THE LIFE OF KENNETH

September 05, 2019

ADDING WEBSEED URLS TO TORRENT FILES

I was recently hanging out on a Slack discussing the deficiencies in the BitTorrent protocol for fast file distribution. A decade ago when Linux mirrors tended to melt down on release day, Bittorrent was seen as a boon for being able to distribute the relatively large ISO files to everyone trying to get it, and the peer-to-peer nature of the protocol meant that the swarm tended to scale with the popularity of the torrent, kind of by definition.

There were a few important points raised during this discussion (helped by the fact that one of the participants had actually presented a paper on the topic):

1. HTTP-based content distribution networks have gotten VASTLY better in the last decade, so you tend not to see servers hugged to death anymore when the admins are expecting a lot of traffic.
2. Users tend to see slower downloads from the Bittorrent swarm than they do from single healthy HTTP servers, with a very wide deviation as a function of the countless knobs exposed to the user in Bittorrent clients.
3. Maintaining Bittorrent seedbox infrastructure in addition to the existing HTTP infrastructure is additional administrative overhead for the content creators, which tends to not be leveraged as well as the HTTP infrastructure for several reasons, including Bittorrent's hesitancy to really scale up traffic, its far from optimal access patterns across storage, the plethora of abstract knobs which seem to have a large impact on the utilization of seedboxes, etc.
4. The torrent trackers are still a central point of failure for distribution, and now the content creator is having to deal with a ton of requests against a stateful database instead of just serving read-only files from a cluster of HTTP servers which can trivially scale horizontally.
5. Torrent files are often treated as second class citizens since they aren't as user-friendly as an HTTP link, and may only be generated as part of releases to quiet the "hippies" who still think that Bittorrent is relevant in the age of big gun CDNs.
6. Torrent availability might be poor at the beginning and end of a torrent's life cycle, since seedboxes tend to limit how many torrents they're actively seeding. When a Linux distro drops fifteen different spins of their release, their seedbox will tend to only seed a few of them at a time and you'll see completely dead torrents several hours if not days into the release cycle.

As any good nerd discussion on Slack goes, we started digging into the finer details of the Bittorrent specification like the Distributed Hash Table that helped reduce the dependence on the central tracker, peer selection algorithms and their tradeoffs, and finally the concept of webseed.

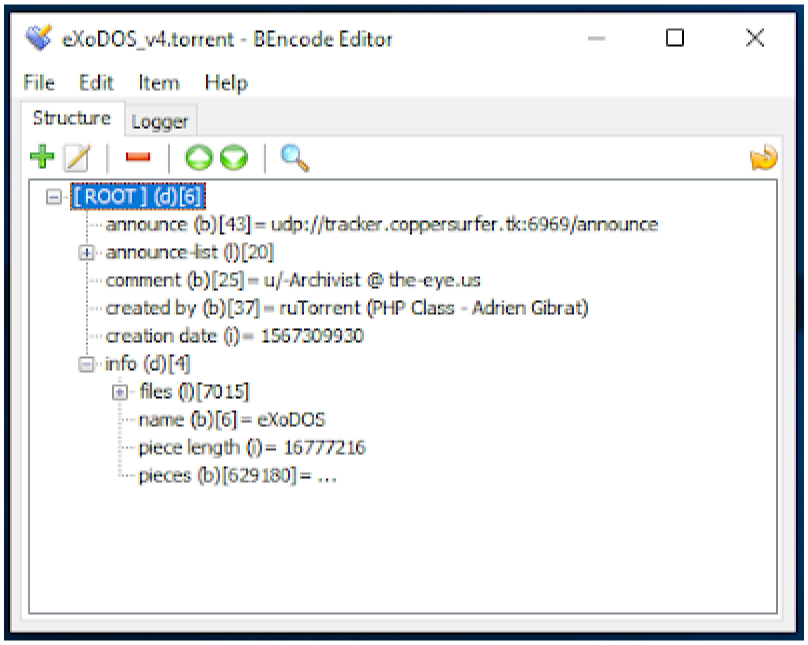
Webseed is a pretty interesting concept which was a late addition to Bittorrent where you could include URLs to HTTP servers serving the torrent contents, to hopefully give you most of the benefits of both protocols; the modern bandwidth scalability of HTTP, and the distributed fault tolerance and inherent scaling of Bittorrent as a function of popularity.

I was aware of webseed, but haven't seen it actually used in *years*, so I decided to dig into it and see what I could learn about it and how it fits into the torrent file structure.

The torrent file, which is the small description database which you use to start downloading all of the actual content of a torrent, at the very least contains a list of the files in the torrent and checksums for each of the fixed-size chunks making up those files. Of course, instead of using a popular object serializer like XML or JSON (which I appreciate might not have really been as popular at the inception of Bittorrent), the torrent file uses a format I've never seen anywhere else called BEncoding.

