LOADER: A Location-Aware Distributed Virtual Environment Architecture

Behnoosh Hariri[1,2], Shervin Shirmohammadi[2], Mohammad Reza Pakravan[1]

[1] *Department of Electrical Engineering, Sharif University of Technology, Tehran, Iran*

[hariri | pakravan]@ee.sharif.edu

[2] *Distributed Collaborative Virtual Environment Research Laboratory, University of Ottawa, Ottawa, Canada*

[BHariri | Shervin]@discover.uottawa.ca

Abstract

This article proposes a new architecture for distributed data management and update massage exchange in massively multi-user virtual environments. The key points in the design of such environments are scalability and QoS-aware message delivery. Therefore it requires robust distributed algorithms in a dynamic peer-to-peer system with frequent node arrivals and departures. The proposed approach is mainly based on distributed hash tables (DHTs) in order to achieve a decentralized distributed system where any participating node can efficiently retrieve the updates associated with a given object. Therefore, the responsibility of maintaining the mapping from names to values is distributed among the nodes, in such a way that a change in the set of participants causes a minimal amount of disruption. This allows the design to scale to extremely large number of nodes and to handle continual node arrivals, departures, and failures. The original form of DHT has several deficiencies for multi-user virtual environment application. This work introduces LOADER: a Location Aware Distributed Virtual Environment Architecture, which is the result of a series of efforts in alleviating a number of these problems including lack of features such as interest-aware ID assignment. Moreover, efficient multicast is addressed as an inevitable requirement for message exchange in multi-user environments and a solution based on Scribe is proposed.

I. INTRODUCTION

Multi-participant virtual environments have been a major research issue for the virtual reality community in recent years due to the achievements in the networking field that further motivated multi-user collaboration over the Internet. Such applications incorporate computer graphics, sound and haptics to simulate the experience of real-time interaction between multiple users in a shared three-dimensional virtual world while each user runs an interactive interface program on a client computer connected to a wide-area network. In order to share the common sense of a single world, the other users should be notified of a user action when navigating through the virtual world or interacting with objects by visual, sensory or audio feedback. To ensure such awareness, messages have to be exchanged between all users. Therefore multiple entities interacting in the shared virtual environment frequently send messages to one another to announce updates to their own geometry or behavior, modification to the shared environment or impact on the other objects. As the number of virtual environment users and entities increase, there would be a huge bulk of update messages exchanged among the users and entities. How these messages are distributed among the users is an important factor that depends on the overlaying framework of the virtual environment. Internet has been and is expected to be the dominant substrate over which virtual environment users are connected to each other. This article discusses the issues related to message exchange in massively multi-user virtual environments over the Internet in an attempt to offer better and more efficient solutions.

When it comes to the area of distributed virtual environments, the first important issue is to find peers in charge of updating the entities in the environment. In a distributed architecture with no centralized server, there would be no single manager to take care of status update for all nodes. Therefore a distributed update mechanism should be used in the network so that peers share the task of status update among themselves. This way each entity would be assigned an owner whose role is to provide a substrate for the distribution of update messages for that entity to all those who are interested in it.

Optimum owner selection would be the next step in the distributed virtual environment (VE) design. There are mainly two considerations in the owner selection mechanism. The first one is the proper owner location in a way that updates do not have to travel a long distance on the network in order to reach the objects of interest. The second one is that there should be no centralized indexing mechanism mapping owners to the objects, as such a centralized node would become a bottleneck and also deviate from a P2P paradigm; therefore we should have a distributed owner searching mechanism in an environment where nodes must be able to join and leave the system frequently without affecting the whole system robustness. Moreover the update message forwarding role should also be balanced across the available nodes.

Peer-to-peer file sharing systems have widely adopted distributed hash table (DHT) as a way to efficiently locate the

node that stores the desired data in a large distributed system. However; DHT in its original format lacks some essential requirements for application in virtual environments. One of these requirements is locality-aware routing. Users in a virtual environment are mainly considered to exchange messages in a clustered manner. Virtual environment is therefore divided into sub areas called zones where the probability of message exchange within each zone is much higher than inter-zone message exchange. Therefore the owner of an object should be selected from the same zone as the object in order for the routing mechanism to be efficient. The ID-based owner assignment in DHT is random and does not contain location information. Hence, we propose in the following sections the idea of locality aware ID assignment and routing for DHT. This way we can still take advantage of several promising features of DHT such as scalability, load balancing, and distributed indexing, while the clustering feature of virtual environment users is also exploited in owner assignment and message routing.

