Examen de Introducción a las Redes de Computadoras y Comunicación de Datos (ref: sirc0703.doc) 26 de febrero de 2007

***Atención**:*

- *La duración del examen de 3 horas.*
- *El examen debe realizarse sin material.*
- *Se debe responder al menos el equivalente a 15 puntos en las preguntas teóricas.*
- *El puntaje mínimo de aprobación es de 60 puntos.*
- *para todos los ejercicios, suponga que dispone de los tipos de datos básicos (p.ej. lista, cola, archivo, string, etc.) y sus funciones asociadas (ej: tail(lista), crear(archivo), concatenar(string, string).*

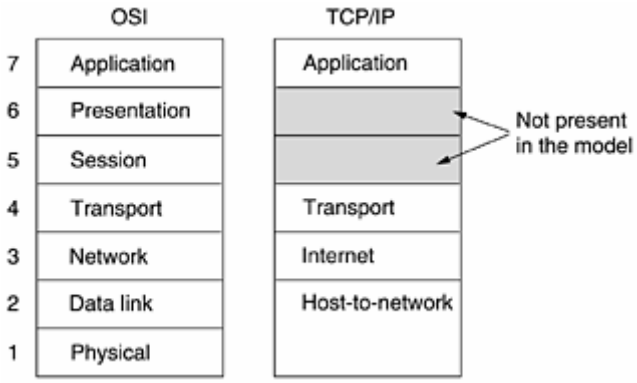
Preguntas Teóricas

Pregunta 1 (5 puntos)

Describa el modelo de capas TCP/IP.

Respuesta:

El modelo TCP/IP surgió a partir de la red ARPANET, cuyo mayor objetivo de diseño era interconectar varias redes en forma transparente, dicha arquitectura evoluciono en el modelo TCP/IP.



La capa Host a Red

Especifica cómo se envían físicamente los datos a través de la red, que incluye cómo se realiza la señalización eléctrica de los bits mediante los dispositivos de hardware que conectan directamente con un medio de red, como un cable coaxial, un cable de fibra óptica o un cable de cobre de par trenzado. No esta definido el protocolo y el mismo puede variar en diferentes implementaciones.

La capa Interred

Esta capa es la que mantiene unida toda la arquitectura. Permite a los host enviar paquetes a cualquier red y que viajen independientemente del destino (potencialmente otra red). Pueden incluso llegar en diferente orden, en cuyo caso es trabajo de capas superiores, reordenarlos. La capa de interred define un formato de paquetes llamado IP. El trabajo de la capa de interred es que dichos paquetes lleguen a donde corresponde, por lo que el ruteo de los paquetes es el mayor problema. La capa de interred es similar en su funcionamiento a la capa OSI de Red.

Capa de Transporte
Sobre la capa de interred se define la capa de transporte, la misma está diseñada para conectar dos entidades y mantener un enlace de datos. Hay dos protocolos definidos: TCP (Transmission Control Protocol), es un protocol orientado a conexión, confinable, que permite que un flujo de bytes sea enviado de una maquina a otra sin errores; fragmenta el flujo de bytes y lo pasa a la capa de interred, al llegar a destino reconstruye el flujo de bytes; TCP también maneja el control de flujo para evitar sobrepasar la capacidad de un receptor más lento. El segundo protocolo, UDP, (User Datagram Protocol) es un protocolo no confiable sin conexión, para aplicaciones que no necesitan secuenciamiento de paquetes o control de flujo, o aplicaciones donde es más importante transmitir rápido que en forma confiable, como voz o video.

Capa de aplicación
El modelo TCP/IP no tiene capas de sesión o presentación, no se pensó que fueran necesarias, por lo que no fueron incluidas. En la práctica se prueba que son muy poco usadas. Por lo que en el modelo arriba de la capa de transporte se encuentra la capa de aplicación, que contiene todos los protocolos de alto nivel, como TELNET, FTP, SMTP, DNS, NNTP, HTTP, etc.

Pregunta 2 (10 puntos)

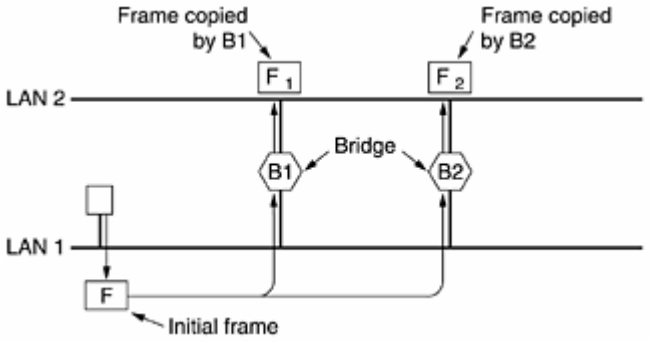
Un bridge (puente) utiliza la siguiente política:

- si la LAN origen y destino son iguales, descarta el frame
- si la LAN origen y destino son diferentes, reenvía el frame al puerto correspondiente de la LAN destino
- si la LAN destino es desconocida, utiliza flooding (envía por todos los puertos).

Describa que problema puede ocasionar este protocolo, mencione cómo se soluciona.

