TIMING - SMALL STEP FOR DEVELOPERS, GIANT LEAP FOR THE MEDIA INDUSTRY

Njål Borch[1], François Daoust[2], Ingar Arntzen[1]

[1]Norut, Norway, [2]W3C, France

ABSTRACT

Timing is a fundamental property of media experiences. In particular, multi-device scenarios require a shared timing to provide engaging, coherent services to users. Trends are also that people have multiple devices available, and that they do not limit the usage to one device at a time.

The industry caters to these developments in a variety of ways, often using expensive, custom solutions to limited areas or targeting short term issues that lock down users to particular devices and services. This paper discusses some existing solutions, and indicates how they could benefit from shared timing, as provided by the suggested HTML Timing Object and online Shared Motions..

INTRODUCTION

People are using and interacting with an increasing amount of devices, often used simultaneously or combined with more traditional TV viewing. While watching, people are also active in product research, online shopping and even interacting with content by voting, commenting or sharing thoughts and content on social networks. The availability of sensor data opens for advanced multi-screen services. Key to engaging users is providing coherent user experiences across devices and technologies, ensuring correctly timed presentations of both media, interactive components and user input.

Popular multi device solutions are however still facing some fundamental limitations, both from a technical and user experience point of view. For example, while Sonos plays audio perfectly in a home and a SmartTV can show IP content, the Sonos speakers can't easily play the audio from the SmartTV while maintaining sync. Twitter might integrate fairly well with live broadcasts, but much less so for time-shifted content. Netflix is great for binge-watching, but it lacks mechanisms for social viewing beyond physically watching the same screen.

We see a pattern where successful multi-device solutions like Sonos and Chromecast seem to focus on small and well defined use cases. They commonly depend on a predictable environment, e.g similar hardware and closed software solutions and a physically shared network. This makes Web integration difficult, and adds usability hurdles as the requirement list for successful operation must largely be met by the user.

A web based online solution could meet user expectations more easily, as users can interact with their account rather than their physical, yet typically invisible, network. Online

solutions are however typically unable to meet user expectations in multi-device scenarios. Streaming a web radio on two devices will more likely than not create a highly confusing or annoying experience as the devices fail to play back content in synchrony. Due to differences in hardware, network latency, buffer sizes and several other factors, providing tight audio/video playback over the Internet might seem a challenging task. There is a relatively simple solution though: Shared timing and control.

The World Wide Web Consortium (W3C) Multi-Device Timing Community Group (1) has been working on a new model that exports timing as a first class citizen on the Web. The Timing Object (2) and Shared Motions (3) provide a programming model and an approach for creating natively multi-device experiences using HTML5 by adding globally shared timing and control. In our work, we have used the open reference implementation of the Timing Object proposal (4) and a hosted implementation of Shared Motion by the Motion Corporation (5).

In this paper, we look at how the industry approaches timing related issues, how they compare to a shared timing model, as well as how shared timing can provide further opportunities.

SHARED TIMING

Timing is typically still regarded as system internal - a player will have its internal clock. The clock might be exported to allow for example progress bars to be visualized, or an API might be available to change position, playback state or other control properties. A different approach is explicit timing, where timing is kept external from the player. The player will rather slave after the external clock as best it can. If there are internal clocks, e.g. hardware decoders, the player will do the necessary adjustments itself.