The other issue in update message routing for virtual environment is the QoS requirement in transport. One of these key requirements is the latency in update message delivery. DHT aims at minimizing the total hop-count during message transmission while it does not take the real physical hop distances into account. Topology–aware routing in DHT has been a research area where several approaches have been proposed for topology-aware DHT networks. In these solutions, real under-lying network benchmarks such as delay are considered in both routing table construction and forwarding decision [1][1].

As the users in a virtual environment are meant to interact in a shared environment, most of the update messages in the environment are destined to a number of users interested in a specific entity for which the update message has been generated. Therefore, update messages are generated by one source and should reach many destinations, implying a multicast distribution mechanism. Hence, selection of an efficient multicast protocol is among the key challenges in the design of distributed virtual environments. IP Multicast seems to be the ideal solution for content distribution in such cases. It can serve content to an unlimited number of recipients, and it is bandwidth efficient. However, it has neither been widely deployed on the Internet nor adopted by most commercial ISPs, thus large parts of the Internet today and for the foreseeable future are still incapable of native IP multicast. Also, the very dynamic nature of node and object behaviour in a VE would not be suitable for the much more static nature of IP multicasting.

The lack of an efficient multicast mechanism in network layer has encouraged application developers to shift this functionality from the third layer (networking) to the top layer (application) where they named it as ALM (application layer multicast). In ALM, data packets are replicated at end users. Therefore end users self-organize into a logical overlay network and transfer data along the edges of the overlay network using unicast. The goal of ALM protocols is thus to construct and maintain efficient overlays for data transmission. Readers are referred to [19][19] for a detailed discussion of ALM. Update message transport in distributed virtual environments also seems to be best handled by ALM. In This article, we propose LOADER: a distributed architecture for object allocation, routing and multicast in a distributed virtual environment using locality aware DHT and ALM.

II. Related Work

In recent years, much research has been done in the area of scalable multi-user virtual environments among which P2P based networked virtual environments (NVE) have been discussed as an option for scalability.

As for DHT based architectures, Knutson et al proposed SimMud [2][2] where NVE is divided into regions and a coordinator is assigned to each region using random hashing. Coordinator would then act as the root of the multicast tree for the region through which all updates are exchanged. It would also be responsible for the shared objects. The whole application was built on top of Pastry and Scribe was used as multicast scheme over it. We will briefly review the principles of Pastry and Scribe in the coming sections. In this architecture, the coordinator can be overwhelmed by the presence of many nodes in its region. Also, the non-interest aware random ID assignment in Pastry will result in non-efficient architecture in which nodes with the same interest maybe located far apart in the ID space.

Limura et al's Zoned proposed Federation of Games Servers [3][3] in which all global states of a multimedia online game (MOG) are separated into several zones and distributed among peers using DHT. However DHT use is only limited to storage and rendezvous and once the authoritative node for a particular global state is found by DHT, a direct connection will be established to avoid the DHT latency. Crowding within zones would still be a concern here. Moreover, the random selection of zone owner may cause updates to be relayed from a far-apart point.

MOPAR[4][4] is a mix of DHT neighbor-list exchange between zone masters, and direct transfer among zone slaves. OPeN [5][5] is built on the top of Chord/Floc and proposed the use of a distributed spatial index to facilitate the discovery and query of relevant entities and an agent-based approach to facilitate real time interactions between the entities. Colyseus [6][6] is also a distributed architecture whose object query interface is based on DHT with the addition of speculative pre-fetch to exploit locality for compensation of DHT delay. However, updates are propagated from primary

copies to replicas directly. Thorsten et al [7][7] Proposed the idea of using Pastry for overlay, Past for object management and Scribe for multicast. Further details on the implementation have not been mentioned while it should be noted that these standards are not efficient to be applied to distributed VE design in the original format. Finally [8][8] proposes Mediator as a Design Framework for P2P MMOG. It adopts a hybrid communication architecture whereby a structured P2P overlay[10][10] is used in the peer bootstrapping process, application layer multicast is used for game zone structure maintenance, and direct P2P connections are established on demand for the time critical events.

