Eero Kelly
Alex Hoover

Group 17

Our system implements both load-balancing and security. We will split our report in to two parts, each detailing half of the system.

Load Balancing:

Our load balancing system has three major systems functioning. They are connecting to a server, adding a new server to the pool, and distributing messages.

Connecting to a Server:

We needed our system to connect users to one of however many available servers. To do this, we have a “transparent” proxy. When a client first navigates to a page, it briefly connects to the websocket of the proxy. While it is doing this, it is also pulling all of the page info from the proxy. This only happens once, as the functionality of the page works from javascript. During this brief connection, the client is asking the proxy to kindly direct it towards a server. The client then makes a connection to this server and will hold this connection through it’s lifetime. By organizing our proxy this way, there is only as much traffic to the proxy as there are clients connecting to the system. All of the messaging is done on the backend servers. Every time our proxy directs a user towards a server, it updates the current users on that server, and it uses that number to choose where to send users in the future. The proxy will always pick the least occupied server. As well as this, every time a user disconnects from a server, the server is able to notify the proxy that it has lost a user, and the proxy can continue to hold a correct count of connected users.

Adding a New Server to the Pool:

We needed our system to be able to have a variable number of servers supporting it. To do this, we have a negotiation process that happens every time a server wants to add itself to the pool. The server is preconfigured to know where the proxy is, something we thought was

reasonable. The server must be configured with it's own websocket address and TCP socket address. We will get to more of that later. When a server wants to connect, it packs both of its addresses into a json object and sends that to the proxy. The proxy will register that server's address as somewhere it can send users, and it will hold onto the server's TCP address to give to other servers later. The proxy will respond to this object with a list of the TCP addresses of all other known servers. The server makes a TCP connection to each of the servers that the proxy tells it about, and it stores these servers for later. This TCP connection contains the location of the server making the connection, so that each of the other servers is now aware of the server being added to the pool. After this initial connection period, the server goes into listen mode. In this listen mode the server can get add requests from other servers, which will allow this server to be aware of new servers as they connect, and it can receive relay instructions from other servers, which we will talk about next. To better demonstrate how this system works we'll use letters.

Connect A

A asks the proxy for servers and the server responds with nothing.

A alerts the proxy of itself and then listens for any new connections

Connect B

B asks the proxy for servers and the server responds with A

B connects to A and tells A to connect to itself

B alerts the proxy of itself and then listens for any new connections

Connect C

C asks the server for servers and the server responds with A, B

C connects to A and B and both tell to connect to themselves

C alerts the proxy of itself and then listens for any new connections

After this is done, A, B, and C are all connected to each other, the proxy knows about each, and all are listening in case a new server connects.

Distributing Messages:

Eero Kelly
Alex Hoover

The servers in this TCP “mesh” network are able to distribute messages across all servers to all clients. When a server receives a message from a client, it has the ability to push that message to all connected clients over websockets. As well as doing this, the server will push its message to all other servers that are in listen mode. When a server in listen mode, it propagates this message to all clients connected to it in the same way as it would if the message came from a client connected to that server. This way, all messages reach all connected clients, no matter the server they are on.

Security:

We do two main things to keep our system secure. They are monitoring for DDOS attacks, and encrypting traffic between servers.

Monitoring for DDoS Attacks:

In order to watch our system for DDos attacks we monitor how quickly we let users send messages so fast. If a user is sending messages too close together for them to be meaningful and not spam, we disconnect them from our servers. This way we aren’t forwarding useless traffic along our network, slowing down servers and bothering clients.

AES Encrypting Traffic Between Servers:

Every message we send between each of our servers is encrypted. By doing this we keep messages from being intercepted by third parties in things like man-in-the-middle attacks. Without breaking our encryption nobody can see what is being passed between servers. Everything sent between the servers and the proxy is also encrypted. This means that if someone doesn’t have our keys, they can’t join our server network, so they won’t be able to hijack clients.