Towards a System for Controlling Client-Server Traffic in Virtual Worlds using SDN

Jason Anderson  
Clemson University  
jwa2@clemson.edu

Jim Martin  
Clemson University  
jmarty@clemson.edu

Abstract—Scaling virtual worlds in the age of cloud computing is complicated by the problem of efficiently directing client-server traffic in the face of agile and dynamic compute resources. In this proposed model, Software Defined Networking and compact encoding of avatar data in packet headers are combined to make a fast, scalable, high capacity proxy server that can hide the server infrastructure while fitting well with the IaaS paradigm of modern cloud providers.

I. INTRODUCTION

Scalability is an important factor in the design of virtual worlds. Content is typically designed to fit within the constraints of the computation, memory, storage, and network capacity available to the system. The "pie in the sky" of virtual worlds is an architecture that does not impose limits on content or the number of connected players. The current state of the art architecture in virtual worlds is the distributed client-server model, where the world state can be spread across resources beyond those available on a single compute node.

One well-researched method for partitioning the world state is a dynamic spatial partition mechanism described in [1]. With DSP, the game state is split and migrated to compute resources allocated from a pool as the load increases, and can be merged to reduce the number of partitions as the load decreases. Partitions can also be replicated to other locations for low latency and redundancy. This compartmentalization and replication of the game state to fit compute resources lends itself well to the cloud computing model, where partitions can be created, replicated, destroyed, and migrated to achieve acceptable quality of service and cost effectiveness [2].

A key challenge in a distributed server model is the efficient direction of network traffic. The server responsible for handling a particular client can change at any time, such as during a partition, merge, or migration event, or when the player crosses a partition boundary in the game world. These changes require extra round-trip traffic as the client’s traffic destination is renegotiated.

To completely hide the compute resources and game architecture from the clients, Quax et al propose a ring of proxy servers that intercept and redirect all traffic between clients and servers [3]. In this model, clients could connect to a nearby proxy to minimize latency. This would act as a more robust form of network address translation (NAT), at the cost of higher overhead due to traffic being passed through the proxy’s network stack for application-layer routing decisions.

Fig. 1. Recursive quadtree partitioning of a world map where each quadrant is identified by two bits. Logic Servers are assigned sections of the map, indicated by like-colored partitions. In this example, the location described by ‘110110’ can be mapped to the ‘red’ section by descending the tree.

A promising technology that can address these issues is Software Defined Networking, in which a central controller manages the data plane of multiple physical switches through a protocol such as OpenFlow. The control software installs flows on the switch hardware that can match data in the packet headers and perform low level manipulation at high speeds.

The primary contribution of this work is to propose the design of an efficient, scalable SDN-based proxy system that uses information about the player encoded in the transport layer header to make fast routing decisions.

II. DESIGN

A. Architecture

In this model, dynamic quadtree partitioning is utilized to progressively subdivide the virtual world [4], as shown in Fig. 1. One or more partitions are assigned to each Logic Server (LS) for computation of all objects and avatars contained within them. A Proxy Server (PS) intercepts all traffic between a client and LS as proposed in [3]. The PS is implemented as an OpenFlow-enabled switch at the edge of the compute cluster. A Region Manager (RM) node controls region partitioning, merging, migration, and allocation of compute resources as the load demands, and also serves as a controller communicating with the PS via OpenFlow. The PS is assigned a virtual IP address to which clients direct their traffic.

B. Function

The client software determines the location of the player’s avatar on the virtual world map using quadtree partitioning as described in [5], up to 10 levels deep. The resulting 20-bit number is written in the Flow Label field of each outgoing IPv6 packet.
with background full-duplex network traffic while client-server machine with 6 Gigabit Ethernet ports. Each proxy was loaded on a Pronto 3240 48-port SDN switch, the other on a dual-core Opteron industry standard switches [7]. To verify this, we built two simple packet redirecting proxy servers: one on a Pronto processing packets at line rate, comparable to the speed of a modern OpenFlow-enabled switch hardware is very fast; the latency cost of the CPU-based implementation increased linearly with the background traffic that it handled, while the SDN proxy stayed close to the manufacturer’s rating of 8 µsec.

Furthermore, this model is well suited for the movement toward combining SDN and IaaS, where datacenter operators can allow clients to control logical ‘slices’ of the network to manage their own traffic by running their own OpenFlow controller [8]. Strategic deployment of proxy servers and dynamic provisioning of compute resources could yield a system that scales to meet the demands of millions of clients.

### IV. Ongoing Work

Our lab has built a 12-node experimental SDN cluster to conduct research on application-aware networks. In the short term, we will continue to evaluate the performance and capacity of an SDN proxy server compared to more traditional solutions such as NAT and application layer proxies. Our long term aim is to use multiple instances of the SDN proxy server in a large-scale simulation to evaluate its ability to scale as the clients and servers increase.

### ACKNOWLEDGMENT

This research is supported in part by NSF Grant #1212680.

### REFERENCES


