Hindawi Publishing Corporation
International Journal of Computer Games Technology
Volume 2008, Article ID 594313, 9 pages
doi:10.1155/2008/594313

Research Article

ALVIC versus the Internet: Redesigning a Networked Virtual Environment Architecture

Peter Quax, Jeroen Dierckx, Bart Cornelissen, and Wim Lamotte

Expertise Center For Digital Media, tUL, IBBT, Hasselt University, Wetenschapspark 2, 3590 Diepenbeek, Belgium

Correspondence should be addressed to Peter Quax, peter.quax@uhasselt.be

Received 31 January 2008; Revised 25 April 2008; Accepted 16 June 2008

Recommended by Jouni Smed

The explosive growth of the number of applications based on networked virtual environment technology, both games and virtual communities, shows that these types of applications have become commonplace in a short period of time. However, from a research point of view, the inherent weaknesses in their architectures are quickly exposed. The Architecture for Large-Scale Virtual Interactive Communities (ALVICs) was originally developed to serve as a generic framework to deploy networked virtual environment applications on the Internet. While it has been shown to effectively scale to the numbers originally put forward, our findings have shown that, on a real-life network, such as the Internet, several drawbacks will not be overcome in the near future. It is, therefore, that we have recently started with the development of ALVIC-NG, which, while incorporating the findings from our previous research, makes several improvements on the original version, making it suitable for deployment on the Internet as it exists today.

1. INTRODUCTION

Witness the media attention that applications such as Second Life [1] and There [2] have gathered, it should be obvious that the technology behind them is an important subject of study. Indeed, historically, many architectures have been proposed, that were designed from the ground up to be scalable. In modern day's terms, however, the figures that can be attained are not exactly state of the art. Looking at examples of [3] or [4], the authors point out that architectures scale up to a maximum of tens of users. Comparing this to a modern-day application such as Second Life, which claims to have around 1.3 million active residents (a subject of debate), or World of Warcraft, with around 10 million active subscribers, these numbers indeed do seem ridiculously low.

Once the architecture behind these success stories is exposed, however, it quickly becomes clear how the developers have tweaked the systems in such a way that the user is tricked into imagining the virtual world being a single-instance massive environment. In practice, World of Warcraft, for example, uses a system of instancing where a limited amount of players are active in a single instance. By adding additional instances of the same virtual world, the community of players can grow indefinitely (as long as servers are added). However, it is clearly not possible for all players in the community to interact—as they need to be active in the same instance. A different approach is taken by Second Life, a single-instance virtual world, where the virtual land is split up into several regions, each managed by a single server. While this approach is definitely simple to implement, there are very obvious limitations, such as the (possibly disastrous) growth of network and processing load on the server once an event takes place in a location that is deemed popular by the community. Several circumventions have been implemented to mitigate these problems, for example, the limitation of the number of polygons that can make up a model in the environment, or the splitting of regions that have become too popular. This latter solution, however, leads to additional costs in terms of server infrastructure and cannot cope with a highly dynamic world, as the region definition, in terms of server assignment, is relatively static.

2. RELATED WORK

Several years ago, we investigated an alternative architecture, able to support the same numbers of users that are needed for todays' applications, that is, the Architecture for Large-Scale Virtual Interactive communities (ALVICs) [5]. ALVIC was designed from the ground up to be scalable to high numbers of users, all present in a single instance of the virtual world. The basics behind ALVIC were founded on several (at the time) next-generation network features that were thought to become available in the near future. It has turned out, however, that the promised improvements are clearly not yet met. This is the reason why a new version of ALVIC is being designed (dubbed ALVIC-NG), that takes into account the limitations apparent in today's version of the global Internet architecture. For reasons of clarity, a brief description of the original ALVIC architecture is given in Section 3, we do, however, refer the reader interested in more details to our previous work published in [5, 6].

There are some commercially available products (and architectures) that show some familiarities with our work. For example, the BigWorld [7] middleware platform is claimed to be the upcoming industry standard by its developers. The software collection that is offered consists of a set of server applications, together with a 3D client and specially developed API's. Unfortunately, as is common with most commercially available products, the technical details are not disclosed; it is, however, clear that a client/server architecture forms the basis of BigWorld. The technology behind World of Warcraft [8] may seem—at first glance—to be able to support millions of simultaneous users. In practice, however, the World of Warcraft system is based on a sharded design, whereby multiple instances of the virtual world (called shards or realms) are run concurrently on a large number of servers. This means that only a small subset of players is—at any given time—able to interact with each other, typically a few thousand. The architecture also supports "instancing," through which a group of players (typically less than 25 players) can indicate that they wish to complete a quest without interference from other players. The World of Warcraft architecture is fundamentally different from the ALVIC-NG architecture in the fact that it uses sever realms to support the total number of players that is subscribed. In case the number of players outgrows the server capacity, a new realm is started on a new server cluster. Readers familiar with the Eve On-line [9] architecture may notice several parallels with the approach used for that particular game. We will point out, as part of the discussion on the new architecture, some key differences, although the intricate details about the implementation of the EVE Online architecture are also not publicly available.

