Survey of P2P Game

Siqi Shen
Parallel and Distributed Systems Group, Delft University of Technology,
Mekelweg 4, Delft, the Netherlands
Email: S.Shen@tudelft.nl

***Abstract*—Millions of players play Massively Multiplayer Online Games (MMOG) daily. Because the cost and limited scalability of client/server architecture of today's MMOG; Peer-to-Peer (P2P) architectures become very popular in MMOG recently, many researches are conducted in this field. This paper presents a survey of recent advancements in P2P game regarding with its state management, overlay management and security.**

I. Introduction

Millions of people play different genres of computer games such as Massively Multiplayer Online games(MMOG), Social Network games and First Person Shooter games(FPS). As it is reported, in World of Warcraft, the most popular MMOG, there are more than ten millions subscribers[1]; FarmVille, a social network game, attract millions players daily[2]. However, in client/server approach the operation of an MMOG is costly, game company should over-provision large number of servers for game as the population of players are unpredictable[3]. Moreover, the capacity of single server/server cluster limit the number of concurrent players in one game session to few thousands, MMOG company have to use *instancing* to host multiple independent world to serve large population of players; by *instancing*, players in different game world usually can not communicate. To lower the costs of game hosting and increase the number of players in one game session to millions, Peer-to-Peer technique emerges as a promising approach to host MMOG. P2P gaming is believed to offer high scalability and is more efficiency than client/server gaming. This paper presents a survey about the advancements of P2P game in recent years.

As the MMOG game world is huge, it is impossible for a single player's computer to offer enough computing and network resource to host the game world. In typical P2P game, the game world is divided into sub-worlds; each sub-world's simulation is handled by one or several peers. The peer which is responsible for a sub-world acts as a game server of this area, the peer receives the other peers' commands, verifies and executes the commands, sends back the result executed by the commands to the other peers.

Many researches[4][5][6][7] are conducted in P2P game regarding its data distribution, consistency, persistency, security, availability etc. For example, game world could be divided using various methods such as static/dynamic method and Area of Interest(AOI); commands/states are delivered via enhanced point-to-point or application layer multicast techniques; game data are managed/transfered by structured/unstructed P2P network. Similar to the categorization of [8], P2P game design research could be categorized as follows:

1) **State management** The distribution of game states onto peers' computer and the maintenance of the game data. For example, ensuring consistency, persistency of game data, fault-tolerance and load-balancing of game.
2) **Overlay management** The construction of overlay that defines how peers inter-connect with the other peers, such as how the game data are delivered in a timely fashion.
3) **Content management** The distribution of game content such as voice, textures via utilization of client resources. As this field is close to P2P content sharing, the authors do not discuss this topic in this paper.
4) **Security** The security methods prevent and detect cheating in P2P games.

The rest of the paper is organized as follows. Section II gives a brief introduction of game interaction model and conventional game design methods. Section III discusses the state management techniques used in P2P gaming. Section IV discusses the overlays used to inter-connect peers and deliver game states/commands. SectionV discusses the security issues in P2P games. Finally, section VII concludes this survey.

II. Background

In this section, we present the design procedure of P2P game, game interaction model, game messages propagation schema and messages filter methods.

A. P2P Game design procedure

To design a P2P game, there are a few steps designer may follows:

1) Determining the story of the game, selecting game interaction model and designing game rules.
2) Deciding the distribution the game data and logic to peer.
3) Selecting proper consistent model and inconsistent mitigation method.
4) Designing persistency infrastructure to store persistent data.
5) Deciding the inter-connection of peers and the propagation of states and commands.
6) Designing security methods for cheat detection and prevention.

B. Game Interaction Model

Due the interaction model adopted by the games, different games have different latency, consistency and bandwidth requirements. Depending on how the player interact with the game world and the other player, games can be classified as either the avatar or the omnipresent model[9]. In *avatar model*, player controls single or several in-game representative of himself called avatar to interact with the game world and the other players' avatars. As the avatars are located in particular locations of the game world, the player could only view and influence the game world around his avatars. Role Playing Games (RPG), First Person Shooter (FPS) games, action games are examples of avatar model games. In RPG games such as World of Warcraft[10], players control his avatars to explore the world and conduct combat through a third person perspective. In FPS games such as Counter Strike[11], players use weapons to combat with each other through first person perspective. In *omnipresent model*, player can view and simultaneously influence portions of game world under his control; player can control hundreds of objects and each objects of player could be interacting with the other players' object simultaneously. Real Time Strategy game(RTS) such as Startcraft II[12] and OpenTTD[13] is an example of omnipresent model game. In Startcraft II, player acts as a general and controls his troops to collect resource, build factories, fight against with the other troops in real time. Due to the interaction model and game perspective(first person/third person), games have different requirements[9]; thus lead to different designs of games. In general, FPS games are the most sensitive to latency while RTS games are least sensitive[9]. Some RPG game such as Second Life[14] allows players to change the game world thus has larger requirement of network bandwidth.

