Deployment Issues of Voronoi Self-organizing Overlays

Man-Chun Li
Department of Electrical Engineering
National Taiwan University
Taiwan
Email: pennytaipei@gmail.com

Shun-Yun Hu and Kuan-Ta Chen
Institute of Information Science
Academia Sinica
Taiwan
Email: syhu, swc@iis.sinica.edu.tw

***Abstract*—Spatial publish subscribe (SPS) is a basic primitive underlying many real-time, interactive applications such as online games or discrete-time simulations. Voronoi Self-organizing Overlay (VSO) is a scalable SPS service designed to adjust workload automatically to avoid system overload or underload. We investigate the deployment of VSO on PlanetLab, to evaluate whether it is feasible to scale up SPS operations in real environments. Our results show that by ensuring enough capacities for super-nodes (called matchers), such automatic load balancing can scale up a Second Life region to over 200 entities while still maintaining proper discovery consistency.**

I. Introduction

Spatial simulations allow coordinate-based entities to move within a Cartesian space, based on certain rules and a logical time. A common requirement in spatial simulations is to know, or send messages to entities within a radius. For example, Massively Multiplayer Online Games (MMOGs) require players to know other players within an *area of interest* (AOI), so that other players' positions and actions can be displayed. Such a requirement in general can be described by a *spatial publish subscribe* (SPS) [1] service. In SPS, each entity subscribes or publishes to an area, and will only receive messages if its subscribed area overlaps with the message's publication area. How to scale up SPS operations thus is important to scale up spatial simulations. Partitioning the space into regions and assign manager nodes for each region is a common technique. However, when entity densities change unexpectedly, or if many entities are simulated, manager could overload or underload, causing quality of service to degrade or resource idling [2].

We have designed a scalable SPS service called Voronoi Self-organizing Overlay (VSO) [1], where the spatial division is done via Voronoi diagrams, and can be adjusted flexibly given the loading at each manager. VSO extends a Voronoi-based Overlay Network (VON) [1] and follows the basic idea of spatially partitioning a virtual space to support SPS. The entire virtual space is divided into various regions, each managed by a super-node called *matcher*. The partitioning is based on Voronoi Diagrams, which partitions n sites into n regions, where all points within a given region is closest to the region's site than to any other site. The matchers are responsible to map a given publication with potential subscribers. We define a *client* as an entity of the system that performs publications or subscriptions. Additionally, clients may also move subscribed areas to receive more relevant and timely update messages. For simplicity, we assume that publications or subscriptions are circular areas. Each subscription thus is defined by a subscription area, and each publication is in the form of $(area, message)$, where $area$ is the publication area, and $message$ is an application-specific message.

Each matcher is the unique authority within the region, such that each client registers its subscription interests with at least one matcher, before it can receive publications. To ensure that each subscription is managed by only a single matcher, the matcher whose region covers the center point of a publication or subscription is its proper *owner matcher*.

The system starts when clients contact one of the existing matchers, and specify their subscriptions. The request can be forwarded greedily based on the subscription center to the actual owner matcher. The first matcher of the system is a well-known host called the *gateway*. Once joined, clients can move to new positions and change their subscribed areas. If a subscription crosses the boundaries of matchers, then the ownership of the subscription is explicitly transferred between the old and new owner. When a publication occurs, the client first sends a request to its owner matcher, which checks for known subscribers to forward the message. If the publication covers other regions beyond the current matcher, the publication is forwarded to the neighbors continuously, until all matchers whose regions overlap with the publication are notified, so publications are guaranteed to deliver.

To adjust loading, an overloaded matcher can ask its neighbors to move their sites closer, effectively shrinking the size of its own region. Subsequent ownership transfer of subscriptions may occur to reduce the requesting matcher's load. If the overload persists, the overloaded matcher will request the gateway to promote one of the capable clients (called *candidate matcher*) as a new nearby matcher.

II. PlanetLab Experiment

Our goal is to evaluate VSO on real networks such as PlanetLab [3], and to find out which deployment approach is more practical. Our main metric is *discovery consistency*,

defined as the percentage of correctly discovered AOI neighbors, out of all potential AOI neighbors [1]. For example, if a node has 10 actual AOI neighbors but only sees 9, then the consistency is 9/10 * 100% = 90%. We first collect logs on the clients' actual positions and their observed neighbors (recorded with synchronized timestamps). After merging into a global view of the clients' authentic positions, we can reconstruct which neighbors are visible to each client at any time. Discovery consistency is then calculated between the observed and the authentic client positions.

