International Journal of Scientific and Technical Advancements

ISSN: 2454-1532

R.A.F.T - Remote Access File Transfer

Krishna Rohila[1], Neuvine D'souza[2,] Kingsley Rodrigues[3]
[1, 2, 3]Computer Engineering, Xavier Institute of Engineering, Mumbai, Maharashtra, India-400016
Email Address: [1]krishna.rohila@gmail.com, [2]neuvine@gmail.com, [3]kingsleyrod@gmail.com

***Abstract*—** This paper focuses on the implementation of peer to peer terminology for a Local Area Network just like the torrent terminology for internet. The product built is used for file transfer from one machine to another in a many-to-many, one-to-many and many-to-one fashion. It is used for transfer of huge data or files within a locally interconnected network of computers at a high speed with features such as many-to-one machine transfer, pause and resume of the file transfer and splitting of the file into pieces and blocks and merging them at the client machine.

***Keywords*—** File transfer; local area network; peer-to-peer; remote acces; torrent.

I. Introduction

R.A.F.T. – Remote Access File Transfer works on the terminology of peer-to-peer file sharing. Peer-to-peer file sharing is the distribution and sharing of digital media using the peer-to-peer (P2P) technology. P2P file sharing allows users to share and access digital media such as music, documents, games, movies and books using a P2P software program that searches for peers and connects to them for download of such media. P2P file sharing eliminates the need for a centralized server. It is also efficient in terms of cost. Since the user is the provider and usually the provider is the administrator as well, the system administration overhead is smaller because. Hence the users themselves can monitor each network. At the same time, large servers require more storage and this increases the cost since the storage has to be rented or bought exclusively for a server. However, usually peer-to-peer file sharing does not require a dedicated server. However, when we talk about peer-to-peer in a local area network, it usually emphasises on one-to-one sharing of files and media.

The proposed software allows the user to transfer all the media form more than one machine to a single machine or more than a single machine. Thereby implementing a many-to-one transfer of the files or any digital media content across the locally interconnected network. The proposed software allows the user easily share large files or data between machines connected in a network.

II. Review of Literature

The review of working of torrent technology helps us in understanding how exactly the peer-to-peer technology works and can be implemented on a local area network. The torrent protocol consists of two logically distinct protocol, namely the Tracker HTTP Protocol (THP), and the Peer Wire Protocol (PWP). THP defines a method for contacting a tracker for the purposes of joining a swarm, reporting progress etc. PWP defines a mechanism for communication between peers, and is thus responsible for carrying out the actual download and upload of the torrent.

In order for a client to download a torrent the following steps must be carried through:

a) A meta-info file must be retrieved.
b) Instructions that will allow the client to contact other peers must be periodically requested from the tracker using THP.
c) The torrent must be downloaded by connecting to peers in the swarm and trading pieces using PWP.

To publish a torrent the following steps must be taken:

a) A tracker must be set up.
b) A meta-info file pointing to the tracker and containing information on the structure of the torrent must be produced and published.

At least one seeder with access to the complete torrent must be set up.

A. Pieces

This section describes how a torrent is organized in pieces and blocks. The torrent is divided into one or more pieces. Each piece represents a range of data which it is possible to verify using a piece SHA1 hash. When distributing data over PWP, pieces are divided into one or more blocks such as one piece may contain one or more blocks for the transfer to take place. The illustration below gives a representation of pieces and blocks.

```
Piece 1 ||           Piece 2              || Piece 3
         ||Block 1|Block 2|Block 3||
```

The number of pieces in a torrent is indicated in a meta-info file. The size of each piece in the torrent remains fixed and can be calculated using the following formula:

Fixed piece size = size of torrent / number of pieces

where '/' is the integer division operator. However, only the last piece of the torrent is allowed to have fewer bytes than the fixed piece size.

