Challenges for Multi-Screen TV Apps

Jean-Claude Dufourd°, Max Tritschler*, Niklas Schmücker*, Stephan Steglich*

°) Telecom ParisTech/LTCI
37-39 rue Dareau
75013 Paris, France
jean-claude.dufourd@telecom-paristech.fr
*) Fraunhofer FOKUS
Kaiserin Augusta Allee, 31
10589 Berlin, Germany
{max.tritschler,niklas.schmuecker,stephan.steglich}@fokus.fraunhofer.de

ABSTRACT
HbbTV, an emerging standard for interactive television, is being deployed in various European countries, creating bandwidth demand for interactive services in existing saturated broadcast networks. It is a first step in the direction of TV apps, but we believe the winning technology in that area is multi-screen apps on and around the TV. In this paper, we describe the current situation with HbbTV and apps, what can be done with existing technologies, high-potential scenarios and our plans to make them possible by combining existing standards with limited extensions, creating a platform implementing a multi-application service model, and open source software tools spanning management and production of this new type of service, within the Coltram project.

Keywords HbbTV, interactive TV applications, resource management in broadcast, multi-screen and distributed applications, inter-widget communication.

1 INTRODUCTION

HbbTV (Hybrid Broadband Broadcast TV), a standard targeting interactive TV, is getting more and more traction since its inception in 2009 and standardization by ETSI in June 2010 [1]. HbbTV is actually a conformance, certification and promotional effort as well as an industry standard adding interactive functionalities to TV by aggregating and profiling existing standards such as CEA for CE-HTML, MPEG for video, audio and transport, DVB for signaling, OIPF (Open IPTV Forum) for APIs to manage the link between the interactive HTML and the various subsystems, etc. It builds on well-known and proven technologies, shared with most of the Web and the mobile environment, such as HTML, CSS and JavaScript to create interactive applications.

HbbTV was created from the premise that interactivity will be useful for the future of TV as we know it. Almost everybody infers that today's TV should become interactive, and this means that the TV screen should display interactive user interfaces and that the current TV command device, i.e. the remote control, should be the input device for the interaction.

There are many examples of attempts to improve on these assumptions, the most recent being the MOVL connect platform [7], which is a proprietary platform for developing multi-user and multi-device applications targeted at connected TVs and mobile devices, Apple's AirPlay [8], which allows controlling media playback on AirPlay-enabled devices remotely from e.g. a MacBook or an iPhone. There were also many versions of improved remote controls, including some with pointer technologies and others with touchscreens.

We do not believe in interactivity limited to the TV screen controllable only with the TV remote. And yet, we believe that HbbTV is a key enabler for an ecosystem of applications and services in the home around the TV. But even if the interactivity is TV-channel-related and received by the TV, it should not be confined to the usage of the TV screen and interaction through the remote control. More enabling technologies are needed.

There is currently intense activity around "multi-screen" scenarios [7][8][10][12][13], i.e. the possible cooperation between the TV and other connected devices in the home: ***smart phones***, representatives of a class of personal interactive mobile devices, ***tablets***, representatives of another class of family/shared mobile devices, ***desktop computers***, a class of non-mobile devices, etc. The activity has multiple forms:

- Developers try to use existing tools to provide services around TV, with varying degrees of synchronization; applications are either adding to the TV experience and focusing the viewer back on to the TV or trying to steal the viewer attention away from the TV towards a more personal, channel-independent experience.
- Larger industry actors are investigating which products or standards would help create an ecosystem around TV in the home, possibly on the basis of existing Web technologies.

With Coltram we aim to create a model for modular applications that are designed as constellations of smaller collaborating components. These components may be individually distributed across devices at runtime on request of the user, without losing their current state. Coltram will provide a location-transparent communication mechanism that lets the system handle the complex task of managing cross-device communication and synchronization between components, so that the developer may focus on his application. The scope of Coltram will go beyond using companion devices as remote controls for the TV [11] and not be confined to static distribution configurations that do not satisfy the highly dynamic availability of additional devices in the home. Further, the employment of Web technologies will enable cross-platform interoperability for Coltram-enabled applications and lower the barrier for developers and manufacturers to adopt this new approach to applications that are suited for multi-screen scenarios by design.

