6. Protocolos de Transporte

6.1 TCP

6.1.1 Introducción

En una red datos, sobre todo cuando en la misma los protocolos fundamentales de la capa de red y enlace de datos son del tipo orientado a la no conexión, existe la necesidad de emplear un protocolo que controle los procesos de trasmisión y responder ante la duplicación, pérdida y recepción desordenada de los paquetes.

La unidad de información en las redes de datos que emplean el protocolo de control de transporte (TCP) se denomina **segmento**. Existen mecanismos del protocolo de control de transporte que permiten controlar la entrega de la información entre el origen y el destino que describiremos a lo largo de este capitulo. TCP permite que los segmentos entre el origen y el destino lleguen en orden, no se pierdan ni se duplique. TCP esta normalizado en la RFC 793.

6.1.2 Envío de datos

En condiciones normales la transmisión entre dos pares se realiza de la siguiente manera, donde para este caso el retardo de transmisión es la mitad del intervalo de transmisión (tics}.

La diferencia entre el tiempo de envío de un segmento y su llegada al receptor constituye el retardo de transmisión que puede variar en unción de la carga de los enrutadores o congestión en la red.

Como puede apreciarse cada vez que una estación de trabajo desea transmitir se cumplen las tres fases de un protocolo orientado a la conexión: establecimiento de la conexión, envío de datos y control de la conexión y finalización de la conexión. Esto puede observarse en la figura 1.

Para el establecimiento de la conexión se emplean los mensajes SYN de dos vías y su correspondiente confirmación ida y vuelta. Durante el inicio de la conexión se fijan los valores iniciales de números de secuencia, el tamaño de la ventana deslizante a emplear para el control inicial de flujo y el tamaño máximo del segmento convenido, que por defecto es 536 bytes.

Para el mantenimiento de la conexión, junto con los datos de usuarios, se emplean segmentos que incluyen números de secuencia, numero de confirmaciones, datos del tipo **URGente** (bandera URG), datos de Empuje (PSH), control de errores.

Para la finalización de la conexión se emplean también mensajes de dos vías del tipo FIN.

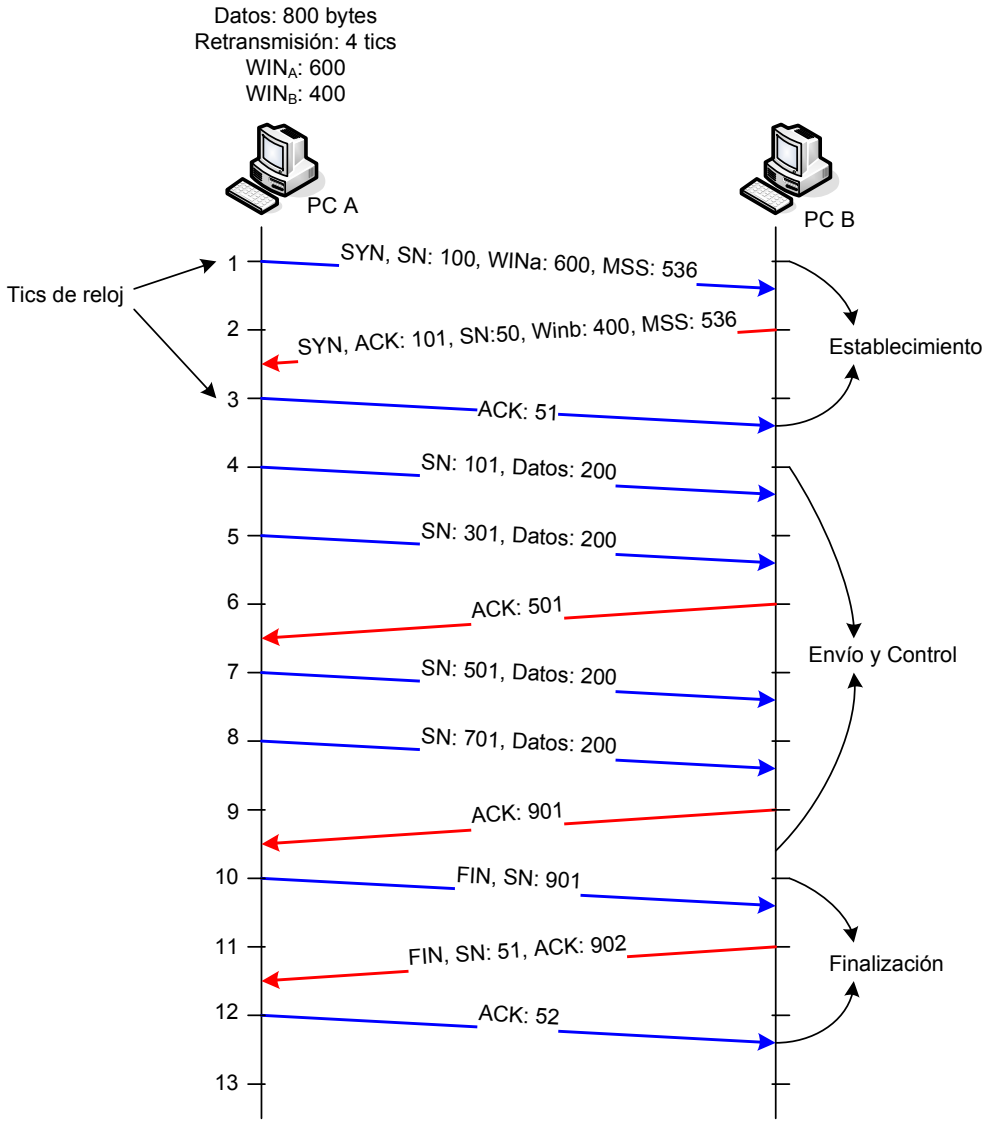


Figura 1. Envío de datos normal

6.1.3 Control de Flujo

El control de flujo es el mecanismo que emplea el protocolo de control de transporte para regular el flujo de datos entre emisores y receptores. En muchas ocasiones, el procesamiento y memoria de los equipos participantes de la transmisión de datos, tienen capacidades diferentes. Para evitar que un transmisor rápido sature o colapse a un receptor mas lento, se implementa en TCP el concepto de Ventana Deslizante.