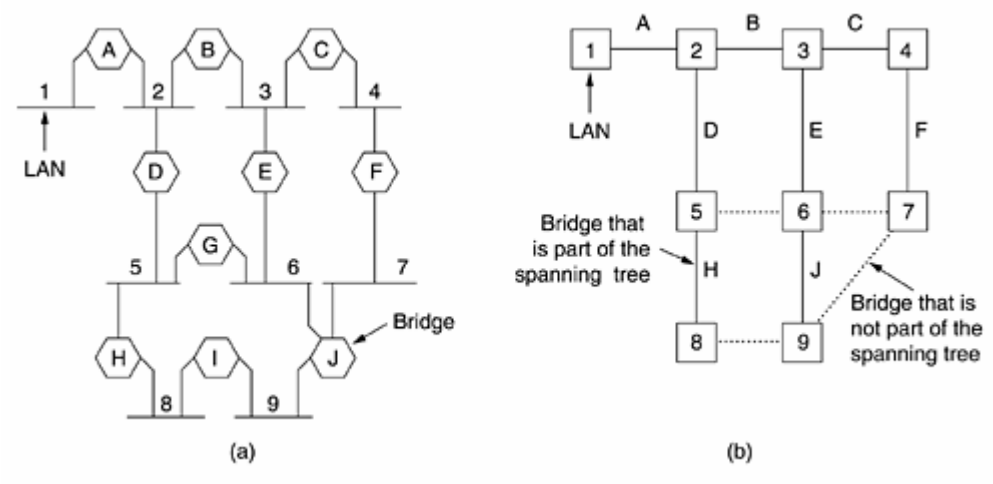
Respuesta:

Para incrementar la confiabilidad, se pueden utilizar varios puentes en paralelo entre LANs, como se muestra en la figura. Sin embargo, esto introduce algunos problemas ya que crea bucles en la topología.



Un ejemplo de este problema se puede ver en la figura. Cada puente usando las reglas descritas, usa flooding, que en ejemplo simplemente copian las tramas a la LAN 2. Luego el puente 1 ve los paquetes y los vuelve a copiar a la LAN 1. De esta forma se siguen copiando los paquetes entre las LAN en forma infinita.

La solución a esto es que los puentes se comuniquen entre si y descubran la topología con un árbol de cubrimiento o expansión, que cubra todas las LAN. Cada arista del árbol conecta dos LAN que están conectadas por un puente. El grafo puede ser reducido a un árbol de cubrimiento quitando las aristas que producen bucles, por lo que queda exactamente un camino de cada LAN a otra. Una vez que los puentes acordaron el árbol de cubrimiento, todos los redirecionamientos de tramas entre las LAN deben seguir el árbol de cubrimiento, y como solo hay un camino entre cada LAN no son posibles los bucles.



Pregunta 3 (10 puntos)

Describa en que escenario se utiliza el protocolo RARP y su funcionamiento.

Respuesta:

ARP resuelve el problema de encontrar que dirección Ethernet corresponde a una dirección IP dada.

A veces se debe resolver el problema reverso: dada una dirección Ethernet, cual es la dirección IP que le corresponde. Este problema ocurre por ejemplo en el escenario en que hay estaciones de trabajo tontas, sin disco y las mismas deben ser reiniciadas. Dicha maquina normalmente obtiene la imagen del sistema operativo de un archivo remoto, pero antes debe tener una dirección IP.

La primera solución que se encontró fue usar RARP (Reverse Address Resolution Protocol), este protocolo permitía a las maquinas reiniciadas hacer un broadcast de Ethernet enviando una consulta de IP con su dirección Ethernet. Un servidor de RARP recibe la solicitud, y realiza una búsqueda en sus archivos de configuración enviando una respuesta con la dirección IP.

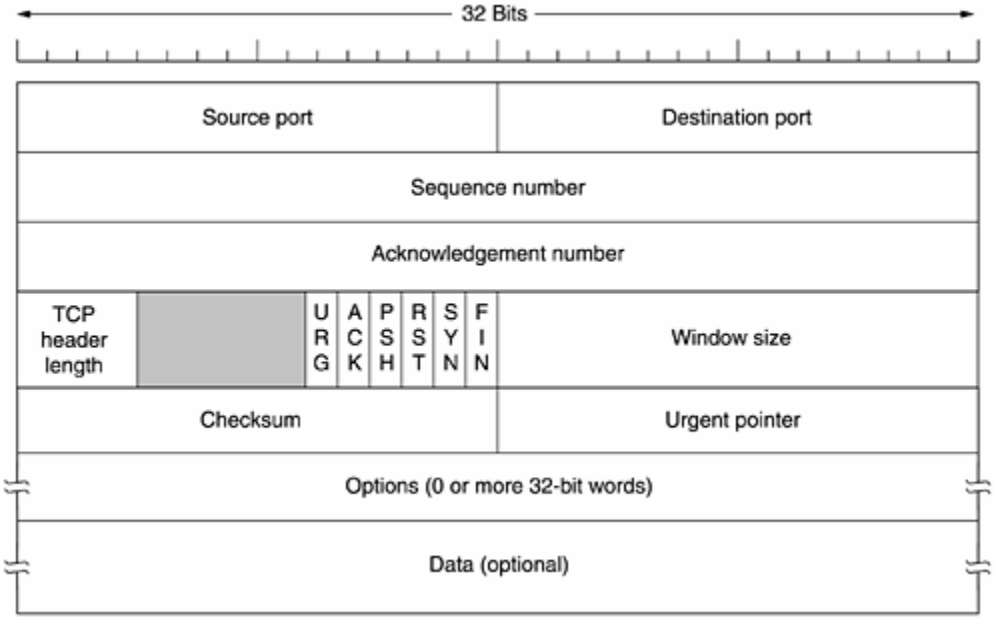
Usar RARP es mejor que grabar la IP en la imagen del sistema operativo ya que permite usar la misma imagen para todos los equipos. La desventaja de RARP es que los broadcast Ethernet no son reenviados por los router, por lo que se necesita un servidor RARP en cada subred. Por ello se diseñaron otros protocolos como BOOTP y DHCP.

Pregunta 4 (10 puntos)

Mencione y explique los campos más importantes de los segmentos TCP y UDP.

