Wireless Display Adapter Using Raspberry Pi

casting screen wirelessly and using it as a remote-control for media displayed

Caihua Li
Department of Computer Science
Peking, China
peterli@pku.edu.cn

Zikai Zhou
Department of Computer Science
Peking, China
zk.spica@qq.com

Siyu Yao
Department of Computer Science
Peking, China
syyao@pku.edu.cn

ABSTRACT

Peripheral devices for computer emerged in recent years, but the wires connecting them greatly limit user experience. What we did is to remove the cable from computer to a remote display device, which is a compromise solution to a big plan that to remove the cable of VR headset. We used a Raspberry Pi to set an access point at a normal offline display device to get connected. Based on this, we developed 4 functions: local video player, online video player, screen mirroring and offline electronic album.

Keywords

802.11; Wireless Communication; Leap Motion; Virtual Reality

1. INTRODUCTION

1.1 Background

In recent years, some cutting-edge concept like Virtual Reality has brought us freshly new prospect of our human-computer interface, games, etc. But on the way of freeing people into this new world, one of the knottiest problems is about its requirement of huge amount of data and incredibly high transmission rate. Thus, many peripheral devices use wired transmission, USB or HDMI cables in most cases. However, wires are limiting user experience. We aim to replace wires with wireless transmission without unacceptable harm to transmission quality. Originally, the cables associated with VR headsets or products for hand gesture capture, Leap Motion, are among our targets, but unfortunately, we somehow failed to find a physical interface for a well-encapsulated VR set and an appropriate compatible architecture for Leap Motion to reconstruct. Indeed, we have tried everything we can, including sending an email to Leap Motion producer to check if there is compatible architecture, but the result was not optimistic.

Considering that the core theory inside our original target lies in control over wireless support, we adapt our idea into some more accessible devices, normal display devices. We completed a remote display adapter, by which any normal display device can update to be a remote display device as long as it has a USB interface.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

WOODSTOCK '97 El Paso, Texas USA

© 2017 ACM. ISBN 123-4567-24-567/08/06…$15.00

DOI: 10.475/123_4

1.2 Goal

Pragmatic What we aim to do is something useful and pragmatic, which prohibits unaffordable device updates. An adapter plugged into USB interface is all you need to make a display into an online device that can communicate with other members, fetch resources from Internet and etc.

Cross Platform We do not want to see our adapter exclusively accessible for users of some systems or companies. To reach this goal, we have to do things at a relatively high level, since some companies forbids access to changes at lower levels.

2. RELATED WORK

To be honest, remote display is anything but new or state-of-the-art. Indeed, there is a variety of competing wireless standards and protocols available.

Miracast

A standard for wireless display launched by Wi- Fi Alliance at the end of 2012. Devices that are Miracast-certified can communicate with each other, regardless of manufacturer. It employs Wi-Fi Direct standard, a technology that enables two devices to form a direct, peer-to-peer Wi-Fi connection so that they can automatically discover and connect without a wireless router. The connections are created via WPS and thus secured by WPA2. On the transport layer, TCP or UDP are used. On the application layer, the stream is initiated and controlled via RTSP, RTP for the data transfer.

One big problem concerning Miracast is that Miracast could drain your device's battery life while mirroring. It's strictly a screen-mirroring protocol and therefore not smart enough to hand over the streaming. Miracast will not allow for multitasking so that you can neither use your mobile device for other purposes while mirroring nor let your device go to sleep.

Chromecast

Chromecast is a small dongle by Google that plugs into your television's HDMI port and uses the Discover-And-Launch protocol. With this setup, you can open an app on your Android or iOS device and command the app to stream video to your Chromecast.

Chromecast, which requires an internet connection from a local wireless router, will then display the video on your television, but while that content streams from your device to the television, you're still free to use your device for other purposes and can even control playback of the stream from that device.Chromecast also has a feature that lets you mirror your desktop onto your television's display.

Airplay

Basically, Airplay can do everything you expect of a remote display. It can display your Mac's desktop on a television, for instance. It can also take video playing in an iPhone app and stream it to your television, while letting you use playback controls on your iPhone, and it can even mirror your iPad's screen onto your television. The performance is said to surpass Chromecast and any other Miracast-certified device: the AirPort Express' streaming use Apple's Remote Audio Output Protocol (RAOP), a proprietary variant of RTSP/RTP. Using WDS-bridging and the AirPort Express can allow AirPlay functionality across a larger distance in a mixed environment and up to 10 wireless clients. But the exact composition of the protocols that AirPlay uses have not yet fully been discovered or reverse-engineered.

Our work can be best described a imitation of Google's Chromecast with one difference that instead of sharing a local internet connection, the display itself becomes a new access point, which is a compromise of static IP address problem but also brings new advantages like convenience for users.

