

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА ИНФОРМАЦИОННО-СЕТЕВЫХ ТЕХНОЛОГИЙ

КУРСОВАЯ РАБОТА (ПРОЕКТ)
ЗАЩИЩЕНА С ОЦЕНКОЙ
РУКОВОДИТЕЛЬ

доцент, к.т.н.		И. А. Пастушок
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О КУРСОВОЙ РАБОТЕ

по дисциплине: Компьютерное проектирование информационных систем

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. №		А. Е. Ковалева
	подпись, дата	инициалы, фамилия

Санкт-Петербург, 2021

Содержание

Постановка задачи	3
Описание проекта	3
API.....	4
Сайт пользователя	4
Сайт администратора	4
Таблицы базы данных	5
LTE.....	6
Работа с маскараром и Ansible Playbook	6
Оконечное устройство	6
Контейнеризация	7
Вклад в проект	9
Задача 1: API. AC: Работа с сетью	9
TCP	9
Теоретическая часть	9
Реализация поставленной задачи	10
Инструкция.....	10
Тестирование кода.....	11
Ссылка на Git (код + тесты).....	11
Особенности сокетов TCP	12
UDP	12
Теоретическая часть	12
Основное задание	13
Инструкция.....	13
Тестирование.....	14
Задача 2: внутренне тестирование найти баги в текущих приложениях	15
как пользователь системы.....	15
Реализация.....	15
Тесты как пользователя.....	15
Тесты JUnit	15
Тестирование. JUnit.....	16
Выводы	16

Постановка задачи

Исследование работы с LTE и masquerade.

Написание API с базой данных устройств.

Реализация способа передачи данных через интерфейс.

Написание программы на пользовательском устройстве.

Описание проекта

Данный проект предназначен для коммуникации пользователей с помощью различных интерфейсов: LTE, Wi-Fi, LoRa.

На данной стадии проекта реализовано только передача данных через Wi-Fi и взаимодействие с masquerade.

Схема проекта представлена на рисунке 1:

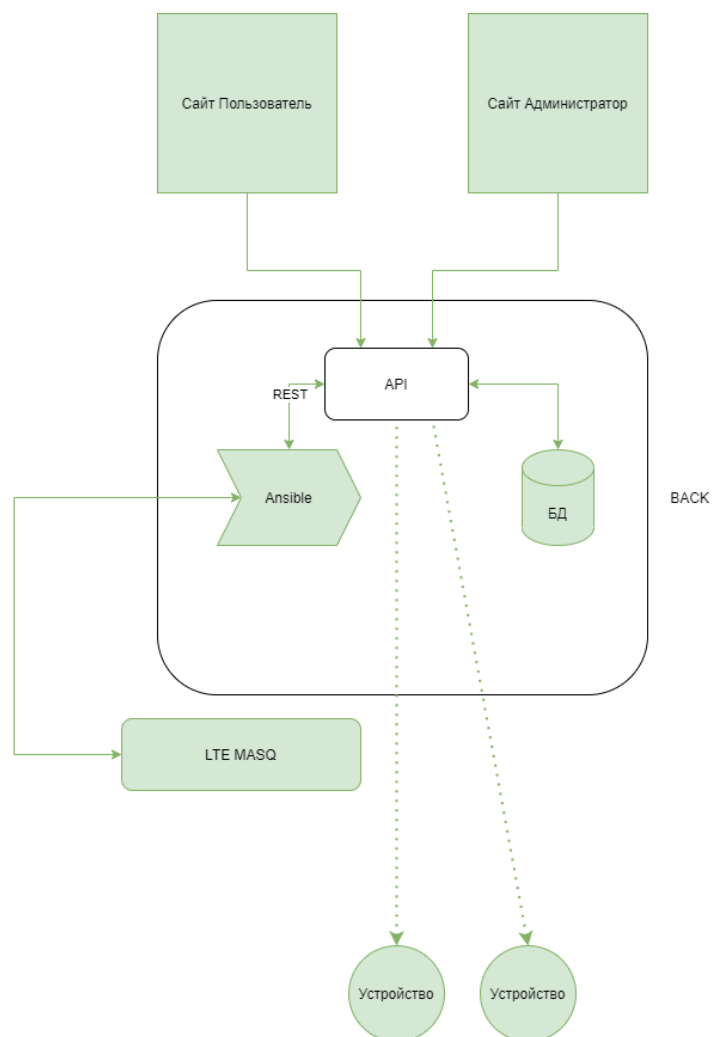


Рисунок 1 – Схема проекта

API

Созданное API имеет функции, такие как:

- Обращение с интерфейсами
(может через интерфейсы передавать данные на конкретные устройства)
- Сопоставление внутреннего и внешнего IP
- Добавление/удаление устройств
- Получение управляющих команд

Данные функции реализованы через сайты пользователя и администратора.

Сайт пользователя

1. Пользователь выбирает ID абонента, которому хочет отправить сообщение.
2. Для него подгружаются доступные данному абоненту интерфейсы передачи.
3. Пользователь выбирает желаемый интерфейс и пишет сообщение, которое хочет передать.

Сайт администратора

Администратор может добавлять новых пользователей или удалять пользователей.

1. Для добавления нового пользователя администратору необходимо будет ввести внешний и внутренний IP адреса и перечислить доступные для данного абонента интерфейсы.

После этого в базу данных добавляется новый пользователь.

2. Для удаления абонента нужно выбрать только ID удаляемого пользователя и после этого выбранный пользователь удаляется из базы данных.

База данных имеет описание всех устройств, то есть ID-устройства и интерфейсы, доступные данному абоненту.

База данных имеет следующую структуру:

Таблицы базы данных

1. **Users** – таблица, содержащая соответствие id пользователя и его интерфейсов
"Id" (INT) – id пользователя
"LTE" (BOOL) – наличие у пользователя интерфейса LTE
"LoRa" (BOOL) – наличие у пользователя интерфейса LoRa
"WiFi" (BOOL) – наличие у пользователя интерфейса WiFi
2. **Log** – таблица с логом переданных сообщений
"Moment" (DATETIME) – время передачи
"IdUser" (INT) – id пользователя
"Interface" (VARCHAR(255)) – название интерфейса
"Message" (VARCHAR(255)) – переданное сообщение
3. **WiFi** – таблица, содержащая соответствие внутреннего и внешнего адреса и id пользователя для интерфейса WiFi
"IdUser" (INT) – id пользователя
"IpIn" (VARCHAR(255)) – внутренний адрес
"IpOut" (VARCHAR(255)) – внешний адрес
4. **LoRa** – таблица, содержащая соответствие внутреннего и внешнего адреса и id пользователя для интерфейса LoRa
"IdUser" (INT) – id пользователя
"IpIn" (VARCHAR(255)) – внутренний адрес
"IpOut" (VARCHAR(255)) – внешний адрес
5. **LTE** – таблица, содержащая соответствие внутреннего и внешнего адреса и id пользователя для интерфейса LTE
"IdUser" (INT) – id пользователя
"IpIn" (VARCHAR(255)) – внутренний адрес
"IpOut" (VARCHAR(255)) – внешний адрес

LTE

При выходе во внешнюю сеть маскарад производит подмену внутреннего IP на внешний.

Работа с маскарадом и Ansible Playbook

В ходе разработки проекта были написаны playbooks, выполняющие добавление, удаление, изменение, просмотр правил masquerade. Запуск нужного playbook определяется действием, которое хочет совершить администратор, и дополнительными параметрами, такими как внутренний и внешний IP, также данные параметры заносятся в БД.

Администратор отправляет запрос на действие через сайт администратора.

В ходе работы с Ansible было настроено SSH подключение между двумя машинами, развернут сервер на JS, а также изучены функции для работы с playbook через JS.

