20. Транспортни протоколи TCP и UDP

TCP и UDP

Двата основни протокола на транспортния слой в TCP/IP модела са Transmission Control Protocol (TCP) и User Datagram Protocol (UDP).

И двата управляват комуникациите между приложения, работещи на компютри в Мрежата и са от типа “край до край” (end to end).

Разликите между двта са във функциите, които реализират.

UDP е по-опростен, осигурява ненадеждно обслужване с неустановена връзка (connectionless), дефиниран в RFC 768.

Предимство е ниското закъснение.

TCP и UDP

Протоколните единици в UDP се наричат дейтаграми, за които подобно на IP пакетите се полагат максимални усилия за доставяне - "best effort".

Типични приложения на UDP са:

- Domain Name System (DNS)
- Video Streaming
- Voice (Video) over IP (VoIP); Video over IP
- мониторинг и управление на мрежите (SNMP)
- опростен пренос на двоични файлове (Trivial FTP).

TCP и UDP

TCP е протокол, който предоставя надеждно обслужване с установена връзка (connection-oriented), дефиниран в RFC 793.

TCP внася допълнително закъснение заради функциите по надеждност, спазване на реда на подаване на единиците с данни (сегменти) и управление на потока.

Всеки TCP сегмент има 20 байта служебна информация, с които опакова приложните данни, докато при UDP дейтаграмите те са само 8 байта.

TCP и UDP

Типични приложения на TCP са:
- Web браузъри и сървъри
- E-mail
- сигурен обмен на файлове (FTP)

TCP сегменти и UDP дейтаграми

TCP Segment

Bit (0)	Bit (15)	Bit (16)	Bit (31)
Source Port (16)		Destination Port (16)	
Sequence Number (32)			
Acknowledgement Number (32)			
Header Length (4) Reserved (6) Code Bits (6)		Window (16)	
Checksum (16)		Urgent (16)	
Options (0 or 32 if any)			
APPLICATION LAYER DATA (Size varies)			

20 Bytes

UDP Datagram

Bit (0)	Bit (15)	Bit (16)	Bit (31)
Source Port (16)		Destination Port (16)	
Length (16)		Checksum (16)	
APPLICATION LAYER DATA (Size varies)			

8 Bytes

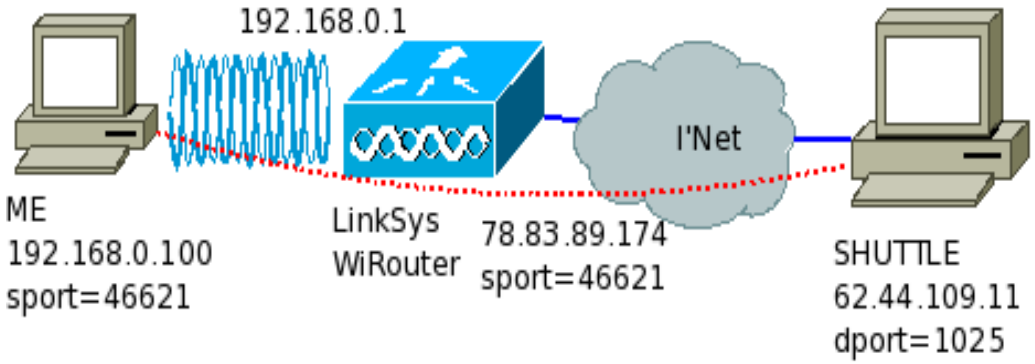
Адресиране с портове. Идентифициране на “разговорите”

TCP и UDP базираните услуги следят комуникациите между различни приложения в Мрежата.

За да разпознаят сегментите и дейтаграмите на всяко приложение, и TCP сегментите, и UDP дейтаграмите имат полета в заглавната част, които уникално идентифицират тези приложения - номерата на портовете.

По-конкретно, порта-източник и порта-местоназначение: source port и destination port.

Установена TCP сесия (SSH)



source port (**sport**) е номера на комуникацията, свързана с приложението – инициатор на “разговора” (сесия).

destination port (**dport**) е номера на комуникацията, свързана с приложението – дестинация, работещо върху отдалечения хост.

conntrack

[root@shuttle]#less /proc/net/ip_conntrack

```
tcp      6 432000 ESTABLISHED src=78.83.89.174
dst=62.44.109.11 sport=46621 dport=1025 packets=243
bytes=20821 src=62.44.109.11 dst=78.83.89.174
sport=1025 dport=46621 packets=156 bytes=28980
[ASSURED] mark=0 secmark=0 use=2
```

[root@me]# less /proc/net/nf_conntrack

```
ipv4     2 tcp      6 431761 ESTABLISHED
src=192.168.0.100 dst=62.44.109.11 sport=46621
dport=1025 packets=262 bytes=22097 src=62.44.109.11
dst=192.168.0.100 sport=1025 dport=46621
packets=167 bytes=30432 [ASSURED] mark=0 secmark=0
use=2
```

destination port

Номерата на портове се присвояват по различни начини, в зависимост от това дали съобщението е заявка или отговор.

Процесите на сървър са със статични номера, а клиентите динамично избират номер на порт при всяка сесия.

Клиент изпраща заявка до сървър: destination port е номер на порт, присвоен на процеса (демона) на услугата, стратирана на отдалечения хост.

Destination port

Клиентският софтуер трябва да знае номера на този порт. Той се конфигурира по подразбиране или ръчно. (*напр. SSH=22, но може 1025*)

Например, web браузър прави заявка към web сървър. Използва TCP и порт 80, ако не е дефинирано нещо друго.

TCP port 80 е по подразбиране присвоен на web сървърите. Повечето известни приложения имат номера на портове по подразбиране.

source port

source port в заглавието на сегмента/дейтаграмата, съдържащ заявката на клиента, е произволно генериран от номера на портове, по-големи от 1023.

source port играе ролята на обратен адрес за приложението, заявяващо услугата.

Комбинацията от номера на порт на транспортния слой и IP адреса на мрежовия (IP:port No) идентифицира конкретен процес, работещ на даден хост. Тази комбинация се нарича сокет (socket).

source port

Двойката (IP:port No) на източник и местоназначение идентифицира конкретна сесия между два хоста.

Например, HTTP заявка за web страница, изпратена към web сървър (port 80) с IPv4 адрес 192.168.1.20 е насочена към сокет 192.168.1.20:80.

Ако web браузъра е на хост с IP: 192.168.100.48 и динамично му е присвоен порт 49152, сокет, където трябва да бъде “свалена” web страницата ще е 192.168.100.48:49152.

Видове портове

IANA (iana.org) се занимава и с присвояване на портове.

Различните видове номера:

Добре известни - Well Known Ports (0 - 1023). Резервирани за популярни услуги и приложения – HTTP, POP3/SMTP, Telnet и др.

Регистрирани - Registered Ports (1024 - 49151). Тези номера се присвояват на потребиъелски процеси и приложения.Ако не се използват в момента, могат да се използват динамично от клиентските процеси.

Видове портове

Динамични или частни портове (49152 - 65535). Известни и като Ephemeral (краткотрайни) портове. Обикновено се присвояват динамично на клиентски приложения, инициращи сесия. Услугите не използват такива портове (с изключение на peer-to-peer file sharing програми).

Номера, използвани и в TCP, и в UDP. За постигане на по-висока производителност някои приложения в даден момент се опират на TCP, в друг - на UDP.

Видове портове

Например, ниското закъснение, внасяно от UDP, позволява на DNS да обслужва бързо много клиентски заявки. Но понякога изпращането на заявената информация може да изисква надеждността на TCP.

Т.е DNS използва известния порт 53 и с двата протокола.

