TJHSST Senior Research Project
End-to-End Publication Using the Bittorrent P2P Filesharing Protocol
2006-2007

Andrew Wang

January 24, 2007

Abstract

Bittorrent is a promising peer-to-peer network that always allows for fast download speeds despite the number of peers downloading the file. Currently, there exist tools to make .torrent files, tools to "track" the peers downloading the file, tools to host .torrent files, and tools to initially upload the file. This project aims to unify this process by making an end-to-end software suite that simplifies the process of publishing a file on the Bittorrent network for download. The key to this will be automating and streamlining the process from the perspective of the user. It will involve a complete implementation of the Bittorrent protocol, including encoding torrent files, peer-to-tracker and peer-to-peer communication, and a greater understanding of the benefits and detriments of the Bittorrent protocol.

Keywords: Bittorrent, Peer-to-peer, Linux, Publishing, Download, Tracker

1 Introduction

1.1 Rationale

The scope of the project is broad, because it aims to be a complete solution to publishing files using Bittorrent. It will have to handle all aspects of the

Bittorrent protocol, from processing the file to make the .torrent metadata file to hosting and tracking the .torrent file and possibly a download client.

A new way of publishing using Bittorrent is important because the current system of publishing is much more complicated and is inaccessible to the normal computer user. Bittorrent has clear advantages over traditional methods of publishing via the Internet, such as HTTP or email, because it can handle a far larger number of users concurrently and thus allows for the publishing of far larger files, such as indie HD movies, podcasts, or other content that would otherwise be unfeisable because of bandwidth constraints. This system would also be superior to other forms of P2P though the use of a "everseed" that would keep the torrent from "dying" (a state where there are no peers with a complete copy of the file).

1.2 Purpose

This project started out as writing a better download client, but there is already a plethora of download clients available and finding an easy to use and mature Bittorrent development library proved difficult. It would also be difficult to surpass the quality and features of other download clients developing by myself, and users would be unlikely to switch their choice in download clients unless there was a very good reason. Thus, the project has been redirected toward streamlining the currently convoluted process of distributing a file using the Bittorrent protocol.

1.3 Expected Results

The research of this project involves the Bittorrent protocol. Given the specification of the protocol given on the Bittorrent website, I aim to implement the server side aspect. This means encoding and dencoding of .torrent metadata files, hosting of .torrent metadata files, and subsequent tracking and peer-to-tracker communication of download peers by the tracker. It will involve the subject areas of large scale networks, encoding and decoding algorithms, and peer-to-peer communication.

1.4 Type of Research

This project will be use-inspired basic research, because the underlying goal is to gain an understanding of the benefits and limitations of the Bittorrent

protocol. There are great practical implications for the end product of this research, but ultimately the project was started to gain an introduction to networking and peer-to-peer technology.

2 Background

Bittorrent is an up and coming filesharing protocol that has emerged in the wake of illegal services such as Kazaa, Napster, or Bearshare that have since been shutdown or forced to end their copyright violations. Bittorrent is a much more legally feisable filesharing protocol than previous attempts, because there is no copyrighted content to be stored on centralized servers that can be subpoenaed or seized, and it has become extremely popular for independent movie makers and other people that need to distribute their legal content without buying an expensive server. A movie distribution method using Bittorrent is also being developed by major movie companies, as they too see the benefits of peer-to-peer technology.

The whitepaper written by the creator of Bittorrent available on the official Bittorrent website is the most useful reference for this project. Additionally, there is a page on wiki.theory.org that takes and expands upon the official protocol specification that is useful for more detailed help. These two documents give a total description of all aspects of Bittorrent, and will be the only necessary references throughout the project's extent.

3 Procedure and Methodology

3.1 Planning

The languages used in this project will be Python, for all parts of the project. Performance is not an issue because the processor and bandwidth requirements are low. A webserver of some sort will be needed to host the .torrent files, but this can be done with a third-party solution, or a basic server can be written if needed. The stages of this project can be split up into a number of clearly defined steps:

1. Study and implementation of encoding .torrent metadata files. These files are "bencoded," which is a translated form of dictionaries, lists,

strings, and integers. This will also coincide with studying of the various kinds of metadata stored in .torrent files as well as an interface for creating these .torrent files.

2. Study and implementation of a Bittorrent tracker. The tracker must process the .torrent metadata value, store it into a database, and then handle processing "announce" and "scrape" requests from the clients that wish to download the file. It will also make use of the "bencoding" algorithm to send data from the tracker to the client.

   A typical "announce" request from a client consists of status and unique identifying information. The tracker stores this information in a database, and the tracker then sends the client identifying information and a list of peers for the client to connect to for downloading and uploading purposes.

   A typical "scrape" request asks the tracker for status information about a single or all torrents that the tracker is tracking. The difference between an "announce" and "scrape" request is determined by the URL used to query the tracker. The tracker will respond with information like the number of peers and seeds connected, and the number of downloads completed.

3. Making a web interface that takes a file, prompts the user for the minimum amount of information regarding the file through the use of automation and intelligent defaults, make a .torrent metadata file for it, add it to the tracker, and put the file up for download via HTTP.

