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» (ИУ7)_____

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОЙ РАБОТЕ

НА ТЕМУ:

Приложение для трансляции аудио с одного компьютера на несколько устройств

Студент ___**ИУ7-73Б**_____ _____ **Фурдик Н.О.**____
(Группа) (Подпись, дата) (И.О.Фамилия)

Студент ___**ИУ7-73Б**_____ _____ **Коновалова О.С.**____
(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель курсового проекта _____ **Рогозин Н.О.**_____
(Подпись, дата) (И.О.Фамилия)

2020 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)
« _____ » _____ 2020 г.

З А Д А Н И Е
на выполнение курсовой работы

по дисциплине Компьютерные сети

Студенты группы ИУ7-73Б

Фурдик Никита Олегович, Коновалова Ольга Сергеевна
(Фамилия, имя, отчество)

Тема курсового проекта Приложение для трансляции аудио с одного компьютера на несколько устройств

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 клиент-серверное приложение с передачей данных через собственный протокол прикладного уровня. Реализовать клиент, подготавливающий данные для отсылки на сервер с помощью протокола на одном компьютере. Реализовать сервер, принимающий данные через протокол на другом компьютере.

Оформление курсового проекта:

Расчетно-пояснительная записка на 20-30 листах формата А4.
Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)
На защиту проекта должна быть представлена презентация, состоящая из 10-20 слайдов. На слайдах должны быть отражены: постановка задачи, использованные методы и алгоритмы, расчетные соотношения, структура комплекса программ, диаграмма классов, интерфейс, характеристики разработанного ПО.

Дата выдачи задания « » _____ 2020 г.

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

Студенты

_____ Фурдик Н.О.
(Подпись, дата) (И.О.Фамилия)

_____ Коновалова О.С.
(Подпись, дата) (И.О.Фамилия)

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

Оглавление

Реферат

Данная работа посвящена разработке программного комплекса для трансляции аудио с одного компьютера на несколько устройств на основе протокола TCP. Программный комплекс состоит из клиента и сервера.

Отчет содержит 20 страниц, 7 рисунков, 4 листинга, 8 источников.

Введение

В настоящее время музыкальные стриминговые сервисы с каждый годом наращивают популярность. Так, например, в 2016 году они сгенерировали 51% выручки всей музыкальной индустрии США, а число платных подписчиков составило 100 миллионов.Прогнозируется, что их будет 500 миллионов в 2020 году. В России стримингом музыки тоже пользуются все активнее — в 2016 году доходы от стриминга музыки удвоились[1], а число платных подписок превысило миллион.

Одновременно с ростом популярности стриминговых сервисов падает количество физических продаж альбомов. Таким образом, можно сказать, что за потоковыми сервисами-будущее. Кроме того, такая возможность избавляет от необходимости хранения аудиофайлов на собственном устройстве, и, следовательно, экономит дисковое пространство.

Целью данной работы является разработка программного комплекса, обеспечивающего трансляцию аудио-файлов с одного компьютера на несколько устройств.

Для достижения поставленной цели необходимо решить следующие задачи:

- проанализировать существующие стриминговые сервисы, имеющие аналогичную цель;
- проанализировать протоколы локальных сетей транспортного уровня;
- выбрать протокол потокового аудио для организации трансляции;
- спроектировать программный комплекс для реализации поставленной цели;
- выбрать средства реализации для спроектированного программного комплекса;
- разработать и протестировать программный комплекс.

Аналитическая часть

В данном разделе производится постановка задачи, анализ предметной области, рассматриваются существующие протоколы.

Постановка задачи

Требуется:

- разработать клиент-серверное приложение для трансляции аудио-файлов с одного компьютера на несколько в режиме реального времени с передачей данных через собственный протокол прикладного уровня;
- реализовать клиент, принимающий данные через протокол на другом компьютере.
- реализовать сервер, подготавливающий данные для отсылки на сервер с помощью протокола на одном компьютере.

