# WEEBCHAT

A dynamically distributed telnet chat server

First of all, **Dynamic**, the word itself means always active or changing. Second, **Distributed,** which is one, if not the best way to write a future ready server. Put them together and you have a server that can scale horizontally in *N* number of servers.

Why horizontal? Well, this helps you accommodate more users and can save you money because the marginal cost of adding (*vertical scaling*) one more core or a hard drive that does a few more I/O operations per second grows exponentially. In the long run, the cost of adding one more node to the system becomes far cheaper than the cost of additional hardware.

With those in mind, I decided to write a load balancer called “Lobby” to distribute workloads across multiple servers I call “School”. Initially, the lobby does not know any school. A school introduces itself to the lobby when it comes online thus, making the number of schools dynamic.

A school is a server that can host a limited number of chatrooms depending on admin configuration. Each time a user creates a chatroom the lobby asks each active school for available slots, the first to respond gets to host the new chatroom. Now when a user tries to join a chatroom the lobby should pass the clients connection to the specific school hosting the room freeing itself from the duty of handling the client.

One of the things to consider when developing a server is the client. A lot of the server’s architecture must be complimented by the client’s design. Given this task that restrains me to only use **Telnet** as a client, I encountered my first major problem: “How do I properly distribute load between schools when I cannot pass the client’s connection from the lobby to a school?”

The usual workflow of a client for a distributed system is:

* Client connects to load balancer.
* Load balancer selects an appropriate server and sends the server info to the client.
* Client receives server info and disconnects from the load balancer.
* Client reconnects directly to the given server.

This allows load to be properly distributed. Problem is, I do not have the liberty of coding my own client. So I must find a way to keep the architecture and comply with the restriction of only using telnet.

With the given constraint I have decided to sacrifice proper load balancer workflow by keeping all connections in the lobby while passing the work to the schools.