САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
КАФЕДРА СИСТЕМНОГО ПРОГРАММИРОВАНИЯ

ЗЕЛЕНЧУК ИЛЬЯ ВАЛЕРЬЕВИЧ
КИРИЛЕНКО ЯКОВ АЛЕКСАНДРОВИЧ
НЕМЕШЕВ МАРАТ ХАЛИМОВИЧ

ТЕЛЕКОММУНИКАЦИИ
МЕТОДИЧЕСКОЕ ПОСОБИЕ

Дисциплина [003730] «Телекоммуникации»
по направлению 09.03.04 «Программная инженерия»
учебный план рег. №. 19/5080/1

САНКТ-ПЕТЕРБУРГ
2020

Введение

Всемирная сеть Интернет стала распространенной и очень доступной для миллиардов жителей земли. Эта сеть предоставляет доступ к различным сайтам, поисковым системам, магазинам и другим сервисам. Ежедневно, миллиарды людей используют Интернет для отправки электронной почты (e-mail), сообщений, чтения новостей, просмотра видео-лекций и других материалов. Курс “Телекоммуникации” посвящен протоколам сети Интернет, дает базовое понимание архитектуры и принципов ее работы.

В это методическое пособие входит тезисное описание лекций и практические задания, которые авторы предлагают студентам направления 09.03.04 «Программная инженерия» и 02.03.03 “Математическое обеспечение и администрирование информационных систем” в рамках курса «Телекоммуникации». Курс посвящен базовым протоколам и службам Интернет: NTP, SNTP, DNS, HTTP, POP3/IMAP, SMTP и другим.

Задания можно выполнять на операционных средах Windows, GNU/Linux и macOS на любом доступном языке программирования.

При составлении практических заданий и курса в целом авторы целенаправленно ориентируется на высокоуровневые протоколы, так как основы работы компьютерных сетей подробно разбираются в курсах 003657 «Компьютерные сети» и [003589] «Компьютерные сети и базы данных».

1. Теоретический материал

1. Автономные системы и регистраторы.

Знакомство с сетью Интернет начинается с разбора понятие «автономная система» (AS). Интернет — это не одна глобальная сеть, а объединение множества

небольших сетей. Автономная система — это система IP-сетей и маршрутизаторов, управляемых одним или несколькими операторами, имеющими единую политику маршрутизации с Интернетом. Более подробную информацию можно найти в RFC 1930. Важную роль в работе Интернет выполняют регистраторы:

IANA — Администрация адресного пространства Интернета. Занимается распределением номеров AS и IP-адресов в глобальном масштабе. Назначает RIR, подчиняется напрямую головной организвции ICANN. Такая организация единственная.

RIR — Региональная регистратура Интернета. Занимается выделением крупных блоков IP-адресов, регистрацией LIR и распределением AS. Россия находится в юрисдикции RIPE NCC, расположенной в Амстердаме.

LIR — Локальная Регистратура Интернета. Занимается поддержкой работы сети, распределением PI (о них — дальше) и номеров AS. Как правило, это провайдеры (ISP). Минимальный блок адресного пространства для LIR — 4096 IP-адресов.

PI — Provider Independent. Провайдеро-независимые IP адреса. Находятся в определенной AS, маршрут к ним зависит только от политики маршрутизации. Принадлежат конечному пользователю [компании или LIR], а не его вышестоящему провайдеру. Следственно, сохраняются при смене ISP\подключении дополнительного ISP.

Все данные об AS и выделенных IP-сетях находятся в открытой базе данных WHOIS. Сервис WHOIS позволяет любому получить регистрационные данные о владельцах доменных имён, IP-адресов и автономных систем.

2. Порты и сокеты.

Данный курс, как было сказано ранее, опирается на базовые понятия и требует понимания принципов работы протоколов TCP, UDP и IP. Напомним кратко основное.

Каждый сервис в сети Интернет имеет уникальный идентификатор, который состоит из IP адреса и порта. Например, 192.168.1.1 и порт 80. Часто можно встретить запись через двоеточие, например. 192.68.1.1:80. Порт — это целое неотрицательное число, записываемое в заголовках протоколов транспортного уровня модели OSI (TCP, UDP, SCTP, DCCP). Используется для определения процесса-получателя пакета в пределах одного хоста. Размер порт 2 байта.

Сокет — название программного интерфейса для обеспечения обмена данными между процессами. Процессы при таком обмене могут исполняться как на одном хосте, так и на различных хостах, связанных между собой сетью. Сокет — абстрактный объект, представляющий конечную точку соединения. Интерфейс сокетов впервые появился в BSD Unix. Программный интерфейс сокетов описан в стандарте POSIX.1 и в той или иной мере поддерживается всеми современными операционными системами.

Чтобы не заниматься угадываем, многие Интернет службы закрепили за собой один или несколько номер портов. Это не значит, что эти службы не могут работать на других портах, также, это не означает, что на закрепленных портах не может работать другая служба. Это скорее договоренность, запускать определенные службы на закрепленных за ними портами. Например:

- HTTP (Hypertext Transfer Protocol) использует TCP порт 80
- SMTP (Simple Mail Transfer Protocol) использует TCP порт 25
- SNTP (Simple Network Time Protocol) закрепил UDP порт 123
- DNS (Domain Name System) использует UDP и TCP порт 53

3. Время в интернет. Служба времени и протоколы синхронизации времени NTP/SNTP.