Таким образом, работу блока, отвечающего за соединение с masquerade, можно описать следующим образом: включённый на выбранном порту сервер, слушает запросы от сайта администратора, и по действиям и параметрам запроса, модифицирует таблицу masquerade.

Оконечное устройство

Реализовано приложение с графическим интерфейсом для оконечного устройства. Оконечное устройство имеет внутренний IP и один или несколько интерфейсов для подключения к сети.

С помощью маскарада IP адресов устройство получает внешний IP адрес, который необходим для подключения к внешней сети.

Абонент с помощью сайта, предназначенного для пользователя, может отправить сообщение на оконечное устройство.

Полученное сообщение выводится на экран.

Пример окна приложения представлено на рисунке 2:

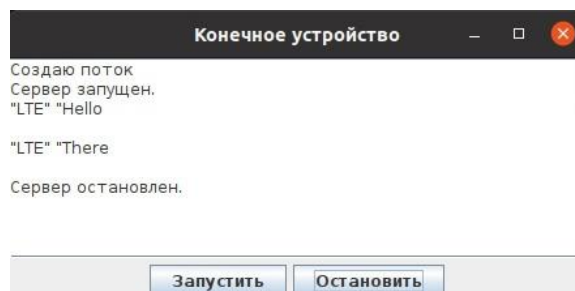


Рисунок 2 – Пример работы приложения

Контейнеризация

Чтобы организовать быстрое развертывание данного проекта, было создано два DockerFile для сборки контейнеров. Они были переданы команде CI/CD для сборки и тестирования.

Результаты тестирования контейнеров представлены на рисунках 3,4:

LIBRARY	VULNERABILITY ID	SEVERITY	INSTALLED VERSION	FIXED VERSION	TITLE
libgnutls-dane0	CVE-2021-20231	CRITICAL	3.6.7-4+deb10u6		gnutls: Use after free in client key_share extension -->avd.aquasec.com/nvd/cve-2021-20231
	CVE-2021-20232				gnutls: Use after free in client_send_params in lib/ext/pre_shared_key.c -->avd.aquasec.com/nvd/cve-2021-20232
libgnutls-openssl27	CVE-2021-20231				gnutls: Use after free in client key_share extension -->avd.aquasec.com/nvd/cve-2021-20231
	CVE-2021-20232				gnutls: Use after free in client_send_params in lib/ext/pre_shared_key.c -->avd.aquasec.com/nvd/cve-2021-20232
libgnutls28-dev	CVE-2021-20231				gnutls: Use after free in client key_share extension -->avd.aquasec.com/nvd/cve-2021-20231
	CVE-2021-20232				gnutls: Use after free in client_send_params in lib/ext/pre_shared_key.c -->avd.aquasec.com/nvd/cve-2021-20232
libgnutls30	CVE-2021-20231				gnutls: Use after free in client key_share extension -->avd.aquasec.com/nvd/cve-2021-20231
	CVE-2021-20232				gnutls: Use after free in client_send_params in lib/ext/pre_shared_key.c -->avd.aquasec.com/nvd/cve-2021-20232
libgnutlsxx28	CVE-2021-20231				gnutls: Use after free in client key_share extension -->avd.aquasec.com/nvd/cve-2021-20231
	CVE-2021-20232				gnutls: Use after free in client_send_params in lib/ext/pre_shared_key.c -->avd.aquasec.com/nvd/cve-2021-20232
libpython2.7-minimal	CVE-2021-3177		2.7.16-2+deb10u1		python: stack-based buffer overflow in PyCArg_repr in _ctypes/callproc.c -->avd.aquasec.com/nvd/cve-2021-3177
libpython2.7-stdlib					
python2.7					
python2.7-minimal					
python3-cryptography	CVE-2020-36242		2.6.1-3+deb10u2		python-cryptography: certain sequences of update() calls when symmetrically encrypting very large payloads... -->avd.aquasec.com/nvd/cve-2020-36242

