

# Компьютерные сети

## лекция 1



Андрей  
Вахутинский



## **Андрей Вахутинский**

Зам.начальника ИТ отдела в АО "ИНТЕКО"





# План модуля

1. Работа в терминале, лекция 1
2. Работа в терминале, лекция 2
3. Операционные системы, лекция 1
4. Операционные системы, лекция 2
5. Файловые системы
- 6. Компьютерные сети, лекция 1**
7. Компьютерные сети, лекция 2
8. Компьютерные сети, лекция 3
9. Элементы безопасности информационных систем



# План занятия

1. [Об истории компьютерных сетей](#)
2. [All People Seem To Need Data Processing\\*, TCP/IP](#)
3. [Некоторые популярные сетевые утилиты](#)
4. [Что происходит, когда вы открываете сайт](#)
5. [Итоги](#)
6. [Домашнее задание](#)

\*All People Seem To Need Data Processing (всем людям необходима обработка данных) - фраза, которая помогает запомнить порядок слоёв: a - application, p - presentation, s - session, t - transport, n - network, d - data link, p - physical.

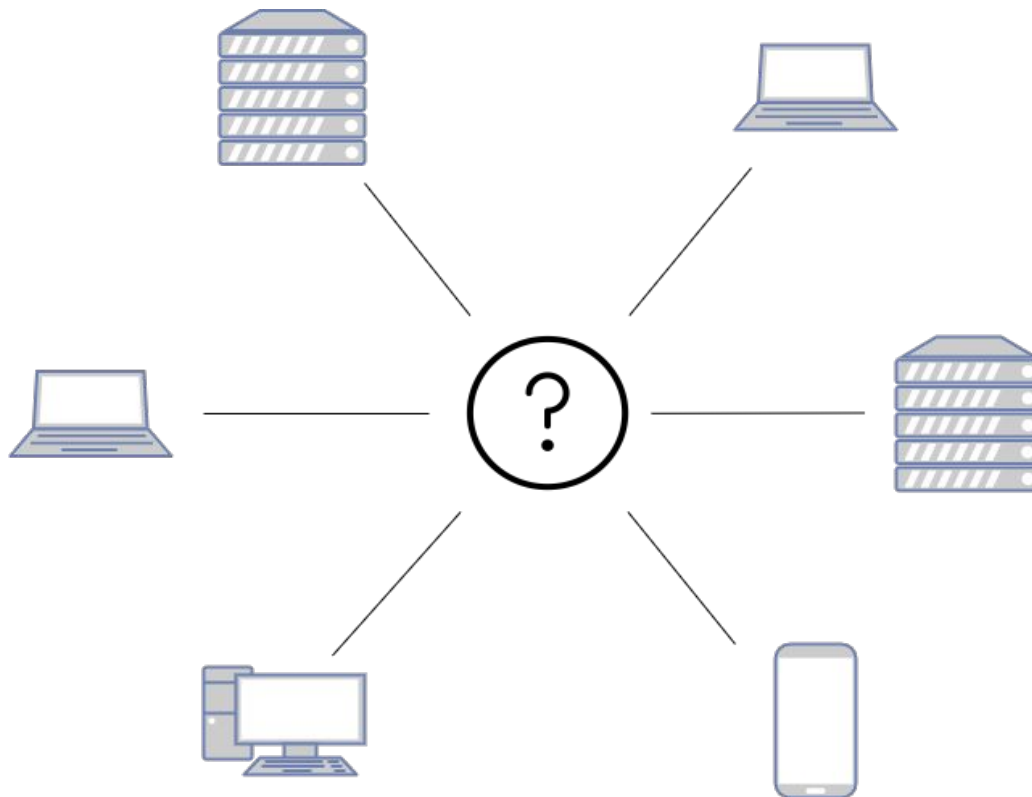


# Об истории компьютерных сетей

---

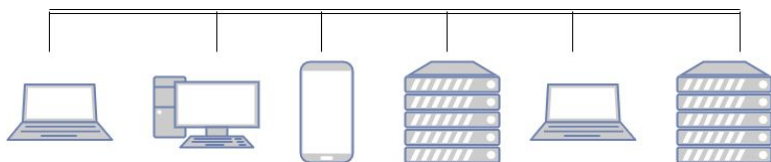
# Как объединить компьютеры в сеть?

В данной лекции делаем акцент на деление **по типу среды** и **по типу коммутации**.