В компьютерных системах часто требуется иметь точно синхронизированное время (при этом не

важно, какая именно величина времени будет использоваться). Примером может быть распределённая система, где есть несколько компьютеров, принимающих запросы и передающих их на центральный сервер для обработки. Чтобы запросы выполнялись точно в том порядке, в котором они были приняты, принимающие компьютеры добавляют в них отметку о времени приёма. Если часы у разных компьютеров в такой системе выставлены по-разному, то может возникнуть ситуация, когда позднее пришедшему запросу будет выставлена меньшая отметка о времени, чем пришедшему ранее, и сервер обработает их в неправильном порядке.

Одним из базовых протоколов сети Интернет является NTP и его упрощенный вариант — SNTP. NTP решает задачу синхронизации часов узлов сети.

Официальный сайт со списком серверов точного времени можно найти по адресу: https://www.ntppool.org/ru/

NTP (Network Time Protocol) — сетевой протокол для синхронизации внутренних часов компьютера с использованием сетей с переменной латентностью. Протокол был разработан Дэвидом Л. Миллсом, профессором Делавэрского университета, в 1985 году. Версия на 2015 год — NTPv4. NTP, основанный на алгоритме Марзулло, использует для своей работы протокол UDP и учитывает время передачи. Система NTP чрезвычайно устойчива к изменениям латентности среды передачи. В версии 4 способен достигать точности 10 мс (1/100 с) при работе через Интернет, и до 0,2 мс (1/5000 с) и лучше внутри локальных сетей[2].

Наиболее широкое применение протокол NTP находит для синхронизации серверов точного времени. Для достижения максимальной точности предпочтительна постоянная работа программного обеспечения NTP в режиме системной службы. В

семействе операционных систем Microsoft Windows — это служба W32Time, Linux — демон Ntpd или chronyd.

SNTP (Simple Network Time Protocol) — протокол синхронизации времени по компьютерной сети. Является упрощённой реализацией протокола NTP. Используется во встраиваемых системах и устройствах, не требующих высокой точности, а также в пользовательских программах точного времени. SNTP протокол является частным случаем NTP протокола с некоторыми упрощениями. Таким образом SNTP клиент может обращаться к любому NTP серверу, как к серверу SNTP.

Серверы NTP и SNTP используют UDP протоколи 135 порт.

4. Влияние TCP на производительность прикладных протоколов. Nagle algorithm, медленный старт.

Из прошлого курса известно, что во время передачи данных TCP протокол контролирует скорость обмена данными и отслеживает переполнения канала (TCP congestion control). Эти алгоритмы приводят к тому, что в начале своей работы скорость передачи данных низкая и увеличивается в процессе передачи данных. Другими словами, первые 10 Кбайт данных передаются медленнее, чем следующие 10 Кбайт. Из этого следует, что скачивать 10 небольших файлов подряд, используя одно TCP соединение, будет выгодней, чем для открывать для каждого такого файла отдельное TCP соединение.

Алгоритм Нейгла (назван в честь Джона Нейгла), является средством повышения эффективности работы сетей TCP/IP, позволяющим уменьшить количество пакетов, которые должны быть отправлены по сети. Документ Нейгла, “Управление перегрузкой сетей IP/TCP” (RFC 896) описывает то, что он назвал «проблемой небольших пакетов», которая заключается в том, что приложение неоднократно посылает данные небольшими порциями, часто размером в 1 байт. Так как TCP-пакеты имеют 40 байт

заголовка (20 байт TCP, 20 байт IPv4), это приводит к тому, что передается пакет размером 41 байт, несущий в себе 1 байт полезной информации, то есть к огромным накладным расходам. Эта ситуация часто встречается в сессии Telnet/SSH, где большинство нажатий клавиш генерируют один байт данных, который немедленно передается. Кроме того, по медленным каналам связи многие такие пакеты могут находиться в пути в одно и то же время, что может привести к перегруженности сети.

Алгоритм Нейгла работает путём объединения нескольких небольших исходящих сообщений, а затем отправки их всех сразу. В частности, пока существует отправленный пакет, для которого отправитель ещё не получил никакого подтверждения о доставке, отправитель должен держать в буфере следующие данные для отправки, до тех пор, пока не наберётся достаточно данных на полный пакет, который можно отправить единожды.

Приложения, для которых важно актуальное время ответа, могут плохо работать с алгоритмом Нейгла. Такие приложения, как сетевые многопользовательские игры, предполагают, что какие-либо действия в игре отправляется сразу, тогда как алгоритм целенаправленно задерживает передачу, увеличивая эффективность использования полосы пропускания сети за счет задержки. По этой причине приложения с низкой пропускной способностью срочных передач обычно используют TCP_NODELAY, чтобы обойти задержки алгоритма Нейгла.

5. Служба DNS: Организация пространства имён. Top Level Domains.

Использование более простого и запоминающегося имени вместо числового адреса хоста относится к эпохе ARPANET. Стэнфордский исследовательский институт (теперь SRI International) поддерживал текстовый файл hosts.txt, который сопоставлял имена узлов

с числовыми адресами компьютеров в ARPANET. Этот файл по прежнему используется. В UNIX системах он находиться в /etc/hosts, а в Windows в папке system32\drivers\etc\hosts.

Адреса назначались вручную. Чтобы запросить имя хоста и адрес и добавить компьютер в главный файл, пользователи связывались с сетевым информационным центром (NIC) SRI, руководимым Элизабет Фейнлер, по телефону в рабочее время.

DNS (Domain Name System) — компьютерная распределённая система для получения информации о доменах. Чаще всего используется для получения IP-адреса по имени хоста (компьютера или устройства). Распределённая база данных DNS поддерживается с помощью иерархии DNS-серверов.

