Support Dynamic Network Architecture for Large-Scale Collaborative Virtual Environment

Liang Zhang, Qingping Lin
Information Communication Institute of Singapore
School of Electrical & Electronic Engineering
Nanyang Technological University
liangz@pmail.ntu.edu.sg, iqplin@ntu.edu.sg

Abstract

Collaborative Virtual Environments (CVEs) is a promising technology which provides shared virtual world to the geographically dispersed people to interact with each others. However, the scalability of existing CVE systems is limited due to the constraints in processing power and network speed of each participating host. In this paper, a Mobile Agent based framework for large scale CVE -- MACVE is proposed to support dynamic network architecture. In MACVE, the CVE is decomposed into a group of collaborative mobile agents, each of which is responsible for an independent system task. Agents can migrate or clone dynamically at any suitable participating host which includes traditional servers and qualified user hosts to change the network architecture. This can avoid bottleneck and improve the scalability of CVE. Our system prototype has demonstrated the feasibility of the proposed framework.

Keywords: Collaborative Virtual Environments, network architecture, scalability, mobile agent,

1. Introduction

Collaborative Virtual Environments (CVEs) are computer generated virtual worlds that allow geographically dispersed participants to interact with each other. With the advancements in 3D graphics techniques and the popularity of Internet broadband services, CVE is gaining more and more attentions by many academia and industries. One of the key research issues of CVE is the scalability, which requires a CVE system to provide a spatially large and content-rich virtual environment and to support a large number of concurrent participants. This kind of CVE is called Large-scale CVE (LCVE). To solve the scalability issue, two types of network architecture are adopted in existing CVE systems: peer-to-peer (P2P) architecture and multi-server architecture.

In the P2P architecture, there is no central server to preserving the whole CVE. Each individual peer maintains its own copy of the VE state and exchanges data directly with other peers. By adopting IP-Multicast, CVE with P2P architecture can be scalable, such as NPSNET[1], DIVE[2], SPLINE[3], SCORE[4], etc. However, P2P architecture may not be suitable for supporting heterogeneous peers, such as hosts on the Internet, because their network connection speed and computational capacity may vary greatly, which leads to difficulty in maintaining the consistency of the virtual world; the stability of peer computer hosts and their network connections also make persistency maintenance difficult.

In the multi-server architecture, a CVE is managed by multiple servers. Each participant exchanges data with one or more servers, which can effectively manage the consistency and persistency of the CVE. The servers can also function as proxies to do the computational intensive jobs for the

participants[5]. The workloads on the servers may include: (1) delivering the scene data to each participant; (2) processing and routing the participant's interactive messages to control the consistency of CVE and delivering this consistent CVE state to newly arriving participants; (3) recording the state changes happening in the CVE to maintain its persistency. We defined these workloads on all the servers as system workloads/tasks. The existing multi-server CVE systems include RING[6], NetEffect[7], Community Place[8], CyberWalk[9], and some commercial multiplayer network games, such as EverQuest[10], Ultima Online[11], etc. However, the scalability is not easy to realize for CVE systems with multi-server architecture, because certain servers may become bottleneck due to the unpredictable workloads on them, even though dynamic load balancing is adopted.

In this paper, different from P2P architecture and multi-server architecture, we proposed a Mobile Agent based framework for LCVE (MACVE). Mobile agent is an autonomous software entity that can migrate from one machine to another in a heterogeneous network. Compared with traditional distributed computing schemes, mobile agents promise (at least in many cases) to cope more efficiently and elegantly with a dynamic, heterogeneous, and open environment which is characteristic for today's Internet [12]. Mobile agent has good features for load balancing, distributed system management, software deployment, etc, so we apply mobile agent paradigm to LCVE to dynamically change network architecture for redistributing the system workloads pervasively. This will improve the scalability of the system.

The paper is organized as follows. Section 2 reviews the related work; Section 3 describes the MACVE framework; Section 4 presents the MACVE prototype and experimental results; Section 5 draws conclusions.

2. Related Work

A number of existing CVE systems have attempted to address the issue of scalability. In this section, we briefly review some of them.