La Ventana, consiste en la cantidad de bytes que un receptor espera recibir y recién ahí genera un mensaje de confirmación. Cuando el receptor estadísticamente determina que esta pronto a saturarse o no poder procesar

Protocolo de Transporte

los segmentos que le están llegando, emite un segmento TCP enviando un tamaño de ventana menor o hasta incluso 0 hacia el transmisor. De esta forma se regula el envío de información entre el receptor y el trasmisor.

En el siguiente esquema se observa un ejemplo de control de flujo con una ventana deslizante de 400 bytes. En un determinado momento la PC B no puede procesar mas información y previendo una saturación envía un mensaje de control cambiando el tamaño de la ventana a 0 bytes.

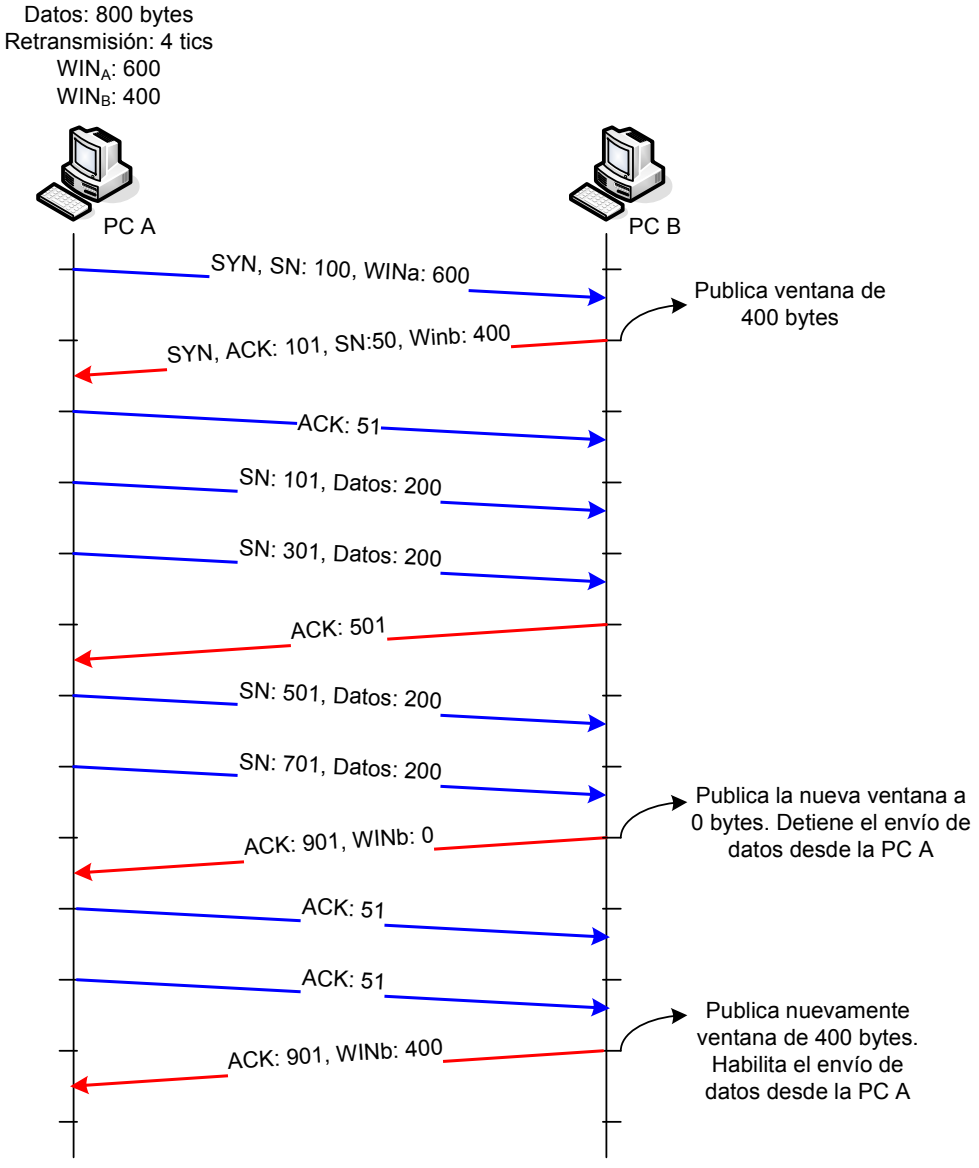


Figura 2. Control de Flujo

6.1.4 Retransmisión de segmentos

Como se observa en la figura 3, la confirmación de la PC B no llega a la PC A, motivo por el cual hace que en la PC A se expire el tiempo de retransmisión. Donde en el tics 8 se vuelve a enviar el segmento numero 101 y en el tics 9 se reenvía el segmento 301.

