Proceedings of the International Conference on Information and Automation, December 15-18, 2005, Colombo, Sri Lanka.

Enabling Massively Multi-Player Online Gaming Applications on a P2P Architecture

Scott Douglas, Egemen Tanin, Aaron Harwood, and Shanika Karunasekera
NICTA Victoria Laboratory
Department of Computer Science and Software Engineering
University of Melbourne, Victoria 3010, Australia
www.cs.mu.oz.au/p2p

***Abstract*— Recent advances in Peer-to-Peer (P2P) technologies have enabled the use of P2P architectures for developing complex applications such as, Massively Multi-Player Online Games (MMOG). In this paper we address a challenging research problem related to P2P MMOGs; efficient entity maintenance and interaction. We have identified two opposing software engineering approaches to address this problem. The first approach assumes that the entire virtual world and the game logic are implemented using database techniques, e.g., with a distributed index over all the peers and the game semantics built into data query/update functions. The second approach is to separate the entities into completely independent processes, using agent-oriented programming techniques. In our work, we propose a combination of these two approaches; the use of a distributed spatial index to facilitate the discovery and querying of relevant entities and an agent-based approach to facilitate real time interactions between the entities. We show the relevant factors for dynamic optimization and the use of a new software architecture with existing game libraries to implement the system.**

I. Introduction

The Peer-to-Peer (P2P) computing paradigm is currently receiving considerable attention. P2P networks originated as a means of distributed file sharing (e.g. Napster, Gnutella, and Kazza). Many recent research advances have enabled P2P systems to be used for complex applications, beyond simple file sharing. We focus on one such class of complex applications, namely Massively Multi-player Online Gaming (MMOG) applications. A P2P architecture offers several advantages over the currently used centralized architecture for MMOGs. The elimination of the need for a central server and high scalability are two key advantages. In this paper we address a challenging research problem related to P2P MMOGs; efficient entity maintenance and interaction.

A MMOG can be modelled as a set of interacting entities in a 3D space, i.e., a virtual world, with a large population of players. Players run an application on their PCs that allows them to interact with the entities and each other, i.e., using avatar entities. Entities can be *static* or *dynamic*. Static entities, e.g., a picture hanging on a wall, can be cached at all users' PCs and displayed whenever required. Dynamic entities, e.g., a rocket propelled grenade launched by a user, need to be continually updated on all relevant PCs. Dynamic entities can be user controlled, changing state as the result of user input, or as the result of pre-programmed logic into the game. The rules that govern these interactions and state changes can be referred to as the *game logic*. In this paper, we are concerned with the efficient entity maintenance and interaction in the context of a P2P network, i.e., when all PCs are peers and there is no central server or administration to coordinate the interactions.

In our preliminary prototypes, we have identified two, opposing design approaches to the problem of efficient entity maintenance and interactions in P2P networks. The first approach assumes that the entire virtual world and the game logic are combined into an entity database, distributed over all the peers. Entity location and state are directly updated on the distributed database. The changes in the entity locations are given to the users through a publish/subscribe scheme or direct querying. The database is responsible for validating the entity interactions. This approach can provide powerful query capabilities across the virtual world, e.g., to determine entities in a given region of the virtual world efficiently. On the other hand, it is difficult to sustain efficient entity interactions between strongly related entities in an indirect manner and updates to this database can become cumbersome. The second approach is to separate the dynamic entities into independent processes, akin to the agent-oriented programming techniques. Interactions are naturally resolved between the related entities, e.g., when they come in contact with each other. Entities can migrate from peer to peer. The agent approach provides for efficient communication and processing between strongly interacting entities, yet it is difficult to run queries or maintain global connectivity in this setting as we do not maintain a global spatial index. Therefore, the solution will not scale well to large applications.

The agent approach can be quickly adopted for MMOG developments and this may be preferred due to the ease in design and implementation of the game. However there are significant advantages given by the database approach for a MMOG. In our work, we propose a combination of these two approaches; the use of a distributed spatial data management system to facilitate the discovery and querying of relevant dynamic entities, combined with the agent approach to facilitate real time interactions.

