A Method and Apparatus for Measuring Media Synchronization

Brian K. Schmidt†, J. Duane Northcutt‡, and Monica S. Lam†

†*Computer Systems Laboratory, Stanford University, Stanford, CA 94305*
‡*Sun Microsystems Laboratories, 2550 Garcia Avenue, Mountain View, CA 94043*

Media synchronization is widely regarded as a fundamental problem in the field of multimedia. While much work has been conducted in this area, and many different solutions have been proposed, no method for obtaining a repeatable, objective measure of synchronization performance exists. Thus, there has been no means for determining the effectiveness of potential media synchronization solutions. In this paper we present an experimental methodology for quantitatively measuring the performance of different media synchronization schemes. We describe a complete (hardware and software) test environment for measuring audio/video synchronization quality of various media players, and we also present empirical performance measurements of an example media player. The results show that external observation is necessary for accurate assessments of synchronization performance. This test and evaluation methodology is applicable to other media delivery systems and can serve as the first step in isolating and quantifying the effects of individual components of a media delivery system.

1 Introduction

Multimedia refers to the use of one or more types of *media data* — data designed to be consumed by humans, such as text, graphics, audio, and video. These data typically possess timeliness requirements with respect to their presentation. For example, digital audio samples may be required to be displayed at a uniform rate of 48KHz. The media synchronization problem is to assure the correct temporal alignment of such time-critical activities relative to a physical clock.

Although there is a relatively large body of research describing various solutions to the synchronization problem, no metrics have been defined to measure the performance and efficacy of this work. Effectiveness assessments have been largely subjective in nature, thus making comparisons between different approaches quite difficult. To determine the degree to which media synchronization is achieved, a means for obtaining a repeatable, objective measure of (externally visible) system performance must exist. We present such a scheme below.

1.1 Background

To help characterize the media synchronization problem we present the following terminology. A *media element* is a single unit of a multimedia data type, such as a video frame, or audio sample. A *media stream* (or stream) is a series of media elements. Common examples of streams include video clips and audio sound bites. A time-ordered collection of media streams is termed a *media sequence* (or sequence). A good example of a sequence is a music album, i.e. a set of audio streams (songs). Given these definitions we decompose the media synchronization problem space into three classes: event-based, stream-based, and element-based.

Event-based synchronization refers to synchronization activities performed in response to external events such as user input, whereas *stream-based* and *element-*

based synchronization refer to activities which attempt to control the timely interactions between related media data. The units of synchronization for stream-based and element-based synchronization are media streams and media elements respectively. The interactions between synchronization units may be within the same sequence (intra-sequence), across different sequences (inter-sequence), within the same stream (intra-stream), or across different streams (inter-stream).

Several methods have been proposed to address all or part of the media synchronization problem. One common approach for dealing with the case of inter-stream element-based synchronization (e.g., managing lip sync in movie clips) is to interleave the associated media streams. However, this scheme suffers from a number of different shortcomings — including the coupling of failure modes resulting from packet loss during transmission, and the inability to prioritize the data streams or handle them in a manner appropriate to their data type ([7], [8], [10]). An approach which attempts to address all types of element-based synchronization is to assign each media element a timestamp that represents the time at which it must be displayed ([1], [4]). While this scheme has the benefit of simplicity, it does not take into account the variations between the different physical clocks involved in actual systems (i.e. system, frame buffer, and audio codec clocks). Some approaches designed to provide general support for media synchronization in distributed settings are based on the use of synchronized system clocks ([5], [14]), while others employ centralized synchronization servers ([13], [15]). Work has also been done in other areas, such as operating systems ([3], [11], [12]) and network protocols ([6], [9], [18]), to provide support for media synchronization. These approaches have vastly different properties and address different aspects of the problem. Without a means for quantitatively measuring their performance, it is difficult to accurately and objectively determine what impact these schemes have on the achieved quality of media synchronization.

1.2 System Overview

This paper presents a methodology for quantitatively measuring the performance of media delivery systems. Below, we present an overview of a complete framework for measuring synchronization quality. In Section 2 we outline our experimental methodology, and Section 3 describes the implementation of our framework. Section 4 presents some experimental results, and we conclude in Section 5.

A generalized media delivery system is capable of capturing, manipulating, and presenting synchronized media data. Evaluating the performance of such a system requires a repeatable procedure for presenting the system with a stimulus and quantitatively measuring its output relative to an ideal response. To meet this goal we have developed a complete synchronization test environment as depicted in Figure 1. The environment is comprised of three components: the media delivery system being tested, a stimulus production system, and a measurement system. The components execute on separate machines in order to avoid interference effects.

