**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ **«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ ИМ. В. Г. ШУХОВА»**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

# КУРСОВАЯ РАБОТА
## Дисциплина: «Сети ЭВМ и телекоммуникации»
## Тема: «Чат без центрального сервера»

Выполнил: ст. группы ВТ-31
Подкопаев Антон Валерьевич
Проверил: ст. пр. ПО и ВТАС
Федотов Евгений Александрович

**Белгород 2020**

# Содержание

# Введение

**Постановка задачи:** необходимо реализовать широковещательное приложение, позволяющее двум или более пользователям общаться, используя сеть интернет.

**UDP (User Datagram Protocol)** представляет собой сетевой протокол, который позволяет доставить данные на удаленный узел. Для этого передачи сообщений по протоколу UDP нет надобности использовать сервер, данные напрямую передаются от одного узла к другому. Снижаются накладные расходы при передаче, по сравнению с TCP, сами данные передаются быстрее. Все посылаемые сообщения по протоколу UDP называются дейтаграммами. Также через UDP можно передавать широковещательные сообщения для набора адресов в подсети. В то же время UDP имеет недостатки по сравнению с TCP. В частности, UDP не гарантирует доставку дейтаграммы конечному адресу. Поэтому данный протокол подходит больше для передачи изображений, мультимедийных файлов, потокового аудио, видео и т.п., когда недостаток небольшого количества потерянных при передаче дейтаграмм не сильно скажется на качестве переданных данных. [4]

Протокол UDP позволяет рассылать сообщения с помощью широковещательной групповой рассылки. При такой рассылке клиенту достаточно отправить одно сообщение, и его получат все остальные клиенты, которые подключены к группе. И нет надобности отправлять одно сообщение каждому отдельному клиенту. При использовании широковещательной передачи надо учитывать, что передача не идет дальше локальных сетей, так как маршрутизаторы подобные рассылки не пропускают.

# Теоретические сведения

## Протокол UDP

Протокол UDP является одним из основных протоколов, расположенных непосредственно над IP. Он предоставляет прикладным процессам транспортные услуги, немногим отличающиеся от услуг протокола IP. Протокол UDP обеспечивает доставку дейтограмм, но не требует подтверждения их получения. Протокол UDP не требует соединения с удаленным модулем UDP ("бессвязный" протокол). К заголовку IP-пакета UDP добавляет поля порт отправителя и порт получателя, которые обеспечивают мультиплексирование информации между различными прикладными процессами, а также поля длина UDP-дейтограммы и контрольная сумма, позволяющие поддерживать целостность данных. Протокол UDP ориентирован на транзакции, получение дейтаграмм и защита от дублирования не гарантированы. Приложения, требующие гарантированного получения потоков данных, должны использовать протокол управления пересылкой (Transmission Control Protocol - TCP). UDP - минимальный ориентированный на обработку сообщений протокол транспортного уровня. UDP не предоставляет никаких гарантий доставки сообщения для протокола верхнего уровня и не сохраняет состояния отправленных сообщений. UDP обеспечивает многоканальную передачу (с помощью номеров портов) и проверку целостности (с помощью контрольных сумм) заголовка и существенных данных.

Заголовок UDP состоит из четырех полей, каждое по 2 байта (16 бит). Два из них необязательны к использованию в IPv4, в то время как в IPv6 необязателен только порт отправителя.

В поле **Порт отправителя** указывается номер порта отправителя. Предполагается, что это значение задает порт, на который при необходимости будет посылаться ответ. В противном же случае, значение должно быть равным 0.

Поле **Порт получателя** обязательно и содержит порт получателя.

Поле **Длина датаграммы** задает длину всей датаграммы (заголовка и данных) в байтах. Минимальная длина равна длине заголовка – 8 байт. Теоретически, максимальный размер поля – 65535 байт для UDP-датаграммы (8 байт на заголовок и 65527 на данные). Фактический предел для длины данных при использовании IPv4 – 65507 (помимо 8 байт на UDP-заголовок требуется еще 20 на IPзаголовок).

