# Requirements of Peer-to-Peer-based Massively Multiplayer Online Gaming

Gregor Schiele[1], Richard Süselbeck[1], Arno Wacker[2], Jörg Hähner[3], Christian Becker[1] and Torben Weis[2]

[1]Universität Mannheim
Schloss, 68131 Mannheim, Germany
{gregor.schiele | richard.sueselbeck | christian.becker}@uni-mannheim.de

[2]Universität Stuttgart
Universitätsstraße 38, 70569 Stuttgart, Germany
{arno.wacker | torben.weis}@ipvs.uni-stuttgart.de

[3]Universität Hannover
Appelstraße 4, 30167 Hannover, Germany
haehner@sra.uni-hannover.de

## Abstract

*Massively Multiplayer Online Games have become increasingly popular. However, their operation is costly, as game servers must be maintained. To reduce these costs, we aim at providing a communication engine to develop massively multiplayer online games based on a peer-to-peer system. In this paper we analyze the requirements of such a system and present an overview of our current work.*

## 1 Introduction

Massively Multiplayer Online Games (MMOG), like [5] or [16] have become increasingly popular in recent years. Today, millions of people meet and play in virtual worlds with friends and strangers. However, the operation of an MMOG is a costly task: new content must be developed, game servers must be maintained, and network bandwidth must be purchased. The provision of a suitable amount of game server resources is crucial for a game. If the game servers are overloaded, players experience high delays. If there are less players than expected, the game company faces costly overprovisioning. One way to solve this problem and to reduce the operational cost of MMOGs considerably is to organize the game as a peer-to-peer (P2P) system instead of a client/server-based one. Our goal is to develop a P2P-based communication engine for MMOGs that allows game developers to run an MMOG without the need to operate any central server for the game. Instead, the game is executed cooperatively on the hosts of its players. To do so imposes huge challenges on the P2P system, as MMOGs have extremely high requirements regarding, e.g., responsiveness, scalability, and availability.

In the past, a number of efforts to design a P2P-based system for MMOGs have been undertaken. However, most of these approaches have focused very narrowly on individual aspects of such systems, like scalability [11][15], consistency [4][17] or security [2][7]. In this paper, we build upon these earlier approaches to provide a detailed and complete requirements analysis for P2P-based MMOGs. In addition, we give an early overview of our project *peers@play*, undertaken cooperatively by the Universities of Mannheim, Duisburg and Hannover to develop a P2P communication engine for such MMOGs.

The paper is structured as follows. First, we discuss our system model. Then we analyze the requirements that a P2P-based MMOG must fulfil and briefly present the peers@play project. Finally, we discuss related work and offer a short conclusion as well as an outlook on future work.

## 2 System Model

Our system model consists of a number of users that want to play an MMOG, their devices, a communication network and the game software. Users may be located at any place in the world. The number of users is a priori undetermined and may change dynamically. Each user owns one or more computing devices, e.g., a personal computer, a laptop, personal digital assistant or mobile phone. Each device is connected to a common communication network, e.g., the Internet. In addition, each device is identifiable with a globally unique identifer. We assume the game software to be installed on each device. To play, a user activates one of her devices and starts the game software. After she is done playing, she stops the software and deactivates the

device. A player may change devices between playing sessions. Each player is represented in the game world by a special character, called his *avatar*.

## 3 Requirements Analysis

The main contribution of this paper is a detailed analysis of the requirements that a P2P-based MMOG must fulfil. In the following section, we present each of these requirements in detail. For each requirement we discuss which properties of an MMOG it is based on. In addition, we comment on whether it is specific to the support of MMOGs or if other P2P systems have the same requirement.

### 3.1 Distribution

In a P2P-based MMOG, both the game data and the computation of game events are completely decentralized. This is the main difference of a P2P-based MMOG architecture compared to a client/server-based one. With no central place to save data in a P2P system, the current world state must be partitioned into different parts and stored on the peers in a distributed manner. Similarly, the computation of active entities, such as a non-player character (NPC), in the game world must be distributed and coordinated between the peers. To do so, suitable data management, computation, and coordination agorithms are needed. Clearly, the need for distributed data management or for distributed computation is characteristic for most P2P systems. However, MMOGs require both.

