Signal Processing: *Image Communication* 26 (2011) 370–377

Contents lists available at ScienceDirect

Signal Processing: *Image Communication*

journal homepage: www.elsevier.com/locate/image

From IPTV to synchronous shared experiences challenges in design: Distributed media synchronization

Ishan Vaishnavi [a,*], Pablo Cesar [a], Dick Bulterman [a], Oliver Friedrich [b], Simon Gunkel [b], David Geerts [c]

[a] *Science Park 123, Amsterdam 1098XG, Netherlands*
[b] *Kaiserin-Augusta-Allee 31, 10589 Berlin, Germany*
[c] *Parkstraat 45 Bus 3605, 3000 Leuven, Belgium*

ARTICLE INFO

Available online 29 March 2011

Keywords:
Synchronization
Local Lag
QoS
Mobility
Shared experiences

ABSTRACT

Recent developments on Social TV point to an evolution from traditional IPTV services towards more social experiences. Newer applications and services have appeared wherein groups of people in different locations can watch multimedia content while synchronously communicating with each other. We name such applications as *synchronous shared experiences*. Realization of these shared experiences requires that users feel that they are *coherently* communicating with each other. This paper identifies and analyzes challenges that need to be tackled to achieve coherence: quality of service, mobility and distributed media synchronization. Furthermore, universal session handling is required to setup these sessions. We then present our solution to one of these challenges: distributed media synchronization. Our design uses the local lag mechanism over a distributed control or master–slave signaling architecture. We validate our implementation via experiments performed with one client located in Amsterdam and the other in Seoul. The experiments demonstrate a bound in play-out skew of 500 ms across these locations. Our results from user tests, presented elsewhere, show that this value is well within tolerance limits.

1. Introduction

In the coming future media consumption services will integrate synchronous communication creating *synchronous shared experiences*. These experiences will exist independent of physical location and network of the users, across service provider and network technology domains. In this paper we explore technical challenges in the synchronization of such experiences. Compared to traditional media synchronization research, synchronization of shared experiences imposes additional requirements. For example, the users should be able to communicate unhindered with each other. This may require quality of service mechanisms over the Internet. This work adopts the term *coherence* when referring to synchronization in these shared experiences. Based on our work within the iNEM4u project[1] and previous work [1], we highlight four requirements for coherence: quality of service (QoS), universal session description, distributed media synchronization and user mobility. This article briefly highlights the challenges and examines potential solutions. We then present our solution for the distributed media synchronization problem in detail.

Fig. 1 shows how synchronous shared experiences are emerging. The figure presents three applications: an IPTV

* Corresponding author.
E-mail addresses: i.vaishnavi@cwi.nl (I. Vaishnavi), p.s.cesar@cwi.nl (P. Cesar), dick.bulterman@cwi.nl (D. Bulterman), O.Friedrich@t-systems.com (O. Friedrich), simongunkel@googlemail.com (S. Gunkel), david.geerts@soc.kuleuven.be (D. Geerts).

0923-5965/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.image.2011.01.006

[1] www.inem4u.eu

Fig. 1. Streaming media and synchronous communication yield synchronous shared experiences. (a) Imagenio IPTV, (b) Skype and (c) Yahoo! Zync.

service, Skype, and Yahoo! Zync. The first one follows a traditional broadcast model providing the user with a high-level of interactivity. The user can use new services such as Video on Demand and, at some extent, interact with the content. The second one is a popular instant messaging and conferencing system, in which the user can communicate synchronously with others; soon, to be integrated in your television set.[2] The third one, Yahoo! Zync, integrates instant messaging (communication) with video sharing. Two users at different locations can watch together a YouTube video, while chatting. The videos at both ends can be synchronized if the users so desire [2]. In this work we explore the software enablers needed for integrating not only chatting, but any kind of synchronous communication over the Internet, while watching IPTV, Internet TV or any other kind of streaming media.

This paper is organized as follows. The next section presents related work on IPTV and Social TV. Section 3 presents an overall architecture for synchronous shared experiences. Section 4 describes the challenges and elaborates on feasible solutions in each of these areas. Then, Section 5 reports on our work on cross-domain media synchronization. In particular, we show how existing synchronization solutions in distributed games, can be extended to achieve distributed media synchronization. We describe in detail our implementations and the validation results, including user studies in Section 6.