III. Overall Network Architecture

One of the important challenges in a multi-user virtual environment is maintaining a consistent state among all participants. Therefore, whenever the state of one object changes, updates should be sent to all other related entities reporting this change. Two distribution approaches may be used for object state management in a shared environment of a virtual reality (VR) system. One approach is active replication in which a copy of each object is distributed among all VR nodes and the updates will be sent by flooding to all entities. The other approach is replication on demand in which an object or parts of it are only sent to a VR node if it expresses interest in that object. In this case, updates are also distributed among all of the interested nodes whenever a change happens in that object. As there are a lot of entities in massively multi-user virtual environments, the selective copy approach seems to work much better due to the huge reduction in update message transmission. However, copy management has its own challenges in this case. Either a centralized server for all objects or local owners of the VR has to be set up as the object manager. This manager would be then responsible for the object copies, i.e. the manager knows where to send object changes. The manager of the object is in fact the virtual owner of the object; it will take care of the multiple copies of the object and handles the issues related to inconsistent copies by allowing only one entity to modify the object at anytime.

The whole virtual environment is therefore partitioned with one participant managing any given partition. Therefore a table is needed that maps a partition to its manager's network address. A simple scheme is to let a central server maintain the mapping. However, there are many participants in a VE system and they span wide-area networks. Their possibly short lifetimes result in frequent arrivals and departures. Also, central servers are single points of failure and potential performance bottlenecks. However, it should be noted that, theoretically speaking, the server only does a kind of relaying job for update messages. That is, the Virtual environment is inherently scalable, but the hard part is finding the nodes from whom to get the update message. Thus, a scalable virtual environment requires, at the very least, a scalable indexing mechanism.

In a distributed virtual environment where there would be no server to act as a single object owner, The object ownership role must be distributed among participating nodes and the challenge would be to find the best owner for one object that could take care of all replicas of the object in the other nodes in the most efficient way. Moreover we would need a distributed index for mapping objects to their owners.

DHT has been widely used in large scale P2P networks as a good option for handling both routing and object allocation. It obviates the need for central servers by creating an overlay network among the participants. Hash lookups are routed to appropriate managers using the overlay.

DHT functionality has been used as a base in several popular distributed resource sharing systems. Chord [9][9], Networks (CAN) [11][11], Tapestry [12][12] and Pastry [13][13] are the most popular ones.

All these networks use DHT hashing mechanism to map hosts to file names. Such Structured P2P systems use the DHT as a substrate, in which data object location information is placed deterministically, at the peers with identifiers corresponding to the data object's unique key. DHT-based systems have a property that consistently assigned uniform random Node IDs to the set of peers into a large space of identifiers. Data objects are assigned unique identifiers called keys, chosen from the same identifier space. Keys are mapped by the overlay network protocol to a unique live peer in the overlay network. The P2P overlay networks support the scalable storage and retrieval of {key, value} pairs on the overlay network. Among the DHT-based networks Tapestry is the oldest and the most complex one. Other networks such as CAN, Chord, and Pastry have also been developed and have deep similarities. CAN uses a Multi-dimensional ID coordinate space, Chord Uni-directional and Circular nodeID space, while Pastry and Tapestry use Plaxton mesh network. Pastry allows greater choice of neighbors due to more flexibility. Therefore Pastry seems to be a promising choice to construct the substrate of a DHT based network for a multi user virtual environment due to its flexibility. Moreover we can later take advantage of Scribe [13][13][15][15] as an efficient multicast scheme over Pastry.

Each Pastry node has a unique, 128-bit nodeId. The set of existing nodeIds are uniformly distributed; given a message and a key, Pastry reliably routes the message to the Pastry node with the nodeId that is numerically closest to the key, among all live Pastry nodes. Assuming a Pastry network consisting of N nodes, Pastry can route to any node in less than $\mathrm{Log}_{2^b}(N)$ steps on average where b is a configuration parameter with a typical value of 4.

