Self-organizing Spatial Publish Subscribe

Shun-Yun Hu
Institute of Information Science
Academia Sinica, Taiwan
syhu@iis.sinica.edu.tw

Kuan-Ta Chen
Institute of Information Science
Academia Sinica, Taiwan
swc@iis.sinica.edu.tw

ABSTRACT

Virtual environments (VEs) such as Massively Multiplayer Online Games have grown in popularity over recent years. However, existing architectures have yet been able to support over one million concurrent users in a single world. A key challenge involved is to allow entities in a VE to discover other relevant entities in constant time regardless of entity distributions. A scalable spatial publish / subscribe (SPS) service, where entities can specify a subscription area and receive messages published by other entities to an area, may be a flexible and fundamental primitive. We describe Voronoi Self-organizing Overlay (VSO), which extends a Voronoi-based Overlay Network (VON) to support SPS operations in constant time and performs automatic load balancing. New classes of million-scale VEs may thus be built.

Categories and Subject Descriptors

C.2.1 [**Computer Communication Networks**]: Network Architecture Design

General Terms

Algorithms

Keywords

publish/subscribe, virtual environment, MMOG

1. INTRODUCTION

Massively Multiuser Virtual Environments (MMVEs) such as Massively Multiplayer Online Games (MMOGs) have become very popular in recent years. Currently, scalability is achieved with two main techniques: 1) replicating many parallel world instances (called *sharding*); and 2) geographic decomposition (i.e., partitioning the world into *regions* and assign each to a server) [5]. However, neither approach can yet support a virtual environment (VE) with millions of concurrent users *in a single world instance*. Each shard has a maximum number of users, who cannot migrate or communicate across worlds, limiting the realism of interactions. While geographic partitioning may scale with user size (as long as enough servers are available and each server can handle the users in its region), existing partitioning is often hard-wired to world design and requires manual adjustment. Given that user distribution often varies greatly across regions and time of day [3], this means prohibitively expensive operating costs (for running servers without users and manually adjusting region sizes), or overloaded and unresponsive servers (if the region has too many users).

Many researchers have proposed alternative peer-to-peer (P2P) architectures [1]. However, while P2P may be more economic by utilizing inexpensive clients, they still face the same geometric partitioning problem due to uneven user distribution. Recent works suggest that by separating *interest management* from *state management* (i.e., the task of determining and delivering the messages of interest can be separated from the tasks of computing state changes [4, 5]), the scalability limit inherent in geometric partitioning thus may be alleviated. Regardless of P2P or client-server, we identify the fundamental interest management required by any VE applications as *spatial publish / subscribe* (SPS), and argue that most existing VE functions can be reduced to SPS operations [1]. An implication is that if SPS operations can be made highly scalable and self-organizing, a VE supporting millions of concurrent users may become possible.

2. SELF-ORGANIZING SPS

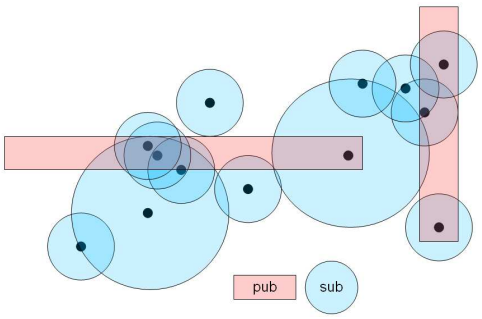

Figure 1: 2D Spatial Publish Subscribe.

In SPS, each *node* in a coordinate space can specify a *publication space* and a *subscription space*. A node may send a message to a specified publication space, which will then be delivered to all nodes whose subscriptions overlap with the publication space (Figure 1). Subscription spaces can further be updated continuously as nodes move.

In Voronoi Self-organizing Overlay (or VSO), the entire virtual space is divided into various *regions*, each managed by an *interest matcher*, or simply *matcher*, whose responsibility is to map a given publication with potential sub-

ICAC'11, June 14–18, 2011, Karlsruhe, Germany.
ACM 978-1-4503-0607-2/11/06.

scribers. A *client* (i.e., entity) joins the system to perform publications or subscriptions. Additionally, clients may also *move* their existing subscribed area to receive more relevant and timely messages. For simplicity, we assume that publications or subscriptions are specified as circles or rectangles, and have a well-specified *center* point. Each matcher is the unique authority in a region, such that a client needs to register its subscription with at least one matcher, before receiving any publications. To ensure that subscriptions are updated authoritatively, each subscription has an *owner* matcher whose region covers the subscription center. A client joins by contacting any existing matchers to notify its subscription. If non-owners receive it, the subscription is forwarded greedily based on the center to the actual owner matcher. It is also forwarded to all relevant matchers, if the subscribed area covers neighboring regions. Once joined, clients can move to new locations by changing their subscribed areas. If the subscription center enters another region, an explicit *ownership transfer* should occur.

