P2P Middleware for Massively Multi-player Online Games*

Shanika Karunasekera, Scott Douglas, Egemen Tanin, and Aaron Harwood

NICTA Victoria Laboratory
Department of Computer Science and Software Engineering
University of Melbourne, Victoria 3010, Australia

www.cs.mu.oz.au/p2p

1 Introduction

The Peer-to-Peer (P2P) computing paradigm is currently receiving considerable attention. Many recent research advances are enabling P2P systems to be used for complex applications, beyond simple file sharing. We are currently focusing on Massively Multi-player Online Gaming (MMOG) applications.

A P2P architecture offers several advantages over centralized architectures for various applications, including MMOG. Decentralization with high scalability and no single point of failure forms the key advantage. Seamlessly unifying many peers under one application, especially for complex P2P applications such as MMOG, forms the main challenge. We demonstrate the use of a P2P middleware which enables the development of such complex applications, and efficient entity maintenance and interactions for the highly interactive and visual P2P MMOG application domain.

A MMOG application can be modeled as a set of interacting entities in 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. *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 *game logic*. We have developed P2P middleware that manages static and dynamic entities. In our architecture, all PCs are peers and there is no central server or administration to coordinate the interactions.

Figure 1: A screen shot of a P2P MMOG prototype that is developed in our labs that utilizes our middleware platform.

The middleware adheres to the Open Peer-to-peer Network (OPeN) architecture first presented in [1]. We have developed a prototype P2P MMOG application using our middleware. A screen shot from our P2P MMOG prototype is shown in Fig. 1.

*This work was supported in part by the National ICT Australia (NICTA).

2 Research Contribution

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 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. On the other hand, it is difficult to sustain efficient entity interactions between strongly related entities in an indirect manner. The second approach is to separate the dynamic entities into completely independent processes, akin to the agent-oriented programming techniques. Interactions are naturally resolved between the related entities. 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.

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 application. 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. With our layered architecture we can easily facilitate easy development of a P2P MMOG application that enables efficient entity maintenance and interactions.

3 Implementation

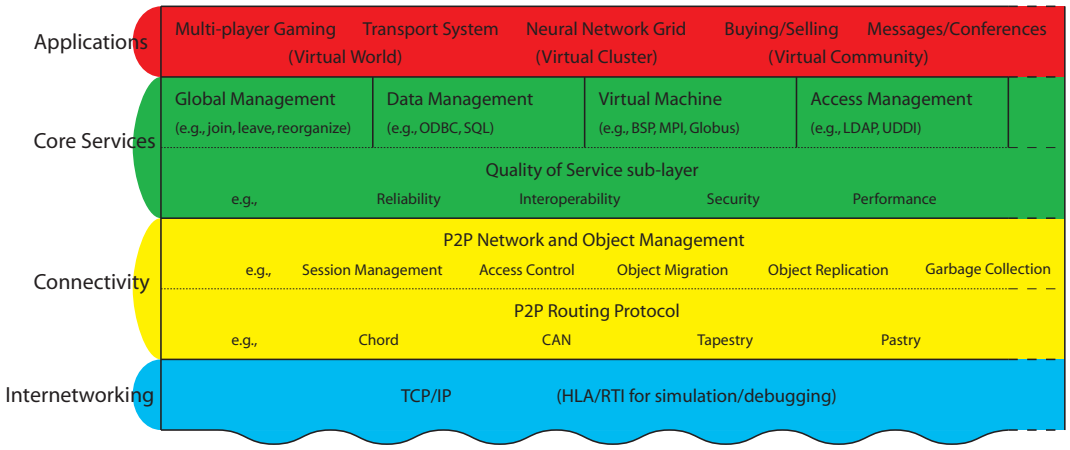
3.1 OPeN Architecture

The OPeN architecture is aimed towards the development of complex P2P applications. The layers of the OPeN architecture are shown in Fig. 2(a). Applications, which reside in the Application layer, can use the services available in 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 provides functionality for the higher layers to use. This allows P2P applications to be developed without tight coupling to the underlying P2P protocol and therefore provide a service-oriented P2P system.

