P2P File Sharing

Назначение

Приложение позволяет производить децентрализованный обмен файлами

Функциональные требования

1. Приложение должно иметь консольный командный интерфейс и поддерживать следующие команды:
   - *share <FILE_PATH>* - Добавить файл к списку файлов, которые доступны для скачивания с этого клиента;
   - *scan* - Обновить список пиров и файлов, которые доступны для скачивания;
   - *ls* - Вывести список файлов, которые данный клиент может скачать;
   - *download FILE_NAME [-o SAVE_PATH]* - Скачать файл с указанным именем и сохранить его по указанному пути. Если путь не указан - сохранить в текущую папку. Скачивание должно начинаться в фоне и не блокировать ввод;
   - *status* - Показать список текущих загрузок и список файлов, которые доступны для скачивания с клиента. Для каждого файла, который доступен для скачивания с клиента, должен показаться список пиров, которые загружают его в данный момент. Также для каждого файла, который скачивается, должен показаться список пиров с которых происходит скачивание файла.
2. Приложение должно работать децентрализовано и не требовать наличия центрального сервера;
3. Приложение не должно получать список пиров от пользователя - они должны находиться автоматически при выполнении команды *scan*;
4. (опционально) Сервисная часть должна использовать *async/.await* с *tokio* (*https://tokio.rs/*);
5. (опционально) Передача файлов должна происходить по защищенному зашифрованному соединению. Лучше использовать *Rustls* (*https://github.com/ctz/rustls*), но можно и *native-tls* (*https://github.com/sfackler/rust-native-tls*);
6. (опционально) Добавить параметр *[-w]* к команде *download*. При получении этого флага команда *download* блокирует ввод пока файл не будет полностью скачан;
7. (опционально) Если несколько пиров позволяют скачать один и тот же файл, скачивание должно производиться с каждого из них одновременно.

Нефункциональные требования

1. Приложение должно быть реализовано с помощью языка программирования Rust (*https://www.rust-lang.org/*);

2. Для реализации можно (и следует) использовать любые библиотеки с crates.io;
3. Для нахождения других пиров в сети *можно* использовать *IP-multicast*, но не следует использовать его для передачи файлов;
4. Приложение должно состоять из 2х частей:
    1. демон/сервис, который выполняет загрузку и раздачу файлов;
    2. CLI клиент, который отвечает за взаимодействие пользователя с демоном/сервисом.

Пример работы с приложением

Пример работы с приложением из терминала Linux*:

1. Запустить демон:

    ```
    $ ./p2pfs_daemon
    ```

2. Обновить список пиров и файлов:

    ```
    $ ./p2pfs scan
    ```

3. Показать доступные для скачивания файлы:

    ```
    $ ./p2pfs ls

    Kreator.Extreme_Aggressions.flac

    LordOfTheRings.mkv

    Learn_CPP_in_21_seconds.pdf

    GloomhavenRuleBook.pdf
    ```

4. Добавить файл к списку файлов:

    ```
    $ ./p2pfs share /home/user/Videos/MyFavouriteMovie.mp3
    ```

5. Скачать файл:

    ```
    $ ./p2pfs download GloomhavenRuleBook.pdf -o /home/user/Docs/
    ```

6. Показать статус:

    ```
    $ ./p2pfs status

    Sharing:

        MyFavouriteMovie.mp3

            192.168.1.106

            192.168.1.234

    Downloading:

        GloomhavenRuleBook.pdf

            192.168.1.120
    ```

Примеру следовать не обязательно, название приложения и вывод команд могут отличаться.

* Приложение не обязательно должно работать на Linux. Достаточно поддержки любой ОС. Linux здесь приведён для примера, т.к. синтаксис терминала может отличаться на других платформах.