МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Забайкальский государственный университет»
(ФГБОУ ВПО «ЗабГУ»)
Факультет: Энергетический
Кафедра: Информатики, вычислительной техники и прикладной математики

КУРСОВОЙ ПРОЕКТ

По дисциплине: Протоколы вычислительных сетей

На тему: «Реализация мобильного приложения с обменом информацией по групповому вещанию»

Выполнил студент группы ВМК–21, Пуртов Георгий Андреевич

Руководитель работы: старший преподаватель кафедры ИВТ и ПМ, Забелин Вячеслав Олегович

Чита
2024

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Забайкальский государственный университет»
(ФГБОУ ВПО «ЗабГУ»)
Факультет: Энергетический
Кафедра: Информатики, вычислительной техники и прикладной математики

ЗАДАНИЕ

на курсовой проект

По дисциплине: Протоколы вычислительных сетей

Студенту: Пуртову Георгию Андреевичу

Специальности (направления подготовки): Вычислительные машины и комплексы

1 Тема курсовой работы: «Реализация мобильного приложения с обменом информацией по групповому вещанию»

2 Срок подачи студентом законченной работы: 20.05.2024

3 Исходные данные к работе: описание предметной области

Дата выдачи задания: 15.02.2024

Руководитель курсовой работы_____/Забелин В.О./

(подпись, расшифровка подписи)

Задание принял к исполнению

«15» февраля 2024 г.

Подпись студента_____ /Пуртов Г.А. /

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Забайкальский государственный университет»
(ФГБОУ ВПО «ЗабГУ»)
Факультет: Энергетический
Кафедра: Информатики, вычислительной техники и прикладной математики

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

По дисциплине: Протоколы вычислительных сетей

На тему: «Реализация мобильного приложения с обменом информацией по групповому вещанию

Выполнил студент группы ВМК-21, Пуртов Георгий Андреевич

Руководитель работы: старший преподаватель кафедры ИВТ и ПМ, Забелин Вячеслав Олегович

Чита
2024

КАЛЕНДАРНЫЙ ГРАФИК

выполнения курсового проекта

УТВЕРЖДАЮ

Зав.кафедрой_____

«___» _____ 2024 г.

Этапы выполнения курсовой работы	Месяцы и недели															
	Февраль			Март				Апрель				Май				
1. Получение задания на курсовую работу	+															
2. Анализ задачи		+														
3. Анализ данных			+													
4. Программная реализация				+	+	+	+									
5. Тестирование								+	+							
6. Документирование										+	+					
7. Представление руководителю чернового варианта работы												+				
8. Корректировка работы в соответствии с замечаниями руководителя													+	+		
9. Защита работы																+

План выполнен: руководитель _____

(подпись, расшифровка подписи)

«___» _____ 2024 г

СОДЕРЖАНИЕ

РЕФЕРАТ

Пояснительная записка - 29 с, 10 рис., 5 источников.

ГРУППОВОЙ ЧАТ, ЯЗЫК JAVA, IGMP, UDP, PEER-TO-PEER, ANDROID, ПАТТЕРН «НАБЛЮДАТЕЛЬ».

В данной работе рассматривается процесс создания peer-to-peer приложения, являющегося групповым чатом на основе протокола IGMP и UDP, на языке программирования Java, для ОС Android.

В работе определены методы разработки и описан процесс их применения при описании группового чата на основе IGMP и UDP для Android.

ВВЕДЕНИЕ

Цель данной работы – реализовать мобильное приложение с графическим интерфейсом, для обмена сообщениями между пользователями в локальной сети без использования отдельно выделенного сервера обработки сообщений, при помощи группового вещания и протокола транспортного уровня – UDP.

Задачи:

1) Изучить понятие «peer-to-peer» приложения с использованием группового вещания, и основы сетевого программирования на языке Java на платформе Android;

2) Разобрать работу протокола IGMP и UDP;

3) Разработать программный комплекс для реализации peer-to-peer (P2P) приложения на платформе Android с использованием группового вещания и протокола UDP для обмена данными между клиентскими устройствами в локальной сети.

1. Теоретическая часть

1.1 Описание предметной области

Мобильные приложения становятся все более популярными для обмена данными и коммуникаций между устройствами. В контексте P2P (peer-to-peer) приложений, которые позволяют устройствам обмениваться информацией напрямую, важно использовать соответствующие протоколы для обеспечения эффективной коммуникации. Одними из таких протоколов являются IGMP (Internet Group Management Protocol) [5] и UDP (User Datagram Protocol) [1].

Обзор протоколов:

IGMP – это протокол управления группами интернета, который используется для управления мультимедийными потоками в IP-сетях. Он позволяет устройствам в сети определить, к какой группе IP-адресов они принадлежат, и присоединиться к мультимедийным потокам, передаваемым в этой группе.

UDP – это протокол пользовательских датаграмм, который предоставляет простую и быструю доставку данных без необходимости установления соединения, в отличие от TCP.

UDP идеально подходит для P2P приложений, где требуется низкая задержка и возможность передачи данных напрямую между устройствами. Протокол UDP (User Datagram Protocol) является одним из ключевых компонентов семейства интернет-протоколов, обеспечивающих передачу данных в компьютерных сетях. Этот протокол описан в документе RFC 768 [4] и представляет собой простой транспортный протокол, работающий поверх IP (Internet Protocol). UDP использует модель передачи без установления соединения, что позволяет отправлять датаграммы (небольшие блоки данных) без предварительного согласования состояния с приёмной стороной. Эта особенность делает протокол идеальным для сценариев, где важна скорость передачи за счет потенциальной потери надежности.

