SimpleMessenger with Apache Thrift

Animesh Baranawal

August 5, 2019

1 Introduction

The objective is to create a chat messenger service that supports point to point communication, broadcasts and file sharing among users. The messenger service uses Apache Thrift[1] for supporting communication across different users. Apache Thrift was chosen because of its cross-language support.

2 Methods

The chat messenger is models as a central server node acting as a relay between different users, which are modeled as client nodes. Since Apache Thrift does not support any direct 2-way communication[2], each client node runs a mini, light weight server on its side. Java Swing is used as the supporting user-interface framework, and the backend services are implemented in Java respectively.

2.1 Basic Structures

- **UserDefinition**: string uniqueID, string ip_addr, i32 port
- **GroupDefinition**: string groupID, list<string> members
- **Message**: i64 timestamp, string msgString, string toID, string fromID, bool multicast
- **FileDefinition**: binary data, string fileID, i32 size
- **FileChunk**: binary data, i32 size, i32 offset

2.2 Backend Services

- **ServerService**: It is hosted by central server for message communication. It supports the following RPC methods: *join*, *connectTo*, *sendMessage*, *joinGroup*, *leaveGroup*. It also spawns dispatcher threads which empties the server message queue.
- **FileTransferService**: It is hosted by central server for file communication. it supports the following RPC methods: *startUpload*, *uploadChunk*, *endUpload*, *startDownload*, *downloadChunk*.
- **ClientService**: It is hosted by client for communication from server. It supports the following RPC methods: *receive*. It also spawns a receiver thread which empties the client message queue.

2.3 Implementation

- Messaging: The client node calls *sendMessage* on ServerService. The server on receiving the message adds it to its message queue. In case the message is to be multicasted to a group, the message is replicated for different user members and then added to the queue. The dispatcher thread takes a message from the server message queue and calls *receive* on the target client node. The ClientService on receiving the message adds it to its message queue. The client receiver thread takes a message from the message queue and passes it to the respective UI widget for displaying.

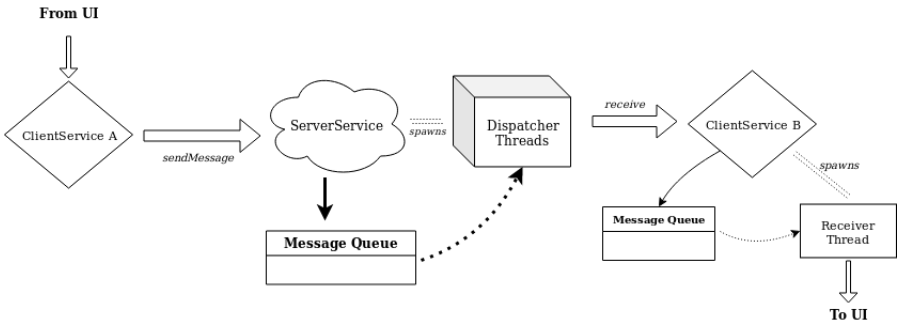

Figure 1: Visual description of messaging workflow

- FileTransfer: The file transfer workflow is similar to the workflow used for the same in WhatsApp. The client starts uploading by calling *startUpload* on FileTransferService and then uploads the file in chunks of 2048 bytes using *uploadChunk*, followed by an *endUpload* call. This stores the file on the server storage. The client then sends a message to recipient node with the file name. The recipient client calls *startDownload* on FileTransferService followed by *downloadChunk* to downloads the file in chunks.

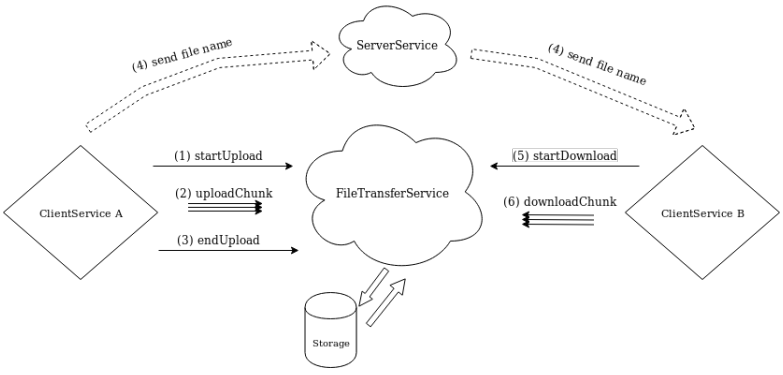

Figure 2: Visual description of file transfer workflow

3 Results

3.1 Validation

`java.util.logging` was used for creating logs of ClientService and ServerService. Every message sent and received by the client is logged along with the sender unique identifier, receiver unique identifier, sent time-stamp and received time-stamp. Latency is measured by calculating the difference between the sent timestamp and received timestamp. Time-stamps for when file upload/download request was sent and when the upload/downloads got completed are also logged. File transfer time is calculated as the difference of these two timestamps.

