

Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Кафедра компьютерных систем и программных технологий

Сети и телекоммуникации

Отчет по лабораторной работе
по сетевым технологиям

Работу
выполнил:
Шустенковс О.
Группа: 43501/1
Преподаватель:
Алексюк А.О.

Санкт-Петербург
2018

1. Цель работы

Ознакомиться с принципами программирования собственных протоколов, созданных на основе TCP и UDP.

2. Описание выполненных работ по TCP и UDP

В ходе выполнения лабораторных работ были написаны простейшие клиент-серверные приложения на базе протоколов TCP и UDP. В приложениях TCP создается сокет, ставится на прослушивание и при подключении клиента создается отдельный сокет, по которому клиент общается с сервером.

2.1. TCP

Для инициализации, запуска и завершения TCP-сервера необходимо выполнить следующие системные вызовы:

- `socket()` - создание сокета
- `bind()` - привязка созданного сокета к заданным IP-адресам и портам (сервер)
- `listen()` – перевод сокета в состояние прослушивания
- `accept()` - прием поступающих запросов на подключение и возврат сокета для нового соединения
- `read()` - чтение данных от клиента (контроль конца сообщения)
- `send()` - отправка данных клиенту с помощью того же сокета
- `shutdown()` - разрыв соединения с клиентом
- `close()` - закрытие клиентского и слушающего сокетов

TCP-клиенты выполняют следующую последовательность действий для открытия соединения, отправки и получения данных, и завершения:

- `socket()` - создание сокета
- `connect()` - установка соединения для сокета, который будет связан с серверным сокетом, порожденным вызовом `accept()`
- `send()` - отправка данных серверу
- `read()` - прием данных от сервера (контроль конца сообщения)
- `shutdown()` - разрыв соединения с сервером
- `close()` - закрытие сокета

Для поддержки работы с несколькими клиентами создается поток, в котором создается сокет для общения с клиентом.

2.2. UDP

В UDP сервер принимает сообщение от клиента и отправляет сообщение об успешной доставке. UDP протокол не подразумевает логических соединений, поэтому не создается слушающего сокета.

Реализация UDP-сервера:

- `socket()` - создание сокета
- `bind()` - привязка созданного сокета к заданным IP-адресам и портам (сервер)
- `recvfrom()` - получение данных от клиента, параметры которого заполняются функцией
- `sendto()` - отправка данных с указанием параметров клиента, полученных на предыдущем шаге
- `close()` - закрытие сокета

UDP-клиент для обмена данными с UDP-сервером использует следующие функции:

- `socket()` - создание сокета
- `recvfrom()` - получение данных от сервера, параметры которого заполняются функцией
- `sendto()` - отправка данных с указанием параметров сервера, полученных на предыдущем шаге
- `close()` - закрывает сокет

3. Индивидуальное задание

Задание: разработать приложение-клиент и приложение сервер электронной почты.

Основные возможности. Серверное приложение должно реализовывать следующие функции:

1. Прослушивание определенного порта
2. Обработка запросов на подключение по этому порту от клиентов
3. Поддержка одновременной работы нескольких почтовых клиентов через механизм нитей
4. Приём почтового сообщения от одного клиента для другого
5. Хранение электронной почты для клиентов
6. Посылка клиенту почтового сообщения по запросу с последующим удалением сообщения
7. Посылка клиенту сведений о состоянии почтового ящика
8. Обработка запрос на отключение клиента
9. Принудительное отключение клиента

Клиентское приложение должно реализовывать следующие функции:

1. Установление соединения с сервером
2. Передача электронного письма на сервер для другого клиента
3. Проверка состояния своего почтового ящика
4. Получение конкретного письма с сервера
5. Разрыв соединения
6. Обработка ситуации отключения клиента сервером

Настройки приложений. Разработанное клиентское приложение должно предоставлять пользователю настройку IP-адреса или доменного имени сервера электронной почты, номера порта, используемого сервером, идентификационной информации пользователя. Разработанное серверное приложение должно предоставлять пользователю настройку списка пользователей почтового сервера.

Методика тестирования. Для тестирования приложений запускается сервер электронной почты и несколько клиентов. В процессе тестирования проверяются основные возможности приложений по передаче и приёму сообщений.

4. Разработанный прикладной протокол

После подключения клиента к серверу, сервер ждем команд от клиента. Существует 5 возможных команд от клиента.

Поддерживаемые команды клиента:

1. login <имя> - авторизация клиента
2. send <сообщение> <имя получателей> - отправка сообщения пользователю (можно нескольким, через ;)
3. list - список принятых сообщений
4. read <номер письма> - полное чтение присланного письма
5. delete <номер письма> - удаление письма под определенным номером.