Respuesta:

TCP

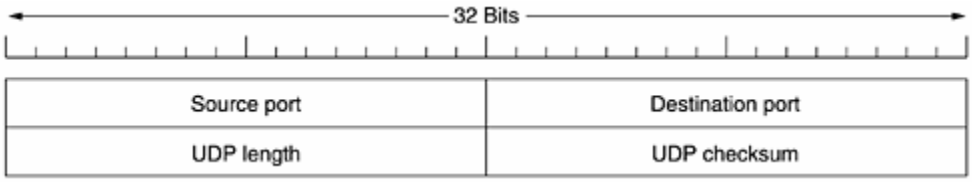


1) Puertos Origen/Destino
   Identifican los puntos de acceso de la conexión. Hay puertos conocidos para protocolos comunes y puertos definidos por el usuario. El puerto más la dirección IP forman un único punto de acceso. Los puntos de acceso origen y destino identifican únicamente a la conexión.
2) Numero de secuencia / confirmación
   El número de secuencia identifica cada paquete. El número de confirmación es el siguiente número esperado no el último recibido. Ambos son de 32 bits porque se numera cada byte de datos.
3) Largo del cabezal
   Cuenta cuantas palabras de 32 bits están contenidas en el cabezal. Esta información se usa porque el campo de opciones es de largo variable. En realidad este campo indica el comienzo de los datos medido en palabras de 32 bits, pero el número indica el largo del paquete por lo que es lo mismo.
4) URG
   El puntero urgente es usado para indicar un desplazamiento del número de secuencia a partir del cual se encuentran datos urgentes.
5) ACK
   Se utiliza para indicar que el paquete es de confirmación, si no es así, el campo de confirmación es ignorado.
6) PSH
   Indica que los datos deben ser enviados a la capa de aplicación inmediatamente luego de ser recibidos, en lugar de almacenarlos.
7) RST
   Se utiliza para reiniciar una conexión caída u por otro problema, o para rechazar un paquete o una conexión. Solo se utiliza en caso de un problema grave.
8) SYN
   Se utiliza para establecer conexiones, se envía un paquete con SYN = 1 y ACK = 0. La respuesta debe ser SYN = 1 y ACK = 1. Dicho campo es utilizado en definitiva para indicar un CONNECTION REQUEST o CONNECTION ACCEPTED, y el ACK distingue las dos.
9) FIN
   Se utiliza para cerrar una conexión, denota que no se tiene más datos para transmitir. Sin embargo luego, de cerrar una conexión el proceso puede continuar enviando datos.
10) Tamaño de ventana
    El control de flujo se realiza con una ventana deslizante. El tamaño de la ventana indica cuantos bytes pueden ser enviados a partir del último byte confirmado. Una ventana de largo 0 se permite e

indica que se el receptor no aceptara más datos. El receptor puede volver a aceptar datos enviando un paquete con un tamaño de ventana distinto de 0.

11) Checksum
Se utiliza un campo de verificación para proveer más confiabilidad. El campo de verificación incluye el cabezal, los datos y un seudo cabezal.
12) Opciones
El campo de opciones es una forma de agregar funcionalidades extra no cubiertas por el cabezal estándar. La opción más importante es la que permite especificar el tamaño de los datos que es aceptada por un nodo. Otra opción posible es la indicación de utilizar selective repeat en lugar de go back n.
13) Datos
Finalmente se incluyen los datos a ser enviados en el paquete

UDP



1) Puertos Origen/Destino
Identifican los puntos de acceso de la conexión. Hay puertos conocidos para protocolos comunes y puertos definidos por el usuario. El puerto más la dirección IP forman un único punto de acceso. Los puntos de acceso origen y destino identifican únicamente a la conexión.
2) Largo del cabezal
Cuenta cuantas palabras de 32 bits están contenidas en el cabezal y en los datos.
3) Checksum
Es un campo de validación y es opcional, si no es calculado se rellena con 0.

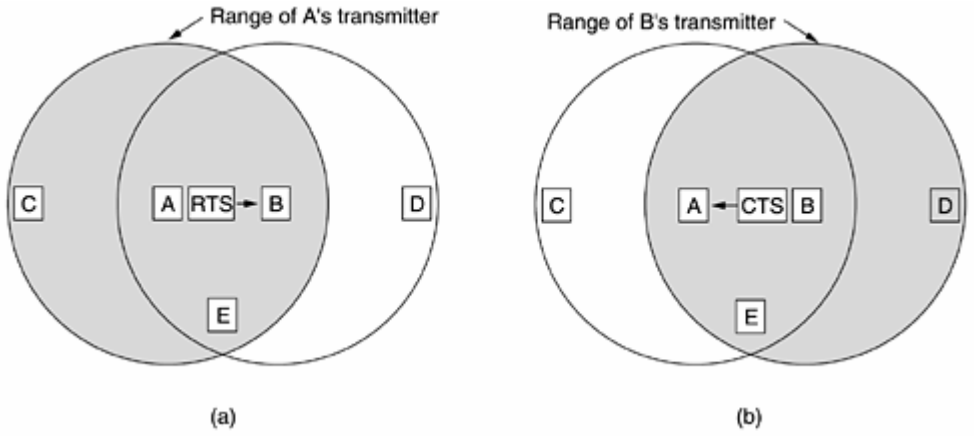
Pregunta 5 (5 puntos)

Describa el mecanismo de acceso al medio CSMA/CA (*Carrier Sense Multiple Access With Collission Avoidance*).

Respuesta:

Uno de los primeros protocolos diseñados para LAN inalámbricas fue CSMA/CA o MACA (Carrier Sense) Multiple Access with Collission Avoidance.

La idea es que el emisor estimula al receptor enviando una trama corta, de forma que las estaciones cercanas puedan detectar la transmisión y eviten transmitir durante el envío de una trama de datos.