Рисунок 3 - Результаты тестирования контейнеров

LIBRARY	VULNERABILITY ID	SEVERITY	INSTALLED VERSION	FIXED VERSION	TITLE
libgnutls-dane0	CVE-2021-20231	CRITICAL	3.6.7-4+deb10u6		gnutls: Use after free in client key_share extension -->avd.aquasec.com/nvd/cve-2021-20231
	CVE-2021-20232				gnutls: Use after free in client_send_params in lib/ext/pre_shared_key.c -->avd.aquasec.com/nvd/cve-2021-20232
libgnutls-openssl127	CVE-2021-20231				gnutls: Use after free in client key_share extension -->avd.aquasec.com/nvd/cve-2021-20231
	CVE-2021-20232				gnutls: Use after free in client_send_params in lib/ext/pre_shared_key.c -->avd.aquasec.com/nvd/cve-2021-20232
libgnutls28-dev	CVE-2021-20231				gnutls: Use after free in client key_share extension -->avd.aquasec.com/nvd/cve-2021-20231
	CVE-2021-20232				gnutls: Use after free in client_send_params in lib/ext/pre_shared_key.c -->avd.aquasec.com/nvd/cve-2021-20232
libgnutls30	CVE-2021-20231				gnutls: Use after free in client key_share extension -->avd.aquasec.com/nvd/cve-2021-20231
	CVE-2021-20232				gnutls: Use after free in client_send_params in lib/ext/pre_shared_key.c -->avd.aquasec.com/nvd/cve-2021-20232
libgnutlsxx28	CVE-2021-20231				gnutls: Use after free in client key_share extension -->avd.aquasec.com/nvd/cve-2021-20231
	CVE-2021-20232				gnutls: Use after free in client_send_params in lib/ext/pre_shared_key.c -->avd.aquasec.com/nvd/cve-2021-20232
libpython2.7-minimal	CVE-2021-3177		2.7.16-2+deb10u1		python: stack-based buffer overflow in PyCArg_repr in _ctypes/callproc.c -->avd.aquasec.com/nvd/cve-2021-3177
libpython2.7-stdlib					
python2.7					
python2.7-minimal					

Рисунок 4 - Результаты тестирования контейнеров

Вклад в проект

Задача 1: API. АС: Работа с сетью

Необходимо было реализовать программу, которая открывает сокеты и по TCP протоколу обменивается данными. Код поместить на Git и написать тесты.

TCP

Теоретическая часть

Взаимодействие между устройствами в рамках стека TCP/IP осуществляется с помощью связки IP адреса и порта.

Пара адрес и порт образует сокет (с английского socket – «гнездо»).

Сокет – является программным интерфейсом, который обеспечивает обмен данными между устройствами на низком уровне (рисунок 5).

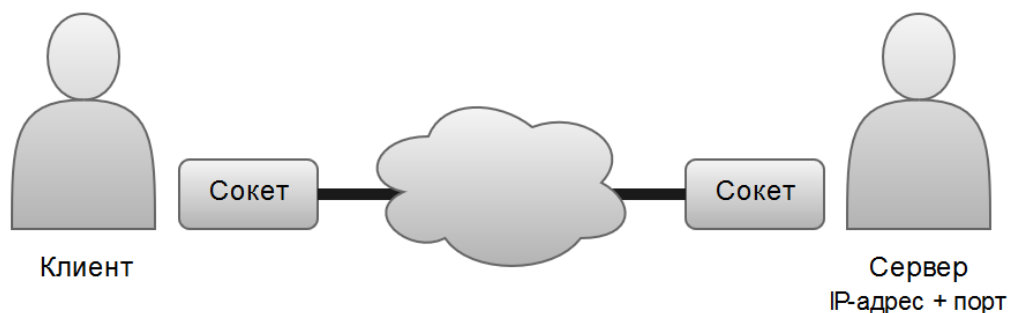


Рисунок 5 - Общение с помощью сокетов

Протокол TCP/IP основывается на соединениях, устанавливаемых между двумя компьютерами, обычно называемых клиентом и сервером. Поэтому, различают сокет клиента и сокет сервера.