B. Blocks

The size of a block is an implementation defined value that is not dependant on the fixed piece size. Once a fixed size is defined, the number of blocks per piece can be calculated using the formula:

Number of blocks = (fixed piece size / fixed block size)
+ !!(fixed piece size % fixed block size)

where "%" denotes the modulus operator, and "!" the negation operator. The negation operator is used to ensure that the last factor only adds a value of 0 or 1 to the sum. Given the start offset of the block its index within a piece can be calculated using the formula:

block index = block offset % fixed block size

C. *The Meta-Info File*

The meta-info file provides the client with information on the tracker location as well as the torrent to be downloaded. Besides listing which files will result from downloading the torrent, it also lists how the client should split up and verify individual pieces making up the complete torrent. In order for a client to recognize the meta-info file it should have the extension .torrent and the associated the media type "application/x-<application_name>".

III. Problem Statement

Many institutions and organizations use traditional FTP file sharing method to transfer files from one machine to another. This method allows for the transfer of files from a single machine to a single machine only. Or else a client-server architecture allows for a transfer of file from one machine (server) to many other machines (clients). However this leads to decrease in overall speed of transfer of files, which leads to wastage of time. Another problem with the traditional method is that if the file transfer is interrupted in between, the whole transfer has to be started again from the start.

R.A.F.T – remote Access File Transfer solves these problems and allows easy sharing of files between locally interconnected networks. A user can transfer any large file from many machines to one machine easily thereby increasing the overall speed of transfer. Plus a user can pause and resume the transfer of file at any given time of the file transfer operation. The interruption in the transfer process will not result in complete failure of the transfer process. This will increase the reliability of the process and overall performance of the transfer process. Not only this software is reliable for large file transfer but it is also easy to use and decreases the overall time taken to transfer large files.

IV. Implementation Methodology

The proposed product or R.A.F.T. – remote Access File Transfer uses three main components: a) Tracker b) Server(s)/Uploader(s)/Sender(s)/peer(s)/seeder(s)
c) Client(s)/downloaders(s)/ receiver(s).

A. *Working*

This product works on the basis of the above given terms. Following steps take place for the transfer of the files.

Step 1: The files which are to be transferred to other machines needs to be selected from within the application. Once the file is selected, a .torrent file or meta-info file is generated containing all the information about the file. This meta-info file is encoded using the ‘bencoding’ method. This meta-info file also contains the IP address of the tracker which is online and is marked as ‘live’ so as to be tracked by the tracker. Once this file is created, it is tracked by the tracker and stored in the tracker information log. While the generation of the meta-info file, the selected file is encrypted by the SHA 1 algorithm and is divided into logical pieces and blocks which are later used for transfer of the file.

Step 2: Along with the log of the meta-info files, the information about the machines containing the files is also stored with the tracker. The meta-info file then can be transferred to the client machine and opened with the application. As soon as it is opened by the application, the client machine sends a request to the trackers asking for the information about the particular file. The tracker then searches for all the information matching the request from the client and locates all the other meta-info files, and the information on the requested file.

Step 3: Once the files are located on the other machines, the tracker provides a medium for a handshake between the client machine and all the other peer/seeder machines. On completion of the handshake between the client and the peers, the transfer of the file is started.

Step 4: Herein, the tracker needs to be always online for the transfer to take place between the machines. However, once the transfer is initiated, the tracker going offline wouldn’t matter as the information of meta-info file is already passed between the client and peers. But, once the tracker is offline, no new peer will be able to join the swarm of the file transfer process.

The progress of the completion of file can be checked in the application. Also the features such as pause and resume are available within the application. In case of many-to many file transfer, once a client completes downloading of the file, it also becomes a seeder for the other clients to whom the files are being transferred.

