1672 E 117th
Cleveland, OH 44106

Jonathan Weedon
Department of English, CWRU
Guilford House
11112 Bellflower Road
Cleveland, Ohio 44106

25 November 2014

Dear Jonathan Weedon:

You will find attached a proposal for a software development project entitled “Low-Latency Game Streaming for AMD Platforms”. This proposal is to be presented to software investors or software engineering companies. This proposal was developed as part of a requirement for ENGL 398 and is being presented to you for evaluation. The proposal looks at several competing technologies and argues for a better solution for gaming on AMD platforms.

This proposal is written for the aforementioned audience and thus uses some technical terms that are common for the field. The proposal contains background information on the subject matter and explains some alternative solutions in simple terms. The background section contains sufficient information about currently available technology and the need for an alternative. It also provides some insight about the desire for such an application, adding to the need for this project. The following sections include more technical information about how the project would be carried out and what technologies would be utilized to make game streaming possible for AMD platforms.

I would like to acknowledge Professors Michael Rabinovich and Gultekin Oszoyoglu, both in the Electrical Engineering and Computer Science department at Case Western Reserve University for their expertise in software engineering and networking. This project would not be possible without their help.

I hope you enjoy the following proposal and if there are any technical terms you feel pertinent to the understanding of the proposal, please do not hesitate to ask for clarification.

Sincerely,

Diego Waxemberg

Low-latency Game Streaming for AMD Platforms

Submitted to Scott Weedon
Submitted By Diego Waxemberg
25 November 2014

Abstract

There is a growing desire for a streaming application that offers low-latency and high performance. This desire has partially been met by some products, however there is still a large market that does not have access to these products. By creating a low-latency streaming application that works with AMD platforms many of these people would be able to utilize this technology. AMD graphics cards have the necessary hardware to encode video and audio into a format that is small enough to be sent over the internet, but still contain the quality necessary for gaming. This proposal discusses the necessary steps in order to create a server and client pair application that harnesses the potential of AMD's platform and makes game streaming possible for consumers who have AMD graphics cards. The application will consist of several protocols: an authentication protocol, video streaming, audio streaming, paring, input, and others to ensure the stream is low-latency, high quality, and secure.

I. Project Description

The need for a low-latency game streaming application is clearly evident from the current market trends, however a solution does not exist for those who choose to use AMD's hardware. This project aims to provide AMD users with the ability to stream their full collection of resource intensive games from their desktop to their mobile devices. There are various solutions to this problem, however they all force the user to make some sort of compromise. Some manufacturers make versions of the games that have less features, and different gameplay in order to reduce the resources required and allow the game to be played on a mobile device. Another approach is to introduce a delay between what the player sees and what is actually happening. This is very common in current implementations of remote streaming such as VNC, or Remote Desktop. The increased latency causes the gameplay to be suboptimal and reduces the user's competitive edge. These issues can all be remedied by creating a system that allows users to take their high performance gaming experience anywhere they want.

II. Background and Current Products

A. Background

For a long time now, people who want to play high-end games require a high-end desktop. There are many reasons for this requirement, but what it comes down to is performance. High-end games require an abundance of resources in terms of processing power, graphic rendering, storage space, etc. This requirement makes it very difficult to take a high-end game and give it the portability of a more basic game. Many game manufacturers have created simplified versions of their games that will run on less powerful platforms such as mobile devices and gaming consoles. However, these simplified games are often times not as fun or immersive as their full-featured counterparts. What if we were to harness to power of the desktop to do all the heavy-lifting of playing a game, but use a mobile or less powerful device to actually play the game?

What I am proposing is a method of remote game streaming. Use a powerful gaming desktop to perform all of the physics calculations, graphics renderings and other resource intensive aspects of a game, then take the final result (ie. the image frame and audio) and quickly send it to a less powerful device that presents it to the player. The mobile device would simply need to be able to display an image, render some audio, and receive input from the user. All of these are fairly simple to accomplish and have been done numerous times in mobile applications and games.

Remote game streaming as described above has already been attempted, and commercialized, however there is room for improvement and plenty of market to capitalize on. The most viable alternative was released by NVIDIA last year and has very promising performance, as well as market penetration. Although NVIDIA's GameStream works quite well, there are a few major setbacks that could be resolved in order to produce a better product and reach a larger share of the gaming community. The commercialization of a better product would be immensely profitable and push the bounds of current technology.