When comparing ALVIC-NG to Second Life [1], one of the best-known and most successful (single-shard) 3D virtual communities, we should remark that Second Life uses a fixed assignment of (virtual) geographical regions to servers. At the time of writing, a maximum of about 35000 simultaneous users were active in the virtual world, which is run on more than 5000 servers, each serving an area of 256 by 256 meters. While this type of design is easy to implement, scalability problems are sure to become apparent as soon as the number of users increases. As users are not typically evenly distributed over the virtual world, some servers are nearly idle, while some are overloaded. The ALVIC-NG architecture is designed to use the available server processing power as efficiently as possible, thereby decreasing the chance of system failure in case a large number of users decide to convene in a single location in the virtual world.

There are also some architecture-only solutions that should be compared to our ALVIC-NG framework. One of these is the Sun Game Server Technology framework [10]. This is fundamentally different from our approach, as the virtual world is not spatially subdivided; every server is able to manage each object through a massive centralized database. For each operation that is to be performed on an object, the information is retrieved from the database, and stored again after the manipulation is completed. While this is clearly scalable in terms of the size of the virtual world, this solution also introduces extra delay for each operation that is to be performed on an object (which may accumulate if interactions involve several objects). This architecture is designed to be scalable up to around 10000 users, and it uses a cluster of database servers that are load-balanced. A second example of a similar architecture is the Multiverse technology platform, which also does not use a "traditional" spatial subdivision scheme for scalability purposes (it does in fact have such a scheme for visibility purposes, but this is rather trivially implemented and not relevant here), but rather tries to scale the number of supported users by defining services that are implemented through plugins. Examples of these services are those that handle combat events, intricate interactions, and so forth. At run-time, those plugins that are able to support a large number of users can be migrated to servers that are minimally loaded. However, it should be noted that some plugins are inherently more processor-intensive than others (or are sure to be used much more than others), so they will probably be assigned to the most powerful servers anyway.

In research, several other architectures have been proposed and discussed, which were designed to support networked virtual environment applications, both client/server and peer-to-peer based. In this section, we have specifically opted to discuss only those that are currently being used in the specific context of games. We do refer the reader to [6, 11, 12] for a comprehensive overview of existing literature on the more general subject of NVEs.

Section 3 provides a brief overview of the original ALVIC architecture, which is required to understand some of the design options that were made for ALVIC-NG. Section 4 describes the problems associated with ALVIC, when one wants to deploy the architecture on real-life networks such as the Internet. In-between solutions that can be used to overcome some of the limitations are presented in Section 5. The next generation of ALVIC is described in Section 6. Conclusions and pointers to future work are presented in Sections 7 and 8.

3. THE ORIGINAL ALVIC ARCHITECTURE

The architecture behind ALVIC was designed to be adaptable to several usage scenario's, ranging from games to virtual interactive communities. Each of these applications should be able to be deployed on the same architecture, preferably even concurrently. However, in practice, this means that each end-user can, at a given time, only be present in one specific world. Because of the extensive size of the virtual world, we followed an approach similar to that in [13], in which each virtual world, running on the server infrastructure, is divided into a number of square regions. Their size depends on the estimated number of active clients in that region and on the type of region. For example, a region that represents a small room inside a building would most likely be scaled to equal the dimensions of the room. Clients that move around the world dynamically enter and leave regions depending on their position.

The reasoning behind this subdivision of the world is to effectively link the physical properties of the virtual world (geographic location) with the underlying network architecture. The relation between the two entities is strong because of the fact that data propagation can easily be coupled to visibility. If an object is invisible to the end-user, there is no need for any data to be received. Furthermore, by assigning a distinctive multicast address to each of the regions defined before, we can reduce unnecessary network traffic.

In fact, event information, originating from a single end-user, should only be sent to the multicast address of the region from which the event originated. When a client enters a region, a simple subscription to the multicast group assigned to that specific region suffices to start receiving state information on all objects present in the region. As all members of a region send their generated events to the same multicast address, it should be clear that they will also receive all events from other members in the same group without the need for an explicit distribution mechanism through a dedicated or ad hoc-defined server.

It should be clear that a mapping of these (geographical) regions onto multicast groups is an efficient way of distributing data. There is no need to maintain open connections with a "number of" server(s) to receive state information. Neither would one need to determine where to send data, as the current location is always known by a client. The key to the entire system is the fact that data distribution within a multicast group is done implicitly.

