

Usage:

Our programming language is Java in which we can use the udp/tcp package and the sha1 hash function easily.

Simply run make in the command line, and a jar executable file named bittorrent-client.jar file will be generated in the directory. The directory has many useful resources you can use to test our implantation.

Below is the main structure of our project. And some statements and explanations will be attach.

| - src

| - |--main->his is where we coding

| - |--bencoding->third-party library we use to encode and decode the .torrent file information

Command:

Download using torrent file:

java -jar client.jar test_download_from_bittorrent.torrent

Start a server using 2333 port:

java -jar client.jar test_server_and_client.torrent server 2333 "BitTorrent.exe"

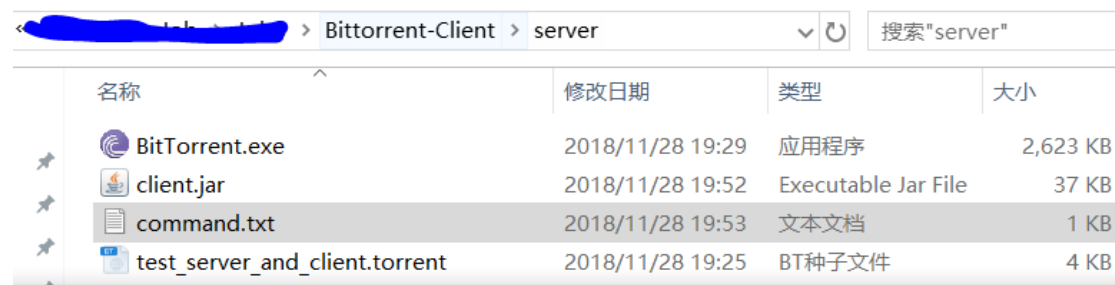
Start a client and establish tcp connection with server:

java -jar client.jar test_server_and_client.torrent client 2333

You can also just type **java -jar client.jar**, the you can see some hint about the usage.

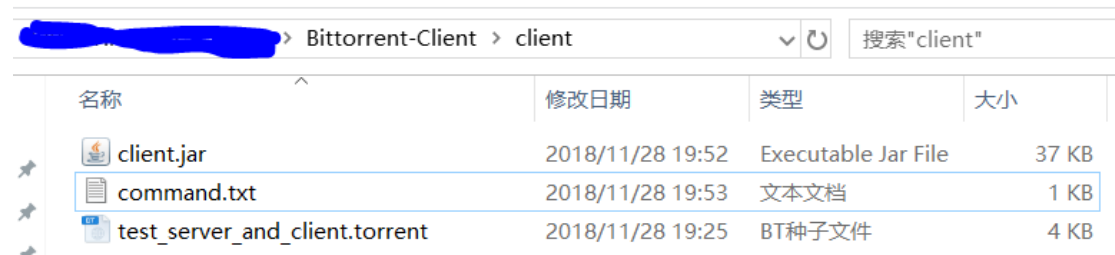
About server and client test, the best way is create two folder like below. I have created two test folder for easy test. (Just type below command, and everything is ok)

!!!!!!Be sure copy bittorrent-client.jar to client and server folder!!!!!!



名称	修改日期	类型	大小
BitTorrent.exe	2018/11/28 19:29	应用程序	2,623 KB
client.jar	2018/11/28 19:52	Executable Jar File	37 KB
command.txt	2018/11/28 19:53	文本文档	1 KB
test_server_and_client.torrent	2018/11/28 19:25	BT种子文件	4 KB