Поле **Контрольная сумма** используется для проверки заголовка и данных на ошибки. Если сумма не сгенерирована передатчиком, то поле заполняется нулями. Поле является обязательным для IPv6.

| 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) | |

Если UDP работает над IPv4, контрольная сумма вычисляется при помощи псевдозаголовка, который содержит некоторую информацию из заголовка IPv4. Псевдозаголовок не является настоящим IPv4-заголовком, используемым для отправления IP-пакета. В таблице приведён псевдозаголовок, используемый только для вычисления контрольной суммы.

| Биты | 0 — 7 | 8 — 15 | 16 — 23 | 24 — 31 |
|---|---|---|---|---|
| 0 | Адрес источника | | | |
| 32 | Адрес получателя | | | |
| 64 | Нули | Протокол | Длина UDP | |
| 96 | Порт источника | | Порт получателя | |
| 128 | Длина | | Контрольная сумма | |
| 160+ | Данные | | | |

Из-за недостатка надежности, приложения UDP должны быть готовыми к некоторым потерям, ошибкам и дублированиям. Некоторые из них (например, TFTP) могут при необходимости добавить элементарные механизмы обеспечения надежности на прикладном уровне. Но чаще такие механизмы не используются UDP-приложениями и даже мешают им. Потоковые медиа, многопользовательские игры в реальном времени и VoIP - примеры приложений, часто использующих протокол UDP. В этих конкретных приложениях потеря пакетов обычно не является большой проблемой. Если приложению необходим высокий уровень надежности, то можно использовать другой протокол. Более серьезной потенциальной проблемой является то, что в отличие от TCP, основанные на UDP приложения не обязательно имеют хорошие механизмы контроля и избежания перегрузок. Чувствительные к перегрузкам UDP-приложения, которые потребляют значительную часть доступной пропускной способности, могут поставить под угрозу стабильность в Интернете. [1]

## Широковещательные сети

Широковещательные сети обладают единым каналом связи, совместно используемым всеми машинами сети. Короткие сообщения, называемые в некоторых случаях пакетами, которые посылаются одной машиной, получают все машины. Поле адреса в пакете указывает, кому направляется сообщение. При получении пакета машина проверяет его адресное поле. Если пакет адресован этой машине, она его обрабатывает. Пакеты, адресованные другим машинам, игнорируются.

В качестве иллюстрации представьте себе человека, стоящего в конце коридора с большим количеством комнат и кричащего: «Ватсон, идите сюда. Вы мне нужны». И хотя это сообщение может быть получено (услышано) многими людьми, ответит только Ватсон. Остальные просто не обратят на него внимания. Другим примером может быть объявление в аэропорту, предлагающее всем пассажирам рейса 644 подойти к выходу номер 12.

Широковещательные сети также позволяют адресовать пакет одновременно всем машинам с помощью специального кода в поле адреса. Когда передается пакет с таким кодом, его получают и обрабатывают все машины сети. Такая операция называется широковещательной передачей. Некоторые широковещательные системы также предоставляют возможность посылать сообщения подмножеству машин, и это называется многоадресной передачей. Одной из возможных схем реализации этого может быть резервирование одного бита для признака многоадресной передачи. Оставшиеся n-1 разрядов адреса могут содержать номер группы. Каждая машина может «подписаться» на одну, несколько или все группы. Когда пакет посылается определенной группе, он доставляется всем машинам, являющимся членами этой группы.

## Чат без центрального сервера

Чат, чаттер (англ. chatter «болтать») — средство обмена сообщениями по компьютерной сети в режиме реального времени, а также программное обеспечение, позволяющее организовывать такое общение. Характерной особенностью является коммуникация именно в реальном времени или близкая к этому, что отличает чат от форумов и других «медленных» средств. То есть, если на форуме можно написать вопрос и ждать, пока кто-нибудь посчитает нужным на него ответить (в то же время можно получить и несколько супов от разных пользователей), то в чате общение происходит только с теми, кто присутствует в нём в настоящий момент, а результаты обмена сообщениями могут и не сохраняться. В последнее время чаты значительно расширили свою функциональность за счёт красивого радующего глаз дизайна, а также большого числа улучшений. Например, появились возможности помещать одного или нескольких пользователей в игнор, сообщения которых после этого перестают быть видимыми тому, кто поместил их в игнор, при том для данной операции необязательно быть модератором или администратором чата. Такое бывает необходимо, когда пользователь не нарушает Правила чата, но в то же время неприятен, либо пользователь по какой-то причине не банится, то есть не наказывается Администрацией чата за своё поведение.

Под словом чат обычно понимается групповое общение, хотя к ним можно отнести и обмен текстом «один на один» посредством программ мгновенного обмена сообщениями, например, XMPP, ICQ или даже SMS.

В настоящее время интерес к чатам падает. Их место заняли социальные сети, с их намного большим функционалом. Популярность набирают только видеочаты.