In this paper we provide a detailed description of our proposed approach. We show how our work can be implemented

Proceedings of the International Conference on Information and Automation, December 15-18, 2005, Colombo, Sri Lanka.

using a software architecture that we recently developed for building complex P2P applications. We also show how an existing game library for high performance graphics rendering, *Crystal Space*, can be integrated with a specific game logic using our approach. We make comparisons to other proposed systems for MMOG games and state why our system is a favourable choice. We present a P2P MMOG prototype we have developed using our approach.

II. Related work

P2P networks have a number of characteristics that make them particularly suitable to MMOGs. They can be scalable, easily deployable, and more robust than client-server systems. The challenge is to dynamically organize the network in a way it provides an efficient communication mechanism between many interested parties. Typically the communication overheads are dominated by frequent updates informing players about opponents' positions, directions, and other characteristics. Consequently, low update latency and update filtering is critically important to MMOGs.

An early P2P game is MiMaze [1], which uses all-to-all communications through IP multicasting for player updates. This approach does not scale well as it suffers from bottlenecks. MMOGs often host many thousands of participants at any time and, therefore, communication must only occur within subsets of distributed entities, i.e., mutable spatial objects in the virtual world, while allowing for a global sense of existence in a larger setting. The performance and scalability of the game is thus dependent on successfully determining subsets, often referred to as interest management [2], from a globe of many participants. Interest management schemes can be broadly classified as either *neighbor based* in which peers maintain a list of nearby entities with which to communicate, and *region based* in which updates are performed through queries on regions.

A. Neighbour based interest management

Kawahara et al. [3] have proposed a neighbor based interest management algorithm which uses Euclidean distance within the virtual world as an heuristic to form connections. The algorithm suffers from network partitions as groups of entities move apart from each other. The authors state that this can be overcome by using random introductions via a bootstrap node, however, this makes the system reliant on a central peer.

The Solipsis protocol [4] addresses network partitions by ensuring that the position of neighboring entities form a convex hull around the player, that is, the angle between any two adjacent neighbours is less than 180 degrees. This scheme has a communication and computational overhead, since each entity must check the distance between each disconnected pair of neighbors and inform them of new connections each time an entity moves.

B. Region based interest management

Region based methods assign a region of the virtual space to a coordinating peer. This peer is responsible for propagating updates to all entities within its area and similarly entities must send their updates to it.

Knutsson et al. [5] describe a P2P game which uses a region based approach called SimMud. SimMud assigns each region an ID which is mapped via a hash function to peers in the network. By assigning the region coordinator to the root of a multicast group, players can easily subscribe to region updates.

Bharambe et al. [6], [7] propose the use of the Mercury routing protocol for updates. A publish-subscribe system is built from this by applying queries over an extended period of time. The source of the query then receives updates on objects within the query range.

C. Our Contribution

Our work can be classified as a region based approach. We use a distributed data structure [8] over a base P2P protocol such as Chord [9] or FLOC [10] to find nearby entities with which we form connections for interactions afterwards. Hence, we are decoupling the process of querying a region of the virtual world (along with global entity maintenance) from interaction management between entities (e.g., a set of players in a virtual room). Named as the spatial data service (SDS), the distributed spatial data index that we developed provides a means to insert, delete, query and modify objects in a multidimensional space. Entities register an event region, a region of space which is guaranteed to contain the entity for some time, with SDS. Entities can query SDS to find nearby objects with which direct connections can be formed. This performs effective interest management by minimizing the number of connections, removes routing delays between peers for updates since peers connect directly, does not rely on introduction by neighboring peers to form new connections, and uses little CPU resources to maintain potential neighbors. We maintain the global connectivity and query facility between all the entities without a need for frequent updates on the index. The interacting entities communicate with each other frequently while they send summary updates to the SDS about their positions when needed.

III. Entity Maintenance and Interactions

Consider the set of entities, V, as points in the virtual space. Each point, $u \in V$, is given a position, p_u, velocity, v_u, and an *event region*, e_u. Position and velocity are examples of entity attributes. The event region defines the area of space the entity is guaranteed to be in within a given time limit and will be interested in.