Для организации общения клиент должен знать IP адрес и номер порта сервера, по которым он подключается к удаленному устройству.

В рамках стека протоколов TCP/IP различают два типа сокетов TCP и UDP. Также, TCP сокеты называют *потокowymi*, а UDP – *датаграммными*.

Реализация поставленной задачи

Написан текстовый многопользовательский чат.

Пользователь управляет клиентом.

На сервере пользователя нет.

Сервер занимается пересылкой сообщений между клиентами.

По умолчанию сообщение посылается всем участникам чата.

Есть команда послать сообщение конкретному пользователю (@senduser Vasya).

Программа работает по протоколу TCP.

Дополнительно была добавлена команда @kill, которая приводит к завершению программы собеседника.

Инструкция

Сервер запускается с аргументом – портом, клиенты – портом и IP-адресами.

На сервере формируется сокет client, в который передается порт и хэшмэп для хранения клиентов – ключи и их IP-значения.

Сокет и хэшмэп передаются в класс InputOutputForServer extends Thread.

Здесь с помощью метода equals происходит отлавливание команды @senduser. Если она найдена, то в хэшмэпе происходит поиск среди ключей никнейма клиента, берется соответствующий никнейму IP и формируется сокет для отправки сообщения по IP.

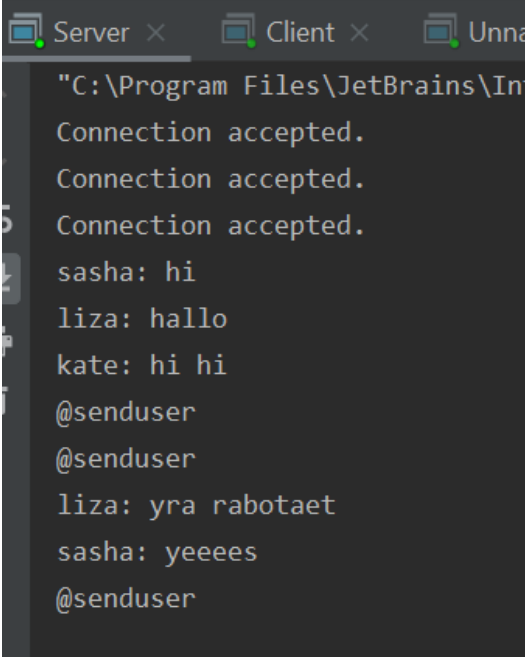
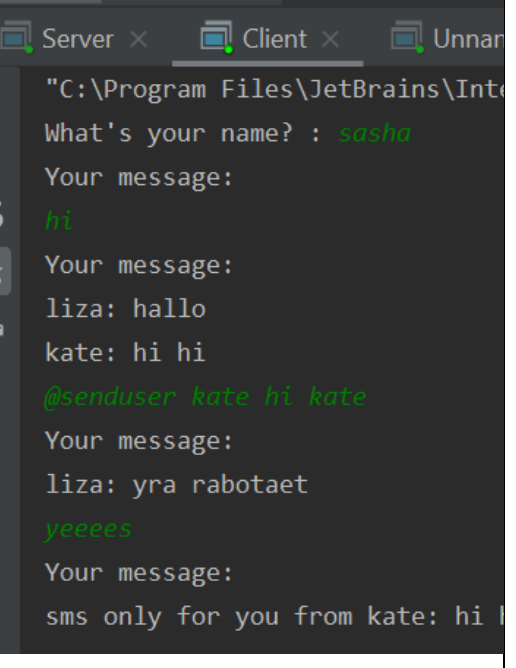
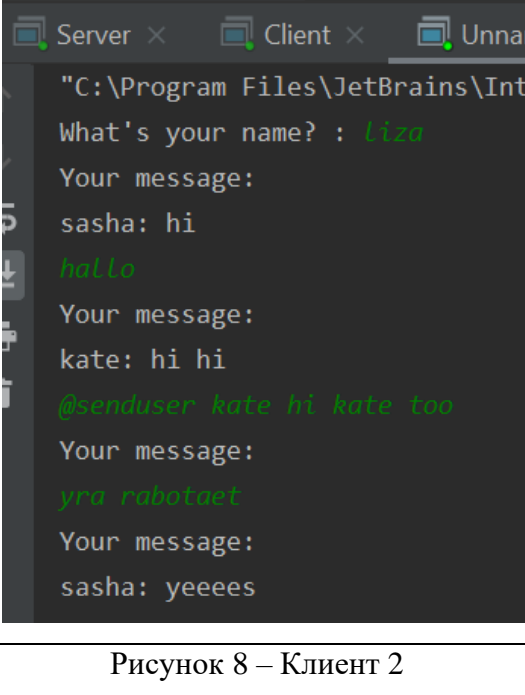
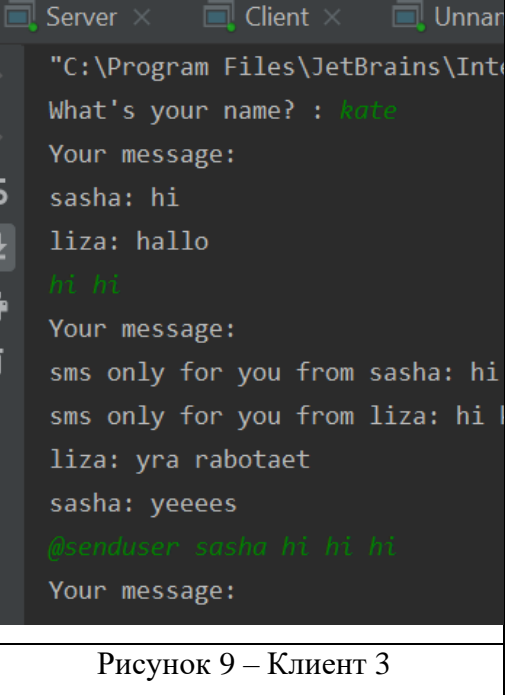
В class OutputThread extends Thread прописаны методы вывода имени и сообщений клиентов, в class InputThread extends Thread – ввода.

