PROYECTO DE SIMULACIÓN POR EVENTOS DISCRETOS (VALOR 30 %)
PROTOCOLO PARA ENVÍO DE MENSAJES EN UNA RED DE DOS COMPUTADORAS

En grupos de **no más de 3 personas** se debe programar un modelo estocástico discreto de simulación del sistema explicado a continuación. El grupo debe escoger un líder de proyecto y comunicarlo a la prof. por correo electrónico a más tardar el día 24 de mayo.
Nota: el lenguaje de programación a utilizar puede ser cualquiera de su agrado, siempre y cuando no sea un lenguaje específico para simulación.
Entrega del proyecto: a más tardar **el Lunes 10 de Junio a las 11:59 pm**

I. DESCRIPCIÓN DEL SISTEMA A SIMULAR

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 t**ransferencia 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 $\mathbf{n(\mu=25,\sigma^2=1)}$. 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**
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:

$$f(x) = 2x/5 \quad 2 <= x <= 3 \text{ segundos}$$

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

- Para **enviar el 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 **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 etc) gasta **0 tiempo** (por simplicidad)

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**)

NOTAR QUE: Un timer mal estimado, puede provocar que se envíe un "frame" otra vez al vencerse ese timer, pero que sí había llegado a B. El problema fue que no se dió tiempo suficiente para que llegara su ACK a A. Cuando llegue a A ese ACK tardío, se supondrá erróneamente que es el correspondiente a este segundo envío. B debería descartar todos los que le lleguen y que ya tenía, pero para cada uno de ellos debe enviar no su ACK, sino el que realmente espera (lo cual provocaría que A reciba un mismo ACK varias veces) Esto puede provocar errores y retrasos (ver punto III)

II. EJEMPLOS PARA EXPLICAR MEJOR LO ANTERIOR:

Se tiene la cola de mensajes para envío en A, en la que la ventana se marca en color amarillo, los "frames" no enviados aún en blanco, y los "descartados" porque ya llegaron bien a **B** en gris (el orden es de izq. a derecha, es decir los "más viejos" a la izquierda):

a) Esta ventana indica que **A** envió los "frames" del 0 al 7 seguidos, sin esperar ACK por parte de **B**. Sin embargo ahora no puede hacer nada hasta que llegue un ACK o un "timer" se venza

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 ...

Ahora **A** recibió un ACK = 1 desde **B**, esto indica que recibió bien el "frame" 0 y que espera el 1 (el cual ya se había enviado), entonces se descarta el "frame" 0, la ventana se corre y se envía el mensaje que acaba de entrar a la ventana, es decir, el identificado con el número de secuencia "8":

0 **1 2 3 4 5 6 7 8** 9 10 11 12 13 14 15 16 17 18 19 20 ...

b) **A** recibió ahora un ACK = 3 desde **B**, esto indica que **B** recibió bien los "frames" 1 y 2 aunque el ACK que envió para el 1 se perdió en la línea de transmisión. Al recibir **A** este ACK corre la ventana y se envían los siguientes mensajes que recién entraron a la ventana identificados con los números de secuencia "9"y "10":

0 1 2 **3 4 5 6 7 8 9 10** 11 12 13 14 15 16 17 18 19 20 21 22 ...

Veamos ahora la situación en B a partir de este instante (marcamos la secuencia de "frames" que ha recibido **B** en celeste, el "frame" que espera en ese momento (el último ACK enviado a **A**) en color **azul** y los que seguirán en blanco:

c) **B** ha recibido los "frames" 0, 1 y 2 y espera el 3 (recordemos que para este momento **A** ha enviado los mensajes 0,1,2,3,4,5,6,7,8,9,10)

0 1 2 **3** 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 ...

d) **B** recibe en este momento el "frame" 5. Viene fuera de secuencia ya que se espera el 3, esto significa que el 3 y el 4 se perdieron en la línea de transmisión. Entonces **B** descarta este "frame" que llegó y vuelve a enviar un ACK=3:

0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3…

e) Para estos momentos en **A** se vence el "**timer**" para el "frame" 3, entonces **A** asume que se perdieron todos los "frames" enviados a partir del número 3 por lo que se devuelve en la cola 8 "frames" y los envía de nuevo ("**go back n**"en donde n vale 8) y de nuevo espera los ACK.

0 1 2 **3 4 5 6 7 8 9 10** 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 ....

