Mediator: A Design Framework for P2P MMOGs

Lu Fan
School of Mathematical and Computer Sciences
Heriot-Watt University,
Edinburgh, UK
lf16@hw.ac.uk

Hamish Taylor
School of Mathematical and Computer Sciences
Heriot-Watt University,
Edinburgh, UK
h.taylor@hw.ac.uk

Phil Trinder
School of Mathematical and Computer Sciences
Heriot-Watt University,
Edinburgh, UK
p.w.trinder@hw.ac.uk

ABSTRACT

With widespread use of the Internet, Massively Multiplayer Online Games (MMOGs) are becoming increasingly popular. As MMOGs scale up, conventional Client/Server (C/S) architectures exhibit various drawbacks in scalability, reliability, and redundancy. This paper presents a new Peer-to-Peer (P2P) MMOG design framework, Mediator, using a super-peer network with multiple super-peer (Mediator) roles. Mediator is novel in integrating four elements: a reward scheme, distributed resource discovery, load-management and super-peer selection. The reward scheme differentiates a peer's contribution from their reputation, and pursues symmetrical reciprocity as well as discouraging misdemeanours. A deadline-driven auction protocol is proposed for distributed resource discovery. Furthermore, both common-peer and super-peer workloads are approximately balanced using a two-level load-management scheme, and super-peers are selected in a flexible policy-based way. In this framework, the functionalities of a traditional game server are distributed, capitalising on the potential of P2P networks, and enabling the MMOG to scale better in both communication and computation. A proof-of-concept prototype of this framework is described, and ongoing work is discussed.

1. INTRODUCTION

A MMOG allows thousands of players to interact simultaneously in a persistent game world over a network. With widespread use of the Internet, MMOGs are becoming increasingly popular. A recent survey [15] indicates how the number of MMOG subscribers have been increasing at a fast growing rate. Traditionally, MMOGs have been implemented as C/S systems, which offers advantages such as centralised control, better security and simplicity of implementation. Though this widely used architecture is suitable for many types of distributed applications, it still suffers from technical and commercial drawbacks:

1)***Scalability*** - The performance of central game servers are a bottleneck that put a limit on the total number of players a game can accommodate.

2)***Redundancy*** - To ensure that the performance of a game server is powerful enough to handle peak usage rates may result in high hardware redundancy.

3)***Reliability*** - The C/S architecture is not very fault tolerant because servers are single points of failure.

4)***Cost*** - It takes 2 to 3 years and about 10 million dollars to launch a medium-sized MMOG, and ongoing support costs consume up to 80% of its revenues [8].

As discussed above, C/S architecture drawbacks undermine its ability to support large scale, sophisticated, interactive multiplayer online games, and thus much research effort has been put into the design and implementation of P2P MMOGs [6, 18, 3, 5, 16]. P2P architectures suggest a scalable and low cost way for building MMOGs. It enables individuals to deploy public domain MMOG software without the major investment required by previous architectures, and affords business opportunities for enterprises to market game services in a more flexible and profitable fashion, without providing a supporting hardware infrastructure and dedicated maintenance staff.

P2P MMOGs have begun to attract significant academic attention since the early 2000s. Substantial research work has been carried out to address issues like game event dissemination [3, 5, 16], interest management [18, 1], and game state persistency [6, 2]. Though many important issues have been clarified and solved to some extent, little work has been published on the distributed hosting of game objects. As of June 2006, 97.6% of the deployed MMOGs fell into the MMO RPG (Role-playing Game) category, and only about 0.3% fell into MMO FPS (First-person Shooter) [15], which shows the relative importance of MMORPGs. One main difference between them is that the former involves considerable numbers of game objects (a.k.a. non-player characters, or NPCs) such as AI-controlled monsters. Traditionally, NPCs are hosted by a central game server, consuming significant processing power and network bandwidth, and thus account for a major part of the server's workload. In order to support a MMORPG in a P2P network, a scheme is needed for hosting game objects using common game participants' computing resources.