NPSNET

NPSNET (Naval Postgraduate School Networked Vehicle Simulator) [13] is a 3D networked virtual environment developed by the U.S. Naval Postgraduate School. NPSNET was the first CVE system to adopt the multicast and also the first successful LCVE system, which can support more than 1000 concurrent users. NPSNET complied with the Distributed Interactive Simulation (DIS) protocol to interoperate with other simulation system, and it incorporates the dead-reckoning algorithms[1] to reduce network traffic to achieve scalability. In NPSNET-IV, it logically partitioned virtual environments by associating spatial, temporal, and functionally related entity classes with network multicast groups. An entity can belong to several groups. Multicast can save the bandwidth, filtering different kinds of traffic in the network interface hardware and does not consume processor cycles. At the same time, it makes network architecture easy to realize. However, NPSNET was designed for military war simulation with dedicated network and hosts. It does not take into account of potentially huge variation in user hosts computing power and network connection speed stability.

Spline

Spline (Scalable Platform for Large Interactive Networked Environments) [14] is a software platform suitable for implementing multi-user interactive environments developed by the Mitsubishi Electric Research Laboratory. Spline aims to make the multi-user Virtual Environment large in spatial extent, large in number of objects, and large in numbers of users interacting with the environment. The Spline platform provides a convenient architecture based on a shared world model. Spline's world model is an

object-oriented database and Spline applications do not communicate directly with each other, but rather only with the world model. To minimize latency, and prevent bottlenecks, the primary communication of the world models is peer to peer. Locales[3] are the central organizing principle of the Spline world model. The concept of locale is based on the idea that even in a very large virtual world most of what a single user can observe at a given moment is nevertheless local in nature. Locales divide a virtual world into chunks that can be processed separately. Each locale is associated with a separate set of multicast addresses, which guarantees the efficiency of the communication. The locales notion can make VE spatially scalable to an arbitary extent. However, an object exists only so long as the application that owns it runs, so to maintain the Spline world to persist overtime is not support by Spline itself.

NetEffect

NetEffect[7] is a highly-scalable architecture for large, media-rich, 3D virtual worlds developed by National University of Singapore. NetEffect partitions a whole virtual world into communities, allocates these communities among a set of servers, and migrates clients from one server to another as clients move through the communities. It provides a load balancing technique which creates a uniform distribution of user density over the servers. To balance the workloads among all the servers, the system dynamically transfers some communities from heavily loaded servers to ones with less workload. However, each communities is a separate virtual worlds, the migration between communities needs the client to request the new community description from the server, which incur a waiting time from sereral seconds to a minute.

CyberWalk

CyberWalk[9, 15] is a web-based distributed virtual walkthrough system developed by the City University of Hong Kong and Hong Kong Polytechnic University jointly. It aims to provide good performance in terms of responsiveness and resolution under the existing constraints of relatively low Internet bandwidth and the large memory demand of virtual objects. CyberWalk adopted a progressive multiresolution modeling technique to reduce the model transmission and rendering time and a caching and prefetching mechanism to reduce the Internet response time. For scalability, it utilizes a multi-server architecture and employs an adaptive data partitioning techniques to dynamically partition the whole VE into regions. All objects with each region are managed by a separate server. The size of each region is dynamically adjusted according to the server loading due to processing the clients' requests and to the network traffics. However, clients' dataflow converge at limited server nodes, which predetermined the maximum numbers of the concurrent participants.

3. MACVE Framework

To effectively decentralize the system workloads, firstly, the expected system tasks of a LCVE should be independent and high granularity. In MACVE, we model a LCVE as a group of collaborative agents. Each agent is a software component which assumes an independent task to provide a certain service for the system. Agents collaborate with each other to maintain the entire LCVE system.

To improve CVE scalability, the system network architecture needs to be dynamically changed according to the system workload distribution. In MACVE, all agents are mobile and do not bond with any fixed host. As the system scales up, agents will be able to migrate or clone to any qualified participating host (include Trusted User Nodes defined in Section 3.1.2) to provide more services. The mutual independence of services and hosts provide large flexibility to utilize the computational and network resources of the system efficiently. Our proposed framework fully takes advantages of such flexibility via Agent Resource Management, Computing Resource Management, Database Resource

Management, VE Content Management and VE Directory Management, which will be discussed in the following paragraphs.