First we describe the multi-screen challenges from a service-oriented perspective (Section 2) and introduce the underlying concepts of the proposed Coltram architecture (Section 2.1). Then we will describe four scenarios for future multi-screen enabled TV apps (Section 3), from which we then derive requirements (Section 4) and challenges (Section 5) that need to be addressed to realize these scenarios. We conclude with a brief presentation of the Coltram components and concepts (Section 6).

2 A Service Point of View

Today, any service to the user is either implemented as a (native) software application, running on a particular device and OS, or if implemented as a Web application, then as one document (view/session) at a time and coming from the Internet (online).

The home environment is constituted of more and more devices with extremely varying characteristics. It is heterogeneous, whatever the point of view: devices have a screen or not, large or

small, have input capabilities or not, are personal and public, have computing power or not… Such an environment is, in a sense, multi-centralized. Specific features of a device implicitly make this device a server for these specific features. Any service requiring these features must connect to that device or to a service exported by that device. Parts of the service may be executed on a choice of devices, and as such, may need to be adapted. A service, on the other hand, should be easy to use. Detailed management of its mapping onto devices should be transparent to the user. More: if the availability of devices changes, then the service should be transparently reconfigured, without loss of execution context.

We believe in a new model of service, which is a collaboration of multiple applications running on multiple devices, including discovered devices, each application not tied to one device and able to move to another device without losing state, and coming from Internet or broadcast or the local network.

And yet, because native software applications will stay as an important component of the home architecture, the new service model should allow the seamless integration of native and Web components, and the seamless switching between native and Web components.

The challenge is to allow the convergence of Internet, the mobile world, the home media (TV, set top boxes, gaming consoles, etc.) world within the home network by providing this new service model, lowering the cost of designing and using these services for all actors.

In the WWW, services are implemented with Flash or Web standards, with one server for the application and possibly others for media. The situation is similar on mobiles, with less reliance on Flash but more dedicated apps and slightly different Web standards. Emerging standards such as HbbTV in the TV domain are pointing toward a similar model (without Flash or dedicated apps).

The home network however is quite different: there are many different devices, many of which are not always available; communications are not naturally centralized, but device-to-device; but Web standards are a common base for the design of services if peer-to-peer discovery and communication interfaces can be added.

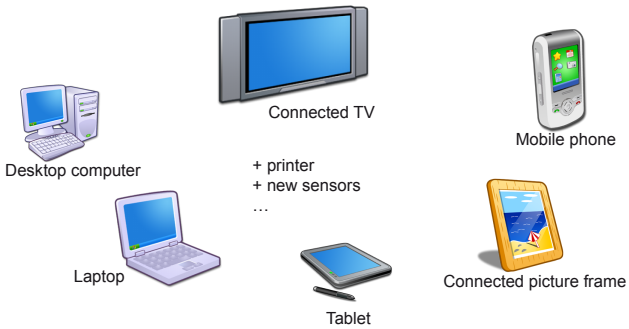

Fig. 1. Typical home network

Services on Internet or on mobiles are designed to run on one single, relatively powerful device, which has computing power and a screen/speakers and interaction capability as well as possible extra features such as a camera.

In the home network, capabilities required by a service are rarely concentrated on a single device: usually there are multiple choices for each requirement, and the best available choice may

change from moment to moment. Concepts from various origins need to be merged: ubiquitous/pervasive computing, where an application can run on any device; distributed computing, where a service may be implemented as several communicating applications running on different devices; and Web applications, where the service is implemented as a document presented in a Web browser (with scripting).

2.1 Enabling distributed TV apps

In this paper, we propose *Coltram*, a platform for distributed services in home networks and beyond. We see Coltram as an enabler for many innovative use cases in future TV scenarios. The Coltram infrastructure builds on top of the *webinos* [6] framework, a federated Web runtime that offers a common set of APIs to allow applications easy access to multi-user cross-service cross-device functionality in an open, yet controlled manner, to which Coltram will add a flexible model for distributed collaborative services and a service platform allowing for

- advanced, context-aided discovery of devices and services,
- seamless migration and ad-hoc distribution of atomic or composite applications between different devices,
- dynamic UI adaption for applications that were moved between devices with e.g. different display sizes.