В отличие от протокола TCP (Transmission Control Protocol), UDP не гарантирует доставку, порядок следования пакетов, отсутствие дубликатов и контроль целостности данных на уровне транспортного протокола. Каждый UDP-пакет включает в себя всего четыре поля в своем заголовке: порт источника, порт назначения, длину датаграммы и контрольную сумму. Такая минималистичная структура заголовка (всего 8 байт) способствует уменьшению общего объема служебных данных и, соответственно, увеличению полезной пропускной способности сети. Однако отсутствие механизмов восстановления порядка и проверки доставки означает, что вышележащие уровни приложений или протоколы должны самостоятельно решать задачи обеспечения надежности, если это необходимо.

Применение UDP оправдано в случаях, когда требуется быстрая передача данных с приемлемыми потерями, например, в реальном времени для видео- и аудиопотоков, онлайн-игр, или для протоколов, в которых реализована собственная система подтверждения доставки и восстановления данных, как, например, в случае с DNS-запросами или при использовании протокола SNMP (Simple Network Management Protocol). В таких ситуациях UDP позволяет достичь меньшей задержки передачи данных по сравнению с TCP, благодаря отсутствию процедур рукопожатия и управления потоком, что делает его незаменимым в условиях, когда время имеет критическое значение.

Рис.1 Модель OSI

Internet Group Management Protocol (IGMP) – это сетевой протокол, используемый для управления членством в многоадресной (multicast) рассылке в сетях IP. Он функционирует на сетевом уровне (уровень 3 модели OSI) и предназначен для контроля маршрутизаторами активности участников многоадресных групп. IGMP позволяет устройствам в сети сообщать маршрутизаторам о своем намерении присоединиться к определенной группе или покинуть ее. Это обеспечивает оптимизацию маршрутизации трафика мультимедийного контента и других данных, отправляемых по технологии IP-мультикаста. Протокол определяет несколько типов сообщений: общие запросы (General Queries), специфические запросы групп (Group-Specific Queries), сообщения о присоединении (Membership Reports) и сообщения о выходе (Leave Group). Эти сообщения позволяют маршрутизаторам эффективно управлять членством в группах.

Сущность IGMP заключается в предоставлении механизма для эффективной доставки данных многим получателям одновременно без необходимости отправлять отдельные копии данных каждому из них.

Протокол обеспечивает возможность маршрутизаторам собирать информацию о присоединении и выходе устройств из групп многоадресной рассылки. Существуют три версии протокола: IGMPv1 предоставляет базовую функциональность управления членством, включая только общие запросы и сообщения о присоединении. IGMPv2 добавляет возможность быстрого выхода из группы, улучшенное управление тайм-аутами и специфические запросы групп. IGMPv3 существенно расширяет возможности протокола, позволяя клиентам указывать конкретные источники, из которых они хотят получать трафик (функциональность known-source multicast или SSM). Это позволяет более точно контролировать трафик и уменьшить количество нежелательных данных.

Область применения IGMP наиболее ярко выражена в сетях, где важна эффективная доставка мультимедийного контента. Протокол широко используется в сервисах IPTV, видеоконференциях, онлайн-играх и других приложениях, где требуется передача данных от одного источника к нескольким получателям. Благодаря использованию IGMP и технологии IP-мультикаста, сетевой трафик значительно оптимизируется, что снижает нагрузку на сеть и улучшает качество обслуживания. Например, в системах IPTV маршрутизаторы могут отправлять телевизионный контент только в те сегменты сети, где есть зрители, подключенные к соответствующему каналу. Однако использование IGMP требует внедрения мер безопасности, чтобы предотвратить возможные атаки, такие как ложные запросы на участие в группах. Для защиты сетей рекомендуется использовать фильтрацию многоадресных групп, а также более продвинутые протоколы управления, такие как MLD (Multicast Listener Discovery) в IPv6-сетях.

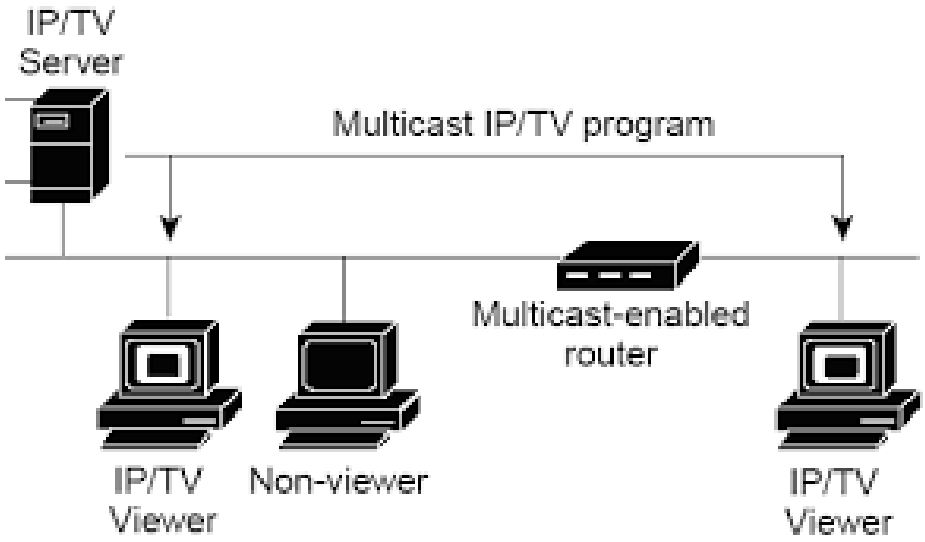

Рис.2 Организация IPTV через IGMP Snooping

1.2 Структура UDP

Заголовок UDP сегмента имеет следующую структуру:

Биты	0 - 15	16 - 31
0-31	Порт отправителя (Source port)	Порт получателя (Destination port)
32-63	Длина датаграммы (Length)	Контрольная сумма (Checksum)
64-...	Данные (Data)	

Рис.3 Структура UDP-заголовка

Порт источника (Source Port)

- Диапазон используемых бит: 0-15
- Определяет порт отправителя, позволяя идентифицировать приложение, отправляющее дейтаграмму. Это поле является опциональным, и если оно не используется, должно быть установлено в ноль.

