Intercambio de datos TCP

Protocolo ARQ de parada y espera

Protocolo de **parada y espera**: el emisor no envía datos nuevos hasta confirmar que el receptor ha recibido correctamente los datos anteriores. También denominado de bit alternante. Obtiene un rendimiento muy bajo. Solución: enviar varios paquetes sin esperar a los mensajes de confirmación, procesamiento en cadena. ¿Qé necesito?

- Aumentar el tamaño de los números de secuencia, varios paquetes podrán estar en la red simultaneamente, sin confirmar.
- El emisor necesitará un buffer para almacenar los paquetes transmitidos pero no confirmados.
- El receptor necesitará un buffer para almacenar los paquetes recibidos correctamente, pero que la capa superior aún no puede procesar. Protocolos ARQ de procesamiento en cadena:
- Retroceder N (GBN - Go-Back-N)
- Repetición Selectiva (SR - Selective Repeat)

Protocolo ARQ retroceder N

El emisor puede transmitir varios paquetes sin esperar a que estén confirmados. Se establece un máximo de N paquetes enviados sin confirmación. Se suelen representar los paquetes que se pueden enviar como una ventana que se va desplazando:

- Cada vez que se recibe un ACK nuevo, se puede enviar otro paquete.
- Protocolo de **venta deslizante** El receptor no tiene buffer, Los números de secuencia son finitos, son circulares. Sólo utiliza ACKs positivos, pero son **acumulativos**.

Protocolo ARQ de repetición selectiva

Problema de retroceder N: un error en un paquete hace que se repitan otros (muchos) paquetes recibidos correctamente. Solución: que el emisor unicamente retransmita los paquetes erróneos, repetición selectiva.

- Los ACKs son individuales.
- Se utiliza una ventana, pero con algunos paquetes confirmados.
- El receptor necesita un buffer.
- Se utiliza un temporizador para cada paquete enviado.

Intercambio de datos TCP

En TCP se consideran dos tipos de tráfico de datos:

- **Interativo**: gran número de segmentos de pequeño tamaño.
- **No Interativo**: segmentos de gran tamaño, normalmente el máximo permitido por las limitaciones de la red. Para implementar la fiabilidad, se basa en el modelo ARQ retroceder N:
- Es un protocolo de ventana deslizante.
- Los ACKs son acumulativos y positivos. Con algunos matices:
- Cuando el receptor recibe un paquete fuera de orden no lo descarta, lo almacena en el buffer y envía un ACK del último paquete correcto.

- Retransmisión rápida: si el emisor recibe tres ACKs repetidos retransmitirá sólo el paquete siguiente al número de ACK.
- El emisor mantiene un temporizador por cada grupo de paquetes enviado. Al tiempo de espera antes de retransmitir se le denomina Retransmission Tiemout, **RTO**. En TCP el RTO se calcula a partir del Round-Trip Time, RTT, que se estima continuamente durante toda una conexión TCP.
- Cuando se envía un segmento se mide el tiempo que tarda en recibirse su ACK. También existe la opción de Timestamp, TSOPT: 10 bytes
- Campo Timestamp Value, TSval, el emisor indica el valor de su reloj en el momento de la transmisión.
- Campo Timestamp Echo Replay (TSecr): el receptor copia el TSval en el segmento de respuesta. Estimación de RTT con TSOPT:
- El emisor indica el TSval en los segmentos que envía.
- Al preparar la respuesta, ACK, el receptor copia el TSval en el campo TSecr.
- El emisor, al recibir el ACK, comprueba el reloj del sistema, le resta el TSecr y tiene la estimación del RTT.

Flujo de datos interactivo

Funcionamiento del ssh:

- Envío de la tecla pulsada por el cliente
- ACK de la tecla pulsada por el cliente
- Eco de la tecla desde el servidor
- ACK del eco **ACKs retardados**:
- Objetivo: enviar el ACK + eco en un único datagrama. TCP no envía el ACK inmediatamente al recibir el dato, sino que retarda la salida del ACK esperando un tiempo para ver si hay datos para enviarlos con el propio ACK. El tiempo de espera es de 200 mseg. No en valor absoluto, sino que se utiliza un reloj que da ticks cada 200 mseg. Tanto en el cliente como en el servidor se utilizan los acks retardados. El tráfico interactivo genera gran cantidad de paquetes de tamaño muy pequeño, denominados **tinygrams**
- En las redes de área local no presenta ningún problema.
- En las WAN supone una gran sobrecarga para la red. El **algortimo de Nagle** pretende resolver este problema, y se aplica en una conexión TCP de tráfico interactivo en una red de área extensa. *"Una conexión TCP puede tener un unico segmento pequeño que no haya sido confirmado. No se pueden enviar otros segmentos hasta recibir un ACK. En cambio, si esos datos se almacenan y son enviados por TCP al llegar el ACK"*. Esto es auto-ajustable: cuanto más rápido lleguen los ACKs más rápido se enviarán los datos. El algoritmo de Nagle convierte a TCP en un protocolo de parada y espera. En algunos casos puede no interesar, por lo que se puede controlar. setTcpNoDelay(boolean) de la clase Socket En este tipo de tráfico se generan "pocos" segmentos, pero de gran tamaño. El principal problema a resolver en este tipo de tráfico es el **control de flujo**: evitar que un emisor rápido sature a un receptor. TCP utiliza una ventana deslizante: permite al emisor enviar múltiples paquetes antes de parar y esperar por el ACK, lo que da una mayor rapidez a este tipo de tráfico. Con un protocolo de ventana deslizante no es necesario confirma todos los paquetes recibidos, sino que se pueden confirmar todos los paquetes simultaneamente.

