A Synchronization and Compensation Protocol for Multimedia Communication Systems*

Chao-Hui Wu Liang-Seng Koh Ming T. Liu
Department of Computer and Information Science
The Ohio State University
Columbus, Ohio 43210-1277

Abstract

Media synchronization and freedom from starvation at destination devices are the two most important problems in a multimedia communication system. In order to solve these problems at the same time, this paper presents a protocol combining the scheduling scheme and the buffering scheme with underflow threshold (SBUF). In this protocol, by controlling the buffer sizes and modifying transmission schedule of the synchronizer, synchronization can be achieved without synchronizing clocks during a connection. In this paper, the algorithm of the protocol is described. The setting of the system parameters and network QOS to support the proposed protocol are also given in this paper. A simulation is performed for comparison of four synchronization schemes: scheduling schemes (with and without clock synchronization), marker scheme, and the proposed SBUF protocol. The results confirm that the proposed protocol guarantees satisfaction of QOS requirements and also performs better than the other schemes in terms of media synchronization and freedom from starvation at destination devices.

1 Introduction

A real-time multimedia application consists of different media types such as text, voice, image, video and audio. Data from various media that are generated at the same time need to be played back simultaneously or within an acceptable range at the destination. Due to the unique characteristics of each medium, different channels are used to carry different media at the same time. Consequently, different delays or jitters will be introduced into different media at the destination. Moreover, different media may go through different encoding/decoding procedures which have varying processing latencies. The frequencies of some physical devices may not match with others; therefore, the existence of different clock drifts between devices is very possible. See Figure 1 [1, 2]. The above factors lead to asynchrony among media streams during playback at the destination. An effective synchronization and compensation protocol is needed to prevent or remedy the possible asynchrony problems so as to guarantee the playback quality of multimedia applications.

Most work on multimedia synchronization protocols can be classified into two categories: scheduling schemes and marker schemes. In scheduling schemes [3, 4, 5], either a global network clock or a clock synchronization technique is assumed to be available. Each node or switch has a corresponding "transmission/display schedule." If all clocks in the system are synchronized, synchronization will be achieved if all devices transmit and play back data frames according to their respective schedules. However, in this scheme, the quality of the media synchronization highly depends on how fine clocks in the system can be synchronized. The current well-referenced clock synchronization technique, the Network Time Protocol (NTP) [6], can synchronize clocks within a range of +/- 5 ms. It still cannot satisfy some applications which need fine synchronization (such as down to 100 ms between media). In marker schemes [7], special control frames are added to mark points that need to be synchronized in the media streams. To ensure that synchronization is achieved among various media streams, a synchronizer realigns these control frames before data frames are played at the destination. However, the maker scheme cannot guarantee Quality of Service (QOS) when clock drift or data loss occurs. In both schemes, the buffering technique is adopted by using buffers to store early arrived data frames. Both the above schemes introduce extra overhead into the system. In some environments with very limited bandwidths, such as wireless communications, periodically synchronizing clocks across the system or inserting control frames can be very costly. Moreover, most of the protocols fail to compensate for the consequences of data loss and clock drift. As a result, the device starvation problem can still occur.

In this paper, a synchronization and compensation protocol is proposed to ensure media synchronization and freedom from starvation at destination devices. The protocol is a combination of the scheduling scheme and the buffering scheme with underflow threshold. The synchronization among media is achieved at the synchronizer by sending out all media frames according to the respective schedules at the synchronizer. To maintain synchronization at the destination, the buffer sizes of the destination devices are minimized to be three. The schedule of the synchronizer is modified to cope with the starvation problem caused by the clock drift between the synchronizer and the destination. Underflow threshold is defined and used to prevent overduplication of frames while compensating

*Research reported herein was supported by U.S. Army Research Office, under contract No. DAAL03-92-G-0184. The views, opinions, and/or findings contained in this paper are those of the authors and should not be construed as an official Department of the Army position, policy or decision.

0-8186-7216-1/95 $04.00 © 1995 IEEE

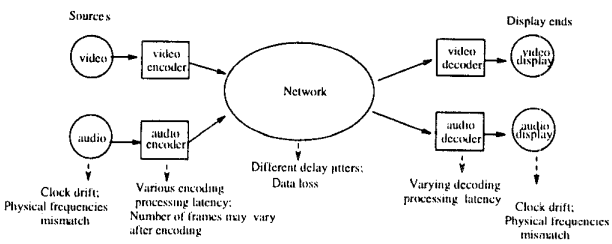

Figure 1: Asynchrony factors of a multimedia system

for the starvation problem. The clock synchronization is not needed during a connection. The assumption of the proposed protocol is that the upper bound of the drift between clocks is known. The simulation performed in this paper shows that the proposed protocol guarantees satisfaction of all QOS requirements. In addition, the proposed protocol is the best of the four schemes in eliminating the destination devices starvation and synchronization problems.