Но и веб-чаты не хотят сдавать своих позиций. Их возможности постоянно растут. Чаты, из мест для просто общения, трансформируются в подобие порталов с разнообразными развлечениями для посетителей. В своём большинстве, современные веб-чаты, кроме простого общения, предлагают посетителям приватное общение с помощью веб-камеры, прослушивание собственных радиостанций с диджеями, принимающими заказы прямо в чате, круглосуточные викторины с призами, «магазины» с подарками, приобретаемыми за чатовскую «валюту», различные игры, казино, предсказания… Активные посетители поощряются повышением рангов и другими наградами. Для украшения сообщений используется возможность изменения стилей и цвета шрифтов, разнообразие смайлов. Также пользователи имеют возможность раскрасить свой никнейм, или установить графический ник-картинку. В функции чатов встраиваются чат-боты, рассказывающие анекдоты и поддерживающие псевдоразумное общение с посетителями. Кроме этого, в чатах могут быть форумы, организовываться службы знакомств и заключаться виртуальные браки между посетителями. [3]

## .NET Framework

.NET Framework - программная платформа, выпущенная компанией Microsoft в 2002 году. Основой платформы является общеязыковая среда исполнения Common Language Runtime (CLR), которая подходит для разных языков программирования. Функциональные возможности CLR доступны в любых языках программирования, использующих эту среду. Считается, что платформа .NET Framework явилась ответом компании Microsoft на набравшую к тому времени большую популярность платформу Java компании Sun Microsystems (ныне принадлежит Oracle).

Хотя .NET является патентованной технологией корпорации Microsoft и официально рассчитана на работу под операционными системами семейства Microsoft Windows, существуют независимые проекты (прежде всего это Mono и Portable.NET), позволяющие запускать программы .NET на некоторых других операционных системах. В настоящее время .NET Framework получает развитие в виде .NET Core, изначально предполагающей кроссплатформенную разработку и эксплуатацию.

Программа для .NET Framework, написанная на любом поддерживаемом языке программирования, сначала переводится компилятором в единый для .NET промежуточный байт-код Common Intermediate Language (CIL) (ранее назывался Microsoft Intermediate Language, MSIL). В терминах .NET получается сборка, англ. assembly. Затем код либо исполняется виртуальной машиной Common Language Runtime (CLR), либо транслируется утилитой NGen.exe в исполняемый код для конкретного целевого процессора. Использование виртуальной машины предпочтительно, так как избавляет разработчиков от необходимости заботиться об особенностях аппаратной части. В случае использования виртуальной машины CLR встроенный в неё JIT-компилятор «на лету» (just in time) преобразует промежуточный байт-код в машинные коды нужного процессора. Современная технология динамической компиляции позволяет достигнуть высокого уровня быстродействия. Виртуальная машина CLR также сама заботится о базовой безопасности, управлении памятью и системе исключений, избавляя разработчика от части работы.

Архитектура .NET Framework описана и опубликована в спецификации Common Language Infrastructure (CLI), разработанной Microsoft и утверждённой ISO и ECMA. В CLI описаны типы данных .NET, формат метаданных о структуре программы, система исполнения байт-кода и многое другое.

Объектные классы .NET, доступные для всех поддерживаемых языков программирования, содержатся в библиотеке Framework Class Library (FCL). В FCL входят классы Windows Forms, ADO.NET, ASP.NET, Language Integrated Query, Windows Presentation Foundation, Windows Communication Foundation и другие. Ядро FCL называется Base Class Library (BCL). [2]

# Разработка программы

## Выбор инструментов разработки

Для проектирования данной курсовой работы необходимо было реализовать чат без центрального сервера на основе широковещательных UDP-протоколов. Для реализации широковещательной передачи в рамках локальной сети OS Windows была использована программная платформа .NET, так как она обладает обширным набором инструментов для работы с UDP. Сама же реализация была исполнена на объектно-ориентированном языке программирования C#, в котором и был представлен необходимый нам фреймворк. Графическое отображение было разработано с использованием интерфейса Windows Forms API.

## UDP клиент

В .NET за работу с протоколом UDP отвечает класс UdpClient, который построен на основе класса Socket. Для создания подключения достаточно создать объект UdpClient и вызывать у него метод Connect(). В метод Connect передается адрес и порт для подключения, то есть настройки внешнего адреса, к которому мы хотим подключиться. В случае с локальным адресом можно указывать один порт. При этом Connect не устанавливает постоянного соединения с удаленным хостом. Он только устанавливает параметры подключения, которые также можно задать с помощью конструктора.

```
UdpClient client = new UdpClient();
client.Connect("www.microsoft.com", 8888);
```

Фактически образование соединения происходит при передаче данных на удаленный хост или, наоборот, при получении с него данных. Для передачи данных применяется метод Send(), который в качестве параметра принимает массив отправляемых байтов и количество байтов, которые надо отправить. Метод Send() возвращает количество реально отправленных байтов, благодаря чему мы можем сравнить, сколько из отправляемых байтов в реальности было отправлено.

```
UdpClient client = new UdpClient(8001);
string message = "Hello world!";
byte[] data = Encoding.UTF8.GetBytes(message);
int numberOfSentBytes = client.Send(data, data.Length);
client.Close();
```

После завершения работы объекта UdpClient его надо закрыть с помощью метода Close().

