Proyecto Redes de Computadoras

Itzel Morales García
Rodrigo Galeana Vidaurri
Ciencias de la Computación: Redes de Computadoras

Semestre 2025-1

Entendimiento Teórico

Funcionamiento del protocolo rdt3.0

A continuación, se describen los pasos para el funcionamiento del protocolo rdt3.0:

1. **Inicio del envío (emisor)**: El emisor, con los datos que tiene para enviar, crea un paquete con un número de secuencia (0/1), lo envía al receptor e inicia un temporizador para detectar posibles pérdidas.

2. **Recepción del paquete (receptor)**: Si el paquete pasa correctamente el *checksum* y contiene el número de secuencia esperado, el receptor envía un ACK correspondiente y entrega los datos a la capa superior. De lo contrario, descarta el paquete y reenvía el último ACK.

3. **Recepción del ACK (emisor)**: Si el ACK recibido corresponde al número de secuencia enviado, el emisor detiene el temporizador y prepara el siguiente paquete (con el número de secuencia invertido). Si no, retransmite el paquete anterior.

4. **Manejo de retransmisiones**: El protocolo rdt3.0 maneja pérdidas y corrupción de paquetes mediante retransmisiones, asegurando que todos los paquetes lleguen correctamente.

Diagrama de Máquina de Estados

A continuación, se presenta el diagrama del protocolo rdt3.0:

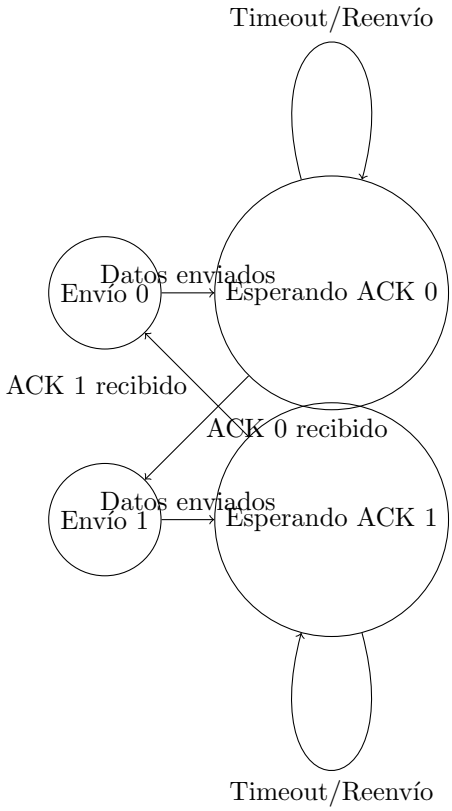

Tabla de funcionamiento

Mecanismo	**Problema que resuelve**	***¿Cómo lo implementa rdt3.0?***
Detección de errores	Detecta corrupción de paquetes	Utiliza un checksum para verificar la integridad del paquete.
Respuesta de reconocimiento	Confirma la recepción correcta de datos	El receptor envía ACK si el paquete es correcto; envía el último ACK válido si el paquete está corrupto.
Retransmisión de segmentos	Maneja pérdida de paquetes	El emisor retransmite un paquete si no recibe un ACK antes de que expire el temporizador.
Números de secuencia	Evita duplicados y garantiza el orden	Usa números de secuencia alternantes (0 y 1) para distinguir paquetes nuevos de retransmisiones.
Temporizador	Detecta pérdida de ACKs o paquetes	Activa retransmisión si el ACK correspondiente no llega dentro del tiempo límite.

Preguntas

3

En el protocolo `rdt3.0`, los paquetes `ACK` no necesitan números de secuencia propios porque:

1. **Ya incluyen el número de secuencia del paquete reconocido**: Esto es suficiente para que el emisor sepa qué paquete fue recibido correctamente.
2. **No hay ambigüedad**: El emisor solo necesita confirmar que su paquete llegó. Si no recibe un `ACK`, simplemente retransmite el paquete.
3. **Es más simple y eficiente**: No agregar números de secuencia a los `ACK` evita redundancias y reduce la cantidad de datos transmitidos.

4

Al analizar el problema podemos llegar a tres conclusiones:

1. **Rdt3.0 asegura entrega confiable:** El protocolo está diseñado para que cada paquete sea reconocido antes de enviar el siguiente. Aunque ocurran retransmisiones, el emisor siempre sabe cuál es el siguiente paquete a enviar basándose en los números de secuencia y los ACK recibidos correctamente.
2. **Retransmisión innecesaria afecta solo al paquete actual:** En nuestro problema se limita a p_2, ya que $ACK'1$ corrupto provoca la retransmisión de p_2. Sin embargo, cuando ACK_2 válido llega al emisor, este pasa al siguiente paquete p_3 de manera normal. Así, p_n no se ve afectado por estos problemas anteriores.
3. **Paquete p_n sigue el flujo estándar:** Para que p_n sea enviado, debe recibirse correctamente el ACK correspondiente a p_{n-1}. Aunque existan retransmisiones antes de llegar a p_n, este se envía una sola vez (salvo que su propio ACK se pierda o llegue corrupto).

Esto no afecta el funcionamiento correcto del Rdt3.0 pero si va a afectar el rendimiento ya que las retransmisiones innecesarias consumen recursos de red, como ancho de banda y procesamiento. En escenarios reales con alta carga de errores o problemas frecuentes de temporizador, esto puede degradar el rendimiento, especialmente en redes con capacidad limitada.

5

Sí, el protocolo `rdt3.0` es susceptible a fallos si el medio de transmisión reordena los segmentos. Esto se debe a que utiliza un esquema de número de secuencia alternante (0 y 1), el cual no es suficiente para distinguir paquetes fuera de orden en caso de reordenamiento. Si un paquete antiguo se procesa después de un paquete más reciente, el receptor puede interpretarlo incorrectamente como un paquete válido, entregando datos duplicados o desordenados a la capa superior.

Descripción del error con números

1. El emisor envía un paquete p_0 con número de secuencia 0.
2. El receptor recibe p_0, verifica su integridad y envía un `ACK 0` para confirmar su recepción.
3. El emisor envía p_1 con número de secuencia 1.
4. Por reordenamiento en el medio de transmisión, p_0 llega nuevamente al receptor **después** de p_1.
5. El receptor interpreta el segundo p_0 como un nuevo paquete, ya que su número de secuencia es válido (0), y lo procesa como si fuera el siguiente en la secuencia.
6. Esto causa que el receptor entregue datos duplicados a la capa superior, rompiendo la integridad de la comunicación.

Y entonces vemos que el protocolo `rdt3.0` no está diseñado para manejar reordenamiento de paquetes en el medio de transmisión. Si esto ocurre, puede causar errores como duplicación de datos o entrega fuera de orden