Polibits

ISSN: 1870-9044

polibits@gelbukh.com

Instituto Politécnico Nacional

México

Olguín Carbajal, Mauricio; Rivera Zárate, Israel; Pérez Romero, Patricia

Protocolos de Control de Flujo

Polibits, núm. 34, 2006, pp. 7-11

Instituto Politécnico Nacional

Distrito Federal, México

Protocolos de Control de Flujo

M. en C. Mauricio Olguín Carbajal,
M. en C. Israel Rivera Zárate,
Ing. Patricia Pérez Romero,
Profesores del CIDETEC-IPN.

A lo largo de la historia de la informática y la computación, de una u otra forma, siempre se ha definido a una computadora como una maquina que procesa datos; sin embargo, este procesamiento requiere de una amplia variedad de aditamentos de soporte.

Toda esta majestuosa orquesta de equilibrios hoy en día está al alcance de una tecla o un clic, manejada por una sencilla idea que dispara un impulso nervioso que viaja hasta la mano, en donde se convierte en energía mecánica y activa un interruptor, enviando a su vez una leve señal eléctrica a un controlador de entrada, que avisa al procesador que «algo» pasó que requiere de su atención. El procesador interpreta esta señal por su origen y decide qué hacer con ella, a dónde enviarla; en esto lo ayuda el sistema operativo, que le dice qué es y a dónde va. En el caso de una tecla, este impulso eléctrico se convierte a su representación codificada y se envia a la memoria de video, en donde el controlador respectivo se encarga de presentarlo en pantalla... resulta fascinante pensar que todo esto puede ocurrir en una millonésima de segundo o menos.

Esta generación de información no siempre es exitosa; para hacerla confiable existen métodos como el control de Flujo, que garantiza que la información después de ser procesada se envié y reciba de una manera íntegra por el Receptor.

Introducción

Primero que nada diremos que el Control de Flujo existe en el intercambio de Datos (Información) solamente entre 2 entidades: *Transmisor* y *Receptor.*

El Control de Flujo es una técnica para que una computadora llamada *Transmisor* (TX) no sobrecargue a otra denominada *Receptor* (RX), al enviarle más información de la que puede procesar, debido a que normalmente tienen velocidades diferentes. Tanto el *Receptor* como el *Transmisor* tienen una zona de memoria temporal llamada *Buffer*, con una cierta capacidad para almacenar la información recibida, procesarla y enviarla.

El *Receptor* reserva generalmente una zona de memoria temporal para éste efecto; el *Receptor* debe realizar una cierta cantidad de procesamiento antes de pasar los datos al software que los utilizará. Si no existieran procedimientos para el control de flujo, la memoria temporal podría llenarse y eventualmente "desbordarse" mientras se estuviera procesando información.

En estos casos, el *Receptor* envía la información de control de flujo al *Transmisor* para que este reduzca la velocidad de transmisión, logrando así el tiempo necesario para poder procesarla. Es aquí donde se deduce la necesidad de una comunicación entre el *Receptor lento* y el *Transmisor rápido,* para que este último se entere de la situación que se está dando al otro lado del enlace; además, el control de flujo se utiliza para que no se sature la red de comunicaciones. De no existir este control de flujo podría suceder que la información se perdiera y los datos no llegaran completos.

Desarrollo

A continuación se explica de manera gráfica el manejo de la información durante la transmisión y recepción, comparando diferentes protocolos, analizando su funcionamiento y control de mecanismos, y el uso para cada situación de transferencia de datos. Estos son los principales protocolos que existen para el control de flujo:

- Protocolo simplex no restringido.
- Protocolo simplex de parada y espera.
- Protocolo simplex para un canal con ruido.

- Protocolo full duplex con información de reconocimiento (piggybacking).
- Protocolo de ventana deslizante con retroceso *n*.
- Protocolo de ventana deslizante con repetición selectiva.

A los tres primeros protocolos se les conoce como simplex por que solamente se puede transmitir información en un solo sentido, ya sea de ida o de regreso, mientras que a los tres restantes se les conoce como Full duplex, ya que se puede estar recibiendo mientras se envía información.

Protocolos Simplex

Los protocolos Simplex transmiten datos en una sola dirección; el regreso es utilizado únicamente para enviar acuses de recibo del receptor (*ACKnowledgements*).