查看



名称	修改日期	类型	大小
client.jar	2018/11/28 19:52	Executable Jar File	37 KB
command.txt	2018/11/28 19:53	文本文档	1 KB
test_server_and_client.torrent	2018/11/28 19:25	BT种子文件	4 KB

```
CAWindows\System32\cmd.exe - java -jar bittorrent-client.jar test_server_and_client.torrent CAWindows\System32\cmd.exe - java -jar bittorrent-client.jar test_server_and_client.torrent client 2333
serverTCP=>/127.0.0.1:8992=>ServerTCP=>/127.0.0.1:8992=>send piece TCP=>/127.0.0.1:2333=>Piece_idx: 68 Send block request:idx:68 offset:12288 size:4096
serverTCP=>/127.0.0.1:8992=>ServerTCP=>/127.0.0.1:8992=>send piece TCP=>/127.0.0.1:2333=>Success get piece 68
serverTCP=>/127.0.0.1:8992=>ServerTCP=>/127.0.0.1:8992=>send piece TCP=>/127.0.0.1:2333=>Piece_idx: 69 Send block request:idx:69 offset:0 size:4096
serverTCP=>/127.0.0.1:8992=>ServerTCP=>/127.0.0.1:8992=>send piece TCP=>/127.0.0.1:2333=>Piece_idx: 69 Send block request:idx:69 offset:4096 size:4096
serverTCP=>/127.0.0.1:8992=>ServerTCP=>/127.0.0.1:8992=>send piece TCP=>/127.0.0.1:2333=>Piece_idx: 69 Send block request:idx:69 offset:8192 size:4096
serverTCP=>/127.0.0.1:8992=>ServerTCP=>/127.0.0.1:8992=>send piece TCP=>/127.0.0.1:2333=>Piece_idx: 69 Send block request:idx:69 offset:12288 size:4096
serverTCP=>/127.0.0.1:8992=>ServerTCP=>/127.0.0.1:8992=>send piece TCP=>/127.0.0.1:2333=>Get piece data error. Retry.
serverTCP=>/127.0.0.1:8992=>ServerTCP=>/127.0.0.1:8992=>send piece TCP=>/127.0.0.1:2333=>Piece_idx: 69 Send block request:idx:69 offset:4096 size:4096
serverTCP=>/127.0.0.1:8992=>ServerTCP=>/127.0.0.1:8992=>send piece TCP=>/127.0.0.1:2333=>Piece_idx: 69 Send block request:idx:69 offset:8192 size:4096
serverTCP=>/127.0.0.1:8992=>ServerTCP=>/127.0.0.1:8992=>send piece TCP=>/127.0.0.1:2333=>Piece_idx: 69 Send block request:idx:69 offset:12288 size:4096
serverTCP=>/127.0.0.1:8992=>ServerTCP=>/127.0.0.1:8992=>send piece TCP=>/127.0.0.1:2333=>Get piece data error. Retry.
serverTCP=>/127.0.0.1:8992=>ServerTCP=>/127.0.0.1:8992=>send piece TCP=>/127.0.0.1:2333=>Piece_idx: 69 Send block request:idx:69 offset:8192 size:4096
serverTCP=>/127.0.0.1:8992=>ServerTCP=>/127.0.0.1:8992=>send piece TCP=>/127.0.0.1:2333=>Piece_idx: 69 Send block request:idx:69 offset:12288 size:4096
serverTCP=>/127.0.0.1:8992=>ServerTCP=>/127.0.0.1:8992=>send piece TCP=>/127.0.0.1:2333=>Get piece data error. Retry.
serverTCP=>/127.0.0.1:8992=>ServerTCP=>/127.0.0.1:8992=>send piece TCP=>/127.0.0.1:2333=>Piece_idx: 70 Send block request:idx:70 offset:0 size:4096
serverTCP=>/127.0.0.1:8992=>ServerTCP=>/127.0.0.1:8992=>send piece TCP=>/127.0.0.1:2333=>Piece_idx: 70 Send block request:idx:70 offset:4096 size:4096
serverTCP=>/127.0.0.1:8992=>ServerTCP=>/127.0.0.1:8992=>send piece TCP=>/127.0.0.1:2333=>Piece_idx: 70 Send block request:idx:70 offset:8192 size:4096
serverTCP=>/127.0.0.1:8992=>ServerTCP=>/127.0.0.1:8992=>send piece TCP=>/127.0.0.1:2333=>Piece_idx: 70 Send block request:idx:70 offset:12288 size:4096
serverTCP=>/127.0.0.1:8992=>ServerTCP=>/127.0.0.1:8992=>send piece TCP=>/127.0.0.1:2333=>Get piece data error. Retry.
serverTCP=>/127.0.0.1:8992=>ServerTCP=>/127.0.0.1:8992=>send piece TCP=>/127.0.0.1:2333=>Piece_idx: 70 Send block request:idx:70 offset:4096 size:4096
serverTCP=>/127.0.0.1:8992=>ServerTCP=>/127.0.0.1:8992=>send piece TCP=>/127.0.0.1:2333=>Piece_idx: 70 Send block request:idx:70 offset:8192 size:4096
serverTCP=>/127.0.0.1:8992=>ServerTCP=>/127.0.0.1:8992=>send piece TCP=>/127.0.0.1:2333=>Piece_idx: 70 Send block request:idx:70 offset:12288 size:4096
serverTCP=>/127.0.0.1:8992=>ServerTCP=>/127.0.0.1:8992=>send piece TCP=>/127.0.0.1:2333=>Get piece data error. Retry.
serverTCP=>/127.0.0.1:8992=>ServerTCP=>/127.0.0.1:8992=>send piece TCP=>/127.0.0.1:2333=>Piece_idx: 70 Send block request:idx:70 offset:4096 size:4096
serverTCP=>/127.0.0.1:8992=>ServerTCP=>/127.0.0.1:8992=>send piece TCP=>/127.0.0.1:2333=>Piece_idx: 70 Send block request:idx:70 offset:8192 size:4096
serverTCP=>/127.0.0.1:8992=>ServerTCP=>/127.0.0.1:8992=>send piece TCP=>/127.0.0.1:2333=>Piece_idx: 70 Send block request:idx:70 offset:12288 size:4096
serverTCP=>/127.0.0.1:8992=>ServerTCP=>/127.0.0.1:8992=>send piece TCP=>/127.0.0.1:2333=>Get piece data error. Retry.
```