Our framework is divided into three Layers: Resource Layer for System Resource Management, Content Layer for VE Content Management and Gateway Layer for VE Directory Management, which is illustrated in Figure 1. Each layer is composed of multiple collaborative mobile agents to achieve the management.

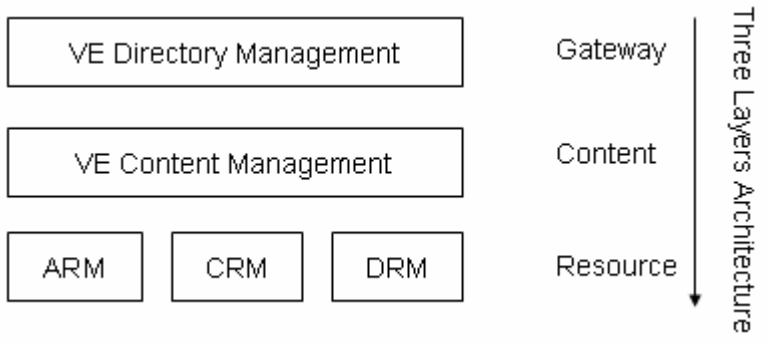

Figure 1 Overall Architecture

3.1 System Resource Management

The bottom layer is resource layer which manages the distribution of system resources. In MACVE, we define mobile agents, System Computing Nodes (defined in Section 3.1.2) and the database as system agent resource, system computing resource and system database resource respectively. Thus, this layer is further subdivided into three modules: Agent Resource Management (ARM), Computing Resource Management (CRM), and Database Resource Management (DRM). Resource layer is independent with different CVE application and scenario. It provides resource management services for the high layers and hides the complexity of the resource distribution.

3.1.1 Agent Resource Management

In a LCVE system, there are many types of agents running at different hosts. Each type of agent provides a certain service. In MACVE, these services does not bond with any host and can be dynamically distributed by agent migrating/cloning, so agents are treated as a kind of resource. To manage all these agents effectively, we use ARM module.

ARM module is realized by an ARM Agent which functions as an Agent Code Repository, Agent Monitor Center, and Agent Command Center. The code of each type of agents should register at ARM Agent who manages these codes and guarantees the consistency and integrity of the agent code running in the whole system. When a new agent begins to run, it will register its network locations and running states at ARM Agent. So ARM Agent provides a directory for all the running agents. When Agents execute some operations, such as clone or migration, ARM Agent will record the Agent operation event log. As an Agent Command Center, ARM Agent can also send commands dynamically to launch agents, transfer agents, clone agents, update agents to new versions, or kill agents etc. to manage network architecture of the LCVE system at runtime.

Since ARM Agent only deal with the messages related to agent command information, which is small in size and does not belong to frequent traffic, there is little chance for an ARM Agent to become a bottleneck. Even though the remote chance happens, ARM Agent can clone itself at multiple SC Nodes (defined in Section 3.1.2) and all these ARM Agents work together to share the workload.

3.1.2 Computing Resource Management

CRM module is designed for effectively managing all the participating hosts to share the system workloads on demand. A typical LCVE system consists of a large number of computer hosts connected through networks. Each host is called a participating node. Based on the belongingness of each participating node, we classify them into two main categories: User Nodes and Service Provider Nodes. User Nodes refer to any user host participating in the LCVE. Service Provider Nodes refer to the nodes belonging to the LCVE system owner.

User Nodes are further classified into Normal User Nodes and Trusted User Nodes based on their functional roles.

- Normal User Node is a host that only allows a user to navigate through the CVE and interact with virtual entities or other users in the CVE.
- Trusted User Node is a host that not only functions as a Normal User Node, but also has spare capacity in terms of computing power, memory and network bandwidth to host system mobile agents. It should at least meet the minimum capability and security requirements set by the LCVE system.

Service Provider Nodes are further classified into Controlling Nodes and DB Nodes based on their functional roles.

- Controlling Node is a host provided by the system owner, which assumes system tasks to manage a LCVE system and maintain the multiuser collaborative interactions in a consistent, persistent, and evolving LCVE.
- DB Node is a host owned by the system owner, which provides the database support for the LCVE system.

