Architecting Scalability for Massively Multiplayer Online Gaming Experiences

Rui Gil
CISUC, University of Coimbra
Polo II – Universidade de Coimbra, 3030-290 Coimbra, Portugal
+351-23979000
rgil@dei.uc.pt

José Pedro Tavares
CISUC, University of Coimbra
Polo II – Universidade de Coimbra, 3030-290 Coimbra, Portugal
+351-23979000
jtavares@dei.uc.pt

Licinio Roque
CISUC, University of Coimbra
Polo II – Universidade de Coimbra, 3030-290 Coimbra, Portugal
+351-23979000
lir@dei.uc.pt

ABSTRACT
In this study, the authors propose to discuss scalability challenges posed by Massively Multiplayer Online Game (MMOG) scenarios, while outlining issues specific to the context of online interactive experiences and game genres. These scalability issues concern: game simulation, content distribution, communication and coordination, and structural scalability. The authors present a critical review of approaches to known issues and outline research goals for an integrated scalability approach, to achieve a balanced, general purpose design, for MMOG infrastructure.

Keywords
Massively Multiplayer Online Game (MMOG), Scalability, system design, system architecture

INTRODUCTION
The emergence of the Internet made possible the development of gaming systems that combine multiple computer resources and that can be played simultaneously by thousands of users from anywhere in the world. The massive use of these systems introduces scalability issues that must be considered at design phase. The scalability level of these systems is perceived by their ability to manage increasing interaction demands without significantly degrading the quality of the playing experience. Scalability appears related with internal system characteristics but also with the specific demands of the context of use, and, as such, a general solution scalable for all

Proceedings of DiGRA 2005 Conference: Changing Views – Worlds in Play.

interactive contexts may be hard or impossible to achieve.

Massively Multiplayer Online Games (MMOGs) are created with the objective of a massive use and its scalability challenges must be approached according to its specificity. E.g. the demands of the First Person Shooter genre of games are very different from those of a turn-based strategy game. The authors are involved in the design of a general infrastructure for MMOG development and it is in the context of that goal game that they select the research questions currently being addressed: Can we design a general and scalable infrastructure for MMOG development that can support diverse game genres? What its architecture should be? What main components and characteristics?

With the previous research questions in mind, the purpose of this article is to identify the scalability issues in the design of MMOG. We will discuss the meaning of scalability for MMOG and what we think the main challenges are. Next, we present each of the scalability issues and their possible solutions by going through the relevant state of the art. The description of individual issues is followed by a brief discussion of how they are related; and how these dependencies can limit the global system scalability and the effective design of a general MMOG infrastructure. Finally, there is a brief description of future work

SCALABILITY FOR MMOG INFRASTRUCTURE

The appearance of networking environments (Internet and LANs) provided a platform for the development of gaming systems that can be used by many players simultaneously. The Internet was the main vehicle for the generalized adoption of this kind of systems, making it possible for a large number of users to simultaneously and cost-effectively access these systems, and for designers to consider the development of massively multiplayer gaming experiences, and thus raising the question of scalability. What is the adequate system design and implementation that supports a certain kind of performance to a growing number of users while keeping the perceived quality of service at enjoyable levels? We think that the possible answers to this apparently subjective question depend on the objective analysis of the various system scalability issues.

By scalability it is understood the ability that a system has to respond to increasing demand from its users, without significantly degrading the quality of the interactive experience. Thus, quality of service degradation in a reasonable and controlled way, the elimination of performance bottlenecks, the expandability of system resources to cope with demand and technology updating to avoid becoming obsolete, the maintenance and system upgrading cost control are all challenges in the design of a scalable solution [1][2][3].

The Internet itself can be invoked as an example of massive use and explosive growth, in the number of components and users, where good scalability examples can be found. The Domain Name System (DNS) – the Internet distributed directory service that has the fundamental role of translating hostnames to IP addresses [1], is a good example of how a system can be made scalable despite the massive growth, keeping maintenance costs and complexity under control. On the other hand, the IPv4 address system is a good example of how a system design became problematic due to massive demand. IPv6, while resolving the addressing problem in a satisfactory way, brings significant transition costs and complex management by the goal of legacy interoperability [4]. The IPv4 example also teaches us not to underestimate the risk of obsolescence.

