Low-Level Multimedia Synchronization Algorithms on Broadband Networks

Miguel Correia , *Paulo Pinto*

Instituto Superior Técnico, Av. Rovisco Pais, 1 P-1000 Lisboa, Portugal
INESC, R. Alves Redol, 9 P-1000 Lisboa, Portugal
+351 1 3100301
{mpc,pfp}@inesc.pt

ABSTRACT

As general purpose non-real-time systems start to be able to capture, present, store and transmit multimedia information, the need for low-level synchronization algorithms for continuous media transmission arises as a requirement for desktop applications such as video-conference and video-on-demand. This paper presents a complete scheme to obtain such synchronization with an intended quality of service (QoS). Algorithms handle both intramedia and intermedia synchronization and act progressively as the problems get more serious. A conditional retransmission mechanism adequate for multimedia information is defined. An adaptive QoS degradation scheme is proposed as the solution for CPU heavy loads or network congestion. The heuristics to calculate the algorithm parameters when using "unknown" networks are sketched. The QoS values obtained are discussed.

KEYWORDS

Low-level synchronization, Distributed multimedia systems, Broadband Networks.

1. INTRODUCTION

The ability of computers and networks to handle high throughput data, and the existence of specific hardware (and software) to capture and present video and audio, are introducing this type of information into the application area. Their handling is becoming as easy as manipulating text and still pictures. Systems composed of "fast" workstations with video and audio hardware, and connected by a broadband network are called "distributed multimedia systems".

However, several problems exist due to the need of using non-real-time operating systems and asynchronous networks, i.e. best-effort systems. The sensible approach to solve the problem is to define a set of conditions that ought to be preserved during the overall handling of the isochronous signals. This set, called "quality of service" (QoS), is composed of several parameters that define quality levels required for the presentation of the data.

This paper addresses algorithms to meet the QoS parameters related with the synchronization of isochronous (or continuous) media. It is broadly accepted to divide the synchronization problem into high level (extrinsic) synchronization; and low level (intrinsic) synchronization. The first refers to the media presentation control problem and is related to the object model and the composition concepts of the basic parts of the objects (when they begin, end, etc.). There are already some standards (HyTime[12], MHEG[7]) and also other academic contributions ([19],[10] and [14]).

Low-level synchronization, which will be the subject of this paper, is concerned mainly with two problems: the continuity of a medium presentation (intramedia synchronization); and the larger problem of synchronization between two or more signals (intermedia synchronization), transmitted from one origin to one destination, from several origins to one destination, or from one origin to several destinations.

2. MOTIVATION AND OVERVIEW

2.1 Problems

In a real-life distributed multimedia system there are a number of aspects that require an explicit low-level synchronization.

(a) Networks introduce transmission delay jitter (delay variation). The existence of this jitter means that the portions of data are not received at a constant rate, and consequently, cannot be presented as soon as they arrive. Another relevant and equivalent jitter is the one introduced by the variation of the transmission instant at the transmitter.

(b) Workstations have different clock rates. This mismatch is modeled with the concepts of skew (the frequency difference) and drift (the difference variation). It can cause the loss of continuity between the transmitter and the receiver, or the loss of intermedia synchronization between different transmitters. Values measured can go as high as $30x10^{-6}$ Hz. An additional skew component is introduced by the granularity of time measurement. This component can reach higher values than clock rate difference itself.

If drift did not exist, a frequency adjustment could be performed only once to compensate the skew (the frequencies difference). Otherwise a permanent running mechanism is necessary.

Skew and drift can be avoided by the use of clock synchronization protocols, such as NTP [11]. However, this is not desirable for two main reasons: because specific mechanisms (like the one that will be described) are more simple and light; and because if the workstations belong to different entities, they may not accept total synchronization.

(c) Instantaneous CPU loads can cause CPU unavailability when time constrained actions such as information presentations are due. A non-real-time CPU can never guarantee an instant so the algorithms consider (reasonable) intervals. These intervals are defined by a beginning instant and a tolerance.

(d) Networks and CPUs can have highly variable loads. A CPU heavy (and not instantaneous -- (c)) load can prevent the synchronization and presentation processes from having time to do their job, causing discrete losses on the stream. Network heavy loads and congestion can prevent the presentation process from having information available to be presented when needed.

(e) The last network relevant effect is only related to intermedia synchronization. When different workstations are sending signals to be synchronized, the average propagation delays may differ considerably due to the effect of delays introduced in intermediate bridges, routers, etc. This difference should be taken into account when the process is started, to avoid the reception of information from one origin in advance of the information from the other sources, as it would lead to prohibitive amounts of buffering.

2.2 Algorithm Characteristics

This paper describes an algorithm to solve the problems just listed with the following characteristics:

(1) It works at the reception instants so the corrections to clock frequencies that have to be made can have a better precision than in schemes based on buffer fill levels, much used in the references. It assumes a fixed delay from reception to presentation of data.

(2) It has different levels of action to handle the problems according to their nature: delay jitters are controlled by a conventional buffering scheme; clock frequency differences are treated by period adjustments (the same as frequency adjustments); instantaneous loads are dealt with information discarding; and heavy loads are treated with QoS degradations. This progressive scheme provides low overhead when it is not needed.

(3) It uses different network channels for each stream of data. The use of a single multi-stream connection, used by MPEG [6], avoids some of the difficulties just described (it does not need an explicit intermedia synchronization). However, the option of different connections makes sense because it is a more general solution allowing for the generation of data in different computers; because different types of streams have different requirements towards resource reservation [13]; and because they have different sensivity toward QoS.

(4) The system has two operation modes: "interactive" and "non-interactive". "Interactive" is used, for instance, on conference systems, that capture the information in run-time and need a low latency. "Non-interactive" is used, for example, on video-on-demand systems, that can afford to have longer start up delays (latency) to achieve better QoS levels. Systems using this last mode transmit information stored in a disk or similar device, so it is possible to advance or delay the whole presentation process.

The first algorithm described in the paper deals with intramedia synchronization. This kind of synchronization guarantees that one channel's information is received before the end of its presentation interval. It also guarantees that the reception and presentation frequency are similar, to avoid buffer overflow or underflow. The second algorithm handles intermedia synchronization. This synchronization has to deal with skew and drift between the transmitters and different average delays between each transmitter and the receiver.

2.3 Final Considerations

Usually, multimedia information does not need a hard real-time handling. Small and temporary QoS losses, such as a few milliseconds of sound or a single image, are not disturbing to human senses. However, some compression algorithms make some information more important than others. A selective discarding method would have to be used for these cases (not considered here). On the other hand, to guarantee that all information is presented on time, real-time systems (like the ones in most of the references) have to consider the worst-case situation, leading to undesirable large buffers and end-to-end delays. Even if this was done for best-effort systems, it would not guarantee 100% of the QoS required because the non-real-time operating system does not guarantee presentation (or other) instants, and because the non-real-time network does not guarantee maximum transmission delays. Consequently, the algorithms described do not work with worst-case values but with average ones that allow QoS level violations with some probability. Increasing or decreasing the latency and buffering decreases or increases the presentation probability. It is interesting to state the fact that on the system used in practice, about 99% of the jitter values are smaller than 1% of the maximum value measured. This allows reasonable QoS values without an excess of resources.