C. State Propagation

State propagation is widely used in P2P or client/server avatar model games. During the gaming process, player issues his commands in his game client and this command is sent to the game server over the Internet; the game server read the command, verify it and execute it, game world state which is affected by this technique are sent back to the game client; the game client receive the change state and visualize it. Typical modern FPS games have one peer manages all the game state and sends updates of game objects to the other peer.

D. Operation Propagation/Simultaneous Simulation

In omnipresent model games, because each player could control hundreds of objects during gaming, it is difficult and not feasible using today's Internet to transmit all the state-update of in-game objects over the Internet. In order to improve the responsive of gaming experience and avoid exceed the limit of network bandwidth, the *simultaneous simulation* or *operation propagation* technique is proposed. It is widely used in popular RTS games such as Starcraft II, Age of Empires[15] and OpenTTD. Instead of transmission of the in-game objects' state, *simultaneous simulation* technique transmits player's operations; the game client transfer the command to all the other peers; each machine receives commands and simulates the game world simultaneously. Because of the simultaneous simulation nature, RTS games are resistent to most cheating techniques.

E. Area of Interest/Interest Set

During the gaming process, due to the limitation of human perception, player will only interest with a portion of the game world around his controlled-objects. The portion of the game world is called *Area of Interest(AOI)*. AOI is a conventional approach to reduce bandwidth demand of each peer. For convenience, AOI's typical shape is circular. During gaming, a peer responsible for a area only sends update of game states to peers who is interest with the area, thus greatly reduce bandwidth consumption comparing to sending updates to all peers. However, AOI works well when the population in any given area is small; as population of players in games follows a power law distribution[16], peer responsible for a populated area will not have enough upload bandwidth or computational resource; thus AOI limits the maximal number of players in one area. In order to mitigate this problem, DonnyBrook[17] introduces the notion of player's *interest set*, the set of objects player is paying attention to. As a human can only focus on a constant number of objects, players will only interest with several objects while receive base information of the other objects. *Interest set* of each player will remains small when density of a area increase.

III. State Management

State Management is central to design of P2P games. This section identifies three important aspects of state management: *State Partition*, *State Persistency* and *State Consistency*.

A. State Partition

As the game world is large, p2p techniques divide the game world into various regions. According to how the entire game world is divided, there are four genres of games: *none*, *static*, *dynamic*, *Area of InterestAOI*. We describe it separately. *None*, there is not partitioning; *static*, the game world is static divided beforehand; *dynamic*, the game world is dynamic divided during the game according to some performance metrics such as load, density of area; *AOI*, there are not explicit game world partitions, but peers only communicate with his AOI neighbors, this technique is usually combine with mutual notification(unstructured P2P overlay).

Static partition This approach divides the game world into many fix-size regions. *SimMub* [18] is an example of static partition which divides the game world rectangularly. It distributes transient game state of MMOG on P2P network while persistent user state(account information and avatar's experience) is handled by a centralize server. SimMub groups players and objects by regions, and distribute regions to peers. Peers are formed into a *Pastry*[19] overlay. Each region and peer is assigned a *Pastry* id; and the region is managed by a peer who's Pastry id is numerical closest to its id. The selected peer acts as a coordinator to resolve game update conflicts involve with share objects, and it is also root of the region's multicast tree. Each region form a *Scribe*[20] multicast tree to transfer updates of game states in that region. During gameing, each peer calculates its own state update and send it to multicast tree, and the messages are forward to peers in its region. If there is a event happen between multi-players they may communicate with each other directly and used the coordinator as a arbiter. Different from *SimMub*, *MOPAR* divides the game world into hexagonal regions. Hexagons have uniform orientation and uniform adjacency, as an typical AOI is circular, hexagonal partition is more approximated to circle than rectangular. In *SimMub*, because the random mapping property of *Pastry*, coordinator peer usually does not positioned in the region his is responsible for; *MOPAR* [21] improve this approach by introduce divide the coordinator's role into registering and coordination. In *MOPAR*, home node is responsible for registering peers in a specific region, the assignment of home node is determined by the DHT key. Each peer lies in a region need to register in the home node. Home node will assign master role to suitable peer. As master role is performed by nodes in the regions, message transfer delay is reduced and it achieve better data locality.

Dynamic partition Game workload is highly unpredictable, players may gather in an area of game world and move to another area quickly. Dynamic partition of game world is needed thus to efficiently use of P2P resource. *Dynamic Status Information Distribution*(DSID)[22] divides game world into multiple small rectangular regions, each region is managed by a sub-server. Each peer only communicate with the sub-server in a specific region. When the region is over-crowd, the sub-server may find some area-servers for help and divides the area into smaller pieces; this division depends on the load information of the area, the area may be divided evenly or divide unevenly. As the introduce of smaller areas increase the latency between commands and updates, and the inter-communication between regions, in DSID when regions can be merged into larger region to reduce inter-region communication. There are several researcher adopt dynamic partition, P2P Second Life[23] adopt similar approach to partition the game world; HYMS[24] partitions the regions of game world by recursive 2-dimensional kd-tree division.

