ТЕМА № 6. ТРАНСПОРТЕН СЛОЙ

1. Функции на транспортния слой
2. Протоколи от транспортния слой:

   TCP

   UDP
3. Програми за следене и диагностика на мрежата:
   - ping – Packet Internet Gropper;
   - arp – Address Resolution Protocol and Reverce ARP (rarp);
   - netstat и tpscan;
   - nbstat.
   - програми за конфигуриране на IP: ifconfig и ip;
   - програми за проследяване на маршрути – traceroute и mtr.

ТАНСПОРТЕН СЛОЙ – отговаря за осигуряването на надеждна директна връзка от тип **точка-до-точка (end-to-end)**. За постигането на тази цел се използват механизми за удостоверяване, че данните са пристигнали до своето местоназначение без загуби или повреди.

ВСИЧКИ ПРОТОКОЛИ ОТ ТОЗИ СЛОЙ СА ПРЕДНАЗНАЧЕНИ ЗА ОСИГУРЯВАНЕ НА ВРЪЗКА ОТ ТИП END-TO-END (от край до край).

За транспортния слой не е важно какви данни предава, откъде до къде ги предава – той само предоставя механизъм за предаване на тези данни. Блоковете от данни се разбиват на поредици от байтове (в протоколния стек TCP/IP - **TCP сегменти или UDP дейтаграми**), размерът на които зависи от конкретния протокол.

В този слой има няколко протокола – едни от тях осигуряват само основни транспортни функции като предаване на данните без потвърждение за тяхното получаване (UDP), други осигуряват доставянето на множество пакети от данни в съответната им последователност, мултиплексират предаването на няколко потока от данни, като осигуряват механизми за тяхното управление и гарантират достоверността на получените данни (TCP).

Протоколът UDP е протокол без установяване на връзка (connectionless), докато TCP е протокол с установяване на връзка (connection-oriented) между крайните възли в мрежата.

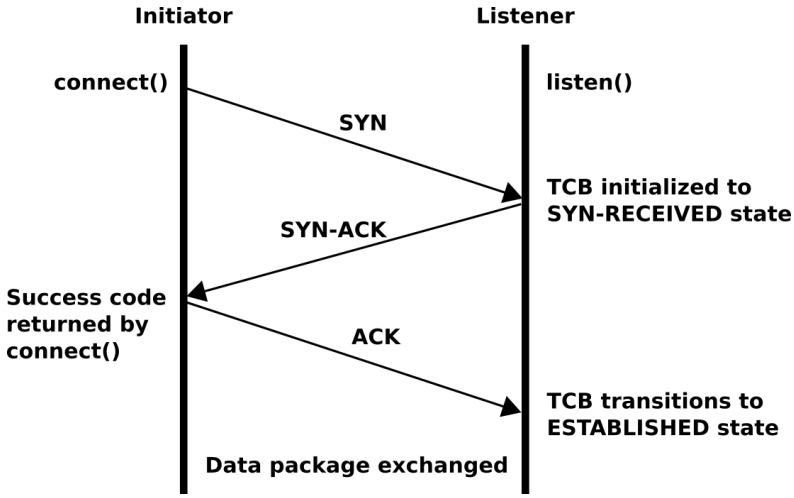
Мултиплексиране на данните – транспортния слой може да управлява едновременно няколко потока от данни, които може да постъпват от различни приложения.

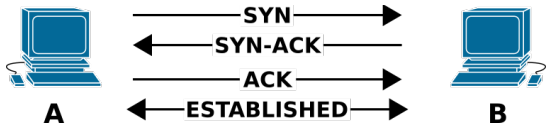
Механизъм за управление на потоците от данни (Flow Control) – позволява регулирането на количеството данни, предавани от един възел към друг.

Протоколите от транспортния слой често изпълняват и функция за контрол на доставянето на данните, изисквайки от приемащия данните възел да изпраща към предаващия възел потвърждения за получаването на данните.

TCP – TRANSMISSION CONTROL PROTOCOL

TCP е протокол, ориентиран към създаването и използването на връзки (тип end-to-end). Той установява сесия между двата общуващи си възела от мрежата, преди да започне да изпраща данни. За установяване на сесията се използват съобщения за потвърждаване и отговор – three way handshaking – syn, syn-ack and ack сегменти.