3. RELATED WORK

Intramedia synchronization algorithms that treat jitter and skew can be found in [4]. A worst-case scenario is assumed, with the problems just referred. The pre-fetch amount of data and the maximum and minimum buffer fill levels to detect the existence of skew are deduced formally using the value for the maximum jitter. Rate matching is based on the values for input and output period times which are difficult to estimate. The skew control scheme is based on the buffer usage, proposed originally by [5], but with a significant difference: the cells considered in [5] have 53 bytes and, consequently, a high rate; the messages considered in [4] have at least some hundreds and, consequently, a much lower rate. The paper considers a real-time operating system and so it does not consider the problems caused by machine and network loads and does not address intermedia synchronization.

A value for the maximum jitter was also used in [17] on a proposal for video-on-demand like systems. The topology considered is not typical: one transmitter and several receivers. Intramedia synchronization is solved with pre-fetch and buffering, using a feedback technique for rate adaptation based on light-weight messages sent by the receiver after receiving selected messages. The rate at which feedback units must be sent to maintain continuity is deduced from the round-trip delay. Intermedia synchronization uses one stream as the master and Relative Timestamps defined at recording time.

The closest proposal to this paper can be found in [1]. Jitter is again compensated by pre-fetch and rate matching is done by "skipping and pausing" units of data. There are no smaller adjustments than these. The handling of continuity of streams is performed by mapping the data units onto a Logical Time System (LTS) instead of sample-level synchronization. Intermedia synchronization is achieved by considering one of the streams as a master and skip or duplicate units on the slaves to keep them synchronous with the LTS. In case of network congestion the system blocks, so this scheme is not very adequate for interactive transmission. TCP was chosen for the transport protocol causing delays and increase of traffic due to retransmission when there are losses.

[22] presents a scheme for network congestion control, based on bit and packet rate scaling, in a non-real-time system. This scaling means a QoS degradation. The problem of heavy loads in the CPUs is not addressed. The key problem is to find sustainable values for those rates, with an acceptable QoS, during a congestion.

[16] presents schemes and policies to achieve intramedia and intermedia synchronization based on Petri Nets. The system is assumed to be real-time and the heavy loads' problem is addressed. [13] gives a two level synchronization scheme that allows the application to specify the synchronization granularity that it can better handle. [8] presents some algorithms for a real-time system. It uses the synchronization channel proposed in [18] and achieve synchronization at the level of synchronization units (SU). The path from low-level synchronization concepts to high-level ones is the following. Several SUs from different streams form a group for the purpose of time constraints. A sequence of SUs in time form an activity which is an application concept. In between there are the transport level concepts of segment to cope with transmission synchronization.

The ATM AAL 1 [5] layer defines algorithms to adjust the receiver clock rate to the transmitter one. Such algorithms will be used to control skew and drift at a very low level, immediately above the ATM layer. This is not of a great help to the problem in hand for several reasons. It does not take care of jitter or loads introduced by the operating system and cannot take any action, such as reducing the QoS, if there is a heavy load or congestion in the machine. Another problem is that the ATM reception clock may be different from the presentation one, introducing a new skew/drift between these clocks. A better approach is to renounce the services of AAL1, and have a new algorithm working as near as possible to the presentation interface and to use ALL5 or AAL null over ATM.

4. INTRAMEDIA SYNCHRONIZATION

Some considerations about the transport protocol, retransmission mechanisms and temporal model of the system are given before the description of the receiver and transmitter roles. The last section focuses on the determination of the parameters.

4.1 Transmission Protocols and the Retransmission Mechanism

The synchronization algorithms are encapsulated in a synchronization layer that has two independent functions: transmit and receive. In normal operation mode, one side is the transmitter, the other side the receiver. They communicate by sending packets as shown in fig. 1. There are two kinds of packets: information and control. The basic unit of data is called an *information unit* (IU). A video IU is a frame; an audio IU is a set of samples correspondent to a time interval. An information packet can carry an IU or a fraction of it. The current implementation sends one audio IU in one information packet and a video IU in more than one (When one or more packets of a multi-packet IU are lost, the decision on the use of the incomplete IU is transferred to the application). There are five control packets: START - to ask for data transmission and negotiate the initial QoS values; ECHO - to measure the round-trip delay; REQ - to request a retransmission; CHNG - to perform a QoS degradation or to change the transmitter clock period; and RESET - to reinitialize the communication. ECHO packets are sent periodically by the receiver and returned by the transmitter. The time of the "trip" is used to get a round-trip delay average value (greater weights are give to recent measures).

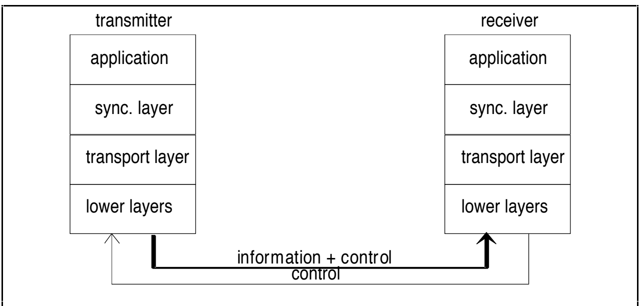

Figure 1. Communication process.

The synchronization layer uses the transport layer services. The transport protocol should be connection oriented as the data is to be transmitted continuously. The protocol may have rate control and error detection but should not have error recovery or flow control, such as XTP [21]. Another good solution is to use the place holders offered by RSVP [23] to put the synchronization information produced by the algorithms. As XTP was not available, a trivial connection establishment and termination mechanism was implemented over UDP. The disadvantages were the overhead caused, at transmission, by the address resolution procedures (and bandwidth used by the addresses in the packet), and the inexistence of rate control.

Error recovery mechanisms can cause indiscriminated retransmission of data when there are losses due to congestion or traffic policing. This is not a good feature for multimedia data (see below). This is the reason why the use of TCP is not acceptable in this cases. TCP has also the disadvantage of using a go-back-n retransmission scheme, that causes the transmission of redundant data. Nevertheless, if possible, a retransmission should be attempted in case of error, because it is important to present as many IUs as possible (it is a QoS parameter). This means that a conditional retransmission mechanism can be an adequate solution for multimedia transmission.

A late IU (an IU that arrives after its presentation interval) is as good as if it was lost. So, the first condition to attempt a retransmission is the existence of time to get the IU before its presentation interval. The second condition is the inexistence of network or CPU heavy loads (this detection is analyzed below). The evaluation of the existence of time is given by:

$$t + d_{echo} < t_p(n_p) + (n - n_p) \times T(n)$$

where t is the present instant; d_{echo} is an estimation of the average round-trip delay based on the measures done with the ECHO packets; n_p is the number of the IU to present next; n is the number of the lost IU; $t_p(n_p)$ is the presentation instant of the unit n_p; and $(n - n_p) \times T(n)$ is an estimative of the time until $t_p(n)$ (T(n) is the period correspondent to the current IU, that serves as an estimation of the T corresponding to the next IUs (T can change if the IU size changes)).

