

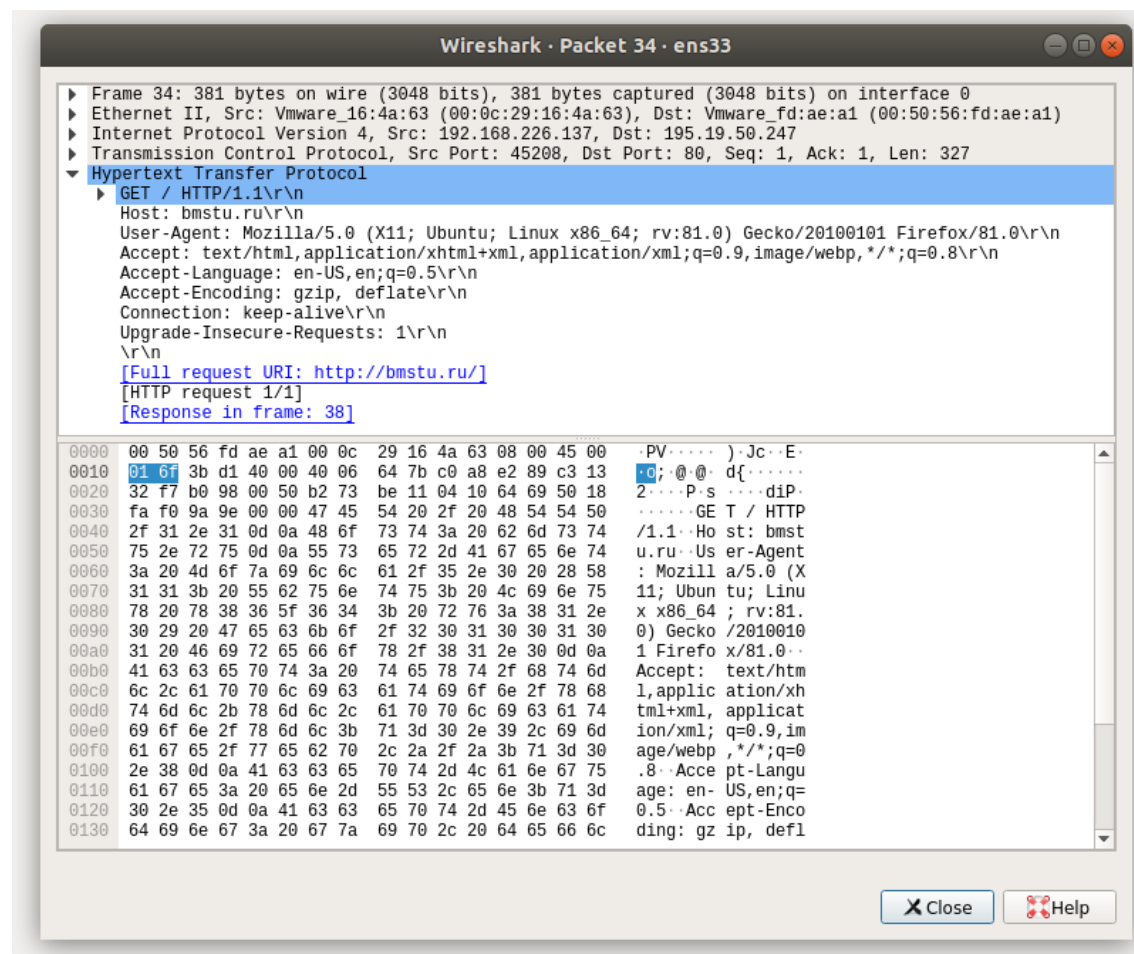
# Лекция VI. Протоколы прикладного уровня, часть II

Рогозин Н.О., каф. ИУ-7

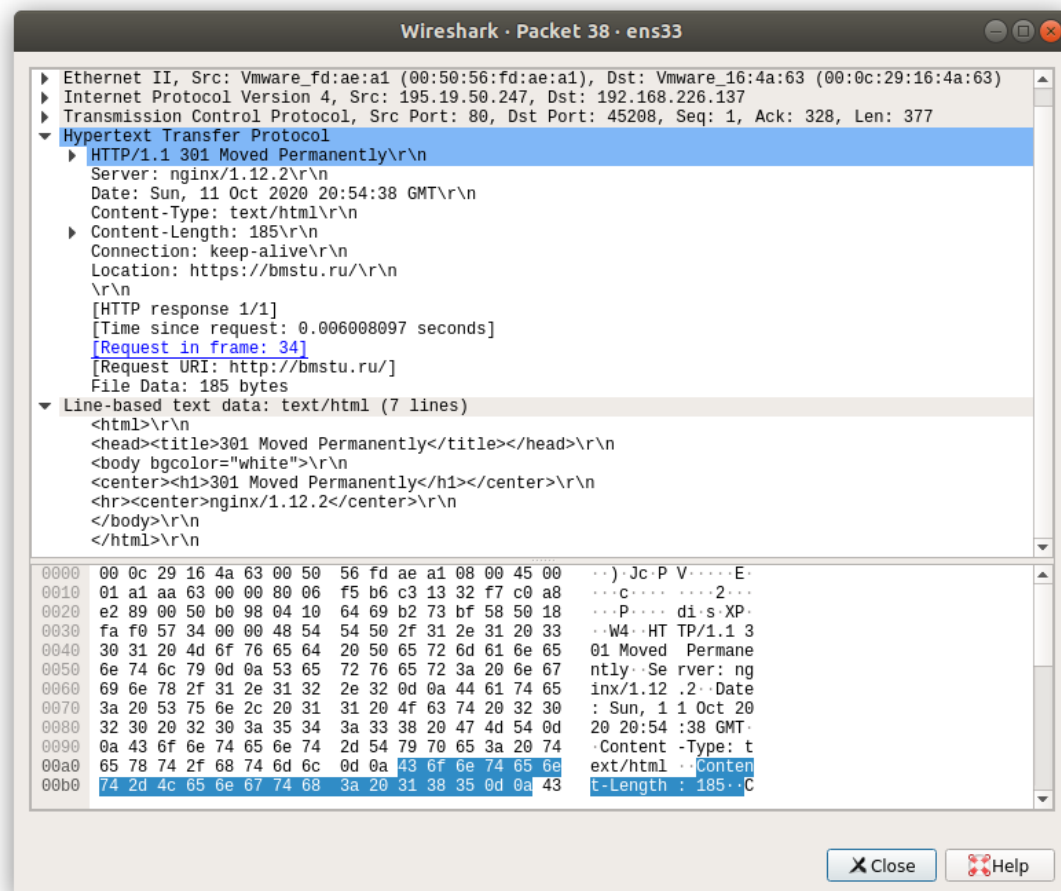
# HTTPS (RFC 2818)

- По умолчанию HTTPS URL использует 443 TCP-порт (для незащищённого HTTP — 80).
- Чтобы подготовить веб-сервер для обработки https-соединений, администратор должен получить и установить в систему сертификат открытого и закрытого ключа для этого веб-сервера.
- В TLS используется как асимметричная схема шифрования (для выработки общего секретного ключа), так и симметричная (для обмена данными, зашифрованными общим ключом).

# Первоначальный http-запрос



# http-ответ



# Обмен сообщениями TLS

The image shows a Wireshark packet capture window titled "ens33". The filter bar at the top displays the filter "ip.dst == 195.19.50.247 || ip.src == 195.19.50.247". The packet list shows 125 packets, with packet 38 selected. The packet details pane shows the structure of the selected packet, and the packet bytes pane shows the raw data in hexadecimal and ASCII.

