Remote Gaming on Resource-Constrained Devices

Waazim Reza, Hari Kalva, and Richard Kaufman
Multimedia Laboratory, Department of Computer Science and Engineering, Florida Atlantic University, FL, USA;

ABSTRACT

Games have become important applications on mobile devices. A mobile gaming approach known as remote gaming is being developed to support games on low cost mobile devices. In the remote gaming approach, the responsibility of rendering a game and advancing the game play is put on remote servers instead of the resource constrained mobile devices. The games rendered on the servers are encoded as video and streamed to mobile devices. Mobile devices gather user input and stream the commands back to the servers to advance game play. With this solution, mobile devices with video playback and network connectivity can become game consoles. In this paper we present the design and development of such a system and evaluate the performance and design considerations to maximize the end user gaming experience.

Keywords: Remote Gaming, video codec, performance, response time

1. INTRODUCTION

Games have evolved over time from simple two dimensional graphics games like Pong to intriguing three dimensional games like James Cameron's Avatar 3D which approaches real world actions. Game publishers and other service companies have used various tactics to bring your favorite games to your living room. There are game services available that provide "Games on Demand"[1]. They provide convenient, 24 x 7 accesses to a growing online library of popular game titles. With the click of the mouse, one can now browse and download full Games. With direct download, it's easy to switch quickly between games and there are no discs to worry about.

But as the market transitions from feature phones to smartphones, the dynamics of gameplay are also shifting towards a higher quality experience. As a result, one can expect to see a profound increase in adoption of this activity, both in terms of audience size and overall engagement. Smartphone gamers are also typically more avid gamers overall. According to ComScore 47.1 percent of smartphone users play games at least once a month on their devices; only 15.7 percent of non-smartphone users play games at the same rate [2]. However there has been a considerable restriction of the type of the games that a user can play on a mobile device inherently because of the device limitations such as memory, processing, and storage capabilities.

In this paper we present the design and development of a remote gaming solution to deliver games to low cost mobile devices. We studied the design alternatives and developed a remote gaming platform to study the games and application needs. We studied the impact of video encoding and game configuration on user experience. Bounds on response time are presented. These response time bounds are used to design remote games.

2. BACKGROUND

Video games are usually played on a dedicated "console," like Sony's PlayStation 2 or Microsoft's Xbox, connected to a television, or on a portable "hand-held" device, like Nintendo's Game Boy. Gaming industry has seen a proliferation of portable gaming devices by many gaming giants. The Sony PlayStation 2 has become a mainstay in the living rooms of video game players around the world. With the PSP (PlayStation Portable), Sony took its first step into the portable,

Applications of Digital Image Processing XXXIII, edited by Andrew G. Tescher,
Proc. of SPIE Vol. 7798, 77981Q · © 2010 SPIE · CCC code: 0277-786X/10/$18
doi: 10.1117/12.863016

handheld video game arena. The Game Boy gaming system from Nintendo comes in two flavors, with the SP version adding rechargeable lithium-ion batteries, a front-lit screen and a new design that features a flip-up panel. Both versions play older Game Boy and Game Boy Color cartridges. The PSP can't play PlayStation or PlayStation 2 games.
Finnish mobile telecommunications giant Nokia dove headfirst into the portable gaming arena with the N-Gage, a device that specializes in multi-player gaming. The N-Gage allows users to face off against other N-Gage owners via wireless Bluetooth technology or through Nokia's N-Gage Arena network. The Nintendo DS, features two screens, one of which is a touch-sensitive pad like you'd find on a laptop. The screens can also be used for innovative game design, showing different information on each of the screens.

However all the above technologies have some shortcomings:

- Requires high hard disk storage space to store the games or requires external cartridges.
- A limited number of games would be available for a particular brand of gaming device. For example, games for Nintendo will not available for a PSP user.
- Most of the gaming devices have a main central processing unit (CPU), a media processor, a 3-D graphics processor, a security processor to prevent piracy and a final processor to manage power and conserve battery life. This makes the gaming devices costlier and the design more complicated.
- Rechargeable lithium-ion batteries are used for most of the gaming devices as most of the gaming devices consumes more battery.

It may be promising to investigate an alternative approach that has been gaining attention recently, where remote servers are responsible for executing the appropriate gaming engines, and streaming the resulting gaming video to the client devices. This approach is being explored for PC-based gaming [1][2], as it enables PC users to play any games on-demand from any PCs. Clearly the same approach can be extended to enable rich multiplayer games on mobile devices, as it will eliminate the need for mobile devices to download and execute the memory and computation intensive gaming engines.