The receiver knows that an IU was lost when it receives a subsequent one. The lost IU can either have been dropped by the transmitter (there is an indication in the received IU), or lost in the network.

4.2 Time Behavior Considerations

The synchronization layer assumes the existence of higher layers to read or present data (fig. 1). At the receiving end they are modeled as introducing only a delay and a corresponding jitter, i.e., the IUs to be presented are delivered to the higher layers synchronously. At the transmitter end they are modeled in two ways: a synchronous model similar to the one for the receptor; or an assynchronous model, for blocks of IUs to be read from a disk-like device. In this latter case, data is requested by the synchronization layer when needed and the device tries to deliver it in a limited time interval.

An important concept is the one of a *presentation instant*, the instant when an IU should be delivered to the higher levels. Some other presentation instants can be defined, such as the real presentation instant (when the output is really performed) or the expected presentation instant (when the output should be really performed). These concepts will not be considered because they are difficult to relate to the algorithms defined. The presentation instant of the nth IU, $t_p(n)$, in terms of the period T(n) is given by:

$$t_p(n) = t_p(n-1) + T(n)$$

It was already said that intervals are used instead of instants. Thus a valid presentation instant is any point of the interval:

$$[t_p(n), \quad t_p(n) + \xi]$$

The value ξ is the presentation instant tolerance and depends on the media device (or the application). In the audio case the (minimum of the) value is given by:

$$\xi = \Delta buf - \Delta p$$

where Δbuf is the audio time buffered in the device (or application) and Δp the time taken to "present", i.e., approximately the time taken to copy the information to the device buffer. For the video, ξ is the maximum acceptable presentation jitter of a frame (a QoS parameter).

4.3 Intramedia Mechanisms at the Receiver

This section describes the intramedia algorithm, with special focus on the mechanisms at the receiver (i.e., the main part of the algorithm). The mechanisms used to control each of the problems referred in section 2 are described in that same order. A stronger mechanism -- reset -- is described. It is used when serious situations with a total loss of synchronization happen. The differences between the running modes, "interactive" and "non-interactive" are also depicted.

The problems to be solved are in increasing order of seriousness: (a) transmission jitter; (b) transmitter and receiver clocks skew; (c) instantaneous heavy loads; and (d) non-instantaneous heavy loads.

The problem (a) is neutralized using the standard procedure of pre-fetching [1, 4, 17, etc]. The pre-fetch time value is equal to the sum of the transmission jitters introduced by the network and the transmitter. This value is adjustable, causing a better or worse presentation probability. In systems working in "non-interactive" mode longer pre-fetch intervals are acceptable.

The problem (b), skew, causes a systematic delay or advance in relation to the "expected". A non-instantaneous heavy load, (d), causes a delay to several consecutive IUs. Although these two effects are very different in nature, it is not simple to detect and distinguish them due to the influence of delay jitter.

To distinguish between the three problems (a), (b) and (d), an *expected reception instant* is defined. This value is initialized with the first reception instant and the other ones are calculated summing the period:

$$tr_{exp}(n) = tr_{exp}(n-1) + T(n-1)$$

The average of some consecutive values of the difference between the reception instant and expected reception instant, *dif*, is used to make the distinction:

$$dif(n) = tr(n) - tr_{exp}(n)$$

If the average belongs to $[\lambda, \alpha]$ or to $[-\alpha, \lambda]$ a "false skew" is detected, i.e., an effect that may be just a jitter that affects several IUs or a small skew (fig. 2). If $avg_{dif} \in \,]\alpha, \beta]$ or $avg_{dif} < -\alpha$ a skew is detected. This distinction between skew and "false skew" will be used to define two levels of corrections. If the average is bigger than a value β the existence of heavy loads is assumed (no negative values are considered because a heavy load cannot accelerate an IU arrival). If $avg_{dif} \in \,]-\lambda,\lambda[$ no action is done because the difference of the values in relation to 0 is considered just jitter.

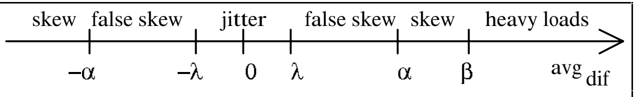

Figure 2. Intramedia effects detection.

The average is calculated with a fixed number of values, Navg. After a detection, no other test is done before Navg new values are available, to avoid several consecutive detections of the same problem without waiting for the correction action. λ is defined considering the maximum jitter allowed. α is a value between λ and β, for example: $\alpha = (\lambda + \beta) / 2$. A good β is more difficult to calculate theoretically. Two principles can be used: to consider the maximum skew allowed; a consider the maximum delay so that the IUs still arrive before their presentation instant. The formulas used, but not deduced here are (considering a constant period T):

$$\beta = \alpha + S \times T \times \frac{Navg - 1}{2}$$

and

$$\beta = P \times T$$

being P the number of IUs pre-fetched and S the skew. An average of the two principles gave good results in practice.

There are three ways to correct the skew:

(1) Adjusting the presentation clock frequency to the reception rate (equal to the transmitter frequency).

(2) Adjusting the transmitter clock rate to the one of the presentation, using control messages sent by the receptor (packet CHNG).

(3) Emulating the transmitter clock rate skipping or duplicating IUs at the receiver.

The second solution was chosen because it is more adequate for the intermedia synchronization case, for which an adjustment of the transmitting rates of all the transmitters to the receiver rate avoids the existence of skew between the streams. The packet CHNG is used to instruct the transmitter.

A good value for the rate correction could be obtained using linear regression, but it is too CPU time consuming. An acceptable but not very exact correction can be calculated with a simpler formula:

$$\Delta f = -signal(avg_{dif}) \times \frac{|avg_{dif}| - \alpha}{Navg}$$

A correction to the transmission instant (the beginning of the corresponding interval) is also performed, to compensate the offset existent at the moment:

$$\Delta t_t = -avg_{dif}$$

The "false skew" detection is used to avoid this mechanism to do its job before there is some certainty about the really existent problem. If a "false skew" is detected a mechanism similar to the one used for the skew is activated but just with local (receiver) implications: a correction is performed on the presentation and expected reception instants. The value of the correction is identical to the one used for the correction of the transmission instant:

$$\Delta t_p = -avg_{dif}$$

This value is summed to an accumulator (*ac*). If this accumulator gets off some bounds $[-\alpha, \alpha]$ it is reset and a skew correction is performed. On the other hand, the accumulator is reset whenever a skew correction is performed. The value for the rate correction is given by the same formula as before but exchanging α for λ.