Last but not least, Coltram will provide application authoring tools which help developers to easily make use of the mentioned Coltram platform features.

Pervasive computing addressed the various problems of mapping applications onto heterogeneous sets of devices. Service adaptation was studied in multiple contexts. However, there are no solutions yet addressing the Coltram combination of features: services implemented as constellations of communicating Web applications; Web applications can be split so that a device-dependent part can stay on that device; devices come and go, and applications may need to be moved during the life of the service while keeping the service state.

3 SCENARIOS

The following sections describe scenarios in the connected TV context which can be realized with the proposed Coltram architecture but not with the HbbTV architecture as it is. We further outline the corresponding challenges that need to be addressed in order to realize each scenario.

3.1 One on One

The general scenario is that of an application received by the TV, and requested for use by one viewer among many.

If the application does not require the use of TV APIs, then the application can be sent as is to the personal device of the viewer. Else, the application needs to be split into two communicating parts: the user interface should be sent to the personal device of the viewer, and the business logic including calls to the TV API should be executed on the TV itself. Communication between the two parts should be supported by the service platform.

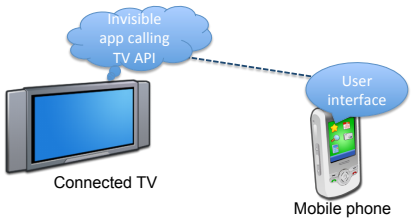

Fig. 2. One on One Scenario

A typical example is the electronic program guide (EPG) associated with the current channel. At this time, with HbbTV, the EPG could be part of the HbbTV application accessible through the red button (either directly, or as a sub-application). If there are multiple viewers, there is a good chance that only one viewer is actually interested in the EPG information. If the EPG information is displayed on the TV screen, only one viewer is satisfied, and the others are actually disturbed. If there is another screen available, it will be optimal to move the EPG display to that other screen.

To implement this scenario, there are many possibilities. HbbTV needs to be extended to be able to deal with this scenario, but how big the extension should be will be discussed below. Here is a list of features of this scenario that are not currently solved within HbbTV:

- the TV should be aware of the other device, i.e. discovery through UPnP or Bonjour;
- the application on the TV should be able to communicate with the application on the other device, to pass information that HbbTV application usually request by calling an JS API;
- the TV screen resolution and aspect ratio are not the same as that of the screen of the other device, so some layout adaptation needs to happen, or possibly a different layout needs to be provided;
- the interaction on the other device could be touchscreen (definitely not remote control), so user interaction should be adapted;
- the application on the TV and the application on the other device should be synchronized

3.2 Play Along TV Game

The general scenario is that of a TV game, with an application that allows viewers to play along. With HbbTV as it stands now, either there is just one local player, or all viewers have to collaborate as one player.

For such a game, the central application running on the TV would be designed to detect all local devices that would allow a viewer to play along, and to propose to all these devices to load the appropriate version of a play along application. Each version of the play along application would have the appropriate user input (touchscreen on a tablet and some phones, keyboard on some phones, etc.), and be synchronized with the TV program and the application running on the TV. Results for each player would be communicated by the play along application to the central application. The central application would possibly not need a user interface, just communication interfaces to all play along applications, and a connection to send results back to the server, e.g. to share high scores with all players in the country.

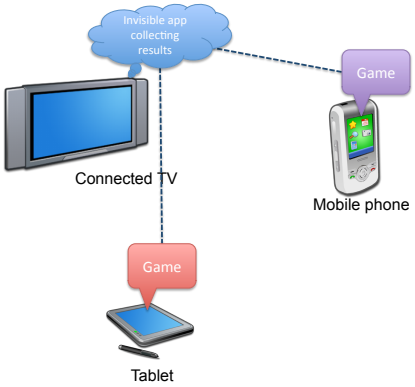

Fig. 3. Play Along TV Game

To implement this scenario, missing features are very close to those for the previous scenario:

- the TV should be able to discover many devices, not just one; UPnP and Bonjour provide that;
- the TV application should be able to communicate with multiple applications running on other devices, which makes the application more complex but is the same requirement;
- same requirement for screen adaptation, interaction adaptation and update propagation.