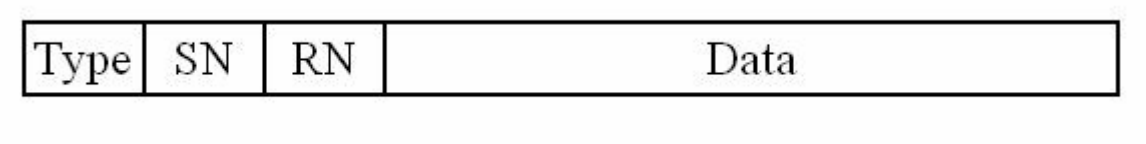


Consideremos que A quiere enviar una trama a B. Comienza enviando una trama del tipo RTS (Request To Send) a B, esta trama corta contiene el largo de la trama que sigue. Luego B responde con una trama CTS (Clear to Send). La trama CTS contiene una copia del largo de la trama siguiente enviada por A.
Luego que A recibe la trama comienza el envío de datos.
Cualquier estación escuchando un RTS está cercana a A y no debe enviar datos para que A reciba el CTS sin interferencia. Cualquier estación cercana a B no debe enviar datos para que A pueda realizar el envío de datos sin interferencia.
Se plantea el problema mostrado en el dibujo donde una estación C escucha la trama RTS pero no la CTS por lo que puede llegar a enviar datos durante el envío de datos de A a B. Por otro lado una estación D puede escuchar la trama RTS pero no la CTS, por lo que deduce que se van a enviar datos, y debe evitar mandar en ese periodo.
Por lo tanto pueden ocurrir colisiones, por ejemplo también si B y C envían RTS al mismo tiempo a A. Cuando ocurre una colisión, el emisor aguarda un periodo aleatorio de tiempo y vuelve a intentar.

Problemas Prácticos

Problema 1 (30 puntos)

Se cuenta con un protocolo de capa 2 (enlace) que opera sobre un enlace punto a punto, el que utiliza el siguiente formato de trama:



Los campos que integran la misma son:

- Type (1 octeto) especifica el tipo de la trama.
- SN (1 octeto) especifica el número de secuencia de la trama.
- RN (1 octeto) especifica el número de trama que espera recibir el equipo origen.
- Data (0..MAX_LARGO) contiene los datos (payload) que transmite la trama.

Se desea contar con un protocolo para manejo del enlace que sea orientado a conexión, con un esquema de ARQ (Automatic Repeat reQuest) de tipo Selective Repeat, que implemente además piggybacking.

Se cuenta con las siguientes primitivas:

- *envioTrama(trama:byte[]).* Envía la trama pasada por parámetro por el enlace agregando un campo de verificación de la misma.
- *reciboTrama():byte[].*Devuelve una trama recibida por el enlace tal que verificó el campo de verificación de la misma.
- *datoAenviar():byte[].*Devuelve datos generados en la capa superior. a enviar por el enlace.
- *datoRecibido(dato:byte[]).*Envía los datos recibidos por el enlace, a la capa superior.

Se pide:

a) Especificar que tipos de tramas requiere para implementar el protocolo (el campo Type de la trama)

b) ¿Que tamaño máximo de ventana le permite implementar con el formato de trama que se cuenta?

c) Especificar en un lenguaje de alto nivel el protocolo, suponiendo que se reciben y entregan datos de la capa superior con la primitiva *datoAenviar()* y *datoRecibido()* respectivamente devolviendo como máximo MAX_LARGO bytes, y se envían y reciben tramas del enlace con las primitivas *envioTrama()* y *reciboTrama()* respectivamente .

Notas:

- Cada invocación de la función *datoAenviar()*, devuelve el contenido que será enviado en una sola trama, debiendo numerarse por trama.
- Considerar que no existirán problemas en la conexión y desconexión del protocolo.

Solución:

a) Especificar que tipos de tramas requiere para implementar el protocolo (el campo Type de la trama)

Tipo de Trama	Utilidad
cnx	Para solicitar/aceptar conexión al/del extremo opuesto.
descnx	Para solicitar/aceptar desconexión.
info	Trama conteniendo datos a transferir por el enlace, que incluye ack
ack	Trama conteniendo información de la última trama recibida. Su identificador se especifica en el campo RN.
nack	Trama conteniendo información sobre trama no recibida. Su identificador se especifica en el campo RN.

b) ¿Que tamaño máximo de ventana le permite implementar con el formato de trama que se cuenta?

La condición para el manejo de ventana en Selective Repeat requiere que M módulo del valor máximo de la ventana y N valor de la ventana cumplan:

$$2N <= M$$

El valor de M es 256, ya que se cuenta con un octeto (0..255) para enviar su valor, por lo que el valor máximo de tamaño de ventana será 128.

c) Especificar en un lenguaje de alto nivel el protocolo, suponiendo que se reciben y entregan datos de la capa superior con la primitiva *datoAenviar()* y *datoRecibido()* respectivamente devolviendo como máximo MAX_LARGO bytes, y se envían y reciben tramas del enlace con las primitivas *envioTrama()* y *reciboTrama()* respectivamente .