Deben considerarse diversos factores que pueden hacer que la información se pierda durante la transmisión, entre los que podemos mencionar:

- Ruido: cualquier alteración durante la transmisión de datos debida a causas tales como campos de energía eléctrica, fallas en los cables, etc.
- Demasiado tiempo de procesamiento de la información.
- Límite en el tamaño de la información, tomando en cuenta que el buffer no fuera suficientemente grande.

Protocolo Simplex No Restringido

Este es un caso ideal, en el que se supone que la comunicación es perfecta; como su nombre lo dice no existen restricciones: no hay errores, no hay ruido, no hay limite en el buffer, la información no requiere ser procesada, por lo que no es necesario comprobar que los datos hayan llegado bien ni retransmitirlos. El *receptor* está siempre disponible y preparado para recibir datos con un espacio de buffer infinito, por lo que no se requiere control de flujo; el *transmisor* está siempre preparado para transmitir, y en este caso el único evento posible es la llegada de información.

Protocolo Simplex de Parada y Espera (Stop & Wait)

Ahora supongamos que el *receptor* no siempre está disponible para recibir información, por tener ocupado su espacio de buffer, o bien porque el mensaje sea muy grande y tenga demasiadas instrucciones que atender. En este caso, lo más sencillo es que el *transmisor* espere confirmación *ACK* después de enviar cada mensaje (*Data*), de forma que sólo después de recibir la confirmación se envíe el siguiente bloque, garantizando así el no saturar al receptor. Esto se conoce como protocolo de *parada y espera*, mismo que se muestra en la **Figura 1**, donde:

WT *Wait Time* = Tiempo de Espera

EOT *End Of Transmission* =Fin de Transmisión

DATA = Datos, información.

Protocolo Simplex para un canal con ruido

Si el canal de comunicación no es perfecto las tramas (datos) pueden alterarse debido al ruido, o incluso perderse por completo. Utilizando la Comprobación de Redundancia Cíclica, *CRC* (*Check Redundance Cycle*), que es la encargada de verificar que los datos hayan llegado sin alteraciones, el *receptor* podrá detectar la llegada de una trama (datos) defectuosa, en cuyo caso pedirá al transmisor que la reenvíe.

Sin embargo, esto puede generar duplicidad en la información; para evitar esta repetición, lo más sencillo es numerar las tramas, y forzar al transmisor a no enviar un bloque hasta recibir el acuse de recibo del anterior. Incluso, bastaría con numerar las tramas como 0,1,0,1, etc., tal como se muestra en la **Figura 2**, donde:

D-0 =Data0 y D-1 =Data1

reTX=Retransmisión

ACK= (*ACKnowledgement*)=Acuse de Recibo

Nota: El Transmisor tiene un temporizador (Timer), que sirve para detectar si en un periodo de tiempo X no recibe un ACK, para entonces reenviar la trama D-0 o D-1.

Como se muestra en la **Figura 2**, el transmisor envia D-0 y el receptor responde mediante un ACK; si en el momento que se recibe D-1 el receptor pasa mas tiempo en contestar que el especificado, el transmisor asumirá que la trama no llegó y la reenvía. Cuando el recep-

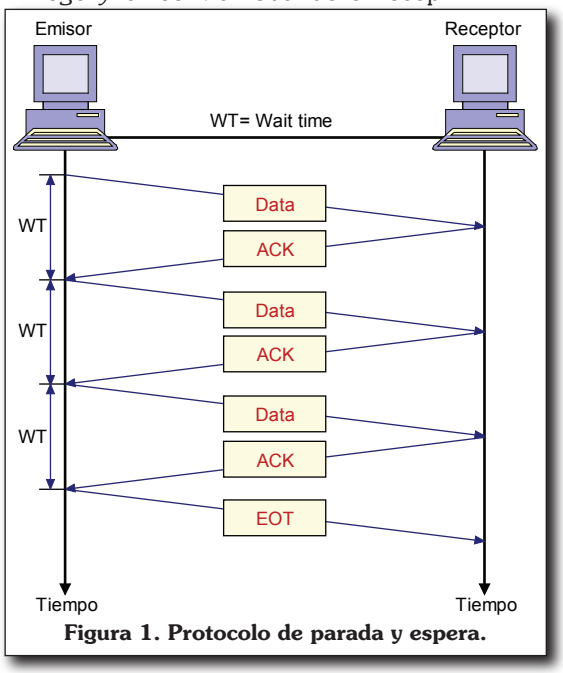

Figura 1. Protocolo de parada y espera.

