Some Considerations on the Design of a P2P Infrastructure for Massive Simulations

Gennaro Cordasco*, Rosario De Chiara*, Ugo Erra† and Vittorio Scarano*

*ISISLab - Dipartimento di Informatica ed Applicazioni "R.M. Capocelli",
Università degli Studi di Salerno, Fisciano, Italy.
Email: {cordasco, dechiara, vitsca}@dia.unisa.it

†Dipartimento di Matematica e Informatica,
Università della Basilicata, Potenza, Italy.
Email: ugo.erra@unibas.it

***Abstract*—Massive Multiuser Virtual Environments (MMVEs) are rapidly expanding both in the number of users and complexity of interactions. Their needs of computational resources offer new challenges for the computer scientists. In this paper we present some ideas on the implementation of a Massive Simulation Environments, a particular MMVE, distributed over a Peer-to-Peer infrastructure. We analyze some of the problems related to the workload balancing on such distributed environments. In particular we discuss an hybrid Peer-to-Peer architecture in order to provide an efficient load balancing strategy. By some assumptions on temporal and spatial coherence, we use a predictor component which exploits previous phase workload as an estimate for next phase workload for load balancing purposes.**

***Keywords*-Peer–to–Peer; Massive Simulation; Load Balancing.**

I. Introduction

Distributed Virtual Environment (DVE) is an emerging research field which combines 3D graphics, networking and behavioral animation with the purpose of simulating realistic and immersive virtual environments offering an high degree of interactivity. The distributed nature of these systems widened the scenarios of use that now ranges from online videogames to serious games for training including online cooperative systems for learning and problem solving.

DVEs have greatly evolved in past times, so that a new term has been coined: Massively Multiuser Virtual Environment (MMVE) which defines these environments where hundreds of thousands of actors interact simultaneously. Recently, *World of Warcraft* and *Second Life*, have reached around 10 million subscribers worldwide and roughly 1 million of active users [1].

The design and management of MMVE, due to their highly interactive nature, poses many unique challenges compared to traditional network domains [2]. A single machine is not able to manage thousands of players at the same time. So, even if the client/server approach is quite common for small-size DVEs, it is mandatory for MMVEs to use the computing power of a group of many servers with dedicated responsibility. Several approaches split the responsibilities of the servers in different ways. For instance, every server can have a different assignment (communication, artificial intelligence, physics, game state) or, on the other hand, a single server acts as a *factotum* server for a portion of the whole environment (aka *shard*). In this case the actors which belong to different shards can not interact with each other (each shard represent a distinct copy of the whole environment). Another approach to achieve scalability is by developing MMVE on top of a Peer-to-Peer (P2P) infrastructure where the responsibility of maintaining the whole environment is shared among all its users. In this approach the workload balancing is essential for both the overall performance and scalability. In the context of MMVEs we are interested to Distributed Massive Simulation Environments (DMSEs), which, for the same reasons, they appear as a suitable problem that can greatly benefit from the use of a P2P approach.

A. Massive Simulation Environments (MSEs)

The simulation of groups of characters moving in a virtual world is a topic that has been investigated since the 1980s with the purpose of simulating a group of entities, dubbed *autonomous actors*, whose movements are related to social interactions among group members.

Flock simulation: A classical example of use of this approach is the simulation of a flock of birds in the most natural possible way. Elements of this simulated flock are usually named *boids* (from *bird-oid*) and got instilled a range of *behaviors* that induces some kind of *personality*. A widespread approach to this kind of simulations has been introduced in [3]. Every boid has its own *personality* (e.g. the trajectory of its flight) that is the result of a weighted sum of a number of *behaviours*. The simulation is performed in successive steps: at each step, for each boid and for each behavior in the personality, the system calculates a request to accelerate in a certain direction in the space, and sums up all of these requests; then the boid is moved along this result. The behaviors are, in the most of cases, simply geometric

9781–4244–3941–6/09/$25.00 © 2009 IEEE

calculations that are carried out for each boid considering the *k*-neighbors it is flying with: for example the behavior called *pursuit* just let the boid to pursuit a moving target (e.g. another boid). Each boid reacts to its *k*-neighbors, which constitute its neighborhood. Given a certain boid out of a flock of n boids, the most simple way of identifying that boid's neighborhood is by an $O(n^2)$ proximity screening, and for this reason the efficiency of the implementation is yet to be considered an issue.

