Deadline-Driven Auctions for NPC Host Allocation in P2P MMOGs

Lu Fan*, Phil Trinder and Hamish Taylor
School of Mathematical and Computer Sciences,
Heriot-Watt University, Edinburgh, UK
E-mail: {lf16, p.w.trinder, h.taylor}@hw.ac.uk
*Corresponding author

Abstract:
This paper presents the design, implementation and evaluation of Deadline-Driven Auctions (DDA), a novel task mapping infrastructure for heterogeneous distributed environments. DDA is primarily designed to support the hosting of Non-Player Characters (NPCs) in P2P Massively Multiplayer Online Games (MMOGs). Experimental and analytical results demonstrate that DDA provides four significant advantages. It is self-organising as the infrastructure can be automatically assembled and managed. It efficiently allocates computing resources for large numbers of real-time NPC tasks in a simulated P2P MMOG with the better part of 1000 players. It supports gaming interactivity by keeping the communication latency among NPC hosts and ordinary players low. Finally, it supports flexible matchmaking policies, and with a friendly incentive policy, can establish a cooperative economic model that helps motivate participants to contribute their resources to the system.

Keywords: NPC host allocation; P2P; MMOGs; real-time; task mapping; heterogeneous environments; communication latency; matchmaking; incentive; simulation.

1 INTRODUCTION

A Virtual Environment (VE) is a computer simulated environment for its users to inhabit and interact via avatars. Many VEs are Massively Multiplayer Online Games (MMOGs). Conventionally, VEs and MMOGs have been implemented using Client/Server (C/S) architectures, because they are relatively easy to implement and secure (Mulligan and Patrovsky, 2003). However, they also exhibit various drawbacks in reliability and cost (Fan et al., 2007), hence Peer-to-Peer (P2P) MMOGs are becoming increasingly attractive as an alternative (Hu et al., 2008).

To adapt MMOGs to P2P architectures, many key issues have to be addressed. A MMOG is different from a typical VE in that it features considerable numbers of non-player characters (NPCs). These NPCs need to be hosted by peers to be run (Bharambe, 2006; Yonekura et al., 2004; Hu et al., 2008). A NPC is an AI-controlled virtual actor, which drives storylines in a game, or combats player characters (PCs) as a monster. MMOGs have to supply their game worlds with large numbers of NPCs as required by game scenarios. Traditionally, NPCs are hosted by game servers, consuming significant processing power and network bandwidth. Therefore, one of the prerequisites for realizing a P2P MMOG is to provide a mechanism that hosts such NPCs using resources available on game participant machines.

Conceptually, a NPC object is an indivisible, computational and interactive task, because:

1. It can only be efficiently hosted by a single computer.

2. It consumes processing power, as a NPC is associated with an AI program that determines the NPC's actions. For example, a monster NPC may determine its move by examining the position of nearby players.

3. It needs to interact with players. For example, a monster NPC may engage in combat with a set of players, and exchange real-time gaming events with them.

A mechanism that schedules NPC tasks on game participant machines is referred to as NPC host allocation. The responsibilities of such a mechanism include the discovery of potential resource providers, the selection of suitable NPC hosts, and the migration of related AI program and state information for a NPC. Among them, the most important one is the selection of a suitable host. Firstly, the selected host must have adequate computing resource, such as CPU cycles, memory and network bandwidth. Secondly, it should also offer a low communication latency to guarantee the gaming experiences of other players that interact with the NPC. This determination must be made in

due course, because the fast pace of a MMOG requires a NPC to appear at a specific position and start working in the order of seconds.

The contribution of this paper is to present the design (Section 3), analysis (Section 4), extensions (Section 5), implementation using Pastry (Rowstron and Druschel, 2001) (Section 6), and evaluation (Section 7) of Deadline-Driven Auctions (DDA). DDA supports real-time NPC host allocation in P2P MMOGs using a heterogeneous task mapping mechanism. DDA's infrastructure is a self-organised super-peer network on top of a structured P2P overlay. This infrastructure is provided by our previous research on the Mediator framework (Fan et al., 2007).

