A Methodology to Assess Synchronization Algorithms for Distributed Applications

Christina Class, Burkhard Stiller
Computer Engineering and Networks Laboratory (TIK)
Swiss Federal Institute of Technology, ETH Zürich
Gloriastr. 35, 8092 Zürich, Switzerland
{class|stiller}@tik.ee.ethz.ch

Abstract

The need for algorithms providing synchronization between audio and video data was driven by the advent of multimedia applications in distributed environments. Numerous algorithms have been proposed, but there is a lack in methodologies to assess these algorithms, to compare them, and to determine algorithms best suited for a given networking and end-system environment and application.

In this paper a methodology to assess synchronization algorithms for distributed multimedia applications is presented. The methodology is based on four Quality of Service (QoS) parameters for synchronization comprising asynchrony, synchronization error probability, synchronization delays, and buffer requirements. An analytical analysis of these parameters is performed. A key feature of this methodology is that it parameterizes the network, the end-system, and the application. Thus, it allows for the assessment of any synchronization algorithm independent from network and end-system. Influences of the infrastructure on synchronization can be determined by the proposed methodology.
Keywords: *Quality of Service, Synchronization, Analysis, Distributed Multimedia Applications*

1 Introduction

The increasing development in computer engineering and technology leads to increased computing performance. Numerous *multimedia applications* became feasible like Video-on-Demand, Video-conferencing, Virtual Museums, and Teleeducation on top of a broad range of different communication networks. These applications are characterized by the integrated processing of different media types, such as audio and video. Some of these media are time dependent, *i.e.* change their values as a function of time. For this reason, recording, storing, and presentation of data must take into account the data's time component. The task of multimedia *synchronization* is to retain data's temporal relationships relative to each other or to a clock. In case of distributed multimedia applications this task is aggravated.

Multimedia applications need to provide a quality acceptable to the user. The quality must be provided by end-systems, the network as well as the application itself. The concept of Quality of Service (QoS) allows for the specification of the quality provided by a system or required by the user.

Different networks with varying characteristics support distributed multimedia applications. Within the literature many approaches and a huge range of synchronization algorithms exist. Firstly, there is a need to relate characteristics of the network and end-systems with characteristics of synchronization. This determines, if there exists a synchronization method that can, within a given environment, provide the required quality to the user. In response to this need, synchronization quality is defined comprising the QoS parameters asynchrony, buffer requirements, synchronization error probabilities, and synchronization delay. Secondly, there is also a need to classify and assess existing and future synchronization algorithms. Therefore, a methodology based on synchronization quality is defined to assess synchronization algorithms.

In this paper the methodology is presented as well as results of the analysis of two sample synchronization algorithms. These analytical results are required to apply the methodology. The application of the methodology to as-

sessments of algorithms is not shown in this paper.

Following a brief discussion of related work in Section 2, Section 3 is devoted to specify characteristics of synchronization quality. Section 4 introduces the methodology based on synchronization quality parameters. While Section 5 determines an excerpt of results, Section 6 concludes the work and sketches future work.

2 Related Work

Synchronization consists of two consecutive steps to be fulfilled. In a first step, synchronization has to be specified, *i.e.*, it has to be defined how data have to be played synchronously. Many approaches have been proposed, *e.g.*, the object composition Petri Net (OCPN) [10], the real-time synchronization model (RTSM) [22] and the inter-operable Petri Nets (IPN) [2]. [15] defines a temporal reference framework that allows for the classification of specification schemes.

The second step of synchronization is to retain the specified time relationships. Many different approaches are proposed in the literature for content-based, *e.g.* [13], event-based, *e.g.* [14], and temporal synchronization, *e.g.* [22], [23]. The two synchronization algorithms [17] and [20] analyzed in this paper are temporal synchronization approaches.

[3] describes a layered reference model for synchronization. [18] compares three basic synchronization methods (multiplexing, out-of-band, and time-stamping) according to a specified set of criteria. [6] defines different general types of synchronization services. [19] describes an approach to build a test environment for synchronization algorithms that measures skew, drift, jitter and losses. These approaches provide useful insight into synchronization but yet do not supply a general classification method for synchronization algorithms.

