# TSync: A New Synchronization Framework for Multi-site 3D Tele-immersion

Zixia Huang, Wanmin Wu, Klara Nahrstedt, Ahsan Arefin, Raoul Rivas
Department of Computer Science
University of Illinois at Urbana-Champaign
{zhuang21, wwu23, klara, marefin2, trivas}@illinois.edu

## ABSTRACT

Synchronization is a challenge in the multi-site 3D tele-immersion (3DTI) because it is complicated by the coexistence of highly correlated heterogenous streams from multiple sources, and the need for multi-stream resynchronization when user views change. To address the problems, we present TSync: a new multi-tier synchronization framework for 3DTI which can effectively reduce the multi-stream sync skews. Our contributions are focusing on (1) the use of timed synchronization points for multi-stream synchronization; (2) Internet bandwidth estimation based on machine learning; (3) the cooperative frame rate allocation for correlated multi-streams and (4) the resynchronization protocol used when user views change. Experimental results show that TSync can successfully achieve the synchronization of multi-source heterogeneous streams in 3DTI under Internet dynamics.

## Categories and Subject Descriptors

H.5.1 [**Multimedia Information Systems**]: Video; C.2.1 [**Network Architecture and Design**]: Network communications

## General Terms

Design, Performance, Experimentation

## Keywords

3D Tele-immersion, synchronization

## 1. INTRODUCTION

3D tele-immersion (3DTI) is an application that creates a joint virtual space among geographically distributed users for realistic collaborations. Each 3DTI system usually includes three tiers. In the *capturing tier*, multiple 3D cameras and their processing computers produce multi-view 3D video streams in real time. Audio signals are sampled and encoded to allow voice conversations. In the *transmission tier*, audio and multi-view 3D video streams are multiplexed at a local gateway and forwarded to the remote sites. In the *rendering tier*, both local and remote 3D video streams are aggregated and rendered into a joint virtual space at the display, while audios are mixed and played at the speakers.

The synchronization of multiple sensory streams (e.g., audio and video) becomes a key challenge in 3DTI due to several characteristics. First, the audio streams have to be synchronized with multiple highly correlated video streams. The correlation results from the deployment of the 3D cameras in different positions to capture multiple views of the same scene. Second, because of 3DTI's multi-tier architecture, a sync skew between any two streams in one tier can be propagated to the next tier. Third, the sync skew can be aggravated during its propagation. The sensory streams in 3DTI have heterogeneous Quality-of-Service (QoS) requirements, because different streams employ their own protocols and adaptation algorithms due to their different characteristics. This heterogeneity will inevitably result in different end-to-end latencies. Fourth, due to the multi-sensory nature of 3DTI systems, the sensory streams have to be carefully coordinated in all tiers in order to achieve proper synchronization. Fifth, Internet dynamics must be taken into account and timely frame rate allocation thus becomes a key issue. Sixth, user dynamics (e.g., change of user views [16]) should also be considered and resynchronization should be supported.

Although there have been numerous studies on multimedia synchronization, their contributions in 3DTI are very limited. Early in 1990s, [3] classified different synchronization techniques used to control jitters. For audio-visual streaming, we can achieve lip synchronization by scheduling the rendering time [9, 10] or warping audio signals in time domain [4]. In distributed multimedia environments, we can synchronize multi-streams at the sender side by multicasting a global timestamp, or applying *Precision Time Protocol* (PTP) [1] at the renderer. However, none of these schemes consider the synchronization of multi-source highly correlated streams under their heterogenous QoS, nor do they discuss the concept of resynchronization under Internet and/or user dynamics.

In this paper, we design TSync: a synchronization framework which can effectively minimize the multi-stream sync skews in all tiers. Unlike previous studies, our major contributions are as follows. (1) We introduce the concept of *timed synchronization point* and apply it in the 3DTI system to synchronize multi-streams with heterogenous QoS


*NOSSDAV'10,* June 2–4, 2010, Amsterdam, The Netherlands.

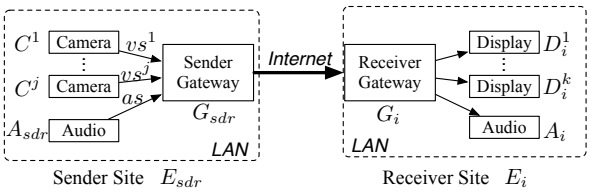


**Figure 1: 3DTI architecture with 2 environments.**

and prevent sync skew propagations. (2) We impose machine learning on bandwidth estimation under Internet dynamics. (3) We develop the cooperative frame rate allocation scheme for correlated multi-streams to facilitate their coordinated delivery. (4) We propose a protocol for smooth resynchronization under user dynamics. We use the recommended in-sync region of a maximum 80-msec skew [13] as a design guideline. We consider the conversational scenario where end-to-end delays for both video and audio streams are important.