### 3.2 Consistency

As a multiplayer game, an MMOG offers a shared game world, in which players operate. The state of this world must be kept consistent between all players. Ideally, an MMOG can be considered consistent, if all players always perceive the effect of an event identically and at the same time. This corresponds to the well known model of *strict consistency* and has been weakened by, for example, the definition of *linearizability* [9]. However, due to the properties of the underlying system, such as unknown network jitter, these models cannot be guaranteed. Therefore, tradeoffs need to be identified which can be achieved while satisfying the players' expectations. As an example, if a (virtual) item in the game is destroyed by a player, it seems acceptable if the effect is seen by other players with some delay. It may not be acceptable, however, for the destroyed item to be used by another player during this delay.

In contrast to MMOGs, many P2P systems – such as content distribution systems – focus on the creation and retrieval of content. In MMOGs the alteration of game world data is essential and needs to be done in a consistent and timely manner. Deleting and updating content is non-trivial in a P2P environment, if consistency among copies needs to be maintained [1]. The ordering issues that arise while altering the state of the game world often involve the ordering of events according to the physical time at which they occur in the game world. This requirement differentiates MMOGs from classical applications such as database systems, where consistency models that do not enforce chronological ordering, e.g., serializability [3], are used.

### 3.3 Self-Organization

Most MMOGs operate continuously, 24 hours a day, 7 days a week. However, users log in and out of the game regularly, leading to fluctuating availability of their devices. MMOGs must cope with this and adapt themselves automatically to the new situation. The sudden loss of a device must be detected and handled. As an example, if an NPC was computed on a device that is lost, the system may reset the NPC to the last known state and transfer control over its behavior to a new device. On the other hand, if a new peer joins the game, the system should try to relocate game tasks to it, in order to balance the load on all peers. As another example, the MMOG may adapt the number of data replicas according to the current frequency of peer losses.

This dynamic adaptation must be performed by the game autonomously. Otherwise, the cost of operation of the MMOG could increase considerably, due to the manpower needed to perform these adaptations manually around the clock.

### 3.4 Persistency

MMOGs often operate for years. Players expect to find their profiles and virtual items, as well as any changes they have affected on the world, to persist between login sessions. A failure of data persistency therefore means a failure to save critical data over longer periods of time. Additionally, the virtual world of the game may evolve while players are not logged in due to actions of other users. When a player logs back in, such changes must be visible to him instantly. This is totally different to, e.g., a classical P2P-based content distribution system, in which a data item may disappear when the last peer offering it leaves the network. To be useable for MMOGs, our system must prevent such data loss reliably.

In addition, both the current state of the game world and its earlier states must be preserved for some time. This historical data is needed for the investigation of virtual crimes, as discussed later.

### 3.5 Availability

Players who pay a regular fee to play an MMOG demand a high availability of the game. They have a very low tolerance for down-time, login problems and also want to be able to access the game on a variety of devices without extensive a priori device configuration. A P2P-based MMOG must therefore be highly fault tolerant, provide multi-client support, and be able to deal with special circumstances such as firewalls and network address translation (NAT). Clearly, this requirement is valid for all P2P systems, especially commercial ones.

### 3.6 Interactivity

MMOGs are typically highly interactive, ranging from games were users give indirect tactical orders, e.g., role-playing games, to action-oriented first person shooters. If the MMOG is not able to provide the necessary level of interactivity, the playing experience may be significantly diminished and players could get frustrated with the game. In order to provide a smooth playing experience and a natural sense of interaction, the delay between a player's actions and the game's observable reaction to them must be kept as short as possible. The maximum tolerable delay depends on the specific game, however. To do so, the P2P system should try to deliver update messages containing changes in the world state as fast and directly as possible to minimize the experienced delay. In addition, the system can use a multitude of different techniques to hide delays from users, e.g., dead reckoning. These latter approaches are game specific and beyond the scope of a generic MMOG communication engine. Interactivity is a requirement that is valid for some P2P systems, e.g., telephony systems, but only of secondary importance for others, e.g., content distribution systems. For MMOGs, providing the needed level of interactivity is essential.