Если клиентом введена несуществующая команда, то будет выведено сообщение о неверной команде и будут предложены существующие команды.

Ответы на команды клиента:

1. Bad name - когда не ввели имя или имя > 64 символов
2. Already logged in as - повторная попытка авторизации
3. Logged in as - успешная авторизация
4. Please log in - попытка использовать команды без авторизации
5. Letter not found - при чтении/удалении сообщения нет введенного номера
6. Deleted successfully - успешное удаления сообщения

7. Failed to send message - по каким то причинам сообщение не отправилось
8. Message sent - сообщение успешно отправлено

Поддерживаемые команды сервера:

1. list - список подсоединившихся клиентов
2. kick <slot> - отключение определенного клиента (<slot> - номер подключения)
3. exit - завершение работы сервера

Отклик-сообщения на сервере:

1. Connection - подключен новый клиент
2. Received from client login/send/list/read/delete - получена команда от определенного клиента
3. Connection terminated by remote user. Terminating client connection - при отключении определенного клиента
4. Server terminated - выключение сервера

5. Описание архитектуры и особенности реализации TSP

Изначально происходит создание серверного сокета, который затем будет прослушивать подключения на определённом, заданном при создании, порту. Для того, чтобы сервер мог параллельно с установкой соединений самостоятельно писать и выполнять команды, создается новый поток, в котором как раз и осуществляется ожидание новых подключений. В основном потоке сервер принимает команды из командной строки, а затем осуществляет выполнение одной из следующих команд: завершение сервера, закрытие определенного клиента и просмотр списка клиентов, подключенных в данный момент. Все операции защищены мьютексами, так как возможно одновременное обращение из нескольких потоков. В поточной функции происходит ожидание новых подключений в цикле. Когда клиент подключается к серверу, клиентский сокет добавляется, а на работу клиента выделяется отдельный поток, в который передается сокет для общения с клиентом. Поточная функция ожидает команды от клиента.

В отдельном текстовом файле хранятся логины ранее зарегистрированных пользователей. После отключения сервера все залогиненные пользователи (имена) отправляются в файл. Если имени нет в файле, то письмо получателю не придет.

6. Описание архитектуры и особенности реализации UDP

Основная логика взаимодействия клиента и сервера осталась такой же, как у TSP (в том числе и хранение логинов). Однако в данном случае нет необходимости создавать отдельные сокеты и потоки для каждого клиента. Обработка всех запросов происходит в одном потоке.

Клиент идентифицируется по ip и port внутри принимаемого пакета, которым ставится в соответствие определенный подключенный клиент - его состояние в виде текущего имени пользователя или его отсутствия. На сервере есть возможность отключить пользователя, в этом случае ip и port попадают в черный список (пакеты от этого клиента дропаются).

7. Тестирование приложения на основе TSP

Для тестирования приложения запускался сервер и несколько клиентов. Проверялись все команды поддерживаемые сервером в различных комбинациях. В результате тестирования ошибок выявлено не было, из чего можно сделать вывод, что приложение работает корректно.

Листинг 1: Пример вывода информации при работе с клиентом

```
1 Connected to server 127.0.0.1:5001
2 Available commands:
3 login <name>
4 send <message> <recepipients>
5 list
6 read <number>
7 delete <number>
8
9 login anna
10 Logged in as: anna
11 list
12 Current messages for anna:
13 0: Hello! How are y
14 read 0
15 Hello! How are you today?
16
17 Received from bob
18 list
19 Current messages for anna:
20 0: Hello! How are y
21 delete 0
22 Deleted successfully
23 list
24 Current messages for anna:
25
26 login bob
27 Already logged in as: anna
```

8. Тестирование приложения на основе UDP

Для тестирования приложения запускался сервер и несколько клиентов. Проверялись все команды поддерживаемые сервером в различных комбинациях. В результате тестирования ошибок выявлено не было, из чего можно сделать вывод, что приложение работает корректно.

Листинг 2: Пример вывода информации при работе с клиентом

```
1 Available commands:
2 login <name>
3 send <message> <recepipients>
4 list
5 read <number>
6 delete <number>
7 exit
8 login oleg
9 Processing package from 127.0.0.1:5001
10 Acknowledged package with number #0
11 Processing package from 127.0.0.1:5001
12 Logged in as: oleg
```

```
13 list
14 Processing package from 127.0.0.1:5001
15 Acknowledged package with number #1
16 Processing package from 127.0.0.1:5001
17 Current messages for oleg:
18 0: Hey! What was th
19 rear 0
20 Incorrect command. Available commands:
21 login <name>
22 send <message> <recepipients>
23 list
24 read <number>
25 delete <number>
26 exit
27
28 read 0
29 Processing package from 127.0.0.1:5001
30 Acknowledged package with number #2
31 Processing package from 127.0.0.1:5001
32 Hey! What was the weather today?
33
34 Received from billy
35 delete 0
36 Processing package from 127.0.0.1:5001
37 Acknowledged package with number #3
38 Processing package from 127.0.0.1:5001
39 Deleted successfully
40 list
41 Processing package from 127.0.0.1:5001
42 Acknowledged package with number #4
43 Processing package from 127.0.0.1:5001
44 Current messages for oleg:
45
46 login billy
47 Processing package from 127.0.0.1:5001
48 Acknowledged package with number #5
49 Processing package from 127.0.0.1:5001
50 Already logged in as: oleg
```