This paper presents a design framework, which takes into consideration a reward-based distributed resource discovery and load-management scheme within super-peer networks, and aims at distributing the functionalities of a traditional game server. The rest of this paper is organized as follows. Section 2 is a brief survey of related work. Section 3 presents the overall design of the Mediator framework. Section 4

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission from the authors.
NetGames'07, September 19-20, 2007, Melbourne, Australia.

introduces a preliminary proof-of-concept prototype that is being built. Section 5 concludes and discusses future work.

2. RELATED WORK

2.1 Game Event Dissemination

Game event dissemination has become a popular research topic, because the bandwidth reduction potential of a P2P network can remove C/S communication bottlenecks at the server side. Fiedler et al. were amongst the first to advocate splitting the game world of a P2P MMOG into smaller pieces and to apply a publisher/subscriber communication model to enhance the scalability of a MMOG [3].

With the publication of a number of P2P overlay infrastructures in the early 2000s, such as Chord [14], CAN [12], Pastry [13] and Tapestry [19], game event dissemination research started to head in two opposite directions. Unstructured approaches [5, 16] emphasised the dynamic grouping of peers, and argued that each peer only needs to establish limited direct P2P connections with neighbouring peers in the same group. In contrast, structured approaches [6, 18] proposed building P2P MMOGs over a P2P overlay infrastructure which provides advantages such as self-organization, scalable Distributed Hash Tables (DHT), efficient routing algorithms, and transparent reconfiguration after node failures. Generally speaking, a structured design can be broken down into three layers:

1)***Structured P2P Overlay Network*** - In this layer, every peer is identified by a random Id generated by a hashing function, and a routing algorithm [14, 12, 13, 19] is employed to route a message from source to destination.

2)***Super-peer Network*** - Within the P2P overlay, some peers are elected to be super-peers, which operate both as a server to a set of clients, and as an equal in a network of super-peers [17].

3)***P2P MMOG Game Zones*** - Every game zone is identified by a zone Id, which serves as a rendezvous point for the peers that play in that zone. The main difference between a peer Id and a zone Id is that the former is random and temporary, whereas the latter is static and well-known by all participants of the application.

Most structured approaches accord with this layered structure, with some flexibilities, e.g. how and for what purpose a super-peer is elected.

2.2 Interest Management

Interest Management (IM) is a classical research topic in Networked Virtual Environments (NVE), and was initially addressed by Morse et al. in the mid 1990s [10]. Various IM algorithms have been proposed in the literature, and their performances are compared in [1].

The MOPAR infrastructure [18] is representative of recent work on distributed IM, which is especially beneficial to the design and implementation of P2P MMOGs. Unlike previous related work, MOPAR relies on a hybrid communication scheme that combines a DHT and direct P2P connections. It is argued that though the DHT can facilitate the maintenance of the game zone structure, it will introduce a considerable amount of communication overhead. Instead, MOPAR devises a completely distributed IM algorithm, which requires each peer to establish direct P2P connections to a small quantity of other peers on demand, for exchanging time critical information.

2.3 Game State Persistency

A MMOG is also called a Persistent World, because of the maintaining and developing of the game world around the clock. So, the problem "how is the persistent data stored, updated, and reloaded?" immediately presents itself to game architects who aim to deploy a P2P MMOG without a centralized game server.

Scott et al. propose that the entire virtual world and the game logic be combined into an entity database distributed over all peers, and adopt an agent-based approach for efficient communication and processing among strongly interacting entities [2]. Takuji et al. propose a zoned federation model, in which a zone owner is elected in each zone and works in the same way as a centralized authoritative server while it is connected to the world [6].

3. THE MEDIATOR FRAMEWORK

3.1 Overview

The Mediator framework adopts a hybrid communication architecture - a structured P2P overlay is used in the peer bootstrapping process, application level multicast is used for efficient game zone structure maintenance, and direct P2P connections are established on demand for the dissemination of time critical events. Secondly, the framework adopts quadrant zoning of a single-realm MMOG to faciliate load-management based upon dynamic zoning, because rectangles are easy to divide and combine. Thirdly, similar to previous structured related work, peers are organized into a hierarchical super-peer network. However, a major novelty here is that multiple super-peer roles are used in each game zone to carry out various management tasks, and some super-peer roles may have multiple instances. A super-peer is also called a Mediator, and four preliminary Mediator roles have been chosen in the current design:

1)***Boot Mediator (BM)*** - A BM is the peer whose peer Id is numerically closest to a zone Id in the P2P overlay, and it is responsible for handling bootstrapping messages. A BM works in a similar way to a Home Node in MOPAR.

