Final Project Report - P2P Music Sharing

by Devante Wilson

Submission date: 03-Dec-2018 07:57PM (UTC-0500)
Submission ID: 1049977036
File name: Distributed_Systems_-_Final_Project_Report.docx (114.39K)
Word count: 1758
Character count: 9624

SOFE 4790: DISTRIBUTED SYSTEMS

P2P Distributed Music Sharing Application – Final Project Report

DECEMBER 4, 2018

UOIT
GROUP 8

P2P Music Sharing

Devante Wilson, Emin Avanessian-zadeh, Sikandar Shahbaz, Ebrahim Merchant

Summary

Our team was given the task of engaging in systems research and development (R&D) to investigate novel ideas for distributed systems and applications/services and tools. We built on the concepts presented in class to investigate a particular topic further and propose a novel solution to a challenging problem. The proposed project in question that our team chose was a peer-to-peer distributed music sharing application. The application allows users connected via a Local Area Network to search for and share music files with each other.

1. Introduction

For our distributed systems term project, our team decided it would be best to experiment with a potential peer-to-peer music sharing application. The reason we are so excited to work on this innovative product is because it is a highly novel idea and would add an exhilarating twist to the standard music playing applications that are in the market. The most exciting aspect of having a peer-to-peer application is that the data is distributed between the clients and not stored in a central server, thus eliminating many overhead expenses and saving bandwidth. The entire application is developed in Java so that the Java Virtual Machine (JVM) can be utilized for simple interoperability for use on various platforms.

A complex project such as this requires for the work to be split up equally to ensure that one team-member is not burdened with more work that they can handle. We decided to designate a group leader who is responsible for dictating which steps are required for each stage of the project. Two programmers worked on developing code that runs efficiently. Finally, there was one member designated for system integration who combined all of the different pieces of the application and finding reasonable fixes to any major errors/bugs. Combining all of these roles enabled our team to work like a fully-functional machine to develop this product in the allotted time we were given.

2. Background and Related Work

There are a number of other popular applications used to share and stream music. Most notably, there are Soundcloud and Spotify. These two application are virtually the same but use different data storage systems. Soundcloud uses a blend of different Amazon storage services to store its enormous, ever-increasing data. The actual audio files are stored inside of Amazon S3 - where the music is deployed from. But to ensure that the audio files are available to users at any time, audio files are copied onto Amazon Glacier - used for data archiving and backup. One potential problem that Soundcloud may encounter in the future is that it may have problems upscaling their data storage as their company increases in size.

The other major music distribution platform is Spotify. At one point, Spotify used their own storage infrastructure but eventually came to the realization that their 75 million users could not be supported on a self-relying data storage system. This is when they decided to issue a partnership with Google in order to use their Cloud Platform to store data. While that system is sufficient right now, Spotify pays a very high fee to Google for their cloud storage services and that takes a huge chunk from their yearly revenue. This is where a peer-to-peer application shines; by having a decentralized application upscaling will prove to be easier in terms of data storage, with no overhead cost. These two features help our distributed music system stand out from the other music sharing platforms.

Napster (what used to be a popular music sharing platform) was the first of its kind to use P2P file sharing for digital audio.

Torrent clients and sites allow peers to download a file by downloading chunks of the file from different peers who have the complete file (called seeders) or from peers who have a portion of the file. Leechers are the peers who are downloading the file. A tracker is a server that maintains an accurate account of active seeders and peers.

3. Peer-to-Peer Music Sharing

3.1 Assumptions

- The Local Area Network connecting the peers is secure (no attempts at a malicious attack)
- The Local Area Network connecting the peers is reliable and supports multicasting/broadcasting
- A peer will search for music file in the form <song name - artist>
- A user has a folder in the source folder named “Music” where music files will be stored

3.2 Architecture

The application uses a peer-to-peer architecture to facilitate communication. Peer-to-peer distributed systems work differently than systems with centralized servers. The audio files are stored within a music folder on a given peer’s file system. Any peer can search for a song on the network; if the song is not present on their local file system in the specified folder, they will broadcast a UDP request to all other nodes on the network.

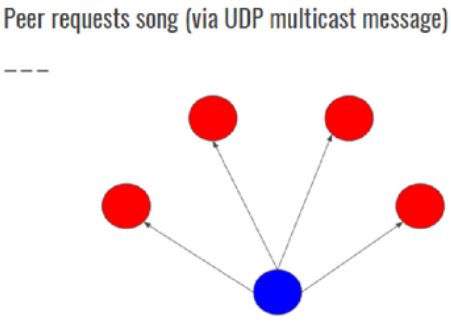

Figure 1

If a peer has a song that another peer on the network is searching for, the peer will let the other know that the song is available for download via a directed UDP message (to the requestor's IP directly). A TCP server socket is created for sending the mp3 file.

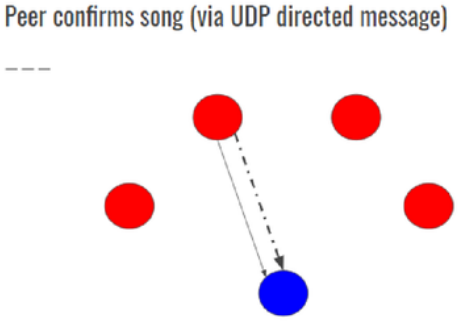

Figure 2

The requestor then receives this confirmation message and opens a TCP client socket to receive the mp3 file from the sender. The file will be saved locally to the peer's file system.

