Clase 8- control de errores

miércoles, 16 de octubre de 2024 12:17

Tiene un mecanismo protocolar que ordena los segmentos que llegan fuera de orden y recuperarse mediante solicitudes o retransmisiones de los segmentos perdidos o con errores.

- Mecanismo que se usa sobre un canal no confiable.
- Se realiza con ARQ: Automatic Repeat reQuest/Automatic Repeat Query, End-to-End.
- Utiliza n'umeros de secuencia y confirmaciones para validar que los datos se recibieron OK(en orden y sin errores)
- Se confirmal los datos recibidos
- Usa timer: RTO o TMOUT
- Requiere mantener buffer de segmentos a transmitir TxBuf y posiblemente para segmentos recibidos RxBuf.
- Puede corroborar mediante checksum/CRC errores de bits en los datos recibidos.

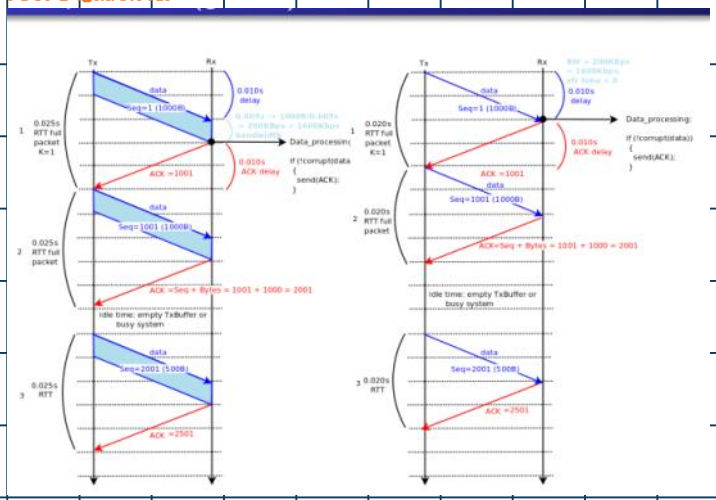
Errores que se pueden dar:

- Que se pierdan datos o acks por descartes o errores en la red
- Que se dupliquen datos o acks por retransmisiones o errores de equipos
- Que se desordenen
- Que se corrompan datos o acks

Stop and wait (no usado por tcp)

El emisor manda un segmento numerado y espera confirmación
El emisor arranca un rto cuando envia el segmennto, si no recibe el ack lo vuelve a mandar
El receptor avisa cada vez que recibe un segmento
El receptor si tiene errores descarta o confirma avisando que numero espera. Es como un nak (not ak)
El emisor si recibe la confirmación manda un nuevo segmento si es que tiene en el txBuf e inicia RTO.
El emisor descarta las confirmaciónes fuera de secuencia

Resumen: el emisor manda un bloque y hasta que no recibe ack no manda otro segmento
Si el receptor recibe, manda ack indicando lo que quiere recibir (o solo ack).
Si el emisor no recibe ack en x tiempo reenvia el segmento y considera que se perdió.
El receptor puede también detectar errores (por ejemplo, mediante un checksum) y, si encuentra uno, puede descartar el segmento. Sin embargo, no envía un ACK para ese segmento, lo que lleva al emisor a retransmitirlo.