AOI-based dynamic partition This technique does not divide the world explicitly, but peers only communicate with its neighbors. Voronoi-based Overlay Network(VON)[25] and *VSM*[4] are scalable P2P virtual environments proposed by Hu etal. In VON, the game world is not explicitly divided, peers only communicate with its AOI neighbor. It works as follow. Each peer computes a Voronoi[26](unstructured P2P) diagram with all his known neighbor peers. There are two types of neighbors, *enclosing neighbor* and *boundary neighbor*. For a given node, nodes whose regions immediately surround the given node's region are *enclosing neighbor*, *boundary neighbors* are nodes whose Voronoi region intersects with the border of the given node's AOI(AOI's shape is circular). When the player moves into game world, he notifies all his neighbors. Upon receiving the movement messages, each peer updates his local Voronoi diagram and figure out the change of neighbor relationship. In *VSM*, each peer manages the Voronoi region he lies in, he acts as a server for this region; game events generated in the region are sent to the peer first, the peer will execute the event and send updates of states to the other peer. VSM uses superpeer and dedicated server resource to achieve load balancing and efficient resource usage. VoroGame[5] and Solipsis2[27] adopts similar approach to partition the world using Voronoi diagram.

B. State Persistency

MMOGs often run for years, the information of player's avatar and items should be preserved when the player temporarily leaves and rejoins the game later. State Persistency is an important issue in P2P gaming, as all the data of game is distributed in each players' computers, if there is not state persistency; when a player shut down his computer, some pieces of game state will be lost thus may lead to the crash of the game. This is different from P2P content distribution system, in which a data item may be lost due to the last peer seeding it disappears. Although state persistency is import to P2P game, to the best of the author's knowledge only few researches were invested on game state persistency using P2P technique.

PastryMMOG[28] is P2P architecture for MMOGs, it uses PAST which is a Pastry-based file system to provide game state persistency. It is unclear whether this approach is suitable for MMOGs as no experiments were conducted.

P2P Second Life[23] is a communication infrastructure for MMOGs where users are organized via Kad which is a structured P2P network implementation of Kademlia[29]. In Second Life(SL) [14], players can participate in the design of virtual environment, thus persistency is important in SL. By the definition of [23], a game is persistent if no objects gets lost during the evolution of the virtual world. They simplifies the in-game object as (name, position) position pair and fits it into Kad Key. Each in-game object is owned by the peer who creates it, by making use of the (re)-publish operation of Kad, data is keep persistent in the network. They conduct emulation of simplify version of SL on the Internet using the resources provided by eMule users. The paper shows that the P2P SL is mostly consistent, persistent and scalable. However the player may experience high latency due to inconsistent state recovery and hops in Kad search. In *PastryMMOG* and *P2P Second Life* game world is not divided, each peer can access any global data via DHT.

Zoned federation P2P [30] divides the game world into multiple zones, the way it manages the zone is similar with MOPAR [21]. Zoned federation P2P decouples the management of transient data and persistent data of zone into different peers. Each zone owner manages the primary copy of the data/transistent data in the zone. Zone owners uses DHT to store persistent data periodically. When a peer becomes zone owners, it uses DHT to retrieve persistent data of zone and become the owner of the primary copy of the zone data.

Mammoth [31] stores persistent data in a relational database. In *mammoth*, different objects has different persistency requirement and they are stored differently. Critical data must be written synchronized with persistent storage and with high exactitude while least important data is stored asynchronously with low exactitude. For example, in *mammoth* the weapons/items acquired by player are synchronized with database write operation while movement/position information is stored using several compression method with low exactitude. *zoned federation P2P* [30] and *Mammoth relational database* could be combine to offer better persistency storage of game data.

C. State Consistency

P2P games are distributed interactive applications deployed in the Internet, due to the latency, jitter, packet loss of network it might cause inconsistent views and states of game which contradict each other; for example in FPS game although player issued movement command to avoid being shoot, his avatar still got shoot because the movement command he issued did not received by the other peers in time. Following the consistency classification and definition in distributed interactive application[32]; this section presents a brief summary of consistency model and inconsistency mitigation methods of P2P games.