Quality of Service has been introduced as a powerful concept to define, monitor, and supervise the quality of communications. In particular, QoS is defined in Open Distributed Processing as "a set of quality requirements on the collective behavior of one or more objects" [9]. Slightly different definitions for QoS exist, *e.g.*, within [12]. The use of QoS parameters to determine general synchronization issues on an algorithmic level has been mentioned in [24], synchronization definitions in terms of skew and jitter are included in [21]. Several authors have included synchronization quality criteria in the discussion of their own work, *e.g.,*buffer utilization and delay [11], delay, losses and skew [17], buffer requirements [7], jitters [22], buffer size and delay [20], asynchrony and losses [16]. However, a detailed specification and utilization of generic synchronization quality parameters have not been reported yet.

Therefore, in this paper four generic QoS parameters for synchronization are defined and a general methodology is introduced that allows for the comparison of synchronization algorithms by analytical analyses.

3 Synchronization Quality

In this section basic terms are introduced and synchronization quality is defined.

3.1 Analytical Environment and Terminology

One *presentation unit (PU)* refers to the atomic information of a media stream. Examples of presentation units are video frames or audio samples. PUs are independent, *i.e.* no PU carries information relevant to others. In compression schemes like MPEG, compression is achieved by creating dependencies between PUs. To overcome these dependencies, the bit error probability has to be increased such as to cover also the probability that PUs cannot be presented due to a bit error in other PUs or due to loss of other PUs. Independent PUs can, thus, be modeled by an adequate bit error probability.

Multimedia applications are characterized by the integrated processing of different media types being *time dependent* or *time independent*. The *valid time interval* is defined as the interval in which a time dependent PU is valid.

The *media schedule* specifies the correct play-out time, called the *media time*, for each PU. The first PU of an application is assigned the media time 0. In applications handling stored data the media schedule must be specified explicitly, while in applications handling live data it is implicitly given by the capturing time.

In this paper, we assume that PUs are sent accordingly to their media schedule. Capturing live data and sending them instantaneously to the receiver is equivalent to data being sent accordingly to the media schedule. In case of stored data, the media schedule has to be read and interpreted on the sender's side and the data must be sent to the receiver accordingly to the media schedule. To do so, a local synchronization mechanism has to be applied in the sender, controlling the synchronized sending of the data.

Each PU i has different times assigned:

- The *sending time* denotes the time, the PU is sent ($t_{s,i}$).
- The *receiving time* specifies the time point the PU was received ($t_{r,i}$).
- The *presentation time* is the time the PU has to be presented ($t_{p,i}$). It is the target time of the synchronization and specifies the presentation time as achieved by the synchronization method.[1]

[1] If some constant time t_x is needed, *e.g.*, in a device to prepare data for

In case of different data streams that have to be distinguished, the indicator of the stream stands as additional index, *e.g.*, $t_{p,(l,i)}$ or $t_{p,(k,i)}$ for streams l and k.

The time in different end-systems is local and in general varies from a global reference time. Time differences and mappings can be clearly specified by the time model as introduced in [5]. This model allows for the analysis of algorithms based on the ideal assumption of perfect global clocks and for the consideration of local times and clock drifts in a second step.

Finally, one *synchronization group* encompasses all streams of an application that have to be synchronized. These streams do not have to be sent necessarily from the same sender. This definition is similar to the one provided in [17].

3.2 Quality of Synchronization (QoSy)

The concept of QoS is extended by parameters indicating the *Quality of Synchronization (QoSy)* [4]. These parameters encompass asynchrony, synchronization error probability, synchronization delays, and synchronization buffer requirements. They can be applied to describe the service quality offered by a given synchronization algorithm.