Порт назначения (Destination Port)

- Диапазон используемых бит: 16-31

- Указывает порт, на который должна быть отправлена дейтаграмма. Порт назначения идентифицирует приложение-получатель на целевом хосте.

Длина (Length)

- Диапазон используемых бит: 32-47
- Указывает общую длину дейтаграммы в байтах, включая заголовок и данные. Минимальное значение этого поля – 8 байт, что соответствует длине только заголовка. Максимальное значение – 65 535 байт, но фактический размер дейтаграммы может быть ограничен максимальным размером кадра (MTU) сетевой технологии.

Контрольная сумма (Checksum)

- Диапазон используемых бит: 48-63
- Используется для проверки целостности заголовка и данных UDP. Контрольная сумма вычисляется на основе псевдозаголовка (pseudo-header), включающего IP-адреса источника и назначения, протокольный номер (равен 17 для UDP) и длину UDP-пакета, а также самих данных UDP-дейтаграммы.

Примечание: Контрольная сумма является опциональной в IPv4 (если не используется, значение устанавливается в ноль) и обязательной в IPv6.

Псевдозаголовок (Pseudo-header) не является частью UDP-заголовка, но включается в расчет контрольной суммы. Его структура зависит от версии протокола IP.

IPv4 Псевдозаголовок:

- IP-адрес источника (Source IP Address): 32 бита.
- IP-адрес назначения (Destination IP Address): 32 бита.

- Нулевой байт (Zero Byte): 8 бит.
- Протокол (Protocol): 8 бит (значение 17 для UDP).
- Длина UDP (UDP Length): 16 бит.

IPv6 Псевдозаголовок:

- IP-адрес источника (Source IP Address): 128 бит.
- IP-адрес назначения (Destination IP Address): 128 бит.
- Длина UDP (UDP Length): 32 бита.
- Нулевые байты (Zero Bytes): 24 бита.
- Протокол (Next Header): 8 бит (значение 17 для UDP).

1.3 Структура IGMP

Структура IGMP-сегмента отличается в зависимости от версии протокола (IGMPv1, IGMPv2 и IGMPv3). Далее рассмотрим структуру сегмента IGMPv2, который используется в данной курсовой работе.

Структура IGMPv2-сегмента:

Тип сообщения (Type)

- Диапазон бит: 0–7.
- Определяет тип сообщения IGMP:
  - 0x11: Membership Query (запрос членства);
  - 0x16: Version 2 Membership Report (отчет о членстве версии 2);
  - 0x17: Leave Group (выход из группы);
  - 0x12: Version 1 Membership Report (отчет о членстве версии 1).

Максимальное время ответа (Max Response Time)

- Диапазон бит: 8–15.
- Используется только в сообщениях типа Membership Query. Указывает максимальное время (в десятых долях секунды), в течение

которого должен быть отправлен ответ. Для других типов сообщений это поле установлено в ноль.

Контрольная сумма (Checksum)

- Диапазон бит: 16–31.
- 16-битное значение, рассчитанное по алгоритму контрольной суммы, определенному в RFC 1071. Включает тип сообщения, максимальное время ответа и адрес группы.

Адрес группы (Group Address)

- Диапазон бит: 32–63.
- Представляет собой IP-адрес группы, к которой относится сообщение. В запросах Membership Query это поле может быть установлено в ноль для общих запросов или содержать адрес конкретной группы для специфических запросов. В отчетах Membership Report и сообщениях Leave Group содержит IP-адрес соответствующей группы.

Рис.4 Структура IGMPv2 пакета

2. Практическая часть

2.1 Описание проекта

Данное приложение реализовано на языке Java [3] для платформы Android, в среде разработки Android Studio Iguana 2023.2.1 [2]. В ходе разработки программного обеспечения были реализованы классы MainActivity, MessageHandler, MulticastManager, UdpChat, UdpManager, DeviceIPAdapter, интерфейса MessageObserver.

Информация передается с использованием протоколов транспортного уровня (UDP) и сетевого уровня (IGMP) модели OSI.

Код структурирован посредством использования классов и интерфейсов. В данной архитектуре пользовательский интерфейс реализуется в активностях, в то время как основная логика приложения инкапсулируется в отдельных классах. Такой подход способствует эффективной архитектурной организации программного обеспечения, обеспечивая разделение ответственности, улучшение модульности и повышение уровня повторного использования кода.