1) Consistency Model: Consistency of game should be user aware, in other words, the consistency should be similar or equal to the human perception. Consistency of game also need to be context aware, different game events have different consistency requirements such as the death event's consistent requirement should be stricter than walking in garden, moreover, for a given event its consistency should be changed to comprise gameplay experience when the latency is high. Game designer should make a tradeoff between consistency and players' satisfaction. By the definition of [32], there are two major types of consistency model of games: *Ultimate consistent model* and *perceptional consistent model*. We only discuss *Ultimate consistent model* in this paper.

Ultimate/Eventual consistent In this model, data are ordered before being displayed to players. The well known "Sequential Consistency(SC)" and "Causal Consistency(CC)" falls into this category. The model only considers the logic time the command issued without takes into consideration of the particular physical time the game data should be displayed. It may take to long time to displayed an action considered consistent. P2P games are distributed continuous application in which application states are not only affect by the order of players' operations but also the physical time them are issued. Because *ultimate consistency* break the physical temporal relationship and does not guaranty responsiveness, some inconsistencies mitigation methods [33][34] are added to make to practical. For example, in order to improve the responsiveness of game, the local action is displayed directly without the communication schema to ensure consistency, and uses inconsistencies mitigation methods such death reckoning to remove inconsistencies.

2) Inconsistencies mitigation method: Inconsistencies mitigation methods suitable for games fall into two categories: *Pessimistic* and *Optimistic*. *Pessimistic method* anticipates inconsistency exists between data replicas when performing local actions. In contrast, *Optimistic method* assumes no inconsistency exists a and perform local actions instantaneously.

Optimistic method *Dead reckoning*[33][34] and TimeWarp[35] are examples of *Optimistic method*. After each peer update its own states, it estimates all the peers' states(self included) via prediction. After it gets the correct states, it compares the estimate state and the actual states; if there are differences, it sends actual states to all the other peers. This technique is used in commercial games such as Counter Strike(CS)[11]. In Colyseus[36], the players' positions and directions are predicted by *death reckoning*, and Colyseus uses this method to resolve inconsistency. In *Timewarp*, the inconsistency is removed via rollback, after the rollback the game state restore to the original state when observed right before inconsistency time. Literature [35] employs this technique, the key to use *timewarp* lies in avoid recursive rollback which reduces gameplay experience.

Pessimistic method Local-lag[37][38] is a typical pessimistic method. After a local action is issued, local-lag delays the execution of the action for a short time. During the short time period this action is transfer to remote peers, all the peers try to execute the same action at the same physical time. This technique increase the response time of players' action and need to synchronized physical time. Local-lag works efficiently when the lag is larger than the largest network latency[35]. Some RTS games such as OpenTTD use local-lag technique.

Hybrid method Several techniques have been proposed to combined *Optimistic method* and *Pessimistic method*. In [38], researchers propose combining death-reckoning with local-lag to achieve high consistency. In [37], researchers propose combining local-lag and timewarp to achieve a tradeoff between responsiveness and consistency.

IV. Overlay Management

Because the "best effort" property of Internet, the Internet doesn't guarantee network capacity or packet delivery, this is a challenge for online games. Various techniques have been proposed to avoid gameplay experience suffer from unpredictable network, ensure a global connect game world, make better use of players' bandwidth etc. Inspired by [39], [40], this section presents how game peer connects and communicates with each other via overlay.

A. Super-peer coordination

In this approach, the game world is divided into multi-regions, each region is managed or coordinate by a supper peer. Normal peer directly connect with the super-peer and forward cross-region messages via acquiring information from super-peer or using super-peer as a relay. MOPAR [21] and Mediator [41] are examples of super-peer coordination.

Mediator [41] is a typical example of *super-peer coordinator*. It divides the game world into multi-regions, each regions is manages by multiple super peers. Each region is organized by Distributed Hash Table(DHT). Normal peer in this region require peers' information of the other region via super-peer and communicate with each other directly. In each region, there are four types of super peers: boot mediator, resource mediator, interest-management mediator, zone mediator. Each mediator has different responsibility, boot mediator is for node registering, resource mediator is for resource discovery and match making, interest-manager mediator is for interest management, zone mediator is for selecting the other super-peer and load-balancing. *Mediator* adopts a condor-like pull-based job allocation method. Each peer get its computation job via biding. It also adopts a reputation system which encourage resource contribution. Moreover, *Mediator* achieves load balancing via a two-level(inter-region and intra-region) load-balancing scheme.

B. Structured P2P Application-layer Multicast

Significant research efforts have been devoted to game states distribution and interest management via multicast in recent years. As the lack of network-layer multicast, Application-layer Multicast(ALM) technique is proposed. In this genre of P2P game, peers disseminate game commands and states using ALM technique. The game world may be divided into multi-regions, each regions or the whole game world form a multicast tree. Colyseus [36] and SimMub [18] are examples of ALM.

