Universidad de Costa Rica
Escuela de Ciencias de la Computación e Informática
Investigacion de operaciones
CI1453

Proyecto de Simulacion por eventos discretos: Protocolo para envio de mensajes en una red de dos computadoras

Elaborado por:
Helena Cubas
Gloriana Garro

Profesora:
Ileana Alpizar

Proyecto de Simulación

a. Manual de Instalación

En la carpeta del proyecto se incluirán los documentos con el código fuente, además en la carpeta bin/Debug hay un ejecutable que deberá ser corrido desde consola.
Al correr el programa, este le solicitará ciertos datos al usuario, estos son el número de corridas, el tiempo en segundos para correr y el tiempo del timer.

b. Descripción del problema

(copia del enunciado)
Suponga que se tiene una LAN (Local Area Network) compuesta solo por dos computadoras A y B. En esta red hay una única línea de comunicación desde A hasta B y otra desde B hacia A. Además la transferencia y la propagación es bit por bit.
Estas dos computadoras (A y B) utilizan el siguiente protocolo para el envío de mensajes desde A hacia B (es una sobresimplificación del protocolo “GO BACK N”):
La computadora A recibe los mensajes para ser enviados a la computadora B con una distribución para su tiempo entre arribos normal, con una media de 25 segundos y una varianza de 1 seg cuadrado o sea es n(μ=25,σ2=1). Todos los mensajes son de igual longitud y se van colocando en una cola, la cual para efectos prácticos es de tamaño infinito. El proceso que coloca los mensajes en cola no tiene relación con el que se encarga de hacer los envíos a B, por lo que no se va a tomar en cuenta y se asumirá entonces que el tiempo para colocar un mensaje en cola es 0.
El proceso en A que hace los envíos maneja una ventana de tamaño 8 (el N = 8). A inicia su trabajo y comienza a enviar uno por uno estos 8 mensajes. Cada vez que entra un mensaje a la ventana, este está disponible para ser enviado. Si el proceso que prepara los frames está libre, atiende al mensaje que acaba de ingresar a la ventana, si está ocupado, el mensaje se queda esperando en la ventana a que le toque su turno. Para enviar un mensaje, el proceso de A debe realizar 2 tareas:

• Primero convierte el mensaje en un "frame", esta es una estructura de datos que contiene el número de identificación del frame, el mensaje, y cierta información para que B pueda revisar si el
mensaje llega o no con errores. Para nuestro caso los números de identificación irán desde 0 en adelante, lo cual probablemente obligue a manejar valores muy altos si se corre por mucho tiempo la simulación. El tiempo que el proceso de A tarda para convertir un mensaje a un "frame" tiene distribución exponencial con una media de 2 segundos.

• Luego se coloca el “frame” en la línea bit por bit (tiempo de transferencia a la línea), como todos los “frames” son de igual longitud este tiempo es fijo de 1 segundo por “frame” (en

resumen, este proceso tarda en total 1 + X segundos, con X exponencial con una media de 2 segundos, preparando y transfiriendo a la línea un mensaje)

Note que este proceso de A se mantiene ocupado mientras prepara el frame y lo pone en la línea. Cuando termina de poner en la línea el último bit del “frame”, se encuentra “libre” para poder preparar y enviar otro “frame”, independientemente de si el “frame” anterior aún fuera o no de camino hacia B.

El "frame" que A acaba de transferir a la línea, llega a B cuando llegue el último bit. El tiempo que tarda este último bit en llegar se llama tiempo de propagación y es fijo de 1 segundo.
La computadora A envía sus "frames" en secuencia según su número de identificación, y la computadora B espera recibir los "frames" en esa misma secuencia.
Cuando se envía un “frame” a B hay una probabilidad de 0.1 de que llegue con error y probabilidad 0.05 de que se pierda.
En B hay un proceso que se encarga de recibir los "frames" que van llegando desde A, si el proceso encargado de revisar los "frames" está ocupado, entonces solo coloca en una cola el recién recibido tardando 0 segundos. La revisión de un "frame" recibido consiste en verificar si llegó en la secuencia esperada y no lleva error, entonces crea y envía a A un número de confirmación o “ACK” (acknowledge) el cual tiene como valor el número del “frame” que B espera como siguiente. Por ejemplo, si B acaba de recibir correctamente el ”frame” con número de secuencia 4, entonces le enviará a A un ACK = 5.

