Immersive second-screen experiences using hybrid media synchronization

Ray van Brandenburg[1], Arjen Veenhuizen[2]

[1, 2]TNO, Delft, the Netherlands;

E-mail: [1]ray.vanbrandenburg@tno.nl, [2]arjen.veenhuizen@tno.nl

Abstract: **Most second-screen services have so far been relying on watermarking and fingerprinting for providing synchronisation with the main TV screen. In the EU FP7 HBB-Next project, a new timestamp-based inter-device synchronization system has been developed, allowing for frame-accurate synchronisation between DVB streams and over-the-top HTTP adaptive streaming content. As a showcase for this technology, this paper describes a system that allows users to navigate through ultra-high resolution content on their tablets, while watching the main director-controlled stream on their TV.**

Keywords: Inter-device synchronization, DVB, tiled streaming, HTTP Adaptive Streaming, clock synchronization

1 INTRODUCTION

In the past few years a number of technologies have become available that allow extending the linear broadcast TV experience with additional interactive features delivered over the Internet. A prime example of such a technology is HbbTV (Hybrid Broadcast Broadband TV), a standardized [1] technology that bridges the gap between broadcast DVB streams and the IP Internet world. By including special packets containing URLs in the DVB streams, it is possible to trigger a HbbTV-enabled TV to start retrieving and showing HTML applications to accompany the TV content. Currently, the somewhat limited feature set of the HbbTV standard has limited most such applications to interactive portals featuring news, catch-up TV and access to video-on-demand content.

Simultaneously, a completely orthogonal set of technologies, most notably watermarking and fingerprinting, have allowed the rise of companion applications on second screen devices such as smartphones and tablets. By either detecting audio watermarks in the TV content stream or by exchanging audio fingerprints with a centralized server, any device equipped with a microphone can roughly synchronize to the TV content. The combination of synchronization with the more advanced feature set of second screen devices have allowed for more advanced second screen applications compared to those offered by HbbTV, such as applications where a user can play along with a TV show or receive streams containing statistics and additional information during sports matches. However, the limited synchronization accuracy provided by both fingerprinting and watermarking, combined with the proneness to error because of the reliance on audio feeds, have restricted such second screen applications from providing truly frame accurately synchronized experiences such as providing auxiliary audio or video feeds on second screen devices.

In this paper we will present a new set of technologies that allow for immersive second screen experiences which are frame-accurately synchronized with an HbbTV application running on the main TV screen. As a showcase of this technology, we have implemented a second screen application that allows users to freely navigate (e.g. pan/tilt/zoom) around an ultra-high resolution video panorama which is synchronized with a main DVB stream shown on the TV. Using this technology, users have the freedom to navigate spatially through a football match, without missing anything of the action on the main TV screen, or can zoom in to specific musicians while watching a concert or festival registration. At the same time, he can still watch the director-controlled feed shown on the main TV screen. As such, the passive lean-back experience of watching TV is combined with the more active and lean-forward experience of interacting with TV content.

Apart from introducing a set of synchronization technologies that allow frame-accurate synchronization both between different devices as well as between different video delivery technologies (DVB versus IP), this paper will further discuss new methods for clock synchronization between devices and auto discovery mechanisms which can be used to provide seamless and user-friendly device discovery between HbbTV applications and second screen devices. These components have been designed making sure that all elements need to be able to be implemented without any significant changes to the broadcast chain. Furthermore, they should be implementable on all major mobile platforms, not requiring any proprietary or hard-to-implement features.

2 USE CASE & REQUIREMENTS

The concept of Tiled Streaming [5] allows for scalable delivery of ultra-high resolution video (4K and higher) to mobile devices. By allowing users to spatially navigate in the video and only sending that part of the video a user is interested in, the total bandwidth necessary does not exceed that of a regular HD stream. In the EU FP7

Corresponding author: Ray van Brandenburg, TNO, Brassersplein 2, Delft, NL, +31 6 51219254, ray.vanbrandenburg@tno.nl

ISBN/EAN – 978-90-5986-439-9

FascinatE project we've developed a mobile application for the Apple iOS platform (both iPad and iPhone) that implements Tiled Streaming.

