Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОМУ ПРОЕКТУ

НА ТЕМУ:

Реализация межсетевого экрана

Студент ИУ7-72Б _____ _____ Е.В. Брянская
(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель курсового проекта _____ Н.Ю. Рязанова
(Подпись, дата) (И.О.Фамилия)

2021 г.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ
Заведующий кафедрой ___ИУ7_____
(Индекс)
_____ __И.В. Рудаков_
(И.О.Фамилия)
« _____ » _____ 2021 г.

З А Д А Н И Е
на выполнение курсового проекта

по дисциплине _____Компьютерные сети_____

Студенты группы ___ИУ7-72Б_____

_____Брянская Екатерина Вадимовна_____
(Фамилия, имя, отчество)
_____Иванов Всеволод Алексеевич_____
(Фамилия, имя, отчество)

Тема курсового проекта _____
_____BitTorrent-клиент_____

Направленность КП (учебный, исследовательский, практический, производственный, др.)
__учебный_____
Источник тематики (кафедра, предприятие, НИР) ___кафедра_____

График выполнения проекта: 25% к _4_ нед., 50% к _7_ нед., 75% к 11 нед., 100% к _14_ нед.

Задание__Разработать торрент-клиент на основе протокола BitTorrent_____
_____
_____

Оформление курсового проекта:

Расчетно-пояснительная записка на _20-30__ листах формата А4.
Расчетно-пояснительная записка должна содержать постановку задачи, введение,_____
аналитическую, конструкторскую, технологическую части, заключение, список литературы.

Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.):_____
на защиту работы должна быть предоставлена презентация, состоящая из 15-20 слайдов._____
На слайдах должны быть отражены: постановка задачи, использованные методы и_____
алгоритмы, расчетные соотношения, структура комплекса программ, интерфейс._____

Дата выдачи задания «8» __октября___ 2021 г.

Руководитель курсового проекта _____ __Н.О. Рогозин_____
(Подпись, дата) (И.О.Фамилия)
Студент _____ __Е.В. Брянская_____
(Подпись, дата) (И.О.Фамилия)
Студент _____ __В.А. Иванов_____
(Подпись, дата) (И.О.Фамилия)

Содержание

ВВЕДЕНИЕ

За последние время существенно возросли объёмы информации, передаваемой по сети Интернет. Очевидно, что подобная тенденция сохранится и в будущем – будет расти число пользователей и объём потребляемого ими трафика.

В подобных условиях актуальным является вопрос производительности серверов. Ввиду описанных выше факторов нагрузка на них будет постоянно расти, что будет вынуждать их владельцев производить их обновление и расширение или снижение скорости обмена информацией с клиентами.

Последнее является чувствительным для загрузки файлов больших объёмов. Решением в таком случае может быть кооперативный обмен файлами. Наиболее популярным протоколом для этой технологии является Bittorrent.

Целью данной работы является разработка Bittorrent клиента.

Для достижения поставленной цели необходимо решить следующие **задачи**:

1) изучить структуру и принцип работы протокола;
2) разработать алгоритм взаимодействия с сервером и клиентами;
3) реализовать программу для загрузки файлов на основе протокола Bittorrent.

1 Аналитическая часть

1.1 Постановка задачи

Результатом работы должна стать программа для загрузки файлов по протоколу Bittorrent, удовлетворяющая следующим требованиям:

- поддерживать файлы расширения .torrent;
- поддерживать функцию загрузки данных как от сервера, так и от других клиентов;
- обладать графическим интерфейсом для удобства выполнения действий и просмотра текущей информации по состоянию загрузки.

Первостепенной задачей для дальнейшей разработки является изучения устройства выбранного протокола.

1.2 Принцип работы протокола

Bittorrent – P2P протокол для кооперативного обмена файлами через интернет.

В данном протоколе выделены две роли:

1) **пир** (клиент) хранит файлы и производит обмен их частями с другими пирами;
2) **трекер** (сервер) хранит таблицу файлов и список пиров, имеющих данный файл в распоряжении.

Пир, желающий получить файл должен обладать **.torrent файлом**, с помощью которого он может обратиться к серверу. Сервер предоставляет адреса клиентов, обладающих запрашиваемыми файлами после чего начинается их загрузка. Передача осуществляется частями (**pieces**), каждый torrent-клиент, скачивая эти части, в то же время отдаёт их другим клиентам, что снижает нагрузку на каждого отдельного клиента.

1.3 Структура .torrent файла

Как было отмечено выше, первым шагом в начале загрузки является получение и парсинг файла специального расширения .torrent.