Scalability is a system characteristic that must be considered at the design phase of MMOG infrastructure, possibly involving all system components and interactions, and their dependencies. Overall system scalability may not be achieved if only some components of the system are taken into consideration, as their performance may be hindered by dependencies on other less scalable components. Therefore, we should analyze the scalability issues at each component of the system and the scalability issues of the system as an aggregate.

The designer should also be aware of the diverse types of constraints and demands involved. For instance, a system that has a high processing scalability and low communication scalability may present a poor overall scalability. Throughout the work in system scalability levels [2][5][6][7][8] the nature of the problem influences the definition of scalability itself. A solution for a given problem situation may not be valid for another problem situation, since the relevant measures or indicators of scalability may change with the situation. E.g., system requirements for the rapid movement and update rates of a "First Person Shooter" differ substantially from turn-based multiplayer strategy game like Civilization.

In order to identify the diverse scalability challenges that we can find in the development of a MMOG, we must understand the structure and dynamics of this kind of systems and their interactive contexts, induced by the various game genres. Although there are significant differences in game dynamics, we intent to analyze a set of relevant scalability issues among the various kinds of MMOG – massive multi-player online role playing games (MMORPG), massive multi-player online real-time strategy (MMORTS), massive multiplayer online first-person shooters (MMOFPS) and virtual environments (VE). This is becomes especially relevant if we want to design a generic infrastructure for supporting different MMOG genres or mix.

Typically, in these systems, the action occurs in a 3D virtual environment, where each player controls a virtual character or avatar or a set of interactive entities within the semantics of the game. Users interact with the virtual world in what can be perceived as real time allowing for a game experience with a high updating frequency. The action universe may be persistent, preserving the state of the world between accesses, providing a sensation of continuity. UltimaOnline [9], EverQuest II [10], World of Warcraft [11] and SecondLife [12] are examples of this kind of games.

Taking the characteristics of this kind of games as prototypical we will try to define the main scalability issues that these systems must consider along the following abstract components:

- game state simulation component that allows thousands of players to share the same virtual event or interactive context;
- storage capacity for all the data that is used to represent the virtual environment (data model and multimedia elements) and its efficient and timely distribution;
- communication performance that enables the interactive system to maintain the quality and coordination of the game experience;
- an overall architecture that enables the integration of new components, resources and technologies for system expandability.

Next, we will present at discussion of these issues and the state of the art relevant to approach these requirements. Whenever solutions are presented, the evaluation of economic implications will be postponed for a future evaluation.

MMOG SIMULATION ISSUES

MMOG simulation is the task of processing events that occur from players' interactions and automatically generated events that make the game experience more realistic (e.g. weather conditions, Non-Player Characters) while maintaining data consistency and, in some games, shared state persistence. The nature of such a system, where thousands of players interact, needs a significant processing capacity for simulating the virtual environment.

Besides the huge number of events to process, a simulation module faces another problem, the size of the data state that models the virtual world representation. The area where the game action occurs might be equivalent to the size of a planet, so, managing the data and simulation model for the virtual environment can be a very complex task. The size and complexity of such a system creates various potential scalability problems – processing, management, persistence and data storage – we are going to present and discuss some possible approaches to its resolution. Although the majority of the existent Internet systems are based on the Client/Server architecture, there is another type of architecture that will be taken in consideration, the Peer-to-Peer architecture.

Client/Server Architecture

In the Client/Server architectural model there are two types of components: a centralized Server and multiples clients. The server can be constituted by one or multiples machines that process and manage the simulation state. Clients are the means to interact with the server, which in the case of MMOG games tends to be an interface to a 3D virtual environment [1]. The more common scalability issues are evident as it is difficult to keep processing power expanding to keep pace with simulation demand. A possible approach to overcome processing difficulties within the Client/Server architecture, is to setup a *server farm* of simulation servers where a part or sector of the game universe is attributed to each one [13][14][15]. The partition of the virtual environment into small areas is done taking into attention factors like the area size and predictable usage level based on statistics. With this mechanism the scalability in event processing is achieved because each server only processes events related to its game area. In the case of data model management scalability is also achieved since this is structurally partitioned through the various servers, with each server managing the relevant data set.