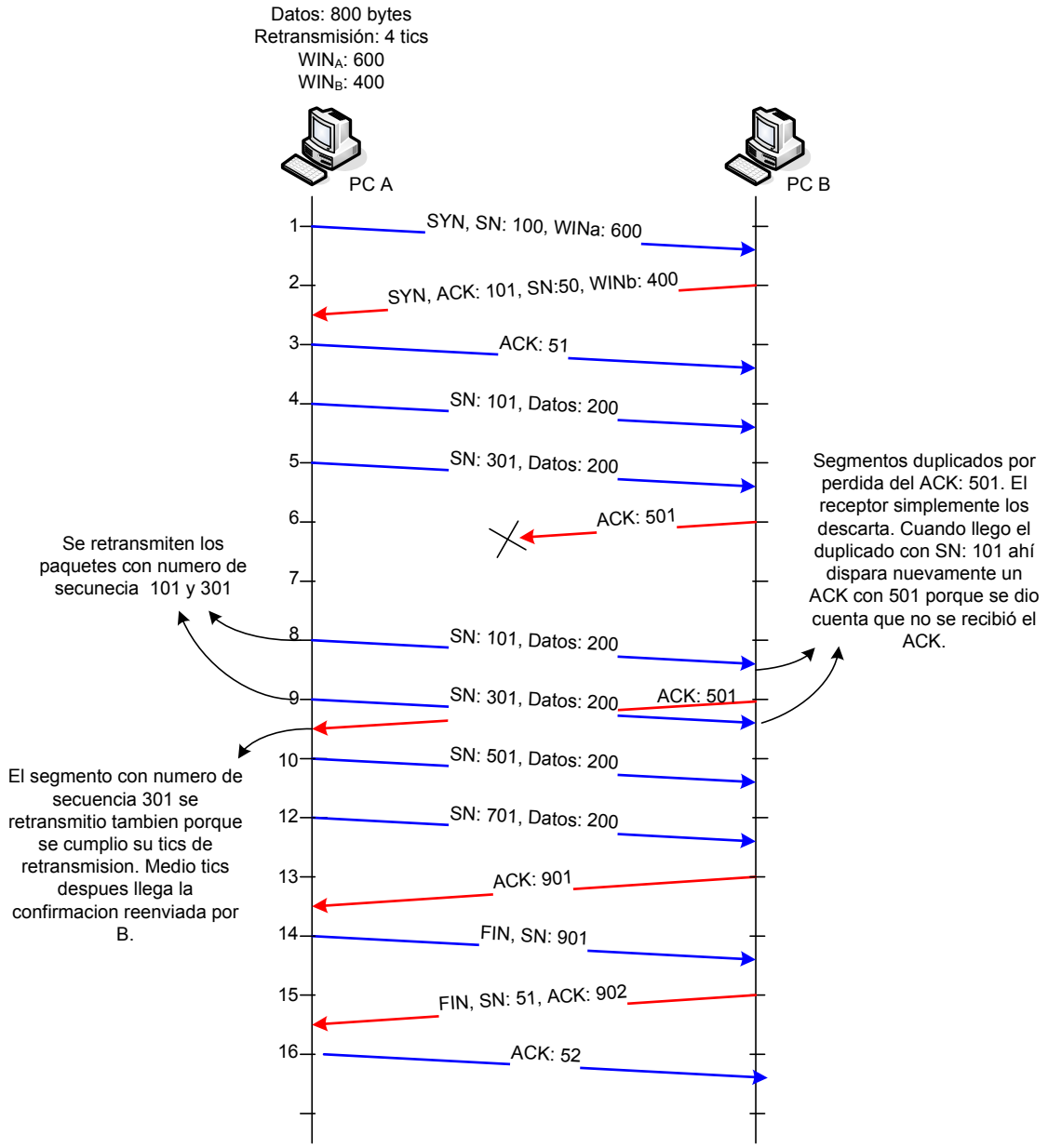


Figura 3. Retransmisión de segmentos

Protocolo de Transporte

Cuando la PC B recibe el segmento número de secuencia 101 nuevamente (duplicado), vuelve a enviar la confirmación ACK: 501 (perdida). Lamentablemente esa confirmación llega después del envío del segmento 301. Estos dos segmentos 101 y 301 son duplicados, motivo por el cual la PC B los descarta. Así es como TCP resuelve los segmentos duplicados.

El problema es encontrar un valor de tiempo de retransmisión adecuado a la aleatoriedad del retardo de la red. En una red de datos existen picos de saturación y momentos de baja carga, lo que hacen que el tiempo de retransmisión deba variar para que:

- Si el retardo de retransmisión es corto pero la congestión es más grande que este, se originan muchos duplicados debido a las retransmisiones excesivas.

- Si el retardo de retransmisión es grande pero la congestión es baja, se demora la transmisión de la información.

Para solucionar este problema se establece un calculo variable del tiempo de retransmisión, para que este se adapte a la congestion de la red.

RTO: retardo de retransmsion.
RTT: retardo ida y vuelta para el calculo del tiempo de retransmisión.

TimeOut = Avg RTT + 4* Desv

Se suele fijar en multiplos de 200, 500 y 1000 milisegundos.

Para evitar ambigüedades en el calculo del RTT como sucede en las figuras siguientes, se emplea el estimador de Karm.