Massive Battle: Massive Battle [4] is a MSE capable of animating autonomous actors with the purpose of reconstructing interactive scenes from a battlefield showing a number of platoons fighting each others. Massive Battle is implemented by expanding the boid model in order to reach an higher degree of realism of the behaviors exhibited by the actors. Massive Battle uses the boid model as the foundation on which to build more complex behaviors: the initial idea of simulating a flock of boids is expanded to simulate a platoon of soldiers obeying to commands imparted by a leader. Soldiers are able to march along a path and are also capable of engaging a fight with enemy platoons (see Fig. 1).

Massive Battle is an example of *serious game* system that offers an effective way of simulating historical battles for the purpose of learning (e.g. providing new insights for battles to engage students) and to carry out historical researches (e.g. what-if scenarios). The simulation of historical battles also imposes some constrains on the number of actors the system is capable of simulate: as an example the Waterloo Battle involved ≈ 190000 soldiers, while Massive Battles, running on an off-the-shelves PC, is capable of simulating only ≈ 5000 units.

Differences between MMVEs and DMSEs: For the sake of the discussion it is worth to clearly state the differences running between MMVEs and DMSEs distinguishing between the typology of users and the kind of interaction they get. In an MMVE actors present in the environment may or not be simulated actors: in the first case they are usually dubbed NPC (Non-Playing Characters), in the latter case they are human players. Considering the presence of human players involved in playing a game, cheating is a serious issue that must be taken into account. In a DMSE every actor present in the environment is simulated. Users connect to the system in order to check how the simulation is evolving. In MMVE situations of uneven balancing cannot be foreseen because they strongly depends on how human players move and act in the environment. In a DMSE the initial conditions of the simulation are pre-determined, this is an important factor that can be taken into account: for example it is reasonable that none of the simulated soldiers will deliberately wander around the map.

B. Designing a P2PMSE

We intend to expand the number of simulated actors by Massive Battle by distributing the computational load to various PCs connected to the system. Each user connected to the system will have two different subsystems running on his PC: a simulation engine and a visualization engine. The simulation engine will take the responsibility of simulating a small part of the soldiers in the system, while the visualization engine will let the user to choose which part of the battlefield to visualize. The system architecture is based on a *Distributed Hash Table* (DHT) that will let the user to dynamically connect to the system in a totally distributed manner: once a new peer is available in the system it will receive part of the simulation and will receive the updates from the system.

The visualization engine will let the user to freely place a camera in the environment and this camera will define an Area of Interest (AOI). AOI is a fundamental concept, as even though many actors and events may exist in the simulated environment, the user, as in the real world, is only interested by nearby actors or events. AOI thus specifies a scope for information which the system should provide to the user. Notice that each subsystem, simulation and visualization, will have its own AOI: the AOI for the simulation, henceforth neighborhood, is defined by the position of actors the peer is simulating, the AOI of the visualization is defined by where the user placed the camera.

II. Peer–to–Peer Background

Peer–to–Peer (P2P) is a class of distributed applications where each node connected to the system (peer) has the same responsibility. In a P2P network, all the peers take advantage of resources, storage, cycles, available at the edges of the Internet, allowing peers to leverage their collective power to the "benefit" of all. Scalability represents the main characteristic that marks the diversity of P2P systems from standard *Client-Server* architectures. Indeed, in the latter case, adding more clients represents solely additional workload for the server(s) and, consequently, less performances for all the clients. On the other hand, P2P systems distribute the whole workload, across all the peers in the network, thus enabling applications to scale without the need for powerful, expensive servers and, consequently, overshadow the capabilities of centralized systems with low costs. Latency on the network is also reduced, thanks to the absence of bottlenecks.

Several scalable P2P systems, based on a DHT approach, have been proposed [5], [6], [7], [8]. A DHT is a self-organizing *overlay network* that allows to add, delete and look up hash table elements. These systems provide: (i) efficient resources' lookup; (ii) load balancing of both resources and query messages; (iii) dynamic maintenance in presence of continuous peers join, leave or fail.