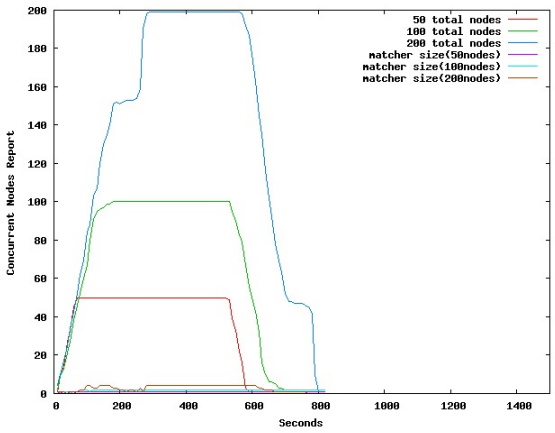

Figure 1. Concurrent Clients Reported

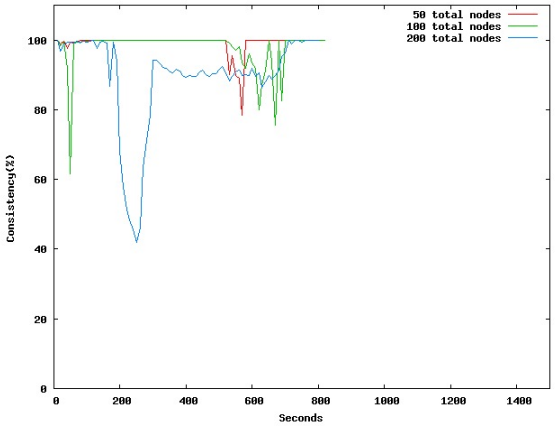

Figure 2. Discovery Consistency

At first, we randomly choose our Planetlab hosts, such that the chosen hosts may be at different sites (i.e., a physical location with several host machines). We start clients on each host at random times within a period. However, we find that the consistency is poor when the node size reaches 100. We thus choose our PlanetLab hosts from the same site (i.e., a LAN environment) to reduce latency and refine control on joining. The hosts belong to UC Santa Barbara (CPU about 2.3GHz and RAM is around 3.67GB). On each selected host, we uniformly place clients and allow at least one client be promoted as a matcher. Clients join the system continuously and move 5 steps per second for 3000 steps (i.e., about 600 seconds). Each client has a different movement path (based on a nodeID), and wait for (1 * nodeID) seconds before joining. The world dimension is 768x768 units, and each move is 3 units in distance (i.e., 15 units per second), to mimic dimensions of a regular Second Life region. We find that too many clients at the same host would cause crashes. So we set up no more than 50 clients on each host.

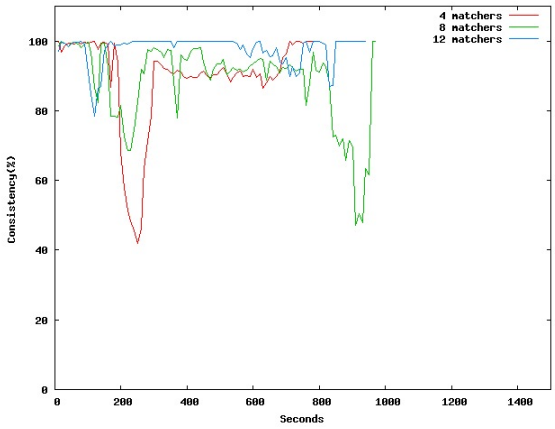

Figure 3. Effect of Matcher Sizes on Consistency

Fig. 1 and Fig. 2 track the changes in concurrent clients and consistency (respectively) as a time-series in seconds for 50 to 200 nodes (i.e., using 1 to 4 hosts). In general, the consistency is above 90% , but may drop if there are continuous join or departures (i.e., churn). The joining slope is almost 1 node/sec, but the join may delay and consistency could drop if there are matcher crashes (e.g., in the 200 node curve in Fig. 1, we see that the delay in client join correlates to matcher departures). However, the consistency tends to restore if churn stops. This shows that churn negatively impacts VSO's correctness, even though VSO can function properly if there are only node movements.

We also want to see how matcher size affects consistency. We disperse 200 nodes uniformly on each host (i.e., 50 nodes/host) and increase the number of candidate matchers. Fig. 3 shows the dependency of consistency on the number of promotable matchers. As before, the consistency drops due to node churn or matcher failure. Still, we see that increasing matcher availability stabilizes consistency.

REFERENCES

[1] S.-Y. Hu and K.-T. Chen, "Vso: Self-organizing spatial publish subscribe," in *Proc. SASO 2011*, 2011.

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

[3] A. Bavier *et al.*, "Operating system support for planetary-scale network services," in *Proc. NSDI*, 2004.