Experimental results demonstrate that DDA provides the following advantages:

1) ***Self-Organisation***: DDA's infrastructure can be automatically assembled and managed. As the system evolves, NPC tasks will be distributed to suitable participants automatically (Section 7.1).

2) ***Real-time Resource Allocation***: DDA efficiently processes large numbers of tasks within a few seconds, and is able to support a simulated P2P MMOG with the better part of 1000 players (Section 7.2).

3) ***QoS Desiderata***: DDA minimises the communication latency among a NPC host and ordinary players, so as to improve interactivity for a P2P MMOG (Section 7.2).

4) ***Cooperative Economic Model***: DDA supports flexible resource selection policies, and it conveniently establishes a cooperative economic model that shares NPC tasks among competent resource providers fairly (Section 5.2).

2 RELATED WORK

Existing NPC host allocation schemes can be classified into static region based approaches and dynamic virtual distance based approaches. The former (Lu et al., 2004; Iimura et al., 2004) partition a game world into multiple regions, and assign each region a super-peer, which works as an authoritative server and hosts all the NPCs within the region. They have several drawbacks. Because only one super-peer is selected to take charge of a region, they might incur excessive computation and communication workloads on the super-peer. Their proposed super-peer selection criteria are also overly simple, as they do not take into consideration peers' actual resource availabilities. Furthermore, the approaches cannot guarantee to fulfil the QoS requirement for game interactivity.

In contrast, dynamic virtual distance based approaches distribute NPC objects to all game participants, where the key idea is to allocate a NPC to the machine of the player, whose avatar is closest to the NPC. Because a player that is closest to a NPC is most likely to interact with it, if the player is hosting the NPC by itself, there is no need for the player to communicate with a remote third party. It has been suggested that this is optimal for minimizing interactive latency and communication overhead (Bharambe, 2006).

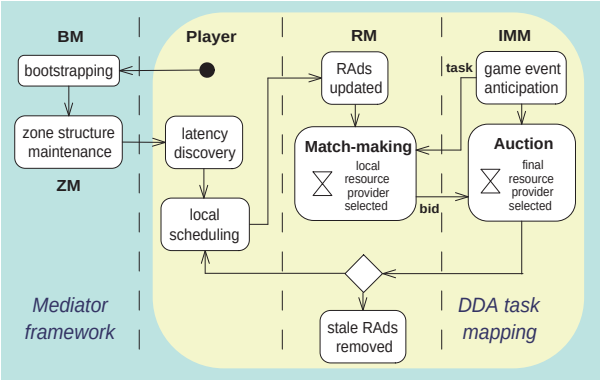

Figure 1: The Mediator framework & DDA.

Colyseus (Bharambe, 2006) has demonstrated the feasibility of virtual distance based object hosting in Quake II. The game object manager of Colyseus allocates mutable objects, e.g. NPCs, doors and weapon items, to the closest players. Similarly, AtoZ (Yonekura et al., 2004) allocates each player avatar a "priority field", which is analogous to the Mahalanobis distance in the domain of quadratic discriminant analysis (Anderson, 1984) to decide which player can access a shared object in the shortest time. Furthermore, the Voronoi diagram discussed in (Hu et al., 2008) seems inherently suitable for virtual distance based NPC host allocation. A Voronoi diagram partitions a game world into multiple non-overlapping regions that contain exactly one player avatar in each region. In this case, it is natural for each player to host the NPC objects within its own Voronoi cell.

Compared to static region based approaches, dynamic virtual distance based approaches are better at utilizing the computing resources of more participant machines. However, they also have the following disadvantages:

1. Some NPCs like shop owners may only need to be present to one player at a time. This might be enforced by environments that limit the NPC's visability and interactivity to one player at a time. Such NPCs would be best hosted by a player's own machine. However, most NPCs are not like this. They can have a real-time effect on players on several hosts at a time. In this case, all the players need to communicate with the NPC host, and virtual distance based approaches cannot ensure that the latency for each player is equally low.