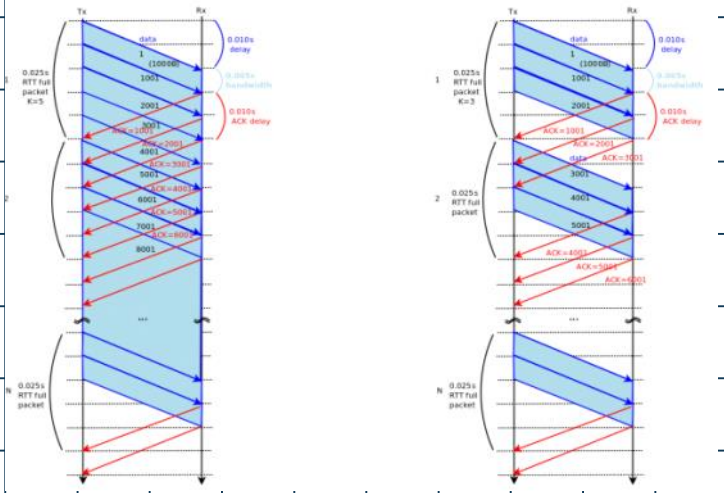


Pipelining/sliding window

Permite enviar vvarios segmentos en ráfaga sin que se haya recibido confirmaciones.
El emisor tiene que saber cuantos segmentos puede mandar sin recibir confirmación--> ventana
Requiere buffering de Tx, TxBuf (lo que deja la capa superior y aún no se confirmó) y de Rx, RxBuf (lo que se va recibiendo hasta entregar a la capa superior).
Por cada mensaje mandado se puede iniciar un timer de retransmisión RTO que se mantiene por ráfaga para el

segmento más viejo que todavía no se confirmó
El receptor tiene que confirmar que recibió las cositas
Al confirmar se desliza la ventana sobre Txbuf permitiendo transmitir nuevos segmentos

Resumen: se mandan varios paquetes en ráfaga. Cada paquete tiene un nro de secuencia para rastrear los segmentos y gestionarlos bien.
El receptor manda akcs. Pueden ser individuales (1 por paquete) o acumulativos: ack para el ultimo paquete recibido en orden lo que implica que tb recibió los anteriores
Menos latencia, más eficiente, usa mejor el ancho de banda.
Es complejo y hay que asegurarse que se recibe en orden



Go-back N

Hay una ventana que limita el max nro de segmentos
No admite segmentos ni confirmaciones fuera de orden. Solo se confirman por la positiva de los segmentos que se colocaron en el buffer en orden.
Se puede confirmar desde N para atrás (ack acumulativos). No se confirma cada segmento
~~Confirmar cada segmento tiene ventajas si se pierden ack~~
El emisor transmite tantos segmentos como w admita según lo que tiene en el buffer.

Emisor:

- Si tienme datos en el TxBuf y la ventana lo permite, los transmite
- Si no tiene RTO activo arranca uno nuevo para el primero que manda
- Si recibe ACK en orden desliza la ventana. Si hay segmentos in flight arranca un nuevo rto y sino lo descarta.
- Si se vence el RTO re-envia todo a partir del segmento más viejo que nmo se confirmó.
- Puede haber más de un segmento "in-flight", más de un ACK "in-flight".

Receptor:

- Si recibe segmento en orden confirma indicando el próximo que espera
- Si el segmento está corrupto lo descarta y espera retransmisión
- Si recibe un segmento fuera de orden confirma indicando que espera uno anterior
    - ~~Puede descartar el segmento fuera de orden, ya que será retransmitido.~~
    - Puede bufferear el fuera de orden, pero no entregar a capa superior, esperando que llegue el/los segmento/s que llena/n el hueco y luego confirmarlo

Resumen: permite mandar varios paquetes antes de recibir confirmación y tiene un mecanismo para gestionar la retransmisión si se detecta error o pierde un paquete
En Go-Back-N, el emisor y el receptor utilizan una ventana deslizante. La ventana define el número máximo de paquetes que se pueden enviar antes de recibir un ACK (confirmación) del receptor.

Característica	Pipelining	Go-Back-N
Tipo	Técnica general de transmisión	Protocolo específico
Manejo de paquetes perdidos	Permite aceptar paquetes fuera de orden	No permite; retransmite desde el último ACK
Eficiencia en la retransmisión	Más eficiente, solo retransmite el paquete perdido	Menos eficiente, retransmite todos los paquetes a partir del perdido
Ejemplos de implementación	TCP, HTTP	Protocolo Go-Back-N

Selective repeat:

Go back n si hay perdidas retransmite segmentos innecesarios si se perdió solo uno en la mitad y hubo timeout.
Este solo restransmite los no confirmaron
El receptopr confirma de a uno o usa intervalos de confirmación
NO SE USAN CONFIRMACIONES ACUMULATIVAS
No se deben confundir los segmentos de diferentes r´afagas. No se deben reusar #ID/SEQ hasta asegurarse que tiene todos los mensajes previos o estos no están en la red.
La ventana se desliza sin dejar huecos, desde los confirmados más viejos