The rest of the paper is organized as follows. Section 2 discusses the 3DTI system model. Section 3 presents a multi-tier overview of 3DTI synchronization issues and the TSync framework. Section 4 describes the design of TSync core components. Section 5 is the performance evaluations. Section 6 concludes the paper.

## 2. 3DTI SYSTEM MODEL

We classify 3DTI sites to be either *sender* or *receiver*. Let $\mathcal{E}_{sdr}$ denote the set of sender sites, and $\mathcal{E}_{rcv}$ be the set of receiver sites. Throughout the paper, we focus on the single sender (denoted as $E_{sdr}$) scenario, i.e., $\|\mathcal{E}_{sdr}\| = 1$. There can be multiple distributed receiver sites $\mathcal{E}_{rcv} = \{E_i\}$. As Fig. 1 shows, the sender site contains a set of source computing nodes $E_{sdr} = \{\mathcal{C}, G_{sdr}, A_{sdr}\}$ where $\mathcal{C} = \{C^j\}$ are the cameras, $G_{sdr}$ is the sender gateway, and $A_{sdr}$ is the audio microphone component. The $i$-th receiver site contains $E_i = \{G_i, \mathcal{D}_i, A_i\}$, where $G_i$ is the receiver gateway, $\mathcal{D}_i = \{D_i^k\}$ the displays or renderers, and $A_i$ the audio speaker component. A path from the sender to a receiver may be routed through an overlay constituting several *intermediate sites* [15]. In terms of receivers, we consider three scenarios: (1) *single-receiver, single-display*: $\|\mathcal{E}_{rcv}\| = 1, \|\mathcal{D}\| = 1^*$; (2) *multi-receiver single-display*: $\|\mathcal{E}_{rcv}\| \geq 2, \|\mathcal{D}_i\| = 1$ and (3) *single-receiver, multi-display*: $\|\mathcal{E}_{rcv}\| = 1, \|\mathcal{D}\| \geq 2$. Due to the space limit, we will leave our *multi-receiver multi-display* scenario (ViewCast [15]: $\|\mathcal{E}_{rcv}\| \geq 2, \|\mathcal{D}_i\| \geq 2$).

The data streams from site $E_{sdr}$ are denoted as $\mathcal{S} = \{\mathcal{AS}, \mathcal{VS}, \ldots\}$, where $\mathcal{AS} = \{as^j\}$ is the set of audio streams, and $\mathcal{VS} = \{vs^j\}$ is the set of video streams. We assume the sender site produces one audio stream, i.e., $\|\mathcal{AS}\| = 1$. Each stream $s$ (either $as$ or $vs^j$)† consists of frames $\{f_1, f_2, \ldots\}$. The set of video frames that is taken from different cameras at the same time at one site are called a *macroframe*. The frame rate of $s$ can be different at its source $C^j$ or $A_{sdr}$, the local gateway $G_{sdr}$, the receiver gateway $G_i$, and the display $D_i^k$, for which the frame rates are denoted as $FR_s^p$, $FR_s^{gs}$, $FR_s^{gr}$, $FR_s^d$, respectively.

We define the concept of a *display dominant stream* (DDS) $dds_i^k$ to be the most important video stream among the correlated multi-source stream bundle for $D_i^k$. The selection

*The subscript $i$ is omitted in the paper when $\|\mathcal{E}_{rcv}\| = 1$.

†The superscript $j$ for the audio stream is omitted in the paper because $\|\mathcal{AS}\| = 1$.

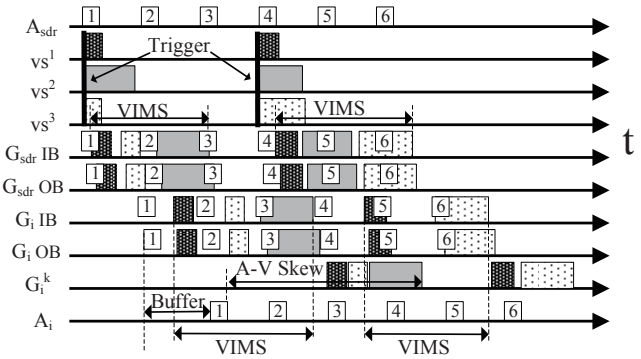


**Figure 2: 3DTI Sync Issues. IB: incoming(receiving) buffer. OB: outgoing(sending) buffer. The second frame of $vs^2$ is dropped by the Internet.**