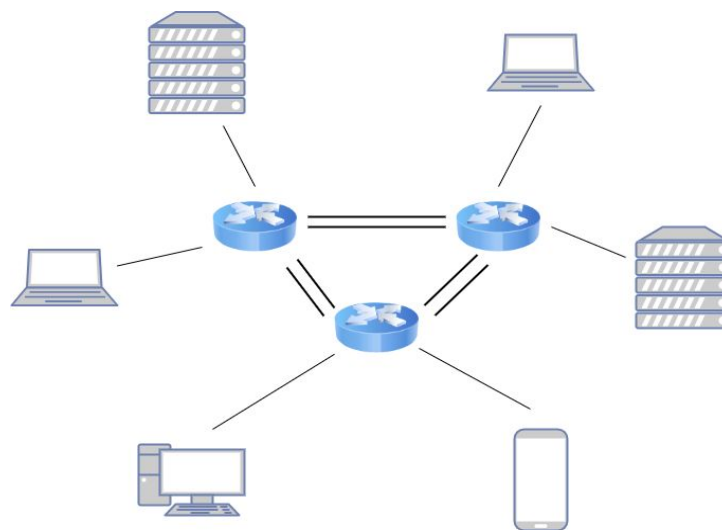


# Сети с разделяемой и коммутируемой средой

*Общая медиа (разделяемая среда)*



*Switched – коммутируемая среда*





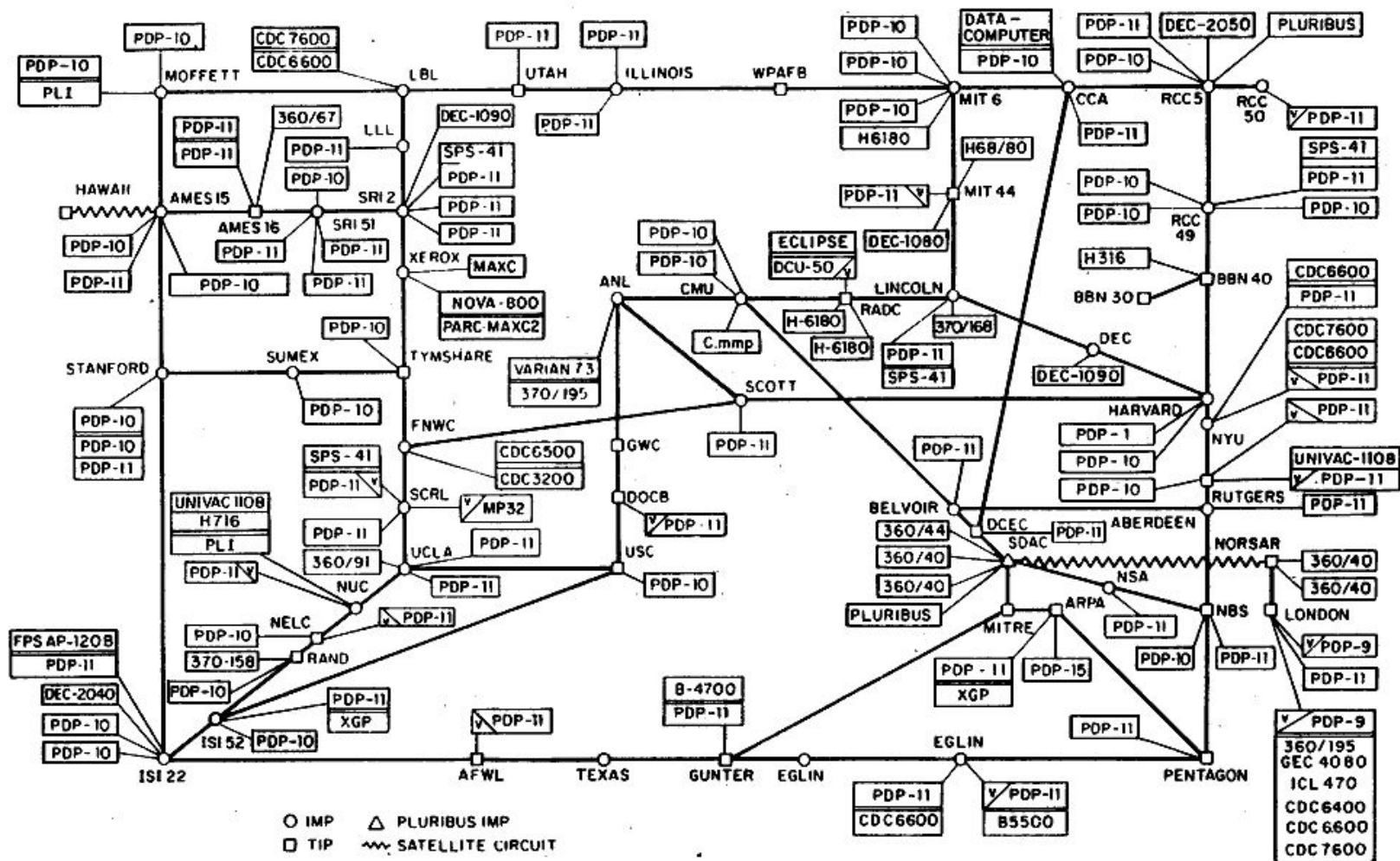
# Сети с коммутацией каналов и пакетов

- **Circuit switching (канальная коммутация)**
  - аналоговая АТС
- **Packet switching (пакетная коммутация)**
  - ARPANET
  - SNA (IBM S/360)
  - Internet



# ARPANET – прообраз Интернета в 1977 году

ARPANET LOGICAL MAP, MARCH 1977



(PLEASE NOTE THAT WHILE THIS MAP SHOWS THE HOST POPULATION OF THE NETWORK ACCORDING TO THE BEST INFORMATION OBTAINABLE, NO CLAIM CAN BE MADE FOR ITS ACCURACY)

NAMES SHOWN ARE IMP NAMES, NOT (NECESSARILY) HOST NAMES

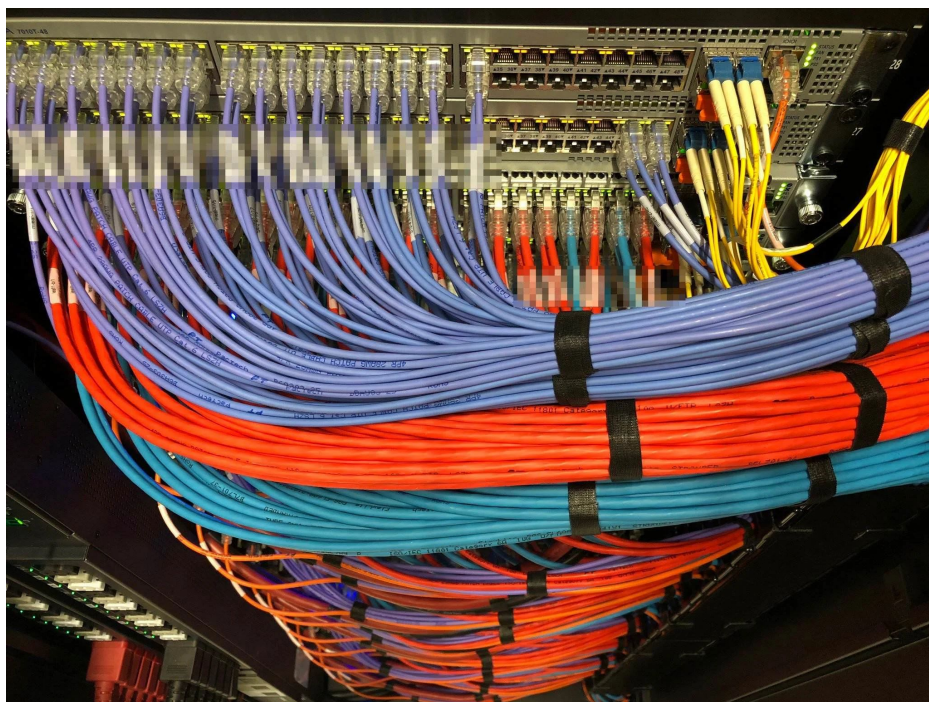
# Необходимость двухуровневой адресации

- **Разделяемая среда – одноуровневая адресация**
  - вся сеть – локальная, поэтому только адреса локальной сети
  - “все слышат всех”, проблем с доставкой до адресата нет
- **Выделенная среда – одноуровневая адресация**
  - появляется дополнительное устройство – коммутатор, который обеспечивает выделенный канал между своим портом и устройством и понимает кому предназначаются данные (= отсутствие коллизий)
  - все уже не слышат всех, но коммутатор знает какой адрес находится за каким физическим портом

Но что будет, если в коммутатор уже невозможно подключить новых участников сети, или они удалены по расстоянию? Можно сделать несколько локальных сетей и объединить их.

- **Двухуровневая адресация – необходимость при объединении нескольких сетей**
  - не пересекающиеся с адресами локальных сетей наборы адресов