No.	Time	Source	Destination	Protocol	Length	Info
30	11.756531879	192.168.226.137	195.19.50.247	TCP	74	45208 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2504779527 TSecr=0 WS=128
31	11.756574871	192.168.226.137	195.19.50.247	TCP	74	45210 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2504779527 TSecr=0 WS=128
32	11.761754857	195.19.50.247	192.168.226.137	TCP	60	80 → 45208 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
33	11.761771685	192.168.226.137	195.19.50.247	TCP	54	45208 → 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
34	11.762143400	192.168.226.137	195.19.50.247	HTTP	381	GET / HTTP/1.1
35	11.762250540	195.19.50.247	192.168.226.137	TCP	60	80 → 45208 [ACK] Seq=1 Ack=328 Win=64240 Len=0
36	11.762468452	195.19.50.247	192.168.226.137	TCP	60	80 → 45210 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
37	11.762477385	192.168.226.137	195.19.50.247	TCP	54	45210 → 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
38	11.768151497	195.19.50.247	192.168.226.137	HTTP	431	HTTP/1.1 301 Moved Permanently (text/html)
39	11.768164913	192.168.226.137	195.19.50.247	TCP	54	45208 → 80 [ACK] Seq=328 Ack=378 Win=63863 Len=0
40	11.774084982	192.168.226.137	195.19.50.247	TCP	74	53682 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2504779545 TSecr=0 WS=128
41	11.779079102	195.19.50.247	192.168.226.137	TCP	60	443 → 53682 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
42	11.779098707	192.168.226.137	195.19.50.247	TCP	54	53682 → 443 [ACK] Seq=1 Ack=1 Win=64240 Len=0
43	11.780246949	192.168.226.137	195.19.50.247	TLSv1.2	571	Client Hello
44	11.780382225	195.19.50.247	192.168.226.137	TCP	60	443 → 53682 [ACK] Seq=1 Ack=518 Win=64240 Len=0
45	11.787879673	195.19.50.247	192.168.226.137	TLSv1.2	3903	Server Hello, Certificate, Server Key Exchange, Server Hello Done
46	11.787896026	192.168.226.137	195.19.50.247	TCP	54	53682 → 443 [ACK] Seq=518 Ack=3850 Win=61320 Len=0
47	11.789869069	192.168.226.137	195.19.50.247	TLSv1.2	180	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
48	11.790009768	195.19.50.247	192.168.226.137	TCP	60	443 → 53682 [ACK] Seq=3850 Ack=644 Win=64240 Len=0
53	11.796056167	195.19.50.247	192.168.226.137	TLSv1.2	312	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
54	11.796076746	192.168.226.137	195.19.50.247	TCP	54	53682 → 443 [ACK] Seq=644 Ack=4108 Win=62780 Len=0
66	11.861080848	192.168.226.137	195.19.50.247	TLSv1.2	414	Application Data
67	11.861221597	195.19.50.247	192.168.226.137	TCP	60	443 → 53682 [ACK] Seq=4108 Ack=1004 Win=64240 Len=0
68	11.875876511	195.19.50.247	192.168.226.137	TLSv1.2	14284	Application Data
69	11.875897074	192.168.226.137	195.19.50.247	TCP	54	53682 → 443 [ACK] Seq=1004 Ack=18338 Win=55480 Len=0
119	12.155774014	192.168.226.137	195.19.50.247	TLSv1.2	401	Application Data
120	12.155914924	195.19.50.247	192.168.226.137	TCP	60	443 → 53682 [ACK] Seq=18338 Ack=1351 Win=64240 Len=0
121	12.156077766	192.168.226.137	195.19.50.247	TCP	74	53690 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2504779927 TSecr=0 WS=128
122	12.156259786	192.168.226.137	195.19.50.247	TCP	74	53692 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2504779927 TSecr=0 WS=128
123	12.156445971	192.168.226.137	195.19.50.247	TCP	74	53694 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2504779927 TSecr=0 WS=128
124	12.156640062	192.168.226.137	195.19.50.247	TCP	74	53696 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2504779927 TSecr=0 WS=128
125	12.156819925	192.168.226.137	195.19.50.247	TCP	74	53698 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2504779928 TSecr=0 WS=128

Frame 38: 431 bytes on wire (3448 bits), 431 bytes captured (3448 bits) on interface 0  
Ethernet II, Src: Vmware\_fd:ae:a1 (00:50:56:fd:ae:a1), Dst: Vmware\_16:4a:63 (00:0c:29:16:4a:63)  
Internet Protocol Version 4, Src: 195.19.50.247, Dst: 192.168.226.137  
Transmission Control Protocol, Src Port: 80, Dst Port: 45208, Seq: 1, Ack: 328, Len: 377  
Hypertext Transfer Protocol

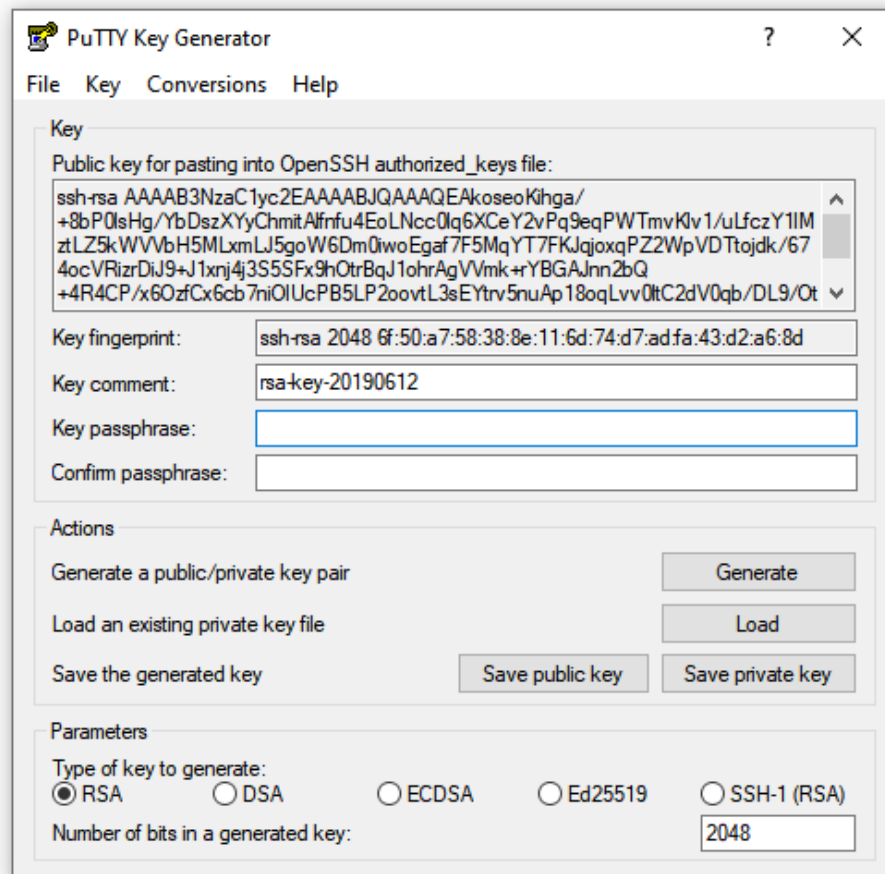
0000 00 0c 29 16 4a 63 00 50 56 fd ae a1 08 00 45 00 ..)JcP V.....E  
0010 01 a1 aa 63 00 00 80 06 f5 b6 c3 13 32 f7 c0 a8 ...c.....2..  
0020 e2 89 00 50 b0 98 04 10 64 69 b2 73 bf 58 50 18 ...P....di.s.XP.  
0030 fa f0 57 34 00 00 48 54 54 50 2f 31 2e 31 20 33 ..W4..HT TP/1.1 3  
0040 30 31 20 4d 6f 76 65 64 20 50 65 72 6d 61 6e 65 01 Moved Perman  
0050 6e 74 6c 79 0d 0a 53 65 72 76 65 72 3a 20 6e 67 ntly..Se rver: ng  
0060 69 6e 78 2f 31 2e 31 32 2e 32 0d 0a 44 61 74 65 inx/1.12 .2..Date  
0070 3a 20 53 75 6e 2c 20 31 31 20 4f 63 74 20 32 30 : Sun, 1 1 Oct 20

wireshark\_ens33\_20201011135425\_u0a2Md.pcapng Packets: 2883 · Displayed: 1840 (63.8%) · Dropped: 0 (0.0%) Profile: Default