### 3.7 Scalability

As the name implies, MMOGs usually involve a large number of concurrent users, making scalability a key issue in their development. Eve Online claims to hold the record of concurrent players with nearly 33000 players [6]. To cope with the potentially large number of players, our engine must deal gracefully with peak numbers while providing the ability for further expansion of the game.

Scalability is often mentioned as one of the main advantages of P2P systems over client/server systems [10]. However, this can not be taken for granted. Without further effort, a P2P-based system may even scale significantly worse than a client/server-based one. As an example, the bandwidth of each individual peer's network connection is usually much lower than that of a game server's. Therefore, it is very easy to overwhelm a single peer with too many update messages. If peers are connected to the network using communication technologies with asymmetric up- and download bandwidth (e.g., ADSL), this is especially true for outbound traffic.

### 3.8 Security

MMOGs bring together players from all over the world. This is one of their major strengths. However, this also means that players often do not know each other and are potentially untrustworthy to each other. Virtual crime may arise, and players may hack their clients in order to gain advantages in the game.

In order to support a sustainable business model, an MMOG must provide secure user accounts and safe payment options. If players do not trust the operator to be able to keep their personal and financial data secure, they will turn away from the game.

Game activities that give a player an unfair advantage over other players are considered *cheating*. Cheat prevention is vital to keep the game fair and equal for everyone, thus preventing frustration among the majority of the players, which in turn can lead those players to abandon the game.

In addition, the operator of an MMOG must be able to deal with *virtual crime*. Virtual crimes are usually violations of the game's terms of service by the players that are not considered cheating. Examples are offensive language or virtual fraud, which – depending on the business model of the game – can spill over into the real world, blurring the distinction between virtual and real crime. This makes it critical for the operator of a game to prevent virtual crime where possible and to be able to track and trace it where not.

Security issues are common for most P2P systems. However, the specific attack scenarios and the solutions for them are highly application-dependent. For MMOGs, cheating attacks can be countered by integrating voting algorithms, which ensure that each game event is checked by a number of peers before executing it. Thus, a single peer cannot cheat without being detected. Against virtual crime a logging engine can be integrated to reconstruct actions in the game. Clearly, access to this logging engine must be restricted to game operators to protect the player's privacy.

### 3.9 Efficiency

A game usually consumes a large amount of CPU resources on the individual peers. Highly detailed 3D graphics in particular use significant CPU time and are a major selling point for many games. A P2P-based MMOG must be able to operate smoothly while leaving the majority of the peer's resources available for running the ac-

tual game itself. Efficiency is especially important for P2P-based systems, as in a client/server-based game the game server can perform resource intensive tasks while the clients can concentrate on the game's 3D presentation. If the game server's performance is too low, the game provider can deploy more powerful servers. However, the performance of the peer computers cannot be influenced by the game developer. Note that efficiency is not restricted to CPU time or memory usage but includes network bandwidth as well.

### 3.10 Maintainability

Like most online games, MMOGs require constant maintenance. Cheat prevention mechanisms need to be updated frequently in order to deal with newly discovered exploits, new content needs to be integrated into the game to provide a reason for players to return, and the game rules need to be tweaked regularly to provide a fair and equal experience for everyone. While a client/server system can simply disconnect all players at a predefined point of time, install updates on the server and therefore guarantee an identical state of the game for everyone, a P2P-based game has to include special mechanisms to provide this functionality.

This requirement is much more relevant for MMOGs than for other typical P2P systems, where users are often able to connect to the network using different versions of the client software. In an MMOG this might lead to unfair situations, caused by data or rule inconsistencies. As an example, two players might use the same item. However, if this item was changed between two versions of the game, it may grant the players different advantages. This would be unacceptable for most players.

