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: ассистент 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: ассистент 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 | | | | | Июнь | |
| 1. Получение задания на курсовую работу | ++ | + |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 1. Анализ задачи |  | ++ | ++ |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 1. Анализ данных |  |  | + | ++ | ++ | + |  |  |  |  |  |  |  |  |  |  |  |  |
| 1. Программная реализация |  |  | + | + |  | ++ | ++ | ++ | ++ | ++ | ++ |  |  |  |  |  |  |  |
| 1. Тестирование |  |  |  | + |  |  |  |  |  | 3+ | ++ | ++ |  |  |  |  |  |  |
| 1. Документирование |  |  |  |  |  |  |  |  |  |  |  |  | ++ | ++ | + |  |  |  |
| 1. Представление руководителю чернового варианта работы |  |  |  |  |  |  |  |  |  |  |  |  |  |  | ++ | + |  |  |
| 1. Корректировка работы в соответствии с замечаниями руководителя |  |  |  |  |  |  |  |  |  |  |  |  |  |  | + | ++ | ++ |  |
| 1. Защита работы |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | + |  | ++ |

План выполнен: руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись, расшифровка подписи)

**«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2024 г**

СОДЕРЖАНИЕ

ВВЕДЕНИЕ7

1 Теоретическая часть8

1.1 Описание предметной области8

1.2 Структура TCP10

2 Практическая часть12

2.1 Описание проекта12

2.2 Руководство пользователя13

ЗАКЛЮЧЕНИЕ17

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 18

ПРИЛОЖЕНИЕ19

# РЕФЕРАТ

Документ – 19 с, 8 рис., 4 источника.

ГРУППОВОЙ ЧАТ, ЯЗЫК JAVA, IGMP, UDP, PEER-TO-PEER, ANDROID.

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;

1. Разобрать работу протокола IGMP и UDP;
2. Разработка программного комплекса 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) и UDP (User Datagram Protocol).

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 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.

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.

**1.2 Структура UDP**

Заголовок UDP сегмента имеет следующую структуру:

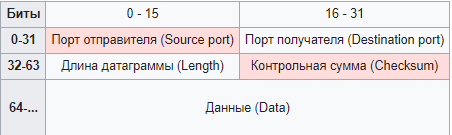


Рис.1 Структура UDP-заголовка

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)

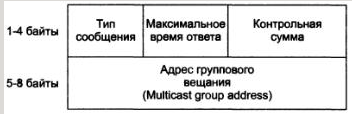
* Диапазон бит: 8–15.
* Используется только в сообщениях типа Membership Query. Указывает максимальное время (в десятых долях секунды), в течение которого должен быть отправлен ответ. Для других типов сообщений это поле установлено в ноль.

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.2 Структура IGMPv2 пакета

**2.Практическая часть**

**2.1 Описание проекта**

Данное приложение реализовано на языке Java для платформы Android, в среде разработки Android Studio Iguana 2023.2.1, с использованием классов MainActivity, MessageHandler, MulticastManager, UdpChat, UdpManager, DeviceIPAdapter, интерфейса MessageObserver.

Информация передаётся при помощи протоколов сетевого и траспортного уровня модели OSI – UDP и IGMP.

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.

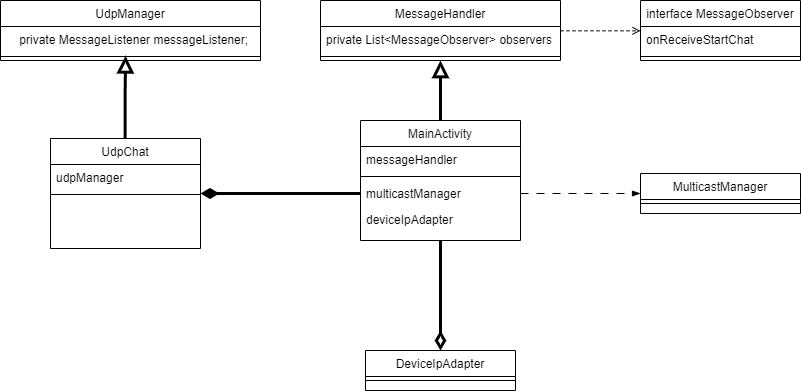


Рис.3 Диаграмма компонентов приложения

Основное взаимодействие происходит в MainActivity, код этого класса описывает последовательность действий, которые необходимы для установления и поддержания связи с групповым чатом. Это включает в себя инициализацию объекта для взаимодействия с групповым вещанием, подписку на мультикаст группу, отправку сообщений участникам о готовности вступить в чат, обработку полученных сообщений, инициализацию потенциальных участников чата и имени пользователя. Это последовательный и логичный процесс, который позволяет пользователям присоединиться к групповому чату и взаимодействовать с другими участниками.