Списък с номерата на портовете можете да намерите на: http://www.iana.org/assignments/port-numbers

TCP портове

Port Number Range	Port Group
0 to 1023	Well Known (Contact) Ports
1024 to 49151	Registered Ports
49152 to 65535	Private and/or Dynamic Ports

Registered TCP Ports:
1863 MSN Messenger
2000 Cisco SCCP (VoIP)
8008 Alternate HTTP
8080 Alternate HTTP

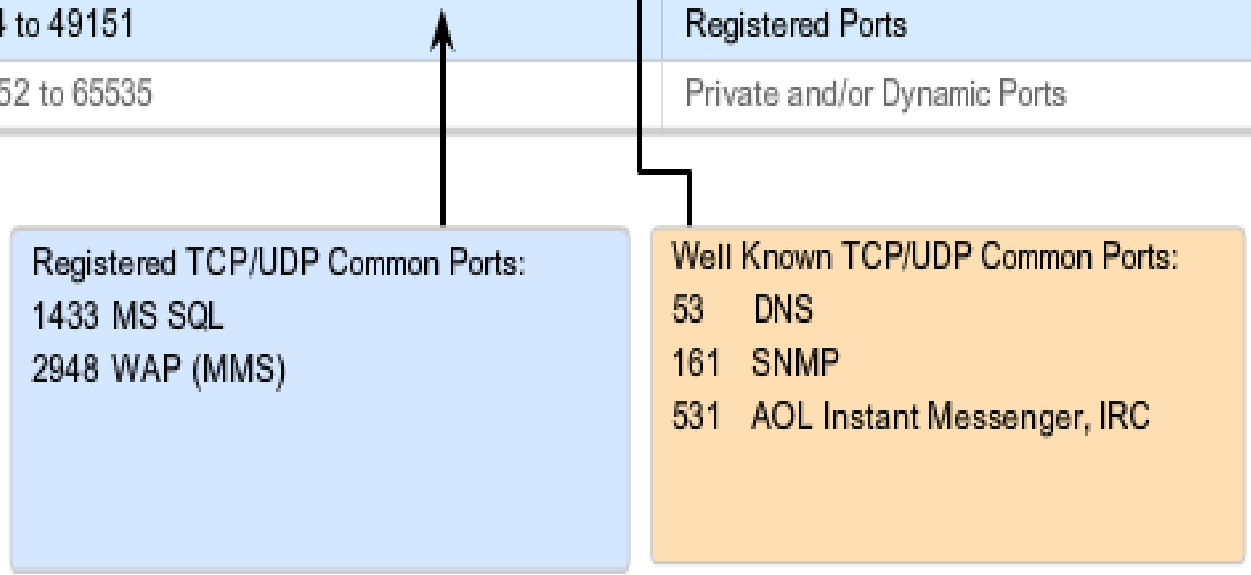
Well Known TCP Ports:
21 FTP
23 Telnet
25 SMTP
80 HTTP
110 POP3
194 Internet Relay Chat (IRC)
443 Secure HTTP (HTTPS)

UDP портове

Port Number Range	Port Group
0 to 1023	Well Known (Contact) Ports
1024 to 49151	Registered Ports
49152 to 65535	Private and/or Dynamic Ports

TCP/UDP портове

Port Number Range	Port Group
0 to 1023	Well Known (Contact) Ports
1024 to 49151	Registered Ports
49152 to 65535	Private and/or Dynamic Ports



Сегментиране и възстановяване

TCP и UDP се справят със сегментирането по различен начин.

В TCP в заглавната част на всеки сегмент се съдържа пореден номер (sequence number). Благодарение на него в Транспортния слой на дестинацията се възстановява реда, по който сегментите са били предадени.

Услугите в UDP също следят сесиите между приложенията, но не и реда, по който се предава информацията, и поддържането на връзката.

Сегментиране и възстановяване

В UDP заглавието липсва последователен номер (sequence number). UDP е с по-опростена структура и внася по-малко излишна за потребителя служебна информация (overhead) от TCP.

Затова осигурява по-бърз пренос на данните.

Но приложенията, които се базират на UDP, трябва да се съобразяват с факта, че е възможно данните в различен ред от този, по който са били предадени.

Transmission Control Protocol

Transmission Control Protocol (**TCP**) осигурява надеждно, подредено доставяне на потока от байтове от програма на един компютър до програма на друг компютър.

TCP следи за размера на съобщенията, скоростта на обмен и натоварването на мрежовия трафик.

Приложимост на TCP

TCP се ползва от най-популярните Internet приложения - WWW, E-mail, FTP, SSH, Telnet и някои групови медийни приложения (streaming media).

TCP е оптимизиран за доставяне на съобщения без грешка в съдържанието, но не и за гарантирано време на доставяне.

TCP понякога внася големи закъснения (в порядъка на секунди) заради подреждането на пристигналите пакети и повторни предавания.

Затова не е подходящ за приложения в реално време като Voice over IP (VoIP) и Video over IP. За тях се препоръчва Real-time Transport Protocol (RTP), който работи върху User Datagram Protocol (UDP).

TCP и IP

Между “подател” и “получател” в Интернет се разменят *съобщения*.

Докато IP се грижи за действителното доставяне на данните, TCP следи отделните единици от данни - *сегменти*, на които е разбито съобщението с цел ефективна маршрутизация.

Например, HTML файл се праща от Web сървър. TCP софтуерът на този сървър разделя последователността от байтове във файла на сегменти, които препраща поотделно към IP софтуера – мрежовия слой.

Мрежовият слой опакова всеки TCP сегмент в IP пакет.

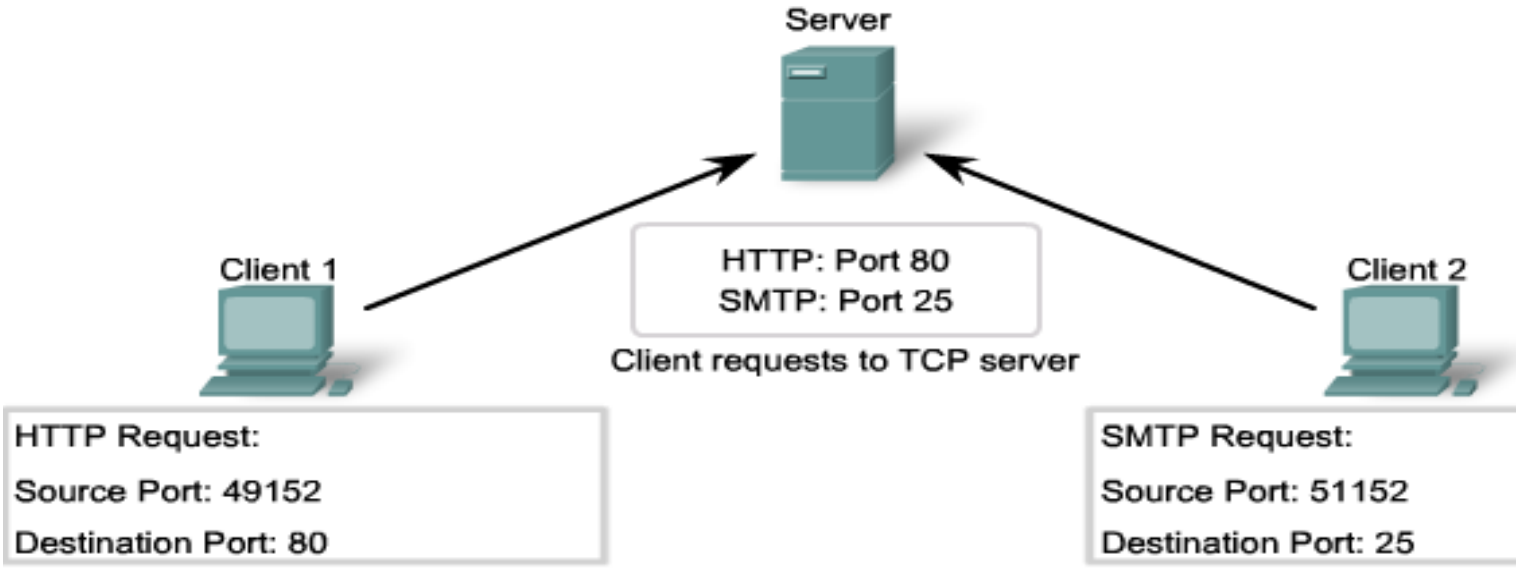
TCP и IP

Всички IP пакети, които произхождат от горния HTML файл са с един и същ адрес на получател, но те могат да минат по различни пътища през Мрежата.

Клиентската програма на компютъра-получател, по-точно TCP софтуера (транспортния слой), възстановява оригиналното подреждане на сегментите, като гарантира, че са получени без грешка, и ги препраща към приложната програма.

TCP процеси на сървъра

На една и същ машина, сървър, не е възможно да има две услуги, присвоени на един и същ номер на порт.