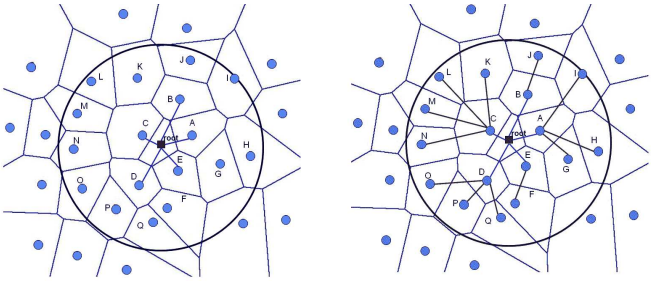

Figure 2: Non-redundant multicast tree.

A client publishes by sending messages to its current matcher, which checks for a list of known subscribers. If the publication covers other regions, it is forwarded to neighboring matchers continuously, until all matchers with overlapped regions are notified. The forwarding is based on VoroCast [2], which builds a spanning tree covering all the matchers within the publication area with non-redundant paths (e.g., in Figure 2 left, a publisher sends four copies of the same message to its immediate neighbors. Its neighbors then forward the message in a non-redundant way, see right).

To manage the pub/sub matching, the matchers form a Voronoi-based Overlay Network (VON) [1] to allow decentralized overlay connectivity and dynamic load balancing. Voronoi diagram is a way to divide a space into n regions based on the positions of n points, or *sites*, such that all points within a region are closest to that region's site than to any other site. Each node on a VON (in our case, a matcher) specifies an Area of Interest (AOI) and organizes other neighbor nodes with a Voronoi diagram. Based on the Voronoi diagram, a node can identify its *boundary neighbors* (i.e., neighbors whose Voronoi regions touch on the AOI boundary), so when a node moves to a new position, its boundary neighbors could help to notify the existence of any new neighbors (Figure 3). This allows a moving node to discover neighbors continuously in a fully distributed manner. In our case, each matcher's AOI is the combined subscription areas of all the clients the matcher manages.

We note that in a Voronoi partitioning, it is fairly easy to adjust region boundary, by moving the *sites* of the Voronoi diagram. The main questions then are: 1) how to move the site locations, as client density changes; and 2) where and when to insert new matchers, if overload persists.

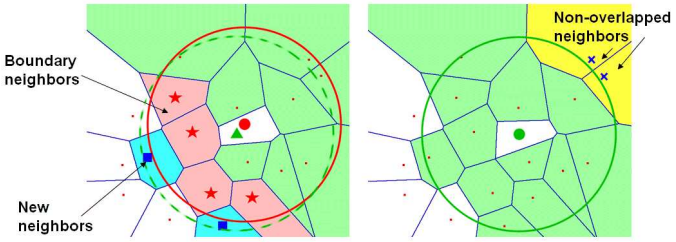

Figure 3: Neighbor Discovery in VON.

We experimented with a number of approaches and found the following simple rules. Intuitively, the ideal partitioning matches the region size with the region's load (which should reflect the matcher's capacity). In other words, the region size should change in such a way that the average client density per square area roughly equals the handling ability of the matcher per square area.

The adjustment rules for a matcher is twofold:

1) shrink region size when overloaded by asking neighbors to move their sites closer.

2) request a matcher insertion from a gateway, if the first method does not work after a while.

Each time a neighbor matcher moves, it uses the following to adjust the distance (where P stands for *position*):

$P\_new = P\_current + (P\_overload - P\_current) * F\_move$

Where F_move is a number between 0 and 1 that determines how speedy the adjustment should be. While the above may appear intuitive, the site positions would soon cluster when clients are moving, thus making the clients to cross region boundaries easily. A second requirement thus is to have each region centers (e.g., the *sites*) to be as far away from each other as possible, so that region crossing can be minimized. We thus devise a third rule as follows:

3) a matcher will continuously move its site to the center of clients that it manages, using the formula:

$M_c = M_c + (L_c - M_c) * f$

Where M_c is the site position of the matcher (i.e., the matcher's center), L_c is the center of the clients' positions (i.e., the load center), and f is an adjustable parameter for how much fractional adjustment should be made each time.

With the above rules, a Voronoi partitioning can be in constant adjustment of its region shape, such that the number of clients in a region will follow the matcher's capacity.

3. REFERENCES

[1] S.-Y. Hu et al. A spatial publish subscribe overlay for massively multiuser virtual environments. In *Proc. ICEIE 2010*, 2010.

[2] J.-R. Jiang, Y.-L. Huang, and S.-Y. Hu. Scalable aoi-cast for peer-to-peer networked virtual environments. In *ICDCS Workshops*, 2008.

[3] Y.-T. Lee and K.-T. Chen. Is server consolidation beneficial to mmorpg? In *IEEE Cloud*, 2010.

[4] M. T. Najaran and C. Krasic. Scaling online games with adaptive interest management in the cloud. In *Proc. NetGames 2010*, 2010.

[5] J. Waldo. Scaling in games & virtual worlds. *ACM Queue*, 6(7), 2008.