Asynchrony

The *presentation times interval* is defined as the time interval that elapses between the triggering of the play-out of two consecutive PUs in the receiver.[2] The *optimal presentation times interval* is defined as the time interval between the triggering of the play-out of two consecutive PUs according to the media schedule. *Asynchrony* is defined as the maximum difference of the length of presentation interval and optimal presentation interval as accepted by the synchronization algorithm.

The *maximum asynchrony for intra-stream synchronization* (or jitter) Δ_{ia} in a stream is defined by the maximum deviation of the presentation times interval of two subsequent PUs ($t_{p,i+1} - t_{p,i}$) from the optimal interval ($t_{s,i+1} - t_{s,i}$, as the sending of PUs follows the media schedule). Figure 1 depicts the minimum and maximum valid presentation times interval as well as the optimal presentation times interval. Out-of-order delivery of PUs is defined as a synchronization error and not accepted.

The *maximum asynchrony for inter-stream synchronization* (or skew) Δ_{ie} for two streams k and l, which may originate from two different senders, is defined as the maximum deviation of the presentation interval of two PUs ($t_{k,p,i} - t_{l,p,j}$) from the optimal presentation interval ($t_{k,s,i} - t_{l,s,j}$).

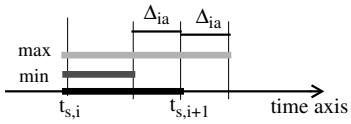

Figure 1. Intra-stream Asynchrony

Synchronization Delays

The *delay* of data that is transmitted live, *e.g.*, during a video conference, as well as the delay before the first data is presented to the user, may be critical for the quality perceived by the user. Synchronization may increase these delays by buffering data to smooth the jitter.

Buffer Requirements

Whenever the network delay can vary, synchronization needs to buffer PUs arriving too early to ensure an in-time play-out of PUs. *Buffer requirements* can, thus, serve as an indication of the cost of synchronization algorithms within the given environment.

Synchronization Error Probability

In general, three different synchronization errors can be defined [4]:

1. There exist one or more consecutive valid time intervals of PUs in which the corresponding PU is not displayed. Reasons are late arriving PUs, losses of PUs, or unreliable end-systems.
2. One or more PUs are displayed outside their valid time intervals. This is referred to as *loss of synchronization*.
3. The mapping between the different time systems [5] is incorrect or the time systems are not mapped at all. In this case a *loss of time* occurs.

The synchronization error probability serves as an indicator for the quality of the synchronization scheme.

The following example illustrates the relevance for four synchronization parameters to synchronization quality:

Considering two scenarios it can be assumed 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 $t_{s,i} + \Delta_{\max}$, ($\Delta_{\max}$ known maximum delay). In scenario B

presentation by decompressing data, the presentation has to be triggered at $t_{p,i} - t_x$. This can be substituted by a new presentation time. If t_x varies, the variation results in additional asynchrony.

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

the receiver displays data at $t_{s,i} + \Delta_{\min}$ ($\Delta_{\min}$ known minimum delay). 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 3.2 indicates that one single synchronization quality parameter alone cannot serve as a valid indicator to describe the overall quality.

Table 1. Synchronization Quality Values

	Asynchrony	Delay	Buffer	Errors
A	0	maximum	maximum	0
B	0	minimum	minimum	maximum

4 The Methodology

In this section the synchronization model and the assessment scheme which underly the proposed methodology are presented.

4.1 A Model of Synchronization

In distributed multimedia applications five different components can be distinguished for the purpose of analyzing synchronization algorithms (cf. Fig. 2): (1) user, (2) application, (3) synchronization algorithm, (4) end-system, and (5) communication system or network.