Структура на TCP сегмента

Bit offset	Bits 0–3	4–7	8–15								16–31
0	Source port										Destination port
32	Sequence number										
64	Acknowledgment number										
96	Data offset	Reserved	CWR	ECE	URG	ACK	PSH	RST	SYN	FIN	Window Size
128	Checksum										Urgent pointer
160	Options (optional)										
160/192+	Data										

Структура на TCP сегмента

TCP сегментът се състои от две части: заглавие (header) и данни (data).

Заглавната част на TCP сегмента се състои от 11 полета, от които 10 са задължителни. 11-то е "options".

Source port (16 бита) – номер на изпращащ порт

Destination port (16 бита) – номер на получаващ порт

Sequence number (32 бита) – двуяка роля:

- ако флаг SYN е вдигнат, това е първоначалният номер. Последователният номер на първия байт ще бъде именно този sequence number + 1.
- ако флаг SYN не е вдигнат, това е sequence number на първия байт с данни.

Структура на TCP сегмента

Полето Acknowledgement number е номерът на първия байт данни, който се очаква да се получи със следващия сегмент, изпратен от другия край на TCP връзката. Например, при успешно получаване на сегмент с размер на полето данни 500 байта и пореден номер на началния му байт n, към източника на този сегмент се изпраща TCP сегмент, в който потвърждението е с номер n+501.

Полето TCP header length е 4-битово и определя дължината на заглавната част на TCP сегмента в 32-битови думи. То е задължително, тъй като полето за опции е с променлива дължина. Фактически с това поле се определя началото на полето Data в рамките на TCP сегмента.

Структура на TCP сегмента

Заглавната част на TCP сегмента съдържа и 6 еднобитови флага:

URG – валиден е указателят за спешни данни (Urgent pointer). Установяването на този флаг означава, че трябва да се преустанови обработката на получените данни, докато не се обработят байтовете, към които сочи указателят за спешни данни;

ACK – валиден е номерът на потвърждение, записан в полето Acknowledgement number на заглавната част;

PSH – при активирането на този флаг, програмните модули управляващи транспортния слой на източника и на приемника трябва да изпратят незабавно наличните данни колкото е възможно по-бързо към техния получател, т.е. източникът не изчаква да се съберат данните за образуване на пълен сегмент с избрания размер и съответно получателят не чака запълването на приемния буфер;

Структура на TCP сегмента

RST – сегмент, в който е установен този флаг, служи за прекратяване на TCP връзката. Използва се в случаите, когато връзката е нарушена (например, поради повреда в хоста) или когато се отхвърля невалиден сегмент или се отказва опит за установяване на връзка;

SYN – сегмент с установен флаг SYN се използва при установяване на TCP връзка и за изпращане на началния номер, от който ще бъдат номерирани байтовете на изходящия информационен поток;

FIN – сегмент, в който е установен този флаг, означава, че изпращачът прекратява предаването на данни. Поради двупосочния характер на информационния обмен това не означава, че TCP връзката е прекратена.

Flow control. Sliding Window.

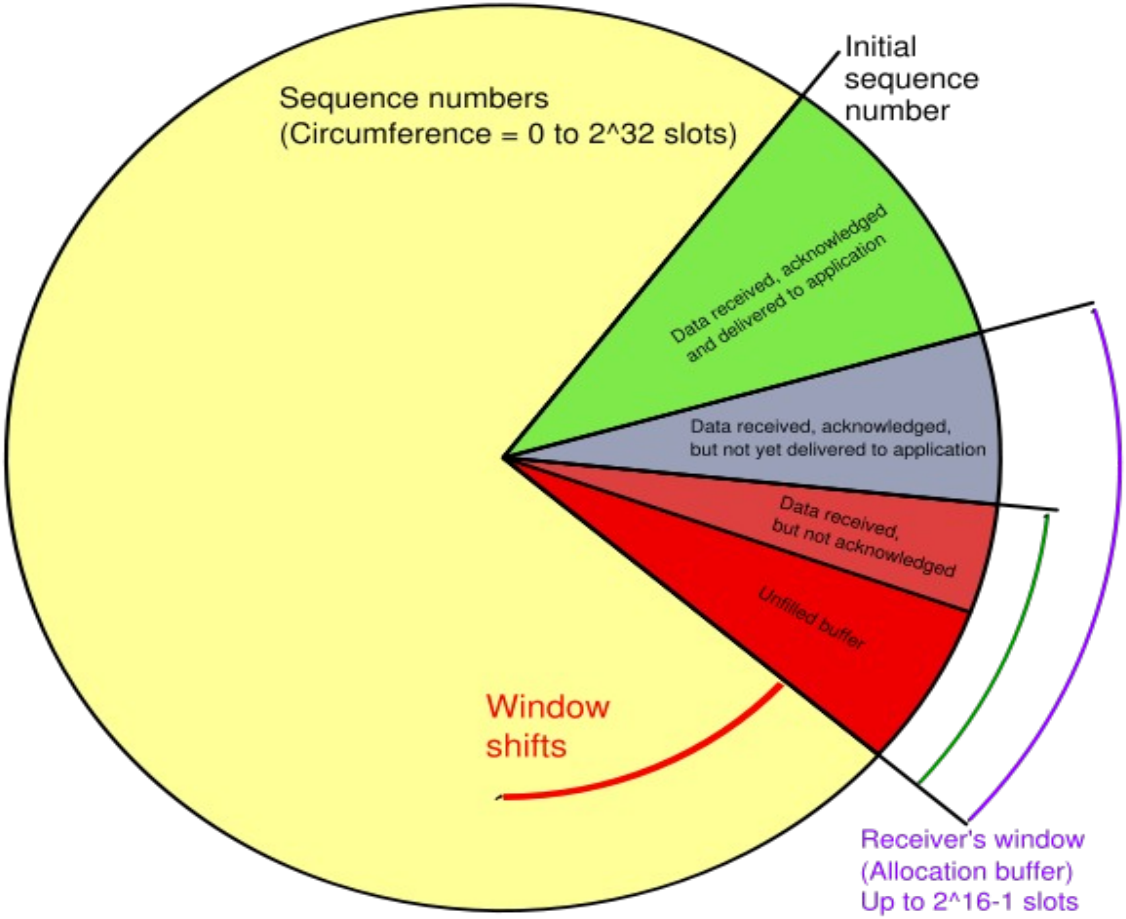
Полето **Window size** определя темпа на информационния обмен от гледна точка на получателя на информационния поток. Това е т.нар. **Flow Control**.

Стойността на прозореца указва на отсрещната страна колко байта могат да бъдат изпратени и съответно приети ***без препълване на входия буфер*** след последния потвърден номер на байт.

При получаване на данни, размерът на прозореца намалява. Ако той стане равен на 0, изпращачът трябва да престане да предава данни.

След като данните се обработят, получателят увеличава размера на своя прозорец, което означава, че е готов да получава нови данни.

sliding window



Структура на TCP сегмента

Полето Checksum се изчислява върху целия TCP сегмент. При неговото изчисляване участват и някои полета от заглавната част на IP дейтаграмата, в която е опакован сегмента.

Полето Options на заглавната част на TCP сегмента е предназначено да предостави допълнителни възможности за управление на обмена, които не се осигуряват от останалите полета на заглавието. Най-важната възможност е указване на **максимална дължина на сегмента** (**MSS**).

TCP Jumbograms

В TCP header няма поле за дължина, т.е няма какво да ограничи дължината на TCP пакета (RFC 2675).

Максималната дължина на сегмента (MSS), която се уговаря в началото на сесията, определя максимална дължина на TCP пакета.

Ако (MTU - 60) >= 65535, MSS = 65535.

При получаване на MSS = 65535, приема се за ∞. (MSS = Path MTU - 60).

UDP Jumbograms

16-бит поле Length в UDP header ограничава дължината на UDP пакета (UDP header + данни) <= 65535 октета.

За да се справим с това ограничение (RFC 2675):

UDP пакети > 65,535 октета се изпращат като UDP Length = 0, получателят извлича фактическата дължина на UDP пакета от IPv6 payload length.

Установяване и терминиране на TCP съединение

Когато два хоста комуникират по TCP, първо трябва да се оформи съединение, преди да започне обмена на данни.

След приключване на комуникацията сесиите се затварят - терминират.

