An Overview of Networking Infrastructure for Massively Multiplayer Online Games

Ahmad Fariz Ali
Dept. of Comp. Graphics & Multimedia
Fac. of Comp. Science & Info Sys
Universiti Teknologi Malaysia
+607-5532326
ahmadfariz@utm.my

Abdul Samad Ismail
Dept. of Comp Sys & Comm.
Fac. of Comp. Science & Info Sys
Universiti Teknologi Malaysia
+607-5532003
abdsamad@utm.my

Abdullah Bade
Dept. of Comp. Graphics & Multimedia
Fac. of Comp. Science & Info Sys
Universiti Teknologi Malaysia
+607-5532324
abade@utm.my

ABSTRACT
A massively multiplayer online game (MMOG) is a networked computer or video game in which tens of thousands to hundreds of thousands of players may interact with one another in real-time in a shared environment, even though these users may be separated by vast geographic distances. In the last decade the popularity of MMOGs has exploded. Unfortunately, the demand has far outweighed the resources commercial game providers can provide. Many MMOGs are suffering from scalability issues, resulting in game world partitioning, service down time and server crashes. The centralized server architecture of most modern day MMOGs is unable to cope with this increase in the number of participating players. Hence, there is a need for a scalable network architecture which can support these large number of players without affecting the overall gaming experience for each player. In this paper, we provide a review on the existing networking infrastructure solutions for MMOGs. This includes description and comparison of different implementation techniques for the deployment of massive multiplayer on-line games, which work as a client/server and peer-to-peer paradigms.

Categories and Subject Descriptors
C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design – *distributed networks, network topology.*

General Terms
Algorithms, Performance, Experimentation.

Keywords
Online games, peer-to-peer, client-server, MMOG.

1. INTRODUCTION
Online games are an increasingly popular form of entertainment on the Internet, with the worldwide online market predicted to be worth over US$10 billion in 2009 [1]. Massively multiplayer online games (MMOGs) distinguish themselves from other online games by allowing thousands of players to share a single virtual world at the same time. As an example of popular MMOG, money making online game, World of Warcraft reached a record of 11 million active subscribers worldwide as of October 2008 [2] who are paying monthly fee and purchasing expansions.

Massively multiplayer online games (MMOGs) differentiate themselves from other multiplayer online games by allowing thousands of players to share a persistent virtual world (online virtual worlds that exist 24 hours a day) at the same time. According to Dolbier (2007) [3], a persistent world is a simulation that keeps running whether players are logged in or not. The integration of players into persistent virtual world creates a new social phenomena. Players therefore even spend more money in order to play in these persistent online virtual worlds as long as they are rich in content and highly interactive. With the widespread availability of broadband internet connection, the number of players are increasing rapidly and further contribute the popularity of these games. As a result, MMOGs are becoming the dominating online game type, because they offer players the crucial game depth and social interaction.

Organization of this paper is as follows: Section 2 highlights challenges on current networking infrastructure of MMOGs. Section 3 discussed in brief about distributed virtual environment, MMOG as an example of distributed virtual environment, networking requirement s for MMOGs and the role of interest management in virtual environment applications. Section 4 described in general about data transmission approaches for supporting multiplayer online games, followed by game state processing for multiplayer online games in section 5. Section 6 discussed about workload distribution techniques for MMOGs. Section 7 described a generic communication architecture for MMOGs. Finally, conclusion is drawn in section 8.

2. CHALLENGES
An online multiplayer game is a virtual environment generated by computers, in which users may be geographically distributed, see a graphical representation of the virtual world and controls their virtual character known as avatar. The avatars communicate with each other to exchange information and do various actions. Some basic of such actions are moving the avatar, picking up objects, or communicating with other players. The scalability problems that MMOGs face rise from the fact that they have to handle a massive amount of connected players, presenting them with a consistent view of the virtual world, providing good gaming performance and an enjoying experience.

Most current commercial MMOGs are based on the client-server communication model where every packet data flows via a centralized server. The benefits of this client-server model include a tight control of the system, which makes data persistence, user authentication and secured communication channel possible. In a single session of typical MMOG, there are two main types of processes participating: (1) the server manages and regularly updates the game state depending on user inputs and the game-specific logic, and (2) the client regularly visualizes the updated game state to the user and forwards user inputs to the server. Several popular multiplayer online game types, also known as game genres, like First Person Shooter (FPS) or Real-Time Strategy (RTS) have evolved using this communication model and usually are played with a few dozens of players [3]. Aided by the increasing broadband connection bandwidth of the Internet and existing processing power at servers, the new type of Massively Multiplayer Online Games (MMOG) has evolved, which allows a high number of users to participate in a single game session. Especially the genre of Massively Multiplayer Online Role Playing Games (MMORPG) is played with possibly thousands of concurrent users in a single session. The highest number of concurrent users in a unique game world was recorded in January 2009 by 'EVE Online' to be near 45, 200 [4].While this centralized communication model is good for administrative control, security and simplicity, increasing complex game play and artificial intelligence computation has turning the server into computational and bandwidth bottleneck.

Some of the very popular and large MMOGs nowadays have managed to scale with a centralized server cluster system. For examples, in order to serve thousands of players at one time, popular MMOG providers like Sony Online Entertainment (developer for EverQuest, a fantasy MMOG) and Linden Lab (producer of Second Life, a 3-D social virtual environment) have invested a huge number of dedicated servers (between 1,500 to 2,600 servers) interconnected by high-speed networks to form a cluster of servers [5] [6]. However, server-clusters demand investment of millions of dollars to develop, deploy and maintain, but the total amount of resources is still limited at any given time. While more servers can be added, the demand placed upon the system by the players has exponential grow, resulting in a performance bottleneck [7]. Although game providers put extra cost by running more server to support more players, MMOGs still face huge scalability problems in order to be able to support hundreds of thousands of simultaneous players in a persistent virtual world.

3. DISTRIBUTED VIRTUAL ENVIRONMENT

Virtual environments, in particular multiplayer online games played over the Internet, have become a very popular class of applications used by an increasing number of users worldwide. Technically, these applications are distributed systems consisting of several clients and servers. There have been different names used for these applications – distributed interactive applications (DIAs) [8], networked interactive entertainment [9], distributed virtual environments [10], networked virtual environments (NVEs) [11], collaborative virtual environment (CVE) [12, 13, 14], such that we use their common trait of distributed virtual environments for referring to this application class.