The video presentation mechanism is (usually) asynchronous due to its "low" presentation rate: maximum of 30 frames per second. So, a correction of the presentation instant just readjusts the moment the driver is called. The audio case is more complex because the rate is too high for the operating system to present each sample asynchronously. In consequence the audio devices have an internal clock and a buffer, so a block of samples (some hundreds, for example) are delivered simultaneously. A correction of the presentation instant implies the presentation of more or less samples than the available. So, samples have to be inserted or deleted. The solution adopted is to try to detect a silence period and decrease or increase its duration. If no silence period is detected, the correction is performed duplicating or deleting the last samples of some blocks.

About the problem (c), instantaneous heavy loads, the presentation intervals are supposed to be met by the algorithm at all times. Nevertheless, the influence of temporary or non-temporary heavy loads may cause the reception of an IU after the end of the presentation interval (or the unavailability of the CPU when the IU should be presented). In these cases the IU is simply discarded.

The problem (d), non-instantaneous heavy loads, is controlled using a QoS degradation scheme[1]. The objective is to decrease the load caused by the synchronization and presentation systems avoiding a total congestion and so allowing the system to go on working (although with a lower QoS). When a heavy load is detected a degradation is performed to the adjacent lower QoS level. If the load goes on being detected, another degradation is performed and the situation is repeated until the lowest level (applications are warned each time in order to take measures). The inverse mechanism is described in the next section because it is local to the transmitter(s).

The quality of service is dealt with in more detail in section 6. A first approach for the definition of QoS levels is just the increase of the size of the IUs. This scheme can be adequate to decrease the load in the CPUs, because the period gets bigger and so the constant processing time operations are realized less times per time unity. This was tested mainly with audio because M-JPEG was used for video and its IUs were already bigger than network packets. The network load is not affected but usually the video throughput is much higher and maintaining the audio quality is more important. The CHNG packet is used to set the new QoS level.

The values of the QoS levels should have a relation to the heavy loads that can affect the system. For example, if only four levels of heavy loads can affect the system, four QoS levels should be defined so the performance of each one of them could control the effects of the corresponding heavy load. In practice there is no sense in talking about discrete load values, so the QoS levels have to consider two arguments: the levels should be distant enough to avoid too many degradations to get to the desired result; they should be close enough to avoid degradations bigger than necessary.

One last aspect to consider is the reset action. Sometimes, if a heavy load remains for a "long" time (1 second, for example) the reception buffer can overflow or underflow in relation to the IU to present. In consequence IUs are not presented. If this happens a reset action will be performed. The reset is always performed by the transmitter but sometimes it is triggered by the receiver.

In "interactive" mode the reset maintains approximately the end-to-end delay and the trade-off is that some IUs are not presented. In "non-interactive" mode the transmission restarts at the moment where it was blocked adjusting all the instants to the new time-scale. So, in this mode, the reset will restart the presentation where it was interrupted.

4.4 Intramedia Mechanisms at the Transmitter

At the transmitter side, the mechanisms are just a complement to the receiver side. Their functions are (in increasing order of seriousness):

(1) Clock rate adjustment when ordered by the receiver: problem (b)

(2) Reaction to instantaneous heavy loads at the CPU: problem (c)

(3) Reaction to non-instantaneous heavy loads at the CPU and others pointed out by the receiver, i.e., heavy loads at the receiver and network: problem (d). The increase of the QoS level to the initial values after a degradation is also made here.

When the receiver detects a skew, it sends a CHNG packet to the transmitter to change its rate, (1). This action is performed changing the transmission period. A practical difficulty found is caused by the granularity (or precision) of the system call used to "sleep" after each cycle of operations -- the Unix usleep instruction. This granularity (10 ms for this instruction) causes that a small correction made to the period has only an average effect: the period between each pair of IUs remains the same, except for one pair once in a while. This causes a considerable skew that can seriously prevent the receiver from detecting the really existent problem: jitter, skew or heavy loads. The solution used is to accept this effect but to send in the (first packet of each) IU the shift in relation to the desired transmission instant. The receiver subtracts the shift when it calculates the difference *dif*. Each time consecutive corrections sum the granularity of usleep a real correction is performed.

The reaction to instantaneous heavy loads, (2), is similar to what happens at the receiver side. There is a transmission instant for every IU and a tolerance, *tol*, to define the corresponding interval. The mechanism is

[1]Applications establish connections by stating the desirable QoS and some acceptable lower QoS levels. This is ouside the scope of this paper.

triggered when the synchronization level does not manage to transmit the IU after the end of the interval. In "interactive" mode the IU is discarded, i.e., not transmitted. The next IU transmitted will carry an indication of this action. In "non-interactive" mode a higher latency is acceptable so a longer tolerance is defined, *tol1*. The violation of this tolerance causes the IU discarding, as for the "interactive" mode.

If Navg consecutive violations of the tolerance *tol* (in both modes) are detected, the existence of a CPU heavy load is assumed, (3), and a transmitter lead QoS degradation is performed. If several IUs transmission instants are missed the reset procedure explained early for both modes is due.

The QoS recuperation operation is not simple to perform at the convenient moment because it is not easy to detect when a heavy load finishes. The solution adopted is to use a timeout. The timeout value has to consider both the average heavy load duration and the number of levels defined. As this is not possible to estimate, a reasonable value had to be determined in practice. Each time there is an interval without further degradation, the QoS level is increased until it reaches the initial level. This mechanism is local to the transmitter and no action is taken by the receiver, except if the application requests a QoS level change directly.

4.5 Automatic Determination of the Parameters

The algorithms described involve several parameters that have to be calculated for each practical system. These parameters are: skew and heavy load detection limits (α and β), number of values used to calculate the averages, reception buffer dimensions, transmission and presentation interval lengths (tolerances).

These parameters are obtained with three sets of values: (1) system characteristics values such as the maximum transmission delay jitter considered, the maximum skew, the network latency and the bandwidth; (2) QoS parameters values such as the desired latency, the presentation jitter, and the presentation probability; and (3) values related to the application such as the IU length and the period.

The second and third sets of values are defined when the applications are designed. The first set depends on the real system available: hardware, operating system and network. Some of them, the bandwidth and the latency, cause direct restrictions to the system performance and have to be considered before defining the QoS that the application will negotiate for the connection. The maximum skew can be defined in advance as a small value, 1% for example, because an exact value is not very important. The maximum transmission delay jitter is a key value because it is used to calculate important parameters, such as the pre-fetch interval and the reception buffer dimension. This value is measured before the beginning of the operation, using the ECHO packets to measure the round-trip delay.

5. INTERMEDIA SYNCHRONIZATION

The objective of intermedia synchronization is to keep intrinsic time relations between two or more media. Some media may need a "strict" intermedia synchronization, such as lip-sync between the audio and the video of a film, or just a "loose" synchronization, in the case of the sound-track of a movie about nature. This kind of synchronization has to treat the skew and drift between transmitters and the initial reception mismatch if the propagation delays are different, (e).

Intermedia synchronization can be thought as a layer that uses continuous channels given by an intramedia synchronization layer. So, intramedia synchronization is done in every channel (fig. 3).