# SSH

- Как и telnet, передает набираемые на терминале пользователя символы на удаленный узел без интерпретации их содержания.
- Предусматривает меры по защите передаваемых аутентификационных и пользовательских данных.
- Поддерживает симметричное, асимметричное шифрование и хеширование (обсуждается в лекции, посвященной безопасности сетевых соединений)

# Генерация ключа в PuttyGen



The screenshot shows the PuTTY Key Generator window. The 'Key' section displays a public key for pasting into an OpenSSH authorized\_keys file. The key fingerprint is shown as 'ssh-rsa 2048 6f:50:a7:58:38:8e:11:6d:74:d7:ad:fa:43:d2:a6:8d'. The key comment is 'rsa-key-20190612'. The 'Actions' section includes buttons for 'Generate', 'Load', 'Save public key', and 'Save private key'. The 'Parameters' section shows the 'Type of key to generate' set to 'RSA' and the 'Number of bits in a generated key' set to '2048'.

PuTTY Key Generator

File Key Conversions Help

Key

Public key for pasting into OpenSSH authorized\_keys file:

```
ssh-rsa AAAAB3NzaC1yc2EAAAABJQAAAQEAkoseoKihga/  
+8bP0lsHg/YbDszXYyChmitAlfnfu4EoLNcc0lq6XCeY2vPq9eqPWtmvKlv1/uLfczY1IM  
ztLZ5kVVVbH5MLxmLJ5goW6Dm0iwoEgaf7F5MqYT7FKJqioxqPZ2WpVDTtojdk/67  
4ocVRizrDiJ9+J1xnj4j3S5SFx9hOtrBqJ1ohrAgVvmk+rYBGAJnn2bQ  
+4R4CP/x6OzfCx6cb7niOIUcPB5LP2oovtL3sEYtrv5nuAp18oqLvv0ltC2dV0qb/DL9/Ot
```

Key fingerprint: ssh-rsa 2048 6f:50:a7:58:38:8e:11:6d:74:d7:ad:fa:43:d2:a6:8d

Key comment: rsa-key-20190612

Key passphrase:

Confirm passphrase:

Actions

Generate a public/private key pair Generate

Load an existing private key file Load

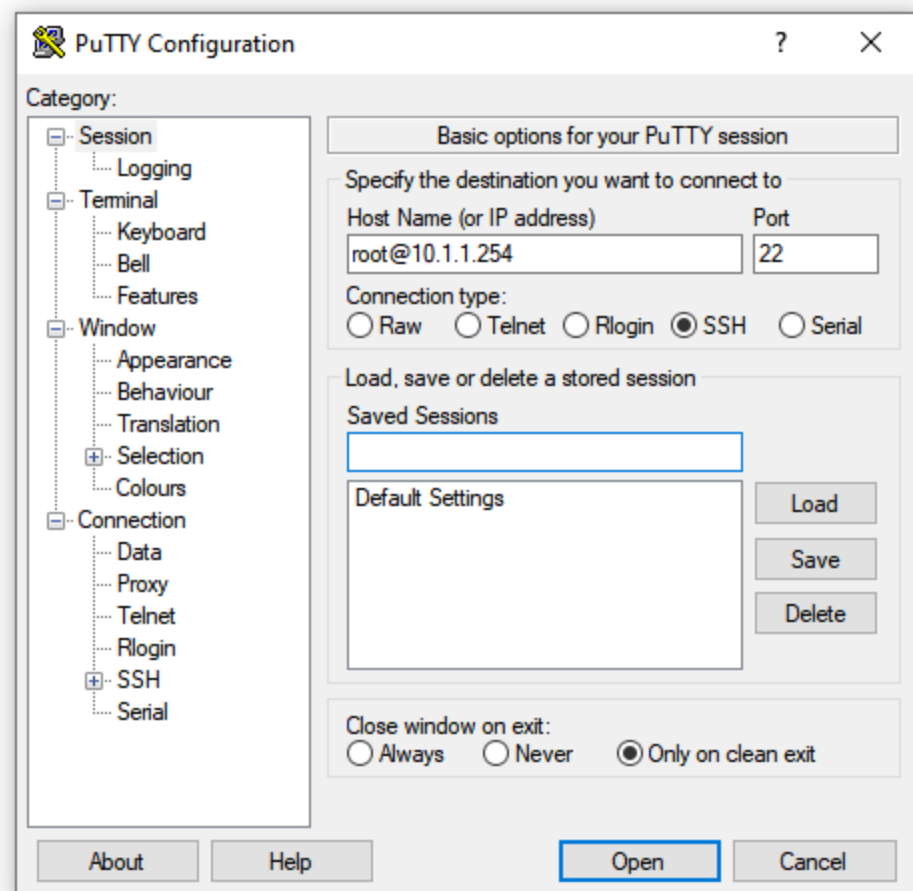
Save the generated key Save public key Save private key

Parameters

Type of key to generate:  
☒ RSA ☐ DSA ☐ ECDSA ☐ Ed25519 ☐ SSH-1 (RSA)

Number of bits in a generated key: 2048

# Выполнение доступа





# FTP

- Один из старейших протоколов в Internet и входит в его стандарты.
- Обмен данными в FTP проходит по TCP-каналу.
- Построен обмен по технологии "клиент-сервер".
- Пользователь FTP может вызывать несколько команд, которые позволяют ему посмотреть каталог удаленной машины, перейти из одного каталога в другой, а также скопировать один или несколько файлов.