B. Current Products

There are several products on the market that perform streaming from a desktop to a less powerful client, however the only one that has the performance required for playing games is NVIDIA's GameStream. NVIDIA's GameStream is able to produce a higher quality stream with less latency and more frames per second than the nearest competitors. This advantage is due to the hardware requirement of GameStream. The only desktops capable of running GameStream are those that have a Kepler Graphics Processing Unit (GPU). Kepler is one of NVIDIA's highest-end and most expensive line of GPUs and includes GTX

650 and above graphics cards. This requirement is due to GameStream's use of hardware video encoding. The only graphics cards that NVIDIA sells with an onboard hardware H.264 video encoder are Kepler.

The GPU requirement imposed by GameStream makes the technology less reachable than it could be, and that is not the only limiting requirement. GameStream will only stream to select NVIDIA SHIELD devices. Currently, there are two SHIELD devices: the SHIELD Tablet and SHIELD Portable both of which cost hundreds of dollars. Forcing gamers to have to purchase specialized hardware that is comparable to hardware they may already own makes this technology less reachable and limits its potential.

C. Possible Alternative

The current technology has many limitations and requirements, but surely this must be a necessity. Actually, it's not. Requiring gamers to purchase an NVIDIA Shield device is purely a marketing tactic. There is nothing intrinsically better about these devices that make them capable of GameStream. These devices run Google's Android operating system making the GameStream application viable for any other Android device. Android is responsible for over 50% of the current mobile market share making it much more accessible than the SHIELD devices. A lot of gamers already own an Android device and would not need to purchase any additional hardware to use GameStream. The mobile market share also attributes over 40% to Apple's iOS. As there is no particular requirement for the Android operating system, there is no reason this technology couldn't be brought to iOS and even other mobile operating systems such as Windows Phone 8.

Although NVIDIA does not allow users to run GameStream on their own devices, there is nothing preventing another application from doing the same thing as NVIDIA's without the need for specialized hardware. This alternative application already exists in the form of Limelight and has had a significant impact on the gaming community surrounding NVIDIA's GameStream. Limelight is an open-source client for NVIDIA's GameStream that runs on Android, OS X, Windows, Linux and has promise of iOS and Windows Phone 8. Limelight shows that there is no need for NVIDIA's specialized SHIELD products, however SHIELD is not the only requirement of GameStream.

GameStream requires a Kepler based GPU in order to perform the low-latency hardware video encoding. Although it appears there is no way around the hardware encoding to produce a low-latency high-quality video stream, there is an alternative to NVIDIA's GPUs. AMD has a similar hardware encoder in their HD 7700 and above graphics cards. AMD and NVIDIA are the two largest producers of consumer GPUs and the ability to stream games from an AMD graphics card would have tremendous impact in the market. AMD's Video Codec Engine (VCE) is able to produce a 60 frame per second, 1080p, H. 264 encoded video faster than real-time. This is a comparable performance to NVIDIA's Kepler GPUs and would provide a similar streaming experience.

AMDs graphics cards not only have the capability of game streaming, but AMD has listed remote game streaming as an intended use-case for VCE. Surprisingly, no application has taken advantage of this technology. A quick browse through NVIDIA's Geforce forums shows that many gamers wish to have the same technology brought to AMD's platform. By creating an application that supports AMD graphics cards, many gamers would no longer need to purchase new, almost identical hardware to get the same benefits.

NVIDIA has paved the way for remote game streaming, but added an asterisk to all of its benefits. Some of these asterisks can be removed by incorporating a larger array of streaming clients as well as hosts. Many gamers prefer to use AMD's graphics cards and therefore do not have access to GameStream. The financial burden of

purchasing a new graphics card and a SHIELD device is often too great to be feasible for some gamers. By creating and marketing an application that will use available technology to bring the same gaming experience to the AMD platform, a large number of gamers will be able to enjoy the benefits. Limelight has already shown that streaming to non-specialized hardware is possible which bodes well for an application that aims to capture as much of the gaming community as possible. By creating and commercializing a remote game streaming application, one could take advantage of this emerging market.

III. Methodology

This application will actually be two applications working in tandem. On the desktop, there will be a server application running and on the mobile device, the client application. The two applications will have a protocol for communicating and setting up and tearing down the stream. The server application will be written to run on the Windows operating system, while the client will be flexible and have support for Windows, OS X, Linux, iOS, Android, and Windows Phone.

Server

The server application will be able to run as a background process or a service both with a user-facing interface for configuring settings. This application will be responsible for all of the heavy-lifting for the system. The hardest component will be the video encoder. In order to encode a video frame, the server will use AMD's VCE to gain access to the hardware encoder, but first the server will need something to encode. The server will need to request a frame from the GPU's image buffer and send that to VCE to be encoded. VCE supports a large array of resolutions and frame-rates, so the user will be able to customize the stream to their liking, allowing for a better-looking or better-performing game. This will be very similar to the settings in most games that allow the user to change the way the game looks to optimize for their specific hardware.