3.3 Device "Unavailable"

The viewer has engaged with the "One on one" scenario with the family tablet. During that usage, the tablet becomes unavailable, e.g. a kid "preempted" the tablet to play some new game. Then there are two possibilities for the viewer to go on with the service:

1. Before the tablet is preempted, the viewer requests a list of possible devices on which the service can be migrated, chooses one, and the service is restarted, keeping the service context, on e.g. the mobile phone of the viewer. We call this the application push migration scenario.

Fig. 4. Push Migration

2. From the viewer personal phone, a list of in-use services is requested, the viewer finds the one running now in background on the tablet, and requests its migration to the phone. We call this the application pull migration scenario.

Fig. 5. Pull Migration

For either push or pull migration of applications, the missing features are:

- the devices, or the browsers on the devices, need to recognize each other as "hosts of movable applications"; so each device needs to advertise itself as provider of that service and implement pull request, push request and the details of application migration;
- an application should be easy to move, so all of its resources should be packaged together and our suggestion is to adopt W3C Widgets-style packaging 5 which provided a simple organization of zip packaging for a web application;
- the execution state of an application should be serialized and sent with the application.

3.4 Service Mobility

In the fourth proposed scenario, a service is made available to a device residing on a different network: Alice and her husband Bob watch a movie on their HbbTV set. Bob decides to leave for work, but would prefer not to miss the rest of the movie. Hence, Alice authorizes Bob's tablet to receive the stream via their home network, using a modal menu on her TV, which automatically suggests nearby devices that are able to play video. Bob chooses to accept the offer by selecting the respective option in a dialog window shown on his tablet. Later in the bus on his way to work, Bob continues watching the stream, served directly from their home network.

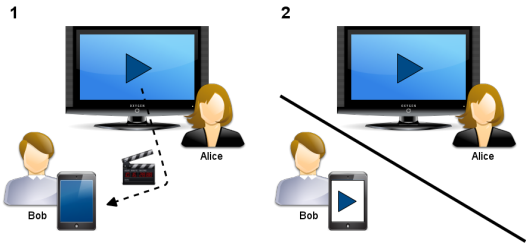

Fig. 6. Service Mobility

In order to enable this scenario, the Coltram platform must deal with several challenging aspects:

- Offering transparent access to local discovery mechanisms. The discovery results must be filterable by device capabilities and other constraints.
- Providing media content to remote devices. In particular, QoE and adaptive streaming mechanisms need to be put in place to ensure that home networks can handle the traffic load. Individual components might have to be placed in the Cloud.
- Political questions concerning content ownership and DRM need to be addressed in the implementation. For example, it should not be possible to stream copyright-protected content to arbitrary, alien devices without honoring restrictions imposed by copyright holders.

4 REQUIREMENTS

A generic requirement for all of the scenarios above is the automatic discovery of devices. It is not realistic to ask for the general public to declare all of their devices on their home network.

It should be possible to build text-based and graphics-based service interfaces. This should be non-controversial: use HTML+CSS for text apps, SVG for graphics apps.

Services should be available as widgets, compatible to W3C Widgets. The extra technology to make HbbTV compatible with W3C Widgets ("Packaging and Configuration" [5] and "Widget Interface" [9]) is comparatively very small.

A service should be available on any device, even if the service is about e.g. TV. A user should be able to vote on her TV (with a remote control), but also on her mobile, especially if there are more viewers. One way to achieve this is by solving another requirement: it should be possible to create a service as a constellation of cooperating elements (widgets) running of possibly multiple devices (distributed services). In the example of the TV vote, there should be one widget on each voting phone and one widget on the TV to collect and send the votes back. This implies communication between widgets.

It should be possible to build services with data from any origin, broadband or broadcast or local. The user should not be aware of these differences, and it should be possible to implement innovative business models on the complementarities between delivery modes. A widget should work the same way whether it is used during the broadcast, or when used with catch-up TV.

The user should also be able to switch devices at any time without losing the context of the current service, for example, for nomadic use, for a better and more organized user interface, and for extending the functionality of running applications through external device capabilities.