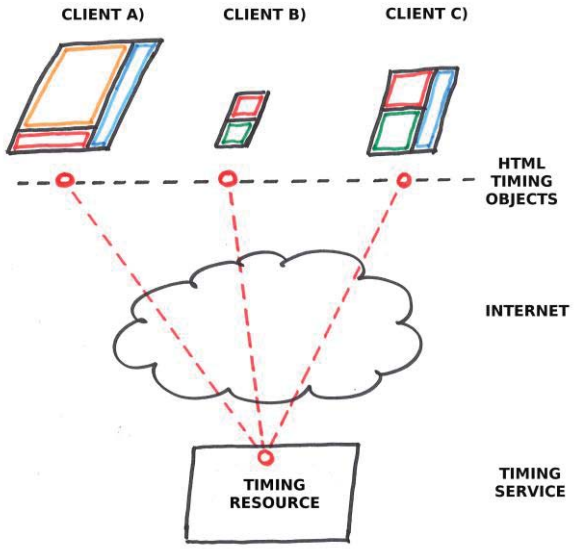

Figure 1 – Shared timing model

At the core of Shared Timing is Shared Motion. Shared Motions can be thought of as hosted, remote controllable, globally shared clocks. A Shared Motion has a URL, and any device connecting to this URL will share the same motion. Control can be symmetric (if access control allows), no master/slave relationships are necessary. Client side implementations can be done in user space, making Shared Motions extremely available. As latency is compensated for, Shared Motion provides similar accuracy regardless of networks, making it independent of underlying network technologies and topologies for accurate operation.

The Shared Motion is represented by a Timing Object running on the local device. Timing Objects running on different devices are precisely synchronized if connected to the same Shared Motion. Timing-sensitive multi-device applications may therefore operate consistently across devices, simply by ensuring that local playback on each device remains as close as possible to the local Timing Object. Hence, the complexity of the synchronization does not increase when more devices are added, and the need for APIs to control media players is vastly reduced. A progress bar can just as easily monitor a Timing Object as it can a video player, and updates to the Timing Object will be reflected by media players, making direct communication between control elements and players unnecessary. In practice, we experience that adding shared timing simplifies development by allowing a much greater modularity and code reuse as well as allowing focus to be put on a single part of an experience at a time

We therefore define shared timing as a globally available service providing shared timing information to all participants. This is substantially different from typical solutions today, where multi-device timing is limited to local networks, closed systems or is too coarse for advanced use cases. The model unifies several issues, providing a simple and efficient way to support natural and global multi-device applications. Perhaps most importantly, a full implementation and hosted service is provided, making it possible to achieve most of the suggested functionality in this paper today.

THE CURRENT STATE OF AFFAIRS

In this section, we will look at some industry solutions and how they approach timing related aspects. Note that these solutions are mostly proprietary and that documentation is not readily available. This exercise should still be useful, as we do not require a particular implementation of a shared timing model in our discussion. Any implementation of explicit and globally shared timing and control should give similar possibilities. Shared Motion and the Timing Object could be thought of as a model rather than a specific implementation in this context.

We see two main approaches to multi-device playback. Internal - exporting timing information from the internal clock of a media player, and external - sharing a common timing source. Exporting the internal clock is relatively easy, but we will argue that this solution lacks flexibility and accuracy as multiple multiple players have individual clocks. Another approach is to share wall clocks using NTP or other time protocols. This can give multiple devices and media players a common reference point, but further control is still necessary. Examples can be to map wall clock time to frame numbers of a video or pausing the playback at a given time and position.

Using shared timing there are two ways to ensure synchronized playback. One is to synchronize playback operations like skip, play and pause. If these are executed at the same time, one would expect playback to be synchronous. However, if there are differences in hardware, buffering latencies, decoding latencies, operating system latencies etc, this solution will not provide a good user experience. It is for example apparent that different devices might play back content at slightly different speeds to each other simply due to differences in decoding hardware or even sound cards. We call this player drift. The other solution is to let the player slave after the reference timing source. The player will itself correct for any discrepancy in its playback, meaning that differences in hardware and software can be adjusted for. This is the solution we have chosen for

synchronization of HTML5 media relative to the Timing Object, as it provides good experiences also in highly heterogeneous scenarios as shown in Borch (6).

Sonos

Sonos is well known for multi-device audio playback on their custom speaker system. Based on available patent applications (7), they appear to use a local network to synchronize playback operations. The patent indicates that local clock synchronization is used from a master device to all others, in order to ensure that the playback operations are executed at the same time on all devices. As such, Sonos are already using a shared timing model. There appears to be no correction for player drift, which means that Sonos might be dependent on the hardware having a very similar player drift. As they produce their own hardware, this is feasible at least in the short term, but combining new and old players could cause problems as hardware evolves. Devices also need to be configured in groups at the beginning of a playback session, as devices cannot join dynamically while the others are playing.