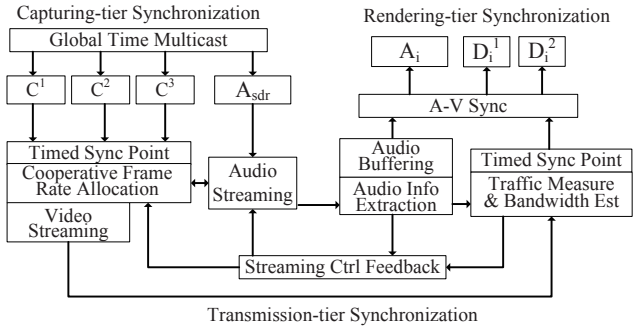


**Figure 3: TSync overview.**

of DDS for each display is based on maximizing *contributing factor* (CF) which is determined by $\vec{O}_{vs^j}$ (orientation of camera for $vs^j$) and $\vec{O}_{u_i^k}$ (orientation of the user view for $D_i^k$): CF = $\vec{O}_{vs^j} \cdot \vec{O}_{u_i^k}$ [16]. We denote a set of DDS for all displays at a receiver site $E_i$ as $\mathcal{DDS}_i = \{dds_i^1, dds_i^2, \ldots\}$. Because it is impossible to synchronize the audio stream with multiple correlated video streams, we also define the *receiver site dominant stream* (RDS) $rds_i$ to be the most important stream among $\mathcal{DDS}_i$ at the receiver site $E_i$ and it is the actual video stream that is used to be synchronized against the audio. The RDS $rds_i$ is selected from $\mathcal{DDS}_i$ such that $rds_i = \arg\max_s \{\sum_{k=1}^{\|\mathcal{D}_i\|} \vec{O}_s \cdot \vec{O}_{u_i^k},\ s \in \mathcal{DDS}_i\}$. Note that we limit both the number of DDS for each display and the number of RDS for each receiver site to be one. Other video streams in the multi-source correlated bundle that are neither RDS nor DDS are called *non-dominant streams* (NDS), and are denoted as a set $\mathcal{NDS}_i$.

## 3. A MULTI-TIER OVERVIEW OF TSYNC

We add the audio components to current 3DTI implementation [15]. The synchronization issues are illustrated in Fig. 2. We provide an overview on how these issues are addressed in TSync framework (Fig. 3).

**• Capturing-tier**

Each 3DTI site should enforce the synchronous capturing of correlated multi-source streams. In our previous work [15] a video trigger has been adopted at each sender site to allow synchronous capturing of images (Fig. 2). Basically the trigger sends a hardware "grab" signal to all 3D cameras at its site through wired cords when they become ready to capture the next frame. When the signals are sent in TSync, we also send a small soft packet piggybacking a global timestamp

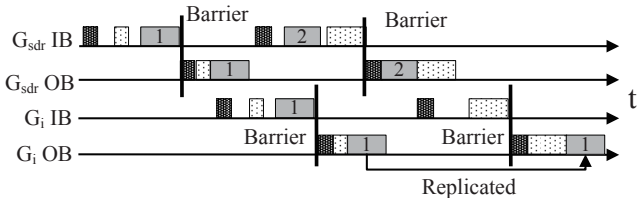


**Figure 4: Timed synchronization points (Barrier) in 3DTI. IB: incoming buffer; OB: outgoing buffer.**

$T_g$ to all cameras as well as the audio microphone component in order to facilitate audio-visual synchronization in the rendering tier. Because audio frames are transmitted more often than video frames, we also maintain a local timestamp in the audio stream for intra-stream synchronization at the audio speaker component at the receiver site.

To guarantee source reliability, we use TCP to send audio and video frames from the source cameras and audio speaker component to the sender gateway. Its congestion control and retransmission mechanism will cause asynchronous packet arrivals. We define *video intra-macroframe skew* (VIMS) as the arrival time skew between the first and the last frame belonging to the same macroframe (Fig. 2). To synchronize the video macroframe at the sender gateway, we apply the timed synchronization point in TSync which, due to the 3DTI real-time characteristics, has a bounded time waiting for the arrival of a video macroframe before it is placed in the outgoing (sending) buffer at the sender gateway (Section 4.1).

**• Transmission-tier**

Traffic stream differentiation affects the multi-stream synchronization. Between the sender and receiver gateways in 3DTI, we use UDP for audio transmissions to guarantee a constant frame period, while video utilizes DCCP [8] to adapt to Internet congestions. When the multi-stream data rate exceeds the network bandwidth capacity, both video and audio frames can be dropped over the Internet, and the DCCP sending rate will be reduced in response to the congestion. These problems will create huge synchronization issues at the receiver gateway. One solution is to estimate the network bandwidth and allocate the frame rates for each stream in different scenarios. Redundant frames can be discarded at the sender gateway. However this solution is complicated by the Internet dynamics, the correlated multi-stream coordination, and the diverse multi-site topology. We will address the issues in Section 4.2 and 4.3.

