This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the WCNC 2010 proceedings.

Addressing Response Time and Video Quality in Remote Server Based Internet Mobile Gaming

Shaoxuan Wang, Sujit Dey
Mobile System Design Lab, Dept. of Electrical and Computer Engineering
University of California, San Diego
{shaoxuan, dey}@ece.ucsd.edu

***Abstract*—A new remote server based gaming approach, where the responsibility of executing the gaming engines is put on remote servers instead of the mobile devices, has the potential for enabling mobile users to play the same rich Internet games available to PC users. However, the mobile gaming user experience may be limited by risks of unacceptably high response time, and low gaming video quality, as gaming control commands, and the resulting gaming video, have to travel through wireless networks characterized by high bandwidth fluctuations, latency and packet loss. In this paper, we propose a set of application layer optimization techniques to ensure acceptable gaming response time and video quality in the remote server based approach. The techniques include downlink gaming video rate adaptation, uplink delay optimization, and client play-out delay adaptation. Experiments conducted on a commercial HSDPA network demonstrate that the proposed optimization techniques can make the remote server based approach feasible, allowing rich Internet games to be played on mobile devices, while ensuring acceptable user experience.**

I. INTRODUCTION

The emergence of new and more capable mobile devices, including smart phones, smartbooks, and netbooks, together with the steady deployment of broadband wireless networks, is making mobile access to rich Internet sites a reality. The above progress opens up a new possibility: ability to play rich Internet games, produced for PCs on wireline networks, from mobile devices. Enabling mobile Internet gaming will significantly change the experience of mobile users from thin, single player gaming possible today to rich, multi-player Internet gaming experience, of their familiar games from anywhere. It will also open the possibility of mobile service providers and Internet game developers translating the tremendous growth experienced in recent years in Internet PC games to the fast emerging mobile eco-system. However, due to the inherent hardware constraint of mobile devices (memory, graphics processing), the above goal will be difficult to achieve using the current client-server gaming architecture for PC-based Internet games, where most of the storage and computational burden of the game lies with the client device.

Instead, 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 Internet 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 Internet games on mobile devices, as it will eliminate the need for mobile devices to download and execute the memory and computation intensive gaming engines.

The Remote Server Based Mobile Gaming (RSBMG) approach, including the associated control and data flows, is shown in Figure 1. It extends the traditional game content server with two additional components: game engine servers and game streaming servers. The mobile user issues a game command on the mobile device, which is delivered to the remote gaming server using the wireless uplink channel, and accepted by the game content server. The game engine server then loads the client's account information and game data from the content server, and starts to process the game logic and user data to render the raw game video. The generated raw game video is encoded by the game streaming server, and finally sent to the mobile client via the downlink wireless connection.

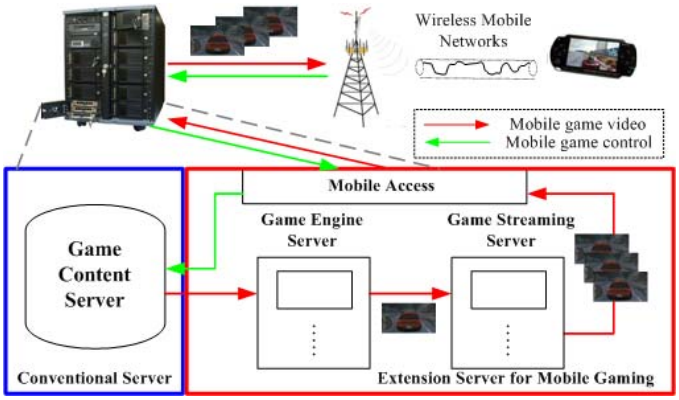

Figure 1. Overview of RSBMG architecture and data/control flow

However, the RSBMG approach will have to cope with new challenges. Firstly, the challenge of streaming gaming video through bandwidth constrained and error-prone wireless networks. Secondly, and more importantly, unlike other multimedia applications like video streaming and video download, video game is a highly interactive application, demanding very fast response time. As opposed to conventional server-client gaming architecture, where the game is executed right on the client, the new approach introduces the possibility of significantly higher response time, from the time a gaming command is issued on a mobile device, to the time the video is streamed back to the device. This risk is evidenced by the data presented in Figure 5 (a), which shows high Response Times experienced by a mobile user on a mobile HSDPA network