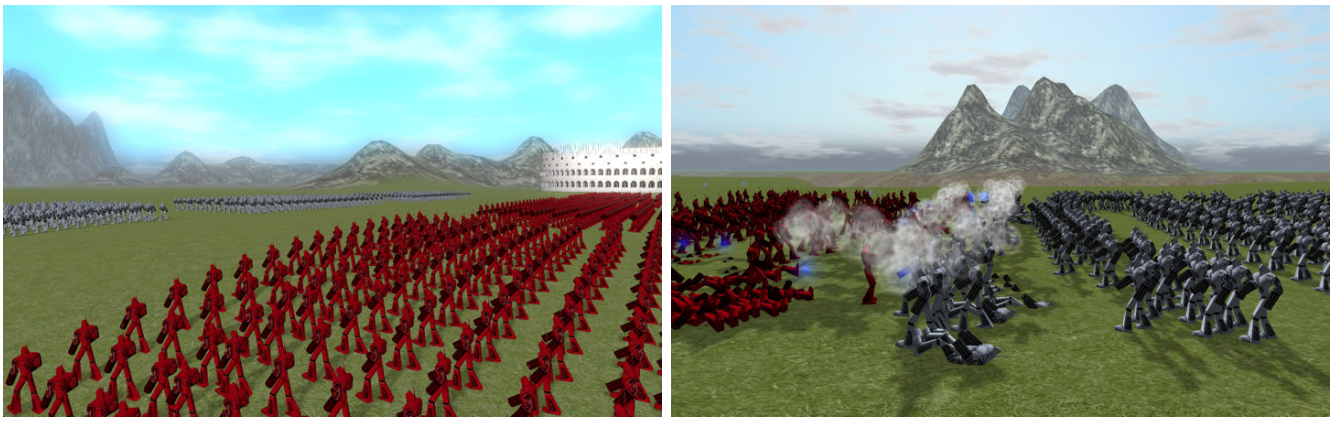

Figure 1. Two screenshot of the Massive Battle MSE: on the left two platoons in formation, on the right platoons confronting by using firearms.

However the design of a P2P DMSEs is quite different from classical P2P applications which are mainly devoted on sharing files. DHT based system in their simple form may result in actors, which belongs to the same zone, to be placed far-apart in the DHT space. Therefore neighbor actors, that probably will interact with one other, will be conncted via large distance and multiple hops resulting in high routing latency and low efficiency. Several approaches [9], [10] are based on the idea of location aware ID assignment, where the actors ID are assigned in order to ensure that actor close in the map are also close in the DHT metric space. Unfortunately such approaches lead to an unbalanced distribution of nodes into the key space which harms the efficiency of the DHT routing algorithm.

The P2P systems offer great benefits in terms of scalability when compared to classical client/server architectures but questions about how divide the system workload and how distribute the workload on the peers are still open topics of discussion. Several research papers proposed dynamic load balancing algorithms to relieve the load imbalance in P2P systems. Most of the papers are based on the concept of virtual servers proposed in [11]. In this work storage and routing occur at virtual servers rather than peers and each peer can host one or more virtual servers. In this scenario each virtual server maintains a sub-region. When a peer is overloaded it transfers some of its virtual servers to an undercharged peer. This decision is based on some peers that act as load balancer.

In [12] the authors propose a load balancing scheme for moving objects that are continuously updated by using a cost model that optimizes the location updates and query processing. The proposed cost model allows the assignment of virtual servers from one peer to another peer if the desired transition optimizes the cost function.

Other papers based on virtual servers are [13] and [14]. In all these approaches, the system dynamically monitors the workload distribution and moves virtual servers from overloaded peers to undercharged peers. In [15] a similar approach is described. The whole map is partitioned into a static set of microcell. Each user is responsible of simulating a given set of microcells. When a user get overloaded, some of its microcells are assigned to another user is such a way to balance the workload while minimizing the communication between users. However, the usage of virtual servers greatly increases the amount of routing data information needed, the management and the communication costs, considering both the latency (more messages are required; therefore, messages may be queued up) and bandwidth (communication overhead for each message), on each peer. Other relevant work can be found in [16], [17], [18].

We propose a strategy that makes a trade-off between balancing the workload and minimizing the extra communication overhead. By using a predictor component our strategy provide a balanced load without resorting to excessively increase the number of sub-regions.