# 48-портовый коммутатор



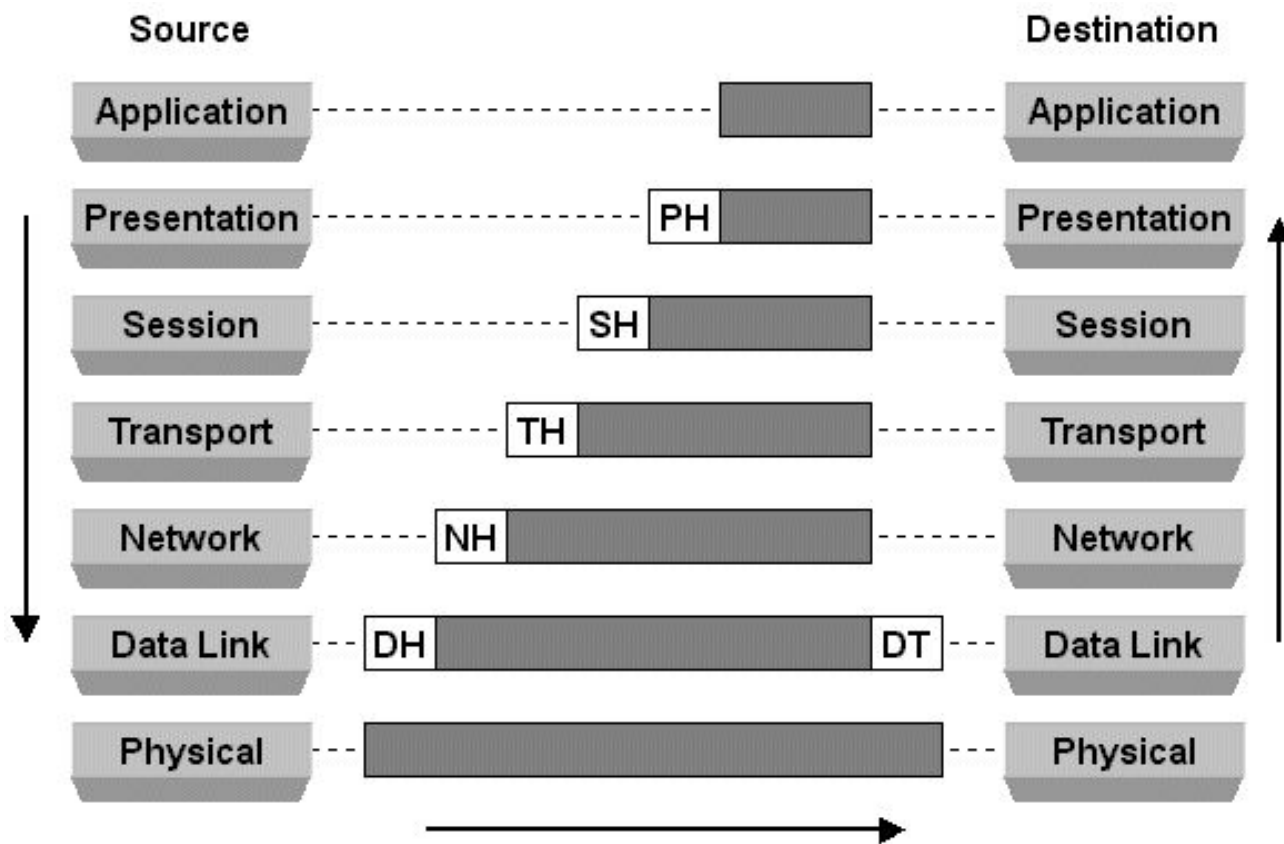


# **All People Seem To Need Data Processing\*, TCP/IP**

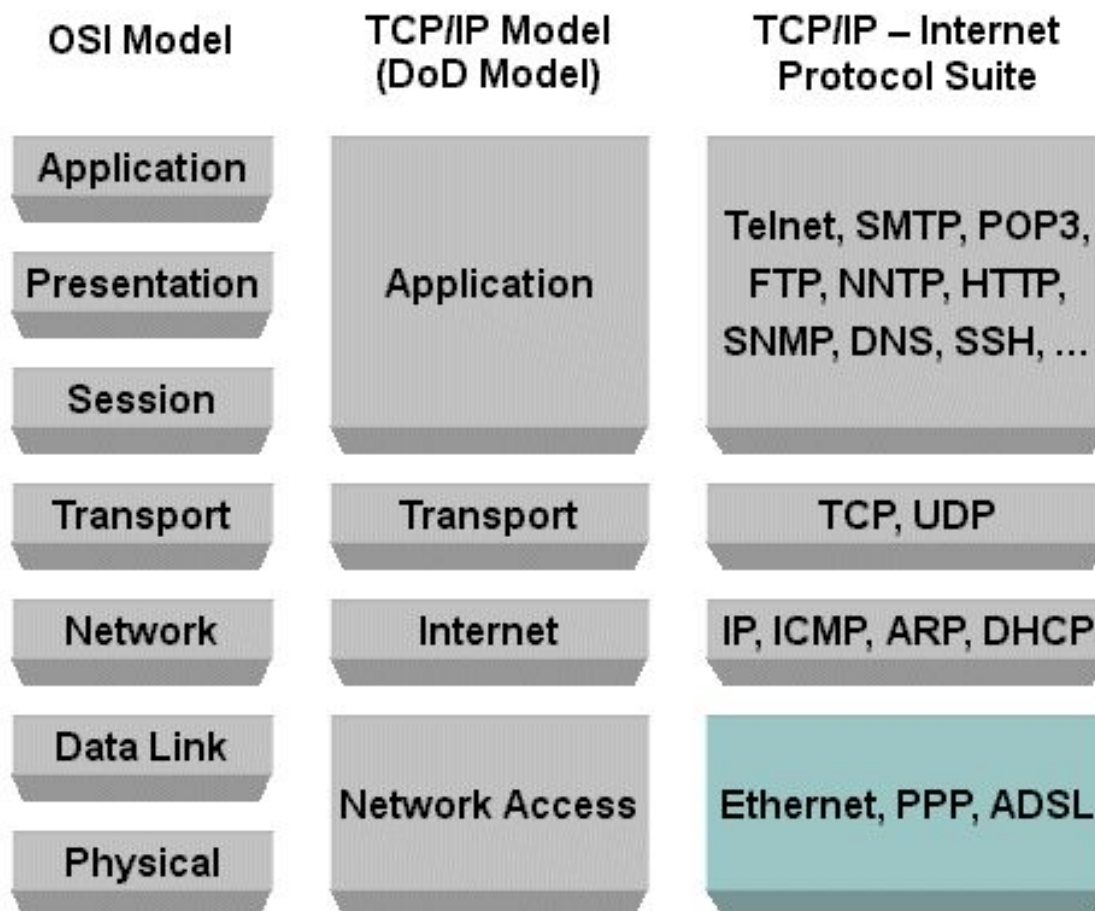
\*All People Seem To Need Data Processing (всем людям необходима обработка данных) - фраза, которая помогает запомнить порядок слоёв: a - application, p - presentation, s - session, t - transport, n - network, d - data link, p - physical.

# ISO/OSI: All People Seem To Need Data Processing

ISO/OSI (Open Systems Interconnection) – сетевая модель стека протоколов передачи данных



# Сетевые модели OSI и TCP/IP



# Термины OSI

## Уровни:

- протокол второго или канального уровня
- протокол третьего или сетевого уровня
- протокол четвертого или транспортного уровня

**PDU, Protocol Data Unit** – блок данных протокола. На каждом уровне он имеет свое название:

- на канальном уровне PDU – фрейм
- на сетевом уровне PDU – пакет
- на транспортном уровне PDU – сегмент или датаграмма

## Устройства/концепции:

- L2 switch – коммутатор второго уровня
- L3 router – роутер третьего уровня
- L4/L7 load balancer – балансировщик нагрузки, работающий на транспортном уровне или на уровне приложений

# Сокращения

**TCP**, Transmission Control Protocol – протокол управления передачей

**UDP**, User Datagram Protocol – протокол пользовательских датаграмм

**MAC**, Media Access Control – протокол управления доступом к среде

**IP**, Internet Protocol – протокол для межсетевых соединений, версии:

- v4
- v6

**ARP**, Address Resolution Protocol – протокол для сопоставления IPv4 и MAC

**NDP**, Neighbour Discovery Protocol – протокол TCP/IP для IPv6, одна из функций которого – замена ARP v4

**ICMP**, Internet Control Message Protocol – протокол межсетевых управляющих сообщений

**DHCP**, Dynamic Host Configuration Protocol – протокол динамического конфигурирования хостов

*Следует понимать, что этими основными протоколами возможности стека не просто не ограничиваются, а это только малая часть, нужная для базового понимания.*

*Не менее важны протоколы класса EGP для маршрутизации внешних шлюзов, External Gateway Protocol, такие как OSPF, туннельные протоколы и т.д.*



# Принадлежность протоколов к уровням

Как отмечалось, конкретные протоколы TCP/IP не соответствуют идеально уровням модели OSI. Тем не менее, понимание *принципиальной* принадлежности протоколов к уровням дает возможность легче запомнить их предназначение.