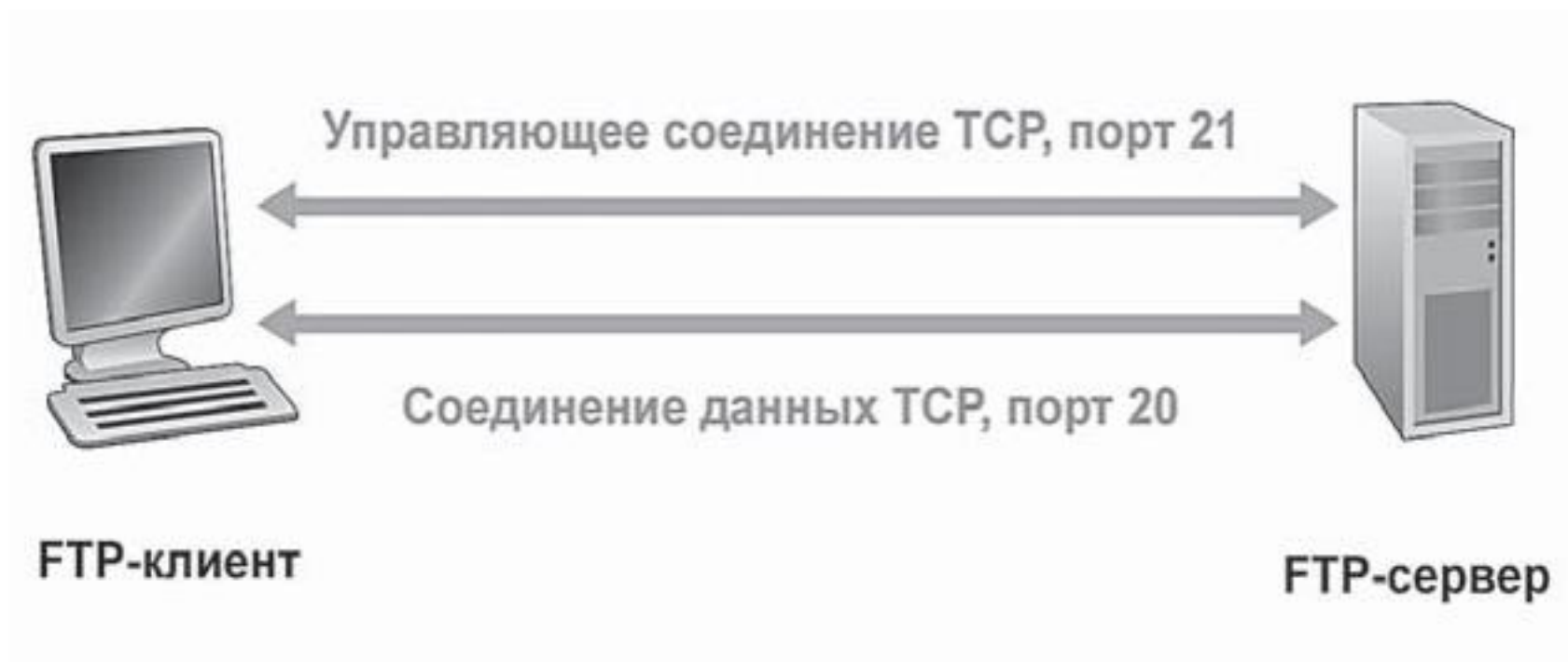
# FTP

- FTP использует два TCP соединения для передачи файла.
- Управляющее соединение устанавливается как обычное соединение клиент-сервер.
- Сервер осуществляет пассивное открытие на заранее известный порт FTP (21) и ожидает запроса на соединение от клиента.
- Клиент осуществляет активное открытие на TCP порт 21, чтобы установить управляющее соединение.
- Управляющее соединение существует все время, пока клиент общается с сервером.

# FTP

- Это соединение используется для передачи команд от клиента к серверу и для передачи откликов от сервера.
- Тип IP сервиса для управляющего соединения устанавливается для получения "минимальной задержки", так как команды обычно вводятся пользователем .
- Соединение данных открывается каждый раз, когда осуществляется передача файла между клиентом и сервером.
- Тип сервиса IP для соединения данных должен быть "максимальная пропускная способность", так как это соединение используется для передачи файлов.

# FTP



# FTP, Представление данных

- **ASCII файлы.** (По умолчанию) Текстовый файл передается по соединению данных как NVT ASCII.
- При этом требуется, чтобы отправитель конвертировал локальный текстовый файл в NVT ASCII, а получатель конвертировал NVT ASCII в текстовый файл.
- Конец каждой строки передается в виде NVT ASCII символа возврата каретки, после чего следует перевод строки. Это означает, что получатель должен просматривать каждый байт в поисках пары символов CR, LF.

# FTP, Представление данных

- **EBCDIC файлы.**

Альтернативный способ передачи текстовых файлов, когда на обоих концах системы EBCDIC.

- **Двоичные или бинарные файлы.**

Данные передаются как непрерывный поток битов.

- **Локальный тип файлов.**

Способ передачи бинарных файлов между хостами, которые имеют различный размер байта. Количество битов в байте определяется отправителем. Для систем, которые используют 8-битные байты, локальный тип файла с размером байта равным 8 эквивалентен бинарному типу файла.

# FTR, управление форматом

- **Nonprint. (По умолчанию)**  
Файл не содержит информацию вертикального формата.
- **Telnet format control.**  
Файл содержит управляющие символы вертикального формата Telnet, которые интерпретируются принтером.
- **Fortran carriage control.**  
Первый символ каждой строки это Fortran символ управления формата.

# FTP, структура

- **Структура файла.**  
(По умолчанию) Файл воспринимается в виде непрерывного потока байтов. Файл не имеет внутренней структуры.
- **Структура записи.**  
Эта структура используется только в случае текстовых файлов (ASCII или EBCDIC).
- **Структура страницы.**  
Каждая страница передается с номером страницы, что позволяет получателю хранить страницы в случайном порядке. Предоставляется операционной системой TOPS-20. (Требование к хостам Host Requirements RFC не рекомендует использовать эту структуру.)



# FTP, режим передачи

- **Режим потока.**

(По умолчанию) Файл передается как поток байтов. Для файловой структуры конец файла указывает на то, что отправитель закрывает соединение данных. Для структуры записи специальная 2-байтовая последовательность обозначает конец записи и конец файла.

- **Режим блоков.**

Файл передается как последовательность блоков, перед каждым из них стоит один или несколько байт заголовков.

- **Сжатый режим.**

Простое кодирование неоднократно встречающихся повторяющихся байт. В текстовых файлах обычно сжимаются пустые строки или строки из пробелов, а в бинарных строки из нулевых байт. (Этот режим поддерживается редко. Существуют более оптимальные способы сжатия файлов для FTP.)

# Команды FTP

Команда	Описание
ABOR	прервать предыдущую команду FTP и любую передачу данных
LIST список файлов	список файлов или директорий
PASS пароль	пароль на сервере
PORT n1,n2,n3,n4,n5,n6	IP адрес клиента (n1.n2.n3.n4) и порт (n5 x 256 + n6)
QUIT	закрыть бюджет на сервере
RETR имя файла	получить (get) файл
STOR имя файла	положить (put) файл
SYST	сервер возвращает тип системы
TYPE тип	указать тип файла: A для ASCII, I для двоичного
USER имя пользователя	имя пользователя на сервере

# Отклики FTP (первая цифра)

Отклик

Описание

- |     |  |
|-----|--|
| 1yz | Положительный предварительный отклик. Действие началось, однако необходимо дождаться еще одного отклика перед отправкой следующей команды.         |
| 2yz | Положительный отклик о завершении. Может быть отправлена новая команда.  |
| 3yz | Положительный промежуточный отклик. Команда принята, однако необходимо отправить еще одну команду.   |
| 4yz | Временный отрицательный отклик о завершении. Требуемое действие не произошло, однако ошибка временная, поэтому команду необходимо повторить позже. |
| 5yz | Постоянный отрицательный отклик о завершении. Команда не была воспринята и повторять ее не стоит.  |

# Отклики FTP (вторая цифра)

x0z	Синтаксическая ошибка.
x1z	Информация.
x2z	Соединения. Отклики имеют отношение либо к управляющему, либо к соединению данных.
x3z	Аутентификация и бюджет. Отклик имеет отношение к логированию или командам, связанным с бюджетом.
x4z	Не определено.
x5z	Состояние файловой системы.

# Пример ftp-клиента

```
from ftplib import FTP
```

```
HOST = 'localhost'
```

```
ftp = FTP(HOST, 'user', 'mypassword')
```

```
ftp.retrlines('LIST')
```

```
with open('README', 'wb') as fp:
```

```
    ftp.retrbinary('RETR README', fp.write)
```

```
ftp.quit()
```

# TFTP

- Простой протокол передачи файлов.
- Как правило, используется при загрузке бездисковых систем (рабочие станции или X терминалы).
- В отличие от протокола передачи файлов FTP, который использует TCP, TFTP использует UDP.
- Это сделано для того, чтобы протокол был как можно проще и меньше. Реализации TFTP (и необходимого UDP, IP и драйвера устройства) могут поместиться в постоянной памяти (ПЗУ).

# TFTP

- Обмен между клиентом и сервером начинается с того, что клиент запрашивает сервер либо прочитать, либо записать файл для клиента. В стандартном варианте загрузки бездисковой системы первый запрос - это запрос на чтение (RRQ).
- Первые 2 байта TFTP сообщения это код операции (opcode). В запросе на чтение (RRQ) и в запросе на запись (WRQ) имя файла (filename) указывает файл на сервере, который клиент хочет либо считать, либо записать.