В классе Client формируется socket client с входными аргументами - порт и ip.

После запуска программы сервер занимается пересылкой сообщений.

Клиентам дается указание представиться и ввести сообщение.

Тестирование кода

 <pre>Server x Client x Unna "C:\Program Files\JetBrains\IntelliJ IDEA\bin\idea.exe" Connection accepted. Connection accepted. Connection accepted. sasha: hi liza: hallo kate: hi hi @senduser @senduser liza: yra rabotaet sasha: yeeeees @senduser</pre>	 <pre>Server x Client x Unna "C:\Program Files\JetBrains\IntelliJ IDEA\bin\idea.exe" What's your name? : sasha Your message: hi Your message: liza: hallo kate: hi hi @senduser kate hi kate Your message: liza: yra rabotaet yeeeees Your message: sms only for you from kate: hi hi</pre>
Рисунок 6 - Сервер	Рисунок 7 – Клиент 1
 <pre>Server x Client x Unna "C:\Program Files\JetBrains\IntelliJ IDEA\bin\idea.exe" What's your name? : liza Your message: sasha: hi hallo Your message: kate: hi hi @senduser kate hi kate too Your message: yra rabotaet Your message: sasha: yeeeees</pre>	 <pre>Server x Client x Unna "C:\Program Files\JetBrains\IntelliJ IDEA\bin\idea.exe" What's your name? : kate Your message: sasha: hi liza: hallo hi hi Your message: sms only for you from sasha: hi sms only for you from liza: hi hi liza: yra rabotaet sasha: yeeeees @senduser sasha hi hi hi Your message:</pre>
Рисунок 8 – Клиент 2	Рисунок 9 – Клиент 3