3. SYSTEM DESIGN

Hardware

Display devices act as service providers in our design, so we expect it to be a server so that all devices, including users' laptops and cell phones, can connect to it right away. And here comes the first obstacle: within some ESS such as *Wireless PKU*, it is impossible to get a static IP address, because all the IP addresses for log-in users are dynamically assigned. And such situation is not rare, people connected to other ESS like airpot wi-fi service face the same problem. To solve this, we decide to enable the display device to be an access point for wi-fi connections, to which all electronic devices can connect and communicate.

We implemented our idea using *Raspberry Pi*, which can be seen as a mini computer supporting all basic functions and possessing all basic physical interfaces. We used 3 parts of the *Raspberry Pi* in total: 2 NIC (one built-in and one external NIC), USB interface, video player. One NIC is for the *Raspberry Pi* to connect to Internet, while the other is used to set access point.

Software

We developed 4 functions for our wireless display adapter covering almost all functions for any display devices.

Local Video Player

Users can play local videos stored in their own laptop or mobile devices.

- Client sets up a *FTP* server program, where stores the video resource.

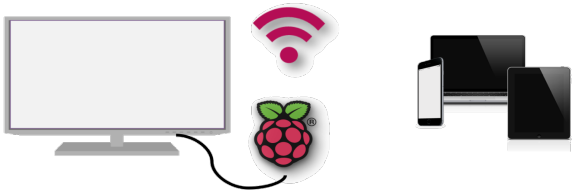

Figure 1: Hardware Implementation

- Client sends a request to give the command of video player
- Raspberry Pi, becoming client at this time, starts a *FTP* request for video resource on the *FTP* server

Things are not as simple as it seems. For instance, we have met the problem that system firewall against resource fetching from outside client.

Online Video Player

- Client sends a URL of a video
- RPI starts a HTTP request for online video source
- A browser plug-in to extract video and control access

Screen Mirroring

- Client launches screen shots at about 40 frames per second
- Client sets up a server program for screen mirroring
- Client sends a request to command of screen mirroring
- Raspberry Pi, at this moment becomes client side and starts a request for transmission of screen shots pictures
- Raspberry Pi draws every frames onto the screen

Offline electronic album

- An eternal file store images desired for display while readiness
- Client sends a request to command wallpaper update
- After connection closes, RIP starts display wallpapers.

4. EVALUATION

We tested 2 aspects of our demo remote display device: the bitrate of video it can support and the delay of screen mirroring.

Bitrate

Under the testing environment of bandwidth around 8MBps, in an environment within 10m without any occulusion, videos with bitrate below 5000kbps are well supported with perfect smoothness. But the performance of videos at about 8000 kbps is not that desirable with video suspended from time to time. And video became too slow to watch after we raise the bitrate to 10000kbps.

Delay

Under the testing environment of bandwidth around 8MBps, in an environment within 10m without any occulusion, the latency of casting a frame from a device to a display is about 3.12 seconds on average, most of which is spent on picture processing and drawing. The major reason for this latency lies in the speed limit of picture draw on *Raspberry Pi*. We have applied multi-thread programming, so that the picture processing and sending at user side is accelerated by 23%, but the bottleneck still lies in the processing speed of *Raspberry Pi*.

5. FUTURE WORK

As we have mentioned, this project is just the first try or a first taste of our real goal. In the future, probably the coming semester, we will look into the physical structure and transmission protocol of VR headset. Besides, to fulfill the challenging task of VR enormous headset transmission rate, we will get to know more about 5G, only through which, from our point of view, can we support the transmission throughput of a VR headset.

6. CONCLUSION

Though embarrassing, it is hard to say that we have come to any conclusion yet. But at least, we have gained some experience in wirelessing originally wired peripheral devices and Raspberry Pi developing skills. In retrospect, we have realized that what we learned at class during this semester's computer networks is far from sufficient to fulfill the task of wirelessing VR headsets. There are new technologies and protocols like 5G arising every day, we should apply the principles we learned to these new things so that we can make good use of them to do something even greater like we stated in our proposal.

7. CONTRIBUTION

All of our members have devoted a lot of time and energy on this project, especially when we were facing challenges at the first stage dealing with Leap Motion and VR headset. And when we were forced to change our project, although it is a really hard decision, we cooperated to do new researches quickly.

Li Caihua %40

- Hardware research about Raspberry Pi
- Setting up access point at display device
- Code for server end
- Research for Leap Motion SDK in original project

Zhou Zikai %30

- Code for client end interface
- Code for 2 applications: local video player and screen mirroring
- Research about VR headsets in original project

Yao Siyu %30

- Research about HDMI protocol and USB protocol
- Code for 2 applications: online video player and offline electronic album
- OS X system compatibility adjustment
- Poster, presentation, report writing and design
- Research on Leap Motion API in original project