The rest of this paper is organized as follows. In Section 2, the asynchrony problems are described, and the QOS requirements used to prevent these asynchrony problems are also specified. In Section 3, the proposed protocol is introduced. The algorithm, the setting of system parameters and the QOS translation between the application and the network are also given. A simulation study comparing four schemes is given in Section 4. Finally, an extension of this protocol is examined and concluding remarks are presented in Section 5.

2 QOS Requirements

Before we present our protocol, the asynchrony problems that this paper is going to address and QOS parameters to be used later are described in this section.

2.1 Asynchrony problems

The goal of a synchronization and compensation protocol is to remedy the asynchrony problems at destinations such that the quality of an application can be maintained. It is necessary to identify every possible consequence and problem caused by asynchrony factors so that the proposed protocol can be designed to prevent and compensate for these undesired asynchrony problems completely. The possible undesired consequences and problems are stated as follows.

Lip-sync problem: The image of a speaker's lips does not match his voice. If the supposedly synchronized picture and voice data arrive at the destinations within the range 70-150 ms, human viewers cannot detect the asynchrony between them.

Destination devices starvation problem: A destination device has no data frame to play when the scheduling deadline is due. The human visual and auditory systems can perceive the remaining effect of a picture and voice for 30-40 ms. Thus, display devices need to playback data periodically at fixed time intervals of 30-40 ms in order to preserve the effect of continuity in the picture and voice.

Duplication gap problem: The duplication gap problem occurs when the same frame is played too many times, so that human viewers are able to detect the abnormal result. It is a result of overduplication by a compensation protocol for lost or late arrival data.

Media skip problem: The media skip problem occurs when two non-consecutive data frames are played consecutively at the destination. Data loss during transmission and data dropped by the synchronization protocol are the main causes of this problem. This consequence cannot be compensated because retransmission is not possible in applications that have real-time constraints. Hence, when a synchronization protocol is designed, it is important not to introduce too much dropping of data frames while trying to overcome the other problems.

The above undesirable problems can be prevented if appropriate QOS requirements are specified in an application and these QOS requirement are maintained during a connection by a designed protocol and the underlying network. Therefore, in the following two subsections, we first identify the QOS requirements to be specified by the application for addressing the problems and then present the network support needed in terms of network QOS requirements. Based on these two sets of QOS parameters, an efficient protocol is proposed to address these problems in Section 3. The translation between two sets of QOSs is shown in Section 3.4.

2.2 QOS requirements specified by an application

In order to eliminate the undesirable problems described above, we define a set of QOS requirements specified by an application in terms of the following parameters:

- **transmission rate (frames/sec) $\hat{\Omega}_i$, a.k.a. sampling rate:**
  the number of data frames transmitted/displayed per second. This parameter is used to address the destination devices starvation problem.

- **maximum end-to-end delay allowed (sec) $\hat{D}_i$:**
  the maximum allowed time interval for a data frame between generating time at the source and displaying time at the destination. This is used to address the real-time constraint for an application. It can also help to prevent the destination devices starvation problem.

- **maximum out-of-sync allowed range (sec) $\hat{\Phi}_i$:**
  the maximum allowed out of sync range which can be tolerated between media. If this requirement can be guaranteed, the lip-sync problem can be eliminated.

- **maximum skip frames allowed at one time (frames) $\hat{K}_i$:**

the maximum number of consecutive frames that can be skipped. This parameter is for preventing media skip problem.

- **maximum duplication allowed per frame $\hat{U}_i$:** the maximum number of duplications for a frame. This parameter is needed for avoiding duplication gap problem.

When service is requested, an application needs to specify a set of values to these QOS parameters. This set of values defines the QOS requirements for the application. For an application, if all its QOS requirements can be satisfied, undesirable asynchrony problems will be avoided at the destination.

2.3 QOS parameters supported by the network

The network support needed are specified in terms of the network QOS parameters. The network QOS parameters include:

- **data transmission rate (frames/sec)** ω_i
  the number of frames transmitted per second by the network
- **data loss rate** p_i
  the percentage of data allowed to be lost by the network
- **maximum network delay allowed** d_i
  the maximum allowed end-to-end network delay
- **network jitter** j_i
  the maximum delay variance allowed in the network

The values of these network QOS parameters are derived from the application QOS requirements plus the possible overhead introduced by the synchronization and compensation protocol. The relationship between the network QOS requirements and the others is shown in Section 3.4.