Рис.4 Диаграмма состояний метода создания сервера

**2.2 Руководство пользователя**

При первом запуске любой пользователь видит окно с пустым полем игры и двумя выпадающими меню – Game и Server (рис.2). Game позволяет начать новую игру (при запущенном на этой машине сервере), либо выйти из игры и закрыть приложение.

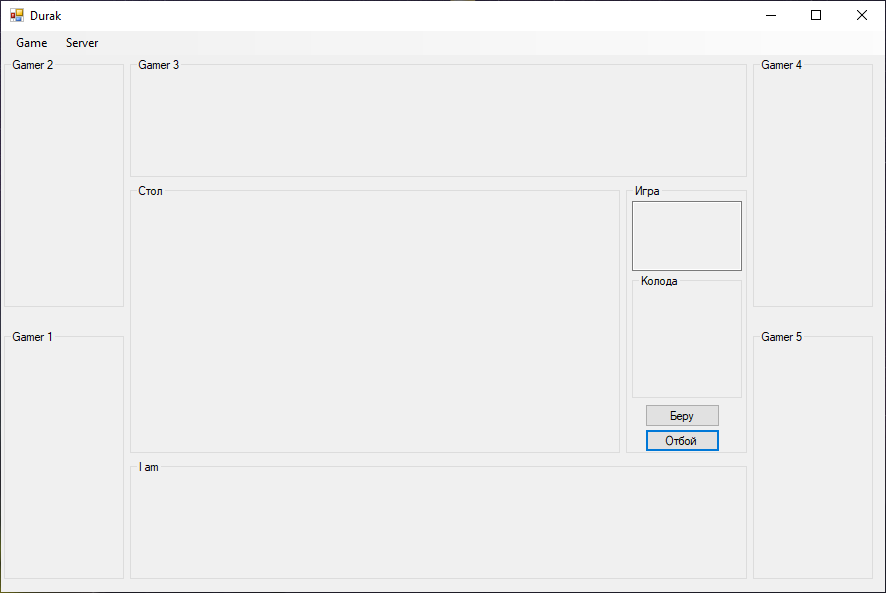


Рис.4 Основная форма игры

Для запуска сервера пользователю требуется нажать на вкладку Server и выбрать пункт Start server. При этом появится окно, которое предложит пользователю ввести свой игровой псевдоним, а также содержит строку с именем машины, на которой разворачивается сервер, в локальной сети.

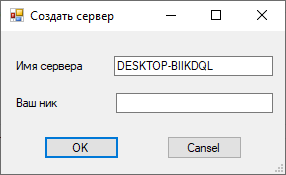


Рис.5 Окно создания сервера

После того, как сервер был запущен, владелец сервера может перейти к созданию игры во вкладке Game, для этого нужно выбрать пункт меню New Game.

Для того, чтобы игрок смог подключиться в созданную игру на запущенном сервере ему необходимо выбрать пункт меню Connect to Server во вкладке Server. После этого отобразится окно, схожее с таковым при запуске самого сервера, игроку нужно ввести псевдоним для игры, а также ввести в поле «Имя сервера» имя машины сервера в локальной сети.

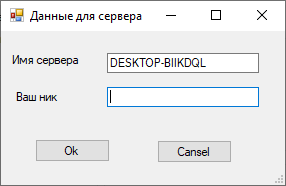


Рис.6 – Окно подключения к серверу

После подключения второго игрока, имена подключенных игроков отобразятся на соответствующих местах игрового поля, и владелец сервера сможет запустить партию игры.

