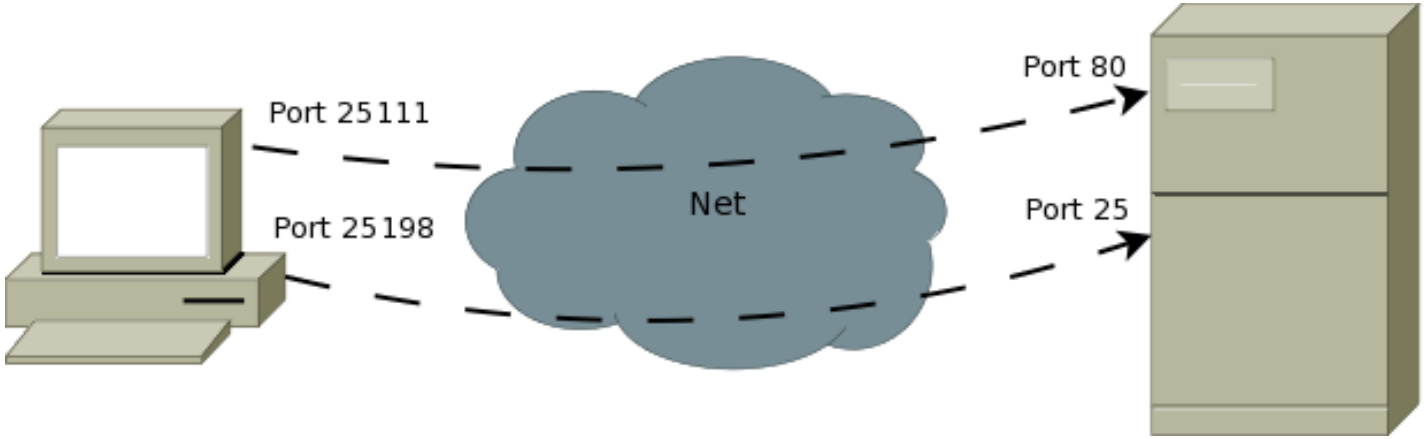
19. Транспортен слой.
20. Транспортни протоколи TCP и UDP.

Общи положения



Транспортният слой (transport layer) осигурява транспортирането на съобщения от източника до получателя. Той е най-ниският слой, който реализира връзка от тип “край-край” между комуникиращите системи.

Изгражда се програмен канал между портовете на приложения, които си “говорят” през мрежата.

Общи положения

Най-общата цел на транспортното ниво е да осигури ефективни и надеждни услуги на своите потребители - процеси в приложното ниво.

Транспортното ниво трябва да прехвърля произволно дълги съобщения между два крайни абоната.

За тази цел използва услугите на по-долното мрежово ниво.

Транспортното ниво предлага два вида обслужване – със съединение (надеждни) и без съединение (ненадеждни).

Протоколи със съединение

При протоколите със съединение имаме три фази – установяване на съединение, трансфер на данни и закриване на съединението.

Транспортното ниво трябва да предоставя интерфейс на приложното ниво.

Примерен интерфейс за надеждно обслужване се състои от следните примитиви LISTEN, CONNECT, SEND, RECEIVE, DISCONNECT.

Да предположим, че имаме приложение с един сървър и много отдалечени клиенти.

Установяване на връзката

Първоначално сървърът изпълнява примитива LISTEN, който го блокира докато не се появи клиент.

Когато клиентът иска да се свърже със сървъра, той изпълнява примитива CONNECT. Това изпълнение блокира клиента и изпраща пакет за установяване на връзка към сървъра.

Ако сървърът е в състояние на слушане (блокиран след изпълнение на LISTEN), той изпраща обратно пакет за потвърждение за създаване на връзка, което отблокира клиента и връзката е създадена.

Обмен на данни

След установяване на връзката сървърът и клиентът могат да обменят данни, като например всеки от тях изпълнява RECEIVE за изчакване на другия да изпълни SEND.

Докато двете страни могат да спазват реда, в който трябва да предават данни тази схема работи добре.

Всеки изпратен пакет данни трябва да се потвърждава обратно към изпращача.

Когато връзката вече не е нужна, тя трябва да се прекрати за да се освободят заетите от нея ресурси.

Освобождаване на връзката

Освобождаването на връзката има два варианта – симетричен и асиметричен.

При асиметричният вариант, само клиентът може да прекрати връзката чрез изпълнение на примитива DISCONNECT, който води до изпращане на пакет за прекратяване на връзката до сървъра.

При симетричния вариант и двете страни трябва да затворят връзката, т.е. да изпълнят DISCONNECT независимо един от друг.

Когато едната страна изпълни DISCONNECT, това означава, че тя повече няма да предава данни, но не означава, че няма да приема данни.

Транспортни протоколи

Услугите, предоставяни от транспортното ниво, се реализират от протокола на транспортно ниво, който действа между транспортното ниво на двата крайни абоната.

Всъщност транспортните протоколи си приличат с протоколите на канално ниво.

И двата типа протоколи се грижат за отстраняването на грешки, за последователно предаване на данните, за управление на потока и др.

Транспортни протоколи

Съществуват и много разлики.

При протоколите от каналното ниво двете точки взаимодействат помежду си през физически канал.

При транспортното ниво този физически канал се заменя от комуникационна подмрежа до целия Интернет.

Друга съществена разлика е, че при изпращане на кадър от каналното ниво той или пристига или се изгубва, докато пакетите при транспортното ниво могат да престояват в маршрутизаторите, да обикалят цялата подмрежа, докато достигнат местоназначението.

Мултиплексиране. Портове и сокети

За да могат множество процеси на един единствен хост да използват по едно и също време транспортния протокол, имаме адрес или **порт** (16-битов номер) за всеки процес.