The tables required in each Pastry node has only $\text{Log}_{2^b}(N)(2^b-1)+1$ entries, where each entry maps a nodeId to the associated node's IP address. A node's routing table is organized into $\text{Log}_{2^b}(N)$ rows with (2^b-1) entries in each row. The (2^b-1) entries in row n of the routing table each refer to a node whose nodeId matches the present node's nodeId in the first n digits, but whose $(n+1)^{th}$ digit has one of the (2^b-1) possible values other than the $(n+1)^{th}$ digit in the present node's id. Each entry in the routing table refers to one of potentially many nodes whose nodeId have the appropriate prefix. In addition to the routing table, each node maintains IP addresses for the nodes in its leaf set, i.e., the set of nodes with the l/2 numerically closest larger nodeIds, and the l/2 nodes with numerically closest smaller nodeIds, relative to the present node's nodeId.

Although our application uses concepts from Pastry, it should be noted that Pastry, in its original format, lacks some essential requirements for application in virtual environments. One of the major requirements for NVE design is minimizing latency in message delivery. In a DHT based network this can be translated into two parts First, we should build the design in a way to minimize hop count for messages and, second, we should select the hops in a way to choose the path with the shortest delay. In order to satisfy the condition of minimum hop routing; we will propose the idea of Interest-aware ID assignment in section IV~~I~~V. This design helps minimizing the hop count by localizing the DHT in the zones so that update messages inside a zone are not relayed by the nodes outside of it.

IV. INTEREST-AWARE ID ASSIGNMENT

Pastry does not consider the location of the nodes for the assignment of identifiers. This random assignment of owners is useful in applications like file-sharing systems where the underlying P2P network can be considered random with no pre-specified clusters. However the case would be totally different in virtual environments where the traffic exchange among users is mostly limited within a zone known as area of interest (AoI). Application of DHT in simple form to VE nodes may results in nodes in the same AoI to be placed far-apart in the Hash space. Therefore they will be connected via a large distance and multiple hops thus resulting in high routing latency and low efficiency where nearby nodes with close node identifiers on the overlay can be located far away from one another in the virtual environment and rarely have a demand for update message exchange.

Topology-based nodeId assignment attempts to map the overlay's logical Id space onto the physical network such that neighboring nodes in the Id space are close in the physical network; hence exploiting network locality into node identifiers. This approach destroys the uniform population of the Id space, causing load balancing problems in the overlay and does not work well in overlays that use a one-dimensional Id space (Chord, Tapestry, Pastry) while it works well for CAN[16]~~[16]~~.

The other approach in location based ID assignment is to divide the whole node space into several regions. A location based node ID is a concatenation of a hierarchical prefix assigned to a node's region and a suffix of randomly generated bits. The scheme is based on nodes zone, that is, different prefixes are assigned to different regions. Chord6 [17]~~[17]~~ is a modified version of Chord with the approach of geographic layout.

In order to build our enhanced DHT, we slightly modify the process of producing node identifiers. In our proposed approach, a node identifier contains two parts: the higher bits are obtained by hashing the node's zone ID, while the remaining lower bits are the hash value of node's IP address. With the new node identifiers production mechanism, all the nodes in a zone will be mapped onto a contiguous key space on the overlay. That is, nearby nodes in the virtual environment shall also be neighbors on the overlay. As a result, it is highly unlikely that messages will be routed many times into and out of the same zone, thus significantly reducing routing latency and improving efficiency.

Theoretically in a system with N nodes dispersed in M different zones, the end-to-end routing latency shall be O(log(N)). However, all the nodes in the same zone don't participate in other zones. Instead they will be resident in a continual logical space. Therefore, routing latency within the same zone will be O(log(N/M)), while between zones it is O(log(M)).

V. MULTICASTING OVER DHT BASED NETWORKS

Update messages in a virtual environment are mostly exchanged in groups. The owner of an object should generally send the update messages for that object to a group of nodes interested in the object. Therefore efficient multicast within zones is among the challenges in overlay design for virtual environments.

We also aim at building a scalable application-layer multicast infrastructure built on top of an overlay. Any VE node owning an object may create a group; other nodes interested in that object can then join the group to receive the update messages for that specified object.