El proceso de B que revisa "frames" y crea y envía ACK´s tarda los siguientes tiempos con sus distribuciones:
• Tiempo de revisión del "frame" para detectar si viene en secuencia y si llegó o no con errores, así como la creación del ACK si se debe hacer, sigue la siguiente distribución de probabilidad:

f(x) = 2x/5 2 <= x <= 3 segundos

Si B recibe un “frame”, pero detecta que viene con errores (utilizando la información que agrega A a mensaje), o si viene mal según la secuencia entonces B desecha el frame y se reenvía a A, el ACK del frame que espera.

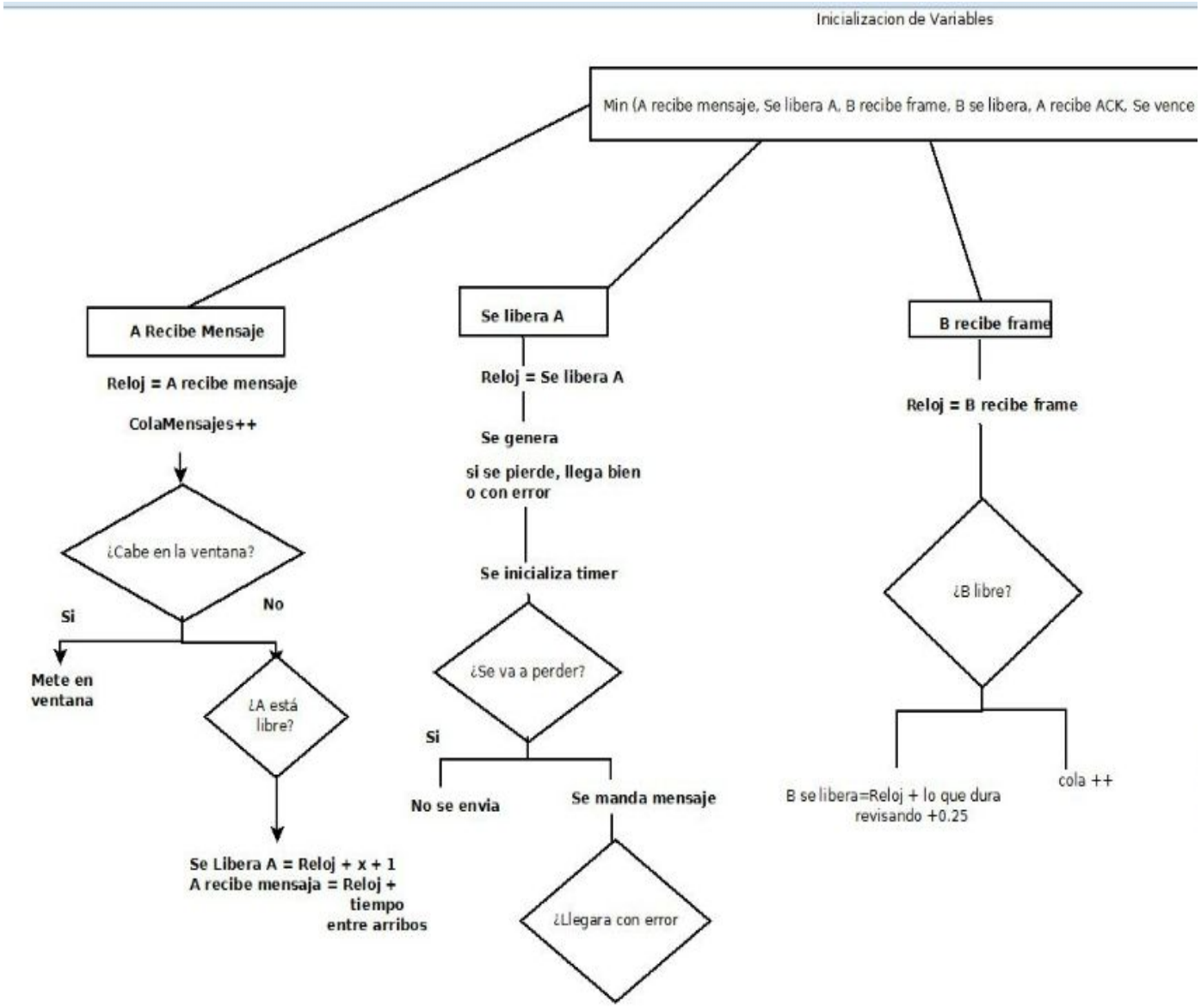
• Si se debe enviar ACK, este proceso en B debe entonces ponerlo bit por bit en la línea de transmisión. Este tiempo es fijo por ACK y es igual a 0.25 segundos
El tiempo que tarda en llegar a A el último bit del ACK (tiempo de propagación) es de 1 segundo.
Cada ACK enviado tiene una probabilidad 0,15 de que se pierda (suponga que si llega no tiene error).
Si A ha enviado un “frame” a B, pero luego de x segundos no ha recibido de vuelta la confirmación de su llegada, A reenvía a B dicho “frame” y todos los que había enviado luego de éste. (Timer de x segundos) Note que esto puede ocurrir por tres 2 razones: una, que el "frame" de camino a B se perdiera, y la otra que el ACK se perdió en su camino a A.