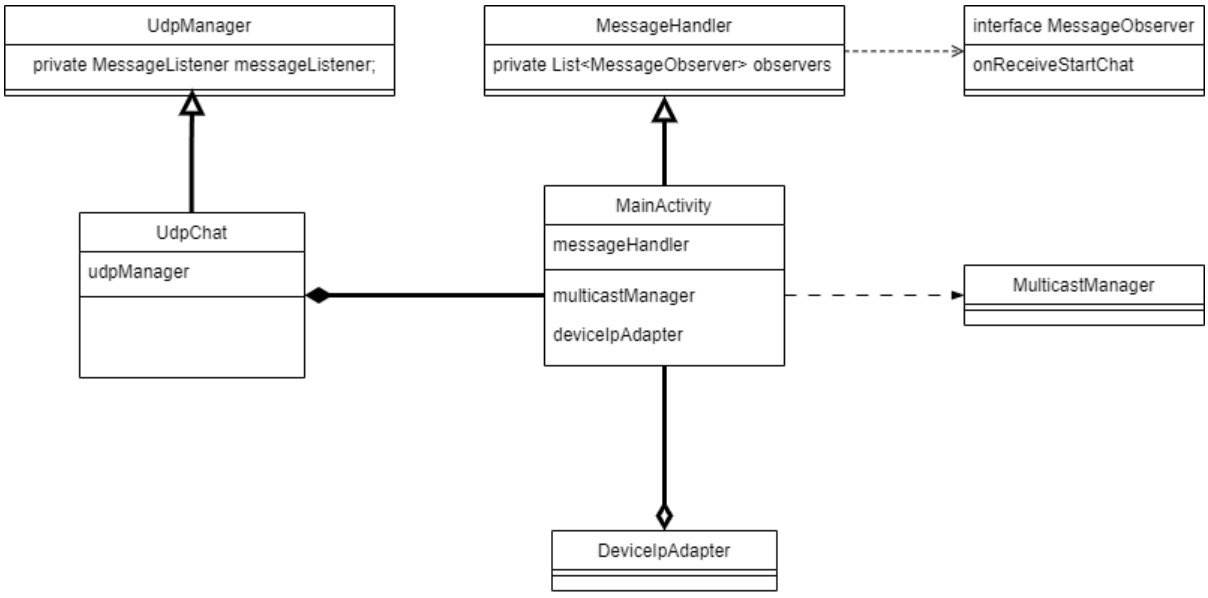

Рис.5 Диаграмма классов приложения

Проект представляет собой приложение на языке Java для обмена сообщениями через UDP и мультикаст-сети, предназначенное для мобильных устройств с операционной системой Android. Основной класс MainActivity обеспечивает пользовательский интерфейс, в котором реализованы основные функции приложения, включая ввод никнейма, управление соединениями и отображение списка подключенных устройств. Пользователь может вводить свое имя, инициировать и принимать чаты через UDP, а также обновлять список доступных устройств. Восстановление состояния приложения происходит посредством сохранения и загрузки данных в SharedPreferences.

Класс MessageHandler выполняет роль обработчика сообщений, принимаемых и отправляемых через мультикаст-сокеты. Он управляет картой соответствия IP-адресов и никнеймов подключенных устройств и реализует механизм наблюдателей для уведомления о событиях начала чата. MessageHandler также включает методы для обработки сообщений IGMP, которые используются для объявления присутствия в сети (hello-сообщения), и управления UDP-приемником для прослушивания определенного порта. Список наблюдателей позволяет различным компонентам приложения реагировать на изменения состояния и поступление новых сообщений.

Интерфейс MessageObserver определяет метод onReceiveStartChat(String ipAddress), который вызывается для уведомления наблюдателей о начале нового чата. Этот интерфейс используется для реализации паттерна "Наблюдатель" в проекте, позволяя объектам, реализующим этот интерфейс, регистрироваться в качестве наблюдателей у MessageHandler и получать уведомления о важных событиях, таких как начало чата. Это способствует гибкости и расширяемости системы, поскольку любые компоненты могут легко подписаться на уведомления о начале чата, реализовав данный интерфейс.

Класс UDPChat представляет собой активность, которая обеспечивает интерфейс чата между пользователями, подключенными через UDP. Он управляет отправкой и приемом сообщений в реальном времени, отображает

полученные сообщения и обеспечивает корректную обработку событий, таких как нажатие кнопок и изменение состояния видимости клавиатуры. Класс использует объект UdpManager для управления сокетами и обмена сообщениями. UdpManager отвечает за низкоуровневую работу с UDP-сокетами, включая отправку и прием сообщений, а также уведомление слушателей об получении новых сообщений. Таким образом, проект демонстрирует структуру с четким разделением ответственности между компонентами, обеспечивая масштабируемость и возможность добавления новых функций.

2.2 Руководство пользователя

При первом запуске любой пользователь видит стартовый экран (рис. 4) с стандартным именем пользователя («никнеймом») User, кнопкой Submit, переключателем режима Discover и кнопку Refresh. Пользователь вводит ник, нажимает Submit (устанавливает его в качестве своего имени), включает режим Discover (готовность вступить в чат, старт рассылки IGMP). После готовности к выходу с сеть, пользователь периодически нажимает Refresh, пока не увидит нужного собеседника (рис.6). После того, как нужный собеседник появился в верхней части экрана, нужно нажать на него и подтвердить свой выбор (рис.5).