Since Controlling Nodes and Trusted User Nodes are used to share the system workloads, we classify both of them as System Computing Nodes (SC Nodes). And system computing resource is defined as the computing and network capability of all the joining SC Nodes in the system.

In MACVE, the system tasks are shared not only by the Controlling Nodes as most conventional LCVE systems do, but also by Trusted User Nodes. When a Trusted User Node logs off, it will transfer all the agents running on it to other SC Nodes. Since User Nodes have relatively less stability, a Trusted User Node may crash before it successfully transfers all the agents on it. MACVE has an agent recovery mechanism to restart the losing agent from the state before it crashes which will be discussed in Section 3.4. Thus, system tasks can be pervasive to more participating nodes.

CRM module is responsible for managing the system computing resource so that each system task will receive enough computing resources. As there may be thousands of SC Nodes in a LCVE system, in order to improve the efficiency of data communication between the SC Nodes, the nodes are grouped according to their IP addresses. Each group will have a group manager to reasonably distribute the workloads among the nodes in this group.

CRM module is achieved by Node Agents, Group Manager Agents, and a CRM Agent whose relationships can be illustrated as Figure 2. A Node Agent runs in each SC Node which monitors the computing load and network traffic of that node. When this node's workload reaches its threshold, Node Agent will decide whether to transfer certain mobile agents in this node to other nodes or to clone certain mobile agents to other nodes to alleviate this node's workloads. A Group Manager Agent runs at a SC Node which is designated as a group manager by the CRM Agent. It will search for the most suitable node in its own group for its member nodes. If it cannot find a suitable node within its own group, the Group Manager Agent will negotiate with CRM Agent to find a suitable node in other groups. The CRM Agent manages all the Group Manager Agents. When joining the LCVE system, every SC Node will register with the CRM Agent and CRM Agent will allocate the SC Node to a group.

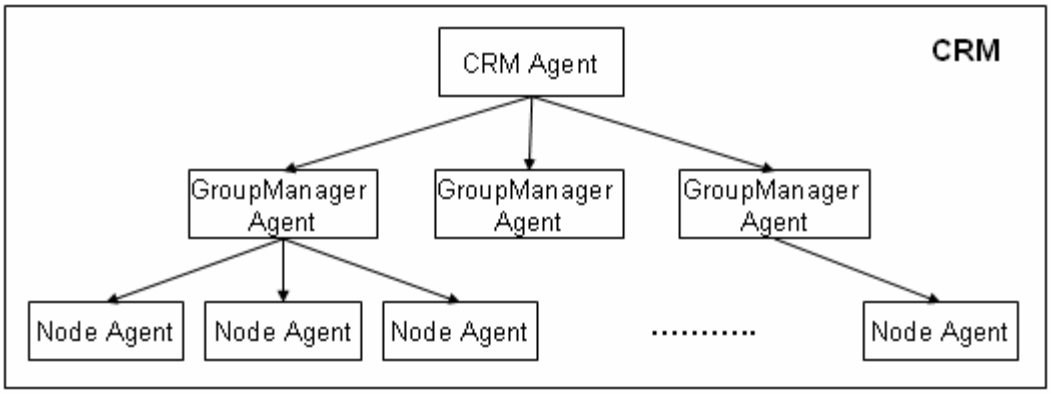

Figure 2 Relationships of Agents for CRM

3.1.3 Database Resource Management

In order to support large VE content, the database is distributed to multiple DB Nodes. As the system evolves, new DB Nodes can be added into the system at any time. DRM module is achieved by multiple DB Agents and a DRM Agent, which is responsible for managing the distributed database. It provides a uniform interface to agents at content layer and gateway layer to access the system database resource.

3.2 VE Content Management

Content layer is on top of resource layer which provides the VE content services to participants. It is application and scenario dependent. In MACVE, the VE is spatially divided into several manageable continuous regions. Each region maybe further divided into cells depending on the requirement of the VE contents. Each cell is a basic unit for VE content retrieving, user group communication, maintenance of VE consistency and persistency. Each cell is associated with a separate set of multicast addresses to guarantees the efficiency of the communication. The management of the VE is in correspondent with the region-cell hierarchy, which is achieved by Region Agents, Cell Agents, Consistency Agents and Persistency Agents as illustrated in Figure 3.