3 Protocol

In this section, a synchronization and compensation protocol is proposed to solve the lip-sync and starvation problems and to avoid the duplication gap and media skip problems. First, the compatibility of the proposed protocol with the current existing network models is discussed. Then, the protocol is described in details to show how each asynchrony problem is prevented. Based on the proposed protocol, how the system parameters are tuned to support the QOS requirements is also shown in this section. Finally, the transformation between the network QOS parameters and the other parameters are presented.

3.1 Network model

In existing networks, since there are no corresponding functions to address synchronization and its associated problems, the proposed synchronization protocol can be added as a layer to an existing network model. In order to absorb all the asynchrony factors, the synchronization layer should be placed as

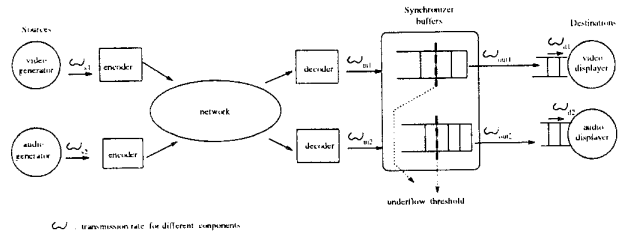

Figure 2: Network architecture for a multimedia system

close as possible to the application layer. For example, in order to absorb various latencies of decoding procedures for different media, a synchronization layer should be placed between the application layer and the presentation layer in the OSI model, and between the ATM adaptation layer and the application layer in the ATM model.

A multimedia communication system consists of several components. The proposed protocol requires a synchronizer near the destination to coordinate synchronization among media streams, as shown in Figure 2. It is assumed that the new data will overwrite the most recently written data frame in the buffer if the buffer is full when new data frames arrive.

Data frames are generated periodically at the sources according to the sampling/display rate of the devices. A time stamp or sequence number is stamped on each data frame, so that the expiration time can be checked at the destination. The network transmits data frames according to the network QOS negotiated when the connection was set up. When data frames arrive at the synchronizer, they will first be stored in the associated buffer. Data frames will be sent to the destination according to the transmission schedule in the synchronizer. At the destination, newly arrived data frames are stored in the buffers of the destination devices before being played back. A device periodically retrieves data from its buffer and plays it back according to the playback (retrieval) rate of the device.

3.2 Protocol description

The synchronization protocol proposed in this paper uses both the scheduling technique and the buffering technique. Basically, asynchrony factors and effects originating from the processes before the synchronizer can be detected and compensated for by the synchronizer. For example, the length of a buffer in the synchronizer indicates the possible clock drift relationship between the synchronizer and the sources. If the length of the data frames in the buffers keeps increasing, it implies that the transmitting speed of the sources may be faster than that of the synchronizer, or vice versa. Using scheduling and buffering techniques, the synchronizer smoothes the skew of the arrival time of data streams and makes sure the outgoing data frames are synchronized or within the out-of-sync range. The lost data or late arrival data will be compensated for by duplicating data frames.

However, asynchrony effects that occur after the synchronizer but before playback are hard to control. The clock drift relationship between the synchronization and destination de-

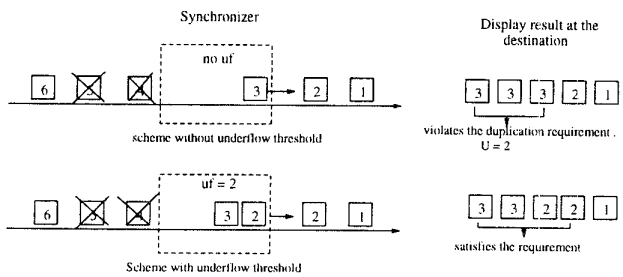

Figure 3: Underflow threshold smoothes the duplication gap

vices cannot be known if there is no feedback from the destination devices. Device starvation may occur if the clock at the destination is faster than the one at the synchronizer. The lip-sync problem may occur if the clock drifts between the destination devices are different, despite media streams being synchronized at the synchronizer. The way the proposed protocol overcomes these two problems is described separately in the sections that follow.