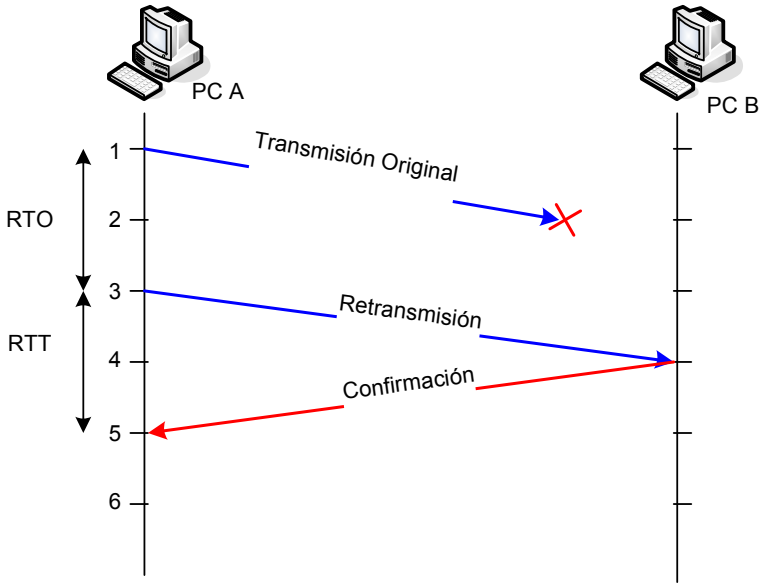
- **Ante una retransmisión**:
  - No actualizar los RTT
  - Timer backoff
    - RTO = 2*RTO

- **Volver a estimar después de una transmisión exitosa**.

- **Las muestras del RTT se toman en base a la opción de Time Stamp de TCP**.

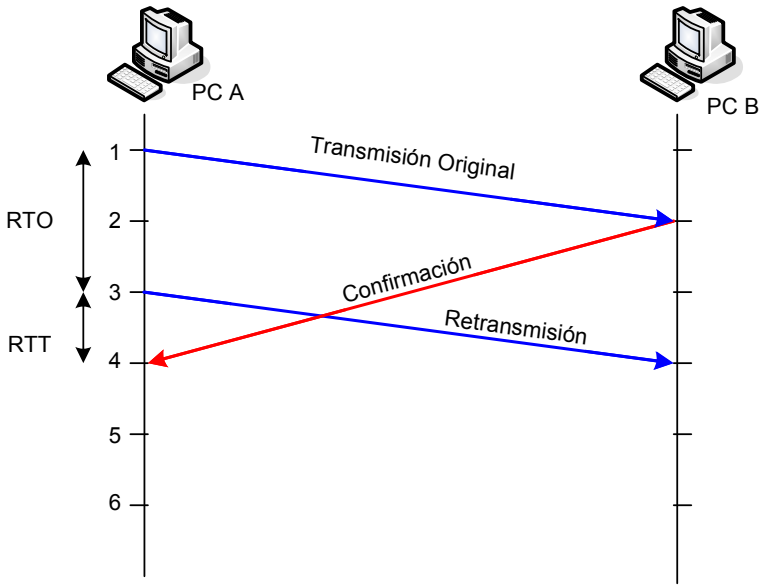
Protocolo de Transporte

Calculo del RTT. Confusión con los paquetes retransmitidos.



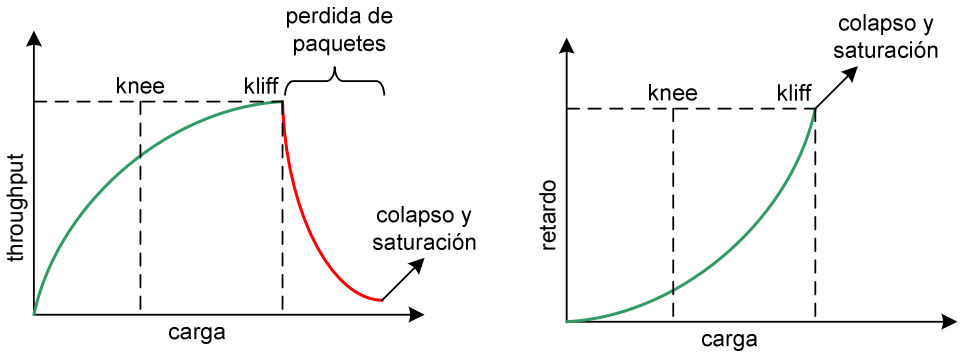
Se pierde el segmento original y se retransmite. Se calcula erróneamente el RTT ya que no se tiene en cuenta el RTO debido a la perdida del paquete lo que es probable que se halla originado por una congestión en la red.

Calculo del RTT. Confusión con la llegada tarde a la confirmación.



6.1.5 Congestión y retardo

En las siguientes figuras se observa la carga de la red en función de la utilización de la misma y en función del retardo. Cuando mas carga existe en la red, mayor será el RTT pero mas eficiente será el uso de la red aumentando el **"throughput"** casi a la capacidad máxima del enlace. El objetivo de los mecanismos de control de congestión es mantenerse antes del punto de saturación máximo conocido como **"kliff"** (o precipicio), permitiendo la maxima eficiencia o **throughput** de la red.



Controlar la congestión es mantenerse en la **izquierda** del punto quiebre o precipicio conocido como **"cliff"**. **Evitar** la congestión es un mecanismo mas conservador donde no se llega al limite del retardo máximo, carga y saturación sino que el mecanismo permite mantenerse a la **izquierda** del codo o **"knee"**.

Existen dos enfoques en el control de congestión, uno extremo a extremo como lo implementa TCP, donde las estaciones que participan de la transmisión son las encargadas de estimar los tiempos y grado de congestión y reducir la demanda y los nodos intermedios son los encargados de monitorear el estado de la red. El otro enfoque es basado en red, donde los datos relevados por los extremos trasmisores no son confiables y el nodo de red tiene el control sobre el tráfico y toma acciones más rápidamente.