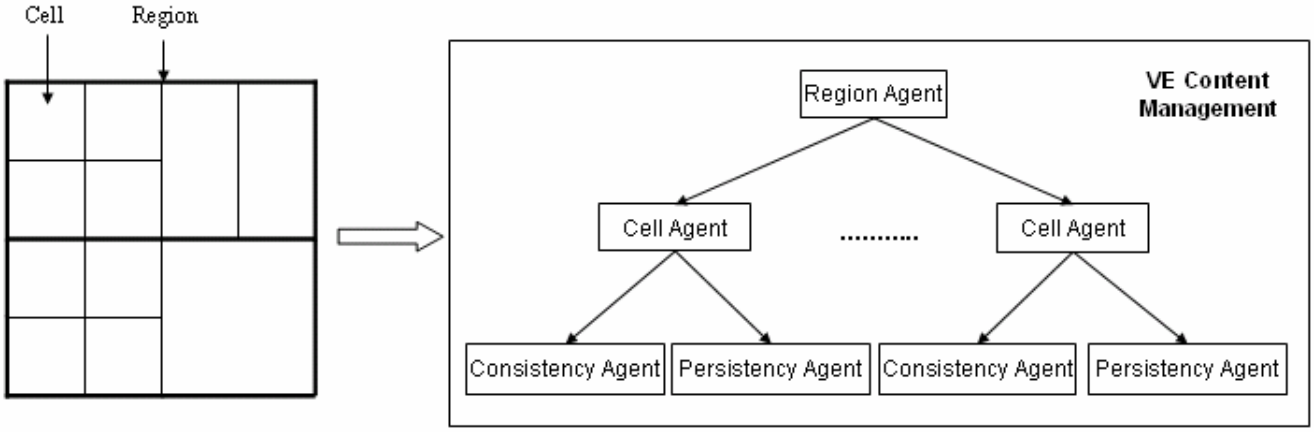

Figure 3 VE Management

Region Agents manage its Cell Agents and make the VE Content Management hierarchically. Region Agent itself does not maintain the VE contents. The VE content is managed by Cell Agents, Consistency Agents and Persistency Agent. Each cell has at least a Cell Agents, a Consistency Agent and a Persistency Agent to provide different aspect of content services and these agents are not necessary to

run at a same SC Node. This can make the system services high granularity so that system workloads are distributed more effectively.

A Cell Agent has two basic components: Scene Data Cache Manager and Scene Data Delivery Manager. Scene Data Cache Manager loads the scene data of a cell from the DB Agent and maintains the local cache of them. Scene Data Delivery Manager delivers the scene data to users on demand. Scene Delivery Manager implements the algorithm to deliver the most needed data with its appropriate level of detail timely, such as predictive pre-fetching or multi-resolution caching mechanism, etc.

When a user navigates through a cell, it will download the scene data of that cell and cache them at local storage to avoid future reloading from the network. As the scene data include 3D graphics data and multimedia data such as texture files, audio, video files, Cell Agents are bandwidth demanding. In order to make use of bandwidth resources of User Nodes, Cell Agent can be cloned at Trusted User Nodes which have enough bandwidth and already cached scene data of that cell. The cloned Cell Agents provide the scene data delivery service together with the original Cell Agent. Thus, the network architecture is dynamically changed to distribute the scene data flow, which avoids bottleneck emerging at limited Controlling Nodes.

A Consistency Agent has three basic components: Concurrency Control, Cell State Manager, and Intelligent Message Router. Concurrency Control preserves consistency and guarantees there is no conflicts when multi-user access a same virtual entity concurrently. Cell State Manager maintains the real-time VE state of the cell and delivers the state to newly arriving users on demand. Intelligent Message Router implements different interaction management strategies and route users' messages to different multicast groups. As Consistency Agents are responsible for processing all the participants' interactive messages, they are computational intensive. In order to make use of idle processor cycles in User Nodes, Consistency Agents can migrate to Trusted User Nodes to execute. Thus, the network architecture is dynamically changed to distribute the consistency services, which also avoids bottleneck emerging at limited Controlling Nodes.