Control de flujo

El envío de mensajes TCP no es determinista, depende de múltiples factores: la carga de la red, la carga del receptor y emisor, ... El emisor no tiene por qué enviar una ventana completa de datos. El receptor no tiene que esperar a que se llene la ventana para enviar un ACK.

Temp. de persistencia

En una conexión TCP, puede producirse una situación de interbloqueo cuando el emisor deja de enviar datos porque no recibe actualizaciones de la ventana de recepción, mientras que el receptor no puede notificar la liberación de espacio en su buffer. Para evitar este problema, TCP implementa un temporizador de persistencia, cuya función es asegurar que el emisor continúe verificando si el receptor ha abierto espacio en su ventana, incluso cuando no llegan notificaciones. Una vez activado, este temporizador hace que el emisor envíe segmentos especiales llamados sondas de ventana. Estas sondas consisten en segmentos muy pequeños, generalmente de un solo byte, y su propósito es comprobar si la ventana de recepción ha cambiado. Si el receptor ha liberado espacio, responderá con una confirmación (ACK) que indica el nuevo tamaño de la ventana. Si no hay espacio disponible, puede responder indicando que la ventana sigue cerrada o simplemente ignorar la sonda. El temporizador de persistencia emplea un mecanismo de retroceso exponencial, es decir, si no se recibe respuesta, el intervalo entre las sondas se incrementa progresivamente. Este proceso continúa hasta que el receptor abre la ventana y se reanuda la transmisión de datos, o bien hasta que las aplicaciones que utilizan la conexión TCP deciden cerrarla.

Control de congestión

El control de congestión de flujo evita que el emisor llegue a saturar al receptor. Esto es correcto en una LAN, sin embargo en un entorno con routers intermedios el control de flujo no es suficiente, los routers intermedios también se pueden saturar. Los routers no tienen nivel de transporte, por eso se utilizan controles de congestión. Algoritmo para evitar la congestión: Se establece el **umbral de inicio lento**.

- Por debajo del umbral, se aplica el algortimo de incio lento: "La velocidad a la que se inyectan paquetes nuevos a la red es la velocidad a la que se reciben ACKs del otro extremo"
- Por encima, se aplica un crecimiento suavizado. ¿Qué pasa si algo va mal?

1. ¿Qué es que algo vaya mal? Cuando se pierde un paquete. Dos posibilidades para "perder" un paquete: saturación en un router o un error en el paquete. Se asume que cuando se pierde un paquete es debido a que, al menos un router, está saturado.
2. ¿Cómo sé que algo va mal? Se utilizan dos indicadores para identificar un problema de congestión:

- Ha vencido un timeout de Retransmisión (Fast Recovery).
- Se han recibido ACKs duplicados (Fast Retransmit).

3. ¿Qué hago cuando algo va mal?

- Retransmitir.
- Y reducir la velocidad de transmisión. Todo esto se integra en el **algortimo para evitar la congestión**.

Temporizador de keepalive

Es una conexión TCP sin intercambio de datos, no se produce ningún intercambio de paquetes, problema en situaciones de fallos de uno de los extremos, normalmente el cliente. Solución: **temporizador de keepalive**, mantiene la conexión activa aunque no es parte del RFC de TCP. Permite liberar recursos al otro extremo, si realmente está desconectado. Funcionamiento: después de 2 horas de inactividad, el servidor enviará una

sonda keepalive (segmento de un byte, correspondiente al último byte enviado). El otro extremo puede estar en cuatro estados: ok, caído, reiniciando o no alcanzable. Se controla con /proc/sys/net/ipv4/tcp_keepalive_time, tcp_keepalive_intvl, tcp_keepalive_probes.