Two entities are said to interact when an event from one entity is received by another entity. In our work we consider all interactions to be pairwise. For any two entities, u and v, the number of events that u receives from v (and vice versa) is proportional to the required accuracy for which changes in v's attributes must be perceived by u. E.g., Fig. 1 shows v changing its position attribute to v' while u stays fixed. The number of events that u must receive from v is a function of attributes like the velocity of v, the distance from u to v, the

Proceedings of the International Conference on Information and Automation, December 15-18, 2005, Colombo, Sri Lanka.

angle that v's motion makes with u's line of sight, etc. Fast perpendicular motion that is very close to u requires many events, while slow motion that is at a great distance from u and/or perhaps obscured by other entities requires few events. Note that the relationship is not symmetrical.

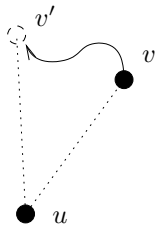

Fig. 1. Motion between entities.

The number of required events to be received by u from v leads to a bitrate requirement, $b_{v,u}$, when u and v are distributed over the Internet. In general $b_{u,v}$ is not equal to $b_{v,u}$.

Similarly, event transmission times must observe a maximum allowable delay if they are to be effectively applied. Let $d_{v,u}$ be the maximum allowed delay for events from v to reach u and similarly $d_{u,v}$ is the maximum allowed delay for events from u to v; these delay constraints are not necessarily equal.

A. Event regions

An event region, e_u, for entity u is defined and is initially centred at u. The entity can move within this event region while the centre of the event region stays fixed.

Fig. 2 shows a number of event regions distributed in two dimensional space and indexed by a MX-CIF quadtree (the entities are shown at the centre of each event region but are not stored in the index). This model basically forms the base for our approach. Overlapping regions are used to establish the extent of network connections between entities that will interact and require frequent communication while the spatial data structure [8] is used for global maintenance, i.e., as a registry for all the entities.

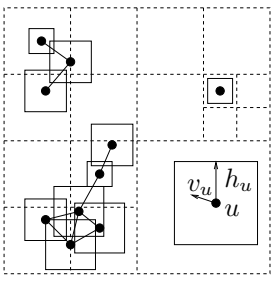

Fig. 2. Event regions, depicted as solid line rectangles, in a quadtree index, depicted by dashed lines. Entities are shown as solid points and thick lines between entities represent possible network connections for communicating events.

B. Moving entities

Generally, as an entity, say u, moves through the space, the spatial data index must be updated with the new event region. Using a buffer zone to obtain hysteresis, updates can be required only when u moves further than a distance $c\,h_u$ from the centre of e_u, where $0 < c < 1$ is a chosen constant. When an update is required, the new centre for e_u becomes the location of u and this information is entered into the spatial data index. It follows that for a constant velocity of v_u, entity u can cause a maximum of $\frac{v_u}{c\,h_u}$ spatial data index updates per time unit.

C. P2P connections

Consider the undirected simple graph $G = (V, E)$, where $(u, v) \in E$ iff e_u and e_v intersect. Let $I_{u,v} = 1$ if the respective event regions intersect and $I_{u,v} = 0$ otherwise. The graph G does not directly represent connections between peers. For generality, we consider the set of peers, P, or PCs, as one peer for each player and a many-to-one assignment function, $Assign\colon V \rightarrow P$. While it may be assumed that player controlled entities are always attached to that player's peer, this may not be optimal at all times for all entities. Computer controlled entities that move large distances through the space may be reassigned to different peers over time.

Two distinct peers $s, t \in P$ may be required to make a P2P connection if for any two entities u and v: $I_{u,v} = 1$, and either (i) $Assign(u) = s$ and $Assign(v) = t$ or (ii) $Assign(u) = t$ and $Assign(v) = s$.