A Persistency Agent has two basic components: Cell State Manager, Cell State Recorder. Cell State Manager preserves the real-time VE state of the cell. In a LCVE, the Persistency Agent and Consistency Agent usually run at different SC Nodes, so the Cell State Manager of Persistency Agent is also a real-time backup of the cell state for the Consistency Agent, if a Consistency crash, a new Consistency Agent can get the latest cell state from its Persistency Agent. Because Consistency Agent and Persistency Agent subscribe to the same multicast group, the duplicated Cell State Manager at both agents will not create any extra traffic to compromise the system scalability. Cell State Recorder sends the changed cell states data to DB Agent to save them periodically to maintain the persistency of the cell.

3.3 VE Directory Management

Since the VE is managed in a distributed manner, it needs a directory service to locate a place, search a virtual entity or a user in the VE, etc. VE Directory Management provides such services. VE Directory Management is achieved by a group of Gateway Agents, which are deployed to multiple well-known Controlling Nodes on the Internet. When a user wants to enter the VE, it connects to a nearest Gateway Agent which directs the user to its intended destination in the VE. Gateway agent also provides user registration, authentication and user log services.

3.4 Agent Recovery

In MACVE, the system workloads can be shared by the Trusted User Nodes. However, User Nodes tend to have less stability compared with Service Provider Nodes. So MACVE provides a mechanism for agent recovery.

In MACVE, the management of all agents is organized as a tree-structure, such as Group Manager Agent manages the Node Agents which belong to its group; or Region Agent manages its Cell Agents. The agent at the root position is called the Parent Agent and the agent at the leaf position is called Child Agent. The Parent Agent is responsible for monitoring the proper execution of its Child Agents. Every Child Agent sends a "heartbeat" message (a living message) periodically to its Parent Agent. If the Parent Agent receives no heartbeat from the Child Agent within a timeout, it detects the crash of the Child Agent. The Parent Agent will ask the ARM agent to send a new Child Agent to other available nodes to resume the system tasks.

For system stability reason, Persistency Agent of a cell is not allowed to be transferred to Trusted User Node, because it maintains the backup of the snapshot of the current cell state. When a Trusted User Node crashes and result in a Consistency Agent lost, the newly launched Consistency Agent will get the latest cell state from the corresponding Persistency Agent at the System Controlling Node. Thus, it will remove the impact of the crashes of Trusted User Node to allow effective recovery of the lost Consistency Agent. If both Consistency Agent and Persistency Agent crash at the same time, the newly launched Persistency Agent will retrieve the cell state from the database. The reloaded cell state is the latest persisted state. Then, the newly launched Consistency Agent gets the cell state from the Persistency Agent and in order to guarantee consistency, all users in this cell roll back their cell state the same as the Consistency Agent.

3.5 Discussion of System Scalability

To support system scalability, MACVE supports dynamic network architecture by migrating and cloning agents to Trusted User Nodes in addition to Controlling Nodes. It combines the strength of both multi-server architecture and distributed P2P architecture while avoiding their weaknesses. Like peer-to-peer, it utilizes the computing resources of user nodes by transferring system tasks to Trusted User Nodes, which avoids the system workloads converging at limited server nodes. At the same time, like the multi-server system, it can maintain the consistency and persistency of the CVE by Consistency Agents and Persistency Agents.

4. System Prototype and Experiment

We have developed a prototype system to conduct experimental studies and evaluate MACVE. The prototype includes three parts: 11 types of Mobile Agents for different system services; a Mobile Agent Environment (MAE) for supporting agent migration and clone; and a web based client user interface for user navigation and interaction.

As the existing mobile agent platforms, such as Aglet[16], Voyager[16], is designed for general purpose which is not suitable for real-time multi-agent communication required in LCVE, we have implemented our own mobile agent platform with our own mobile agent communication protocol designed for LCVE application.

Our experimental CVE consists of 11 cells. Each cell is populated with interactive entities and concurrent users. Users in the CVE can add, remove or interact with entities and chat with each other. We have developed a program --- RobotGroup to simulate a certain number of concurrent users, which send out messages for each simulated user at a constant rate. In the whole CVE, we collect experimental data for various numbers of concurrent users and virtual entities. Figure 4 is the screenshot of user interface during our experiment with 1000 concurrent users and 5000 virtual entities.