6.1.5.1 Control de Congestión

Para el control de congestión TCP emplea tres variables.

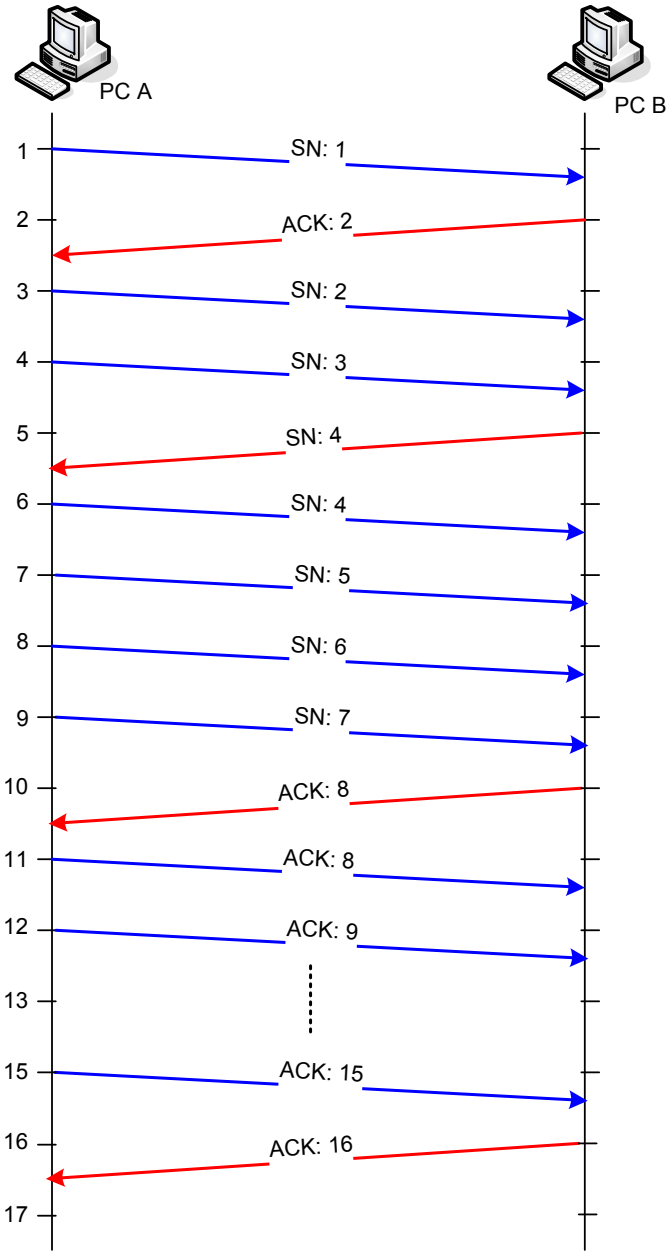
- **cwnd**: ventana de congestión.
- **rcv_win**: ventana de recepción. Publicada en el segmento.
- **ssthresh**: valor de umbral.

Para enviar datos un transmisor debe calcular su tamaño de ventana en base a:

$$win = \min(rcv\_win, cwnd)$$

Protocolo de Transporte

Para descubrir la congestión se emplea el algoritmo **slow atart**.



- Inicializa el sistema y descubre la congestión rápidamente.
- Incrementa **cwnd** hasta la congestión, donde se estima el óptimo **cwnd**.
- Se Detecta congestión por pérdida de segmentos.
- Desventajas
  - Detección tardía
  - Enlaces de alta velocidad, tienen ventanas de envío pero ante perdidas es mayor la cantidad de información que se pierde.

Protocolo de Transporte

- Interacción con el algoritmo de retransmisión y timouts.

- Inicialmente cuando arranca el algoritmo o después de la congestión es valor de **cwnd=1**.
- Después de cada ACK recibido la ventana de congestión es:
  - **cwnd=(cwnd + 1)**

Como se observa en la figura el emisor (PC A) emite primero un segmento, al recibir confirmación envía dos segmentos, al recibir confirmación envía 4 segmentos, al recibir confirmación envía 8 segmentos y así sucesivamente hasta que pierde un segmento. En ese momento determino la ventana de congestión.

6.1.5.2 Evitar la Congestión

Consiste en mmantenerse a la izquierda del **knee** para lo cual se emplea una combinación de **"Slow Start"** y evitación de la congestión.

1. Inicialmente cuando comienza el envío de segmentos se inicia con un cwnd=1 y un **"ssthresh"** de infinito.

2. A medida que va recibiendo los ACK confirmando segmentos el envío de estos sigue el patrón de **"Slow Start"** cuando **cwnd < ssthresh**.