Recent papers on Remote Server Based Mobile Gaming (RSBMG) approach discusses a quantitative model for measuring Gaming User Experience [5][6]. The authors studied the impact of the wireless network conditions such as network congestion and outages. Since users typically do not have control over network conditions, game experience can be improved by appropriately adapting the other components of the system. RSBMG does not consider the effects of video codec selection and the broader question of maximizing gaming experience using the right configuration of video codecs and game configurations. Another important problem is the selection of the right game for a set of conditions (network and device).

We propose a gaming service that would enable a user to run any high graphics game which require heavy computing and processing capabilities in any low cost remote device which is assumed to be connected a network that satisfies particular minimum requirements of bandwidth and traffic. The gaming service is implemented on a high speed wireless network like Wimax or Wifi that would guarantee a seamless experience for the user. Remote servers are responsible for executing the appropriate gaming engines, and streaming the resulting gaming video to the resource constrained client devices. Our system would classify games that can be played for a particular network stage. This approach is being explored for Internet PC-based gaming [3][4], as it enables PC users to play any games on demand from any PCs.

The problem worsens, when the video is encoded from a server and sent to a user on a resource constrained mobile device. Encoder delays, network delays at various routers and decoder delays will hamper the performance of the system. Time difference between the user's key input at the client and the server reaction has to be minimal. Also Extensibility of the application would be another important concern. Accommodating to a multi-user environment is also an important task. The server should have enough flexibility to handle network variations. These requirements led to the following design consideration:

- No Downloading of the Game is required

- No waiting time for physical media to spin up - Instantly Available
- Any game can be played in a cheap mobile device without any GPU requirements.
- Platform independent games.

3. SYSTEM DESIGN AND ARCHITECTURE

The remote gaming system developed is shown in Figure 1. The system consists of a server that streams the game to a device with a video playback and user input control. The components of the system are described in the following sub-sections.

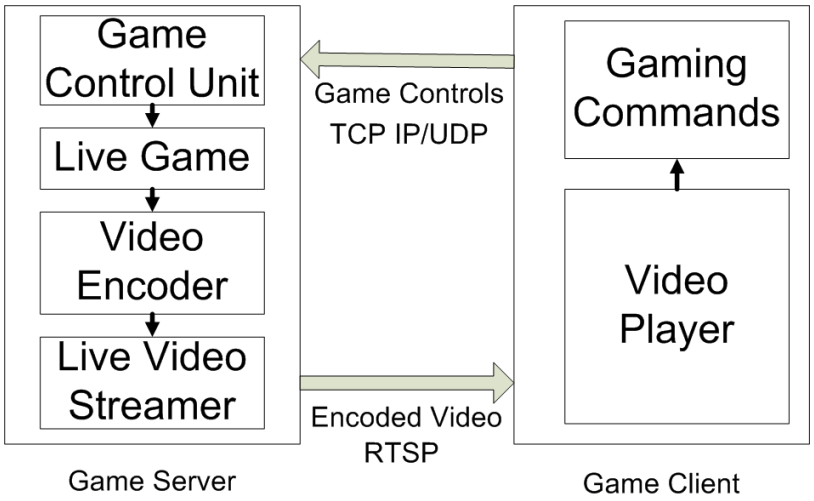

Figure 2. System design and architecture

3.1 Game Control Unit

Game control unit receives the gaming commands send from the remote client. The gaming commands are captured from user input, encoded, and streamed to users. The commands are typical control such as key and mouse inputs. The gaming commands are processed by the gaming control unit at the server and appropriate game parameters are set in the live game module. The game control unit is implemented using a socket program which continuously checks for game controls in a particular port.

3.2 Live Game

We developed a simple live game using windows sdk and directx APIs. It is a space shooting game in which there are only two characters- the player and the enemy. If the player gets hit by a bullet, his health level will reduce. Both the characters shoot at each other and try to avoid being hit. If the health of the player reduces below a particular threshold, the color of the background and the player changes. So, the motion in the game are basically motion due to the player, enemy, motion of bullets and the changes in background.

The pace of the game can be controlled using server side game configuration. The frames of the rendered game to be encoded and streamed are extracted periodically at a particular frame rate. Also a set of scripts can be run in the live game code, (e.g. high motion of the player and the enemy) which can be used to test a particular test case scenario. YUV frames are extracted from the game and fed to the MPEG encoder.