III. Main issues in designing DMSEs

In order to devise a fully distributed infrastructure for DMSEs, three main issues need to be addressed: World partitioning, World state propagation and Load Balancing.

A. World Partitioning

A simple and efficient solution to achieve scalability using a P2P system is to partition the whole environment map into regions. Regions are assigned to peers, named *region masters*, by mapping both regions and peers to the DHT key space: regions as well as peers are associated with an ID computed by using a consistent hash function [19]. The peer whose ID is the closest to the region ID become the region master for the region. Region masters are responsible to:

- Simulate all the actors which belong to the region;
- Deliver the state of the region (that is the state of each actor which belongs to the region) to the peers whose AOI overlaps with the region;
- Handle *handovers* of actors between regions.

The choice of the world partitioning technique is important for the efficiency of the whole system. Two key factors need to be considered:

- Static or Dynamic Partitioning;
- The *granularity* of the world decomposition.

A standard approach is to divide the environment map into static equal sized regions. This approach is quite easy to develop, each region has a unique ID and, therefore, region masters can be easily discovered and kept up to date by peers. Unfortunately, this approach does not guarantee load balancing: a region masters can be overcharged by crowding in its region. To address the issue of overcrowded regions, several techniques use dynamic partitioning schemes, where regions might have different sizes but comparable computing requirements. The problem with this approach is that the management of dynamic regions requires a huge amount of communication between region masters that consumes bandwidth and introduces latency [20], [21]. For instance actors could continuously be migrated from one region to another even if they did not change their positions.

The region size and, consequently, the number of regions, which a given map is partitioned into, determines the *granularity* of the world decomposition. It may appear that the time required to simulate the actors could be easily reduced by simply increasing the granularity of the decomposition, in order to perform more and more simulations in parallel, but this is not always true. Typically, interactions between actors (the behavior of actors is influenced by their neighborhood), and/or other important factors, such as the number of messages required to propagate region state, limit the choice to coarse-grained granularity. Indeed, the finer is the adopted granularity, the more is the generated communication between regions. The interaction between regions is a direct consequence of the fact that exchanging information between regions (e.g. actors positions, events, or transition across regions) is usually needed. Other considerations that would suggest to use coarse-grained granularity are (a) the *locality of interaction* and (b) the *spatial coherence* that are motivated because (i) the behavior of an actor usually relies on nearby actors and (ii) two close actors usually need to access some common data. Then, it is important that the regions are large enough, so that each user can exploit spatial coherence of regions, having a good degree of (local) cache hits.

B. World State Propagation

In order to implement a P2P architecture for MMVEs, a communication infrastructure is needed to deliver messages to a wide group of users. Being multicast communication not available on geographical network, application-layer multicast provides a workaround [22]. For instance, SimMud [23] uses Scribe [24], an application-layer multicast built on top of the DHT Pastry [8]. Scribe is decentralized and highly efficient because it leverages the existing Pastry overlay.

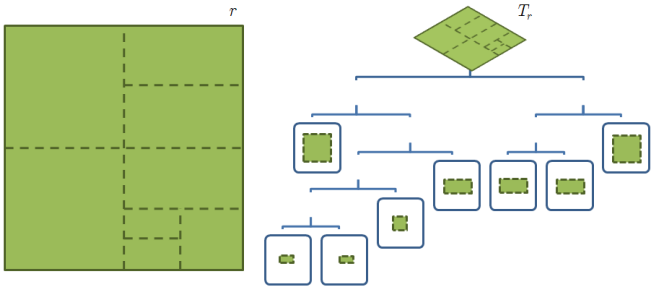

Figure 2. An example of a PBT decomposition.

A well-known mechanism used to propagate world state information is based on the Publish/Subscribe design pattern: a multicast channel is assigned to each region; users then simply subscribe to the channels associated with the regions which overlap with their AOI to receive relevant message updates. As users' cameras moves, they will need to subscribe to new channels and unsubscribe from old ones. This approach raises again the problem of determining the granularity of the world decomposition [25]: using a coarse-grained granularity, regions are often bigger than users' AOI, then region subscribers will receive unnecessary information. On the other hand, using finer granularity, provides a greater number of channels, then a complex subscription management is needed, as more regions overlaps with a given AOI.