Для получения данных применяется метод Receive(). Этот метод принимает один параметр типа System.Net.IPEndPoint с модификатором ref. Объект IPEndPoint представляет удаленную точку, с которой поступили данные. Полученные данные возвращает метод Receive в виде массива байтов.

Как правило, рекомендуется выносить вызов метода Receive в отдельный поток, поскольку данный метод блокирует вызывающий поток пока данные не будут получены. При этом если мы указали информацию о подключении (название хоста, номер порта) в конструкторе или в методе Connect, то метод Receive будет получать данные только с указанной удаленной точки, остальные подключения будут игнорироваться. Если же мы использовали пустой конструктор и не вызывали метод Connect, то UdpClient будет принимать данные от всех подключений.

## Широковещательная рассылка

Ключевым в данном случае является метод JoinMulticastGroup(), который позволяет присоединиться клиенту к группе в локальной сети. Благодаря этому объект UdpClient сможет получать дейтаграммы, которые предназначаются всей группы объектов UdpClient. Для присоединения к группе в метод JoinMulticastGroup передается адрес группы. В качестве адреса может использоваться один из адресов в диапазоне от 224.0.0.0 до 239.255.255.255.

```
private static void ReceiveMessage()
{
      UdpClient receiver = new UdpClient(localPort);
      receiver.JoinMulticastGroup(remoteAddress, 50);
      IPEndPoint remoteIp = null;
      //  остальное содержимое метода
}
```

Метод JoinMulticastGroup() принимает два параметра: адрес групповой рассылки (должен быть одним и тем же для всей группы клиентов) и число проходов через маршрутизаторы. То есть пока сообщение достигнет получателя, оно может пройти через некоторое число маршрутизаторов, которые направят его по нужному пути. В данном случае число 50 означает, что чтобы пройти от компьютера-отправителя к компьютеру-получателю, дейтаграмма может миновать 50 маршрутизаторов.

## Windows Forms

На форме есть две кнопки loginButton и logoutButton соответственно для входа и выхода из чата, а также кнопка sendButton для отправки сообщений. Для ввода имени, которое будет использоваться в чате, есть текстовое поле userNameTextBox. Для ввода сообщения внизу имеется текстовое поле messageTextBox, которое является многострочным, то есть у которого свойство MultiLine имеет значение true. И в самом центре размещено многострочное текстовое поле chatTextBox, в которое будут выводиться полученные сообщения.

Все клиенты будут подключаться к общему адресу "235.5.5.1". По нажатию на кнопку входа будет происходить отправка сообщения о входе нового пользователя и запускаться новая задача на прием входящих сообщений. Для приема сообщений будет использоваться метод ReceiveMessages(). Так как в этой задаче мы обращаемся к элементам, созданным в главном потоке, то для добавления сообщения надо использовать делегат MethodInvoker.

```
this.Invoke(new MethodInvoker(() =>
{
      string time = DateTime.Now.ToShortTimeString();
      chatTextBox.Text = time + " " + message + "\r\n" + chatTextBox.Text;
}));
```

В качестве параметра он принимает либо ссылку на метод, либо лямбда-выражение.

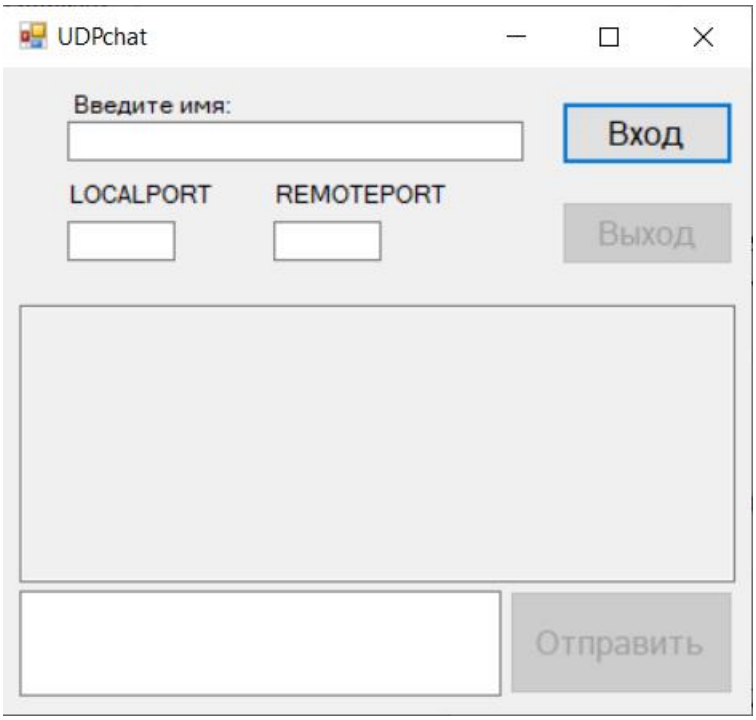
По нажатию на кнопку sendButton происходит отправка сообщения. И по нажатию на кнопку выхода или по закрытию формы срабатывает метод ExitChat(), которые прекращает поток и осуществляет выход из чата.

