BitTorrent Client

GitHub link: https://github.com/GhermanCristian/BitTorrentClient

Abstract

Transferring large files over the internet is not a novel concept. However, most of these methods require some form of server which sends or receives the data from some clients. In addition to operational costs, communicating through a server may be slower and more susceptible to privacy breaches.

This is where **peer-to-peer** protocols come in. Peer-to-peer (P2P) data transfer means that the involved clients (peers) can communicate directly. The software mostly associated with peer-to-peer nowadays is the **bittorrent** (or just "torrent") client. Popular bittorrent programs, such as µTorrent, Vuze and BitComet have been around since the early 2000s, with the first version of the BitTorrent protocol being released in 2001.

The aim of this work is to create a lightweight bittorrent client, which does not offer any advertisements or hidden malware. Because of its open-source availability, most transparency issues will be addressed, therefore more users will be incentivized to use it. Additionally, this paper can be used as a means of studying the peer-to-peer and the BitTorrent protocol.

ACM Classification:

Computer systems organization -> Architectures -> Distributed Architectures -> **Peer-to-peer architectures**

Networks -> Network properties -> **Network privacy and anonymity**

Networks -> Network algorithms -> Control path algorithms -> **Network resources allocation**

Security and privacy -> Security services -> **Digital rights management**

Security and privacy -> Systems security -> **Distributed systems security**

Computing methodologies -> Distributed computing methodologies -> **Distributed algorithms**

AMS Classification:

68M14 – **Distributed systems**

68W15 – **Distributed algorithms**

Introduction

The **BitTorrent** protocol was developed by Bram Cohen and first released in 2001. It involves several **seeders** (clients which already own the entire file) uploading parts of it to other clients, until eventually they receive it completely and become seeders themselves. The connection between the clients is usually done through a server ("**tracker**"); however, this server has no access to the distributed content, just to the clients involved. There is also the

option to use distributed hash tables, which would render trackers useless, but it is not as common.

The current application aims to provide a user with such functionalities in a safe and advertisement-free manner.

Original contributions

To begin with, this BitTorrent client does not provide as much security as other similar clients, nor does it use sophisticated algorithms to improve the download or upload speed. Therefore, it does not consume as many resources, being a good alternative for systems with limited processing power.

Moreover, it is not bloated with functionalities such as live data streaming, embedded search engines, prioritization or super-seeding. This further reduces power requirements.

Lastly, the product does not contain any advertisements, nor any hidden malware or crypto-mining services. It is provided as an open-source application, which will address issues regarding the lack of transparency and trust in such products, therefore users will be more incentivized to use it.

Programming languages. Frameworks

The program will be predominantly written in **Python 3**. In addition to being easy to use, debug and test, it is also one of the languages used by Bram Cohen, the creator of this protocol, to develop the original BitTorrent client.

Another potential alternative was using **node.js**. However, due to my lack of experience using it, I chose not to use it in this project. Moreover, I believe that **Python** is more suitable for the chosen application architecture (single-module, not client-server).

Some of the more important libraries and frameworks that will be used are:

- **bencode3** - used for parsing bencoded content, such as .torrent files and the tracker response
- **hashlib** - a native **Python** library used in order to encrypt / decrypt some checksum values specific to torrents
- **requests** - used to make HTTP requests
- **Twisted** – framework used for handling simultaneous connections to multiple other users. An alternative to this would have been to create my own event-driven programming loop: however, I chose an existing library because it has already been properly tested and optimized.

The application will have a GUI, which will be implemented in the **tkinter** library. I consider this library to be fairly easy to use, not very wasteful with resources, and able to fulfill the necessary visual and interaction requirements of the program. The **pygame** library was also a good option, and the same could be said for **pyqt**.

Application lifecycle

The product is a desktop application for Windows; as such, the development will be done entirely on Windows. The feature-driven development technique will be used, and the source code repository will be hosted on GitHub.

The first stage is **planning**, which involves listing the application features, evaluating and selecting the most suitable programming languages, frameworks and design patterns.

The **development** stage consists of improving the application one feature at a time. Before being released, each feature is tested using unit testing. As more features are implemented, integration tests will be run to ensure that the application as a whole works properly.

The **release** stage consists of testing the product with several real-life users. Malfunctions detected during each release will be patched in future builds, which will again be released to some users.

Related work

As previously mentioned, the original BitTorrent client, BitTorrent, was also mostly written in **Python**. As with most such clients, it offered features like downloading and uploading multiple files (in parallel), pausing and restarting the download process of a file, or displaying various information such as the completion rate, download speed and ETA. These functionalities will also be present in the current program.

Another popular client, µTorrent, was written in **C++** using the Win32 API. In its early stages it was considered very lightweight; however, nowadays it contains lots of new features which some consider unnecessary. Some of these features include streaming, anti-virus protection, embedded search engines – none of which will feature in this program.

An important difference between several programs available on the market and the current one is their dependency on **advertising** for revenue. This client is not created for-profit and therefore does not need to display advertisements, which use valuable bandwidth and resources and can also be malicious toward users.

Deployment

The application will be ready-to-use (as an **.exe** file) and will not require any installation process. It will be published on a platform such as sourceforge.net, from where the latest builds will be available for download.

