Based on a definition of guaranteed synchronization QoS parameter values of synchronization algorithm can be analyzed.

The presented assessment methodology consists of a QoS based analysis of synchronization algorithms and an evaluation of analysis results. Both steps are introduced and discussed in detail in this report. In order to demonstrate the application of the assessment methodology, analysis and evaluation results of two well-known synchronization algorithms are presented in this report. Based on these results an assessment of the two algorithms is provided.

If assessment parameter sets can be defined for other algorithms, our methodology can also be applied in order to assess algorithms aiming to solve different problems.

ACKNOWLEDGMENTS

Many thanks go to Dr. Burkhard Stiller for his advice and support during the described work as well as for the proof-reading of this report. Many thanks go also to Prof. Bernhard Plattner, the advisor of my Ph.D. for the possibility to carry out my research in his laboratory. Part of my work is described in this report. Many thanks go also to all colleagues in our research group for all comments and input during my work.

REFERENCES

- [1] P. Hoepfner, "Synchronizing the Presentation of Multimedia Objects," *multimedia*, vol. 15, no. 9, pp. 557–564, Nov. 1992.
- [2] K. Rothermel and T. Helbig, "Clock Hierarchies: An Abstraction for Grouping and Controlling Media Streams," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 1, pp. 174–184, Jan. 1996.
- [3] T. D. C. Little and A. Ghafoor, "Synchronization and Storage Models for Multimedia Objects," *IEEE Journal on Selected Areas in Communication*, vol. 8, no. 3, pp. 413 – 427, April 1990.
- [4] C.-C. Yuan and J.-H. Huang, "A Multimedia Synchronization Model and Its Implementation on Transport Protocols," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 1, pp. 212–225, Jan. 1996.
- [5] M. J. Pérez-Luque and T. D. C. Little, "A Temporal Reference Framework for Multimedia Synchronization," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 1, pp. 36–51, January 1996.
- [6] G. Coulson, G. S. Blair, J. B. Stefani, F. Horn, and L. Hazard, "Supporting the Real-time Requirements of Continuous Media in Open Distributed Processing," *Computer Networks and ISDN Systems*, vol. 27, no. 8, pp. 1231–1246, July 1995.
- [7] R. Steinmetz, "Human Perception of Jitter and Media Synchronization," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 1, pp. 61–72, Jan. 1996.
- [8] A. Vogel, B. Kerkervé, G. von Bochmann, and J. Gecsei, "Distributed Multimedia and QoS: A Survey," *IEEE MultiMedia*, pp. 10–19, summer 1995.
- [9] C. Aurrecochea, A. T. Campbell, and L. Hauw, "A Survey of QoS Architectures," *Multimedia Systems*, , no. 6, pp. 138–151, 1998.
- [10] G. Coulouris, J. Dollimore, and T. Kindberg, *Distributed Systems*, Addison-Wesley, 1994.
- [11] E.A. Isaacs and J.C. Tang, "What Video Can and Can't do for Collaboration: A Case Study," in *ACM Multimedia '93, Proceedings*, June 1993, pp. 199–206.
- [12] C. Class, *Synchronization in Distributed Multimedia Applications*, Ph.D. thesis, Swiss Federal Institute of Technology, Zürich, Sept. 1999.
- [13] C. Class and B. Stiller, "A Methodology to Assess Synchronization Algorithms for Distributed Applications," in *The 23rd Annual Conference on Local Computer Networks (LCN'98)*, Boston, MA, USA, Oct. 1998, pp. 140–149.
- [14] K. Rothermel and T. Helbig, "An Adaptive Protocol for Synchronizing Media Streams," *Multimedia Systems*, vol. 5, no. 5, pp. 324–336, Sept. 1997.
- [15] C. J. Sreenan, "A Service Oriented Approach to Continuous Media Synchronization," in *IEEE INFOCOM '94*. IEEE, 1994, vol. 2, pp. 936–943.
- [16] N. Shivakumar, C. J. Sreenan, B. Narendran, and P. Agrawal, "The Concord Algorithm for Synchronization of Networked Multimedia Streams," in *Proceedings of the International Conference on Multimedia Computing and Systems*. May 15-18, 1995, pp. 31–40, IEEE Computer Society Press.
- [17] J.-P. Courtiat, L. F. Rust da Costa Carmo, and R. Cruz de Oliveira, "A General-purpose Multimedia Synchronization Mechanism Based on Causal Relations," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 1, pp. 185–195, Jan. 1996.
- [18] S. H. Son and N. Agarwal, "Synchronization of Temporal Constructs in Distributed Multimedia Systems with Controlled Accuracy," in *Proceedings of the International Conference on Multimedia Computing and Systems*, Boston, MA, USA, May 1994, pp. 550–555.
- [19] F. Alvarez-Cuevas, M. Bertran, F. Oller, and J. M. Selga, "Voice Synchronization in Packet Switching Networks," *IEEE Network*, pp. 20–25, Sept. 1993.
- [20] I. F. Akyildiz and W. Yen, "Multimedia Group Synchronization Protocols for Integrated Services Networks," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 1, pp. 162–173, Jan. 1996.
- [21] D. Le Gall, "MPEG: A Video Compression Standard for Multimedia Applications," *Communications of the ACM*, vol. 34, no. 4, pp. 103–111, Apr. 1991.
- [22] B. Furht, S. W. Smoliar, and H. Zhang, *Video and Image Processing in Multimedia Systems*, Kluwer Academics Publishers, 1995.
- [23] C. Class, B. Stiller, and B. Plattner, "Die Verwendung von Dienstgüteparametern zur Bewertung von Synchronisationsalgorithmen," *Praxis der Informationsverarbeitung und Kommunikation (PIK)*, vol. 21, no. 4, pp. 219–227, Dec. 1998.