This solution presents as main challenges: finding a general or algorithmic way to divide the virtual environment into sectors and the overhead in synchronization efforts to keep the simulation data between different server mirrors in order to maintain coherence. Any arbitrary subdivision can face the challenges of possible social mob behaviors that can occur when a specific event is promoted or becomes too popular. In such cases a great number of players can attempt to use a very small number of partitions instead of spreading throughout the game universe. Characters and other entities can move among sectors that are being simulated in different servers further complicating coherence efforts. In this simulation model there are also some possible optimizations to enhance the performance, e.g., the introduction of specialized servers to perform tasks like calculating world physics.

Still within the scope of Client/Server architecture is the use of Grid Computing techniques for accessing processing power. In this case, the server is built by a computer grid where the simulation of the virtual world distributed transparently. An example of this approach is IBM's Butterfly.net, a game distribution platform [16]. In this architecture, the processing of game events proceeds in parallel on several machines, having as base a common simulation logic that

can be defined according to some metrics like the event processing priority [16] [17]. The adaptation to game processing necessities is one of the main advantages, because resource allocation in the grid is made according to ongoing processing needs. The main challenge becomes one of load balancing to assure an optimal available resource management and in the synchronization of the simulation state among the several server units to keep consistency.

Peer-to-Peer Architecture

Despite its obvious scalability problems client/server architectures are still the most common in current MMOG implementations. Peer-to-Peer architectures are gaining ground as more experiences are made and some of its problems are overcome. Lately, various distributed systems emerged based in this architecture, with file sharing as the most common application [18] [19]. The original idea for Peer-to-Peer architecture is based on the concept of a system where all components are treated as equally important and thus avoiding the scenario where a specific resource becomes more critical and the overall system performance dependent on it [20]. Another main characteristic of Peer-to-Peer architecture is the design that tries to take advantage of the existing resources available on the Peers' – load or processing capacity, bandwidth capacity and storage capacity – and use them for common benefit.

One of the possible solutions to achieve *simulation scalability* is again based on the game space subdivision. Each of resulting game zones is attributed to a peer that became responsible for simulating the events that occur in that zone [21] thus taking advantage of each peers' load capacity. In this approach some of the challenges we already referred when we discussed Client/Server architecture also apply and others emerge. Firstly, in the process of attributing a game zone to a peer, it is important to take in consideration some metric of load capacity, available memory and network bandwidth. If not, game simulation could be attributed to a peer that doesn't have the necessary resources thus damaging the quality of game experience for all players in the sector. Other challenges are inherent to Peer-to-Peer architecture and relate to the unpredictability that is characteristic of the peer-to-peer networks, with the arbitrary entering and leaving of peers. While we can manage to have backup servers to compensate for those leaving, this characteristic introduces a new problem in keeping simulation object addressing updated and simulation data persistent. There is already some work being done in this area, e.g. [22][23][24].

Another solution that is based on the Peer-to-Peer architecture and is more radical than the last one, is to have no central simulation. In this case we would need a distributed simulation model where all peers would have to virtually simulate the space where the player is interacting [25][26]. Each peered simulation would process the events that are produced by all players, in order to maintain consistency. This solution approach would potentially increase the complexity of the synchronization mechanisms and the resolution of possible inconsistencies that could occur between simulation states. Main limitations are the possibility of a peer to lag behind if it does not have the necessary capacity to simulate the game space and the potential for fraude by forging events or simulation states. Further readings on Peer-to-Peer approach to MMOG design can be found at based in [27].