2. Related work

According to the European Commission's Networked Media cluster's vision, Future Media Internet [3] will carry high quality multimedia content and communications. Actual developments on operator's IPTV networks reflects this vision using a bottom-up approach, starting with the edge networks. Current research on IPTV concentrates mainly on three aspects:

- Creation of standardized service signaling, to drive the convergence of content and communications, such as Next Generation Networks [4].
- Technologies such as Widgets,[3] CE-HTML for the unified deployment of interactive applications over "three screens": PC, TV and mobile devices.
- Efficient distribution of content through Content Delivery Networks.

From a human factors perspective, the related research field of interactive digital television is being transformed into a study of human-centred television [5], in which viewers become active users with communication and (re-)distribution capabilities. As an active user, the viewer might want to communicate with others, for example: by sharing enriched fragments of multimedia content [5]. Boxee,[4] iPlayer, and Watchitoo[5] provide content sharing functionality, acknowledging that direct recommendations are more effective and personal than computed recommendations. Similarly, the first commercial social webTV applications Joost[6] and Lycos Cinema enabled users to text chat with each other while watching online TV or movies. More recently, the web based application Watchitoo[7] not only enables chatting, but also audio and video conferencing while watching the same content. All these services are however bound to one service provider/web-site. We do not impose any such restrictions on our algorithm.

This convergence of IPTV and Internet Video with social aspects has been predicted elsewhere [1]. The authors focused on the requirements of the future shared experiences: standards, mobility of content and people. While the authors there focused on social experiences in general, this article compliments their work by addressing synchronous shared experiences in particular.

3. Architecture

This section presents a high-level architecture for synchronous shared experiences. Fig 2 represents application and network level views for a typical shared experience. The figure provides a new perspective to the integration of the Home Network to the public Internet presented in [1]. In our vision different clients may have different service providers. The clients, however, should have the ability to be part of the same session, identify identical content even from different sources and render it in a synchronized manner. Furthermore, the clients should be able to differentiate between the QoS

[2] http://www.skype.com/allfeatures/tv/
[3] http://www.w3.org/TR/widgets/
[4] http://www.boxee.tv
[5] http://www.watchittoo.com
[6] www.joost.com
[7] http://www.watchittoo.com

requirements for the media and communication streams. Lastly, users should be able to move the entire shared experience from one client to another while preserving its state.

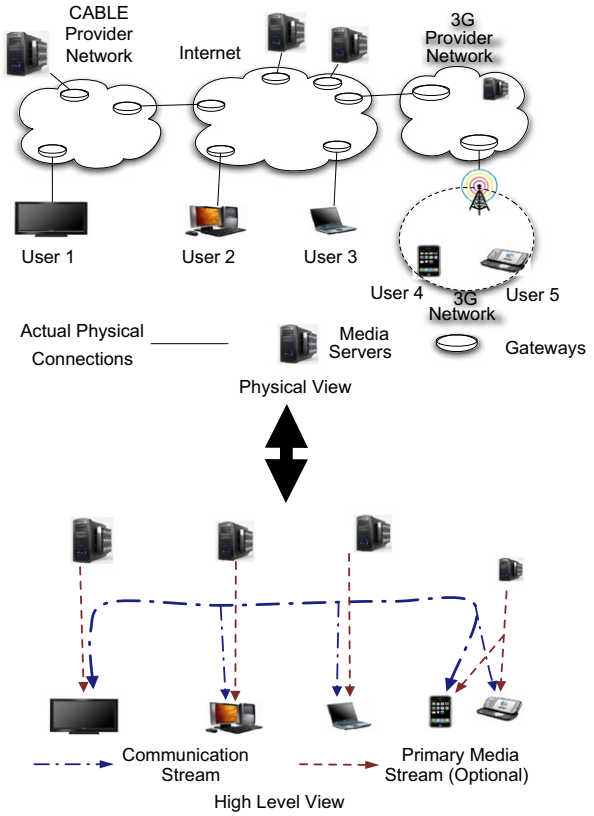

Fig. 2. Network and application level views.

The term *communication streams*, Fig 2, refers to the stream of synchronous communication between the distributed users. These can be chat messages and audio or video conferencing streams. The network path used by the communication stream is the *common communication channel*.