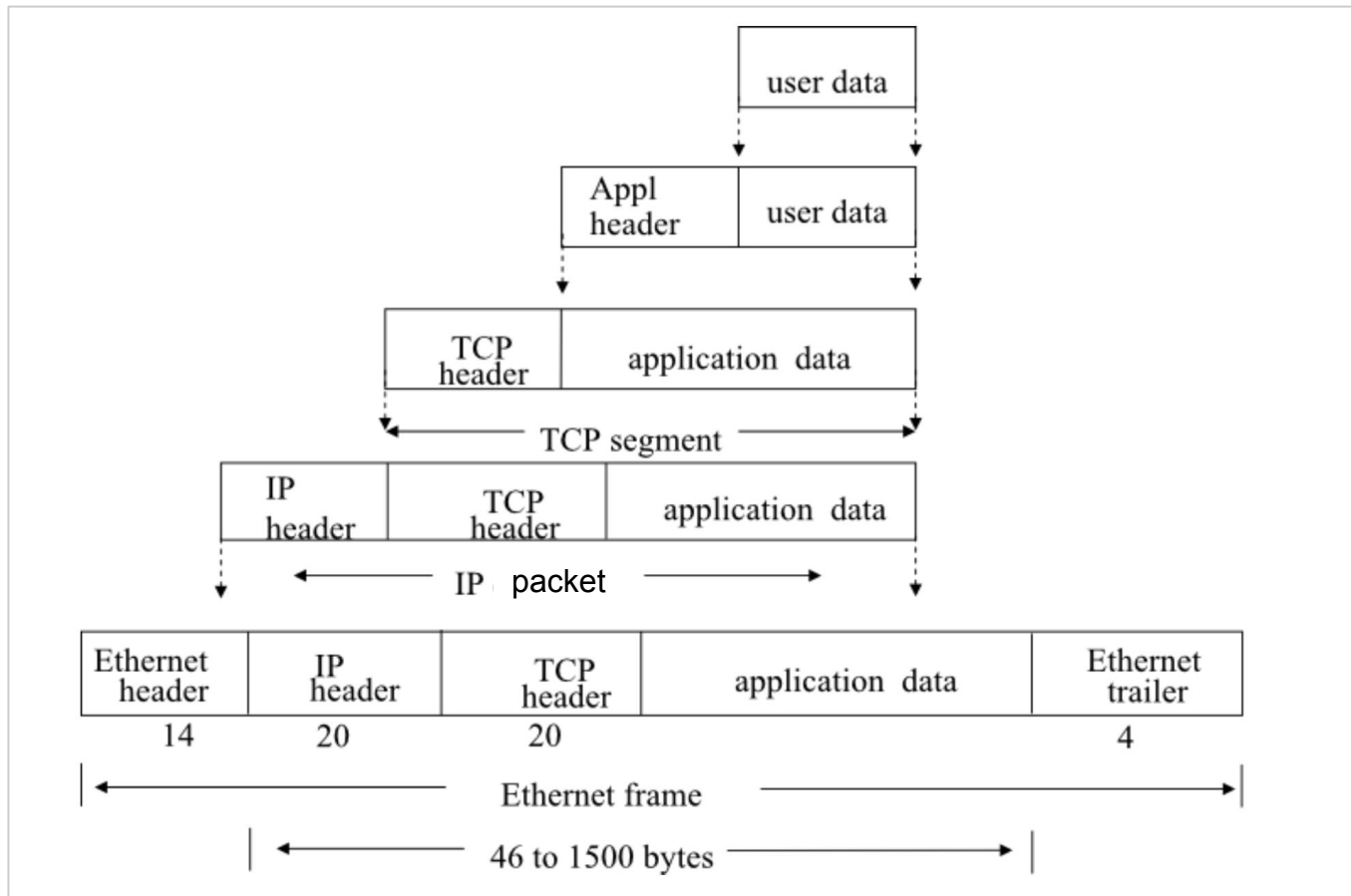
Так, ARP работает на канальном уровне L2, то есть он никогда не выходит за рамки домена сети, адресация в котором возможна без использования IP адресов. Хотя сам сервис, который он предоставляет, по сути необходим и для сетевого уровня L3 (сопоставление MAC и IP адресов).

Принадлежность ICMP тоже является предметом каверзных вопросов на собеседованиях, ведь хотя сам ICMP инкапсулирован в IP, то есть по сути должен являться протоколом транспортного уровня L4, он не реализует задач транспортного уровня, не использует порты (что типично для TCP/UDP), а предназначен для передачи сервисных сообщений о работе IP L3. Поэтому формально его относят к L3 а не к L4.

Также физические устройства могут сочетать в себе несколько уровней: все мы знакомы с L3 роутерами, которые одновременно являются и L2 коммутаторами, хотя это могут быть и независимые устройства.

# MTU, MSS\*

TCP/IP Illustrated, Vol. 1



\*MTU (Maximum Transmission Unit) – размер полезных данных в одном фрейме (размер фрейма минус заголовки Ethernet, минус трейлер Ethernet),

MSS (Maximum Segment Size) – размер полезных данных (payload) в одном сегменте (MTU минус заголовки TCP, минус заголовки IP)

# Формат заголовков Ethernet + хвостовик

## Ethernet (802.3) Frame Format

7 bytes	1 byte	6 bytes	6 bytes	2 bytes	42 to 1500 bytes	4 bytes	12 bytes
Preamble	Start of Frame Delimiter	Destination MAC Address	Source MAC Address	Type	Data (payload)	CRC	Inter-frame gap

For TCP/IP communications, the payload for a frame is a packet

## WiFi (802.11) Frame Format

2 bytes	2 bytes	6 bytes	6 bytes	6 bytes	2 bytes	6 bytes	0 to 2312 bytes	4 bytes
Frame Control	Duration	MAC Address 1 (Destination)	MAC Address 2 (Source)	MAC Address 3 (Router)	Seq Control	MAC Address 4 (AP)	Data (payload)	CRC