В комбинация с мрежовия (IP) адрес се формира **сокет** (socket):

IP:port

Двойка сокети идентифицира едно съединение.

Видове портове

Според **RFC 6335** портовете се делят:

- Системни или Добре известни, **0-1023**, (присвояват се от IANA);

- Потребителски или Регистрирани, **1024-49151** (присвояват се от IANA);

- Динамични (Частни или Ephemeral - краткотрайни), **49152-65535** (не се присвояват)

http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml

Видове портове (2)

Добре известните портове са резервирани за популярни услуги и приложения – HTTP, POP3/SMTP, Telnet и др.

Регистрираните портове се присвояват на потребителски процеси и приложения.

Динамичните портове се присвояват динамично на клиентски приложения, инициращи сесия. Услугите не използват такива портове (с изключение на peer-to-peer file sharing програми).

Видове портове (3)

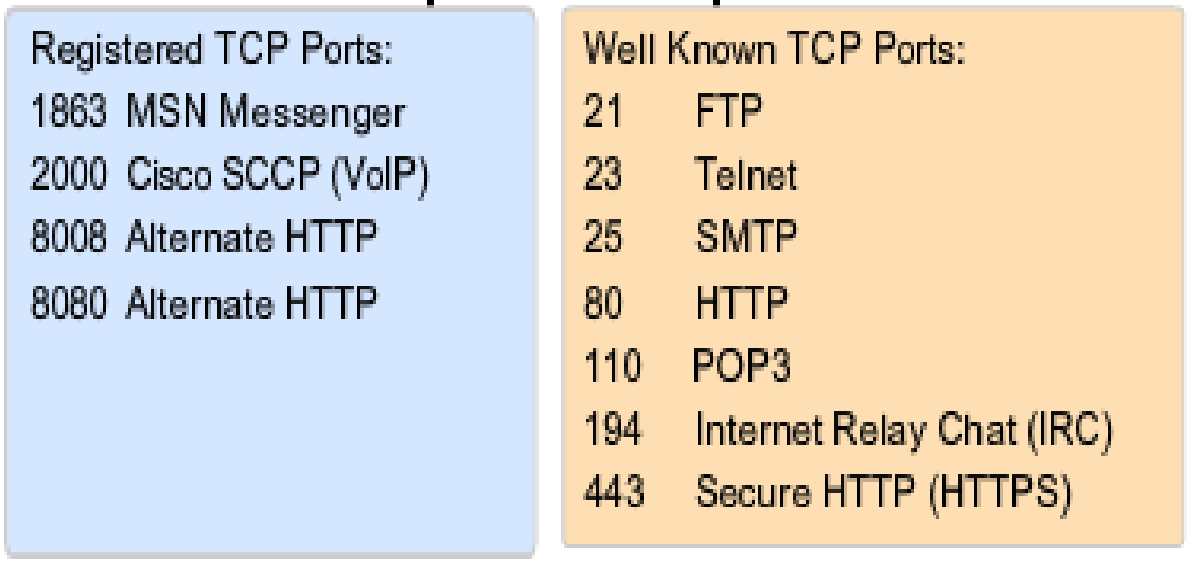
Номера, използвани и в TCP, и в UDP. За постигане на по-висока производителност някои приложения в даден момент се опират на TCP, в друг - на UDP.

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.

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



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

Registered UDP Ports:

1812 RADIUS Authentication Protocol

5004 RTP (Voice and Video Transport Protocol)

5060 SIP (VoIP)

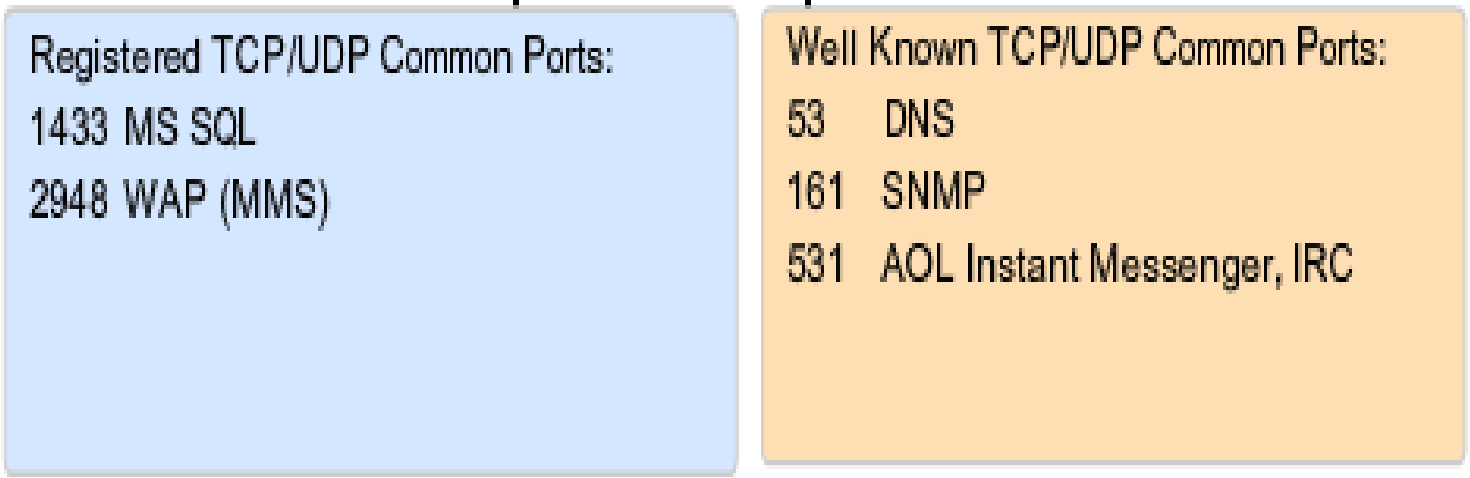
Well Known UDP Ports:

69 TFTP

520 RIP

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



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 двата са във функциите, които реализират.