Para la solución se utilizan dos estructuras de datos, donde se almacenan las tramas enviadas y las tramas recibidas. Las mismas se acceden con las siguientes funciones:

Para las tramas enviadas:

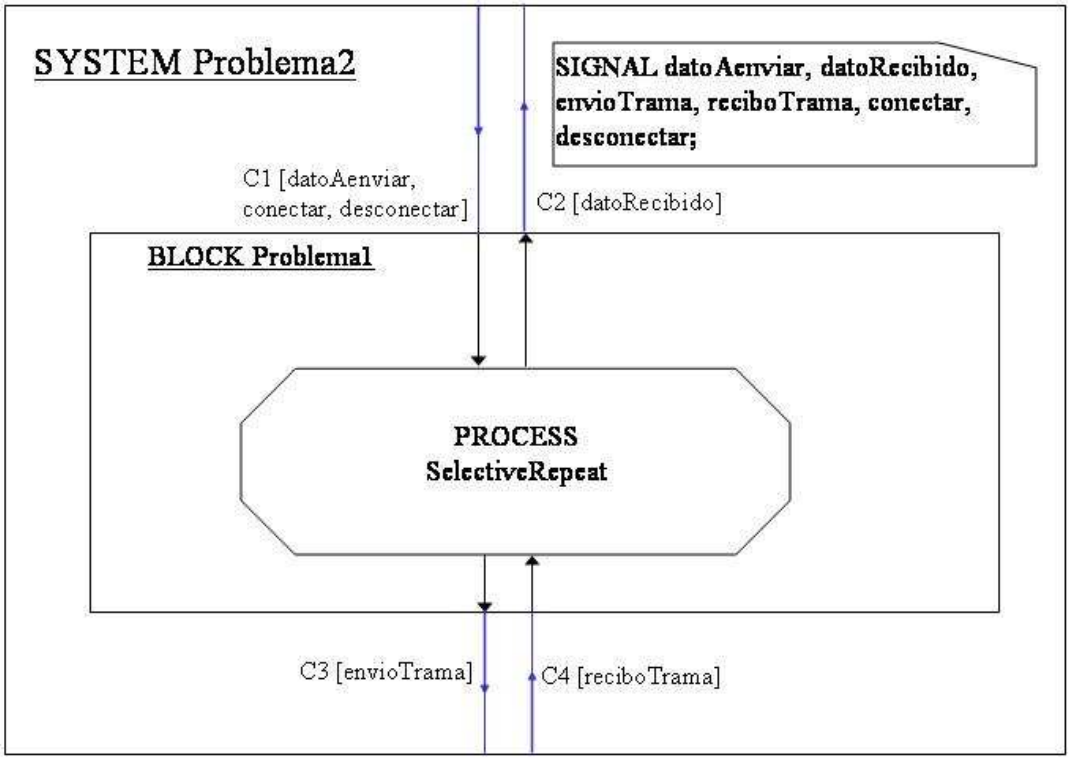
```
getSND(int nroSec)
setSND(int nroSec, string data)
```

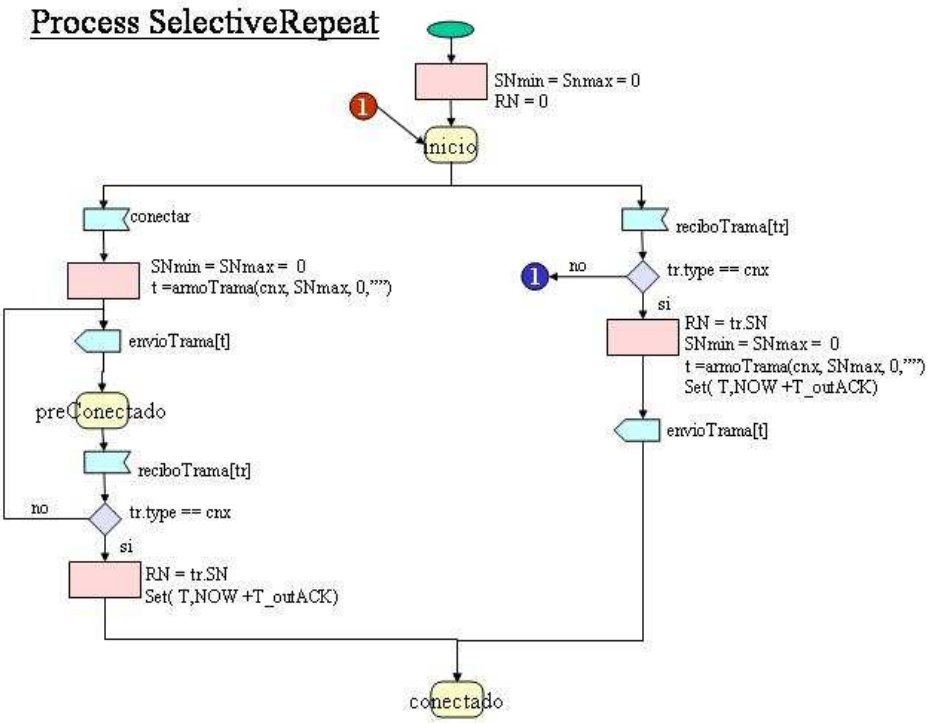
Para las tramas recibidas:

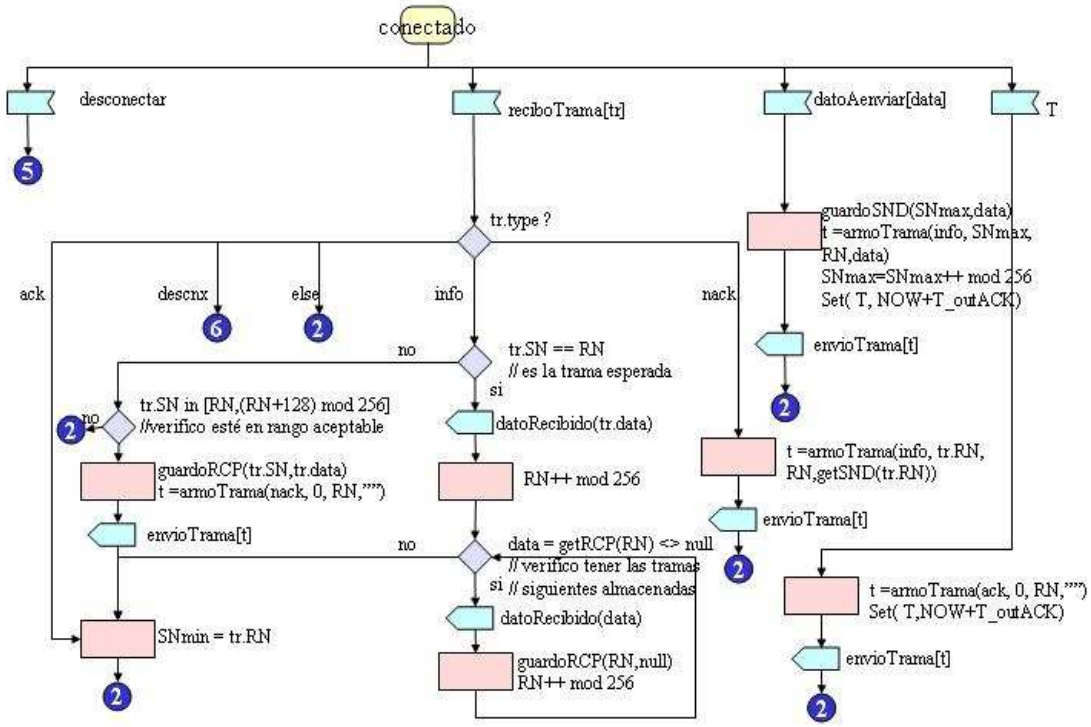
```
getRCP(int nroSec)
setRCP(int nroSec, string data)
```

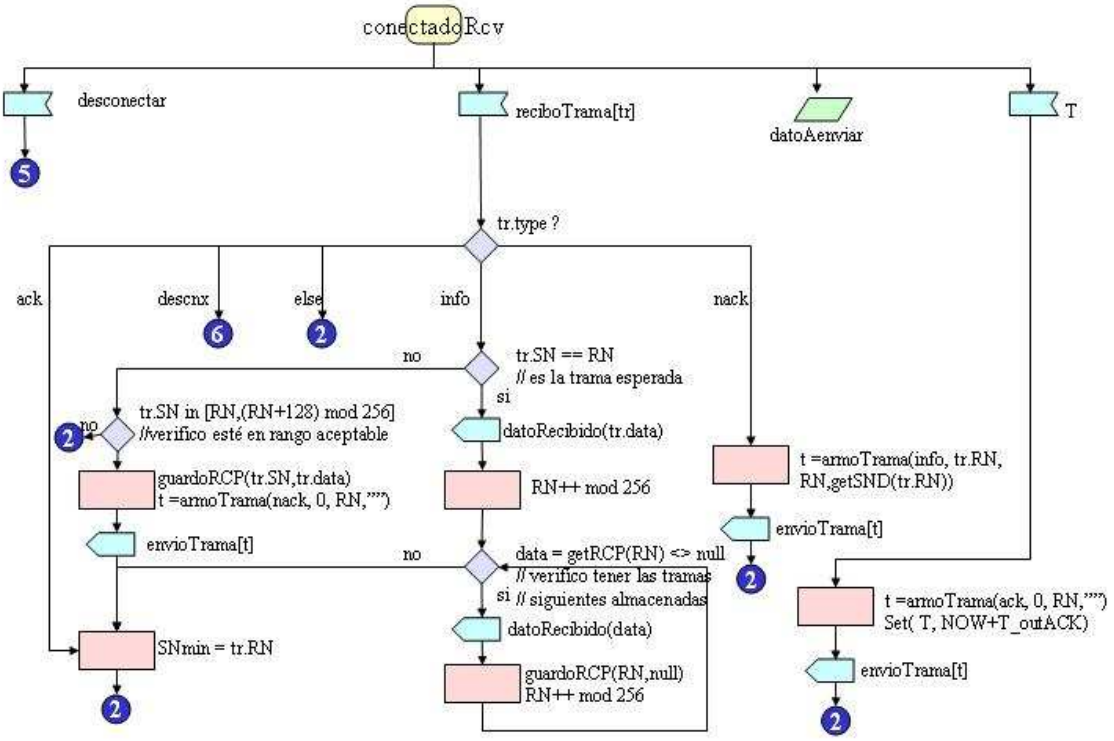
Se utilizan además las siguientes funciones auxiliares:

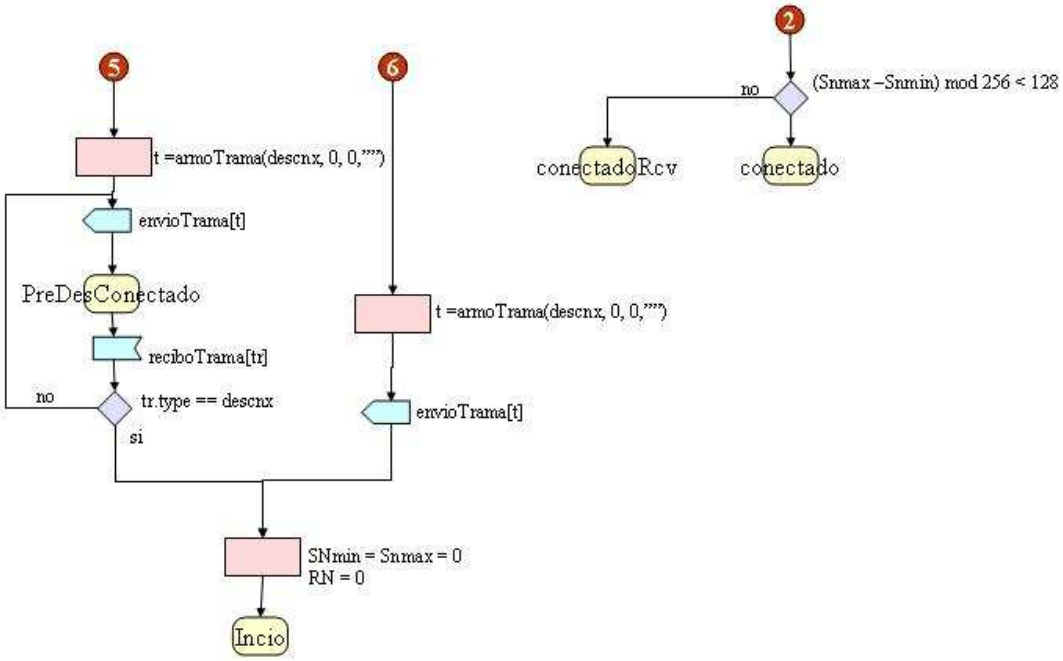
```
armoTrama(byte tipo, byte SN, byte RN, string data)
```