Анализ предметной области

Локальная сеть

Локальная вычислительная сеть – это совокупность компьютеров и других средств вычислительной техники (активного сетевого оборудования, принтеров, сканеров и т. п.), объединенных с помощью кабелей и сетевых адаптеров и работающих под управлением сетевой операционной системы.

Вычислительные сети создаются для того, чтобы группа пользователей могла совместно задействовать одни и те же ресурсы: файлы, принтеры, модемы, процессоры и т. п. Каждый компьютер в сети оснащается сетевым адаптером, адаптеры соединяются с помощью сетевых кабелей и тем самым связывают компьютеры в единую сеть. Компьютер, подключенный к вычислительной сети, называется рабочей станцией или сервером, в зависимости от выполняемых им функций.

Эффективно эксплуатировать мощности локальной сети позволяет применение технологии «клиент/сервер». В этом случае приложение делится на две части: клиентскую и серверную. Один или несколько наиболее мощных компьютеров сети конфигурируются как серверы приложений: на них выполняют-

ся серверные части приложений. Клиентские части выполняются на рабочих станциях; именно на рабочих станциях формируются запросы к серверам приложений и обрабатываются полученные результаты.

Различают сети с одним или несколькими выделенными серверами и сети без выделенных серверов, называемые одноранговыми сетями.

В случае с локальной сетью с выделенным сервером при выборе компьютера на роль файлового сервера необходимо учитывать следующие факторы:

- быстродействие процессора;
- скорость доступа к файлам, размещенным на жестком диске;
- емкость жесткого диска;
- объем оперативной памяти;
- уровень надежности сервера;
- степень защищенности данных.

Локальные сети используются на многих предприятии. Также, они есть в частных домах и квартирах. Удобство такой организации девайсов заключается в том, что между компонентами сетки можно легко передавать файлы, создавать режим общего доступа к информации и одновременно проигрывать один и тот же контент на нескольких устройствах.

Стриминг

Стриминговые (они же потоковые) сервисы работают по принципу передачи контента от провайдера к пользователю. Весь контент уже загружен на стороннем сервере, конечному пользователю не требуется ничего скачивать для просмотра или прослушивания. Контент транслируется в режиме реального времени.

Современные потоковые сервисы позволяют смотреть абсолютно всё: от новостей в прямом эфире до классических фильмов и последних сериальных новинок. Условно они делятся на поставщиков музыки, видео, видеоигр и программного обеспечения.

Среди достоинств стриминговых сервисов по сравнению с хранением медиаданных непосредственно на устройстве можно выделить следующие:

- экономия дискового пространства;
- увеличение доступности контента и возможность его воспроизведения на любом устройстве сети;
- готовые подборки и персональные рекомендации.

Обзор стриминговых средств для локальной сети

В качестве примера стримингового сервиса для локальной сети рассмотрим кросс-платформенный медиаплеер VLC. VLC media player - это инструмент для работы с музыкой и видео, с его помощью можно: воспроизводить онлайн-трансляции и медиа-файлы почти любого формата, записывать происходящее на экране, передавать видео или музыку по локальной сети или с помощью сети интернет и многое другое.

Рассмотрим передачу аудио по локальной сети с помощью VLC media player. Для начала разберемся с основными понятиями. DLNA (Digital-Living-Network-Alliance) – стандарты, по которым различные устройства могут принимать и передавать медиа-файлы, транслировать их в режиме «онлайн». Объединение электроники в единую сеть может быть: беспроводным (Wi-Fi) и проводным (Ethernet). Сетью объединяются мобильные телефоны, ноутбуки, настольные ПК, смарт-часы и другие аппараты. В общем смысле это три категории девайсов: многофункциональные приборы, сетевые бытовые устройства, мобильные устройства. DLNA описывает все методы совместного использования разного оборудования. Так, можно с видеокамеры или смартфона посмотреть фото на телевизоре или распечатать их на принтере. Смотреть фильм из «облака» на компьютере или вывести экран монитора на телевизор тоже помогает DLNA. Плеер VLC имеет два способа DLNA-взаимодействия: DLNA-клиент и DLNA-сервер. Первый позволяет проигрывать аудио, видео-файл, находящийся на другом устройстве. Второй обеспечивает раздачу материала другим гаджетам [2].