The stimulus production system must generate a controlled stimulus and present it as input to the system under test. This stimulus must be a well-defined media sequence that requires synchronization, and it must be produced in a reliable, accurate, and repeatable manner. In practice, a media delivery system typically receives its input either from a storage system (as in a movie player) or from a live analog source (as in a teleconferencing application). To simulate typical use, the stimulus production system must be able to generate a sequence for both situations.

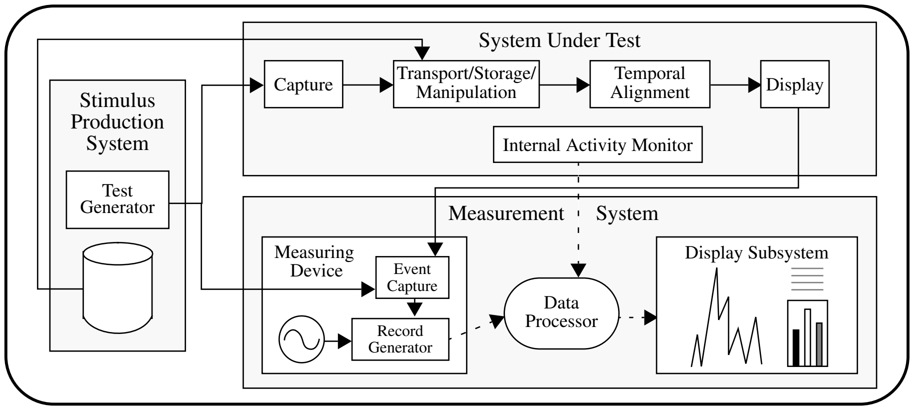

Figure 1. Schematic view of a synchronization test environment.

A stored sequence can be generated once and recorded in a file (in a format suitable for input by the system under test). All timing information of the media elements within the sequence can then be recovered by simply examining the properties of the stored data. If, however, live data are to be used, it is necessary to produce the sequence using a test sequence generator — a specialized media delivery system. The test generator must be able to display the defined sequence in a repeatable fashion and with a small error bound. In addition, the accuracy of the generated sequence must exceed the acquisition and synchronization accuracy of the system under test. Although the sequence generated by this method may be highly accurate, it is still not perfect. So, the output from the test generator is not only sent to the media capture device(s) of the system under test, but it is also sent to the measurement system (described below), which then characterizes the actual sequence timing for later comparison with the output from the system being tested.

Once the sequence is input to the test system, it may be stored, manipulated, transmitted across a network, etc. When the system is ready to display the sequence, it will analyze the synchronization requirements, attempt to temporally align the media elements to their ideal presentation times, and then finally display them. The system under test might also record information regarding its activity and pass it to the measurement system (described below). This information can then be correlated with the measured output of the system to establish causal relationships between externally- and internally-observed system behavior.

The measurement system contains three components: a measuring device, a data processor, and a display subsystem. The measuring device is responsible for collecting the output of both the test generator and the system under test. It then analyzes these data, looking for particular *media events*. A media event corresponds to the display of one or more semantically meaningful media elements, e.g. an audible click, a certain type of video frame, etc. For each media event that it receives, the measuring device generates an *event record* (or record). Records consist of a timestamp and an identifying tag. All generated records are output to the data processor, which is responsible for analyzing the data from the measuring device and compiling statistics on the timing of events. These statistics are then passed to the display subsystem, which presents graphical and tabular summaries of media synchronization quality for the system being tested. The measurement system provides an indication of the end-to-end synchronization performance of the system under test, and it can be used to isolate and quantify the effects of individual system components on that performance.

2 Experimental Methodology

Since displaying synchronized audio and video is such a common activity in practice, we have chosen to measure element-based intra-stream audio, intra-stream video, and inter-stream audio/video synchronization as the initial test cases. We use a timestamp-based audio/video test generator and a timestamp-based movie player as two sample media delivery systems. However, since we make no *a priori* assumption on the manner in which the problem is attacked, any system capable of displaying audio and video can be measured. We will conduct experiments with other media delivery systems in the future, and we also plan to measure other forms of synchronization performance. This section presents the design issues of the input sequence as well as the types of synchronization measurements that can be made.

