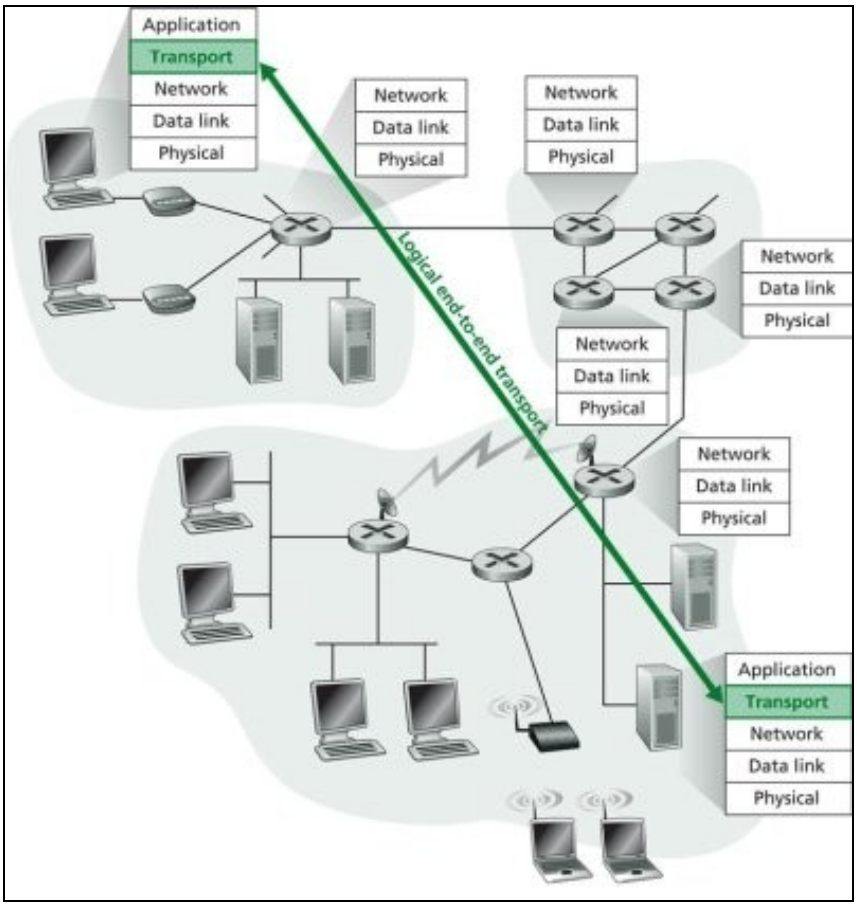
O nivel de transporte

Sumario

Introdución



No nivel de enlace os datos envíanse entre equipos dentro da mesma LAN, conectados por un *switch* fisicamente. No nivel de rede os datos envíanse a través de distintas redes conectadas físicamente mediante *routers*. No nivel de transporte hai unha conexión directa entre os dous extremos que interveñen na comunicación pero é unha **conexión lóxica**.

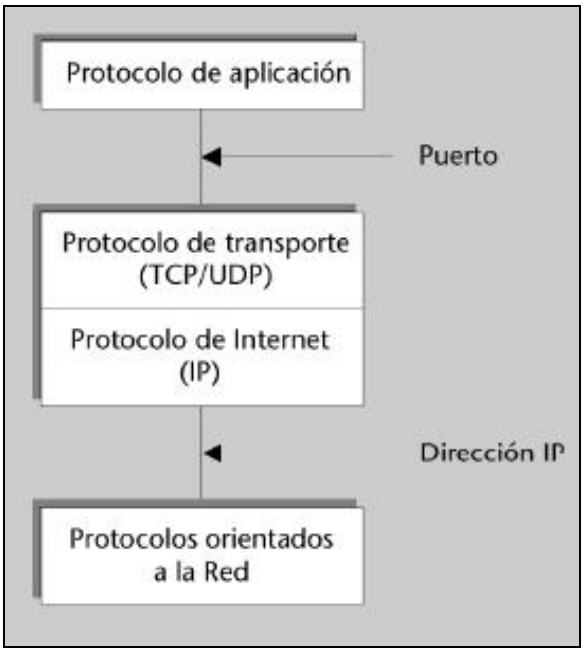
Por conexión lóxica entendemos que hai unha comunicación directa entre os programas ou equipos que interveñen na comunicación, independentemente da rede onde se atopen pero **non existe conexión física directa entre eles**. Existe, polo tanto, un **circuíto virtual** entre o emisor e receptor. A capa de transporte proporciona comunicación lóxica extremo a extremo, no canto de comunicación física. O nivel de transporte non é consciente, nin debe selo, de como están interconectados fisicamente os dous equipos (por unha LAN, unha WAN ou unha combinación de múltiples redes de ambos tipos). Que a comunicación sexa extremo a extremo quere dicir que non se coñecen os detalles da subrede, polo tanto, o nivel de transporte non sabe nada de *routers*, fragmentación, *switches*, *hubs*, etc.

Servizos orientados a conexión

Xeralmente as aplicacións requiren que o nivel de transporte lles garanta a entrega dos datos ao destinatario, sen erros, perdas, nin duplicados. Para que isto sexa posible o nivel de transporte pode ofrecer un **servizo orientado a conexión**, con retransmisións en caso necesario. Este é o caso do protocolo TCP (*Transport Control Protocol*), utilizado en moitas aplicacións (FTP, SMTP, HTTP...).