3.2.1 Solving the starvation problem

The device starvation problem occurs either when data frames are lost during transmission or when the clock at the destination is faster than the clock at the synchronizer. In the first case, the synchronizer uses an *underflow threshold (uf)* to indicate a potential insufficient data situation. When the number of data frames in the buffer drops to the level of the uf threshold, the synchronizer will start making copies of the outgoing data frames so that the duplication per frame can be minimized without exceeding the duplication requirement. For example, as shown in Figure 3, a scheme without the underflow threshold does not save enough frames in the buffer. If the data frames with sequence numbers 4 and 5 are lost during transmission, the scheme without uf can only compensate for the lost data by repetitively transmitting a copy of the previous frame. As a result, the duplication per frame at the destination may exceed the QOS requirement specified by the application of a maximum duplication rate of $U = 2$. However, in the scheme using uf threshold, the duplication gap can be smoothed by always saving uf frames in the buffer.

In the second case, starvation is prevented by modifying the transmission rate of the synchronizer so as to guarantee that the destination playback rate is not faster than the synchronizer transmission rate no matter how bad the clock drift is. That is, the schedule for the synchronizer is modified to always send out more frames as if the clock at the destination were the fastest and the clock at the synchronizer were the slowest. Assuming the clock drift is bounded, the bound is defined as B per second. In order to prevent the starvation problem, the synchronizer assumes itself to have the slowest clock rate, and the destinations to have the fastest clock rate. If the original scheduled transmission rate for the synchronizer is Ω frames/second, it will be modified to a rate of at least $\Omega(1 + B)$ to match the fastest possible transmission rate at the destination. In this way, the transmission rate of the synchronizer will always be the fastest or one of the fastest of all the components. As a result, starvation at the destination will never occur. With the aid of uf threshold, data frames can be properly duplicated so that sufficient data is supplied for the synchronizer to transmit.

Using this method, though a duplication problem is created, it can be solved easily, thus avoiding the starvation problem at the destination. At the same time, by applying uf threshold, the duplication per frame is kept under control. If the clock drift between the destination and the synchronizer is not as predicted, extra frames transmitted by the synchronizer will be overwritten by new arrival data frames that are newly arriving at the destination buffers.

3.2.2 Solving the lip-sync problem

The size of the buffers at the destination site will affect the synchronization quality of the multimedia system. If clock drifts among the destination devices differ, their real data display rates will be different from what they declare. Despite all data frames being synchronized at the synchronizer, a destination device with a faster clock will display more data frames than a destination device with a slower one. The queue for the slower device's buffer will gradually increase. This out-of-sync effect among media may occur after a period of time. To avoid this situation, and to consume the extra data frames sent by the synchronizer, the buffer size should be as small as possible so that newly arriving data frames can be played back right away and the synchronization achieved at the synchronizer can be maintained at the destination. The shorter the period of time data frames wait at the buffers, the less the chance that they will be out of synchronization.

However, the read-write conflict at the destination buffer can be avoided only if the buffer size is large enough to contain at least three data frames. This is illustrated via counter-examples. Note that writing by the synchronizer is more frequent than reading by the destination devices. This is because, after modification of the schedule, the synchronizer has the fastest transmission rate of all components. When the buffer is full, the newly arriving data frame will overwrite the data frame the synchronizer wrote just before it; otherwise, the new data frame will be written in an empty buffer. If the buffer size is one, then while the display device is reading, the newly arriving frame may rewrite it, and a read-write conflict occurs. Possible effects may be that the display frame proves incomplete or the display of a frame is interrupted. If the buffer size is two, as shown in Figure 4, the read-write conflict may occur when the destination device finishes reading the first buffer and starts to read the second buffer. If the buffer size is three, there is always one more buffer that no read or write will access. Whichever agent finishes first can move to that buffer. Therefore, the read-write conflict can be avoided.

In summary, to avoid possible asynchrony at the destination, the smaller buffer size of the destination should be as small as possible. However, as discussed above, the minimum size needed to avoid a read-write conflict is at least three. Therefore, the optimal buffer size for the destination device should be three.

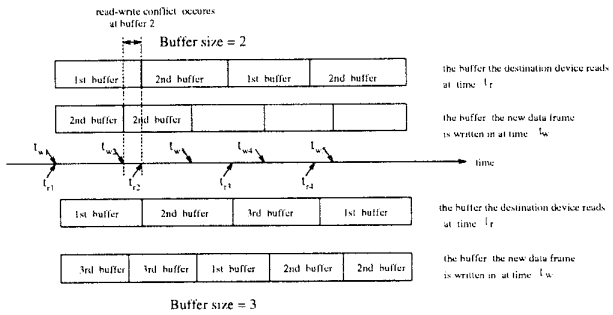

Figure 4: Examples of read-write conflict

3.3 Algorithm and system parameters setting

To sum up the protocol described above, an algorithm of the proposed protocol for the synchronizer is given in this section. Whenever needed, system parameters will be set to support the QOS of the application.