Обзор протоколов аудиопередачи

Стремительный рост передачи мультимедиа-информации предъявляет новые требования к скорости и объемам передачи данных. И для того чтобы удовле-

творить все эти запросы, одного увеличения емкости сети недостаточно, необходимы разумные и эффективные методы управления трафиком и контролем загруженности линий передачи.

В приложениях реального времени отправитель генерирует поток данных с постоянной скоростью, а получатель (или получатели) должен предоставлять эти данные приложению с той же самой скоростью. Такие приложения включают, например, аудио- и видеоконференции, живое видео, удаленную диагностику в медицине, компьютерную телефонию, распределенное интерактивное моделирование, игры, мониторинг в реальном времени и др.

Наиболее широко используемый протокол транспортного уровня — это TCP. Он обеспечивает гарантированную доставку пакетов и проверку их целостности и отлично подходит для задач, связанных с передачей важной информации, не привязанной к времени. Протокол TCP устанавливает соединение между отправителем и получателем, регулируя скорость передачи данных, после этого начинается передача медиа-данных, затем осуществляется проверка доставки данных без изменений и в нужной последовательности, и в случае неудачной доставки осуществляется повторная передача.

Существует еще один протокол транспортного уровня-UDP. В отличие от TCP, UDP не устанавливает предварительного соединения, а вместо этого просто начинает передавать данные. UDP не следит за тем, чтобы данные были получены и не дублирует их, если отдельные части пропали или пришли с ошибками. UDP менее надежен, чем TCP. Но с другой стороны, он обеспечивает более быструю передачу потоков благодаря отсутствию механизма повторения передачи потерянных пакетов.

Вывод

Так как механизмы проверки доставки пакетов являются более предпочтительными, чем низкие задержки при передаче данных, в качестве используемого протоколов был выбран протокол TCP.

Конструкторская часть

В данном разделе рассматривается структура разрабатываемого программного комплекса, который будет состоять из сервера и клиента.

Функциональная модель

На рисунке 1 представлена функциональная модель программного комплекса IDEF0 нулевого уровня.

Рис. 1: IDEF0 диаграмма программного комплекса

Сценарий использования

Ниже на рисунке 2 представлена диаграмма взаимодействия пользователя на стороне сервера с приложением и на рисунке 3 - аналогичная диаграмма для пользователя на стороне клиента.