To avoid Internet congestions and losses at the sender side, we apply the time spacing technique to video streaming in TSync. We use a simple strategy that two consecutive frames within a synchronized video macroframe are sent in a separation of 5 msec and two consecutive fragmented packets within a video frame in a separation of 10 $\mu$sec. The time spacing and the Internet dynamics will introduce jitters (and thus VIMS variations in Fig. 2) at the receiver site. In order to resynchronize all video streams so that a complete macroframe can be sent to the renderer, we need to apply the timed synchronization point at the receiver gateway as well (Section 4.1).

**• Rendering-tier**

In TSync, we use a simplified version of our previous study [11] to allow lip synchronization. The audio buffering time is the sum of average 3D video rendering time (15 msec on average per macroframe including 4 video streams [14]), and an empirical 80th percentile audio-visual arrival skew (the arrival time skew between the last frame of a video macroframe and the audio frame with the same global timestamp $T_g$) in the previous 2 seconds at the receiver gateway. We assume the audio playing time is negligible. Note that the audio buffer should only be adapted during silence periods and a minimal audio buffering time of 60 msec should be guaranteed [6]. We also propose a resynchronization protocol for the smooth view change in Section 4.4.

# 4. TSYNC CORE DESIGN

## 4.1 Timed Synchronization Points

Fig. 4 shows the deployment of timed synchronization points at both sender and receiver gateways to deal with the asynchronous multi-stream arrivals. This can be achieved by the timed *barrier* implementation. Only video streams are affected by the barrier in the current design.

The timed barrier will immediately be released if a video macroframe is completely received within the bounded time. Otherwise it needs to wait until the bounded time expires before it releases the macroframe. If a frame is missing or incomplete at the release, the barrier needs to replicate the previous correctly received frame belonging to the same video stream. The amount of waiting time depends on where the barrier is located. At the sender gateway, an incomplete video macroframe will not be placed in the outgoing buffer until the gateway starts to receive the next macroframe from the source cameras. At the receiver gateway, the latest barrier release time is either when it starts to receive the next macroframe from the sender gateway, or the 60 msec behind the audio playing time with the same timestamp $T_g$, whichever is earlier. We use 60 msec because the recommended in-sync region is 80 msec and we leave 20 msec for video rendering time [14]. If the multi-streams are relayed by an intermediate site, a barrier is deployed at this site to synchronize a video macroframe before it is released and forwarded to other sites. The waiting time incurred on the barrier from the last release is $TD + \Delta T$ in msec, where $TD$ is the duration between the last release and the arrival of the first frame belonging to the new macroframe, and $\Delta T$ is the expected VIMS. Note that a longer $\Delta T$ will accommodate larger VIMS, reduce the missing frame rate and hence improve 3D video quality. However, $\Delta T$ is directly related to the end-to-end latency, so a longer $\Delta T$ will degrade the interactivity of 3DTI systems. We should balance a tradeoff between the video quality and the interactivity in deciding $\Delta T$. In Section 5, we set $\Delta T = 50$msec so that the one-way end-to-end latency can be bounded by 400 msec recommended in ITU G.114 [7] for up to three intermediate sites [15].

## 4.2 Bandwidth Estimation based on Machine Learning

A real-time 3DTI system requires huge amounts of bandwidth for its multi-stream transmission. Hence it is important to determine the network's *effective bandwidth* $EB$, the maximum bandwidth allowed for 3DTI streaming such that no congestions will be incurred (overall packet loss rate (LR) for the multi-stream bundle is less than 5% in this paper). However $EB$ is unlikely to be measured at run time using existing tools such as [5], because the traffic flows used for

bandwidth measurement can affect the throughput of the 3DTI multi-streams. Simple techniques like measuring the DCCP throughput at the receiver gateway are good only when $EB$ is less than or equal to the multi-stream bandwidth requirement. $EB$ can be under-estimated when it far exceeds the multi-stream bit rate. In addition, the measured DCCP throughput at the receiver gateway may not automatically tell if $EB$ increases due to a congestion reduction, and this drawback will prevent future adaptations. Our approach is to give a rough estimation of $EB$ between the sender and receiver gateways at run time using the machine learning approach based on the eight traffic statistics: (1) audio jitter (>60 msec) percentage , (2) average video throughput, (3) average VIMS, (4) maximum VIMS, (5) average macroframe size (6) the macroframe size corresponding to the maximum VIMS, (7) overall LR and (8) number of video streams. We have noticed that the eight parameters will affect $EB$ either through linear or non-linear relations, and we believe they are sufficient to model $EB$. We take the number of video streams into account because it will affect VIMS due to the time spacing.

In order to find a mapping $\mathcal{F}$ between $EB$ (called dependent variable, denoted as $y$) and the eight statistics (called independent variables, denoted in a vector expression $\vec{x}$ with each statistic $x_i$), we use *support vector regression* (SVR) [12] for machine learning. The general idea of SVR is to find a linear hyperplane function $\mathcal{F} = \vec{w} \cdot \vec{x} + b$ such that the deviations of $\mathcal{F}(\vec{x})$ from the target $y$ are minimized. To solve nonlinear relations between $\vec{x}$ and $y$, a mapping of $\vec{x} \rightarrow \phi(\vec{x})$ using a radial basis kernel function has to be applied.