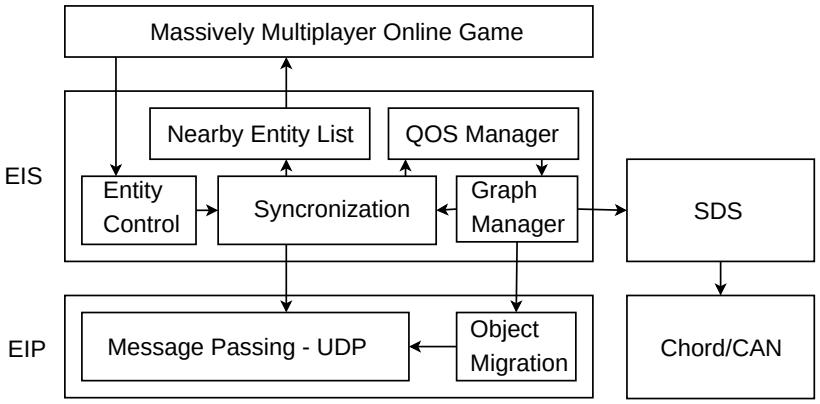
3.2 P2P MMOG Middleware

Figure 2(b) shows the high level architecture for our P2P MMOG prototype, which uses an Entity Interaction Service (EIS), combined with a Spatial Data Service (SDS) for efficient entity maintenance and interaction. EIS uses an Entity Interaction Protocol (EIP) for communication between peers.

A distributed P2P virtual world, represents entities in a virtual game space [5]. Users can participate in the virtual world by locating themselves in the space. 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 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 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.



(a) OPeN layered architecture.



(b) Game communication architecture.

Figure 2: Implementation details.

SDS supports the querying, insertion, and deletion of spatial data. Our querying capabilities support range queries (also known as window queries) and nearest neighbor queries on multi-dimensional data. This is based on our recent research which supports distributed queries for spatial data in a P2P network [2, 4]. 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 [3]). SDS can be used in a P2P MMOG application 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 are inserted into the SDS by routing queries to the appropriate control points.

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 2(b)) in the EIS handles the topology of interactions through queries to the SDS to determine intersecting event regions of objects. Event regions are minimum bounding boxes that objects are interested in. 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 regions 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 via UDP. It is also responsible for event ordering and time synchronization. The interoperability of the service layer allows objects to be migrated at any time. 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.

4 Demonstration Description

Our demonstration includes a prototype P2P MMOG application. Multiple PCs are used to demonstrate the application in a local setting, i.e., all connected to each other with a local area network.

We first demonstrate the data persistency in our system. As peers join and leave the network, data for the game, i.e., spatial objects can be easily maintained. We also demonstrate the fault tolerance of the system with peers that leave abruptly. Failing peers will not disrupt the game because of data replication.

We then demonstrate efficient interactions between peers. Interactions between three or more dynamic objects is resolved using dynamically formed communication cliques.

We finally show the scalability of the approach. Players will be allowed to travel to the different parts of the virtual world and easily locate each other. Using a spatial index provides efficient large scale operations like jumping from one location to another in the virtual world.

Currently, we are in the process of developing a widely available version of this demonstration for Internet users.

References

[1] 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.

[2] 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.

[3] 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*, pages 149–160, San Diego, CA, August 2001.

[4] E. Tanin, A. Harwood, and H. Samet. A distributed quadtree index for peer-to-peer settings. In *IEEE International Conference on Data Engineering*, pages 254–255, Tokyo, Japan, April 2005.

[5] E. Tanin, A. Harwood, H. Samet, S. Nutanong, and M. T. Truong. A serverless 3D world. In *ACM GIS*, pages 157–165, Washington, DC, November 2004.