Colyseus [36] is a P2P FPS, which supports hundreds of players; this is an order of magnitude more players than existing client/server designs. As P2P FPS games are upload bandwidth bound [36], it does not divide the game world. Each object in Colyseus has a *primary* copy and it is reside exactly on one peer. Each peer is responsible for *primary* copy of his own avatar and related objects; the synchronization of *replica* and *primary* data are handled by replica manager in each peer. Colyseus use *Mercury* [42] to place and locate primary and replica objects. *Mercury* is multi-attribute range-based dynamic load-balancing Distribute Hash Table(DHT). Mercury allows users to express ranged subscription expression of battlefield information which is akin to $0 < x < 100 \; and \; 0 < y < 100$. Because lookup in DHT is slow, Colyseus predict the players' interest and use publish/subscribe to fetch and store objects such as opponent's position in advance, and use direct point-to-point to synchronize primary and replica data.

C. Unstructured P2P protocol

In Unstructured P2P approach, peer only communicate with it neighbor. In recent years the researches of unstructured P2P protocol support for games are focus on *mutual notification*.

Mutual Notification This technique does not partition the world into multi-regions. It partitions the world using AOI instead. Each peer communicate with peers located inside its AOI, thus peer need to know all the peers in his AOI in advance. In order to achieve this requirement, peer learn his AOI neighbors via mutual neighbor notification. VON [25], VSM [4], VoroGame [5], Solipsis [43], Blue Banana [44] are examples of *mutual notification*. The drawback of this approach is that movement is the most frequent activities in game, it is costly to maintain the overlay due to avatar movement.

Solipsis [43] is a P2P virtual environment. In Solipsis, each peer is only connect with its AOI neighbor. Neighbor peers act as "watchmen" for approaching new peers. Each peer watches every movement of their neighbor and notifies the neighbor peer when new peer is found. Each peer need to inside the convex hull formed by its outmost neighbors to achieve *global connectivity*. An improved version of *Solipsis* [27] uses n-dimensional Voronoi diagram to maintain neighbor relationship.

Blue Banana [44] combines avatar mobility prediction with Solipsis. In *mutual notification*, when avatar reaches a new area, he need to get game data to construct his *playing area*, it will cause transient data failure when the game data are not delivered on time. Comparing to *Solipsis*, this method is more resilient to avatar mobility.

D. Enhanced Point-to-Point

According to the interaction model of game, some genres of games are highly affected by network latency such as FPS games. In FPS games players' performance will be greatly drop when network latency is larger than 100ms. Taking into account today's network situation(in UK, inter-server latency is on average below 55ms [45]), it means that messages need to be delivered less than 2 hops. *Enhanced Point-to-Point* technique transfer game data directly or via at most one relay to reduce network latency. The another advantage of this approach is that, it makes use of peers' heterogeneity upload bandwidth efficiently.

Donnybrook[17] is a P2P FPS game. Because AOI limited the maximum number of players in one area, *Donnybrook* proposes *interest set* which contains the objects the player is paying attention to. The calculation of *interest set* takes into account temporal and spatial factors. Each peer subscribes information of the other peers in his *interest set*, and the peers in his *interest set* sent updates of states to subscribers directly when there states are changed. For a given peer, peers which are not reside in the *interest set*, the information of them are sent to it every 2 seconds. Donnybrook uses Artificial Intelligent(AI) bots to imitate the positions of players during the two seconds. If some peers become very popular thus they are in many peers' *interest set*, Donnybrook uses peers with high upload bandwidth to help them to deliver states updates, and the maximum routing path is limited to two hops. One of the interesting result is that, some authors of donnybrook are also authors of Colyseus[36] but they did not use their DHT-based overlay Mercury[42] to locate objects anymore.

V. SECURITY

Any system which is not designed to withstand malicious behaviors is going to crash easily, P2P games are without exception. A malicious game peer may response erroneously to request or try to reveal in-game data which are not suppose to be viewed by this peer. Erroneous response could be at game application level; the peer might send fake data to request such as larger virtual wealth his company owns. Erroneous response could also happen in network level, the peer might return false route information. As network level security is common to all P2P applications, this section only focus on game application level security. We call game activities which give a unfair in-game advantage to a player *cheating*. This section presents a classification of cheating and a summary of techniques which deal with cheating.

A. Cheating Classification

Extended from the classification of [46], we classify cheating from the view of game data manipulation. Cheater may use any combination of those to achieve a specific cheating method.