978-1-4244-6398-5/10/$26.00 ©2010 IEEE

when playing the Internet game World of Warcraft (WoW) using the RSBMG prototype (Figure 1). Hence, it is uncertain whether the remote server based approach can enable mobile Internet gaming, with adequate quality of experience.

In [3], we proposed a model to capture the effects of response time and video quality on Mobile Gaming User Experience (MGUE) with the RSBMG approach. We developed a new metric termed Game Mean Opinion Score (GMOS) as a measure for MGUE, and developed techniques to measure the GMOS when playing games on mobile devices on wireless networks.

In this paper, we develop optimization techniques to ensure high user experience while playing Internet video games using the RSBMG approach. We analyze the possible causes of high response time in the RSBMG approach, establish acceptable targets, and propose optimization techniques which consider the prevailing wireless network conditions to meet the response time targets. We also propose techniques that consider the gaming content and the wireless network conditions to ensure the best gaming video quality possible.

Several techniques have been proposed to address the problems of streaming video in congested networks [4][5][6]. These techniques concentrate on reducing packet loss by video rate adaptation, but do not address reducing delay, which is the main objective of this paper. Moreover, they do not make use of knowledge of the video content; in contrast, we utilize the knowledge of the gaming video to enhance the video quality delivered. There have been other networking and scheduling level efforts to address delivery delay in video streaming [7][8]. However, since video streaming is not an interactive application, naturally these techniques do no address response time, which is the primary optimization objective in this paper.

In Section II, we analyze the various contributors to response time in the RSBMG approach, and establish optimization targets and response time thresholds that need to be met. In Section III, we propose several optimization techniques for response time and gaming video quality, including a gaming video rate adaptation technique that ensures acceptable downlink delay as well as optimized video quality, an uplink delay optimization scheme, and a play-out delay adaptation technique which dynamically adjusts the play-out delay to satisfy the response time while mitigating the risks on video quality. In section IV, we demonstrate the performance and benefits of our optimization techniques in a commercially available cellular HSDPA network. Section V summarizes our findings and points out future works.

II. Response Time in Remote Server Based Mobile Gaming Approach

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 2 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 2, 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.

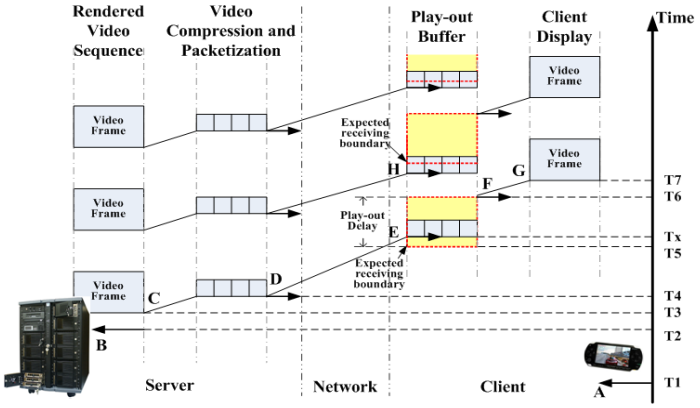

Figure 2. Round-trip flow of gaming response time

Based on the above analysis, gaming response time in the RSBMG 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:

$$RT = D_{UL} + D_S + D_{DL} + D_C \quad (1).$$

TABLE I. The RT Thresholds For Three Games: WoW, NFS, and PES

Quality	RT_E (Excellent)			RT_A (Acceptable)		
Game	WoW	NFS	PES	WoW	NFS	PES
RT(ms)	280	200	160	440	320	240

From the user experience model developed in [3], and the associated GMOS, we can calculate the RT thresholds that need to be met to achieve excellent (GMOS > 4.0), and acceptable (GMOS > 3.0) mobile gaming user experience. We term these thresholds RT_E and RT_A respectively. Table I shows the values of RT_E and RT_A for three different games, World of Warcraft (WoW), Need For Speed (NFS), and Professional Evaluation Soccer (PES), representing three different game types.