We use LIBSVM [2] to do the regression. We first need to build the regression model and establish a mapping between $\vec{x}$ and $y$ in the off-line training process. We take the following steps: **Step 1**: we throttle the target bandwidth capacity (i.e. $EB$ or $y$) to be 10Mbps using *tc* software in Linux. We vary the transmission data rate of 3DTI video streams at the sender gateway, so that the resulting eight traffic statistics ($\vec{x}$) can exhibit variations. **Step 2**: we iterate step 1 by varying target bandwidth capacity from 10Mbps to 50Mbps (in a separation of 5Mbps), hence SVR can see a variety of situations. Based on $\vec{x}$ and the corresponding $y$, LIBSVM will solve the optimal hyperplane parameters and the prediction mean squared error (MSE) $\sigma^2$ of the trained model.

Given the trained regression model, we can now apply it at run time to estimate $EB$ in TSync by inputing the eight independent variables to LIBSVM. Due to the Internet dynamics, we only use the average of the most recent 10-sec traffic information. In our experiment, we can achieve a prediction MSE $\sigma^2 = 27.2$ ($\sigma = 5.2$ Mbps).

Of course, there may be unseen network conditions that can over-estimate $EB$ (overall LR is larger than 5%), we record $\vec{x}$ that predicts this $EB$. Every second we reduce the frame rate by one for each video stream, until the overall LR is within the 5% threshold. We compute the new incoming throughput at the receiver gateway and use it to replace $EB$. Along with all the off-line training data, the updated $EB$ and the recorded $\vec{x}$ can be used to retrain the regression model and improve the prediction accuracy.

## 4.3 Cooperative Frame Rate Allocation

In [16], we adapt the bandwidth availability by the dynamic allocation of multi-stream frame sizes based on CF. However this method does not address the synchronization issues and will create additional computational overhead. In TSync framework, we propose an alternative cooperative frame rate allocation scheme. Its main purpose is to allow the largest possible frame rate for the audio stream $as$, video RDS and DDS in the scheme to facilitate audio-visual synchronization and achieve good audio and 3D video quality. The allocation is based on the regression result $EB$.

To determine the proper frame rate $FR_s$ of a stream $s$ (either $as$ or $vs^j$) at a given $EB$, we need to estimate the future frame size $FS_s(n)$, where $n$ is the current frame index. This can be achieved by a linear predictor: $FS_s(n+1) = \sum_{l=0}^{L} a_l \times FS_s(n-l)$, where $L$ is the predictor order and $a_l (l = 0, 1, \ldots, L)$ are the coefficients. In TSync these coefficients are computed using the Levinson-Durbin recursion approach based on the frame size of the first 50 consecutive frames of stream $s$ received at the sender gateway from the source cameras and audio microphone component. We tested different order $L$, and found that any $L \in$ [6 10] can achieve a prediction accuracy of 90% of samples within 5% deviation and 99% samples within 10% deviation. We simply choose $L = 8$ in TSync. Note that to make the computations consistent, in this section we represent $EB$ in 'bits/sec', $FR_s$ in 'frames/sec' and $FS_s$ in 'bits'.

We discuss our frame rate allocation scheme in the three scenarios mentioned in Section 2. In all scenarios, the $as$ should synchronize to RDS. In the first two scenarios, because each receiver site has only one display, RDS of the receiver site is the same as the DDS of its display. In scenario 3, RDS should be determined among all DDS at the receiver site based on the strategy mentioned in Section 2.

**• Scenario 1: Single-receiver Single-display**

The sender gateway determines $FR_s$ for each stream $s$ based on the RDS information and regression estimation $EB$. We simply prescribe that all other non-dominant video streams share the same frame rate at the receiver gateway and the renderer. We denote $rds$ as the receiver site RDS (same as the DDS in this scenario), and $EB_{\text{NDS}} = EB - (FR^p_{rds} \times FS_{rds} + FR^p_{as} \times FS_{as})$. If $EB_{\text{NDS}} \geq 0$, $FR^{gs}_s$ can be determined by:

$$FR^{gs}_s = \begin{cases} FR^p_{as} & s = as \\ FR^p_{rds} & s = rds \\ EB_{\text{NDS}}/(\sum_{s \in \mathcal{NDS}} FS_s) & s \in \mathcal{NDS} \end{cases} \quad (1)$$

If $EB_{\text{NDS}} < 0$, there is not enough bandwidth for $rds$. Let $EB_{rds} = EB - FR^p_{as} \times FS_{as}$, $FR^{gs}_s$ is computed as:

$$FR^{gs}_s = \begin{cases} FR^p_{as} & s = as \\ EB_{rds}/(FS_{rds}) & s = rds \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

