From Centralization to Distribution:

A Comparison of File Sharing Protocols

Xu Wang, Teng Long and Alan Sussman

Department of Computer Science, University of Maryland, College Park, MD, 20742

August, 2015

Abstract

File sharing service is a fundamental module for many modern collaboration and communication services. The traditional server/client repository model of file sharing has been widely used in many systems, but when it comes to fully decentralized systems, a decentralized file sharing service is also greatly needed. In this paper, we have selected three different protocols, SSH File Transfer Protocol (SFTP), Trivial File Transfer Protocol (TFTP) and BitTorrent (BT), implemented an experimental file sharing service for each of them, and evaluated their performance and scalability over different shared file size and number of sharing participants. The results show that BT has shortest time cost with the best scalability, while the TFTP is the slowest in downloading and uploading files whereas the SFTP speed is in between.

Introduction

File sharing service is an essential module for many collaboration and communication services. Traditionally, the server/client model is widely used when there exists a

centralized server that can be accessed by all sharing parties, and light-weight protocols such as FTP and similar ones can be easily adopted. However, with high bandwidth internet available to more and more ordinary users, some systems may require a broader sharing of files with larger size. The traditional server may become a bottleneck in this case. Besides, there are also many decentralized systems that do not have a centralized server but need to implement their file sharing services. For these reasons, we need to investigate alternative protocols for such file sharing services.

File sharing protocols play a key role in communication network and govern the computers' communication and interaction when the computers are connected in the network [1] to deliver data between computers or between computers and terminals [2]. A communication protocol defines the rules allowing blocks of data to communicate from one node in a network to another node. Protocols are normally defined in a layered manner and provide all parts of the services specified by a layer of the OSI (Open Systems Interconnection) reference model [3] and are usually implemented by writing a number of programs.

In the past, many protocols had been developed such as Transmission Control Protocol (TCP), Hypertext Transfer Protocol (HTTP), and so on. Decided by the underlying network protocol used, there are UDP based ones such as File Service Protocol, Trivial File Transfer Protocol (TFTP), and TCP based ones like FTAM, HTTP and Secure copy. Also judged by the topology of file transportation parties, the most traditional way is the client/server model, where there is one server which is the central location and multiple clients can exist to share and access network resources. Shared data is in one

location so it is easy to back up information. A lot of protocols implement it including FTP and SFTP, HTTP etc. P2P is another model, which also consists of multiple computers, but there is no central location for authenticating users, files or resources. Instead, each client functions not only as a client and but also as a server, and there is no central security. The popular implementations include Bittorrent, Ares and OpenNap, etc.

In these protocols, FTP is the most typical server client protocol over TCP-based networks and is built on a client-server architecture. FTP protocol has a lot of advantages. It allows to add items to a queue to be uploaded or downloaded. Compared to HTTP, there is no size limitation on a single transfer. It is also able to resume a transfer after reconnection. However, FTP also has some disadvantages, for example, it does not encrypt user name and password, so it could make user account security an issue. Direct connect protocol, on the other hand, differs from FTP and it is a P2P protocol. It is a text-based computer protocol, where the information are sent in clear text.

All these protocols have the same basic functionalities to facilitate communicating and sharing computer resources. However, each of them has different purposes and accomplishes different tasks. They also have their own advantages and restrictions. The goal of this paper is to analyze and compare different protocols by evaluating their correctness and performance.

We selected three representatives of communication protocols, SFTP, TFTP, and BT, then evaluated their performance from functional requirements, quality requirements and design

constraints. The SFTP and TFTP protocols are based on server-client model, however, the BT protocol represents the decentralized model which may better fit the needs of today's decentralized systems. The functional requirement includes system behavior and data format to be transferred; the quality requirement refers to the communication reliability and performance while the design constraint involves the communication environment and interfaces.

Experimental Setting

Stability and time cost are the key factors used for evaluating the performance of file sharing protocols. In this work, our first step is to set up the connection for transferring files between client and server by using the three different protocols, SFTP, TFTP, and BT, respectively.