D. Bitrate and delay requirements

The bitrate and delay requirements between two peers, $s, t \in P$, is a function of the entities assigned to s and t and hence ultimately on the bitrate and delay requirements between entities. Let

$$B_{s,t} = \sum_{u,v\colon\, Assign(u)=s, Assign(v)=t} b_{u,v}$$

be the bitrate required from s to t and similarly $B_{t,s}$ is the bitrate required in the opposite direction; these bitrates are not necessarily equal. Let

$$D_{s,t} = \min_{u,v\colon\, Assign(u)=s, Assign(v)=t} \{d_{u,v}\}$$

be the maximum allowed delay from s to t and similarly $D_{t,s}$ is the maximum allowed delay from t to s; these delay constraints are not necessarily equal.

Let N_s be the neighbour set of s. Then the total bitrate requirements for s is

$$B_s = \sum_{t \in N_s} (B_{s,t} + B_{t,s}).$$

Similarly the maximum allowed delay by s is

$$D_s = \min_{t \in N_s} \{D_{s,t}\}.$$

If peer s cannot supply the required bitrate then some interactions will need to be discarded. If the maximum allowed delay is exceeded then synchronization errors may be observed.

Note that in general the peer s will need to provide more bitrate than B_s since the spatial data index also requires communications to maintain. The additional bitrate requirements on peer s due to the spatial data index accesses is essentially the superposition of update rates over entities assigned to s. Let us simply write ω_s to represent this bitrate requirement.

E. Dynamic optimization of interactions

Find an assignment function, $Assign^*$ that allows the largest h_u for each $u \in V$, while maintaining for all s that

$$B_s + \omega_s \leq B_s^{max},$$

where B_s^{max} is the maximum bitrate provided by peer s, and

$$D_s \geq D_{t,s}^{actual} \text{ for all } t \in N_s,$$

where $D_{t,s}^{actual}$ is the actual network delay from t to s.

Note that the optimization is dynamic because the computation of bitrate and delay requirements assumes that a given h_u has been specified in order to locate entities that are near enough to interact (and hence generate bitrate/delay requirements). Our proposed solution lets the optimization to grow and shrink h_u in iterations.

Clearly, the size of N_s is partly determined by h_u for entities assigned to s. A small h_u admits less connections than a large h_u. However as h_u becomes small then the frequency of spatial data index updates for s becomes large, given a constant v_s. In the other extreme, if h_s is allowed to grow unboundedly then the number of spatial data index updates approaches zero. At the same time though, the size of N_s increases to include all peers in the system (since all entity event regions will intersect) which places unscalable demands on the bitrate and delay requirements for s.

The assignment function also partly determines the size of N_s since if two entities are assigned to s then interactions between them do not lead to communication. The assignment function must also satisfy delay requirements. In this case it ensures that delays incurred between peers do not lead to delay in excess of what is specified by entity interaction requirements. In some cases, it may be necessary to migrate a player's avatar entity to a different peer in order to satisfy the delay requirements. In other cases, the requirements cannot be satisfied and the solution may then lie in using specific degradation rules given the MMOG semantics.

IV. Software architecture

We used the OPeN (Open Peer-to-peer Network) application development framework [11] developed in our previous work to build the initial prototype of a P2P MMOG. In this section we introduce the OPeN architecture framework and some of the services that are used for building the game prototype.

A. OPeN architecture

The OPeN architecture, is a layered architecture which facilitates the development of complex P2P applications. The layers of the OPeN architecture are shown in Fig. 3. Applications, which reside in the top layer, the Application layer, can use the services available in the next layer, the Core Services layer. The Core Services layer provides a number of reusable services. P2P protocols and connectivity related functions are managed by the Connectivity layer. Each layer in the architecture, through well-defined interfaces, provides functionality for the higher layers to use. This allows P2P applications to be developed with the appropriate level of abstraction without tight coupling to the underlying P2P protocol.

Following are some of the highlights of the OPeN architecture:

- Abstraction, provided by a layered architecture with each layer providing a higher level of abstraction, completely hides the network and protocol details from applications.
- Interoperability across protocols, provided by the design of the connectivity layer, provides a protocol independent interface to the Core Services layer.
- Delegation, supported through a service-oriented architecture, allows a service to delegate tasks to other peers that support the same service.
- Extendibility, provided by a plug-in architecture where new services and protocols can be plugged.
- QoS Management, supported at each layer of the architecture.