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

Предимство е ниското закъснение.

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

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.

Типични приложения на TCP са:

- уеб браузъри и сървъри
- E-mail
- обмен на файлове (FTP)

TCP

Описание на TCP в RFC 793:

- реализира взаимодействие в режим с установяване на логическо (виртуално) съединение;
- осигурява двупосочна дуплексна връзка;
- организира потоково (от потребителска гледна точка) предаване на данните;
- предоставя възможност за предаване на части от данните, като «спешни»;
- за идентификация на партньорите при взаимодействието на транспортния слой се използват 16 битови «номера на портовете»;
- реализира се принципа «плаващ прозорец» (sliding window) за повишаване скоростта на предаването;
- поддържат се редица механизми за осигуряване на надеждно предаване на данните.

UDP

Описание на UDP в RFC 768:

- реализира взаимодействие в режим без установяване на логическо (виртуално) съединение;
- организира дейтаграмно (пакетно) предаване на данните;
- за идентификация на партньорите при взаимодействието на транспортния слой се използват 16 битови «номера на портовете»;
- не гарантира надеждно предаване на данните (възможна е както загуба на UDP пакети, така и дублиране на UDP пакети);
- не притежава средства за уведомяване на източника на UDP пакета за успешно/неуспешно получаване от приемника;
- не осигурява правилна последователност на доставката на UDP пакетите от източника до получателя (местоназначение);
- може да гарантира цялостност на данните в UDP пакета поради използването на контролна сума;
- много е прост (особено в сравнение с TCP).

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)			

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)			

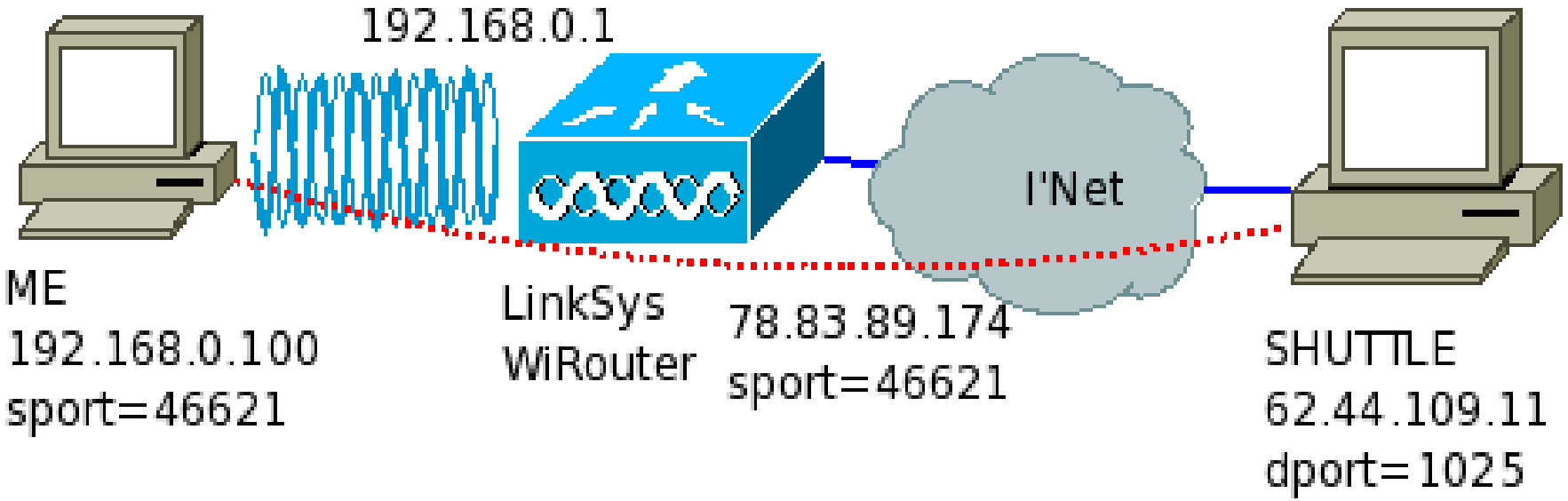
Адресиране с портове. Идентифициране на “разговорите”