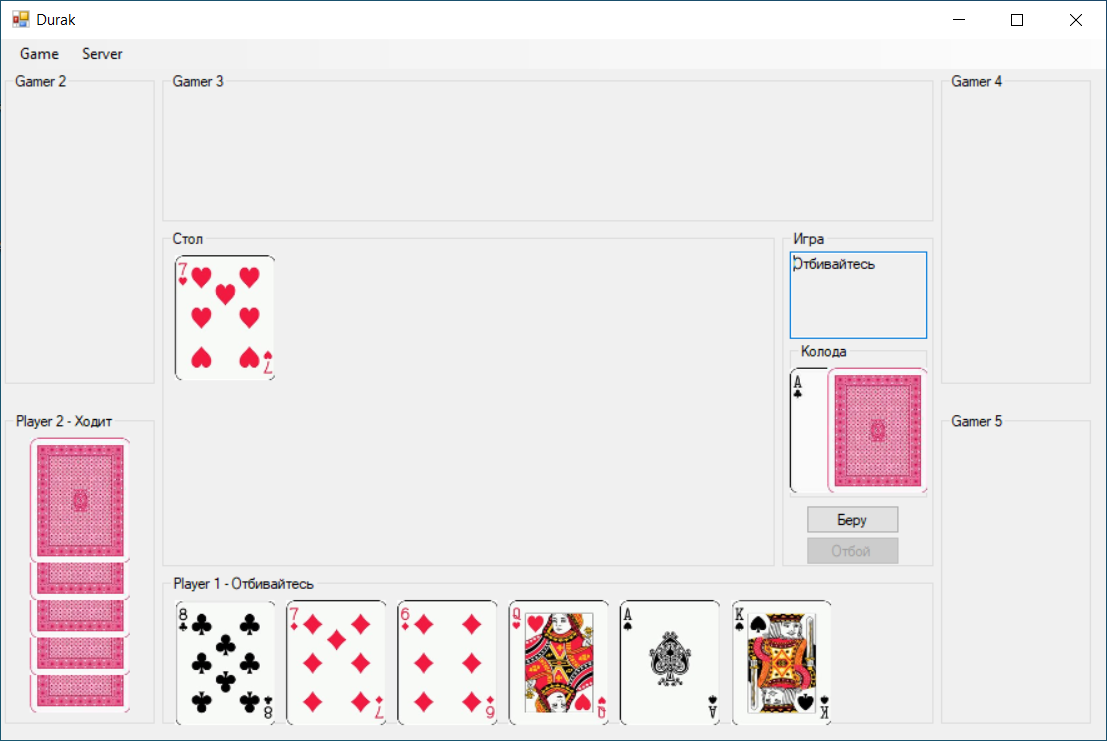


Рис. 7 Основной интерфейс запущенной партии

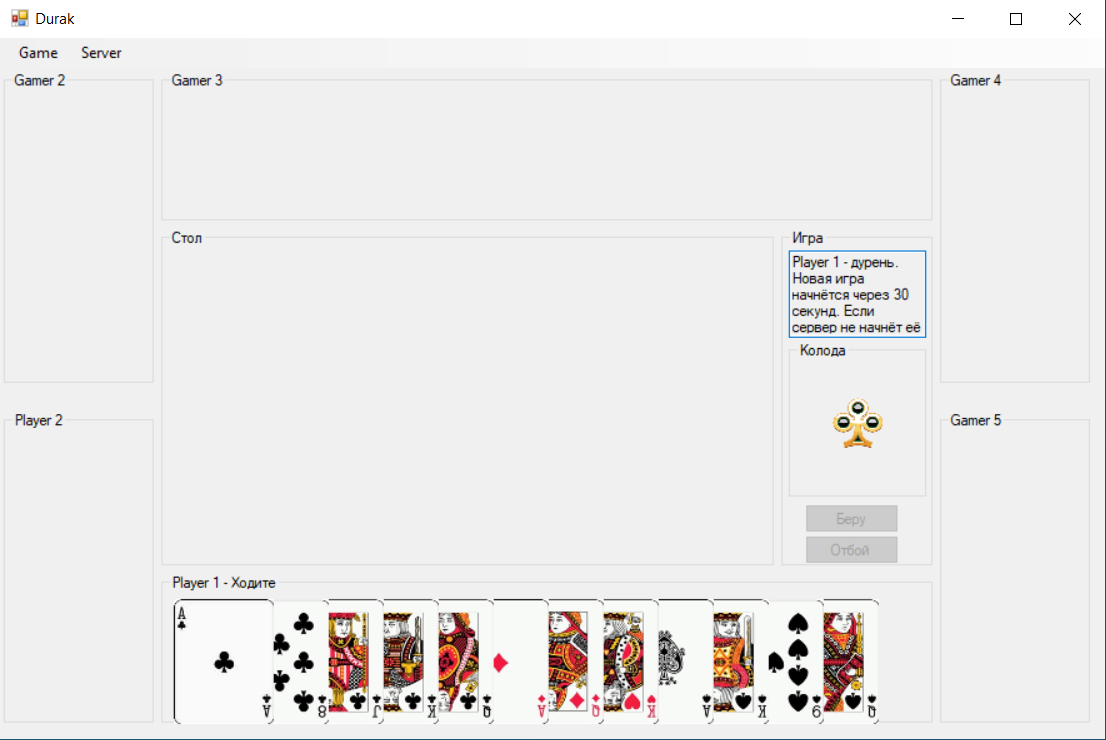
****

Рис.8 – Конец партии от лица проигравшего игрока

# ЗАКЛЮЧЕНИЕ

В результате проделанной работы была реализована карточная игра «Дурак» на архитектуре вида «клиент – сервер».

Были реализованы все части программного комплекса. Было проведено тестирование отдельных частей комплекса, а также взаимодействие между ними.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Википедия – Свободная энциклопедия [Электронный ресурс] –  
   URL: <https://ru.wikipedia.org/wiki/IGMP> (Дата обращения: 10.02.2024)
2. Википедия – Свободная энциклопедия [Электронный ресурс] –  
   URL: <https://ru.wikipedia.org/wiki/UDP> (Дата обращения: 25.02.2024)
3. 2.2.6. User Datagram Protocol – udp [Электронный ресурс] -  
   URL: <https://studfile.net/preview/7390796/page:6/> (Дата обращения: 25.02.2024)
4. Документация Android Studio [Электронный ресурс] -  
   URL: <https://developer.android.com/guide> (Дата обращения: 03.03.2024)
5. Документация Java [Электронный ресурс] -   
   URL: [docs.oracle.com/javase/8/docs/api/java/net/MulticastSocket.html](https://docs.oracle.com/javase/8/docs/api/java/net/MulticastSocket.html) (Дата обращения: 04.03.2024)

ПРИЛОЖЕНИЕ

Код класса MainActivity:  
public class MainActivity extends AppCompatActivity implements MessageObserver {