The system parameters used in the *synchronizer* are as follows. See also Figure 2.

- ω_{out}: new transmission rate after modification
- uf: underflow threshold to indicate any possible insufficient data situation
- $counter$: the maximum number of duplications per frame
- buf: the buffer size in the synchronizer

When the connection is set up, the clocks in the system are synchronized, and each component is given a schedule, which normally is the transmission rate/sampling rate of the application. The transmission schedule for the synchronizer is modified to match the possible clock drift between the synchronizer and the destination so that the starvation problem can be avoided. When a new deadline is up, if the number of data frames in the buffer is more than or equal to uf threshold, the synchronizer retrieves a data frame from the associated buffer and sends it out. However, if the number of data frames in the buffer is less than the uf threshold, and if the data frame at the head of the associated buffer has not been duplicated more than $counter$ times, the synchronizer sends out a copy of that data frame. The detailed algorithm is shown in Figure 5.

In order to support the proposed protocol, system parameters need to be tuned accordingly. The setting of system parameters is explained further in the following subsections.

3.3.1 Setting a new transmission rate ω_{out}

It is assumed that clock drift is bounded at B per second between a clock and a precision timing source (calibrated cesium clock.) When a clock is operating as precious as a calibrated cesium clock, the transmission rate of a media stream i for each

Modify the schedule of the synchronizer
For every media stream i:
  $counter_i = U_i$
  Set the value of uf_i based on (5)
For every media stream i:
  while a deadline on the new schedule is due
    if $0 <$ the queue length of the $buffer_i < uf_i$ **then**
      if $counter_i > 0$ **then**
        synchronizer sends out a copy of the first data frame in the $buffer_i$
        $counter_i = counter_i - 1$
      else
        synchronizer sends out the first data frame in $buffer_i$
        $counter_i = U_i$
    else if the queue length of the $buffer_i \geq uf_i$ **then**
      synchronizer sends out the first data frame in its buffer
      $counter_i = U_i$
end

Figure 5: The algorithm for the synchronizer

component in the system is Ω, as specified in the QOS parameters. However, because of local clock drift, the real transmission rates of components may not be Ω as compared to the precision timing source. The starvation problem may occur if the clock at the destination is faster than the one at the synchronizer. In the worst case, the real display rate of a display device would be $\Omega(1+B)$, and the real transmission rate of the synchronizer would be $\Omega(1-B)$, according to a precision timing source. In the worse case, to match the transmission rate of the destination the synchronizer must be x times faster than its real rate. That is,

$$x * \Omega(1-B) = \Omega(1+B)$$

The new transmission ω_{out} rate for the synchronizer is set to:

$$\omega_{out} = x * \omega_{syn} = \frac{1+B}{1-B}\omega_{syn} \tag{1}$$

where ω_{syn} is the original transmission rate of the synchronizer as compared to a precision timing source.

3.3.2 Setting the *counter*

In order to guarantee QOS requirements regarding the maximum duplications per frame at the destination, the maximum number of duplications per frame ($counter$) at the synchronizer is set to be U, the maximum allowable duplications per frame specified by the application.

$$counter = U \tag{2}$$

Since the number of copies of each frame transmitted from the synchronizer is less than or equal to U and since the end device will not duplicate data frames, the maximum number of copies of a frame displayed at the destination will be less than or equal to U. Thus the situation of overduplication of frames at the destination can be eliminated.

However, if duplicating U copies of every data frame generated at the source still cannot meet the display rate of the destination, more than U copies per frame are necessary to achieve freedom from starvation at the destination. Thus, the proposed protocol can only guarantee freedom from the overduplication problem if the clock drift bound, B, and the duplication, U, are related as shown below.

$$\omega_{source} * U > \omega_{destination} \tag{3}$$

The data generation rate at the source is expressed as ω_{source}; $\omega_{destination}$ is the data playback rate at the destination (indicated as ω_{s_i} and ω_{d_i} in Figure 2). In the worst case, $\omega_{source} = (1-B)\Omega$ and $\omega_{destination} = (1+B)\Omega$. Therefore, according to inequality (3), it can be derived as:

$$U \geq \frac{1+B}{1-B} \tag{4}$$

Fortunately, the bound of clock drift B is hardly likely to be more than 10^{-1} per second. In that case, according to (4), U as specified by the application will be at least 1.3, which is generally acceptable for most of the applications. Therefore, in most cases, the proposed protocol can satisfy the overduplication constraints of QOS requirements.