SYSTEM Problema2
SIGNAL datoAenviar, datoRecibido, envioTrama, reciboTrama, conectar, desconectar;
C1 [datoAenviar, conectar, desconectar]
C2 [datoRecibido]
BLOCK Problema1
PROCESS SelectiveRepeat
C3 [envioTrama]
C4 [reciboTrama]


Process SelectiveRepeat
SNmin = Snmax = 0
RN = 0
inicio
conectar
SNmin = SNmax = 0
t =armoTrama(cnx, SNmax, 0,"")
envioTrama[t]
preConectado
reciboTrama[tr]
no
tr.type == cnx
si
RN = tr.SN
Set(T,NOW +T_outACK)
reciboTrama[tr]
no
tr.type == cnx
si
RN = tr.SN
SNmin = SNmax = 0
t =armoTrama(cnx, SNmax, 0,"")
Set(T,NOW +T_outACK)
envioTrama[t]
conectado

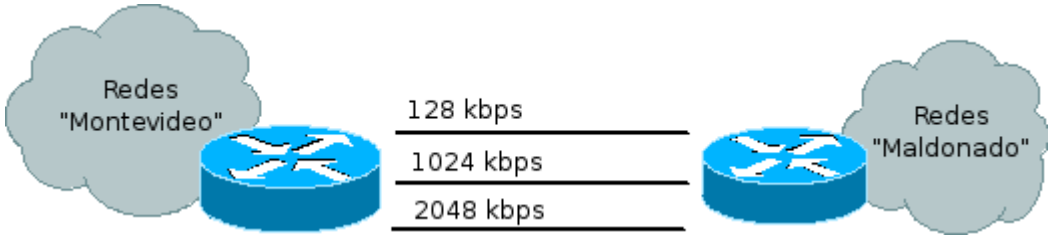

conectado
desconectar
reciboTrama[tr]
datoAenviar[data]
T
5
tr.type ?
ack
descnx
else
info
nack
6
2
no
tr.SN == RN
// es la trama esperada
si
tr.SN in [RN,(RN+128) mod 256]
//verifico esté en rango aceptable
no
2
datoRecibido(tr.data)
guardoRCP(tr.SN,tr.data)
t =armoTrama(nack, 0, RN,"")
RN++ mod 256
envioTrama[t]
no
data = getRCP(RN) <> null
// verifico tener las tramas
si // siguientes almacenadas
SNmin = tr.RN
datoRecibido(data)
guardoRCP(RN,null)
RN++ mod 256
guardoSND(SNmax,data)
t =armoTrama(info, SNmax,
RN,data)
SNmax=SNmax++ mod 256
Set(T, NOW+T_outACK)
envioTrama[t]
2
t =armoTrama(info, tr.RN,
RN,getSND(tr.RN))
envioTrama[t]
2
t =armoTrama(ack, 0, RN,"")
Set(T,NOW+T_outACK)
envioTrama[t]
2


conectadoRcv
desconectar
reciboTrama[tr]
datoAenviar
T
5
tr.type ?
ack
descnx
else
info
nack
6
2
no
tr.SN == RN
// es la trama esperada
si
tr.SN in [RN,(RN+128) mod 256]
//verifico esté en rango aceptable
no
2
datoRecibido(tr.data)
guardoRCP(tr.SN,tr.data)
t =armoTrama(nack, 0, RN,"")
RN++ mod 256
envioTrama[t]
no
data = getRCP(RN) <> null
// verifico tener las tramas
si // siguientes almacenadas
SNmin = tr.RN
datoRecibido(data)
guardoRCP(RN,null)
RN++ mod 256
t =armoTrama(info, tr.RN,
RN,getSND(tr.RN))
envioTrama[t]
2
t =armoTrama(ack, 0, RN,"")
Set(T, NOW+T_outACK)
envioTrama[t]
2


5
6
2
t =armoTrama(descnx, 0, 0,"")
envioTrama[t]
PreDesConectado
reciboTrama[tr]
no
tr.type == descnx
si
t =armoTrama(descnx, 0, 0,"")
envioTrama[t]
SNmin = Snmax = 0
RN = 0
Incio
no
(Snmax –Snmin) mod 256 < 128
conectadoRcv
conectado

Problema 2 (30 puntos)

Se desea analizar y mejorar el comportamiento de un algoritmo de enrutamiento link-state en una topología particular como la de la figura, en la que hay dos grupos de redes interconectadas por un haz de enlaces paralelos.