The BEncoding format is relatively simple; key-value pairs can be stored as byte strings or integers, and the file format supports dictionaries and lists, which can contain sets of further byte strings, integers, or even other lists/dictionaries. Bittorrent then uses this BEncoding format to create a dictionary named "info" which contains a list of the file names and chunk hashes which define the identity of a torrent swarm, but beyond this one dictionary in the file, you can modify anything else in the database without changing the identity of the swarm, including which tracker to use as "announce" byte-strings, or "announce-list" lists of byte-strings, comments, creation dates, etc.

Fortunately, the BEncoding format is relatively human readable, since length fields are encoded as ASCII integers, field delimiters are characters like ':', 'l', and 'i', but unfortunately this is all encoded as a single line with no breaks, so trying to edit this database by hand with a text editor might be a little hairy.



```
[ ROOT ] (d)[6]
  announce (b)[43] = udp://tracker.coppersurfer.tk:6969/announce
  announce-list (l)[20]
  comment (b)[25] = u/-Archivist @ the-eye.us
  created by (b)[37] = ruTorrent (PHP Class - Adrien Gibrat)
  creation date (i) = 1567309930
  info (d)[4]
    files (l)[7015]
    name (b)[6] = eXoDOS
    piece length (i) = 16777216
    pieces (b)[629180] = ...
```

I wasn't able to find a tremendous amount of tooling for interactively editing BEncode files; there exists a few online "torrent editors" which give you basic access to changing some of the fields which aren't part of the info dictionary, but none of them seemed to give the arbitrary key-value editing capabilities I needed to play with webseed, so I settled on a Windows tool called BEncode Editor. The nice thing about this tool is that it's designed as an arbitrary BEncode editor, instead of specifically a torrent editor, so it has that authentic "no training wheels included" hacker feel to it. User beware.

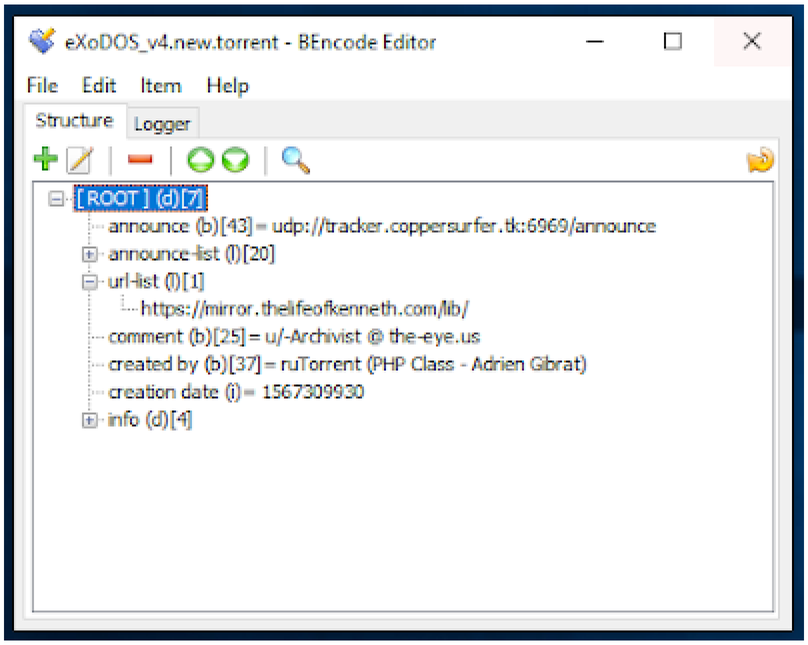
As an example, I grabbed the torrent file for the eXoDOS v4 collection, which is a huge collection of 7000 DOS games with various builds of DOSBOX to make it all work on a modern system. Opening the torrent file in BEncode Editor, you can see the main info dictionary at the end of the main root dictionary, which is the part you don't want to touch since the info dictionary is what defines the identity of the torrent. In addition to that, you can see five other elements in the root dictionary, including a 43 byte byte string named "announce" which is a URI to a primary tracker to use to announce yourself to the rest of the swarm, a list of 20 elements named "announce-list" which is alternative trackers to use (the file likely contains both the single tracker and a list of trackers for backwards compatibility for Bittorrent clients which predate the concept of announce-lists?) and some byte strings labeled "comment", "created by", and an integer named "creation date", which looks like a Unix timestamp.

Cool! So at this point, we have an interactive tool to inspect and modify a BEncode database, and know which parts to not touch to avoid breaking things (The "info" dictionary).