3. Si no se reciben ACK, ahora la ventana de congestión va creciendo de a 1 segmento (cesa el slow start exponencial).

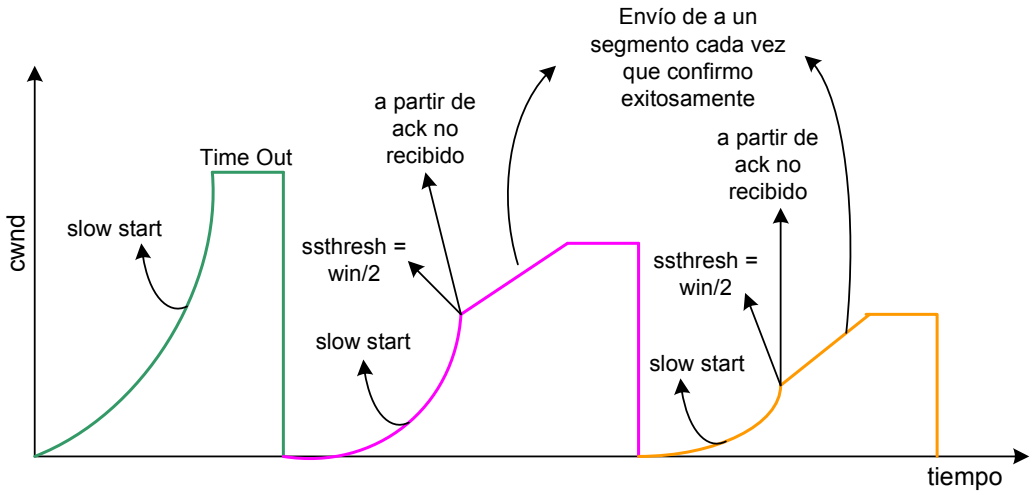
   cwnd = cwnd + 1/cwnd

4. Cuando comienza la perdida de paquetes (tiempo de retransmisión o timeout).

   ssthresh = win/2; donde win = min(cwnd,rcv_win);

   cwnd = 1;

Protocolo de Transporte

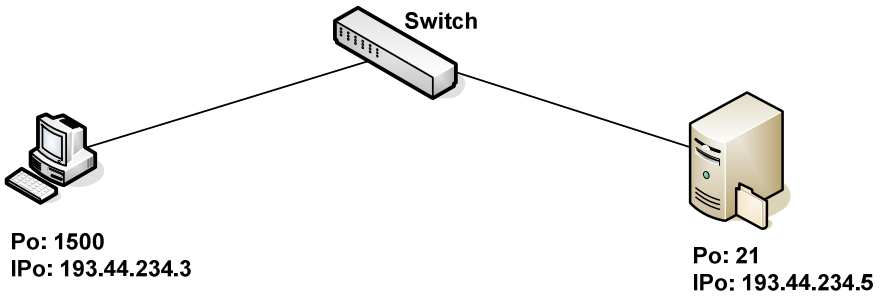


6.1.6 Sesión TCP

Una sesión TCP entre pares de aplicaciones transmitiendo se identifica por los siguientes 5 parámetros.

Sesión TCP = (numero de protocolo, Puerto Origen, IP de Origen, Puerto Destino, IP de Destino)

Por ejemplo, la sesión = (tcp, 1500, 193.44.234.3, 21, 193.44.234.5), indica que la aplicación de origen de esta sesión emplea el protocolo TCP como transporte, utiliza el puerto de origen aleatorio 1500, con IP de origen 193.44.234.3, se dirige hacia el puerto de destino 21 que identifica a un aplicativo destino FTP, con la IP de destino 193.44.234.5.

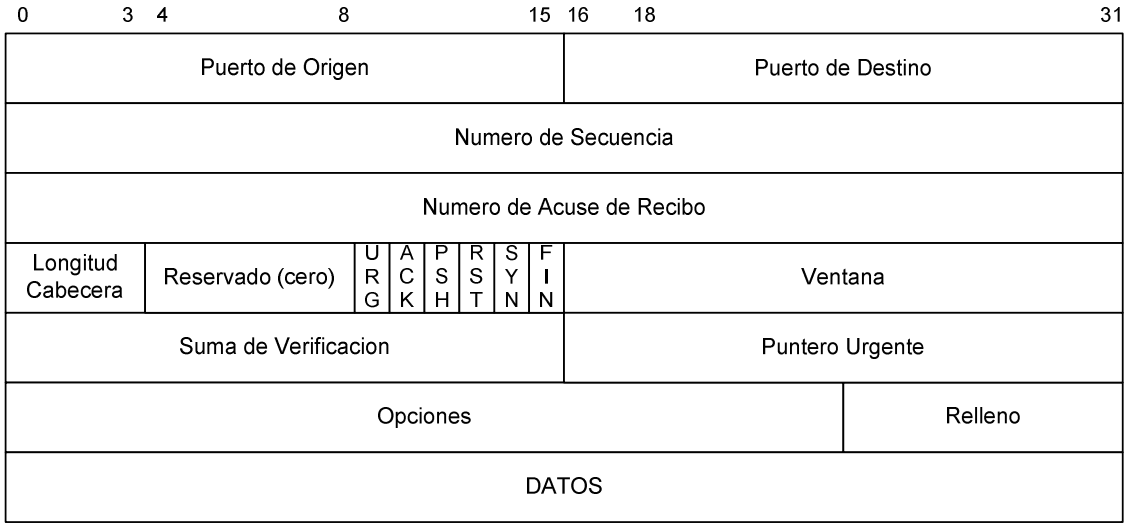


6.1.7 Formato del Segmento TCP

Los segmentos de TCP se envían como datagramas de Internet. La cabecera del protocolo de Internet transporta varios campos de información, entre los que se incluyen las direcciones de los 'host' de origen y de destino. Una cabecera de TCP sigue a la cabecera de Internet, aportando

Protocolo de Transporte

información específica del protocolo de TCP. Esta división permite la existencia de otros protocolos de la capa de 'host' distintos de TCP.



- **Puerto de origen (16 bits):**
  El número del puerto de origen.

- **Puerto de destino (16 bits):**
  El número del puerto de destino.