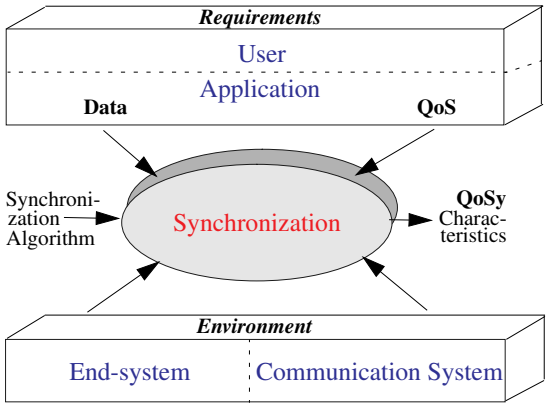

Figure 2. Model of Synchronization Analysis

Applications offer data, where at least a part of them is time dependent and, thus, require mechanisms to ensure a synchronized play-out. Data inherently own characteristics being relevant to synchronization, *e.g.*, the play-out period.

On one hand, the user of an application imposes requirements with respect to the entire application as well as to the synchronization itself. On the other hand, there exist a variety of characteristics of the underlying end-system and the communication system influencing the synchronization.

The network and the end-system offer a service with given characteristics concerning the synchronization investigation. In general, this service may not allow the application to play out data synchronously, particularly, if there exist delay variances in transmission. For this reason, synchronization methods have to be applied to fill the gap between application requirements and users and the "service" characteristics of the end-system and the network. To describe the quality offered by these synchronization mechanisms a method is required allowing for the evaluation of their quality, independently of the specific characteristics of network, end-system, and application.

4.2 Analytical Assessment

Applying the model above and the Quality of Synchronization parameters as defined in Section 3, synchronization algorithms can be analyzed.

Analyzing an algorithm is a demanding task which is not supported by rules or tools. However, clearly separated parameters allow for the direct calculation of the asynchrony. In a first step algorithms have to be analyzed, assuming perfectly synchronized clocks as the simplest case defined by the algorithm. Afterwards, all factors are taken into account step by step that aggravate synchronization by determining changes to the parameters introduced by these factors. The analysis was supported by the clear definition of different time systems [5].

The analysis remains independent of characteristics of the underlying network and end-system which are parameterized. Data characteristics in terms of parameters are included into the analysis, too.

The results of this analysis are expressions of parameters describing the environment (see Section 5.1 and Section 5.2). In this context environment refers to the environment of the synchronization algorithm, *i.e.* communication system, end-system, data, and application. In addition they determine, when evaluated, the synchronization quality characteristics of the analyzed algorithm within this specified environment. Furthermore a list of parameters describing characteristics of the environment and influencing the synchronization quality can be derived (see Section 5.3). These results allow for a deeper, analytical understanding of the synchronization service provided by the studied algorithm and they allow 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 [20] belongs to

the class of rigid synchronization algorithms, whereas the *Adaptive Synchronization Protocol (ASP)* [17] defines an adaptive synchronization algorithm. Since QoSy parameters as defined in Section 3.2 are referred to as guaranteed parameters, the analyses have to take into account the worst case.

The analyses have been performed for unicast communications and periodic live data or data that are sent according to a media schedule, respectively. Specially focussed on asynchrony synchronization errors, partly buffer requirements, and delays depend to a high degree on the communication system's characteristics. The transmission delay is analyzed, whereas the start-up delay is not considered, since this occurs only once during the beginning of a presentation and does not have an impact on the quality.[3] Since these analyses determine the synchronization quality in the receiver, resulting expressions also resolve the synchronization quality of one receiver in a multicast group. This holds as long at it is evaluated with end-system characteristics of this receiver and characteristics being valid for the transmission from the multicast sender(s) to this specific receiver.

The second type of 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 and are, therefore, not included in the analysis. For the same reason, an incorrect time mapping resulting in the third type of synchronization errors is not included in the analysis of synchronization quality. Thus, the parameter *synchronization error probability* is restricted for analysis purposes to the case in which a PU is not presented.

5 Determined Results

Central resulting expressions of the performed analyses are presented here while the detailed results can be found in [5].

5.1 Analyzing the Concord Algorithm

The *Concord algorithm* [20] 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 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.