9. Листинги программ

Листинги программ находятся по адресу:

<https://github.com/OlegShust/NetworksLab2018/tree/individual-task>

10. Дополнительное задание

Исследуем реальные прикладные протоколы. Притворимся клиентом и подключимся к одному из существующих общедоступных серверов.

10.1. Подключение к HTTP-серверу и запрос веб-страницы

С помощью команды telnet подключаемся к HTTP-серверу:

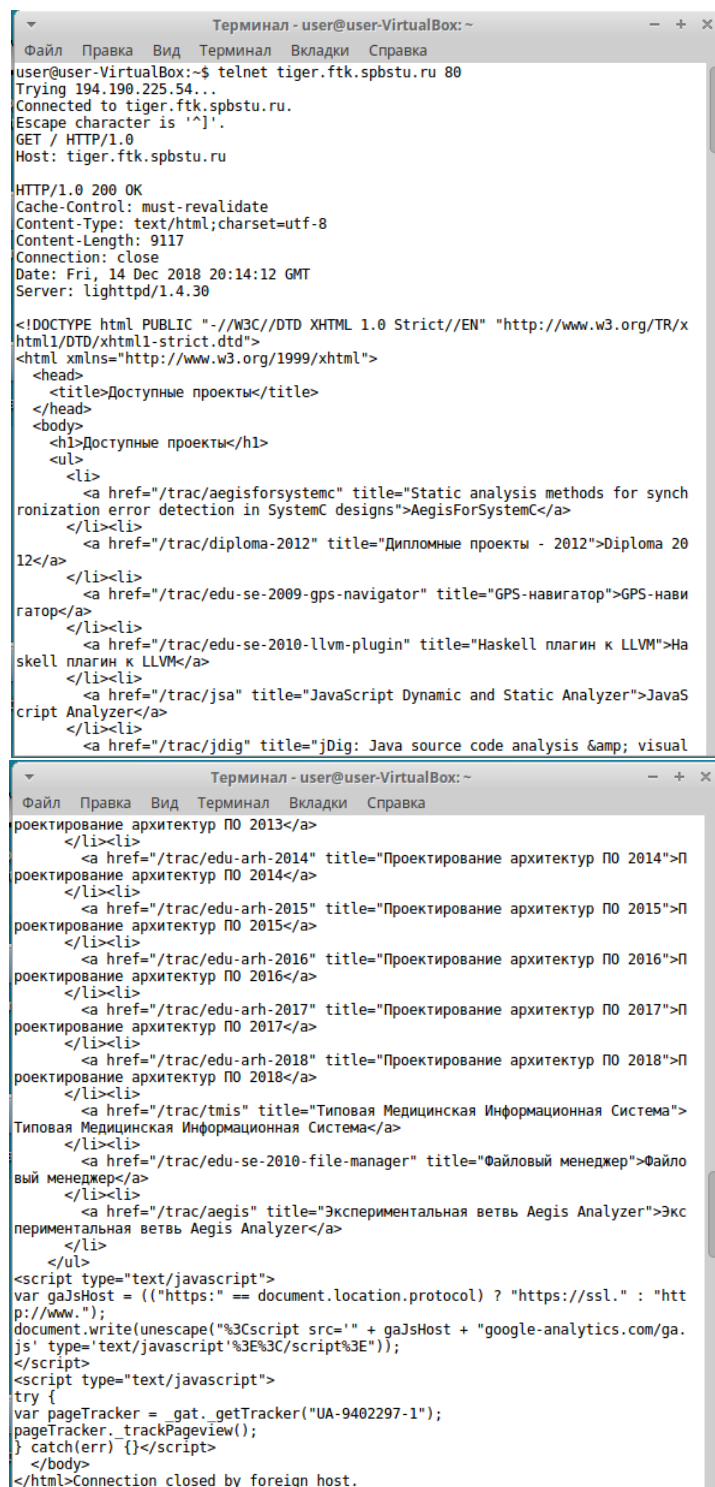
```
1 telnet tiger.ftk.spbstu.ru 80
```

tiger.ftk.spbstu.ru - подключаемый сервер, 80 - номер порта.

