tversity

Blending OTT with Traditional TV

Enable a TV App Ecosystem

via an

HTML5 Browser in the Cloud

Synopsis

HTML5 is the ultimate technology for cross platform TV apps, however most TV sets and set-top boxes do not support HTML5. A cost effective cloud based solution to this fundamental issue is presented in this paper, with the intention to enable an ecosystem of 3rd party TV apps on top of the existing pay TV platform.

Introduction

According to Netflix’s CEO, the future of TV will be app heavy. When it comes to apps, recent attempts by all the smart TV manufacturers to create app stores based on proprietary technologies have failed miserably. It is now commonly accepted by industry pundits that web technologies hold the key to cross-platform TV apps - and that, of-course, means HTML5.

It is, therefore, quite unfortunate that most TV sets and set-top boxes do not support HTML5, and those that do support it, offer typically partial implementations that do not truly abstract the underlying hardware. This is a fundamental roadblock for adoption and usage.

The purpose of this paper is to present a cloud solution that allows any connected device (a recent smart TV or STB, ultra-cheap dongles or even legacy boxes are all supported) to deliver HTML5 content to existing smart TV users and pay TV subscribers all over the world.

Why HTML5?

HTML5 ubiquity on desktops and mobile platforms is well known, and, in fact, is now taken for granted. What is less known, is the popularity of HTML5 for creating TV apps. As an example, consider the TV apps from YouTube and Netflix (arguably the TV apps that matter the most to consumers) - both are now offering HTML5 based solutions for TV. YouTube, which used to allow native app development based on their API, is now mandating transition to HTML5 (see their HTML5 app for TV at https://www.youtube.com/tv) and Netflix has been basing their TV apps on WebKit for a while now (see here and here).

Like YouTube and Netflix, many online video publishers, are targeting the big screen with unique 10 foot experiences based on HTML5 (some example are available here). This should not be surprising, given the fragmentation of devices and architectures in the TV space. Without HTML5, online video publishers would need to create a few dozen versions of their app to reach all the relevant TV devices, something that is clearly not manageable.

Why HTML5 in the cloud?

In order for all these great HTML5 apps to be available on televisions, the TV needs an HTML5 browser (whether built into the TV, or provided via a separate box). Unfortunately, the kind of hardware deployed today in households all over the world, is not capable of delivering a good HTML5 experience, this includes smart TVs, cable/satellite/IPTV set-top boxes, and OTT media devices.

Given that the typical refresh cycle for living room TV sets and set-top boxes is about 6-7 years, it is clear that this situation is not going to be amended anytime soon. At the same time, consumer demand for playing web videos on the big screen is ever increasing, creating a wider and wider gap between market demand and the solutions provided by the industry - we call this the hardware incompatibility challenge.