Our approach for multicast is mainly inspired by Scribe that is a large-scale, decentralized application level multicast infrastructure built on the top of Pastry. Scribe is fully decentralized: all decisions are based on local information, and each node has identical capabilities. Each node can act as a multicast source, a root of a multicast tree, a group member, a

node within a multicast tree, and any sensible combination of the above.

Each group has a unique groupId. The Scribe node with a nodeId numerically closest to the groupId acts as the rendezvous point for the associated group. The rendezvous point is the root of the multicast tree created for the group. We can assign the creator of a group to be the rendezvous point for the group with the groupId selected as the concatenation of the nodeId of the creator and the hash of the textual name of the group.

As the groupId should be available to those who are interested to receive multicast messages for a specified object, the group name can be chosen the same as the object name corresponding to the multicast group.

To create a group, a Scribe node that is the owner of an object in our case sends a CREATE message using the groupId as the key. A multicast tree is then created and rooted at the rendezvous point that is the same as object owner. To disseminate the multicast messages in the group the tree is formed by joining the Pastry routes from each group member to the rendezvous point. Group joining operations are managed in a decentralized manner to support large and dynamic membership.

Virtual environment nodes that are part of a group's multicast tree are called forwarders with respect to the group; they may or may not be members of the group. Each forwarder maintains a children table for the group containing an entry (IP address and nodeId) for each of its children in the multicast tree.

When a VE node has an object in its area of interest, it can find the groupId related to that object knowing only the object Id. Therefore it can join the right group to receive related update messages for that object; it sends a JOIN message with the group's groupId as the key. This message is routed by overlay towards the group's rendezvous point. At each node along the route, the corresponding node checks its list of groups to see if it is currently a forwarder; if so, it accepts the node as a child, adding it to the children table. If the node is not already a forwarder, it creates an entry for the group, and adds the source node as a child in the associated children table. It then becomes a forwarder for the group by sending a JOIN message to the next node along the route from the joining node to the rendezvous point.

When a VE node is no longer interested in an object and wishes to leave the corresponding group, it checks its children table. If there are no other entries in the children table, it sends a LEAVE message to its parent in the multicast tree. The message proceeds recursively up the multicast tree, until a node is reached that still has entries in the children table after removing the departing child.

The properties of underlying Pastry based routes ensure that this mechanism produces a tree. There are no loops because the nodeId of the next node in every hop of a Pastry route matches a longer prefix of the groupId than the previous node, or matches a prefix with the same length and is numerically closer, or is the nodeId of the root.

VI. PERFORMANCE EVALUATION

In this section, we present experimental results obtained with an initial development of the whole architecture in FreePastry [18]~~[18]~~ that is an open-source implementation of Pastry. The preliminary results are presented here while we have left the whole evaluation to the full paper version.

Simulations have been done in a virtual environment of 2500 nodes with 25 zones where zones have an average of 100 members. Figure 1 compares the average hop count in data transmission with and without our interest-aware ID assignment policy.

The Simulation results further demonstrate the effectiveness of interest–aware ID allocation on the message exchange efficiency.

(Note: more results in the final version)

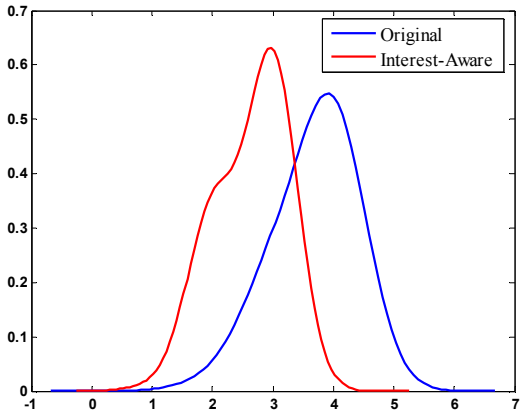

Figure. 1 Hop Count Distribution for Original Pastry and Interest- aware modified version

VII. SUMMARY AND CONCLUSION

This paper presents a novel distributed architecture for efficient update message exchange in massively multi-user virtual environments. The architecture design aims at decentralized reliable message delivery scheme in a continuously changing environment where nodes join and leave frequently. The proposed architecture is mainly based on Pastry that is a DHT-based scalable, distributed object location and routing substrate for wide-area peer-to-peer applications running over the Internet. However Pastry that was mainly motivated by peer-to-peer file sharing systems does not directly address a number of challenges in distributed virtual

environments. In this article we have proposed an architecture that is specifically customized for efficient update message exchange in distributed massively multi-user virtual environments. We proposed a new interest–aware ID assignment scheme to address the issue of local routing inside VE zones. Moreover we have proposed a multicast scheme based on Scribe that is a decentralized publish/subscribe system based on Pastry for its underlying route management and host lookup.