1) **Authorized Automated Read** This method automatically collect authorized data which are present to player and uses some artificial intelligent technique to help/replace player to perform actions or make decisions such as automatically aiming in Halflife.
2) **Unauthorized Data Read** This method is also called information exposure, it reads unauthorized data which are not suppose view by players. For example, in Real Time Strategy(RTS) games such as WarCraft III, cheater uses this technique(maphack) to remove fog-of-war, thus cheater could see the opponents' activities to gain strategic advantage.
3) **Unauthorized Data Write** Cheaters may modify game executable, data file or game protocol to modify unauthorized game data thus achieve the advantage which is impossible to gain in unmodified version of game. A example of this is modifying the game executable to give the cheater's avatar greater strength.
4) **Prevent Authorized Read/Write** This method prevents/delays the authorized data which should be delivery to the other players. By delaying/droping the messages sent to the other players, the cheater view game information earlier than the other player.
5) **Boost Authorized Read/Write** This method boosts the reading/writing of authorized data. For example, by sending more movement commands than it is allowed, the player will move much faster than the other player. Cheater may uses cheating softwares such as *speed gear* to change the speed of game thus achieve this kind of cheating.

B. Cheat Detection and Prevention

In P2P games, as game states and logics are distributed to peers, it is easier to cheating in P2P than in client/server architecture. Dealing with cheating is non-trivial in academic field and commercial field [47]. Techniques dealing with cheating in P2P games can be classified as *game log verification*, *tamper resistent software/hardware* and *hybrid P2P and Client/Server approach*.

Game log verification Many works conduct game log analysis technique to find anomaly or detect bots [48]. Chambers etal [49] proposes a technique for detecting information exposure based on bit-commitment in P2P RTS games such as Warcraft III. Because Warcraft III adopts *operation propagation* schema, all the operations player issued are viewable to the other players. [49] only exchange operations that are located in the viewable area, while exchanging cryptographic hash value of actions lies outside viewable area; in the begin of game, each player need to exchange the hash value of initial game state; at the end of the game, each player exchanges all the operations he issues and then simulate the game again to check the validity of hash value and states/operations.

Tamper Resistent software/hardware Because preventing cheating is difficult, researchers from academia and industry proposes to increase the difficulty of cheating thus reduce the chance of cheating. Mönch etal [50] proposes *mobile guard* which aims at preventing cheating through modification of game client. *mobile guard* are small segments of code downloaded from trusted server, it will validate the game client using checksums and encrypt game data. Feng etal [46] proposes using tamper-resistant hardware to detect cheating in clients, they analysis various cheating technique and identify cheating techniques hardware could detect.

Hybrid P2P and Client/Server approach This approach combines the scalability of P2P with the centralized authority of client/server to achieve both scalability and security. Literature [51] divides the user action into no-state-change-action and state-change-action. Take movement for example, the most common events in RPG, it is classified into state-changing movement and positional movement: state-changing movement is the movement changes game state such as movement when the player attacks the other players; positional movement is movement without changing the game state such as exploring. The no-state-change-action in a region is handled by a appointed peer, the peer is used as a regional server to handle all the no-state-change-action, such as moving in safe zone. While the state-change action need to send to the server and wait for the server's state update. This approach limits the peers which can cheat to super-peers; it slightly mitigates the cheating problem but not completely prevents it. *Peer Auditing* [52] is similar to the idea of N-Version programming in fault-tolerance field. In *Peer Auditing* server distributes each computation task to peers. Each computation task is assigned to two peers, the two peers calculate the task and return the result to the server, the server will check whether these results are identical, similar or dissimilar; if the result is not similar, the computation is redone in a trusted peer or in the server and the malicious/erronous behavior is identified. In order to reduce the false positive of identifying malicious peers, *Peer Auditing* employs a reputation score based on the history of peer's correct/errornous behaviors. If the reputation decreases to some threshold, the peer is banned. [51] [52] can be combined to achieve better scalability and security. The hybrid architecture is promising in P2P games.

VI. Related Work

Schiele etal [7] identify the requirements of P2P MMOG games, it complements with this survey. Krause [39] surveys three types of P2P protocols for games, it is focus on mutual notification. Hu [40] classifies games based on *neighbor discovery*, *connectivity*, *partitioning*, *AOI type*, *latency*. It is focus on game architecture and maintains many game architectures in recent years, we follow several classifications in [40] and take into account consistency, persistency, security, [40] complements with this survey. Webb etal [53] survey on cheating in network games and gives a in-depth discussion; we classifies cheating from view of data manipulation and discusses two hybrid approaches to prevent cheating. Bouillot [32] survey consistency in distributed interactive multimedia applications; we focus on game consistency and its inconsistency compensation methods.

VII. Conclusion

This paper present an analysis of P2P game design technique. We compares game design technique from three categorizes: *state management*, *overlay management*, *security*. As it is shown in the paper, the procedure of design a P2P game is a tradeoff between distribution and communication of game data/logic, game world should be properly divided to make use of peers' resource efficiently and avoid saturate the network bandwidth. The P2P game design procedure is also a tradeoff between scalability and security, P2P technique improve the scalability of game but reducing its security, a P2P architecture couples with peer auditing may address this problem. The design procedure is also a tradeoff between distribution application performance and human gameplay performance, the design technique limits the game rule and vice versa. We expect a comprehensive architecture meet the scalability, consistency, persistency, security requirements of P2P games in the future.