Затем с помощью команды запрашиваем определенную веб-страницу:

```
1 GET / HTTP/1.0
2 HOST: tiger.ftk.spbstu.ru
```

Веб-страница успешно выведена:



```
Терминал - user@user-VirtualBox: ~
Файл Правка Вид Терминал Вкладки Справка
user@user-VirtualBox:~$ telnet tiger.ftk.spbstu.ru 80
Trying 194.190.225.54...
Connected to tiger.ftk.spbstu.ru.
Escape character is '^'.
GET / HTTP/1.0
Host: tiger.ftk.spbstu.ru

HTTP/1.0 200 OK
Cache-Control: must-revalidate
Content-Type: text/html;charset=utf-8
Content-Length: 9117
Connection: close
Date: Fri, 14 Dec 2018 20:14:12 GMT
Server: lighttpd/1.4.30

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Доступные проекты</title>
  </head>
  <body>
    <h1>Доступные проекты</h1>
    <ul>
      <li>
        <a href="/trac/aegisforsystemc" title="Static analysis methods for synch
ronization error detection in SystemC designs">AegisForSystemC</a>
      </li>
      <li>
        <a href="/trac/diploma-2012" title="Дипломные проекты - 2012">Diploma 20
12</a>
      </li>
      <li>
        <a href="/trac/edu-se-2009-gps-navigator" title="GPS-навигатор">GPS-нави
гатор</a>
      </li>
      <li>
        <a href="/trac/edu-se-2010-llvm-plugin" title="Haskell плагин к LLVM">Ha
skell плагин к LLVM</a>
      </li>
      <li>
        <a href="/trac/jsa" title="JavaScript Dynamic and Static Analyzer">JavaS
cript Analyzer</a>
      </li>
      <li>
        <a href="/trac/jdig" title="jDig: Java source code analysis &amp; visual
роектирование архитектур ПО 2013</a>
      </li>
      <li>
        <a href="/trac/edu-arh-2014" title="Проектирование архитектур ПО 2014">П
роектирование архитектур ПО 2014</a>
      </li>
      <li>
        <a href="/trac/edu-arh-2015" title="Проектирование архитектур ПО 2015">П
роектирование архитектур ПО 2015</a>
      </li>
      <li>
        <a href="/trac/edu-arh-2016" title="Проектирование архитектур ПО 2016">П
роектирование архитектур ПО 2016</a>
      </li>
      <li>
        <a href="/trac/edu-arh-2017" title="Проектирование архитектур ПО 2017">П
роектирование архитектур ПО 2017</a>
      </li>
      <li>
        <a href="/trac/edu-arh-2018" title="Проектирование архитектур ПО 2018">П
роектирование архитектур ПО 2018</a>
      </li>
      <li>
        <a href="/trac/tmis" title="Типовая Медицинская Информационная Система">
Типовая Медицинская Информационная Система</a>
      </li>
      <li>
        <a href="/trac/edu-se-2010-file-manager" title="Файловый менеджер">Файло
вый менеджер</a>
      </li>
      <li>
        <a href="/trac/aegis" title="Экспериментальная ветвь Aegis Analyzer">Экс
периментальная ветвь Aegis Analyzer</a>
      </li>
    </ul>
    <script type="text/javascript">
var gaJsHost = (("https:" == document.location.protocol) ? "https://ssl." : "htt
p://www.");
document.write(unescape("%3Cscript src='" + gaJsHost + "google-analytics.com/ga.
js' type='text/javascript'%3E%3C/script%3E"));
</script>
<script type="text/javascript">
try {
var pageTracker = _gat._getTracker("UA-9402297-1");
pageTracker._trackPageview();
} catch(err) {}</script>
</body>
</html>Connection closed by foreign host.
```

Рис. 10.1.1. Подключение к HTTP-серверу

Сервер вернул код 200 в заголовке ответа, что говорит об успешной обработке запроса.

10.2. Подключение к FTP-серверу, запрос списка файлов в директории и получение файла

С помощью команды `telnet` подключаемся к FTP-серверу:

```
1 telnet ftp.stat.duke.edu 21
```

ftp.stat.duke.edu - ftp-сервер, к которому происходит подключение, 21 - порт.

Вводим имя пользователя и пароль. Затем необходимо перейти в пассивный режим с помощью команды `pasv`. С помощью цифр кодируется IP-адрес и порт для соединения. Создаем соединение к указанному IP-адресу через необходимый порт (порт высчитывается по формуле $p1*256 + p2$, где $p1$ и $p2$ - это два последних числа в присланном сообщении сервера). Стоит уточнить, что здесь используется одно соединение на одну команду. С помощью команды `rwd` и `cwd` переходим по папкам, пока не найдем необходимый файл. Для скачивания файла используем команду `retr <имя файла>`.