Besides this first trivial task, each client is responsible for managing its own *area of interest* (AOI), analogous to, for example, the work in [14]. It is of vital importance to note (as stated before) that there is a coupling between geographical regions and their associated multicast addresses. It can clearly be seen that at a specific moment in time, a limited number of other regions will be located in the view frustum of a client. It is, therefore, only necessary for a client to subscribe to exactly those regions. The view frustum size is entirely client-side determined, and can be adapted dynamically to either expand or shrink depending on several factors, such as available bandwidth or processing power. We point out here that a large view frustum does not have any impact on the upstream traffic needed for sending out state information, as this data only needs to be sent to the local multicast address.

While, in theory, it is entirely possible to design a networked virtual environment architecture using only multicast traffic, we opted to include a set of governing servers into the architecture. Their purpose is threefold: authentication, network resource management (e.g., multicast addresses), and server resource management. The minimal load on these servers in the architecture allows for a large number of clients to be simultaneously connected to a single server (for more details, see our previous work, as referred in Section 2) and facilitates the distribution of load over several physical machines.

4. ALVIC-SPECIFIC DEPLOYMENT ISSUES

While ALVIC has been shown to scale to several thousands of users using only a very limited number of servers (see [15]), the features it relies on to make this possible have still not become widely available on the Internet as it is available to typical end-users. In this section, we will identify the main issues that still exist, and will remain problematic in the near future. While some intermediary solutions exist for some of the problems, a good example of this is the use of TURN for NAT traversal, they cannot always be applied to the specific transmission methods used in ALVIC (i.e., multicast transmissions). Also, not all solutions provide satisfactory results due to the special requirements posed on the multicast transmissions employed by ALVIC (e.g., IGMP snooping would introduce prohibitive amounts of delay, IPv4 tunneling of IPv6 traffic is rather inefficient, etc.). These restrictions have been the main reasoning behing the development of the next-generation ALVIC architecture.

4.1. IPv6 deployment

When ALVIC was first proposed, the mass introduction of IPv6 was touted as being the solution to many of the problems facing the Internet community at the time. Several features, such as a massive increase in the machine-addressing space and the support for large numbers of multicast addresses (together with improved supporting protocols), would make it possible to manage a large set of multicast groups, necessary for ALVIC to be deployed for massive environments. It has turned out, however, that while the backbone networks of ISP's do support or actually run on IPv6, the availability of this technology to typical end-users is still severely limited, and will probably remain so for the next few years.

4.2. NAT gateways

The main reason behind this is the proliferation of IPv4 NAT gateways and firewalls, which mitigate the problems associated with the limited number of addresses available. By hiding several machines behind a common IP address, these machines are at the same time able to connect to the Internet, and they are (relatively) protected against attacks from the

outside. For ALVIC, however, NAT gateways present a major obstacle, as direct peer-to-peer connections are required for the architecture to work as it was originally designed. Even worse, the support for multicast applications behind NAT gateways is practically nonexistent. To provide an optimal experience to the end-user, it is clearly undesirable that major reconfiguration of network equipment (such as port forwarding and/or definition of DMZs) is required to run an application.

4.3. Multicast address space and scope

Multicast applications, as they exist today, are based around a very limited set of content producers that distribute their data to large amounts of "listeners." This is especially true for the case of digital TV distribution (possibly on-demand), where a single producer (the broadcasting company and/or network provider) sources all data watched by subscribers. As these networks are managed by a single entity (the network provider), the scope of the multicast transmissions can be limited to the provider-owned network. For ALVIC purposes, addresses with a global scope are clearly required, as a single instance of the virtual world is required for all users. Coming back to the example of Digital TV, it should also be clear that only a limited number of addresses is required (e.g., one for each stream in a set of TV channels). ALVIC, on the other hand, requires large amounts of multicast group addresses, if it is to be deployed with a fine-grained spatial subdivision scheme. At the same time, real-life wide-area networks are optimized specifically for one-to-many multicast applications. However, in the case of ALVIC, users generate their own multicast traffic, which needs to be transmitted from their own computers or devices to the other participants, something which cannot be done on these types of networks due to the possible explosive growth in traffic.

5. INTERMEDIATE SOLUTIONS

As it became clear during the final stages of the development of ALVIC that the Internet would not quickly evolve in the direction that was required for deployment, a temporary solution was envisaged that would overcome several practical issues, while retaining the advantages a multicast-based architecture could offer.

We used the CastGate [16] project, which, in practice, consists of two entities. One of these entities, the router, is placed in the local, multicast-enabled LAN. Its role is to intercept the packages that are to be transmitted to the multicast-enabled backbone network. The link between the router and the multicast-enabled backbone network is unicast only. A separate entity needs to be placed in the backbone network as an end point for the tunnel between the different networks.

