COMP90015 Distributed System

Project 1

Bit-Box

Group ZWPL

Group Members:

Jialiang Zhu JialiangZ jialiangz@student.unimelb.edu.au

Yanzi Wang YanziW yanziw@student.unimelb.edu.au

Yunqiang Pu YunqiangP yunqiangp@student.unimelb.edu.au

Miaoze Li MiaozeL miaozel@student.unimelb.edu.au

1. Introduction

The goal of the project was to create a distributed system that could enable the users to share and synchronize files with each other. To be more specific, it is an unstructured P2P network. The design language for the project is Java 1.8. Before transferring data, the connection between peers needs to be established through TCP connections and the messages will be in JSON format. Two-way handshakes protocols are implemented in order to establish the connection. After the connection being established, the user can create or modify files remotely. The biggest challenge we met during the building of the systems was the implementation of the protocols since it is difficult to test the codes half way and we met a lot of bugs when we were trying to test the system. Eventually, we built the P2P system that could enable the file creation and transferring between peers using the specified protocols.

2. Fault detection/tolerance/recovery

In this section, issues regarding the fault that could potentially happen would be discussed. The fault sources will be addressed first, followed by the detection, tolerance and recovery method. Suggestions regarding the revision and improvement of the protocol will also be provided in this section.

2.1 Fault sources:

The event that could cause the most significant impact on the performance of the system is the disconnection of a peer during the synchronization phase. The reason is that the system will not try to re-establish the connection to finish the synchronization. Therefore, the file with the same name will be different for the two peers. For the next time the connection is established, the file which shares the same name will not be automatically updated based on the last modified time. As a result, this could cause an inconsistency in the system. This could cause a major flaw in our distributed system where the files should be synchronized throughout the system.

2.2 Fault Detection, Tolerance & Recovery:

The fault detection is imperfect for this system. As described in the detailed protocol, there lacks a confirmation message between peers after the data transfer. As a result, if connection loss occurs during the transferring phase, the system will not be able to identify the loss and request a re-transmit. The consequence will be an inconsistency in the distributed system and incomplete file at a local level. Currently, the fault tolerance is the treated method if the packet loss happens and no recovery methods are implemented to address the issue.

2.3 Suggested Revision to the protocol:

The suggestion we proposed that could potentially improve the fault detection and recovery is to add a time-out in the protocol. After several packet transmission without the ACK message from the peer in a given period of time, the system should identify the connection loss and try to reconnect the peer to start over the transmission. Therefore, the consequence brought by the connection loss will be revised properly.

3. Scalability

3.1 problems for scalability

As for the scalability challenge it can be concluded as three part as follow:

1. The system automatically synchronizes all files every 60 seconds. It may not be suitable for a bigger system because that there might be more than one modification instruction to the same file from the different places during the time block, therefore there may be some confusion about this file, because there is not any function to choose which one of the modification will be used according to the last modified time.

2. In this procedure we did not achieve the function that the modification instruction will be apply to all the peers in the connection. There might be a problem that the peer a pass a modification instruction about file X to peer b and in a short time (less than 60 second) the peer read the file X will get the wrong data because the latency in updating. These problems can never be acceptable in some cases for example the banking files which have large scale of files and change frequently.

3. Obviously according to the procedure, there is the maximum connection limit of 10, when the incoming connection request is more than 10 it will be refused. It is also a problem that in an extreme big procedure that there are hundreds or even thousands of potential peers there might be a problem that over ten peers are requesting to make connection, once this happens, I will be returned refused in some request to connections.

3.2 suggest revision

To overcome these problems above there are also three potential revisions to the protocols to address these problems.

1. Increase the limit of maximum connection of 10 to the suitable number which fit the volume of the potential peers to avoid the congestion of modification instructions toward the same files.

2. In the same system there are always some files modified a lot and some files modified little, it is an efficient way to make some specific rules for the modification to the files those accessed frequently like reduce the waiting time for updating.

3. Add one more function to choose the modification instructions based on the latest modified time when there is more than one modification instruction to the same file.

4. Security

According to the subject, the project implemented a file-sharing function. The linking is built after twice handshake. However, in this situation, there are still some security problems that great impact on users' information. Here are four problems:

1. The most serious problem is eavesdropping. The files which are shared in this system are easy to be eavesdropped, because the files are not encrypted. In addition, an unauthorized user is able to access to data, and the system cannot stop it. Building a firewall is a effective way to solve this problem, due to the fire wall can protect the trusted users from untrusted external users.

2. The data lost is another problem in this system. Assuming that the system is attacked, and the linking have to be interrupted. On the one hand, the interruption can stop others accessing to the data. On the other hand, the trusted cannot get the data, which may cause data lost and data error. To solve this problem, this system can use Transmission Control Protocol (TCP). In order to ensure that no data lost, TCP gives each packet a serial number, and the number also ensures the sequential reception of the data transmitted to other peer.

3. Physical security problem should also be concerned in this project. For example, a user delete an important file which is shared by mistake, which may cause many peers can not get the file. Therefore, the function of data backup and recovery is necessary. Besides, the important data, files or historical records in the computer, whether for business users or individual users, are crucial. If anyone accidentally lose it, it will cause immeasurable loss. In case of this situation, data backup can solve most of physical security problem.

4. This system is a peer-to-peer network, which can be used by some hackers. For example, hacker can pose as a normal peer and send wrong instruction to other peers, which can shut down the whole network. Thus, this system does need the function of authentication. People must sign in before send or access data.

5. Other Distributed System Challenges

Except the scalability and security issues discussed above, there are also some other challenges that distributed systems may face need to be concerned. Such as concurrency and heterogeneity. Sometimes there may also be some problem appear toward data consistency. And single-point failure is also possible to occur.

As this system is in peer-to-peer model, the data are distributivity stored in different peers. There are some advantages of this model with comparing to a client-server one. For example, it has less chance to get bottle-necked, and each peer can be both client or server for providing and request services, which makes the data retrieving faster. However, it also brings some drawbacks. A peer-to-peer system need data redundancy to make it more reliable. And maintaining it can be bandwidth expensive. As each time a file is modified, the other replications in other peers with the same name need to be updated. This cause a burden to the storage resource. Besides, with regarding to the unimplemented function, once a peer lost the connection, the system may not try to re-connect with it. This could also cause a problem for maintaining the data consistency. If a peer requests the file from the peer with the old version one, this will cause a further confusion in future file sharing. As for the single-point failure, if a file is hold by only one peer, once the data in that peer is damaged, that part of data meant to be lost and unavailable for any other peers.

For the improvement towards the problem, future enhance development of the system could be the solution. To be specific, realizing the system function of re-connection after connection lost could help maintain the data consistency. And a peer-to-peer and client-to-server hybrid could to some extent improve the overall performance. It can combine the advantages of the both models, which help achieves a safer, faster and cheaper cost system.