Distributed Computing Group, ETH Zurich

BitThief New UX

Semester Thesis

Vishrant Vasavada

v.vasavada@mail.utoronto.ca

Department of Computer Science,

University of Toronto

Supervisors:

Michael König

Prof. Roger Wattenhofer

Acknowledgements

I would like to express my gratitude, especially to my mentor Michael König for supporting me during the whole project with a constant input of new ideas and improvement proposals and also for providing all the assistance needed when I was stuck. I would also like to thank Prof. Roger Wattenhofer who gave me the opportunity to be the part of this project. Lastly, I would like to thank Prof. Francois Pitt and Prof. Christiana Amza at University of Toronto who agreed to read my project report and feedback given by Prof. Wattenhofer, and allowed me to pursue this project.

Abstract

User should be able to have one stop destination for torrent file search, information retrieval (such as number of seeders, file size, upload date) and download, rather than visiting multiple torrent websites to obtain the torrent files they want and transfer them to the BitTorrent client. With the BitThief's (a BitTorrent client) new User Experience (UX), we try to make this possible by providing the user with a fully configurable system, integrated into the BitThief, which can act as a meta-search engine, along with result filters, and take their torrent search, information retrieval and download experience to a whole new level.

Introduction

BitThief is a “freeloading” BitTorrent client which downloads files from BitTorrent swarms without returning the favor by uploading. The goal of this project was to implement a flexible user-configurable system for integrating existing torrent websites like piratebay.se and kat.to into the BitThief. This would then allow the users to add, edit or delete any torrent website configurations and get the torrent search results from the sites, which integrates a meta-search engine into the BitThief in a unique way.

The main goal of this project was to emphasize the concepts of “flexibility” and “power user” which will be discussed later in this report. While the BitThief already had features of any other normal torrent client such as downloading from magnet links and viewing some upload/download statistics, now it also gives its users the ability to perform some amazing operations like executing queries on search results, creating profiles to auto-propose the download of newly created torrents on the pages configured by the user and retrieve any information about a torrent from its torrent page as per the user configurations.

The purpose of this report is to further explain all these new flexible features that were added to the client, why they were added and how they can enhance the user experience.

Background

MOTIVATION

Many websites are out there which host links to torrent files and the cached information like IMDB rating, number of files, file sizes, torrent type and such other details. Whenever a user executes a search on a torrent website, they often find themselves running through the list of search results while giving special attention to the specifics like number of seeders, number of times the file has been downloaded, whether it was uploaded by a verified user, and many such things. Also, some torrent websites might have some torrent type (e.g. movies) categorized better and more informatively than the others (e.g. games). Sometimes, a user might be interested in new or "hot" torrents listed on torrent websites, but only in torrents with some specifics (e.g. size greater than 1 GB or torrent title that contains "YIFY"). The torrent websites are also often unstable with their domains and their website layouts. All these would be frustrating and lacking in details as desired by a power user who wants to save time, get only specific results, automate certain functions and have flexibility. These features are not offered by the torrent websites and most of the torrent clients today. Absence of these features in the existing torrent websites and clients and the needs of a power user were a good motivation to come up with a client that serves as a good search engine, allows users to filter the search results based on queries and also allows users to update torrent websites' URLs as needed. What makes BitThief's new system unique is that it is a fully-configurable system with nothing hard-coded. If a certain torrent website is down or its web content layout changes, users can simply reconfigure BitThief without requiring its developers to release a fixed version.

RELATED WORK

Torrentz.eu is a web-based meta-search engine which indexes torrent results from major torrent websites. "To perform a search, users simply type in the string of keywords and the list of matching torrent files is displayed on screen for the user to choose from."[1] It also allows users to sort the results based on seeders and good or verified uploaders. Selecting the torrent from the result list, then takes the user to another page, listing the torrent websites currently hosting the specified torrent through which a user can view more details and download the file.