Using this approach, we were able to interconnect several sites using a number of routers, every one of them connected to the multicast-enabled BELNET network. While it would, in theory, be feasible to have each of the connected parties install this additional piece of software and to reconfigure their network equipment, this is clearly undesirable from a user's point of view. It is, therefore, only interesting for academic reasons.

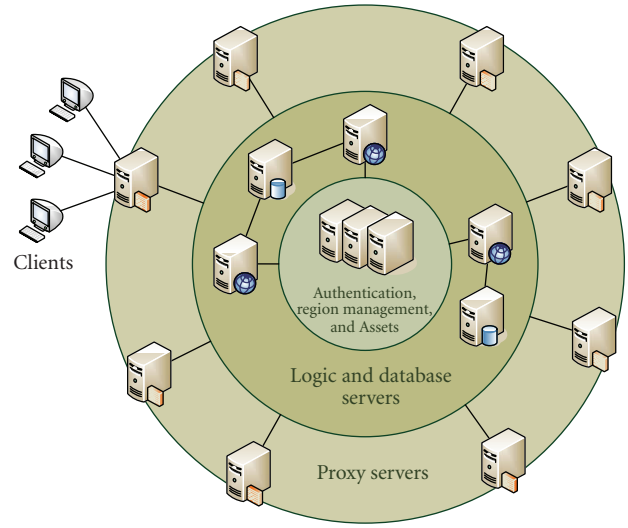

Figure 1: Conceptual overview of ALVIC-NG.

6. ALVIC-NG

The limitations as described above, combined with the unsatisfactory intermediate solution described in Section 5, have triggered a major redesign of the ALVIC framework, now designated as ALVIC-NG (next generation).

We have been very careful to retain the strong points of ALVIC, while translating them into a more real-life network-friendly architecture, mainly based on the client-server paradigm instead of peer-to-peer.

6.1. Overall overview

The main entities of the architecture are shown in Figure 1, represented in a set of concentric circles. At the outer perimeter, the clients are shown that want to connect to the virtual world. Instead of connecting to a variety of supporting servers as is often the case in current-generation examples, such as Second Life, nearly all traffic is tunneled over the client-proxy link. The proxies are responsible for handling a number of clients at the same time, and are assigned based on several properties. These may include, for example, their processing load and/or the network properties of the link between the client and the proxy (e.g., typical RTT values and/or packet loss). More on this subject can be found in next section. Proxies are assigned from a pool of available servers, managed by a centralized entity, which is also responsible for other authentication and accounting tasks. This entity, as mentioned in what follows, is based on the master server of the original ALVIC architecture.

As the spatial subdivision and area of interest management scheme used in ALVIC provided us with a powerful way to manage downstream bandwidth, we wanted to retain this system for the new architecture. However, the peer-to-peer approach needed to be substituted with a client-server-based

equivalent. The new entities responsible for managing parts of the world are referred to as Logic servers. They are notably different from, for example, the simulators in the Second Life architecture, in the way they are assigned to geographical regions in the virtual world. Instead of using a fixed allocation scheme, as is traditionally used, a new entity is created, responsible for managing the relationship between virtual locations and Logic servers, called the Region Management (RM) system. Analogies can easily be drawn between these RM servers and the DNS system that is currently in use on the Internet. The RM system can be queried by the proxy servers to find out which region(s) is/are managed by a specific server. At the same time, the RM system is responsible for keeping track of the load on the various logic servers. A control link is, therefore, established between the RM system and the individual logic servers, over which several parameters are sent, comparable to the SNMP querying system. In case the RM system detects either a Logic server failure or an impending overload of a specific server, the Logic servers are re-assigned to remediate the problems. Possible solutions include splitting the management of a single part of the virtual world over a number of servers or transferring the complete responsibility to a new instance, for example, in case of complete logic server failure. A more detailed scenario is described in Section 6.2. When compared to the original ALVIC design, the Region Management system is roughly comparable to the Game server entities.

Logic servers can also be used as entities that control the behavior of objects, such as non-playable characters (NPC's) or autonomous interactive objects such as virtual video walls. Behaviors are triggered by scripts that are assigned to specific objects. As the logic servers are responsible for handling all objects present in a specific part of the virtual world, which will traverse the virtual word, the scripts need to be shared between all logic servers. These scripts, together with the information regarding the visual representation of objects, are stored in asset databases.

The reason behind the introduction of the intermediary layer of proxy servers in the architecture is threefold. First of all, it reduces the number of connections each client needs to initiate and maintain with other servers (which may lead to issues as discussed in Section 4 due to the presence of firewalls and NAT gateways). Secondly, the proxies reduce the number of connections for the logic servers, which is important if a high number of clients is to be supported on a single machine due to the overhead associated with connection tracking. Finally, the proxies can "cache" a lot of data, possibly reducing the response time (and load) on the logic servers, as these servers can be assigned in such way that they provide a better response time than the entire path between the client and the logic server(s).