3.3 Video Encoder

The video encoder was implemented using the Intel Media Software Development Kit(Intel Media SDK), a software development library that exposes the media acceleration capabilities of Intel platforms for decoding, encoding and video preprocessing [7]. The API library covers a wide range of Intel platforms.

3.4 Live video streamer

The LIVE555 Media Server is an open source media streaming application [8]. The Media Server uses RTSP protocol for streaming. It can stream several kinds of media files including MPEG videos. These streams can be received and played by any standard RTP/RTSP media clients. VLC media player and Quick time media player are the two examples of RTSP/ RTP compliant media players we have used for evaluation. The open RTSP command-line RTSP client can receive/store the stream data, without playing the received media. The key features of LIVE555 Media Server are:

- The server can transmit multiple streams of same file or multiple streams of different files concurrently.
- The media streams are transmitted as RTP/UDP packets by default or tunneled through HTTP
- The server can also support raw UDP streaming for non-standard clients.

3.5 Video player

VLC is a multimedia player which supports most audio and video formats such as H.264, Ogg, DivX, MKV, TS, MPEG-2, mp3, MPEG-4, aac from files, physical media such as DVDs, VCD, Audio CD, TV capture cards and also many streaming protocols [9]. We use the VLC media player as a client for receiving the audio/video stream.

3.6 Gaming commands

Gaming commands at the client are detected and send back to server using TCP protocol.

Figure 2. Live Game Snaphot

4. PERFORMANCE EVALUATION

Video game is a highly interactive application. In a game session, players manipulate the game on the input devices, while their actions are perceived when the resulting video presents on the display device. In this paper, the term "Response Time" refers to the elapsed time from the time the control command is issued on the input device till the resulting video is perceived by the user. Figure 3 presents a round-trip flow of response time, from the issuance of a gaming command on the mobile device to the receiving of the resulting video frame back to the mobile device. At time T1 (point A), game client sends out an action command. Game server receives this command at time T2 (point B), and then renders a new game video frame at time T3 (point C). The generated video frame is compressed and packetized, and then is sent to the client at time T4 (point D). T5 is the expected boundary for completely receiving all the packets of a frame. However, due to the downlink delay variation, the exact time of the whole frame arriving at the client is different (earlier or later than the boundary). For instance, the first video frame has been fully received at point E (time Tx) which is later than the expected boundary, while at point H the second frame is received earlier. Early coming packets can be cached in a data buffer. To eliminate the impairment of late coming packets, we postpone displaying a received frame in the buffer for a short period, which is called play-out delay. As shown in Figure 4, the yellow windows indicate the period of play-out buffer delay. The packets of the compressed frame have been de-packetized and leave the buffer at Time T6 (point F). At time T7 (point G), the user perceives the decoded video frame on his display.