According to Smed, Kaukoranta, and Hakonen (2002)[15], over the past twenty years three distinct classes of virtual environment have become important: (1) military simulations, (2) networked virtual environments (NVEs), and (3) multiplayer computer games (MCGs). The focus of scientific research has shifted from the military simulations (the 1980s) through NVEs (the 1990s) and currently to MCGs (see Figure 1). Military terminology prefers the word 'simulation', because they can be more than NVEs (e.g., logistical simulations). Moreover, the entertainment industry is investing seriously on MCGs, mobile gaming and online gaming in general. Thus, we survey the literature and research regarding distributed military simulation, networked virtual environments, and commercial online gaming in search of patterns for network architectures which are applicable to massively multiplayer online games.

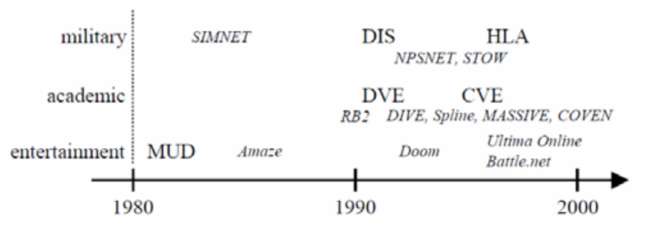

Figure 1. History of distributed virtual environments.

The relationship between games and simulations is not straightforward (see Figure 2). There are games that are simulations (e.g., football manager games) and games that are especially experience in a virtual environments (e.g., flight simulators or first person shooters). While virtual environments simulate possibly real-world environments, computer games do not necessarily belong to simulations or virtual environments. As an example, as the games get more abstracted from collaborative interactions among players, they are less and less simulations like puzzles and board games which are personally played on a single machine.

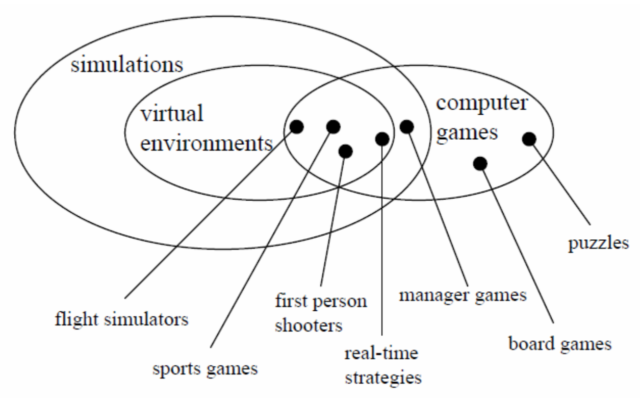

Figure 2. Relationship of simulations, virtual environments and computer games.

3.1 Distributed Virtual Environment Classes

All applications belonging to the class of distributed virtual environment systems have a common feature that for each user a client software graphically presents a virtual environment. The users can move and interact in real-time in this environment in a very responsive way. Besides online games, there are other application classes which belong to this type of distributed systems. Distributed Virtual Environment (DVE) systems like

DIVE [16] or MASSIVE [17] target at Computer Supported Collaborative Work (CSCW) scenarios like virtual conferences and distributed cooperative work. As a different application, life simulations like Second Life [18] allow to build virtual communities, chat with other users, develop the user's avatar as its personal alter ego; even building and furnishing a virtual estate is possible. There has been some work on graphical distributed operating systems in which the user interface is realized as an virtual environment. For example, the user in Croquet [19] as such a system moves an avatar around and uses portals to start and interact with different applications.

3.2 Massively Multiplayer Online Games

Massively multiplayer online games (also known as MMOGs) are a variation of the standard multiplayer online game, but with the addition that they can support thousands of simultaneous players in a persistent game world. Regarding multiplayer online games as a main application type of virtual environments, there are several different game design types called genres. There are three main genres of multiplayer online games currently played over the Internet:

3.2.1 Role-Playing Games

Role-Playing Games (RPG) for small sessions of about up to a dozen players, respectively their counterpart for huge sessions – Massively Multiplayer Online Role-Playing Games (MMORPG). MMORPGs like World of Warcraft [2], Ultima Online [20] or EverQuest II [21] can be played by thousands of concurrent users in a single game world and usually require the user to subscribe to the game service on a monthly basis.

3.2.2 First Person Shooter

First Person Shooter (FPS) allow users to play a fast-paced action game in a collaborative or competitive way. In such games like Counter-Strike: Source [22], Quake 4 [23] or Unreal Tournament , the user observes the game directly through the in-eye view from the avatar. FPS games are usually played with 2 to 32 users, the maximum number of players in most games currently is 64 users [3]. Although there are some few examples of games featuring more simultaneous users in a single session like Joint Operations [24] (up to 150 users) or PlanetSide [25] (up to 400 users), the FPS genre cannot be considered as being playable in a large-scale setting in currently available titles.

3.2.3 Real-Time Strategy

Real-Time Strategy (RTS) games are usually played by 2 to 12 players. Each of these players can control an army consisting of several different avatars, usually called units. The number of units each player can control ranges from several tens as in Command and Conquer Generals [26] over hundreds to even thousands as in Cossacks [27], for example. However, the number of human players is limited in existing RTS games, which do not allow large-scale sessions.

3.3 Networking Requirement for MMOGs

The development of MMOGs comprises many technical challenges: limited network bandwidth, computational power, latency, distributed consistency, scalability, and others. One of the biggest challenge is to scale the game state while providing distributed consistency (i.e., the shared sense of space) to a massive number of concurrent users. In order to provide a shared sense of space among players, each player must maintain a copy of the (relevant) game state on his computer. When one player performs an action that affects the world, the game state of all other players affected by that action must be updated. As the number of objects that can change and the number of players in the world increase, the amount of information that must be exchanged between players also increases.