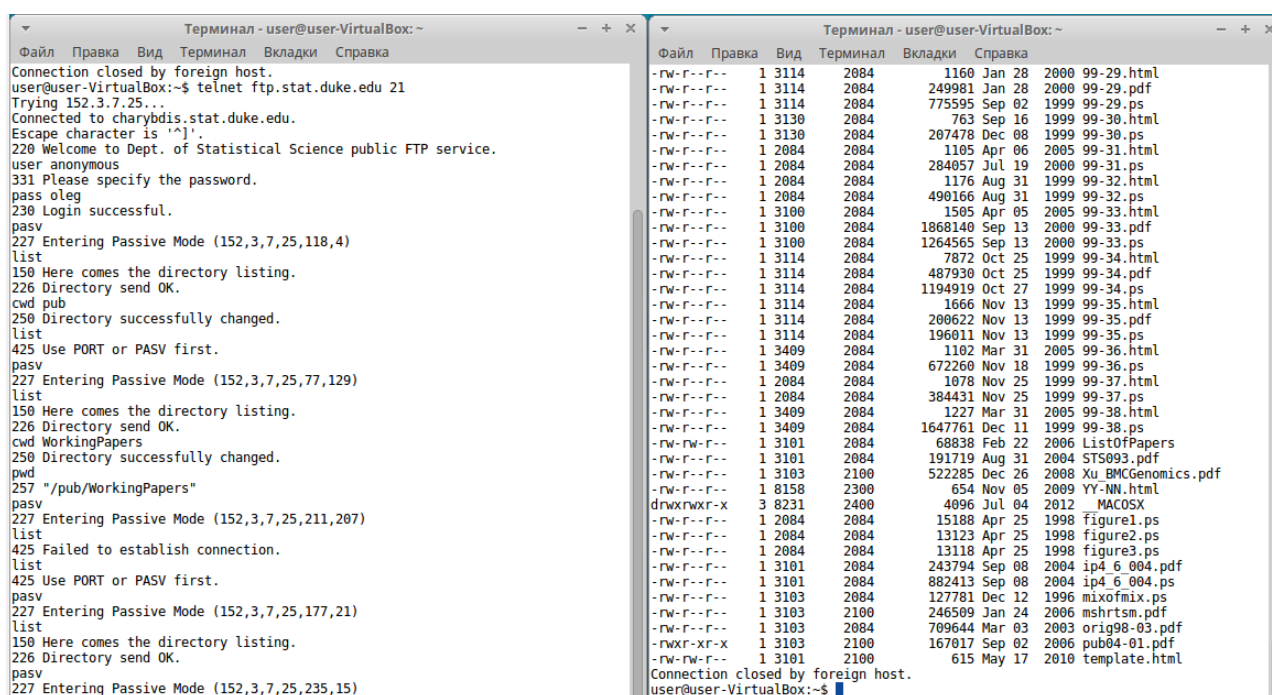


Рис. 10.2.1. Подключение к FTP-серверу

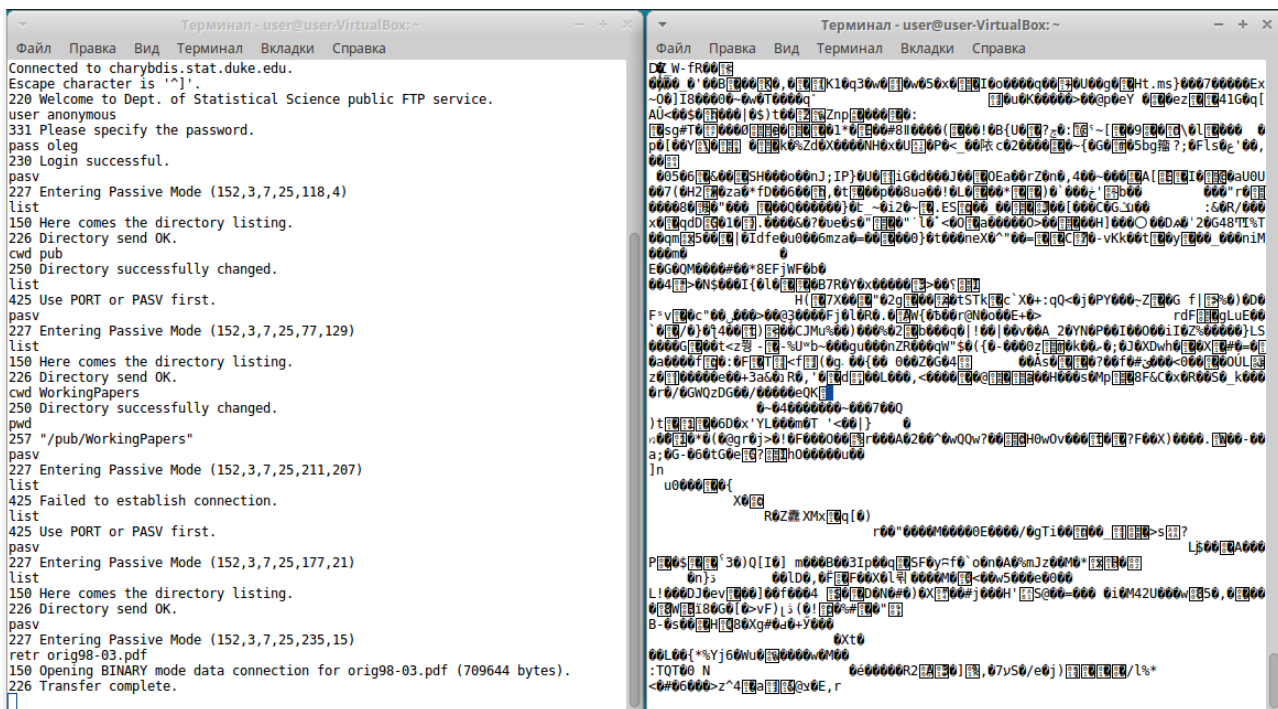


Рис. 10.2.2. Скачивание файла с FTP-сервера

Файл успешно скачан.

10.3. Подключение к SMTP-серверу и отправка письма

С помощью команды, указанной ниже, подключаемся к SMTP-серверу smtp.yandex.ru:

```
1 gnutls-cli -p 465 smtp.yandex.ru
```

```

Терминал - user@user-VirtualBox: ~
Файл  Правка  Вид  Терминал  Вкладки  Справка

user@user-VirtualBox:~$ gnutls-cli -p 465 smtp.yandex.ru
Processed 148 CA certificate(s).
Resolving 'smtp.yandex.ru'...
Connecting to '213.180.204.38:465'...
- Certificate type: X.509
- Got a certificate list of 3 certificates.
- Certificate[0] info:
- subject 'C=RU,O=Yandex LLC,OU=ITO,L=Moscow,ST=Russian Federation,CN=smtp.yand
ex.ru', issuer 'C=RU,O=Yandex LLC,OU=Yandex Certification Authority,CN=Yandex CA
', RSA key 2048 bits, signed using RSA-SHA256, activated '2017-10-11 13:27:26 UT
C', expires '2019-10-11 13:27:26 UTC', SHA-1 fingerprint '57bed6b3bce8e3b8c1a67a
9bc7faf28d480cabb'
Public Key ID:
29937c5abb1a07bc8f944012d6092af73f2306b8
Public key's random art:
+--[ RSA 2048 ]-----+