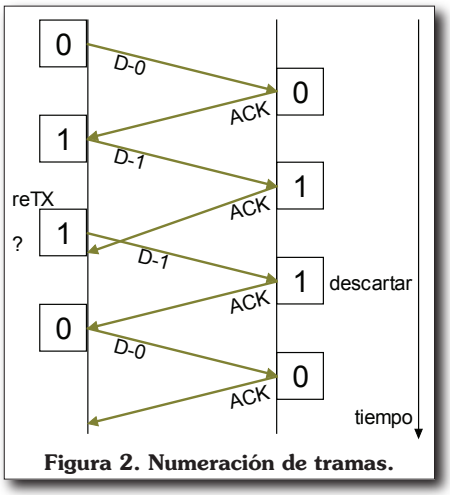

Figura 2. Numeración de tramas.

tor recibe nuevamente a D-1, se da cuenta de que ya lo tiene, por lo que simplemente lo descarta y espera el siguiente envio.

Protocolos de Ventana Deslizante

Los protocolos de ventana deslizante permiten transmitir datos en ambas direcciones utilizando canales Full-dúplex.

Protocolo Full-Duplex con Información de Reconocimiento (Piggybacking)

Recordemos que en el protocolo simplex para un canal con ruido las tramas se numeraban 0,1,0,1,…; en este protocolo el numero 0 o 1 solo servirá para confirmar la recepción de la trama. En este caso se envía un ACK de un numero, no como confirmación de trama correcta, sino para indicar que no llego la trama esperada y le solicita la retransmisión al emisor.

En este caso el ACK se monta en la trama y se ahorra un envío; esta técnica se conoce con el nombre de *piggybacking* (en inglés piggyback significa llevar algo a cuestas).

Protocolo de Retroceso n

Cuando se utiliza un protocolo de ventana deslizante de más de un número (BIT) el emisor no actúa de forma sincronizada con el receptor; por ello, cuando el receptor detecta una trama defectuosa hay varias posteriores ya en camino, que llegarán irremediablemente a él, aún cuando reporte el problema inmediatamente.

Existen dos posibles estrategias en este caso:

1. El receptor ignora las tramas recibidas a partir del error (inclusive) y solicita al emisor retransmisión de todas las tramas subsiguientes. Esta técnica se denomina *retroceso n.*

2. El receptor descarta la trama errónea y pide retransmisión, pero acepta las tramas posteriores que hayan llegado correctamente. Esto se conoce como *repetición selectiva* y corresponde a una ventana deslizante mayor de 1 en el receptor (normalmente de igual tamaño que la ventana del emisor).

En cualquiera de los dos casos el emisor deberá almacenar en su buffer todas las tramas que se encuentren dentro de la ventana, ya que en cualquier momento el receptor puede solicitar la retransmisión de alguna de ellas.

Ejemplo:

a) Caso ideal
b) Caso con retroceso n

Como se muestra en la **Figura 3a**, en un caso ideal el transmisor envía DATA 0123, y el receptor responde con acuses de recibo por cada envio.

En la **Figura 3a** existe una falla, D-2 no llega. El receptor envía un acuse negativo (NACK); sin embargo, el transmisor continua enviando tramas, hasta que recibe el NACK, entonces revisa y empieza a retransmitir a partir de la trama con error, con la desventaja de que se ocupa un ancho de banda considerable.

Protocolo con Repetición Selectiva

La repetición selectiva consiste en aprovechar aquellas tramas correctas que lleguen después de la errónea, evitandose así tráfico en la red al pedir al *emisor* que retransmita únicamente la trama dañada.

Logicamente, la técnica de repetición selectiva da lugar a protocolos más complejos que la de retroceso n, y requiere mayor espacio de buffer en el receptor; a cambio de ello ofrece mayor rendimiento, dado que permite aprovechar todas las tramas correctas.

Conclusiones

El control de flujo nos permite sincronizar el envio de información entre dos entidades, evitando así sobrecargar la red.

Problema: Emisor enviando con mayor velocidad de transmisión que la que el receptor es capaz de procesar.

