Control 2
Redes
Plazo de entrega: 16 de octubre 2024

José M. Piquer

P1: Protocolos Clásicos

1.1 Algoritmos clásicos

En general, se considera que Stop-and-Wait es mejor con delay muy bajo y poca pérdida, Go-Back-N es mejor con delay alto y poca pérdida y Selective Repeat es mejor con delay alto y pérdida alta.

Esto, porque no vale la pena pagar la complejidad de un algoritmo avanzado si el más simple funciona bien.

Ahora, supongamos que un físico loco inventó un enlace de gran capacidad (1 Gigabyte/s), casi sin pérdidas, pero de alto delay (5 segundos de RTT).

Responda las siguientes preguntas (justificando sus respuestas, puede usar el simulador del curso para experimentar):

1. ¿Qué protocolo recomendaría usar?
2. ¿Qué configuración del protocolo usaría? (parámetros que correspondan: timeout, tamaño ventana, etc).
3. ¿Qué riesgos de ineficiencias tiene su elección?
4. Suponga un enlace más normal, con una pérdida de tipo 1 %, RTT de 10ms y ancho de banda 100 Megabytes/s: ¿qué protocolo recomendaría? ¿Con qué parámetros?
5. Si tuviera que elegir cuál de estos dos enlaces comprar (al mismo precio), ¿cómo decidiría cuál tomar?

1.2 Números de secuencia

Selective Repeat tiene una ventana de recepción: si el paquete recibido cae dentro de la ventana, se acepta. Si no, hay dos posibilidades: si es un paquete anterior a la ventana debo descartarlo y enviar un ACK; si es un paquete posterior a la ventana, debo descartarlo sin enviar ACK.

El problema es que usamos siempre números de secuencia que se reutilizan, no son infinitos. Entonces, ¿Cómo hace el receptor del protocolo, cuando recibe paquetes fuera de la ventana, para no confundirse entre paquetes del pasado de paquetes del futuro si comparten el espacio de números de secuencia?

P2: Simulador

2.1 BDP y Pérdida

Un problema con las ventanas de los protocolos clásicos es definir qué tamaño óptimo deben tener cuando conozco el BDP y la probabilidad de pérdida. Claramente es bueno que sean mayores al BDP, pero ¿qué tanto más grande?

Un ingeniero postula que la ventana tiene que ser lo más grande posible, es decir, todo lo que nos permita el protocolo y la memoria disponible.

Usemos el simulador para testear esta hipótesis, usando Selective Repeat con CACK.

El objetivo a lograr en el simulador es aproximarse al Useful Bandwidth obtenido sin pérdidas. Entonces, en una configuración con 5.000 de delay, 10.500 de timeout y 59 paquetes/minuto, una ventana de 11 paquetes cubre el BDP y (sin pérdidas) me da casi 1 de Useful Bandwidth. Al haber pérdidas, aumentará el consumo de ancho de banda. El Useful Bandwidth sólo disminuye si hay bloqueos: momentos en que el protocolo no puede seguir enviando paquetes nuevos a 1 paquete/s.

Usen el simulador: `http://users.dcc.uchile.cl/jpiquer/srgbn2.html` para poder definir ventanas grandes y no usar ventanas de congestión.

OJO: El simulador corre en javascript en el navegador, y eso hace que si la pestaña del simulador no está visible, el navegador la inactiva y deja de ejecutar hasta que es visible otra vez. Por lo tanto, para hacer pruebas un poco más largas, deben dejar la pestaña en primer plano y visible en la pantalla para medir bien los anchos de banda.

Responda las siguientes preguntas:

1. Una ventana de tamaño 40, ¿qué probabilidad de pérdida logra soportar sin bloquearse nunca?

2. Con pérdidas más grandes, ¿lograremos siempre encontrar una ventana que las soporte sin bloquearse nunca?

3. Si pudiéramos tener ventanas de tamaño infinito, ¿soportaríamos cualquier pérdida?

4. ¿El ingeniero, en teoría, tiene razón en su hipótesis?

5. ¿Qué problema podríamos generar en un ambiente real si usamos ventanas enormes?

6. ¿Qué pasaría con los números de secuencia en el caso de ventanas enormes?

2.2 Consumo de Ancho de Banda

En el simulador del curso, se muestran tres valores para el ancho de banda: el Consumo Total (gasto de ancho de banda acumulado), el Consumo Actual (consumo total en un delta de tiempo) y el Consumo útil (transmisión real lograda, sería como lo que mediría la aplicación).

Al haber pérdidas y retransmisiones, es normal que el consumo total suba, mientras que el útil baja. Aunque la generación de paquetes sea fija (típicamente 1 paq/s en nuestros ejemplos), el consumo total puede ser mayor que 1, ya que las retransmisiones se hacen por sobre la generación de paquetes nuevos. Ojo que la probabilidad de pérdida se aplica tanto a los paquetes como a los ACKs respectivos.

Experimentando con el simulador, en Selective Repeat con CACK, pruebe con 59 paquetes/minuto y 0.1 probabilidad de pérdida. Pensando en cómo funcionan los protocolos de retransmisión, responda las siguientes preguntas:

1. Mire cuál es el mejor valor de Consumo Útil que logra. ¿Cuál sería el valor teórico óptimo de Consumo útil?

2. Mire cuánto gasta de ancho de banda total. ¿Cuál sería el valor teorico óptimo que uno esperaría obtener de consumo de ancho de banda total?

P3: Congestión

El algoritmo de la ventana de congestión logra efectivamente controlar el ancho de banda utilizado. Pero, la ventana en realidad fue inventada para otra cosa: para adaptarse al BDP del enlace.

Responda las siguientes preguntas:

1. Si hay pérdidas por congestión, ¿es como si cambiara el BDP de la conexión y por eso es bueno cambiar la ventana?

2. Al achicar la ventana de congestión: ¿se queda bloqueada la conexión esperando ACKs? ¿Es esto lo que se buscaba al achicar la ventana?

3. Al achicar la ventana de congestión: ¿estamos modificando el BDP de la conexión?

4. Proponen una solución alternativa: Podríamos controlar el ancho de banda utilizado (cuando comienzan las pérdidas) agregando un `sleep()` pequeño y adaptable según la pérdida, después de cada envío de paquete, manteniendo una ventana de tamaño constante. Esto ¿es como si mantuviera el BDP de la conexión? ¿Lograría el objetivo de limitar la congestión?

5. Compare esta solución con la ventana de congestión: ¿sería mejor o peor?