A fact that became immediately obvious while developing Tiled Streaming and seeing users interact with it, is that interacting with video is an inherently active and lean-forward experience. While such interaction can definitely add a level of immersion to experiencing a video, it is hard to imagine it becoming the primary method with which users experience a football match or a musical performance. TV has traditionally been a passive lean-back experience and this is unlikely to change with the advent of Tiled Streaming and similar interactive services. Such applications however, might be interesting value-added services when added as a second-screen application to regular TV broadcasts. It is this application of Tiled Streaming that is the subject of this paper. In particular, the combination of Tiled Streaming on a tablet or smartphone synchronized with a DVB stream on the main TV screen.

The rest of this paper will describe how the existing Tiled Streaming application was extended with functionality that allows it to be synchronized with an external DVB stream. During this discussion, a number of new synchronization techniques will be presented.

2.1 Requirements

2.1.1 Inter-device and Clock Synchronization

In existing second-screen applications that implement synchronisation, such as used for interactive quiz-shows and sports statistics, the level of synchronization accuracy needed is typically in the order of half a second to a couple of seconds. When synchronizing two audio/video streams on multiple devices however, the level of accuracy required is an order of magnitude higher. Especially in the case where audio streams are played from multiple speakers, studies have shown the need for accuracy to be in the order of 50ms or less. In general we can therefore say that for this type of *inter-device* synchronization, frame-accurateness is a requirement (i.e. 40ms for 25fps content).

A concept closely tied to inter-device synchronization is that of clock synchronization. In order for two devices to synchronize playback of content streams to a frame-accurate level, it is first necessary for these two devices to share a common wallclock. In synchronization mechanisms that assume such a common clock, the level of synchronization accuracy is therefore determined by the accuracy of the clock synchronization method. For the purposes of the use case described in this document, where two 25fps video stream will be synchronized across multiple devices, it is therefore vital that the clock synchronization is as accurate as possible (error at least <40ms).

2.1.2 Hybrid timelines

Another important element when considering the presented use case is the issue of hybrid media timelines. Whenever it is necessary to synchronize two distinct streams, one of the requirements is to have a common content timeline between the two streams.

In most video delivery mechanisms, the content timeline is tied together with the used video container. However, where in DVB environments the MPEG Transport Stream (MPEG-TS) is the audio-visual delivery method, the same is not true for the over-the-top IP world. In recent years the method of choice for internet streaming to mobile devices has become HTTP Adaptive Streaming (HAS) in one of its many variants (e.g. Apple HLS, Microsoft SS or MPEG DASH). The problem here is that MPEG-TS and HAS use two very different methods of providing a content timeline. While MPEG-TS uses a concept of Presentation Timestamps and Decoding Timestamps (PTS/DTS), in HAS the timeline is typically communicated via the so-called manifest file. (One could argue that both Apple HLS and a specific version of MPEG DASH actually use MPEG-TS as the underlying video container, however, in these cases the PTS/DTS values are only used within segments and not as the primary synchronization method between segments. In addition, typical HAS client side implementations don't make these values available to the application).

What therefore is needed is a common content timeline that unifies the DVB PTS/DTS system and the HAS manifest-based timelines.

2.1.3 Device discovery

Before synchronization between the TV and the second screen device can occur, it is necessary to first create a communication link between the two devices. Device discovery is a topic which has received significant attention from both the research community as well as the industry. For the purposes of the presented use case, the most important requirements are that the solution is user friendly and thus does not require any active input from the user (e.g. pairing, typing passcodes, scanning QR codes, tc.) and that the solution can be implemented relatively easy on multiple devices which are typically limited by their respective platforms (e.g. TV/STBs with simple browsers, Android, iOS, etc.).

2.1.4 Application bootstrapping & Communication

A final element to include in the to-be-developed system is application bootstrapping: how to make sure that the proper application is launched on the TV and how to notify the second screen app of the appropriate content? For the former problem the HbbTV specification already utilizes the so-called Application Information Table (AIT) [11]. With this table, which can be included in the DVB stream, a broadcaster is able to signal a URL that points to an HbbTV application that should be launched upon switching to a particular TV channel. For the latter problem, signalling a content identifier to the second screen, no standardized solution exists to date.