2.1 Designing a Media Sequence for Controlled Input

To provide reliable measurements, the media sequence which is used as a controlled input must be well-defined so that all timing information is known *a priori*. Thus, given that the ideal display time of each media event is known, and since the actual display time is measurable, it is possible to detect any deviations. The sequence must also be defined in such a manner that the measurement system can reliably quantify these timing differences. For example, typical quartz oscillators have a frequency offset from ideal which is on the order of 100 parts per million (ppm). Without some method of compensation, this error accumulates over time and will result in media elements being displayed increasingly farther from their ideal presentation times. This effect is usually termed *drift*. Since this type of error becomes noticeable (to a human) only very slowly, the test sequence must be sufficiently long-running to allow the potential error to accumulate to an appreciable level. Also, the event inter-arrival times must be varied in order to avoid resonance effects with any other clocks in the system. For instance, if the scheduler executes every 10ms, a media delivery system displaying a stream with a period of 30ms between element presentation times may appear to have better (or worse) performance than the same system displaying a stream with a 33ms period.

For our initial experiments with simple audio/video synchronization, we defined the following sequence to satisfy the above requirements. To simplify matters, monaural audio is used, and the video is comprised of only two different frames: a solid black frame, and a black frame with a white square in the center. Two types of media events are recognized by the measuring device: video events, and audio events. A *video event* occurs whenever the type of video frame being displayed is changed, and an *audio event* occurs when a click sound is played. The sequence is defined so that audio and video events should occur simultaneously. This limits the granularity of synchronization to a video frame time. Clearly, this is the smallest meaningful unit for intra-stream video and inter-stream audio/video synchronization, but intra-stream audio synchronization could be measured at a finer granularity. While this is possible, it adds unneeded complexity since measuring the rate at which audio samples are output from the system can be much more easily achieved by monitoring the output voltage levels of the audio codec (a technique supported by the system we have implemented).

The test sequence runs for at least two hours — a sufficient length to allow drift to accumulate to a noticeable level. The sequence displays video at rates of 30Hz, 29.97Hz, 29Hz, and then cycles down to 1Hz with a step size of 1Hz. For each rate, the ratio of the duty cycles of white-square frames to black frames is varied as follows: 1:1 (one white-square frame, then one black frame), 1:2, 1:3, 1:5, 1:7. A pseudo-random sequence of frame types is also displayed for each rate. The sequence is defined in this way to avoid any possible resonance effects with clocks in the system. This pattern is repeated until the time limit for the test is reached.

2.2 Performance Measurements

The above measurement environment is defined in such a way that for each media event ε, the following values are available: the generation time $\tau_G(\varepsilon)$, the ideal display time $\tau_I(\varepsilon)$, and the actual display time $\tau_A(\varepsilon)$. Based on these values, we define six quantitative performance measurements. The *end-to-end latency* of a media event is defined as $\lambda_\varepsilon = \tau_A(\varepsilon) - \tau_G(\varepsilon)$. *Absolute asynchrony* represents the deviation of a particular media event from its ideal display time and is given by $\delta_\varepsilon = \tau_I(\varepsilon) - \tau_A(\varepsilon)$. *Relative asynchrony* is given by $\pi_{\varepsilon_i, \varepsilon_j} = \tau_A(\varepsilon_i) - \tau_A(\varepsilon_j)$, where ε_i and ε_j denote events from streams i and j respectively. It corresponds to the relative display time of nominally simultaneous events from different streams.

With these measures we can derive additional quantities for *skew*, *drift*, and *jitter*. Intuitively, given two sets of values over time (e.g. actual and ideal presentation times of media elements from a given stream), skew refers to a constant offset between pairs of values, drift denotes the amount by which the difference between value pairs changes over time, and jitter is characterized by the instantaneous variations in these values when skew and drift effects have been removed. These quantities can be computed for both absolute and relative asynchrony through simple statistical analysis techniques. For example, given a series of media events $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_N$ and the corresponding series of absolute asynchronies $\delta_{\varepsilon_1}, \delta_{\varepsilon_2}, \ldots, \delta_{\varepsilon_N}$, for each event ε_i we can plot absolute asynchrony (δ_{ε_i}) vs. ideal display time ($\tau_I(\varepsilon_i)$) on a graph and use a curve interpolation technique (e.g. linear least squares) to fit a straight line through these points. Then, the skew is given by the intercept of this line on the absolute asynchrony axis, the drift corresponds to the slope, and the dispersion about the line gives an indication of the amount of jitter. We can also plot a series of relative asynchrony values ($\pi_{\varepsilon_i, \varepsilon_j}$) against a series of actual display time values ($\tau_A(\varepsilon_i)$) and perform a similar analysis to yield skew, drift and jitter figures for stream j relative to stream i.