Solución: Los protocolos incluyen reglas que permiten al transmisor conocer de forma implícita o explícita si puede enviar otra trama al receptor, de manera sincronizada y segura.

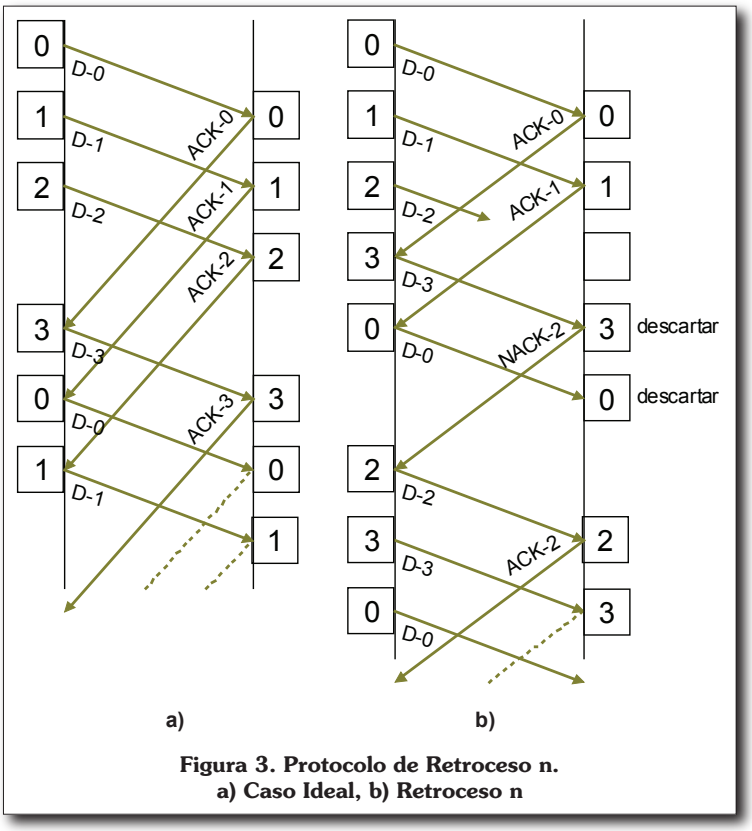

Figura 3. Protocolo de Retroceso n.
a) Caso Ideal, b) Retroceso n

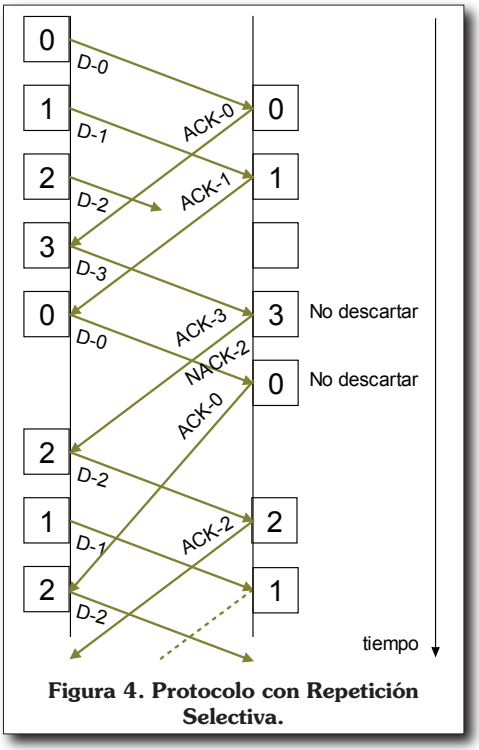

Figura 4. Protocolo con Repetición Selectiva.

Referencias

[1] Tanenbaum, Andrew S., *Redes de Computadoras*. 4ra. ed. Prentice-Hall, 1996.

[2] Forouzan, Behrouz A., *Transmisión de Datos y Redes de Comunicaciones*. 2a ed. McGraw-Hill.

[3] http://gsyc.escet.urjc.es/docencia/asignaturas/itig-transmision_datos/transpas/node5.html

[4] http://www.ilustrados.com/publicaciones/EpVAZVFApFreZSeEml.php

[5] http://deltha.uh.cu/~redes/resources/conf3.htm

[6] http://www.angelfire.com/ga/metalsystem/Redes.txt

[7] http://www.ddominguez.net/teknicas/CURSOS/redes.htm