Для кодирования данных в .torrent-файлах используется формат Bencode. Само содержимое – ассоциативный массив с полями:

- **info** – вложенный ассоциативный массив который описывает файлы, передаваемые торрентом;
- **announce** – URL трекера;
- **announce-list** – список трекеров, если их несколько, в Bencode-виде — список списков;
- **creation date** – дата создания;
- **comment** – текстовое описание торрента;
- **created by** – автор торрента.

info и announce являются обязательными полями, всё остальные — опционально. Первый в свою очередь состоит из:

- **piece length** – размер одного куска;
- **pieces** – конкатенация SHA1-хешей каждого куска (каждый хеш - 20 байт);
- **name** – имя файла (если файл один);
- **length** – содержит длину файла (если файл один);
- **files** – если файлов несколько, то содержит список ассоциативных массивов (с указанием length и path).

Данная информация используется на всём протяжении загрузки файла и его последующей раздаче.

1.4 Взаимодействие клиента и сервера

Чтобы перейти к загрузке файла клиент должен получить список пиров у трекера. Для этого он должен отправить GET-запрос, называемый **анонсом**, по адресу announce по пути /announce.

После данного действия трекер узнаёт о наличии нового клиента и может выдать его адрес другим клиентам. Указываются следующие URL-параметры.

- **info_hash** – SHA1-хеш словаря info. Используется для поиска файла в таблице трекера, то есть фактически является его уникальным идентификатором.
- **peer_id** – уникальный ID клиента. Имеет вид *-<2-символьный id><номер версии из 4 цифр>-<12 случайных цифр>*. Такой код может быть сгенерирован клиентом самостоятельно, так как вероятность коллизии с другими клиентами крайне мала (число возможных вариантов peer_id одной версии превышает количество IPv4 адресов более чем в 200 раз).
- **uploaded**, **downloaded**, **left** – количество отправленных, загруженных и незагруженных байтов.
- **port** – TCP-порт, прослушиваемый клиентом. Общепринятыми значениями являются 6881-6889.
- **compact** – признак того, принимает ли клиент компактный список пиров.

В случае, если запрос прошёл успешно и по info_hash был найден необходимый torrent, трекер посылает ответ (также по протоколу HTTP). В его теле содержится следующие поля в формате Bencode:

- **interval** – интервал в секундах до того, как клиент должен сделать новый запрос к трекеру;
- **peers** – список пиров. В случае, если в запросе compact был равен 1, в ответе будет список будет заменён бинарной строкой, которую потребуется разбить на группы по 6 байт для выделения IPv4 адреса и порта каждого

пира.

Подобные запросы будут повторяться раз в interval секунд для поддержания сервера в курсе актуального состояния загрузки и для получения новых адресов пиров.

1.5 Структура сообщений

Протокол BitTorrent определяет следующий способ обмена сообщениями для клиентов, его особенности:

- использует стек TCP/IP;
- файл передаётся по кускам фиксированного размера, не в порядке их следования в файле.

Определена следующая структура p2p сообщения:

1) **длина**, Len (4Б) – размер типа и полезной нагрузки сообщения;
2) **тип**, ID (1Б) определяет вид сообщения и способ его обработки;
3) **полезная нагрузка**, Payload (0 - 32КБ) содержит передаваемую информацию.

Различаются следующие типы сообщений.

- **handshake**: <len=49+X><info_hash><peer_id>. Сообщение рукопожатия. Отправляется один раз в начале обмена информацией. Содержит название протокола, хеш код файла и собственный id.
- **keep-alive**: <len=0000>. Содержит только нулевую длину. Используется чтобы один из пиров не закрыл соединение по истечению времени без сообщений.
- **choke**: <len=0001><id=0>. Используется для запрета другому пиру отправки сообщений до момента посылки ему unchoke.
- **unchoke**: <len=0001><id=1>.

- **interested**: <len=0001><id=2>. Состояние говорит о том, что пир заинтересован в получении фрагментов.
- **not interested**: <len=0001><id=3>. Обратно interested.
- **have**: <len=0005><id=4>. Сообщает о появлении в своём распоряжении куска с указанным индексом.
- **bitfield**: <len=0001+X><id=5>. Содержит в себе карту битов, описывающую статус всех кусков файла.
- **request**: <len=0013><id=6>. Используется для запроса блока байт размером length, начинающимся с позиции begin из куска с номером index.
- **piece**: <len=0009+X><id=7>. Сообщение содержит в себе блок байт block по формату, описанному в request.

1.6 Взаимодействие клиентов

Первым шагом после получения адреса пира требуется выполнения ”рукопожатия”(handshake). Он нужен для обмена id и проверкой совпадения протоколов и контрольного хеша файла. В случае неудачного рукопожатия TCP соединение разрывается.

После рукопожатия устанавливается состояние Choked. Для выхода из него сразу отправляется сообщение Interested для перехода к обмену.

В первую очередь после взаимной заинтересованности пиры обмениваться информацией о наличии кусков с помощью сообщения bitfield. Это требуется для определения отсутствующих кусков, которые можно запросить у данного пира.

В тот момент, когда клиент может запросить кусок (т.е. не находится уже в состоянии загрузки с данным пиром, не является choked и not interested), он выбирает блок для запроса у данного пира. Приоритет выбора следующий:

1) блоки, для которых истекло время ожидания;
2) блоки из неполностью загруженных кусков;
3) блоки из наиболее редких кусков.