# TFTP

- Если файл может быть прочитан клиентом, сервер отвечает пакетом данных с номером блока равным 1. Клиент посылает подтверждение (ACK) на номер блока 1.
- Сервер отвечает следующим пакетом данных с номером блока равным 2.
- Клиент подтверждает номер блока 2.
- Это продолжается до тех пор, пока файл не будет передан. Каждый пакет данных содержит 512 байт данных, за исключением последнего пакета, который содержит от 0 до 511 байт данных.
- Когда клиент получает пакет данных, который содержит меньше чем 512 байт, он считает, что получил последний пакет.



# TFTP

- В случае запроса на запись (WRQ) клиент посылает WRQ, указывая имя файла и режим. Если файл может быть записан клиентом, сервер отвечает подтверждением (ACK) с номером блока равным 0. Клиент посылает первые 512 байт файла с номером блока равным 1, сервер отвечает ACK с номером блока равным 1.
- Так как TFTP использует UDP, то именно от TFTP зависит, как будут обработаны потерянные и дублированные пакеты.
- В случае потери пакета, отправитель отрабатывает тайм-аут и осуществляет повторную передачу. (Возможно появление проблемы, называемой "синдромом новичка" которая может возникнуть, если с обеих сторон будет отработан тайм-аут и осуществлена повторная передача.

# Сетевая почтовая служба

- Распределенное приложение, главной функцией которого является предоставление пользователям сети возможности обмениваться электронными сообщениями.
- Обмен почтой с использованием ТСП осуществляется посредством агентов передачи сообщений (MTA - message transfer agent).

# Почтовый клиент / Mail User Agent

- программа, предназначенная для поддержания пользовательского интерфейса (обычно графического), а также для предоставления пользователю широкого набора услуг по подготовке электронных сообщений.
- Outlook, Thunderbird и т.д.

# Программа передачи сообщений / Mail Transfer Agent

- Наиболее распространенные MTA для Unix систем это Sendmail.
- При общении между двумя MTA используется NVT ASCII. Команды посылаются клиентом серверу, а сервер отвечает с помощью цифровых кодов и опциональных текстовых строк (для чтения человеком).

# MIME

- Multipurpose Internet Mail Extensions — многоцелевые расширения почты Интернета
- В заголовке каждой части сообщения имеется также информация о том, каким образом почтовый клиент должен обрабатывать тело части — отображать ее немедленно при открытии сообщения (например, встраивая изображение в текст) или считать это тело вложением (attachment), которое пользователь будет обрабатывать сам.

# Типы данных

- ASCII
- текст в 8-битном формате
- текст не в формате ASCII, преобразованный в ASCII-код
- гипертекст (HTML)
- изображение
- видеоклип
- звуковой файл

# S/MIME

- RFC 1847
- Цифровая подпись (Mutipart/Signed);
- Шифрованное тело (Multipart/Encrypted).

# Состав электронной почты

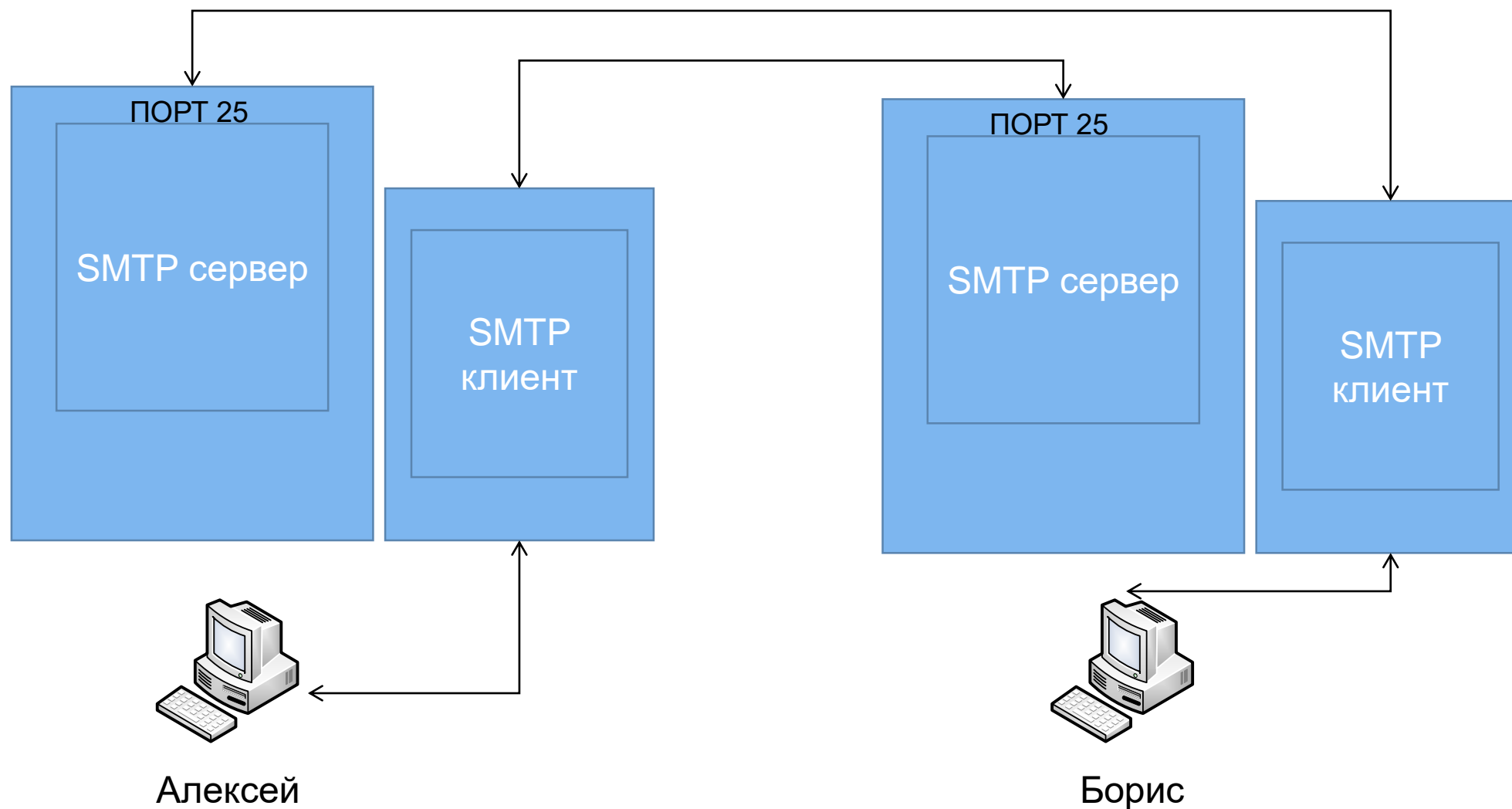
- **Конверт** используется МТА для доставки.
- Например:
  - MAIL From: <staff@bmstu.ru>
  - RCPT To: <student@bmstu.ru>
- RFC 821 определяет содержимое и интерпретацию конверта, а также протокол, который используется для обмена почтой по TCP соединению.
- **Заголовки** используются пользовательскими агентами. Например: Received, Message-Id, From, Date, Reply-To, X-Phone, X-Mailer, To и Subject. Каждое поле заголовка содержит имя, после которого следует двоеточие, а затем следует значение этого поля. RFC 822 определяет формат и интерпретацию полей заголовка. (Заголовки, начинающиеся с X-, это поля, определяемые пользователем.)
- **Тело** это содержимое сообщения - текстовые строки в формате NVT ASCII.



# SMTP

- Simple Mail Transfer Protocol — простой протокол передачи почты
- Является одним из первых стандартизованных протоколов прикладного уровня -1982
- Реализуется несимметричными взаимодействующими частями: SMTP-клиентом, работающим на стороне отправителя, и SMTP-сервером, работающим на стороне получателя. SMTP-сервер должен постоянно быть в режиме подключения, ожидая запросов со стороны SMTP-клиента.