SFTP is based on SSH protocol to provide secure access to shell accounts on remote servers for exchanging files over a TCP/IP (Internet Protocol) network. With SFTP protocol, all data sent between client and server is encrypted using an agreed upon encryption cipher. SFTP sessions can also be further protected through the use of public and private keys, which offer an alternative form of authentication known as public key authentication [4].

As the name suggests, TFTP is a simple and lock-step protocol for transferring files between machines of different networks by implementing UDP (User Datagram Protocol) and it is designed to be small and easy to implement. TFTP can't list directories and

currently has no provisions for user authentication, therefore it not only lacks most of the features of SFTP but also lacks the security offered by SFTP. The only thing it can do is to read files from and write files to a remote server. In common with other internet protocols, it passes 8 bit bytes of data [5].

The third protocol we used in this work is BT which is used to distribute large amounts of data over the internet [6] for the practice of peer-to-peer file sharing. Although SFTP and TFTP use different protocols, both of them use a client-to-server connection. The client of SFTP and TFTP connects directly to a server to download and upload files via the protocol and the download speed is limited by both the speed of client's connection and the speed on the server's connection. In contrast, there is no server in BT to host the file where all the clients can download the parts of the file from other users who already received that part. To get BT connection, three fundamental parts are required, torrent file, tracker and computer. The torrent file contains metadata about files or folders to be distributed and the tracker is a server to regulate the communication between each computer, and the computers share the resources.

We evaluated the performance of protocols in different environment by using two computers with different operating systems. The Mac OS X system computer was used for a client, and the Linux Mint acts as a remote server. The experiment is in Verizon 15 M upload and download internet connection.

We used "***org.apache.commons.net*** " library to implement the client side of TFTP and "***com.jcraft.jsch.Channel"*** library to implement the client of SFTP. The ***"org.apache.commons.net"*** library provides fundamental protocol access to make the

global functionality of a protocol accessible. It has TFTP packet classes and the TFTP packet send and receive methods so that we are able to construct our custom implementations. By using JSch (Java Secure Channel), programmers are able to connect to an SSH server and use port forwarding. Therefore we can use this library to setup client side to connect with the server. Simultaneously, we created a torrent file by using UTorrent software and selected one of the public active trackers and two computers as peers to share the file where one is creating seeding and uploading the file and the other is downloading from it.

We selected three different sizes of files, 10 MB, 100 MB, 500 MB and 1GB to test and compare the protocols' performance. To test each protocol's performance, we uploaded and downloaded each file with different size and calculate the time cost for SFTP, TFTP and BT. We did each test 10 times and took their average and standard deviation values to check the protocol's correctness and stability. We also tracked the time cost to evaluate the efficiency of protocols.

Result Analysis

Figure 1 shows the screen shot of representative time cost results for downloading 100 MB and 10 MB files using SFTP. It takes 90.659 and 117.848 seconds to download 100 MB files for the first and second time by SFTP and it takes 103.218 seconds to download 10 MB files.

```
Session created.
works here.
Session connected.
Opening Channel.
Connected to xxx.xxx.x.xxx.
success!
/home/xwang125/Desktop/files/100MB
Downloading file takes 90659 milliseconds
/home/xwang125/Desktop/files/100MB
Downloading file takes 117848 milliseconds
/home/xwang125/Desktop/files/10MB
Downloading file takes 103218 milliseconds
```

Figure 1 Screen shot downloading 100 MB file by SFTP

From time costs for 10 transfers, we are able to average the time cost for each protocol for downloading and uploading the same size files. Table 1 summarized the main results of average time cost for 10 transfers to download and upload 10 MB, 100 MB, 500 MB and 1 GB files. Obviously, the speed of download and upload using BT protocol is fastest and TFTP is slowest. For example, it takes TFTP and SFTP 55.687 seconds and 13.068 seconds to download 10 MB file, respectively, however, it only takes BT 11.584 seconds. The upload speed has the same tendency as the download for SFTP, TFTP and BT. SFTP acts more efficiently than TFTP over transferring and uploading. It might be because TFTP uses UDP, which has no windowing and acknowledgment, therefore TCP need to acknowledge each packet by itself and slows down transferring. BT protocol shows a bigger advantage over the other two protocols for larger size download and upload. For example, BT download of 1 GB file is more than two times faster than SFTP. This is due to the fact that the BT protocol is a peer-to-peer transfer protocol and there is no central location for authenticating users, instead, there are a group of computers connected

together to share resources. Each client could be also a server to download files and simultaneously upload files for other users in BTP and the download speed is only limited by your internet connection. Therefore, the time cost of upload and download by BT is the same, as shown in table 1. BT also has the best scalability comparing other two.

