Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 95 (2016) 201 – 208

Procedia Computer Science

Complex Adaptive Systems, Publication 6
Cihan H. Dagli, Editor in Chief
Conference Organized by Missouri University of Science and Technology
2016 - Los Angeles, CA

MMO Smart Servers Using Neural Networks for Intelligent, Client-Handling Decisions and Interactions

Ben Smith*

Graduate Student at Missouri University of Science & Technology, 1825 E. Republic Rd. Apt 5203 Springfield, MO. 65804

Abstract

This paper proposes a complex, adaptive System of Systems architecture whose goal is to intelligently and dynamically host Massive Multi-Player Online (MMO) games. Furthermore, the use of a Perceptron employed with a Sigmoidal Membership Function trained by the Least Mean Squares learning algorithm is proposed. The network intelligently manages communications and interactions between equal, subordinate, and client level servers based on a reward returned by each respective neural network. The success of these techniques allows for a fully open and interactive world with minimal server/client maximums, minimal load times, and minimal network down-time. This freedom is due to the dynamic trading of resources and/or hosted clients during execution. This paper also outlines a proof of concept application designed to demonstrate the viability of this concept. Experimental results show that the complex system of systems is able to quickly adapt to the quickly changing environment thereby proving its plausibility as an adaptive and dynamic server management system.

Peer-review under responsibility of scientific committee of Missouri University of Science and Technology

Keywords: System of Systems; Server-Client; Least Mean Squares Algorithm; Perceptron; Sigmoidal Membership Function

1. Introduction

In real world applications, Neural Networks have proven to be an effective, expedient, and resource light solution to solving complex problems. The very nature of a Neural Network is that with a little bit of training and resources up front, you can save a tremendous amount of resources and architecture design headaches in the end. A good example of this concept would be the use of a Neural Network to identify letters in an image or to track objects travelling in space with little to no assistance from a human being. These concepts, if written in raw programming

Peer-review under responsibility of scientific committee of Missouri University of Science and Technology
doi:10.1016/j.procs.2016.09.314

code, could take hundreds if not thousands of lines of logic to cover all the basis of one of the concepts. With a Neural Network, this coding effort can be reduced tremendously requiring only a bit of training logic and the implementation of the equation in the program. It was for this reason the author chose to implement this concept into a complex System-of-Systems architecture for Server-Client cross-talk on a Massive Multi-Player Online (MMO) Server Network. The goal of this proof-of-concept application is to allow an abnormally large number of users to seamlessly integrate with one another, and their completely open world, without load or lag while using minimal server resources at all times. Optimistically, this would allow the user to move throughout a vast, open world with no load times and with an unrestricted amount of players moving around the world with them, synchronously. Furthermore, it would also allow for a more dynamic and fluid maintenance platform for developers and server administrators.

The use of simulated Neural Learning in Smart Servers is not entirely unique as can be seen in examples such as [1] [2], however, the concept of Servers intelligently spawning and communicating with other servers, to the knowledge of the author, is unique as the majority of servers today suffer cross-talk limitations that, either, do not allow unrestricted, synchronous player interaction or cut the world into smaller, separate loading areas of the world. In using a Neural Network, the author aims to allow a network of servers the ability to make intelligent, trained decisions to determine the optimal method of serving all clients while constantly equalizing the load amongst all possible servers.

2. The Structure

The System-of-Systems architecture proposed includes the use of three primary server types which are used recursively and a client type. The server types are the *Host Server*, *Map Server*, and *Player Server*. Each server entity has a responsibility to maintain a specific aspect of the network, but may overlap in their coverage.

2.1. Hosting Server

The *Hosting Server* is the primary point of contact for all *Subordinate Servers*. It is important to note that Hosting Servers may nest themselves in order to provide better coverage over clients and fellow servers more effectively. For this reason, the Hosting Server may, itself, be an “equal subordinate” of a different Hosting Server, but as it doesn’t function with a Neural Network, nor is it submissive to any other server than one of its own pedigree, it is not referred to as a Subordinate throughout the remainder of this paper.

The responsibilities of the Hosting Server include:

1. Listen for New Client connections
    a. Poll Subordinate Servers for Hosting Bids based on individual clients
    b. Evaluate Hosting Bids to determine optimal client ownership
    c. Serve New Clients with Map and Player Servers
2. Monitor and Serve Subordinate Servers
    a. Notify subordinates of proximity with other, like subordinates
    b. Negotiate “Transition Requests” to trade clients between subordinates
    c. Spool new subordinates when no subordinates are available
    d. Spool new Host servers when unable to maintain its own subordinates

2.2. Subordinate Servers

Subordinate servers may be any server controlled by another server. The primary Subordinate relationship is between the Hosting Server and its Map and Player subordinates. The fine print of a subordinate is that it evaluates its client-server ownership relationship by running their distances through a simple Neural Network called a *Perceptron*, described in detail in section 4, is used to produce a single value or reward. This reward is then used to determine the ownership of that client. Evaluation takes place for three reasons, *Creation*, *Optimization*, and *Transition*.

Creation is the act of finding an initial Map and Player to own a client when it first connects to the network. The request is sent from the Hosting Server to evaluate a new client, each of the Hosting Server’s subordinates evaluate said client and return a Bid which is an accurate representation of the Subordinates’’