Peer-to-Peer Support for Low-Latency Massively Multiplayer Online Games in the Cloud

Richard Süselbeck, Gregor Schiele and Christian Becker
Universität Mannheim, Germany
{ richard.sueselbeck | gregor.schiele | christian.becker }@uni-mannheim.de

Abstract—**Cloud gaming has recently been proposed as an alternative to traditional video game distribution. With this approach, the entire game is stored, run and rendered on a remote server. Player input is forwarded to the server via the Internet and the game's output is returned as a video stream. This adds network delay, which can negatively impact the gameplay. The delay is acceptable as long as the user is located geographically close to the cloud servers. However, for Massively Multiplayer Online Games (MMOGs), this delay is added on top of the existing delay between MMOG client and server. As MMOGs are highly delay-sensitive, this can significantly degrade their playability. To deal with this issue, we propose to use peer-to-peer techniques to distribute the MMOG server functionality and place it at the cloud server centers. This allows us to reduce the additional delay introduced by running the MMOG clients in the cloud.**

I. Introduction

Cloud gaming has recently been proposed as an alternative to traditional video game platforms. In this type of cloud computing, the player's input is forwarded to a remote server, where the game is stored, run and rendered. The rendered images are streamed to the user as a video over a broadband internet connection. This means the player no longer needs to own powerful, up-to-date gaming hardware such as a video game console or high-end PC to play current games. Even the most hardware-demanding games can be played using hardware just powerful enough to decode the video stream. Such hardware can be a specialized "micro-console" or a standard home PC, where games can be accessed in the cloud via a special client or even the web browser. In addition, this provides an alternative distribution channel for games, which is not based on physical media or software downloads.

While this approach has several advantages, it also introduces network delay between a player's controller input and the game's response. As video games are highly interactive systems, the player must therefore be located geographically close to the server that runs the game. OnLive [2], one of several companies currently preparing a cloud gaming serving for launch, have stated that that a good gameplay experience requires the client to be located within 1000 miles of the server where the game is run and rendered [1]. If the player is located too far away, the network delay becomes too large and the gameplay experience suffers or the game becomes unplayable. As a consequence, a cloud gaming service needs a number of geographically distributed server centers in order to keep delay acceptable for all players.

When running an MMOG in the cloud, the delay introduced by running the client on a remote server is added to the delay generated by communicating with the MMOG server. A player's actions not only have to be sent to the cloud server, but then need to be forwarded to the MMOG server and then a second cloud server before they can be sent to another player. This makes it difficult to achieve acceptable delays for MMOGs, which are highly delay-sensitive applications.

We propose to utilize the peer-to-peer-based MMOG middleware prototype developed by the peers@play project in order to distribute the MMOG server functionality across several server centers. The peers@play project [3] is a cooperative project of the Universities of Mannheim, Duisburg-Essen and Hannover to develop protocols and algorithms for peer-to-peer-based massively multi-user virtual environments and games. By using peer-to-peer techniques, it becomes possible to co-locate the MMOG servers with the cloud servers. This allows us to reduce the additional delay introduced by running the MMOG clients in the cloud.

II. Our Approach

Currently, any MMOG that is run in the cloud is incurring a delay penalty. After a player initiates a game event, e.g., an attack on another player, it takes four hops for the associated update to reach the other player. This situation is depicted in Figure 1. First, the player's input is sent to the server center that runs the MMOG client. Then, the update is sent to the MMOG server. After processing the update, the MMOG server sends the resulting update to the server centers which run the clients of those players to whom the update is relevant. Finally, the servers running these clients process the update, generate the resulting video and stream it to the players. Without running the MMOG in the cloud, an update reaches an another player after only two hops, so this increases the delay experienced by the player.

However, if all players of an MMOG were running their MMOG clients on the same cloud server center, the MMOG server could be co-located with the cloud server. In this case, the communication between the MMOG clients and the MMOG server would be completely within the local network of the server center, which results in minimal added delay. This means the delay traditionally experienced between the client and the MMOG server would be almost eliminated, with only the input and streaming delay remaining. Unfortunately, the geographic locations of an MMOG's players are widely distributed, sometimes across the entire world. This approach is therefore not feasible.