In this paper, we will develop RT optimization techniques that can be used to satisfy the RT threshold values for the

corresponding game type. Our approach consists of a set of application layer RT optimization techniques aimed towards proactively avoiding unexpected variations/increases in the various components of Response Time (Equation 1). Since the delays due to gaming engine, video encoding, and video decoding are largely fixed, and since the server delay can be assumed to be negligible if the gaming servers are not overloaded by live clients, in this paper we focus on optimizing the remaining components of RT, namely D_{DL}, D_{UL}, and D_{PL}. While ensuring desired RT, we will also have to ensure gaming video quality, so as to maximize the mobile gaming user experience.

III. Optimization Techniques for Response Time and Gaming Video Quality in RSBMG Approach

To ensure the desired response time and sufficient gaming video quality in RSBMG, in this section we will propose several optimization techniques. We start by describing a downlink gaming video rate adaptation scheme to optimize the downlink delay, as well as the gaming video quality. Subsequently, we present an uplink delay optimization and a client play-out delay adaptation scheme for response time optimization. Lastly, we describe the techniques we use to monitor the network conditions during the gaming sessions.

A. Gaming Video Rate Adaptation

Streaming gaming video over a rate-varying wireless channel can cause unexpected delay and packet loss [9], leading to undesirable increase in response time, besides adverse impact on the quality of the video streamed. One of the approaches that can be used to address this problem is video rate adaptation, which adjusts the rate of the streamed video to the fluctuating network bandwidth. While several video rate adaptation techniques have been proposed [4][5][6] for general video streaming applications, they cannot be used for gaming video. The reason is that while existing video rate adaptation techniques primarily aim to minimize packet loss during video streaming, the primary objective for gaming video should be to minimize any increase in response time. In addition, the effect on user experience of encoding parameters like bit rate and frame rate used, will vary depending on the type of game. Therefore, additional knowledge of the video game can be leveraged to develop optimal encoding settings for bit rate adaptation of gaming videos, such that the impact on user experience is minimized while adapting to lower video bit rates.

In this paper, we develop a gaming video rate adaptation scheme, which includes: 1) a methodology that utilizes the knowledge of the game to find the optimal encoding settings for each adaptation bit rate, such that the Mobile Gaming User Experience (MGUE) is maximized for that bit rate; 2) a rate-selection algorithm that aims to reduce downlink delay below a user-acceptable threshold, as well as to reduce the packet loss caused by network congestion. We start by describing how we select the optimal encoding settings for different bit rates. Later in this section, we present the rate-selection algorithm.

Optimal encoding settings for video rate adaptation

In [3], we had analyzed the effects of various factors, including video encoding settings and network conditions, on Mobile Game User Experience (MGUE). Subsequently, we had proposed and validated Impairment Functions, which model the impairment (negative effect) of the factors on MGUE. We observe that different game types have different impairments to the parameters like frame rate and frame quality (PSNR). Hence, we propose a technique that makes use of the knowledge of the game type to identify the optimal encoding parameters for the corresponding gaming video, which minimizes the impairment on MGUE. Note that the technique described below is a pre-processing step, resulting in a table of optimal encoding parameters for each game type (Table III), which is subsequently used run-time by the rate-selection algorithm described in the following subsection.

For each game, we first use video frame rate along with bit rate as the video encoding parameters. Then we measure the video frame quality (PSNR) corresponding to various settings of these parameters for a set of example video clips for the selected game. For each setting, we use the resulting PSNR, together with the frame rate, to compute the impairment on MGUE using the corresponding Impairment Function [3]. We can then use the impairment values to identify the best encoding parameters (bit rate, frame rate) for the selected game.