Всичките тези процедури реализират функциите по надеждност в TCP.

За установяване на връзка хостовете изпълняват трипосочна процедура – “ръкостискане” (three-way handshake). Контролните битове в TCP заглавието регистрират прогреса и състоянието на връзката.

Three-way handshake прави:

Установяване и терминиране на TCP съединение

Установява, че отсрещното устройство съществува в мрежата;

Уверява се, че отсрещното устройство има активна услуга и приема заявки на отдалечения порт, който инициатора смята да използва за сесията

Информира отсрещното устройство, че клиент източник има намерение да установи сесия с този номер на порт

3 стъпки на TCP съединение

При TCP съединенията хостът-клиент инициира сесията към сървъра.

За да си изясним как работи “three-way handshake”, нека видим какви стойности на параметри си разменят двете старани. Трите стъпки в установяване на TCP съединение са:

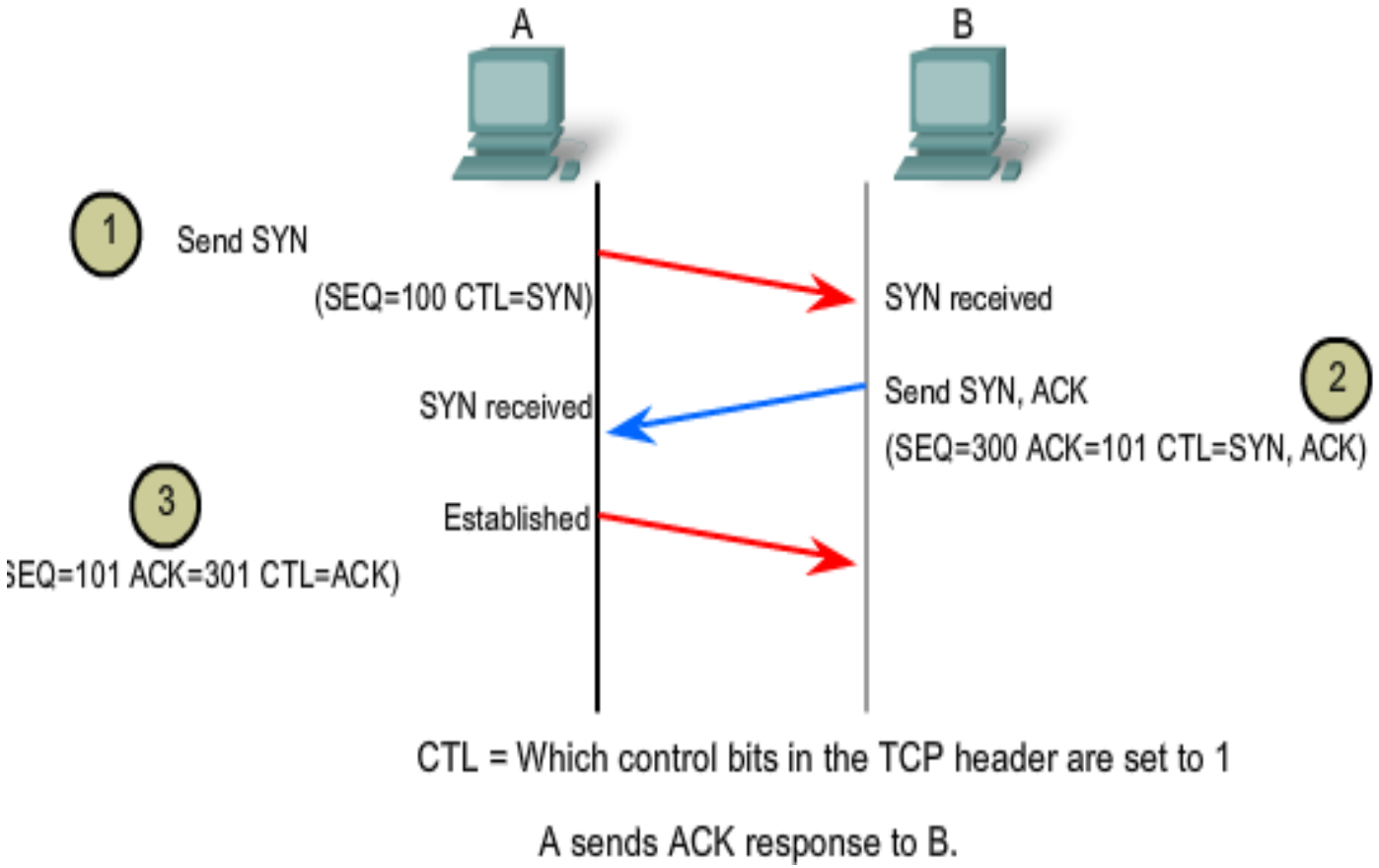
3-way handshake

1. Клиентът-инициатор изпраща сегмент SYN, съдържащ начална стойност на последователността SEQ, и представляващ заявка за начало на сесия.

2. В отговор сървърът изпраща сегмент, съдържащ стойност за потвърждение ACK (SEQ + 1). Освен това - собствената си синхронизираща стойност на последователност SYN SEQ, която е по-голяма от получения и потвърден номер ACK – следващия очакван байт.

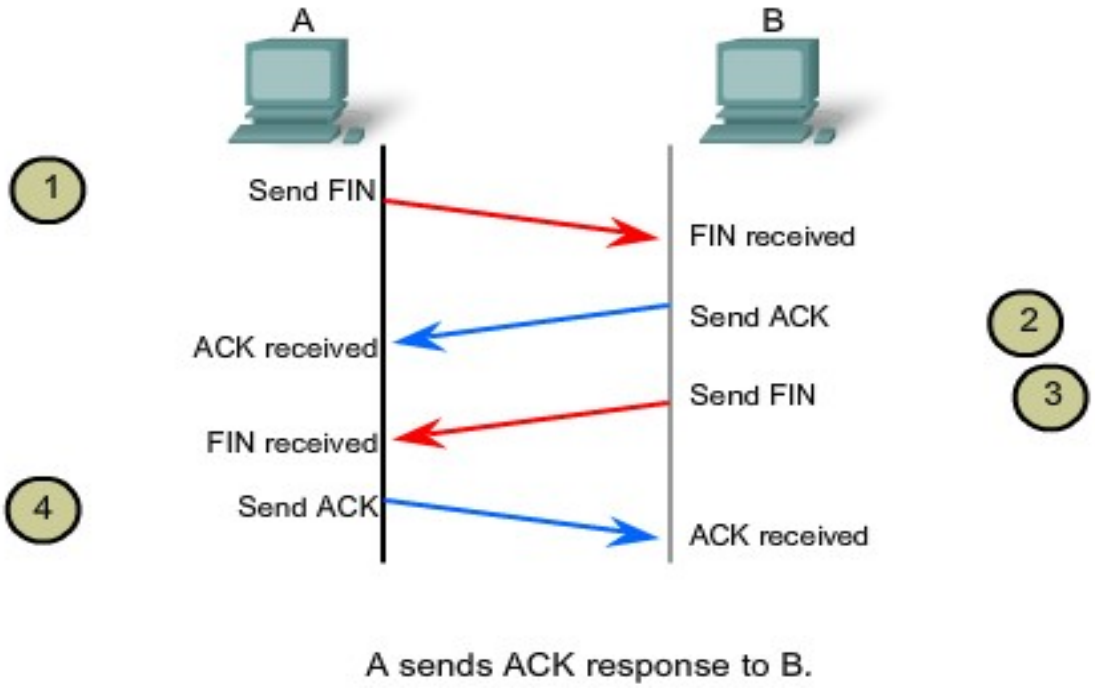
3. Клиентът-инициатор отговаря със стойност ACK, равна на (получена SEQ + 1). С това съединението е установено.

3 стъпки на TCP съединение



Терминиране на TCP сесия

По 4-стъпкова процедура се разменят флагове за терминиране на TCP съединение.



TCP. Потвърждение и прозорци.

Стойностите SEQ и ACK в заглавието на сегмента съвместно служат за потвърждение на получените байтове с данни.

SEQ е относителния брой байтове, които са предадени в дадената сесия + 1 (номера на първия байт с данни в дадения сегмент).

TCP използва числото ACK в сегментите, които се изпращат обратно на източника, за да покаже кой е следващия байт, който приемника очаква да получи в настоящата сесия. Това се нарича очакваното потвърждение.