2. The computation of accurate NPC host allocation can be expensive, and because a large proportion of the players in a MMOG are constantly moving, switches of host may be frequent. Therefore, the overall computation and communication overhead may be still high.

3. Cheating may become easier for unscrupulous players who might abuse their hosting of NPC objects to their own advantage. Even worse, because no third party is required in a local interaction, it is rather hard to detect such a breach.

The DDA infrastructure adopts a task mapping mechanism that is different from all the related work. DDA is good at load-balancing, because it takes into consideration each game participant's actual resource availability and ensures that NPC tasks are always allocated to capable hosts. Also once a NPC is allocated to a game participant, the hosting relationship remains stable, unless the NPC is destroyed, or the host needs to leave the system. Therefore, NPC task migration among hosts is less frequent than in virtual distance based approaches. Furthermore, because a player is unlikely to host a NPC for itself, cheating is relatively harder in DDA. DDA also provides several additional advantages such as self-organisation, real-time resource allocation, QoS support and a cooperative economic model.

3 DDA DESIGN

3.1 Overview

The system model for DDA involves two different parties: a work source and a set of resource providers. The work source is the virtual game world that constantly generates NPC tasks. The resource providers are game participants that have spare resource on their machines. The key idea in DDA is to bridge between resource requirements and their availability using distributed matchmaking via resource matchmakers.

Because a P2P system lacks centralized configuration, a self-organizing mechanism is needed to set up DDA's working infrastructure. Previous work on the Mediator framework (Fan et al., 2007) supplies such an infrastructure. It automatically organizes application participants into a hierarchical super-peer network using a structured P2P overlay (Rowstron and Druschel, 2001). Four super-peer roles have been defined in the framework, Boot Mediator (BM), Zone Mediator (ZM), Resource Mediator (RM) and Interest-Management Mediator (IMM).

The collaboration among the Mediator roles is illustrated by Figure 1. Firstly, the entire game world is partitioned into multiple game zones, and a BM is selected in each zone for bootstrapping purposes. Secondly, the BM promotes a resource-rich peer from existing zone members to be the ZM for that zone with responsibility for zone structure maintenance, e.g. the selection and monitoring of the IMM and multiple RMs. Thirdly, an IMM is essentially a gaming event anticipator. By running a hybrid interest-management scheme, e.g. MOPAR (Yu and T.Vuong, 2005), the IMM respawns NPC objects and updates other zone members with forthcoming gaming events. Finally, RMs serve as matchmakers that facilitate real-time resource discovery. For load-balancing and performance reasons, multiple RMs are selected in each game zone and work in parallel. Once a NPC is about to be respawned, the IMM notifies the RMs in the game zone, and each RM replies with a locally optimised resource provider as a "bid". When the IMM has received

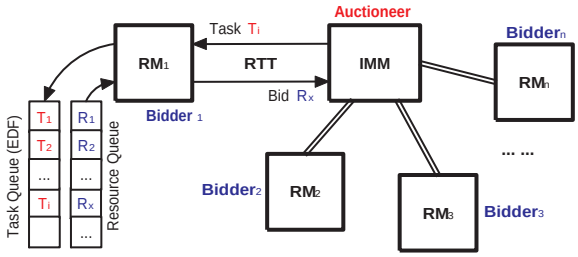

Figure 2: DDA zone-level scheduling.

their bids, or the deadline for the spawning task is sufficiently near, the IMM closes the "auction" and determines which resource provider hosts the NPC.

3.2 Local Scheduling

Local scheduling refers to the activities carried out by a peer to manage its local computing resources, and to contribute spare resource to the system. Typical local scheduling activities include finding out communication latencies with other zone members, and updating resource availabilities at an RM.

DDA resorts to incentive mechanisms to convince participants to contribute their resources. DDA uses DCRC (Gupta et al., 2003) to quantify peers' contributions, so free-riders can be identified and discouraged. DDA charges peers in term of credits, according to the time that they play the game. Ideally, a peer that actively contributes usable resources to the system should be able to earn enough credits to pay for its playing time. However, if resource-rich peers eagerly volunteer for tasks, less competitive peers will be starved of opportunities for earning credits. As a result, some virtuous peers may become poorer and poorer, and finally be regarded as free-riders. To cope with this problem, DDA adopts a friendly matchmaking policy (Section 5.2) that fairly shares credit earning opportunities among all competent resource providers.