2)***Resource Mediator (RM)*** - A RM is a super-peer that is responsible for the distributed resource discovery and match-making job, whose working protocol is discussed in section 3.3. In each game zone, multiple RM instances may coexist and work in parallel.

3)***Interest-Management Mediator (IMM)*** - As indicated by its name, an IMM is responsible for the interest-management job. Its working protocol is like but is more complex than a Master Node in MOPAR (details about the collaboration between the IMM and the RM are given in section 3.3).

4)***Zone Mediator (ZM)*** - A ZM is responsible for the selection of other super-peers and for roughly balancing out the workload among multiple super-peer instances. Moreover, a ZM also monitors the working state of other super-peers within the game zone, and replaces failed super-peers with capable backups in good time. More details about the ZM working protocol are given in section 3.4.

The contribution of the Mediator framework lies in providing a way of distributing the functionalities of a traditional game server in a P2P network, especially the hosting of non-player game objects, which is a significant issue but has been addressed by little related work. The framework is flexible and extensible. On the one hand,

new Mediator roles can be easily introduced into the framework according to newly identified requirements, and on the other hand, the framework is compatible with various distributed anti-cheating, interest management, reputation management, and anti-free-riding algorithms. The Mediator framework comprises four sub-topics that are discussed from section 3.2 to 3.5.

3.2 Reward Scheme

3.2.1 Rationale of the reward scheme

Well-known P2P applications, such as Napster, Gnutella and Bit torrent show that P2P systems are by nature voluntary resource sharing systems, in which there is always a tension between individual concerns and collective welfare. Due to this characteristic, Mediator adopts a pull-based scheduling strategy which is more appropriate for maintaining the viability of a P2P MMOG. In this case, a reward scheme is crucial to keep a record of the resources that a peer has contributed to the system, and accordingly, to entitle the peer to consume roughly equivalent resources from other peers. Thus, selfish peers can be identified and discouraged, and a sufficient level of resource sharing can be ensured to make use of the P2P application beneficial.

3.2.2 Design of the reward scheme

Determining how contributors are rewarded raises two sub-issues: the quantification and the qualification of contributions. Firstly, Mediator quantifies contributions in two different ways. On the one hand, common peers can contribute their computing resources by pulling and hosting game objects. Such jobs come with a computational complexity, which can be measured, and a contributor rewarded in a per-job fashion. However, on the other hand, a Mediator's workload may vary over time, so it is relatively harder to trace and measure. In this case, it is proposed to reward Mediators purely according to their hardware configuration and online time. Therefore, strong peers should be motivated to take Mediator jobs, as well as to stay online for a longer time.

Secondly, because a contribution scheme is only an accounting mechanism that facilitates symmetrical reciprocity, it is inadequate in discouraging disadvantageous behaviour, e.g. a player pretends to have a stronger machine in order to earn more contribution points, but the machine is overloaded by excessive tasks and consequently degrades other players' gaming experiences; or, a Mediator disconnects from the system abruptly, putting the system into an inconsistent state which takes much time and inconvenience to recover from. In this case, a reputation scheme becomes necessary for qualifying a peer's resources and creditability. By tracking a peer's historical behaviour, its overall manner towards P2P collaborations can be made accountable for its positive or negative contribution to the system.

Thus, Mediator rewards contributors by both increasing their reputation score and their contribution points. The former is helpful to promote future "sales" of their resources, and the latter entitles them to consume equivalent resources from other peers. It is important to notice that the reputation and contribution scores can be used in various beneficial ways rather than only within the reward scheme, e.g. the resource discovery scheme described in the next section attempts to enhance the gaming experience, such as minimizing latency, of favourable peers with good reputation and high contribution scores.

