Задание

Разработать приложение для операционных систем семейства Windows или Linux, обеспечивающие функции клиента протокола BitTorrent.

Основные возможности

Приложение должно реализовывать следующие функции:

1. Задание пути к скачиваемому .torrent файлу.
2. Открытие и разбор .torrent файлов.
3. Задание пути, по которому должен располагаться скачиваемый файл.
4. Получение списка пиров (участников обмена) с трекера по протоколу HTTP.
5. Задание порта, который будет прослушиваться для обмена информацией с пирами.
6. Скачивание файла с других BitTorrent-пиров в соответствии со спецификацией BEP-0003.
7. Раздача скачанного файла.

Полезные ссылки

[1] http://www.bittorrent.org/beps/bep_0000.html - Формальное описание протокола.
[2] https://wiki.theory.org/BitTorrentSpecification - Описание деталей для реализации протокола
[3] http://www.kristenwidman.com/blog/33/how-to-write-a-bittorrent-client-part-1/ - Первая часть подробного описания реализации BitTorrent клиента.
[4] http://www.kristenwidman.com/blog/71/how-to-write-a-bittorrent-client-part-2/ - Вторая часть подробного описания реализации BitTorrent клиента.
[5] https://ru.wikipedia.org/wiki/Bencode - Описание формата Bencode.
[6] http://torrenteditor.com/ - Online парсер .torrent файла.
[7] https://www.wireshark.org/ - Wireshark для анализа траффика.

Реализация протокола

Реализацию скачивания и раздачи файлов по протоколу BitTorrent условно можно разделить на 4 части:

1. Парсер .torrent файла.
2. Анализ загружаемых файлов.
3. Получение списка пиров от трекеров.
4. Получение частей по протоколу BitTorrent.

Парсер .torrent файла

.torrent файл использует формат Bencode [5]. Этот формат поддерживает следующие типы данных:

1. Строка байт (<размер>:<содержимое>) - Размер – это положительное число в десятичной системе счисления, может быть нулём; Содержимое – это непосредственно данные, представленные цепочкой байт, которые не подразумевают никакой символьной кодировки. Для реализации лучше всего использовать тип String в Java или std::string в C++.
2. Целое число (i<число в десятичной системе счисления>e) - Число не должно начинаться с нуля, но число ноль записывается как i0e. Отрицательные числа записываются со знаком минуса перед числом. Для реализации лучше всего использовать тип long в Java или тип __int64 в C++.

3. Список (l<содержимое>e) - Содержимое включает в себя любые Bencode типы, следующие друг за другом. Для реализации лучше всего использовать типы ArrayList, Array в Java или тип std::vector в C++.
4. Словарь (d<содержимое>e) - Содержимое состоит из пар ключ-значение, которые следуют друг за другом. Ключи могут быть только строкой байт и должны быть упорядочены в лексикографическом порядке. Для реализации лучше всего использовать тип HashMap в Java или тип std::map в C++.

Можно использовать готовый парсер формата Bencode, однако для удобства работы лучше придерживаться вышеописанных реализаций структур данных. Важным моментом является использование 64-битных целых чисел, потому что некоторые целые значения не поместятся в 32 бита.

Поля .torrent файла описываются в разделе «Metainfo File Structure» в описаниях деталей протокола [2]. Большинство из этих полей являются необязательными и могут не обрабатываться. Самыми важными являются следующие поля:

1. Словарь info – Словарь от которого рассчитывается sha-1 хэш (впоследствии info hash).
2. Строка с ссылкой на основной трекер – Основная ссылка для получения пиров (обычно используются протоколы udp:// и http://).
3. Список ссылок на запасные трекеры - Дополнительные ссылки для получения пиров (обычно используются протоколы udp:// и http://).

Словарь info может быть в двух режимах (одиночный файл, или множественные файлы). В обоих режимах словарь содержит следующие обязательные поля:

1. piece length – Размер каждой части, на которые разбиваются загружаемые файлы, в байтах.
2. pieces – Sha-1 хэши от каждой части, на которые разбиваются загружаемые файлы. Размер каждого sha-1 хэша 20 байт. Таким образом количество частей в 20 раз меньше количества байт в этом поле.

Если файл одиночный, то словарь info содержит следующие обязательные поля:

1. name – Имя одиночного файла.
2. length – Размер одиночного файла в байтах.

Если файлов несколько, то словарь info содержит следующие обязательные поля:

1. name – Имя директории в которой содержатся все файлы.
2. files – Список с описанием всех файлов. Каждое описание - это словарь со следующими обязательными элементами: path – список, каждый элемент которого это вложенные директории файла, а последний элемент - это название файла; length – размер файла в байтах.

Количество частей должно равняться pieces count = ceil(summary length / piece length), где ceil – функция округления в большую сторону.