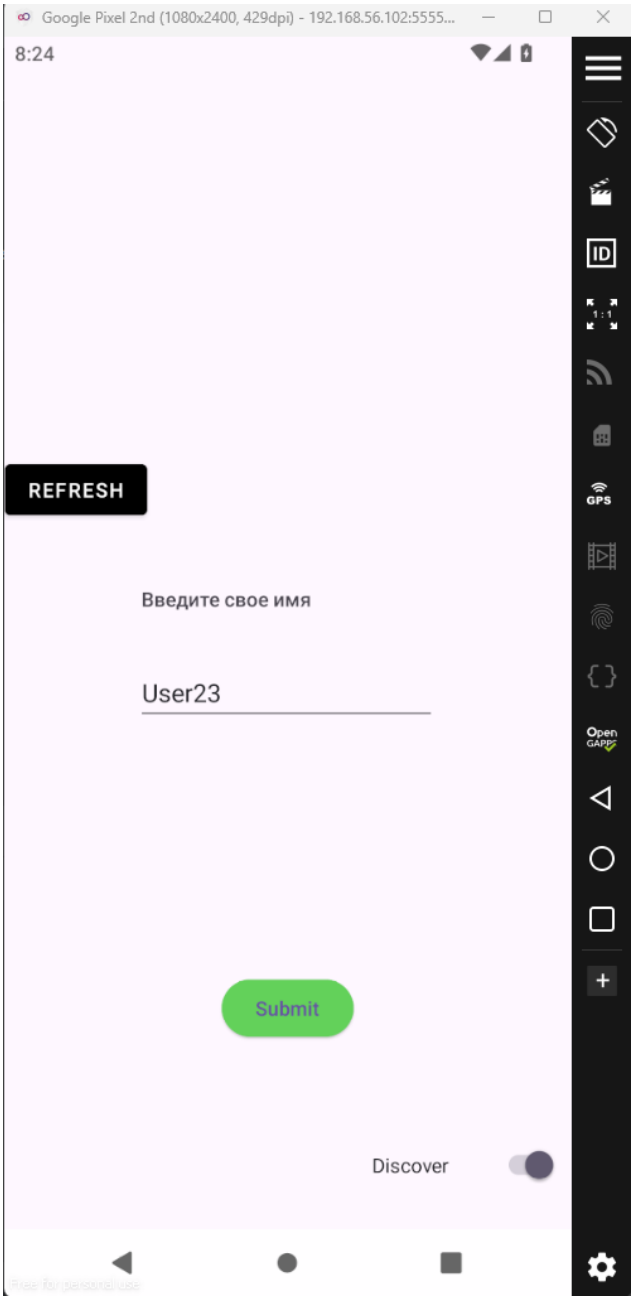

Рис.6 Основной экран приложения

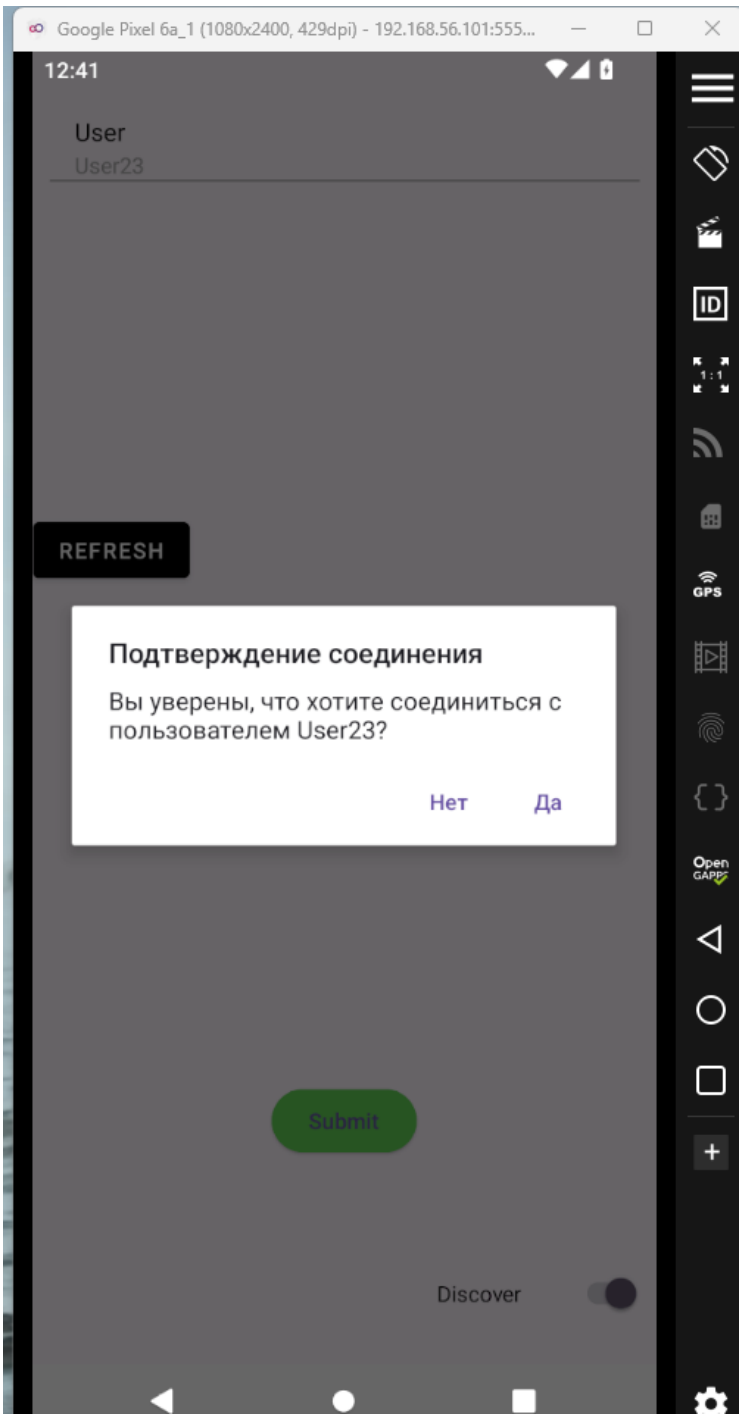

Рис.7 Подтверждение выбора собеседника

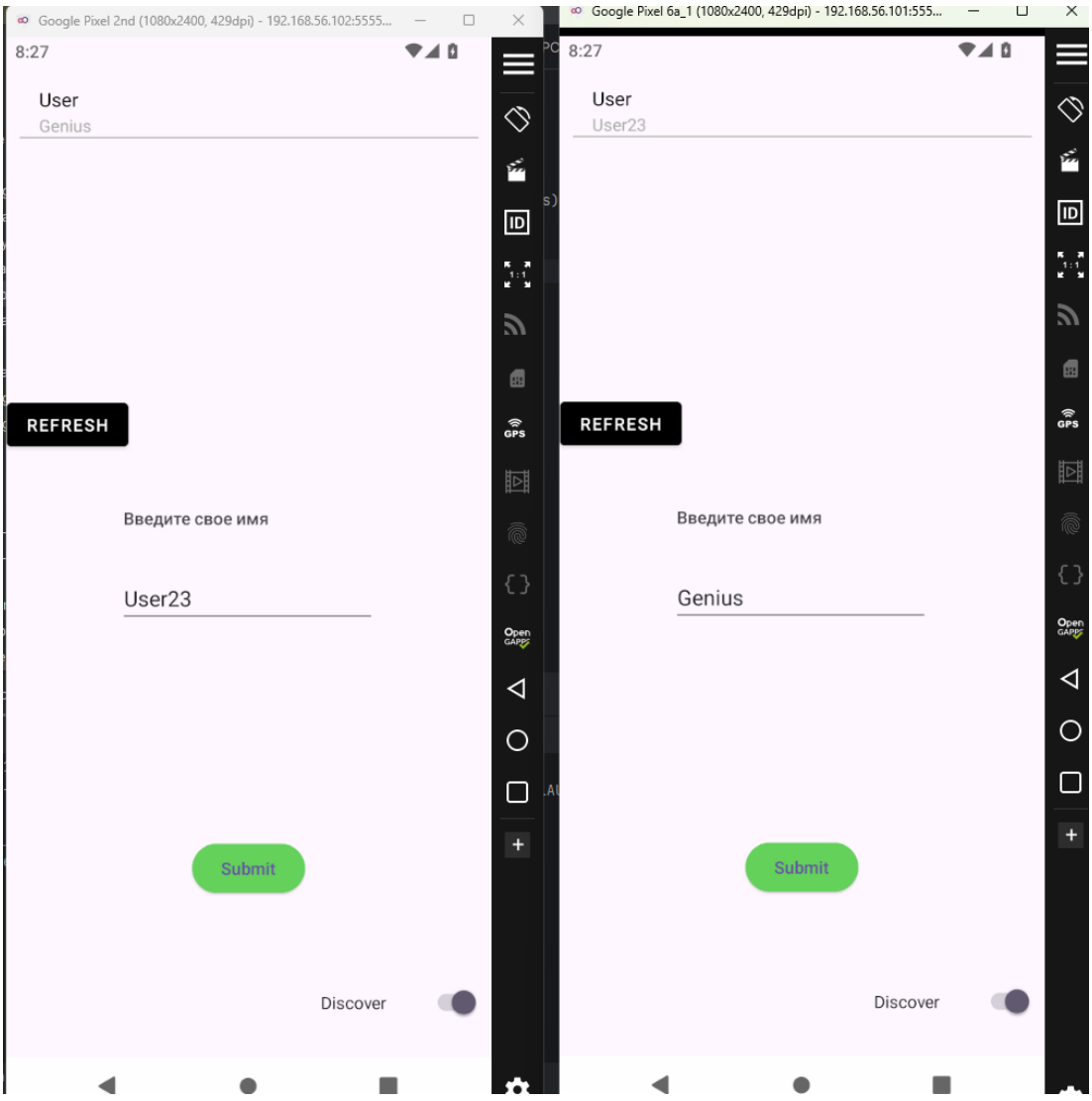

Рис.8 Основной экран приложения с потенциальными участниками чата

После подтверждения соединения на стороне инициатора, у второго участника переход на экран чата произойдет автоматически, теперь можно выбрать поле ввода нажатием на него, ввести сообщение и отправить его нажатием на кнопку “Send” (рис. 7).