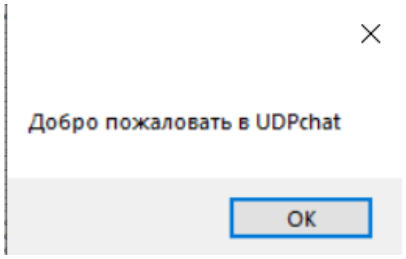
В ожидании приема сообщений. Когда происходит выход из чата по закрытию формы, то всем, в том числе и этому ожидающему клиенту посылается сообщение о выходе. Однако при этом идет закрытие и уничтожение формы, в связи с чем мы можем столкнуться с исключением ObjectDisposedException. Поэтому при приеме сообщений включается дополнительный блок обработки исключений.

```
catch (ObjectDisposedException)
{
      if (!alive)
            return;
      throw;
}
```

# Руководство пользователя

В момент запуска приложения пользователь видит приветственное сообщение, после закрытия которого появляется главная панель управления и сам чат.



Для подключения в первую очередь необходимо ввести свое имя, а так же обозначить порты локальной сети, по которым будет происходить передача сообщений. В поле «LOCALPORT» указывается порт отправителя, а в поле «REMOTEPORT» порт получателя. Важно, чтобы у пользователей били правильно выбраны порты, иначе сообщения не будут доставлены.

По нажатии кнопки «Вход» пользователь будет подключен к беседе, о чем будут оповещены остальные пользователи соответствующим сообщением.

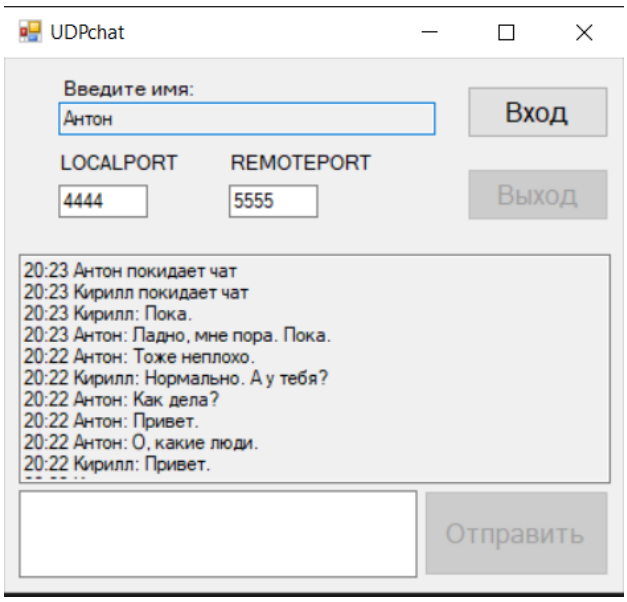
Сообщение набирается в нижнем текстовом окне, после чего оно может быть отослано нажатием кнопки «Отправить». Оно появится в основном окне чата с припиской имени пользователя, который его отправил.

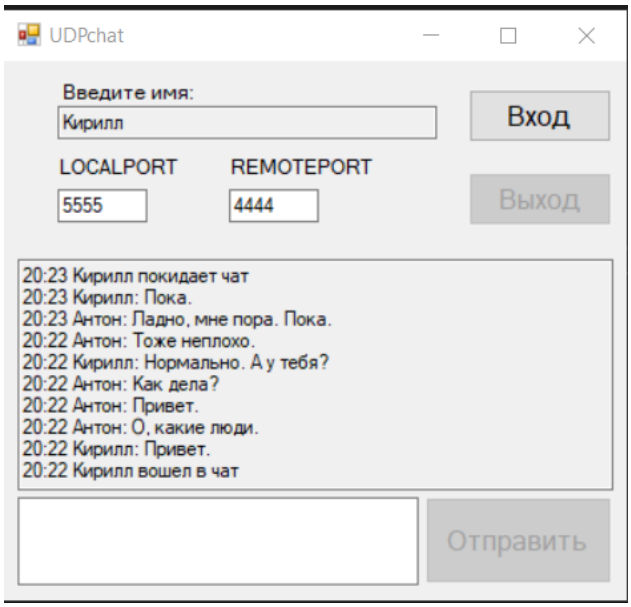
По завершению общения пользователь может нажать на кнопку «Выход», после чего он покинет чат, о чем будут оповещены остальные пользователи. Если же пользователь закроет программу, не выходя из чата, он все равно его покинет, и соответствующее сообщение будет доставлено.