Блоки будут отсутствовать во всех перечисленных категориях только в случае, если загрузка почти полностью завершена. Такая ситуация называется **end-game**. В этом случае в качестве очередных блоков для запроса выбираются уже загружаемые блоки.

После получения блока (сообщения piece), он записывается с указанным смещением в нужный кусок. По окончанию загрузки куска подсчитывается его контрольная сумма и сравнивается с той, которая изначально хранилась в torrent файле. Если они совпали, кусок помечается загруженным и сохраняется в загружаемый файл, а всем хостам без данного куска отправляется сообщение Have с его номером.

Файл считается загруженным полностью кода скачены и проверены все его куски.

Вывод

Результатом аналитического раздела стал анализ устройства протокола BitTorrent, алгоритма взаимодействия с сервером и другими пирами.

2 Конструкторская часть

2.1 Основной алгоритм

На Рисунке 2.1 приведен основной алгоритм работы Bittorrent-клиента.

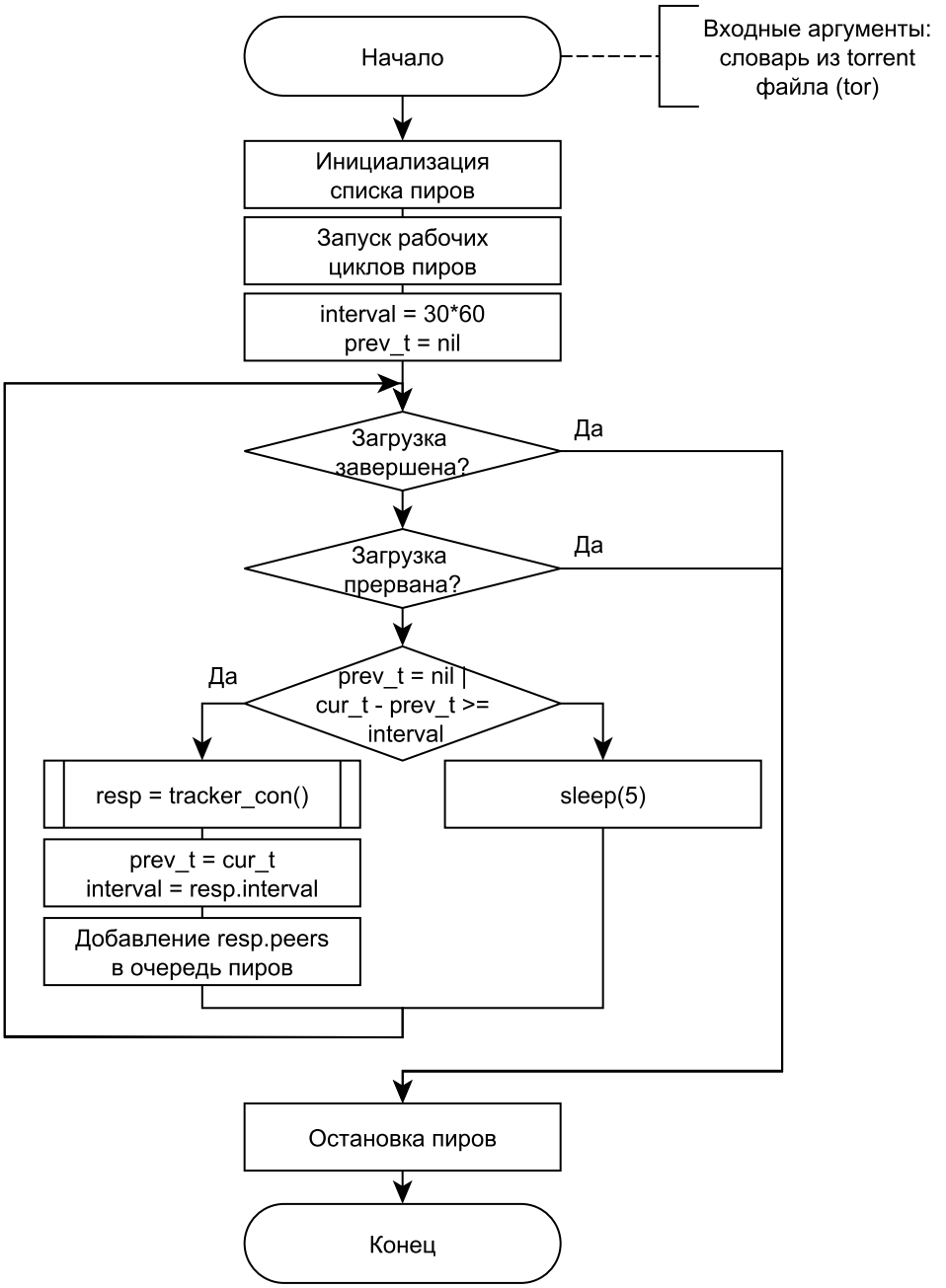

Рисунок 2.1 – Основной алгоритм

2.2 Алгоритм взаимодействия с сервером

Детали алгоритма взаимодействия с сервером продемонстрированы на схеме 2.2.