Acknowledgment

The author would like to thank Alexandru Iosup for his help and Shun-Yun Hu for his related work section in his VAST website which lists and summaries many P2P gaming resources.

References

[1] Blizzard Inc., "World of Warcraft subscriber base reaches 11.5 million worldwide," 2008, [Online] Available: http://us.blizzard.com/en-us/company/press/pressreleases.html?081121, Accessed Jan 2011.
[2] "FarmVille User Daily Statistics." [Online]. Available: http://www.appdata.com/apps/facebook/102452128776-farmville
[3] V. Nae, A. Iosup, and R. Prodan, "Dynamic resource provisioning in massively multiplayer online games," *IEEE TPDS*, no. 3, pp. 380–395, 2010.
[4] S.-Y. Hu, S.-C. Chang, and J.-R. Jiang, "Voronoi State Management for Peer-to-Peer Massively Multiplayer Online Games," *CCNC*, pp. 1134–1138, 2008.
[5] E. Buyukkaya, M. Abdallah, and R. Cavagna, "VoroGame: A Hybrid P2P Architecture for Massively Multiplayer Games," *CCNC*, pp. 1–5, Jan. 2009.
[6] D. Liang and P. Boustead, "Using local lag and timewarp to improve performance for real life multi-player online games," in *NetGames*. ACM Press, 2006.
[7] G. Schiele, R. Suselbeck, A. Wacker, J. Hahner, C. Becker, and T. Weis, "Requirements of Peer-to-Peer-based Massively Multiplayer Online Gaming," in *CCGrid*. Ieee, May 2007, pp. 773–782.
[8] S. Hu, C. Wu, E. Buyukkaya, C.-h. Chien, T. Lin, M. Abdallah, J. Jiang, and K. Chen, "A spatial publish subscribe overlay for massively multiuser virtual environments," in *ICEIE*, vol. 2. IEEE, 2010, p. V2.
[9] M. Claypool and K. Claypool, "Latency and player actions in online games," *Communications of the ACM*, vol. 49, no. 11, pp. 40–45, Nov. 2006.
[10] Blizzard Inc., "World of Warcraft," http://us.battle.net/wow/en/.
[11] Valve Corporation., "Counter Strike," www.counter-strike.net/.
[12] Blizzard Inc., "StartCraft II," http://www.starcraft2.com/.
[13] OpenTTD team, "OpenTTD," http://www.openttd.org/.
[14] Linden Research Inc., "Second Life," http://secondlife.com/.
[15] M. T. Paul Bettner, "1500 archers on a 28.8: Network programming in Age of Empires and beyond," 2001, Game Developer Conference.
[16] D. Pittman and C. GauthierDickey, "A measurement study of virtual populations in massively multiplayer online games," in *NetGames*. ACM Press, 2007, pp. 25–30.
[17] A. Bharambe, J. Douceur, J. Lorch, T. Moscibroda, J. Pang, S. Seshan, and X. Zhuang, "Donnybrook: Enabling large-scale, high-speed, peer-to-peer games," in *SIGCOMM*, vol. 38, no. 4. ACM, 2008, pp. 389–400.
[18] B. Knutsson, H. Lu, W. Xu, and B. Hopkins, "Peer-to-peer support for massively multiplayer games," in *INFOCOM*, vol. 1, no. C. IEEE, 2004, pp. 96–107.
[19] A. Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," *Middleware*, no. November 2001, pp. 329–350, 2001.
[20] M. Castro, M. Jones, A.-M. Kermarrec, A. Rowstron, M. Theimer, H. Wang, and A. Wolman, "An evaluation of scalable application-level multicast built using peer-to-peer overlays," in *INFOCOM*, vol. 2. IEEE, 2003, pp. 1510–1520.
[21] A. P. Yu and S. T. Vuong, "MOPAR : A Mobile Peer-to-Peer Overlay Architecture for Interest Management of Massively Multiplayer Online Games," in *NetGames*. ACM, 2005, pp. 99–104.
[22] H.-H. Lee and C.-H. Sun, "Load-balancing for peer-to-peer networked virtual environment," in *NetGames*. ACM Press, 2006, p. 14.
[23] M. Varvello, C. Diot, and E. W. Biersack, "P2P Second Life: Experimental Validation Using Kad," in *INFOCOM*. IEEE, Apr. 2009, pp. 1161–1169.
[24] K. Kim, I. Yeom, and J. Lee, "Hyms: A hybrid mmog server architecture," *IEICE transactions on information and systems*, vol. 87, no. 12, pp. 2706–2713, 2004.
[25] S. Hu, J.-f. Chen, and T.-h. Chen, "VON: a scalable peer-to-peer network for virtual environments," *IEEE Network*, vol. 20, no. 4, pp. 22–31, Jul. 2006.
[26] F. Aurenhammer, "Voronoi Diagrams ł A Survey of a Fundamental Data Structure," *ACM Computing Surveys*, vol. 23, no. 3, pp. 345–405, 1991.
[27] D. Frey, J. Royan, R. Piegay, A. Kermarrec, E. Anceaume, and F. Le Fessant, "Solipsis: A decentralized architecture for virtual environments," in *MMVE*. Citeseer, 2008, pp. 29–33.
[28] T. Hampel, T. Bopp, and R. Hinn, "A peer-to-peer architecture for massive multiplayer online games," in *NetGames*. ACM, 2006, pp. 48–52.
[29] P. Maymounkov and D. Mazieres, "Kademlia: A peer-to-peer information system based on the xor metric," in *IPTPS*. Springer, 2002, pp. 53–65.
[30] T. Iimura, H. Hazeyama, and Y. Kadobayashi, "Zoned federation of game servers: a peer-to-peer approach to scalable multi-player online games," in *NetGames*. ACM, 2004, pp. 116–120.
[31] K. Zhang, B. Kemme, and A. Denault, "Persistence in massively multiplayer online games," in *NetGames*. ACM Press, 2008.
[32] N. Bouillot and E. Gressier-Soudan, "Consistency models for distributed interactive multimedia applications," *ACM SIGOPS Operating Systems Review*, vol. 38, no. 4, pp. 20–32, 2004.
[33] Y. Bernier, "Latency compensating methods in client/server in-game protocol design and optimization," in *Game Developers Conference*, vol. 98033, no. 425, 2001.
[34] L. Pantel and L. Wolf, "On the suitability of dead reckoning schemes for games," in *NetGames*. ACM, 2002, pp. 79–84.