Now back to the original point of somehow adding webseed URLs to a torrent file

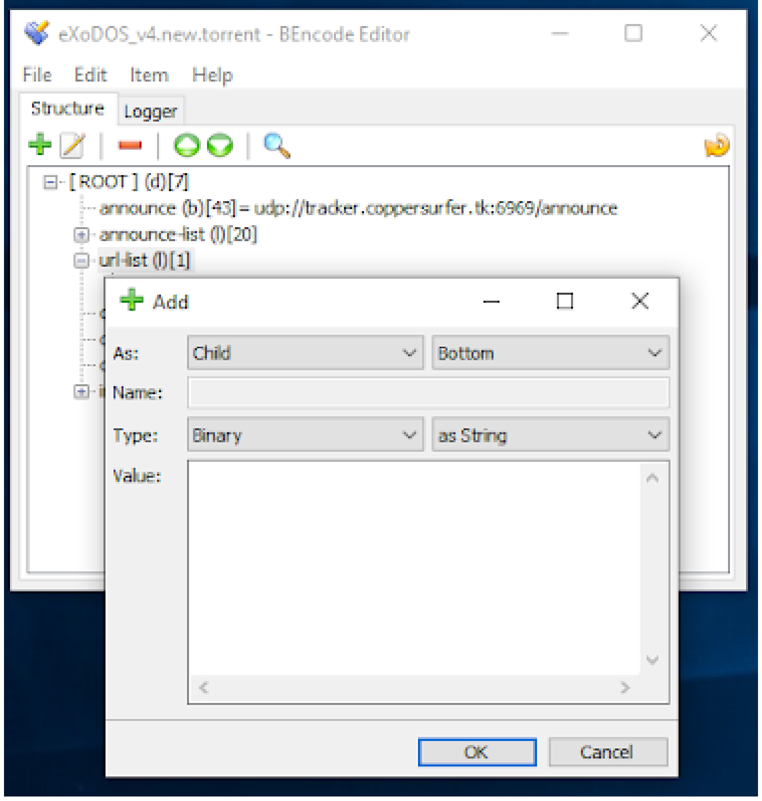
Webseeding is defined in Bittorrent specification BEP_0019, which I didn't find particularly clear, but the main takeaway for me is that to enable webseeding, I just need to add a list to the torrent named "url-list", and then add byte-string elements to that list which are URLs to HTTP/FTP servers serving the same contents.

So first step, log into one of my web servers and download the torrent and throw the contents in an open directory. (In my case, https://mirror.thelifeofkenneth.com/lib/) For actual content creators, this should be part of their normal release workflow for HTTP hosting of the content, so this is only really needed for when you're retrofitting webseed into an existing torrent.



```
[ROOT] (d)[7]
  announce (b)[43] = udp://tracker.coppersurfer.tk:6969/announce
  announce-list (l)[20]
  url-list (l)[1]
    https://mirror.thelifeofkenneth.com/lib/
  comment (b)[25] = u/-Archivist @ the-eye.us
  created by (b)[37] = ruTorrent (PHP Class - Adrien Gibrat)
  creation date (i) = 1567309930
  info (d)[4]
```

Now we start editing the torrent file, by adding a "url-list" list to the root dictionary, and the part I found a little tricky was figuring out how to add the byte-string child to the list, which is done in BEncode Editor by clicking on the empty "url-list" list, and clicking "add" and specifying that the new element should be added as a "child" of the current element.


eXoDOS_v4.new.torrent - BEncode Editor

[ROOT] (d)[7]
announce (b)[43] = udp://tracker.coppersurfer.tk:6969/announce
announce-list (l)[20]
url-list (l)[1]

Add
As: Child, Bottom
Name:
Type: Binary, as String
Value:

Referring back to BEP_0019, if I end the URL with a forward slash, the client should append the info['name'] to the URL, so the binary string I'm adding as a child to the list is "https://mirror.thelifeofkenneth.com/lib/" such that the client will append "eXoDOS" to it, looking for the content at "https://mirror.thelifeofkenneth.com/lib/eXoDOS/", which is correct.

Save this file as a new .torrent file, and success! Now I have a version of the eXoDOS torrent with the swarm performance supplemented by my own HTTP server! The same could be done for any other torrent where the exact same content is available via HTTP, and honestly I'm a little surprised that I don't tend to see Linux distros using this, since it reasonably removes the need for them to commit to maintaining torrent infrastructure since the torrent swarm can at least survive off of an HTTP server, which the content creator is clearly already running.

Share

POPULAR POSTS

November 15, 2017

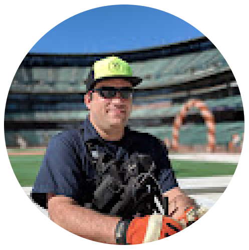
CREATING AN AUTONOMOUS SYSTEM FOR FUN AND PROFIT

Share

May 09, 2023

BUILDING THE MICRO MIRROR FREE SOFTWARE CDN

Share



Kenneth Finnegan

VISIT PROFILE

Archive