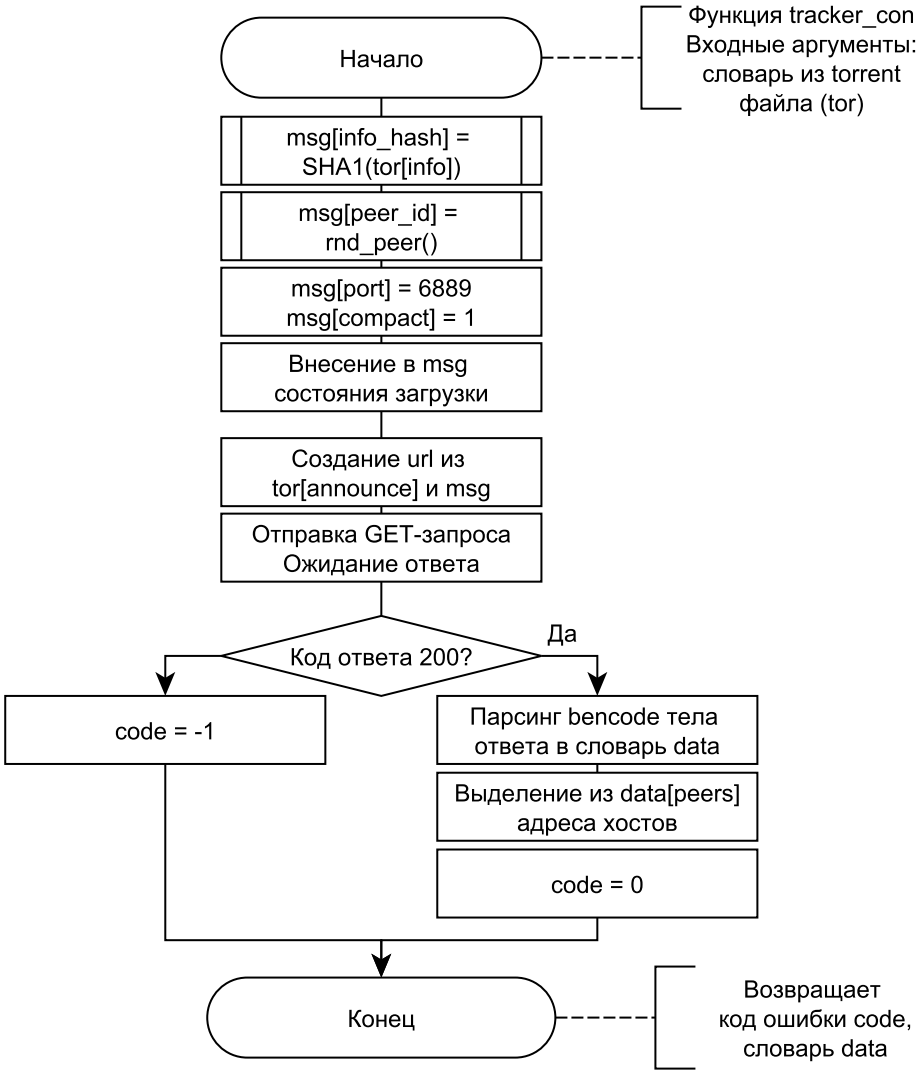

Рисунок 2.2 – Алгоритм взаимодействия с сервером

2.3 Алгоритм рукопожатия

Этот алгоритм приведён на Рисунке 2.3.