VCE will then provide the server with an H.264 encoded image and the server will need to send this to the client. Fortunately, there is no more compression required because H.264 is already dense enough to be streamed. The way H.264 can be so small, is because it uses a system of deltas, or changes, from the previous frame to the current frame. This means each H.264 "frame" does not contain the data for the entire image. Along with the deltas, H.264 also has a system for requesting the entire frame. This will be used at the beginning of the stream, as well as periodically during the stream. If any of the images are lost due to network issues, the server will need to send a full frame in order to resynchronize with the client.

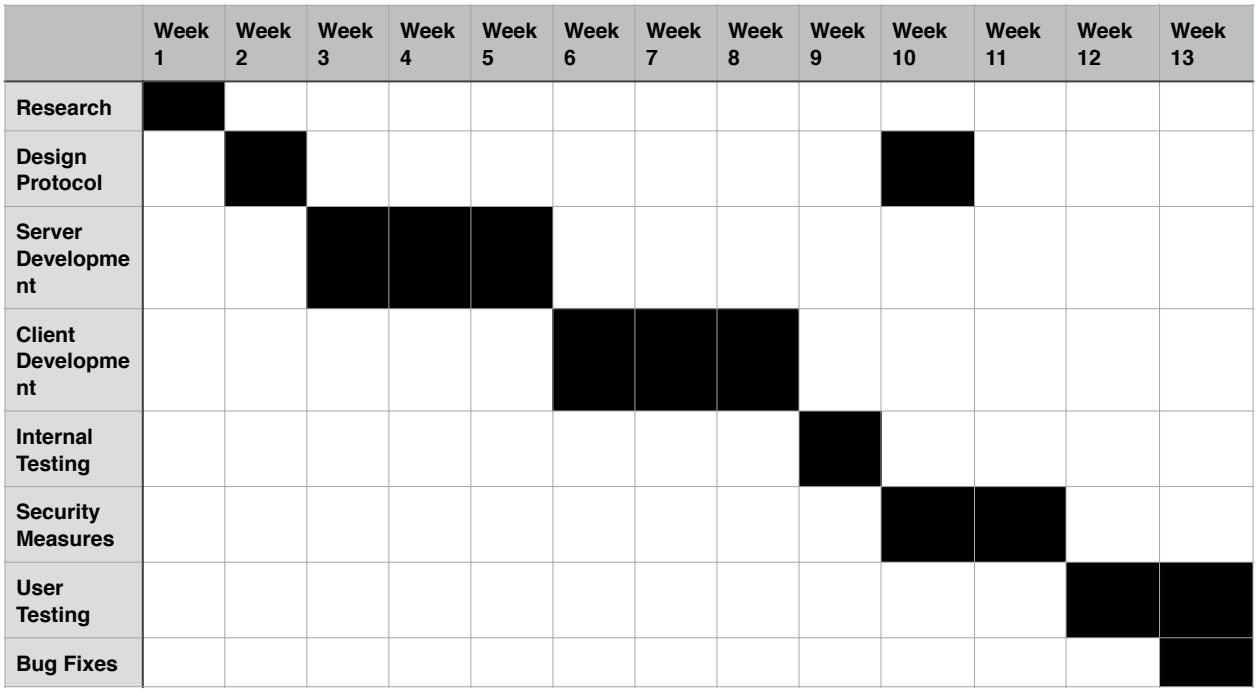
The next component of the server is the audio stream. The audio will also be provided by VCE as it has a system for emulating a sound card and is as simple as changing what output the operating system is set to. This can all be done just prior to beginning the stream and then restored to default as soon as the stream ends. The audio data will be 2 channel, 16-bit, PCM which is small enough to be streamed without compression. However, if compression is necessary there are many simple solutions such as Opus.

The final component of the server is the input handling. The client will send the user's input and that input will need to be injected into any running application and the operating system itself. This can all be achieved using Windows's built in drivers for keyboard and mouse. In order to support a gamepad the server will need to use XInput, which is also provided by Microsoft for this purpose.

Client

The clients will be fairly simple applications that receive an audio and video stream and provide them to the user. In order

Figure 1: Project Outline



to reduce latency, the client will utilize the H. 264 hardware decoders present in almost every modern device or ffmpeg for software decoding when that is not available. After decoding the video the client will render the image to the screen and the user will be able to see what is on the server's monitor.

The audio will be very easy for the client to handle, since it will not be compressed. The client will simply extract the data from the network packets and send it to the audio renderer. The renderer will handle sending the data to the sound card appropriately and the operating system will take it from there.