Noutros casos as aplicacións confórmanse (ou ata prefiren) un servizo menos fiable no que os datos se envían sen pedir confirmación, de forma independente uns doutros (como fai o IP cos paquetes). Este tipo de servizo ofréceo un protocolo **non orientado a conexión** que é UDP (*User Datagram Protocol*). UDP úsase, por exemplo, nas aplicacións en tempo real, onde non se quere incorrer no retardo propio dun protocolo orientado a conexión.

Portos



En TCP/IP defínense dúas direccións que permiten a relación cos niveis superiores e inferiores:

- A dirección IP, que xa coñecemos
- O **número de porto**, que identifica a aplicación que require a comunicación

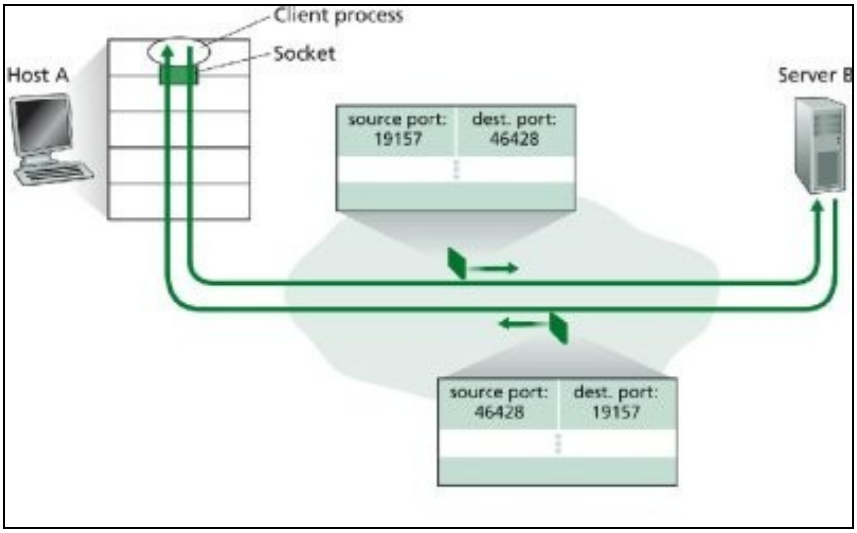
Os portos son os enderezos do nivel de transporte. Cada porto está asociado a unha aplicación. Pódense asignar de dous xeitos dependendo de se falamos da aplicación cliente ou da aplicación servidor:

- **Aplicación cliente**: cando se abre unha aplicación, por exemplo un navegador web, cliente ftp, etc. o sistema operativo asígnalle un porto dos que teña libres.
- **Aplicación servidor**: as aplicacións servidor están sempre escoitando nun porto chamado **ben coñecido** (*well known port*). Este porto configúrase manualmente. Un porto ben coñecido é como un número de teléfono no que sabemos que existe un servizo determinado, por exemplo, 091 é a policía, 112 son emerxencias, etc.

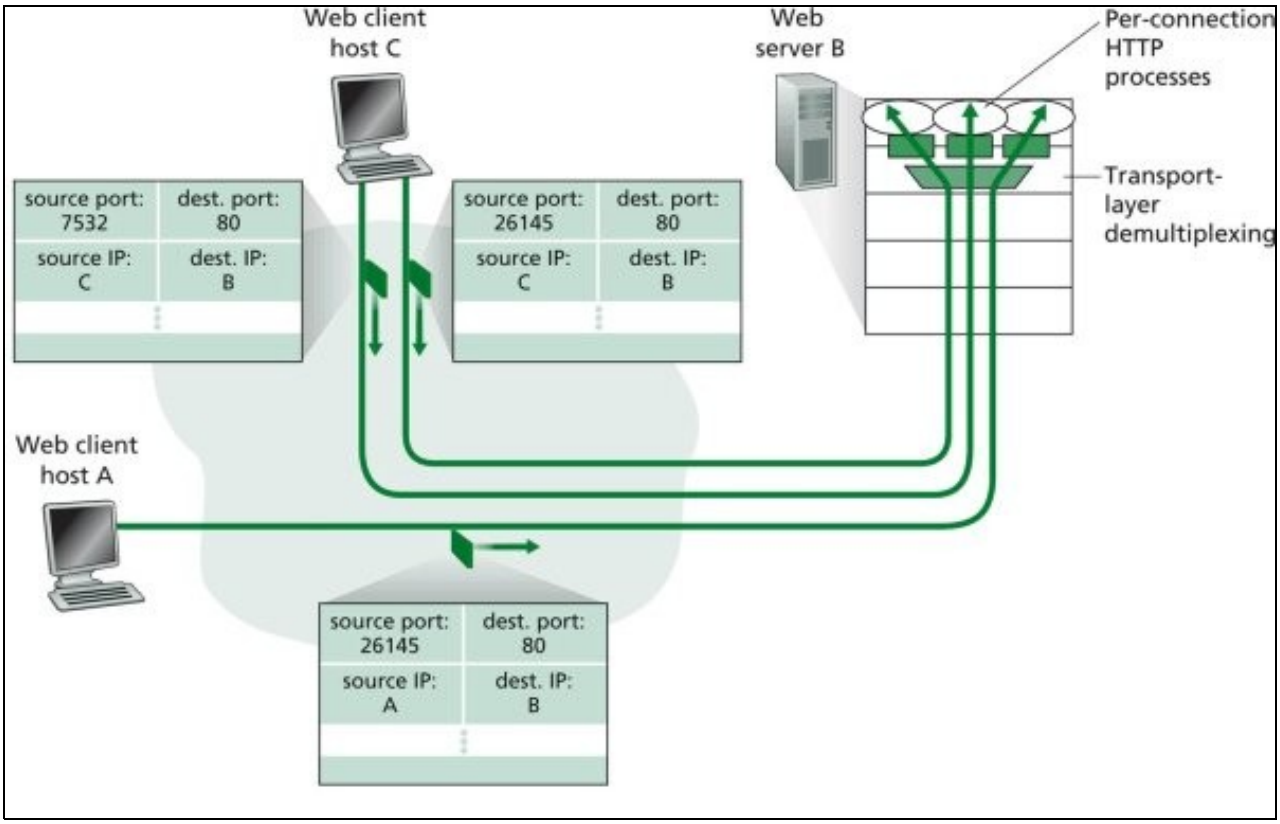
Toda conexión TCP idéntificase polo par **(IP_orixe, Porto_orixe)-(IP_destino, Porto_destino)**. Vexamos o seguinte exemplo:

Un usuario con IP 194.145.10.5 fai dobre clic sobre o navegador web. Nese intre o sistema operativo asígnalle un porto a esa aplicación (1500). O servidor ten, por exemplo, a IP 200.50.100.45. A aplicación cliente sabe en que porto está escoitando o servidor as peticións (neste caso no 80, xa que é o servizo Web). Se a aplicación servidor está escoitando nun porto distinto ao que lle corresponde, o usuario debe expresar cal é ese porto, por exemplo, o 81. A conexión TCP anterior identifícase polo par (194.145.10.5, 1500) - (200.50.100.45, 80).

As conexión TCP son full dúplex, é dicir, a información envíase en ambos sentidos por canais independentes. Isto fai que os números de porto se invirtan, dependendo de quen sexa o emisor ou o receptor, tal e como podemos ver na seguinte figura:



No seguinte gráfico podemos ver dous clientes usando o mesmo porto de destino (80) para comunicarse co mesmo servidor Web.



A Unidade de Datos do Protocolo (PDU)

O conxunto de bytes que transmite o nivel de transporte TCP coñécese como **segmento TCP**, mentres que ao conxunto de bytes que transmite o protocolo de transporte UDP chámase **datagrama UDP**. Ambos son PDU (*Protocol Data Unit*) do nivel de transporte en TCP/IP.

O UDP (*User Datagram Protocol*)

Vimos que TCP ten a robustez e funcionalidades propias dun protocolo orientado a conexión o cal conleva unha certa complexidade. O UDP, que está definido no RFC 768, é o protocolo da capa de transporte non orientado a conexión. Xa que logo, non garante que os datos se entreguen en orde (nin sequera que se entreguen, xa que non existen asentementos ou ACK), nin que a conexión se recupere de erros. En consecuencia, é máis rápido que TCP pero tamén máis inseguro.

Por que UDP

As aplicacións que non requiren unha fiabilidade total e non poden tolerar o retardo producido pola complexidade de TCP usan UDP. Algúns exemplos deste tipo de aplicacións son:

- Transmisión de vídeo ou audio en tempo real
- Aplicacións que requiren o envío dunha ou dúas mensaxes unicamente
- Sincronización de reloxos (NTP)
- Consultas ao servidor de nomes (DNS)
- Mensaxes de xestión da rede (SNMP)
- Obtención dinámica de dirección IP (DHCP)

Tamén usan UDP as aplicacións interesadas en transmitir información en modo *multicast* ou *broadcast*, é dicir, a un grupo de usuarios ou a todos os usuarios da rede, respectivamente. Isto só é posible cun protocolo non orientado a conexión, xa que pola súa propia natureza os protocolos orientados a conexión son punto a punto (en TCP non é posible establecer conexións multipunto).

A PDU de UDP

A Unidade de Datos do Protocolo no UDP é moi simple e os seus campos son os que se ven na seguinte táboa:

Campo	Lonxitude
Porto orixe	16 bits
Porto destino	16 bits
Lonxitude	16 bits
Checksum	16 bits
Datos	65507 bytes

O significado dos campos é o seguinte:

- **Porto orixe**: especifica o porto da aplicación que xera o datagrama. Este valerá normalmente cero, salvo que a aplicación solicite unha resposta.
- **Porto destino**: especifica o porto da aplicación á que vai dirixido o datagrama.
- **Lonxitude**: indica a lonxitude do datagrama, incluíndo os campos de cabeceira.
- **Checksum**: é opcional, como en IPv4. Cando UDP recibe un datagrama e determina que hai erros, descártao e non o entrega a ningunha aplicación sen avisar ao emisor.
- **Datos**: contén os datos a transmitir. Un datagrama UDP vai encapsulado nun datagrama IP, o cal fixa a lonxitude máxima deste campo.

O TCP (*Transport Control Protocol*)

UDP non garante a entrega de información que lle proporciona unha aplicación. Tampouco reordena a información no caso de que chegue nunha orde diferente daquela en que se transmitiu. Todo o contrario que o TCP, especificado na RFC 793, que proporciona fiabilidade á aplicación xa que garante a entrega de toda a información no mesmo orde en que foi transmitida e proporciona un servizo orientado a conexión con control de fluxo e erros, ta e como veremos a continuación.