Furthermore, only to quantify available resources is not sufficient. In order to fulfil the QoS requirement for game interactivity, a peer also needs to qualify its resources by finding out its communication latencies with other peers. For zone structure maintenance purposes, the ZM for each game zone publishes heartbeat messages periodically. Such messages carry a list of existing zone members, so that a peer can check whether new zone members have shown up. If so, the peer sends out a set of `Ping` messages to them, and they reply with `Pong` messages, so that communication latencies among them can be measured.

DDA adopts a matchmaking approach similar to Condor (Litzkow et al., 1988), and represents both the resource offers and NPC tasks using ClassAds (Raman et al., 2000). A peer periodically reports its local resource quantity and quality to an RM using a Resource Ad (RAd), and an IMM notifies related RMs about a NPC task using a Job Ad (JAd). Detailed structures for a RAd and a JAd can be found in (Fan et al., 2007).

Currently, cheating mitigation is not a primary focus of DDA's design, but obviously it is possible for unscrupulous peers to cheat on local scheduling. For example, in order to earn more credits, a peer may lie about the spare computing resource it can provide. As a result, the peer will be allocated tasks that are beyond its capability, which consequently damages other peers' gaming experiences. One avenue of future work is to enhance DDA's security by discouraging disadvantageous peer behaviour using a reputation system.

3.3 Zone-Level Scheduling

Figure 2 depicts DDA's zone-level scheduling, where an IMM works as an auctioneer, and multiple RMs serve as bidders. Each RM maintains two queues: a resource queue (RQ) for storing RAds from resource providers, and a task queue (TQ) for buffering JAds given by the IMM. A TQ suits being made a priority queue, where tasks are ordered on an earliest deadline first (EDF). The processing cycle of the RM involves selecting the task with the shortest deadline from the TQ, scanning the RQ to find the most adequate resource provider, and returning it to the IMM as a bid. From the IMM's perspective, every task T_i must be completed within its deadline D_i. This requirement is captured in Equation 1, in which $C(T_i)$ means the completion time for task i.

$$\forall_{i \in 1...n_\bullet} C(T_i) < D_i \tag{1}$$

According to Figure 2, $C(T_i)$ comprises two periods of time: the round-trip time (RTT) between the IMM and an RM, and the time that an RM takes to discover a locally optimised resource provider. To simplify the analysis, it is assumed that by carefully selecting super-peers from candidates, each RM has an equally short RTT with the IMM. Furthermore, the RQs maintained by different RMs are equally long, each containing l RAds. If it takes t milliseconds for an RM to match one RAd to the JAd, it will take $l * t$ milliseconds in total for the RM to complete matching all the RAds to decide which one is the best. Some preordering of RAds might reduce the search time a little, but would incur the overhead of computing a preordering for every new RAd received. So, for task T_i:

$$\forall_{i \in 1...n_\bullet} C(T_{i+1}) = RTT + i * l * t \tag{2}$$

4 DESIGN ANALYSIS

The respawning algorithm used in a MMOG is important to estimate the reasonable scale of i in Equation 2. Currently, well-known commercial MMOGs (WoW, 2001; Woo, 2004) employ a timer-based approach that respawns NPCs to keep the number of NPCs and PCs to a stable ratio. Suppose that the number of players in a game zone is P, the NPC to PC ratio is R, and on average a NPC's life time is TTL seconds, $\frac{P*R}{TTL}$ NPCs should be respawned every second. Accordingly, if a timer is set to respawn NPCs every Int th second (i.e. the respawning interval), each time $\frac{P*R*Int}{TTL}$ NPCs are respawned. In fact, the respawning interval determines the deadline for a set of NPC tasks. If the last task can be processed before its deadline, previous tasks can meet their deadlines as well. In this case, Equation 1 is changed to:

Variable	Meaning	Nominal Value
P	zone population	
R	NPC : PC ratio	5 : 1
TTL	NPC life time expectation	300 (second)
r	event triggering rate	1/60 (per second)
Int	respawning interval	
RTT	round trip time	0.5 (second)
l	RM resource queue length	50 (RAds)
t	matchmaking time	1 (ms per RAd)

Figure 3: DDA variables & nominal values.

$$C(T_{last}) = RTT + \frac{P * R * Int * l * t}{TTL} < Int \tag{3}$$

In the Mediator framework a player may also trigger other events that require resource providers to be located. Assuming that each player triggers r such events in every second, there would be $r*P$ events in total. It is likely that the player-triggered events have shorter deadlines than regular respawning events, and correspondingly, tasks for the former have higher priorities in an RM's TQ. The worst case is that in every respawning interval, the RM needs to complete all player-triggered tasks before handling the last respawning task, so Equation 3 is changed to:

$$RTT + (\frac{P * R}{TTL} + r * P) * Int * l * t < Int \tag{4}$$

Figure 3 lists all the key parameters of the DDA model, together with nominal values typical of commercial MMOGs (WoW, 2001; Woo, 2004). Applying the nominal values to Equation 4 creates Equation 5, and let us draw the following inferences:

$$(600 - P) * Int > 300 \tag{5}$$

- Based on these assumptions, DDA can support up to 600 peers in each game zone, which corresponds to a zone size of around 500 peers in many typical MMOGs.
- When implementing a P2P MMOG with a zone size of 500 peers, the minimal respawning interval is 3 seconds.
- On average a peer may obtain a task in every 10 respawning intervals, so the credits rewarded for running a task should cover the payment charged for corresponding playing time.

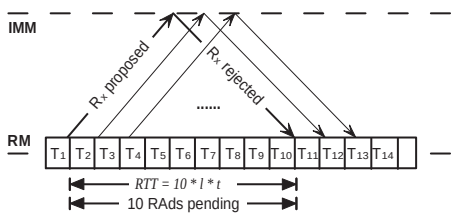

Figure 4: The resource tie-up problem.

5 DDA EXTENSIONS

5.1 Multilevel Feedback Delay Queue

Though theoretically DDA supports up to 600 peers in each game zone, the actual population might be smaller, especially when dynamic zoning is supported. Suppose that there are only 200 peers in a zone, and a respawning interval of 6 seconds is used. Hence, at the beginning of every interval, the IMM sends out 20 tasks to the RMs, specifying the deadline as 6 seconds. For T_1, RM Alice proposes RAd R_x to the IMM as a bid, then carries on processing T_2. Unfortunately, at the IMM end, R_y proposed by RM Bob is better than R_x, so R_x is rejected and returned to Alice. In spite of the time for IMM's decision making, it will take at least RTT time before Alice finds out that her bid has been rejected. Here, a potential problem is that in expression 2, $l * t$ is too small compared to RTT. Figure 4 demonstrates this "resource tie-up" problem, in which 10 RAds are tied up throughout every respawning interval. To address this problem, an RM may need to slow down its matchmaking, but still guarantee that all tasks meet their deadlines. This can be achieved with a multilevel feedback delay queue (MFDQ).

A MFDQ is similar to a multilevel feedback queue for Unix process scheduling, where the main difference is that in the MFDQ, each run queue is executed only when its delay has expired. If the inter run queue delay is d, then a MFDQ comprises $n = ceil(Int/d)$ run queues in total. Each run queue represents a different urgency level. Queue `Instant` buffers the most urgent tasks. Other run queues are arranged in a circle, with a `Head` pointer indicating the queue to be executed next. A MFDQ mainly supports two operations:

1) $offer(Task\ t)$ — the enqueue operation:

```
offer(Task t){
  t.enqueue_deadline = t.original_deadline-RTT-l*t;
  t.urgency_level = floor(t.enqueue_deadline/d);
  if(t.urgency_level == 0)
    Instant.add(t);
  else if(t.urgency_level >= (n-1))
    Tail.add(t);
  else
    getQueue((t.urgency_level+Head.index)%n).add(t);
}
```

The algorithm recalculates a task's enqueue deadline by subtracting RTT and $l*t$ from its original deadline. Then, it evaluates the task's urgency level. If the task is not able to bear one inter-queue delay, the task is added to queue `Instant`. Otherwise, the task is added to a run queue.