Otro ejemplo: (independiente de la secuencia anterior)

a) **A ha enviado los "frames" 0, 1 y 2.** No han llegado más mensajes a A

0 1 2

b) Si llega **otro mensaje a A,** se mete a la ventana, se identifica con la secuencia 3 y se envía también a B

0 1 2 3

c) Si ahora se recibe un **ACK = 4 en A,** se sacan de la ventana los "frames" 0, 1, 2 y 3, y quedan la ventana y la cola vacías , pues no hay más mensajes que enviar

0 1 2 3

III. NOTE QUE EL PROTOCOLO POR SÍ MISMO PUEDE PROVOCAR ERRORES Y NO ES MUY EFICIENTE

Como se indicó anteriormente, problemas como los ocasionados por un valor del timer "malo" que no da tiempo a que los procesos trabajen normalmente, se darán en el "GO BACK - N". Así como el reenvío de mensajes que ya habían sido recibidos en B, pero que se perdió su ACK. Usted puede, si desea, idear mejoras a este protocolo, sin embargo no se pide que lo haga. Tan solo debe programarlo correctamente, de acuerdo con el comportamiento "típico" de "GO BACK - N".

En el curso de redes aprenderá otros protocolos que no solo evitan estos errores, sino que también son más eficientes en cuanto a tiempo y almacenamiento.

IV. ESTADÍSTICAS Y PRESENTACIÓN EN PANTALLA DE LA SIMULACIÓN

1. **Al inicio se le debe solicitar al usuario:**
   a) Número de veces que se va a correr la simulación
   b) Tiempo total en segundos para correr cada vez la simulación (Máximo de tiempo)
   c) El tiempo de espera en **A** para reenvío de un "frame"o sea el valor de **x** para el "timer".
   d) Si desea ver la simulación correr en modo lento o no (Se implementa solo con un **delay**)

2. **Mientras corre la simulación** en pantalla debe presentarse (como mínimo):
   a) **El reloj** del sistema,
   b) La **longitud de la cola** en **A**,

c) La lista de los primeros **20 "mensajes" en la cola de A** (indicando **claramente** la ventana), (no incluye los descartados)
d) El valor del **último ACK** recibido por **A**,
e) El valor del **último ACK** enviado por **B**,
f) El total de "**frames**" correctamente recibidos por **B**
g) La lista de los **últimos "frames" recibidos correctamente por B (20 máximo)**,
h) El tipo de **evento** que se está procesando.

3. **Al final de cada corrida** deben desplegarse las siguientes estadísticas:
   a) Tamaño promedio de la cola en **A (incluyendo la ventana)**.
   b) **Tiempo promedio de "permanencia de un mensaje en el sistema"** (desde que llega el mensaje a **A** hasta que su ACK es correctamente recibido por A (o sea, hasta que A lo descarta de la cola)
   c) **Promedio del tiempo de transmisión** para un mensaje **= [**el tiempo que se tardó en sus puestas en línea (transferencia) + sus tiempos de propagación**]** (no se incluyen los tiempos de transferencia ni de propagación de los ACK para ese mensaje)
   d) **Tiempo promedio de servicio** (para un mensaje)= **[**tiempo calculado para b) **menos** el tiempo calculado para c)**]**
   e) Eficiencia:
   tiempo prom. de transmisión (calculado para c) / tiempo prom de servicio (calculado para d)

4. **Al final de la serie de corridas solicitada por el usuario** deben también desplegarse las estadísticas anteriores, pero como un promedio de todas las de cada corrida y un **intervalo de confianza** para el **tiempo promedio de permanencia en el sistema** para cada "mensaje"

V. DOCUMENTACIÓN MÍNIMA ESPERADA Y ENVÍO DEL PROYECTO: (TODO ESTO DEBE SER ENVIADO POR CORREO ELECTRÓNICO) **A MÁS TARDAR EL DÍA LLUNES 10 DE JUNIO A LAS 11:59:59 PM)**

a) Un **manual de instalación** describiendo el ambiente necesario para correr el programa así como la forma de correrlo.
b) Descripción del sistema a simular (este enunciado)
c) Un **diagrama de flujo o un diagrama que utilice pseudocódigo** simple describiendo claramente la manera en la que se hará la simulación del sistema. Deben usarse nombres descriptivos, o una lista describiendo cada "nombre" del diagrama.
d) Archivos con **código fuente** y con el ejecutable (**debe presentarse el ejecutable**) debe haber una clara y completa documentación interna.
e) **Estadísticas** obtenidas: impresión de la pantalla con las estadísticas obtenidas para los valores de prueba: **correr la simulación 10 veces, correrlo por 2000 segundos cada vez, con un "timer" de 12 segundos**. (para cada una de las 10 veces y para el conjunto)
f) Un **análisis del sistema** con base en las estadísticas obtenidas (del sistema simulado, no el de su programa de simulación)
g) Descripción de los **problemas NO RESUELTOS** al momento de enviar la simulación y una idea de cómo resolver cada uno.
h) Una **nota** para todos y cada uno de los miembros del grupo de trabajo puesta por el líder de acuerdo con el desempeño del integrante en el desarrollo de esta parte del proyecto (0 a 100) Con ese valor se calculará la nota del proyecto correspondiente a cada uno de los integrantes del grupo como:

nota obtenida en el proyecto por el grupo * porcentaje asignado a ese integrante.