Currently, the Mediator framework recommends the EigenTrust reputation management algorithm [7] and the DCRC anti-free-riding algorithm [4] as possible implementations for the reward scheme. Though originally the two algorithms were designed for P2P file-sharing applications, they can be easily adapted and used in a P2P MMOG.

Resource Ad

[**Type** = String: *"Resource"*
Owner = int: peer id **PRI** = int: [0,100]
Rep = int: [-100,100]
CPU = double: free processing power
RAM = double: free memory (KB)
BW = double: free bandwidth (Kbps)
Mini = int: minimum reward accepted
LV = String: serialized latency vector
Fav = String: serialized friends' ids
Rank = int: Other.Reward
Requirements = boolean: (Other.Owner != Owner) && (Other.CPU*Other.Amount <= CPU) && (Other.BW*Other.Amount <= BW) && (Other.RAM*Other.Amount <= RAM) && (Other.Reward >= Mini)
Latency = int: latency value
Preference = int: Other.Rank]

Job Ad

[**Type** = String: *"Job"*
Owner = int: peer id
TTL = int: TTL of the object (minute)
Deadline = String: auction deadline
TimeStamp = String: issue time
Mini = int: minimum Rep required
ObjectType = String: object type
Amount = int: number of the object
CPU = double: processing power
RAM = double: memory (KB)
BW = double: bandwidth (Kbps)
Reward = int: round((BW /*BWFactor* + RAM/*RAMFactor* +CPU/*CPUFactor*) * TTL * Amount)
Rank = int: round(Other.PRI + Other.Rep - Other.Latency /*LatencyFactor*)
Requirements = boolean: Other.Rep >= Mini]

Figure 1: Structures of Resource Ad and Job Ad

3.3 Distributed Resource Discovery

3.3.1 Condor-like opportunistic match-making

Mediator adopts a Condor-like match-making approach for resource discovery, and represents both the available resources and the job requests using ClassAds [11]. Though, Condor and Mediator share being pull-based and opportunistic [11], a major difference is that the former is static and centralized, whereas the latter is dynamic, decentralized and features multiple match-makers working in parallel. Figure 1 shows the structure of a Mediator Resource Ad and Job Ad.

1) ***Descriptions of the Resource Ad*** - The **Owner** field contains a resource provider's Id as a 128bit code. The **PRI** and **Rep** fields respectively indicate the resource provider's current contribution points and reputation score. The **CPU**, **RAM**, and **BW** fields reflect a peer's available processing power[1], memory (KB) and network bandwidth (Kbps). A resource provider is able to specify a minimum job weight that it accepts in the **Mini** field, because on the one hand a powerful provider may not want to be bothered by tiny jobs, and on the other hand, less competent providers would be starved of contribution opportunities if all jobs in their power were seized by stronger peers. The **LV** field is a serialized HashMap with peer Ids as keywords, and latency values as contents (more details are discussed in section 3.3.2). The **Rank** and **Requirements** fields are required by the ClassAds match-making mechanism. By the evaluation of those two fields, it is determined whether a given resource provider is able to host a specific job or not, and to what extent both

[1]A benchmark-based way of evaluating the performance of a CPU product is suggested for simplicity and compatibility.

parties are a good match for each other. The last two fields, **Latency** and **Preference** are completed by the RM that carried out the match-making. A RM finishes the Resource Ad from the selected resource provider, and passes it to the IMM as a bid for the corresponding job (more details are discussed in section 3.3.2).

2)***Descriptions of the Job Ad*** - The **Owner** field reflects a peer Id, for whom RMs should minimize latency while selecting resource providers. The owner of a specific job is determined by the IMM that issued the Job Ad, according to a game event prediction algorithm which is tightly related to in-game logics and beyond the scope of this paper. The **TTL**, **Amount**, and **ObjectType** fields respectively represent the life time, the cardinality, and the type of a game object. These three fields will be used for a selected resource provider to initialize the game objects. The IMM can specify a **Mini** field in a Job Ad to restrict the minimum reputation score for a required resource provider, according to the significance of the game object to be hosted. The **Deadline** and **TimeStamp** fields are utilized by the deadline-driven auction protocol. Finally, **CPU**, **RAM** and **BW** fields describe the weight of a job.