For our audio/video experiment, analyzing the statistical distributions of the absolute asynchronies for either type of stream provides an indication of intra-stream synchronization quality, while the distribution of relative asynchronies describes the effectiveness of inter-stream synchronization. The relative importance of each of the above measures is application- and media-dependent. For example, low latency is extremely important for interactive applications, and drift is only a concern for long-running sequences. Also, the level at which jitter becomes noticeable depends on the media type (e.g., tolerable video jitter is large compared to tolerable audio jitter). Further, the extent to which asynchronies can be tolerated by a viewer is dependent upon human perceptual limits as well as personal taste. Steinmetz and Engler conducted user studies in [17], and they report several figures of merit for quantifying tolerable asynchrony limits. This type of information must be taken into account when judging the merits of any media delivery system.

3 System Design

Utilizing the above experimental methodology, we implemented a complete test environment for measuring the synchronization performance of arbitrary media delivery systems. As an initial exercise of the capabilities of this system, our test generator and a locally developed media player were used as test cases for measurement. In this section we describe the implementations of our example media player, test sequence generator, and measurement system.

3.1 Media Player Implementation

The chosen test case media player was developed at Sun Microsystems Laboratories. It is a timestamp-based system capable of playing synchronized audio, video, and text streams. Media data are displayed at a single site, but may originate from arbitrary sources, including network connections and local capture devices. Each media stream flows under the direction of an independent thread of control. At the receiver, these threads communicate through a shared memory region, and use time stamps to synchronize the display of their media elements.

Since humans are more sensitive to intra-stream audio asynchronies (i.e. audio delays and drop-outs) than to asynchronies involving the video or text, the clock from the audio codec is used as a master reference to which all threads attempt to synchronize the display of their media elements. The thread controlling the audio stream free-runs, and the other "slave" threads use the information it posts into the shared memory region to determine when to display their elements. If a slave thread is ready to display its element early, then it delays until the appropriate time; but if it is late (>20ms behind audio), it discards its current element on the assumption that continued processing will cause further delays later in the stream.

The player uses the following data format. Video frames are captured at 320x240 pixel resolution and stored using JPEG compression. The audio data are standard 8-bit μ-law monaural samples. The average decompression and display time for the black and white-square frames was measured to be about 33.5ms, and the maximum data rate required (at 30Hz display) is 60.35KB/s for video and 7.8KB/s for audio. Given these values, the media player should be able to sustain the necessary display rates without encountering frequent overload conditions.

In addition, the environment we used to conduct our experiments provides a lightweight event tracing facility. Applications can make calls to tracing routines in order to log the time (relative to the system clock) that some internal event occurred. These tracing routines are very low-cost (a few microseconds). Thus, when used judiciously, they do not significantly perturb an application's normal behavior. By inserting tracepoints into the system under test, it is possible to correlate tracing information to the measured output data so that causal relationships between system activity and noted asynchronies can be determined.

3.2 Test Generator Implementation

We implemented a test generator capable of producing the audio/video sequence defined in Section 2.1. The test generator is an application which executes in the real-time scheduling class at the highest priority on a dedicated machine. It utilizes a simple peripheral I/O device to generate an NTSC composite video signal. This device can store two video frames and allows the user to select which frame is displayed. We use this feature to avoid moving data through the system so that the generator is not limited by any bandwidth constraints. When started, the test generator downloads the pixels for the black and white-square frames into these buffers, and then enters a control loop, waiting for the presentation times of the media events. It displays video events by sending a command to the video encoder telling it which frame to display, and it displays audio events by sending two digital audio samples to the machine's standard audio output device. These samples have the maximum allowable difference in amplitude and thus create an audible click.

This produces highly accurate results since the high priority level of the process and lack of other system activity ensure that deadlines can be met. Also, since each set of media events requires only three bytes of data to be moved, there are sufficient resources to ensure no overload exists. Further, we avoid any issues that normally arise when multiple clock domains are involved, i.e. the system, audio codec, and NTSC encoder clocks. This is because the audio/video data/commands are delivered to their respective devices under the control of the system clock and asynchronously with respect the audio codec and NTSC encoder clocks. While many implementations of test generators are possible, this choice represents a balanced trade-off between flexibility and performance.