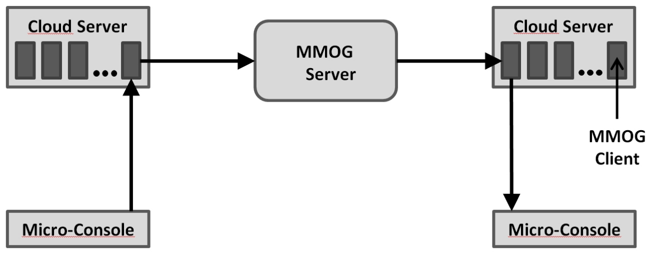

Fig. 1. Client/Server-based MMOG in the Cloud

To enable the co-location of Cloud and MMOG servers, while allowing multiple geographically distributed server centers, we propose to use techniques developed for peer-to-peer-based MMOGs. These techniques allow us to distribute the MMOG server's functionality among several servers. The servers can then be co-located with the cloud servers. Specifically, we suggest to base the MMOG on the peer-to-peer middleware that we are developing in the peers@play project.

In our approach, the combined cloud/MMOG server runs both the game client and an instance of our peer-to-peer middleware for each player that connects to it. As far as the middleware is concerned, it is still running a fully distributed peer-to-peer system, unaware that its peers are running on only a few separate server centers, instead of the systems of the individual users. Our prototype of the middleware supports this without any modifications, as we have in fact run experiments with a similar setup, where we ran a large number of peers on our IBM BladeCenter (see [4] for details).

The update propagation scheme of our middleware sends updates directly from one peer to another, i.e. with a single hop in the peer-to-peer overlay [4]. If a peer does not have sufficient bandwidth to send all updates itself, it uses a super-peer to forward the update to its recipients. These updates thus traverse two hops in the overlay. However when the peers are run on the cloud server, they are not as bandwidth-constrained as they would be when running on a user's system. In fact, we can assume that the server provides sufficient bandwidth to propagate all events. Therefore this case should not occur and when utilizing our update propagation system in the cloud, we can propagate all updates directly between the peers. This results in one hop in the overlay for each update.

The result can be seen in Figure 2. There are two possible cases for each update. First, the update can affect a player whose instance of the middleware is run on the same server center as the player who originated the update. In this case, the update can be sent from one local middleware instance to another local instance. This means the update has two traverse two hops over the internet, resulting in a total delay comparable to the traditional non-cloud client/server model. This case is shown as the dashed line in Figure 2.

In the second scenario, the update needs to be sent to a player whose instance of the middleware is run on a another server center. In this case, the update is sent to the appropriate instance via the Internet. This results in a total of three hops for the update. While this is still worse than the traditional client/server-model, it is better than the four hops required for propagation without using the peer-to-peer techniques. This is shown as the solid line in Figure 2.

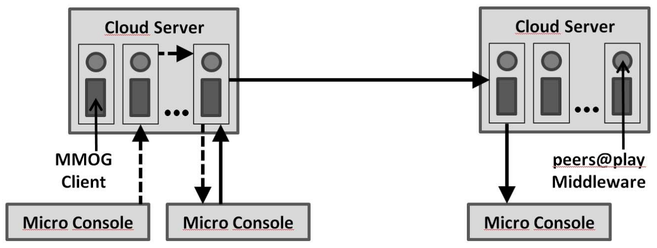

Fig. 2. Peer-to-Peer-based MMOG in the Cloud

III. Conclusions

In conclusion, our approach allows an MMOG developed as a peer-to-peer-based system to be run in the cloud. This reduces the delay penalty that would normally be associated with cloud-based MMOGs. Our approach can reduce the number of hops an update has to traverse from four to between two and three. While our prototype supports this concept without modifications, we are currently implementing several optimizations in order to evaluate our concept.

References

[1] Game developer's conference 2009: Onlive press conference, March 2009.
[2] OnLive, Inc. http://www.onlive.com/.
[3] Peers@Play Project. http://www.peers-at-play.org/.
[4] R. Sueselbeck, G. Schiele, S. Seitz, and C. Becker. Adaptive update propagation for low-latency massively multi-user virtual environments. In *Proceedings of the 18th IEEE International Conference on Computer Communications and Networks (ICCCN)*, August 2009.