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

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
```

source port

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

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

Ако уеб браузърът е на хост с IP: 192.168.100.48 и динамично му е присвоен порт 49152, тогава сокета, където трябва да бъде “свалена” уеб страницата ще е 192.168.100.48:49152.

destination port

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

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

Клиентът изпраща заявка до сървър: destination port е номер на порт, присвоен на процеса (демона) на услугата, стартирана 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, уеб браузър прави заявка към уеб сървър. Използва TCP и порт 80, ако не е дефинирано нещо друго.

TCP port 80 е по подразбиране присвоен на уеб 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 динамичните портове.

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).

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 да са в различен ред от този, по който са били предадени.

Transmission Control Protocol

Transmission Control Protocol (**TCP**) (**RFC 793**)
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 (от 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* (приложен слой).

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 уеб 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.

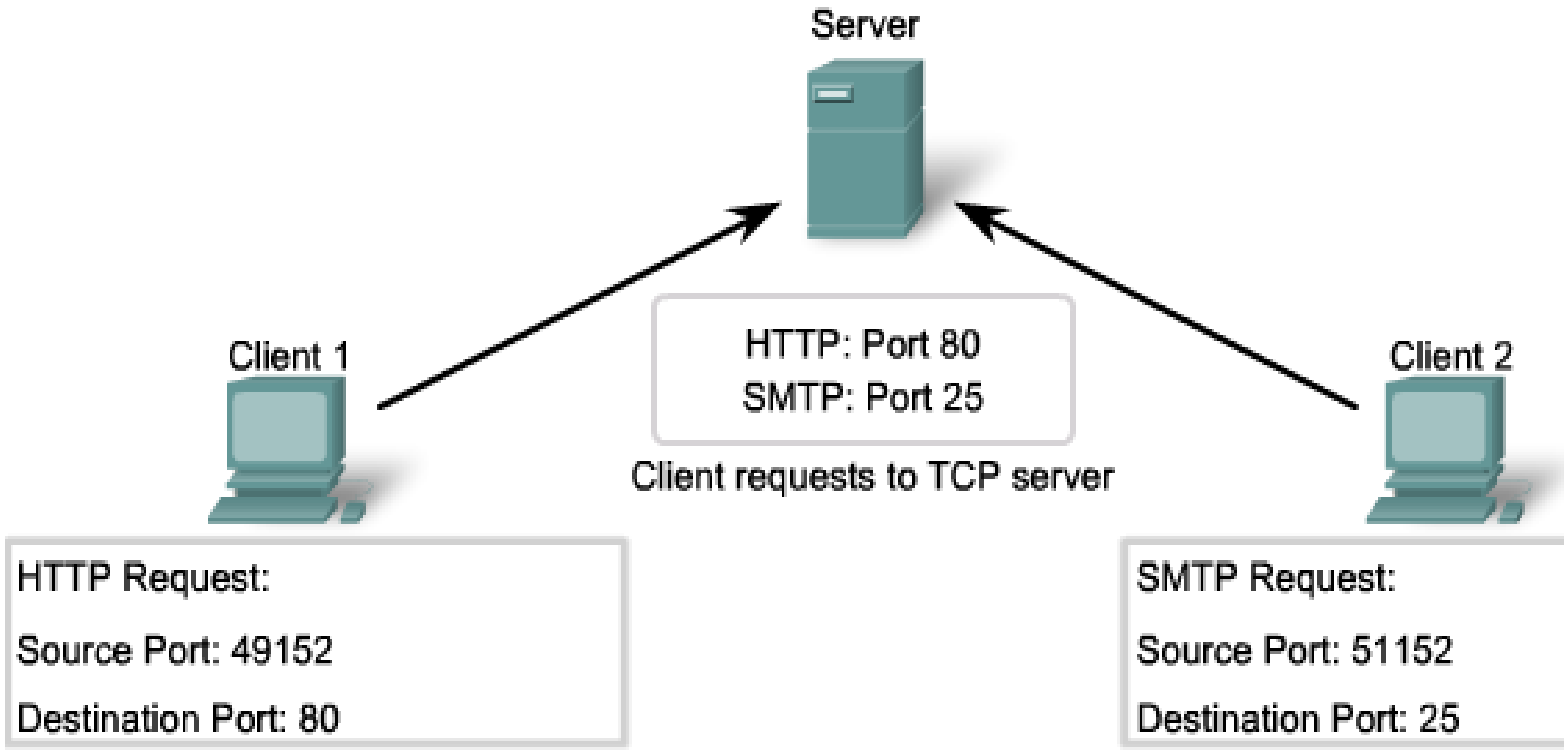
TCP и IP

Всички IP пакети, които произхождат от горния HTML файл са с един и същ адрес на получател, но те могат да преминат 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 съща 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

Поле **Checksum** (гарантира точността на сегмента). Изчислява се върху:

- TCP header, TCP data и

- псевдозаглавие (pseudoheader) – source IP, destination IP и поле дължина (length) в IP заглавието. (гарантира че няма промяна в IP адресите)

checksum е задължително в TCP и в IPv4, и в IPv6.

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.

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)**.

Още за **Sequence Numbers**

В TCP всеки байт (октет) се номерира (sequenced) и може да се потвърди.

Потвърждението на Seq.No. X показва, че всички байтове до (но не и) X са получени.

Байтовете се номерират така: първият след хедъра е с най-малък номер, следващите се увеличават с по 1.

Още за **Sequence Numbers**