2) $poll(\ )$ — the dequeue operation:

```
poll(){
  int to_move = ceil(total_number_of_tasks/n);
  int done = 0;
  for(each Task t in Head){
    Instant.add(Head.remove());
    done++;
  }
  Head = getQueue((Head.index+1)%n);
  int current = Head.index;
  while(done < to_move){
    if(! getQueue(current).isEmpty()){
      Instant.add(getQueue(current).remove());
      done++;
    }
    else
      current = (current+1)%n;
  }
}
```

The algorithm firstly moves all tasks in queue `Head` to queue `Instant`. Because each run queue may have a different length, it moves the same number of tasks each time. If less tasks are moved from `Head` to `Instant`, more tasks in lower level queues are moved as well. If a task's deadline can be satisfied in a common EDF queue, the MFDQ can satisfy its deadline as well.

5.2 Matchmaking Policies

A matchmaking policy is the set of criteria that an RM uses for selecting locally optimised resource providers (by default, the term "optimised" refers to minimised communication latency). In the Mediator framework, an IMM anticipates forthcoming gaming events in a game zone, as discussed in Section 3.1. So, an IMM is able to provide in a JAd a list of target players which are able to interact with a newly respawned NPC. At matchmaking time, an RM takes out every available RAd from its resource queue, matching it to the JobAd using two conditions:

- Does the resource provider have adequate computing resource for running the task?
- Is the mean communication latency among the resource provider and the target players the shortest?

Because this policy strictly selects the resource provider that provides minimum latency, it is called a "strict incentive policy". However, a problem is that less competitive peers are likely to be starved of opportunities for earning credits, and finally to be regarded as free-riders, as discussed in Section 3.2. To stop this problem from happening, a factor τ can be introduced to relax the selection criteria as below:

- Does the resource provider have adequate computing resource for running the task?
- Is the mean communication latency within the range of $shortest * \tau$?
- Is the resource provider low on credit?

This policy favours the poorest peer providing a latency that is not greater than $shortest * \tau$, and making it a "friendly incentive policy". Experimental results demonstrate that with $\tau = 1.2$, DDA shares tasks among the peers more fairly than using the strict incentive policy of $\tau = 1$ (Section 7.3). Actually, DDA allows flexible matchmaking policies to be adopted. For example, when a rep-

utation mechanism is brought into effect in future work, it will allow policies that favour more dependable resource providers.

6 IMPLEMENTATION

Currently, a proof-of-concept prototype for DDA has been implemented using FreePastry 2.1, an open-source implementation of Pastry (Rowstron and Druschel, 2001). The RM's matchmaking mechanism has been implemented using ClassAds 2.2 (Litzkow et al., 1988). Furthermore, to evaluate the prototype, a test-bed application has been developed, which simulates a P2P MMOG using a 2-dimensional game world and hundreds of virtual player avatars moving according to a random way point algorithm.

The test-bed employs the Direct discrete event simulator integrated in FreePastry. The simulator takes in a network topology model generated by GT-ITM, and simulates Internet scale communication latencies among peers in a P2P network. Each peer is assigned some virtual computing resource, e.g. CPU cycles and network bandwidth, and local scheduling activities are simulated by virtual resource managers.

7 EXPERIMENTAL RESULTS

7.1 Implementation Adequacy

On a workstation with 2.4GHz Intel Core2 Duo CPU and 2GB memory the test-bed simulates game sessions with 800 players. The maximum population supported in the experiment is mainly limited by the scalability of the Direct simulator. Experimental results show that during the bootstrapping time, various super-peer roles are selected successfully. Furthermore, under an average churn rate of 5% per minute, game zone structures are maintained correctly, and the leaving of both super and common peers are handled properly.