COMMUNICATION AND COORDINATION

In Internet real-time interaction games, communication scalability is one of the main issues that can hinder the gaming experience. By communication scalability we mean not only the adaptive capacity to support the communication between a growing numbers of users, but also the capacity to maintain a communication performance that not hinder the game experience. In the

MMOG context this scalability aspect is paradigmatic, because any massively multiplayer space will harbor thousands of players which are permanently originating information that has to be timely delivered to a changing set of destinies over a heterogeneous and quite unpredictable network infrastructure. Improper treatment of this issue can lead to situations where a single new player entering the game can drag the overall experience to a halt. Depending in specific design options the state of all objects as well as the events originating from player activity must be propagated to others players to maintain perceived game coherence. The approach we adopted to analyze this scalability issue was, firstly, to study forms to reduce communication volume and frequency and then study mechanism to increase the communication performance and predictability.

Reducing Communication Volume and Frequency

The objects that compose the virtual environment are the first point of analysis. We can consider two main object groups in the simulation of a game: objects that have a deterministic behavior and the objects that have a non-deterministic behavior. In the non-deterministic case and because is necessary to maintain coherence, we need to synchronize the object state in all clients. In the deterministic case we can assume that object instances departing from similar states process events in a similar form leaving a similar final state. Thus, if we can determine the next object state, why not do it in each peer or client? With this method we would reduce significantly the volume of information that is necessary to communicate object states because a big part of the simulation state could be made of deterministic objects like the ones that we use to create the environment, landscapes, trees, lights, building, furniture, etc.

In non-deterministic objects we can divide the information that we need to propagate two types: the one that can't be lost in transit because it will put in danger game consistency; and the information that, because of the high update frequency expires easily, and is lost could be discarded. An example of this type of information is the events from certain player movements in the virtual environment. Although a best effort should be made to communicate it player movement typically generates updating events at reasonable high frequency. The loss of part of these events can easily be compensated by trajectory interpolation and extrapolation significantly decreasing communication without a loss in quality of the game experience [30]. Since a big part of the events that occur in MMOG report character movement it is important to adopt some filtering mechanism to reduce the possibility of network overload, preferably an adaptive one that considers current network performance.

As previously referred, a virtual environment can become quite big, and a significant part of it is not relevant to the action context of a player at a certain time. Therefore, each player does not need to receive notice of all the changes that are happening everywhere else in the virtual environment. In agreement with what has been referred in the manner of dividing the simulation data space, it is possible to use techniques to determine the information that is relevant for updating each player's current experience. The decision of which events or other information is relevant to a player at a certain moment is based on metrics like virtual proximity, field of vision, action context, etc. [15] [31][32]. Another approach to event filtering is the Public-Subscribe communication model. The functioning of this type of mechanism is quite simple to implement and is based on the subscription and unsubscription of distribution channels according to relevance or interest in the information that circulates in them. With this mechanism the decision of which information is relevant to whom, is actively made by the receptor [33] [34].

The reduction of the number of events circulating, besides contributing to a reduction of the information volume, also provides a reduction on the number of events that need to be processed and a reduction of the dependency on latency and on transitory network congestion situations,.

Optimizing Communications

In spite previous ideas aimed at traffic reduction, some of that traffic becomes inevitable, therefore we are going to present some points were we can try to optimize this "inevitable" communication. The first point is what communication protocols should be adopted and in which circumstances. TCP/IP and UDP/IP protocols are the ones that are currently widely used in the Internet. The main differences between the two are that UDP is generally faster than TCP for small communications since it requires less control; the TCP protocol guarantees the delivery of the data which UDP doesn't; and the TCP protocol keeps state and session, guaranteeing the information order and if the connections are still active [1].

These basic characteristics make these protocols more or less adapted to our situation according to the type of data that we need to transmit and the communication architecture, i.e., connection topology. E.g., since some player movement events can be discarded and occupy little coding space they are best fit to take advantage of the UDP protocol characteristics, resulting, in a communication reduction and in an increased sense of flow in the game. On another hand, the use of TCP protocol is advantageous when the information that is being sent must be delivered. TPC protocol assures the order and delivery of the information, reducing the complexity of implementing MMOG communications. A main problem with TCP connections is due to the session-based nature which takes resources that can be limited and thus becomes a scalability issue both in servers and in peers. E.g., it is common for a PC operating system to limit the number of active connections to a few dozen or a few hundred, thus limiting the number of peers that can be cross-connected at a given time. In some designs this makes it better to use UDP protocol complemented with delivery verification and message ordering algorithms [35].