3.3 Measurement System Implementation

The measurement system must be capable of measuring the media output from both the test generator and the system under test, and it must tag and timestamp any events that it detects and then pass them on for processing and display. To accomplish these goals we built a general-purpose, timestamp-generating I/O device called CHAOS (Chronological Hardware Activity Observation System). CHAOS consists of a set of input ports and a counter driven by the system clock. When a signal is detected on a port, a tag identifying the port is generated and attached to the current value of the counter. This timestamp record is then made available for processing. Since this device merely responds to signals at input ports, it is completely generic in its ability to log the times of external events. Thus, the actual display time of any type of media event can be accurately detected and used to measure synchronization quality for any of the classes mentioned in Section 1.1. In particular, we use the event records generated by CHAOS to compute the synchronization quality metrics presented above in order to obtain an accurate, quantitative measure of how well a given media delivery system performs.

For our initial experiments we utilized three inputs. The first comes from a microphone which acts as a transducer to gather audio data. When an audible click registers, a pulse is generated on an input port, causing CHAOS to create an *audio event record*. We use a photo-diode-based transducer to detect the white-square video frame and then send a pulse to a second input port. CHAOS generates a *video event record* for each frame time that the white square is displayed. The final input comes from a Global Positioning System (GPS) receiver and generates *time correction event records*, which are used to improve timestamp accuracy.

One of the most important aspects of the measuring device is the accuracy and precision of its clock, which must surpass that of the clocks used by the system under test. A precision time reference such as an atomic clock would provide an excellent timer for the device, but this is generally not practical due to the high cost and special handling requirements associated with such a device. Furthermore, the exceedingly high resolution of an atomic clock is unnecessary for measuring media synchronization, where the smallest meaningful units tend to be in the microsecond range (e.g. for stereo audio synchronization [17]). A much better approach is to use the standard quartz oscillator found in modern computers as the time base, but also have CHAOS tag and timestamp the input from a highly reliable chronometer.

An inexpensive GPS receiver can provide a precise 1Hz signal with short term error on the order of 10 parts per billion (ppb) and long term error on the order of 1 part per trillion (ppt) [2]. Given the nominal frequency of the system clock, the correct number of ticks between time correction events is known (e.g. for a 25MHz oscillator, the difference between consecutive time correction event records should be 25 million ticks). The measured value represents the true oscillator frequency and can be used to convert timestamps to real time. This frequency changes so slowly that it is sufficient to measure it once at the beginning of an experiment, and a mean computed over 60 seconds is adequate for obtaining a figure that accurately represents the true value [19]. For example, using an atomic-clock-based frequency counter, we measured the actual frequency of a 25MHz oscillator to be 25,001,109Hz. Using the GPS receiver we calibrated the same clock over a one minute interval and arrived at a frequency of 25,001,117Hz. This indicates that the clock is fast by a perceived amount of around 44µs/s.

4 Experimental Results

Using the test sequence, test generator, media player and measurement system described above, we conducted six experiments to obtain an initial assessment of the quantitative effects of some of the causes of asynchrony in modern systems. In the first five experiments we measured the test sequence generator under various conditions. Initially, the generator was run as a process in the real-time scheduling class on a stand-alone Sun SparcStation 20. No peripheral devices (except for the local disk) were attached, and no other processes (except standard system processes) were running. The output was generated using the standard speaker device and the NTSC video encoder. In the second experiment we attached the machine to the local network and repeated the test. In the next experiment we moved the test generator process into the time-sharing scheduling class, and in the fourth experiment we introduced additional load/processes by performing the test with the window system running. In the fifth experiment we rendered the video directly to the frame buffer. For each of these tests the sequence was played out of non-paged memory so that no disk effects are observed. In the final experiment we measured the performance of the media player. We used the same machine and executed the player in the time-sharing class. The same sequence was used, but the player read the data from a local disk in the format described in Section 3.1. For each experiment the system under test maintained internal state regarding the times at which it considered media elements to have been displayed (or discarded due to missed deadlines), and external measurements were made with the CHAOS device.

4.1 Internal Measurements

During each experiment the system under test gathered internal statistics by querying the system clock after the display of each media event and recording those timestamps. This is done for several reasons. First, such measures were the only indication of synchronization performance in the past, and are thus useful for comparison. Next, it provides an indication of how well the test system believes it is performing, and finally, it helps to isolate causes of poor synchronization. In the future we plan to utilize the tracing facility described in Section 3.1 to acquire more detailed information. The results of our experiments are summarized in Table 1.