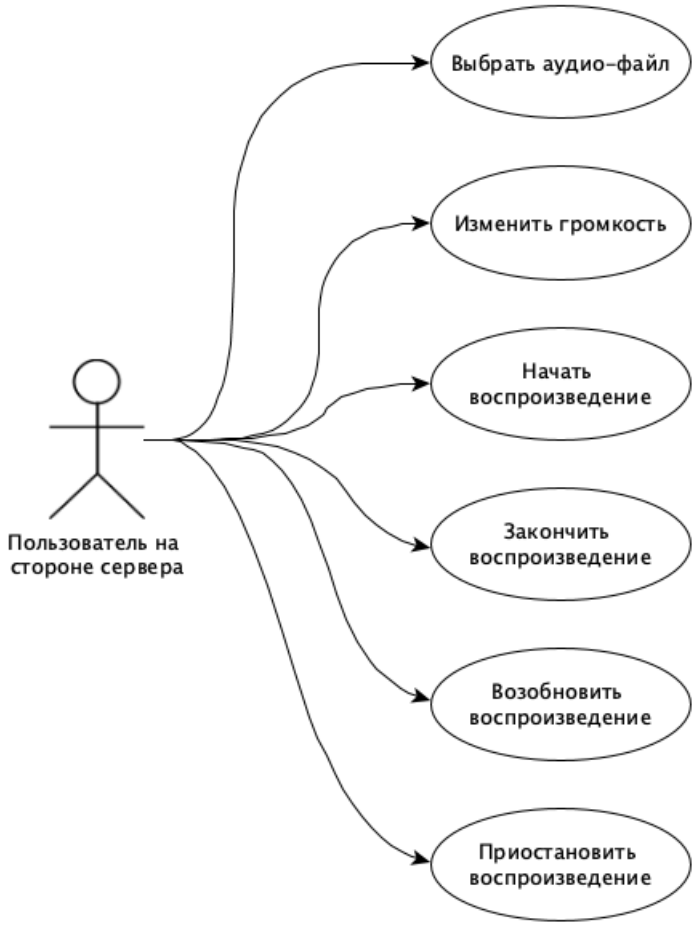

Рис. 2: UseCase диаграмма взаимодействия пользователя на стороне сервера с приложением

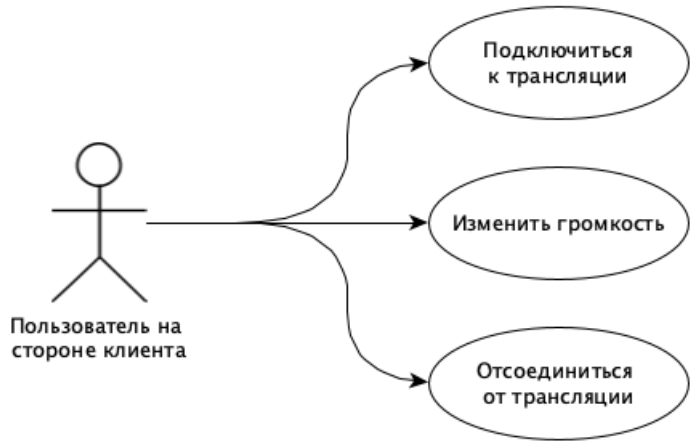

Рис. 3: UseCase диаграмма взаимодействия пользователя на стороне клиента с приложением

Архитектура приложения

Для реализации поставленнои задачи было принято решение разработать клиент-серверное приложение.

Архитектура клиента

Ниже на рисунке 4 приведена схема работы клиента.

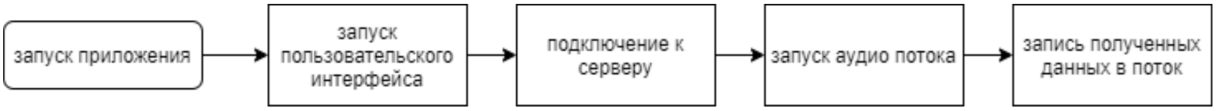

Рис. 4: Схема работы клиента

Архитектура клиента

Ниже на рисунке 5 приведена схема работы клиента.

Рис. 5: Схема работы сервера

Технологическая часть

В данном разделе производится выбор средств программнои реализации и приводится графическии интерфеис пользователя приложения.

Выбор инструментов разработки

В качестве языка программирования выбран язык Python, так как:

- это интерпретируемый язык, его интерпретаторы существуют для многих платформ (кроссплатформенность);
- с Python доступно множество сервисов, сред разработки, и фреймворков. Не составит труда найти подходящий продукт для работы;
- возможность подключить библиотеки, написанные на C. Это позволяет повысить эффективность, улучшить быстродействие;
- простота: в Python реализация сокетов (о них речь пойдет далее) значительно проще, чем в C/C++.

Для обеспечения информационного обмена между клиентом и сервером необходимо использовать сокеты. Сокеты — название программного интерфейса для обеспечения обмена данными между процессами. Процессы при таком обмене могут исполняться как на одной ЭВМ, так и на различных ЭВМ, связанных между собой сетью [3]. В Python для работы с сокетами используется модуль socket. Сокеты используют транспортныи уровень согласно семиуровневои модели OSI (Open Systems Interconnection, взаимодеиствие открытых систем).