The use of compacting and correspondence dictionary methods is another way to reduce the size of the information to be transmitted. The choice of the data compression algorithms should take into account the quality of the compression but also on the relation between the bandwidth and latency gain in communication and the time and computational resources spent on their compression and decompression. Ideally, these mechanisms should be made to adapt to the environment were they are working, e.g., with the adoption of load bandwidth profiling [36].

Finally, the point where communication optimizations are made (or possible) is related with the architectural model. In a Peer-to-Peer architecture all management and optimization strategies are made in the peers, which decide what should be communicated and to whom. In a pure Client/Server architecture all event traffic goes through the servers, where the decision to whom to retransmit to can be is made, possibly taking advantage of the optimization strategies previously described. Although the Client/Server architecture is essentially centralized, some optimizations can be made in information communication without it all going to the server [37]. The dialog between two users is an example of the information that can be directly sent between two clients, reducing the servers load.

CONTENT DISTRIBUTION

MMOGs, as said earlier, are typically becoming 3D environments where players interact. These environments can become quiet dynamic, making it impossible for the clients to save the virtual

world state hoping that when the player returns the game state remains the same. Therefore it is necessary to provide the client with all that is necessary to render the scenes when the player enters the virtual world. We can partition that information in two kinds: the data model that represents the virtual world and the multimedia elements that are needed to render it (visually, audibly and kinesthetically). The transmission of the multimedia elements is the most problematic, since they typically represent more storage space. So, combining the data volume to the possibility of massive download accesses, we find yet another scalability issue we must consider when distributing these media online.

We think that a solution to this scalability problem involves not only an increase in data transfer ratios, but also by being selective about which information to transfer. I.e., a faster transfer rates are not enough and increase costs, it is also necessary to prioritize data, based on urgency or relevance to the current or future context (which requires some form of predictability). This made us look into the possible integration of caching and pre-fetching techniques for improving the content distribution and management, which we will present and relate to how they impart on *distribution scalability*.

A comparison is possible with web caching systems currently in use and that operate at the URL level [28]. This process can be made more efficient if it is made at the elements that compose a page, so if the same multimedia element is reused in a different page it can be reused from cache. For this to become possible the elements must be univocally identified (in the www case, e.g., by reusing an image from the same URL). Caching of multimedia elements used to render virtual worlds is a possible solution to adopt in a MMOG content management and distribution. This solution adapts better in this kind of systems, due to the fact that the content reuse level (e.g. images, sounds, 3D model) can potentially be much bigger than in web pages. For these systems to work best the media elements must be identified univocally (e.g. through file digest or universal IDs). Cache management on the client side can be done according to the degree of usage: locally, by the player, and globally, by the player community.

Media elements could also be pre-fetched at certain moments while other gaming activities take place. The main goal would be to try to anticipate the future needs of the player and avoid or significantly reduce delays in transitions between game levels or scenes. This process can start when the player already has all the multimedia elements needed to render the current space, level or scene. So, taking advantage of marginal bandwidth, the client could begin downloading media elements that, predictably, will be needed. Techniques like the recognizing social patterns of behavior and community usage ratios are starting points to determine what are the next priority elements [28] [29].

Caching and pre-fetching techniques can minimize the amount of network traffic between player accesses and pull prioritized elements making them readily available when needed. For these mechanisms to work we need services to store and deliver these media elements. A possible approach is to implement them as several servers making the media elements available. A general directory service can then be used to translate media IDs to the possible URLs to access them. A mirroring approach could be used for increased availability, in which case an indexation service can translate a request for media elements and respond by using scalability relevant metrics such as: server load, bandwidth and the geographic localization [28] [29].