Primary Media Streams refer to the multimedia content that users are watching together. This can, for example, be an IPTV stream. This content needs to be rendered at the different locations in a synchronized manner, so that all users can watch coherently.

3.1. Client architecture

From the client's perspective, the architecture for enabling synchronous shared experiences is shown in Fig 3. It includes a number of servers (session, presence, and media servers) with two cross-domain clients. We have deliberately skipped a number of needed blocks (e.g., conditional access) in order to better scope our contribution. Each client includes five major components: the synch agent, the session manager, the parent renderer, the media content renderer, and the communication stream renderer.

The *parent renderer*, Fig 3, is the overall composite renderer that is responsible for the final layout of all the media elements and communication streams that compose the shared experience. The *communication stream renderer* is a plug-able component responsible for rendering the shared session stream, such as a Skype client for video/audio conferencing. The *media content renderer* is a plug-able component responsible for handling the primary media stream in accordance with the synchronization agent's control, such as an IPTV service client.

Our main concern in this paper is the *synchronization agent*, whose responsibility is providing coherence to the

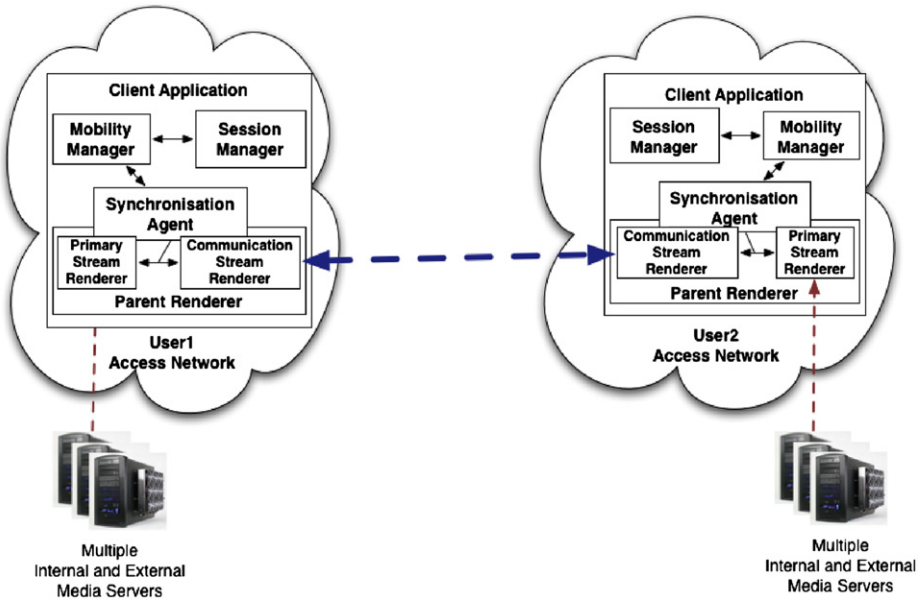

Fig. 3. A generic architecture.

shared experience. This may involve maintaining causality, providing time synchronization, quality of service semantics and the synchronized play-out of the media content across the network. The synchronization agent is supported by the underlying *session manager*. The session manager's role is to maintain media content and the shared session details, including user presence. It should also be accessible throughout a generic API providing interfaces for multiple protocols such as SIP, Webservices or JSON-RPC. It also acts as a mediator between users in different network domains (e.g. managed Telco IPTV and Over-The-Top Portals) and allows them to share content independent of their access network or used IPTV standards.

Lastly, there are challenges in the infrastructure design. In particular towards providing efficient QoS guarantees to both the primary stream and the communication stream. Each one of these challenges is presented in the following section.

4. Challenges in shared experiences

In this section we discuss the challenges for providing coherence in synchronous shared experiences, beginning with quality of service.

4.1. QoS

The QoS guarantees required by our architecture can be split into three categories. The QoS guarantees required by the (i) shared experience, (ii) media content and the (iii) control messages (including play-out position updates).

Communication streams require time bounded delivery, so that every word a user may say at one end can be heard immediately by the other users. However, how fast is immediate? Research in telecommunications puts this value at 150 ms [6]. The current standard for providing such QoS guarantees is the Differentiated Service networks architecture. Differentiated networks work by classifying packets into behavior classes, known as *per hop behaviors* (*phb*). The real time group (or *Expedited Forwarding Class*) must be used for communication streams. All packets in a class (phb) require a similar type of service.