фиг. 1 Схема на трикратното ръкостискане
при установяване на TCP сесия

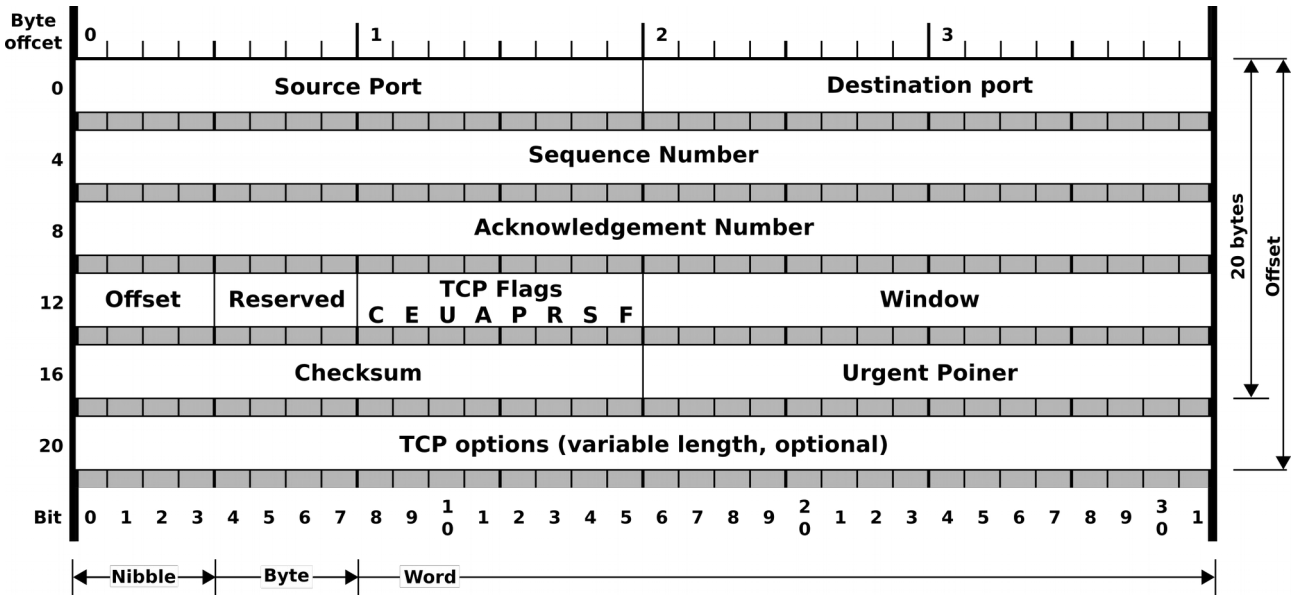
Последователността при установяване на една TCP сесия между два възела от мрежата е следната:

1. Възелът A изпраща към възел B TCP синхронизиращ пакет SYN;
2. Възелът B го получава;
3. Възелът B връща към възела A потвърждаващ TCP пакет SYN-ACK;
4. Възелът A получава този потвърждаващ пакет от възела B;
5. Възелът A **изпраща** TCP пакет **ACK** към възел B;

6. Възелът B получава този пакет ACK;
7. **TCP socket connection is ESTABLISHED.**

След установяването на връзката се извършва проверка за грешки и тяхното коригиране като данните се разделят на пакети.

Пакетите SYN и ACK се отбелязват с допълнителен SYN или ACK бит в заглавната част на пакета.



Фиг. 2 Заглавна част на TCP пакет

Към всеки пакет се добавя информация за неговата последователност, така че отделните части от съобщението да могат да се сглобят в обратен ред при тяхното получаване. Тази информация позволява на приемащия възел да открие дали няма липсващи пакети. Всичко това прави TCP протокола по-надежден от UDP, но на цената на по-ниската му производителност.

Datagram (RFC 1594 **–** FYI on Questions and Answers или RFC 2664 – FYI on Questions and Answers) – независима единица данни, носеща достатъчно информация за маршрутизирането ѝ от нейния източник до нейното местоназначение без да се разчита на предишен обмен между тези възли и транспортиращата мрежа (между тях).

Packet (пакет) – единица от данни, която е част от група последователни единици или „парчета“, на които е разбито дадено съобщение.

Пакетите може да бъдат доставени по различни маршрути в мрежата до крайното си местоназначение, където се сглобяват в обратен ред.