The file sharing Peer-to-Peer systems are also a good example of an effective world wide content

sharing system [18] [19] that can inspire us in the pursuit of scalable distribution methods. Based on the Peer-to-Peer approach and taking advantage of a service for univocally identification of the multimedia elements, we can have peers distributing directly among themselves, thus unloading the servers. The success of this solution is conditioned by a low level of content replications, scenario that, typically, only exists at the beginning of the distribution. In conclusion, we think that an high performance directory service associated with an efficient Peer-to-Peer distribution method is a scenario worth exploring to tackle the issue of scalability in online distribution contexts.

STRUCTURAL SCALABILITY

The structural scalability issue has three aspects: scalability through the incorporation of new resources (processing capacity, memory, bandwidth and storage); architectural scalability through the updating of components; and scalability through the extension of system functionalities and by exploiting new technologies.

Scalability through the incorporation of new resources is an issue that has to be thought of in a system lifetime perspective. When a MMOG is being implemented and deployed the decision on the resources being allocated is based on conservative usage level estimates, the assumption of certain usage patterns and cost control issues. But contingency scenario should also be considered in which the system has to respond to an exponentially increasing number of users. In that case the acquisition and deployment of new resources may be a solution if the system was designed to support that kind of structural scalability. Therefore, the system is thought to be as scalable as possible to the inclusion of these new resources, taking total advantage of them. In this way, depending of the system usage level it is possible to add or remove resources maintaining an appropriate quality of service. The Butterfly.net grid platform is an example of system designed with this scalability issue in mind [16].

The architecture of the system can also be thought of as a structural scalability factor. A scalable architecture is one that makes component redesigning or updating consistent with the overall design. A design based on a separation of concerns, e. g., simulation, communication middleware, graphic interface, storage management, complemented with interface definition (e.g. API) for interoperation between the components can make a gaming engine system more scalable in its future development and updating.

The third aspect of structural scalability is the integration of new system functionalities and new technologies. The goal of this point is to prevent the system from technological or functional stagnation due to the incapacity to extend to new platforms or to incorporate new functionalities. Factors like generality of the underlying abstractions and specifications, e.g. the data model for representing virtual worlds, the dependency of graphics engine implementation to the hosting platform, etc., must be accounted for when a MMOG infrastructure is being developed. We think this aspect of structure scalability is not adequately resolved a MMOG lifetime and commercial success could be significantly reduced by platform displacement.

CONLUSION: AN INTEGRATED APPROACH TO SCALABILITY

Considering the scalability issues previously discussed we now propose an approach to the challenge of balancing these for the purpose of designing a generic MMOG infrastructure. For this integrated approach we will need to analyze the dependencies between these requirements. We have taken as the basis for this approach a Peer-to-Peer architecture with a distributed

simulation model where each peer computes, locally, the game arena where the player interacts. Arena is the structural element that delimits the context of action and the virtual environment is created by an hypertext of interactive arenas were the game action takes place. Links between arenas represent possible moves or transitions between arenas. A distributed peer-to-peer simulation brings clear advantages by lowering the cost of centralized infrastructures and by distributing the processing load to where it is benefited from, while complexity increases for maintaining the consistency and persistence of virtual world.

With the adoption of such a model the dependency on network performance increases because it is necessary to communicate a high volume of information between players in order to maintain simulations consistent. Thus, techniques as the extrapolation and interpolation of players' movement, and event filtering based on the relevance of the information to the player's action context are needed to limit communication volume. A live profile of communication latency and packet loss between peers could be used to monitor the dynamics of network characteristics such as "communication proximity" between peers.

Consistency between simulations could be maintained using a conflict detection and resolution mechanism for probable conflicts (e.g. two players catching one object at the same virtual time). Conflict resolution could be done by having a legitimizing simulation peer. Solutions like distributed voting methods don't seem to fit well with MMOG contexts because they increase the network traffic and significantly delay conflict resolution. To safeguard scalability in conflict resolution an adequate method is necessary to select the legitimizing simulation, possibly a random selection weighted by system availability.