Se pide:

a) Describa en términos generales el funcionamiento de un algoritmo de enrutamiento del tipo “link state” (OSPF).

b) ¿Como estima que el mismo se comportará en una topología como la de la figura? ¿Que problemas podrían ocurrir en situaciones de alto tráfico y congestión? Estudiar en particular los casos de tráfico mayor a 2048 kbps y entre 1024 y 2048 kbps.

c) Proponga y especifique en un lenguaje de alto nivel, un mecanismo para estimar el ancho de banda de los enlaces. Este mecanismo deberá ser lo menos intrusivo posible, por lo que se sugiere aprovechar el envío de LSAs (Link State Advertisements) para ello. Suponga que puede marcar el tiempo al momento de salida/entrada del paquete al enlace, y los routers tienen su tiempo exactamente sincronizado.

d) ¿Cómo puede mejorar la situación de la parte (b) el conocer el ancho de banda de los enlaces, para la selección del mejor camino?

Solución:

a) Describa en términos generales el funcionamiento de un algoritmo de enrutamiento del tipo “link state” (OSPF).

El funcionamiento de los algoritmos de enrutamiento de tipo “link state” se puede describir en términos generales de la siguiente forma:

1. *Descubrimiento de los vecinos*
2. *Medición de la distancia a los vecinos*
3. *Construir un paquete con la información recopilada y enviarlo a todos los demás routers de la red*
4. *Escuchar la información de topología enviada por otros routers e ir armando con ella el grafo de la red*
5. *Calcular el camino más corto a todos los destinos posibles conocidos en función del grafo de red conocido*

Comentarios:

- *Para los pasos 1 y 2 se utilizan los paquetes HELLO que solamente tienen significado local entre vecinos y no deben ser enrutados.*
- *Para el paso 3 se arma un paquete LSA (link state advertisment) que resume la información de vecinos y métricas para llegar a c/u de ellos*
- *El paso 5 se hace corriendo el algoritmo de Dijkstra para el grafo conocido de la red*
- *Para asegurarnos que los LSA llegan a todos los routers los mismos de inundan (flooding) con mecanismos para descartar los LSA ya vistos y números de secuencia para garantizar que todos los routers tienen información consistente de la topología*
- *La métrica puede ser cualquier cosa que se pueda medir. Desde numero de saltos pasando por ancho de banda y retardo*

b) ¿Como estima que el mismo se comportará en una topología como la de la figura? ¿Que problemas podrían ocurrir en situaciones de alto tráfico y congestión? Estudiar en particular los casos de tráfico mayor a 2048 kbps y entre 1024 y 2048 kbps.

El algoritmo tal como fue descripto en el teórico corre el algoritmo de Dijkstra simple, que para cada destino va a devolver solamente el camino más corto en cada momento de corrida. Por ello la red en cada momento que se corra el D. va a elegir uno de los enlaces.

Discutimos casos, suponiendo que la métrica es algún valor condicionado por el tráfico (si fueran saltos no queda muy claro que enlace sería el elegido y la pregunta como está planteada no tendría sentido)

- *Mientras el tráfico ofrecido es < que 2048 kbps, el enlace elegido es el de 2048 kbps*
- *A medida de que el tráfico ofrecido se acerca a 2048 la métrica sobre este enlace empeora mientras que la métrica por el de 1024 se mantiene constante (pensar en que si mando HELLOs los mismos tienen que esperar cada vez más en cola para ser transmitidos)*
- *A medida que nos adentramos en congestión más profunda, hay un momento de corte en el cual la métrica por el enlace de 1024 es mejor que por el saturado enlace de 2048 y el enlace elegido pasa a ser el de 1024*
- *¿PERO qué pasa? El tráfico ofrecido es mucho mayor que 1024 por lo que este nuevo enlace satura casi inmediatamente y un breve tiempo después el enlace de 2048 vuelve a ser el mejor*
- *Este ciclo de oscilaciones se puede repetir varias veces mientras la carga ofrecida no varíe*

Esta situación es muy perniciosa para la red y ocasiona importantes pérdidas de paquetes.

c) Proponga y especifique en un lenguaje de alto nivel, un mecanismo para estimar el ancho de banda de los enlaces. Este mecanismo deberá ser lo menos intrusivo posible, por lo que se sugiere aprovechar el envío de LSAs (Link State Advertisements) para ello. Suponga que puede marcar el tiempo al momento de salida/entrada del paquete al enlace, y los routers tienen su tiempo exactamente sincronizado.

Dadas las hipótesis se puede aplicar un muy simple algoritmo que consiste en los siguientes pasos:

1. *Armar los LSAs de un tamaño conocido y fijo, eventualmente con padding (relleno) para hacerlos de un largo no muy corto ni demasiado largo, por ejemplo, 1024 bytes.*
    a. *Esto es para eliminar efectos de borde de paquetes muy cortos o muy largos*
2. *Enviar el LSA marcando el timestamp de salida, posiblemente en un campo de opciones adicionales*
3. *El router que recibe el LSA con estas características determina el ancho de banda realizando una simple operación:*

    Bw = L / (Ts2 – Ts1)

    ***Donde**: Ts1 y Ts2 son los timestamps correspondientes (salida y llegada) y L es el largo del LSA*

    ***Observación 1:** La hipótesis de sincronización de todos los routers es **fundamental**, sin ella este proceso no es posible y hay que recurrir a otro tipo de técnicas mucho más complejas*

    ***Observación 2:** La medida debería repetirse en varias oportunidades y promediar, a los efectos de eliminar efectos interferentes del resto del tráfico. No hay que olvidar que los LSAs no son los únicos paquetes que están presentes en la red*

d) ¿Cómo puede mejorar la situación de la parte (b) el conocer el ancho de banda de los enlaces, para la selección del mejor camino?

En el caso más elemental, conocer el ancho de banda me permitiría simplemente evitar oscilaciones porque puedo saber a-priori que el nuevo enlace no va a soportar la carga

Conocido el ancho de banda en este escenario es posible utilizar versiones refinadas del algoritmo de Dijkstra que no solo determinan el mejor camino sino eventualmente también caminos alternativos.