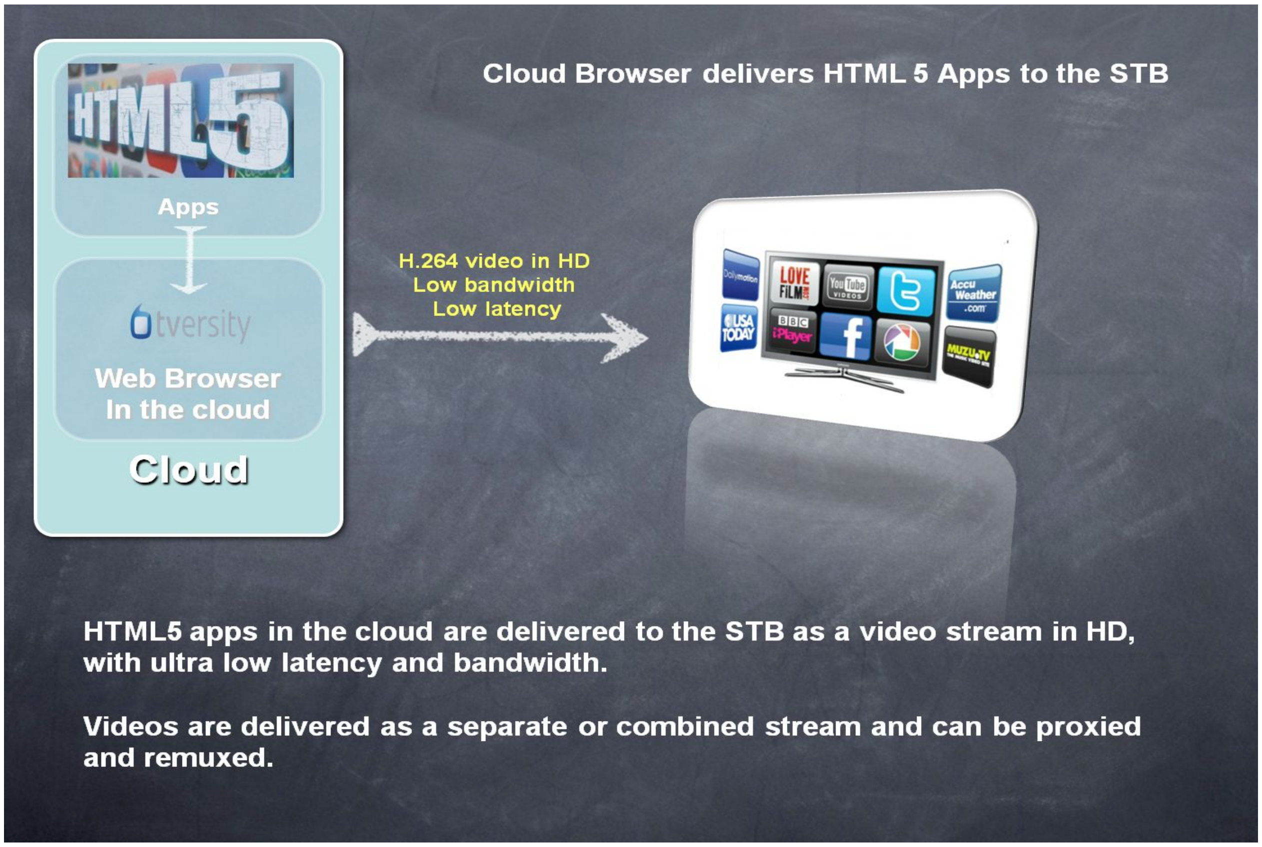
Running the HTML5 browser in the cloud, and delivering the content of the browser window as a video stream, makes it possible to bring HTML5 apps to the big screen - today. The one thing that every TV and set-top box can do well is play a video and therefore virtually every IP or QAM enabled device instantly becomes HTML5 capable thanks to the cloud.

Hardware incompatibility is not the only reason to utilize the cloud. The cost of a set-top box (hardware and software) and associated on-site services have been a real pain point for pay-TV operators and many have been looking to drop these boxes altogether. However dropping these boxes means losing control of the living room. Without these boxes it becomes impossible to deliver a consistent user experience and to guarantee a fully working end-to-end solution. It also may imply paying a commission on every transaction to some new living room gatekeeper (similar to the app store fee paid by app publisher in the mobile world).

Once again the cloud holds the key to effectively dealing with this challenge (the set-top box challenge). By moving most of the functions of a typical set-top box to the cloud, one can replace a typical box with a significantly cheaper counterpart that rarely needs to be updated, eliminating most of the hardware costs as well as most of the costs associated with on-site services. Furthermore by utilizing web technologies one can eliminate the traditional software costs of set-top boxes and enjoy web pace innovation. All this is achieved without relinquishing control of the living room and thus the cloud along with the adoption of web technologies appear to be the only good solution to this paradox.

Finally, the security and stability of set-top boxes are absolutely critical. While web browsers running on set-top-boxes, have been used (mainly by IPTV operators) to deliver interactive user interfaces, those were typically created by the operators. Allowing third-party publishers to target such a browser while using the public Internet, gives rise to a whole new set of security risks and challenges. The cloud eliminates those issues by isolating the box from the apps thus ensuring that the box can never be compromised or made unstable due to an ill-behaved or malware infested web application.

To better understand the idea behind a web browser in the cloud, consider the following diagram:



HTML5 apps in the cloud are delivered to the STB as a video stream in HD, with ultra low latency and bandwidth.

Videos are delivered as a separate or combined stream and can be proxied and remuxed.

The concept seems straightforward, however it is not at all obvious that this kind of a solution can be delivered in a cost effective way.

The Required Solution

The best way to understand the challenge of running an HTML5 browser in the cloud, is to consider the requirements from such a solution:

- The picture quality should be HD (720p or 1080p).
- The overall latency should ideally be under 150 milliseconds and not more than 300 milliseconds or otherwise end-users won't consider the experience responsive.
- Full HTML5 support is needed including the <video> tag, media source API extensions (MSE), and encrypted media extensions (EME). Overlays on top of videos should be fully supported as well.
- The cloud infrastructure (and associated costs) needed to deliver this experience in a cost-effective manner i.e. it should be in line with the typical web video streaming infrastructure. More specifically, the solution should not decode and then re-encode web videos in order to support overlays, since this would result in significantly higher infrastructure costs and potential video quality issues.
- Bandwidth should be similar to the bandwidth consumed by the streamed app.