A puede enviar todos los "frames" de su ventana (hasta 8), uno detrás de otro, sin esperar “ACK”
(acknowledge). Si no ha recibido ningún ACK luego de enviar la ventana, A espera sin enviar más. Apenas A recibe un ACK correcto corre la ventana, descartando los "frames" que ya sabe llegaron bien a B, y envía los nuevos frames que ingresaron a la ventana (aunque no completen 8). Cuando se espera el siguiente ACK, si este no llega en el tiempo esperado (x segundos) reenvía a B todos los "frames" de la ventana. Note que A puede recibir un ACK con una valor mayor al esperado, pero esto solo significa que B recibió correctamente los "frames" pero que los ACK´s enviados se perdieron en su camino a A. Si A recibe un ACK, pero no hay mensajes que enviar, entonces el proceso se queda libre, pero apenas ingrese un mensaje a la ventana, continuará con su envío. Si la ventana está llena, solo ingresan "mensajes" de la cola cuando hay
corrimiento por un ACK recibido,

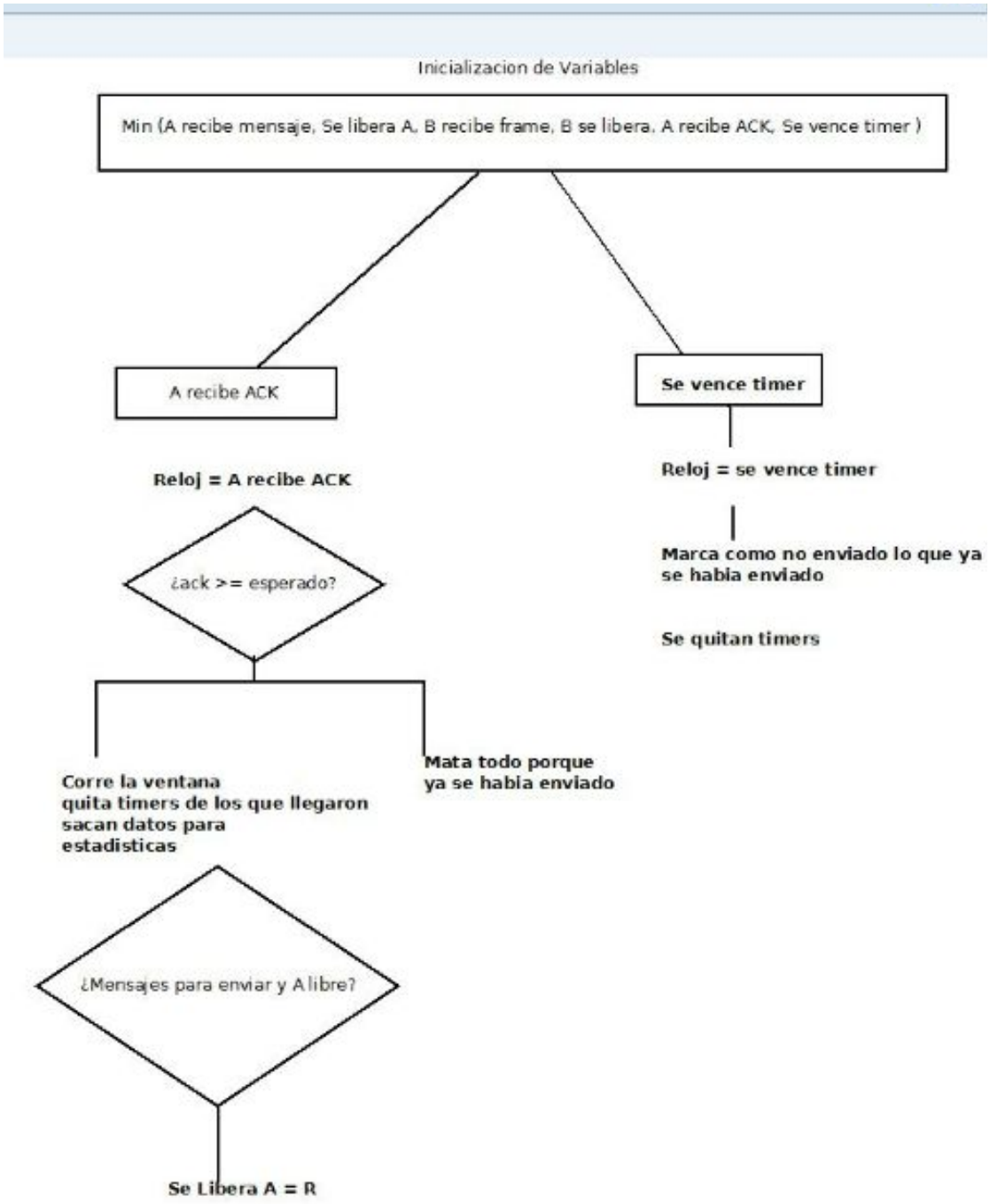
Suponga que el proceso en A que se encarga de recibir los ACK´s no es el mismo que hace los envíos a B, ni el mismo que recibe los mensajes en A. De esta manera no se pregunta si está ocupado o no, se supondrá que al igual que el que recibe los mensajes que van llegando a A, siempre puede recibir y procesar un ACK. Suponga además que el tiempo de análisis del ACK recibido y el tiempo de lo que se deba hacer a consecuencia de este análisis (correr la ventana descartando mensajes que ya B recibió etc , gasta 0 tiempo.