As with any virtual environment system, persistent storage is a requirement to keep the world up and running over long periods of time. It also offers enhanced features such as roll-back capabilities in case of system failure and/or, more applicable to the virtual world scenario, in case of malevolent users that have exploited the system. Instead of using a single, high capacity database, as is typical in existing applications (e.g., Eve Online), the ALVIC-NG architecture provides a fine-grained mechanism for determining the degree of persistency that is required. In case transactions being handled have financial repercussions (e.g., the exchange of virtual currency between users), it is likely that these transactions need to be logged and written to disk immediately, as an in-between state, where currency is "floating" between users would clearly not be desirable. However, it should be clear that not all objects and actions require an immediate storage of state to disk. This enables the ALVIC-NG architecture to retain as much state as possible in the main memory of the Logic servers, which improves both response time and the load on the database servers. It is, in any case, the goal of ALVIC-NG not to use a single server (farm) for persistent storage, as the requirements on this type of server would increase in a nonlinear fashion with a growing number of users. A clear demonstration of this fact is the limitation that is put on the objects that are present in a single simulation server (analogous to our logic servers) in Second Life. In reality, only about 15000 prims (primitive objects such as spheres, cones, etc.) can be supported on a single server [17]. Also, relational database systems are CPU-intensive applications. An example is given in [18], where a cluster of more than one hundred machines is required to support about 30000 transactions per second in a game context. The persistency modules of ALVIC-NG are designed in such a way that they can employ a number of distinct servers, again based on factors such as load or network link capacity. As the update rate of the database system is low due to the in-memory processing and adaptive storage requirements for different classes of operations (e.g., player movement versus financial transactions), we are able to use a basic MySQL infrastructure, were a number of instances of this software can run on the same hardware as the logic servers. Persistency and the inclusion of logic servers that do processing on parts of the environment is something that is entirely new to the ALVIC-NG framework, as the (previous) peer-to-peer approach necessitated the individual clients to be responsible for the distribution of their own state information. In case the client would disconnect, there was no way to store his/her data in a central location. Please note that ALVIC-NG only provides an interface to a back-end, which will in most cases consist of an off-the-shelf database management system (either object-oriented, relational, or any other type). A benchmark for the performance of this back-end is outside the scope of this paper. The back-end should provide functionality such as roll-back capabilities and redundant storage of information, which will automatically provide the ALVIC-NG framework with the same capabilities.

6.2. *Typical usage scenario*

To clarify the interdependence of the various entities in the architecture, we will describe the typical workflow for a client that connects to the system and subsequently moves around in the virtual world. We refer to Figure 2 for a graphical representation.

Initially, the client is unaware of the existence of the proxy and logic servers. The only publicly known entities

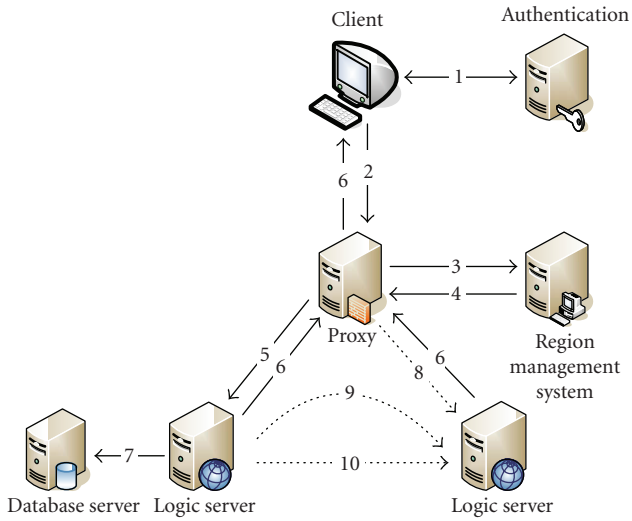

Figure 2: Typical usage scenario of ALVIC-NG.

are the Authentication servers. A (possibly minimal) pool of proxy servers is available, and the members of this pool are known by the Authentication servers (through the region management facility, which was described in the previous section). Once a connection has been established by the client (1), a specific proxy is assigned. The decision on which proxy to select is based on several metrics, including the current "load" on the proxy. In practice, the load calculation on the proxy servers is achieved through several parameters, including the number of clients it is currently serving, as well as the 5-minute-average CPU load (which is available, e.g., on linux systems through the /proc file system). At regular intervals, this load information is exchanged between the proxy servers and the region management infrastructure, enabling the authentication servers to choose a proxy server that currently has a minimal amount of "load." Of course, a proxy server may still decide to reject clients based on momentary information, available only to the server itself (after which the process repeats from the beginning). Beside the load factor, another metric is used in determining the appropriate proxy server to assign, namely, the network delay between the client and the server. The network delay is sampled from (existing) active connections between clients and proxies, and is also communicated through the region management infrastructure. If this data is not available (in case of a newly introduced proxy server), the geographic location of the proxy server can be compared to that of the client (available through, e.g., a WHOIS database lookup) and used as as intuitive metric. While this does not guarantee an optimal assignment, it does severely decrease the chance that a server is chosen on a very impaired network path. In practice, a combination of several metrics can be used to provide satisfactory results.