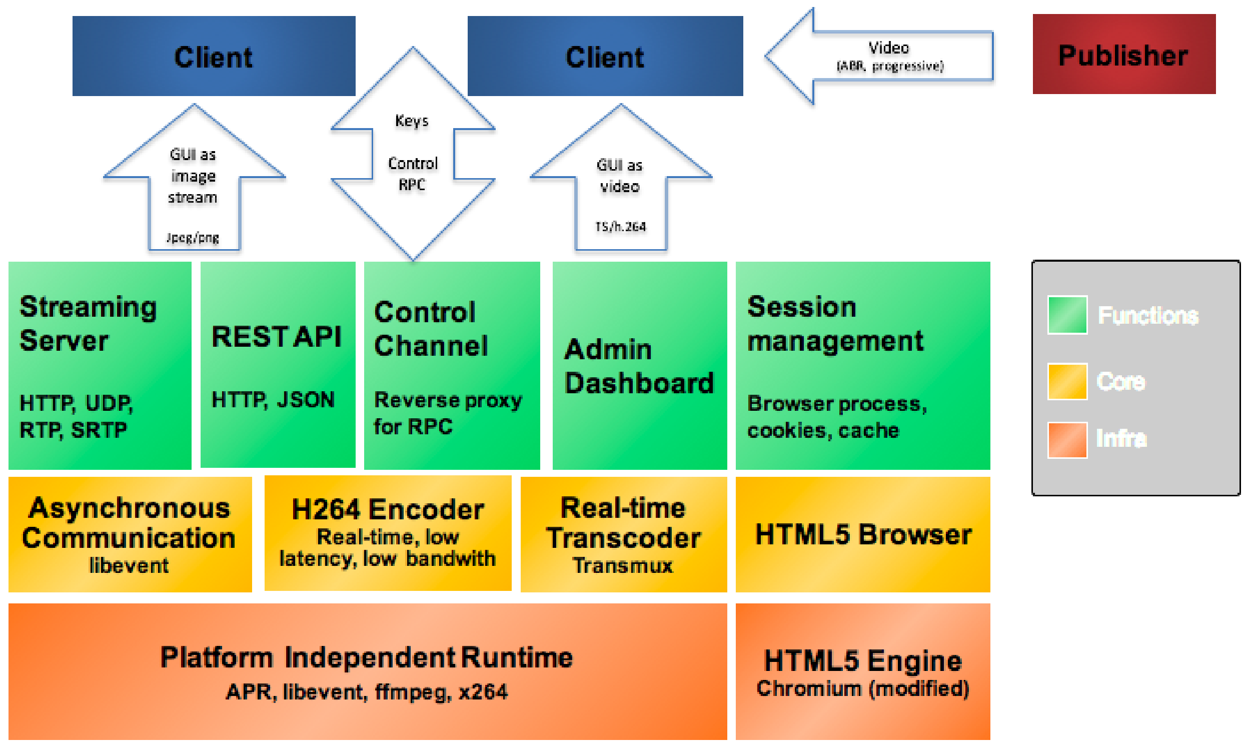
AppFlinger - The TVersity Solution

The TVersity HTML5 cloud browser is comprised of the following subsystems:

- HTML5 user interface delivery:
    - Ultra low latency via UDP streaming and highly optimized H.264 encoding.
    - Resource consumption is directly correlated to the extent of change on screen, with zero CPU and zero bandwidth when nothing changes on screen.
    - Up to 1000 active sessions per single rack unit for the youtube.com/tv workload.
- Video delivery:
    - Real-time transmuxing of the original video (typically H.264+AAC in MP4 container) to MPEG2-TS.
    - Streaming via UDP for QAM support.
    - Trick play support.
    - The same economics as a streaming server used on the web today i.e. few thousand sessions per rack unit.
- Control, pairing and Input injection:
    - REST/JSON services for injecting remote control keys.
    - Control via mobile apps or device's own remote control unit.
    - For IP enabled client devices:
        - Automatic pairing, discovery and launch via DIAL.
        - Automatic pairing, discovery and up to the second multi-screen synchronization via AirFlinger.
- Provisioning and administration:
    - REST/JSON services for provisioning and session management.
    - Web based admin dashboard.