3.3.2 Deadline-driven Auction (DA) Protocol

Figure 2 represents the overall flow of control of the framework, where sub-regions are labelled by circled numbers to establish a connection between the diagram and corresponding texts that explain it. As discussed in section 3.1, a BM and an IMM work in a similar way to a Home Node and a Master Node in MOPAR. At the bootstrapping stage, a peer routes an enquiry message to the target Zone Id through the structured P2P overlay, which will be received and handled by the BM for that zone ⓪. If a ZM already existed in the zone, the BM would reply to the enquiry with the current ZM's Id. Otherwise, the BM would promote the peer to be a provisional ZM ①, with a more capable ZM selected later on. Once a peer has successfully joined a game zone, it updates the IMM when its moving speed or direction are changed. In this way, the IMM obtains a global view of the zone-level game world, and is able to predict every peer's position in the near future. Peers, whose Area of Interest (AOI) are going to overlap, will be notified by the IMM to establish direct P2P connections with each other, getting prepared for potential real-time interactions.

In MOPAR, a Master Node is only required to predict player-vs.-player interactions. However, Mediator also requires an IMM to predict player-vs.-object events ⑨, e.g. a player avatar is approaching a hidden spawn point of monsters and will trigger a game event that releases three dragons in the next five seconds. In abstract, given a frame interval T_F, the IMM will predict the game events that satisfy $E_t \in [T_F, 2T_F]$, where E_t is the estimated period of time before event E actually takes place, as well as a deadline before which the IMM must locate a capable resource provider to host the game objects involved in event E, through cooperation with multiple RMs ⑧. The process for locating a resource provider is directed by the DA protocol:

I. After the bootstrapping stage, a peer enrols at the ZM to become a zone member. The ZM responds with an empty Latency Vector (LV) and the communication endpoint of a selected RM ②. The empty LV contains a list of all existing zone members, but with all latency values set to infinity. The peer then starts to ping every other zone member to complete the LV. More up-to-date empty LVs will be issued by the ZM via a zone-level multicast for zone structure maintenance purposes.

II. Once the peer has finished pinging a majority of the zone members, it may create a Resource Ad according to local load-management policy and upload it to the RM assigned by the ZM. Because what computing resources are available is transient information, the peer should update the RM when its resource availability is changed ③.

III. At the RM end, large numbers of Resource Ads may be received from peers that are currently attached to it. These Resource Ads are queued and wait to be accepted. Every time a RM receives a Resource Ad, it inspects the queue. If an existing Resource Ad from the same peer is identified, the previous Ad would be replaced by the new one. The RM also monitors the TTL of each entry in the queue, and removes stale Resource Ads ④.

IV. At the IMM end, game events are predicted, and Job Ads are created and delivered to multiple RM instances within the game zone. When an RM receives a Job Ad, it estimates how much time is left for match-making by subtracting the round-trip-time (RTT) from the deadline indicated by the Job Ad, and then finds out a most suitable resource provider from the local resource queue in a best efforts manner ⑤. Because there will be real-time interaction between the selected resource provider and the resource consumer, the word "suitable" here mainly refers to a minimum latency, accompanied by other considerations such as a resource provider's reputation and contribution. When it reaches the deadline for the match-making or the RM has finished examining all existing Resource Ads in the queue, a resource provider with the highest **Preference** value is selected as the most suitable one.

V. Next, the RM completes the selected Resource Ad and uploads it to the IMM as a bid for the job. Because the IMM only aims at employing the best resource provider, it becomes an auction among the multiple RMs, and any less preferred bid is rejected immediately by the IMM ⑥. In other words, the worst case for a RM is to wait for $E_t - \frac{RTT}{2}$ time before confirming that its bid has won the auction.

VI. Finally, the winning RM passes the Job Ad to the selected resource provider, and the latter will initialize the game objects accordingly ⑦. If the game objects are stateless AI-controlled monsters, the resource provider can load the AI programs from a local copy of the game logic. Otherwise, if the game objects are stateful, the resource provider may acquire related information through a P2P MMOG persistency scheme such as was discussed in section 2.3.