3.3.3 Setting the uf

Suppose the the data loss rate of the network is p, which is also the probability that a data frame will be lost. Denote $jitter$ as network jitter and ω_{in} as the data frames arrival rate from the network. Then uf should contain sufficient data frames to make it certain that a new data frame will arrive before all the data frames in the associated buffer are gone. That is, the following inequality should hold:

$$\frac{uf * U}{\omega_{out}} \geq (\frac{1}{\omega_{in}} + jitter) * ((1-p) + 2p(1-p) + 3p^2(1-p) + ...)$$

Since $(1-p) + 2p(1-p) + 3p^2(1-p) + 4p^3(1-p) + ... \rightarrow \frac{1}{1-p}$, it can be rewritten as:

$$uf \geq \frac{(\frac{1}{\omega_{in}} + jitter) * \omega_{out}}{counter * (1-p)}$$

In the worst case, $\omega_{in} = (1-B)\Omega$ and $\omega_{out} = \frac{1+B}{1-B}(1+B)\Omega$, thus

$$uf \geq \frac{(1+B)^2(\frac{1}{1-B} + jitter * \Omega)}{(1-B)(1-p)U} \tag{5}$$

3.3.4 Setting the buffer size buf

The buffer overflow problem will not occur with a synchronizer because ω_{out} is always greater than ω_{in}. However, the buffer size (buf) at least should be larger than the uf in order to hold the data frames needed to be duplicated. Therefore, the buffer size buf can be set as

$$buf = uf + 1 \tag{6}$$

3.4 Setting the corresponding network QOS parameters

In order to support the QOS requirements of the application, the QOS parameters of the network are set as follows :

- **data loss rate** (p): The allowed data loss rate is mapped from the maximum skip allowed parameter (K) specified by the multimedia application. Data loss is not possible in the synchronizer buffers since $\omega_{out} \geq \omega_{in}$. It will occur only if data frames are overwritten by newly arriving data. Thus, p needs to satisfy the inequality (5).
- **data transmission rate** (ω): If possible, the data transmission rate of the network should be set as $\Omega(1+B)$, which is the fastest possible transmission rate at the sources. However, it is very rare that clocks at both sources and destinations will drift so much. In order to achieve better network utilization, the data transmission rate is set as the scheduled transmission rate Ω. The simulation result confirms that it is sufficient to guarantee the application QOS satisfaction.
- **end-to-end delay** (d): The time spent on the synchronizer buffer will be $\frac{buf-uf}{\omega_{out}} + \frac{uf*c}{\omega_{out}} + \frac{3}{\omega_{destination}}$. Therefore, the end-to-end delay supported by the network should be

$$d \leq D - (\frac{buf - uf}{\omega_{out}} + \frac{uf * counter}{\omega_{out}} + \frac{3}{\omega_{destination}})$$

In the worst case, ω_{out} and $\omega_{destination}$ are both slowest; that is, $\omega_{out} = \frac{1+B}{1-B}(1-B)\Omega = (1+B)\Omega$ (by (1)) and $\omega_{destination} = (1-B)\Omega$. Together with (2) and (5),

$$d \leq D - (\frac{1 + uf * U}{(1+B)\Omega} + \frac{3}{(1-B)\Omega})$$

- **network jitter** ($jitter$): The network jitter should satisfy the inequality (5).

4 Simulation Results

In this section, a simulation is carried out to compare the performance of the proposed protocol with other synchronization schemes, namely, the marker scheme, the scheduling scheme with synchronized clocks and the plain scheduling scheme without synchronized clocks. Four schemes are implemented as follows:

1. **Marker scheme (MK):**
   As described in Section 1, in the marker scheme special control frames are inserted into the media stream periodically. These special control frames are then realigned at the synchronizer. In this simulation, controlled frames are inserted every 3.3 seconds, or approximately every 100 frames for the video application.
2. **Scheduling scheme with synchronized clocks (SSC):**
   Data frames are transmitted according to their schedules and their local clocks. In the system, all clocks are synchronized within -/+ 5 ms.

The QOS requirements specified by an application[4, 8]:

media	$\hat{\Omega}$ frms/s	$\hat{D}$ ms	$\hat{\Phi}$ ms	$\hat{U}$ per frm	$\hat{K}$ frms
video	30	250	100	2	2
voice	3000	250	100	2	2

The QOS parameters supported by the network:

media	ω frames/sec	d ms	j ms	p
video	30	220	10	10^{-3}
voice	3000	220	10	10^{-1}

Figure 6: QOS parameters used in simulation

3. **Plain scheduling scheme without synchronized clocks (PS):**
   This scheme is similar to the SSC but clocks in the system are not synchronized.

