**CC4303-1 Redes**
**Profesor:** José M. Piquér
**Auxiliar:** David Miranda
**Estudiante:** Andrés Calderón Guardia

# Control 2

Redes

## P1. Protocolos clásicos

### P1.1. Algoritmos clásicos

1. Dado el alto delay y la baja pérdida, el protocolo Go-Back-N sería más adecuado como menciona el enunciado, puesto que este protocolo permite seguir enviando múltiples paquetes sin esperar la confirmación de cada uno, lo que es ideal en un escenario donde el tiempo de ida y vuelta es largo.

2. **Tamaño de ventana**: Debe ajustarse al BDP (Bandwidth-Delay Product) para maximizar el uso del enlace. Dado que el ancho de banda es de 1 GB/s y el RTT es de 5 segundos, el BDP sería:

   $$\text{BDP} = 1\ \text{GB/s} \cdot 5s = 5\ \text{GB}$$

   El tamaño de ventana debería ser 5GB, lo que asegura que se use toda la capacidad del enlace sin esperar a que lleguen los ACKs, además de no generar sobrecosto al utilizar una ventana muy grande.

   **Timeout**: El timeout debe ser ligeramente mayor que el RTT, por lo que elegimos un valor de 5.5 segundos, permitiendo margen para retrasos adicionales o variaciones en la red.

   **Número de paquetes/minuto**: Si se tiene un tamaño de paquete P entonces la cantidad de paquetes emitidos por minuto será:

   $$n^\circ\ \text{paquetes/s} = \frac{\text{BDP}}{\text{Tamaño del paquete} \cdot \text{RTT}} = \frac{5\ \text{GB}}{P\ \text{KB} \cdot 5s} = \frac{1.000.000}{P} \cdot \frac{\text{paquetes}}{s}$$

3. El principal riesgo sería la retransmisión innecesaria en caso de pérdida de un solo paquete, ya que Go-Back-N obliga a retransmitir todos los paquetes desde el que se perdió, generando ineficiencia si ocurre aunque sea una pérdida, considerando adicionalmente el gran tamaño de ventana elegido.

4. Para estas especificaciones elegiría Selective Repeat, dado que aunque Stop & Wait podría funcionar considerando el delay, la elección es más eficiente por el simple hecho de la diferencia entre el envío secuencial de Stop & Wait en contraste con la paralelización de Selective Repeat, considerando que un RTT de 10ms no es tan instantáneo, además que mantiene el flujo de datos activo incluso cuando hay pérdidas, ya que solo retransmite los paquetes que fallaron, sin necesidad de detener toda la transmisión.

   Esto último es particularmente útil si la red transfiere una alta cantidad de paquetes o lo hace de forma continua, ya que los errores serán más recurrentes dado el 1% de pérdida.

   Los parámetros a elegir considerando la elección son:

   **Tamaño de ventana**: Calculamos el BDP:

   $$\text{BDP} = 100\text{MB/s} \cdot 0.01s = 1\text{MB}$$

   El tamaño de ventana debería ser al menos de 1 MB para garantizar que se utilice toda la capacidad del enlace.

   **Timeout**: Un timeout adecuado sería ligeramente superior al RTT, por ejemplo, 12ms para asegurar una respuesta oportuna sin esperar demasiado tiempo ante fluctuaciones pequeñas en el RTT.

   **Número de paquetes/minuto**: Ídem del punto 1. considerando de nuevo un tamaño de paquete P.

5. La elección dependería del tipo de aplicación que se planea ejecutar en la red:
   - Elegiría el enlace de 1 GB/s con alto delay (5 segundos de RTT) si las aplicaciones que se van a ejecutar involucran transferencias masivas de datos o descargas de archivos grandes, donde la latencia no es tan crítica, pero el ancho de banda es esencial para la eficiencia.
   - Elegiría el enlace de 100 MB/s con bajo delay (10ms de RTT) si las aplicaciones son interactivas, como videoconferencias, servicios de streaming o cualquier tipo de aplicación donde la baja latencia sea más importante que la capacidad de transferencia masiva.