Funcións

Entre as funcións encargadas a TCP están as seguintes:

- **Transmisión libre de erros**: entrega á aplicación de destino exactamente a mesma información que lle entregou a aplicación de orixe
- **Garante de entrega da información**: toda a información transmitida pola aplicación de orixe entrégase á aplicación de destino. Se non é posible, o TCP debe avisar á aplicación.
- **Garante de mantemento da secuencia de transmisión**: garante a entrega no mesmo orde en que lle foi entregado pola aplicación de orixe.
- **Eliminación de duplicados**: o TCP garante que só entregará unha copia da información transmitida á aplicación de destino. No caso de que reciba copias a causa do funcionamento da rede ou dos protocolos que se implementan por baixo do nivel de transporte, eliminaranse.

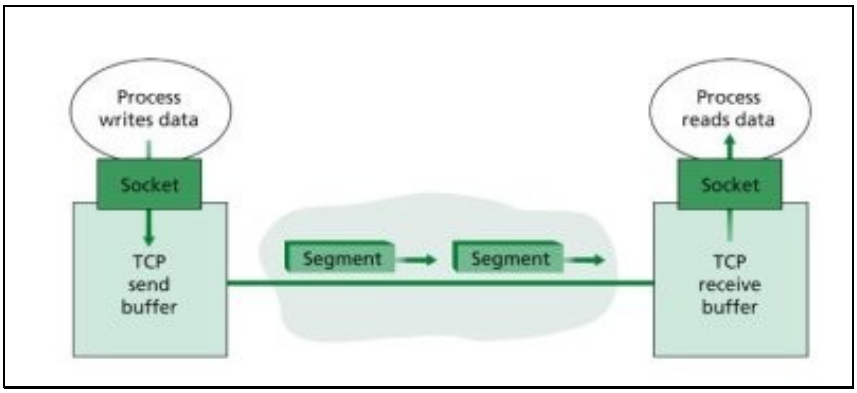
Características

Entre as principais características deste protocolo están as seguintes:

- **Orientado a conexión**: para realizar unha comunicación entre dous puntos extremos, séguense tres pasos:

1. Establecer a conexión (o cliente indícalle ao servidor que quere comunicarse con el)
2. Unha vez establecida a conexión realízase o intercambio de información
3. Finalizado o intercambio, libérase a conexión

- **Orientado a fluxos de bytes** (*stream oriented*): as aplicacións non teñen ningún modo de indicar ao TCP os límites en que queren transferir a información. É o TCP o que decide en cada momento cantos bytes transfire nun segmento. Para transmitir eses fluxos usa **buffers de envío e recepción**, tamén coñecidos como ventás de recepción. Se a aplicación xera un byte o TCP pode esperar que a memoria intermedia (o buffer) estea máis chea antes de transferir a información. Se xera fluxos de gran tamaño transfírea de inmediato (mecanismo *push*).



- **Ten un tamaño máximo de segmento**: o MSS (*Maximun Segment Size*) é o tamaño do campo de datos do segmento que se especifica durante o establecemento da conexión. Polo tanto, o MSS no inclúe as lonxitudes das cabeceiras IP e TCP (MSS = MTU ? 20 ? 20). Cada extremo especifica un MSS pero non existe unha negociación entre os extremos. Se son distintos escollerase o menor. O feito de elixir o MSS non é trivial. En xeral, canto maior sexa o MSS, mellor, posto que as cabeceiras IP e TCP se ?amortizan? máis. Con todo, se a MTU é pequena, será preciso fragmentar o datagrama IP (é dicir, o segmento TCP). Xa que logo, por norma xeral non interesa elixir MSS maiores que a MTU.
- **Usa asentimentos**: Son acuses de recibo, é dicir, segmentos que envía o receptor ao emisor para informalo de se recibiu correctamente o que o emisor enviou. A estes asentementos chámaselles ACK, do inglés *acknowledgement*. Contempla a técnica de *piggybakcing* que permite incluír ACK nun segmento de datos, co correspondente aforro fronte ao envío de segmentos específicos de recoñecemento.
- **O TCP utiliza unha conexión full dúplex**: xa o comentamos. A transferencia de información é en ambos sentidos. A aplicación ve dous fluxos independentes de bytes. No caso de que a aplicación peche un dos fluxos, a conexión pasa a ser half dúplex. Iso significa que un dos extremos (o que non pechou a conexión) pode continuar enviando información pola canle, mentres que o outro extremo (o que pechou a conexión) limítase a recoñecer a información.