В качестве инструмента работы со звуком был выбран модуль PyAudio. Он обеспечивает привязки Python для PortAudio, кросс-платформенной библиотеки аудио ввода-вывода. С помощью PyAudio можно использовать Python для воспроизведения и записи звука на различных платформах.

Для создания графического пользовательского интерфейса была использована кроссплатформенная встроенная библиотека Python - TKinter. Визуальные элементы отображаются через собственные элементы текущей операционной системы, поэтому приложения, созданные с помощью Tkinter, выглядят так, как будто они принадлежат той платформе, на которой они работают. С помощью Tkinter обычно создаются неброские интерфейсы, зато ему отдают пред-

почтение, когда в приоритете стоят функциональность и кроссплатформенная скорость.

Интерфейс приложения

На рисунке 6 приведен интерфеис пользователя для сервера, выводящиися на экран при запуске программы. На этои странице пользователь может в меню выбрать одно из шести деиствии: выбрать аудио-файл (из файлов, приведенных справа на панели songs), начать воспроизведение (кнопка PLAY на панели control panel), закончить воспроизведение (кнопка STOP на панели control panel), возобновить воспроизведение (кнопка RESUME на панели control panel), приостановить воспроизведение (кнопка PAUSE на панели control panel), изменить громкость (ползунок на панели currently playing). Воспроизведем конкретный выбранный аудио-файл. Для этого выберем на панели songs файл и нажмем кнопку PLAY на панели control panel. Начнется воспроизведение, параллельно будет работать эквалайзер. Пользователь может приостановить, возобновить, изменить громкость или полностью остановить воспроизведение текущей композиции.