VIII. REFERENCES

[1] M. Castro, P. Druschel, Y. C. Hu, and A. Rowstron, "Topology-aware routing in strucutured peer-to-peer overlay networks," Technical Report MSR-TR-2002-82, Microsoft Research, Redmond, WA, USA, 2002

[2] B. Knutsson, H. Lu, W. Xu, B. Hopkins, "peer-to-peer Support for Massively Multiplayer Games," In INFOCOM,Mar, 2004

[3] T. Limura, H. Hazeyama, Y. Kadobayashi, "Zoned Federation of Game Servers: a Peer-to-Peer Approach to Scalable Multi-player Online Games," In Proc. SIGCOMM'04.

[4] A. Yu and S. T. Vuong, "MOPAR: a Mobile Peer-to-Peer Overlay Architecture for Interest Management of Massively Multiplayer Online Games," Proc. NOSSDAV, Jun. 2005, pp. 99–104.

[5] S. Douglas, E. Tanin, A. Harwood, and S. Karunasekera, "Enabling Massively Multi-Player Online Gaming Applications on a P2P Architecture," in Proc. IEEE Intl. Conf. Information and Automation, Dec. 2005, pp. 7-12.

[6] A. Bharambe, J. Pang and S. Seshan, "Colyseus: A Distributed Architecture for Multiplayer Games," in Proc. ACM/USENIX NSDI, May 2006.

[7] T. Hampel, T. Bopp, and R. Hinn, "A Peer-to-Peer Architecture for Massive Multiplayer Online Games," in Proc. Netgames, Oct. 2006.

[8] L. Fan, H. Taylor and P. Trinder, "Mediator: A Design Framework for P2P MMOGs," in Proc. Netgames 2007, Melbourne, Australia, Sept. 2007, pp. 43-48.

[9] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," In Proc. of the ACM SIGCOMM '01, San Diego, California, August 2001.

[10] E.K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim, "A Survey and Comparison of Peer-to-Peer Overlay Network Schemes," IEEE Communications Survey and Tutorial, 2004.

[11] S. Ratnasamy, P. Francis, M. Handley,R. Karp and S. Shenker ," A scalable content-addressable network," In Proc. ACM SIGCOMM ,San Diego, CA, August 2001, pp. 161–172.

[12] B. Zhao, j. Kubiatowicz and A. Joseph, "Tapestry: An infrastructure for fault-tolerant wide-area location and routing," Tech. Rep. UCB/CSD-01-1141, University of California at Berkeley, Computer Science Department, 2001.

[13] P. Druschel and A. Rowstron, "pastry: Scalable, distributed object location and routing for large-scale peer-to peer systems,"

[14] M. Castro, P. Druschel, A.-M. Kermarec, and A. I. T. Rowstron, "Scribe: A large-scale and decentralized application-level multicast infrastructure," in IEEE J. Sel. Areas Commun., vol. 20, Oct. 2002, pp. 1489–1499

[15] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast", Proceedings of ACM SIGCOMM 2002 Pittsburg, PA, August 2002. Proc. of Middleware 2001, Nov 2001.

[16] M. Castro, P. Druschel, and Y. C. Hu, "Topology-aware routing in structured peer-to-peer overlay networks," Microsoft research technical report msr-tr-2002-82.

[17] J. Xiong, Y. Zhang, P. Hong and J. Li, "Chord6: IPv6 based topology-aware Chord," in Proc. ICNS'05, 2005.

[18] http://freepastry.org/FreePastry/

[19] M. Hosseini, D.T. Ahmed, S. Shirmohammadi, and N.D. Georganas, "A Survey of Application-Layer Multicast Protocols", IEEE Communications Surveys and Tutorials, Vol. 9, Issue. 3, 2007, pp. 58 – 74.