Последователните номера са едно крайно число: (0 - 2**32 - 1).

Аритметиката с тези номера е modulo 2**32.

Това е беззнакова аритметика, която запазва отношението на SeqNos при броене в обратна посока: от 2**32 - 1 до 0.

modulo аритметиката има някои тънкости, трябва много внимателно да се програмира сравнението на стойности.

32-bit SeqNo и бързите мрежи

Table 5.1 Time Until 32-Bit Sequence Number Space Wraps Around

Bandwidth	Time until Wraparound
T1 (1.5 Mbps)	6.4 hours
Ethernet (10 Mbps)	57 minutes
T3 (45 Mbps)	13 minutes
Fast Ethernet (100 Mbps)	6 minutes
OC-3 (155 Mbps)	4 minutes
OC-12 (622 Mbps)	55 seconds
OC-48 (2.5 Gbps)	14 seconds

32-bit SeqNo беше до скоро адекватно за мрежите, но с днешния гигабитов Ethernet (10, 40, 100 Gbps) се изчерпва много бързо. Затова IETF разработи разширения на TCP.

TCP Extensions

RFC 7323 дефинира разширения на TCP (extensions) за повишаване на производителността по високоскоростни мрежи.

TCP Window Scale (WS) поддържа по-големи прозорци.

TCP Timestamps (TS) се прилага за защита от препълване на полето SeqNo - Protection Against Wrapped Sequences (PAWS) и измерване на времето за “отиване и връщане” Round-Trip Time Measurement (RTTM).

TCP Extensions. Препълване на 32-bit SeqNo.

В TCP хедъра се въвеждат:

– поле TCP header length – хедър с променлива дължина, но обратно съвместим (основните полета не са променени)

За бързите мрежи:

– 64-bit SeqNo – разширен с **32-bit timestamp**

– timestamp постоянно се увеличава и служи за различаване на сегменти с един и същ SeqNo. (но вторият е превъртян).

Стартиране на TCP съединение

Двете крайни точки - TCP модулите, съдържат информация, дефинираща виртуален канал.

Виртуалният канал е пълен дуплекс.

Приложението записва данни в TCP порта, които се прочитат от крайната точка.

TCP - sliding window протокол

TCP е протокол с “плъзгащ прозорец” с time-out и повторни предавания. Предадените данни се потвърждават от отдалеченото TCP.

Управление на потока се упражнява и от двете страни, за да не се получи препълване на буферите.

Размерът на прозореца определя количеството на данните, които се предадат, преди да се получи потвърждение.

В протокола TCP **количеството на данните се измерва с броя на байтовете**, а не с броя на TCP сегментите.

Основни функции на TCP

- Трасфер на данни
- Надеждност
- Управление на потока
- Мултиплексиране
- Съединения
- Приоритети и сигурност

Initial Sequence Number

Генератор на начален последователен номер (**ISN**) избира нов 32-bit ISN при предавателя и при приемника.