## P1.2. Números de secuencia

Para evitar confusiones entre paquetes antiguos y nuevos, el protocolo utiliza un mecanismo en el que la ventana de recepción es más pequeña que el rango total de números de secuencia, en particular, debe ser menor a la mitad de esta cantidad.

Esto asegura que, en cualquier momento, el receptor pueda distinguir entre un paquete "viejo" (previo a la ventana actual), uno "actual" (dentro de la ventana) y uno "nuevo" (posterior a la ventana actual), ya que los números de secuencia se reutilizan pero no solapan con la ventana anterior:

**Estrategia del receptor**:

- Si un paquete cae dentro de la ventana de recepción (es decir, el número de secuencia está dentro del rango actual): El paquete es aceptado y el receptor envía un ACK para confirmar su recepción.
- Si el número de secuencia del paquete es anterior a la ventana (pertenece a un paquete ya recibido y procesado en una ronda anterior): El paquete es descartado, pero se envía un ACK para notificar que ya se ha recibido.
- Si el número de secuencia del paquete es posterior a la ventana de recepción actual (un paquete demasiado adelantado): El paquete es descartado sin enviar un ACK, ya que aún no está en el rango de secuencias esperadas.

# P2. Simulador

## P2.1. BDP y Pérdida

1. Tras probar con una probabilidad de pérdida del 20%, por alrededor de media hora, no se evidenció ningún bloqueo durante todo este periodo.
2. No necesariamente. Si bien una ventana más grande permite retransmitir más paquetes y manejar mayor cantidad de paquetes en vuelo, cuando las pérdidas son significativas, podría haber bloqueos. En casos de pérdidas altas, ni siquiera ventanas muy grandes pueden prevenir bloqueos si el RTT se ve afectado o los ACKs no llegan.
3. No, cualquier protocolo eventualmente fallará, ya que no podrá gestionar retransmisiones y nuevas transmisiones de manera eficiente, puesto que se acumularían hasta un punto en el que el ancho de banda no sería capaz de manejar todos los paquetes que siguen viajando en la red.
4. No completamente. Si bien las ventanas grandes ayudan a mejorar el rendimiento hasta cierto punto (aproximando al BDP), existe un límite después del cual el tamaño de la ventana no puede compensar los bloqueos causados por pérdidas, en especial cuando estas son muy grandes en donde es muy difícil obtener ventanas de un tamaño razonable para evitar que se bloquee o congestione.
5. Las ventanas grandes en la vida real podrían causar desperdicio de memoria y congestionamiento en las redes dada la constante retransmisión de múltiples paquetes los cuales no reciben sus respectivos ACK, en especial cuando se tienen varios usuarios conectados a la vez, afectando negativamente la estabilidad de la red al saturar la misma.
6. El número de secuencias en los protocolos debe ser lo suficientemente grande para manejar todos los paquetes en vuelo. Si la ventana es enorme pero el número de secuencia es limitado, podría generarse una reutilización de números de secuencia, lo que puede finalmente causaría confusión entre paquetes "viejos" y "nuevos".

## P2.2. Consumo de Ancho de Banda

Para realizar la experimentación se dejó ejecutando el simulador con las especificaciones dadas, considerando además un delay de 5s y timeout de 10.5s, hasta llegar a los 1000 paquetes enviados, obteniendo los siguientes resultados:

1. Respecto al Consumo Útil se obtuvo 0.9706855319674009, según la teoría el óptimo correspondería a la cantidad de paquetes por segundo que registra el receptor, en este caso, pese a la pérdida dada, considerando que en la primera pregunta del punto anterior se concluyó que con las mismas condiciones pero un 20% la ventana no se bloquea, entonces con una menor probabilidad como lo es en este caso tampoco ocurriría, de forma que los paquetes aunque estén siendo retransmitidos convergen a que todos logran ser recibidos, por lo que el valor esperado sería de $0.98\overline{3}$ paq/s (59 paq/min), siendo un valor muy cercano al obtenido.
2. Y para el Consumo Total se llegó a 1.171463425905849, considerando que para cada paquete enviado hay una probabilidad de pérdida de 0.1 para el paquete y 0.1 para el ACK correspondiente, la probabilidad combinada de una pérdida (ya sea del paquete o del ACK) es:

   $$1 - (1 - 0.1) \cdot (1 - 0.1) = 0.19$$

   Entonces, el Consumo Total teórico sería aproximadamente:

   $$(59 + 59 \cdot 0.19) \cdot \text{paq/min} = (59 + 11.21)\ \text{paq/min} = 70.21\ \text{paq/min} = 1.1701\overline{6}\ \text{paq/s}$$

   Lo cual es posible evidenciar nuevamente que en este caso el valor empírico se acercó bastante al teórico.

# P3. Congestión

1. Si es cierto que estas pérdidas provocan un aumento en el BDP, dado que la congestión es señal de que la latencia está subiendo, y considerando que el BDP depende proporcionalmente del ancho de banda y el RTT, significa que este valor está incrementando. Ahora bien, la reducción de la ventana de congestión responde a un intento de reducir la cantidad de tráfico en la red, ya que cuando ocurre congestión, pese a aumentar el BDP no significa que se cuente con mayor capacidad para enviar paquetes, sino más bien, que el ancho de banda efectivo disminuyó, porque los paquetes siguen viajando por la red y aumentan los tiempos de retransmisión. Entonces, reducir la ventana de congestión ayuda a adaptarse a este cambio en el comportamiento de la red y por ello es que si conviene cambiar su tamaño.

2. Efectivamente el bloqueo si ocurre, ya que al achicar la ventana de congestión, esta se quedará esperando a que terminen de viajar los ACKs más antiguos. Sin embargo, el bloqueo de la ventana en sí no es lo que se busca, sino que lo que realmente se espera es aliviar la congestión en la red. En la práctica esto no supone un problema, sino que resulta ser parte del proceso esperado de adaptación de la ventana de congestión.

3. Sí, achicar la ventana de congestión modifica el BDP, ya que producto de ello se disminuye la congestión, reduciendo el delay y por ende el RTT, de forma que el BDP disminuye. Pero el objetivo de achicar esta ventana no es simplemente reducir el BDP, sino más bien, la consecuencia de ello, que corresponde a mejorar la congestión, dado que ahora hay menos paquetes pendientes y ACKs innecesarios en la red.

4. Esto no es directamente equivalente a mantener el BDP de la conexión, ya que después de cada envío de paquete, lo que este `sleep()` estaría haciendo es controlar artificialmente la tasa de envío de datos sin cambiar el tamaño de la ventana, por lo que más bien podría considerarse una consecuencia de ello. Por esto es que igualmente podría resultar en un efecto similar a reducir la congestión, ya que se está limitando la velocidad de transmisión y evitando saturar la red. Sin embargo, es esencialmente distinto a mantener el BDP, ya que este refleja la capacidad total de la red, y en este caso, se está modificando el comportamiento del emisor para ajustarse a la congestión, no modifica el ancho de banda ni el delay de ninguna forma.

5. La propuesta de implementar un `sleep()` adaptativo tras cada envío de paquete, manteniendo una ventana de tamaño fijo, resulta desfavorable en comparación al algoritmo de ventana de congestión, ya que este se adapta dinámicamente a las condiciones de la red, ajustando su ventana en función de la congestión, lo que le permite utilizar de manera eficiente el ancho de banda disponible.

   Además, la ventana de congestión es capaz de detectar y recuperarse de la congestión a través de la pérdida de paquetes y ajusta su ventana de transmisión en consecuencia, mientras que el `sleep()` adaptativo simplemente reduce la frecuencia de transmisión sin gestionar el tamaño de la ventana, lo que resulta en una recuperación menos efectiva.