[1] Source: https://en.wikipedia.org/wiki/Torrentz

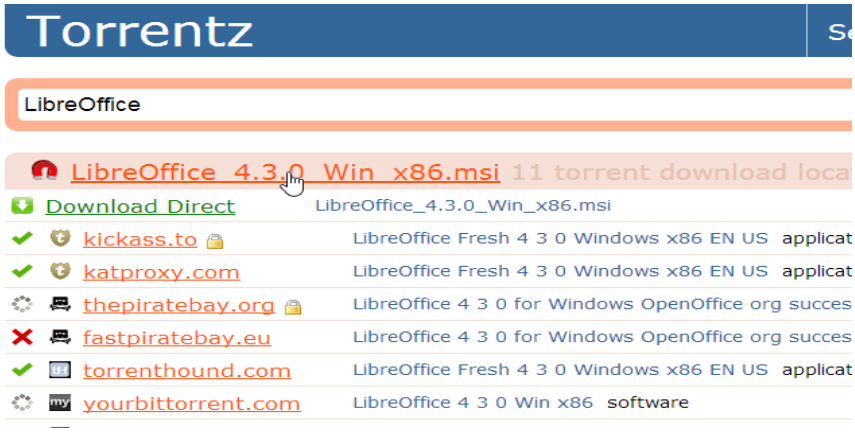

Figure 1: Torrentz displaying the list of external websites hosting the torrent file links[2]

Though it automatically keeps track of which torrent website is down or has its domain name changed, it actually isn't providing its users with any other information on torrents. Every time the user performs the query, they have to go through 3 steps to get to the host torrent website and view more information.

One of the most popular clients, μTorrent, does have a torrent search feature where users are allowed to manage the search providers by entering the search query URLs. Using this feature, users can add, delete or edit a search engine anytime. This prevents the hard-coding of the search engines.

Figure 2: µTorrent's Search Engines Manager[3]

However, it lacks most of the other useful features which were developed in this project and were discussed above in brief.

[2] Source: https://chrome.google.com/webstore/detail/torrentzeu-magnet-izer/kcagpojojhhdoiphfpmipackjjchlboj
[3] Source: http://www.techsupportalert.com/%20utorrent-help-using-utorrent-torrent-search

Design and features

We initially started with an idea to integrate the support for finding and filtering new movie torrents based on the information obtained from IMDB into the existing client. The idea was to retrieve new releases and the popular movies list from IMDB, based on the user settings like minimum IMBD rating, movie genre, and other such data, and perform a meta-search on different torrent websites' search engines to get related torrents. Two basic input fields were used to reach this goal. The user had to configure the torrent website URL and the search query URL format for the websites they wanted, like μTorrent.

The shortcomings to this approach were that there are many types of torrent files available other than movies and that there is no reliable API similar to IMBD available for most of these types which would limit our client to only a few torrent categories. Also, such APIs do not tell us anything about the torrent files that the client retrieves, which users might be interested in, such as number of seeders, quality of the torrents' contents, size of the torrent files and other such data.

We had to come up with another way of retrieving the torrent information from the torrent download page itself. The simple solution to this was to use XPaths, which further opened many other doors for flexibility. The user is simply allowed to enter the XPaths to all the pieces of information they want our client to retrieve from the torrent page. Further, the user may enter queries to filter the results and only display the content they care about.

The backbone behind being able to do all of these lies in our Search Engines Manager tool and the Query Strings.

QUERY STRINGS

A Query String is simply a statement that evaluates to true or false. It is used to filter the torrent results retrieved from the torrent websites. BitThief supports following operators in a Query String:

- is (e.g. title is inception)
- is-not (e.g. title is-not inception)
- contains (e.g. title contains harry)
- not-contains (e.g. title not-contains harry)

- before (e.g. uploaded before “03 december 2015”)
- after (e.g. uploaded after “03 december 2015”)
- on (e.g. uploaded on “03 december 2015”)
- Logic symbols (==, >=, >, <=, <, !=) (e.g. seeders > 300)

It is also possible to group multiple expressions or query statements. For example, `uploaded:year < 2016 AND (size < 3 OR size > 500)`. This will simply return all the torrents which were uploaded before the year 2016 and whose size is either less than 3 units or greater than 500 units.

SEARCH ENGINES

Figure 3 shows the Search Engines Manager.