To study the agent migration impacts on CVE users, we do experiments to study the effect of number of concurrent users and number of entities in a cell on the migration time spans of a Consistency Agent. The reason we choose to study Consistency Agent migration is that this type of agents maintains users'

real-time interaction in the cell so that its migration is most complex among all types of agents and it has most impacts to users.

Figure 4 Screenshot of MACVE User Interface

The migration of a Consistency Agent includes 3 steps: (1) transferring the agent code to the destination; (2) synchronizing the agent state; (3) shifting the agent control. Transfer & Synchronization Time measures the time required to transfer the agent code and synchronize the agent state. The Transfer & Synchronization process will not directly affect users' collaborative interaction in the CVE as it can be done in a separate thread while the CVE system tasks are performed by the current mobile agents. Handover Time measures the time required to shift agent control which will affect users' collaborative interaction in the CVE. Thus it is particularly important to evaluate the delay caused by the Handover Time.

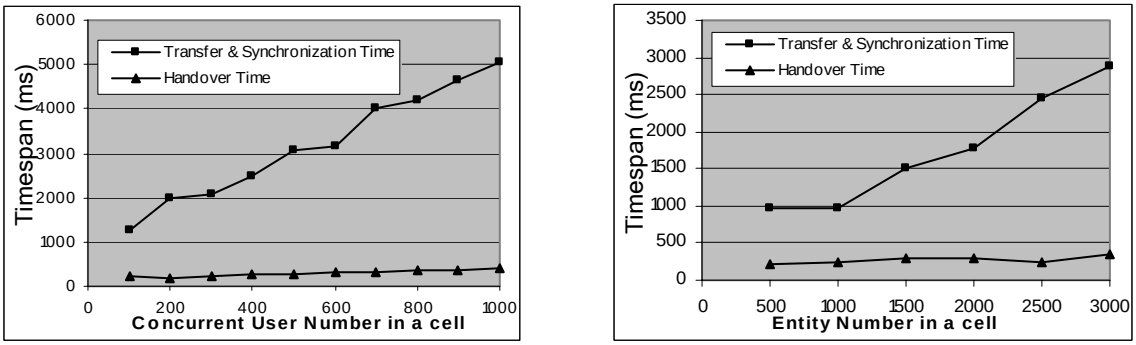

Figure 5 Consistency Agent Migration Time on Concurrent User Number and Entity Number

As shown in Figure 5, we observe from our experiment that the Transfer & Synchronization Time increases with the increase of the number of concurrent users, whereas the Handover Time is relatively stable which is 425.0 ms when the concurrent user number reaches 1000 in a cell. We also observe that the Transfer & Synchronization Time increases with the increase of the number of entities, whereas the Handover Time is relatively stable which is 344.4 ms when the number of entities in the cell reaches 3000. A temporary short delay of less than half a second in updating scene state caused by the Handover Time will have little impact on the performance of a CVE with a large number of concurrent users and virtual entities. We found from the experiment that the users would not feel the migration of the Consistency Agent as the user interaction with the LCVE will not be affected during the agent transfer and synchronization period while the agent control handover time is short enough to avoid apparent interruption.

Our experiments show that the migration of Consistency Agent does not affect the real-time interaction of the CVE users, and so do other types of agents. Therefore, our proposed framework to dynamically change the network architecture will improve the system scalability without compromising CVE performance.

5. Conclusion

In this paper, we have proposed a mobile agent-based framework—MACVE to support dynamic network architecture for LCVE. In MACVE, the system tasks are decomposed into a group of collaborative mobile agents. By migrating and cloning these agents at System Controlling Node and Trusted User Node, the network architecture changed dynamically to pervasively distribute system workloads. This improves the system scalability. Our experiments have demonstrated its effectiveness in supporting large number of concurrent users with real-time interaction in LCVE.

References

[1] M. Macedonia, M. Zyda, D. Pratt, P. Barham, and S. Zestwitz, "NPSNET: A Network Software Architecture for Large-Scale Virtual Environments," *Presence: Teleoperators and Virtual Environments*, vol. 3, pp. 265--287, 1994.