Ссылка на Git (код + тесты)

https://github.com/SUAI-Smart-Space-Team/LTE/tree/dev_test/test/TCP_test-main

Особенности сокетов TCP

Использование TCP сокетов позволяет приложениям клиента и сервера обмениваться данными почти прозрачно, не заботясь о поддержании сетевого соединения, доставке пакетов по сети, порядке передачи пакетов и буферизации. TCP сокет гарантирует доставку сообщений и правильный порядок пакетов, а также пересылает пакеты повторно, если подтверждение о передаче не приходит в течение определенного промежутка времени. Таким образом, использовать TCP сокет уместно там, где необходима гарантированная доставка данных сетевыми средствами.

Несмотря на многие преимущества, TCP сокет имеет и негативные стороны.

Например, необходимость поддержания TCP-соединения уменьшает пропускную способность обмена данными в распределенных системах. Также, в системах обмена данными реального времени повторная передача потерянных пакетов может привести к тому, что система получит данные, которые утратили свою актуальность.

В связи с чем, несмотря на поставленную задачу, в дальнейшем было принято решение использовать UDP.

UDP

Теоретическая часть

Все перечисленные недостатки TCP сокетов связаны с особенностью TCP-протокола. Если в системе присутствие данных факторов крайне нежелательно, а гарантированность доставки сообщений не является критичным требованием, то в качестве альтернативы TCP сокетов могут использоваться UDP (датаграммные) сокет.

UDP сокет устроен проще, чем TCP. В качестве транспортного уровня используется протокол UDP, который не требует установления соединения и подтверждения приема. Информация пересылается в предположении, что принимающая сторона ее ожидает. Датаграммные сокет не контролирует ничего, кроме целостности полученных датаграмм.

Сервер в ответ на запросы клиента передает по сети текущие (мгновенные) значения некоторого параметра контролируемого технологического процесса, а клиент

формирует управляющий сигнал на основе принятых значений. Если темп опроса сервера клиентом много больше требуемого времени реакции алгоритма управления на изменение значения контролируемого параметра, то потеря одного-двух сообщений от сервера несущественно повлияет на качество формирования управляющего сигнала.

В случае использования TCP соединения потерянное сообщение будет автоматически передано повторно, что может привести к получению клиентом неактуального значения контролируемого параметра и к формированию неправильного управляющего сигнала.

Основное задание

Написан текстовый чат для двух пользователей на сокетах.

Чат реализован по принципу клиент-сервер. Один пользователь находится на сервере, второй - на клиенте. Адреса и порты задаются через командную строку: клиенту - куда соединяться, серверу - на каком порту слушать.

При старте программы выводится текстовое приглашение, в котором можно ввести одну из следующих команд:

- 1) Задать имя пользователя (@name Vasya)
- 2) Послать текстовое сообщение (Hello)
- 3) Выход (@quit)

Принятые сообщения автоматически выводятся на экран.

Программа работает по протоколу UDP.

Инструкция

После запуска программы клиента спрашивают к какому порту подключаться, а у сервера - на каком порту слушать.

Затем клиенту предлагают ввести имя, сообщение и также озвучено условия выхода из чата – слово quit.

Клиент вводит текст, имя через знак @, которое в последствии будет считано сервером как имя пользователя и в какой-то момент выходит из чата.

Все сообщения клиента отображаются на сервере.

Сервер аналогичным образом может посылать сообщения клиенту.

Все сообщения в программе упакованы в пакеты

DatagramPacket pack = new DatagramPacket(data, data.length, addr, port),

а так же созданы сокеты DatagramSocket ds = new DatagramSocket().

IP-адреса берутся как localhost, порты задаются через командную строку.

Дейтаграммы отправляются с помощью метода ds.send(pack).