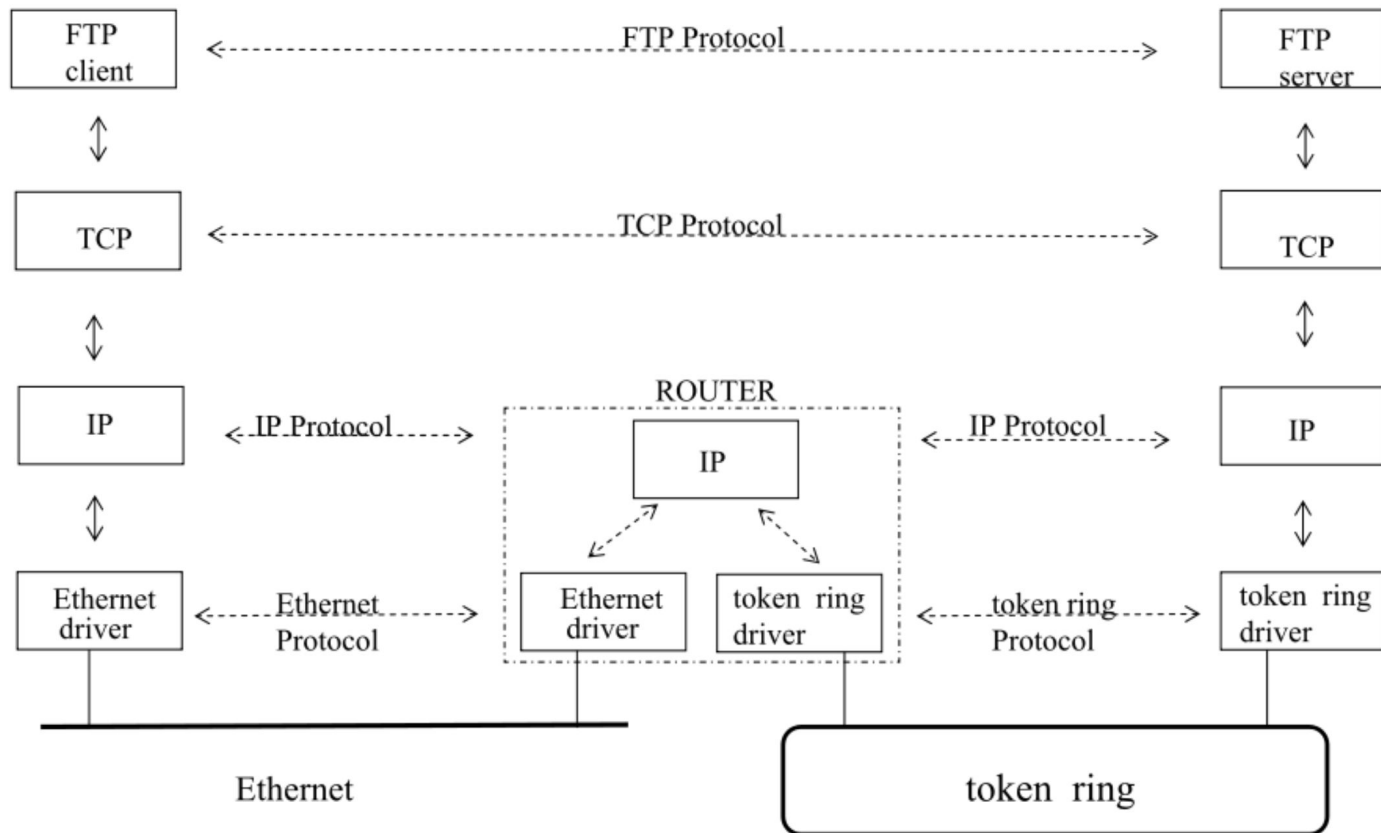
# Формат заголовков IPv4

## IPv4 Packet Header Format

Bit #	0		7	8	15		16	23	24	31
0	Version		IHL	DSCP		ECN	Total Length			
32	Identification					Flags	Fragment Offset			
64	Time to Live			Protocol			Header Checksum			
96	Source IP Address									
128	Destination IP Address									
160	Options (if IHL > 5)									

взято с сайта: [microchipdeveloper.com](http://microchipdeveloper.com)

# Пример клиент-серверного взаимодействия



# TCP vs. UDP, надежность

*Есть две основные проблемы в распределенных системах:*

*2. Строго однократная доставка.*

*1. Гарантированный порядок сообщений.*

*2. Строго однократная доставка.*

Надежность доставки	TCP	UDP
Работа с соединениями	Сессионный протокол	Концепции сессии нет
Повторная передача потерянных сегментов и контроль скорости через механизм окон	Да	Нет
Нумерация сегментов	Да	Нет
Подтверждение доставки	Да	Нет

# TCP vs. UDP, формат заголовков

Относительная сложность TCP и простота UDP находят отражение и в формате заголовков транспортного уровня.

## TCP Segment Header Format

Bit #	0	7	8	15	16	23	24	31
0	Source Port				Destination Port			
32	Sequence Number							
64	Acknowledgment Number							
96	Data Offset	Res	Flags		Window Size			
128	Header and Data Checksum				Urgent Pointer			
160...	Options							

## UDP Datagram Header Format

Bit #	0	7	8	15	16	23	24	31
0	Source Port				Destination Port			
32	Length				Header and Data Checksum			

# TCP vs. UDP, применимость

Характеристики протоколов определяют их применение:

- вы не будете рады, если отправленные на веб-сервер данные формы потеряются по пути и не будут гарантированно доставлены – логично использовать TCP
- напротив, при разговоре по Zoom лучше пожертвовать качеством (потерять часть фрагментов картинки видео или часть спектра частот звука), но сохранить режим реального времени – на помощь придет UDP.

Легковесность UDP позволяет оптимизировать по времени и другие операции. Так классические легковесные **DNS запросы и ответы** используют UDP (ведь потерять эти данные не страшно, а в целом UDP быстрее TCP из-за отсутствия необходимости обмена сегментами для сервисных операций, таких как установление сессии). Впрочем, переключение на TCP возможно специальным флагом, если, например, ответ DNS сервера слишком велик и не вмещается в определенные стандартом для UDP 512 байт.

Еще одной особенностью, связанной скорее с UNIX, а не самими протоколами, – пользовательские дейтаграммы (UDP) можно отправить не обладая правами root, тогда как для TCP или ICMP это обязательно.