Making the players slave after the common timing resource, Sonos should be able to ensure correct playback on any device. In fact, using Shared Motion, Sonos should be able to work on different networks as well, removing the limitation of physical proximity or shared mediums. This would also open for integration with other systems, such as playing the correct sound for an OTT video service, by simply being handed the URL for the content as well as the URL for the Shared Motion. No further integration should be necessary. Allowing devices to dynamically join would be highly useful in larger setups, in particular commercial systems where devices might need changing while the rest of the system keeps playing (e.g. a shopping mall).

Chromecast (video):

The very popular Chromecast from Google is by nature multi-device (two-device) in the sense that there is always a control device involved. A user appears to push content from one device to a Chromecast. A fairly exhaustive API allows the controller to monitor and interact with the target Chromecast, including positions, volumes etc, using direct messaging on a local network. The Chromecast does not appear to use any multi-device timing, but relies either on having video pushed from the controller or exchanging playback positions and control messages with the controller. The Chromecasts can fetch the data directly from the source, and as such the model is highly open for a multi-device timing model. The requirement to be online in this scenario also means that online timing sources are readily available.

By integrating multi-device timing into the Chromecast it will be possible to cast to multiple Chromecasts simultaneously. This also includes playing the audio to alternative destinations, such as the Chromecast Audio (discussed below). Furthermore, online controls should fit very well with the web and online nature of the Chromecast, removing the need for local reconnections to control playback but rather making the playback part of online state. It should be mentioned that while the Timing Object and Shared Motion is typically thought of as a way to synchronize playback position, it can as easily describe the

current item in a playlist, global volume and so forth. Shared timing will also allow devices to leave or join playback dynamically, e.g. add a phone to a setup mid session.

Chromecast audio:

Chromecast audio provides multi-device audio playback. The addition of this functionality appears to also include Sonos equipment as Chromecast audio destinations. As such, it seems likely that Google is using the same protocols as Sonos for the Chromecast Audio.

Adding an online timing model to the Chromecast Audio is as such similar to both the Sonos and the Chromecast video cases discussed above. It would provide a natural bridge between the systems, making it possible to create services that seamlessly use both Sonos, Chromecast and Chromecast Audio devices in highly flexible setups..

Spotify

Spotify provides a number of multi-device features - it allows the user to log in to a number of devices, it shares playlists and some playback history. Spotify also provides an overview of which devices are available, which are playing and even provides remote controls and easy movement of a playback session between devices.

Adding shared timing to Spotify could open for vastly improved multi-device functionality. While Spotify is able to play music using both Sonos and Chromecast Audio and through that local multi-device playback, native multi-device playback could be done. Collaborative listening would allow groups of people to listen to the exact same content together, and adding interactive second screen material could be a fun way to experience music.

Netflix:

Netflix provides a highly popular online video service. It's compatible with a number of playback devices, and in some instances can be used with multiple devices, with one playback device and one control device. The media clock is however internal to the player, and might be exported through APIs of the playback system. As opposed to the Sonos and Chromecast scenarios, Netflix does not provide their own hardware, and are therefore not in the same position of control of the end user experience.

The lack of control of the full chain from backend to client side hardware makes shared timing even more compelling for Netflix. Any web enabled device should be able to provide a seemingly well integrated and coherent service, with multi-device playback and control a natural part of that. For example, visiting Netflix on your smartphone could give you the remote control to whatever experience you are currently having, regardless of which device is playing the content. Selecting an alternative audio track on said phone should be trivial, providing both accessibility options, multi language options and even silent TV style consumption without adding complexity for developers or end users. Social functionality like collaborative viewing are easily within range, as inviting another Netflix user to share your timing source will ensure correct playback on all devices. Similarly, moving from one device to another is as easy as opening Netflix on the target device and turning off the source device - there is no need for them to even speak to each other. Having two TV sets showing the same content in two different rooms without producing an echo might seem

trivial to users, and they might even be surprised if they do not play back synchronously. With proper support for shared timing, this is trivial.