# Непосредственное взаимодействие



# Процесс передачи сообщения

- Алексей, используя графический интерфейс своего почтового клиента, вызывает функцию создания сообщения, в результате чего на экране появляется стандартная незаполненная форма сообщения
- В поля которой Алексей вписывает свой адрес, адрес Бориса и тему письма, а затем набирает текст письма

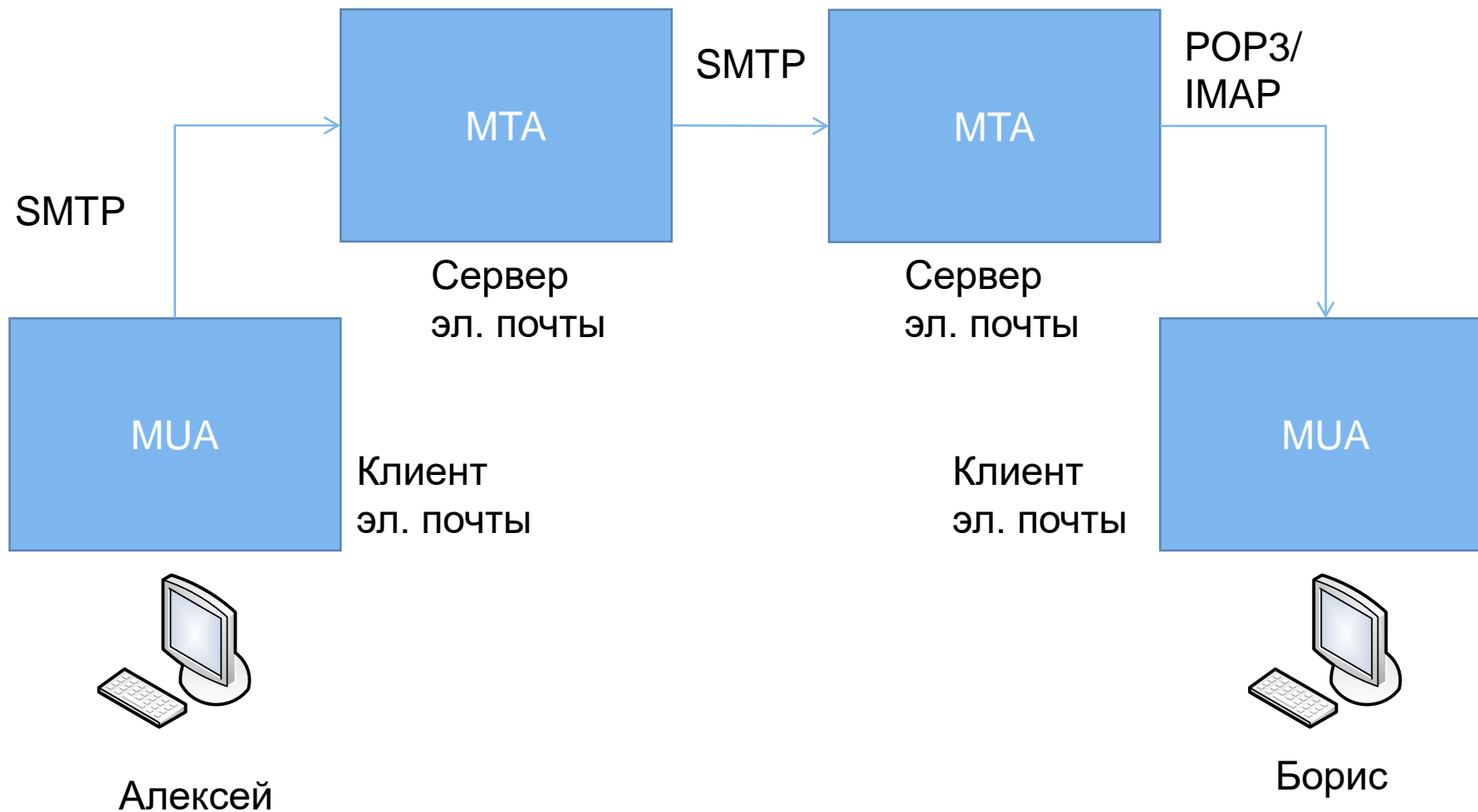
# Процесс передачи сообщения

- Когда письмо готово, Алексей вызывает функцию отправки сообщения, и встроенный SMTP-клиент посылает запрос на установление связи SMTP- серверу на компьютере Бориса.
- В результате устанавливаются SMTP и TCP-соединения, после чего сообщение передается через сеть.
- Почтовый сервер Бориса сохраняет письмо в памяти его компьютера, а почтовый клиент по команде выводит его на экран, при необходимости выполняя преобразование формата.

# Выделенный почтовый сервер

- Достаточно мощный и надежный компьютер, способный круглосуточно передавать почтовые сообщения от многих отправителей ко многим получателям (предоставляется организацией)
- Для каждого домена имен система DNS создает записи типа **MX**, хранящие DNS-имена почтовых серверов, обслуживающих пользователей, относящихся к этому домену.

# Использование промежуточных серверов



# Порядок обмена данными

- Он пишет текст сообщения, указывает необходимую сопроводительную информацию, в частности адрес получателя `boris@bmstu.ru`.
- Поскольку готовое сообщение должно быть направлено совершенно определенному почтовому серверу, клиент обращается к системе DNS, чтобы определить имя почтового сервера, обслуживающего домен `bmstu.ru`.
- Получив от DNS ответ `mail.bmstu.ru`, SMTP-клиент еще раз обращается к DNS — на этот раз чтобы узнать IP-адрес почтового сервера `mail.bmstu.ru`.

# Порядок обмена данными

- SMTP-клиент посылает по данному IP-адресу запрос на установление TCP-соединения через порт 25 (SMTP-сервер).
- Начинается диалог между клиентом и сервером по протоколу SMTP.
- Направление передачи запроса от клиента на установление SMTP-соединения совпадает с направлением передачи сообщения. Если сервер оказывается готовым, то после установления TCP-соединения сообщение Алексея передается.



# Порядок обмена данными

- Письмо сохраняется в буфере почтового сервера, а затем направляется в индивидуальный буфер, отведенный системой для хранения корреспонденции Бориса (т.е. почтовый ящик)
- В какой-то момент Борис запускает свою почтовую программу и выполняет команду проверки почты.
- После этой команды почтовый клиент должен запустить протокол доступа к почтовому серверу. Однако это будет не SMTP.

# Порядок обмена данными

- Инициатором передачи сообщений от почтового сервера почтовому клиенту по протоколу POP3 или IMAP является клиент.
- Почтовый сервер ожидает запрос на установление TCP-соединения по протоколу POP3 через порт 110, а по протоколу IMAP — через порт 143.
- В результате работы любого из них письмо Алексея передается в память компьютера Бориса.

# Пример smtp-клиента

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

```
msg = MIMEMultipart()
message = "Sample message"
```

```
password = "mypassword"
msg['From'] = "sender@gmail.com"
msg['To'] = "receiver@gmail.com"
msg['Subject'] = "Subscription"
```

```
msg.attach(MIMEText(message, 'plain'))
```

```
server = smtplib.SMTP('smtp.gmail.com: 587')
```

```
server.starttls()
```

```
server.login(msg['From'], password)
```

```
server.sendmail(msg['From'], msg['To'], msg.as_string())
server.quit()
```

```
print
"successfully sent email to %s:" % (msg['To'])
```

# SMTP команды

- **MAIL** запускает пользовательского агента. Затем необходимо ввести тему сообщения, после чего можно печатать тело сообщения. Ввод точки в начале строки завершает сообщение, и пользовательский агент передает почту в МТА для доставки.
- Клиент осуществляет активное открытие на TCP порт 25, после чего ожидает приветственного сообщения (отклик с кодом 220) от сервера.
- **HELO** позволяет клиенту идентифицировать себя.
- **MAIL** идентифицирует автора сообщения (или отправителя). Следующая команда, RCPT, идентифицирует получателя. Если сообщение предназначено нескольким получателям, может быть исполнено несколько команд RCPT.
- **DATA** отправляет содержимое почтового сообщения. Строка, содержащая только точку, указывает на конец сообщения.
- **QUIT** прекращает обмен почтой.