## 4 The peers@play Project

The peers@play project is performed cooperatively by the Universities of Mannheim, Duisburg, and Hannover. Our common goal is the development of a communication engine for P2P-based MMOGs. At the moment we are defining the overall architecture of this engine. The engine is structured into two parts: a basic communication middleware and a distribution framework.

The *communication middleware* is used to find and distribute game data using a P2P network. Game developers may choose one of two APIs to distribute data: the *proactive world state propagation* is used to send game events proactively to all peers requiring them. The *reactive world state propagation* can query the system reactively for specified game data, e.g., the activities of a given player in a given timeframe. On top of the middleware, a *distribution framework* offers extended support for distributed game data management and computation. Most prominently, the framework includes a number of consistency models for different types of game data.

Currently, we are working in three main areas: P2P networking, world state propagation, and consistency management. In the following, we briefly address each of these areas and present the work done so far.

### 4.1 Peer-to-Peer Networking

One central area of research in the peers@play project is the development of a suitable P2P networking layer. The task of this layer is the establishment and management of a P2P network containing all currently active player computers. The design of the layer is influenced by a number of the requirements discussed before, most prominently by the requirements for consistency and availability.

To provide a consistent game world, our P2P network must not be partitioned. Otherwise, world state and player information cannot be send to all clients. Hence, we must guarantee that the P2P network is always connected. To do so, we are constructing a *$k$-connected network graph*. A graph is said to be $k$-connected if any set of $(k-1)$ nodes can be removed from the graph and the remaining graph is still connected. We have used a similar approach successfully for sensor networks, where we provided algorithms which provably construct $k$-connected graphs [18]. Furthermore we also provided algorithms for adding and removing nodes from the graph. We intend to adapt these algorithms for the P2P network in order to keep the network connected at all times.

In addition to keeping the network connected, we must guarantee that a new player is always added to an existing network, instead of initiating a new one. Otherwise multiple networks and thus game worlds may evolve. From a technical point of view this is the so-called *bootstrapping problem* for P2P networks. In [14] we analyzed different possibilities to solve the bootstrapping problem and proposed the use of Internet Relay Chat (IRC) as a suitable solution. We continue to investigate this solution.

### 4.2 World State Propagation

The goal of the world state propagation is to distribute important changes to the state of the game world to peers. We offer two kinds of state propagation, *proactive* and *reactive* propagation. For the normal operation of the game we use proactive world state propagation, as reactive propagation would lead to longer delays for players. Reactive state propagation is only used for special activities like virtual crime investigation. As an example, if a player is suspected of having acted inappropriately, a game master may actively query the network for all state information about the player's actions in a given time period. In the remain-

der of this section we concentrate on proactive world state propagation.

The main challenge for the proactive world state propagation is to distribute state changes as fast as possible while staying scalable enough to support up to thousands of nearby player avatars. Clearly, it is impossible to distribute all changes in the game world to all peers directly. This would quickly overwhelm the P2P network. Fortunately, it is sufficient to notify peers of changes which are perceivable by their players, such as changes occuring near the player's avatar. However, this introduces a new problem: for each event changing the game world, all nearby avatars and the peers executing them must be determined efficiently. A commonly used approach to do so (see, e.g., [12][15][19]) is to partition the game world into a number of sub-areas, called *zones*. For each zone, one peer is elected dynamically to act as its *coordinator*. The coordinator manages all events occuring in its zone and determines the peers that must be notified of them.

We are currently developing a world state propagation algorithm based on this approach. In addition to existing work, we plan to switch between different coordinator modes dynamically, according to a zone's current population density. In a sparsely populated zone, peers send state changes directly to each other. The coordinator simply observes the peers and determines which peers should exchange events. As an example, if two avatars come close to each other, the coordinator notifies their peers and instructs them to send each other update messages. This way, updates are sent with little delay and a high level of interactivity can be achieved. However, if the number of nearby avatars becomes large, the direct exchange of update messages may become inefficient and limit scalability. Therefore, the coordinator switches to a second algorithm and instructs the peers to stop sending update messages to each other. Instead, the coordinator collects all changes and sends a single, integrated update message to all peers in its zone. This results in a higher delay for update messages but allows a more efficient use of the peers' communication bandwidth.