Table 1 Performance comparison results of TFTP, SFTP, BT. Avg refers to average time; STD represents the standard deviation; Proto means protocol.

Time (s) / Proto	Download				Upload			
	10 MB	100 MB	500 MB	1 G	10 MB	100 MB	500 MB	1 G
	Avg/ STD	Avg/ STD	Avg/ STD	Avg/ STD	Avg/ STD	Avg/ STD	Avg/ STD	Avg/ STD
TFTP	55.687/ 17.943	512.780/ 83.383	1037.69/ 138.972	N/A	58.991/ 19.226	524.906/ 76.319	1198.608/ 137.618	N/A
SFTP	13.068/ 2.85	119.654/ 12.43	667.542/ 51.869	1153.56/ 98.267	11.426/ 3.53	121.645/ 14.639	695.544/ 45.929	1246.33/96.279
BT	11.584/ 1.362	67.387/ 5.927	361.268/ 6.318	494.32/ 6.981	11.584/ 1.362	67.387/ 5.927	361.268/ 6.318	494.32/ 6.981

We see from Table 1 that the time cost of download/upload by TFTP and SFTP is proportional to the file size. For example, downloading 10 MB file by TFTP costs 55.687 seconds and it takes the file 512.780 seconds to downloading 100 MB file. Similarity, the time cost for SFTP is 10 times longer for downloading 10 times larger files.

However, for BT protocol, such correlation is different from TFTP and SFTP and the time consumption is not proportional to the file size in BT protocol. In this case, the user spends 11 seconds for downloading a 10 MB file but it takes the user 67 seconds to download a 100 MB file. Therefore, the download/upload by BT protocol has the most benefit for file with large scale. At the same time, BT test shows the best stability since the time consumption for the download of same size files changes much less than SFTP and TFTP. In table we can see the standard deviation (STD) result from 10 times of tests for each protocol. For instance, the value of STD of BT for 500MB files is 6.318, while SFTP and TFTP's are 51.869 and 138.972 respectively, which means the time-cost range of SFTP and TFTP are much wider than BT in the experiment, TFTP takes 53.424 to 105.927 seconds to download a 10 MB file, and SFTP takes 12.327 to 15.429 seconds, while BT takes from 9.372 to 11.966 seconds. Also, when the file's size increases, the STD of TFTP and SFTP are getting larger, while BT remains almost the same value. It is noted that we could not get the transferred file in client side after the program ran for 1 hour when we studied TFTP performance for the 1 GB file.

Conclusion

In this paper, we have evaluated and tested the performances of three different communication protocols, SFTP, TFTP and BT measuring the time cost with download/upload with various size files. The speed of download/upload of smaller size file (≤ 100 MB) by BT is more than 5 times faster than the time for TFTP and it is 3 time faster for a larger size file (≥ 500 MB). For smaller size files, the speed of BT protocol download is close to the speed of SFTP but it is much faster than SFTP for the download of larger size files. At the same time, BT protocol transfers files more stably since in each transfer the time cost doesn't change too much.

References

[1]http://www.jscape.com/blog/bid/75602/Understanding-Key-Differences-Between-FTP-FTPS-and-SFTP

[2] https://en.wikipedia.org/wiki/Trivial_File_Transfer_Protocol

[3] https://en.wikipedia.org/wiki/BitTorrent

[4] https://tools.ietf.org/html/rfc1350

[5] http://www.tcpipguide.com/TCPIPGuide_2-0_sec1.pdf

[6] http://pluto.ksi.edu/~cyh/cis370/ebook/ch06b.htm

[7] Cerf, V. G., & Kahn, R. E. A protocol for packet network intercommunication. *IEEE Transactions on Communications*, **22**, 5 (1971).

[8] https://en.wikipedia.org/wiki/SSH_File_Transfer_Protocol