3.4 Load-management

3.4.1 Peer-level Load-management

Because Mediator adopts a pull-based scheduling strategy, every individual peer monitors its own computing resources and decides at what time to ask for new jobs, and how many jobs to ask for. The set of configuration parameters related to peer level scheduling is called the Contribution Enthusiasm, which controls the following aspects of a peer's local scheduling behavior:

1)***The amount of dedicated resources*** - A player is able to specify what percentage of its computing resources it would like to contribute to a P2P MMOG. To encourage voluntary contribution, the framework allows a player to

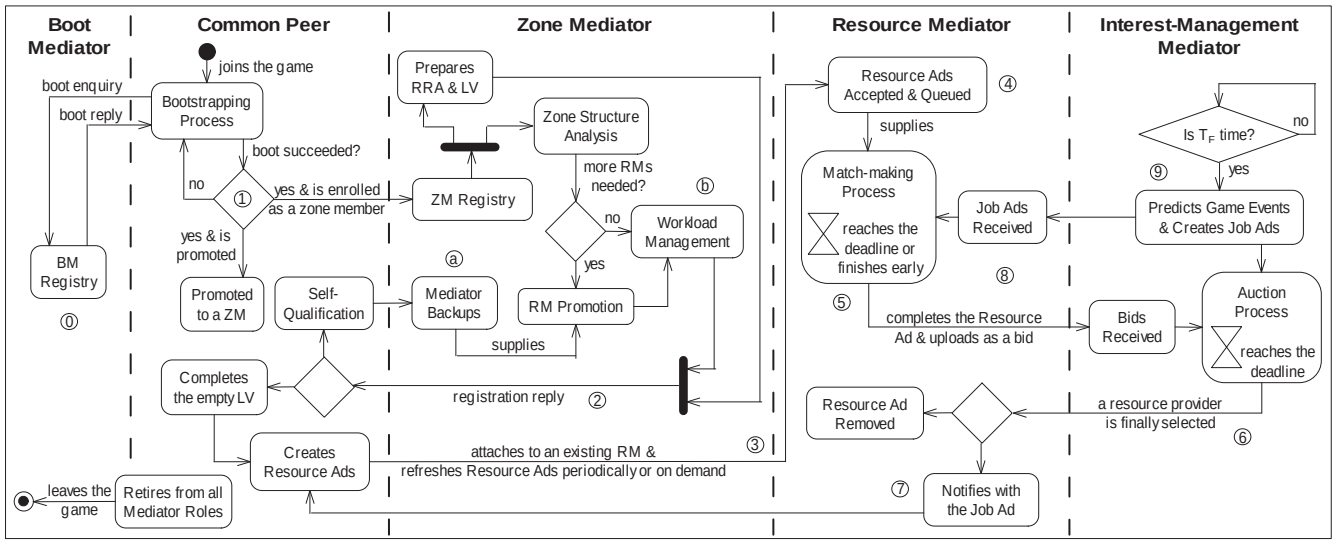

Figure 2: Activity diagram for a typical gaming session

join the application as a dedicated contributor, when the player is not playing the game but the player's computing resources are being utilized.

2)***Pulling frequency and job selection*** - A peer can determine how often it uploads Resource Ads to a RM, and specify which job types it prefers, accepts, and refuses. The framework encourages stronger peers to seek a large bundle of jobs or to work as a super-peer, and leave the lesser jobs to weaker or slower to access peers.

3)***Super-peer role solicitation*** - A peer can specify whether it is interested in super-peer jobs. If it is interested, the peer can evaluate its hardware configuration against the Reference Resource Ad (RRA) that is periodically issued by the ZM, to find out whether it is presently qualified for a super-peer role. According to the self-qualification result, the peer decides whether to register at the ZM as a super-peer backup. Generally speaking, a super-peer job would be better rewarded than a common resource contributor. However, a super-peer is required to commit to providing the service for a long enough gaming session, and if the peer violates this commitment, its reputation will be diminished.

