Connect - A study in distributed chat applications

Irtiza Ahmed Akhtar, Zainab Ghadiyali

November 4, 2012

1 Abstract

Distributed Systems are widely popular and increasingly important given the yearly increase in number of internet users and scalable systems. In this paper, we describe the design, implementation and analysis of connect, a distributed system built on top of EC2 service.

2 Introduction

The increasing ubiquity of distributed and chat systems has paved way for an increasing number of advances in both. Deploying a chat system over a distrbuted system can be done in several ways. Internet Relay Chat (IRC) is one of the oldest chat systems found on the internet. The backbone structure is a spanning tree of servers and users connect to one of these servers. The message then trickles down to all servers across the spanning tree. The protocol is simple and widely studied. Another type of chat dsystem is web based with each chat application differing in interface and functionality. AOL Instant Messenger, Facebook Messenger and Google Chat are some examples dominating this space.

3 Background

3.1 Distribtued System

3.2 Chat System

As mentioned in 2, chat systems are of two main types.

3.2.1 IRC

Internet Relay Chat(IRC[2]) was a rather widely used chat system. Each IRC network comprises of a spanning tree structure of servers who typically listen

on port 6667. A user idetnfies himself within the IRC-network through a designation username. The username is hence unique within a particular network. IRC networks comprise of several channels which form meeting and discussion avenues. Since channels are so critical to IRC functioning, it is imperative that these channels be unique and consistent throughout the network. An IRC operator works as an admin to override actions of remove users from network.

3.2.2 Web-chat

A web chat system uses a browser as a user interface and HTTP as underlying application protocol. Typically, a web chat service has a login system to authenticate users and a session id to maintain consistency and ensure that each user gets the chat message broadcasted.

3.3 Distributed Chat System

4 Methodology

4.1 Design

In our chat system we use the TCP protocol for communication between the chat server and client while communication between the chat client and location server is via HTTP and TCP. TCP is a connection oriented reliable protocol and states related to each TCP connection need to be maintained at both client and server side. UDP could be used for transfer of chat messages, however we choose TCP over UDP due to TCP's reliability and the fact that packet loss may occur. The chat application is to be implemented using a clientserver architecture:

1. Clients connect to a chat server who in turn is responsible for broadcasting these messages.

2. Since the chat server has the key role of transmitting and storing messages, it is important to automate the chat server as much as possible in order to improve operational performance.

3. The network must be secure to protect messages irrespetive of whether the clients connect through an insecure/public network.

4. The client must be able to use the chat service irrespective of changes to the chat server and internal application network.

5. The clients must be able to connect to the chat server irrespective of firewalls.

6. The client must have easy access to the chat application through all major browsers

4.2 Implementation

5 Results

5.1 Design

5.2 Operational Efficiency

5.3

6 Discussion

7 Conclusion

References

[1] Lamport, L., *LaTeX : A Documentation Preparation System User's Guide and Reference Manual*, Addison-Wesley Pub Co., 2nd edition, August 1994.

[2] J. Oikarinen and D. Reed, Internet Relay Chat Protocol RFC 1495, 1993.