Conclusions

This program is not a perfect replacement for the current popular BitTorrent clients. However, it being free and not containing any advertisements or malware may be appealing to many users.

Bibliography

1. Aberer, Karl, Zoran Despotovic. "Managing trust in a peer-2-peer information system." In Proceedings of the tenth international conference on Information and knowledge management, pp. 310-317. 2001.
2. Allen, Kim. *How to make your own bittorrent client*. https://allenkim67.github.io/programming/2016/05/04/how-to-make-your-own-bittorrent-client.html. 2018. Retrieved November 27, 2021.
3. Anton, B., and A. Norbert. "Peer to peer system deployment." Acta Electrotech. Inform 16 (2016): 11-14.
4. *bencode3 - pypi*. https://pypi.org/project/bencode3/. Retrieved November 28, 2021.
5. Bharambe, A. R., et al. “Analyzing and Improving a BitTorrent Networks Performance Mechanisms.” Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications, 2006, pp. 1–12.
6. *BitTorrent Protocol Specification v1.0*. https://wiki.theory.org/index.php/BitTorrentSpecification. Retrieved November 27, 2021.
7. Cohen, Bram. “Incentives Build Robustness in BitTorrent”. Bittorrent. May 22, 2003. http://bittorrent.org/bittorrentecon.pdf
8. Cohen, Bran. *The BitTorrent Protocol Specification*. http://www.bittorrent.org/beps/bep_0003.html. Retrieved December 1, 2021
9. Fox, Geoffrey. "Peer-to-peer networks." *Computing in Science & Engineering* 3, no. 3 (2001): 75-77.
10. Gardiner, Jesse; Sheppard, Travis; Lasa, John. “User Guide: Bit Torrent Server”. Southern Institute of Technology. April 15, 2010.
11. *hashlib - Secure hashes and message digests*. https://docs.python.org/3/library/hashlib.html. Retrieved November 28, 2021.
12. Izal, Mikel, Guillaume Urvoy-Keller, Ernst W. Biersack, Pascal Felber, Anwar Al Hamra, and Luis Garcés-Erice. 2004. “Dissecting BitTorrent: Five Months in a Torrent’s Lifetime.” In International Workshop on Passive and Active Network Measurement, 1–11.
13. Legout, Arnaud, Nikitas Liogkas, Eddie Kohler, and Lixia Zhang. 2007. “Clustering and Sharing Incentives in BitTorrent Systems.” In Proceedings of the 2007 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, 35:301–12.
14. Liu, Qiong, Reihaneh Safavi-Naini, and Nicholas Paul Sheppard. 2003. “Digital Rights Management for Content Distribution.” In ACSW Frontiers ’03 Proceedings of the Australasian Information Security Workshop Conference on ACSW Frontiers 2003 - Volume 21, 49–58.
15. Piatek, Michael, Tomas Isdal, Thomas Anderson, Arvind Krishnamurthy, and Arun Venkataramani. 2007. “Do Incentives Build Robustness in Bit Torrent.” In NSDI’07 Proceedings of the 4th USENIX Conference on Networked Systems Design & Implementation, 1–1.

16. Qiu, Dongyu, and R. Srikant. “Modeling and Performance Analysis of BitTorrent-like Peer-to-Peer Networks.” Acm Special Interest Group on Data Communication, vol. 34, no. 4, 2004, pp. 367–378.
17. *Requests: HTTP for humans*. https://docs.python-requests.org/en/latest/. Retrieved November 28, 2021.
18. Schollmeier, Rüdiger. "A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications." In *Proceedings First International Conference on Peer-to-Peer Computing*, pp. 101-102. IEEE, 2001.
19. Shen, Xuemin, Heather Yu, John Buford, and Mursalin Akon, eds. Handbook of peer-to-peer networking. Vol. 34. Springer Science & Business Media, 2010.
20. Singh, M. "Peering at peer-to-peer computing." IEEE Internet computing 5, no. 6 (2001): 4-5.
21. Stutzbach, Daniel, and Reza Rejaie. 2006. “Understanding Churn in Peer-to-Peer Networks.” In Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement, 189–202.
22. *The basics – Twisted 17.9.0 documentation*. https://docs.twistedmatrix.com/en/twisted-17.9.0/core/howto/basics.html. Retrieved November 28, 2021.
23. *tkinter — Python interface to Tcl/Tk*. https://docs.python.org/3/library/tkinter.html. Retrieved November 28, 2021.
24. Widman, Kristen. *How to write a BitTorrent Client, Part 1*. http://www.kristenwidman.com/blog/33/how-to-write-a-bittorrent-client-part-1/. November 23, 2012. Retrieved November 27, 2021.
25. Widman, Kristen. *How to write a BitTorrent Client, Part 2.* http://www.kristenwidman.com/blog/71/how-to-write-a-bittorrent-client-part-2/. November 27, 2012. Retrieved November 27, 2021.
26. Yadav, Megha, Ms Shalini Bhadola, Ms Kirti Bhatia, and Rohini Sharma. "Torrent Poisoning: Antipiracy and Anonymity." International Journal of Innovative Analyses and Emerging Technology 1, no. 3 (2021): 60-63.