Генераторът е обвързан с фиктивен 32-bit часовник, чийто младши бит се инкрементира на всеки 4 μs. Т.е. ISN превърта на всеки 4.55 часа при 2 Mb/s. (при 100 Mb/s, 5.4 минути).

Сегментите престояват в мрежата не повече от Maximum Segment Lifetime (MSL= 2 минути) < 4.55 ч., следователно ще бъдат уникални.

3-way (или three message) handshake

При инициализиране на TCP съединение трябва да си синхронизират двете крайни точки ISN.

Те си обменят сегменти за установяване на съединение, които са с вдигнат бит "SYN" и ISN.

Този процес се нарича “ръкостискане” (**handshake**).

Стъпки при ръкостискането

1) A --> B SYN моят sequence number е X

2) A <-- B ACK твоят sequence number е X

3) A <-- B SYN моят sequence number е Y

4) A --> B ACK твоят sequence number е Y

Стъпки 2 и 3 се комбинират в едно съобщение, затова имаме трипосочно (с три съобщения) ръкостискане.

Защо трипосочно?

Защото SeqNos не са обвързани с глобален часовник и TCP-та може да имат различни механизми за избор на ISN.

Получателят на първия SYN няма как да знае дали сегментът е закъснял или не, затова иска от подателя да верифицира този SYN.

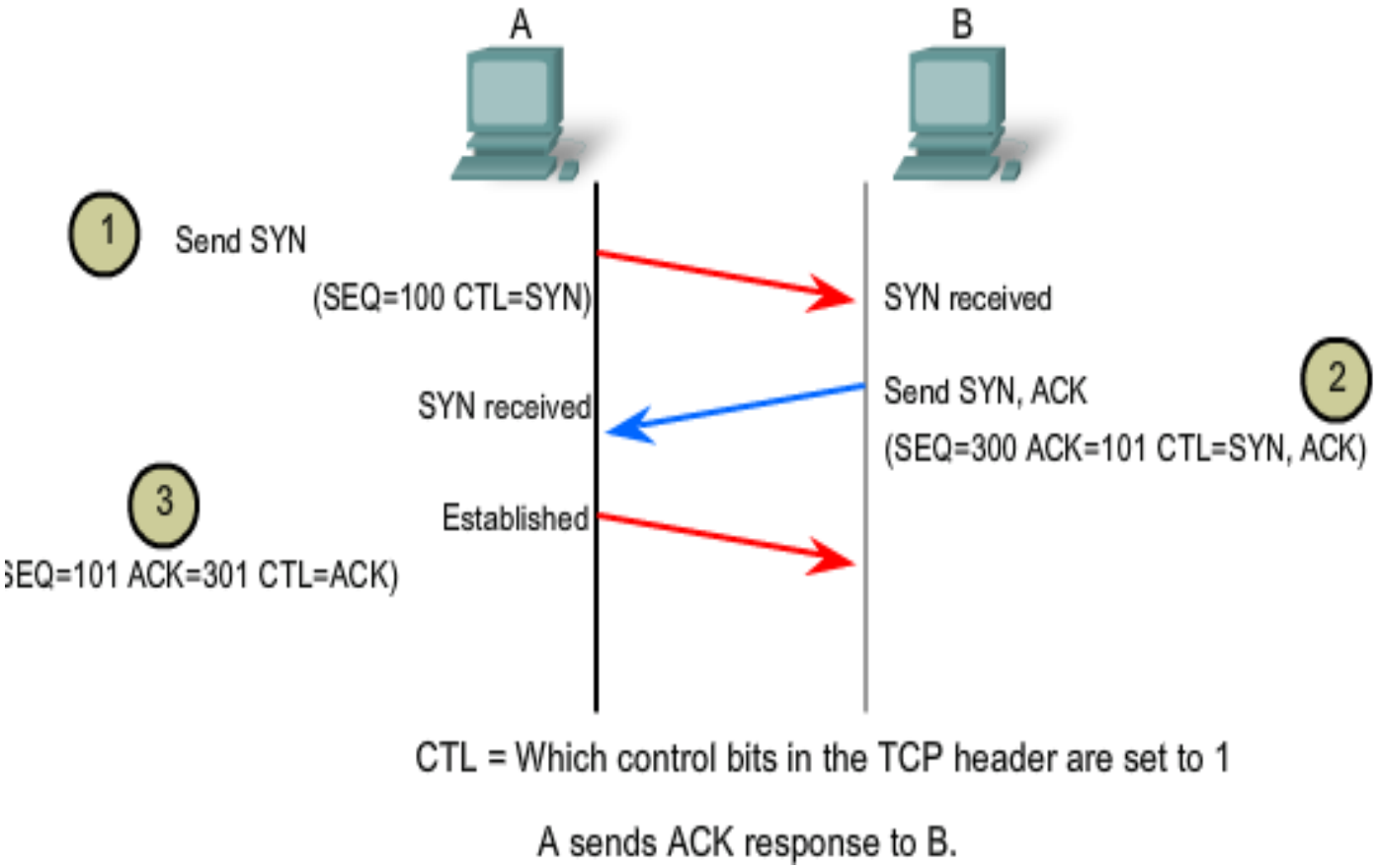
Примери

Сървър B “слуша” на порт:

TCP A TCP B

1. CLOSED LISTEN
2. SYN-SENT --> <SEQ=100><CTL=SYN> --> SYN-RECEIVED
3. ESTABLISHED <-- <SEQ=300><ACK=101><CTL=SYN,ACK> <-- SYN-RECEIVED
4. ESTABLISHED --> <SEQ=101><ACK=301><CTL=ACK> --> ESTABLISHED
5. ESTABLISHED --> <SEQ=101><ACK=301><CTL=ACK><DATA> --> ESTABLISHED

3 стъпки на TCP отваряне на съединение



Примери

Инициране на сесия едновременно от двете страни:

TCP A TCP B

1. CLOSED CLOSED
2. SYN-SENT --> <SEQ=100><CTL=SYN> ...
3. SYN-RECEIVED <-- <SEQ=300><CTL=SYN> <-- SYN-SENT
4. ... <SEQ=100><CTL=SYN> --> SYN-RECEIVED
5. SYN-RECEIVED --> <SEQ=100><ACK=301><CTL=SYN,ACK> ...
6. ESTABLISHED <-- <SEQ=300><ACK=101><CTL=SYN,ACK> <-- SYN-RECEIVED
7. ... <SEQ=101><ACK=301><CTL=ACK> --> ESTABLISHED

Терминиране на TCP сесия

Имаме три случая:

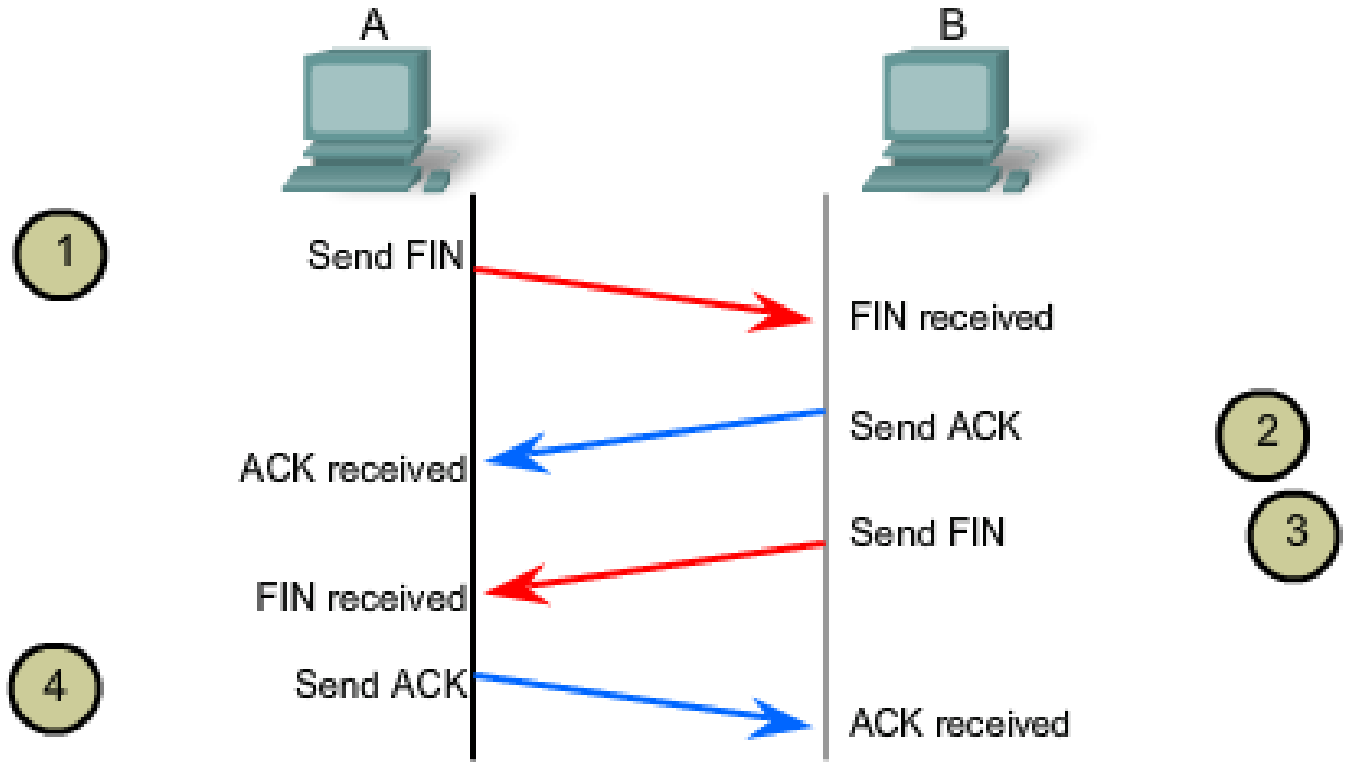
1) Потребителят го инициира, като “казва” на TCP да затвори (CLOSE) съединението

2) Отдалеченото TCP инициира като изпраща сигнал FIN

3) И от двете страни едноременно затварят (CLOSE)

Терминиране на TCP сесия

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



A sends ACK response to B.

Предаване и приемане на данни

След установяване на съединението започва обмен на данни (сегменти).