A main challenge with this approach is to guarantee virtual environment persistence in a scalable way. As previously referred, information that is used to represent the virtual world could become very dynamic, specially is players actions are able to effect persistent changes in it. This case makes it impossible for players to keep its state locally, between accesses. Under this model a service designed to assure persistence, availability and distribution of updated simulation state becomes a necessity. Persistence can be achieved through special purpose servers that keep the changeable part of the state of the virtual environment or through peer-to-peer distribution.

As for content distribution our approach will combine a centrally managed content server for media registration and networks injection point, with a peer-to-peer distribution method. With these mechanisms will take advantage of the storage and bandwidth capabilities available in the peers. In order to achieve timely availability of content it will be necessary to implement caching and pre-fetching methods. Additionally, network use will have to strike a balance between game state updating and background content pre-fetching in order to avoid a deterioration of the game experience. The proposed architecture will be based on components with clear competences and interactions, implemented by a set of clearly defined APIs and protocols. Making it possible to develop and to update or adapt components independently.

These scalability issues and the integrated approach that has been present are now being studied through a process of prototyping of a general purpose MMOG infrastructure that is being designed to enable collaborative online creation of games by authors and players alike. In addition to enable an experimental study of the scalability issues, the platform is being designed with such characteristics as robustness, low cost and fault tolerance. The authors expect to be able to disclose general system architecture and detailed design in a near future.

REFERENCES

1. Coulouris, G., J. Dollimore, and T. Kindberg, Distributed System: Concepts and Design, Addison-Wesley, England, 2001.
2. P. Jogalekar, “Evaluating the Scalability of Distributed Systems”, IEEE Transactions on Parallel and Distributed Systems, 2002 IEEE, Vol. 11, No. 6, June 2000, pp. 589-603.
3. A.B. Bomdi, “Characteristics of Scalability and Their Impact on Performance”, Workshop on Software and Performance:Proceedings of the second international workshop on Software and performance, ACM Press, Ottowa Canada, 2000, pp. 195 - 203.
4. A.B. Bomdi, “Realizing the Transition to IPv6”, IEEE Communications Magazine, 2002 IEEE, June 2000.
5. A. Grama, A. Gupta, and V. Kumar, "Isoefficiency: Measuring The Scalability of Parallel Algorithms And Architectures," IEEE Parallel and Distributed Technology, vol. 1, no. 3, August 1993, pp. 12-21.
6. C. Yoshikawa, B. Chun, P. Eastham, A. Vahdat, T. Anderson and D. Culler, “Using Smart Clients to Build Scalable Services", Internal report, Computer Science Division, University of California, Berkeley. Available at http://www.now.cs.berkeley/edu/SmartClients
7. J. Howard, M. Kazar, S. Menees, D. Nichols, M. Satyanarayanan, R. Sidebotham and M. West. “Scale and Performance in a Distributed File System”, ACM Transactions on Computer Systems, ACM Press, February 1988, p.p 51–82.
8. F. Sheikh, J. Rolia, P. Garg, S. Frolund and A. Shepherd, “Performance Evaluation of a Large Scale Distributed Application Design", World Congress on Systems Simulation, Singapore, September 1997.
9. Electronic Arts, UlimaOnline; http://www.uo.com
10. Sony Online Entertainment INC., The EverQuest II homepage; http://everquest2.station.sony.com/
11. Blizzard Entertainment, The World of Warcraft homepage; http://www.worldofwarcraft.com/
12. Linden Research, Inc., Second Life; http://secondlife.com/
13. P. Rosedale and C. Ondrejka, “Enabling Player-Created Online Worlds with Grid Computing and Streaming”, September 2003, Available at: http://www.gamasutra.com/resource_guide/20030916/rosedale_01.shtml
14. Zona Inc. “Terazona: Zona application frame work whitepaper” (2002) Available at www.zona.net/whitepaper/ Zonawhitepaper.pdf
15. L. Aarhus, K. Holmqvist and M. Kirkengen, “Generalized Two-Tier Relevance Filtering of Computer Game Update Events”, Proceedings of the first workshop on Network and system support for games, ACM Press, April 2002, pp. 10-13.
16. Butterfly.net Inc., The butterfly grid: Powering Next-Generation Gaming with On-Demand Computing (2003), Available at www.ibm.com/grid/pdf/butterfly.pdf
17. – D. Saha, S. Sahu, A. Shaikh, “A service Platform for On-Line Games”, NetGames03, ACM Press, May 2003.
18. Napster.com Inc., Napster; http://www.napster.com
19. Gnutella.com Inc., Gnutella; http://www.gnutella.com/
20. Oram, A. , Peer-to-peer : harnessing the benefits of a disruptive technology, O'Reilly, 2001.
21. T. Limura, H. Hazeyama and Y. Kadobayshi, “Zoned Federation of Games Servers: a Peer-to-Peer Approch to Scalable Multi-player Online Games” SIGCOMM’04, ACM Press, August 2004.
22. I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek and Hari Balakrishnan, “Chord: a scalable peer-to-peer lookup protocol for internet applications”, IEEE/ACM Transactions on Networking, ACM Press, February 2003, V. 11 , pp. 17 – 32.
23. – B. Y. Zhao, J. Kubiatowicz, and A. D. Joseph, “Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing”, Internal Report, Computer Science Division, University of California, April 2001.
24. J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao “OceanStore: An Architecture for Global-Scale Persistent Storage”, ACM, 2000.
25. B. Knutsson, H. Lu, W. Xu and B. Hopkins, “Peer-to-Peer Support for Massively Multiplayer Games”, INFOCOM 2004, Hong Kong, China, March 2004.
26. – C. Diot and L. Gautier, “A Distributed Architecture for Multiplayer Interactive Applications on the Internet”, IEEE Networks magazine, IEEE, July/August 1999, pp. 6-15.
27. – C. GauthierDickey, “A Distibuted Architecture”, Available at: www.cs.umanitoba.ca/~maheswar/anc2002/ PAPERS/JoW00.pdf
28. Rabinovich, M. and O. Spartscheck, Web Caching and Replication, Addison-Wesley, USA, 2001.
29. T. Henderson and S. Bhatti, “Modelling user behaviour in networked games”, Proceedings of ACM Multimedia 2001, ACM Press, October 2001, pp. 212–220.
30. S. I. P., H. J. Shin, T. Kim, and S. Y. Shin, “On-line Motion Blending for Real-time Locomotion Generation”, Journal of Computer Animation and Virtual Worlds, V. 15, July 2004, pp. 125-138.