3 RELATED WORK

A technique very similar to Tiled Streaming was first proposed by Mavlankar et. al. [3]. In their work, they proposed an interactive video streaming system that

ISBN/EAN – 978-90-5986-439-9

allows users to change the region-of-interest while watching online lectures. Related techniques were explored by Khiem et. al in [2]. Furthermore, Alface et al. [4] proposed a server-centric method for tiled streaming that allows browser-based client to interactively navigate around a video panorama without any processing on the client-side. The main differences between these systems and the Tiled Streaming system discussed in [5] is the focus on the client as the node responsible for selecting the appropriate tiles instead of an intelligent network node. Using a so-called spatial manifest in combination with standard HLS streams, all intelligence is placed on the client, allowing the use of standard off-the-shelf HTTP servers and making the system highly scalable. Another difference is the use of an overlapping tiling scheme, reducing the number of necessary decoders on the client side, allowing Tiled Streaming to be performed on performance-restricted mobile clients.

One of the first projects to focus on synchronisation between the broadcast world and the over-top internet world was the SAVANT project [19], developing a system that allows a sign language video stream to be shown as an overlay on a broadcast stream or on a second-screen device. More recent research has included work on re-using the PCR/PTS timestamp system present in DVB and adding this information to IP streams as well, either by the broadcaster [20] or by a third party [21]. A downside of such systems is that the PCR and PTS values are often changed within the delivery network, as a result of re-multiplexing operations. Finally, the combination of DVB and RTP has also been the topic of significant study, such as that by Leroux et. al. [22] who proposed a system for synchronizing DVB-H with RTP streams on mobile devices.

In recent years, with the advent of tablets and smartphones, the topic of inter-device synchronization has received renewed interest from both the industry as well as the research community. However, this attention has been mostly directed at fingerprinting [8] and watermarking-based [7] techniques. An exception is [6], where Howson et. al. propose using synchronised auxiliary data packets to carry timeline data in DVB streams in order to allow synchronization with RTP streams.

4 COMPONENT DESIGN

This section discusses the main design decisions that were made while developing the system. Before proceeding with the details behind the various components, Figure 1 shows an architectural overview of the proposed demonstrator in an HbbTV context.

On the TV/STB side, there is an HbbTV application which features modules for clock synchronization, inter-device synchronization and device discovery. On the second-screen side, there exists the Tiled Streaming application which has been extended with modules that form the counterparts of those in the TV/STB. Where the main input for the TV/STB consist of the DVB stream, the second-screen application retrieves Tiled Streaming segments from a standard web server via HTTP.

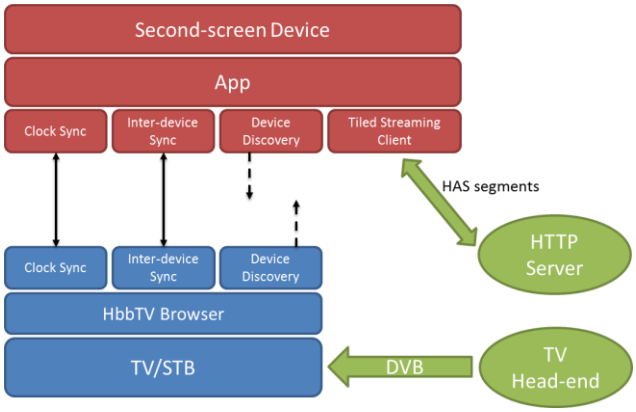

Figure 1: Architectural overview

4.1 Content Timelines

In order to synchronize the DVB broadcast stream with the HAS-based streams used in Tiled Streaming, a common timeline between the two video delivery mechanisms is necessary.

The PTS/DTS values which form the basis of the inter-media synchronization mechanism present in an MPEG-TS are not suitable for any type of synchronization with streams not present in the same MPEG-TS multiplex. The main reason for this is that PTS/DTS values are not a product of the content itself but of the transport mechanism. In a typical DVB stream, the PTS/DTS values form a continuous stream, irrespective of program changes or advertisements. While this is a positive thing from a broadcaster-perspective, in the sense that it is impossible to use PTS/DTS values to automatically remove ads from a recording, it also makes it difficult to distinguish different programs. Furthermore, since PTS/DTS values are not inserted by the broadcaster but by the network operator, these values may also change depending on the network. This means that a particular program being broadcasted on Network A will typically have different PTS/DTS values than the exact same program at the same time on Network B. The PTS/DTS values may even change within the same network, for example after a re-multiplexing operation. Clearly, this disqualifies PTS/DTS values as a candidate for providing content timelines for inter-device synchronization purposes in a real-world scenario.