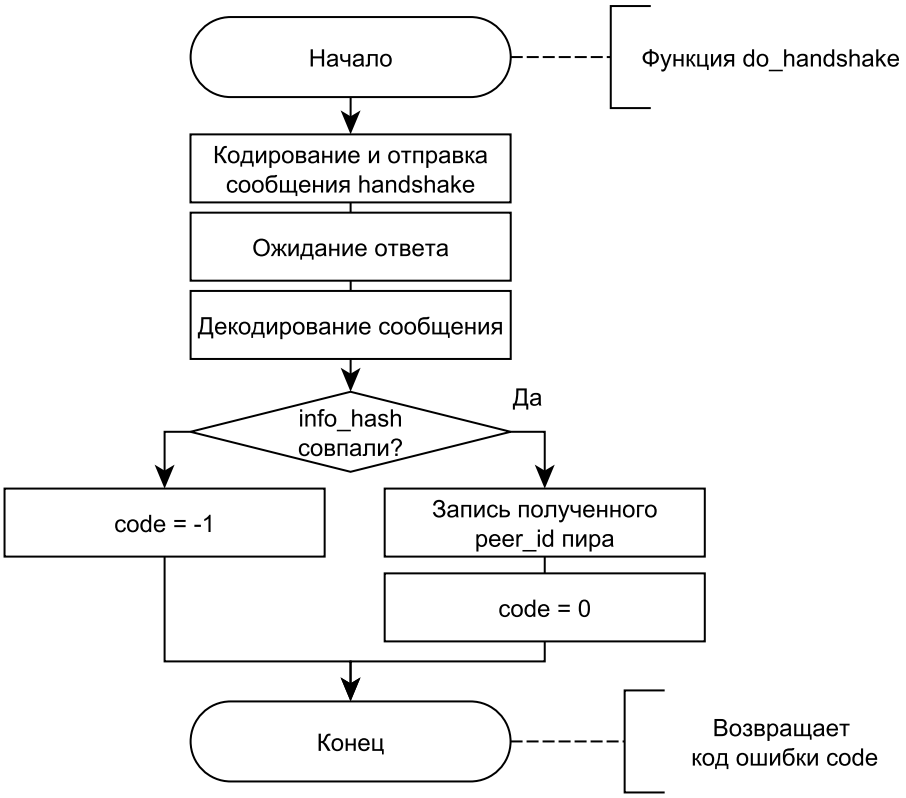

Рисунок 2.3 – Алгоритм рукопожатия

2.4 Алгоритм взаимодействия с пирами

Детали взаимодействия с пирами приведены ниже, на Рисунке 2.4.

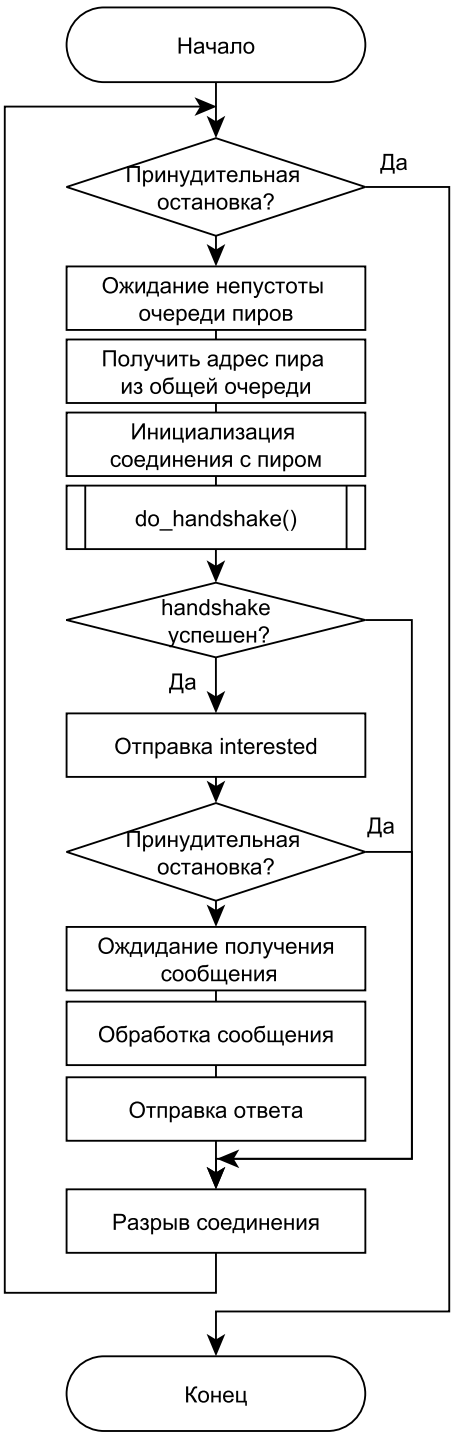

Рисунок 2.4 – Алгоритм взаимодействия с пирами

3 Технологическая часть

3.1 Выбор технологических средств

В качестве языка программирования был выбран Python, поскольку он предоставляет множество необходимых для реализации поставленной задачи библиотек, такие как aiohttp, socket, bencodepy и прочие, а также ввиду имеющего опыта работы с этим языком.

Была выбрана среда разработки PyCharm, поскольку она бесплатна для студентов и хороша знакома, так как активно использовалась в процессе обучения.

Для создания удобного, интуитивно понятного интерфейса использовался набор библиотек PyQt5.

3.2 UML диаграмма классов

На Рисунках 3.5-3.6 приведена UML-диаграмма основных разработанных классов. На диаграмме 3.6 приведены все виды сообщений, которыми могут обмениваться участники процесса скачивания.