C. Load balancing using the Prediction Binary Tree

We developed in [26] a decomposition strategy for mesh-like computations that exploits both spatial and temporal coherence, among computation phases, to perform load balanced decomposition. The strategy uses *temporal coherence* (the amount of time required to elaborate a region r in phase f is comparable to the amount of time required to elaborate r in phase $f+1$) to estimate the computing time of a new computation phase using previous phase computing time. The strategy performs a semi-static load balancing (decisions are made before each computing phase). Temporal coherence is exploited using a *Prediction Binary Tree* (PBT) where each leaf represents a region which will be assigned to a peer as a task.

The proposed approach particularly suites to the World partitioning problem. Indeed, the actors of a MSE usually have limited movement and speed and interact mainly with nearby actors. Thus, MSEs exhibit both *temporal* and *spatial* localities.

The proposed strategy makes a trade-off between balancing the workload and minimizing the extra communication overhead due to regions's interactions, by using a Predictor component (the PBT), with a negligible overhead, keeps the load balanced without resorting to excessively increase the number of regions.

A PBT T_r describes a decomposition of a given area r through a full binary tree. The root of T_r represents

the whole area r. The two children of an internal node v represent the two halves of the area represented by v. Consequently, the set of the leaves of T_r represents a partition of the area r (cf. Fig. 2). Hence, the PBT stores the decomposition of r and each leaf of T_r represents a region to be simulated by a user. At the end of each phase, the PBT receives the information about the time needed to simulate each region. By using the previous phase times as estimates, the PBT is efficiently updated for the next phase. The variance between regions' time-computation is defined as a metric to measure the (estimated) computational unbalance that is expected given the decomposition provided by the PBT T_r. Then a simple algorithm update the PBT (via some split/merge operation) in such a way that the variance of the estimated times is improved and consequently the PBT describes a balanced decomposition, without excessively increasing the number of regions. More details can be found in [26].

While the PBT was devised for a very different scenario (it obeys the Master-workers paradigm which is based on a centralized approach and was designed for a cluster of workstations where the number of workers is fixed and known *a priori*) it is an effective tool to tackle balancing problems and minimizing overhead, as we show in the next Section.

IV. Our Proposal

We propose here an hybrid approach which allows to exploit all the features of the PBT in a decentralized environment. We recall that every user of our system is at the same time a worker (peer) and has the opportunity to visualize a portion of the environment (camera).

Our architecture is based upon a DHT overlay network where the simulation engine of each peer works in a distributed manner and uses the DHT as a data storage while the visualization engine is managed by each user on its own.

We notice that a global load balancing approach is quite hard to develop (at each simulation phase the information about the load of each peer should be distributed in some way). For this reason we chose a local workload balancing. Obviously, the use of a local load balancing scheme cannot guarantee an optimal load balancing. However, a carefully choice of the granularity of regions and the use of a DHT approach to assign regions to peers in a "pseudo-random" way allows to obtain a good workload balance without introducing a huge amount of communication.

The whole environment map is partitioned into a set of zones using a 2-levels hierarchical approach (cf. Fig 3): first of all, the map is partitioned into a predefined set of static regions. Each region is marked by an ID and consequently assigned to a peer (region master). Users interested on visualizing a given region will subscribe to that region, that is they will inform the region master of their interest. For instance, as in [23], one can use a Pastry network for maintaining the state of the environment and the mapping between regions and peers, and on top of that, Scribe can be used for the World State propagation. Region masters of neighboring regions can always be connected in order to speed up the handover.

As observed in Section III-A such a static approach cannot provide load balancing. On DMVEs in particular, region master can be overcharged either because there are too many actors to simulate or because there are too many subscribers to inform about the state of the region. In order to provide load balancing without increasing the amount of inter-region communication (or generating undesired actors migrations) we exploit the idea of the PBT on top of each region. We use a forest of PBTs where each static region corresponds to the root of a PBT. During each computing step a region is dynamically partitioned into a set of sub-regions which corresponds to the leaves of the PBT managed by the region master. It is worth to recall that the use of PBT ensures load balancing and minimizes inter-region communication. A new ID is assigned to each sub-region which is mapped into the DHT space. Consequently the actors belonging to each sub-region are simulated by a peer (sub-region worker). For each sub-region a new multicast channel, managed by the sub-region worker, is generated. The region master is still in charge of managing the PBT and act as a proxy between region subscribers and sub-region worker, while the task of simulating the region is delegated to the sub-region workers.

