Content type: Opinion
Not peer reviewed
Innovations in Ubicomp Products

Pervasive Interaction Across Displays

Lars Lischke and Dominik Weber, University of Stuttgart

Scott Greenwald, MIT Media Lab

Digital screens are becoming more and more ubiquitous. Resolution and size are increasing, and, at the same time, prices for displays are falling. Large display installations are increasingly appearing in public spaces as well as in home and office environments. We expect this trend to continue, making wall-size displays commonplace in the next decade. With this development, all three classes of devices described by Mark Weiser—pads, tabs, and boards[1]—will be mainstream.

Pads (tablets), tabs (smartphones), and boards (displays) let us show and interact with data in different situations, because each device class is optimized for a certain use case. Consequently, the use of multiple devices becomes common—for example, the use of second screens while watching TV is becoming the norm. However, the use of multiple devices requires seamless transitions between devices, mechanisms for exchanging data, and the ability to move content from one device to another and to remotely access or control the data.

Back in 1998, Michael Beigle and his colleagues proposed dynamically and automatically distributing Web-based content to different output devices in a smart environment.[2] A few years later, Roy Want and his colleagues suggested using interfaces in our environment to interact with our personal data.[3] Because mobile devices or notebooks often provide only a small screen for output and limited input techniques, they proposed using office screens or public displays to create a more enjoyable user experience. They also argued for having physical access to private data. These examples highlight that research in ubiquitous computing was already early on exploring interaction across pervasive devices, displays, and content.

Current products support both visions. On one hand, there are devices that provide options to present remote data on a screen in the environment with the control residing on the mobile device. On the other hand, there are means to easily present content from mobile devices on remote displays.

There are now also many cloud-based products for interacting with data on multiple devices. For example, Dropbox provides access to all text documents and images. Spotify lets you enjoy your favorite music on smartphones, tablets, notebooks, and music systems. Furthermore, people are starting to use mobile devices as remote controls for large screens,[4] smart TVs, or music systems.

All these examples show that streaming and connecting different devices ubiquitously are key technologies for smart environments. Here, we present a few commercially available technologies supporting this and provide an outlook on how displays might become a service themselves.

Streaming Content on TVs

Although smart TVs have been available for a while now, there is no common dominant platform like on mobile devices and traditional PCs. Instead, manufacturers use a broad range of platforms, from modified versions of Android, to specific operating systems such as WebOS or Tizen, to custom implementations. The drawback of this fragmentation of platforms is that there is no common application repository (like the App Store), and the overall user experience is still poor. Often, apps are only available for some TVs, and the functionality is limited compared to similar apps on mobile phones or tablets.

A range of third-party devices that bring their own computer and only use the screen are benefiting from this situation and are becoming increasingly popular. The main contenders that we discuss here are the AppleTV, Amazon Fire TV, and Google's Chromecast (see Figure 1).

Figure 1. Computer systems that connect to any TV with HMDI input: (a) AppleTV, (b) Amazon Fire TV, and (c) Google Chromecast.

Apple TV

Apple TV is a set-top box developed by Apple that lets users display content on their TV with minimal setup. The device is connected via HDMI to the TV and receives content via Wi-Fi or Ethernet. The most popular streaming apps are pre-installed on the Apple TV and can be controlled using an Apple TV remote. Furthermore, Apple's wireless protocol AirPlay lets users select content on their iPhone, iPads, and Macs to be streamed to the Apple TV. Another option is to mirror the sender's display on the TV.

Fire TV

With the Fire TV, Amazon offers a product similar to Apple TV. Based on Android, the functionality of the Fire TV can be extended by downloading apps from the Amazon App Store. The device can also be controlled using a remote with a microphone for voice commands that let users search content and use basic playback controls.

Although there is a USB port for connecting external storage devices, the Fire TV is mostly focused on streaming media from the cloud. This has been further emphasized by the Fire TV Stick, a version of the Fire TV that has been reduced to the bare minimum. It also connects via HDMI to the TV, but instead of using an HDMI cable, the small form factor allows it to be plugged right into the TV. The only other port on the Fire TV Stick is a micro USB port to supply the device with power. All content is streamed to the device via Wi-Fi.

Chromecast

An even more simplified approach is used by Google's Chromecast. The device has the same form factor as the Fire TV Stick and purely acts as a receiver. When plugged into the TV, the device only displays that it is ready to receive content. Users can use their mobile devices to initiate playback directly from apps that support the Chromecast or from the Google Chrome Web browser on their desktop computer. One example is the YouTube app; when using Chromecast, the content is shown on the TV screen, and the mobile device becomes a remote control with a familiar user interface. The content to be shown is directly streamed from the cloud to the Chromecast, and the user's other devices are only used to select the content and control the playback. Additionally, it is possible to stream the whole desktop of the connected computer to Chromecast.

Distributed Interfaces

All mentioned devices extend the functionality of TVs, and a common aspect is that they do not actually store content on the device itself. Instead, either the content to be displayed on the TV is streamed from the cloud or from other devices, which (to some extent) are also used to control the playback.

For developing distributed interfaces, a fast starting point is to use Web sockets. For example, the Node.js Web socket implementation Socket.IO lets multiple clients quickly exchange information with a few lines of code. Thus, it is easy to prototype remote controls for smart displays.

Figure 2. The devices support two basic concepts: (a) mirroring of the screen (same content on mobile and screen) and (b) applications that are aware of the external screen (content on screen, control on mobile).