Internal Measurements of Synchronization Performance											
Type	SUT	Dest	Sched Class	Net	Win	Skew (μsecs)	Drift (unit time/ unit time)	Jitter (μsecs)			Losses (%)
								Min	Max	Std Dev.	
A	TG	TV	RT			-61	3.6×10^{-10}	-20	610	17	0.003
A	TG	TV	RT	✔		-63	1.2×10^{-9}	-25	670	28	0.007
A	TG	TV	TS	✔		-46	5.7×10^{-9}	-54	61245	536	0.6
A	TG	TV	TS	✔	✔	-37	1.5×10^{-9}	-33	67266	471	0.8
A	TG	FB	TS	✔	✔	-16	6.1×10^{-11}	-49	64920	706	0.9
A	MP	FB	TS	✔	✔	0	0	0	0	0	0
V	TG	TV	RT			13	-1.1×10^{-10}	-13	597	22	0.002
V	TG	TV	RT	✔		14	6.2×10^{-10}	-19	754	37	0.003
V	TG	TV	TS	✔		27	6.0×10^{-9}	-57	61252	536	0.6
V	TG	TV	TS	✔	✔	35	1.6×10^{-9}	-36	67273	471	0.8
V	TG	FB	TS	✔	✔	3342	-1.1×10^{-8}	-586	250136	1583	0.9
V	MP	FB	TS	✔	✔	-109329	0.000051	-360736	464338	136877	0.9
AV	TG	TV	RT			62	1.4×10^{-10}	-5	11	2	0.004
AV	TG	TV	RT	✔		63	1.9×10^{-10}	-5	274	3	0.007
AV	TG	TV	TS	✔		74	3.1×10^{-10}	-15	496	10	0.6
AV	TG	TV	TS	✔	✔	72	1.4×10^{-10}	-14	293	4	0.8
AV	TG	FB	TS	✔	✔	3674	-1.6×10^{-8}	-180	249852	1689	0.9
AV	MP	FB	TS	✔	✔	-106879	0.000051	-361860	341950	137159	0.9

KEY:

SUT - system under test
TG - test sequence generator
MP - SML media player
TV - video displayed on television
FB - video displayed on frame buffer

A - intra-stream audio synchronization
V - intra-stream video synchronization
AV - inter-stream synchronization of video relative to audio

RT - real-time scheduling class
TS - time-share scheduling class
Net - machine connected to network
Win - window system running

Table 1. Synchronization performance as reported by the application displaying the test media sequence. Skew, drift, and jitter are the quantities described in Section 2.2, and losses refers to the percentage of media elements which were not displayed due to missed deadlines.

It can be seen from this table that in general the players performed very well. Measured skew is insignificant in the first four experiments, and drift is negligible in the first five. Except for the media player, jitter values are within reason, and in all cases few elements were discarded since sufficient resources were available. There are, however, some interesting observations that can be made.

First, note that the media player reports that it displayed the audio data with perfect synchrony. This is because it uses the audio codec clock as a master reference and assumes that the display of audio events cannot deviate from it, i.e. it does not generate timestamps corresponding to when the audio clicks are actually played. This also helps explain the reason for the non-negligible drift values it reports for the intra-stream video and inter-stream audio/video synchronization. Since the internal time stamps for the video events are generated by querying the system clock when the audio codec clock reports that display is complete, this drift reflects the fact that the system clock is about 51μs/s slow relative to the audio clock. This drift is an artifact of the way in which internal timestamps were generated, and hence the media player itself perceives no such drift.

Next, note that even in the best case, intra-stream audio jitter is high relative to the needs of stereo synchronization ([17]). This suggests that separate control of related audio streams is not feasible with this type of approach. In addition, all intra-stream measurements show dramatic increases in the amount of observed jitter when the system under test is placed in the time-sharing class, indicating that lack of appropriate scheduling support can adversely affect synchronization quality.

Finally, there is a dramatic drop in intra-stream video and inter-stream audio/video synchronization quality in experiment five, in which the test generator displayed video on the frame buffer. This is because the test generator was originally designed to use only the NTSC encoder to produce video, and so the expected time to display a video frame is hard-coded into the application. Thus, when a display path of differing length is encountered, it cannot adjust appropriately. This emphasizes the need for adaptive display algorithms which scale automatically to account for varying display lengths.

4.2 External Measurements

In addition, to the internal measures reported above, we used the CHAOS device to gather performance information for each experiment. These data represent what is actually observed by viewers. The results are summarized in Table 2, from which several interesting observations can be made.