Прием и распаковка дейтаграмм производится в обратном порядке, вместо метода send () применяется метод receive (DatagramPacket pack).

Тестирование

<pre>C:\11proga>java UDP2Client kuda podklutchatsia ? 9876 @name? sms? break = quit @User Never say never Everyone is the creator of one's own fate Tolerance is more powerful than force. quit FROM SERVER: Server:- God never makes errors.</pre>	<pre>C:\11proga>java UDP2Server na kakom portu slushat' ? 9876 Client User :-@User Client User :-Never say never Client User :-Everyone is the creator of one's own fate Client User :-Tolerance is more powerful than force. Client User :-quit Client sent quit.....EXITING Your answer: God never makes errors. quit C:\11proga></pre>
--	---

Рисунок 10 – Результат работы программы

<pre>C:\11proga>java UDP2Client kuda podklutchatsia ? 9876 @name? sms? break = quit @User 2 Chop your own wood and it will warm Whose statement is this? quit FROM SERVER: Server:- Henry Ford</pre>	<pre>C:\11proga>java UDP2Server na kakom portu slushat' ? 9876 Client User 2 :-@User 2 Client User 2 :-Chop your own wood and it will warm you twice. Client User 2 :-Whose statement is this? Client User 2 :-quit Client sent quit.....EXITING Your answer: Henry Ford quit C:\11proga></pre>
---	---

Рисунок 11 - Результат работы программы

Задача 2: внутренне тестирование найти баги в текущих приложениях как пользователь системы

Реализация

Тесты как пользователя

1. Была найдена ошибка при повторном открытии сокета по одному и тому же номеру порта.
2. Было определено, что можно сокеты открывать сколько угодно раз подряд (не закрывая) с разных портов, но нельзя их повторять.
3. Не удалось подключиться с двух приложений к одному сокету.
4. При вводе вместо числа в графу «порт» других знаков - выдается ошибка о некорректном вводе. Окно с ошибкой не свернуть, пока не будет введен порт. Кнопки cancel не помогают.
5. Не было реализовано окно для ввода сообщения и не формировался список пользователей, указанный в коде.

В последствии после данных замечаний код был отредактирован.

Тесты JUnit

1. Написан тест, создающий потоки для прослушивания порта

Первый поток не получает никакого отклика после отправки второго сокета.

Код на Git: https://github.com/SUAI-Smart-Space-Team/LTE/blob/dev_test/test/LTE-main/src/test/java/Server/ServerTest2.java

2. Написан тест на проверку сервера

Код на Git: https://github.com/SUAI-Smart-Space-Team/LTE/blob/dev_test/test/LTE-main/src/test/java/Server/ServerTest.java

Отдельно о том, как были реализованы тесты:

Тестирование. JUnit

Для тестирования модели была использована библиотека JUnit.

Для подключения библиотеки JUnit используется maven.

Класс Controller послужил основой для написания юнит тестов. Тестирующие классы находятся в пакете test.

В IntelliJ IDEA тестирующие классы были созданы автоматически. Для этого нужно было нажать alt + enter на классе и выбрать «Create test». Далее были выбраны методы, которые нужно было протестировать.

В результате были созданы классы, в названии которых присутствует слово Test выбранными методами.

В классе для указания на тестирующие методы, перечисленные выше, была использована аннотация @Test.

Фикстурой @Before задан момент создания объекта класса перед каждым тестом.

Такой аннотацией обозначен метод в class ServerTest2.

Данная аннотация говорит о том, что этот метод выполняется один раз перед всеми тестами.

Выводы

1. В ходе работы над данным проектом были изучены принципы работы с LTE и masquerade.
2. Получены навыки работы с программным интерфейсом приложения.
3. Была создана база данных устройств для работы с API.
4. Реализован способ передачи данных через интерфейс Wi-Fi и взаимодействие с masquerade.
5. Написано приложение с графическим интерфейсом для получения сообщений от пользователя.