However, the amount of information that can be exchanged between computers in an MMOG is bounded by at least two technical limitations: network bandwidth and processing power [11] [28]. Another important requirement of an MMOG is that the information about the game state must not only be exchanged between players, but it must be exchanged in a timely manner. Game state changes must be propagated within a delay that does not affect the game play, usually below human perception. Delivery of information in a timely manner must deal with the technical limitation of network latency [11] [28].

3.3.1 Network Bandwidth

The bandwidth is the amount of data that can be transmitted over a network in a fixed amount of time. The bandwidth is limited by the underlying hardware that connects the computers. On the Internet many factors can affect the available bandwidth: the user's connection to its Internet Service Provider (ISP), the ISP's hardware and software infrastructures, traffic congestion, and others. Furthermore, each user of an MMOG is likely to have different bandwidth capabilities. Ultimately, the amount of information that can be exchanged between computers in order to keep the game state consistent in an MMOG is bounded by the network bandwidth capabilities of the users. In order to scale an MMOG it must be possible to add new users without dramatically increasing the bandwidth need of other users. Interest management is a technique that has been widely used by developers to cope with high bandwidth demand of multiplayer online game.

3.3.2 Server Processing Power

The processing power is the amount of calculations that can be performed by a game server in a fixed amount of time. Every client computer's CPU has a fixed processing power capacity. In a game, processing power is required for multiple concerns such as physics, collision detection, graphic rendering. In networked games such as MMOGs, processing power is also required to send and receive network messages. The amount of information exchanged between client computers and servers does not only affect network bandwidth, but it also consumes CPU resources.

3.3.3 Network Latency

Latency is the amount of time it takes a message to travel over a network from source to destination. Although latency is ultimately bounded by the physical limitation of the underlying hardware, congestion caused by a large amount of information sent over the network also affects latency. When one network application transmits data over a network, the network's latency determines when the data becomes available for consumption by the receiving application. As players expect to interact with each other and with the environment in real-time [29, 30, 31], the MMOG developer wishes to give the user the illusion that the entire game world is located on their local machine. Network latency means that incoming data about the state of the game environment is already somewhat out-of-date by the time it arrives for processing.

3.3.4 Distributed Consistency

Latency and distributed communications make it difficult to maintain consistent world interpretations for each participant. The MMOG developer wishes to present users with a single shared environment, which is challenging if the messages communicating changes in the environment arrive at different times at each node. Further, if participants act based on differing views of the environment, some means must be provided to determine which resulting version of the environment is correct.

A lack of consistency hampers the level of immersion of the player, taking away from their enjoyment of the game and decreasing their desire to play. Fortunately, the military, academic, and gaming communities have all found that human players are often satisfied given the appearance of consistency. Overall, maintaining consistency in an online game has proven to be a difficult task for game developers; players commonly discover defects that violate the consistency of the game world, many of which may be exploited to gain an unfair advantage over other players [32, 33, 34]. These consistency violations tend to stem from implementation or design errors. Achieving true consistency in a distributed environment is often quite complex, having subtle failure modes and timing or ordering considerations.

3.3.5 Scalability

The challenge for the massively multiplayer online game developer is to design and implement a system that addresses each of the other technical challenges in the presence of large numbers of concurrent users. The required scalability aspects of a particular distributed virtual environments application depend on the actual application category and design. For example, in multi-user training simulations for natural disaster recovery scenarios, physical simulations of earthquake or behavior simulation of virtual crowds in panic are usually required to be scaled [35, 36]. On the other hand, the MMOG developer seeks to partition and distribute computational tasks and game state such that; a) each participant is aware of any and all game information which is relevant to that participant, b) each computing device participating in the game is not over-burdened, c) the capacity of the network resources available to the game is not exceeded and, d) the size of the concurrent users and the virtual environment can be increased without requiring architectural changes to the game.

3.4 Interest Management in Virtual Environment

Messages are generated by user actions (for example, moving to a particular location) and are communicated among participating players of MMOG to maintain distributed consistency. However, if messages are sent to all users in the virtual environment, the amount of transmission and processing grows at $O(n^2)$, which is clearly not scalable. Real-world observation tells us that each individual only has a limited visibility or 'area of interaction'. We cannot look around corners, through doors or walls, into pockets, listening on conversations in a busy room and so on. In other words, our interest is localized [37]. Interest management therefore deals with relevant information filtering, to reduce network resource consumption while maintaining adequate interactivity.

Interest management in virtual environment can be generally classified into three groups [15]:

- Area-of-Interest Management
- Multicast Routing
- Subscription-based Aggregation

Area-of-Interest Management and Multicast Routing are within the scope of this research and are described in the following paragraphs.

3.4.1 Area-of-Interest Management

Area-of-Interest (AOI) Management is a mechanism that filter messages such that users ideally do not receive and process messages that they do not need in order to correctly proceed in the virtual environment. Thus this mechanism provide MMOGs with intelligent information dissemination through the network that has the potential to considerably reduce the amount of information that must be exchanged between users and the resources it consumes. . It can be distance-based (by geography), class-based (by object or user attributes), or some combination of both [37].

3.4.2 Multicast Routing

Multicast Routing is a network protocol technique that realizes the area-of-interest mechanism described previously. Multicast routing is an efficient network data dissemination protocol compared to unicast and broadcast. Some distributed applications require that widely separated processes work together in groups, for example a group of users participating in virtual meeting via video conferencing system [38]. Multicast routing is ideally used when groups are large but small compared to the entire communication network. By applying multicasting in this scenario the problems associated with broadcasting and unicasting messages in large groups can be avoided. Broadcasting sends out messages to all the nodes in the network regardless whether they are interested or not. Unicasting requires the sender to send one message per receiver and this is expensive when the groups are large.

4. DATA TRANSMISSION TECHNIQUES

Data communication in larger multiplayer games can be performed using a variety of packet delivery methods. This includes basic unicast as the most popular current choice, but also broadcast and multicast approaches[39]:

4.1 Unicasting

Unicasting is communication between a single sender and a single receiver, which allows to control and direct the traffic from point to point. If the same message is intended for multiple receivers, unicasting wastes bandwidth by sending redundant messages.

4.2 Broadcasting

Broadcasting is communication between a single sender and all recipients, which means that every node has to receive and process every broadcast message. Obviously, this leads to problems as the number of participants grows, which is why broadcast transmissions are not guaranteed on the WANs.