**• Scenario 2: Multi-receiver Single-display**

In this scenario, $\|\mathcal{E}_{rcv}\| = M$ and $\|\mathcal{D}_i\| = 1$. Each multi-stream bundle destined to one receiver behaves as background traffic of other receivers, so we estimate the bandwidth and allocate the frame rate for each receiver individually. We use Eq. 1 and 2 to compute $FR^{gs}_s$ for each stream $s$ targeted to receiver site $E_i$ based on its $rds_i$ and $EB_i$. $M$ regression estimations should be conducted in this scenario. The audio stream $as$ only synchronizes to $rds_i$ at each receiver site $E_i$, and there is no guarantee for inter-site synchronous rendering among multiple receiver sites.

**• Scenario 3: Single-receiver Multi-display**

In this scenario, $\|\mathcal{E}_{rcv}\| = 1$ and $\|\mathcal{D}\| = M$. We select $rds$ among $\mathcal{DDS} = \{dds^1, \ldots\ dds^M\}$. $EB$ is estimated

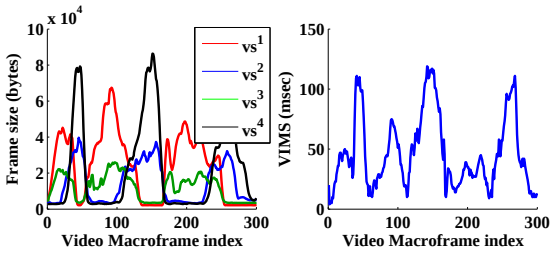


**Figure 5: Video frame variations of the 4 video streams and corresponding VIMS at the sender gateway.**

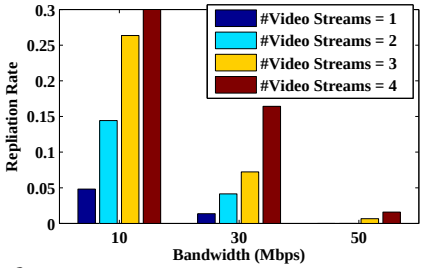


**Figure 6: Replication rate at the renderer. Replication rate for 4 streams at 10Mbps is more than 50%, and is not shown completely in the figure.**

between the sender and the receiver gateways. If $EB \leq (FR^p_{as} \times FS_{as} + FR^p_{rds} \times FS_{rds})$, we use a similar approach as in Eq. 2 to determine the frame rate of $rds$. Otherwise we let $EB_{\text{NDS}} = EB - (FR^p_{as} \times FS_{as} + FR^p_{rds} \times FS_{rds} + \sum_{s \in \mathcal{DDS}, s \neq rds} FR^p_s \times FS_s)$. We prescribe that all $s \in \mathcal{DDS}, s \neq rds$ share the same $FR^{gs}_s$ in this paper. If $EB_{\text{NDS}} \geq 0$, we compute $FR^{gs}_{s \in \mathcal{NDS}}$ similar to Eq. 1. If $EB_{\text{NDS}} < 0$, two methods can be applied.

• **Method 1**: we can reduce the uniform $FR^{gs}_{s \in \mathcal{DDS}, s \neq rds}$ to:

$$\frac{EB - (FR^p_{as} \times FS_{as} + FR^p_{rds} \times FS_{rds})}{\sum_{s \in \mathcal{DDS}, s \neq rds} FS_s} \quad (3)$$

However this method may degrade the user experiences if the video frame rate is below a threshold $FR_{\text{th}}$, the minimal frame rate that can provide acceptable rendering quality (e.g. 5 frames/s). Hence we propose the second method.

• **Method 2**: we can reduce the number of DDS(excluding $rds$) until $EB_{\text{NDS}} \geq 0$. This can be achieved by iterations. In each iteration, we remove the $dds^j$ whose $FS$ is the largest among all DDS(excluding $rds$). We replace the $dds^j$ with $dds^k$ from the remaining DDS(including $rds$) such that the resulting CF can be maximized. For example, in a $M = 3$ scenario, we substitute $dds^1$ (the $rds$) for $dds^2$ because $FS_{dds^2}$ has the largest frame size and $\vec{O}_{dds^1} \cdot \vec{O}_{u^2} > \vec{O}_{dds^3} \cdot \vec{O}_{u^2}$.

## 4.4 Resynchronization Protocol

When a user requests a view change at a display, the DDS may change to another video stream which either may not be available or does not have enough frames to render due to frame rate allocation. Hence a resynchronization framework should be developed to allow smooth view changes for all the three scenarios. Suppose the original DDS is $vs^k$, the details of the protocol are illustrated as follows.

• **Step 1**: upon the request of a view change, the new DS ($vs^j$) is determined and $FR^{gs}_s$ of each stream $s$ is updated based on the allocation scheme in Section 4.3.