TCP. Потвърждение и прозорци.

Източникът на сесията е информиран, че дестинацията е получила всички байтове в този поток от данни с изключение на байта под номер, равен на номера, съдържащ се в потвърждението.

Очаква се хоста-инициатор да изпрати сегмент със SEQ = (ACK).

Запомнете. Всъщност всяко съединение представлява две еднопосочни сесии. Стойностите SEQ и ACK се разменят и в двете посоки.

TCP. Потвърждение и прозорци. Пример.

В примера на фигурата на по-следващия слайд левият хост изпраща данни към десния. По-точно, изпраща сегмент с 10 байта данни и SEQ = 1.

Хостът отдясно получава сегмента и вижда, че SEQ = 1 и има 10 байта с данни.

Хостът отдясно връща обратно сегмент към левия хост, за да потвърди получаването на данните. В този сегмент ACK = 11, с което показва, че очаква да получи следващ байт с данни под номер 11.

TCP. Потвърждение и прозорци. Пример.

Забележете! Стойността ACK левия хост остава 1, за да покаже, че сегментът е част от вървящата сесия и стойността в полето Acknowledgment Number е валидна.

След като левият хост е получил потвърждение, вече може да продължи текущата сесия, като изпрати следващия сегмент с данни, започващи от байт 11.

TCP. Потвърждение и прозорци. Пример.

Ако левият хост в този пример трябва да чака потвърждение на всеки 10 байта, закъснението ще е много голямо.

За да се намали служебния трафик от тези потвърждения, възможно е да се изпратят множество сегменти, които да се потвърдят с едно единствено TCP в обратната посока.

В това потвърждение се съдържа число, показващо общия брой на байтовете, получени в тази сесия.

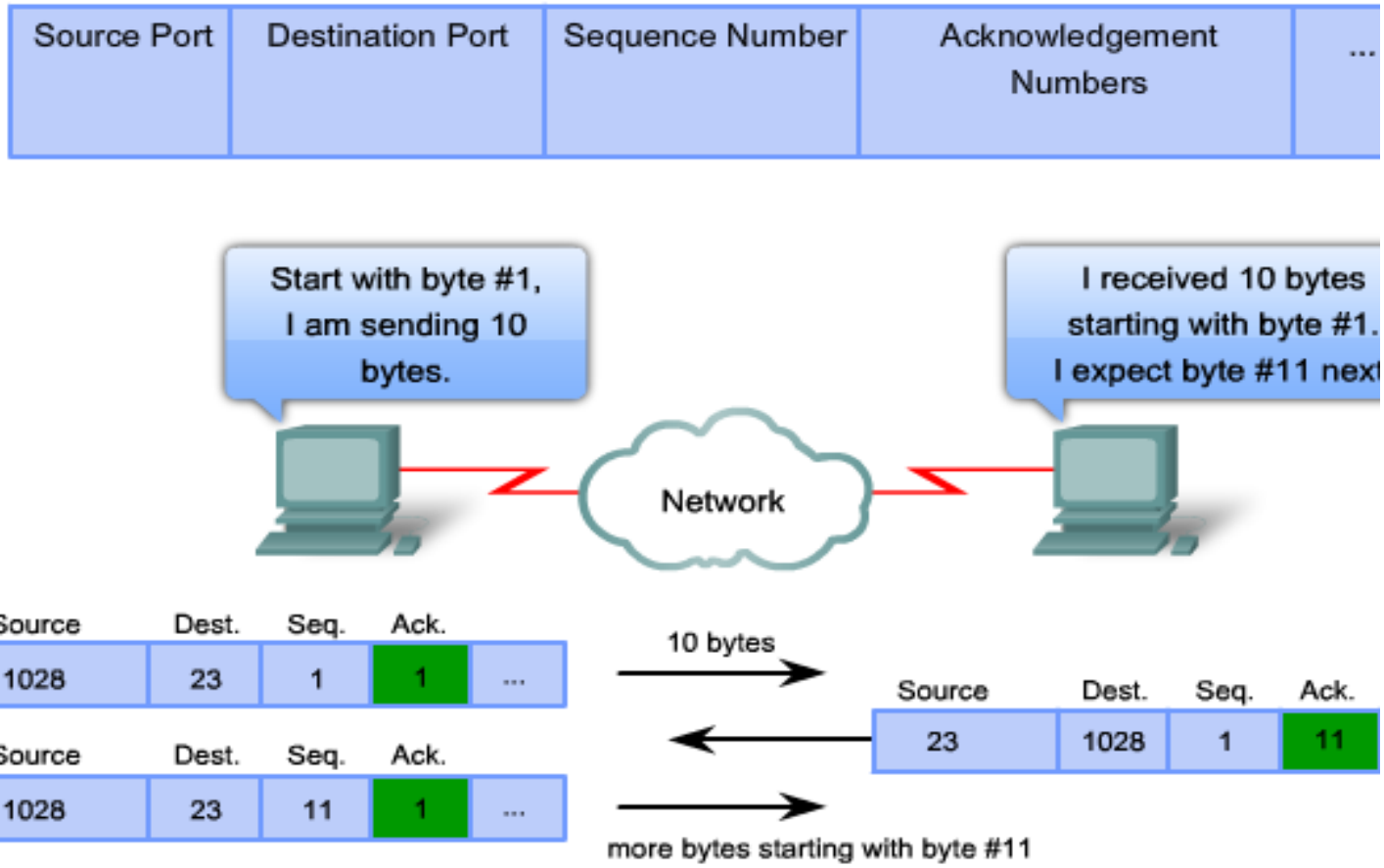
TCP. Потвърждение и прозорци. Пример.

Например, стартираме със SEQ = 2000. Получени са 10 сегмента с по 1000 байта всеки. На източника (левия хост) ще бъде върнат ACK = 12000.

Количеството данни, които източникът може да изпрати, преди да получи потвърждение, се нарича размер на прозореца.

Window Size е поле в заглавието на сегмента, което улеснява улавянето на загубени данни и управлението на потока.

TCP. Потвърждение и прозорци. Пример.



TCP. Повторно предаване.

Колкото и добре да е проектирана дадена мрежа, не може без загуби на данни.

TCP осигурява методи за управление на тези загуби на сегменти. Един от тях е механизмът за повторно предаване на сегменти с непотвърдени данни.

TCP услугата в хоста-приемник потвърждава само данни, състоящи се от непрекъсната последователност от данни. Ако липсват един или повече сегменти, потвърждават се само данните в сегментите, които попълват плътно потока.

TCP. Повторно предаване.

Например, получени са сегменти със SEQ = 1500 - 3000 и 3400 – 3500. Тогава ACK = 3001. Това е така, защото има сегменти със SEQ = 3001 – 3399, които не са получени.

TCP в хоста-източник не е получил потвърждение след предварително зададено време. Тогава той ще се върне към последния ACK, който е получен, и ще предаде повторно данните от тази точка нататък.

TCP. Повторно предаване.

Процесът на повторно предаване не е дефиниран в RFC, зависи от конкретната реализация на TCP.

В типичната TCP реализация хост предава сегмент, поставя копие от него в опашка за препредаване и стартира таймер. След получаване на потвърждение сегментът се изтрива от опашката.

Ако след нулиране на таймера не се получи потвърждение, сегментът се предава отново.

Днес хостовете могат да поддържат и опцията “Селективно потвърждение”. В такъв случай дестинацията би потвърдила байтове в непоследователни сегменти, без да е необходимо препредаване на липсващи данни.

TCP. Управление на потока и задръстванията.

TCP има и механизми за управление на потока (flow control).

Flow control синхронизира скоростите на потока от данни между двете услуги в сесията.

Когато източникът е информиран, че оределено количество данни в сегментите е получено, тогава може да продължи с изпращане на повече данни за дадената сесия.

TCP. Управление на потока и задръстванията.

Полето Window Size определя количеството данни, което може да бъде предадено, преди да бъде получено потвърждение.

Първоначалният размер на прозореца се определя в началото на сесията чрез three-way handshake.

Механизмът за обратна връзка в TCP нагласява ефективната скорост на предаване на данните към максималния поток, който мрежата и устройството-получател могат да поддържат без загуби и без повторни предавания.