External Measurements of Synchronization Performance

Type	SUT	Dest	Sched Class	Net	Win	Skew (µsecs)	Drift (unit time/ unit time)	Jitter (µsecs)			Losses (%)
								Min	Max	Std Dev.	
A	TG	TV	RT			—	0.000101	-70	587	30	0.004
A	TG	TV	RT	✔		—	0.000101	-139	730	51	0.007
A	TG	TV	TS	✔		—	0.000102	-61	867	35	0.6
A	TG	TV	TS	✔	✔	—	0.000101	-106	728	53	0.8
A	TG	FB	TS	✔	✔	—	0.000101	-50	723	22	0.9
A	MP	FB	TS	✔	✔	—	0.00015	-348425	231271	125036	0
V	TG	TV	RT			—	0.000100	-17645	17749	9586	0.002
V	TG	TV	RT	✔		—	0.000101	-17423	16771	9624	0.003
V	TG	TV	TS	✔		—	0.000101	-17153	62690	9690	0.6
V	TG	TV	TS	✔	✔	—	0.000101	-17348	66388	9610	0.8
V	TG	FB	TS	✔	✔	—	0.00019	-247842	1.027×10^9	426948	0.9
V	MP	FB	TS	✔	✔	—	0.00015	-364994	337882	136359	0.9
AV	TG	TV	RT			26095	-3.06×10^{-7}	-17584	17812	9587	0.004
AV	TG	TV	RT	✔		25114	9.4×10^{-8}	-17309	16667	9620	0.007
AV	TG	TV	TS	✔		25616	-1.5×10^{-7}	-17127	62552	9689	0.6
AV	TG	TV	TS	✔	✔	24411	1.7×10^{-7}	-17117	66326	9606	0.8
AV	TG	FB	TS	✔	✔	-138529	0.000092	-247917	1.027×10^9	4302378	0.9
AV	MP	FB	TS	✔	✔	-28377	-0.0000020	-55798	278188	26680	0.9

Table 2. Synchronization performance as measured with the CHAOS device. See Table 1 for a description of the symbols used in this chart.

First, note that no intra-stream skew numbers are reported. This is due to the fact that the actual starting time of the experiment cannot be determined with certainty, i.e. externally, we can only determine when the first event is displayed, not when the system under test began to display it. Inter-stream skew, on the other hand, is internally measured as being relatively small (except for the media player) but is observed as being substantially larger. In fact, the value for the experiment in which the test generator displayed video on the frame buffer falls within a range that would be easily noticed by a human observer ([17]). These differences between internal and external measures can be attributed to the fact that the system under test has no means for determining how long it takes for a media element to appear as output. Instead, it only knows when the element was last under its control, and there can clearly be a substantial time difference between those two events.

Next, since the tested systems control the display time of media elements using a single time base (either the system clock or the audio clock) and since no external reference is utilized, the internally measured values for drift were negligible (except as noted above). However, most of the external measurements exhibit a significant amount of drift. For example, in the first four experiments the test generator references only the system clock to produce its output. Hence, the measured intra-stream drift values in these cases indicate that the system clock is about 100µs/s slow relative to real-time. Additional measurements of the system clock accuracy confirmed that this was indeed the case. Similarly, the other observed drift values indicate differences between other pairs of clocks (e.g. audio clock vs. real-time, frame buffer clock vs. real-time). In addition, the intra-stream video and inter-stream audio/video drift values reported internally by the media player in the last experiment do not correspond to the externally measured values. This is because the internally reported drift is due to differences between the audio and system clocks, whereas the external drift is due to differences between the audio clock and real-time. Drift can result whenever multiple time references are used, and these observations make it clear that without an external reference, a media delivery system has no means to characterize it and will operate under the false notion that it is correctly displaying media elements.

Other interesting features can be observed by comparing the internal and external values for jitter. Most significantly, it is clear from the tables that externally observed jitter is substantially worse than jitter reported by the systems under test. In addition, when the test system is moved into the time-sharing class in the third experiment, the external measurements do not reflect the drop in jitter quality that was reported internally. This can most easily be understood by examining intra-stream video synchronization for the first four experiments, which display video using the NTSC encoder. In these cases the dispersion of the jitter is around 9.6ms about the mean. The NTSC encoder outputs video frames with a period of about 16.7ms. When a toggle command arrives at the encoder, it will be deferred until the next period. This suggests that the output should on average be delayed about 8.35ms, which corresponds well to the observed jitter. Since the period is so large relative to the internally observed jitter values, it will effectively mask the large drop in jitter quality when the time-sharing class is utilized. The lack of discontinuities in the other experimental data can be attributed to similar effects. This is significant since it suggests that accounting for the clock rates of the output devices is more important over a long interval than scheduling decisions.