In addition to this approach, we plan to investigate different algorithms to elect coordinators and to adapt the size of zones dynamically to changing populations.

### 4.3 Consistency Management

For the purpose of consistency management, we interpret the changes in the game world as a series of events. Each event is either caused by a player interacting with the game world through his avatar or by an active entity in the game world, e.g., an NPC. The primary goal of our consistency management is to execute the series of events on all peers in such an order that no player experiences unexpected or contradictory behavior at any point in time. As an example for *unexpected behavior*, consider two players simultaneously interacting with the same (virtual) item. While one player destroys the item in his view of the game, the other player places it in his inventory. If the two peers execute the resulting events in different order, the second player may find that the item was unexpectedly removed from his inventory. An example for *contradictory behavior* is the different perception of a character's position by multiple players.

Just as in real life, the chronological ordering of events in the game world plays an important role for the behavior experienced by the players. Previously we have investigated the chronological ordering of events perceived by sensors in mobile ad hoc and sensor networks. This work has lead to the definition of a consistency model called update-linearizability [8]. It defines the ordering of events in a system to be correct, if they are executed according to the physical time at which they were captured by sensors. Accompanying algorithms guarantee this model in mobile ad hoc networks where the network topology changes frequently and where nodes may join and leave the network at any time. We believe that this work is a solid basis for developing appropriate consistency management algorithms in P2P-based MMOGs, since the handling of events in the virtual game world has great similarities to the management of events that occur in the physical world.

## 5. Related Work

As mentioned above, a common approach to scalability issues is to partition the game world into a number of zones, dynamically promoting certain peers to coordinators responsible for these zones. Knutssen et al. [15] use an application level multicast to communicate game state information within the zones, while Iimura et al. [12] use direct connections from the coordinator to the peers in order to avoid the increased network delay of the multicast approach. Yu et al. [19] use hexagonal zones and call their coordinators *master nodes*. Their solution allows players a continous view of the world, across zone borders.

Approaches that do not use coordinators include Solipsis by Keller et al. [13] and Hu et al.'s Voronoi scheme [11]. In Solipsis each peer is connected to its neighbours and observes them, reporting any changes or new nodes to all neighbours. Hu et al.'s approach utilizes a Voronoi diagram to solve the scalability problem. The Voronoi scheme fails to maintain a consistent topology if two or more adjacent boundary neighbours leave the game simultaneously [19], while Solipsis sometimes fails to detect incoming nodes [11].

Other issues that have been approached include consistency and security. Pellegrino et al. [17] propose a hybrid architecture that uses a central server to enforce a consistent game state, while all other information is distributed via a

P2P approach. The Colyseus [4] architecture takes advantage of many games' high tolerance for weak consistency and allows every peer to make local read-only copies of any game object which are then updated at appropriate times from a single primary copy. This approach also works under low-latency requirements.

Baughman et al.'s Lockstep protocol [2] is a stop-and-wait-type protocol with a commitment step, which prevents certain types of cheats, such as lookahead cheats. Their asynchronous synchronization technique then relaxes the requirements of the Lockstep protocol, by requiring lockstepppping only when interaction is required. The NEO protocol [7] is an improvement of the Lockstep protocol, which deals with a broader range of cheats, but can block players with high latency connections from playing the game.

## 6. Conclusion and Future Work

In this paper we propose a P2P-based communication engine for MMOGs and thoroughly analyze the requirements of such an engine. In addition, we present the current state of our project *peers@play*. Clearly, our analysis is only the first step towards the development of our engine. We are currently working on refining its architecture and implementing a first prototype. In order to gain more practical experience, we will deploy this prototype at the participating Universities.

## References


[1] S. Androutsellis-Theotokis and D. Spinellis. A survey of peer-to-peer content distribution technologies. *ACM Computing Surveys*, 36(4):335–371, 2004.

[2] N. E. Baughman and B. N. Levine. Cheat-proof playout for centralized and distributed online games. In *Proceedings of the 20th IEEE Conference on Computer Communications (INFOCOM'01)*, pages 104 – 113, April 2001.