For "interactive" applications there is little room for intermedia synchronization because all streams must meet the real time. Experiences have proven that independent intramedia in each stream solves the problem, if all clock rates are synchronized to a single one (section 4).

For "non-interactive" applications intramedia synchronization can make the presentation smother with regard to the problems. The trade off is the extension of the presentation time in relation to the real duration of the media. One stream is considered to be the master and the other(s) the slave(s). The master transmission rate is always adjusted to the presentation by the intramedia algorithm (the master should be a media with more strict QoS values, usually the audio). For the slave, the rate is adjusted only when necessary, i.e., the intramedia rate adjustment mechanism (used to correct the skew) is disabled (the "false skew" corrections are still enabled as they are local to the receiver). So it is possible to define a mechanism that acts on the slave(s) transmission rate(s) only when needed, according to the parameters defined for the QoS. The algorithm is applied over each slave-transmitter pair (with the master transmitter and the receiver) so the case of only one slave is described.

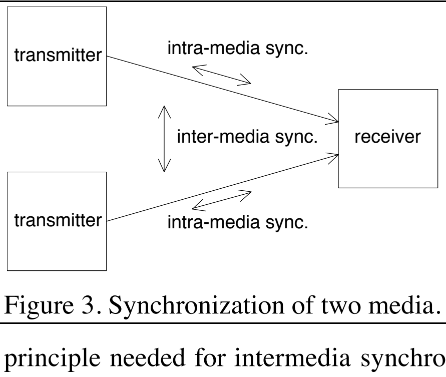

Figure 3. Synchronization of two media.

A basic principle needed for intermedia synchronization is the correspondence between the IUs of both channels. Two corresponding IUs should be presented with an offset lower than an allowed maximum value (a QoS parameter). The relative presentation instant of an IU is derived from a timestamp. There are several formats possible for the timestamps. They can have a time format (microseconds, milliseconds, etc) or a non-time format (a packet number, an IU number, etc). The format chosen for IUs with constant length was the IU number because the time interval of an IU is known and it can be represented with a small number of bits (one byte for example). If the IU length is variable, the timestamp unity is the maximum divider of the possible lengths.

To maintain the intermedia synchronization the algorithm must keep the reception instants within certain bounds. So, the measure of synchronization (for each slave channel) will be the average of the difference

$$\text{dif1} = \text{tr}(n_s) - \text{tr}(n_m)$$

where $\text{tr}(n_m)$ and $\text{tr}(n_s)$ are the mapped reception instants of the master and the slave. *dif1* is calculated for each IU arrived on the slave channel. The average is calculated over the last Navg1 values.

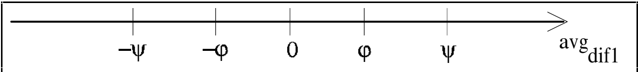

Figure 4. Intermedia effects detection.

The algorithm is similar to the intramedia one (fig. 4) in the sense that it acts according to threshold values of the average:

(1) If the average is small but bigger than φ, $|\text{avg}_{\text{dif1}}| \in [\varphi,\psi]$, there is the danger of violating the maximum allowed offset. So a delay or advancement of the slave presentation has to be performed. This also means to decrease or increase the presentation instants.

(2) If it is higher, $|\text{avg}_{\text{dif1}}| > \psi$, there is a violation of the maximum offset or IUs are not presented. So, a reset has to be performed: the whole presentation is delayed, and the offset is reset to 0.

The meaning of the parameters ψ and φ is deduced from the statements above: ψ is the maximum acceptable offset value; and φ is a smaller value that means a danger of violating the maximum offset.

A less important intermedia mechanism is related to the different average transmission delays of each stream, caused by the existence of routers, bridges, etc. Some ECHO packets are sent before the transmission of data a the answers collected to make an estimation of the round-trip delays. Afterwards, an order to the transmitters to begin the transmission is sent. The orders are sent considering the delay differences so that the IUs from the different transmitters will start arriving approximately at the same time maintaining the filling level of the buffers balanced.

6. QUALITY OF SERVICE

It should be clear now that there are two types of QoS parameters: some direct ones handled by the algorithms; and some indirect ones, at the level of the application, which influence the overall performance. The direct QoS parameters are: (1) presentation jitter; (2) end-to-end delay; (3) presentation probability; (4) temporal distortion; (5) intermedia presentation offset.

T indirect ones are the sample rates of video and audio, the quality of the video compression algorithm or the coding algorithm for audio. These parameters are related with the QoS levels and are most of the time constrained by the hardware. Their influence to the overall system will not be discussed here and will be subject of a future report.

The presentation jitter, (1), is not limited because of the non-deterministic delays the operating system can introduce anytime before an operation. The same approach as for the network jitter was used: set the value equal to the presentation interval and accept a probability of violation. In general terms, this value should be long enough to maximize the presentation probability, and small enough so the overall jitter is acceptable. A good compromise was obtained with a 20 ms interval: the probability was high and the jitter not noticeable.

The end-to-end delay, (2), can be proved to be maximized by:

$$l_{max} = d_{max} + d_p + d_a + \alpha$$

where d_{max} is the maximum transmission delay; d_p is the pre-fetch interval; and d_a is the delay between the presentation instant and the real presentation. Assuming a 50 ms pre-fetch interval and α = 10 ms a maximum latency of 82 ms was calculated. This value is very good both to "interactive" and "non-interactive" systems. For systems such as video-conference 150 ms is usually considered a good value.

The presentation probability, (3), is highly influenced by the system load. 100% values were obtained without heavy loads and values as low as 0% were obtained with very heavy loads.

Temporal distortion, (4), exists due to the difference of clock rates at the transmitter(s) and the receiver (real rates, the nominal ones are *always* the same). The values obtained are not detectable due to the very low difference. To achieve intermedia synchronization it was accepted as a principle that every media is captured at the same rate. The presentation rate may be (slightly) different. The video presentation mechanism is (usually) asynchronous so it poses no problems (v. 4.3). The audio presentation is synchronous (v. 4.3). So, the different clock rates implies that the number of IUs delivered to the device to be presented will be played during an interval slightly longer or shorter than the desired one. This effect can be neutralized by the receiver, inserting or deleting a few samples from time to time. This procedure is independent from the algorithms and must be performed by the receiver before the algorithms get the data. This number is easy to calculate given the difference of the rates. The operation can be done as referred in 4.3.

The objective of intermedia synchronization is to keep the intermedia offset within bounds, (5). [20] determines a maximum value of 80 ms for this parameter for the case of lip-sync. Two mechanisms were given for "interactive" and "non-interactive" systems. For "interactive" systems, the offset is limited by the imposed maximum presentation jitter (given by *tol*). For "non-interactive" systems, intermedia synchronization guarantees that this parameter does not get off an interval [-ψ, ψ]. So ψ has to be defined in accordance.