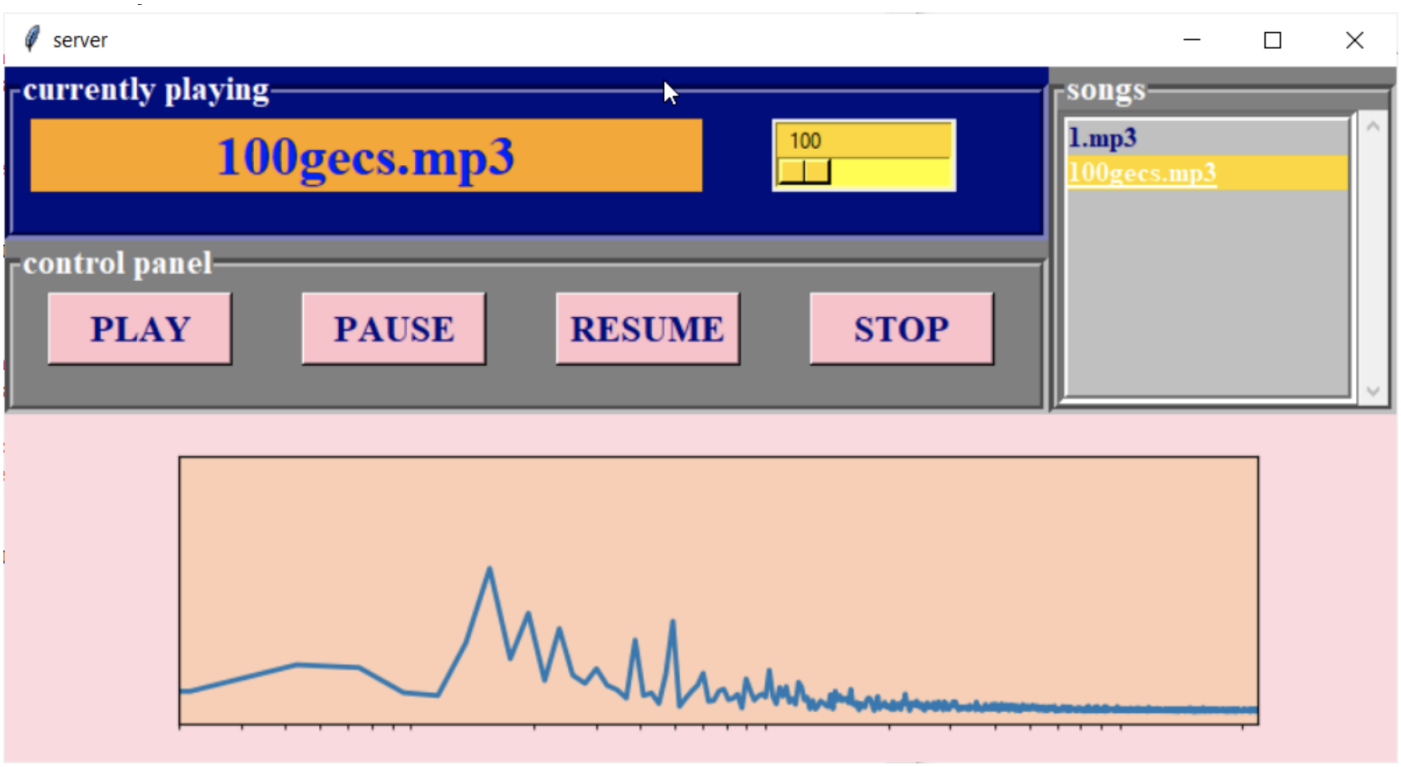

Рис. 6: Интерфейс сервера

На рисунке 7 приведен интерфейс пользователя для клиента, выводящийся на экран при запуске программы. Пользователь может подключиться (кнопка CONNECT на панели control panel) к трансляции или отсоединиться от нее (кнопка DISCONNECT на панели control panel), а также изменить громкость (ползунок на панели currently playing).