A more suitable form of timeline might be an absolute content-time, starting at 00:00:00 at the start of the program and counting the number of minutes and seconds since. Such a timeline is particularly well-suited to be synchronized with an HAS stream, since the manifest files used in the various HAS variants all include the length of the individual segments, from which the absolute time since the start of the content can be deduced. The question is how this content timeline can be signalled in the DVB stream.

As mentioned by Howson et. al. earlier [6], DVB has standardized so-called DVB Synchronized Auxiliary Data (SAD) packets in 2005 [12]. Using these packets, it is possible to include additional packets in an MPEG-TS

ISBN/EAN – 978-90-5986-439-9

that contain arbitrary data which is synchronized to the media streams present in the MPEG-TS. In short, each of these SAD packets contains a PTS/DTS value, which allows it be accurately synchronized with audio/video packets. By including the absolute content time in each SAD packet up to that point, one can create a mapping from PTS/DTS values to absolute content time. Furthermore, since these packets can be packetized as any other Packetized Elementary Stream (PES), they are transparent from a network perspective. Should any intermediate (re-)multiplexer change the PTS/DTS values in the audio/video PESs, the packets in the SAD stream will be changed accordingly. This property allows SAD packets to be added to a stream by a broadcaster and arrive unharmed at the client side.

Using a combination of SAD packets in the DVB stream and either an implicit or explicit absolute content timeline in the HAS manifest files, a common content is achieved between these two heterogeneous media delivery methods.

Now that the content timeline is available, the next step is using it to synchronize the two streams on different devices.

4.2 Inter-device synchronization

As discussed in section 2.1.1, there are multiple approaches for achieving inter-device synchronization. The mechanism developed for the system being discussed here, in which frame-accurate synchronization is a requirement, is to exchange media timestamps. Specifically, repeatedly communicating a single tuple between the to-be-synchronized devices: a content time and a wallclock time.

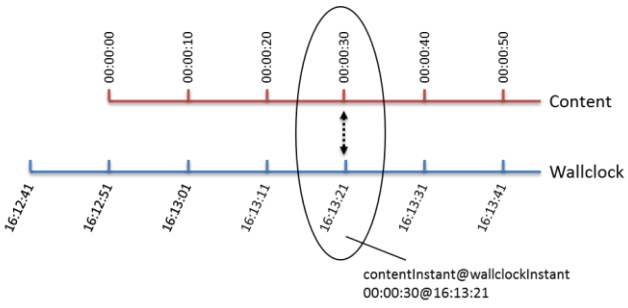

Figure 2: Synchronization Tuple

Figure 2 shows the two relevant timelines schematically: a content timeline starting at 00:00:00 and increasing monotonically with the media stream, and a wallclock timeline representing the moment in time at which the media was played back. The circled area in the figure displays a single synchronization tuple which can be signalled to other players to achieve inter-device synchronization. In this particular case, it shows that the media samples with content time 00:00:30 where played back at 16:13:21. Based on this single tuple, the receiver has enough information to calculate the appropriate playback instant for any given media sample in the content stream. Assuming successful clock synchronization, it can simply calculate that the media samples at 00:00:50 should be played back at 16:13:41. The advantage of sending timestamps tuples instead of control commands over the synchronization interface is that it the synchronization accuracy is independent of the network delay. Even if the tuple arrives at the receiver a couple of seconds after it was sent, it still provides all the information the receiver needs to calculate the appropriate playback time for any given media sample.

In order to prevent desynchronization resulting from clock drift and packet-loss, the synchronization tuple can be sent repeatedly, e.g. once every 5 seconds.

While in the current implementation the synchronization tuples were communicated over a WebSocket [14] between the TV and the second-screen device, the same mechanism can be applied over any underlying transport mechanism. In large-scale synchronization architectures, one could even broadcast the synchronization tuples to all receivers.

Depending on device capabilities and the availability of the media streams, either the TV/STB or the second-screen device can delay its media playback depending on the tuples it receives. For example, a given STB might only be able to buffer incoming DVB streams for a short period of time due to memory limitations. In this case, it might be easier for the second-screen device to adjust its playback to match the DVB stream. Alternatively, in live situations the Tiled Streaming segments might only be available a certain period of time after the DVB broadcast stream, which means that in these cases the DVB stream playback might have to be delayed.

