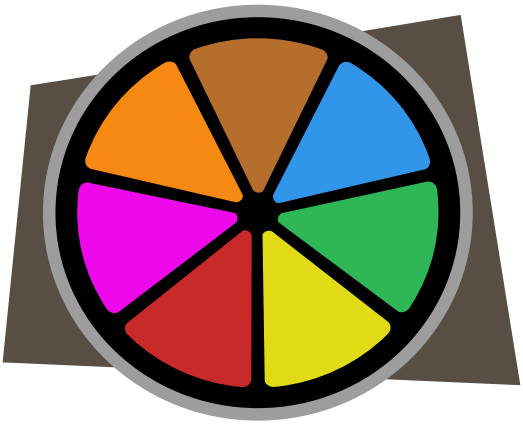
Trivial Torrent Specification

Xarxes, Grau d'Enginyeria Informàtica



Document version: 1.0
Date: Jan. 2020

Universitat Autònoma de Barcelona

Departament d'Enginyeria de la Informació i de les Comunicacions

Document Information

Document identifier: trivial_torrent_specification.md
Title: Trivial Torrent Specification
Number of pages: 3
Date: Jan. 2020
Author(s): Ian Blanes

Change Control

Revision	Date	Changes
1.0	Jan. 2020	(initial version)

Contents

1 Introduction

The trivial torrent protocol is a highly simplified version of the bittorrent protocol [`https://en.wikipedia.org/wiki/BitTorrent`]. This document describes the protocol as well as the supporting binary formats.

2 Overview

The trivial torrent protocol is a peer-to-peer communications protocol designed to share files among multiple peers. In the trivial torrent protocol, a peer can be a *client* peer or a *server* peer, but not both at once. Servers listen to a single TCP port and make available a single file for clients to download. This file is referred to as the *downloaded file*, regardless of whether it is currently being downloaded, uploaded, or simply stored on disk.

2.1 Blocks

For the purposes of the protocol, files are split into consecutive *blocks* of data, which are transmitted from servers to clients. All blocks are of 64 KiB, except possibly for the last one, which may be smaller. Blocks are indexed, starting from block 0, in order of occurrence within the file.

When serving a file, it is not necessary for a server to have all blocks comprising that file. A single file may be split among multiple servers, and clients are expected to connect to multiple servers and download the blocks that are needed.

For example, file `"A"` has a length of 65537 Bytes. Hence, it has two blocks: block 0 comprises bytes from 0 to 65535 (included), and block 1 comprises just one byte (at file offset 65536).

A block always contains at least one byte. Hence, empty files (i.e., that have a length of 0 bytes) have 0 blocks.

2.2 Metainfo file

A *metainfo file*, with extension `".ttorrent"` describes the information necessary to share a file for both clients and servers. Metainfo and downloaded files share their name, except for the additional `".ttorrent"` extension of metainfo files.

For example, downloaded file `"A"` is associated with metainfo file named `"A.ttorrent"`, while downloaded file `"name.txt"` is associated with metainfo file named `"name.txt.ttorrent"`.

A metainfo file contains, among others:

- The downloaded file length.
- The SHA256 hash of each block [`https://en.wikipedia.org/wiki/SHA-2`].
- A list of address of multiple server peers from which this file can be downloaded.

2.3 Communication process

2.3.1 Clients

A simple client of the trivial torrent protocol follows this procedure:

1. Load a metainfo file (functionality is already available in the file_io API).
   a. Check for the existence of the associated downloaded file.
   b. Check which blocks are correct using the SHA256 hashes in the metainfo file.
2. For each server peer in the metainfo file:
   a. Connect to that server peer.
   b. For each incorrect block in the downloaded file (the hash does not match in 1b).
      i. Send a request to the server peer.
      ii. If the server responds with the block, store it to the downloaded file.
      iii. Otherwise, if the server signals the unavailablity of the block, do nothing.
   c. Close the connection.
3. Terminate.

2.3.2 Servers

A simple server of the trivial torrent protocol follows this procedure:

1. Load a metainfo file (functionality is already available in the file_io API).
   a. Check for the existence of the associated downloaded file.
   b. Check which blocks are correct using the SHA256 hashes in the metainfo file.
2. Forever listen to incoming connections, and for each connection:
   a. Wait for a message.
   b. If a message requests a block that can be served (correct hash), respond with the appropriate message, followed by the raw block data.
   c. Otherwise, respond with a message signaling the unavailability of the block.

3 Wire format

Peers exchange only one type of message. A message consists of four tightly packed fields as follows.

```
uint32_t magic_number;
uint8_t message_code;
uint64_t block_number;
uint8_t payload[payload_size];
```

All fields shall be transmitted in network byte order.

The `magic_number` field shall always be set to the constant `MAGIC_NUMBER` available in the template source code [`https://en.wikipedia.org/wiki/Magic_number_(programming)#In_protocols`].

The `message_code` field shall always be set to `MSG_REQUEST` by client peers, and either to `MSG_RESPONSE_OK` or `MSG_RESPONSE_NA` by server peers, whether the requested block is available or not, respectively.

```
static const uint8_t MSG_REQUEST = 0;
static const uint8_t MSG_RESPONSE_OK = 1;
static const uint8_t MSG_RESPONSE_NA = 2;
```

The `block_number` field shall be set to the requested block by client peers, and shall be copied into the `block_number` field of their responses by server peers.

For messages with `message_code` set to `MSG_REQUEST` or `MSG_RESPONSE_NA`, the `payload` field shall be empty and `payload_size` shall be zero. Hence, messages with `message_code` set to `MSG_REQUEST` or `MSG_RESPONSE_NA`, have a constant size of 13 Bytes.

For messages with `message_code` set to `MSG_RESPONSE_OK`, the `payload` field shall contain the data block with index `block_number`.

4 Disk format

The contents of `".ttorrent"` metainfo files is as follows.

1. The SHA256 hash of the whole downloaded file (as an array of 32 bytes).
2. The downloaded file size (as a 64-bit unsigned integer).
3. The number of server peers in the metainfo file (as a 64-bit unsigned integer).
4. For each file block:
   a. The SHA256 hash of that block (as an array of 32 bytes).
5. For each server peer:
   a. The server peer IP address (as an array of 4 bytes).
   b. The server peer port (as a 16-bit unsigned integer).

All fields shall be stored in network byte order.