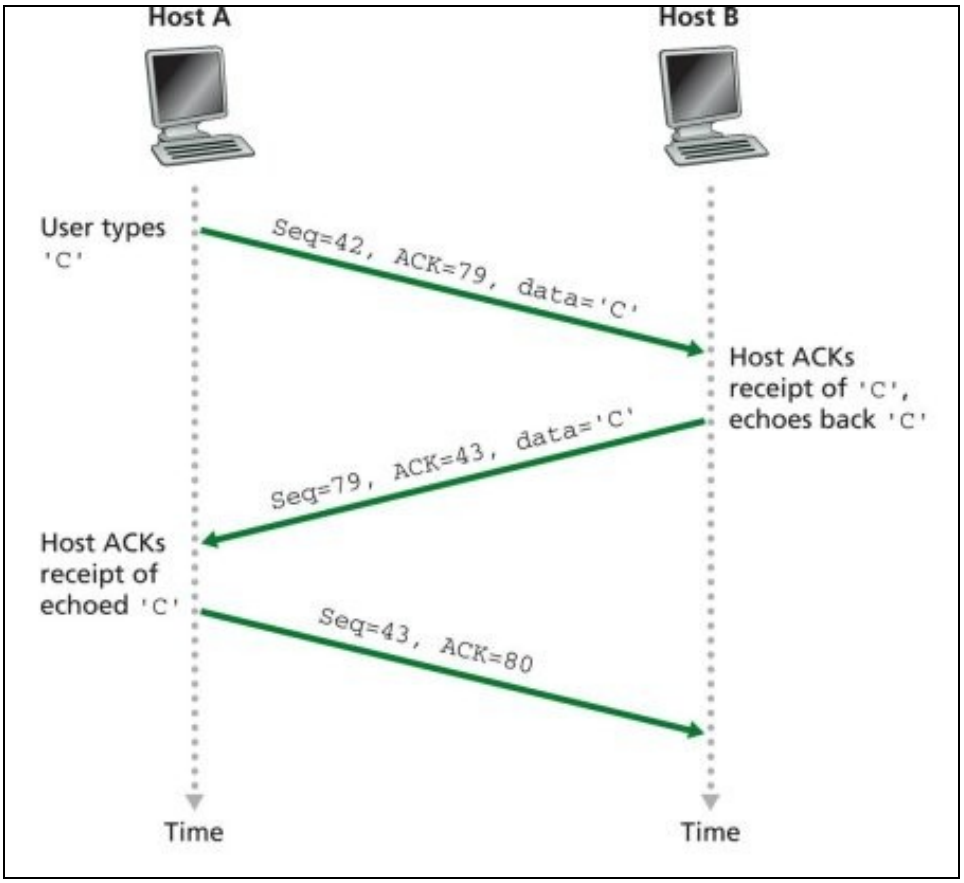
A PDU de TCP

Xa vimos que á Unidade de Datos do Protocolo TCP chámaselle segmento e o seu formato é o seguinte:

Campo	Lonxitude (bits)
Porto orixe	16
Porto destino	16
Número de secuencia	32
Número de ACK	32
Lonxitude da cabeceira TCP	4
Reservado	6
URG (Urgent)	1
ACK (Acknowledgement)	1
PSH (Push)	1
RST (Reset)	1
SYN (Synchronize)	1
FIN (Finish)	1
Tamaño da ventá	16
Checksum	16
Punteiro de datos urxentes	16
Opcións	0, 32, 64, ?
Datos	0-523960 (65495 bytes)

Os seus campos teñen o seguinte significado:

- **Porto orixe e porto destino**: identifican os portos que van a utilizar en cada equipo as aplicacións.
- **Número de secuencia**: indica o primeiro byte do campo datos dese segmento. En TCP numéranse os bytes non os segmentos, xa que é un protocolo orientado a fluxos de bytes.
- **Número de ACK**: TCP recoñece datos por medio de *piggybacking*. Ao activar un bit da cabeceira TCP (o bit ACK), indica ao outro extremo o próximo byte que está disposto a recibir, tal e como podemos ver no seguinte exemplo:



Na figura anterior vese como o host A envía un segmento co número de secuencia 42. O host B resposta cun segmento de datos que ten o bit ACK activado. O número de ACK é 43, é dicir, Número de secuencia +1, xa que logo "recoñece" o segmento anterior ou o que é o mesmo, indícalle ao host A que recibiu correctamente o segmento 42 e espera recibir o 43. Por último, o host A envía o segmento 43 co bit ACK activado e número de ACK = 80, polo que tamén recoñece o semento enviado polo host B.

- **Lonxitude de cabeceira TCP**: especifica a lonxitude en palabras de 32 bits, excluído o campo datos (o campo opcións fai que dita lonxitude poida variar).

A continuación hai 6 bits non utilizados, seguidos por outros 6 bits indicadores que se están activados (a 1) teñen un significado específico, que é o seguinte:

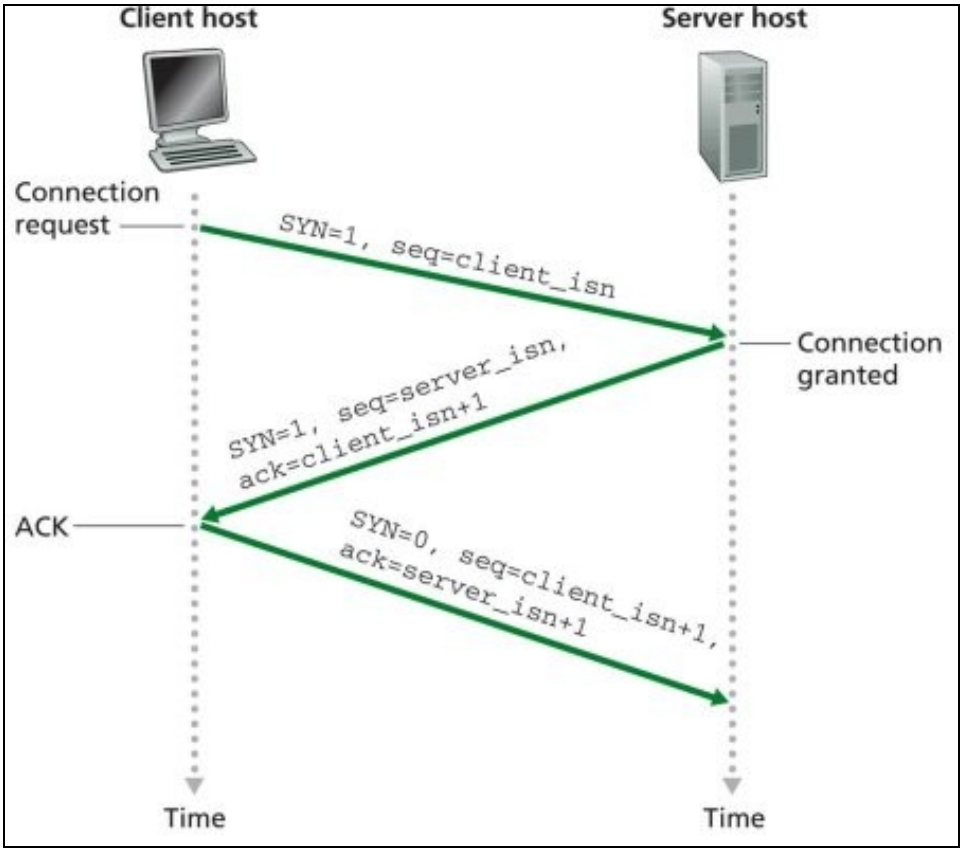
- **Bit URG** (*urgent*): serve para indicar que o segmento contén datos urxentes. Utilízase en aplicacións como telnet e rlogin cando se pulsa a tecla de interrupción, ou no FTP cando se aborta a transferencia dun ficheiro.
- **Bit ACK** (*acknowledgement*): indica que é un segmento ACK.
- **PSH** (push): indica que se lle entreguen á aplicación todos os datos que estean na memoria intermedia de recepción (no buffer), sen esperar a recibir un segmento de tamaño máximo.
- **RST** (*reset*): indica que se debe abortar unha conexión porque se detectou un erro de calquera tipo; por exemplo que se recibiu un segmento cun valor inadecuado do número de secuencia ou número de ACK.
- **SYN** (*synchronize*): este bit indica que se está establecendo a conexión. Está activado só no primeiro segmento enviado por cada un dos dous equipos no inicio da conexión.
- **FIN** (*finish*): indica que non se teñen máis datos que enviar e que se quere pechar a conexión.

Os seguintes campos da PDU de TCP son:

- **Tamaño de ventá**: indica a cantidade de bytes que se está disposto a aceptar do outro lado **en cada momento**. Suponse que se garante unha cantidade suficiente de espazo nos buffers. Mediante este parámetro o receptor establece un control de fluxo sobre o caudal de datos que pode enviar o emisor. Cada extremo terá a súa ventá de recepción.
- **Checksum**: utilízase para detectar erros.
- **O campo Urgent pointer**: ten sentido cando o bit de control URG está activo polo que os datos que envía a orixe son urxentes. Así, este campo identifica o último byte do campo de datos que é urxente.
- **Opcións**: permite engadir funcionalidades extras ao protocolo.

Establecemento da conexión

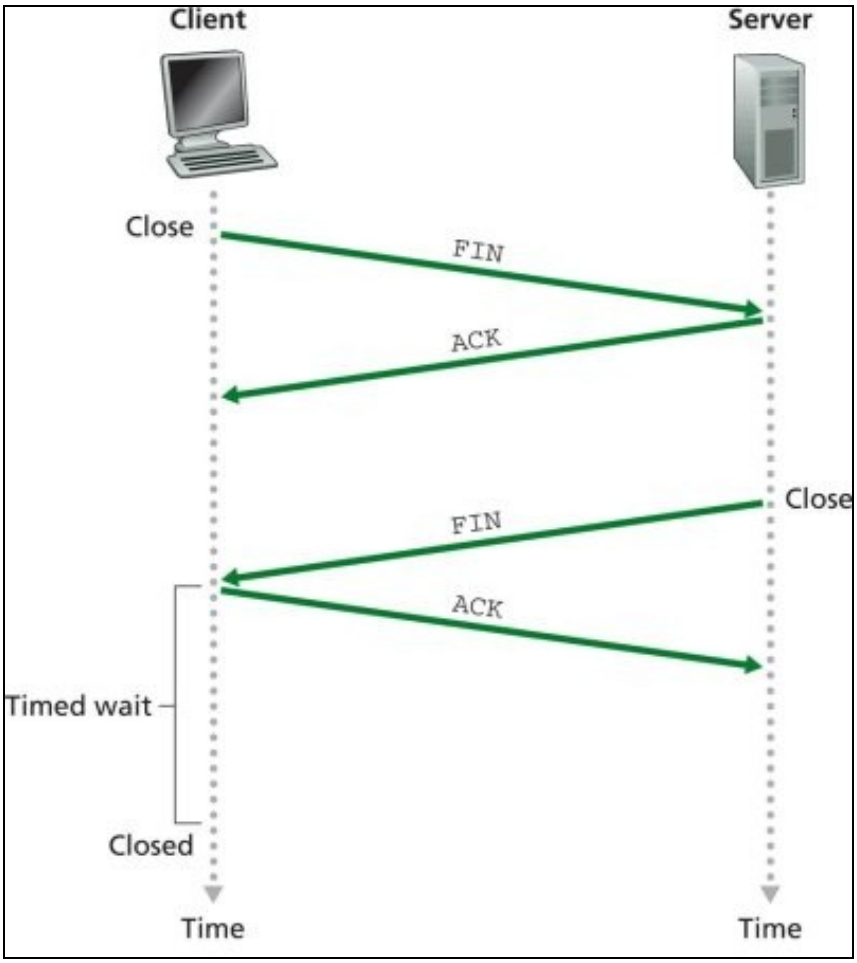
Para establecer unha conexión o TCP utiliza un mecanismo coñecido como **three-way handshake**, ou conexión a tres bandas, porque necesita tres segmentos TCP para poder establecer dita conexión. Inicialmente o servidor está nun estado de escoita, chamado *listen*. O cliente quere establecer unha conexión co servidor polo que o TCP da máquina cliente iniciará a petición de conexión, que será contestada polo TCP da máquina servidor. O intercambio de segmentos podemos velo no seguinte gráfico:



- **Paso 1: petición de conexión**. O TCP cliente envía unha petición de conexión ao servidor mediante un segmento SYN (ten o bit SYN da cabeceira activado). Especifica tamén nese segmento o ISN ou **Número de Secuencia Inicial** para empezar a numerar os bytes que se enviarán, así como outros parámetros como o MSS. O ISN escóllese ao chou.
- **Paso 2: confirmación de conexión**. O servidor responde á petición de establecemento da conexión cun segmento SYN que indica o número de secuencia inicial que utilizará el. Este segmento contén un recoñecemento (ACK) do segmento SYN do cliente que indica o ISN do cliente máis 1. TCP numera os ACK co número de secuencia do próximo byte que espera recibir, tal e como xa se comentou anteriormente.
- **Paso 3: recoñecemento da conexión**. O cliente recoñece o segmento SYN do servidor cun ACK que contén o ISN do servidor máis 1, establecendo nese momento a conexión lóxica entre o cliente e o servidor.

Quen envía o primeiro segmento SYN (neste caso, o cliente) efectúa unha **apertura activa** (*active open*). Quen recibe o primeiro segmento SYN e envía o seguinte segmento SYN (neste caso, o servidor) leva a cabo unha **apertura pasiva** (*passive open*)

Peche da conexión



Cando a transferencia de datos remata, TCP dispón dun mecanismo de finalización da conexión para pechala. Unha conexión TCP é full dúplex, os datos flúen en ambos sentidos, independentes o un do outro, polo que calquera conexión debe pecharse independentemente. O intercambio de segmentos para o peche dunha conexión TCP podémolo ver na seguinte figura:

1. **Paso 1: peche da conexión nun sentido**. O cliente envía un segmento TCP co bit FIN activado o que significa que non haberá máis datos dende o cliente ao servidor, nese sentido (cliente->servidor). O servidor envía unha confirmación de peche por medio dun segmento ACK. O TCP servidor indica á súa aplicación que o cliente pecha a conexión.
2. **Paso 2: peche da conexión no outro sentido**. O servidor envía un segmento TCP de tipo FIN ao cliente. O TCP cliente responde automaticamente cun ACK.

Quen envía o primeiro segmento FIN (neste caso o cliente) leva a cabo un **peche activo** (*active close*). Quen o recibe (neste caso o servidor) realiza un **peche pasivo** (*passive close*).

Transferencia de datos

Unha vez establecida a conexión o TCP pode empezar a transferencia de segmentos en ambos os dous sentidos. Para transmitir información de xeito fiable, TCP implementa protocolos de control de erros e de fluxo.

Cando o TCP envía datos mantén un temporizador (*timeout*) ata que recibe un ACK do receptor. Se o temporizador salta, o TCP retransmite os datos (**técnica de envío - espera**). Cando o TCP recibe un segmento de datos envía un recoñecemento. Este último pódese retornar atrasado (non de inmediato) se o TCP o considera necesario.

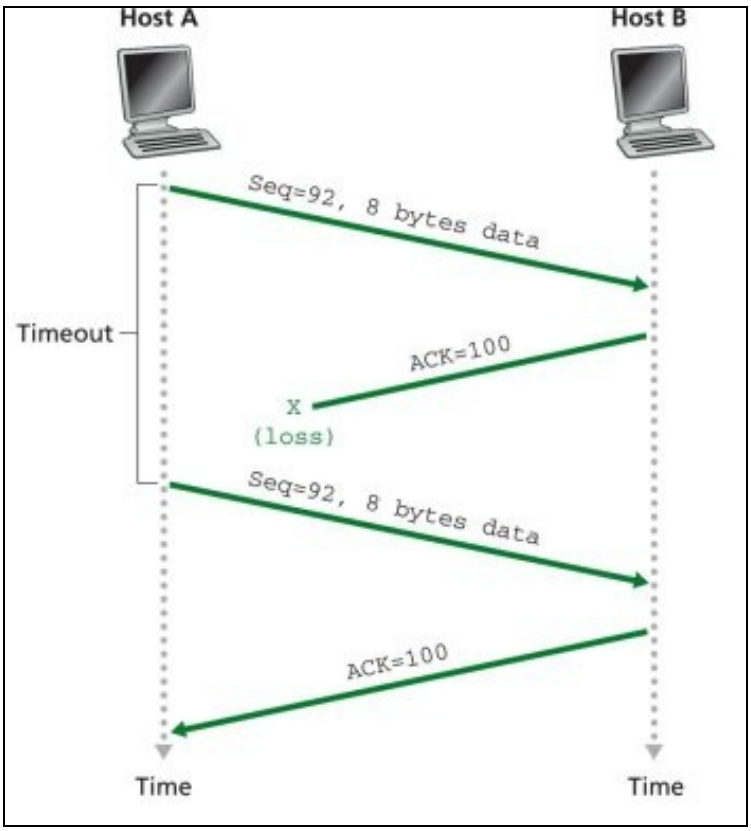
Se un segmento recibido é incorrecto (o *checksum* indícao), o TCP descártao e devolve un segmento co mesmo número de ACK que recoñecera a última vez (**ACK duplicado**). O transmisor verá un ACK cun número repetido e interpretará que non lle recoñecen a información e, xa que logo, terá que volver a enviala.