- **Número de secuencia (32 bits)**:
  El número de secuencia del primer octeto de datos de este segmento (excepto cuando el indicador SYN esté puesto a uno). Si SYN está puesto a uno es el número de secuencia original (ISN: 'initial sequence number') y entonces, el primer octeto de datos es ISN+1. TCP emplea número de secuencias basado en flujo de bytes enviados, es decir si se enviaron 300 bytes de datos con un numero de secuencia inicial de 150, el próximo segmento a enviar tendrá un numero de secuencia de 450 bytes. Si este nuevo segmento tiene que enviar 500 bytes datos, el próximo segmento a enviar tendrá un número de secuencia de 950 bytes.

- **Número de acuse de recibo (32 bits):**
  Si el bit de control ACK está puesto a uno, este campo contiene el valor del siguiente número de secuencia que el emisor del segmento espera recibir. Una vez que una conexión queda establecida, este número se envía siempre.

- **Posición de los datos (4 bits):**
  El número de palabras de 32 bits que ocupa la cabecera de TCP. Este número indica dónde comienzan los datos. La cabecera de TCP (incluso una que lleve opciones) es siempre un número entero de palabras de 32 bits.

- **Reservado (6 bits):**
  Reservado para uso futuro. Debe valer 0.

- **Bits de control (6 bits - de izquierda a derecha):**

Protocolo de Transporte

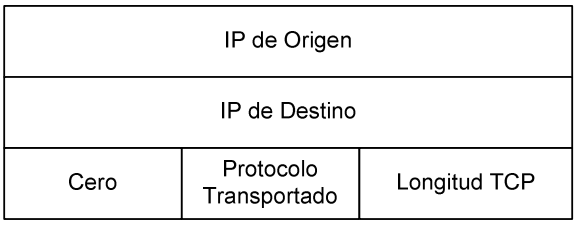
- **URG**: Hace significativo el campo "Puntero urgente"
- **ACK**: Hace significativo el campo "Número de acuse de recibo"
- **PSH**: Función de "Entregar datos inmediatamente" ('push')
- **RST**: Reiniciar ('Reset') la conexión
- **SYN**: Sincronizar ('Synchronize') los números de secuencia
- **FIN**: Últimos datos del emisor

- **Ventana (16 bits):**
  El número de octetos de datos, a contar a partir del número indicado en el campo de "Número de acuse de recibo", que el emisor de este segmento está dispuesto a aceptar.

- **Suma de control (16 bits):**
  El campo "Suma de Verificación" es el complemento a uno de 16 bits de la suma de los complementos a uno de todas las palabras de 16 bits de la cabecera y del texto. Si un segmento contiene un número impar de octetos de cabecera y texto, el último octeto se rellena con ceros a la derecha para formar una palabra de 16 bits con el propósito de calcular la suma de control. En el cálculo de la suma de control, el propio campo suma de control se considera formado por ceros.

  La suma de control también incluye una pseudocabecera de 96 bits prefijada imaginariamente a la cabecera TCP. Esta pseudocabecera contiene la dirección de origen, la dirección de destino, el protocolo, y la longitud del segmento de TCP. Esto proporciona una protección ante segmentos mal encaminados. Esta información es transportada por el protocolo de internet y es transferida a través de la interfaz TCP/Red en los argumentos o en los resultados de las llamadas de TCP a IP.

  La "longitud TCP" consiste en la suma de la longitud de la cabecera de TCP más la de los datos en octetos (esto no es una cantidad transmitida explícitamente, sino que ha de calcularse), y no incluye los 12 octetos de la pseudo cabecera.



- **Puntero urgente (16 bits)**:
  Este campo indica el valor actual del puntero urgente como un desplazamiento positivo desde el número de secuencia de este segmento. El puntero urgente apunta al número de secuencia del octeto al que seguirán los datos urgentes. Este campo es interpretado únicamente si el bit de control URG está establecido a uno.

- **Opciones: variable**

Los campos de opciones pueden ocupar un cierto espacio al final de la cabecera de TCP, pero siempre de una longitud múltiplo de 8 bits.

6.2 UDP

6.2.1 Introducción

UDP es un protocolo, definido en la **RFC 768**, que opera en la capa de transporte que solo agrega identificación de las aplicaciones a través de los puertos. No realiza ningún tipo de control sobre los segmentos enviados, es por ello que se dice que es un protocolo orientado a la no conexión.

Permite el envío de datagramas a través de la red sin que se haya establecido previamente una conexión, ya que el propio datagrama incorpora suficiente información de direccionamiento en su cabecera. Tampoco tiene confirmación ni control de flujo, por lo que los paquetes pueden adelantarse unos a otros tampoco se sabe si ha llegado correctamente, ya que no hay confirmación de entrega o recepción.

Su uso principal es para protocolos como DHCP, BOOTP, DNS y demás protocolos en los que el intercambio de paquetes de la conexión/desconexión son mayores no son rentables con respecto a la información transmitida, así como para la transmisión de audio y vídeo en tiempo real, donde no es posible realizar retransmisiones por los estrictos requisitos de retardo que se tiene en estos casos.

UDP sólo añade multiplexado de aplicación y suma de verificación de la cabecera y la carga útil. Cualquier tipo de garantías para la transmisión de la información deben ser implementadas en capas superiores.