# SMTP команды

- **RSET** прекращает текущую передачу почты и заставляет оба конца "сброситься". Любая сохраненная информация об отправителе, получателе или содержимое почты уничтожается.
- **VRFY** позволяет клиенту попросить отправителя проверить адрес получателя, не отправляя ему почту. Этим часто пользуются системные администраторы, чтобы вручную определить проблемы с доставкой почты. **NOOP** не делает ничего, однако заставляет сервер ответить, что все нормально, а именно откликом с кодом 200.
- Дополнительно: **EXPN** расширяет список почты и часто используется системными администраторами, так же как и **VRFY**. Более того, большинство версий Sendmail обрабатывают эти две команды одинаково.

# SMTP команды

- **TURN** позволяет клиенту и серверу поменяться ролями, чтобы послать почту в обратном направлении, не разрывая TCP соединение и не создавая новое. (Sendmail не поддерживает)
- **SEND**, **SOML** и **SAML** редко реализуются и призваны заменить собой команду MAIL. Позволяют доставлять почту непосредственно на пользовательский терминал (если пользователь находится терминалом в системе) или складывать ее в почтовый ящик получателя.

# Интервалы между ретрансляциями

- Когда пользовательский агент передает новое почтовое сообщение своему МТА, попытка доставить сообщение обычно осуществляется немедленно. Если доставить сообщение не удалось, МТА поставит сообщение в очередь и повторит попытку позже.
- Требования к хостам Host Requirements RFC рекомендует устанавливать первоначальный тайм-аут по крайней мере в 30 минут.
- Отправитель должен повторять свои попытки по меньшей мере 4-5 дней.
- Если сбои в доставке происходят часто (получатель вышел из строя или произошла временная потеря сетевого соединения), имеет смысл делать две попытки установить соединение в течение первого часа, когда сообщение находится в очереди.

# POP3

- Предназначен для получения сообщений, находящихся в почтовом ящике пользователя на удаленном сервере электронной почты.
- Сервер **SMTP** должен быть доступен постоянно, а рабочие станции обычно включают только на время работы пользователя, соединение с сервером они нередко устанавливают по коммутируемым линиям только для того, чтобы забрать накопившуюся почту.
- Почта доставляется только в хранилище сообщений, откуда пользователь может ее забрать в удобное для него время.



# Алгоритм работы POP3

- После установления соединения сервер посылает клиенту строку приветствия, свидетельствующую о готовности к диалогу, и сеанс переходит в состояние авторизации (AUTHORIZATION State). На этом этапе выясняется, доступ к какому именно почтовому ящику запрашивает клиент и имеет ли он соответствующие права. Успешное прохождение авторизации необходимо для продолжения работы.
- Если авторизация проходит успешно, то сеанс переходит в состояние транзакции (TRANSACTION State). На этом этапе клиент может проделывать все необходимые манипуляции с почтовым ящиком: он может просмотреть информацию о состоянии ящика и отдельных сообщений, получить выбранные сообщения и пометить письма, подлежащие удалению.

# Алгоритм работы POP3

- По окончании всех операций, клиент сообщает об окончании связи, и сеанс переходит в состояние обновления (UPDATE State). На этом этапе сервер стирает из ящика сообщения, помеченные на предыдущем этапе как подлежащие удалению, и закрывает соединение.
- Переход в состояние обновления в принципе возможен, только если клиент выходит из состояния транзакции по команде QUIT . Ни при каких других обстоятельствах, например, если сеанс связи прерывается по таймауту или из-за обрыва связи, переход в состояние обновления происходить не должен. То есть, если состояние транзакции прерывается не по команде QUIT , никакие удаления не должны производиться, пометки для удаления должны быть аннулированы. К сожалению, как показывает практика, это требование выполняется не всег

# Алгоритм работы POP3

- В ходе сеанса клиент посылает серверу команды, а сервер сообщает о результате выполнения каждой из них.
  - Ответ состоит из индикатора состояния (status indicator) и, если нужно, дополнительной информации, отделенной пробелом.
  - Строка ответа может содержать до 512 символов, включая последовательность **CRLF**, обозначающую конец строки.
- Предусмотрено два индикатора состояния:
  - "+OK" – успешное завершение и
  - "-ERR" – неуспешное завершение.

Если строка ответа не содержит дополнительной информации, то после индикатора состояния сразу должна идти последовательность **CRLF**. Однако некоторые клиенты ожидают пробела после индикатора состояния.

# Алгоритм работы POP3

- Если команда предусматривает многострочный ответ, то индикатор состояния передается только в первой строке, а последняя строка ответа должна состоять из одной точки. Эта строка не является частью ответа, а только обозначает его завершение.
- Чтобы сделать возможным использование строк, состоящих из одной точки, в ответах сервера, ко всем строкам ответа, начинающимся с точки, добавляется еще одна точка.

# Основные команды POP3

- В ответ на команду **STAT** сервер возвращает количество сообщений в почтовом ящике и общий размер ящика в октетах. Сообщения, помеченные для удаления, при этом не учитываются. Например, ответ "+OK 4 223718" означает, что в почтовом ящике имеется 4 сообщения общим объемом 223718 октет.
- Ответ на команду **LIST** без аргумента: список сообщений в почтовом ящике, содержащий их порядковые номера и размеры в октетах.
- Команда **RETR** Требует в качестве аргумента номер существующего и не помеченного для удаления сообщения. В ответ сервер присылает запрошенное сообщение.

# Основные команды POP3

- Команда **DELE** требует в качестве аргумента номер существующего и не помеченного для удаления сообщения. Указанное сообщение помечается для удаления. До конца сеанса обращаться к нему становится невозможно. После окончания диалога, когда сеанс переходит в состояние обновления, сообщение удаляется окончательно.
- На команду **NOOP** команду сервер должен дать положительный ответ. Никаких других действий не производится.
- Команда **RSET** сервер снимает все установленные ранее пометки для удаления.
- Команда **QUIT** завершает сеанс. Если в ходе сеанса какие-то сообщения были помечены для удаления, то после выполнения команды QUIT они удаляются из ящика.

# Дополнительные возможности POP3

- Кроме обязательных команд, перечисленных выше, программное обеспечение, реализующее взаимодействие по протоколу **POP3**, поддерживает дополнительные возможности (capabilities), вводящие новые команды, влияющие на исполнение основных команд, облегчающие взаимодействие клиента и сервера, информирующие об особенностях реализации сервера и хранилища сообщений.
- В число дополнительных возможностей входят, например, команды авторизации. Хотя бы один механизм авторизации должен быть реализован, так как доступ к почтовому ящику предоставляется только после аутентификации. Но, поскольку таких механизмов несколько, и их выбор оставляется на усмотрение разработчиков и администраторов, соответствующие команды не входят в число обязательных.
- Предусмотрена команда **CAPA**, позволяющая клиенту получить информацию о дополнительных возможностях, реализованных на сервере, и их параметрах.

# Протокол IMAP

- Используется на участке между **MUA** получателя и хранилищем сообщений.
- предоставляет более широкие возможности работы с почтовыми ящиками, чем **POP3**: он позволяет работать с несколькими почтовыми ящиками на одном или нескольких серверах **IMAP** как с файлами и каталогами на собственной машине пользователя.
- Сервер **IMAP** способен анализировать сообщение: выделять заданные поля заголовка и разбирать структуру тела сообщения.