Note that the delivery of user interface stream and the video stream typically do not need to exist simultaneously and can in fact be alternating, however for user interface overlays on top of a playing video, both need to exist at the same time and are rendered on top of the other by the client device.

The Architecture



The cloud browser is composed of the following main components:

- **H.264 encoder** - optimized for ultra fast, low latency, low bandwidth, real-time streaming of the browser window over UDP (or TCP/HTTP/RTP/SRTP/DASH and more). The video is variable frame rate such that no encoding is performed and no bits are streamed when nothing changes on screen, resulting in a more scalable and cost-effective solution.
- **HTML5 browser** - optimized for off-screen rendering of the HTML and streaming (instead of playing) of web videos. The browser is based on Google's Chromium and therefore offers state of the art performance and compliance with HTML5.
- **Transmuxer** (optional) - optimized for real-time transmuxing of various video formats (and in particular MP4) to MPEG2-TS, while streaming the output over UDP (or TCP, HTTP and more). The transmuxer offers full support for many streaming protocols and for trick play delivering the kind of experience consumers expect on the big screen.
- **Asynchronous HTTP Server** - provides REST/JSON API for provisioning, administration and input injection.

The diagram above also shows the interaction between the cloud browser and client devices. Up to two video streams may exist at any given point in time:

1. The content of the browser window (user interface) is rendered in memory by the HTML5 browser, encoded to H.264 and streamed to the TV with very low latency.
2. A web video (when played) is optionally transmuxed and streamed to the TV as a separate stream either directly from the publisher’s CDN or from the cloud transmuxer.

The TV or STB receiving those two streams needs to alternate between them and either overlay them one on top of the other (when overlays are required) or utilize an image stream for overlays. Remote control input can be posted to the server via a REST/JSON API. Additionally mobile apps can use the DIAL protocol to discover the TV and launch their HTML5 counterpart app for a multi-screen experience (effectively automating the pairing process). Alternatively the AirFlinger mobile SDK can be used for automatic pairing, and up to the second synchronization between the mobile app and the HTML5 app.

Finally, a web based dashboard is available for administrative purposes.

New Sources of Revenues

The smart TV industry has long realized the potential in a TV based app store. Similarly pay TV operators and service providers have been looking to establish new sources of revenues based on the app store paradigm. Unfortunately, none of them have been able to deliver a solution compelling enough for both consumers and app developers, resulting in spectacular failures.

HTML5 may be the key to solving this mystery; With its well-known characteristics, cross-platform appeal and unparalleled simplicity and openness, app developers can, for the first time, create a single TV app that targets eyeballs across all devices.

However HTML5 is not enough; The creation of a synchronized mobile and TV experience via a TV App and a counterpart mobile app that work together to deliver an experience better than each can deliver alone, will finally allow a flourishing ecosystem of TV Apps to emerge.

The utilization of the cloud means that hundreds of millions of TV devices deployed in living rooms worldwide can all become part of this app-ecosystem distribution.

The combination of a best of breed app technology, with massive distribution, is known, based on experience in the mobile space, to unleash an unparalleled wave of innovation that can change TV forever and create significant opportunities for app ecosystem gatekeepers and app publishers as one.

It is ultimately up to the existing living room dominant players to usurp this opportunity, leverage their current leadership to maintain their position in the TV market of the future, and expand their business in the process - it is also theirs to lose.

Summary

The adoption of the cloud along with web technologies hold the key for the TV of tomorrow. Addressing the hardware incompatibility challenge, set-top box challenge and the TV app ecosystem challenge will change TV forever.

The TVersity cloud browser can deliver today the TV of tomorrow, enabling smart TV vendors and pay TV operators to provide a best of breed experience with minimum hardware investments.

New revenue models can be created via an app aggregation model, future proofing the business of TV against OTT erosion.

Most importantly, the deployment of a cloud based HTML5 browser will allow the Internet's pace of innovation to be introduced to the world of TV - making this a highly strategic technology for pay-TV operators and smart TV vendors worldwide.

Finally, with TVersity's unique solution, all this can be achieved in a highly cost effective way.

About TVersity

Established in 2005, with headquarters in NYC and R&D in Israel, TVersity offers a comprehensive software platform and a suite of solutions for operators, service providers and smart TV vendors that solves the PC to TV, Mobile to TV and Web to TV problems, utilizing set-top boxes, connected TVs and mobile devices.