**Содержание**

Введение

1. Теоретические сведения
   1. Общие сведения
   2. Файлы .torrent
   3. Трекеры
   4. Соединение между пирами
   5. Диалог между пирами
2. Обзор аналогов
3. Определение и анализ требований
   1. Назначение разработки
   2. Выполняемые функции
   3. Входные данные
   4. Выходные данные
   5. Технические требования
   6. Программные требования
4. Постановка задач
5. Разработка программного средства

**Введение**

В наш век бурного развития информационных технологий появление нового решения в одной области вызывает скачок остальных. Растет скорость обработки данных, их объем, скорость передачи. Одним из локомотивов прогресса являются технологии сетевого обмена.

Классической стала схема с сервером в роли источника и приемника информации. Через один узел сети проходит основная масса данных, что делает его узким местом и необходимым звеном для функционирования всей сети. Поэтому была разработано альтернативное решение – модель p2p (peer-to-peer, равный к равному), в которой функции сервера распределяются между узлами – участниками обмена.

Торрент (или BitTorrent) - протокол для обмена данными по сети, разработанный в 2001 году Брэмом Коэном. Следуя модели p2p, он оптимизирует обмен путем минимизации роли сервера. Отныне сервер выполняет чисто справочную роль, а основная нагрузка по хранению и передаче данных делится между пользователями. После загрузки файла узел сам может стать его источником, т.е. сеть динамична и изменяется самостоятельно. Согласно данным BitTorrent Inc., в 2012 году количество пользователей торрент-сетей достигало 150 млн человек.

Основные функции процессов передачи выполняются программой на стороне каждого пользователя – торрент-клиентом. Он выполняет обращения к серверу и диалог с другими пользователями , обеспечивает целостность и корректность передаваемых данных.

В данной работе будет разработан торрент-клиент, способный функционировать в реальной сети обмена. В частности, будет реализована функция загрузки файлов на основании их .torrent описаний.

* 1. **Теоретические сведения**
  2. **Общие сведения**

Протокол BitTorrent снижает нагрузку на сервер и сеть при передаче больших файлов. Вместо централизованного скачивания пользователи включаются в «улей», в котором могут загружать и отдавать данные друг другу. Протокол не требует непрерывности процесса и может работать в низкоскоростных сетях. Также он предотвращает локальные скачки интернет-трафика.

Пользователь, который хочет поделиться данными, создает их .torrent-описание и распространяет его (через трекер, по почте и пр. методами). Затем с помощью торрент-узла (клиента) файл делается доступным по сети, и пользователь становится «сидом» (*seed*). Получившие .torrent-дескриптор могут загрузить его в свои клиентские приложения, чтобы стать «пирами» (*peer*) или «личерами» (*leecher*), и затем связаться с сидом (или другими пирами) для скачивания целевого файла.

Распространяемый файл делится на сегменты (*pieces*) одинакового размера (кроме последнего). Когда пир получил сегмент файла, он сам становится источником для других пиров (делится нагрузка на сид).

Каждый сегмент защищен криптографическим ключом, хранимым в торрент-файле. Это позволяет узлам проверять части и целые файлы на корректность.

Сегменты загружаются неупорядоченно, а затем сохраняются в соответствующие им места назначения торрент-клиентом, который следит за статусом каждого сегмента. Это позволяет приостанавливать скачивание на произвольный промежуток времени, в котором клиент может быть неоднократно перезапущен. Также такой подход дает возможность загружать несколько сегментов одновременно.

Когда пир полностью загрузил целевой файл, он сам становится сидом. Таким образом, скорость скачивания и количество пиров растут лавинообразно (до некоторых пределов).

* 1. **Файлы .torrent**

.torrent файл – дескриптор, хранящий названия, пути и размеры целевых файлов, хэш-суммы их сегментов, размер сегментов, адреса трекеров и данные о создавшем их торрент-клиенте.

Информация в файле закодирована в Bencode – специализированным форматом, используемым протоколом BitTorrent. Формат включает типы данных: целые числа, строки, списки и словари.

Запись целого числа:

*i<значение>e*

Запись строки:

*<длина строки>:<значение>*

Запись списка:

*l<элемент 1><элемент 2><..>e*

Запись словаря:

*d<строка – имя элемента 1><элемент 1><..><..>e*

Обычный .torrent файл представляет собой иерархическую структуру из вышеописанных элементов. Корневым элементом является словарь, включающий в себя элементы:

* Info – описание целевого файла/файлов и их хеш-сумм.
* Announce – адрес трекера.
* Announce-list (необяз.) – список адресов трекеров.
* Creation date (необяз.) – время создания дескриптора.
* Comment (необяз) – ссылка на сообщение трекера, в котором был опубликован данный файл.
* Created by (необяз.) – название и версия клиента, создавшего дескриптор.
* Encoding (необяз.) – кодировка хэш-сумм из элемента Info.