|  +o .
|  o.o
| o o .
|..o + o . .
| . . o B S
| . . o X .
|E   o X o
| . o B .
|      o.o
+-----+

- Certificate[1] info:
- subject 'C=RU,O=Yandex LLC,OU=Yandex Certification Authority,CN=Yandex CA', i
ssuer 'C=PL,O=Unizeto Technologies S.A.,OU=Certum Certification Authority,CN=Cer
tum Trusted Network CA', RSA key 2048 bits, signed using RSA-SHA256, activated '
2015-01-21 12:00:00 UTC', expires '2025-01-18 12:00:00 UTC', SHA-1 fingerprint '
ddf10e6da72c447ecad874eb531b49662d2c6ed2'
- Certificate[2] info:
- subject 'C=PL,O=Unizeto Technologies S.A.,OU=Certum Certification Authority,C
N=Certum Trusted Network CA', issuer 'C=PL,O=Unizeto Sp. z o.o.,CN=Certum CA', R
SA key 2048 bits, signed using RSA-SHA256, activated '2008-10-22 12:07:37 UTC',
expires '2027-06-10 10:46:39 UTC', SHA-1 fingerprint '929badf26081523490edc91154
b380a4776e2185'
- Status: The certificate is trusted.
- Description: (TLS1.2)-(ECDHE-RSA-SECP256R1)-(AES-128-GCM)

```

Рис. 10.3.1. Подключение к SMTP-серверу

Данная команда обеспечивает защищенное TLS-подключение. Команды telnet/nc здесь не подходят по причине их работы только с нешифрованными соединениями.