4. **Scheduling scheme with underflow threshold (SBUF):**
   This is the protocol proposed in this paper. Parameters are set up as described in Section 3.

Given the same QOS requirements from an application and the same network support, the percentage of the QOS requirements violations between schemes is compared. The application selected combines video and audio media. The QOS requirements and the parameters supported by the network are listed in Figure 6 and Section 2.

The performance parameters investigated are :

out-of-sync rate (%): the percentage of out-of-sync frames as compared to the total frames generated

overduplication rate (%): the percentage of overduplicated frame as compared to the total frames generated

over skip rate (%): the percentage of frames where the skip range is more than that allowed by the QOS parameter as compared to the total frames generated

starvation rate (%): the percentage of destination starvation as compared to total frames generated

average max out-of-sync (ms): the average value of the maximum out-of-sync occurrences displayed at the destination

duplication (%): the percentage of duplicated frames as compared to the total frames generated

skip (%): the percentage of frames skipped as compared to the total frames generated

The first four parameters are indicators of QOS violations. They should all be zeroes in order to satisfy the QOS requirements specified by the application. The last three parameters are quality indices which show the quality of service for the application. The smaller their values are, the better the quality is. The simulation is carried out under different clock drift bounds (B) and under different clock drift situations among the sources, synchronizers and the destination devices. The results are shown in Figure 7.

B = 5 ms, U=2 , and K=2					
Scheme	out of sync (%)	starv. (%)	max out-sync (ms)	dupl. (%)	skip (%)
SSC	10.52	0.4	105.3	0.32	0.32
PS	10.52	0.4	105.3	0.32	0.32
MK	0	0.54	40.49	0	0.29
SBUF	0	0	46.47	1.35	1.23
B = 10 ms, U=2, and K=2					
Scheme	out of sync (%)	starv. (%)	max out-sync (ms)	dupl. (%)	skip (%)
SSC	10.52	0.4	105.3	0.32	0.32
PS	14.93	0.61	158.9	0.53	0.47
MK	0	0.71	44.02	0	0.41
SBUF	0	0	51.25	2.28	2.25
B = 100 ms, U=2, and K=2					
Scheme	out of sync (%)	starv. (%)	max out-sync (ms)	dupl. (%)	skip (%)
SSC	10.52	0.4	105.3	0.32	0.32
PS	40.00	6.58	1620.46	3.97	3.85
MK	0	5.90	47.90	0	4.04
SBUF	0	0	52.37	15.43	15.3

Figure 7: Simulation results

From the results, the average over duplication rate and the average over skip rate are all zeroes in all four schemes. This means that four schemes will not introduce the duplication gap and media skip problems while compensating the other problems. However, only the proposed protocol, SBUF, has zero in the both the out-of-syn and starvation rates measurement. That is, only SBUF successfully remedies the out-of-sync and starvation problems. Besides, the value of the maximum out-of-sync parameter also indicates that the proposed protocol can achieve fine synchronization. Two of the quality indices, the duplication rate and the skip rate, are higher as compared to the other schemes. This is because the synchronizer creates extra duplication and transmission to prevent destination starvation from occurring. However, as long as duplication and skip are within the ranges of QOS requirements, the results are acceptable.

The marker scheme (MK) performs well except on the freedom from starvation requirement. The performance of the scheduling scheme with synchronized clocks (SSC) is stable. However, starvation and out-of-sync problems may still occur even when clocks are synchronized within +/- 5 ms. As expected, the performance of the plain scheduling scheme without synchronized clocks (PSS) is the worst of the four schemes. When clock drifts between clocks in the system increase, the quality of the application decreases.

The data we collected is based on a three minutes long application. If the duration lasts longer, the performance parameters values with percentages in the Figure 7 will remains the same. However, the average of the maximum out-of-sync (ms) will increase proportionally to the duration length.