Анализатор загружаемых файлов

После распознавания основных полей .torrent файла известны размеры и пути к каждому загружаемому файлу. Анализатор должен создать пустые файлы соответствующих размеров, если файл еще не содержится в указанной директории. Если файлы уже содержатся, то необходимо произвести сравнение sha-1 хэш сумм каждой части файла с указанными суммами в графе pieces словаря info. Для этого также необходимо правильно разбить файлы на части. Файлы делятся на части в том же порядке, в котором они представлены в словаре info. Каждая часть нумеруется с 0, имеет размер, указанный в графе piece length словаря info. Часть может быть растянута на

несколько файлов (если она находится на их стыке), а также последняя из частей может быть меньше размера, указанного в графе piece length.

Также разумно, чтобы анализатор предоставлял интерфейс для записи в файл по индексу части (для скачивания файлов), а также интерфейс для получения из файла байтов конкретной части (для раздачи).

Получение списка пиров от трекеров

Трекеры могут быть реализованы по UDP или HTTP протоколам. UDP трекеры далее рассматриваться не будут.

Для анализа траффика общения с трекером (а также общения с пирами) удобно воспользоваться программой Wireshark [7]. Для этого необходимо запустить любой торрент клиент и запустить Wireshark.

Некоторые трекеры поддерживают scrape конвенцию, которая описывается в разделе «Tracker 'scrape' Convention» в описаниях деталей протокола [2]. Поддержка scrape конвенции является необязательной и далее рассматриваться не будет.

Список пиров возвращается в формате Bencode с основного и дополнительных трекеров, ссылки на которые располагаются в полях announce и announce-list .torrent файла. Для запроса к трекеру используется HTTP GET запрос. Параметры HTTP GET запроса описываются в разделе «Tracker Request Parameters» в описаниях деталей протокола [2]. Важнейшими параметрами запроса являются:

1. info_hash – sha-1 хэш от словаря info .torrent файла.
2. peer_id – 20-байтный идентификатор пира. Обычно начинается с -xxnnnn-, где xx это код программы, а nnnn это версия программы.
3. key – 8-байтный случайный ключ, также используется для идентификации пира.
4. event – При первом подключении имеет значение 'started', при последующих не указывается, при внезапном отключении – 'stopped', при завершении скачивания – 'completed'.
5. port – Порт для подключения других пиров.

Формат ответа трекера описывается в разделе «Tracker Response» в описаниях деталей протокола [2]. Важнейшие поля ответа – это интервал для повторных обращений к трекеру, а также список пиров, участвующих в раздаче. Обычно список пиров указывается в компактном формате (6 байт на пира: 4 байта на ip адрес, 2 байта на порт).

Запросы к трекерам должны повторяться с определенным интервалом (обычно указывается в ответе). После каждого ответа, список пиров пополняется.

Получение частей по протоколу BitTorrent

Перед осуществлением обмена, необходимо выделить несколько незавершенных частей (количество влияет на использование оперативной памяти) и разделить их на блоки. Блоки обычно имеют размер 16 килобайт и характеризуются смещением от начала части. Количество блоков в части равно blocks count = ceil(piece length / block length) , где ceil – функция округления в большую сторону. Количество блоков в каждой части фиксированное, кроме количества блоков в последней части. Последняя часть обычно содержит меньше блоков, чем остальные. После того, как все блоки части завершены, они должны быть объединены в часть, а после этого должна быть проверена sha-1 хэш сумма части. Если сумма совпадает, то часть можно записать в файл.

Обмен между пирами описывается в разделе «Peer wire protocol» в описаниях деталей протокола [2].

Для связи между пирами используется свой протокол, работающий поверх TCP. Каждый

пир должен как слушать порт, так и подключаться к другим пирам, то есть работать и как сервер, и как клиент. После установления соединения обе стороны отправляют сообщение-рукопожатие (handshake). После этого пиры могут отправлять друг другу любые команды. Сообщения имеют следующий формат:

1. Длина сообщения (не считая этого заголовка), 4 байта.
2. Тип сообщения, 1 байт.
3. Данные (payload, если есть).

Список поддерживаемых команд:

1. Handshake – Начальное сообщение рукопожатия.
2. Keep alive – Сообщение для проверки достижимости пира.
3. choke (тип 0) – Прекратить обмен с пиром.
4. unchoke (тип 1) – Разрешить обмен с пиром.
5. interested (тип 2) – Ожидаются запросы от пира.
6. not interested (тип 3) – Запросы от пира не ожидаются.
7. have (тип 4) – Посылается при получении очередного блока.
8. bitfield (тип 5) – Имеющиеся у пира части файла.
9. request (тип 6) – Запросить блок.
10. piece (тип 7) – Запрашиваемый блок.
11. cancel (тип 8) – Отмена запрашиваемого блока.
12. port (тип 9) – Порт для DHT трекера (необязательно).
13. extended (тип 20) – Различные расширения в формате Bencode (необязательно).

После рукопожатия пиры обмениваются сообщениями bitfield, в котором они сообщают, какие части файла у них имеются. По умолчанию считается, что у пира нет никаких частей.

Пока от пира не поступит сообщение unchoke, обмен запрещен, поэтому после рукопожатия обычно отправляется сообщение interested. В ответ пир может отправить unchoke. После того, как пиры разблокировали друг друга, можно запрашивать блоки.