1) Supported features

All the core features and the support for udp-tracker protocol in extra credit.

2) Design and implementation choices

Our torrent file choosing.

Before we start, we want to get better result when we use the protocol, so we made some choices below:

- We choose some hottest torrent file to do the experiment because we can get a bunch of peers. After carefully choosing, we eventually selected two torrent files. Because many site doesn't offer torrent files instead just a magnet url, we then copy the magnet url and make torrent files use the function of the bittorrent application on our local machine.
- We close the bittorrent DHT function.
- After we did a) and b) we get download rate at 200kb/s and always get 4~5 peers per download.

After we started coding, we found there are so many problem use the torrent file which we downloaded on the Internet. The worst problem is the connection to the tcp. We find the truth when we do our experiment: all tcp connection encountered timeout but the same tcp connection work well in Bittorrent application.

```

=====>>>>End Connect To Tracker=====>>>>
=====>>>>Start TCP To Peers=====>>>>
TCP==>/45.41.132.133:14891==>start establish tcp
TCP==>/77.229.99.66:18359==>start establish tcp
TCP==>/139.180.216.203:2334==>start establish tcp
TCP==>/41.244.243.216:22595==>start establish tcp
TCP==>/186.220.89.38:32194==>start establish tcp
TCP==>/141.85.0.123:21689==>start establish tcp
TCP==>/92.58.61.122:7703==>start establish tcp
TCP==>/121.7.41.47:29240==>start establish tcp
TCP==>/139.180.216.203:16314==>start establish tcp
TCP==>/37.116.201.166:28353==>start establish tcp
TCP==>/149.28.149.75:16372==>start establish tcp
TCP==>/184.173.25.79:1337==>start establish tcp
TCP==>/101.177.23.162:4109==>start establish tcp
TCP==>/81.84.86.70:28848==>start establish tcp
TCP==>/139.180.216.203:16314==>ERROR: can't get the socket use the ip and port==>Connection refused: connect
TCP==>/139.180.216.203:2334==>ERROR: can't get the socket use the ip and port==>Connection refused: connect
TCP==>/45.41.132.133:14891==>ERROR: can't get the socket use the ip and port==>Connection timed out: connect
TCP==>/149.28.149.75:16372==>ERROR: can't get the socket use the ip and port==>Connection timed out: connect
TCP==>/141.85.0.123:21689==>ERROR: can't get the socket use the ip and port==>Connection timed out: connect
TCP==>/184.173.25.79:1337==>ERROR: can't get the socket use the ip and port==>Connection timed out: connect
TCP==>/92.58.61.122:7703==>ERROR: can't get the socket use the ip and port==>Connection timed out: connect
TCP==>/186.220.89.38:32194==>ERROR: can't get the socket use the ip and port==>Connection timed out: connect
TCP==>/45.41.132.133:14891==>ERROR: can't get the socket use the ip and port==>Connection timed out: connect
TCP==>/37.116.201.166:28353==>ERROR: can't get the socket use the ip and port==>Connection timed out: connect
TCP==>/41.244.243.216:22595==>ERROR: can't get the socket use the ip and port==>Connection timed out: connect
TCP==>/77.229.99.66:18359==>ERROR: can't get the socket use the ip and port==>Connection timed out: connect
TCP==>/101.177.23.162:4109==>ERROR: can't get the socket use the ip and port==>Connection timed out: connect
TCP==>/121.7.41.47:29240==>ERROR: can't get the socket use the ip and port==>Connection timed out: connect
=====>>>>After TCP and first handshake below peer are working fine
=====>>>>End TCP To Peers=====>>>>