4.3 Multicasting

Multicasting is communication between a single sender and multiple receivers, which allows receivers to subscribe to groups that interest them. The sender does not need to know all the subscribers but sends only one message, which is received by multiple receivers belonging to the group. Because no duplicate messages are sent down the same distribution path, multicasting provides an efficient way to transmit information among a large number of nodes.

Multicast transmission is not quite as efficient as basic broadcast, but is usually much more efficient than multiple unicast operations. Interest management systems in games often specify multicasting as a mean of efficiently implementing interest groups and associated network communication [40].

5. GAME STATE PROCESSING FOR MULTIPLAYER ONLINE GAMES

From the application level point of view, real-time MMOGs typically simulate a spatial virtual world where a particular player moves and acts through an avatar, which represents the player within the game. The state of the virtual world can be conceptually separated into a static part and a dynamic part. The static part covers the elements of the game world that never change, e. g., environmental properties like the landscape, buildings and other non-changeable objects. Since the static part is pre-established, no information exchange about it is required between the game participants. The dynamic part of the game state covers objects like the players' avatars, non playing characters (NPCs) controlled by the computer, items that can be collected by players or, generally, objects that can change their state. These objects are called entities and the sum of all entities is the dynamic part of the game state, such that dynamic information need to be exchanged between the game participants.

Typical multiplayer online real-time games implement an endless loop, called state update loop, which repeatedly updates the game state in real time. Figure 3 shows one iteration of the server state update loop for multiplayer games based on the client-server architecture. An iteration of state update loop consists of three steps: (1) At first the clients process the users' input and transmit them to the server. (2) The server then calculates a new game state by applying the received user actions and the game logic, including the artificial intelligence of NPCs and the virtual world simulation, to the current game state . As the result of this calculation, the states of several dynamic entities have changed. (3) The final step transfers the new game state back to the clients.

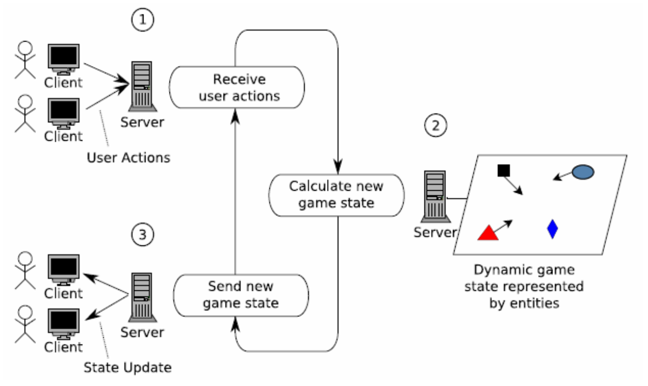

Figure 3: Typical steps in state update loop processing

There exist several possible network architectures which provide the actual placement of this state update loop component inside the distributed system [28]. In order to support game applications with distributed clients, several architectural solutions have been presented in the research literature and in commercial products. All architectures are classified as either Centralized (as illustrated in Figure 4 (a) client-server), Distributed (P2P) (as illustrated in Figure 4 (b) peer-to-peer), or Hybrid - depending on where the game state is stored. Centralized and distributed architectures store the game state on centralized servers and client machines respectively. Hybrid architectures store the game state on both centralized and client machines.

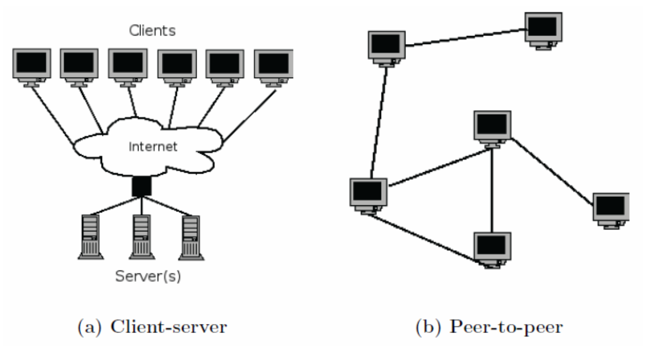

Figure 4: Network architectures

5.1 Client-server Architecture

In this traditional architecture, a single server takes care of the overall state processing and incorporates the application logic. Figure 5 illustrates the client-server architecture for game states processing. The clients run on the users' desktop computers, send user inputs to the server and present the received application state graphically. For a typical MMOG sessions running over the

Internet, the server usually runs at a dedicated server host connected to the network over a high-bandwidth link.

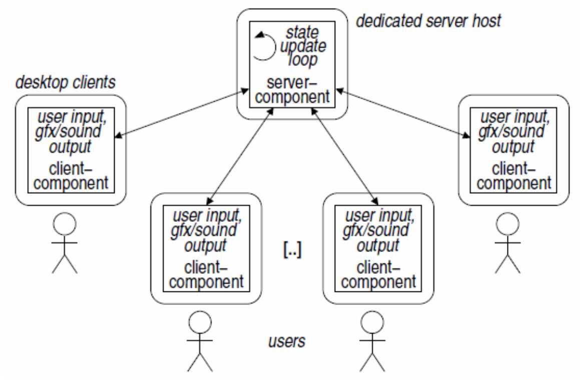

Figure 5: Game states processing in client-server architecture

One major drawback of the architecture is that this server constitutes a single point of failure for the complete session, because it is the only component of the distributed architecture knowing the complete application state: if the server fails, then there is no other component able to take over the state computation and the complete session is lost.

The server has to receive the user actions from all clients, compute the new overall state of the application and send each client the new state information according to its region of interest of the virtual environment. Therefore, the computational and communication load of the server grows at least linearly with the number of connected clients. If this load exceeds the computational or communication resources of the server host, then the server will become congested and not be able to finish the games state processing within the particular deadlines, thus violating the real-time constraints of the MMOG. Furthermore, this architecture is not scalable with the number of increasing players [41, 3, 42, 43, 5, 6] and generally support between 8 and 64 players [44]. It will continue to be the most common architecture for games with small numbers of players such as First Person Shooters (FPS) and cannot handle more than a few hundred players.