Emisor:

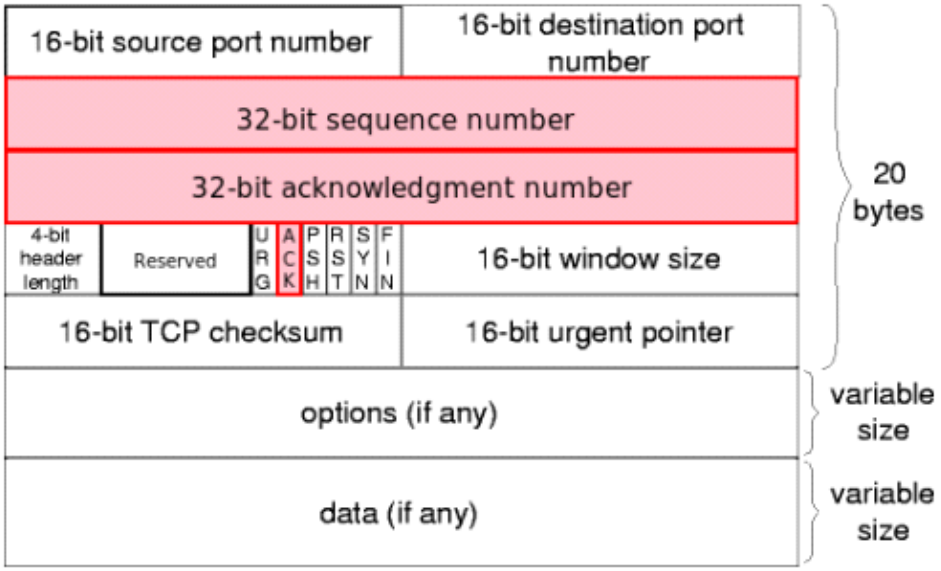
- Si tiene datos en el buffer y la ventana lo permite los retransmite como GBN(go back n)
- Cada segmento requiere un RTO. Se puede simular con solo un timer.
- Si recibe un ack en orden, desliza la ventana y para el RTO para el segmento que se confirmó.
- Si recibe un ack fuera de orden, no actualiza la ventana, para el rto para el segmento confirmado y lo marca como recibido
- Si vence el rto, reenvía el segmento asociado

Receptor:

- Si recibe el segmento en la ventana esperada confirma el segmento particular
- Si recibe segmento dentro de la ventana esperada y no lo tiene lo almacena, si ya lo tiene lo descarga. Confirma siempre, necesita bufferear los rxbuf
- El receptor actualiza su ventana a medida que recibe los segmentos en orden

Resumen:permite recibir paquetes desordenados. Si uno se pierde el receptor puede seguir aceptando. El receptor almacena lo que recibe hasta que le mandan el que le falta.Solo se retransmiten los paquetes perdidos o dañados. Esto es más eficiente que el enfoque de Go-Back-N, donde se retransmiten todos los paquetes a partir del último ACK recibido.

Control de errores en tcp



1. Go-Back-N con Ventana Dinámica (Flow Control)

- TCP utiliza un protocolo de control de flujo basado en **ventanas deslizantes**, que determina cuántos bytes se pueden enviar sin esperar una confirmación (ACK) del receptor.
- La **ventana dinámica** cambia de tamaño dependiendo del estado de la red, permitiendo que más datos se envíen si el receptor puede manejarlos, y menos datos si el receptor está saturado.
- TCP utiliza **Go-Back-N**, lo que significa que si un paquete se pierde, TCP retransmite todos los paquetes desde el perdido hacia adelante, pero esto se puede mejorar mediante la negociación de **Selective Repeat**.

2. Piggy-backing

- En lugar de enviar un ACK independiente para confirmar la recepción de datos, **piggy-backing** permite que TCP combine un ACK con un segmento de datos que se envía en la dirección opuesta, ahorrando ancho de banda.