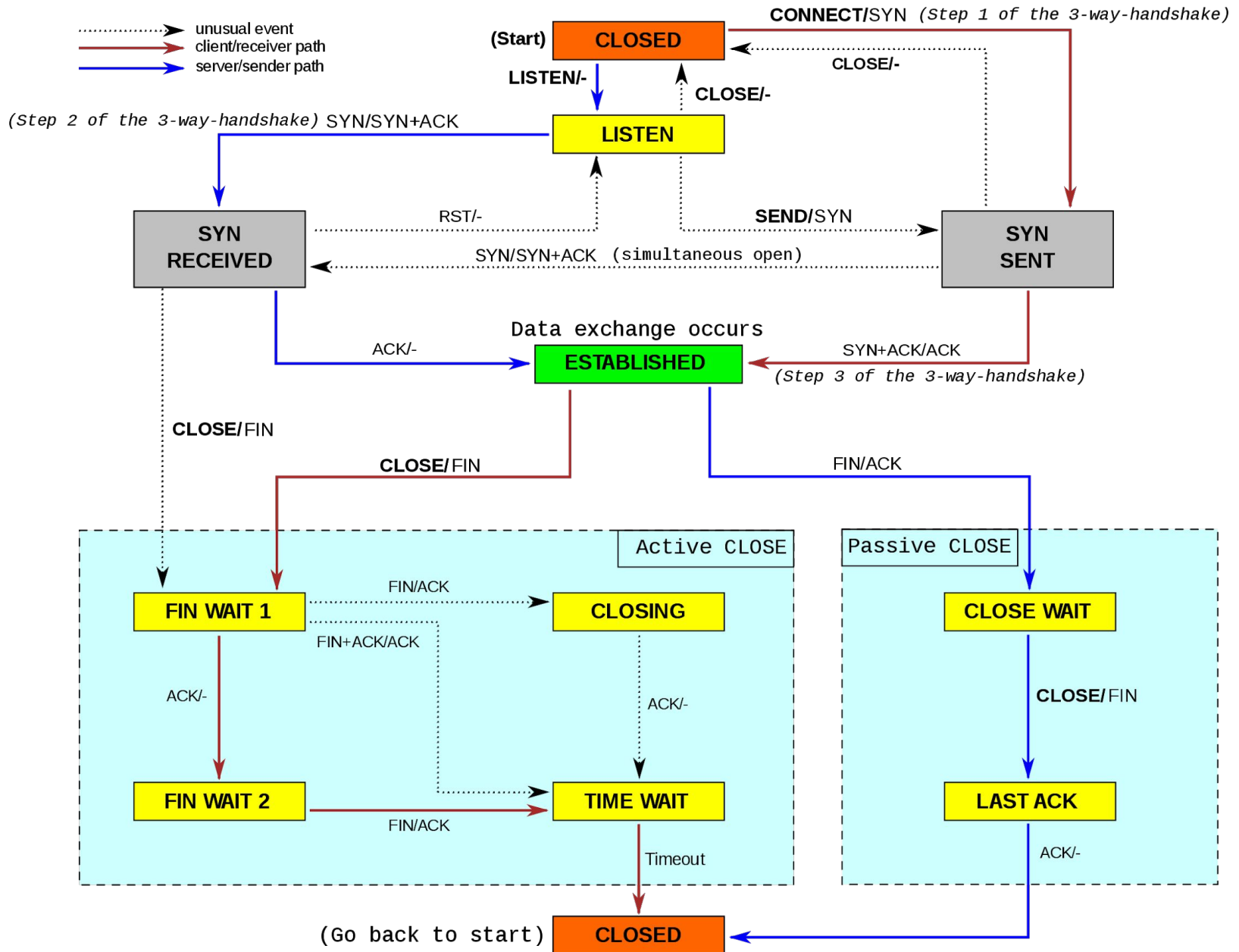
# TCP vs. UDP, протоколы сегодня


Для сегодняшних сетей с большим количеством мобильных беспроводных устройств и “тяжелых” сайтов механизмы контроля и управления потоком TCP могут быть не всегда оптимальны (хотя алгоритмы для этого постоянно совершенствуются и в ядре Linux присутствует целый набор, из которого можно принудительно выбирать алгоритм, если известно что они лучше работают для конкретной конфигурации сети).

```
osboxes@osboxes:~$ grep TCP_CONG /boot/config-$(uname -r)
CONFIG_TCP_CONG_ADVANCED=y
CONFIG_TCP_CONG_BIC=m
CONFIG_TCP_CONG_CUBIC=y
...
osboxes@osboxes:~$ cat /proc/sys/net/ipv4/tcp_congestion_control
cubic
```

Стали появляться альтернативы стандартной связке HTTP 1.1 с TCP, такие как HTTP/3 и QUIC (HTTP, HyperText Transfer Protocol; QUIC созвучно с quick – быстрый, Quick UDP Internet Connections). Они используют UDP на транспортном уровне и берут на себя функции гарантированной доставки и управления сессиями, стремясь их оптимизировать именно для потребления контента в Интернете, а не для сетей общего назначения, для которых предназначался TCP..

# Схема переходов состояния соединения ТСР





# **Некоторые популярные сетевые утилиты**

# ARP, NDP: механизм

Суть протоколов ARP и NDP в части сопоставления IP и MAC адреса в следующем:

- подключенный к L2 сегменту клиент хочет узнать, какому MAC адресу соответствует известный IP адрес. Например, при конфигурации по DHCP ему стал известен IP адрес шлюза, однако нужно все еще узнать его MAC для формирования фрейма с адресатом из другой сети
- для этого используется специально сформированный broadcast широковещательный запрос. Ранее мы обсуждали unicast соединения, в которых только два участника. Этот же запрос предназначен всем участникам L2 сегмента
- особенностью запроса является использование виртуального MAC адреса FF:FF:FF:FF:FF:FF, тогда коммутатор отправляет данный фрейм на все свои порты
- если MAC одного из клиентов, получивший такой ARP запрос, совпадает с его адресом, он формирует ответ, информирующий о соответствии, и отправляет его источнику запроса по ICMP

# ARP, NDP: Linux

Таким образом, клиент получает знание о соответствии IP и MAC. Данная информация кэшируется:

```
osboxes@osboxes:~$ ip neigh show dev enp0s3  
10.0.2.2 lladdr 52:54:00:12:35:02 REACHABLE
```

А так после удаления записи из кэша выглядят ARP request и ARP reply:

```
osboxes@osboxes:~$ sudo ip neigh del 10.0.2.2 dev enp0s3  
osboxes@osboxes:~$ sudo tcpdump -i any arp -nn  
15:28:08.484515 ARP, Request who-has 10.0.2.2 tell 10.0.2.15, length 28  
15:28:08.484643 ARP, Reply 10.0.2.2 is-at 52:54:00:12:35:02, length 46
```

# TTL и traceroute

Одно из важных полей в заголовках IP – TTL или time to live (время жизни). При прохождении IP пакета через роутер, счетчик в этом поле уменьшается на единицу. Пакеты, в которых TTL достиг нуля, уничтожаются. Когда это происходит, роутер формирует сообщение ICMP Time Exceeded.

Данное свойство используется в traceroute:

- формируется пакет с TTL = 1, первый на пути следования роутер уменьшает TTL до 0 и отвечает Time Exceeded (первый хоп),
- traceroute формирует пакет с TTL = 2, он успешно преодолевает первый роутер, и уже второй отвечает Time Exceeded,
- когда traceroute вместе Time Exceeded получает Connection Refused или случайно выбранный порт совпадает с каким-то слушающим сервисом, искомый хост считается достигнутым.

Таким образом, используя src IP и метки времени, traceroute узнает и трассу прохождения пакета, и время его прохождения.

По умолчанию traceroute в Linux использует UDP пакеты со случайным портом, однако ключами можно выбрать TCP (-T) или ICMP режим (-I). mtr вместо разового прохождения трассы собирает статистику.

# ip

Посмотреть интерфейсы:

```
osboxes@osboxes:~$ ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT
   group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode
   DEFAULT group default qlen 1000
    link/ether 08:00:27:f4:39:50 brd ff:ff:ff:ff:ff:ff
```

Посмотреть адреса (для IPv4):

```
osboxes@osboxes:~$ ip -4 address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
   qlen 1000
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group
   default qlen 1000
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic noprefixroute enp0s3
        valid_lft 86328sec preferred_lft 86328sec
```

# telnet, netcat

Установить соединения, эмулируя curl или браузер (получили редирект на HTTPS):

```
osboxes@osboxes:~$ telnet netology.ru 80
Trying 172.67.75.22...
Connected to netology.ru.
Escape character is '^]'.
GET / HTTP/1.1
Host: netology.ru
HTTP/1.1 301 Moved Permanently
```

Завершить сессию: Ctrl + ], Enter, Ctrl + D.