We have already implemented, using the OPeN architecture, several protocols, services, and applications to demonstrate how it meets the architectural objectives stated above [12]. The following sections describe the Spatial Data Service (SDS), Entity Interaction Service (EIS) and the P2P MMOG application which are relevant to the work presented in this paper.

B. P2P MMOG

Figure 4 shows the high level architecture for the P2P MMOG prototype we have developed, which uses an Entity Interaction Service (EIS), combined with SDS for efficient entity maintenance and interaction. EIS uses an Entity Interaction Protocol (EIP) for communication between peers.

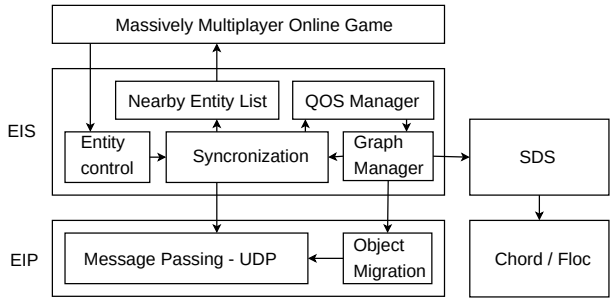

Fig. 4. Game communication architecture.

A distributed P2P virtual world, represents entities in a virtual game space [13]. Users can participate in the virtual world by locating themselves in the space, which would then display the space visible to the user. Users can then interact with the environment by inserting/deleting entities in the space and can move through the space. They can also jump to remote

Proceedings of the International Conference on Information and Automation, December 15-18, 2005, Colombo, Sri Lanka.

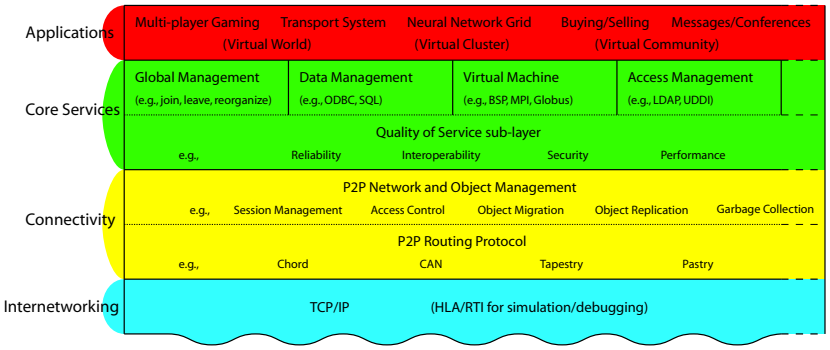

Fig. 3. OPeN layered architecture.

locations in space using the SDS querying mechanisms. The game application uses the services provided by the SDS, to manipulate the virtual world through insert, delete, and query operations. SDS manages the entities in the virtual world, and therefore, the application developer does not have to know the details of how or where the entities are stored in the system. Given a region, all the entities (using their individual event regions) can be retrieved for that region. The interactions between these entities are facilitated through EIS. We used an existing library, Crystal Space, for high performance graphics rendering. Fig. 5 shows the screen shots from the P2P MMOG prototype.

Fig. 5. A screen shot of a P2P MMOG prototype that is under development in our labs.

C. Spatial Data Service (SDS)

SDS supports the querying, insertion, and deletion of spatial data. Our querying capabilities support range queries (also known as window queries) and nearest neighbour queries on multi-dimensional data. This is based on our recent research which supports distributed queries for spatial data in a P2P network [8]. SDS uses a distributed quadtree index where nodes of the tree are service objects, called control points, which are hashed onto a P2P network using a base P2P protocol (e.g., Chord). SDS can be used in a P2P MMOG as a global registry for entities.

Our current SDS implementation uses the Chord protocol but any key based routing protocol can be used. Spatial objects (in our case, the event regions for the entities in a game) are inserted into the SDS by routing queries to the appropriate control points.