DNS была разработана Полом Мокапетрисом в 1983 году; оригинальное описание механизмов работы содержится в RFC 882 и RFC 883. В 1987 публикация RFC 1034 и RFC 1035 изменила спецификацию DNS и отменила RFC 882, RFC 883 и RFC 973 как устаревшие.

Основой DNS является представление об иерархической структуре имени и зонах. Каждый сервер, отвечающий за имя, может делегировать ответственность за дальнейшую часть домена другому серверу (с административной точки зрения — другой организации или человеку), что позволяет возложить ответственность за актуальность информации на серверы различных организаций (людей), отвечающих только за «свою» часть доменного имени.

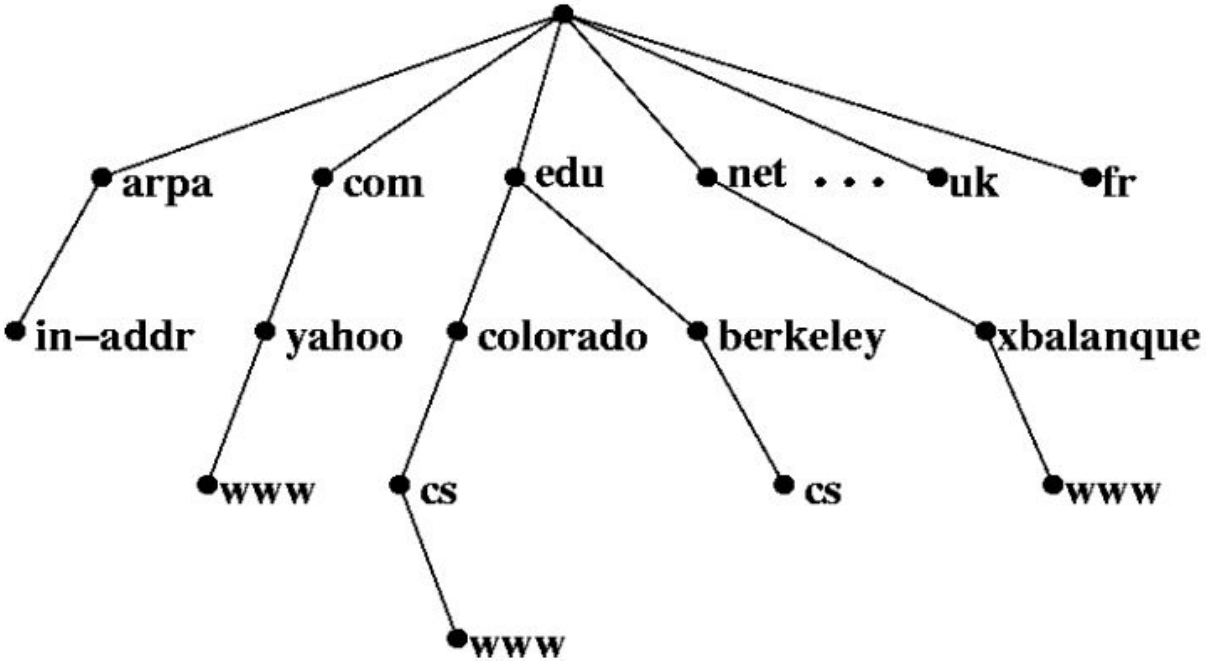

Рис. 1. Иерархия DNS.

Записи в DNS состоят из:

- названия (например, spbu)
- типа записи (A)
- адреса (195.70.219.101)

Типы записей:

- A (address record) или запись адреса связывает имя хоста с адресом протокола IPv4. Например, запрос A-записи на имя spbu.ru вернёт его IPv4-адрес — 195.70.219.101.
- Запись MX (mail exchange) или почтовый обменник указывает сервер(ы) обмена почтой для данного домена.
- Запись NS (name server) указывает на DNS-сервер для данного домена.
- Запись SOA (Start of Authority) или начальная запись зоны указывает, на каком сервере хранится эталонная информация о данном домене, содержит контактную информацию лица, ответственного за данную зону, тайминги (параметры времени) кеширования зонной информации и взаимодействия DNS-серверов.
- Запись CNAME (canonical name record) или каноническая запись имени (псевдоним)

используется для перенаправления на другое имя.
- и другие.

Типы запросов:
- Рекурсивный. Клиент посылает серверу DNS запрос, в котором требует дать окончательный ответ даже если DNS-серверу придется отправить запросы другим DNS-серверам.
- Прямой. Клиент посылает серверу DNS запрос, в котором требует дать наилучший ответ без обращений к другим DNS-серверам

Протокол DNS использует для работы TCP- или UDP-порт 53 для ответов на запросы. Традиционно запросы и ответы отправляются в виде одной UDP-датаграммы. TCP используется, когда размер данных ответа превышает 512 байт.

Общие домены верхнего уровня (ранее категории) первоначально состояли из gov, edu, com, mil, org и net. Авторитетный список существующих доменов верхнего уровня в корневой зоне публикуется на веб-сайте IANA по адресу https://www.iana.org/domains/root/db/

6. Протокол SMTP. Служба электронной почты.

SMTP (Simple Mail Transfer Protocol) — это широко используемый сетевой протокол, предназначенный для передачи электронной почты в сетях TCP/IP. SMTP впервые был описан в RFC 821 (1982 год); последнее обновление в RFC 5321 (2008) включает масштабируемое расширение — ESMTP (Extended SMTP). В настоящее время под «протоколом SMTP» как правило подразумевают и его расширения. Протокол SMTP предназначен для передачи исходящей почты с использованием порта TCP 25 (без шифрования), порт 587 порт (для TLS) и порт 465 (для SSL).