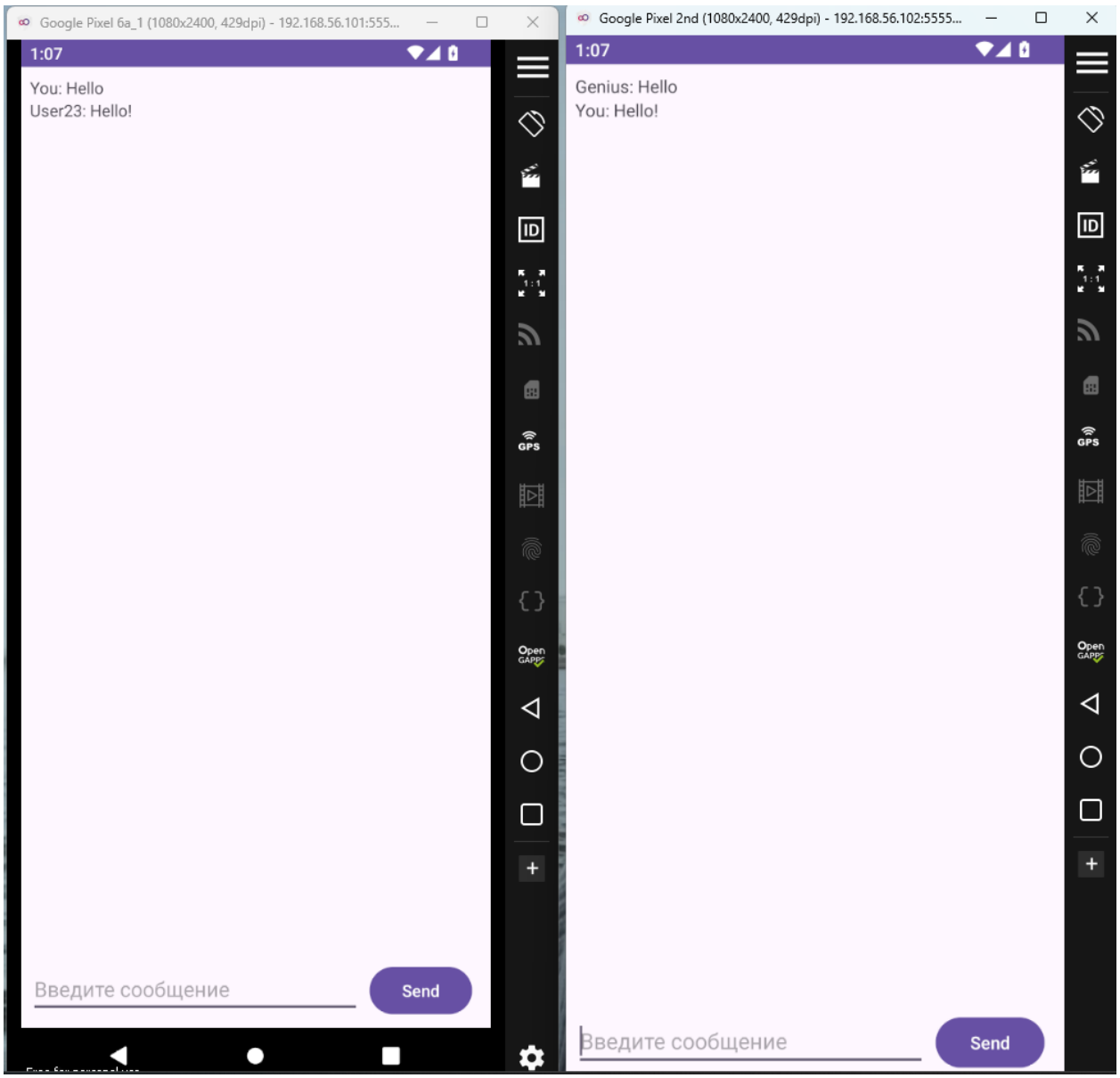

Рис.9 Экран чата с отправленными сообщениями

Когда пользователь решит прекратить чат, нужно дважды нажать на кнопку «Назад» на устройстве, его собеседник получит уведомление о том что он остался один, и ему будет предложено покинуть текущий диалог.

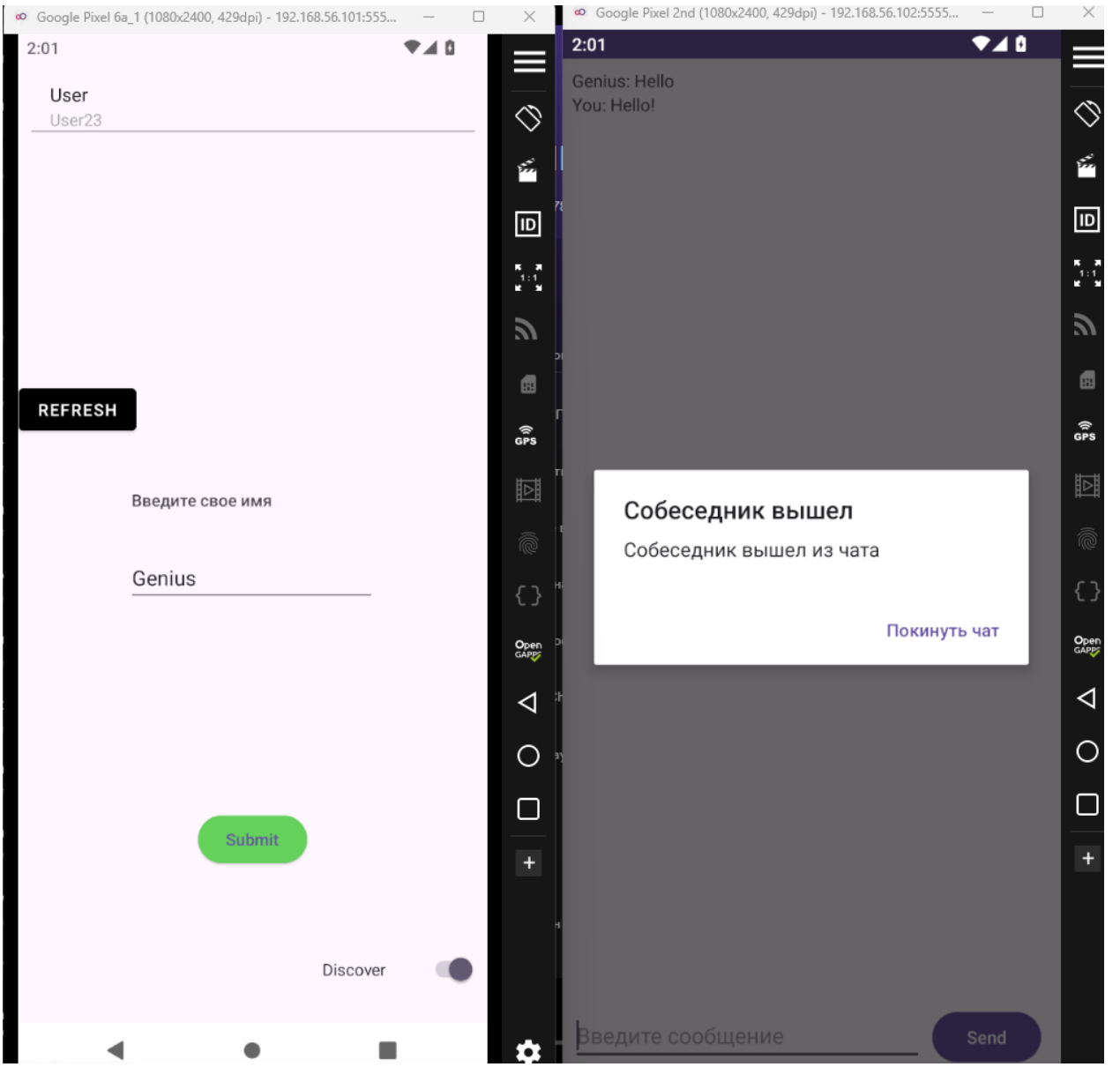

Рис.10 Экран чата с уведомлением о выходе

ЗАКЛЮЧЕНИЕ

В ходе выполнения данной курсовой работы было разработано и реализовано мобильное peer-to-peer (P2P) приложение для ОС Android, функционирующее как групповой чат на основе протоколов IGMP и UDP. Основная цель заключалась в создании надежной и эффективной платформы для обмена сообщениями в реальном времени между несколькими пользователями, подключенными к одной мультикаст группе.

Спроектирована и реализована архитектура приложения, обеспечивающая взаимодействие между компонентами для установления и поддержания связи в групповом чате. Применены принципы объектно-ориентированного программирования для повышения модульности и поддерживаемости кода.

Реализованы механизмы подписки на мультикаст группы с использованием IGMP для эффективного распространения сообщений, а протокол UDP использован для передачи сообщений, что позволило обеспечить минимальные задержки при обмене данными. Разработан интуитивно понятный интерфейс, позволяющий пользователям легко присоединяться к групповым чатам, отправлять и получать сообщения.