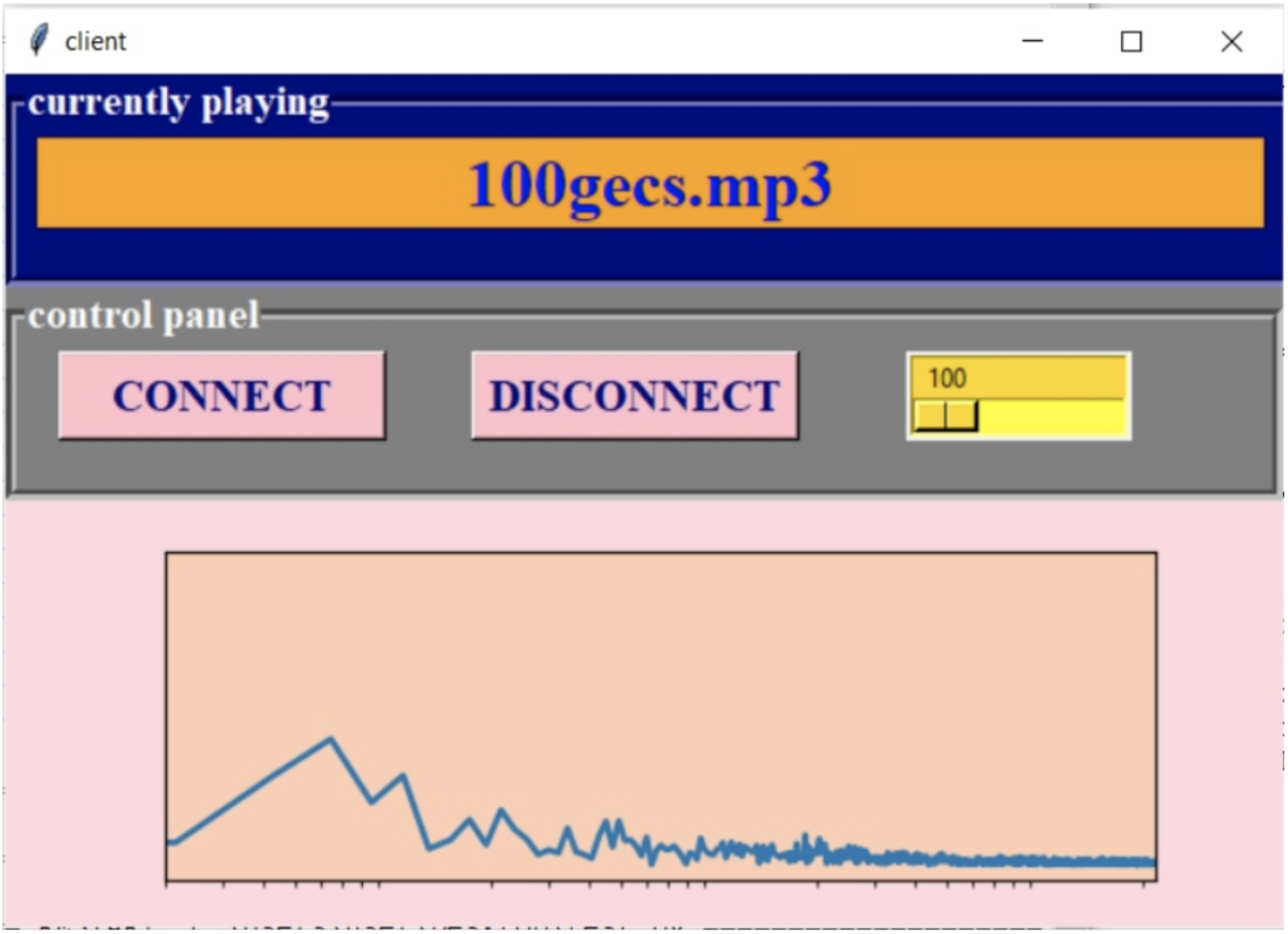

Рис. 7: Интерфейс сервера

Требования к системе

Для нормального функционирования программного комплекса система должна удовлетворять следующим требованиям:

- операционная система Windows (если 7 то со всеми обновлениями);
- установленный ffmpeg в случае работы с не WAV файлами;
- redist 2015 c++.

Описание программных модулей

Описание модулей серверного приложения

Серверное приложение абстрактно можно представить в виде трех модулей.

1) Модуль создания соединений – один из основных модулей серверного приложения. Содержит информацию о сервере – адрес и порт, в нем также инициализируется сокет для передачи данных клиентам.

   Также в модуле содержится главный поток серверного приложения, который принимает запросы на подключение клиентов, в последствии помещая успешные в список установленных соединений. Этот модуль осуществляет ожидание нового подключения, а когда подключение инициировано – создает новое соединение для передачи данных.

2) Модуль проигрывания аудио:

   - считывает аудио файл;
   - запускает аудио поток;
   - занимается пересылкой (листинг 1);
   - в нем также формируются данные для генерации эквалайзера (листинг 2);
   - изменяет громкость (листинг 3).

Листинг 1: Пересылка данных

```
# функция посылки строки определенной длины
def send(sock, data):
    length = len(data)

    sock.sendall(pack('!I', length))
    sock.sendall(data)
```

Листинг 2: Генерация эквалайзера

```
# транслируем байтовый массив в числовой
dataint = np.array(struct.unpack(str(2*CHUNK) + 'B', data),
                                 dtype = 'b')[::2] + 128
# производим дискретное преобразование Фурье
yf = fft(dataint)
# строим график
line_fft.set_ydata(np.abs(yf[0:CHUNK])  / (64 * CHUNK))
```

Листинг 3: Изменение громкости

```
# получаем текущее значение громкости
volume = int(val)/100.
# транслируем байты в числа
chunk = np.frombuffer(data, np.uint8)
# изменяем громкость
chunk = chunk * volume
# транслируем обратно в байты
data = chunk.astype(np.uint8)
```

3) Модуль пользовательского интерфейса

   Пользовательский интерфейс осуществляет взаимодействие пользователя с программой. Пользовательский интерфейс представлен главной формой приложения, которая содержит кнопки для управления аудио потоком.

Описание модулей клиентского приложения

Клиентское приложение абстрактно можно представить в виде трех модулей.

1) Модуль создания соединений – один из основных модулей клиентского приложения. Он получает информацию о сервере – адрес и порт, в нем также инициализируется сокет для получения данных от сервера.