3.4.2 Zone-level Load-management

The zone-level load-management mechanism is designed to balance super-peer workloads approximately. Because a ZM is responsible for promoting various super-peers in the framework, zone-level load-management is mainly embodied by the ZM decision-making algorithms to promote adequate super-peers in a game zone, and prevent any super-peer from becoming overloaded.

On the one hand, a ZM should maintain the super-peer to common peer ratio in a game zone to approximately a constant. Every time a new peer joins its zone, the ZM evaluates whether more super-peers are needed. If the answer is positive, the ZM would promote a given number of super-peers from a super-peer backup queue ⓐ. Moreover, it is also necessary for a ZM to monitor the length of the backup queue, in order to protect a game zone from super-peer paucity. The ZM will also regulate the length of the backup queue by tuning the super-peer qualification criteria specified in each RRA that it issues.

On the other hand, each working super-peer should periodically report its workload to the ZM, and the ZM will either shift part of a heavy-loaded super-peer's work to other super-peers, or just assign the lightly-loaded super-peers to new participants from then on ⓑ. Workload information can be delivered to a ZM when a super-peer periodically refreshes the ZM with knowledge of its existence. If a super-peer fails to report for a sufficient period of time, the ZM will assume that the super-peer has left the game zone silently, and replace the super-peer with a capable backup.

3.5 Super-peer Selection

Super-peer selection is a common problem that emerges across a variety of P2P applications with many different selection protocols presented in the literature [17, 9]. In the Mediator framework, except for the BM that is a light-weight super-peer selected in a structured way, other Mediator roles are selected using a policy-based approach. Currently, the framework just suggests some fundamental policies and more application specific ones can be flexibly introduced by different P2P MMOG designs.

Policy 1 Only qualified, high performance peers can be selected as Mediators, unless the zone is bootstrapping when any peer can be selected as a provisional Zone Mediator, or during a resource paucity period when super-peer qualification criteria are temporally lowered.

Policy 2 A ZM should avoid assigning multiple Mediator roles to a single peer, if other qualified candidates are available, unless the peer works as the BM for multiple game zones, or happens to be both a Mediator for its native zone, and a BM for a foreign zone.

Policy 3 A relatively stable ratio between common peers and super-peers should be maintained, in order to prevent, on the one hand, any super-peer from being overloaded, and on the other hand, too many super-peers being promoted and sitting idle.

4. IMPLEMENTATION

A proof-of-concept prototype for the Mediator framework is being built using FreePastry 2.0 and the ClassAds Java library 2.2. The prototype employs the Direct Simulator integrated in the FreePastry package, and has successfully simulated 1000 peers on an AMD 64 3000+ workstation with 2GB memory. The simulator takes in a table of float latency values that is generated by a standard network topology generator such as GT-ITM. According to the topology file, both the direct latency between two peers and the accumulated latency for routing a message through the P2P overlay are simulated. In the prototype, each peer is assigned a certain amount of virtual computing resources, and the local scheduling behaviours are simulated by a resource monitor stub. Preliminary experiment results indicate that various Mediator roles are allocated as peers join the application, zone structures are maintained correctly when peers moving from one zone to another, and the virtual computing resources owned by each peer are dynamically consumed by satisfying job requests.

5. CONCLUSION AND FUTURE WORK

This paper presents Mediator, a novel P2P MMOG design framework that addresses four sub-issues - the reward scheme, distributed resource discovery, load-management, and super-peer selection. These issues are crucial to distribute the functionalities of a traditional game server , but have not been much addressed in the literature. In Mediator, game objects are hosted using player machines' computing resources, and hence the MMOG scales better in both communication and computation. One avenue of future work is to complete experiments to validate and measure:

1)***Effectiveness*** - Given a deadline (T_F) that is sufficiently long, a resource provider with an optimal **Preference** value should be located.

2)***Efficiency*** - There should be a direct proportionality between T_F and the **Preference** value, so that an efficient T_F can be predicted to afford an acceptable **Preference**.

3)***Scalability*** - All experiments will be repeated several times against different peer populations, to test whether the framework is scalable or not.

4)***Robustness*** - Exceptional events, e.g. Mediators stopping functioning, will be simulated to test whether the framework is robust enough to recover from potential failures.