[2] O. Stahl and M. Andersson, "DIVE - a Toolkit for Distributed R Applications," presented at the 6th ERCIM workshop, Stockholm, 1994.

[3] J. W. Barrus, R. C. Waters, and D. B. Anderson, "Locales: Supporting Lange Multiuser Virtual Environments," *IEEE Computer Graphics and Applications*, vol. 16 (6), pp. 50-57., 1996.

[4] E. lety, T. Turletti, and F. Baccelli, "SCORE: A Scalable Communication Protocol for Large-Scale Virtual Environment," *IEEE/ACM Transactions on Networking*, vol. 12, pp. 247-260, 2004.

[5] C.-L. W. Tianqi Wang, Francis Lau, "A Grid-enabled Multi-server Network Game Architecture," presented at 3rd International Conference on Application and Development of Computer Games, 2004.

[6] T. A. Funkhouser, "RING: a client-server system for multi-user virtual environments " in *Proceedings of the 1995 symposium on Interactive 3D graphics* Monterey, California, United States ACM Press, 1995 pp. 85-ff. .

[7] T. K. Das, G. Singh, A. Mitchell, P. S. Kumar, and K. McGee, "NetEffect: A Network Architecture for Large-scale Multi-user Virtual World," presented at the ACM symposium on Virtual reality software and technology, 1997.

[8] R. Lea, Y. Honda, K. Matsuda, and S. Matsuda, "Community Place: Architecture and Performance," presented at the second symposium on Virtual reality modeling language, 1997.

[9] R. W. H. L. Jimmy Chim, Va Leong, Antonio Si, "CyberWalk: A Web-Based Distributed Virtual Walkthrough Environment," *IEEE Transactions on Multimedia*, vol. 5, pp. 503-515, 2003.
[10] "EverQuest." Available at http://everquest.station.sony.com/.
[11] "Ultima Online." Available at http://www.uo.com/.
[12] B. W. Cheng-Zhong Xu, "A mobile agent based push methodology for global parallel computing," *Concurrency: Practice and Experience*, pp. 705-726, 2000.
[13] M. Macedonia, R. M. J. Zyda, D. R. Pratt, and P. T. Barham, "Exploiting Reality with Multicast Groups: A Network Architecture forLarge-Scale Virtual Environments," presented at IEEE VRAIS 95, Las Alamitos, CA, 1995.
[14] D. A. R. Waters, J. Barrus, D. Brogan, M. Casey, S. McKeown, T. Nitta, I. Sterns, W.Yerazunis, "Diamond Park and Spline: A Social Virtual Reality System with 3D Animation, Spoken Interaction, and Runtime Modifiability," *Presence: Teleoperators and Virtual Environments*, pp. 461-81, 1997.
[15] B. Ng, A. Si, R. W. H. Lau, and F. W. B. Li, "A Multi-Server Architecture for Distributed Virtual Walkthrough," presented at ACM VRST'02, Hong Kong, 2002.
[16] A. R. T. N. M. Karnik, "Design Issues in Mobile-Agent Programming Systems," *IEEE Concurrency*, vol. 6, pp. 52-61, 1998.

Liang Zhang received his B.S. and M.S. degrees in Electrical Engineering from Xi'an Jiaotong University, P.R.China in 1999 and 2002. He is currently a Ph.D candidate in Information Communication Institute of Singapore (ICIS) at School of Electrical & Electronic Engineering, Nanyang Technological University. His present research interests have centered around large-scale collaborative virtual environment (CVE), mobile agent, and Grid-Computing.

Qingping Lin received his PhD in Computer Applications from University of Strathclyde, UK, in 1997.He is currently an assistant professor with Information Communication Institute of Singapore, School of Electrical & Electronic Engineering, Nanyang Technological University, Singapore. His current research interest is in the area of virtual reality and large-scale collaborative virtual environment (LCVE). His research work has led to two international patents filed and one spin-off company (cu3D Technologies Pte Ltd, invested by Singapore Technologies Group). He also serves as a member of Editorial Boards of the following international journals: International Journal of Information Technology, International Journal of Computational Intelligence, and International Journal of Signal Processing.