D. Entity Interaction Service (EIS)

EIS is responsible for coordinating entity communication, both initiating connections and the sending and receiving of update messages. It provides a consistent view of the virtual world by maintaining a list of nearby up to date entities, which can be accessed by the application for rendering and other game specific processing. The graph manager module (Figure 4) in the EIS handles the topology of interaction through queries to the SDS to determine intersecting event regions. Essentially, it finds nearby entities with which to interact. The quality of service (QOS) manager keeps account of bandwidth usage which can be altered by adjusting the event region as required. Entity logic such as avatar movement and action is handled by the application. Changes in entity state are passed to the entity control module in the EIS, which in turn results in publication of this change to interested entities through messages passed by the synchronization module.

Update messages are handed to the EIP which is responsible for passing them over the Internet with UDP. It is also responsible for event ordering and time synchronization. The interoperability of the service layer allows objects to migrated at any time. As described in section III-E this can be used to ensure the latency requirements are met. The logic which decides when and to which peer this takes place is also located in the protocol.

V. Conclusion

In this paper, we propose the combination of a distributed spatial data management strategy and an agent-oriented programming approach to maintain entities and allow for efficient interactions in a P2P MMOG. We define a dynamic optimization problem which takes into account factors such as bitrate and delay requirements between entities, in this context. We also show how our existing software architecture for building complex P2P applications can be applied to implement our proposed system. We make use of a spatial data service combined with an interaction management service to build a P2P MMOG.

Proceedings of the International Conference on Information and Automation, December 15-18, 2005, Colombo, Sri Lanka.

VI. Acknowledgement

The authors thank National ICT Australia for funding this research.

References

[1] C. Diot and L. Gautier, "A distributed architecture for multiplayer interactive applications on the Internet," *IEEE Network*, vol. 13, no. 4, pp. 6–15, 1999.

[2] K. L. Morse, "Interest management in large-scale distributed simulations," University of California, Irvine, CA, Tech. Rep. ICS-TR-96-27, 1996.

[3] Y. Kawahara, T. Aoyama, and H. Morikawa, "A peer-to-peer message exchange scheme for large-scale networked virtual environments." *Telecommunication Systems*, vol. 25, no. 3-4, pp. 353–370, 2004.

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

[5] B. Knutsson, H. Lu., W. Xu, and B. Hopkins, "Peer-to-peer support for massively multiplayer games," in *IEEE Infocom*, Hong Kong, China, March 2004.

[6] A. R. Bharambe, M. Agrawal, and S. Seshan, "Mercury: supporting scalable multi-attribute range queries," in *ACM SIGCOMM*, Portland, OR, September 2004, pp. 353–366.

[7] A. R. Bharambe, S. Rao, and S. Seshan, "Mercury: a scalable publish-subscribe system for Internet games," in *NetGames*, Bruanschweig, Germany, April 2002, pp. 3–9.

[8] A. Harwood and E. Tanin, "Hashing spatial content over peer-to-peer networks," in *Australian Telecommunications, Networks, and Applications Conference-ATNAC*, Melbourne, Victoria, December 2003.

[9] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for Internet applications," in *ACM SIGCOMM*, San Diego, CA, August 2001, pp. 149–160.

[10] A. Harwood, E. Tanin, and M. T. Truong, "Fast learning of optimal connections in a peer-to-peer network," in *IEEE ICON*, Singapore, November 2004, pp. 16–19.

[11] A. Harwood, S. Karunasekera, S. Nutanong, E. Tanin, and M. Truong, "Complex applications over peer-to-peer networks," in *ACM Middleware (Poster Proceedings)*, Toronto, Ontario, October 2004.

[12] S. Karunasekera, A. Harwood, E. Tanin, M. Troung, and S. Douglas, *OPeN: An Architecture for Complex Applications over Peer-to-Peer Networks*, Submitted to Software-Practice and Experience Journal, March 2005.

[13] E. Tanin, A. Harwood, H. Samet, S. Nutanong, and M. T. Truong, "A serverless 3D world," in *ACM GIS*, Washington, DC, 2004, pp. 157–165.