Электронные почтовые серверы и другие агенты пересылки сообщений используют SMTP для отправки и получения почтовых сообщений. Для получения сообщений

клиентские приложения обычно используют либо POP (Post Office Protocol), либо IMAP (Internet Message Access Protocol).

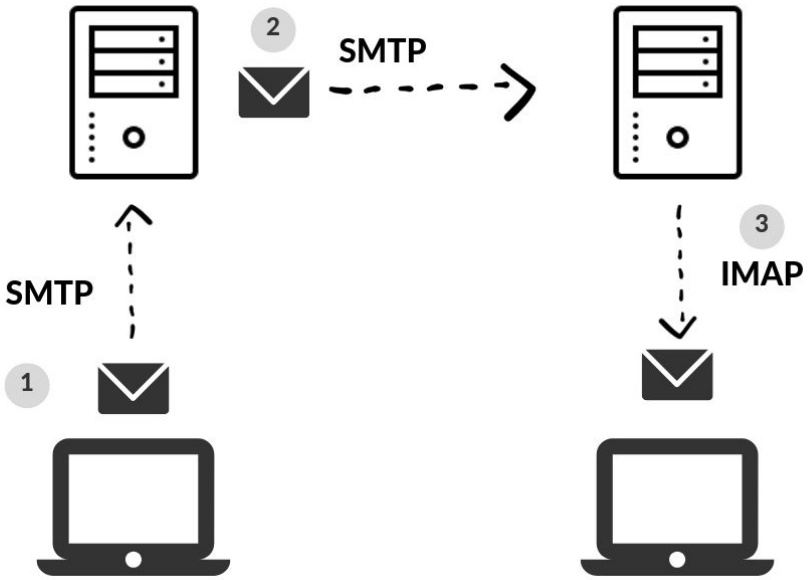

Рис.2. Схема работы электронной почты.

Электронное письмо состоит из следующих частей:

- Заголовков SMTP-протокола (MAIL FROM, RCPT TO)
- Заголовков письма.(отправитель, обратный адрес, адресат, отметки о спам-проверках, тема письма, MIME-тип, кодировка и т.п.)
- Тела письма.(отделяется от заголовков пустой строкой, обычный ASCII текст либо соответствующий mime типу набор данных).

Подключение к SMTP серверу:

- nc smtp.mail.ru 25
- openssl s_client -connect smtp.mail.ru:465 -quiet (для SSL)

Для преобразования логина и пароля можно воспользоваться вот такой командой:

- echo -ne "\0name@damain.ru\0password" | base64

Команды SMTP:

- EHLO
- MAIL FROM
- RCPT TO
- DATA
- AUTH

Заголовки письма:

- From: "Ilya Zelenchuk" <ilya@hackerdom.ru>
- To: ilya@bigxp.ru
- Subject: Заголовок письма
- Reply-to: ilya@hackerdom.ru
- CC, BCC и так далее

От тела письма заголовки отделяются пустой строкой. Пример взаимодействия с SMTP сервером:

> 220 smtp63.i.mail.ru ESMTP ready (Looking for Mail for your domain? Visit https://biz.mail.ru)
< EHLO inbox.ru
> 250-smtp63.i.mail.ru
> 250-SIZE 73400320
> 250-8BITMIME
> 250-PIPELINING
> 250 AUTH PLAIN LOGIN XOAUTH2
< AUTH PLAIN BlaBlaBla
> 235 Authentication succeeded
< MAIL FROM: dnetc@inbox.ru
> 250 OK
< RCPT TO: ilya@hackerdom.ru
> 250 Accepted
< DATA
> 354 Enter message, ending with "." on a line by itself
< From: "Ilya Zelenchuk" <dnetc@inbox.ru>
< To: ilya@hackerdom.ru
< Subject: Привет Илья!
< Reply-to: ant@lanit-tercom.ru
<
< Привет)
< .
> 250 OK id=1jNwwu-0006Pz-Tv
< quit
> 221 smtp63.i.mail.ru closing connection

Пример программы на Python для отправки почты:

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import smtplib, ssl
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
```

```
from email.header import Header
import getpass

sender_email = "dnetc@inbox.ru"
receiver_email = "ilya@hackerdom.ru"

password = getpass.getpass(prompt='Password entered:')

message = MIMEMultipart("alternative")
message["Subject"] = "Привет, Илья!"
message["From"] = Header(u'"Ilya Zelenchuk" <dnetc@inbox.ru>', 'utf-8')
message["To"] = receiver_email

# Create the plain-text and HTML version of your message
text = """\
Привет, Илья!
Это письмо я отправил сам себе используя скрипт на Python.
"""

# Turn these into plain/html MIMEText objects
part2 = MIMEText(text, "html")

# Add HTML/plain-text parts to MIMEMultipart message
# The email client will try to render the last part first
message.attach(part2)