For example, the program guide may be easier to browse on a tablet, even if the data still comes from the TV.

To be able to deal with a dynamic set of available devices, the environment should provide discovery and service protocols. Discovery is necessary to deal with dynamic networks, to remove the need for the author of a service to know the address of a subservice in advance, as well as to deal with devices coming in or going out of the home network. You should be able to vote even if you are at a friend's home.

A service should behave the same, whether it is provided by hardware, by a native application, or by a widget. This is to be inclusive about the type of device that can be integrated in the home environment.

It should be possible to build services on top of other services, so as to easily create mash-ups, as well as multiple interfaces for the same service.

Even if we may have technology preferences, it should be possible to replace any chosen technology by an equivalent, without breaking the service model: replace a discovery protocol with another, a scripting language by another, a format by another, etc. The service model needs to be future-proof.

5 CHALLENGES

Although our every day's devices are getting more and more connected due to the latest technological achievements - almost every home has a wireless and wired network installation, TV sets and set-top boxes are connected to the Internet, mobile devices have WLAN and 3G data flat rates - the devices can talk to each other but there is no common application model yet. Some standards like UPnP addressing the interworking are partially used for interconnecting home devices for multimedia services, but often rely on communicating devices being on the same local network, which is typically not the case between, for example, the TV and the phone a user owns, or the tablet device of a visiting friend.

A real breakthrough is still missing! We need to address exactly this missing step in the ongoing convergence of home networks, TV, and Internet: developers of applications need to be supported to reflect the advantages of this convergence. They should not be forced to develop against all the different characteristics of devices (features, screen-size, I/O support, resources, etc.). Instead there is a clear need for a standardized, open cross-device platform, on which application run and can be automatically (or with the best possible support of the underlying platform) adapted to the respective execution context.

This goal is the innovating characteristic of our project. The pre-conditions are complex and requirements very high, as especially in the consumer electronics domain there is a very high claim for very good usability and user experience. Any solution not fully satisfying these claims will hardly be successful on the market.

On the other hand, the demand for this type of support for application developers is huge. With the introduction of Web 2.0 related technologies (JS, JSON, AJAX, etc.), which led the Web to become an interactive application platform rather than only a document exchange and presentation service. This led to an explosion of the Web and the content (including apps) in it. Today, it is the most dominant application platform used.

Coltram will support application developers in the consistent development of their applications across devices. This will also include the support for distributed execution / deployment; i.e. an "application" can be used on more than one device at the same time, where the different installations will synchronize and coordinate each. In this way the same application might represent a video playback widget on a TV device and a UI control on a tablet.

These different but adaptive representations of the same services need to be reflected by Coltram and the developer needs to be supported to deal with its complexity. Beside the technical aspects of the actual (distributed, orchestrated) execution of the application, the authoring aspect during the development of an application is a major objective and innovation of Coltram.

6 IMPLEMENTATION

In the previous sections we have described the requirements and challenges for innovative multi-screen scenarios in which consumers have the freedom to use any available device to serve as additional screen and even to control applications running on e.g. the TV. Within this section we present the underlying concepts of the Coltram service model and platform and how they meet the identified requirements. Coltram targets a wide range of device types including but not limited to smart phones, tablets, HbbTV connected TV sets and computers.

6.1 Coltram Service Model

In Coltram, services will be modeled as sets of communicating applications. Each application can be dynamically moved to the "best possible device" at any point in time, if its required features do not tie it to one device. The W3C widget specification [5] defines the basic model for a communicating application, extended with discovery, service capabilities and requirements on hard- and software, reaching forward in the direction of the work of the Home Network Task Force of the W3C Web and TV interest group [10]. Developers may use this model to combine multiple communicating applications into one complex service, which may be packaged and deployed as a single component. This composite service may in turn again be use within an even more complex service.

Inter-widget communication in Coltram will be implemented using event-based publish-subscribe messaging patterns [2][3][4], allowing for loose coupling of connected services and for full location transparency from developer perspective.