7. EXPERIMENTAL RESULTS

This section presents three examples that show the main features of the algorithms. The first two are intramedia synchronization examples of an audio stream transmitted between two workstations. The third shows intermedia synchronization. The distributed system included three Sun Sparc 10 workstations with Sun OS 4.1.3 and a LAN ATM network (thus, with a very limited jitter). The audio stream used is a speech sampled at 8KHz (ulaw). The QoS degradations are performed simply by IU size augmentation. The initial IU size is 50 ms (400 bytes) and the maximum size allowed is 150 ms, with a 20 ms variation. The experiments where made in "non-interactive" mode with a pre-fetch value of 3 IUs.

7.1 Experiment 1

The first experiment shows the system normal operation under an imposed considerable skew. The parameters' values used were λ = 2 ms, α = 10 ms and β = 20 ms. 10 values were used to calculate the average. The sound quality obtained was very good.

The value of *dif* (fig. 5) starts to increase influenced by the skew. After some time, around IU number 30, the skew is detected as the average of *dif* crosses λ and a local correction is made (fig. 6). The correction is local because it is detected as a "false skew", i.e. the receiver is still not sure that there is a real skew. There are some spikes (high delay values) caused by some network or machine effect.

The local correction does not change any rate so the influence of the artificial skew remains. So, the value of *dif* continues to increase. Around IUs number 150, 200 and 250 the skew is detected again and local corrections are made.

Around IU number 300 the receiver detects the skew again. This time it considers the existence of a real skew because the sum of the corrections already made (stored in the accumulator *ca*) with the one that would be done this time, is higher than α. So a correction is made to the transmitter rate.

After this remote correction, the influence of the skew is lower, but it is still felt. This is a consequence of the formula used for the corrections not being very good (as mentioned above). Nevertheless the problem gets better and after some future corrections it gets really small.

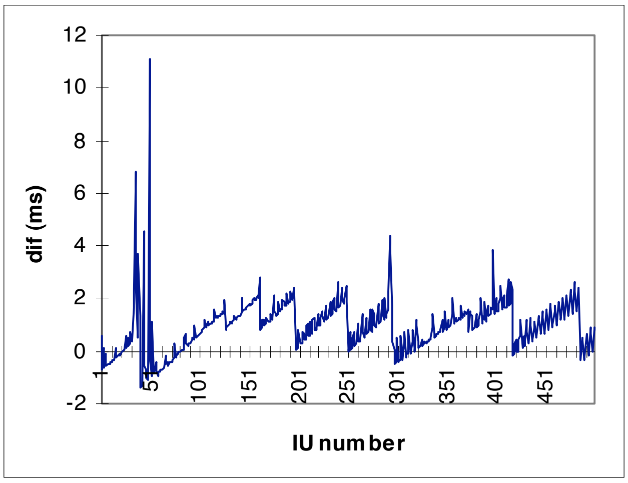

Figure 5. dif(n) chart

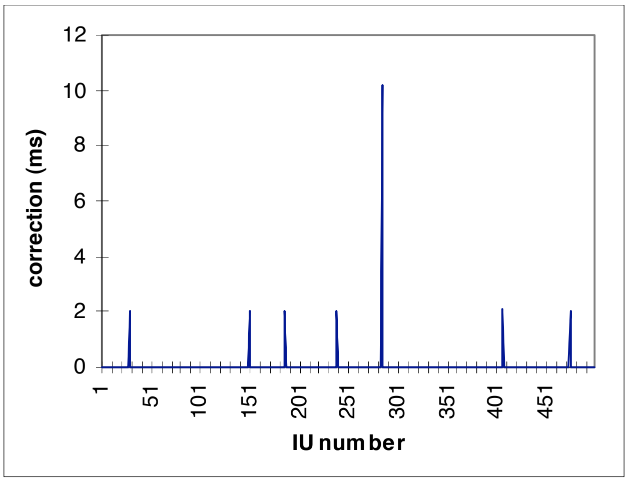

Figure 6. Chart of the corrections made in the receiver (lower than 10 ms) or in the transmitter (higher than 10 ms).

7.2 Experiment 2

T second experiment shows the influence of a heavy load at the receiver. The problem happened around IU number 50 caused by the beginning of several CPU time consuming programs that display charts.

The problem caused some correcting reactions: instantaneous heavy load corrections (some IUs are discarded) and QoS degradations (fig. 9). It also caused some wrong reactions like "false skew" and skew corrections (fig. 8).

Near IU number 60 a heavy load was detected and a QoS degradation was made. Some time after the transmitter decides to try to bring the QoS to the initial values. The attempt was made too early and some milliseconds later another degradation is performed. The transmitters does the same thing again and the receiver manages to keep the QoS for about one second (around IUs 80 to 100). Then it performs another degradation and a little later another one. After the end of the heavy load the systems gets stable and the QoS returns to the initial values.

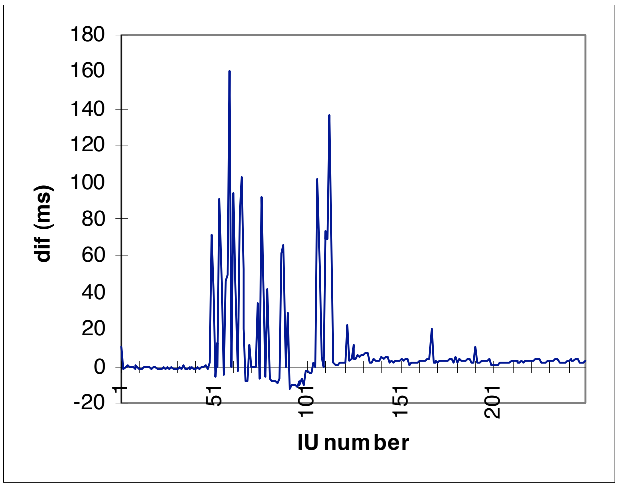

Figure 7. dif(n) chart

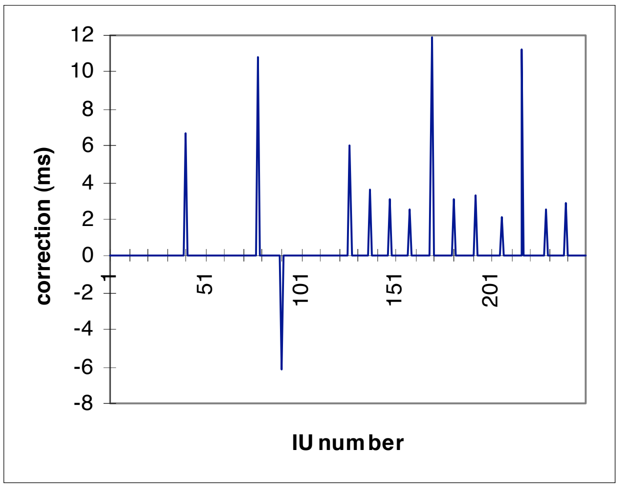

Figure 8. Corrections chart (see fig. 6)