Термините **дейтаграма** и **пакет** понякога се използват взаимозаменяемо.

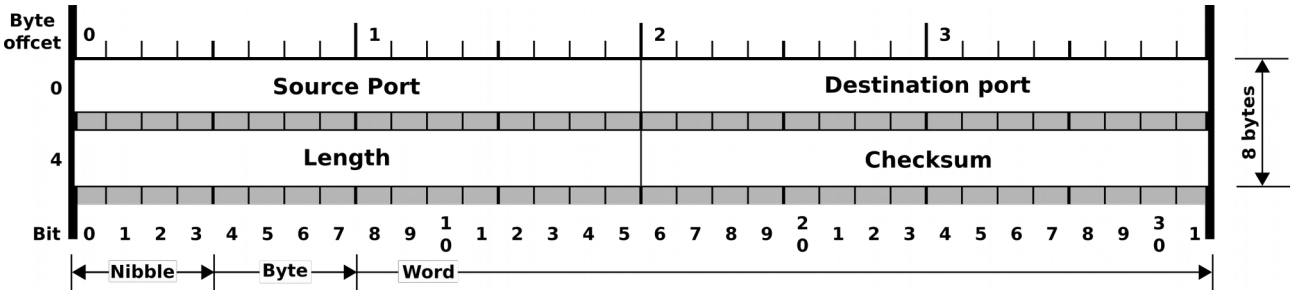
Терминът дейтаграма се използва за описване на по-простите и неподредени единици от данни, предавани по UDP протокол.

UDP – USER DATAGRAM PROTOCOL

За разлика от TCP, той е протокол, който не изисква установяването на връзка между крайните възли в мрежата (connectionless protocol) – между изпращащият възел и възела, за който са предназначени изпращаните данни.

Той не поставя последователни номера (sequences numbers) на пакетите, които изпраща, което го прави подходящ за изпращане на малки съобщения, които могат да се съберат в един пакет. UDP не следи какво изпраща и не изисква потвърждение дали е получено. Но все пак генерира контролна сума на изпратените данни за да гарантира, че те са пристигнали неповредени.

По тези причини (не генерира последователност на пакетите и не проверява за грешки), UDP протоколът е бърз. Неговите заглавни части са по-прости от тези на TCP протокола.



Фиг. 3 Заглавна част на UDP пакет

Протоколи, които използват UDP протокол:

- RIP – Routing Information Protocol;
- TFTP – Trivial File Transfer Protocol;
- DNS – Data Name Service.

Кой от двата протокола (TCP или UDP) ще се използва за предаване на данните се определя от характера и нуждите на това предаване на данни.

TCP се използва, когато най-важна е надеждността на връзката, а UDP – когато с най-висок приоритет за нас е производителността (скоростта) на връзката.

Всеки TCP сегмент се разделя на две части:

1. заглавна част (header) – с фиксиран размер от 20 байта;

2. данни – с максимална дължина 65535 байта.

Всеки краен възел на връзката се идентифицира с комбинацията от IP адрес и номер на използван порт (така наречения **транспортен адрес** от упражнението за NAT). Номера на порта се определя от съответната програма от приложния слой, която използва тази връзка.

Socket – комбинацията от IP адреса и номер на порта (транспортен адрес) на двата възела от мрежата, участващи във връзката.

Всеки TCP сегмент съдържа номерата на портове на източника и на приемника → те определят за коя приложна програма е предназначен съответния сегмент.

Транспортния адрес на приемника и този на източника образуват уникална комбинация, идентифицираща TCP връзката. Един socket (гнездо) може да се използва едновременно от няколко TCP връзки.

Портове с номера от 1 до 1023 се наричат **добре известни портове**. Портовете от 1024 до 65535 са свободни за използване.