The primary stream requires reliable as well as time bounded delivery. The bounds in time here are larger than those of a communication streams and depend on other factors, such as, buffering. Diffserv architectures provide a reliable delivery group (*Assured Forwarding Class*) with subclasses with various parameters, including low delay. This class is suitable for the primary stream.

To ensure end to end predictability the Diffserv-EF and the AF-low delay class are recommended short queue lengths. This is done under the assumption of correct provisioning. There are two problems with this assumption: (i) it is difficult to correctly provision networks and (ii) users, especially using in multimedia, do not always transmit at a constant bandwidth. In our work [7] we showed that this (i) leads to an under-utilization of available bandwidth and (ii) requires that the networks are over-provisioned. We proposed Estimated Service networks which try to ensure time bounded delivery by making the network (IP layer) utilize the time bounds on the data it delivers. In other words we are referring in essence to (adapted) deadline based scheduling mechanism. Real time community has been using this for decades starting primarily by the work in [8]. Our results, presented in [7], show much better bandwidth utilization.

Finally, control flow semantics are required to assure prioritized delivery of control packets such as the play-out position updates. By giving control flow priority over all other classes in the Diffserv architecture these semantics can be achieved. The real-time class expedited forwarding comes second and is used for the communication stream. Lastly, the primary stream can be sent via the assured forwarding low-delay class.

4.2. Universal sessions

Cross-domain session management is one of the main challenges for creating shared experiences between users in different network domains. Sessions for shared experiences extend the basic multimedia session concepts towards a multi-user, multi-content approach. Our solution is called iSession [9]. iSession provides a logical representation of interactive multimedia sessions that can span across multiple domains. It contains all the information that is required by a client to connect to an existing session. This includes information on the services and the content that are consumed and on the users who are participating in the session. It can also contain metadata information that may be required for synchronization of media streams across domains and layout information to create a common experience across devices. The actual delivery of the content is achieved within each domain by clients using domain-specific technology.

An iSession is described by a Session Description Protocol document (SDPD). This document contains information on the content associated with the session, the participants, and the layout and timing information needed to render or replay the session. The SDPD is used by clients to establish connections to content sources. However, it does not contain media specific information like media format or media delivery type. This information is directly exchanged between the client and the content sources providing the session content.

Either users or service providers can create an iSession . In the first case, a service provider gives users the possibility to create a session document and to share it with others. In the second case, the session is created by the service provider and users are able to join in. Users can discover a session through invitation from other users, by subscribing to sessions of specific types or by selecting a session from an EPG containing session information.

4.3. User mobility

User mobility is an important requirement for Social IPTV services [1]. As illustrated in Fig 2, users are surrounded by a multitude of devices all with varying

rendering capabilities. Users are bound to move their focus between these devices for various reasons identified in [10]. In such situations session mobility is required to move relevant sessions from one domain/device to another. The cross-domain sessions described in Section 4.2 can, however, be composed of multiple individual media sessions. Current methods [11] of achieving user mobility in such cases, is as a composition of multiple media session mobility requests. This process is repetitive and therefore inefficient in the control plane and may not provide the best available user experience [12].

To address this inefficiency our solution notices that each user is associated with one universal session composed of one presentation, which in turn maybe composed of multiple media sessions. Thus for efficient user mobility a presentation-layer mobility mechanism is required. The presentation-layer mobility mechanism describes the entire shared experience in structural description languages, such as SMIL.[8] At the time of executing a mobility request, the current state of the experience can be saved in a structured file, using solutions, such as SMIL-state [13]. This file can be transferred using the any appropriate network-dependent means to the target location. The shared experience can be re-started at the new location and the saved state re-applied, yielding coherent user mobility. Since the file is only moved once there is no repetitive signaling in the control plane. Re-starting the experience at the target location ensures the best possible user experience. Further details are presented in [12].

4.4. Distribute media synchronization

The existence of a common synchronized content is another requirement of coherence. The main challenge in this area is to achieve cross-domain synchronization in play-out of primary streams across multiple destinations streaming from respective servers. We assume that the primary streams rendered at different clients have a common or related play-out times. For VoD content this is just the current play time. For broadcast, the ETSI-DVB standard [14] provides such a time line. For more complex presentation structures, such as, those which are written in descriptive languages, such as, SMIL or NCL solutions such as SMIL state [13] can be used. Section 5 presents our implementation of the distributed synchronization algorithm which addresses these challenges. The implementation builds on the universal cross-domain sessions, used for identifying users and content across domains.