The current prototype does not support load-management for Zone and Interest Management Mediators and future work will investigate algorithms for dynamic zoning, so that overcrowded game zones can be divided into disjoint or even parallel sub-zones, each with its own ZM and IMM. Similarly, sparsely populated game zones can be combined into a single zone that is controlled by a joint ZM and IMM.

6. REFERENCES

[1] J. Boulanger, J. Kienzle, and C. Verbrugge. Comparing Interest Management Algorithms for Massively Multiplayer Games. In *Proceedings of the 5th NetGames workshop*. ACM, 2006.

[2] S. Douglas, E. Tanin, and A. Harwood. Enabling Massively Multi-Player Online Gaming Applications on a P2P Architecture. In *Proceedings of the IEEE International Conference on Information and Automation*, pages 7–12. IEEE, 2005.

[3] S. Fiedler, M. Wallner, and M. Weber. A communication architecture for massive multiplayer games. In *Proceedings of the 1st NetGames workshop*, pages 14–22. ACM, 2002.

[4] M. Gupta, P. Judge, and M. Ammar. A reputation system for peer-to-peer networks. In *Proceedings of the 13th NOSSDAV workshop*, pages 144–152. ACM, 2003.

[5] S.-Y. Hu and G.-M. Liao. Scalable peer-to-peer networked virtual environment. In *Proceedings of the 3rd NetGames workshop*, pages 129–133. ACM, 2004.

[6] T. Iimura, H. Hazeyama, and Y. Kadobayashi. Zoned federation of game servers: a p2p approach to scalable multi-player online games. In *Proceedings of the 3rd NetGames workshop*, pages 116–120. ACM, 2004.

[7] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The EigenTrust Algorithm for Reputation Management in P2P Networks. In *Proceedings of the 12th WWW Conference*, pages 640–651. ACM, 2003.

[8] J. Kesselman. Server Architectures for Massively Multiplayer Online Games. In *Session TS-1084, Javaone conference*. SUN, 2005.

[9] V. Lo, D. Zhou, Y. Liu, C. G. Dickey, and J. Li. Scalable supernode selection in peer-to-peer overlay networks. In *Proceeding of the 2nd HOT-P2P Workshop*, pages 18–25. IEEE, 2005.

[10] K. L. Morse. Interest management in large-scale distributed simulations. Technical report, University of California, Irvine, CA, 1996.

[11] R. Raman, M. Livny, and M. Solomon. Resource Management through Multilateral Matchmaking. In *Proceedings of the 9th IEEE Symposium on HPDC*, pages 290–291. IEEE, August 2000.

[12] S. Ratnasamy, P. Francis, M. Handly, R. Karp, and S. Schenker. A scalable content-addressable network. In *Proceedings of SIGCOMM'01*, pages 161–172. ACM, 2001.

[13] A. Rowstron and P. Druschel. Pastry: scalable, decentralized object location and routing for large scale peer-to-peer systems. In *Proceedings of 18th IFIP/ACM Middleware*, pages 329–350. ACM, 2001.

[14] I. Stoica, R. Morris, D. Karger, and F. Kaashoek. Chord: A scalable peer-to-peer lookup service for Internet applications. In *Proceedings of the 1st NetGames workshop*, pages 149–160. ACM, 2001.

[15] B. S. Woodcock. An analysis of MMOG subscription growth. Technical report, www.mmogchart.com, 2006.

[16] S. Yamamoto, Y. Murata, K. Yasumoto, and M. Ito. A Distributed Event Delivery Method with Load Balancing for MMORPG. In *Proceedings of the 4th NetGames workshop*, pages 1–8. ACM, 2005.

[17] B. B. Yang and H. Garcia-Molina. Designing a super-peer network. In *Proceedings of ICDE'03*, pages 49–60. IEEE, 2003.

[18] A. P. Yu and S. T.Vuong. MOPAR: a mobile peer-to-peer overlay architecture for interest management of massively multiplayer online games. In *Proceedings of the 15th NOSSDAV workshop*, pages 99–104. ACM, 2005.

[19] B. Zhao, J. Kubiatowicz, and A. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical report, UC Berkeley, 2001.