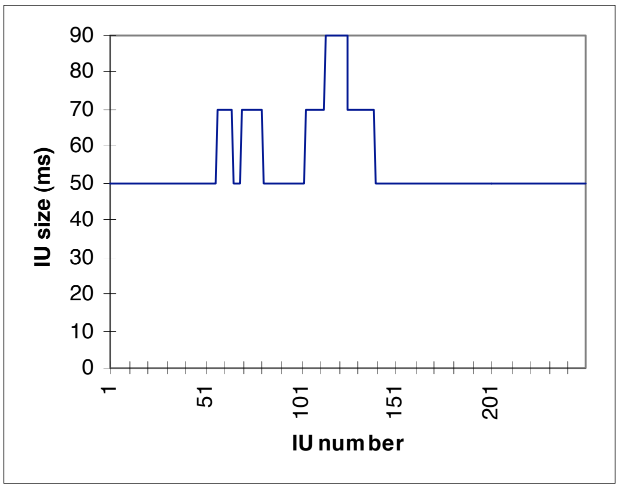

Figure 9. IU size chart (QoS degradation)

7.3 Experiment 3

The last experiment shows intermedia synchronization. The intramedia parameters values used were the same as before and the inter-media parameters were: φ = 10 ms and ψ = 60 ms. The number of samples used to calculate the average *dif1* was 10. There was a considerable transmitter-transmitter skew introduced artificially at the slave transmitter to test the algorithm. The method was the same as for experiment 1. This skew was introduced as an initial period greater than the nominal one (50 ms).

The experiment consisted of playing the same piece of audio from two different workstations. If one of them gets behind, the voice pitch changes. If the difference gets bigger echo is noticed. During the experiment some changes on the pitch were felt but this was expected as the audio is the most sensitive media towards these synchronization problems.

The intramedia algorithm is applied to the master stream normally. Around IU number 130 a local correction is made (figs. 10 and 11). The intramedia algorithm for the slave stream has the "false skew" and skew corrections disabled: the rate adjustment is made in relation to the master stream by the intermedia algorithm.

The intermedia difference *dif1* increases influenced by the skew (fig. 12). When its average crosses φ, around IU number 220, the skew is detected and a correction to the slave transmitter period is performed (figs. 12, 13 and 14). The skew influence gets lower, as seen in the second "section" of fig. 12, but its influence is still felt and another correction is made around IU number 490. As referred for the intramedia case, the skew influence takes some time to disappear.

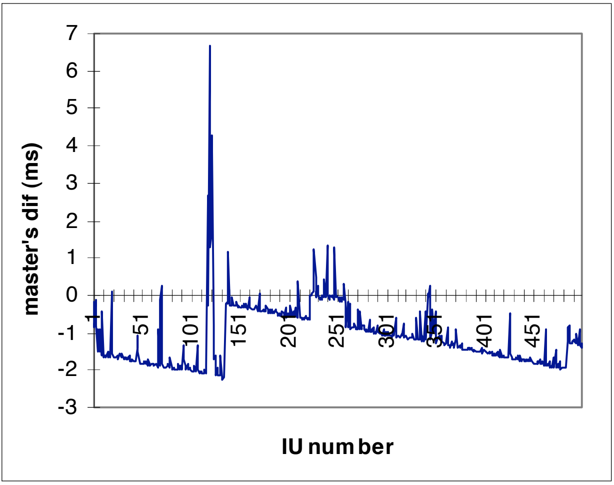

Figure 10. master's dif(n) chart

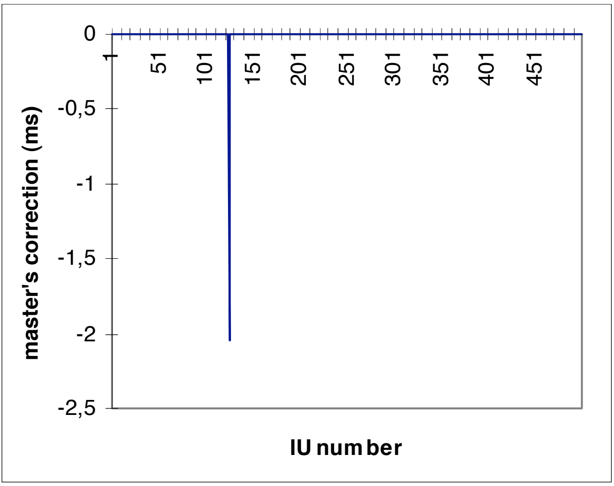

Figure 11. Chart of the master's intramedia corrections (the correction is local because it is lower than 10 ms)

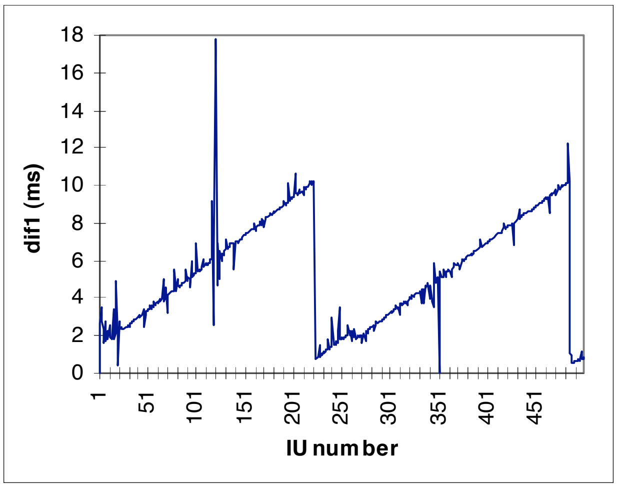

Figure 12. dif1(n) chart

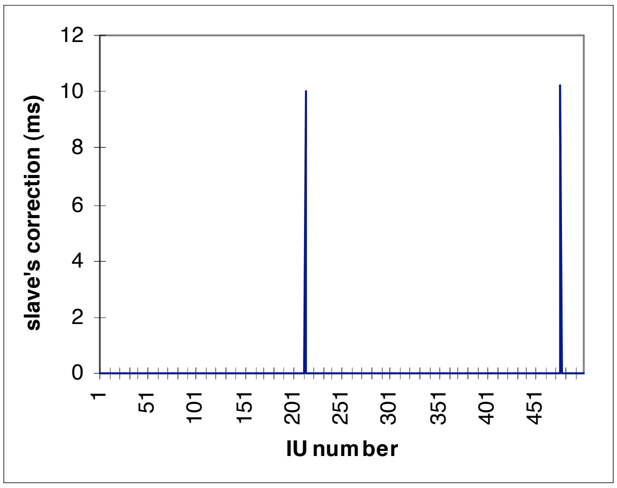

Figure 13. Chart of the corrections made to the slave caused by the intermedia algorithm (the corrections are made to the transmitter period)