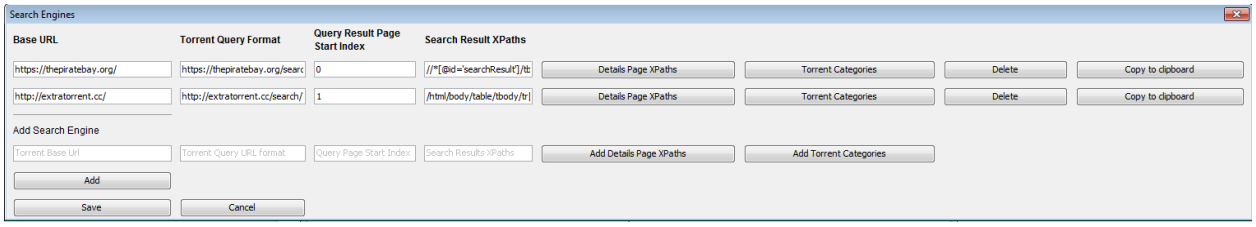

Figure 3: BitThief’s Search Engines Manager

A search engine has following fields:

- **Torrent Website Base URL** (e.g. https://thepiratebay.org): This is simply the torrent website URL

- **Torrent Query Format** (e.g. https://thepiratebay.org/search/<query>/<index>/7/): This is the torrent query result website URL format. To retrieve this, the user needs to go to the site, do a search, replace the search term with “<query>”, and first page number with “<index>” and copy the resulting string to the client.

  These first two fields are enough for carrying out the normal search.

- **Query Result Page Start Index**: Since the number of results would be big most of the time, and all results can’t be displayed on the first page, our results retriever will need a first page index (usually 0 or 1) to automatically jump to the next page after it is done browsing through the current page.

- **Search Results XPaths**: This field is XPath provided by the user to get a list of hyperlinks to all the results on the page.

- **Details Page XPaths**: By "Details Page" we mean torrent file download page. This is the page which contains a magnet link, or .torrent file link and other information like seeders, upload date, torrent size and other such data depending on the website.

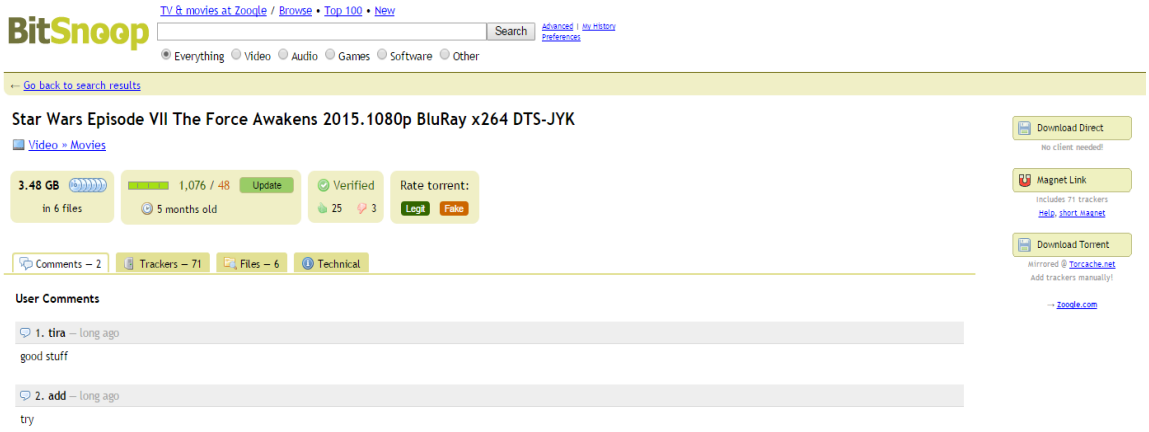

Figure 4: BitSnoop's Torrent Download Page

As shown in figure 4 above, BitSnoop's details page contains some useful information that a user might be interested in. Users can simply enter the XPaths to these fields and name them in their client. These names and information obtained from the XPaths can then be used to perform simple queries (e.g. `Title excludes "HDCam" AND (Seeders > 200 OR Size contains "GiB")`) to filter the search results. One key element added to the XPaths was the support for the regular expressions. As you can see in the figure 4, the file size mentioned is "1.48 GB". The XPath might give us the complete value, but to have a query with number comparisons, the user needs a way to extract the precise data ("1.48") from the string result obtained using the XPath.