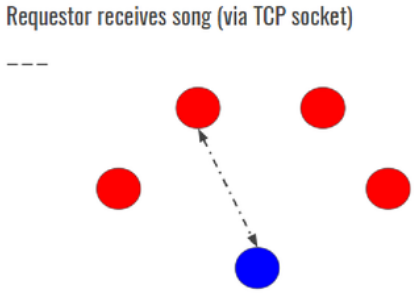

Figure 3

3.3 Implementation

As mentioned in the previous architecture section, the application uses the User Datagram Protocol (UDP) for issuing a broadcast message to all other nodes and for receiving the

confirmation message. UDP was chosen for these functions as it is a connectionless protocol. In other words, peers need not perform any connection setup nor do they need to retransmit lost packets - sending data incurs less delay, making this portion of communication more responsive. We created a timeout period of five seconds when trying to find a peer who has the requested song.

The sending of music files is facilitated with the use of the Transmission Control Protocol (TCP), which is connection-oriented. With a reliable protocol such as TCP, we can ensure that the transferred music file is delivered with data integrity (no data loss, or data being delivered in ambiguous order), which would pose a detrimental issue for a music sharing system, otherwise.

The application consists of a single class that houses all the methods needed to fulfill the application's functionalities. The first method *serveUserRequests()* is for prompting the user and taking their input for searching for a song. The method will first search for the song in their local file system in the specified music folder via a method *getAllMusic()*. If the song was not found, another method *searchForSongOnNetwork(String message, String ipAddress, int port)* will be invoked to begin a search on the network. Meanwhile, each peer has a thread running in the background (daemon process) that is handling incoming song requests - the method invoked by the daemon is *receiveUDPMessage(String ip, int port)*.

3.4 Scalability

The amount of connected users that the given LAN can handle is directly proportional in determining the scalability of the application. The application is only intended for use on a Local Area Network such as a university/college student residence or in a home. Therefore, factors that need to be pondered for a system using servers are not factors that need to be discussed for our peer-to-peer system; overhead and transmission times are topics more suited for systems that utilize centralized servers.

3.5 Challenges and Solutions

The first challenge we encountered during the development of this product was to determine the best method to use to begin developing a decentralized music sharing application. Much brainstorming and planning was required before we developed a proper plan of action. All of the members provided various ideas; through combining many of them together, were able to create a master plan. Once all this was decided, we were able to get started on the implementation phase.

During the testing of the searching feature, we encountered much friction with a certain Local Area Network or router not allowing broadcasting or multicasting on the network. Additionally, network firewalls also prevented receiving a broadcast message sometimes. We resolved these by trial and error testing by using different routers and disabling/enabling the firewall on the computers used for testing.

4. Evaluation and Results

In terms of performance, the throughput of the network, music file size, and limitations of the LAN all play a role in determining how well the application will run. Network speed will determine how fast communication takes place between peers. Though, since most LANs offer decent data throughput, this is not an area of large concern. File sizes also determine how quickly a peer will receive their requested song. Since the application only supports the sending of single music files and not albums or larger compilations, file size is not of immediate concern. The amount of connected users that the given LAN can handle is directly proportional in determining the scalability of the application.

Security-wise, since the application is intended for use over a Local Area Network, less security concerns are present. Connectivity to the Internet presents more opportunity for malicious attacks.

The only privacy concern is that anyone running the application can access the music files of other peers; music files are not a form of sensitive data, thus there is no need for concern in this area.

5. Conclusion and Future Work

By the end of this group project, all of our team members received beneficial experience with distributed systems concepts and how to apply them. We gained new knowledge with developing a peer-to-peer system as none of our team members were familiar with its intricacies beforehand. Furthermore, we had the opportunity to utilize various protocols and mechanisms related to distributed systems such as User Datagram Protocol (UDP), Transmission Control Protocol (TCP), multicast and broadcast, socket communication, IP addresses, threading, buffers and file transfer via byte streams, and resource sharing.

If our team were to pursue the improvement of the application we have developed, we would implement certain mechanisms to streamline the communication process. A possible improvement could be to check metadata to ensure correct music file is found/transferred. We could implement some high-level security checks to maintain safe communication.

References

1. AWS Case Study: SoundCloud. (n.d.). Retrieved from https://aws.amazon.com/solutions/case-studies/soundcloud/
2. Leygues, G. (2016, February 23). Spotify chooses Google Cloud Platform to power data infrastructure | Google Cloud Blog. Retrieved from https://cloud.google.com/blog/products/gcp/spotify-chooses-google-cloud-platform-to-power-data-infrastructure

Final Project Report - P2P Music Sharing

ORIGINALITY REPORT

3%	3%	1%	2%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to University of Hong Kong Student Paper	1%
2	Submitted to University College London Student Paper	1%
3	tel.archives-ouvertes.fr Internet Source	1%
4	cclab.kw.ac.kr Internet Source	1%
5	browserzone.com Internet Source	<1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	Off		