2) Модуль проигрывания аудио:

   - считывает аудио файл;
   - запускает аудио поток;
   - получает аудио данные и записывает их в поток (листинг 4);
   - в нем также формируются данные для генерации эквалайзера.

Листинг 4: Получение данных

```
# функция получения сообщения
def get(sock):
    lenbuf = recvall(sock, 4)
    len, = unpack('!I', lenbuf)

    return recvall(sock, len)
# получение строки по длине
def recvall(sock, count):
    buf  = b''

    while count:
        newbuf = sock.recv(count)
```

```
        if not newbuf:
            return None

        buf +=  newbuf
        count -=len(newbuf)

    return buf
```

3) Модуль пользовательского интерфейса

   Пользовательский интерфейс осуществляет взаимодействие пользователя с программой. Пользовательский интерфейс представлен главной формой приложения, которая содержит поле, содержащее список пользователей и набор кнопок для управления процессом аудиопередачи.

Тестирование

Было проведено функциональное тестирование белого ящика, в ходе которого программа отработала правильно. Также проведено тестирование пользовательского интерфеиса, все элементы интерфеиса реагируют корректно. Демонстрация различных сценариев работы приведена в пункте "Интерфеис приложения". Ожидаемое поведение приложения совпадает с полученными результатами. Программа показала полную работоспособность реализованного функционала.

Вывод

В данном разделе были выбраны средства реализации, рассмотрен интерфейс программы, а также проведено тестирование.

Заключение

В ходе проведенной работы были выполнены следующие задачи:

- проанализированы существующие стриминговые сервисы;
- проанализированы протоколы локальных сетей транспортного уровня;
- выбран протокол потокового аудио для организации трансляции;
- спроектирован программный комплекс для реализации поставленной цели;
- выбраны средства реализации для спроектированного программного комплекса;
- разработан и протестирован программный комплекс.

Таким образом, достигнута цель - разработан программный комплекс, обеспечивающий трансляцию аудио-файлов с одного компьютера на несколько устройств. В будущем планируется дальнеишее развитие проекта. Среди будущих направлений развития хочется выделить следующие:

- переход на web-интерфеис для большего удобства, массовости продукта и комфортного сбора статистики, и, соответственно, переход на другой протокол;
- написание серверноии части на Java, так как скорость и стабильность работы краине важна в данном проекте;
- поддержка кроссплатформенности;
- расширение функционала приложения.

Литература

1. Доходы от музыкального стриминга в России удвоились в 2016 году. [Электронный ресурс]. – Режим доступа: https://www.vedomosti.ru/technology/articles/2017/05/03/688474-dohodi-striminga (дата обращения: 05.12.2020)

2. Настройка DLNA клиента VLC Media Player [Электронный ресурс]. – Режим доступа: https://media-player-s.ru/vlc-dlna (дата обращения: 07.12.2020)

3. Сокеты [Электронный ресурс]. – Режим доступа: https://lecturesnet.readthedocs.io/net/low-level/ipc/socket/intro.html (дата обращения: 01.12.2020)

4. Таненбаум Э. Компьютерные сети. – СПб.: Питер, 2003. – 992 с. (дата обращения: 28.11.2020)

5. Эндрю Кровчик, Винод Кумар, Номан Лагари и др. – .NET Сетевое программирование для профессионалов. – Издательство «ЛОРИ», 2005. – 400стр. (дата обращения: 30.11.2020)

6. How to Play Sound in Python [Электронный ресурс]. – Режим доступа: https://linuxhint.com/play_sound_python/ (дата обращения: 24.11.2020)

7. PyAudio Documentation [Электронный ресурс]. – Режим доступа: https://people.csail.mit.edu/hubert/pyaudio/docs/ (дата обращения: 01.12.2020)

8. threading — Thread-based parallelism [Электронный ресурс]. – Режим доступа: https://docs.python.org/3/library/threading.html (дата обращения: 04.12.2020)