To handle large numbers of concurrent players developers use clusters of computers, also known as Server-Cluster. This is the approach used by EVE Online [4], which holds the record for the greatest number of concurrent players in one shard to be near 45, 200 and was recorded in January 2009. Each computer in the cluster is responsible for managing a different portion of the virtual world. Load balancing transfers players between computers in the cluster, keeping every machine's processing and bandwidth requirements from reaching saturation.

Although the client-server architecture exhibits a single point of failure for the online gaming session and has a limit of the maximum number of participating users, it is widely used by commercial online games, such as Quake [23], Torque Network Library [45], due to several advantages. The architecture is easy to implement and consistency of the game state is automatically achieved, because only the single server is allowed to alter the games state and thus illegal manipulations of the game state are extremely difficult [46]. The clients do not alter their copies of the entity directly, but send user inputs to the server and receive and visualize the according state change.

Another very important feature of the client-server approach is that all user inputs can be verified by the server before applying them to the overall state and that clients only need to know the state of the particular region of interest of the user. This concept of only transferring state information concerning the particular region of interest to the clients saves communication bandwidth [47, 48].

Additionally, especially in highly competitive online games, clients must not be considered trustworthy, because malicious users can hack and alter the client application to provide unfair functionality or sabotage the session. This so-called cheating [49, 50] in online games is a major problem which has to be actively countered by implementing cheat-secure mechanisms for communication and game state processing.

The client-server approach naturally implements the region of interest concept by sending the clients only limited information about the overall virtual environment, which prevents hacked clients to reveal information to the user which should be hidden, like the positions of hidden opponents in an action game. Additionally, the server verifies each user action before executing it, which allows to implement particular anti-cheating mechanisms like plausibility checks at the central server host. Often, these advantages push game middleware providers to adopt this architectural solution [51].

5.2 Peer-to-peer Architecture

In the common, fully connected peer-to-peer (P2P) architecture there is no particular dedicated process computing the state updates. Instead, each process running at a user's desktop computer has its own built-in server that processes the complete virtual world. All peers consisting of the two components are interconnected with each other and have to synchronize their world states as illustrated in Figure 6.

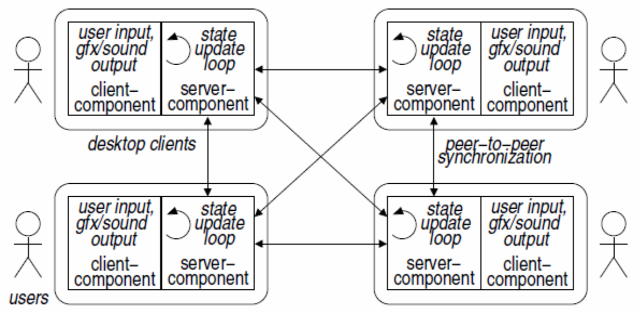

Figure 6: Game states processing in P2P architecture.

This peer-to-peer architecture approach does not require a central server and, therefore, allows users to run virtual environment sessions without accessing a server machine. Additionally, the absence of a central server means that there is no single point of failure, such that when a particular peer fails, the remaining clients are able to continue the application session. Each peer has to receive and process all actions of remote users and send the actions of its own user to all other peers. The distributed processing at the server-components has to realize a particular consistency model which ensures that the distributed application state at all peers is the same. In online game implementations, user actions are commonly timestamped and appropriately ordered at the peers before they are executed, as for example described in [52] for the real-time strategy game Age of Empires [53].

All peers have to execute the same computations and send and receive the same amount of state and action of game data. Therefore, the maximum number of connected peers is determined by the host with the lowest computation power and communication bandwidth. If this particular peer is not able to maintain the processing update rate or communicate state information in time anymore, then the complete session will become unresponsive. Consequently, the fully connected peer-to-peer architecture has only been implemented for small-scale online action and strategy games.

The use of the peer-to-peer architecture for virtual environments has been discussed in detail during the last decade [54] [55]. However, with Internet-based dedicated servers becoming less costly, the advantage of peer-to-peer approach lost its importance. Recent online games do not use the fully connected peer-to-peer concept anymore. However, different heterogeneous concepts using concepts related to the peer-to-peer approach have been presented, as for example [56, 57, 58] that resort to P2P protocols which dynamically build overlay networks to support communication among peers.

6. PARALLELIZATION OF MMOGS WORKLOAD DISTRIBUTION

In MMORPGs, which incorporate a huge virtual world, the scalability aspect of increasing the total player number is mostly happened. This aspect is made scalable by using the zoning approach: the world is subdivided into independent regions, so-called zones, which are then processed in parallel on several servers. This results in a very good scalability if the virtual world is very large and thus can be subdivided in many zones. Besides zoning, instancing is another commonly used approach which creates multiple, independent copies of particular areas of the virtual world, called instances. Instancing limits the user interactivity in the application, because users residing in a particular instance cannot interact with those in other instances.

6.1 Zoned Virtual World

Zoning [59, 60] partitions the virtual world into disjoint parts, called zones, and which are processed in parallel on different servers, as illustrated by Figure 7. A particular client is connected to the server responsible for the region in which the single avatar of the user is located and has to change the server connection if the user moves the avatar into another zone. Although clients can move between zones, no interaction between clients in different zones is possible. The zoning approach fits best for games where the players are reasonably distributed in a very large virtual world.

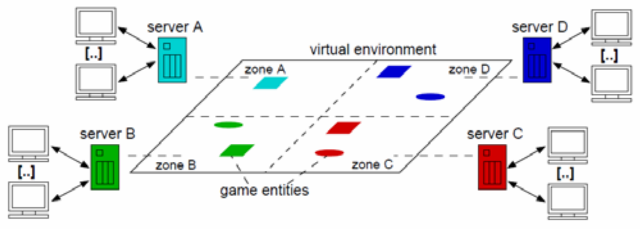

Figure 7: Partitioning a virtual world into several zones.