Once authentication is finished, the client is redirected to the proxy server and establishes a (reliable) control connection (2). At the same time, a UDP data channel is established by sending out datagrams to a specific port on the server (dynamically assigned). This enables the packets to be sent in the reverse direction, possibly traversing the NAT gateway at client side.

Subsequently, the client will announce its initial position to the proxy server it is connected to. We should point out here that only the proxy servers are aware of the assignment of regions to servers, not the clients. This enables regular updates of the mapping, without requiring notifications to be sent to all connected clients. The starting position of the client determines its starting region, and the associated Logic server address is determined (by the proxy) by querying the Region Management servers (3 and 4). If the client is located in a region for which the proxy server does not yet have an active connection to the logic server, the connection is established (5) and the client position is announced to the Logic server in question. At the same time, all updates originating from the specified region are, through the proxy server, sent to the client (6). This may also include additional regions, as required by the AOI management scheme applied.

On a regular basis, determined by the persistency requirements as explained above, the data stored in the Logic server's memory is stored in one of the many database servers (7).

If a client is moving around in the virtual world, and traverses a region boundary, the proxy will detect this, request the associated Logic server address by querying the RM system, and connect to the new Logic server (8). The information state associated with the client is removed from the old server and uploaded to the new one (9).

At any given time, the RM system may determine that one of the Logic servers is overloaded and/or has failed. In the former case, a migration of data will take place, called a region split. This involves storing the Logic server's state in a database and/or directly exchanging information between servers (10) (depending on connectivity) and subsequently assigning the newly created regions to the appointed logic servers. If required, it is possible for the new Logic servers to update their state memory by reading it from the persistent storage medium. In case of a region split, the region boundaries will be updated, and these updates will be announced to the proxy servers (the clients remain unaware of the world buildup). An analogous scenario can be envisaged for the merger of two regions with minimal client load. In case of server failure, a new server is assigned and the state is recreated from what is available in persistent storage.

The decision of splitting and merging regions (at runtime) is left to the region management system. As we stated before, the load on several entities in the architecture is communicated in specific intervals between these entities and the RMS. In this case, the logic servers gather "local" information on the amount of regions they are currently managing, the amount of open connections to proxy servers and the amount of objects in memory (or persistent storage). This information is compared to static information on the available processing/handling power of each of the logic servers (determined by link capacities and raw CPU power). Based on these metrics, a decision will be taken to either split or merge regions if the server becomes overloaded or even superfluous (due to client inactivity in specific regions). However, a single strategy cannot be cited as being the "best"

solution under all circumstances. Depending on the type of game or overall player behavior, it may or may not be desirable to have frequent updates in the spatial subdivision scheme. The ALVIC-NG architecture does not depend on a single metric to determine region management, but is rather developed in such a way that new metrics can easily be added. We will come back to this issues when discussing some of the simulation results in Section 6.4.

6.3. Advantages of ALVIC-NG

In this section, we will look at how the new ALVIC-NG architecture overcomes the issues described in Section 4.

First of all, tunneling all traffic required for a session through the proxy servers enables us to have a severely limited number of open connections and streams at any given time. They also remain the same during an entire session, which is an ideal situation when considering NAT gateways and firewalls. All TCP connections can be initiated at client side, and UDP sessions can easily be kept alive as the port numbers remain the same. There is no need for any incoming peer-to-peer traffic that may be blocked by the network configuration.

The spatial subdivision scheme, proposed in ALVIC, was retained but redesigned to be independent of multicasting capabilities of the network. We should, however, point out here, that the software architecture underlying ALVIC-NG is designed in such a way that the previous implementation using multicast is still supported. The fact that regions are now assigned to Logic servers instead of multicast groups in a dynamic way enables us to exchange the data using unicast connections, albeit with the additional overhead caused by this distribution method. To mitigate this clear disadvantage, the proxy servers are also able to act as caching servers, and distribute "known" data to users without having to fetch the data for each client individually.

Using a (possibly large) set of proxy and logic servers, globally distributed, relieves the need for multicast addresses with a global scope. At the same time, it enables optimal connection circumstances for clients (in terms of delay and link capacity), without the additional delays associated with the propagation of IGMP messages required for multicast traffic.