[20] 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 2), 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 2. 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	$\left\lceil \frac{\Delta_{opt}-\Delta_{min}}{T_r} \right\rceil$ 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-

[3] This may be different, if schemes for stored data are analyzed allowing for the in advance transmission of data to minimize synchronization errors.

out rate, a buffer overflow may occur and in total $\left\lfloor \frac{\lambda}{T_r} \right\rfloor$ 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 QoSy 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.

5.2 Analyzing ASP

The *Adaptive Synchronization Protocol (ASP)* [17] 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

[a] During 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 3 and 4 depict asynchrony and delay parameter values for master and slave streams.

Table 3. 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	$T_{r,M} - \frac{T_{r,M} \cdot L_M}{\left\lceil \frac{\frac{1}{2} \cdot c \cdot j_M}{T_{r,M}} \right\rceil + L_M}$	
delay	decreasing	

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 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.

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.

Table 4. 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	$\frac{\frac{1}{2} \cdot c \cdot j_M}{\left\lceil \frac{l}{T_{r,S}} \right\rceil}$	
delay	increasing	
speeding up adaption		
asynchrony	$T_{r,S} - \frac{T_{r,S} \cdot \left\lceil \frac{l}{T_{r,S}} \right\rceil}{\left\lceil \frac{\frac{1}{2} \cdot c \cdot j_M}{T_{r,S}} \right\rceil + \left\lceil \frac{l}{T_{r,S}} \right\rceil}$	
delay	decreasing	

5.3 Qualitative Results

The analyses presented so far result in formal results on an abstract level. As network, application, and end-system characteristics are numerically parameterized, they are to be included into the calculation of synchronization quality, *i.e.,* the synchronization quality expressions have to be evaluated by assigning values to these parameters.

The analyses carried out revealed relevant environment characteristics for the transformation from general results to specific numerical results. Figure 3 shows relevant influencing factors for the Concord algorithm, whereas Figure 4 displays these factors for the Adaptive Synchronization Protocol.

5.4 Assessment

The results of the analysis of the *Concord* algorithm concur with the values for buffer size and delay as discussed in [20], validating our results. As asynchrony has been partly discussed in [20], parts of our results concerning asynchrony as well as the discussion on synchronization

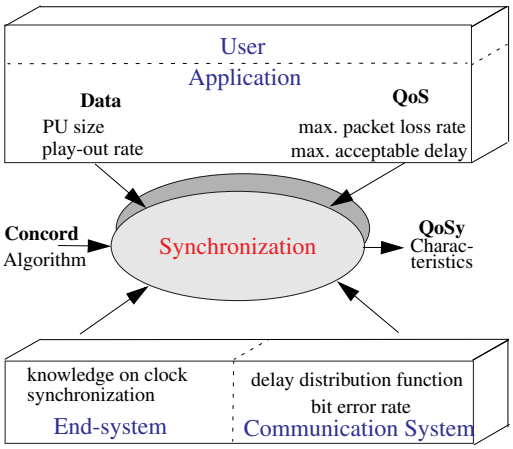

Figure 3. The Analysis Model Applied to Concord

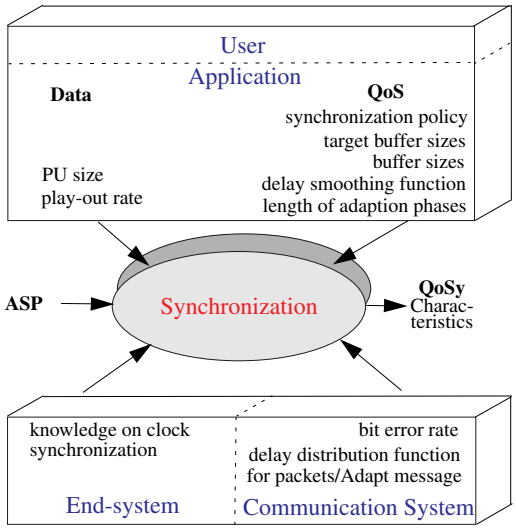

Figure 4. The Analysis Model Applied to ASP

errors provided in [5] and mentioned above determine new results. These results corroborated our assumptions on the quality and performance of the algorithm. The identified list of influencing parameters as depicted in Figure 3 determines for the first time parameters that should be tuned to improve synchronization quality in a given implementation providing insufficient quality.