private EditText editMessage;

private MulticastManager multicastManager;

private MessageHandler messageHandler;

private Button myButtonRefresh;

private RecyclerView recyclerViewDeviceIPs;

private DeviceIPAdapter deviceIPAdapter;

private String currentIP;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

EdgeToEdge.enable(this);

setContentView(R.layout.activity\_main);

// Инициализация RecyclerView

recyclerViewDeviceIPs = findViewById(R.id.recyclerViewDeviceIPs);

recyclerViewDeviceIPs.setLayoutManager(new LinearLayoutManager(this));

// Кнопка обновления списка RecyclerView

myButtonRefresh = findViewById(R.id.buttonRefresh);

// Кнопка submit

Button myButton = findViewById(R.id.button);

// поле ввода никнейма

editMessage = findViewById(R.id.editMessage);

// свитч режима discover

Switch igmpHelloSwitch = findViewById(R.id.igmpHelloSwitch);

// Восстановление значения из SharedPreferences

SharedPreferences preferences = getSharedPreferences("AppPrefs", MODE\_PRIVATE);

String message = preferences.getString("editMessage", "User"); // Второй параметр - значение по умолчанию

editMessage.setText(message);

multicastManager = new MulticastManager("239.255.255.250", 1900);

try {

multicastManager.connect();

messageHandler = new MessageHandler(multicastManager.getSocket(),

multicastManager.getMulticastGroup(),

multicastManager.getMulticastPort());

messageHandler.startListening();