Отличный инструмент диагностики: быстро даст понять по Connection refused что порт не слушается, по Timeout что есть проблемы/ограничения по сетевой доступности и т.д.

```
osboxes@osboxes:~$ nc -l 89 # привилегированные порты ниже 1024
nc: Permission denied
osboxes@osboxes:~$ nc -l 1089
```

Можно даже передавать файлы:

```
osboxes@osboxes:~$ nc -l 9090 > /tmp/bash_from_nc
..

osboxes@osboxes:~$ nc localhost 9090 -q 1 < $(which bash)
..
osboxes@osboxes:~$ md5sum $(which bash) /tmp/bash_from_nc
77506afebd3b7e19e937a678a185b62e /usr/bin/bash
77506afebd3b7e19e937a678a185b62e /tmp/bash_from_nc
```



# ping

ping (ICMP echo request + ICMP echo reply) в представлении не нуждается:

```
osboxes@osboxes:~$ ping -c 1 netology.ru
PING netology.ru (104.26.8.143) 56(84) bytes of data.
64 bytes from 104.26.8.143 (104.26.8.143): icmp_seq=1 ttl=63 time=3.88 ms
--- netology.ru ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 3.877/3.877/3.877/0.000 ms
```

Поэтому попробуем что-то поинтереснее:

```
osboxes@osboxes:~$ ping -M do -s $((2000-28)) -c1 netology.ru
PING netology.ru (172.67.75.22) 1972(2000) bytes of data.
ping: local error: message too long, mtu=1500
--- netology.ru ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms

osboxes@osboxes:~$ ping -M do -s $((1500-28)) -c1 netology.ru
PING netology.ru (172.67.75.22) 1472(1500) bytes of data.
1480 bytes from 172.67.75.22 (172.67.75.22): icmp_seq=1 ttl=63 time=10.1 ms
--- netology.ru ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 10.135/10.135/10.135/0.000 ms
```

Где 28 – длина заголовков IP (20) + ICMP (8).

## SS

socket statistics – наиболее актуальная утилита для сбора информации о сокетах, в частности сетевых сокетах. netstat считается устаревшим, тогда как ss развивается.

```
osboxes@osboxes:~$ ss -4 state listening state unconnected -n | column -t
Netid  State  Recv-Q  Send-Q  Local        Address:Port  Peer  Address:Port  Process
udp    UNCONN  0        0       127.0.0.53%lo:53  0.0.0.0:*
udp    UNCONN  0        0       0.0.0.0:5353    0.0.0.0:*
tcp    LISTEN  0       4096    127.0.0.53%lo:53  0.0.0.0:*
tcp    LISTEN  0        5     127.0.0.1:631    0.0.0.0:*
```


В домашнем задании подумайте, почему udp в состоянии UNCONN = unconnected?  
А так посмотрим установленные TCP соединения не на порт SSH:

```
osboxes@osboxes:~$ ss state connected sport != :ssh -t | column -t
State  Recv-Q  Send-Q  Local        Address:Port  Peer  Address:Port
Process
ESTAB  0        0       10.0.2.15:48668  104.26.8.143:http
```

# Isof

Мы уже рассматривали lsof ранее. Среди прочего он поможет вам узнать, какому процессу принадлежит прослушиваемый порт:

```
osboxes@osboxes:~$ sudo lsof -ni :22
COMMAND  PID    USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
sshd     722    root    3u   IPv4  21337      0t0  TCP *:ssh (LISTEN)
sshd     722    root    4u   IPv6  21348      0t0  TCP *:ssh (LISTEN)
sshd     778    root    4u   IPv4  22479      0t0  TCP 10.0.2.15:ssh->10.0.2.2:52115
(ESTABLISHED)
sshd    1179  osboxes  4u   IPv4  22479      0t0  TCP 10.0.2.15:ssh->10.0.2.2:52115
(ESTABLISHED)
sshd    1538    root    4u   IPv4  30466      0t0  TCP 10.0.2.15:ssh->10.0.2.2:52283
(ESTABLISHED)
sshd    1607  osboxes  4u   IPv4  30466      0t0  TCP 10.0.2.15:ssh->10.0.2.2:52283
(ESTABLISHED)
```



# **Что происходит, когда вы открываете сайт**

# strace, curl

Рассмотрим системные вызовы, относящиеся к сетевой подсистеме (**-e network**), которые делает **curl** при стандартном HTTP запросе:

```
osboxes@osboxes:~$ strace -e network curl -s '127.0.0.1:8000' > /dev/null

socket(AF_INET, SOCK_STREAM, IPPROTO_IP) = 5
setsockopt(5, SOL_TCP, TCP_NODELAY, [1], 4) = 0
setsockopt(5, SOL_SOCKET, SO_KEEPALIVE, [1], 4) = 0
setsockopt(5, SOL_TCP, TCP_KEEPIDLE, [60], 4) = 0
setsockopt(5, SOL_TCP, TCP_KEEPINTVL, [60], 4) = 0
connect(5, {sa_family=AF_INET, sin_port=htons(8000),
sin_addr=inet_addr("127.0.0.1")}, 16) = -1 EINPROGRESS (Operation now in progress)
getsockopt(5, SOL_SOCKET, SO_ERROR, [0], [4]) = 0
getpeername(5, {sa_family=AF_INET, sin_port=htons(8000),
sin_addr=inet_addr("127.0.0.1")}, [128->16]) = 0
getsockname(5, {sa_family=AF_INET, sin_port=htons(32860),
sin_addr=inet_addr("127.0.0.1")}, [128->16]) = 0
sendto(5, "GET / HTTP/1.1\r\nHost: 127.0.0.1:8000\r\nUser-Agent:
curl/7.68.0\r\nAccept: */*\r\n\r\n"..., 78, MSG_NOSIGNAL, NULL, 0) = 78
recvfrom(5, "HTTP/1.0 200 OK\r\nServer: SimpleH"..., 102400, 0, NULL, NULL) = 154
recvfrom(5, "<!DOCTYPE HTML PUBLIC \"-//W3C//D\"...", 722, 0, NULL, NULL) = 722

osboxes@osboxes:~$ echo -en 'GET / HTTP/1.1\r\nHost: 127.0.0.1:8000\r\nUser-Agent:
curl/7.68.0\r\nAccept: */*\r\n\r\n' | wc -c
78
```

# **tcpdump, установление соединения**

[S] – флаг SYN (synchronize, синхронизировать)

[.] – флаг ACK (acknowledge, подтвердить)

Устанавливаем соединение для подтверждения готовности получателем принимать данные:

```
16:43:02.594671 IP 127.0.0.1.32860 > 127.0.0.1.8000: Flags [S], seq 1142656750, win 65495, options [mss 65495,sackOK,TS val 2004484948 ecr 0,nop,wscale 7], length 0
16:43:02.594685 IP 127.0.0.1.8000 > 127.0.0.1.32860: Flags [S.], seq 1882050644, ack 1142656751, win 65483, options [mss 65495,sackOK,TS val 2004484948 ecr 2004484948,nop,wscale 7], length 0
16:43:02.594696 IP 127.0.0.1.32860 > 127.0.0.1.8000: Flags [.], ack 1, win 512, options [nop,nop,TS val 2004484948 ecr 2004484948], length 0
```