6.2 Coltram Application Platform

The Coltram service platform will be built on top of webinos [6], a federated Web runtime comprising a common set of APIs to allow applications easy access to multi-user cross-service cross-device functionality. Coltram will extend and augment the underlying technologies provided by webinos to support high-level functionalities, where webinos provides the lower-level technology enablers.

Coltram will provide a device and service discovery mechanism, which extends webinos' discovery with the additional features that are defined in the Coltram service model: service capabilities and requirements on hard- or software. Devices advertise their availability and will be registered in Coltram until they are not available anymore.

When services migrate from one device to another, Coltram has to ensure that the current state of the service does not get lost. When the user triggers a complete migration of any service, Coltram serializes the local state, pushes it to the new device und deserializes the data. For unexpected shutdowns of a device, e.g. when the battery died, this is not easily possible, although there are workarounds, ranging from storing periodical snapshots to continuous synchronization of the service's local state to a server. For services that are running distributed on different devices, Coltram will provide an efficient real-time synchronization mechanism to developers, which allows quickly propagation of changes between all of the distributed parts of the service.

6.3 Coltram Authoring Tools

The Coltram model induces extra complexity for the author, on top of the complexity of widget authoring. To compensate this, one part of the project is dedicated to making the creation of Coltram services easier. The following authoring aspects will be considered: authoring services consisting of a constellation of communicating widgets; authoring for hybrid delivery, i.e. using synchronized media coming from multiple delivery mechanisms, such as broadcast and broadband; authoring of applications that are mostly script.

The emphasis will be on extensible, collaborative tools, making it easy to reuse previous designs, and with facets for different authoring expertise levels, including one facet for a wide audience.

7 CONCLUSION

In this paper, we have presented several ideas for such use cases and outlined how Coltram will address the most important requirements that must be met. We strongly believe that Coltram will enable developing innovative and compelling scenarios for future TV apps that will increase user engagement and interaction.

Results from on-going research activities in Coltram regarding local and remote device discovery, service discovery, concepts for adaptive cross-device user interfaces and seamless synchronization of distributed services will be contributed to the standardization efforts within OIPF, HbbTV and W3C.

8 REFERENCES

1. HbbTV, ETSI Technical Specification 102 796, “Hybrid Broadcast Broadband TV”
2. C. Concolato, J. C. Dufourd, J. Le Feuvre, K. Park and J. Song, “Communicating and migratable interactive multimedia documents”. Multimedia Tools and Applications, May 2011 [DOI 10.1007/s11042-011-0805-2].
3. B. Hoisl, H. Drachsler and C. Waglechner, “User-tailored Inter-Widget Communication Extending the Shared Data Interface for the Apache Wookie Engine”. Proceedings of ICL 2010, Hasselt, Belgium
4. S. Sire, M. Paquier, A. Vagner, J. Bogaerts, “A Messaging API for Inter-Widget communication”. Proceedings of WWW 2009, Madrid, Spain
5. W3C, “Widget Packaging and XML Configuration”, September 2011, http://www.w3.org/TR/widgets [online; accessed 28/03/2012]
6. webinos, “webinos whitepaper”. June 2011
7. MOVL, http://movl.com [online; accessed 28/03/2012]
8. Apple AirPlay, http://www.apple.com/de/itunes/airplay/ [online; accessed 28/03/2012]
9. W3C, “The Widget Interface”, Candidate Recommendation, December 2011, http://www.w3.org/TR/widgets-apis/ [online; accessed 30/03/2012]
10. W3C, “Requirements for Home Networking Scenarios”, December 2011, http://www.w3.org/TR/hnreq/ [online; accessed 30/03/2012]

11. J.P. Barrett, M.E. Hammond and S.J.E. Jolly, “The Universal Control API v.0.6.0”. BBC Research & Development, Research White Paper WHP 194, June 2011.
12. S. Glaser, B. Heidkamp and J. Müller, „ Next-Generation Hybrid Broadcast Broadband: D2.1 Usage Scenarios and Use Cases”, HBB-NEXT Project Report, December 2011.
13. S. Slovetskiy and J. Lindquist, “Combining TV service, Web, and Home Networking for Enhanced Multi-screen User Experience”. Ericsson Position Paper for the 3rd W3C Web and TV workshop, September 2011.