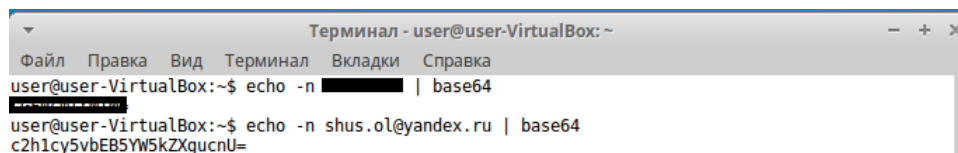
SSL (Secure Sockets Layer) и TLS (Transport Level Security) - это криптографические протоколы, обеспечивающие защищенную передачу данных в компьютерной сети. Они активно используются при работе с электронной почтой. Соединение, защищенное протоколом TLS, обладает определенными свойствами: безопасностью, аутентификацией и целостностью.

Затем необходимо пройти аутентификацию:

```
- Simple Client Mode:
220 smtp3o.mail.yandex.net ESMTP (Want to use Yandex.Mail for your domain? Visit
  http://pdd.yandex.ru)
ehlo yandex.ru
250-smtp3o.mail.yandex.net
250-8BITIME
250-PIPELINING
250-SIZE 42991616
250-AUTH LOGIN PLAIN XOAUTH2
250-DSN
250 ENHANCEDSTATUSCODES
AUTH LOGIN
334 VXNlcm5hbWU6
c2hlcy5vbEB5YW5kZXgucnU=
334 UGFzc3dvcmQ6
[REDACTED]
235 2.7.0 Authentication successful.
```

Рис. 10.3.2. Аутентификация

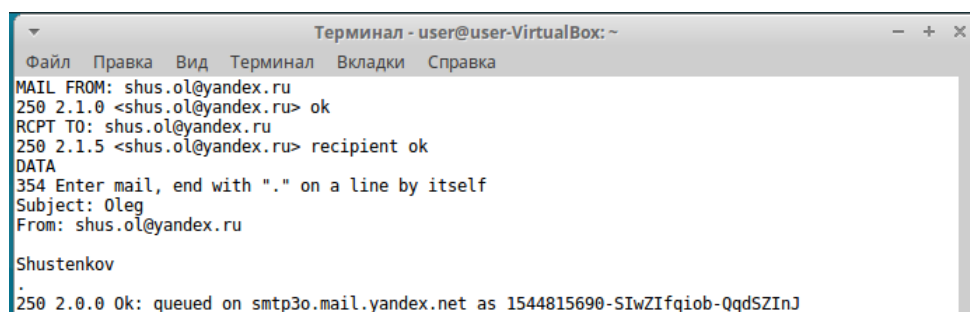
Для аутентификации использовался BASE64, с помощью которого были закодированы логин и пароль:



```
Терминал - user@user-VirtualBox: ~
Файл  Правка  Вид  Терминал  Вкладки  Справка
user@user-VirtualBox:~$ echo -n [REDACTED] | base64
[REDACTED]
user@user-VirtualBox:~$ echo -n shus.ol@yandex.ru | base64
c2h1cy5vbEB5YW5kZXgucnU=
```

Рис. 10.3.3. Кодирование логина и пароля

Используя команды MAIL FROM (от кого письмо), RCPT TO (кому письмо), DATA (само письмо с указанием темы (Subject) и From (от кого)), было отослано письмо с содержанием 'Shustenkov' и subject 'Oleg':



```
Терминал - user@user-VirtualBox: ~
Файл  Правка  Вид  Терминал  Вкладки  Справка
MAIL FROM: shus.ol@yandex.ru
250 2.1.0 <shus.ol@yandex.ru> ok
RCPT TO: shus.ol@yandex.ru
250 2.1.5 <shus.ol@yandex.ru> recipient ok
DATA
354 Enter mail, end with "." on a line by itself
Subject: Oleg
From: shus.ol@yandex.ru
Shustenkov
.
250 2.0.0 Ok: queued on smtp3o.mail.yandex.net as 1544815690-SIwZIfqiob-QqdSZInJ
```