# Create secure connection with server and send email
server = smtplib.SMTP_SSL("smtp.mail.ru", 465)
server.login(sender_email, password)
server.sendmail(sender_email, receiver_email, message.as_string())
```

7. Служба электронной почты. Протокол POP3 и IMAP.

POP и IMAP (Internet Message Access Protocol) — наиболее распространённые интернет-протоколы для извлечения почты. Практически все современные клиенты и серверы электронной почты поддерживают оба стандарта. Протокол POP был разработан в нескольких версиях, нынешним стандартом является третья версия (POP3). Большинство поставщиков услуг электронной почты (такие как Hotmail, Gmail и Yahoo! Mail) также

поддерживает IMAP и POP3. Предыдущие версии протокола (POP, POP2) устарели.

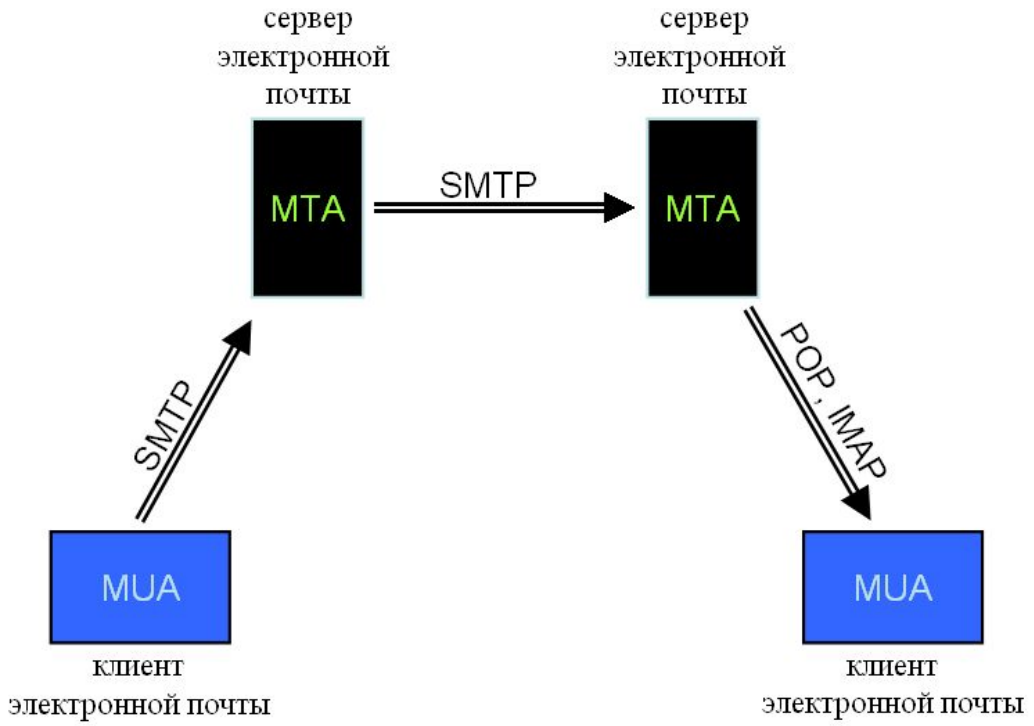

Рис. 3. Схема работы POP3 и IMAP протоколов.

POP3 (Post Office Protocol Version 3) — стандартный интернет-протокол прикладного уровня, используемый клиентами электронной почты для получения почты с удалённого сервера по TCP-соединению. Протокол описан в RFC 1939 (1996 год, J. Myers, M. Rose). Это текстовый протокол, что позволяет его легко отлаживать и работать с ним прямо из консоли.

Протокол работает по схеме загрузить и удалить. Т.е. все письма хранятся локально на клиенте, а после загрузки письмо удаляется с сервера. Такой подход позволяет получать доступ к письмам даже без доступа к серверу. С другой стороны, при такой работе только один клиент может работать с почтовым сервисом. Если у вас почтовый клиент стоит дома, на работе и на телефоне, то у вас будет разный набор писем во всех этих клиентах.

В качестве транспорта POP3 использует TCP протокол и порты 110 (без шифрования) и 995

(SSL/TLS). Протокол использует следующий набор команд:

- USER test@ya.ru - указать полное имя пользователя
- PASS qwerty - указать пароль
- QUIT - закончить сессию и выйти
- STAT - общее количество и размер писем
- LIST 123 - список сообщений и их размер
- RETR 123 - получить указанное сообщение
- DELE 123 - пометить указанное сообщение на удаление
- NOOP - пустая команда, используется для поддержания активного соединения.
- RSET - сбросить пометки.

Во время получения и обработки команд POP3 сервер дает следующие ответы:

- + OK
- - ERR

Альтернативой POP3 является протокол IMAP, он предоставляет пользователю широкие возможности для работы с почтовыми ящиками, находящимися на почтовом сервере. Почтовая программа, использующая этот протокол, получает доступ к хранилищу корреспонденции на сервере так, как будто эта корреспонденция расположена на компьютере получателя. Электронными письмами можно манипулировать с компьютера пользователя (клиента) без постоянной пересылки с сервера и обратно полного содержания писем.

Протокол IMAP описан в RFC 3501 (4-я версия, 2003 год, M. Crispin). Как и POP3, IMAP текстовый протокол, но в отличие от POP3, хранит письма на сервере. Это делает протокол более сложным, но позволяет одновременно нескольким клиентам успешно работать в письмами.

В качестве транспорта IMAP использует TCP протокол и порты 143 (без шифрования) и 993 (SSL/TLS). Протокол использует следующий набор команд:

- . login user@mail.ru password
- . list "" "*"
- . status INBOX (messages)

- . select inbox
- и д р у г и е.

8. П р о т о к о л HTTP. Ф о р м а т п а к е т о в. М е т о д ы з а п р о с а и к о д ы о т в е т о в. З а п р о с к с е р в е р у и о т в е т к л и е н т у - ч т о в н и х.

HTTP (HyperText Transfer Protocol) - п р о т о к о л п р и к л а д н о г о у р о в н я п е р е д а ч и д а н н ы х и з н а ч а л ь н о - в в и д е г и п е р т е к с т о в ы х д о к у м е н т о в в ф о р м а т е «HTML», в н а с т о я щ и й м о м е н т и с п о л ь з у е т с я д л я п е р е д а ч и п р о и з в о л ь н ы х д а н н ы х. О с н о в о й HTTP я в л я е т с я т е х н о л о г и я «к л и е н т - с е р в е р».

К а ж д о е HTTP-с о о б щ е н и е с о с т о и т и з т р ё х ч а с т е й, к о т о р ы е п е р е д а ю т с я в у к а з а н н о м п о р я д к е:

- С т а р т о в а я с т р о к а (Starting line) — о п р е д е л я е т т и п с о о б щ е н и я;
- З а г о л о в к и (Headers) — х а р а к т е р и з у ю т т е л о с о о б щ е н и я, п а р а м е т р ы п е р е д а ч и и п р о ч и е с в е д е н и я;
- Т е л о с о о б щ е н и я (Message Body) — н е п о с р е д с т в е н н о д а н н ы е с о о б щ е н и я. О б я з а т е л ь н о д о л ж н о о т д е л я т ь с я о т з а г о л о в к о в п у с т о й с т р о к о й.

Т е л о с о о б щ е н и я м о ж е т о т с у т с т в о в а т ь, н о с т а р т о в а я с т р о к а и з а г о л о в о к я в л я ю т с я о б я з а т е л ь н ы м и э л е м е н т а м и. И с к л ю ч е н и е м я в л я е т с я в е р с и я 0.9 п р о т о к о л а, у к о т о р о й с о о б щ е н и е з а п р о с а с о д е р ж и т т о л ь к о с т а р т о в у ю с т р о к у, а с о о б щ е н и я о т в е т а — т о л ь к о т е л о с о о б щ е н и я. Д л я в е р с и и п р о т о к о л а 1.1 с о о б щ е н и е з а п р о с а о б я з а т е л ь н о д о л ж н о с о д е р ж а т ь з а г о л о в о к Host. Н а п р и м е р, ч т о б ы п о л у ч и т ь к о р н е в у ю с т р а н и ц у с с а й т а https://spbu.ru н у ж н о и з к о н с о л и в ы п о л н и т ь:

- openssl s_client -connect spbu.ru:443 -quiet (у с т а н о в к а з а щ и щ е н н о г о с о е д и н е н и я)
- GET / HTTP/1.1 (з а п р о с к о р н е в о г о р е с у р с а)
- Host: spbu.ru (у к а з а н и е о б я з а т е л ь н о г о п а р а м е т р а Host д л я в е р с и и п р о т о к о л а HTTP/1.1)
- Д в е п у с т ы х с т р о к и (д в а р а з а Enter).

М е т о д ы в HTTP:

- OPTIONS - Используется для определения возможностей веб-сервера или параметров соединения для конкретного ресурса. В ответ серверу следует включить заголовок Allow со списком поддерживаемых методов. Также в заголовке ответа может включаться информация о поддерживаемых расширениях. Предполагается, что запрос клиента может содержать тело сообщения для указания интересующих его сведений. Формат тела и порядок работы с ним в настоящий момент не определён; сервер пока должен его игнорировать. Аналогичная ситуация и с телом в ответе сервера.
- GET - Используется для запроса содержимого указанного ресурса. С помощью метода GET можно также начать какой-либо процесс. В этом случае в тело ответного сообщения следует включить информацию о ходе выполнения процесса.
- HEAD - Аналогичен методу GET, за исключением того, что в ответе сервера отсутствует тело. Запрос HEAD обычно применяется для извлечения метаданных, проверки наличия ресурса (валидация URL) и чтобы узнать, не изменился ли он с момента последнего обращения. Заголовки ответа могут кэшироваться. При несовпадении метаданных ресурса с соответствующей информацией в кэше — копия ресурса помечается как устаревшая.
- POST - Применяется для передачи пользовательских данных заданному ресурсу. Например, в блогах посетители обычно могут вводить свои комментарии к записям в HTML-форму, после чего они передаются серверу методом POST и он помещает их на страницу. При этом передаваемые данные (в примере с блогами — текст комментария) включаются в тело запроса. Аналогично с помощью метода POST обычно загружаются файлы на сервер.

- PUT - Применяется для загрузки содержимого запроса на указанный в запросе URI. Если по заданному URI не существует ресурс, то сервер создаёт его и возвращает статус 201 (Created). Если же был изменён ресурс, то сервер возвращает 200 (Ok) или 204 (No Content). Сервер не должен игнорировать некорректные заголовки Content-*, передаваемые клиентом вместе с сообщением. Если какой-то из этих заголовков не может быть распознан или не допустим при текущих условиях, то необходимо вернуть код ошибки 501 (Not Implemented).
- CONNECT - Преобразует соединение запроса в прозрачный TCP/IP-туннель, обычно чтобы содействовать установлению защищённого SSL-соединения через нешифрованный прокси.
- и другие.

Код состояния является частью первой строки ответа сервера. Он представляет собой целое число из трёх цифр[5]. Первая цифра указывает на класс состояния. За кодом ответа обычно следует отделённая пробелом поясняющая фраза на английском языке, которая разъясняет человеку причину именно такого ответа:

- 1xx - Информирование о процессе передачи.
- 2xx - Информирование о случаях успешного принятия и обработки запроса клиента. В зависимости от статуса, сервер может ещё передать заголовки и тело сообщения.
- 3xx - Сообщает клиенту, что для успешного выполнения операции необходимо сделать другой запрос (как правило по другому URI). Из данного класса пять кодов 301, 302, 303, 305 и 307 относятся непосредственно к перенаправлениям (редирект).
- 4xx - Указание ошибок со стороны клиента. При использовании всех методов, кроме HEAD, сервер должен вернуть в теле сообщения гипертекстовое пояснение для пользователя.

- 5xx - Информирование о случаях неудачного выполнения операции по вине сервера. Для всех ситуаций, кроме использования метода HEAD, сервер должен включать в тело сообщения объяснение, которое клиент отобразит пользователю.

2. Задачи

В течение семестра студентам необходимо выполнить задачи с 1 по 11. Задачи 9, 10 и 11 - это домашние задачи.

1. Узнать номер своей автономной системы, используя сервисы BGP Looking Glass (https://stat.ripe.net/widget/looking-glass) и http://myip.ru/.
   i. Используя сервис http://myip.ru/ узнать IP адрес, с которого пользователь выходит в сеть Интернет.

Рис. 1. Сервис myip.ru (http://myip.ru/).

   ii. Используя сервис BGP Looking Glass узнать номер автономной системы и кому она принадлежит.
2. Посмотреть список активных TCP/UDP портов на локальном хосте используя утилиту netstat. Для этого (на Linux) выполним:
   i. открыть консоль
   ii. выполнить команду netstat -antlp