3. Selective Repeat y SACK (Selective Acknowledgment)

- TCP puede usar **Selective Repeat** para mejorar la eficiencia. En lugar de retransmitir todos los segmentos después de una pérdida, retransmite solo los que no han sido recibidos correctamente.
- **SACK** es una extensión que permite al receptor indicar qué segmentos específicos ha recibido correctamente, permitiendo al emisor retransmitir solo los segmentos que faltan.

4. TCP y Numeración de Bytes (Byte-Oriented)

- TCP no trabaja a nivel de segmentos, sino a nivel de **bytes**. Cada byte enviado a través de TCP tiene un número único, y los segmentos de datos se numeran según el **número de byte inicial** en el segmento.
- Al iniciar la conexión, los números de secuencia se **negocian** entre el cliente y el servidor mediante el **ISN (Initial Sequence Number)**, que es elegido aleatoriamente.

5. Confirmaciones Anticipativas (ACKs)

- Los ACKs en TCP son **anticipativos**, lo que significa que el receptor envía un ACK que indica cuál es el siguiente byte que espera recibir.
- Si el receptor envía un ACK con un número de secuencia mayor que el último byte recibido, significa que ha recibido todos los bytes anteriores correctamente.

6. Control de Errores en TCP: Timer y Retransmisión

- TCP establece un **temporizador (RTO: Retransmission Timeout)** para cada segmento enviado con datos. Si el temporizador expira antes de recibir el ACK correspondiente, se retransmite el segmento.
- Cada vez que un segmento se confirma (ACKed), el temporizador asociado se cancela y el segmento se elimina del búfer de retransmisión (TxBuf).
- Si los segmentos se reciben fuera de orden, el receptor descarta esos segmentos y solicita la retransmisión del que falta enviando un ACK repetido.

7. RTO Dinámico y el Cálculo de RTO

- El **RTO (Retransmission Timeout)** en TCP no es estático, sino que se ajusta dinámicamente basándose en el **RTT (Round Trip Time)**, que mide el tiempo que tardan los datos en ir y regresar del receptor.
- El cálculo de RTO se basa en el RTT estimado (SRTT) y la desviación de RTT (DevRTT) usando la fórmula: RTO=SRTT+4×DevRTTRTO = SRTT + 4 \times DevRTTRTO=SRTT+4×DevRTT

- Esto le permite adaptarse a las condiciones variables de la red. TCP ajusta el temporizador de retransmisión cada vez que recibe una nueva muestra de RTT.

8. Back-off Exponencial

- Cuando un temporizador RTO expira, TCP retransmite el segmento no confirmado más antiguo y **duplica el valor del RTO** para las siguientes retransmisiones. Esto se conoce como **back-off exponencial** y evita que TCP sobrecargue la red en caso de congestión.
- El valor de RTO máximo es recomendado en 60 segundos (según RFC-6298).

9. Confirmaciones Acumulativas y Retransmisión de Segmentos Antiguos

- TCP puede utilizar **ACK acumulativos**, lo que significa que cuando se recibe un ACK, todos los bytes anteriores al número de secuencia indicado por el ACK también se consideran confirmados.
- Si el **RTO expira**, el segmento más antiguo que no ha sido confirmado se retransmite, y se reinicia el temporizador para este segmento.

10. Timestamp en TCP (RFC-1323)

- La opción **Timestamp** en TCP se utiliza para medir el RTT de manera más precisa. Cada segmento TCP puede incluir un timestamp que permite calcular el tiempo exacto que tardan los datos en llegar al receptor y retornar.
- Además, la opción de timestamp protege contra el "wraparound" (desbordamiento) de números de secuencia cuando se alcanzan los valores máximos.