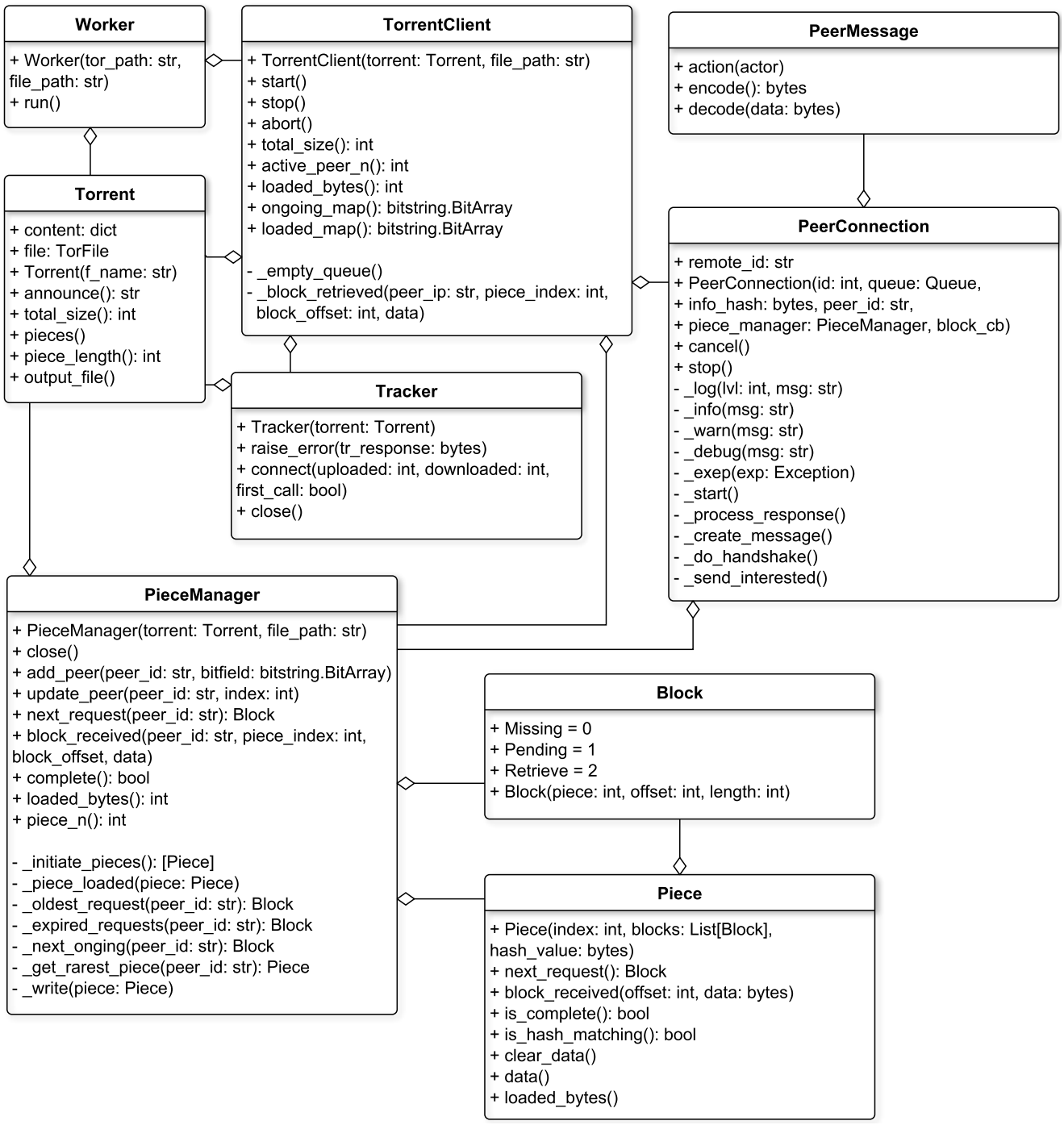

Рисунок 3.5 – UML-диаграмма классов

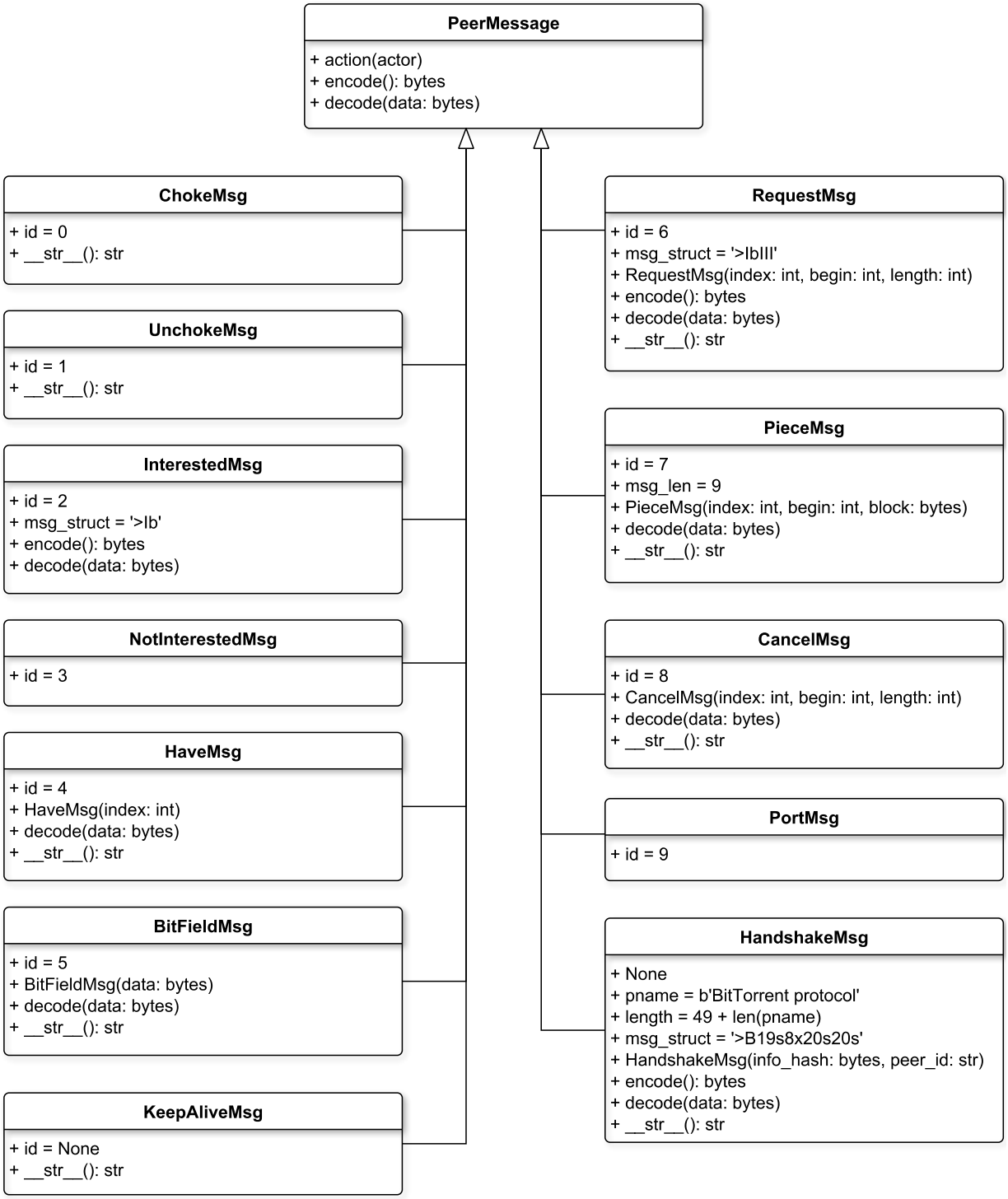

Рисунок 3.6 – UML-диаграмма классов сообщений

3.3 Интерфейс программы

На Рисунках 3.7-3.8 представлен разработанный графический интерфейс. Для того, чтобы начать процесс скачивания, необходимо сначала указать путь до .torrent файла и выбрать директорию загрузки. Нажатие кнопки «Старт» запустит процесс.