```
~$ netstat -anltp
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:2222            0.0.0.0:*               LISTEN      23/sshd
tcp        0      0 0.0.0.0:6000            0.0.0.0:*               LISTEN      8/node
tcp        0      0 192.168.72.232:6001     0.0.0.0:*               LISTEN      8/node
tcp        0      0 192.168.72.232:6001     192.168.123.228:60772   ESTABLISHED 8/node
tcp        0      0 192.168.72.232:6000     192.168.23.59:43420     ESTABLISHED 8/node
tcp6       0      0 :::43789                :::*                    LISTEN      8/node
tcp6       0      0 :::2222                 :::*                    LISTEN      23/sshd
~$
```

Рис.2. Пример вывода команды netstat с флагами antlp

3. Узнать адреса корневых DNS серверов используя ресурс с https://www.iana.org/.
4. Используя утилиту nslookup от авторитетного источника самостоятельно узнать IP адреса yandex.ru для следующих типов записи: NS, MX и A.
5. Используя утилиту nslookup узнать IP адреса NS, MX и A mail.ru без использования рекурсивного запроса.
6. Используя утилиту telnet, netcat (nc) или openssl получить ресурс с HTTP сервера используя HTTP протокол версии 1.1:
   i. В консоли подключаемся к HTTP серверу: openssl s_client -connect yandex.ru:443 -quiet
   ii. Вводим запрос: GET / HTTP/1.1\r\nHost: yandex.ru\r\n\r\n
   iii. Получаем ответ.