5 Conclusions

This work provides a means of determining the actual effectiveness of arbitrary media synchronization schemes. First, several forms of media synchronization were defined, and then a methodology was presented for measuring the degree of synchronization which is achieved both within and among streams of media elements. The defined approach was validated by implementing a complete test environment for measuring the audio/video synchronization performance of media delivery systems. The measurements match the expected performance of the tested systems extremely well, and thus give us confidence in our experimental design. In addition, the results clearly indicate that internal assessments of synchronization performance are insufficient. Accurate appraisals of the quality of media synchronization cannot be achieved without an external reference.

In the future we will use this methodology and test environment to evaluate different mechanisms for media synchronization, as well as the effects that changes in the system have on the provided synchronization quality. Furthermore, this tool forms the foundation for an effort to identify and quantify the various components of media players that are the major contributors of display time variation.

Acknowledgements

We extend our sincere thanks to David Lee who developed the internal logic of the CHAOS board, to Marc Schneider who also helped construct the test setup, and to Jim Hanko and Jerry Wall for many insightful discussions.

References

1. D. P. Anderson and G. Homsy, "A Continuous Media I/O Server and Its Synchronization Mechanism," *IEEE Computer*, 24(10), October 1991, pp. 51–57.
2. Bancomm, *bc700VME GPS Satellite Receiver Operation and Technical Manual*, October 1991.
3. D. C. A. Bulterman and R. van Liere, "Multimedia Synchronization and UNIX," in *LNCS*, 614, R. Herrtwich (Ed.), Springer-Verlag, 1992.
4. J. A. Boucher, Z. Yaar, E. J. Rubin, J. D. Palmer, and T. D. C. Little, "Design and Performance of a Multi-Stream MPEG-I System Layer Encoder/Player," in *IS&T/SPIE Symposium on Electronic Imaging Science & Technology*, San Jose, CA, February 1995.
5. J. Escobar, C. Partridge, and D. Deutsch, "Flow Synchronization Protocol," *IEEE/ACM Transactions on Networking*, 2(2), April 1994, pp. 111–121.
6. D. Ferrari, "Design and Applications of a Delay Jitter Control Scheme for Packet-Switching Internetworks," *Computer Communications*, 15(6), July/August 1992, pp. 367–373.
7. K. Jeffay, D. L. Stone, and F. D. Smith, "Kernel Support for Live Digital Audio and Video," in *Computer Communications*, 15(6), July/August 1992, pp. 388–395.
8. P. Leydekkers and B. Teunissen, "Synchronization of Multimedia Data Streams in Open Distributed Systems," in *LNCS*, 614, R. Herrtwich (Ed.), Springer-Verlag, 1992, pp. 94–104.
9. T. D. C. Little and F. Kao, "An Intermedia Skew Control System for Multimedia Data Presentation," in *LNCS*, 712, V. Rangan (Ed.), Springer-Verlag, 1993.
10. C. Nicolaou, "An Architecture for Real-Time Multimedia Communication Systems," *IEEE JSAC,* 8(3), April 1990, pp. 391–400.
11. J. Nieh, J. G. Hanko, J. D. Northcutt, and G. A. Wall, "SVR4 UNIX Scheduler Unacceptable for Multimedia Applications," in *LNCS*, 846, D. Shepherd, et. al. (Eds.), Springer-Verlag, 1994.
12. J. D. Northcutt and E. M. Kuerner, "System Support for Time-Critical Applications," *Computer Communications*, 16(10), Oct. 1993, pp. 619–636.
13. S. Ramanathan and P. V. Rangan, "Adaptive Feedback Techniques for Synchronized Multimedia Retrieval over Integrated Networks," *IEEE/ACM Transactions on Networking*, 1(1), February 1993.
14. L. A. Rowe and B. C. Smith, "A Continuous Media Player," in *LNCS*, 712, V. Rangan (Ed.), Springer-Verlag, 1993.
15. S. H. Son and N. Agarwal, "Synchronization of Temporal Constructs in Distributed Multimedia Systems with Controlled Accuracy," *International Conference on Multimedia Computing and Systems*, May 1994, pp. 550–555.
16. R. Steinmetz, "Synchronization Properties in Multimedia Systems," *IEEE JSAC,* 8(3), April 1990, pp. 401–412.
17. R. Steinmetz and C. Engler, "Human Perception of Media Synchronization," *Technical Report 43.9310*, IBM European Networking Center, Heidelberg.
18. D. L. Stone and K. Jeffay, "An Empirical Study of Delay Jitter Management Policies," To appear in *Multimedia Systems*.
19. D. B. Sullivan, D. W. Allan, D. A. Howe and F. L. Walls, "Characterization of Clocks and Oscillators," *NIST Technical Note 1337*, March 1990.