5. Distributed media synchronization

This section describes how synchronization algorithms, typically used in distributed gaming, can be adapted for generic synchronous shared experiences. In particular, we look at the scenario, in Fig 4, in which users, watching the same (or related) video at two separate locations, can communicate with each other synchronously. The videos need not originate at the same source. A number of simplifying assumptions are made as enlisted below:

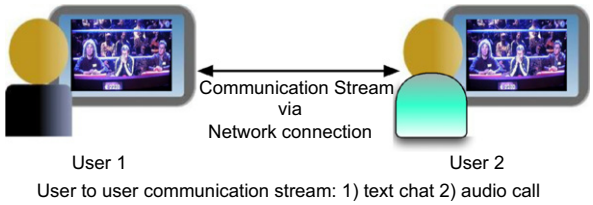

Fig. 4. Scenario and user test setup.

- Play-out position updates can be sent over the communication stream.
- The rendering devices are capable of running the synchronisation algorithm and are powerful enough to process events within time bounds.
- Cross-domain sessions can be setup. We utilized iSessions [9].
- The clients are time synchronized within certain bounds.

Given these assumptions, the "Local Lag with Time Warp" [15] algorithm can be adapted to achieve *event synchronization*. Event synchronisation is the coherent execution of a user's actions at all the clients, so that a consistent version of the experience is seen by all the users. This is important in a distributed gaming infrastructure. In our use case if the main media stream is paused at one end, then, the pause should be executed at other clients within a bounded tolerance limit. The algorithm consists of two parts: (i) local lag to compensate for short term inconsistencies and (ii) time warp to undo inconsistencies that may still occur due to various network delay and jitter factors.

The concept of local lag is illustrated in Fig 5. Instead of executing the user action as soon as possible, a time in the future is chosen. This delay in event execution, known as local lag, ensures that the event has time to travel to all destinations before being executed. The errors in execution are now dependent only on the skew in time synchronization. The value of the local lag needs to be small enough so that the user does not notice the delayed execution, but large enough that the event has enough time to travel to other clients. A minimum value of 150 ms is recommended [15] for applications that span the globe. These events need to reach their destinations before this specified delay. Since these events are sporadic in nature and require very little bandwidth, they can be sent as network control messages, which have strict priority.

The local lag technique however only attempts at resolving small term inconsistencies, it does not provide guarantees. Time warp is the process of rolling back changes to the last known consistent state in case, inconsistencies are detected. Time warping is therefore application-dependent and thus not directly addressed

[8] http://www.w3.org/AudioVideo/

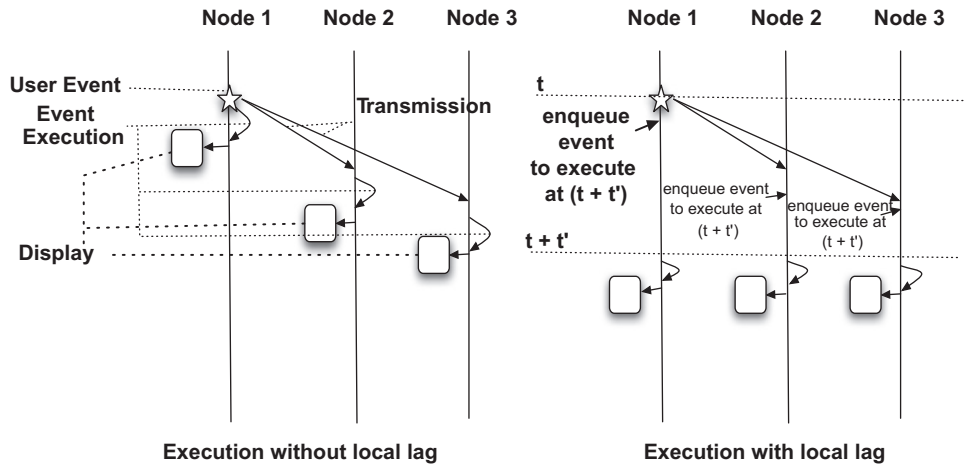

Fig. 5. Local inconsistencies can be avoided by using local lag.

here. For our specific scenario of distributed video watching, time warp resolves to jumping back or forward in the play-out time. This however implies that the renderer needs to cache frames which have already been played out for a short period of time to facilitate the jumping back operation.