No caso de que non tivese datos para enviar en sentido contrario, o TCP pode enviar un segmento que non conteña datos. Este segmento tería o indicador ACK activado e recoñecería os bytes pertinentes no campo Número de ACK. **O número de secuencia non se incrementaría, posto que non se envían datos**.

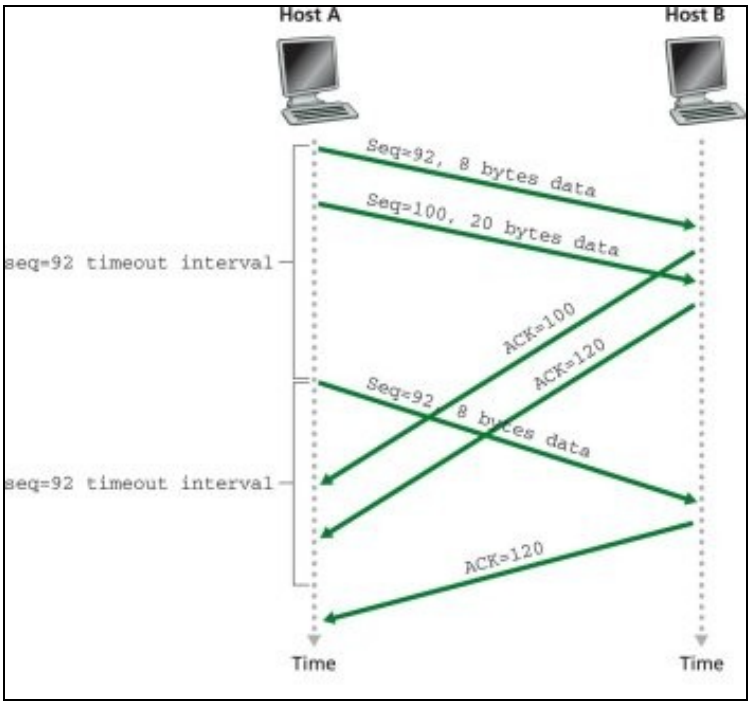
Se os segmentos chegan desordenados o TCP reordénaos e pasa os datos correctamente ordenados á aplicación. Se recibe segmentos duplicados, o TCP descarta as copias.

TCP posúe unha memoria limitada polo que é necesario que efectúe un control de fluxo: cada extremo avisa dos datos que está disposto a recibir en cada momento utilizando o campo Tamaño da Ventá da cabeceira TCP visto anteriormente. Trátase dun mecanismo de **ventá deslizante**.

Na seguinte figura vemos un escenario interesante dende o punto de vista da transferencia da información, no que se reflicte unha retransmisión debida a unha perda de segmento.



Outro escenario interesante é o da seguinte figura na que se amosa o caso dun ACK duplicado.



Ambos escenarios usan **ACK acumulativos**, é dicir, un único segmento ACK pode recoñecer varios segmentos de golpe.

Control de fluxo

As memorias intermedias de recepción dos extremos TCP pódense encher, xa que logo, é necesario un protocolo de ventá deslizante (*sliding window*) para controlar o fluxo de datos. A idea é que cada extremo TCP regula a cantidade de datos que o outro extremo pode transmitir. Por iso, cada extremo notifica ao outro, cada vez que envía un segmento, a ventá que pode aceptar nese momento.

A cabeceira do segmento TCP especifica tres parámetros para a técnica de ventá deslizante:

- **O número de secuencia**, que indica á súa conexión oposta o primeiro byte de datos que contén o segmento transmitido.
- **O número de ACK**, que indica á súa conexión oposta o próximo byte que espera recibir e, xa que logo, o último byte recibido correctamente.
- **O tamaño da ventá**, que indica á súa conexión oposta o tamaño da memoria intermedia de recepción e, xa que logo, o tamaño da fiestra que o transmisor debe utilizar.

O protocolo de ventá deslizante consiste en establecer límite no números de bloques de información que o emisor pode enviar sen recibir acuse de recibo deles. O tamaño da ventá deslizante en TCP mídese en bytes, isto é, cantos bytes se van poder enviar sen estar pendente do acuse de recibo.

--Arribi 12:04 6 oct 2009 (BST)