Promising New Approaches

Overall, the streaming devices we reviewed present convincing approaches for connecting displays and exploiting available infrastructure in the environment. At the moment, all consumer devices focus on streaming multimedia content from mobile devices or computers to additional screens. However, applications such as the Android Auto (www.android.com/auto) and Display as a Service (www.daas.tv) show the great potential for streaming approaches to build services using input and output devices in the environment.

Android Auto

Device interactions while driving must take a back seat to the driver's most important attentional load—maintaining safety on the road (at least until autonomously driving cars become mainstream). Applications for the automobile thus must require minimal visual attention and should offer speech input and audio output whenever possible. They should not distract the driver in a way that would compromise safety.

One infotainment platform that is being adopted in the auto industry is Android Auto. A core architectural feature of this platform is that applications and services run on the user's mobile phone. The current driver connects his or her phone to the car, and the display show messages while the stereo plays his or her favorite music. The car environment becomes instantly customized to the current driver, regardless of whether that person owns the vehicle. In addition, the

phone can use the car's high-quality hardware, including speakers, microphones, displays, control panels, and car sensors. Furthermore, upgrading the car's infotainment system is as easy as buying a new phone.

From the perspective of the user experience, Android Auto aims to present cross-platform content, adapted to suit the driving environment: it uses the car's touchscreen, microphone, and speakers for input and output; applies fixed visual templates for easy test legibility; can read content aloud; and can apply voice-recognition for text input. In terms of the app developer's perspective, positioning the tuning of information presentation as a problem of the platform rather than the developer is advantageous, because it means that app developers need not master a plethora of specialized knowledge about the car user experience. Finally, from a regulatory perspective, giving the platform control over information presentation makes it easier to ensure safety. If the software framework is the gatekeeper of all device-related attentional loads on the driver, and it's responsible for preventing disruptive notifications from being delivered, then only the framework needs to be regulated and not every individual app.

Comparable solutions to Android Auto are Apple's CarPlay and MirrorLink, invented by the Car Connectivity Consortium (http://carconnectivity.org). CarPlay's concepts work similar to Google's solution—that is, the user just connects the smartphone to the car using a USB cable. Then, the user can access applications running on the phone through the car's hardware.

Display as a Service

Connecting multiple displays or projectors to one larger or wall-sized screen is challenging. Specialized graphics hardware is often needed to drive all displays. Furthermore, keeping all signals synchronous is a complex job. Display as a Service uses frame-buffer and display virtualization.[5] Because of the software abstraction, multidisplay environments can be configured without specialized hardware. One or multiple render servers write into one or multiple virtual framebuffers. Every virtual display receives all pixel data of a traditional IP-network. Consequently, a virtual display can consists of one or multiple displays. In contrast to classical screens, every single display unit must be connectable over a network to receive the dedicated pixel data.

From a technical viewpoint, every smart TV, smartphone, tablet, or display or projector equipped with a small computer is suitable as part of a multidisplay setup. This streaming technology lets users easily connect heterogeneous form factors of devices to display content as one system. For future development, we see approaches like Display as a Service as key to building wall-size displays. Furthermore, such approaches will provide the opportunity to include commonly used mobile devices, such as tablets and smartphones, for interaction.

Using larger screens, high-quality speakers, or regular-size keyboards found in user's environment, along with applications running on mobile devices, can greatly enhance the user experience. We thus see streaming as a key element for future ubicomp products.

Acknowledgment

This work is partially funded by the European Community's H2020 Program under the funding scheme "FETPROACT-1-2014: Global Systems Science (GSS)," grant agreement # 641191 CIMPLEX.

References

1. M. Weiser, "The Computer for the 21st Century," *Scientific American*, vol. 265, no. 3, 1991, pp. 94–104.
2. M. Beigl et al., "The UbicompBrowser," *Proc. 4th ERCIM Workshop on User Interfaces for All*, 1998, pp. 51–86.
3. R. Want et al., "The Personal Server: Changing the Way We Think about Ubiquitous Computing," *Proc. 4th Int'l Conf. Ubiquitous Computing*, 2002, pp. 194–209.
4. O. Chapuis, A. Bezerianos, and S. Frantzeskakis, "Smarties: An Input System for Wall Display Development," *Proc. 32nd Ann. ACM Conf. Human Factors in Computing Systems* (CHI 14), 2014, pp. 2763–2772.
5. A. Löffler et al., "Networked Displays for VR Applications: Display as a Service," *Proc. Int'l Conf. Artificial Reality and Teleexistence/Eurographics Workshop on Virtual Environments/EuroVR*, 2012, pp. 37–44.

Lars Lischke *is a PhD student at the Institute for Visualization and Interactive Systems at the University of Stuttgart. Contact him at lars.lischke@vis.uni-stuttgart.de.*

Dominik Weber *is a PhD student at the Institute for Visualization and Interactive Systems at the University of Stuttgart. Contact him at dominik.weber@vis.uni-stuttgart.de.*

Scott Greenwald *is a PhD student at the MIT Media Lab. Contact him at scottgwald@media.mit.edu.*

Streaming and connecting different devices ubiquitously are key technologies for smart environments. Here, the authors present a few commercially available technologies supporting this and provide an outlook on how displays might become a service themselves.

pervasive computing, TVs, Internet/Web technologies, mobile, AppleTV, Fire TV, Chromecast, Android Auto, Display as a Service

Edited version: L. Lischke, D. Weber and S. Greenwald, "Pervasive Interaction Across Displays," in IEEE Pervasive Computing, *vol. 14, no. 4, pp. 12-15, Oct.-Dec. 2015.*
doi: 10.1109/MPRV.2015.77
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7310813&isnumber=7310795