For scenario 1 and 2, simulations were done on a single machine with the server and clients using different port numbers for running the services. The average of maximum latency observed across simulations is used for plotting results. For scenario 3, 2 VMs are set up for simulating clients with original machine simulating the server.

3.1.1 Scenario 1: P2P communication

N different users communicate point to point with each other in this scenario. To validate this scenario, message each sent message in the sending client log is checked in the log of the recipient client node. The simulation was run for N=2,5,10,15,20,25 with the server running 1 dispatcher thread. Figure 3 shows the latency observed. This scenario was simulated 10 times.

3.1.2 Scenario 2: Multicast communication

To simulate this scenario, N different users first join a chatroom. Every message sent to the chatroom will be multicasted to all the other user members. To validate this scenario, the list of messages received by a client node was compared to the list of messages sent by all the other client nodes. The scenario was simulated for N=2,5,10,15 with the server running T=1,2,3,5,7 dispatcher threads. Figure 4 shows the latency observed. This scenario was simulated 10 times.

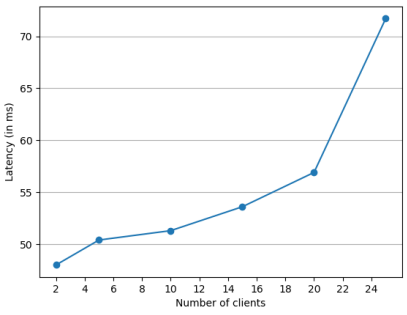

Figure 3: P2P Latency

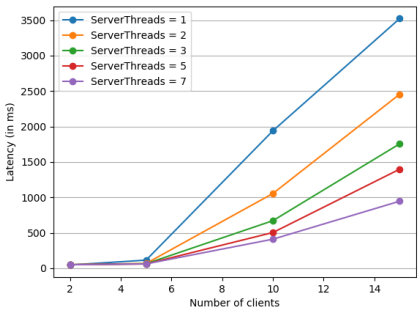

Figure 4: Multicast Latency

3.1.3 Scenario 3: File Transfer between 2 clients

2 clients simulated by two VMs join the server and transfer file of sizes S = 1,10,100,1000,10000 Kb between them respectively. The scenario was simulated 5 times. To validate, `diff` was taken between the sent file and received file to find differences if any. Figure 5 shows the average file transfer time across the simulations.

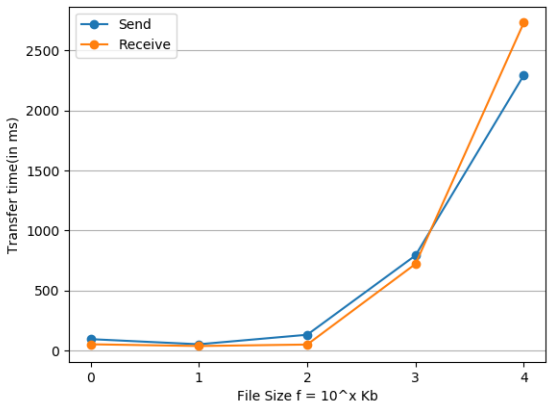

Figure 5: File transfer time for different file sizes

4 Conclusion

We developed a simple messenger service using Apache thrift supporting point to point and multicast communication with file sharing. However, the work can be easily extended.

An extension of the work could be to incorporate fault-tolerance in the application which has not been addressed in either the implementation or scenario simulation. Validation can also be extended to check if the order of messages sent at the sender client node and received at the recipient client node match. Recording the chat conversations to restore the history is yet another extension.

References

[1] Apache Thrift: `https://thrift.apache.org/`

[2] How to send a message from Thrift server to client?

[3] So you want to send a message using Apache Thrift?

[4] Medium: WhatsApp-Engineering

[5] File Transport between Server/Client using Apache Thrift