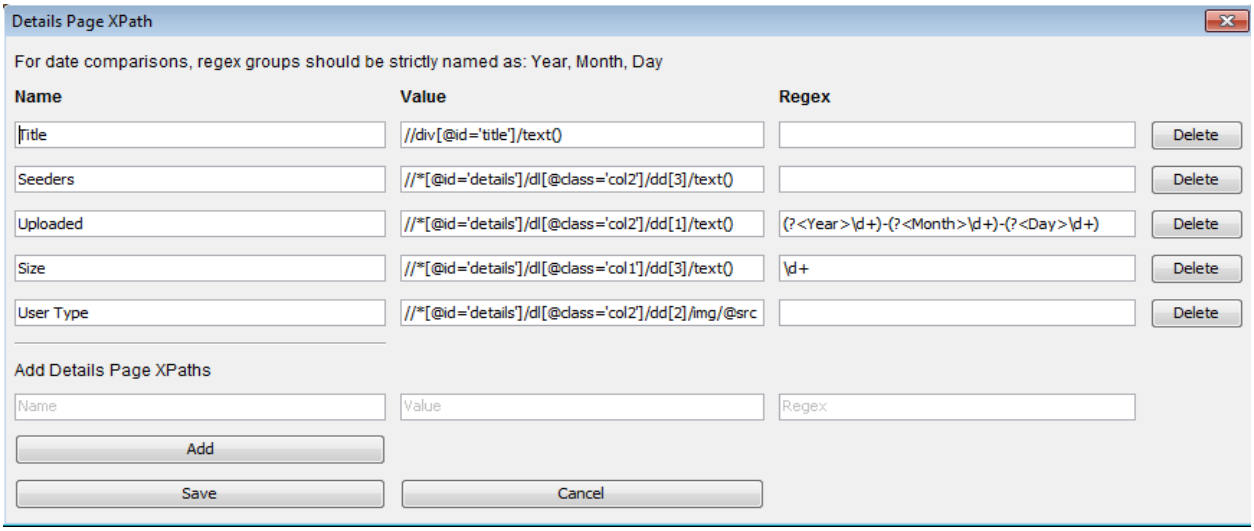

Figure 5: BitThief's Details Page XPath Manager

- **Torrent Categories**: As mentioned before, a torrent website can have multiple torrent categories (e.g., movies, games). Since we were interested in making the BitThief a one stop destination for everything, we needed a way to update its users with newly available torrents in their chosen categories. For this, we came up with the concept of "Profiles" and "Torrent Categories".

  Torrent Categories maintain a list of category names and URLs (e.g. https://kat.cr/movies) and is linked to a search engine.

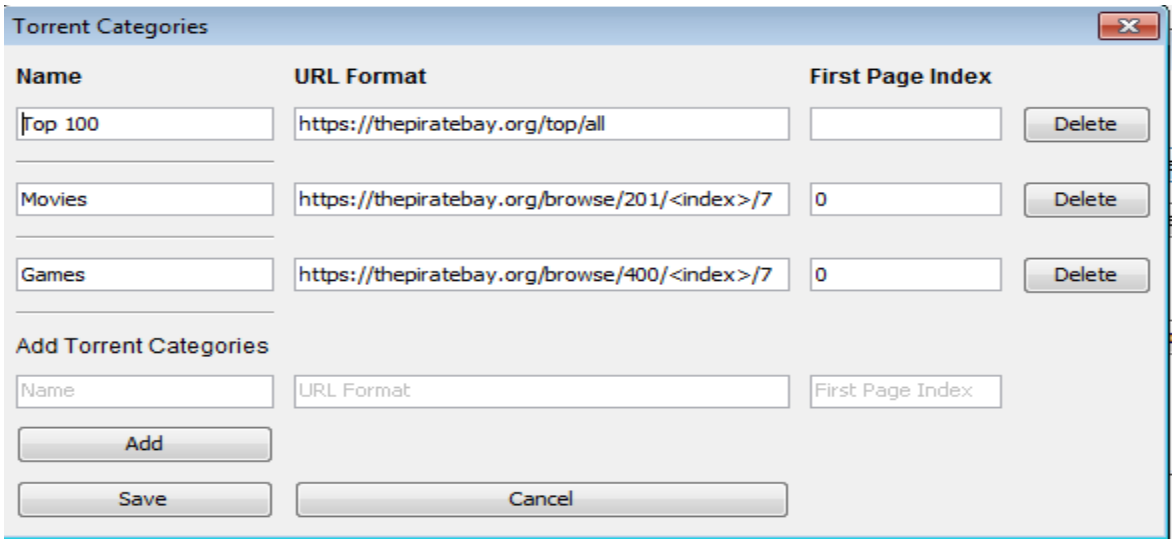

Figure 6: BitThief's Torrent Categories Manager

A profile is simply like a “saved search” containing such categories and query strings that runs every time the BitThief is opened if the profile is active and displays the new available torrents in the chosen categories which match the queries.