7.2 Real-time Resource Allocation

Figure 5 depicts the distribution of communication latencies among resource providers located by DDA, and corresponding resource consumers. Firstly, the experimental results demonstrate that for all the tasks that have been measured, none of them missed their deadlines. DDA is capable of processing large number of tasks, whose deadlines vary between 3 and 8 seconds. So, DDA seems able to satisfy the real-time requirement on resource discovery and allocation needed by a P2P MMOG.

Secondly, the "Network Latency" curve demonstrates the actual communication latency distribution for the network topology used by the Direct simulator. By default, the distribution is approximately a Gaussian distribution $N(350, 100)$. If resource providers are randomly selected,

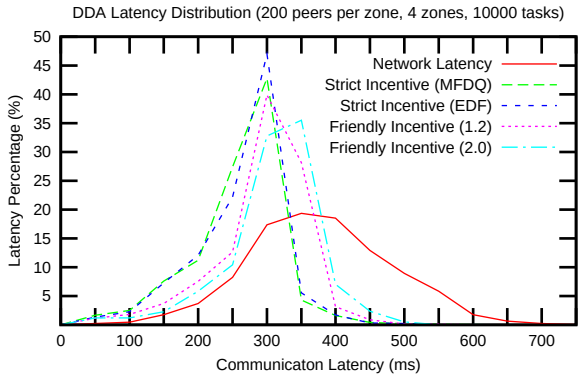

Figure 5: DDA latency distribution.

the latency distribution for DDA should be similar to the default curve. However, as demonstrated by Figure 5, when RMs schedule tasks using an EDF queue, around 90% of the latencies are below mean latency. Furthermore, a MFDQ increases the probability for DDA to discover optimal resource providers. As a result, around 95% of the latencies are below mean latency. So, by carefully selecting resource providers, DDA is able to satisfy the QoS requirement on interactive NPC tasks needed by a P2P MMOG.

7.3 Cooperative Economic Model

Figure 6 depicts the distribution of credits earned by the 800 players after 10000 tasks. With the strict incentive policy, the gap between the richest and poorest players is wide. Specifically, around 5% of the players have earned more than 500 credits, but around 20% other players are in debt to the system for -300 credits. In contrast, with a friendly incentive policy ($\tau = 1.2$), the gap between the rich and poor can be significantly narrowed (some peers are still in debt, due to their inability to provide usable resources).

The impact of different incentive policies on resource quality is displayed in Figure 5. Using the strict incentive policy, only 5% of resource providers are selected when their latencies are higher than the mean. However, this ratio increases to about 30% using the friendly incentive policy ($\tau = 1.2$). Similarly, when τ is more relaxed, e.g. $\tau = 2.0$, the gap can be further narrowed in Figure 6, but the resource quality becomes worse. A P2P MMOG can use τ to customize its own cooperative economic model to strike a balance between fairness in task sharing and QoS for maintaining acceptable gaming experience.

8 CONCLUSION & FUTURE WORK

This paper presents Deadline Driven Auctions, a novel task mapping infrastructure for heterogeneous environments. The strength of DDA is to support self-organised real-time NPC host allocation in P2P MMOGs, as well as to meet the QoS requirements for game interactivity.

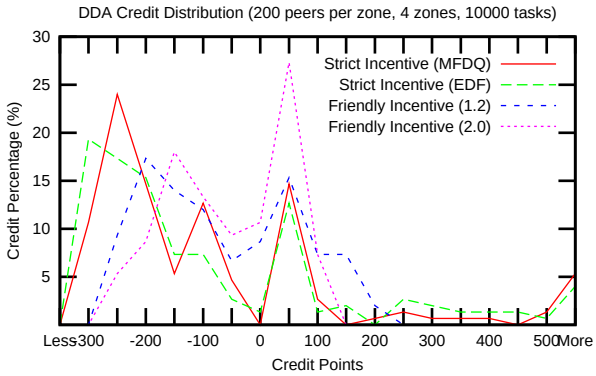

Figure 6: DDA credit point distribution.