Figure 7 depicts an example of a virtual environment partitioned into four zones, where each zone-server handles its entities independently of the rest of the environment. The overall workload partitioning of the entity processing is done in an independent, side-by-side manner based on the virtual geography, i.e., the large virtual world. Communication between servers only occurs when a user moves the avatar from one zone into another: The client has to establish a new connection to the server responsible for the new zone and the servers have to migrate the client's state from one to the other. Zoning is used especially in Massively Multiplayer Online Role-Playing Games like EverQuest 2 [21] or World of Warcraft [2] which feature a very large game world suitable to be partitioned this way. Figure 8 shows the world map of the MMORPG Ragnarok Online split up into 80 independent zones.

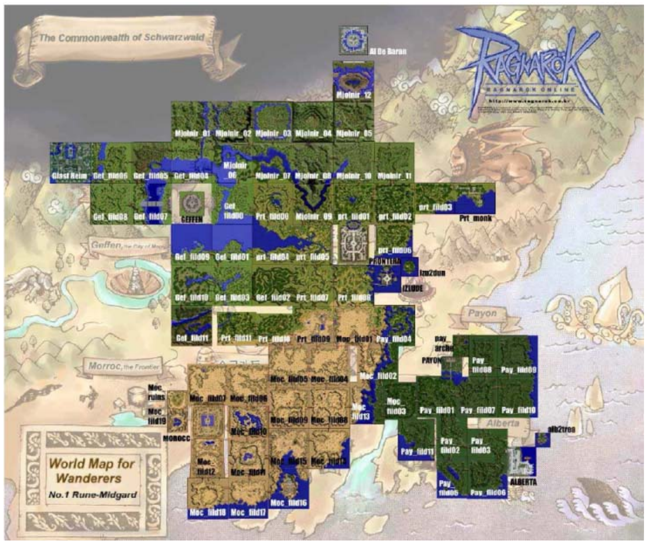

Figure 8: Virtual world of the MMORPG Ragnarok Online partitioning.

6.2 Instanced Virtual World

The instancing approach creates several independent copies of particular regions of the virtual environment, which can be processed in parallel [61]. Instances allow to run independent copies of such spaces or regions on several servers as illustrated in Figure 9. Ultima Online refers to these instances of regions as shards [20]. Usually, some particular regions of virtual worlds are more frequently visited than others, as for example city areas to meet and interact or dungeon areas to adventure in By creating independent copies of a same region of the virtual world, the instancing concept allows to handle congestion of particular areas. If for example, a lot of users move into the same dungeon in an MMORPG, independent instances of that dungeon allow to parallelize the processing of that area. In role-playing games like World of Warcraft, users arrange in teams (so-called groups) to play together in a particular instance of a certain region. This way, a high number of teams are able to adventure in the same region without congesting the real-time processing of a single server, because the independent instances can be operated on several servers. Instancing also allow the effects of server failure to be isolated to the players who are using the failed server.

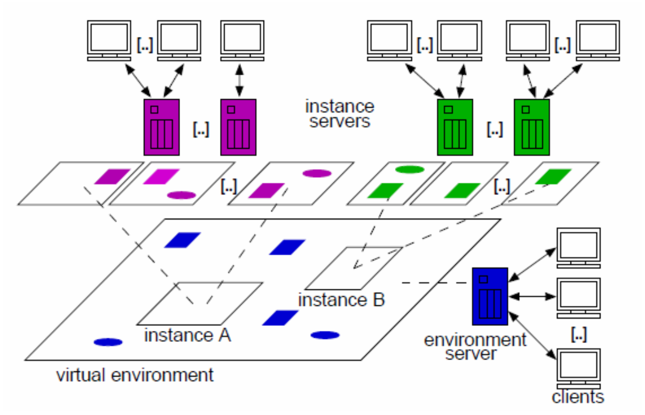

Figure 9: Independent instanced copies of particular virtual world regions.

7. GENERIC MMOGs COMMUNICATION ARCHITECTURE

A generic multiplayer online game core components can be generalized by a layered communication architecture as illustrated in Figure 10. At the top level are the virtual world or scenarios with which the players interact. This layer is composed of virtual world objects (as described previously) and events (for representing user input and game world actions). The level below is the game state management component, which continuously processes user actions and virtual world events, according to the rules of the game logic, to compute a new game state. Network Bandwidth Management is responsible to control network packet transmission of data messages generated from previous two upper layer, in a bandwidth efficient manner. This layer may comprises of 'Interest Management', 'Packet Compression & Aggregation' and 'Dead Reckoning' techniques. The Interest Management component implements Area-of-Interest concept which assigns each avatar in the game world a specific area where dynamic game information is relevant and thus has to be transmitted to the avatar client. This component optimizes network bandwidth by omitting irrelevant information in the communication. The purpose of 'Packet Compression is to reduce the size of network packet (number of bits) needed to represent particular game generated messages. Thus the compression of network packet size helps to minimize overall network traffic. In addition, 'Packet Aggregation' [28] further reduces high bandwidth multiplayer online games requirement by merging several packets and transmitting their content into one larger packet. In 'Dead Reckoning' [29, 30], the missing network packet during a running online game session is computed with an approximation techniques. Based on previously received packets, the node predicts movement of particular object. The predicted state is used until new information is received from the sender node. Depending on the prediction accuracy, the approximated location can be some distance from its actual location, as illustrated in Figure 11.

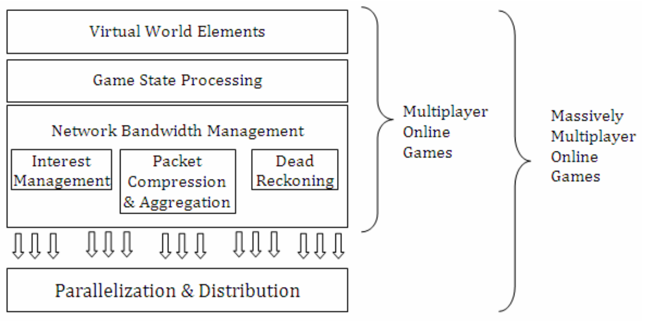
Virtual World Elements
Game State Processing
Network Bandwidth Management
Interest Management
Packet Compression & Aggregation
Dead Reckoning
Parallelization & Distribution
Multiplayer Online Games
Massively Multiplayer Online Games

Figure 10: Communication architecture for (massively) multiplayer online games.