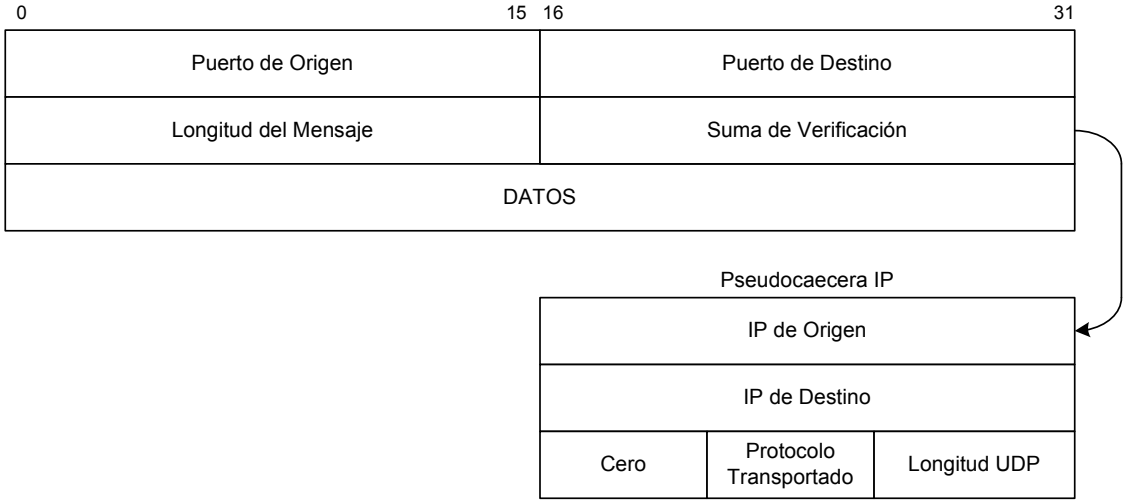


Figura. Formato del segmento UDP

La cabecera UDP consta de 4 campos de los cuales 2 son opcionales (con fondo rojo en la tabla). Los campos de los puertos fuente y destino son campos de 16 bits que identifican el proceso de origen y recepción. Ya que UDP carece de un servidor de estado y el origen UDP no solicita respuestas,

el puerto origen es opcional. En caso de no ser utilizado, el puerto origen debe ser puesto a cero.

A los campos del puerto destino le sigue un campo obligatorio que indica el tamaño en bytes del datagrama UDP incluidos los datos. El valor mínimo es de 8 bytes. El campo de la cabecera restante es una suma de comprobación de 16 bits que abarca una pseudo-cabecera IP (con las IP origen y destino, el protocolo y la longitud del paquete UDP), la cabecera UDP, los datos y 0's hasta completar un múltiplo de 16.

El protocolo UDP se utiliza por ejemplo cuando se necesita transmitir voz o vídeo y resulta más importante transmitir con velocidad que garantizar el hecho de que lleguen absolutamente todos los bytes.

UDP utiliza puertos para permitir la comunicación entre aplicaciones. El campo de puerto tiene una longitud de 16 bits, por lo que el rango de valores válidos va de 0 a 65.535. El puerto 0 está reservado, pero es un valor permitido como puerto origen si el proceso emisor no espera recibir mensajes como respuesta.

Los puertos 1 a 1023 se llaman puertos "bien conocidos" y en sistemas operativos tipo Unix enlazar con uno de estos puertos requiere acceso como superusuario.

Los puertos 1024 a 49.151 son puertos registrados.

Los puertos 49.152 a 65.535 son puertos efímeros y son utilizados como puertos temporales, sobre todo por los clientes al comunicarse con los servidores.

6.2.2 Comparación TCP y UDP

- **UDP**: proporciona un nivel de transporte no fiable de datagramas, ya que apenas añade la información necesaria para la comunicación extremo a extremo al paquete que envía al nivel inferior. Lo utilizan aplicaciones como NFS (Network File System) y RCP (comando para copiar ficheros entre ordenadores remotos), pero sobre todo se emplea en tareas de control y en la transmisión de audio y vídeo a través de una red. No introduce retardos para establecer una conexión, no mantiene estado de conexión alguno y no realiza seguimiento de estos parámetros. Así, un servidor dedicado a una aplicación particular puede soportar más clientes activos cuando la aplicación corre sobre UDP en lugar de sobre TCP.

- **TCP**: es el protocolo que proporciona un transporte fiable de flujo de bits entre aplicaciones. Está pensado para poder enviar grandes cantidades de información de forma fiable, liberando al programador de la dificultad de gestionar la fiabilidad de la conexión (retransmisiones, pérdida de paquetes, orden en el que llegan los paquetes, duplicados de paquetes...) que gestiona el propio protocolo. Pero la complejidad de la gestión de la fiabilidad tiene un coste en eficiencia, ya que para llevar a cabo las gestiones anteriores se tiene que añadir bastante información a los paquetes que enviar. Debido a que los paquetes para enviar tienen un tamaño máximo, cuanta más información añada el protocolo para su gestión, menos información que proviene de la aplicación podrá contener ese paquete (el segmento TCP tiene una sobrecarga de 20 bytes en cada segmento, mientras que UDP solo añade 8 bytes). Por eso, cuando es más importante la velocidad que la fiabilidad, se utiliza UDP. En

cambio, TCP asegura la recepción en destino de la información para transmitir.

6.2.3 Transmisión de vídeo y voz

UDP es generalmente el protocolo usado en la transmisión de vídeo y voz a través de una red. Esto es porque no hay tiempo para enviar de nuevo paquetes perdidos cuando se está escuchando a alguien o viendo un vídeo en tiempo real.

Ya que tanto TCP como UDP circulan por la misma red, en muchos casos ocurre que el aumento del tráfico UDP daña el correcto funcionamiento de las aplicaciones TCP. Por defecto, TCP pasa a un segundo lugar para dejar a los datos en tiempo real usar la mayor parte del ancho de banda. El problema es que ambos son importantes para la mayor parte de las aplicaciones, por lo que encontrar el equilibrio entre ambos es crucial.