A prototype for DDA and a test-bed application have been implemented (Section 6). Experimental results show that, compared to previous work (Section 2), DDA has four main advantages. Firstly, DDA is self-organising. Its infrastructure can be automatically assembled and maintained as peers join and leave the system (Section 7.1). Secondly, DDA is able to process large numbers of real-time NPC tasks and to support a simulated P2P MMOG with 800 players effectively (Section 7.2). Thirdly, DDA satisfies the QoS requirement for game interactivity by keeping the communication latency among NPC hosts and ordinary players low, e.g. 95% of the latencies are below mean latency (Section 7.2). Fourthly, by applying a friendly incentive policy, DDA can establish a cooperative economic model that shares tasks among application participants fairly (Section 7.3).

In future work DDA's security will be enhanced by a distributed reputation system, and a new resource matchmaking policy will take into consideration a resource provider's trustworthiness and dependability. Furthemore, though DDA is primarily designed to support NPC host allocation, it may also apply to general P2P applications with real-time computational or interactive tasks. In future work DDA's potential usage for other application types will be explored.

ACKNOWLEDGMENT

The authors gratefully acknowledge the contributions of Jeff Hoye for tirelessly providing vital software maintenance for the FreePastry, the Direct simulator, and for many helpful suggestions that expedited this research.

REFERENCES

(2001). World of Warcraft. worldofwarcraft.com, Blizzard Entertainment.

(2004). World of Legend. woool.sdo.com, Shanda Entertainment.

Anderson, T. W. (1984). *An Introduction to Multivariate Analysis 2nd Edition*. John Wiley & Sons.

Bharambe, A. (2006). Colyseus: A Distributed Architecture for Online Multiplayer Games. In *Proceedings of NSDI*, pages 3–6. USENIX.

Fan, L., Taylor, H., and Trinder, P. (2007). Mediator: A Design Framework for P2P MMOGs. In *Proceedings of NetGames*, pages 43–48. ACM.

Gupta, M., Judge, P., and Ammar, M. (2003). A Reputation System for Peer-to-Peer Networks. In *Proceedings of NOSSDAV*, pages 144–152. ACM.

Hu, S.-Y., Chang, S.-C., and Jiang, J.-R. (2008). Voronoi State Management for Peer-to-Peer Massively Multiplayer Online Games. In *Proceedings of CCNC*, pages 1134–1138. IEEE.

Iimura, T., Hazeyama, H., and Kadobayashi, Y. (2004). Zoned Federation of Game Servers - a Peer-to-peer Approach. In *Proceedings of NetGames*, pages 116–120. ACM.

Litzkow, M., Livny, M., and Mutka, M. (1988). Condor - A Hunter of Idle Workstations. In *Proceedings of ICDCS*, pages 104–111.

Lu, H., Knutsson, B., Xu, W., and Hopkins, B. (2004). Peer-to-Peer Support for Massively Multiplayer Games. In *Proceeding of INFOCOM*, pages 7–11. IEEE.

Mulligan, J. and Patrovsky, B. (2003). *Developing Online Games - An Insiders Guide*. New Riders Publishing, ISBN: 1592730000.

Raman, R., Livny, M., and Solomon, M. (2000). Resource Management through Multilateral Matchmaking. In *Proceedings of HPDC*, pages 290–291. IEEE.

Rowstron, A. and Druschel, P. (2001). Pastry: Scalable, Decentralized Object Location and Routing for Large Scale Peer-to-Peer Systems. In *Proceedings of Middleware*, pages 329–350. ACM.

Yonekura, T., Kawano, Y., and Hanawa, D. (2004). Peer-to-peer networked field-type virtual environment by using atoz. In *Proceedings of CW*, pages 241–248. IEEE.

Yu, A. P. and T.Vuong, S. (2005). MOPAR: a Mobile Peer-to-Peer Overlay Architecture for Interest Management of Massively Multiplayer Online Games. In *Proceedings of NOSSDAV*, pages 99–104. ACM.