[3] P. Bernstein, V. Hadzilacos, and N. Goodman. *Concurrency Control and Recovery in Database Systems*. Addison Wesley, 1987.

[4] A. Bharambe, J. Pang, and S. Seshan. Colyseus: A distributed architecture for online multiplayer games. In *Proceedings of the 3rd ACM/USENIX Symposium on Network Design and Implementation (NSDI'06)*, San Jose, CA, USA, 2006.

[5] Blizzard Entertainment. World of Warcraft homepage. published on the WWW at http://www.worldofwarcraft.com/, 2007.

[6] CCP. Eve online news: A new pcu record - close to 33,000 users. published on the WWW at http://www.eve-online.com/news/newsOfEve.asp?newsID=385, December 2006.

[7] C. GauthierDickey, D. Zappala, V. Lo, and J. Marr. Low latency and cheat-proof event ordering for peer-to-peer games. In *Proceedings of the 14th international workshop on Network and operating systems support for digital audio and video (NOSSDAV'04)*, pages 134–139, New York, NY, USA, 2004.

[8] J. Hähner, K. Rothermel, and C. Becker. Update-linearizability: A consistency concept for the chronological ordering of events in MANETs. In *Proceedings of the 1st IEEE Conference on Mobile Ad Hoc and Sensor Systems (MASS'04)*, 2004.

[9] M. P. Herlihy and J. M. Wing. Linearizability: A correctness condition for concurrent objects. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 12(3):463–492, 1990.

[10] S.-Y. Hu, J.-F. Chen, and T.-H. Chen. VON: a scalable peer-to-peer network for virtual environments. *IEEE Network*, 20(4):22–31, July-August 2006.

[11] S.-Y. Hu and G.-M. Liao. Scalable peer-to-peer networked virtual environment. In *Proceedings of the 3rd ACM SIGCOMM workshop on Network and system support for games (NetGames'04)*, pages 129–133, New York, NY, USA, 2004.

[12] T. Iimura, H. Hazeyama, and Y. Kadobayashi. Zoned federation of game servers: a peer-to-peer approach to scalable multi-player online games. In *Proceedings of the 3rd ACM SIGCOMM workshop on Network and system support for games (NetGames'04)*, pages 116–120, New York, NY, USA, 2004.

[13] J. Keller and G. Simon. Toward a peer-to-peer shared virtual reality. In *Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCSW'02)*, pages 695–700, Washington, DC, USA, 2002.

[14] M. Knoll, A. Wacker, G. Schiele, and T. Weis. Decentralized bootstrapping in pervasive applications. In *Proceedings of the 5th IEEE International Conference on Pervasive Computing and Communications (PerCom'07), Work in Progress Session (to appear)*, White Plains, NY, USA, March 2007.

[15] B. Knutsson, H. Lu, W. Xu, and B. Hopkins. Peer-to-peer support for massively multiplayer games. In *Proceedings of the 23rd AnnualJoint Conference of the IEEE Computer and Communications Societies (INFOCOM'04)*, March 2004.

[16] NCsoft. Lineage II homepage. published on the WWW at http://www.lineage2.com/, 2007.

[17] J. D. Pellegrino and C. Dovrolis. Bandwidth requirement and state consistency in three multiplayer game architectures. In *Proceedings of the 2nd ACM SIGCOMM workshop on Network and system support for games (NetGames'03)*, pages 52–59, New York, NY, USA, 2003.

[18] A. Wacker, M. Knoll, T. Heiber, and K. Rothermel. A new approach for establishing pairwise keys for securing wireless sensor networks. In *Proceedings of ACM SenSys'05*, San Diego, CA, USA, November 2005.

[19] A. P. Yu and S. T. Vuong. MOPAR: a mobile peer-to-peer overlay architecture for interest management of massively multiplayer online games. In *Proceedings of the international workshop on Network and operating systems support for digital audio and video (NOSSDAV'05)*, pages 99–104, New York, NY, USA, 2005.