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

Based on the simulation in the CINEMA environment [1], more detailed discussions on buffer requirements and stability issues of ASP are included in [8] and [17]. In contrast, the provided analytical results above, especially concern asynchrony. The list of influencing factors as depicted in Figure 4 together with the analysis results allows for the tuning of ASP to provide high quality synchronization. Asynchrony is necessarily introduced by entering an adaption phase. 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 [21] even during adaption phases. This was shown by simulation in [8] and [17].

6 Conclusions and Future Work

In this paper a methodology to assess synchronization algorithms is presented. The methodology is based on the concept of analyzing synchronization algorithms analytically. The analysis is performed by determining values for synchronization quality parameters, comprising asynchrony, synchronization delays, buffer requirements, and synchronization error probabilities. Characteristics of end-systems, the communication system, the application as well as the data are parameterized to allow for the analysis of the isolated synchronization algorithm and, thus, of the synchronization quality provided by this algorithm. Two exemplified analyses of a rigid synchronization algorithm (Concord) and an adaptive one (ASP) have been carried out for unicast applications including live data.

The fact that data may be retrieved in advance and can be stored in the receiver has not yet been taken into account determining future work. QoSy criteria will also be studied in multi-party applications, especially, in video-conferencing including a critical delay.

As numerous synchronization algorithms have been proposed in the literature, two problem fields can be identified: Firstly, a synchronization algorithm must be selected for a given problem field that can fill the gap between (1) application requirements concerning synchronization and presentation quality and (2) characteristics and guarantees provided by end-systems and the network. This task also includes the determination of the existence of such a synchronization algorithm. Secondly, existing synchronization algorithms need to be comparable with respect to different criteria which also include characteristics of applications, end-systems, and communication systems.

The presented methodology is a first step to answer both problem fields. To identify a synchronization algorithm that fills the gap between a given environment and application, parameter values describing the environment and the application can be included into expressions resulting from analyses of these algorithms. Thus, the synchronization quality parameters of these algorithms for a specific case can be evaluated. If these quality values fulfill specified criteria,

i.e. the application/user requirements for synchronization, this algorithm solves the investigated problem. Based on parameter values for the analyzed algorithms the best suited algorithm for the given problem can be chosen.

In addition, a classification of synchronization algorithms can be undertaken based on the analysis of different sample parameter values. This will result in classes of synchronization algorithms focusing on different criteria as perfect synchronization, low delay, or low error rate. For instance, ASP focuses on (1) low delay and low buffer requirements or (2) low loss rate for the price of a less optimal synchronization during adaption phases.

The presented synchronization quality parameters QoSy provide means to better understand synchronization algorithms and to perform an analytical analysis. By this, future applications will profit since a perfectly suitable synchronization algorithm can be selected.

Acknowledgments The authors are grateful to Bernhard Plattner for the possibility to conduct the described research and Thomas Walter for commenting in detail on earlier versions of this work.

References

[1] CINEMA Project. http://www.informa-tik.uni-stuttgart.de/ipvr/vs/projekte/cine-ma.engl.html.

[2] Y. Y. Al-Salqan and C. K. Chang. Temporal Relations and Synchronization Agents. *IEEE Multimedia*, 3(2):30–39, Summer 1996.

[3] G. Blakowski and R. Steinmetz. A Media Synchronization Survey: Reference Model, Specification, and Case Studies. *IEEE Journal on Selected Areas in Communications*, 14(1):5–35, January 1996.

[4] C. Class. Synchronization Issues in Distributed Applications: Definitions, Problems, and Quality of Synchronization. Technical Report 31, Computer Engineering and Networks Laboratory, ETH Zürich, Switzerland, December 1997.