4.2.1 Clock Synchronization

The inter-device synchronization method discussed in the previous section assumes clock synchronization between the different devices. However, such synchronization is not a given.

Traditionally, NTP [13] has been the main method for clock synchronization employed on devices connected to the public internet. One of the advantages of using NTP is that it has been deployed widely, with client-side libraries available for most platforms (although most implementations focus on SNTP, a simpler and less accurate version of NTP [REF]) and a large set of public NTP servers to synchronize to is available. The downside of using NTP is that it is generally not accurate enough for all use cases [How accurate is it typically?]. For this reason, the Precision Time Protocol (PTP) was standardized [15]. However, PTP implementations are not yet commonly available and require, in order to work with the highest precision, PTP-enabled layer 2 elements in the network.

There are a number of factors that contribute to the inaccuracy of NTP, most notably the fact that it assumes symmetric network connections. Since measuring one-way delay over a network is quite difficult, NTP measures the round-trip-time (RTT) and assumes the one-way delay is exactly half the RTT. Of course, data latencies over the internet are never 100% symmetric, especially if wireless networks are involved.

However, despite the fact that NTP is not perfect, it is still the only standardized clock synchronization algorithm

ISBN/EAN – 978-90-5986-439-9

that is currently widely available, so it was selected as one of the *two* clock synchronization mechanisms included in the system described here.

In order to try to increase the accuracy of the clock synchronization for the inter-device synchronization scenario, a second, new, clock synchronization mechanism was developed for specifically this purpose: the so-called *ping-pang-pong* protocol.

The ping-pang-pong protocol works on the assumption that the shorter the network paths are, the lower the latencies and the smaller the absolute error in determining the one-way delay between two network elements. Based on this assumption, the idea is to have the two devices (TV and second-screen device) try to synchronize their clocks peer-to-peer, without an external server.

The method developed for this is similar to how the well-known ping protocol works: one device sends a short ping packet to another device, with the second one responding with a short pong packet. By including in the message the timestamp at the moment it was sent, it is possible for the first device to calculate the RTT between the two devices. In the newly developed ping-pang-pong protocol, a third message is added, so that the second device can also determine the RTT. In addition, by not only including the transmission time of the messages but also the reception time of all earlier messages (in 64bit Unix epoch time format), both devices can not only calculate the one-way delay between the devices, but their absolute time difference as well, providing clock synchronization.

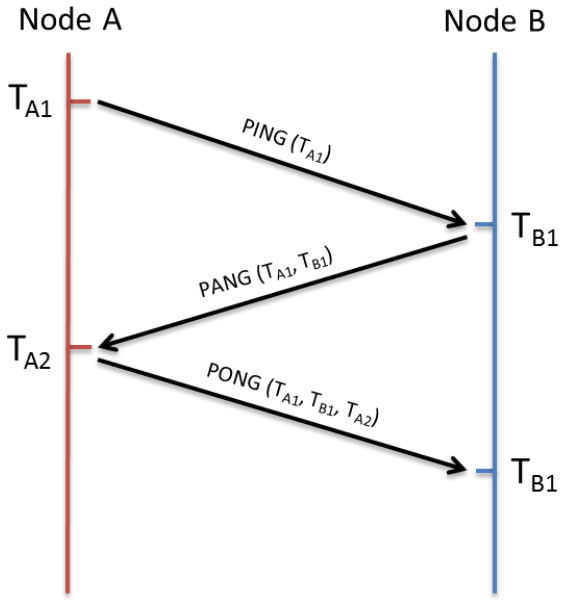

Figure 3: Ping-pang-pong protocol

Similar to NTP, this new protocol also assumes symmetric network paths. However, in this case the network paths are significantly shorter, typically only including a single hop when on the same Wifi network, which makes the absolute error smaller. In addition, well-known statistical algorithms can be applied to reduce the error in the one-way delay measurement even further.

4.3 Device Discovery

For device discovery, it was decided to use a mechanism that is simple to implement and can be deployed on a wide variety of devices. While standards like UPNP and DLNA have been around for many years and have aimed to solve the device discovery problem, interoperability has historically been a major problem for both. While the situation is slowly getting better, guaranteed interoperability and frictionless use between two devices running either of the two systems is still not a given due to the complexity of the protocols and many proprietary extensions. In addition, and maybe more importantly, implementing a full DLNA or UPNP client requires a lot of effort on the part of the app developer. Current mobile platforms such as iOS and Android do not include implementations in the platforms themselves which means that any app developer wishing to support them has to implement the standards themselves. For these reasons UPNP and DLNA have been rejected as candidates for device discovery.