A. PBT managing

The activities of a region master is to maintain the structure of the PBT by monitoring the performances of the workers which simulate actors in its region. The rationale behind this idea is to keep the variance in the simulation time spent by sub-region workers, as low as possible. This is achieved by splitting overloaded sub-regions and merging undercharged sub-regions, following the PBT load balancing schema. A well-known risk is that under uneven load distribution or when the peers are low on computing resources, this mechanism could degenerate by producing a huge number of splits. In order to mitigate this effect, limitation mechanisms on the size of PBT (either height or number of leaves) can be deployed.

B. Communication

The subscription of a peer to a sub-region is made in two phases: region identification and sub-region subscription. When a user is interested on visualizing a certain region (e.g. its AOI overlaps with that region), then the peer subscribes to that region master. Being each region static, the regions' ID do not change and therefore each user can easily find the region masters. Each region master by using a multicast channel, continuously publish the information about the current state of its PBT. In particular each region master sends, to each subscriber, the information about the current

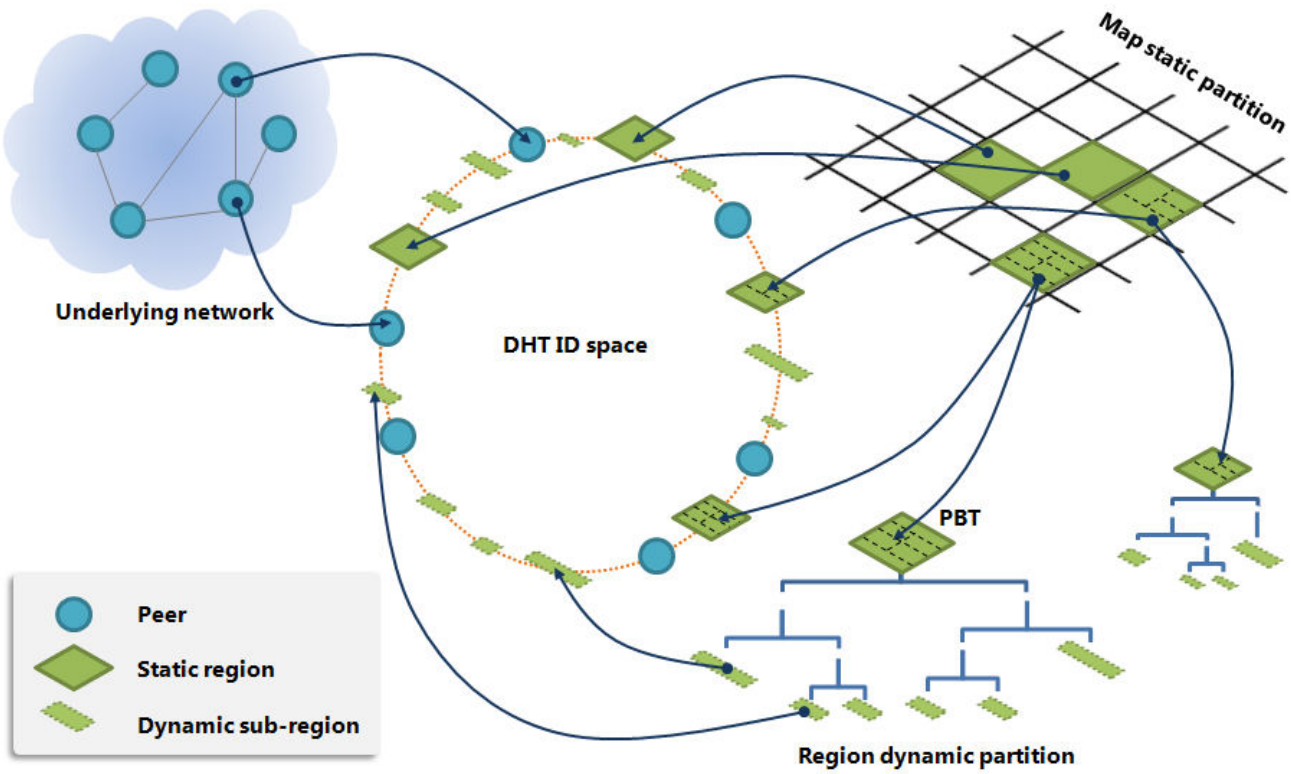

Figure 3. The system architecture split in its logic modules

partition of its region and the IDs associated to each sub-region. In this way each peer is able to choose which sub-regions to subscribe to. Sub-region workers simply publish the current status of all the actors which belongs to it.