De igual manera si se vence el timer de un "frame" enviado, el proceso encargado de reaccionar ante este evento es otro específico para esto (no es el que recibe mensajes a A, ni el que recibe ACK´s desde B, ni el que envía frames a B) y a este es el que le toca hacer las modificaciones (corrimiento de ventana en la dirección opuesta, y etc etc, gastando tiempo 0)

c. Diagrama de flujo

Parte 1 del diagrama .



Parte 2 del diagrama



Además de esto, después de cada evento se pregunta si el reloj actual es mayor que el máximo de tiempo que se le solicitó al usuario.
El diagrama anterior pretende ilustrar la idea que se tiene para lograr la simulación. Al comenzar la simulación se inicializan las variables y se selecciona el mínimo de estas. Idealmente el primer evento que se llevará a cabo va a ser el de A recibe Mensaje, como no va a haber nada en la ventana en ese momento, programará el tiempo en el que sucederá A se

libera y en ese momento se envía el frame a B, el evento se libera B se encarga de revisar que el frame haya llegado correctamente y también programa que se envíe ACK, si no se pierde A recibe ACK, recibe el ACK esperado y si hay mensajes en cola y hay espacio en la ventana, se encarga de meterlos en esta.
Cada evento modifica el valor del reloj y modifica a la hora que se daran los siguientes eventos.

Problemas no resueltos:

El programa se cae al correr A recibe ACK, lo cual no permite que se puedan sacar las estadísticas, ni el intervalo de confianza.
Sin embargo, sí se programaron los métodos necesarios para realizar los cálculos necesarios para obtener tanto las estadísticas como el intervalo de confianza.