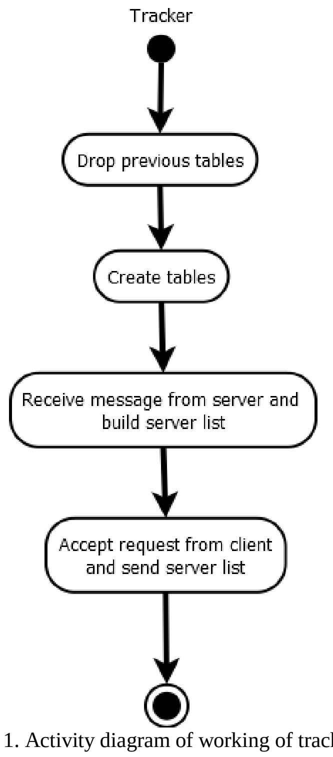
Tracker
Drop previous tables
Create tables
Receive message from server and build server list
Accept request from client and send server list

Fig. 1. Activity diagram of working of tracker.

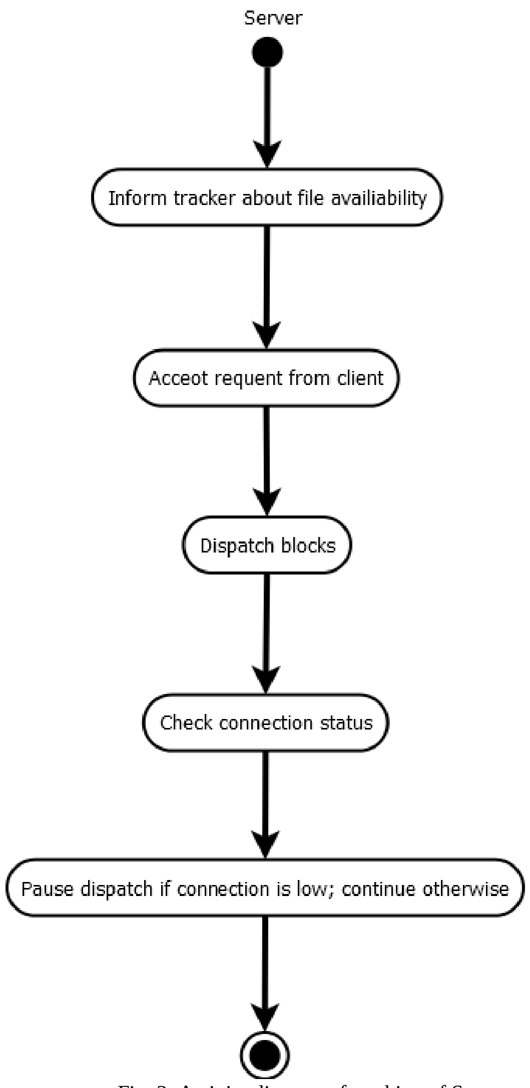
Server
Inform tracker about file availiability
Acceot requent from client
Dispatch blocks
Check connection status
Pause dispatch if connection is low; continue otherwise

Fig. 2. Activity diagram of working of Server.

B. Internal Working

1) Tracker

The Tracker will be any one machine which has to be always online for updating all of its logs/tables regarding the new meta-info files being generated by the users and also the log of previous meta-info files and all the associated information. The tracker keep a track of the peers and clients and the files being transferred between machines. The tracker, if goes offline and is restarted will drop all the previous tables/logs and will create new ones on the basis of the information received by the peers/servers and the requests received from the client machines. The working of the tracker can also be understood from the flowchart given below.

2) /Server(s)/uploader(s)/sender(s)/peer(s)/seeder(s)

The Server(s)/Uploader(s)/Sender(s)/peer(s)/seeder(s) will be referred to as Server in this section. The Server is the machine which has a copy of the file requested by the client. The server machine also has the meta-info file of the same file requested by the client and informs the tracker about the availability of the file. This machine needs to be online in order to transfer the file to the client machine. The server will accept the request from the client machine via the tracker. It will then dispatch he blocks and transfer the file block by block to the client machine. While the transfer takes place, the server will continuously check for the connection status and will pause the transfer if it is low. Or continue otherwise. The working of server can be also be understood from the flowchart given below.