C. Timing

We use a simple approach to achieve a consistent synchronization of the distributed simulations. A fixed time slot is established for each simulation. For instance, for a real time simulation the time slot could be ≈ 40 milliseconds, for ≈ 25 FPS. In each time slot each worker computes a new step of the simulation (i.e. a new state) based on the previous state and on the events received by other workers. The number of steps since the beginning of the simulation is used as a clock so that each event can be associated with a system-wide timestamp. Timestamps can also be useful for synchronizations: when a node is late on the simulation then the simulation is sped up by exploiting other peers' computing resources.

V. Conclusion

DMSEs, due to their scalability requirements, appear to be a natural application for P2P architectures. However DMSEs are quite different from classical P2P applications which are mainly devoted on sharing files as well as storages. On DMSEs the shared resources consist of CPU cycles while the purpose of the architecture is to maintain a distributed data storage (that represents the state of the simulated environment) keeping the latency as small as possible.

We presented some consideration on the use of a P2P infrastructure for DMVEs. The proposed architecture exploits the features of both DHT schemes and a novel data structure called PBT in order to provide a balanced workload on each peer without introducing extra inter-region communication. By some assumptions on temporal and spatial coherence, we use a predictor component which exploits previous phase workload as an estimate for next phase workload and, accordingly, each region is dynamically partitioned into a set of sub-regions in an efficient way. This stands even if a coarse-grained granularity of the static region partition is used, that is without introducing too much communication overhead.

Future works include the implementation of the proposed ideas and a validation through a series of tests, based on a distributed version of Massive battle.

References

[1] D. Pittman and C. GauthierDickey, "A Measurement Study of Virtual Populations in Massively Multiplayer Online Games," in *Proc. of the 6th ACM SIGCOMM workshop on Network and system support for games (NetGames '07)*. New York, NY, USA: ACM, 2007, pp. 25–30.

[2] J. Waldo, "Scaling in games and virtual worlds," *Commun. ACM*, vol. 51, no. 8, pp. 38–44, 2008.

[3] C. W. Reynolds, "Flocks, Herds, and Schools: A Distributed Behavioral Model," *Computer Graphics*, vol. 21, no. 4, pp. 25–34, 1987.

[4] A. Boccardo, R. De Chiara, and V. Scarano, "Massive Battle: Coordinated Movement of Autonomous Agents," in *Proc. of the Workshop on 3D Advanced Media In Gaming And Simulation (3AMIGAS)*, 2009.

[5] S. P. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content-addressable network," in *Proc. of ACM Special Interest Group on Data Communication (ACM SIGCOMM '01), San Diego, CA, US*, Aug. 2001, pp. 161–172.

[6] I. Stoica, R. Morris, D. Liben-Nowell, D. Karger, M. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications," in *IEEE/ACM Transactions on Networking (TON), Volume 11, No. 1*, Feb. 2003, pp. 17–32.

[7] B. Zhao, J. Kubiatowicz, and A. Joseph, "Tapestry: An infrastructure for fault-tolerant wide-area location and routing," in *Tech. Report No. UCB/CSD-01-1141, Computer Science Division (EECS), University of California at Berkeley*, Apr. 2001.

[8] P. Druschel and A. Rowstron, "Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems," in *Proc. of the 18th IFIP/ACM Inter. Conference on Distributed Systems Platforms (Middleware '01)*, Nov. 2001, pp. 329–350.

[9] S. Ratti, B. Hariri, and S. Shirmohammadi, "NL-DHT: A Non-uniform Locality Sensitive DHT Architecture for Massively Multi-user Virtual Environment Applications," in *Proc. of International Conference on Parallel and Distributed Systems*, 2008, pp. 793–798.

[10] B. Hariri, S. Shirmohammadi, and M. Pakravan, "LOADER: A Location-Aware Distributed Virtual Environment Architecture," in *Proc. of IEEE Conference on Virtual Environments, Human-Computer Interfaces and Measurement Systems (VECIMS '08)*, 2008, pp. 97–100.