Рис. 10.3.4. Посылка письма

10.4. Подключение к POP3-серверу, проверка почты и получение письма

Попробуем прочесть посланное письмо с помощью протокола POP3. Для начала подключимся к POP3-серверу:

```
Терминал - user@user-VirtualBox: ~
Файл Правка Вид Терминал Вкладки Справка
user@user-VirtualBox:~$ gnutls-cli -p 995 pop.yandex.ru
Processed 148 CA certificate(s).
Resolving 'pop.yandex.ru'...
Connecting to '87.250.251.37:995'...
- Certificate type: X.509
- Got a certificate list of 3 certificates.
- Certificate[0] info:
  - subject 'C=RU,O=Yandex LLC,OU=ITO,L=Moscow,ST=Russian Federation,CN=pop.yandex.ru'
  , issuer 'C=RU,O=Yandex LLC,OU=Yandex Certification Authority,CN=Yandex CA', RSA key
  2048 bits, signed using RSA-SHA256, activated '2018-04-23 09:35:54 UTC', expires '201
  9-04-23 09:35:54 UTC', SHA-1 fingerprint '097e52ca28074daa571f4695fd89fed272d9e028'
  Public Key ID:
    feddd2f2c302e1628c111cb50c0170d85bd4ac7e
  Public key's random art:
    +---[ RSA 2048 ]-----+
    |..+0+==*..|
    |...+00..|
    | 0 00 |
    | . 0 . |
    | .+S. . |
    | o.E o |
    | o.. . o |
    | .+.+ |
    | . =0. |
    +-----+
- Certificate[1] info:
  - subject 'C=RU,O=Yandex LLC,OU=Yandex Certification Authority,CN=Yandex CA', issuer
  'C=PL,O=Unizeto Technologies S.A.,OU=Certum Certification Authority,CN=Certum Truste
  d Network CA', RSA key 2048 bits, signed using RSA-SHA256, activated '2015-01-21 12:0
  0:00 UTC', expires '2025-01-18 12:00:00 UTC', SHA-1 fingerprint 'ddf10e6da72c447ecad8
  74eb531b49662d2c6ed2'
- Certificate[2] info:
  - subject 'C=PL,O=Unizeto Technologies S.A.,OU=Certum Certification Authority,CN=Cer
  tum Trusted Network CA', issuer 'C=PL,O=Unizeto Sp. z o.o.,CN=Certum CA', RSA key 204
  8 bits, signed using RSA-SHA256, activated '2008-10-22 12:07:37 UTC', expires '2027-0
  6-10 10:46:39 UTC', SHA-1 fingerprint '929badf26081523490edc91154b380a4776e2185'
- Status: The certificate is trusted.
- Description: (TLS1.2)-(RSA)-(AES-128-GCM)
```

Рис. 10.4.1. Подключение к POP3-серверу

Затем введем логин и пароль, используя команды USER и PASS:

```
Терминал - user@user-VirtualBox: ~
Файл Правка Вид Терминал Вкладки Справка
+OK POP Ya! na@3j 64uJC7DXfmI1
user shus.ol@yandex.ru
+OK password, please.
pass [REDACTED]
-ERR [AUTH] login failure or POP3 disabled, try later. sc=64uJC7DXfmI1_142004_3j
*** Fatal error: The TLS connection was non-properly terminated.
*** Server has terminated the connection abnormally.
```

Рис. 10.4.2. Аутентификация

Yandex отключил поддержку pop3 протокола, поэтому проверить почту не удалось. Была сделана попытка подключиться к pop3 gmail, но двойная аутентификация не позволила проверить почту.

11. Выводы

В данной лабораторной работе было реализовано клиент-серверное приложение электронной почты. Данная система обеспечивает параллельную работу нескольких клиентов.

В случае данного варианта индивидуального задания, было необходимо организовать работу почтового клиент-серверного приложения, реализующего многопоточность для общения с множеством клиентов, а также реализовать собственный протокол.

В ходе курса были получены основные навыки разработки прикладных сетевых приложений; навыки разработки собственных прикладных протоколов обмена; навыки разработки многопоточных сетевых приложений; знания по организации сетевого обмена по транспортным протоколам TCP и UDP.

На примере данной разработки были изучены основные приемы использования протокола транспортного уровня TCP – транспортного механизма, предоставляющего поток данных, с предварительной установкой соединения, за счёт этого дающего уверенность в достоверности получаемых данных, осуществляющего повторный запрос данных в случае потери данных и устраняющего дублирование при получении двух копий одного пакета. Данный механизм, в отличие от UDP, гарантирует, что приложение получит данные точно в такой же последовательности, в какой они были отправлены, и без потерь. При подключении нового клиента создается новый сокет, что значительно упрощает создание многоклиентского приложения на основе нитей. Нити в многопоточном приложении позволяют таким образом "разгрузить" обработку данных от нескольких клиентов сервером, который каждому клиенту в отдельном потоке назначает свой обработчик.

Также были получены навыки работы с такими прикладными протоколами, как HTTP, FTP, SMTP и POP3. Эти навыки могут оказаться полезными при разработке серьезных сетевых приложений. Некоторые задачи данной лабораторной очень схожи с задачами, решаемыми этими протоколами. Так SMTP ожидает ответа на каждую операцию и присылает обратно код - выполнена ли операция или завершилась с какой-либо ошибкой. В POP3, SMTP, а так же FTP перед тем, как выполнить какую-либо операцию, необходимо было произвести аутентификацию.