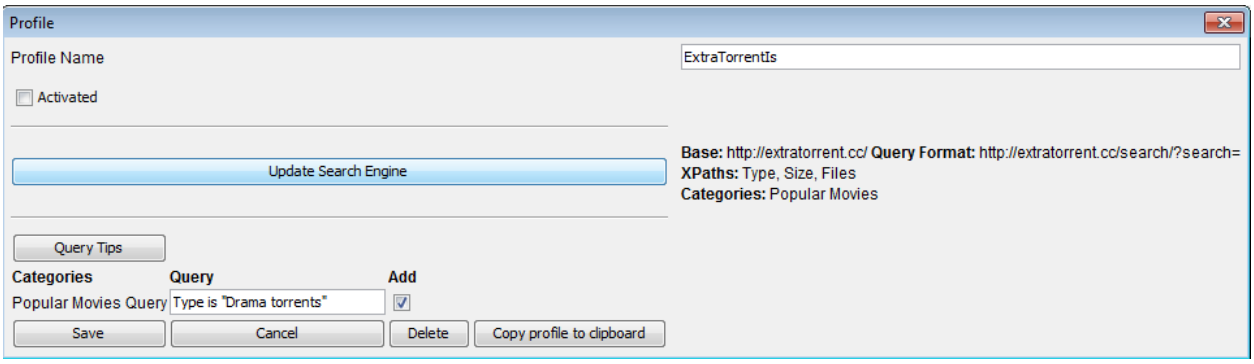

Figure 7: BitThief’s Profile Manager

SEARCH RESULTS

Based on the query string and the XPaths, search results are nicely displayed in the search results and information panel. The name of the XPath is also chosen by the user. BitThief just uses the path value entered by the user to get the information, and displays the name and information obtained in the information panel.

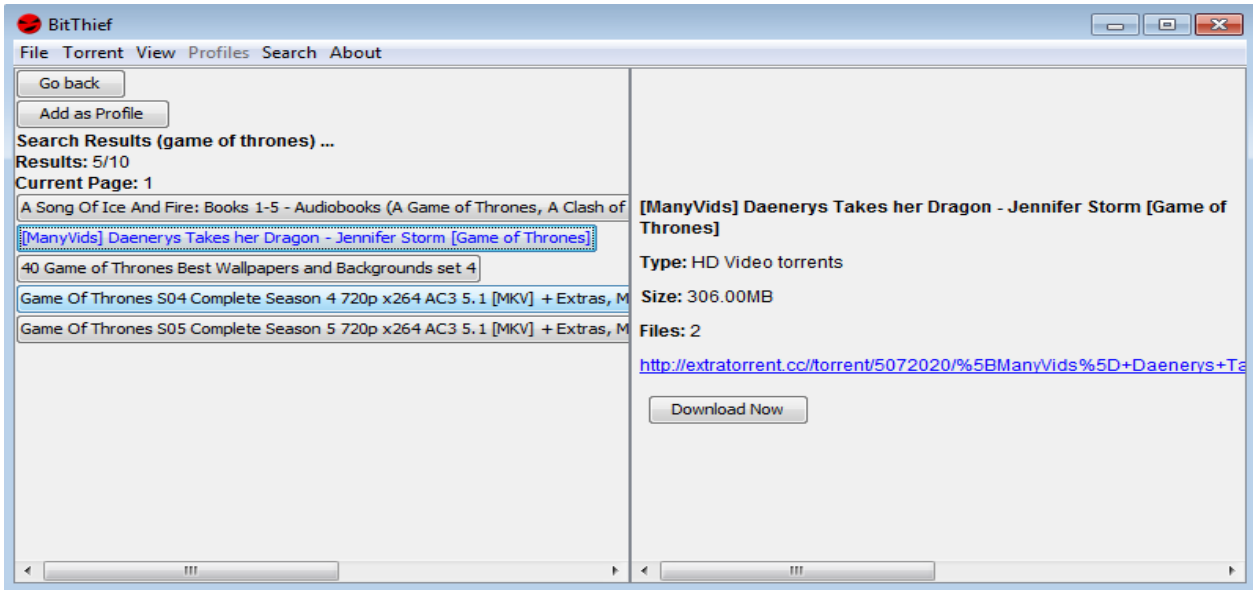

Figure 8: BitThief’s Torrent Search Result Information Panel

If the user has entered the query string and it fails to retrieve the information using the XPath, it simply skips evaluation of entire query string and also the result.