Помощни програми за следене и диагностика

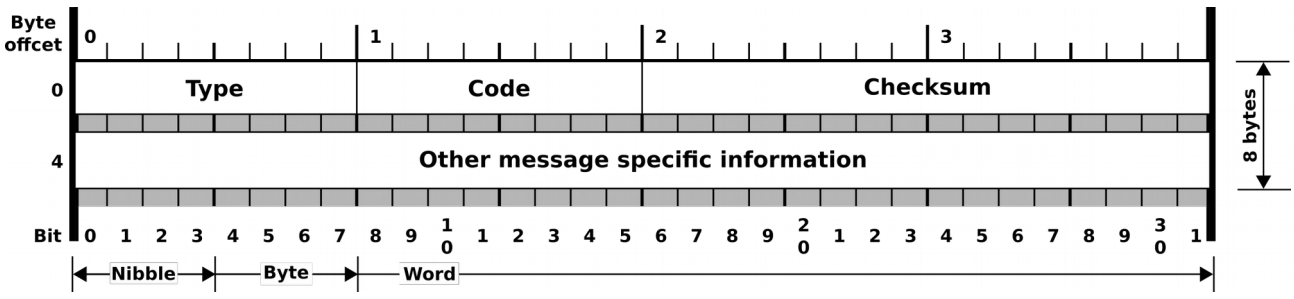
ping – проста, но полезна програма за работа от команден ред. Включена е в повечето реализации на TCP/IP стека. Може да се използва както с името на възела, така и с неговия IP адрес.

Командата ping изпраща ICMP echo request пакет към възела, чийто адрес сме посочили в нея:

```
# ping 62.44.96.142

# ping google.bg
```

Възелът, който получи такъв ICMP echo request пакет трябва да отговори с echo reply пакет.



Фиг. 4 Заглавна част на ICMP пакет

Командата **nslookup** връща IP адреса на дадено име на възел или името на възела на въведен IP адрес:

```
[nick@sakurajima ~]$ nslookup google.bg
```

```
Server:        95.87.194.5
Address:  95.87.194.5#53

Non-authoritative answer:
Name:     google.bg
Address: 216.58.211.35
```

или

```
[nick@sakurajima ~]$ nslookup 62.44.96.142
Server:        95.87.194.5
Address:  95.87.194.5#53

Non-authoritative answer:
142.96.44.62.in-addr.arpa     name = ns.uni-sofia.bg.
```

ARP и RARP

ARP – отнася се до самия протокол (ARP – Address Resolution Protocol) и до самата команда, използвана за разглеждане и манипулиране на ARP кеша.

Протоколът ARP е средството, с което възлите в мрежата съпоставят логическите (IP) адреси с физическите (MAC) адреси. Протоколът ARP изгражда и поддържа таблица, наричана ARP кеш, в която се съдържат тези съпоставяния. Протоколът RARP се използва от машина, която не знае собствения си IP адрес, за да получи информация за него на базата на своя MAC адрес.

Програмата arp може да се използва за разглеждане и промяна на съпоставянията между IP и MAC адресите.

`# arp -s address hw_address` – въвежда ново съответствие MAC/IP адрес

`# arp -d address` – изтрива съответствието IP/MAC адрес.

Повече настройки за използване с командата arp могат да се видят на man страницата на командата:

```
# man arp
```

netstat – команда, показва информация за TCP/IP връзките и протокола, използван за тези връзките:

netstat

netstat | grep TCP

```
tcp        0      0 sakurajima:17595        88.87.13.191:51714      ESTABLISHED
tcp        0      0 sakurajima:17595        212.5.152.1:53766       ESTABLISHED
tcp        0      0 sakurajima:17595        212.5.152.48:51879      ESTABLISHED
tcp        0      0 sakurajima:45774        db3msgr5012709.ga:https ESTABLISHED
tcp        0      0 sakurajima:53157        xmpp.org:xmpp-client    ESTABLISHED
tcp        0      0 sakurajima:17595        212-5-158-212.btc:35809 TIME_WAIT
tcp       32      0 sakurajima:51365        6-55-236-85.rev.c:https CLOSE_WAIT
tcp        0      0 sakurajima:48331        server-54-192-96-:https ESTABLISHED
tcp        0      0 sakurajima:55749        kyufte.mnet:xmpp-client ESTABLISHED
tcp        0      0 sakurajima:43488        157.56.116.204:12350    ESTABLISHED
tcp        0    587 sakurajima:17595        212-5-158-212.btc:33324 ESTABLISHED
tcp        0      0 sakurajima:17595        212.5.152.48:51759      ESTABLISHED
tcp        0      0 sakurajima:17595        87-104-159-150-dy:35073 ESTABLISHED
tcp        0      0 sakurajima:50637        213.199.179.166:40016   ESTABLISHED
tcp        0      0 sakurajima:17595        188-254-235-254.s:51868 ESTABLISHED
tcp        0      0 sakurajima:17595        92.247.150.1:raw-serial ESTABLISHED
```

netstat ни показва списък на връзките, които са активни в момента.