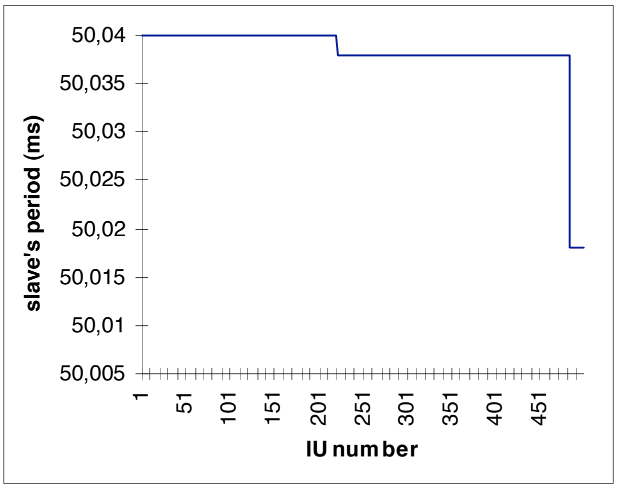

Figure 14. Chart of the slave transmitter period (fig. 13)

8. DISCUSSION AND CONCLUSIONS

This paper presents intramedia and intermedia synchronization algorithms for multimedia distributed systems with non-real time operating systems and an asynchronous network. The algorithms synchronize information streams using reception times and take into account real-life problems such as heavy loads on the machines and network. There are different mechanisms of correction actions depending on the severity of the problem. The two modes of operation, "interactive" and "non-interactive" make the algorithms suitable for a large range of applications, from video-conference to video-on-demand and cooperative work systems.

The use of the average of the differences of reception times, tr(n) - trexp(n) and tr(n_s) - tr(n_m), gives a good and easy to calculate measure of the effects involved but deserve some discussion. Averages are used in spite of singular values because values are affected by several effects, such as jitter, that can mislead any quick correction. The number of values to be considered for the average has to have several aspects in mind. If just a few values (2 or 3) are used, the algorithm gets too sensitive reacting to almost any variation on the values. If a large number is used, the problems will take too long to be detected and corrected, causing, for example, the lost of some IUs.

The relation between high-level concepts and low-level ones is not constrained by any static correspondence, as opposed to [13]. The algorithms were originally designed to work with a high-level synchronization system [14, 2] that can define blocks anywhere in the stream (eg., when the word "hello" is spoken, each time a blue car passes). Blocks can be composed with other blocks from other streams and the QoS can vary from block composition to block composition according to the high-level synchronization. The low-level algorithms explained here can support this paradigm entirely and the only thing needed is to be able to change α and β during the call, when the stream passes from one block to the next.

There are still some features that will be addressed in the near future:

- The correspondence between the indirect QoS parameters, the direct ones and the QoS levels, together with an assessment of the system characteristics and performance.
- The introduction of some sensor on the error rate (mainly due to network policing) in order to influence the QoS. The current QoS is only influenced by temporal synchronization features.
- The definition of the full interface offered at synchronization level to drive the algorithms taking into account the two issues above.
- The integration with the high level synchronization system described in [2] will be done.
- The synchronization of MPEG streams is not so simple as for sequences of JPEG images or sound. The selective discard mechanism has to consider that a loss of a single I frame can cause a serious QoS problem.

The QoS degradation scheme has to discard preferably B, P and lastly I frames.

- An interesting case not considered is the adaptation to multicast algorithms.
- Another aspect to explore is the feedback used to adjust the slave transmitter period. A better quantitative analysis can be done with the help of distributed control techniques.

REFERENCES

1. Anderson, D. and Homsy, G., A Continuous Media I/O Server and Its Synchronization Mechanism, *IEEE Computer* (October 1991)
2. Bernardo, L. and Pinto, P., Sharing Multimedia Information: a Basis for Assisted Remote Training, *Boadband Islands '95* (1995)
3. Correia, M., Multimedia Intrinsic Synchronization in Distributed Systems (in Portuguese), MSc Thesis, Instituto Superior Técnico, Lisboa, Portugal (June 1995)
4. Dairaine, L., Drift Matching Techniques for Time Signature Conservation Service, *Broadband Islands '94* (1994)
5. ETSI, B-ISDN ATM Adaptation Layer (AAL) specification - type 1 (January 1994)
6. ISO/IEC DIS 11172, Information Technology - Coding of moving pictures and associated audio for digital storage media up to about 1,5 Mbit/s (1992)
7. ISO/IEC 1/SC 29/WG 12, Information Technology - Coded Representation of Multimedia and Hypermedia Information Objects (February 1993)
8. Li, L., Karmouch, A. and Georganas, N., Real-Time Synchronization Control in Multimedia Distributed Systems, *ACM SIGCOM* / CCR (July 1992)
9. Li, L. and Georganas, N., MPEG-2 Coded- and Uncoded- Stream Synchronization Control for Real-time Multimedia Transmission and Presentation over B-ISDN, *ACM Multimedia 94* (1994)
10. Mey, V. and Gibbs, S., A Multimedia Component Kit *ACM Multimedia 93* (1993)
11. Mills, D., Network Time Protocol (Version 3). Specification, Implementation and Analysis, Internet RFC 1305 (March 1992)
12. Newcomb, S., Kipp, N., and Newcomb, V., The HyTime: Hypermedia/Time-based Document Structuring Language, *Communications of the ACM* (November 1991)
13. Nicolaou, C., An Architecture for Real-Time Multimedia, *IEEE Journal on Selected Areas in Communications*, vol. 8 n°3 (April 1990)
14. Pinto, P. and Linington, P., A language for the specification of interactive and distributed multimedia applications, Open Distributed Processing, II, Editors J. de Meer, B. Mahr and S. Storp, *IFIP Transactions*, North-Holland (1994)
15. Pinto, P., Bernardo, L., and Pereira, P., A Constructive Type Schema for Distributed Multimedia Applications, *Boadband Islands '94* (1994)
16. Qazi, N., Woo, M. and Ghafoor, A., A Synchronization and Communication Model for Distributed Multimedia Objects, *ACM Multimedia 93* (1993)
17. Ramanathan, S. and Rangan, P., Feedback Techniques for Intramedia Continuity and Intermedia Synchronization in Distributed Multimedia Systems, *Computer Journal*, vol. 36 n°1 (1993)
18. Shepherd, D. and Salmony, M., Extending OSI to Support Synchronization Required by Multimedia Applications, *Computer Communications* (September 1989)
19. Stefani, J., Hazard, L. and Horn, F., Computational model for distributed multimedia applications based on a synchronous programming language, *Computer Communications* (March 1992)
20. Steinmetz, R. and Engler, C., Human Perception of Media Synchronization, IBM ENC Tech. Rep. n. 43.9310, Heidelberg (1993)
21. Strayer, W., Dempsey, B. and Weaver, A., XTP, The Xpress Transfer Protocol, Addison Wesley (1992)
22. Talley, T. and Jeffay, K., Two-Dimensional Scaling Techniques for Adaptive, Rate-Based Transmission Control of Live Audio and Video Streams, *ACM Multimedia 94* (1994)
23. Zhang, L., Braden, R., Estrin, D., Herzog, S. and Jamin, S., Resource Reservation Protocol (RSVP) - Version 1 Functional Specification, Internet Draft (July 1994)