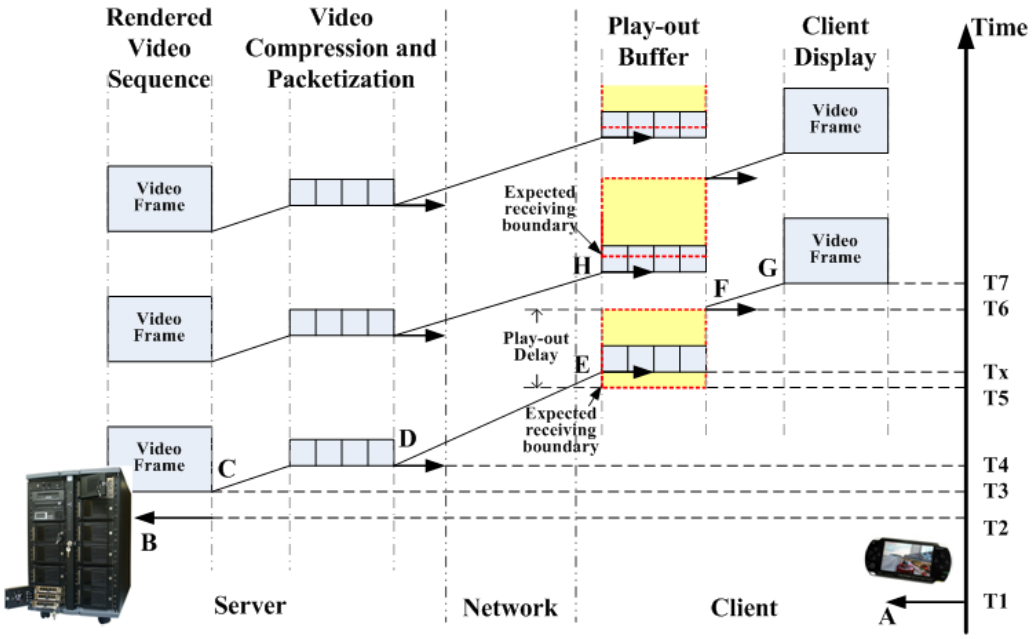

Figure 3. Response Time

Based on the above analysis, gaming response time in our approach mainly includes four sub-components: *network uplink delay* D_{UL} (T1-T2); *server delay* D_S, including game engine delay (T2-T3) and video encoding delay (T3-T4); *network downlink delay* D_{DL} (T4-T5); and *client delay* D_C (T5- T7), including *client play-out delay* D_{PL} (T5-T6) and decoding delay (T6-T7). Thus, Response Time (RT) can be formulated as:

$$\text{Network Uplink Delay} = D_{UL} = (T1\text{-}T2)$$
$$\text{Game Engine Delay} = D_{GE} = (T2\text{-}T3)$$
$$\text{Video Encoder Delay} = D_{VE} = (T3\text{-}T4)$$

Network Downlink Delay = D_{DL} = (T4-T5)
Client Caching Delay = D_{CC} = (T5-T6)
Decoding Delay =
D_D = (T6-T7)

$$Rt = D_{UL} + D_{GE} + D_{VE} + D_{DL} + D_{CC} + D_D$$

5. RESULTS

Game response time is a critical parameter for optimizing gaming user experience. Better the response time, better is the gaming user experience. Different approaches that can be used to dynamically adjust the response time in our system are adjusting encoder parameters (Frame rate, bit rate), the pace of the game in the game engine, compromising video quality over response time and re-configuring streaming packet size. Various delays components were measured in our system which is as follows:

Network Uplink Delay	D_{UL} – order of micro seconds – minimal
Video Encoding Delay	D_{VE} – 0.02 (QCIF) to 0.06 (1080p) seconds per frame.
Client Delay	$D_{CC} + D_D$ – 100 ms

We know that, the response time is:

$$Rt = D_{UL} + D_{GE} + D_{VE} + D_{DL} + D_{CC} + D_D$$

$D_{UL} + D_{DL}$ will be negligible for the assumed home wifi or wimax network and delay in the game engine is negligible when there is less interaction from the user.

Minimum bound of	$Rt > Rtmin = D_{VE} + D_{CC} + D_D$
Maximum Bound of	$Rt < Rtmax = D_{GE} + D_{VE} + D_{CC} + D_D$

The bounds of the response time are calculated as above. The response time of the system will be between the Rtmin and Rtmax. Various timing related information of the game was studied to have a very strong control on the game and its adaptation in our system.

Players fire - 900 ms to reach their target
Enemy fire takes around 1800-2000 ms
Enemy fires volleys of 3 shots each every 500ms

6. CONCLUSION

In this paper, we presented techniques to address the risks of unacceptable response time. The design and development of a system that would provide a platform to measure delays in all stages of the gaming system and optimize the response time is discussed in the paper. While in this paper, we primarily focused on the effect of delays on the server on gaming response time and video quality, in the future we intend to address codec performance in the system, and provide optimization techniques to mitigate the associated delays.

REFERENCES

[1] "Games On Demand" - http://www.microsoft.com/games/en-us/games/pages/gamesondemand.aspx

[2] Comscore report : http://www.comscore.com/Press_Events/Press_Releases/2010/4/Smartphone_Adoption_Shifting_Dynamics_of_U.S._Mobile_Gaming_Market
[3] "Onlive", http://www.onlive.com.
[4] I. Nave, H. David, A. Shani, A. Laikari, P. Eisert, and P.Fechteler, "Games@Large Graphics Streaming Architecture," in Proc. of the 12th Annual IEEE International Symposium on Consumer Electronics, pp. 1–4, Algarve, Portugal, Apr. 2008.
[5] S. Wang, S. Dey,"Addressing Response Time and Video Quality in Remote Server Based Internet Mobile Gaming", in Proc. IEEE WCNC 2010,USA.
[6] S. Wang, S. Dey, "Modeling and Characterizing User Experience in a Cloud Server Based Mobile Gaming Approach," in Proc. IEEE GLOBECOM 2009, Honolulu, USA, Nov. 2009.
[7] Intel Media SDK http://software.intel.com/en-us/articles/media-sdk-1point5-product-brief/
[8] A complete RTSP server application, http://www.live555.com.
[9] VLC (http://www.videolan.org/vlc/): Media player and streaming server