Словарь Info может иметь различный вид, в зависимости от количества целевых файлов.

* 1. **Трекеры**

Трекер (tracker) — специализированный [сервер](http://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%B5%D1%80_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5)), нужный для того, чтобы клиенты могли найти друг друга. Фактически, на трекере хранятся [IP-адреса](http://ru.wikipedia.org/wiki/IP-%D0%B0%D0%B4%D1%80%D0%B5%D1%81), входящие порты клиентов и [хэш-суммы](http://ru.wikipedia.org/wiki/%D0%A5%D0%B5%D1%88%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5" \o "Хеширование), уникальным образом идентифицирующие объекты, участвующие в закачках. По стандарту, имена файлов на трекере не хранятся, и узнать их по хэш-суммам нельзя. Однако на практике трекер часто помимо своей основной функции выполняет и функцию небольшого [веб-сервера](http://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80). Такой сервер хранит файлы метаданных и описания распространяемых файлов, предоставляет статистику закачек по разным файлам, показывает текущее количество подключённых пиров и пр.

Классический торрент-трекер работает на протоколе HTTP и имеет соответствующий адрес (*http://*). Клиенты посылают ему GET запросы. Основные параметры:

* Info\_hash – хэш-сумма (по алгоритму SHA1) секции Info торрент-файла, длиной 20 байт. Значение должно быть экранировано в соответствии с требованиями к HTTP-запросу.
* Peer\_id – идентификатор клиента, хранящий сведения о названии, версии (для стандартных клиентских приложений).
* Port – порт клиента, на котором он будет слушать обращения от других клиентов.
* Event - событие, статус. Может сообщать серверу о начале, остановке и завершении загрузки данным клиентом.

Ответ сервера закодирован в формате Bencode. Он может содержать либо статистику по целевому файлу и список пиров (идентификатор, IP-адрес, порт), либо текст ошибки.

В настоящее время все более распространяется обращение к трекерам по протоколу UDP. В этом случае адрес сервера будет начинаться с ‘*udp://*’ и содержать номер порта. UDP – легковесный протокол, обладающий высокой скоростью передачи сообщений. Однако, так как он не устанавливает соединение, как TCP, то не может гарантировать доставку сообщения (требуется перепосылка при отсутствии ответа).

Для клиента и сервера определены по 3 типа сообщений: соединение (*connection*), анонсирование (*announce*) и так называемые *scrape*-запросы. Каждое сообщение от клиента содержит некоторый идентификатор (*Transaction ID*), дублируемый сервером при ответе, по которому клиент сможет определить соответствие «запрос - ответ».

Connection шлется клиентом для получения идентификатора соединения (*Connection ID*), который будет использоваться в остальных двух запросах. Announce возвращает статистику и список пиров (IP- адрес, порт), а Scrape – только статистику.

* 1. **Соединение между пирами**

Стандартный протокол для общения пиров – TCP. Сегодня также получили распространение новые протоколы типа uTP (uTorrent Transport Protocol), но они менее документированы и интереса для нас не представляют.

Получив от трекера адреса пиров, клиент пытается создать с ними соединение. Затем выполняется операция рукопожатия (*handshake*).

Для инициации рукопожатия клиент по установленному соединению посылает последовательнось:

*<pstrlen><pstr><reserved><info\_hash><peer\_id>,*

где pstrlen и pstr унифицированы и говорят пиру о типе протокола (значения 19 и “BitTorrent protocol” соответственно), reserved – восемь нулевых байтов, которые могут использоваться расширениями протокола BitTorrent, info\_hash – знакомая нам хэш-сумма секции Info торрент файла (уже не требующая экранирования), peer\_id – идентификатор нашего клиента.

В ответ клиент должен получить сообщение такого же формата, уже с идентификатором другого пира. Если хэш-суммы совпадают, можно начинать диалог.

* 1. **Диалог между пирами**

Собеседники обмениваются сообщениями одного из 11 типов:

* Keep-alive – четыре обнуленных байта, говорящих собеседнику о поддержании соединения (необходимо при задержках больших, чем 2 минуты).
* Choke – сообщение о том, что пир в данный момент не намерен отдавать данные (используется в расширениях протокола, для задержки передачи и пр.). Формат (1 байт):

*<id=0>*

Здесь и далее: сообщения предваряются четырьмя битами, хранящими длину сообщения. Для Choke полное сообщение будет выглядеть как 00010.