[11] B. Godfrey, K. Lakshminarayanan, S. Surana, R. Karp, and I. Stoica, "Load balancing in dynamic structured P2P systems," in *Proc. of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2003)*, vol. 4, March 2004, pp. 2253–2262.

[12] M. E. Ali, E. Tanin, R. Zhang, and L. Kulik, "Load Balancing for Moving Object Management in a P2P Network," in *Proc. of 13th International Conference on Database Systems for Advanced Applications (DASFAA '08)*, 2008, pp. 251–266.

[13] J. Ledlie and M. Seltzer, "Distributed, secure load balancing with skew, heterogeneity and churn," in *Proc. of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '05)*, March 2005, pp. 1419–1430.

[14] S. Surana, B. Godfrey, K. Lakshminarayanan, R. Karp, and I. Stoica, "Load balancing in dynamic structured peer-to-peer systems," *Perform. Eval.*, vol. 63, no. 3, pp. 217–240, 2006.

[15] D. Ahmed and S. Shirmohammadi, "A microcell Oriented Load Balancing Model for Collaborative Virtual Environments," in *Proc. of IEEE Conference on Virtual Environments, Human-Computer Interfaces and Measurement Systems (VECIMS '08)*, 2008, pp. 86–91.

[16] D. T. Ahmed and S. Shirmohammadi, "A dynamic area of interest management and collaboration model for p2p mmogs," in *Proc. of the 12th IEEE/ACM International Symposium on Distributed Simulation and Real-Time Applications (DS-RT '08)*, 2008, pp. 27–34.

[17] T. Hampel, T. Bopp, and R. Hinn, "A Peer-to-Peer Architecture for Massive Multiplayer Online Games," in *Proc. of 5th ACM SIGCOMM workshop on Network and system support for games (NetGames '06)*, 2006, p. 48.

[18] A. Rhalibi and M. Merabti, "Interest Management and Scalability Issues in P2P MMOG," in *Proc. of 3rd IEEE Consumer Communications and Networking Conference (CCNC '06)*, Jan. 2006, pp. 1188–1192.

[19] D. Karger, E. Lehman, F. Leighton, R. Panigrahy, M. Levine, and D. Lewin, "Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the world wide web," in *Proc. of the 29th Annual ACM Symposium on Theory of Computing (STOC '97)*, 1997, pp. 654–663.

[20] E. Buyukkaya, M. Abdallah, and R. Cavagna, "VoroGame: A Hybrid P2P Architecture for Massively Multiplayer Games," in *Proc. of the 6th IEEE Consumer Communications and Networking Conference (CCNC '09)*, Jan. 2009, pp. 1–5.

[21] H.-Y. Kang, B.-J. Lim, and K.-J. Li, "P2P Spatial Query Processing by Delaunay Triangulation," in *Proc. of the 4th InternationalWorkshop on Web and Wireless Geographical Information Systems (W2GIS 2004)*, 2004, pp. 136–150.

[22] M. Castro, M. B. Jones, A.-M. Kermarrec, A. Rowstron, M. Theimer, H. Wang, and A. Wolman, "An Evaluation of Scalable Application-Level Multicast Built Using Peer-to-Peer Overlays," in *Proc. of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '03)*, 2003.

[23] B. Knutsson, H. Lu, W. Xu, and B. Hopkins, "Peer-to-Peer Support for Massively Multiplayer Games," in *Proc. of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '04)*, 2004, p. 107.

[24] M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron, "SCRIBE: A large-scale and decentralized application-level multicast infrastructure," *IEEE Journal on Selected Areas in Communications (JSAC)*, vol. 20, pp. 100–110, 2002.

[25] S.-Y. Hu, "Spatial Publish Subscribe," in *Proc. of IEEE Virtual Reality (IEEE VR) workshop, Massively Multiuser Virtual Environment (MMVE'09)*, 2009.

[26] G. Cordasco, B. Cosenza, R. De Chiara, U. Erra, and V. Scarano, "Experiences with Mesh-like computations using Prediction Binary Trees," *Scalable Computing: Practice and Experience, Scientific international journal for parallel and distributed computing (SCPE)*, vol. 10, no. 2, pp. 173–187, June 2009.