3) Client(s)/downloaders(s)/ receiver(s)

Client machine is the one which request the tracker or server for some file(s). A client machine needs to have the meta-info file of the file which it is requesting for. When a client requests for a particular file, the tracker uses the information from the meta-info file and locates the file from among all the listed servers in tis tables. Once the connection is found, the client and server initiates a handshake after which the transfer of the file takes place. The client can pause and resume the transfer of the file at any given point of time.

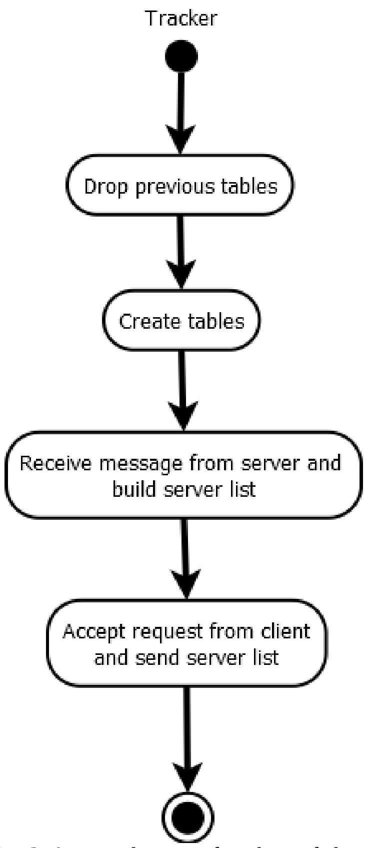
Tracker
Drop previous tables
Create tables
Receive message from server and build server list
Accept request from client and send server list

Fig. 3. Activity diagram of working of client.

V. CALCULATIONS

The formula for calculation for size of each piece of a file is:

fixed_piece_size = size_of_torrent / number_of_pieces

where "/" is the integer division operator. Only the last piece of the torrent is allowed to have fewer bytes than the fixed piece size

The formula for calculation for number of blocks per piece and block index is:

number_of_blocks = (fixed_piece_size / fixed_block_size)
+ !!(fixed_piece_size % fixed_block_size)

where "%" denotes the modulus operator, and "!" the negation operator. The negation operator is used to ensure that the last factor only adds a value of 0 or 1 to the sum. Given the start offset of the block its index within a piece can be calculated using the formula:

block_index = block_offset % fixed_block_size

The meta-info file is encoded using the Bencoding in the following manner:

Byte string are encoded as:
*<string length encoded in base ten ASCII>**:***<string data>*
There isn't any beginning or ending delimiter for this.
Example 5: piece represents the string "piece" of 5 letters.
Integers are encoded as:
i*<integer encoded in base ten ASCII>****e***
The initial **i** and trailing **e** are beginning and ending delimiters
Example: i5e represents the integer "5".
Lists are encoded as:
l*<bencoded values>****e***

The initial **l** and trailing **e** are beginning and ending delimiters. Lists may contain any bencoded type, including integers, strings, dictionaries, and even lists within other lists. Example l5:*hello5: worlds**e*** represents the list of two strings: ["spam", "eggs"]
Dictionaries are encoded as:
d*<bencoded string><bencoded element>****e***
The initial **d** and trailing **e** are the beginning and ending delimiters

VI. Conclusion

The proposed application provides user with a reliable and faster alternative way of transferring files between machines which are connected in a local area network. It will also provide the user with the features such as pause and resume of the transfer process in a local area network. It can be adopted by many institutions, organization, and workplaces.

References

[1] K. Widman, *How to Write a Bittorrent Client – Part 1*, 2012.
[2] K. Widman, *How to Write a Bittorrent Client – Part 2*, 2012.
[3] Theory.org–"BitTorrentSpecification," https://wiki.theory.org/BitTorrentSpecification