**tcpdump** по-умолчанию удобно считает нам смещения относительно случайно выбранных номеров последовательности (sequence number). Добавив флаг -S:

```
16:45:22.716513 IP 127.0.0.1.32862 > 127.0.0.1.8000: Flags [S], seq 3597932188, win 65495, options [mss 65495,sackOK,TS val 2004625070 ecr 0,nop,wscale 7], length 0
16:45:22.716525 IP 127.0.0.1.8000 > 127.0.0.1.32862: Flags [S.], seq 2229564668, ack 3597932189, win 65483, options [mss 65495,sackOK,TS val 2004625070 ecr 2004625070,nop,wscale 7], length 0
16:45:22.716536 IP 127.0.0.1.32862 > 127.0.0.1.8000: Flags [.], ack 2229564669, win 512, options [nop,nop,TS val 2004625070 ecr 2004625070], length 0
```

Обратите внимание на наличие timestamp в сегменте!

# tcpdump, MSS localhost

Обратите внимание на присутствие в дампе максимального размера сегмента, о котором мы говорили ранее, mss:

```
15:57:57.509967 IP 127.0.0.1.32820 > 127.0.0.1.8000: Flags [S], seq 2819152561, win 65495, options [mss 65495,sackOK,TS val 2001779863 ecr 0,nop,wscale 7], length 0
```

Откуда такое огромное значение mss, когда только недавно упоминался максимальный размер в 1500 байт для стандартных и 9000 для jumbo фреймов?

Ответ: этот пример использует соединение на localhost, то есть на самого себя, которое происходит через виртуальный loopback интерфейс – сетевые пакеты не покидают хост. MTU loopback на виртуальной машине с Ubuntu 20.04 – 65536 байт.

Разница 65536 и 65495 – 41 байт, 20 байт на заголовок TCP, 20 байт на заголовок IP.

*Вообще, должно бы было быть 40, но в структуре ядра MSS – [int16](#), поэтому диапазон доступных значений – 0-65535. Стоит отметить, что упомянутые 20 байт на заголовок TCP являются минимальным значением, наличие опций может увеличивать его.*

## tcpdump, MSS external

При установлении соединения с любым внешним сервером, например, netology.ru, сразу можно увидеть:

```
16:50:57.578691 IP 10.0.2.15.48996 > 104.26.9.143.80: Flags [S], seq 2167993563, win 64240, options [mss 1460,sackOK,TS val 2818082966 ecr 0,nop,wscale 7], length 0
```

... что соответствует MTU 1500 на интерфейсе -40 байт на заголовки.



# tcpdump, передача данных

[P] – флаг PSH (push, подтолкнуть)

```
16:45:22.720609 IP 127.0.0.1.32862 > 127.0.0.1.8000: Flags [P.], seq
3597932189:3597932267, ack 2229564669, win 512, options [nop,nop,TS val 2004625074
ecr 2004625070], length 78
16:45:22.720619 IP 127.0.0.1.8000 > 127.0.0.1.32862: Flags [.], ack 3597932267,
win 511, options [nop,nop,TS val 2004625074 ecr 2004625074], length 0

16:45:22.722473 IP 127.0.0.1.8000 > 127.0.0.1.32862: Flags [P.], seq
2229564669:2229564823, ack 3597932267, win 512, options [nop,nop,TS val 2004625076
ecr 2004625074], length 154
16:45:22.722485 IP 127.0.0.1.32862 > 127.0.0.1.8000: Flags [.], ack 2229564823,
win 511, options [nop,nop,TS val 2004625076 ecr 2004625076], length 0
16:45:22.723681 IP 127.0.0.1.8000 > 127.0.0.1.32862: Flags [P.], seq
2229564823:2229565545, ack 3597932267, win 512, options [nop,nop,TS val 2004625077
ecr 2004625076], length 722
16:45:22.723690 IP 127.0.0.1.32862 > 127.0.0.1.8000: Flags [.], ack 2229565545,
win 506, options [nop,nop,TS val 2004625077 ecr 2004625077], length 0
```

length сегмента от клиента серверу совпадает с возвращаемым sendto значением (78).  
length двух сегментов ответа сервера совпадает с recvfrom (154, 722).

## tcpdump, завершение соединения

[F] – флаг FIN (finish, завершить)

```
16:45:22.723690 IP 127.0.0.1.32862 > 127.0.0.1.8000: Flags [.], ack 2229565545, win 506, options [nop,nop,TS val 2004625077 ecr 2004625077], length 0
16:45:22.724578 IP 127.0.0.1.32862 > 127.0.0.1.8000: Flags [F.], seq 3597932267, ack 2229565545, win 512, options [nop,nop,TS val 2004625078 ecr 2004625077], length 0
16:45:22.727592 IP 127.0.0.1.8000 > 127.0.0.1.32862: Flags [F.], seq 2229565545, ack 3597932268, win 512, options [nop,nop,TS val 2004625081 ecr 2004625078], length 0
16:45:22.727614 IP 127.0.0.1.32862 > 127.0.0.1.8000: Flags [F.], ack 2229565546, win 512, options [nop,nop,TS val 2004625081 ecr 2004625081], length 0
```

Это штатный путь завершения соединения. Кроме FIN есть и RST (reset), когда TCP стек ОС по какой-либо причине считает соединение невалидным.

Цветом выделено, что ACK при установлении бита FIN и инициировании завершения соединения относится к прошлому сегменту.

# tshark, запись дампа tcpdump

tcpdump с флагом -w позволит записать файл дампа на файловую систему:

```
root@osboxes:~# tcpdump -i any port 80 -w netology_http.pcap
tcpdump: listening on any, link-type LINUX_SLL (Linux cooked v1), capture size 262144
bytes
^C11 packets captured
11 packets received by filter
0 packets dropped by kernel
```

tshark в командной строке сразу можно посмотреть структуру фреймов:

```
root@osboxes:~# tshark -r netology_http.pcap -V
```

Аналогичная по функциональность утилита GUI: Wireshark.

# Итоги

- Узнали о классификации сетей по типу используемой среды и типу коммутации,
- об истории развития, который привели к появлению двухранговых сетей и двух типов не связанных адресов,
- о стеке протоколов TCP/IP и конкретных его представителях,
- о важных терминах, применяемых при обсуждении сетей,
- о разнице протоколов транспортного уровня TCP и UDP,
- о фазах жизни соединения TCP,
- о том, как можно посмотреть на сетевое взаимодействие средствами операционной системы как на уровне системных вызовов, так и собирая фреймы на интерфейсе с помощью tcpdump.



# Домашнее задание

Давайте посмотрим ваше [домашнее задание](#).

- Вопросы по домашней работе задавайте **в чате** мессенджера Slack.
- Задачи можно сдавать **по частям**.
- Зачёт по домашней работе проставляется после того, как **приняты все задачи**.

**Задавайте вопросы и  
пишите отзыв о лекции!**

**Андрей Вахутинский**