# Состояния сеанса IMAP

- **Неаутентифицированное состояние** (Not Authenticated State): клиент должен пройти процедуру аутентификации прежде, чем сможет выполнять большинство команд;
- **Аутентифицированное состояние** (Authenticated State): клиент аутентифицирован и должен выбрать почтовый ящик, прежде чем сможет работать с отдельными сообщениями;
- **Выбранное состояние** (Selected State): почтовый ящик выбран;
- **Состояние выхода** (Logout State): сеанс завершается.

# Команды любого состояния сеанса

- **CAPABILITY.** В ответ на эту команду сервер присылает непомеченную строку с ключевым словом CAPABILITY, содержащую список поддерживаемых возможностей (расширений) и их параметров. В число возможностей входит в частности поддерживаемая версия протокола **IMAP** – IMAP4rev1 и механизмы аутентификации ( AUTH =механизм\_аутентификации), описанные в RFC 2595 .
- **NOOP.** Не выполняет никаких действий. Однако эта команда сбрасывает таймер неактивности, что позволяет избежать разрыва соединения по таймауту. Кроме того, при определенных обстоятельствах эта или другая команда служит неявным запросом информации об обновлениях, произошедших на сервере. Таким образом, с помощью команды NOOP можно периодически проверять, не появились ли новые сообщения или не изменился ли статус старых.
- **LOGOUT.** Конец сеанса.

# Команды неаутентифицированного состояния

- Информация о поддерживаемых способах аутентификации передается сервером клиенту в ответе на команду CAPABILITY.
- Так запись STARTTLS свидетельствует о поддержке одноименной команды, описанной в RFC 2595 , LOGINDISABLED – расширение, исключающее аутентификацию с использованием незашифрованных имени и пароля, параметры AUTH указывают, какие механизмы аутентификации с использованием **SASL** поддерживает сервер.

# Команды неаутентифицированного состояния

- **STARTTLS.** Переводит сеанс в защищенный режим. После получения сервером команды STARTTLS клиент и сервер согласовывают параметры дальнейшего взаимодействия. Все данные, которыми обмениваются клиент и сервер после успешного завершения этой команды, передаются в зашифрованном виде. Однако аутентификация при помощи этой команды не производится, сеанс остается в неаутентифицированном состоянии.
- **LOGIN регистрационное\_имя\_пользователя пароль** Аутентификация при помощи регистрационного имени и пароля, передаваемых открытым текстом.
- **AUTHENTICATE механизм** - передача зашифрованных аутентификационных данных с использованием **SASL** .

# Команды аутентифицированного состояния

- **SELECT имя\_ящика**
- Открывает доступ к указанному почтовому ящику. Сеанс переходит в состояние выбора, после этого клиент может работать с отдельными сообщениями в ящике.
- В ответ на эту команду сервер присылает ряд непомеченных ответов, содержащих информацию о почтовом ящике: количество сообщений, список допустимых флагов (см. описание команды APPEND), количество новых сообщений, номер первого непрочитанного сообщения, идентификатор почтового ящика.

# Команды аутентифицированного состояния

- **EXAMINE имя\_ящика** - аналогично команде SELECT , но почтовый ящик открывается только для чтения.
- **CREATE имя\_объекта** Создает новый почтовый ящик или каталог. Если объект создается не в корневом каталоге, то надо указать путь к нему.
- Если на конце указанного имени стоит символ, используемый в качестве иерархического разделителя, создается каталог.
- **DELETE имя\_ящика** - удаляет указанный почтовый ящик. Эта же команда удаляет также и каталоги, если они не содержат почтовые ящики.

# Команды аутентифицированного состояния

- **RENAME** имя\_ящика новое\_имя\_ящика
- Переименование почтового ящика.
- **SUBSCRIBE** имя\_ящика
- Почтовый ящик помечается как "активный". Эта пометка используется для вывода списка почтовых ящиков при помощи команды LSUB.
- **UNSUBSCRIBE** имя\_ящика
- Снимает с почтового ящика пометку "активный". Эта пометка может быть снята с почтового ящика только при помощи команды UNSUBSCRIBE. Даже если ящик больше не существует, это не может само по себе стать причиной снятия пометки "активный".

# Команды аутентифицированного состояния

- **LIST** *путь\_к\_ящику имя\_ящика* возвращает список каталогов и почтовых ящиков, соответствующих указанным аргументам.
- **LSUB** *путь\_к\_ящику имя\_ящика* команда LSUB аналогична команде LIST , но она возвращает только имена почтовых ящиков с пометкой "активный".
- **APPEND** *имя\_ящика (флаги\_сообщения) метка\_времени сообщение*
- Добавляет сообщение в конец указанного почтового ящика. В качестве аргументов указываются имя ящика, флаги сообщения (не обязательно), метка времени (не обязательно) и само сообщение – заголовок и тело.



# Команды аутентифицированного состояния

- **STATUS имя\_ящика (имена\_элементов)**
- Возвращает запрошенные элементы информации об указанном почтовом ящике. Имена элементов информации разделяются пробелами и все вместе заключаются в скобки. Предусмотрены следующие имена элементов информации:
  - **MESSAGES** – общее количество сообщений в ящике;
  - **RECENT** – количество новых сообщений;
  - **UIDNEXT** –уникальный идентификатор, который изменяется всякий раз, когда в почтовый ящик помещается новое сообщение, используется для того, чтобы определить, появились ли в ящике новые сообщения за время, прошедшее после предыдущей проверки;
  - **UIDVALIDITY** – уникальный идентификатор почтового ящика;
  - **UNSEEN** – количество сообщений, не помеченных как прочитанные.

# Команды выбранного состояния

- **CHECK**

- Команда производит проверку выбранного почтового ящика, характер которой зависит от реализации программного обеспечения сервера.

- **CLOSE**

- Выбранный почтовый ящик закрывается. При этом, если почтовый ящик был открыт для чтения и записи, все помеченные для удаления сообщения в ящике удаляются. Сеанс возвращается в аутентифицированное состояние.

# Команды выбранного состояния

- **EXPUNGE**
- Из выбранного почтового ящика удаляются все помеченные для удаления сообщения. Для каждого удаляемого сообщения посылается непомеченный ответ, содержащий номер сообщения и ключевое слово EXPUNGE .
- **SEARCH кодировка\_символов критерии\_поиска**
- Поиск в выбранном почтовом ящике сообщений, отвечающих указанным критериям поиска.
- **FETCH x:y имя\_элемента\_сообщения\_или\_макрос**
- Сервер возвращает информацию, относящуюся к сообщениям, обозначенным первым аргументом команды. Это может быть либо число, обозначающее номер сообщения, либо интервал от номера x до номера y, записанный в формате x:y.
- Во втором аргументе перечисляются запрашиваемые информационные элементы.

# Команды выбранного состояния

- **STORE x:y имя\_элемента\_данных (список\_флагов)**
- Команда STORE изменяет значения флагов для указанного в первом аргументе сообщения или сообщений. В ответ сервер возвращает непомеченный ответ с ключевым словом FETCH, содержащий значения флагов, если в имени элемента данных не используется суффикс .SILENT.
- **COPY x:y имя\_ящика**
- Копирует сообщения с номерами от x до y из текущего почтового ящика в указанный почтовый ящик.

# Команды выбранного состояния

- **UID** команда аргументы
- Команда **UID** принимает в качестве аргументов команды COPY, FETCH или STORE с их аргументами, но наряду с номерами сообщений в ответах указываются уникальные идентификаторы сообщений.
- Также команду **UID** можно использовать совместно с командой SEARCH. В этом случае интерпретация аргументов команды SEARCH не изменяется, но в ответах на эту команду будут приведены уникальные

# Список использованных источников

- В. Олифер, Н. Олифер “Компьютерные сети. Принципы, технологии, протоколы”
- Куроуз, Росс “Компьютерные сети. Нисходящий подход.”
- <https://docs.python.org/>

Спасибо за внимание!