Instead, the choice has been to utilize the combination of Multicast DNS (mDNS) [9] and DNS Service Discovery (DNS-SD) [10], together also known as ZeroConf or Bonjour. The main advantage of this combination is that it provides a simple to implement, lightweight but powerful method for service and device discovery. In addition, numerous libraries exist for almost all major programming languages and support is present in both iOS and Android. Most importantly, and industry adoption has shown this in the past few years, the relative simplicity of mDNS and DNS-SD means that implementations just work and interoperability is not an issue.

4.4 Application Bootstrapping & Communication

As discussed in section 2.1.4, the HbbTV application running on the TV is launched upon the TV receiving a so-called AIT tablet with embedded URL in the DVB stream. Once the HbbTV application itself is launched, all other functionality that needs to be present in the TV can be part of that HbbTV application through JavaScript code.

The remaining open issue is how the TV knows which content to tell the second screen device to start playing.

For this particular problem, two categories of solutions are available: server-based or stream-based. In the server-based case, the HbbTV application running on the TV queries a central server (e.g. hosted by the broadcaster) to request the location of the second-screen content. As part of this query the HbbTV application might include some identifier representing the content currently being watched on the TV (e.g. a channel or program ID obtained from the DVB stream).

The second type of solution, and the one that is implemented in the demonstrator, is placing the content URL in the DVB stream itself. In this case, a new field, the < auxiliary_content_url> field, is included in the AIT.

ISBN/EAN – 978-90-5986-439-9

The AIT specification allows for such new fields through so-called private descriptors [1][11].

Upon finding the AIT and extracting both the HbbTV application URL as well as the auxiliary content URL, the HbbTV application which is launched, communicates the auxiliary content URL to the second-screen application over the same WebSocket interface that is used to exchange synchronization tuples (see Section 4.2). To this purpose, a lightweight JSON based message protocol has been developed.

5 IMPLEMENTATION

Since the proposed demonstrator consists of two distinct components, a second-screen component and a TV/STB component, an important element of this project was to create both elements in a way that they could also work independently. As an example, instead of synchronizing with a DVB stream shown on a TV or STB, the second-screen application should also allow for synchronization between multiple tablets and/or smartphones on the same network. Similarly, the TV/STB components should also allow for synchronization with other TV/STBs at remote locations (although in this case using NTP is probably more accurate than the ping-pang-pong protocol).

As part of the FP7 FascinatE project, a Tiled Streaming application for iOS has already been developed [16]. For this new demonstrator, this application was extended with the described clock synchronization, inter-device synchronization and device discovery functionality.

For the TV/STB part of the demonstrator, the existing HBB-Next Synchronization platform [17] was leveraged. With this platform, it was already possible to perform heterogeneous inter-media synchronization, allowing for different types of media streams, such as DVB streams, HLS streams and MPEG DASH streams to be frame-accurately synchronized on a single device. As part of the project described in this paper, the existing platform was extended with inter-device synchronization and device discovery capabilities.

The HBB-Next synchronization platform builds upon the open source GStreamer framework [18], a pipeline-based environment that allows for flexible handling of media flows. While the demonstrator discussed in this document runs on a Linux-based PC, the GStreamer framework has also been implemented in various commercial chipsets, finding its way into commercial TVs and STBs, and making it an ideal environment for developing and testing new features with.

5.1 Impressions and performance

The following figures give an overview of the developed application.

Figure 4: Tiled streaming application running on an iPad

Figure 5: DVB-S stream on TV, synchronized with a video panorama of the same event running on a tablet

Since the research presented in this document is still on-going, extensive performance measurements have yet to be performed to determine the achieved synchronization accuracy. However, initial experiments with a video camera recording both devices in a single shot have shown the synchronization accuracy between the two devices to be at least within the 40ms boundary that determines frame-accuracy for 25fps content. Future experiments will have to confirm these observations.

In addition, the authors are planning to conduct tests to compare the performance of the developed ping-pang-pong protocol with the performance of NTP when used on local Wifi networks.