TABLE II. Average PSNR (left) and Impairment (right) under Various Encoding Settings of Game WoW

FPS / **Bitrate**(kbps)	**25**	**15**	**12**	**10**	**6**
700	34.3/0	36.8/0	37.3/5	38.1/15	38.5/45
600	33.6/4	**34.8/0**	36.6/5	37.2/15	37.8/45
500	33.0/10	**33.6/4**	35.0/5	36.4/15	36.9/45
400	31.0/30	32.8/12	**33.8/7**	34.4/15	35.6/45
300	27.2/54	30.6/34	32.2/23	**33.3/22**	34.0/45
200	25.4/63	28.2/51	30.1/45	**31.5/40**	32.2/58

TABLE III. Optimal Encoding Settings: Bitrate (kbps) and Framerate (fps) for 3 Different Games

Bitrate	**700**	**600**	**500**	**400**	**300**	**200**
WoW	N/A	15	15	12	10	10
NFS	25	15	12	12	10	10
PES	N/A	25	15	15	12	12

Table II presents the average frame quality PSNR values (left), and the corresponding impairment values (right), using each pair of frame rate (FPS) and bit rate for the game WoW, averaged over a random sample of 20 video clips generated when playing WoW. For each bit rate, the best frame rate setting is the one that has the least impairment value in the corresponding row of the Table. For instance, for the bit rate 300kbps, the optimal frame rate is 10, because the impairment of 22 is the least impairment in the row of 300kbps. The high-lighted settings in Table II are the optimal choices of encoding settings for game WoW. It should be noted that in the row of 700kbps in Table II, we do not select any optimal settings, because there is already a lower bit rate setting (600kbps, 15 fps) in the row below which has no impairment to MGUE, and hence is sufficient to provide the excellent gaming quality for WoW.

We can apply the above optimal encoding settings selection methodology on any other games. Table III presents the results

of optimal encoding settings for WoW and two other games: NFS and PES. From the Table III, we can clearly see that the optimal encoding settings are distinct for different games.

Rate-selection algorithm

Having known the optimal encoding settings for different bit rates, we need to develop a rate-selection algorithm, which can decide when and how to switch the bit rate. Unlike real-time video streaming, gaming video streaming is an extremely time-sensitive application, due to the stringent response time requirements for gaming (Table I). Hence, both network delay and packet loss should be taken care of when streaming gaming videos. Therefore, instead of only relying on packet loss as in most video rate adaptation techniques, gaming video rate adaptation should be sensitive to both delay and loss.

We start by giving several delay thresholds needed by the proposed rate-selection algorithm to make the adaptation decisions. First, the threshold of user-acceptable downlink delay, termed as D_{DL}^{TH}, can be calculated using uplink delay D_{UL}, play-out delay D_{PL}, and user-acceptable RT threshold RT_A:

$$D_{DL}^{TH} = RT_A - D_{UL} - D_{PL} \quad (2).$$

A primary objective of the gaming video rate adaptation algorithm is to ensure that the resulting downlink delay, D_{DL}, is lower than the downlink delay threshold D_{DL}^{TH}. Second, there is a certain minimum downlink delay, denoted by *MinDelay*, which will be experienced even in the best network conditions (no congestion or loss), due to the minimum time needed to transmit a packet through the core network and the RF link. If D_{DL}^{TH} required is smaller than *MinDelay*, we cannot meet the adaptation target D_{DL}^{TH} through lowering the application bit rate, and will need to resort to application of the uplink and/or play-out delay adaptation schemes to achieve the desired response time. Third, the traditional rate adaptation schemes have a problem of short-term rate oscillations, primarily due to the adaptation for the losses which are not caused by congestion but the other factors such as channel fading. Since it is known that the downlink delay will have noticeable increase when network is congested, we can define a delay threshold to estimate if the network is congested or not, so that we could avoid the unnecessary adaptation for non-congestion loss. We empirically use 0.2 times of *MinDelay* as the threshold to estimate the congestion condition.