[35] S. Hlavacs, "A Testbed for P2P Gaming Using Time Warp," *Edutainment*, pp. 33–47, 2009.

[36] A. Bharambe and J. Pang, "Colyseus: a distributed architecture for online multiplayer games," *NSDI*, 2006.

[37] M. Mauve, J. Vogel, V. Hilt, and W. Effelsberg, "Local-Lag and Timewarp: Providing Consistency for Replicated Continuous Applications," *IEEE Multimedia*, vol. 6, no. 1, pp. 47–57, Feb. 2004.

[38] Y. Zhang, L. Chen, and G. Chen, "Globally synchronized dead-reckoning with local lag for continuous distributed multiplayer games," in *NetGames*. ACM Press, 2006.

[39] S. Krause, "A case for mutual notification: a survey of p2p protocols for massively multiplayer online games," in *NetGames*. ACM, 2008, pp. 28–33.

[40] Shun-Yun Hu, "Vast Related work," http://vast.sourceforge.net/relatedwork.php.

[41] L. Fan and H. Taylor, "Mediator: a design framework for P2P MMOGs," in *NetGames*, 2007, pp. 43–48.

[42] A. Bharambe, M. Agrawal, and S. Seshan, "Mercury: Supporting scalable multi-attribute range queries," in *SIGCOMM*. ACM, 2004, pp. 353–366.

[43] J. Keller and G. Simon, "Solipsis: A massively multi-participant virtual world," in *PDPTA*, 2003, pp. 262–268.

[44] S. Thomas, "Blue Banana: resilience to avatar mobility in distributed MMOGs," in *DSN*, 2010, pp. 171–180.

[45] J. Miller and J. Crowcroft, "The Near-Term Feasibility of P2P MMOGs," in *NetGames*, 2010.

[46] W.-c. Feng, E. Kaiser, and T. Schluessler, "Stealth measurements for cheat detection in on-line games," in *NetGames*. ACM, 2008, pp. 15–20.

[47] M. Pritchard, "How to hurt the hackers: The scoop on internet cheating and how you can combat it," 2000, Game Developer Conference.

[48] K.-T. Chen, H.-K. K. Pao, and H.-C. Chang, "Game bot identification based on manifold learning," in *NetGames*. ACM Press, 2008.

[49] C. Chambers and W. C. Feng, "Mitigating Information Exposure to Cheaters in Real-Time Strategy Games," in *NOSSDAV*. ACM, 2005, pp. 7–12.

[50] C. Mönch, G. Grimen, and R. Midtstraum, "Protecting online games against cheating," in *NetGames*. ACM, 2006, pp. 20–31.

[51] J. Jardine and D. Zappala, "A hybrid architecture for massively multiplayer online games," in *NetGames*. ACM, 2008, pp. 60–65.

[52] J. Goodman and C. Verbrugge, "A Peer Auditing Scheme for Cheat Detection in MMOGs," in *NetGames*, 2008.

[53] S. Webb, "A survey on network game cheats and P2P solutions," *Australian Journal of Intelligent Information*, vol. 9, no. 4, pp. 34–43, 2008.