* Unchoke – пир готов к отдаче:

*<id=1>*

* Interested – посылается клиентом, который хочет загрузить данные:

*<id=2>*

* Not interested – клиенту не нужны данные собеседника:

*<id=3>*

* Have – пир сообщает, что владеет определенным сегментов целевого файла:

*<id=4><piece index>*

* Bitfield – пир показывает битовую маску имеющихся у него сегментов (один бит на сегмент, значение “0” при отсутствии сегмента и “1” при наличии):

*<id=5><bitfield>*

* Request – запрос на получение определенного сегмента (его части):

*<id=6><segment index><offset><length>*

* Piece – передача данных сегмента:

*<id=7><segment index><offset><data>*

* Cancel – отмена запроса на получение данных (напр., данные уже получены из другого источника):

*<id=8><segment index><offset><length>*

* Port – передача номера порта, по которому пир слушает запросы протокола DHT:

*<id=9><port>*

Последовательность и размер большинства сообщений заранее неизвестны, поэтому любой ответ собеседника может содержать как несколько сообщений, так и часть одного сообщений.

Изначальное состояние диалога: Choke, Not interested (ни одна из сторон не готова к передаче данных).

После рукопожатия пир, к которому был обращен запрос передает информацию об имеющихся у него сегментах посредством сообщений Have и Bitfield.

Клиент-проситель определяет номера сегментов, отсутствующих у него и имеющихся у собеседника. Если таковые существуют, клиент посылает сообщение Interested. Далее он должен получить ответ Unchoke, и только после этого начать слать запросы на данные.

1. **Обзор аналогов**

[uTorrent](http://utorrent.com/download.php) – один из самых компактных торрент-клиентов (120kb) с графическим интерфейсом. Веб-интерфейс отсутствует. Возможность форсированной закачки. Не требуется установка. Высокая стрессоустойчивость, настройки не теряются даже при внезапном выключении питания. Эффективный алгоритм кэширования. Умеет выбирать файлы из директории для выборочной закачки из торрента. Ведется статистику, есть логи. Высокая скорость опроса трекеров, скорость скачивания иной раз оставляет позади FastTorrent, директории качаются быстрее. Встроенный поисковик по торрент-сайтам. Возможность самостоятельной раздачи. Использование широкого диапазона портов (недостаток).

BitTorrent – простой клиент без дополнительных возможностей. Относительно низкая скорость опроса трекеров и закачки. Нет возможности форсирования. Нет возможности выбора файлов для закачки. Дает невысокую нагрузку на сеть.

Vuze (Azerus) – мощный клиент, написанный на Java. Создан 2003. Выделяется оригинальным графическим интерфейсом. Поиск торрентов через vuze.com, пользовательское HD-видео, прокси, доступ к социальной сети vuze.com. Исходные коды ядра в открытом доступе.

1. **Определение и анализ требований**
   1. **Назначение разработки**

Цель разработки – создание программного средства для загрузки файлов по сети с помощью протокола BitTorrent.

* 1. **Выполняемые функции**
* Чтение метафайлов .torrent;
* Опрашивание торрент-трекеров на информацию о скачиваемом файле и списке пиров;
* Соединение с пирами и получение требуемых данных;
* Управление данными, их проверка и организация в виде конечного файла.
  1. **Входные данные**

Метафайлы .torrent, протокол BitTorrent.

* 1. **Выходные данные**

Целевой загружаемый файл, описанный метафайлом .torrent.

* 1. **Технические требования**
* Процессор тактовой частотой 1.0 ГГц и выше;
* Не менее 512 Мб памяти ОЗУ.
  1. **Программные требования**
* Интернет-соединение;
* Операционная система Windows XP/7/8;
* .Net Framework 4.5;
* Разрешение от брандмауэра на передачу и получение пакетов UDP.

1. **Постановка задач**

С учетом требований определены следующие задачи процесса разработки:

* Разработать средство для чтения и интерпретации файлов .torrent;
* Реализовать протокол связи с торрент-трекером;
* Обеспечить соединение с трекером и получение данных о целевом файле;
* Реализовать протокол соединения и диалога с пирами;
* Обеспечить соединение и диалог с пирами по адресам, полученным от трекера;
* Разработать средство определения статуса загрузки файла;
* Реализовать проверку соответствия получаемых данных;
* Разработать средство управления и организацией получаемых данными.

1. **Разработка программного средства**
2. **Руководство пользователя**