# Тестирование

Тестирование работоспособности приложения происходило в рамках локальной сети, хостом групповой рассылки был пользователь с адресом 235.5.5.1 и временем жизни пакета данных TTL - 40.

В ходе сессии работы с программой на одной машине было запущено две копии разработанного проекта. Первый пользователь с локальным портом 4444 подключился первым, после чего в чат зашел еще один человек с локальным портом 5555.



# Заключение

В ходе выполнения данной курсовой работы была реализована программа пользовательского чата без центрального сервера на основе широковещательных протоколов. В ходе работы были изучены основополагающие принципы работы UDP-протоколов, широковещательной рассылки, а также получены базовые знания в области создания пользовательских чатов. Были улучшены навыки создания интерфейса путем использования Windows Form, а также была изучена программная платформа .NET для C#, используемая для гарантированной работы сетевых служб. Были рассмотрены и использованы соответствующие библиотеки для создания широковещательных каналов в локальной сети, вместе с тем и основные компоненты API для создания графического исполнения программ в OS Windows.

По ходу работы дополнительно была изучена история появления онлайн чатов, их значимое положение в пути развития дистанционной коммуникации посредством общения в интернете.

В заключении хотелось бы отметить, что несмотря на безостановочно растущий уровень технологий, чаты продолжают занимать значимую позицию во всех ведущих коммуникационных приложениях и соц. сетях. Не говоря даже о мессенджерах, которые являются прямым развитием идеи чатов, которые до сих пор нужны, чтобы консультировать клиентов: помогать выбрать между несколькими товарами, рассказывать об особенностях продукта, отвечать на вопросы об оплате и доставке. Посетителю сайта может быть проще написать пару предложений, чем звонить или переходить в почту для отправки письма. Таким образом значимых применений чатов со временем становиться только больше, ведь они позволяют пользователям общаться, оставаясь анонимным.

# Список литературы

[1] Интернет-ресурс *Википедия. Протокол UDP.*
URL:https://ru.wikipedia.org/wiki/UDP

[2] Интернет-ресурс *Википедия. .NET framework.*
URL: https://ru.wikipedia.org/wiki/.NET_Framework

[3] Интернет-ресурс *Википедия. Онлайн чат.*
URL: https://en.wikipedia.org/wiki/Online_chat

[4] Интернет-ресурс *Metanit.com. Сайт о программировании.*
URL: https://metanit.com/

[5] Интернет-ресурс *Семенов Ю.А. Протокол UDP.*
URL: http://book.itep.ru/4/44/udp_442.htm

[6] Интернет-ресурс *CyberForum.ru. Форум программистов.*
URL: https://www.cyberforum.ru/

Form1.cs

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Windows.Forms;
using System.Threading.Tasks;

namespace UdpChat
{
    public partial class Form1 : Form
    {
        bool alive = false; // будет ли работать поток для приема
        UdpClient client;
        int LOCALPORT; // порт для приема сообщений
        int REMOTEPORT; // порт для отправки сообщений
        const int TTL = 40;
        const string HOST = "235.5.5.1"; // хост для групповой рассылки
        IPAddress groupAddress; // адрес для групповой рассылки

        string userName; // имя пользователя в чате
        public Form1()
        {
            InitializeComponent();

            loginButton.Enabled = true; // кнопка входа
            logoutButton.Enabled = false; // кнопка выхода
            sendButton.Enabled = false; // кнопка отправки
            chatTextBox.ReadOnly = true; // поле для сообщений

            groupAddress = IPAddress.Parse(HOST);
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            MessageBox.Show("Добро пожаловать в UDPchat");
        }

        // обработчик нажатия кнопки loginButton
        private void loginButton_Click(object sender, EventArgs e)
        {
            userName = userNameTextBox.Text;
            userNameTextBox.ReadOnly = true;

            try
            {
                client = new UdpClient(LOCALPORT);
                // присоединяемся к групповой рассылке
                client.JoinMulticastGroup(groupAddress, TTL);

                // запускаем задачу на прием сообщений
                Task receiveTask = new Task(ReceiveMessages);
                receiveTask.Start();

                // отправляем первое сообщение о входе нового пользователя
                string message = userName + " вошел в чат";
                byte[] data = Encoding.Unicode.GetBytes(message);
                client.Send(data, data.Length, HOST, REMOTEPORT);
                client.Send(data, data.Length, HOST, LOCALPORT);

                loginButton.Enabled = false;
                logoutButton.Enabled = true;
                sendButton.Enabled = true;
            }
```