BBC iPlayer / Web TV solutions

There are many providers of online TV, providing a range of live, catch-up and on demand functionality. These systems largely provide local playback, or support external devices like Chromecasts, AppleTV etc. They therefore use the player clock, possibly with the player clock being exported to a controlling device. These cases are all rather similar to Netflix, sharing the same lack of end-to-end control of the experience. They can also benefit similarly by gaining end-to-end control using shared timing. As an added point, broadcasters might be interested in outsourcing second screen material such as commentary, graphical representations etc. Shared Motions will provide a very simple mechanism for integrating with such externally produced and hosted content, as no direct communication is required - only sharing the motion..

SmartTV

There are a number of SmartTVs on the market. We have had a focus on the Hybrid Broadcast Broadband TV (HbbTV (9)), as the prevailing standard in Europe. While the currently deployed version of HbbTV (1.5) is known for its lack of synchronization functionality, this is being addressed in the next version (2.0). HbbTV 2.0 will be able to share it's timing information on a local network, making it possible for other devices to play back content in sync with the TV. Slaving after another HbbTV device is however optional, and is as such not likely to be supported. In other words, an HbbTV 2.0 can allow a tablet or phone to play in sync with the TV, but two TVs might not be able to play back the same content without producing echos.

Interestingly, we have done some experiments with HbbTV 1.5, and we were able to produce consistent, echoless playback on TV and web browsers for IP delivered content over the Internet in Borch (8). Live content delivered through the broadcast tuner lacks timing resolution, and therefore limits the possibilities to synchronization to within about one second. In our experiment, we let the TV become the master, as it was unable to slave accurately after a Shared Motion. Instead, we let the TV try to synchronize as best it could, and adjust the Shared Motion to reflect what the TV was actually playing back. In this way, the TV can be allowed to only do coarse playback control, while fine playback control is done by more capable devices, like tablets, phones or computers.

Shared timing can thus already be useful in order to include existing hardware solutions into an online world. Newer hardware could of course benefit vastly by being able to slave closely after shared timing objects. None the less, we have already demonstrated that it is feasible to add shared timing to existing HbbTVs, and the next version of this equipment should provide even better user experiences as they are more aware of timing issues and likely report their playback state more accurately.

Apple AirPlay

Apple provides a variety of devices that support AirPlay. There are a number of ways to perform playback on these devices, from synchronizing files locally to the different devices, playing directly from online sources using apps or streaming from a source to one or more

targets. AirPlay can do multi-device playback for up to six devices from a Mac. iOS devices appear limited to sending audio to only one AirPlay capable device at a time.

While Apple devices appear to have a number of different ways to operate, shared timing could help in joining them together, in particular from an end user point of view. The various modes of operation might get different benefits from shared timing, but if the limitations to multi-device playback are due to a lack of resources or persistence of mobile devices rather than artificial, shared timing should be a very helpful tool to enable this functionality on both iOS devices as well as any other browser capable device. Allowing precise sync with other services as suggested for Sonos is also a possibility.

Miracast and other direct stream transfers

Transferring the data to be played back to a remote device has been a popular way to share a screen. The advantages include that no support is necessary from the content provider and that it can provide relatively consistent results. In fact, the Chromecast supports such streaming if the content providers do not have native Chromecast support, as does AppleTV and others. The main issue with this approach is that the controller device is now responsible for fetching data from the content provider, possibly decoding and re-encoding it, and then transferring it to the target player. This is a very resource consuming approach, in particular as many people use their mobile phones as controlling devices. While still useful as a sort of wireless cable, the industry seeks to move away from this solution in the use case of linear media consumption. Multi-device playback is further complicated by the need for streaming the data to multiple devices as well as providing the timing information. Shared timing could provide accurate information about actual playback time, allowing other devices or outputs to delay their presentation to match the (locally added) delay of the streamed content.