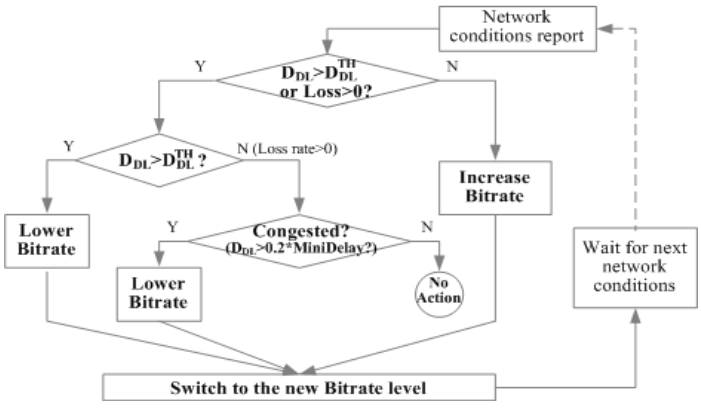

Figure 3. Work flow of rate-selection algorithm

Figure 3 shows the proposed gaming video rate selection algorithm. Depending on the network conditions, D_{DL} and packet loss, measured using the mechanisms described in Section III-D, the rate selection algorithm decides to select a lower or higher bit rate level, or keep the bit rate the same, such that 1) the downlink delay threshold D_{DL}^{TH}, is met, and 2) the packet loss due to congestion is minimized so as to reduce the impact on video quality. The rate selection mechanism is as follows:

- If the downlink delay is greater than D_{DL}^{TH}, the next lower bit rate level is selected (unless D_{DL}^{TH} is smaller than *MinDelay*, or unless the video is already using the lowest bit rate level).
- If there are packet losses (Loss > 0), and these losses are highly possible caused by network congestion (D_{DL} > 0.2**MinDelay)*, a lower bit rate level will be selected. Otherwise, the bit rate will not be changed.
- The bit rate is increased to a higher level (unless it is already at the highest level) when there is no loss and D_{DL} is smaller than D_{DL}^{TH}.

For the new bit rate selected by the rate-selection algorithm above, the optimal encoding parameters are selected by the encoder using Table III.

B. Uplink Delay Optimization

Game commands from the client to the game server must be delivered within a user-acceptable latency threshold, not only due to its important effect on user experience, but also because the delayed delivery of the commands may lead users to repeat actions, causing duplicate commands to be issued. The reason is that in the gaming session the game player will likely send the duplicate commands if he does not see the response of his first command within an expected period. If the first issued command and the later duplicated commands are both processed by the gaming server, the gaming effect will not be the same as what the game player expects.

It is generally known that TCP-based command transportation may occasionally incur long delays primarily due to its retransmission mechanism, and thus cannot meet the low latency requirement of gaming command transmission. Therefore, we select UDP transportation protocol for uplink channel to provide fast delivery of control commands, while controlling the transmissions in application layer to ensure the reliability against packet loss and out-of-order delivery.

We first calculate the uplink delay threshold D_{UL}^{TH} based on downlink delay D_{DL}, play-out delay D_{PL}, and user-acceptable RT threshold RT_A:

$$D_{UL}^{TH} = RT_A - D_{DL} - D_{PL} \quad (3).$$

When the user issues a gaming command, the client will send this command to the server with an assigned index number via UDP transportation. The server will send back an acceptance notification for the received command to the client through a TCP connection. The client periodically retransmits the command with the same index number. It will stop the retransmission if the user issues a new command, or it receives the command acceptance notification from the server, or it does not receive the acceptance notification within the user-acceptable threshold D_{UL}^{TH}.

Since the server has the knowledge of the command sequence from the index number, it can ignore the out-of-order and redundant commands. Moreover, a lost packet will not lead to a lost command, because the command in the lost packet can be recovered from the next coming packet, with a small increase of delay, which is the interval between two consequent packets. Therefore, besides the fast transmission speed achieved by UDP transportation, our proposed approach also provides reliability for the delivery of gaming commands.