Загубените сегменти (загуба на връзка, задръстване или “лош” checksum) се предават повторно.

Това става след timeout. Възможно е да се получи дублиране на сегменти. Затова TCP тества SeqNos и ACKNos.

Изпращачът следи поредния номер чрез променливата SND.NXT и последния непотвърден номер чрез SND.UNA.

Приемникът следи чрез RCV.NXT.

Ако има временно “затишие” в потока от данни и всички изпратени данни са потвърдени, трите променливи ще са равни.

Интерфейс User/TCP

Тук се дават най-общите характеристики на командите, които трябва да генерират потребителските програми към TCP. За всяка ОС потребителския интерфейс е различен.

Но всички TCP-та трябва осигуряват един минимум от услуги, за да гарантират функционалността на протокола.

Интерфейс User/TCP (2)

При междупроцесните комуникации TCP трябва не само да приема команди, но и да връща информация към процесите, които обслужва:

(a) обща информация за съединението (прекъсвания, отдалечено затваряне, обвързване с външен сокет).

(b) отговори на определени потребителски команди (успех или пропадане).

Open

Формат:

OPEN (local port, foreign socket, active/passive [, timeout] [, precedence] [, security/compartment] [, options])

-> local connection name

Send

Format:

SEND (local connection name, buffer address, byte count, PUSH flag, URGENT flag [,timeout])

Ако съединението не е било отворено, SEND връща грешка.

Ако извикващият процес не е оторизиран да използва това съединение, връща грешка.

Receive

Format: RECEIVE (local connection name, buffer address, byte count) -> byte count, urgent flag, push flag

Тази команда алокира приемен буфер.

Ако не е предхождана от OPEN или извикващият процес не е оторизиран да използва това съединение, връща грешка.

Close

Format: CLOSE (local connection name)

CLOSE означава "I have no more to send", но не означава "I will not receive any more."

Възможно е, ако потребителският протокол (програма) не е добре обмислен, затварящата страна да не може да се освободи от данните си преди някакъв time out.

Тогава CLOSE става ABORT и затварящият TCP се отказва.

Status

Format: STATUS (local connection name) -> status data

Тази команда връща блок със следната информация:

local socket, foreign socket,
local connection name,
receive window, send window,
connection state,
number of buffers awaiting acknowledgment,
number of buffers pending receipt,
urgent state, precedence,
security/compartment,
transmission timeout.

Abort

Format: ABORT (local connection name)