```

And for the Bittorrent application.

IP	Client	Flags	%	Down Spe...	Up Speed	Reqs	Uploaded	Download...
149.28.149.75	BitTorrent 7.10.4	D	100.0	112.7 kB/s	0.2 kB/s	28 0		736 kB
81.84.86.70	Vuze 5.7.6.0	D	100.0	29.5 kB/s	0.1 kB/s	9 0		80.0 kB
101.177.23.162	BitTorrent 7.10.4	D	100.0	15.3 kB/s	0.1 kB/s	9 0		80.0 kB
37.116.201.166	µTorrent 3.5.4	D	100.0	12.1 kB/s	0.0 kB/s	3 0		64.0 kB
92.58.61.122	Vuze 5.7.6.0	D	100.0	3.4 kB/s	0.1 kB/s	3 0		16.0 kB

Name	Status	Update In	Seeds	Peers	D
[Peer Exchange]	working		0	3	
udp://tracker.opentracker.org:1337/announce	working	27m 23s	11	3	
[DHT]	disabled		0	0	
[Local Peer Discovery]	disabled		0	0	
udp://tracker.openbittorrent.com:80/announce		updating...	0	0	

We finally solve this problem by deploying another server and make seed on it. So it's better to use our torrent for this experiment test.

Our server is deploy at cloud. When you finish download, the file structure should be like below.

it 7.10.4 (build 44847) [32-bit]

is Help

ide to Pro


its (1)

s

i (0)

es (0)

ADVERTISEMENT



50% on

Take

#	Name	Size
	bittorrent	10.2 MB

File Home Share View

bittorrent

Quick access

- Desktop
- Downloads
- Documents
- Pictures
- bittorrent
- This PC
- Network

Name

- t1
- t2
- BitTorrent.exe

Files

Downloaded

Availability:

Transfer

Time Elapsed

Downloaded

Download Limit


Status:

General

Report

6 : a look at the most beautiful competition

ADVERTISEMENT



50% on

Take

#	Name	Size
	bittorrent	10.2 MB

File Home Share View

bittorrent

Quick access

- Desktop
- Downloads
- Documents
- Pictures

Name

- BitTorrent1.exe
- BitTorrent2.exe

Files

Downloaded

Availability:

Transfer

Take our survey

#	Name	Size	Status
	bittorrent	10.2 MB	Seeding

File Home Share View

bittorrent > t2

Quick access

- Desktop
- Downloads
- Documents
- Pictures

Name

- BitTorrent.exe

Files

Downloaded

Availability:

Transfer

Why we use udp mainly to fetch peers' information.

In the beginning, we just implement http protocol to fetch tracker peers. But after many tries, we realize that http nearly response data with no peers. We googled that the http protocol used in p2p cause too much overload on the Internet and the suggest solution is connect server using udp protocol. As said above, we need get as much peers as possible so that the program can get fast down speed. So It's reasonable that we also implement udp protocol.

3) Problems

The most strangest problem we met is the magic number for the `udp_protocol_id(0x41727101980L)`. Even a slight change will cause fall to connect the tracker. Wiki's resources don't figure out its' importance but we finally find this magic number shouldn't be changed when we googled the solution.

For our two torrent file, only one torrent file contains the http tracker. But none of them has the peers data. That is why we add udp protocol implementation.

4) Bugs

None as we operate properly.