C. Client Play-out Delay Adaptation

In the RSBMG system, a play-out delay buffer is implemented in the client to protect against play-out interruptions due to buffer underflow caused by random network delays. However, as explained in the previous section, the play-out delay D_{PL} also contributes to Response Time, RT. While increasing D_{PL} reduces the likelihood of video interruption, it also increases RT. In this section, we develop a play-out delay adaptation algorithm (Figure 4), which dynamically adjusts the value of D_{PL} in response to changes in the network conditions, and the uplink and downlink delays reported, such that the RT thresholds (Table I) are met, while mitigating the risk for video interruption.

Input: D_{UL}, D_{DL}, *Jitter,* RT_E, RT_A;
Output: D_{PL};
If $(D_{UL}+D_{DL}+C_{SAFEST}*Jitter < RT_E)$
    $D_{PL}=C_{SAFEST}*Jitter$;
Else if $(D_{UL}+D_{DL}+C_{SAFE}*Jitter < RT_A)$
    $D_{PL}=C_{SAFE}*Jitter$;
Else $D_{PL}=Jitter$; *End;*

Figure 4. Play-out delay adaptation algorithm

During a gaming session, we periodically measure the average uplink delay D_{UL}, downlink delay D_{DL}, and downlink delay Jitter (variation) at the application layer, using mechanisms described in Section III-D. After each such periodic measurement, the play-out delay adaptation algorithm, shown in Figure 4, is executed to determine the best value for D_{PL}. Ideally, we would like to set D_{PL} to the safest level, C_{SAFEST}*Jitter, to ensure no video interruption due to jitter. However, we do this only if the resulting response time, $D_{UL} + D_{DL} + C_{SAFEST}$*Jitter, meets the excellent RT threshold, RT_E. Else, we attempt a trade-off between achievable RT and probability of causing video interruption, by lowering the play-out delay to C_{SAFE}*Jitter, only if the resulting response time meets at least the acceptable threshold RT_A. Otherwise, we make the next level of trade-off, by setting D_{PL} to the value of observed Jitter. In our experiments, reported in Section IV, we have observed the values of 1.5 and 1.25 to be good choices for the factors C_{SAFEST} and C_{SAFE} respectively.

D. Monitoring Network Conditions

Next, we describe how the network conditions (D_{DL}, D_{UL}, downlink delay jitter, and packet loss), as well as the client play-out delay D_{PL}, are monitored while executing the optimization techniques.

We first set up a network probing mechanism, which can collect the values of D_{DL} and D_{UL} during a gaming session. We let the RSBMG server periodically send a UDP probe, which includes the probe send out time t1, to the gaming client. The client puts the probe receive time t2 and client play-out delay D_{PL} into the received probe, and returns it back to the server. When the server receives the returned probe, it will record the receive time t3. Subsequently, downlink delay D_{DL} (t2-t1), uplink delay D_{UL} (t3-t2), play-out delay D_{PL}, and downlink delay jitter can be computed by the server. The packet loss rate can be obtained from the RTCP reports sent by the client.

IV. Experimental validation

We have prototyped the Response Time and Video Quality optimization techniques on a Remote Server Based Mobile Gaming (RSBMG) platform shown in Figure 1. In this section, we report on experiments conducted to test the effectiveness of the proposed optimization techniques to enable playing Internet Games on mobile devices using the RSBMG approach. The experimental results reported here demonstrate the significant improvement in mobile gaming user experience when using our proposed optimization approach, in comparison to using the RSBMG approach by itself.

