Supported Features

Parsing and respecting all documented metainfo file entries (with the exception of md5sum and encoding).
Support for all documented tracker response fields, including compact/binary model peers response.
Private torrents are respected, if only because PEX peer exchange, DHT, etc are unsupported.

Optimistic unchoking and rarest-first are implemented.

I have chosen **not** to support magnet links.
I have chosen **not** to support lazy bitfields. I just send the whole thing.
I have chosen **not** to support anti-snubbing, super seeding, or any Fast Extensions.

Design and Implementation Choices

I opted to implement my client in Ruby for several reasons. The most important ones are:

1. Familiarity; I've used Ruby for many a school project. The reason being,
2. The Standard Library; Many languages have thriving communities that create wonderful and convenient libraries for common use cases. For school projects however, this is often undercut by the fact that external libraries are forbidden. Ruby is somewhat unique in that its built-in standard library covers most of what I would normally reach to an external dependency to accomplish. In fact the only time I've found myself wishing it supported something that it did not was during this project for dealing with BEncoded data.

For the sake of simplicity and my sanity I've decided to implement a CLI UI and support only a single active torrent.

I generate peer_ids in the form ["-du-0001" <12 random bytes>]. I chose this form, "Azureus-style", as it seems to be the most common way of encoding client type info into peer_id. I don't have a real reason to need to do this, but do anyway, simply to allign my implementation more closely with others.

I store incomplete and seeding torrent data in its own directory. Each piece is a separate file, again for simplicity. I have not bothered to implement seeding from completed data, so this pieces directory must continue to exist to keep seeding after a download is complete.

Problems Encountered (and their Solutions)

Many parts of the bittorrent specification are left up to interpretation.
Fortunately, there are many notes and annecdotes on https://wiki.theory.org/index.php/BitTorrentSpecification
that shed some light on how one ought to implement the protocol in practice.

Most problems encountered where typos. Gotta love weakly typed languages.

Known Bugs

This client does not make any attempt to deal with changing network conditions. If the host IP address changes, TCP connections will time out and connections to peers will end ungracefully.

My experiements show that download speed is only around a third what standard clients get. I had to play with the algorithm a bunch to even get it to that point.

Contributions

There's only one contributer.