4. The final part of the project is the addition of the "everseed." The "everseed" is the initial and permanent uploader for the file that prevents the torrent from "dying" (a state where a complete copy of the file does not exists among the peers in the swarm, preventing the file from ever reaching completion). This is a problem when a torrent has been around for a long time or is not that popular.

3.2 Testing and Analysis

Testing of encoding .torrent files is done using examples on the Bittorrent website and others. The program will transparently handle errors because it will simple treat the invalid input as a string. This will result in an incorrect

.torrent file though, so I will build in checking when I make the frontend for making .torrents. Performance is also not an issue for the bencoding program because it takes minimal time even with the use of Python. The torrent files have also been verified using the official Bittorrent client as well as popular clients such as Azureus or Shad0w's.

```
[03:42:11] awang::hermit $ ./torrentfile.py

torrentfile.py 0.2
Enter a string: hello world!
Bencoded string: 12:hello world!
Enter an integer: 12345
Bencoded integer: i12345e
Enter a list: apple,orange,pear,grape
Bencoded list: l5:apple6:orange4:pear5:grapee

Example bencoded dictionary
Dictionary:  {'myname': ['andrew', 'wang'],
'dozen': 12, 'apple': 'red', 'banana': 'yellow'}
Bencoded: d6:mynamel6:andrew4:wange5:dozeni12e5:
apple3:red6:banana6:yellowe
```

The following is the step-by-step process through which the tracker handles an announce request from a client. First is a typical HTTP GET announce request from the test Bittorrent client that I am developing concurrently to test the tracker. The data is passed in the GET request as urlencoded key-value pairs:

```
GET /announce/?uploaded=314159&compact=YES%21&numwant=3&
ip=127.0.0.1&info_hash=abcdefghijklmnopqrstuvwxyz&event=started&
downloaded=951413&trackerid=&key=AWANG
&peer_id=evertestclient000000&port=6881&left=1 HTTP/1.0
Host: localhost:6969
User-agent: Python-urllib/1.16
```

The tracker then urldecodes this text, and turns it into a much more useful python dictionary:

```
{'uploaded': '314159', 'compact': 'YES!', 'numwant': '3', 'ip': '127.0.0.1',
```

```
 'info_hash': 'abcdefghijklmnopqrstuvwxyz', 'event': 'started', 'downloaded':
  '951413', 'key': 'AWANG', 'peer_id': 'evertestclient000000', 'port': '6881',
  'left': '1'}
```

The data in this dictionary is then used to form an appropriate response. Various parts of this, such as the status, downloaded, key, and peer_id, are stored in the database for later use. For example, the tracker honors the numwant optional variable from the client, which is of the value 3, and forms the following dictionary (please note that the peers in this scenario are generated randomly):

```
{'peers': [{'ip': u'68.150.132.78', 'peer_id': u'EVERCLIENT1111111111',
 'port': 12454}, {'ip': u'106.190.185.236', 'peer_id': u'EVERCLIENT8888888888',
 'port': 41582}, {'ip': u'8.56.239.117', 'peer_id': u'EVERCLIENT8888888888',
  'port': 64733}], 'min interval': 240, 'complete': 1, 'interval': 720,
  'warning message': '', 'tracker id': 'EVERTRACKER', 'incomplete': 0}
```

This is then bencoded, and sent to the client as a text/plain document, which is seen as follows:

```
d8:completei1e10:incompletei0e8:intervali720e12:min interval
i240e5:peersld2:ip13:250.192.86.977:peer_id20:EVERCLIENT2222222222
4:porti18252eed2:ip13:156.51.108.137:peer_id20:EVERCLIENT7777777777
4:porti1171eed2:ip13:64.166.125.537:peer_id20:EVERCLIENT4444444444
4:porti25819eee10:tracker id11:EVERTRACKER15:warning message0:e
```

The client then bdecodes it, yielding the following dictionary:

```
{'peers': [{'ip': '250.192.86.97', 'peer_id': 'EVERCLIENT2222222222',
'port': 18252}, {'ip': '156.51.108.13', 'peer_id': 'EVERCLIENT7777777777',
'port': 1171}, {'ip': '64.166.125.53', 'peer_id': 'EVERCLIENT4444444444',
 'port': 25819}], 'interval': 720, 'complete': 1, 'min interval': 240,
  'warning message': '', 'tracker id': 'EVERTRACKER', 'incomplete': 0}
```

This data will then be enough such that the client can try to connect to other peers and actually start downloading the file.

3.3 Goals and Requirements

The goals for this project are as follows:

1. Easy to use, automated front end for the user
2. Bencoded .torrent file creation and parsing.
3. Correct implementation of tracker software and tracker-peer communication
4. Implementation of an automatic, permanent "everseed" that prevents the torrent from dying

4 Expected Results

I expect a complete, easy to use frontend that will handle and automate as much of the process of publishing a file through Bittorrent as possible. These results will be represented with screenshots and flowcharts describing the process. The website should be easy enough to use and well designed so that the proper steps to take are obvious.

This could be a very useful way to easily distribute files within a bandwidth limited environment. It could be useful to any project that needs to distribute large files or other people who want to use the Bittorrent protocol, because it will be a complete implementation.