The experiments are conducted using a commercially available cellular HSDPA network. We set up the remote game server in our lab in the UCSD campus, while the Internet games are played on a mobile HSDPA client in three different scenarios: noisy network conditions (low Carrier Interference Noise Ratio (CINR)) in indoor locations, loaded network conditions in outdoor locations, and mobility conditions (30-50 miles per hour). Figure 5 shows a representative sample of data collected from numerous gaming sessions of the WoW Internet game. The original WoW gaming video settings are: 600kbps video bit rate, frame rate of 15, and VGA resolution. The left column in Figure 5 presents results using the original Remote Server approach without using the optimization techniques, while the right column shows results when additionally using the RT and VQ optimization techniques. In each column from top to bottom, we sequentially present the results of the video bit rate and packet loss rate; response time (sum of uplink delay, downlink delay, and play-out delay); and mobile gaming user experience represented by the GMOS [3]. We provide below a summary of the key observations from our experiments:

- When using the remote server (RSBMG) approach without using the optimization techniques, the user experiences unacceptably high and fluctuating response times in noisy and mobility network conditions, and response times above the excellent threshold (280ms) for WoW during loaded network conditions. There is also significant packet loss, in particular in loaded network conditions. The above leads to great instability in GMOS in the noisy and mobility conditions, amounting to poor user experience, primarily due to the RT behavior. In the loaded network conditions, the user has very poor experience, reflected by the very poor GMOS, primarily due to the high packet losses.
- As shown in Figure 5 (b), the RT optimization techniques are able to significantly reduce and stabilize the response times experienced by the user under the same network conditions, including the Uplink Delay optimization scheme