Método/Característica	Go-back-N (GBN)	Selective Repeat (SR)	TCP con SACK	Piggy-backing	Timers (TCP)
Ventana Deslizante	Sí, ventana de tamaño fijo. Retransmite todos los segmentos desde el perdido hasta el más reciente.	Sí, ventana deslizante con retransmisión selectiva de segmentos perdidos.	Sí, similar a SR, pero basado en bytes. Con **SACK** se permite confirmar solo los segmentos perdidos.	No usa ventana, pero combina ACK con datos en la dirección inversa para optimizar el ancho de banda.	Se usa un temporizador dinámico (RTO) para retransmitir si el ACK no llega dentro del tiempo estimado (RTT).
Tamaño de la Ventana	Determinado por el tamaño máximo del búfer del receptor.	Determinado por el tamaño máximo del búfer del receptor.	Ajustado dinámicamente según la congestión y capacidad de la red.	No tiene ventana, optimiza confirmaciones durante la transmisión de datos.	Solo se maneja un temporizador activo, ajustado dinámicamente para el segmento más antiguo no confirmado.
Retransmisión de Segmentos	Retransmite todos los segmentos desde el perdido, incluso si los posteriores han llegado correctamente.	Retransmite solo los segmentos perdidos, no afecta a los que fueron recibidos correctamente.	Solo retransmite los bytes específicos que faltan o tienen errores, basado en la confirmación SACK.	No aplica retransmisión, solo mejora la eficiencia combinando ACK con datos.	Si expira el **RTO**, se retransmite el segmento más viejo no confirmado.
ACKs	ACK acumulativo para confirmar todos los segmentos hasta el número indicado.	ACKs selectivos para confirmar solo los segmentos recibidos, sin afectar los perdidos.	ACK acumulativo y uso opcional de **SACK** para indicar cuáles bytes específicos fueron recibidos correctamente.	Combina la confirmación (ACK) con datos en la misma transmisión para optimizar el tráfico.	Los ACK se utilizan para ajustar o cancelar el temporizador RTO. Si todos los segmentos son confirmados, se detiene el temporizador.
Eficiencia en Redes con Errores	Baja, ya que retransmite muchos segmentos	Alta, retransmite solo los segmentos	Alta, ya que utiliza **SACK** y retrasa menos la transmisión con	Aumenta la eficiencia en redes de bajo ancho de banda	Depende del ajuste del **RTO** dinámico para retransmitir de manera

	aunque algunos ya fueron recibidos correctamente.	con errores.	confirmaciones anticipativas y selectivas.	combinando mensajes.	eficiente, ajustándose a la condición de la red.
Complejidad	Menor complejidad, más simple de implementar, pero menos eficiente.	Mayor complejidad, requiere más memoria y control para manejar las confirmaciones selectivas.	Alta complejidad, ya que incluye control por bytes, confirmaciones selectivas y opciones como **SACK**	Baja complejidad, pero su impacto es solo a nivel de optimización.	Complejo por el cálculo dinámico de **RTO** y el uso de estimaciones de RTT.
Congestión de Red	Retransmitir muchos segmentos puede empeorar la congestión.	Menos congestión porque solo retransmite lo necesario.	Optimiza el control de congestión al minimizar retransmisiones innecesarias con **SACK**	Optimiza el tráfico en la red combinando ACK y datos, reduce el uso del ancho de banda para confirmaciones.	Optimiza la congestión ajustando dinámicamente el RTO basado en las condiciones de la red.
Control de Flujo	El tamaño de la ventana se adapta al receptor, pero es menos flexible.	Mejor control de flujo con ventanas deslizantes más flexibles.	Ajusta dinámicamente el tamaño de la ventana según la congestión y la capacidad del receptor.	No afecta directamente al control de flujo, solo mejora el rendimiento en el envío de datos.	El control del temporizador afecta el flujo al retransmitir solo cuando es necesario.
Uso en TCP	No directamente. Algunos principios se usan, pero TCP es más avanzado con **Selective ACK**	No directamente, pero es el principio básico de **SACK** en TCP.	Método nativo en TCP con opciones avanzadas como **SACK** para mejorar la eficiencia en redes con errores.	Sí, TCP usa **piggy-backing** para combinar ACKs con datos.	Usado ampliamente en TCP para controlar las retransmisiones y evitar congestión con back-off exponencial.