```
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}

// метод приема сообщений
private void ReceiveMessages()
{
    alive = true;
    try
    {
        while (alive)
        {
            IPEndPoint remoteIp = null;
            byte[] data = client.Receive(ref remoteIp);
            string message = Encoding.Unicode.GetString(data);

            // добавляем полученное сообщение в текстовое поле
            this.Invoke(new MethodInvoker(() =>
            {
                string time = DateTime.Now.ToShortTimeString();
                chatTextBox.Text = time + " " + message + "\r\n" + chatTextBox.Text;
            }));
        }
    }
    catch (ObjectDisposedException)
    {
        if (!alive)
            return;
        throw;
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}

// обработчик нажатия кнопки sendButton
private void sendButton_Click(object sender, EventArgs e)
{
    try
    {
        string message = String.Format("{0}: {1}", userName, messageTextBox.Text);
        byte[] data = Encoding.Unicode.GetBytes(message);
        client.Send(data, data.Length, HOST, REMOTEPORT);
        client.Send(data, data.Length, HOST, LOCALPORT);
        messageTextBox.Clear();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}

// обработчик нажатия кнопки logoutButton
private void logoutButton_Click(object sender, EventArgs e)
{
    ExitChat();
}

// выход из чата
private void ExitChat()
{
    string message = userName + " покидает чат";
    byte[] data = Encoding.Unicode.GetBytes(message);
```

```
            client.Send(data, data.Length, HOST, REMOTEPORT);
            client.Send(data, data.Length, HOST, LOCALPORT);
            client.DropMulticastGroup(groupAddress);

            alive = false;
            client.Close();

            loginButton.Enabled = true;
            logoutButton.Enabled = false;
            sendButton.Enabled = false;
        }

        // обработчик события закрытия формы
        private void Form1_FormClosing(object sender, FormClosingEventArgs e)
        {
            if (alive)
                ExitChat();
        }

        private void label1_Click(object sender, EventArgs e)
        {
            MessageBox.Show("Подсказка: рекомендуется использовать имя Антон");
        }

        private void textBox1_TextChanged(object sender, EventArgs e)
        {
            LOCALPORT = int.Parse(textBox1.Text);
        }

        private void textBox2_TextChanged(object sender, EventArgs e)
        {
            REMOTEPORT = int.Parse(textBox2.Text);
        }
    }
      }
```

Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace UdpChat
{
    static class Program
    {
        /// <summary>
        /// Главная точка входа для приложения.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}
```