6 CONCLUSION

In this paper we have described a novel use case for immersive second screen experiences. By allowing users to use their tablet to navigate around an ultra-high resolution video panorama of the sports event they are watching on the main TV screen, the lean-back experience of watching TV has been merged with the more active experience of interacting with content.

In order to solve the synchronization problems present in the proposed use case, a set of synchronization components was described, among them a simple yet

ISBN/EAN – 978-90-5986-439-9

powerful method for synchronizing heterogeneous media streams between different devices and a new clock synchronization protocol.

Future work will focus on determining the accuracy of the proposed inter-device and clock synchronization protocols. In addition, the case of inter-destination synchronization, where the two devices are at different geographical locations, will be explored in more detail. Finally, work is currently on-going on scaling-up the used synchronization algorithms to be used in large groups of devices, where one-to-one communication between all nodes in a synchronization group is no longer efficient.

7 ACKNOWLEDGEMENTS

The research leading to these results has received funding from the EC Seventh Framework Programme (FP7/2007-2013) under Grant Agreement n°287848 (HBB-Next) and Grant Agreement no. 248138 (FascinatE).

References

[1] ETSI TS 102 796: "Hybrid Broadcast Broadband TV 1.5", v1.2.1, November 2012

[2] Khiem, Ravindra, Carlier and Ooi, "Supporting zoomable video streams with dynamic region-of-interest cropping", Proceedings of the first annual ACM SIGMM conference on Multimedia systems (MMSys '10). 2010.

[3] Mavlankar, Agrawal, Pang, Halawa, Cheung and Girod. "An interactive region-of-interest video streaming system for online lecture viewing", Proceedings of 2010 IEEE 18th International Packet Video Workshop. 2010.

[4] Alface, Macq and Verzijp. "Interactive omnidirectional video delivery: a bandwidth-effective approach", Bell Labs Technical Journal, March 2012.

[5] Brandenburg, Niamut, Prins and Stokking, "Spatial segmentation for immersive media delivery", Proceedings of the 15th International Conference on Intelligence in Next Generation Networks (ICIN), 2011.

[6] Howson, Gautier, Gilberton, Laurent and Legallais. "Second screen TV synchronisation", Proceedings of the 2011 IEEE International Conference on Consumer Electronics. 2011.

[7] Civolution website, http://www.civolution.com/home/, last accessed August 6th 2013.

[8] Never.no website, http://never.no/platform/, last accessed August 6th 2013.

[9] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762, February 2013.

[10] Cheshire, S. and M. Krochmal, "DNS-Based Service Discovery", RFC 6763, February 2013.

[11] ETSI TS 102 809: "Digital Video Broadcasting (DVB); Signalling and Carriage of Interactive Applications and Services in Hybrid Broadcast/broadband Environments", version 1.1.1, January 2010.

[12] ETSI TS 102 823, "Digital Video Broadcasting (DVB); Specification for the Carriage of Synchronized Auxiliary Data in DVB Transport Streams", version 1.1.1, Nov 2005.

[13] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, June 2010.

[14] Fette, I. and A. Melnikov, "The WebSocket Protocol", RFC 6455, December 2011.

[15] IEEE 1588-2008: "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems", 2008.

[16] Vimeo, 'TNO iCombine', http://vimeo.com/67199886, May 2013, last accessed August 7th 2013.

[17] Veenhuizen, van Brandenburg, "Frame accurate media synchronization of heterogeneous media sources in an HBB context.", Proceedings of the Media Synchronization Workshop 2012.

[18] GStreamer website, http://www.gstreamer.net, last accessed August 7th 2013.

[19] SAVANT website, http://dea.brunel.ac.uk/project/savant, last accessed August 7th 2013

[20] Matsumura, Evans, Shishikui and McParland, "Personalization of broadcast programs using synchronized internet content", IEEE International Conference on Consumer Electronics, January 2010.

[21] M. Armstrong, J. Barrett and M. Evans, "Enabling and enriching broadcast services by combining IP and broadcast delivery", BBC Research White Paper WHP 185, Sep 2010.

[22] Leroux, Verstraete, De Turck and Demeester, "Synchronized Interactive Services for Mobile Devices over IPDC/DVB-H and UMTS", 2nd IEEE/IFIP International Workshop on Broadband Convergence Networks. 2007

[23]