5.1. Signaling architectures

Previous research in distributed synchronization mechanisms recommends three different signaling architectures: (i) master–slave [16] (ii) sync-maestro [17] and (iii) distributed control [18]. In the chosen scenario there are significant argumentative reasons not to choose sync-maestro. Sync-maestro requires the maintenance of a central sync-server which receives play-out position updates from all the nodes, re-calculates their individual adjustments and signals them to re-adjust their play-out position. This technique requires the maintenance of a dedicated sync server, which is not scalable [19]. Furthermore it assumes that all clients can contact the server directly, which may not be the case in cross-domain applications. This architecture is typically used in distributed games to maintain a world wide view of the game, as a single server simplifies problems relating to causality and replication consistency.

The other two approaches are quite similar and more feasible. In the master–slave approach one of the nodes is considered a master and multicasts play-out update messages to all other nodes in the system, whereas in a distributed control system all nodes can send play-out position updates. Each receiving node chooses to follow one of these update packets as the reference timeline. As long as all the nodes choose the same reference timeline the system will remain synchronized. The choice between these two architectures is largely application dependent. For example: in an e-classroom environment the teacher should be the master node which directs the student nodes as to what part of the presentation to play-out. In a friends watching football scenario the distributed control scheme is more appropriate, since there is no need for master re-election if the current master drops out. Lastly a fourth network architecture, which is a hybrid between the sync-maestro and the distributed control approach can also be used. In this approach a sync-maestro in the local domain collects the status updates from all the end-nodes in that domain and controls their play-out. In a cross-domain session, each of the domain sync-maestros further run a distributed control signaling based algorithm amongst themselves. This approach is suitable if a very large number of nodes belong to the same session, such as massively multiplayer online games.

In our prototypes we implemented the local lag and time warp algorithm [15] over both the master–slave and the distributed control signaling architectures to achieve event synchronization.

5.2. Execution

Given event synchronization, media synchronization can be achieved by replacing the event message by play-out position updates. These play-out position updates are generated periodically. The value of this periodicity depends on the particular application. Since these events are not user generated, the local lag is determined by the period of updates and the particular application semantics, rather than user tolerance to local event execution.

When two or more users join a session to watch media content together, a distributed session needs to be set up. A number of steps are required to achieve media synchronization during the session setup. In particular, a multicast channel needs to be setup and the users need to negotiate on a time synchronization sources. The multicast channel can be set up re-utilising the communication stream infrastructure. The control messages of the communication stream can be extended to create a multicast channel. In our implementation XMPP was used to signal user presence and XMPP IQ extended to include position updates.

During execution, the synchronization agent at each node acts as a smart event queuing and handling thread. All events are placed in a queue, insertion sorted over

execution time. When the execution time of a particular event is reached, the associated event is executed. For periodic position updates in the distributed signaling architecture, all nodes send position updates but only one, the reference node's update needs to be executed. In our implementation the reference node is the node that has the smallest media buffer size.

6. Implementation and validation

Two reference implementations were developed. The *first one* was implemented in Java, using Java Media Framework as the media renderer component. All nodes played the identical video file located at different sources. This implementation was used for testing between clients in Amsterdam and Seoul for a 30 min video clip. Every 30 s each client logged on the current play position, CPT_i, along with the NTP time, NTP_i, at that moment. Fig. 6 plots the results of $CPT_{Amsterdam} - CPT_{Seoul} + \{NTP_{seoul} - NTP_{Amsterdam}\}$ for two runs; 1st run: with the master–slave signaling architecture and 2nd run: with distributed control signaling architecture. A total of five repetitions for each signaling architecture were carried out with indistinguishable results. The graph suggests a constant error in play-out of around $300\,\text{ms} \pm 100\,\text{ms}$. A further uncertainty in these measurements due to error in NTP measurements is $< \pm 100$ ms resulting in an overall worst case error of $300\,\text{ms} \pm 200\,\text{ms}$. This constant skew value is observed due to constant differences between the renderers in the time taken to execute the seek operation, at every position update.[9] The boundedness of this seek operation was assumed at the start of Section 5. Nonetheless this worst case error of 500 ms is acceptable as will be shown by user tests presented in Section 7.