TCP. Управление на потока и задръстванията.

В примера на следващия слайд първоначалният размер на прозореца е 3000 байта.

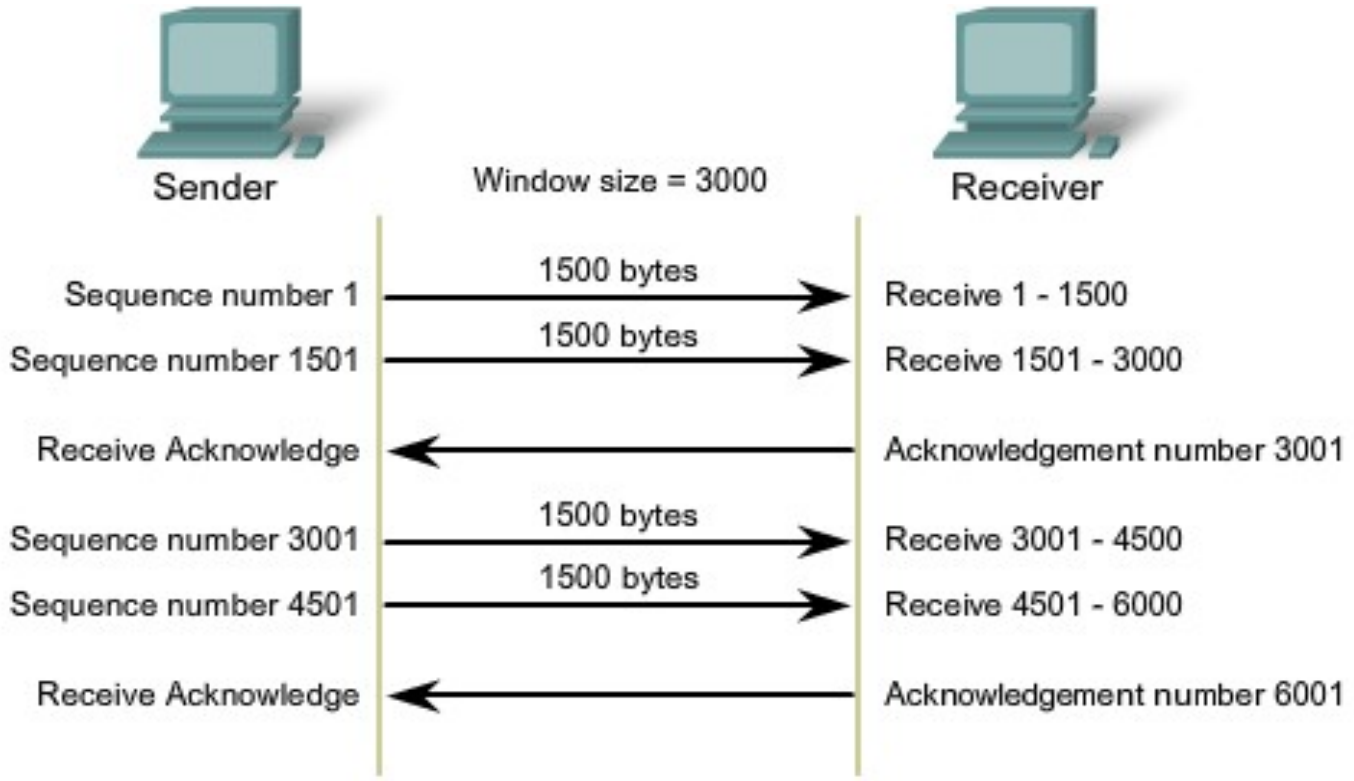
След предаването на тези 3000 байта изпращачът очаква потвърждение, преди да продължи със следващи сегменти. След получаването му може да предаде още 3000 байта.

В периоди, когато мрежата е задръстена или на получателя му липсват ресурси, закъснението може да се увеличи.

И ефективната скорост на предаване на данните за тази сесия да намалее.

Така се справяме с проблема с недостиг на ресурси.

TCP. Управление на потока и задръстванията.



TCP. Редуциране на загубите.

Друг начин за контролиране на потока от данни е да определяме размера на прозореца динамично.

При недостиг на мрежови ресурси TCP редуцира размера на прозореца, с което намалява и скоростта на предаване.

Хостът-получател в TCP сесията изпраща към подателя стойност на размера на прозореца, която показва броя на байтовете, които е в състояние да получи.

Както се вижда на фигурата в по-следващия слайд, имаме загуба на един сегмент. Получателят променя размера на прозореца от 3000 на 1500.

TCP. Редуциране на загубите.

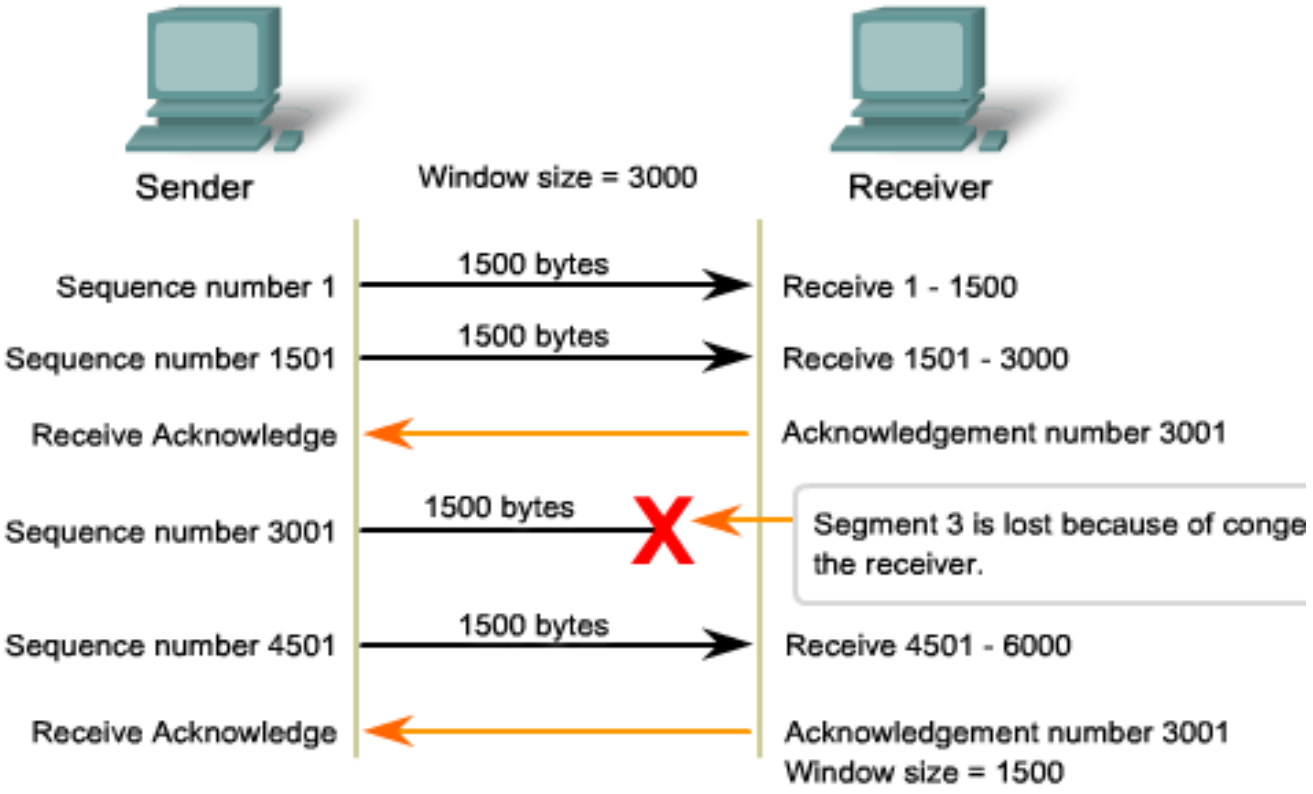
След периоди без загуби на данни или липса на ресурси получателят ще започне да увеличава прозореца. Това ще вдигне ефективната скорост и ще продължи, докато се появят загуби и трябва да се намали прозореца.

Динамичното увеличаване и намаляване на прозореца е непрекъснат процес в TCP, който определя оптималния размер във всеки един момент за дадена сесия.