OTHER FEATURES

As a part of this project, we also implemented a few other small, but useful features. Since a search engine can have many fields filled in with complicated XPaths and regular expressions, we support copying a search engine to the clipboard and adding a search engine from the clipboard. A similar feature was implemented for the user profiles. These features blend in really well with the concept of "power users". Using this, smart users and friends can easily share their configurations on social media platforms or forums and exchange it between friends.

Results

With the design and features of BitThief's new system, we were able to meet our goals of providing flexibility and the support to the power users. Below, we explain how these goals were achieved.

FLEXIBILITY

From the figures of search engines and their components above, it is apparent that nearly each and every field is user-editable. Whenever a good torrent website comes out, the user can simply add it. If any torrent website is down or changes its layout, a user can simply remove or edit the URLs and XPaths. Even the profile query strings are kept configurable. The profiles also automatically sync to search engine changes. In this way, we meet our first goal of providing "flexibility".

POWER USERS

BitThief, while it can act as a simple search-and-download tool, it is apparent now that much amazing stuff can be done with it. It does a very simple task: take input from user, retrieve and process the data as user specified, and spit out the results. It does not care whether the data entered by the user is fully correct or not. If a given XPath is wrong, it will simply not return any value. A smart user can further use regular expressions to make the experience even better. For example, https://thepiratebay.org shows a torrent upload date as `2015-08-02 17:02:24 GMT.` While XPath can retrieve this value, regular expressions then allow the user to extract "2015" as year, "08" as month and "02" as day. Using this, they can filter results using simple query strings like "Uploaded after '03 november 2014'".

The behavior could further vary by how the user provides the URLs. For example, for torrent categories, if the user doesn't provide the URL with <index>, BitThief won't automatically hop on to the next page to find more results. Also, a smart user might provide BitThief with the modified link based on his need. Instead of providing the default link https://thepiratebay.org/browse/201/<index>/3 for pirate bay movie torrents, if the user is interested in movies having more seeders, they can rather use the "sort" feature on the website and provide a modified link which is https://thepiratebay.org/browse/201/<index>/7.

We also have a debug dump file saved in the user's home "bitthief" directory where they can easily check out the retrieved page source, XPath information and query filter results. In this way, BitThief also expects the user to be a debugger on their own in case the output isn't as expected. It basically expects its users to be "power users".

Conclusions

Starting with a simple user-configurable meta-search engine, BitThief evolved into two different versions in this project - the current one and an intermediary IMDB-integration version.

Throughout the project many new things were learned. Before coming up with this UI, an intermediary version of BitThief in this project provided only some limited flexibility.

Figure 9: BitThief's Old Profile Manager showing non-editable fields

Users were able to carry out only two functions - adding and deleting. None of the fields were kept editable, which would turn out to be very frustrating as for one XPath change, users will have to recreate an entire search engine entry. It was the same for the profiles as depicted by figure 9.

Also, the naming of components wasn't clear to general users. For example, torrent categories were called "lookups". The dialog boxes had very complicated design with confusing layout and buttons. All of these were pointed out by supervisors and corrected by the end of the term. "To keep the UI simple" was something that I learned in this project.

FUTURE WORK

Even though the current version of BitThief is really robust with regards to flexibility, it still lacks support for good debugging features. Currently, it dumps everything into a single file and it could be painful and frustrating for a user to go over so much of the dumped data to find the information they want. In future, maybe this could be enhanced for the debug files to contain indexed or mapped results for each torrent search request, result or a link. If BitThief is unable to find data through XPath or unable

to connect to the URL, a debug button could be displayed which directly fetches the indexed or mapped data from the debug file, particularly for the current search, result, or a link. The UI can still be improved a lot and made less confusing.

Now that we have a way to filter the torrent results easily, we can bring back the concept of integrating different APIs like IMDB with the BitThief as plugins. A platform could be prepared for 3rd party developers to develop for the BitThief where they can fetch a media list based on media category, rating and other such filters and have a “search torrent” button which also asks user to enter the query string to filter torrent results. In this way, users will be able to keep track of newly released or popular media, search if a “good quality” torrent exists and download it, if required. If certain API goes down, user should be simply able to get rid of that plugin. However, this would only be possible once the BitThief is out there as a famous product and there are developers who are willing to make such small plugins.