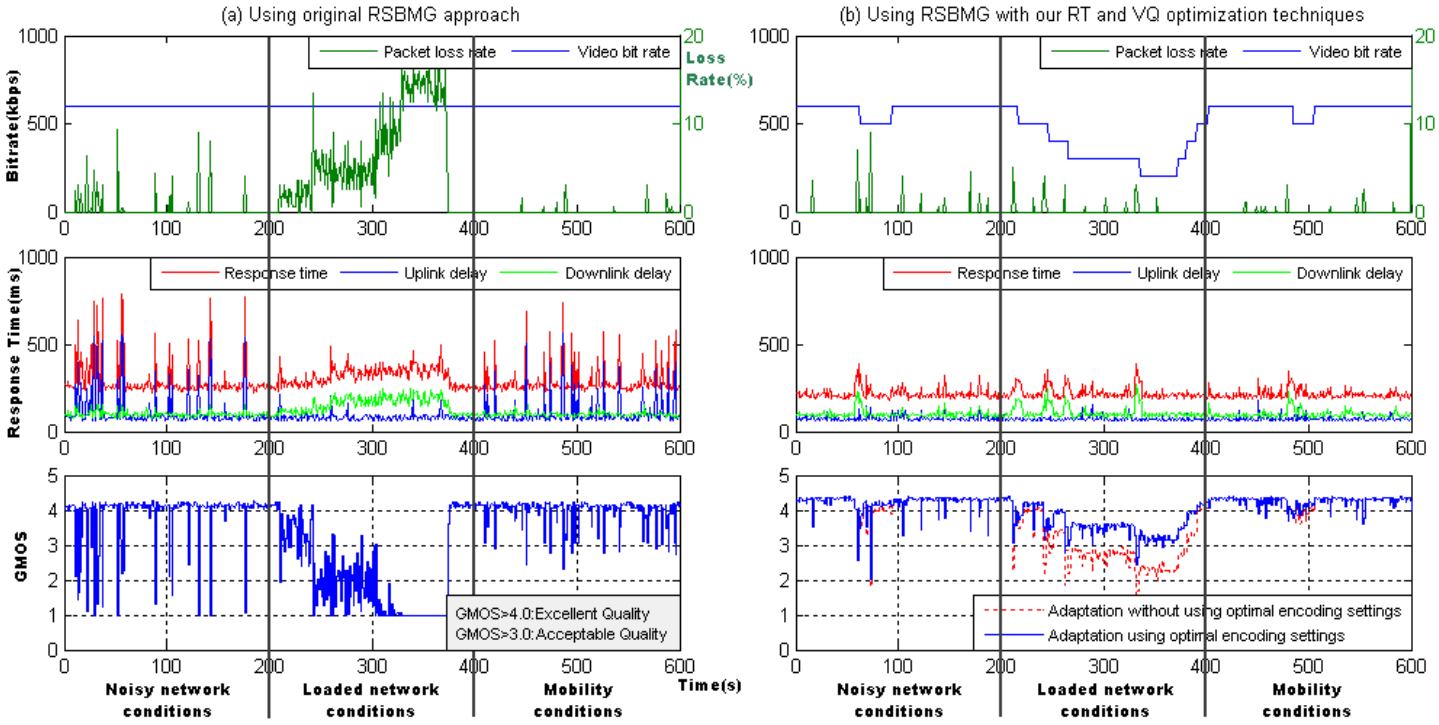

Figure 5. Results for WoW game in HSDPA cellular network: (a) using original RSBMG approach (b) using RSBMG with our RT and VQ optimization techiniques

significantly reducing the uplink delays experienced in noisy and mobility conditions.

- The high levels of packet loss and downlink delay experienced by the original RSBMG approach in loaded network conditions are successfully addressed by the Gaming Video Rate Adaptation technique. Figure 5 (b) shows the gaming video bit rates resulting from the rate adaptation, as well as the significant reductions in packet loss and uplink delay. Note that as desired, rate adaptation is not performed when packet losses are not due to congestion (like packet losses during 100-200 seconds that are accompanied by low downlink delays – they are caused by RF noise).
- Consequently, the user experience is significantly enhanced in all three network conditions, reflected by the relatively high and stable GMOS (Blue line), dipping below 3.0 (acceptable user experience threshold) only very occasionally. We also indicate (Red line) what the GMOS would have been had when the video rate adaptation algorithm is not able to use the game aware optimal encoder settings (Section III-A). We observe the significant impact of using the optimal encoder settings: on a large number of occasions, using the optimal encoder settings enhances the GMOS by a full point, making the user experience acceptable (> 3.0).

The above experimental results demonstrate that the proposed optimization techniques can make the RSBMG approach feasible, allowing rich Internet games to be played on mobile devices, while ensuring expected gaming user experience.

V. CONCLUSION

In this paper, we presented techniques to address the risks of unacceptable response time and gaming video quality in a Remote Server Based Mobile Gaming (RSBMG) approach. The experiments conducted on commercial wireless networks indicate significant improvement in response time and video quality, and thereby user experience, using the proposed techniques. While in this paper we primarily focused on the effect of wireless networks and network conditions on gaming response time and video quality, in the future we intend to address latency added by server loading and gaming engines, and provide optimization techniques to mitigate the latter effects.

REFERENCES

[1] 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.

[2] "Onlive", http://www.onlive.com.

[3] 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.

[4] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-Based Congestion Control for Unicast Applications", in *Proc. ACM SIGCOMM 2000*, Stockholm, Sweden, Aug. 2000.

[5] R. Rejaie, M. Handley, and D. Estrin, "RAP: An End-to-end Rate-based Congestion Control Mechanism for Realtime Streams in the Internet", in *Proc. IEEE INFOCOM 1999*, New York, USA, Mar. 1999.

[6] S. Mascolo, C. Casetti, M. Gerla, M. Sanadidi, and R. Wang, "TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links", in *Proc. ACM MobiCom 2001*, Rome, Italy, Jul. 2001.

[7] M. van der Schaar and D. Turaga, "Cross-layer packetization and retransmission strategies for delay-sensitive wireless multimedia transmission," *IEEE Transactions on Multimedia*, 9(1):185–197, Jan. 2007.

[8] YC Tu, et al., "Enhanced bulk scheduling for supporting delay sensitive streaming applications," Computer Networks 52 pp. 971–987, 2008.

[9] D. Wu, Y. T. Hou, and Y.-Q. Zhang, "Transporting real-time video over the Internet: Challenges and approaches," in *Proc. IEEE*, vol. 88, pp.1855–1875, Dec. 2000.