31. H. Abrams, K. Watsen, and M. Zyda, “Three-tiered interest management for large-scale virtual environments”, Proceedings of 1998 ACM Symposium on Virtual Reality Software and Technology (VRST’98), ACM Press, November 1998.
32. K. L. Morse, “Interest management in large-scale distributed Applications” Technical report, Department of Information & Computer Science, University of California, Irvine, 1996.
33. S. Fiedler, M. Wallner and M. Weber, “A communication Architecture for Massive ”, Proceedings of the first workshop on Network and system support for games, ACM Press, April 2002, pp. 14-22.
34. A. R. Bharambe, S. Rao and S. Seshan, “Mercury: A Scalable Publish-Subscribe System for Internet Games”, Proceedings of the first workshop on Network and system support for games, ACM Press, April 2002, pp. 3-9
35. S. Pack, E. Hong, Y. Choi, I. Park, J. S. Kim and D. Ko, “Game Transport Protocol: A Reliable Lightweight Transport Protocol for Massively Multiplayer On-line Games” Internal Report, 2001, Korea. Available at: http://mmlab.snu.ac.kr/publications/docs/itcom(shpack).pdf
36. J. Smed, T. Kaukorante and H. Hakonen, “Aspects of Networking in Multiplayer Computer Games”, Proceedings of International Conference on Application and Development of Computer Games in the 21st Century, Hong Kong SAR, China, November 2001, pp. 74-88.
37. P. Bettner and M. Terrano, “GDC 2001: 1500 Archers on a 28.8: Network Programming in Age of Empires and Beyond” 22 March 2001; Available at: http://www.gamasutra.com/features/20010322/terrano_01.htm