A *second implementation* for three different clients in three different domains was developed. The three clients were a PC, an unmanaged TV and a mobile phone. The PC version was implemented using .NET and with VLCplayer used as the media renderer. A Philips NET TV was used to implement an unmanaged TV case and was coded in javascript and the in built renderer used for media rendering. The mobile phone was implemented using HP's reference implementation of the JSR 309 standard http://jcp.org/. This work also lead to the enhancement of the JSR 309 standard. Fig. 7 shows the screenshots of this implementation.

7. User studies

A Social TV use case with either text or audio communication streams was tested with real users on two occasions. The objective of the test was to identify the user tolerance towards skew in video play-out. In a first series of tests in March 2010, the implementation shown in Fig 7 was used. Three users, friends or family of each other, were asked to watch the same video on three different locations and on three different devices: a PC, a television set and a mobile phone. At the same time, they could voice chat with each other while watching the video. At specific intervals, the synchronization of the videos was automatically changed so that one of the participants was not in sync with the others. In total, seven groups of three users participated in this test. The results showed that there exist too many independent parameters that influence the users experience, such as the user's environment and communication channel. Thus no statistically significant results on user tolerance to synchronization skew could be derived from these experiments.

A more controlled experimental setup was created in August 2010, enabling two users to watch a video together at two locations, communicating with each other using text chat or voice chat as shown in Fig 4. The skew in video play-out was changed, with five different skew times: 0, 0.5, 1, 2 and 4 s with 36 participants. The results showed that voice chat and active text chat users noticed synchronization differences only beyond 1 s. Further details of this user study and the results thereof can be found in [20].

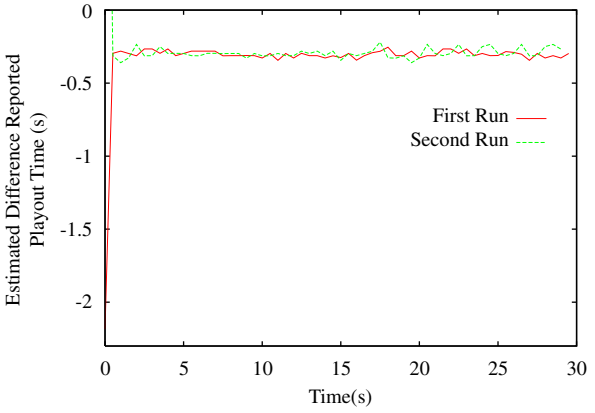

Fig. 6. Validation results: (1st run) master–slave signaling, (2 run) distributed control.

8. Conclusion and future work

In this paper we identify the architectural challenges involved in the transition from IPTV services to synchronous shared experiences. In particular we looked at: quality of service, universal session handling, user mobility and synchronization. We elaborated on our solution for the distributed media synchronization and presented the validation results thereof. Our design uses the local lag mechanism over a distributed control or master–slave signaling architecture. Our implementation achieved synchronization precision of around $300\,\text{ms} \pm 200\,\text{ms}$ between Amsterdam and Seoul. User studies conducted as a part of this work demonstrate this is sufficient for Social TV applications. In the future we plan to perform more user tests to identify the exact user tolerance towards differences in synchronization level and how

[9] This fact was verified when the computers were again co-located in Amsterdam.

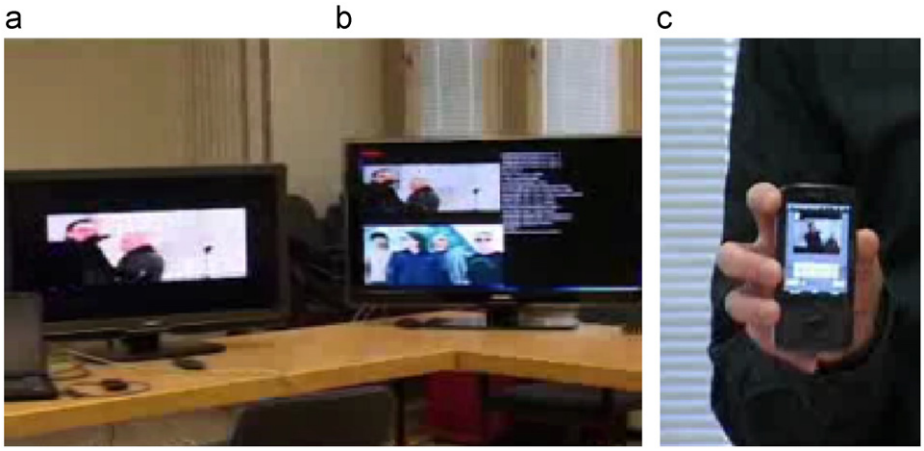

Fig. 7. Example implementation in the iNEM4u project. (a) SIP Clint (Laptop), (b) TV Client and (c) 3G Mobile Clint.