Of course, there is a tradeoff when switching to a client-server-based architecture from a peer-to-peer approach. For one, we loose the automatic distribution mechanisms offered by multicast transmissions. At the same time, the additional investment in terms of server infrastructure may be a hindrance to the uptake of applications based on ALVIC-NG. We do feel, however, that the added advantages, in terms of being able to be deployed on any current-generation broadband access network technology, as well as the ease of management and moderation outweighs these disadvantages.

6.4. Simulation and results

To test the concepts introduced in the ALVIC-NG architecture, we have implemented the various elements to serve as a test-bed for scalability tests. The applications are deployed on a dedicated 16-node PC cluster, interconnected through a gigabit LAN. For testing purposes, the "client" application was developed with a dual interface: one that uses 2D/3D visualization (to confirm the correct functioning through the eyes of a user) and a command-line version; the latter enables us to deploy a large number of instances of the client software on a single machine. As static clients (in other words, clients that do not move around in the virtual world) are not representative of real-life users, a scripting language (LUA [19]) is used to move the avatars in the virtual world, based on predefined movement patterns. As the behavior is scripted, and scripts are assigned on a random basis, the result is a semirandom population of the virtual world. The client with visualization enabled allows us to check whether the system works as intended. Besides the normal client functionality, this version also is able to query the servers in the world to get an overall outlook on the region assignment to logic servers.

A sample is shown in Figure 3, where the world is divided into seven regions. We should point out here that, for these simulation results, we have opted to manage the virtual world as a quad-tree, as this is an efficient data structure for fast detection of boundary passing and can easily be split/merged. The ALVIC-NG architecture, in principle, can be extended to work with a generic region definition. The "active" avatar is shown by the blue dot, surrounded by the circle indicating its area of interest. The "active" regions are indicated by the slightly brighter colors (in this case, these are the ones that overlap with the clients' AOI). The other avatars, which are, as stated before, steered by scripting, of which state information is being received, are also visualized. By moving around in the virtual world, this simulation allows us to test that handovers between logic servers can be handled without major delays and impact on the user experience. It should be noted that a certain delay cannot be avoided, as there is a propagation delay for the new data to arrive at the proxy. The simulation setup will enable us to effectively determine worst-case figures for this delay value and examine how this deficiency can be masked (e.g., through tweaks in the graphical rendering engine). At the same time, it allows us to test the efficiency of the region splitting/merging algorithm that is implemented, which may depend on a number of metrics, as mentioned in Section 6.2. The application also allows us to force the split/merge operation on regions to simplify the testing process. In Figure 4, another scenario is visualized, in which a more intricately subdivided world is shown, together with a client with a reduced AOI (indicated by the smaller surrounding circle).

7. FUTURE WORK

The load and scalability tests on the ALVIC-NG framework are ongoing work. Based on our findings, the metrics used for determining the optimal time to split/merge regions can be adapted. The ALVIC-NG architecture is to be used as a basis for story-telling applications, games, and community-related features in the IBBT Teleon Project, the goal of which

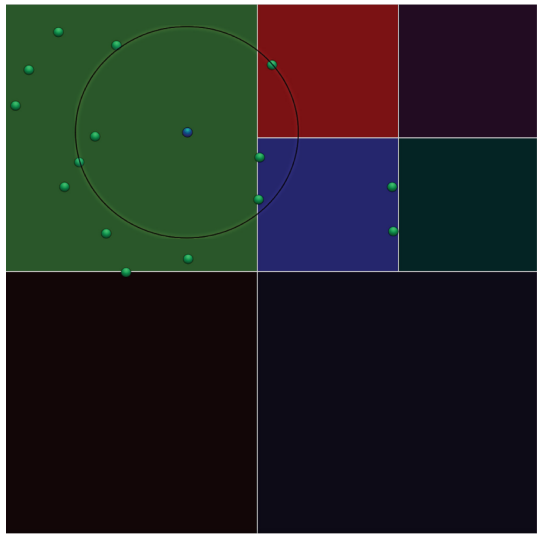

Figure 3: Sample spatial subdivision based on a quad-tree. Active regions are brightly colored.

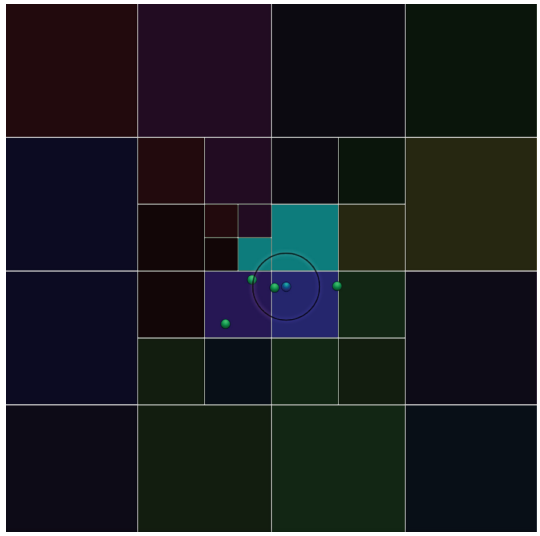

Figure 4: A client with reduced AOI in a more intricately subdivided virtual world (distributed over several logic servers).

is to deploy the architecture as the base for a nation-wide platform (under supervision of the Flemish radio and TV broadcasting company VRT).