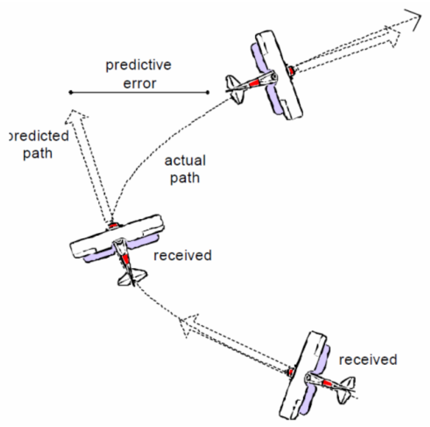

Figure 11: Dead Reckoning Example

The first three layers, as illustrated in Figure 10, are fundamental components for any client-server-based multiplayer online game. However, in the case of massively multiplayer online games, an additional layer of 'Parallelization & Distribution' components is required for achieving high scalability.

8. CONCLUSION

With scalability in the area of distributed virtual environment, we refer the capability of the MMOG application to sustain the real-time processing constraints for an increasing application demand. There are several different aspect of scalability in which the processing demand of the distributed interactive applications increases; the common characteristic of these aspects is that they require more computing resources, such as computational processing power, communication bandwidth, storage space, etc. when being increased. The underlying architecture of such distributed system must be expandable for adding such resources in order to scale at a particular level and maintain timely virtual world state computations.

9. REFERENCES

[1] Worldwide Market Forecasts for the Video Game and Interactive Entertainment Industry. DFC Intelligence 2008 June 30 [cited 2008 16 December]; Available from: http://www.dfcint.com.

[2] World of Warcraft Surpasses 11 Million Subscribers Worldwide. Blizzard Entertainment (Press Releases) 2008 October 28 [cited 2008 16 December]; Available from: http://www.blizzard.com/us/press/081028.html.

[3] Dolbier, G., 'Massively multiplayer online games, Part 1: A performance-based approach to sizing infrastructure', IBM developerWorks (Technical Report), 10 Apr. 2007.

[4] Welsh, O. EVE Breaks Concurrent User Record. EuroGamer News. 2009 January 6. [Online]. Available at: http://www.eurogamer.net/articles/eve-breaks-concurrent-user-record

[5] Kushner, D., Engineering EverQuest: Online Gaming Demands Heavyweight Data Centers. Spectrum, IEEE, 2005. 42(7): p. 34-39.

[6] Terdiman, D. 'Second Life': Don't worry, we can scale. ZDNet News (Personal Technology) 2006 June 6. [Online]. Available at: http://news.zdnet.com/2100-1040_22-148320.html.

[7] Brandt, D. Networking and Scalability in EVE Online. in Proceedings of 4th ACM SIGCOMM workshop on Network and system support for games. 2005. Hawthorne, New York, USA: ACM.

[8] Delaney, D., T. Ward, and S. McLoone, On Consistency and Network Latency in Distributed Interactive Applications: A Survey. Presence: Teleoperators & Virtual Environments, 2006. 15(2): p. 218–234.

[9] Capps, M., McDowell, P., & Zyda, M. A Future for Entertainment-Defense Research Collaboration. IEEE Computer Graphics and Applications, 2001. 21(1) :p. 37–43.

[10] Stytz, M. R. Distributed Virtual Environments. IEEE Computer Graphics and Applications, 1996. 16(3):p. 19–31.

[11] Singhal, S. and Zyda, M., Networked Virtual Environments: Design and Implementation (Siggraph Series). 1999, New York: ACM Press.

[12] Park, K. S., & Kenyon, R. V. Effects of Network Characteristics on Human Performance in a Collaborative Virtual Environment. In Proceedings of Virtual Reality (VR '99), 1999.

[13] Vaghi, I., Greenhalgh, C., & Benford, S. Coping with Inconsistency Due to Network Delays in Collaborative Virtual Environments. In Proceedings of Symposium on Virtual Reality Software and Technology (VRST '99), 1999.

[14] Benford, S., et al., Collaborative Virtual Environments. Communication of the ACM, 2001. 44(7): p. 79-85.

[15] Smed, J., T. Kaukoranta, and H. Hakonen, A Review on Networking and Multiplayer Computer Games, in TUCS Technical Report No 454. 2002.

[16] Frecon, E. and M. Stenius, DIVE: A Scalable Network Architecture for Distributed Virtual Environments. Distributed Systems Engineering Journal, 1998. 5:p. 91-100.

[17] Greenhalgh, C., J. Purbrick, and D. Snowdon. Inside Massive-3: Flexible Support for Data Consistency and World Structuring. in Proceedings of the Third ACM Conference on Collaborative Virtual Environments (CVE 2000). 2000: ACM.

[18] Linden Research. Second Life. [Online]. Available at: http://secondlife.com

[19] Smith, D.A., et al. Croquet - A Collaboration System Architecture. in Proceedings of First Conference on Creating, Connecting and Collaborating through Computing. 2003. Kyoto, Japan.

[20] Electronic Arts. Ultima Online. [Online]. Available at: http://www.uo.com

[21] Sony Entertainment. Everquest 2. [Online]. Available at: http://everquest2.station.sony.com.

[22] Valve Software. Counter-Strike: Source . [Online]. Available at: http://www.valvesoftware.com

[23] ID Software. Quake 4. [Online]. Available at: http://www.idsoftware.com/games/quake/quake4

[24] NovaLogic. Joint Operations: Combined Arms. [Online]. Available at: http://www.novalogic.com

[25] Sony. PlanetSide. [Online]. Available at: http://planetside.station.sony.com

[26] Electronic Arts. Command & Conquer Generals. [Online]. Available at: http://www.commandandconquer.com

[27] CDV Software Entertainment AG and GSC Game World. Cossacks game series. [Online]. Available at: http://www.cossacks2.de

[28] Smed, J., T. Kaukoranta, and H. Hakonen. Aspects of Networking in Multiplayer Computer Games. in International Conference on Application and Development of Computer Games in the 21st Century. 2001.