Form1.Designer.cs

```
namespace UdpChat
{
    partial class Form1
    {
        /// <summary>
        /// Обязательная переменная конструктора.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Освободить все используемые ресурсы.
        /// </summary>
        /// <param name="disposing">истинно, если управляемый ресурс должен быть удален; иначе
ложно.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Код, автоматически созданный конструктором форм Windows

        /// <summary>
        /// Требуемый метод для поддержки конструктора — не изменяйте
        /// содержимое этого метода с помощью редактора кода.
        /// </summary>
        private void InitializeComponent()
        {
            this.loginButton = new System.Windows.Forms.Button();
            this.sendButton = new System.Windows.Forms.Button();
            this.logoutButton = new System.Windows.Forms.Button();
            this.userNameTextBox = new System.Windows.Forms.TextBox();
            this.chatTextBox = new System.Windows.Forms.TextBox();
            this.messageTextBox = new System.Windows.Forms.TextBox();
            this.label1 = new System.Windows.Forms.Label();
            this.label2 = new System.Windows.Forms.Label();
            this.textBox1 = new System.Windows.Forms.TextBox();
            this.label3 = new System.Windows.Forms.Label();
            this.textBox2 = new System.Windows.Forms.TextBox();
            this.SuspendLayout();
            //
            // loginButton
            //
            this.loginButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F);
            this.loginButton.Location = new System.Drawing.Point(374, 21);
            this.loginButton.Name = "loginButton";
            this.loginButton.Size = new System.Drawing.Size(114, 40);
            this.loginButton.TabIndex = 0;
            this.loginButton.Text = "Вход";
            this.loginButton.UseVisualStyleBackColor = true;
            this.loginButton.Click += new System.EventHandler(this.loginButton_Click);
            //
            // sendButton
            //
            this.sendButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F);
            this.sendButton.Location = new System.Drawing.Point(338, 323);
            this.sendButton.Name = "sendButton";
            this.sendButton.Size = new System.Drawing.Size(150, 64);
            this.sendButton.TabIndex = 1;
            this.sendButton.Text = "Отправить";
            this.sendButton.UseVisualStyleBackColor = true;
            this.sendButton.Click += new System.EventHandler(this.sendButton_Click);
```

```
//
// logoutButton
//
this.logoutButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F);
this.logoutButton.Location = new System.Drawing.Point(374, 82);
this.logoutButton.Name = "logoutButton";
this.logoutButton.Size = new System.Drawing.Size(114, 40);
this.logoutButton.TabIndex = 2;
this.logoutButton.Text = "Выход";
this.logoutButton.UseVisualStyleBackColor = true;
this.logoutButton.Click += new System.EventHandler(this.logoutButton_Click);
//
// userNameTextBox
//
this.userNameTextBox.Location = new System.Drawing.Point(44, 33);
this.userNameTextBox.Name = "userNameTextBox";
this.userNameTextBox.Size = new System.Drawing.Size(303, 22);
this.userNameTextBox.TabIndex = 3;
//
// chatTextBox
//
this.chatTextBox.Location = new System.Drawing.Point(12, 147);
this.chatTextBox.Multiline = true;
this.chatTextBox.Name = "chatTextBox";
this.chatTextBox.Size = new System.Drawing.Size(476, 170);
this.chatTextBox.TabIndex = 4;
//
// messageTextBox
//
this.messageTextBox.Location = new System.Drawing.Point(12, 323);
this.messageTextBox.Multiline = true;
this.messageTextBox.Name = "messageTextBox";
this.messageTextBox.Size = new System.Drawing.Size(320, 64);
this.messageTextBox.TabIndex = 5;
//
// label1
//
this.label1.AutoSize = true;
this.label1.Font = new System.Drawing.Font("Microsoft Sans Serif", 9F);
this.label1.Location = new System.Drawing.Point(44, 12);
this.label1.Name = "label1";
this.label1.Size = new System.Drawing.Size(101, 18);
this.label1.TabIndex = 6;
this.label1.Text = "Введите имя:";
this.label1.Click += new System.EventHandler(this.label1_Click);
//
// label2
//
this.label2.AutoSize = true;
this.label2.Font = new System.Drawing.Font("Microsoft Sans Serif", 9F);
this.label2.Location = new System.Drawing.Point(41, 68);
this.label2.Name = "label2";
this.label2.Size = new System.Drawing.Size(98, 18);
this.label2.TabIndex = 7;
this.label2.Text = "LOCALPORT";
//
// textBox1
//
this.textBox1.Location = new System.Drawing.Point(44, 95);
this.textBox1.Name = "textBox1";
this.textBox1.Size = new System.Drawing.Size(70, 22);
this.textBox1.TabIndex = 8;
this.textBox1.TextChanged += new System.EventHandler(this.textBox1_TextChanged);
//
// label3
//
```

```
            this.label3.AutoSize = true;
            this.label3.Font = new System.Drawing.Font("Microsoft Sans Serif", 9F);
            this.label3.Location = new System.Drawing.Point(178, 68);
            this.label3.Name = "label3";
            this.label3.Size = new System.Drawing.Size(115, 18);
            this.label3.TabIndex = 9;
            this.label3.Text = "REMOTEPORT";
            //
            // textBox2
            //
            this.textBox2.Location = new System.Drawing.Point(181, 95);
            this.textBox2.Name = "textBox2";
            this.textBox2.Size = new System.Drawing.Size(70, 22);
            this.textBox2.TabIndex = 10;
            this.textBox2.TextChanged += new System.EventHandler(this.textBox2_TextChanged);
            //
            // Form1
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(8F, 16F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(500, 399);
            this.Controls.Add(this.textBox2);
            this.Controls.Add(this.label3);
            this.Controls.Add(this.textBox1);
            this.Controls.Add(this.label2);
            this.Controls.Add(this.label1);
            this.Controls.Add(this.messageTextBox);
            this.Controls.Add(this.chatTextBox);
            this.Controls.Add(this.userNameTextBox);
            this.Controls.Add(this.logoutButton);
            this.Controls.Add(this.sendButton);
            this.Controls.Add(this.loginButton);
            this.Name = "Form1";
            this.Text = "UDPchat";
            this.Load += new System.EventHandler(this.Form1_Load);
            this.ResumeLayout(false);
            this.PerformLayout();

        }

        #endregion

        private System.Windows.Forms.Button loginButton;
        private System.Windows.Forms.Button sendButton;
        private System.Windows.Forms.Button logoutButton;
        private System.Windows.Forms.TextBox userNameTextBox;
        private System.Windows.Forms.TextBox chatTextBox;
        private System.Windows.Forms.TextBox messageTextBox;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.TextBox textBox1;
        private System.Windows.Forms.Label label3;
        private System.Windows.Forms.TextBox textBox2;
    }
}
```