```
ScrumBook:Zoom ilya2$ openssl s_client -connect yandex.ru:443 -quiet
depth=2 C = PL, O = Unizeto Technologies S.A., OU = Certum Certification Authority, CN = Certum Trusted Network CA
verify return:1
depth=1 C = RU, O = Yandex LLC, OU = Yandex Certification Authority, CN = Yandex CA
verify return:1
depth=0 CN = yandex.ru, O = Yandex LLC, OU = ITO, L = Moscow, ST = Russia, C = RU
verify return:1
GET / HTTP/1.1
Host: yandex.ru

HTTP/1.1 200 Ok
Accept-CH: Viewport-Width, DPR, Device-Memory, RTT, Downlink, ECT
Accept-CH-Lifetime: 31536000
Cache-Control: no-cache,no-store,max-age=0,must-revalidate
Content-Length: 164625
Content-Security-Policy: img-src https://leonardo.edadeal.io https://*.strm.yandex.net https://favicon.yandex.net https://an.yandex.ru https://strm.yandex.ru htt
ps://mc.admetrica.ru https://*.verify.yandex.ru https://www.maximonline.ru https://www.kinopoisk.ru data: https://yandex.ru https://resize.yandex.net https://aut
o.ru https://mc.yandex.ru https://thequestion.ru https://avatars.mds.yandex.net https://yastatic.net 'self';object-src https://avatars.mds.yandex.net;script-src
'unsafe-inline' https://mc.yandex.ru https://an.yandex.ru https://yastatic.net https://yandex.ru 'self';report-uri https://csp.yandex.net/csp?project=morda&from=
morda.big.ru&showid=1587641698.58116.85208.84864&h=stable-morda-man-yp-340&csp=new&date=20200423&yandexuid=9802933921587641698;frame-src https://yandex.ru https:
//yastatic.net 'self' https://st.yandexadexchange.net https://yandexadexchange.net https://mc.yandex.ru;style-src 'unsafe-inline' https://yastatic.net;media-src
https://*.cdn.ngenix.net blob: https://*.strm.yandex.net;connect-src https://mobile.yandex.net https://www.kinopoisk.ru https://yandex.ru https://portal-xiva.yan
dex.net https://zen.yandex.ru wss://portal-xiva.yandex.net https://www.maximonline.ru https://yastatic.net 'self' https://frontend.vh.yandex.ru https://auto.ru h
ttps://mc.yandex.ru https://thequestion.ru https://yastat.net https://*.strm.yandex.net https://*.cdn.ngenix.net https://mc.admetrica.ru https://games.yandex.ru
https://api.market.yandex.ru https://an.yandex.ru https://strm.yandex.ru;default-src https://yastatic.net https://yastat.net;font-src https://an.yandex.ru data:
https://yastatic.net
Content-Type: text/html; charset=UTF-8
Date: Thu, 23 Apr 2020 11:34:58 GMT
Expires: Thu, 23 Apr 2020 11:34:59 GMT
Last-Modified: Thu, 23 Apr 2020 11:34:59 GMT
P3P: policyref="/w3c/p3p.xml", CP="NON DSP ADM DEV PSD IVDo OUR IND STP PHY PRE NAV UNI"
Set-Cookie: yp=1590233699.ygu.1; Expires=Sun, 21-Apr-2030 11:34:58 GMT; Domain=.yandex.ru; Path=/
Set-Cookie: mda=0; Expires=Fri, 21-Aug-2020 11:34:58 GMT; Domain=.yandex.ru; Path=/
Set-Cookie: yandex_gid=2; Expires=Sat, 23-May-2020 11:34:58 GMT; Domain=.yandex.ru; Path=/
Set-Cookie: yandexuid=9802933921587641698; Expires=Sun, 21-Apr-2030 11:34:58 GMT; Domain=.yandex.ru; Path=/
Set-Cookie: i=3vtfTddpqERcr2MfkNzh42N86NzGmvwUSNJP86DIMNfDb+uLskkpvzjTGbbEkFEICNiY4PQ8mj/hkGeNtafSXxXTRt4=; Expires=Sun, 21-Apr-2030 11:34:58 GMT; Domain=.yandex
.ru; Path=/; Secure; HttpOnly
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
X-Yandex-Sdch-Disable: 1