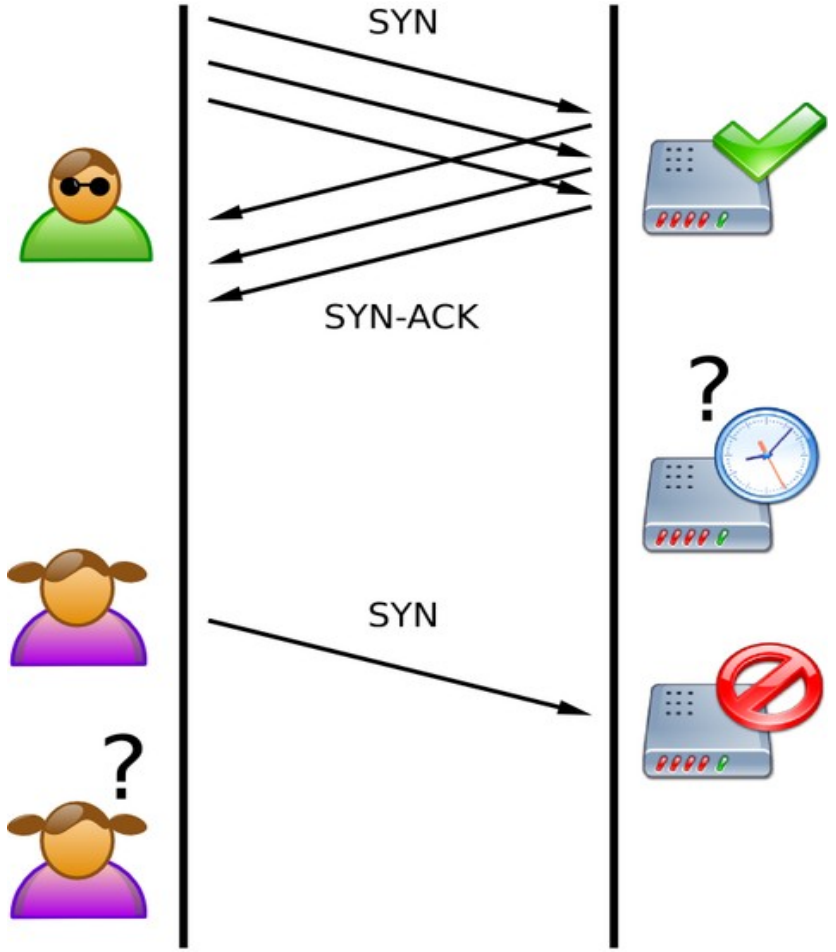
Подробности за справяне със задръстванията в TCP има в RFC 2581.

http://www.ietf.org/rfc/rfc2581.txt

TCP. Редуциране на загубите.



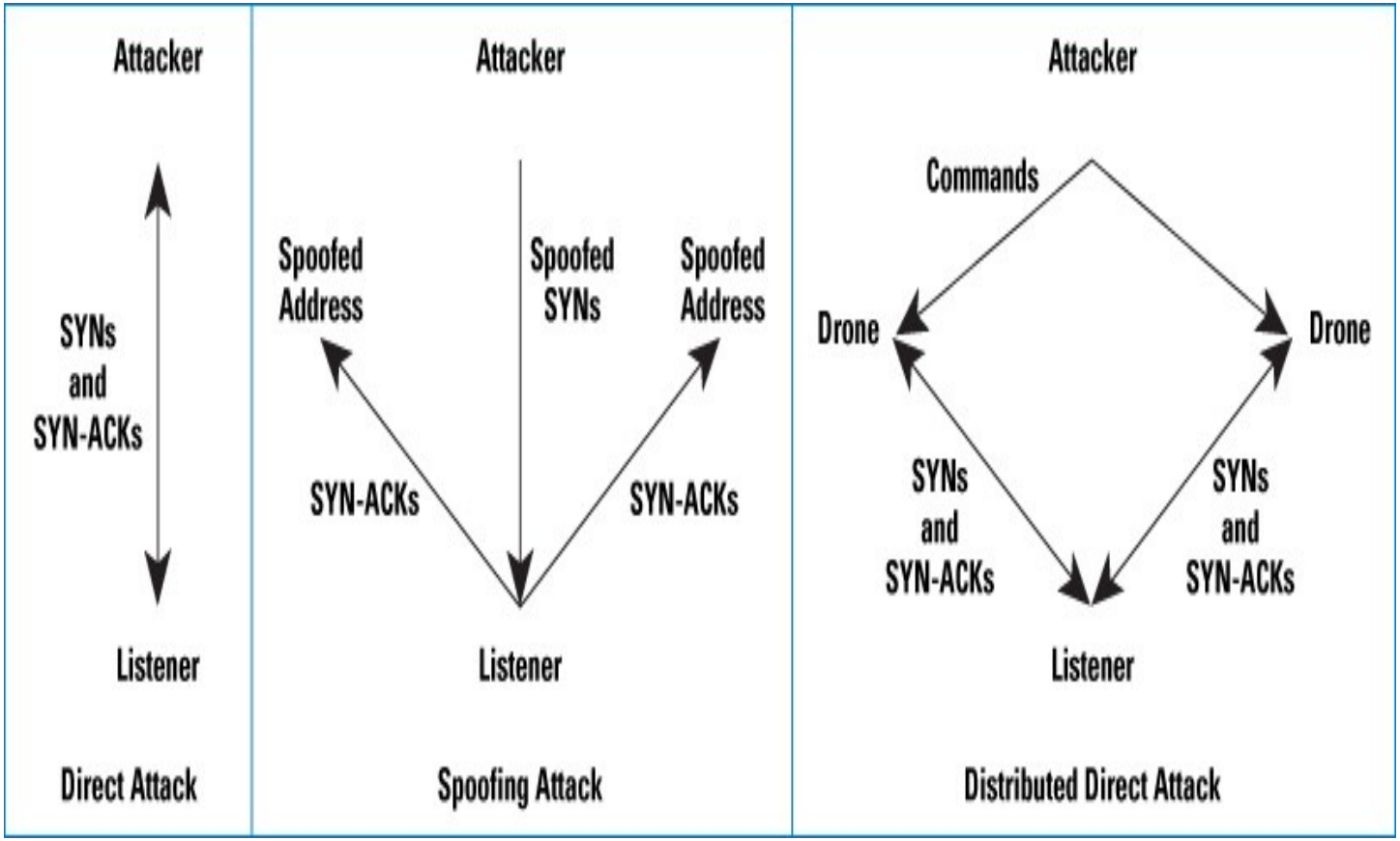
SYN flood атаки



Атакуващият изпраща няколко SYN, но не връща ACK

В резултат: полуотворени конекции. Няма място за легитимния потребител,

SYN flood атаки. Варианти.



UDP

UDP е по-опростен транспортен протокол с неустановена връзка.

Това не означава, че приложенията, които се базират на UDP, са непременно ненадеждни. Функциите по надеждност се осъществяват другаде.

Основни приложни протоколи, които “стъпват” на UDP са:

- Domain Name System (DNS)
- Simple Network Management Protocol (SNMP)
- Dynamic Host Configuration Protocol (DHCP)
- Routing Information Protocol (RIP)
- Trivial File Transfer Protocol (TFTP)
- онлайн игри

UDP

Онлайн игрите или VoIP могат да понесат някои загуби на данни. Но не и закъсненията, които внася TCP.

Други приложения като DNS или TFTP ще повторят заявката, ако не получат отговор. И не им трябват гаранциите на TCP.

UDP. Възстановяване на дейтаграми.

UDP е connectionless и сесии не се установяват като в TCP.

UDP е по-скоро транзакционен протокол. Т.е, ако приложението има данни за предаване, то ги предава.

Много UDP-базирани приложения изпращат малки количества данни, които се побират в един сегмент - дейтаграма.

Но някои изпраща по-големи количества, които се разделят на множество сегменти.

UDP. Възстановяване на дейтаграми.