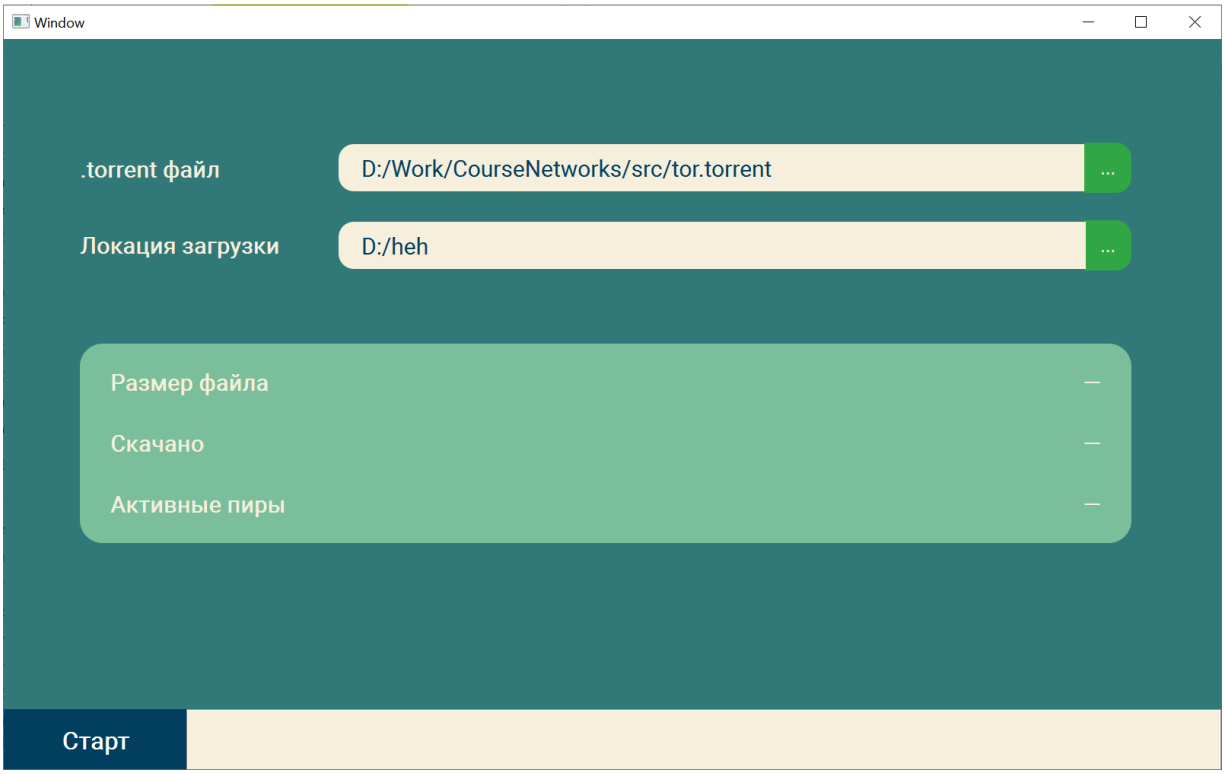

Рисунок 3.7 – Графический интерфейс до начала скачивания

В процессе скачивания необходимого файла на экран выводится статистика: размер файла в килобайтах, успешно полученный объём (килобайты) и число активных пиров, с которыми на данный момент происходит взаимодействие.

Для наглядности снизу была добавлена специальная шкала, в которой жёлтым цветом помечаются куски (pieces), находящиеся в процессе скачивания, зелёным – успешно полученные. Очевидно, когда файл будет полностью скачен, вся полоса будет покрашена в зелёный цвет.

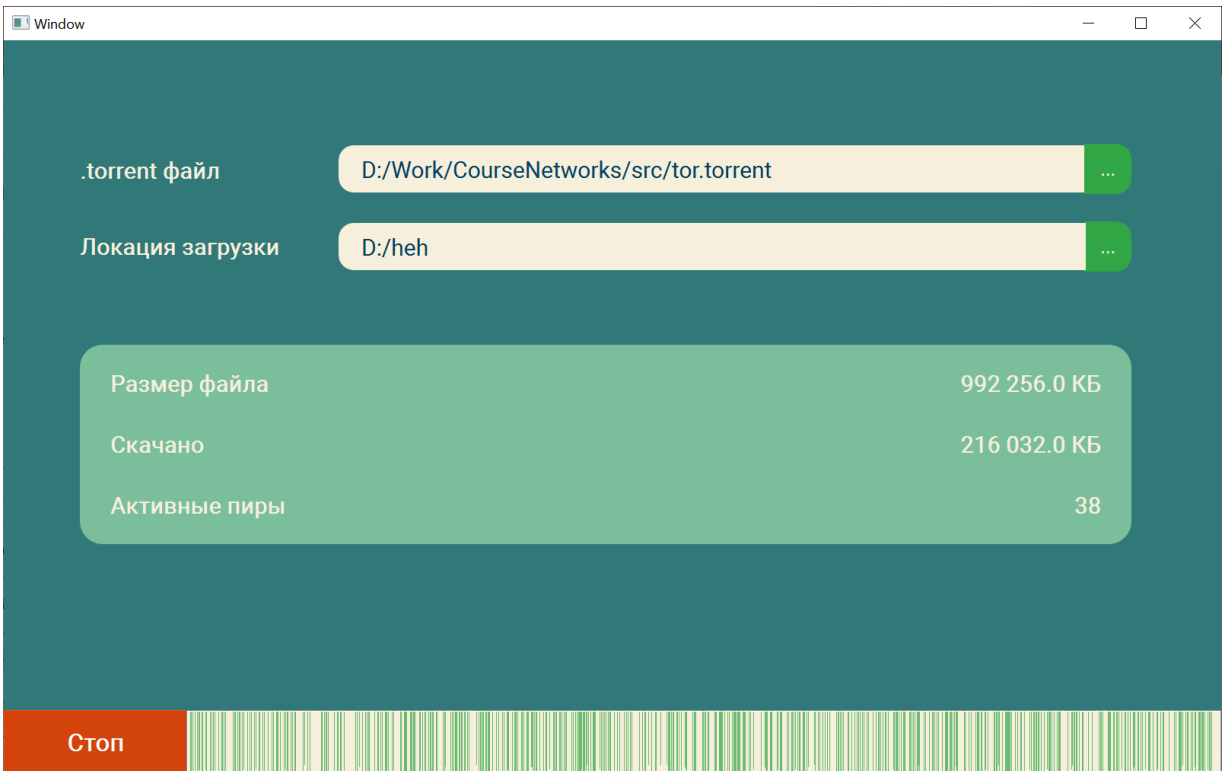

Рисунок 3.8 – Графический интерфейс в процессе скачивания

Для того, чтобы прервать операцию, достаточно нажать на кнопку «Стоп».

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

Список литературы

1. Hari Balakrishnan, M. FransKaashoek , David Karger, Robert Morris, and Ion Stoica. Looking up DATA in P2P systems. In Proc. Acm SIGCOMM’01, San Diego, CA, Aug. 2001.