<!DOCTYPE html><html class="i-ua_js_no i-ua_css_standart i-ua_browser_ i-ua_browser_desktop document_sticky-extra-logo_yes i-ua_platform_other" lang="ru"><head x
mlns:og="http://ogp.me/ns#"><meta http-equiv=Content-Type content="text/html;charset=UTF-8"><meta http-equiv="X-UA-Compatible" content="IE=edge"><title>Яндекс</t
itle><link rel="shortcut icon" href="//yastatic.net/iconostasis/_/81FaTHLDzmsEZz-5XaQg9iTWZGE.png"><link rel="apple-touch-icon" href="//yastatic.net/iconostasis/
_/5mdPq4V7ghRgzBvMkCaTzd2fjYg.png" sizes="76x76"><link rel="apple-touch-icon" href="//yastatic.net/iconostasis/_/s-hGoCQMUosTziuARBks08IUxmc.png" sizes="120x120"
```

Рис.3. Пример запроса ресурса с использованием HTTP протокола версии 1.1.

7. Отправить письмо через SMTP используя консоль (nc или openssl).

8. Получить письмо со своего ящика через nc или openssl
9. Реализовать SNTP клиент и узнать точное время с сервера, чей стратум не ниже 3.
10. Написать программу (скрипт) для отправки электронного письма HTML формата через SMTP сервер.
11. Реализовать HTTP клиент для получения всех “<a href=” ссылок с заданного адреса.

Приложение 1. Примерные списки вопросов к теоретическому зачету

1. Понятие «автономная система». Регистраторы. Определение автономной системы, их виды. Кто такие RIR (перечислить) и LIR (что нужно, чтобы ими стать). Содержимое базы данных WHOIS.
2. Понятие «открытая система». RFC. Определение открытой системы (интерфейсы, стандарты), примеры. Виды (STD/BCP/FYI/...), статусы RFC, процесс утверждения.
3. Понятия «порт» и «сокет». Примеры портов.
4. Служба времени NTP/SNTP. GMT, Атомное время, UTC. Организация сети серверов NTP, понятие «стратум». Алгоритм определения точного времени через интернет с использованием SNTP.
5. Влияние TCP на производительность прикладных протоколов. Nagle algorithm, медленный старт. Интерактивная работа и передача больших файлов. Работа алгоритмов, когда они применяются.
6. Служба DNS: Организация пространства имён. Международные имена (IDNA). Общая организация пространства имён DNS. Top Level Domains, в т.ч. IN-ADDR.ARPA. Алгоритм Punycode.
7. Служба DNS: Записи типа NS, понятия «зона» и «домен». Регистрация DNS-имён. Взгляд с боку. Primary/Secondary DNS-серверы. Регистрация, делегирование.
8. Разрешение DNS-имён в IP-адреса. Взгляд со стороны клиента. Последовательность действий (кеш/hosts/dsn-server).

9. Служба электронной почты. Протокол SMTP. Идентификация почтового ящика. Алгоритм доставки письма. От кнопки «Send» в MUA до папки mailroot\Drop.
10. Служба электронной почты. Протокол POP3. Идентификация почтового ящика. Как почта хранится на сервере. Что нужно для получения (настройки и команды)
11. Протокол HTTP. Формат пакетов. Методы запроса и коды ответов. Запрос к серверу и ответ клиенту - что в них.
12. Архитектура и функции веб- и прокси-сервера. Обработка HTTP-запроса. Алгоритм работы сервера при обработке клиентского запроса.
13. Безопасность в HTTP. Аутентификация пользователя. Контроль доступа на сервере. Коды 401/403. Заголовки Authorization и WWW-Authenticate, понятие realm. Принципы работы HTTPS, алгоритм Диффи-Хеллмана. Анонимный пользователь. Переход от виртуальной к реальной ФС сервера.

Список литературы

1. Олифер, В. Г. Основы сетей передачи данных : учебное пособие / В. Г. Олифер, Н. А. Олифер. — 2-е изд. — Москва : ИНТУИТ, 2016. — 219 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/100346 (дата обращения: 28.04.2020).
2. Андрей Созыкин, Компьютерные сети - https://www.asozykin.ru/courses/networks_online
3. https://www.ietf.org/rfc/
4. http://cs.usu.edu.ru/study/ntpsntp/