The user's input will be converted to a bitmap and then sent to the server. This will make handling the input very fast and effortless for the server since parsing the values will be trivial. The mobile platforms all have built-in support for gamepads and there are open-source libraries for the desktop operating systems.

In order to reduce the amount of duplicate code that would need to be written, the client for Windows, OS X, and Linux will be written in Java. Java is already a cross-platform optimized language so no additional work would need to be put in to make the clients work. Since Android also runs Java,

	Cost	Qty.	Total
Testers	$10	20	$200
Hardware	$200	10	$2000
Less Available Hardware	-$200	5	-$1000
Total:			$1200

there will be a shared library between Android and the desktop clients. This common library will contain all of the network protocol implementation minimizing the amount of client-specific code required.

Communication
The client and server will communicate using multiple network protocols and a few custom protocols. The initialization will be done using HTTP over TCP due to the ease of HTTP. This will provide a simple interface for the client and server to exchange credentials and authenticate as well as set up the stream. The actual stream will be done using custom protocols that use both TCP and UDP. Input will be handled using TCP due to its guaranteed delivery. This will ensure that the user's input will always be received even if the user has a poor connection. The audio and video will be sent using UDP due to TCP's large overhead. Since losing a few packets of audio or video is easy to recover from, it will not pose many performance issues. UDP also offers much higher transfer rates which will be key to provided a low-latency stream.

IV. Schedule

This system will be developed in phases starting with some research. After getting a firm understanding of all the required components, the communication protocols will need to be designed. Once a protocol is established the server and clients will be developed. After the system is operational, the security and authentication system will need to be implemented before allowing user-testing to begin. The final step will be to fix any bugs the users find and then package the software for commercialization. Figure 1 shows a gantt chart detailing the amount of time expected for each phase. The entire project should be completed within 13 weeks.

V. Qualifications

As a 4th year Computer Science student, I have had several experiences that give me insight into the requirements of this project. I have held multiple Software Engineering internship positions and am currently employed as a Software Engineer. I have worked on other large projects which have taught me the requirements of producing such an application as this. I have also worked with networking and video streaming in the past which will provide useful knowledge about what systems work and do not work together.

VI. Faculty Mentor

For this project I will consult with Gultekin Ozsoyoglu and Michael Rabinovich of the Electrical Engineering and Computer Science department. Dr. Ozsoyoglu has experience in software development and he will be very beneficial in the design of the application. He will also be able to help commercialize the system once it is finished. Dr. Rabinovich is the current Networks professor at Case Western Reserve University and will be knowledgeable on how to design an efficient protocol for the system. He will also be able to assist in the security of the protocol.

VII. Budget
The only monetary requirements of this project will be during the user testing phase. I will need to compensate all of the testers for their help and time. I would also need to purchase different hardware in order to ensure the system functions on all of the desired devices. A lot of hardware is already available and would not need to be purchased, so this cost would be fairly small. Figure 2 outlines these costs.

Figure 2: Budget

VIII. References

[1] *Advanced Video Coding for Generic Audiovisual Services.* [Online]. Available: http://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-H.264-200305-S!!PDF-E&type=items

[2] *AMD Unified Video Decoder (UVD).* [AMD White Paper]. Available: http://www.amd.com/Documents/UVD3_whitepaper.pdf

[3] *Introducing the NVIDIA SHIELD Tablet: The Ultimate Gaming Tablet.* [Online]. Available: http://shield.nvidia.com/gaming-tablet/

[4] *Introducing the Video Coding Engine (VCE) - AMD.* [Online]. Available: http://developer.amd.com/community/blog/2014/02/19/introducing-video-coding-engine-vce/

[5] *Limelight Game Streaming*. [Online]. Available: http://limelight-stream.com/

[6] *NVIDIA SHIELD User Guide: Get The Most From The Ultimate Gaming & Entertainment Portable.* [Online] Available: http://www.geforce.com/whats-new/guides/nvidia-shield-portable-user-guide#4

[7] *SHIELD Forum*. [Online]. Available: https://forums.geforce.com/default/board/111/nvidia-shield/

[8] *The Smartphone Reinvented Around You Windows Phone.* [Online]. Available: http://www.windowsphone.com/en-us

[9] Whitney, Lance. (2014, Sept 05). *Android Loses Some US Market Share but Remains Top Dog.* [Online]. Available: http://www.cnet.com/news/android-loses-some-us-market-share-but-remains-top-dog/

[10] *Opus Codec.* [Online]. Available: http://www.opus-codec.org/

[11] *FFmpeg* [Online]. Available: https://ffmpeg.org/