[5] C. Class. Analysis of the Concord Algorithm and the Adaptive Synchronization Protocol Using QoSy. Technical Report 37, Computer Engineering and Networks Laboratory, ETH Zürich, Switzerland, February 1998.

[6] L. F. Rust da Costa Carmo, P. de Saqui-Sannes, and J.-P. Courtiat. Basic Synchronization Concepts in Multimedia Systems. In P. V. Rangan, editor, *Network and Operating System Support for Digital Audio and Video, 3rd International Workshop*, pages 94–105, La Jolla, California, USA, November 1992.

[7] W. Geyer, C. Bernhardt, and E. Biersack. A Synchronization Scheme for Stored Multimedia Streams. In B. Butscher, E. Moeller, and H. Pusch, editors, *Interactive Distributed Multimedia Systems and Services, European Workshop IDMS'96*, pages 277–296, Berlin, Germany, March 1996. Springer.

[8] T. Helbig. *Kommunikation und Synchronisation multimedialer Datenströme in verteilten Systemen*. PhD thesis, Universität Stuttgart, Germany, June 1996.

[9] International Telecommunications Union Telecommunications Standardization Sector, Recommendation X.902. *Information Technology - Open Distributed Processing - Reference Model*, November 1995.

[10] T. D. C. Little and A. Ghafoor. Synchronization and Storage Models for Multimedia Objects. *IEEE Journal on Selected Areas in Communication*, 8(3):413 – 427, April 1990.

[11] T. D. C. Little and A. Ghafoor. Multimedia Synchronization Protocols for Broadband Integrated Services. *IEEE Journal on Selected Areas in Communications*, 9(9):1368–1381, December 1991.

[12] K. Nahrstedt and R. Steinmetz. Resource Management in Networked Multimedia Systems. *IEEE Computer*, 29(5):52–63, May 1995.

[13] D.-Y. Oh, S. S. Kumar, and P. V. Rangan. Content-based inter-media synchronization. In *Multimedia Computing and Networking 1995*, volume 2417, pages 202 – 214, San Jose, California, SPIE, February 1995.

[14] M. Papathomas, G. S. Blair, G. Coulson, and P. Robin. Addressing the real-time synchronization requirements of multimedia in an object-oriented framework. In *Multimedia Computing and Networking 1995*, volume 2417, pages 190 – 201, San Jose, California, SPIE, February 1995.

[15] M. J. Pérez-Luque and T. D. C. Little. A Temporal Reference Framework for Multimedia Synchronization. *IEEE Journal on Selected Areas in Communications*, 14(1):36–51, January 1996.

[16] S. Ramanathan and P. V. Rangan. Adaptive Feedback Techniques for Synchronized Multimedia Retrieval over Integrated Networks. *IEEE/ACM Transactions on Networking*, 1(2):246–259, April 1993.

[17] K. Rothermel and T. Helbig. An adaptive protocol for synchronizing media streams. *Multimedia Systems*, 5(5):324–336, September 1997.

[18] 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*, pages 229–239, Singapore, November 1995.

[19] 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*, pages 190–201, Durham, New Hampshire, USA, April 18-21, 1995.

[20] 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*, pages 31–40. IEEE Computer Society Press, May 15-18, 1995.

[21] R. Steinmetz. Human perception of jitter and media synchronization. *IEEE Journal on Selected Areas in Communications*, 14(1):61–72, January 1996.

[22] C.-C. Yuan and J.-H. Huang. A Multimedia Synchronization Model and Its Implementation on Transport Protocols. *IEEE Journal on Selected Areas in Communications*, 14(1):212–225, January 1996.

[23] P. N. Zarros, M. J. Lee, and T. N. Saadawi. Interparticipant Synchronization in Real-Time Multimedia Conferencing Using Feedback. *IEEE/ACM Transactions on Networking*, 4(2):173–180, April 1996.

[24] M. Zitterbart, B. Stiller, and A. Tantawy. A Model for Flexible High-Performance Communication Subsystems. *IEEE Journal on Selected Areas in Communications*, 11(4):507–518, May 1993.