[29] Aronson, J., Dead Reckoning: Latency Hiding for Networked Games. 1997, Gamasutra. [Online]. Available at: http://www.gamasutra.com/features/19970919/aronson_01.htm

[30] Lincroft, P., The Internet Sucks: Or, What I Learned Coding X-Wing vs. TIE Fighter. 1999, Gamasutra. [Online]. Available at: http://www.gamasutra.com/features/19990903/lincroft_01.htm

[31] Pantel, L. and L.C. Wolf. Network Issues for Video and Games: On the Impact of Delay on Real-Time Multiplayer Games. in 12th International Workshop on Network and Operating Systems Support for Digital Audio and Video. 2002. Miami, FL: ACM Press.

[32] IMG News, DiabloII.Net Recaps Real Issues. in Inside Mac Games. 2002. [Online]. Available at: http://www.insidemacgames.com/news/story.php?ArticleID=4741

[33] Kirmse, A. and C. Kirmse, Security in Online Games. 1997. [Online]. Available at: http://www.gamasutra.com/features/19970707/security.htm

[34] Shachtman, N., 'Blizzard' of Cheaters Banned. Wired News. [Online]. Available at: http://www.wired.com/gaming/gamingreviews/news/2002/09/55092

[35] Sung, M., M. Gleicher, and S. Chenney. Scalable Behaviors for Crowd Simulation. in Computer Graphics Forum 23, (Eurographics '04). 2004.

[36] [56] Treuille, A., S. Cooper, and Z. Popovic, Continuum Crowds. ACM Transactions on Graphics, 2006. 25(3): p. 1160-1168.

[37] Morse, K.L., Bic, L. and Dillencourt, M., Interest Management in Large-Scale Virtual Environments. Presence: Teleoperators & Virtual Environments, 2000. 9(1): p. 52-68.

[38] Tanenbaum, A. and M.v. Steen, Distributed Systems: Principles and Paradigms (2nd Edition) 2007, Upper Saddle River, New Jersey: Pearson Education Inc.

[39] Forouzan, B.A., TCP/IP Protocol Suite (3rd Edition). 2007: McGraw-Hill.

[40] Zou, L., M.H. Ammar, and C. Diot. An Evaluation of Grouping Techniques for State Dissemination in Networked Multiuser Games. in Proceedings of the ninth Internation Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems. 2001.

[41] Cecin, F. R. et. al., Freemmg: A Hybrid Peer-to-Peer and Client-Server Model for Massively Multiplayer Games. In Proceedings of ACM SIGCOMM 2004 workshops on NetGames '04, 2004.

[42] Jenkins, K. Designing Secure, Flexible, and High Performance Game Network Architectures, December 2004, Gamasutra. [Online]. Available at: http://www.gamasutra.com/view/feature/2174/designing_secure_flexible_and_.php

[43] Zhuang, S. Q., Zhao, B. Y., and Joseph, A. D. Bayeux: An Architecture for Scalable and Fault-tolerant Wide-area Data Dissemination. In Proceedings of NOSSDAV'01, Jun. 2001.

[44] Zaichen, Z. and V.O.K. Li, Network-Supported Layered Multicast Transport Control for Streaming Media. Parallel and Distributed Systems, IEEE Transactions on, 18(9): p. 1332-1344, 2007.

[45] GarageGames Inc., Torque Network Library (TNL), [Online]. Available at: http://www.opentnl.org

[46] Mauve, M., Fischer, S., and Widmer, J., A Generic Proxy System for Networked Computer Games, in Proceedings of the 1st Workshop on Network and System Support for Games, 2002.

[47] Lambright, R., Distributing Object State for Networked Games Using Object Views. September 2002, Gamasutra. [Online]. Available at: http://www.gamasutra.com/resource_guide/20020916/lambright_01.htm

[48] Masa, M. and Zara, J. Interest Management for Collaborative Environments Through Dividing Their Shared State. In Cooperative Design, Visualization, and Engineering, volume 3190 of Lecture Notes in Computer Science, pages 156–163. Springer-Verlag, 2004.

[49] Jenkins, K. Designing Secure, Flexible, and High Performance Game Network Architectures, December 2004, Gamasutra. [Online]. Available at: http://www.gamasutra.com/view/feature/2174/designing_secure_flexible_and_.php

[50] Yan, J. and Randell, B. A Systematic Classification of Cheating in Online Games. In NetGames '05: Proceedings of 4rd ACM SIGCOMM Workshop on Network and System Support for Games. ACM Press, 2005.

[51] Bauer, D., Iliadis, I., and Scotton, P., Communication Architectures for Massive Multi-Player Games. Multimedia Tools and Applications, 23:47–66, 2004.

[52] Bettner, P. and Terrano, M. 1500 Archers on A 28.8: Network Programming in Age of Empires and Beyond. In Game Developer Conference Proceedings, San Jose, California, USA, March 2001.

[53] Ensemble Studios. Age of Empires. [Online]. Available at: http://www.microsoft.com/games/empires

[54] Diot, C. and Gautier, L. A Distributed Architecture for Multiplayer Interactive Applications on the Internet. IEEE Networks magazine, 13(4), July/August 1999.

[55] Gautier, L. and Diot, C. Design and Evaluation of MiMaze, a Multi-player Game on the Internet. In International Conference on Multimedia Computing and Systems, pages 233–236, 1998.

[56] Knutsson, B., et al., Peer-to-peer Support for Massively Multiplayer Games. In Proceedings of IEEE Infocomm, March 2004.

[57] Boukerche, A. Araujo, R.B. and Laffranchi, M. Multiuser 3D Virtual Simulation Environments Support in the Gnutella Peer-to-Peer Network. Journal of Parallel and Distributed Computing, 65(11):1462–1469, 2005.

[58] Yu, A. 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 System Support for Digital Audio and Video, 2005.

[59] Wentong, C., et al. A Scalable Architecture for Supporting Interactive Games on the Internet. in Proceedings of the 16th Parallel and Distributed Simulation, 2002.

[60] Dibbell, J., What's in a Shard?, in Terra Nova "Blogs". 2004 April 2. [Online]. Available at: http://terranova.blogs.com/terra_nova/2004/04/whats_in_a_shar.html

[61] Frank, G., et al., RTF: A Real-Time Framework for Developing Scalable Multiplayer Online Games, in Proceedings of the 6th ACM SIGCOMM workshop on Network and system support for games. 2007.