- [24] C. Class, "Analysis of the Concord Algorithm and the Adaptive Synchronization Protocol Using QoS," Tech. Rep. 37, Computer Engineering and Networks Laboratory, ETH Zürich, Switzerland, Feb. 1998.
- [25] "CINEMA Project," <http://www.informa-tik.uni-stuttgart.de/ipvr/vs/projekte/cine-ma.engl.html>.
- [26] T. Helbig, *Kommunikation und Synchronisation multimedialer Datenströme in verteilten Systemen*, Ph.D. thesis, Universität Stuttgart, Germany, June 1996.
- [27] G. Wallace, "The JPEG Still Picture Compression Standard," *Communications of the ACM*, vol. 34, no. 4, pp. 30–44, Apr. 1991.
- [28] G. Fankhauser, M. Dasen, N. Weiler, B. Plattner, and B. Stiller, "WaveVideo - An Integrated Approach to Adaptive Wireless Video," Accepted for publication in ACM Monet, Special Issue on Adaptive Mobile Networking and Computing.
- [29] B. Melamed, "An Overview of TES Processes and Modeling Methodology," in *Joint Tutorial Papers Performance 93 and Sigmetrics 93*, Rome, Italy, and Santa Clara, CA, USA, 27. Sept. - 1. Oct and 10-14 May 1993, pp. 359–392.
- [30] P. Scherer, "Multimediasoftware für Qualitätsuntersuchungen im ATM-Netz," M.S. thesis, TIK, Swiss Federal Institute of Technology, Zürich (ETHZ), Mar. 1999.
- [31] T. D. C. Little and A. Ghafoor, "Network Considerations for Distributed Multimedia Object Composition and Communication," *IEEE Network Magazine*, vol. 4, no. 6, pp. 32–40, 45–49, Nov. 1990.
- [32] G. Blakowski and R. Steinmetz, "A Media Synchronization Survey: Reference Model, Specification, and Case Studies," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 1, pp. 5–35, Jan. 1996.
- [33] T. Meyer and W. Effelsberg, "Multimedia-Synchronisation: Eine vergleichende Analyse der bestehenden Methoden," in *Verteilte Multimedia Systeme*, W. Effelsberg and K. Rothermel, Eds., Stuttgart, Germany, 18./19. Februar 1993, GI/ITG Arbeitstrreffen, pp. 189–205.
- [34] R. Steinmetz and K. Nahrstedt, *Multimedia: Computing, Communications and Applications*, Prentice Hall, P T R, 1995.
- [35] L. Ehley, B. Furht, and M. Ilyas, "Evaluation of Multimedia Synchronization Techniques," in *Proc. of the Int. Conf. on Multimedia Computing and Systems*, Boston, MA, USA, May 1994, pp. 514–519.
- [36] V. Saparamadu, A. Senevirante, and M. Fry, "A Review of Inter Media Synchronization Schemes," in *Proceedings of the First Int. Conf. on Multi-Media Modeling*, Singapore, Nov. 1995, pp. 229–239.
- [37] B. K. Schmidt, J. D. Northcutt, and M. S. Lam, "A Method and Apparatus for Measuring Media Synchronization," in *5th Int. Workshop on Network and Operating System Support for Digital Audio and Video*, Durham, NH, USA, April 18-21, 1995, pp. 190–201.