8. CONCLUSIONS

In this paper, we have introduced ALVIC-NG, a generic framework supporting networked virtual environment applications. The technology presented is applicable to both games and virtual community applications. Based on our original findings during development of the ALVIC architecture, the NG version is designed from the ground up to be deployable on heterogeneous networks, independent of the availability of next-generation network features such as user-generated multicast data distribution and large numbers of globally scoped multicast addresses. The unique selling points of ALVIC, in terms of its spatial subdivision scheme and scalability, have been retained in a novel, mainly client-server paradigm-based architecture. Besides purely scalability-related features, ALVIC-NG also offers solutions for issues that plague many current-generation applications, such as intricate NAT traversal and the additional problems associated with peer-to-peer traffic flows.

ACKNOWLEDGMENTS

Part of this research is funded by the European Fund for Regional Development (EFRD). The authors are grateful to the partners involved in the IBBT Teleon project, as well as the members of the NVE research group at EDM.

REFERENCES

[1] BigWorld Technology, *BigWorld*, http://www.bigworldtech.com/.

[2] Blizzard, *World of Warcraft*, http://www.worldofwarcraft.com/.

[3] MULTIVERSE, *Multiverse*, http://www.multiverse.net/.

[4] P. Quax, *An architecture for large-scale virtual interactive communities*, Ph.D. thesis, Transnationale Universiteit Limburg, Limburg, Belgium, 2007.

[5] K. L. Morse, "Interest management in large-scale distributed simulations," Irvine Technical Report TR 96-27, University of California, Berkeley, Calif, USA, 1996.

[6] P. Quax, T. Jehaes, P. Jorissen, and W. Lamotte, "A multi-user framework supporting video-based avatars," in *Proceedings of the 2nd Workshop on Network and System Support for Games*, pp. 137–147, ACM Press, Redwood City, Calif, USA, May 2003.

[7] Linden Labs, *Second Life Forums*, https://jira.secondlife.com/browse/MISC-210.

[8] C. Greenhalgh and S. Benford, "A multicast network architecture for large scale collaborative virtual environments," in *Proceedings of the 2nd European Conference on Multimedia Applications, Services and Techniques (ECMAST '97)*, pp. 113–128, Milan, Italy, May 1997.

[9] M. Capps, D. McGregor, D. Brutzman, and M. Zyda, "NPSNET-V: a new beginning for dynamically extensible virtual environments," *IEEE Computer Graphics and Applications*, vol. 20, no. 5, pp. 12–15, 2000.

[10] CastGate, *CastGate*, VUB ETRO-TELE, http://www.castgate.net/.

[11] There, *There.com*, http://www.there.com/.

[12] R. C. Waters, D. B. Anderson, J. W. Barrus, et al., "Diamond park and spline: a social virtual reality system with 3d animation, spoken interaction, and runtime modifiability," Tech. Rep. TR-96-02a, Mitsubishi Electric Research Laboratories, Cambridge, Mass, USA, November 1996.

[13] SUN, *Game server technology white paper*, Sun, http://www.sun.com/solutions/documents/white-papers/me_sungame-server.pdf.

[14] P. Quax, P. Monsieurs, T. Jehaes, and W. Lamotte, "Using autonomous avatars to simulate a large-scale multi-user networked virtual environment," in *Proceedings of the ACM SIGGRAPH International Conference on Virtual Reality Continuum and Its Applications in Industry (VRCAI '04)*, pp. 88–94, ACM Press, Singapore, June 2004.

[15] Microsoft, *SQL Server 2008 benchmarks*, http://www.microsoft.com/sqlserver/2008/en/us/benchmarks.aspx.
[16] C. Joslin, T. Di Giacomo, and N. Magnenat-Thalmann, "Collaborative virtual environments: from birth to standardization," *IEEE Communications Magazine*, vol. 42, no. 4, pp. 28–33, 2004.
[17] Linden Labs, *SecondLife*, 2003, http://www.secondlife.com/.
[18] LUA, *The LUA programming language*, http://www.lua.org/.
[19] M. Matijasevic, "A review of networked multi-user virtual environments," Tech. Rep. TR97-8-1, The Center for Advanced Computer Studies. Virtual Reality an Multimedia Laboratory. The University of Southwestern Louisiana, Lafayette, La, USA, 1997.