various parameters, such as, the communication channel and media content may influence them.

References

[1] M. Montpetit, N. Klym, T. Mirlacher, The Future of IPTV: Adding Social Networking and Mobility, ConTEL, 2009.
[2] D. Shamma, M. Bastea-Forte, N. Joubert, Y. Liu, Enhancing online personal connections through the synchronized sharing of video, 2008.
[3] T. Zahariadis, P. Daras, I. Laso-Ballesteros, Towards future 3d media internet, in: NEM Summit 2008, St. Malo, 13–15 October 2008.
[4] T. Kovacikova, P. Segec, NGN standards activities in ETSI, in: Sixth International Conference on Networking, 2007. ICN'07, 2007, p. 76.
[5] P. Cesar, D.C.A. Bulterman, L.F.G. Soares, Introduction to special issue: Human-centered television—directions in interactive digital television research, 2008.
[6] R. ITU-T, I. Recommend, G. 114, One-way transmission time, 2003.
[7] I. Vaishnavi, D.C. Bulterman, Estimate and serve: scheduling soft real-time packets for delay sensitive media applications on the internet, in: ACM NOSSDAV, ACM, New York, NY, USA, 2009.
[8] C. Liu, J. Layland, Scheduling algorithms for multiprogramming in a hard-real-time environment, Journal of the ACM (JACM) 20 (1) (1973) 61.
[9] D. Goergen, J. Zoric, J. O'Connell, O. Friedrich, B. Zachey, A session model for cross-domain interactive multi-user IPTV, in: IEEE CCNC, 2010.
[10] S. Mate, U. Chandra, I.D.D. Curcio, Movable-multimedia: session mobility in ubiquitous computing ecosystem, in: Proceedings of MUM, 2006, p. 8.
[11] M.-X. Chen, C.-J. Peng, R.-H. Hwang, Ssip: Split a sip session over multiple devices, Computer Standards & Interfaces 29 (5) (2007) 531–545.
[12] I. Vaishnavi, P. César, A.J. Jansen, B. Gao, D.C.A. Bulterman, A presentation layer mechanism for multimedia playback mobility in service oriented architectures, in: Proceedings of ACM MUM, 2008.
[13] J. Jansen, D.C. Bulterman, Enabling adaptive time-based web applications with smil state, in: Proceeding of DocEng '08, 2008.
[14] T. ETSI, 102 823 v1. 1.1, Digital Video Broadcasting (DVB); Specs for the carriage of synchronized auxiliary data in DVB streams, 2005.
[15] M. Mauve, J. Vogel, V. Hilt, W. Effelsberg, Local-lag and timewarp: Providing consistency for replicated continuous applications, IEEE Transactions on Multimedia 6 (1) (2004) 47.
[16] Y. Ishibashi, A. Tsuji, S. Tasaka, A group synchronization mechanism for stored media in multicast communications, in: Proceedings of the INFOCOM'97, 1997.
[17] Y. Ishibashi, S. Tasaka, A media synchronization mechanism for live media and its measured performance, IEICE Transactions on Communications E81-B (10) (1998) 1840–1849.
[18] Y. Ishibashi, S. Tasaka, Y. Tachibana, Adaptive causality and media synchronization control for networked multimedia applications, IEEE International Conference on Communications, 2001, ICC 2001, vol. 3, 2001.
[19] Y. Ishibashi, S. Tasaka, Causality and media synchronization control for networked multimedia games: centralized versus distributed, in: Proceedings of Netgames, 2003.
[20] D. Geerts, I. Vaishnavi, R. Merkuria, P. Cesar, O. Deventer, Are we in sync? Synchronization requirements for watching video online together, in: ACM CHI, 2011.