Проведено всестороннее тестирование приложения в различных сетевых условиях для обеспечения надежности и устойчивости работы и на основе него внесены коррективы для улучшения производительности и устранения ошибок.
Разработанное приложение демонстрирует высокую эффективность и надежность при обмене сообщениями в реальном времени, что достигается за счет использования протоколов IGMP и UDP. Данное решение позволяет создать основу для дальнейших улучшений и расширений функциональности, таких как добавление шифрования сообщений,

улучшение пользовательского интерфейса и интеграция с другими сервисами.

Работа по разработке группового чата на основе P2P технологии для ОС Android завершена успешно, и приложение готово к дальнейшему использованию и развитию. Полученные результаты показывают, что выбранный подход и технологии оправдывают себя в контексте мобильных сетевых приложений, требующих быстрой и эффективной передачи данных, без необходимости использования дополнительного сервера обработки данных.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1) 2.2.6. User Datagram Protocol – udp [Электронный ресурс] – URL: https://studfile.net/preview/7390796/page:6/ (Дата обращения: 25.02.2024)

2) Документация Android Studio [Электронный ресурс] – URL: https://developer.android.com/guide (Дата обращения: 03.03.2024)

3) Документация Java [Электронный ресурс] – URL: docs.oracle.com/javase/8/docs/api/java/net/MulticastSocket.html (Дата обращения: 04.03.2024)

4) Компьютерные сети. Принципы, технологии, протоколы: Юбилейное издание [Текст]. Учебник для вузов. / Олифер Виктор, Олифер Наталья – СПб.: Питер, 2020 – 1008. (Дата обращения 01.04.2024)

5) Протокол igmp [Электронный ресурс] – URL: https://studfile.net/preview/8864966/page:70/ (Дата обращения: 29.02.2024)

ПРИЛОЖЕНИЕ

Код класса работы с протоколом IGMP:

```
public class MulticastManager {
    private MulticastSocket socket;
    private InetAddress multicastGroup;
    private int multicastPort;

    public MulticastManager(String multicastGroup, int multicastPort) {
        try {
            this.multicastGroup = InetAddress.getByName(multicastGroup);
        } catch (UnknownHostException e) {
            throw new RuntimeException(e);
        }
        this.multicastPort = multicastPort;
    }

    public void connect() throws IOException {
        socket = new MulticastSocket(multicastPort);
        socket.joinGroup(multicastGroup);
        Log.d("MulticastManager", "Connected to multicast group " +
multicastGroup);
    }

    public void disconnect() {
        if (socket != null && !socket.isClosed()) {
            try {
                socket.leaveGroup(multicastGroup);
                socket.close();
                Log.d("MulticastManager", "Disconnected from multicast
group");
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }

    public MulticastSocket getSocket() {
        return socket;
    }

    public InetAddress getMulticastGroup() {
        return multicastGroup;
    }

    public int getMulticastPort() {
        return multicastPort;
    }
}
```

Код класса работы с протоколом UDP:

```
public class UdpManager {
    private static final String TAG = "UdpManager";
    private DatagramSocket socket;
    private boolean running;
    private String ipAddress;
    private int port;
    private String nickname;
```

```
    private MessageListener messageListener;
    private Handler handler;

    public UdpManager(String ipAddress, int port, String nickname) {
        this.ipAddress = ipAddress;
        this.port = port;
        this.nickname = nickname;
        running = false;
        handler = new Handler(Looper.getMainLooper());
    }

    public void setMessageListener(MessageListener listener) {
        this.messageListener = listener;
    }

    public void startListening() {
        if (!running) {
            new Thread(() -> {
                try {
                    socket = new DatagramSocket(port);
                    byte[] buffer = new byte[1024];
                    DatagramPacket packet = new DatagramPacket(buffer,
buffer.length);
                    running = true;
                    while (running) {
                        socket.receive(packet);
                        String message = new String(packet.getData(), 0,
packet.getLength());
                        Log.d(TAG, "Received message: " + message);
                        if (messageListener != null) {
                            // Оповещаем слушателя о новом сообщении на
основном потоке
                            handler.post(() ->
messageListener.onMessageReceived(message));
                        }
                    }
                } catch (IOException e) {
                    Log.e(TAG, "Error while listening for UDP messages: " +
e.getMessage());
                }
            }).start();
        }
    }

    public void stopListening() {
        running = false;
        if (socket != null) {
            socket.close();
        }
    }

    public void sendMessage(String message) {
        new Thread(() -> {
            try {
                InetAddress address = InetAddress.getByName(this.ipAddress);
                String finalMessage = this.nickname + ": " + message;
                byte[] sendData = finalMessage.getBytes();
                DatagramPacket sendPacket = new DatagramPacket(sendData,
sendData.length, address, port);
                DatagramSocket sendSocket = new DatagramSocket();
                sendSocket.send(sendPacket);
                sendSocket.close();
```

```
                Log.d(TAG, "Sent message: " + finalMessage + " " +
this.ipAddress);
            } catch (IOException e) {
                Log.e(TAG, "Error while sending UDP message: " +
e.getMessage());
            }
        }).start();
    }

    public void sendLeaveMessage() {
        new Thread(() -> {
            try {
                InetAddress address = InetAddress.getByName(this.ipAddress);
                String finalMessage = "CODE___200___EXIT";
                byte[] sendData = finalMessage.getBytes();
                DatagramPacket sendPacket = new DatagramPacket(sendData,
sendData.length, address, port);
                DatagramSocket sendSocket = new DatagramSocket();
                sendSocket.send(sendPacket);
                sendSocket.close();
                Log.d(TAG, "Sent leave message: " + finalMessage + " " +
this.ipAddress);
            } catch (IOException e) {
                Log.e(TAG, "Error while sending UDP leave message: " +
e.getMessage());
            }
        }).start();
    }

    public interface MessageListener {
        void onMessageReceived(String message);
    }
}
```