// Создание адаптера и установка его в RecyclerView

deviceIPAdapter = new DeviceIPAdapter(messageHandler.getIpNicknameMap());

recyclerViewDeviceIPs.setAdapter(deviceIPAdapter);

messageHandler.initUDPReceiver(12346); // Настройка приемника для прослушивания порта 12346

messageHandler.receiveUDPMessage();

messageHandler.addObserver(this); // Добавляем MainActivity в качестве наблюдателя

} catch (IOException e) {

e.printStackTrace();

}

// Обработчик события нажатия на элемент RecyclerView

deviceIPAdapter.setOnItemClickListener(position -> {

if (messageHandler.getNickName() == null) {

showAlert(MainActivity.this, "Не введено имя", "Пожалуйста, введите свое имя и нажмите Submit");

return;

}

// Обработка нажатия на элемент списка

currentIP = (new ArrayList<>(messageHandler.getIpNicknameMap().keySet())).get(position).split("\_")[0];

messageHandler.setCurrentIpUdp(currentIP);

String currentNick = (new ArrayList<>(messageHandler.getIpNicknameMap().keySet())).get(position)

.split("\_")[1];

// Отображаем диалоговое окно подтверждения

AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);

builder.setTitle("Подтверждение соединения");

builder.setMessage("Вы уверены, что хотите соединиться с пользователем " + currentNick + "?");

builder.setPositiveButton("Да", (dialog, which) -> new Thread(() -> {

messageHandler.sendMessageUDPStart();

Intent intent = new Intent(MainActivity.this, UDPChat.class);

intent.putExtra("currentIpaddress", currentIP);

intent.putExtra("nickName", messageHandler.getNickName());

startActivity(intent);

}).start());

builder.setNegativeButton("Нет", new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialog, int which) {

// Пользователь отказался от соединения, ничего не делаем

}

});

AlertDialog dialog = builder.create();

dialog.show();

});

myButton.setOnClickListener(view -> {

String nickname = String.valueOf(editMessage.getText());

new Thread(() -> messageHandler.setNickName(nickname)).start();

});

myButtonRefresh.setOnClickListener(view -> {

if (messageHandler == null)

return;

// Обновление списка устройств при нажатии на кнопку Refresh

Map<String, String> updatedIpNicknameMap = messageHandler.getIpNicknameMap(); // Получите новый словарь

// IP-адресов и никнеймов

deviceIPAdapter.updateDeviceIPs(updatedIpNicknameMap);

});

igmpHelloSwitch.setOnCheckedChangeListener((buttonView, isChecked) -> {

if (isChecked) {

messageHandler.setNickName(String.valueOf(editMessage.getText()));

// Запуск потока для отправки IGMP Hello

messageHandler.enableIGMPHello();

} else {

// Остановка отправки IGMP Hello

messageHandler.disableIGMPHello();

}

});

ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {

Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());

v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);

return insets;

});

}

@Override

protected void onDestroy() {

super.onDestroy();

messageHandler.removeObserver(this);

if (multicastManager != null) {

multicastManager.disconnect();

}

}

// Метод для отображения диалогового окна Alert

public void showAlert(Context context, String title, String message) {

AlertDialog.Builder builder = new AlertDialog.Builder(context);

builder.setTitle(title) // Установка заголовка диалогового окна

.setMessage(message) // Установка сообщения диалогового окна

.setPositiveButton("OK", null); // Добавление кнопки "OK", без действия

AlertDialog dialog = builder.create(); // Создание диалогового окна

dialog.show(); // Отображение диалогового окна

}

@Override

public void onReceiveStartChat(String ipAddress) {

// Реакция на событие "START\_CHAT"

runOnUiThread(() -> {

String nickname = String.valueOf(editMessage.getText());

// Ваш код для запуска новой активности или выполнения других действий

currentIP = ipAddress;

Intent intent = new Intent(MainActivity.this, UDPChat.class);

intent.putExtra("currentIpaddress", currentIP);

intent.putExtra("nickName", nickname);

startActivity(intent);

});

}

@Override

protected void onPause() {

super.onPause();

// Получаем значение из EditText

String message = editMessage.getText().toString();

// Сохраняем это значение в SharedPreferences

SharedPreferences preferences = getSharedPreferences("AppPrefs", MODE\_PRIVATE);

SharedPreferences.Editor editor = preferences.edit();

editor.putString("editMessage", message);

editor.apply();

}

}