netstat -s – показва мрежова статистика

[root@sakurajima ~]# netstat -s

```
Ip:
    313526 total packets received
    157 with invalid addresses
    0 forwarded
    0 incoming packets discarded
    301849 incoming packets delivered
    241589 requests sent out
    24 outgoing packets dropped
Icmp:
    217 ICMP messages received
    0 input ICMP message failed.
    ICMP input histogram:
        destination unreachable: 217
    125 ICMP messages sent
    0 ICMP messages failed
    ICMP output histogram:
        destination unreachable: 125
IcmpMsg:
        InType3: 217
        OutType3: 125
Tcp:
    2663 active connections openings
    3025 passive connection openings
    35 failed connection attempts
```

```
    595 connection resets received
    7 connections established
    201424 segments received
    201507 segments send out
    2200 segments retransmited
    145 bad segments received.
    720 resets sent
Udp:
    129245 packets received
    52 packets to unknown port received.
    0 packet receive errors
    38005 packets sent
    0 receive buffer errors
    0 send buffer errors
UdpLite:
TcpExt:
    10 invalid SYN cookies received
    28 resets received for embryonic SYN_RECV sockets
    866 TCP sockets finished time wait in fast timer
    8260 delayed acks sent
    4 delayed acks further delayed because of locked socket
    Quick ack mode was activated 498 times
    1 SYNs to LISTEN sockets dropped
    88 packets directly queued to recvmsg prequeue.
    1448 bytes directly in process context from backlog
    28761 bytes directly received in process context from prequeue
    104730 packet headers predicted
    15 packets header predicted and directly queued to user
    27655 acknowledgments not containing data payload received
    30870 predicted acknowledgments
    247 times recovered from packet loss by selective acknowledgements
    7 congestion windows recovered without slow start by DSACK
    22 congestion windows recovered without slow start after partial ack
    1 timeouts after SACK recovery
    6 timeouts in loss state
    247 fast retransmits
    19 forward retransmits
    23 retransmits in slow start
    537 other TCP timeouts
    TCPLossProbes: 1075
    TCPLossProbeRecovery: 493
    33 SACK retransmits failed
    647 DSACKs sent for old packets
    185 DSACKs received
    184 connections reset due to unexpected data
    204 connections reset due to early user close
    14 connections aborted due to timeout
    TCPDSACKIgnoredOld: 5
    TCPDSACKIgnoredNoUndo: 68
```

```
    TCPSpuriousRTOs: 7
    TCPSackShiftFallback: 277
    TCPRcvCoalesce: 23955
    TCPOFOQueue: 999
    TCPChallengeACK: 147
    TCPSYNChallenge: 145
    TCPSpuriousRtxHostQueues: 69
    TCPAutoCorking: 14017
    TCPWantZeroWindowAdv: 30
    TCPSynRetrans: 543
    TCPOrigDataSent: 90208
IpExt:
    InMcastPkts: 36275
    OutMcastPkts: 293
    InBcastPkts: 5347
    OutBcastPkts: 4
    InOctets: 467732627
    OutOctets: 40231400
    InMcastOctets: 11493421
    OutMcastOctets: 42982
    InBcastOctets: 477160
    OutBcastOctets: 196
    InNoECTPkts: 511090
    InECT1Pkts: 11
    InECT0Pkts: 107
    InCEPkts: 1
```

ss –

ifconfig – извежда информация за TCP/IP конфигурацията на конкретен възел от мрежата;

ip –

traceroute – за проследяване на маршрута, по който даден пакет минава от източника до приемника.

Следене на трафика

Два типа начина за събиране на информация:

1. **monitoring** – не събира самите пакети, а само статистика за трафика през даден мрежов интерфейс;
2. **capturing** – залавянето на самите пакети позволява следенето на същите тези статистики, но със запазване на пакетите за последващ анализ – декодиране на заглавните части, добавени от различните протоколи.

Предназначение на програмите за следене на трафик:

1. Откриване и диагностициране на проблеми в мрежата;
2. Оценка на натовареността на мрежата и разпределяне на трафика по протоколите

Програми – wireshark и tshark; tcpdump