The results of the out-of-sync rate and the starvation rate are also plotted in the Figure 8 and Figure 9, respectively. In Fig-

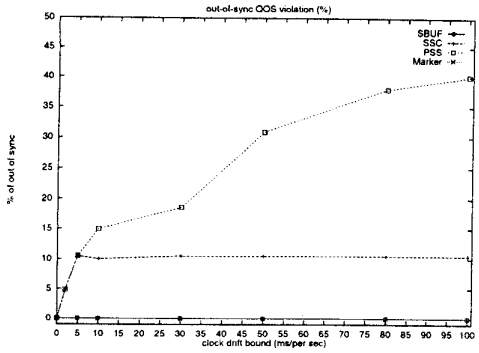

Figure 8: Comparisons in terms of out-of-sync rate

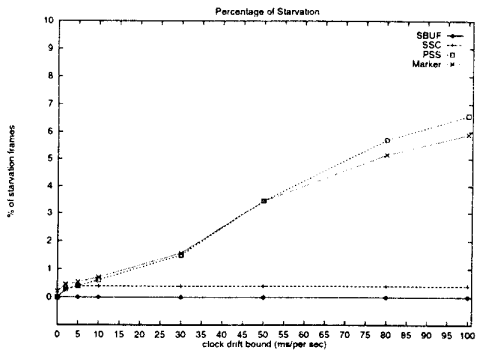

Figure 9: Comparisons in terms of starvation rate

ure 8, the marker scheme and the proposed protocol (SBUF) maintain no out-of-sync violation. The out-of-sync frame rate of SSC becomes stable at approximate 10% when the clock drift bound exceeds 5 ms since all clocks are synchronized within +/- 5 ms. For the PSS scheme, its out-of-sync rate is almost proportional to the clock drift bound. In Figure 9, the proposed SBUF protocol achieves starvation-free under any clock drift bounds. The SSC remains stable at 0.4%. The starvation rates for the marker scheme and the PSS scheme both increase proportionally to the clock drift bound. When the clock drift bound is less than 50 ms, the PSS has a slightly less starvation rate than the marker scheme. However, when the clock drift bound exceeds 50 ms, the marker scheme has smaller starvation rates comparing with the PSS.

5 Conclusion

In this paper, a protocol is proposed to simultaneously solve the synchronization and destination starvation problems. Clock synchronization is not needed during connection in this protocol. In this protocol, the synchronization is achieved at the synchronizer by transmitting data according to the respective schedules of the synchronizer. Synchronization is maintained at the destination by minimizing the buffer sizes of the destination devices to be three. The schedule of the synchronizer is modified to cope with the possible starvation problem. Underflow threshold is defined and used in the buffer of a synchronizer to prevent the possible violation of QOS requirements. The corresponding system parameters and the QOS parameters of the network are stated and set so as to support the proposed protocol. A simulation is done for comparing the four synchronization schemes. The results confirm that the proposed protocol guarantees the satisfaction of QOS requirements and also show better performance than the other schemes in terms of media synchronization and freedom from starvation at destination devices. Although the proposed protocol has a higher duplication rate and skip rate than the other schemes, the results are acceptable as long as they satisfy the QOS requirements.

This protocol could be further improved if the devices at the destination were given a feedback function. Only one feedback from each destination device would be needed. Based on the feedback, the synchronizer could accurately adjust its schedule to meet the display rates of the destination devices. The duplication percentage and skip percentage could be well controlled and it would be expected that they would decrease a great deal. The quality of the service would be improved dramatically.

References

[1] B. Furht, "Multimedia systems: An overview," *IEEE Multimedia Magazine*, pp. 47–59, Spring 1994.

[2] S. Browne, "Communication and synchronization issues in distributed multimedia database systems," in *Advanced Database Systems, Springer-Verlag*, pp. 381–396, 1993.

[3] T. D. Little and A. Ghafoor, "Multimedia synchronization protocols for broadband integrated services," *IEEE Journal on Selected Areas in Communications*, vol. 9, pp. 1368–1382, December 1991.

[4] J. Escobar, C. Partridge, and D. Deutsch, "Flow synchronization protocol," *IEEE/ACM Transactions on Networking*, vol. 2, pp. 111–121, Aprial 1994.

[5] M. Woo, N. U. Qazi, and A. Ghafoor, "A synchronization framework for communication of pre-orchestrated multimedia information," *IEEE Network*, vol. 1, pp. 52–61, January/February 1994.

[6] D. L. Mills, "Precision synchronization of computer network clocks," *Computer Communication Review*, vol. 24, pp. 28–43, April 1994.

[7] C.-J. Wang, L.-S. Koh, C.-H. Wu, and M.-T. Liu, "A multimedia synchronization protocol for ATM networks," in *Proc. IEEE International Conference on Distributed Computing Systems*, (Poznan, Poland), June 1994.

[8] A. Campbell, G. Conlson, F. Garlia, D. Hutchison, and H. Leopold, "Integrated quality of service for multimedia communications," in *Proc. IEEE INFOCOM '93*, pp. 732–739, 1993.