• **Step 2**: if $FR^{gr}_{vs^j}$ is above the threshold $FR_{\text{th}}$, the renderer immediately changes DDS to $vs^j$. Otherwise it determines the video stream set $\mathcal{S}_{\text{rvs}}$ that satisfies: $\forall\ s$ in $\mathcal{S}_{\text{rvs}}$, $FR^{gr}_s \geq FR_{\text{th}}$. If $\mathcal{S}_{\text{rvs}}$ is not empty, during the changing period we temporarily use the stream in $\mathcal{S}_{\text{rvs}}$ with the largest resulting CF (to the new view). Otherwise, we simply keep the original stream $vs^k$ for DDS until $FR^{gr}_{vs^j}$ is greater than $FR_{\text{th}}$.

# 5. EXPERIMENT RESULTS

We evaluate our synchronization scheme in the 3DTI testbed. One microphone and four 3D cameras are used at the sender site. To make our results repeatable, we record at the sender gateway the size and the arrival time of each video and audio streams. We use *tc* software in Linux to simulate real Internet environment. The delay, jitter and loss data are collected from PlanetLab sites in the US.

**• Sender Gateway VIMS**

Fig. 5 shows the frame sizes of the four streams from the source cameras, and the corresponding VIMS at the sender gateway. We find that VIMS is related to frame size diversity within each macroframe, and can be larger than 100 msec. This is due to the fact that larger frames usually require more computation time at the cameras. The bandwidth capacity at the sender gateway and the gateway's thread scheduling may also affect VIMS. These results confirm that capturing-tier synchronization is necessary.

**• Renderer Video Replication Rate**

We evaluate our regression estimation and cooperative adaptation scheme under different bandwidth configurations. We define the *replication rate* as the percentage of video frames that need to be replicated due to the missing or incomplete frames at the receiver gateway. We vary the bandwidth (between the sender and the receiver gateway) at 10, 30, 50 Mbps in scenario 1 respectively with no intermediate site, and the results (Fig. 6) show that the replication rate is less than 3% for four video streams with 50Mbps bandwidth, and a reasonable 15% with 30Mbps bandwidth. 10Mbps bandwidth is not sufficient to support concurrent 4 streams, and will lead to a replication rate of more than 50%. In order to achieve similar replication rates in scenario 2, the bandwidth capacity at the sender gateway should increase in proportion to $\|\mathcal{E}_{rcv}\|$.

**• Receiver VIMS and Audio-Visual Skew**

In this section, we use 50Mbps upload/download bandwidth for all sites in our measurement. Fig. 7(a) presents the sizes of each video macroframe constituting 2 or 4 streams, and Fig. 7(b) shows the corresponding VIMS at the receiver gateway in scenario 1. We notice that VIMS is affected by the video macroframe size. The majority of VIMS is below 25 msec for 2 streams and 80 msec for 4 streams. We present the percentile of the *aggregate barrier latency* (the latency incurred on barriers at all intermediate sites and the receiver gateway) in Fig. 7(c) using 4 video streams. We compare two cases: (1) no intermediate sites (2) two intermediate sites between the sender and receiver gateways. The results show that only 50% of VIMS are below 150 msec for the second case, and the highest VIMS can be as large as 185 msec. The numbers will definitely degrade the 3DTI interactivity.

Unlike the video streams, the audio stream will not be affected by the barrier in the current 3DTI implementation. Fig. 8(a) presents the audio-visual arrival skew at the re-

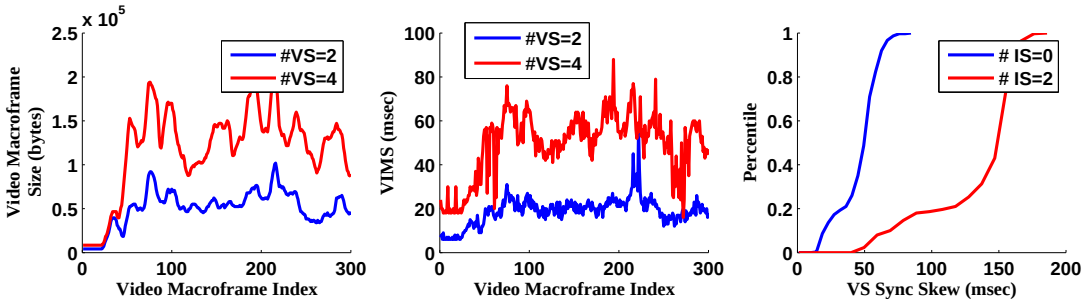


**Figure 7: (a)Left: video macroframe size; (b)Middle: VIMS at the receiver gateway; (c)Right: Percentile of aggregate barrier latency for 4 video streams for 0 and 2 intermediate sites (IS). VS: video stream.**