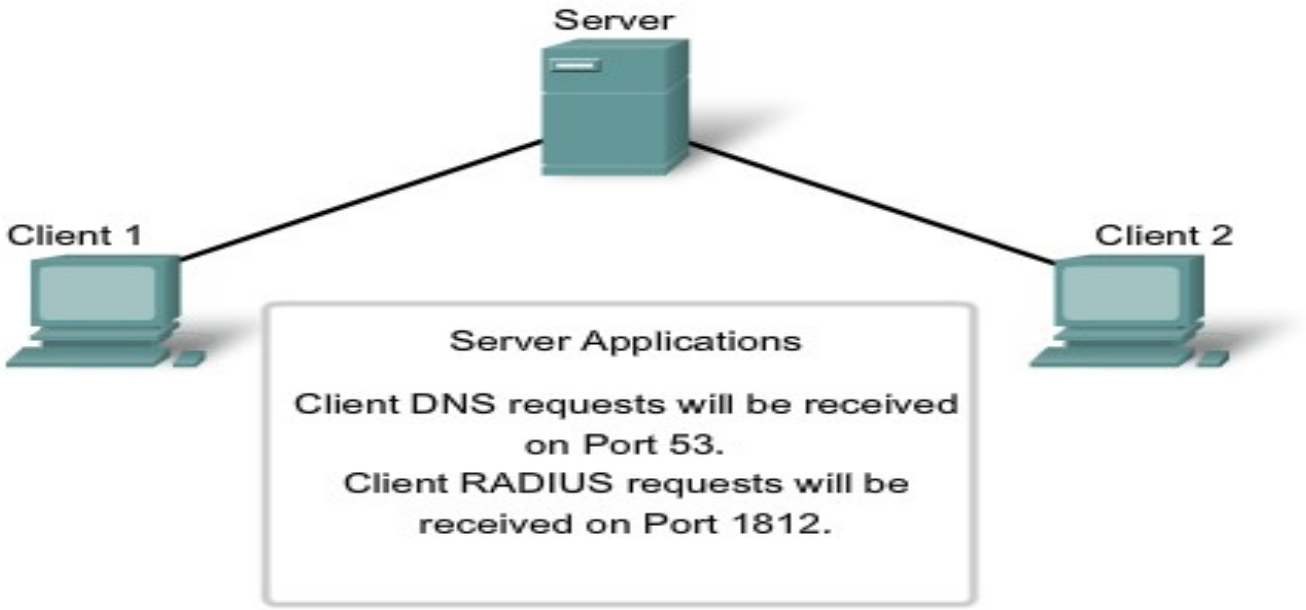
При изпращане на множеството дейтаграми от едно приложение те могат да поемат различни пътища в мрежата и да пристигнат в разбъркан ред.

UDP не следи последователността на дейтаграмите при приемане като TCP.

Ако тя е от значение, за това се грижи приложната програма.

UDP. Сървърски процеси и заявки.

И на UDP-базирани сървърски приложения се присвояват Well Known или Registered портове.



UDP. Клиентски процеси.

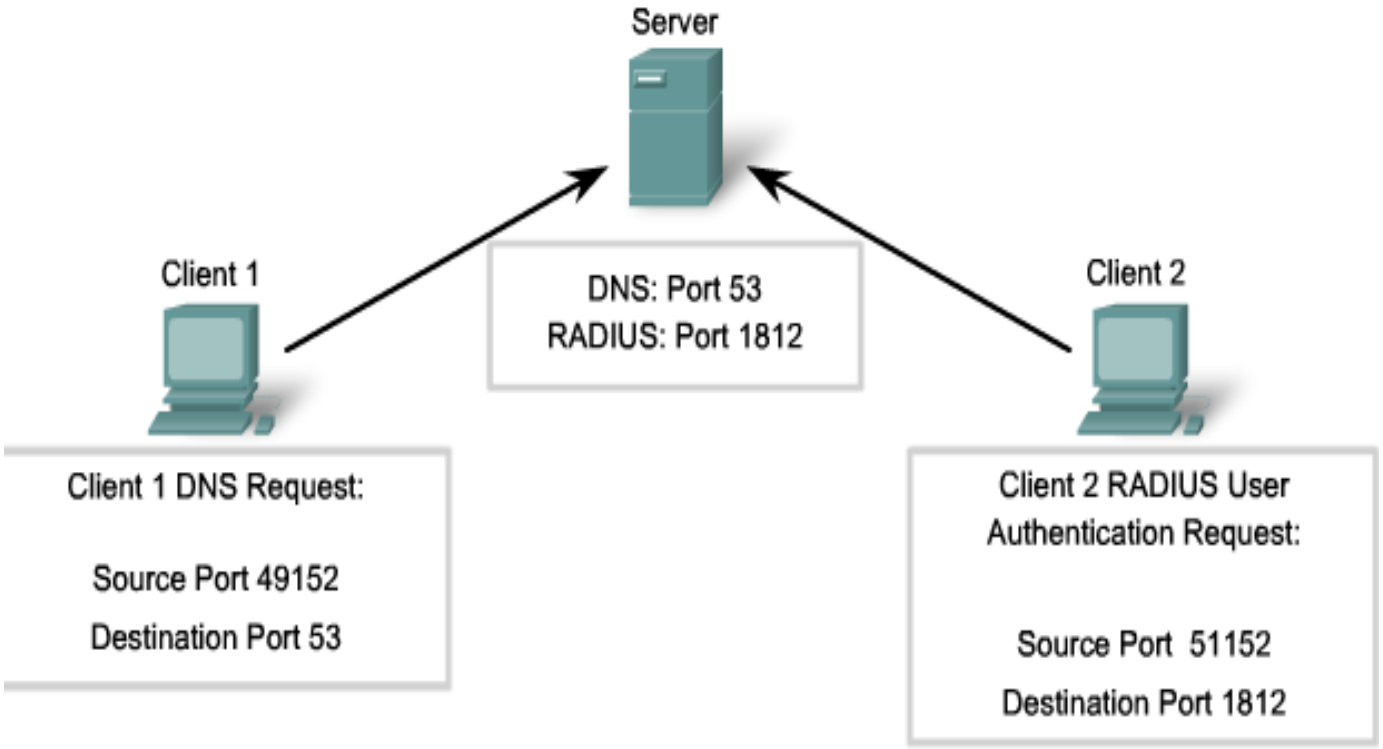
И тук като в TCP клиентското приложение изпраща заявка към сървъра.

Клиентският UDP процес избира на случаен принцип номер на порт от свободните.

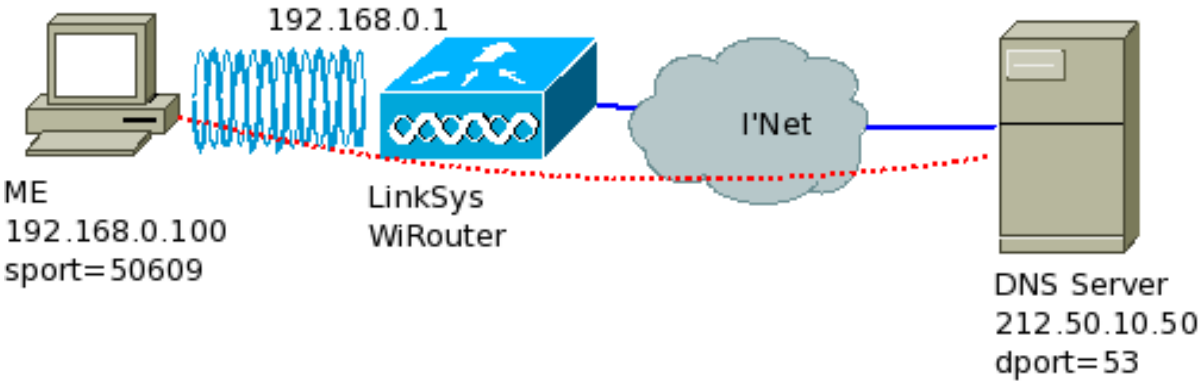
Случайният избор на порт помага за повишаване на сигурността. Номерът няма да е предварително известен на злосторника.

В UDP не се създават сесии. След като данните са готови и портовете са идентифицирани, UDP формира дейтаграма и и я подава към мрежовия слой за изпращане по мрежата.

UDP. Клиентски процеси.



UDP сесия по DNS




```
ipv4     2 udp      17 93 src=192.168.0.100
dst=212.50.10.50 sport=50609 dport=53
packets=2 bytes=150 src=212.50.10.50
dst=192.168.0.100 sport=53 dport=50609
packets=2 bytes=210 [ASSURED] mark=0
secmark=0 use=2
```