POTENTIAL OF SHARED TIMING

In the following table we have mapped the discussed solutions vs. functionality. Currently existing technology (even if it's limited) is marked with an X or text in green. The blue fields indicate that we've identified areas that can be easily improved using Shared Timing. White fields might still be of interest, but has not been discussed in this paper.

Remote control indicates using a secondary device (typically a phone) as remote control of an experience. Device independence allows users to move between devices without needing manual interaction (e.g bookmarks), Local multi-device playback allows multiple devices to play back the content simultaneously while connected to the same local network, without creating annoying echoes. Online multi-device playback is the same thing, but over and across any network. Collaborative viewing allows multiple users to share the experience across devices in a social group. Second screen is anything from split-times in sport to quizzes, comments, maps or other interactive material (e.g. web pages with timed content). Integration indicates integrating with other external services, such as playing the sound from a TV on a stereo, or time shifting tweets of a sporting event. Multi-device accessibility provides accessibility features on a personal device, e.g. provide a customized audio track for visually impaired, or audio with frequencies tuned to

match hearing impairments. The list is in no way exhaustive, but is only provided as a way to illustrate the discussions of this paper.

	Remote Control	*Device independence*	*Local multi-device playback*	*Online multi-device playback*	*Collab. viewing*	*Second screen*	*Integration*	*Multi device Accessibility*
Sonos	*X*		*X*					
Chromecast Video	*X*							
Chromecast Audio	*X*		*X*					
Netflix		*X*						
iPlayer & WebTV		*X*				*embed*		
SmartTV	*v2*					*v2*		
AirPlay	*X*		*X*					
Miracast	*X*							
Spotify	*X*	*X*						

It appears that a number of different solutions are aiming at similar multi-device features. Remote controls are particularly easy to implement, as the level of synchronization can be quite coarse. Shared Timing moves a lot of advanced functionality well into the realms of practicality.

CONCLUSION

We have described a model for shared timing based on the proposed W3C Timing Object in combination with the Shared Motion concept. We have also looked at some of the industry leaders and how they approach timing related issues. Where practical experiments have been performed, we use the reference implementation of the Timing Object (4) and the Shared Motion implementation by the Motion Corporation (5). A brief discussion has been made about how the media industry could benefit from using a shared timing model, including some specific examples. We have argued that Shared timing both unifies a number of issues, while at the same time removes complexity, and we argue that this will open for vastly improved services and integration between systems. Even more excitingly, these possibilities are within the realms of current web browsers and

a number of existing devices. We expect that the industry will be eager to provide smooth end-to-end solutions that put the content and the user in focus, and that media can find it's way into engaging, online social scenarios. New business opportunities should become apparent as creativity is unleashed on natural multi-device platforms.

Shared timing is a game changer.

REFERENCES

1. W3C Multi-Device Timing Community Group. https://www.w3.org/community/webtiming/
2. Arntzen, I., Daoust, F. Borch, N. 2015. Timing Object draft specification http://webtiming.github.io/timingobject/
3. Arntzen, I., Borch, N.,and Needham, C. 2013. "The Media State Vector: A unifying concept for multi-device media navigation". In Proceedings of the 5th Workshop on Mobile Video (MoVid '13). ACM, New York, NY, USA, 61-66
4. W3C Multi-Device Timing Community Group. 2016. timingsrc Timing Object implementation, http://webtiming.github.io/timingsrc/
5. Motion Corporation, http://motioncorporation.com/
6. Borch, N. and Arntzen, I. 2014. "Distributed synchronization of HTML5 media", Norut report, ISBN 978-82-7492-295-2
7. Sonos. 2005. Patent application WP 2005/013047 A2
8. Borch, N. 2016. HbbTV 1.5 experiments, https://www.w3.org/community/webtiming/2016/04/04/hbbtv-1-5-experiments/
9. HbbTV standards documents https://www.hbbtv.org