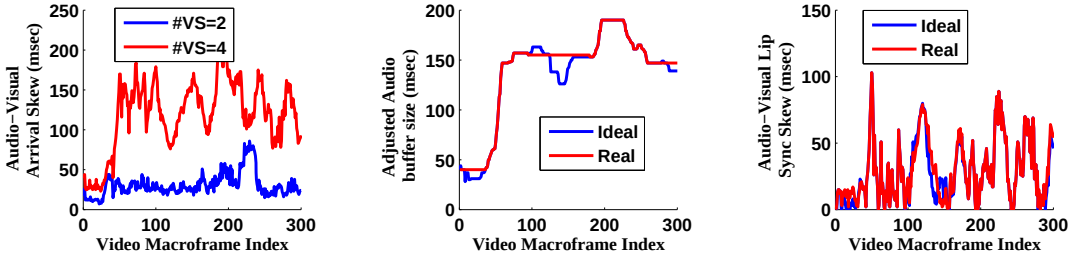


**Figure 8: (a)Left: audio-video frame arrival skew; (b)Middle: audio buffer adjustment for 4 VS. Ideal: all silences, real: talkspurts and silences; (c)Right: Resulting audio-visual sync skew for 4 VS. VS: video stream.**

ceiver gateway for scenario 1. It shows that there is an approximate average of 130-msec skew for 4 video streams and 30-msec for 2 streams. So without audio buffer adjustment at the renderer, the 130-msec number already exceeds the 80-msec audio-visual sync threshold guideline. Fig. 8(b) shows the adjusted audio buffer size in 4 video stream scenario. The ideal case is that the audio source remains silent, hence audio buffer can be adapted at any time. The real situation is that silences only happen at certain time, so audio buffer size will remain unchanged during talk-spurts (video macroframe 1-35, 95-179, 263-300 in Fig. 8(b)). Fig. 8(c) demonstrates the resulting audio-visual lip sync skew in both cases. More than 95% of the sync skews are within 80-msec threshold. Due to the space limit, we will not present scenario 2 and 3 which exhibit similar TSync effectiveness.

## 6. CONCLUSIONS

We present TSync, the multi-tier audio-visual synchronization scheme for the 3DTI system. Our scheme can actually be extended to any correlated multi-source distributed multimedia environment with diverse QoS. Experiment results show that with TSync the audio-visual sync skew at the renderer can satisfy the 80-msec threshold. We would like to acknowledge the support by NSF CNS 0720702 and 0834480, by UPCRC grant from Intel and Microsoft, and by Grainger Grant.

## 7. REFERENCES


[1] IEEE 1588 standard: Precise time synchronization as the basis for real time applications in automation, 2008.

[2] C. C. Chang and C. J. Lin. LIBSVM: A library for Support Vector Machines.

[3] L. Ehley, B. Furth, and M. Ilyas. Evaluation of multimedia synchronization techniques. In *Proceedings of IEEE International Conference on Multimedia Computing and Systems*, pages 110–119, May 1994.

[4] S. Goldenstein. Time warping of audio signals. In *Proceedings of IEEE Conference on Computer Graphics*, pages 52–57, 1999.

[5] N. Hu and P. Steenkiste. Evaluation and characterization of available bandwidth probing techniques. *IEEE JSAC Special Issue*, Aug. 2003.

[6] Z. X. Huang, B. Sat, and B. W. Wah. Automated learning of play-out scheduling algorithms for improving the perceptual conversational quality in multi-party VoIP. In *Proc. IEEE ICME*, pages 493–496, July 2008.

[7] ITU-G.114. One-way transmission time, 2003.

[8] E. Kohler, M. Handley, and S. Floyd. Datagram Congestion Control Protocol (DCCP), Mar. 2006.

[9] T. Little. A framework for synchronous delivery of time-dependent multimedia data. *Multimedia Systems*, 1(2):87–94, 1993.

[10] H. Liu and M. Zarki. An adaptive delay and synchronization control scheme for Wi-Fi based audio/video conferencing. *Springer Wireless Networks*, 12(4):511–522, July 2006.

[11] K. Nahrstedt and L. Qiao. Stability and adaptation control for lip synchronization skews. Tchnical report, University of Illinois, USA, 1997.

[12] A. Smola and B. Scholkopf. A tutorial on support vector regression. Technical report, Statistics and Computing, 2003.

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

[14] W. Wu, Z. Yang, and K. Nahrstedt. Implementing a distributed 3D tele-immersive system. In *Proceedings of IEEE International Symposium on Multimedia*, 2008.

[15] Z. Yang, W. Wu, K. Nahrstedt, G. Kurillo, and R. Bajcsy. Enabling multi-party 3d tele-immersive environments with viewcast. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 2010.

[16] Z. Yang, B. Yu, K. Nahrstedt, and R. Bajscy. A multi-stream adaptation framework for bandwidth management in 3D tele-immersion. In *NOSSDAV*, 2006.