Тази команда предизвиква прекратяване на всички чакащи SENDs и RECEIVEs, премахва се TCB и към отсрещното TCP се изпраща специално съобщение RESET.

В зависимост от реализацията ABORT може да се получи за всеки SEND или RECEIVE, или само да получи ABORT-acknowledgment.

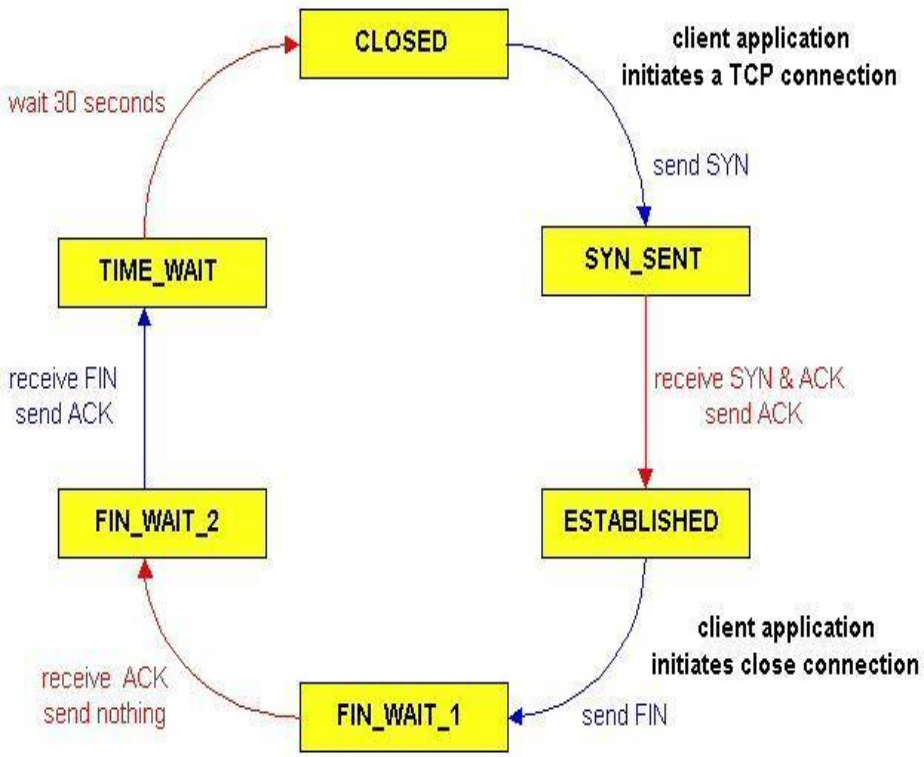


Figure 1: A typical sequence of TCP states visited by a client TCP

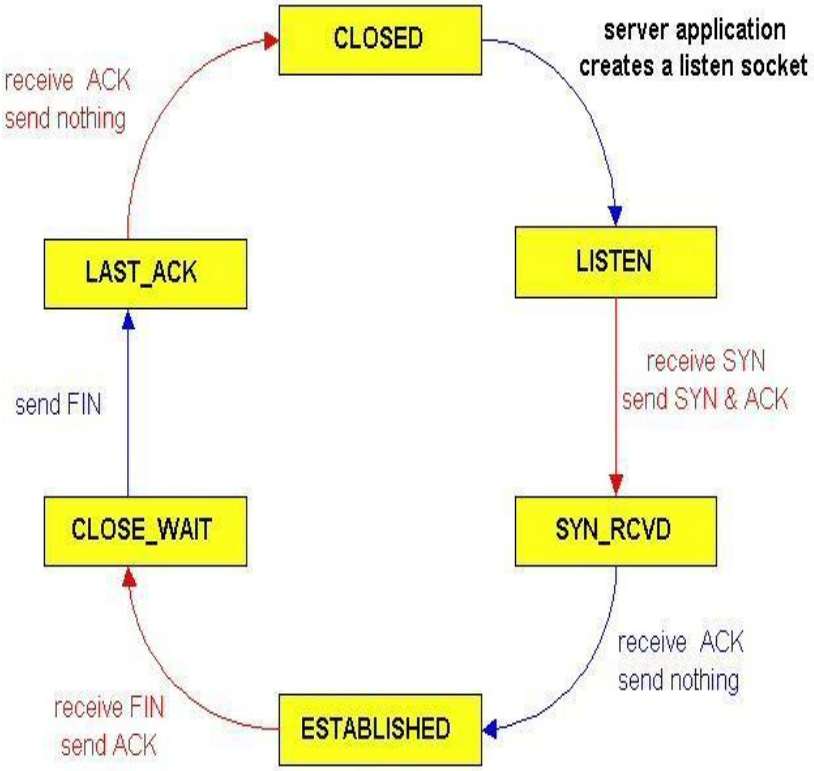


Figure 2: A typical sequence of TCP states visited by a server-side TCP

Отношение на TCP към IP

IP дава аргументи като type of service и time to live. TCP използва следните стойности:

Type of Service = Precedence: routine, Delay: normal, Throughput: normal, Reliability: normal (00000000).

Time to Live = 1 minute (00111100).

Заб. maximum segment lifetime = 2 minutes.

Но ако използваме стойността от IP протокола, MSL ще е 1 minute.

IP протоколът трябва да осигури:

- source address,
- destination address и протоколни полета
- да определи "TCP length".

Flow control. Sliding Window.

Полето **Window size** определя темпото 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 входящия 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.

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 приемникът 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).

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.

Стойността ACK за 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 номер 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.

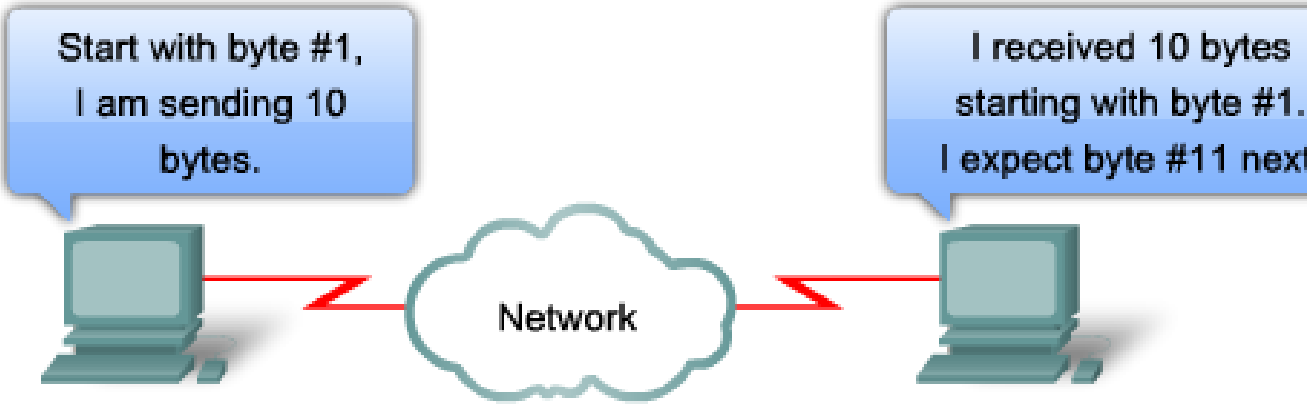
Например, стартираме със SEQ = 2000. Получени са 10 сегмента с по 1000 байта всеки. На източника (левия хост) ще бъде върнат ACK = 12001.

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.

Source Port	Destination Port	Sequence Number	Acknowledgement Numbers	...



Source	Dest.	Seq.	Ack.	...
1028	23	1	1	...

10 bytes

Source	Dest.	Seq.	Ack.
23	1028	1	11

Source	Dest.	Seq.	Ack.	...
1028	23	11	1	...

more bytes starting with byte #11

TCP. Повторно предаване.

Колкото и добре да е проектирана дадена мрежа, винаги има 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.

Това е познатият ви Selective Repeat. Тук Selective Acknowledge (**SACK**).

TCP. Повторно предаване.

Например, получени са сегменти със SEQ = 1500 - 3000 и 3400 – 3500.

Сегменти със SEQ = 3001 – 3399 не са получени.

TCP в хоста-източник съгласно SACK (RFC 2018 в Linux kernel >= 2.4) ще предаде повторно данните със SEQ = 3001 – 3399.