Examen de Introducción a las Redes de Computadoras y Comunicación de Datos (ref: eirc0712.doc) 27 de diciembre 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)

Explique resumidamente los 5 mensajes que se implementa en OSPF.

Pregunta 2 (10 puntos)

Explique el algoritmo de cubeta con goteo *Leaky Bucket* y cubeta con tokens *Token Bucket*.

Pregunta 3 (10 puntos)

Explique en que consiste la conmutación de circuitos, conmutación de paquetes y conmutación de mensajes. Realice un análisis comparativo.

Pregunta 4 (10 puntos)

Explique en que consiste la fragmentación transparente y la fragmentación no transparente.

Pregunta 5 (5 puntos)

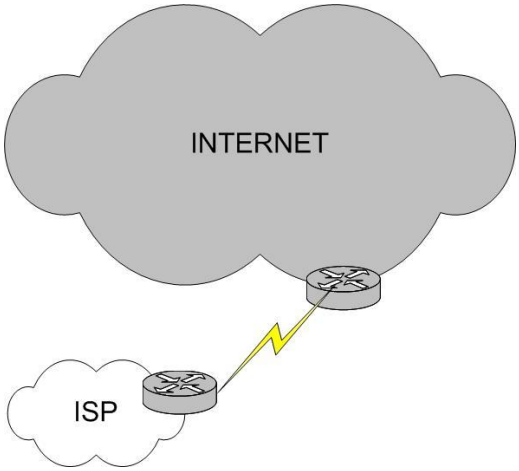
¿Que función cumple el router de ingreso y egreso en MPLS?

Problemas Prácticos

Problema 1 (30 puntos)

Sea un ISP, que arrienda servicios de conectividad a Internet, a un proveedor que brinda dos clases de enlaces (con costos relativos según la calidad del mismo):

- DEFAULT: que brinda un servicio "*best effort*", sin asegurar su resultado
- G0LD: enlaces que aseguran una baja perdida, retardo y jitter.



El ISP desea brindar un servicio que sea percibido como mejor que los otros ISP's por el cliente, enviando el tráfico TCP/UDP, de aplicaciones que requieren parámetros bajos de pérdida retardo y jitter (como ser VoIP -Voice over Internet Protocol o IPTV - IP Internet Protocol Televisión), por un enlace con éstas características y el tráfico restante por un enlace estándar.

Para esto se cuenta con un analizador del tráfico, que para el paquete IP pasado por parámetro, intenta determinar cual es el tipo de servicio requerido por la aplicación que generó el paquete, y devuelve el enlace sugerido:

- [UNKNOWN,DEFAULT,GOLD] tipoTrafico(pkg:IP_pkg)

En caso de no conocer el tipo de enlace requerido por la aplicación que genera el tráfico, devuelve UNKNOWN.

Para notificar al proveedor del servicio de Internet, que enlace utilizar, se cuenta con el campo TOS (Type of Service), donde se colocará la constante:

- DEFAULT_TYPE para enlace *best effort*
- GOLD_TYPE para enlace *gold*.

Además se cuenta con un procedimiento que configura el campo TOS del datagrama IP

- `void setTOS (tipo: [DEFAULT_TYPE,GOLD_TYPE],pkg:IP_pkg)`

Se pide:

a) ¿Cómo podría implementarse la función *tipoTráfico()*? ¿Que ventajas y desventajas encuentra a las implementación planteada?

b) Analice políticas a aplicar al tráfico clasificado como UNKNOWN, y las consecuencias que la misma genera.

c) Especifique en un lenguaje de alto nivel, un procedimiento que impelente lo especificado, a nivel de Capa 3 (Red).

Nota:

- El analizador de tráfico *tipoTrafico()* y la función *setTOS()*, se suponen ya implementadas.
- Suponga que todos los servicios se encuentran definidos sobre *well known ports*

Solución:

a) ¿Cómo podría implementarse la función *tipoTráfico()*? ¿Que ventajas y desventajas encuentra a las implementación planteada?

Bajo la suposición de que todos los servicios se encuentran definidos sobre *well known ports*, el mecanismo para la implementación de la función tipoTrafico(), podría ser el siguiente:

- Implementar un lista de puertos, con tipo de servicio que requiere (GOLD, DEFAULT)
- Dado un paquete, verificar si contiene un segmento TCP/UDP, y en caso afirmativo, determinar puerto origen y destino.
- En caso que puerto origen o puerto destino, se encuentre en la lista, devolver el tipo definido, y en caso contrario devolver UNKNOWN

El pseudocódigo propuesto es:

```
[UNKNOWN,DEFAULT,GOLD] tipoTrafico(pkg:IP_pkg) {
      if ( IP_Pkg.PROTOCOL == TCP_CODE || IP_Pkg.PROTOCOL == UDP_CODE ){
            payload = IP_Pkg.payload;
            sourcePort = payload.sourcePort;
            destinationPort = payload.destinationPort;
            if ( sourcePort in listaTipoTrafico )
                  return (listaTipoTrafico(sourcePort));
            if ( destinationPort in listaTipoTrafico )
                  return (listaTipoTrafico(destinationPort));
            else
                  return UNKNOWN;

      } else
            return UNKNOWN;
}
```

Ventajas :

- El cómputo de la función es muy simple. El algoritmo accede a la tabla y devuelve el valor configurado.
- Ante el surgimiento de nuevas aplicaciones, simplemente debe agregarse una línea en la tabla.
- Se puede clasificar el tráfico, simplemente con la presencia de un paquete, no requiere tener presente información referente a varios paquetes del flujo.

Desventajas:

- Para el cliente del ISP, conociendo el esquema utilizado, podría seleccionar la utilización de un puerto que utilice GOLD, y realizar un Proxy para cambiar alguno de los puertos. Simplemente con esta operación conseguiría tener un enlace GOLD para su aplicación.
- Ante la aparición de nuevas aplicaciones, se requiere de la intervención humana para que comience a registrarlas.

Los otros métodos exigen la visualización del contenido del paquete, y a través de algún algoritmo determinar que aplicación lo generó.

b) Analice políticas a aplicar al tráfico clasificado como UNKNOWN, y las consecuencias que la misma genera.

La decisión debe tomarse en función de la relación entre costos del los enlaces GOLD y DEFAULT.

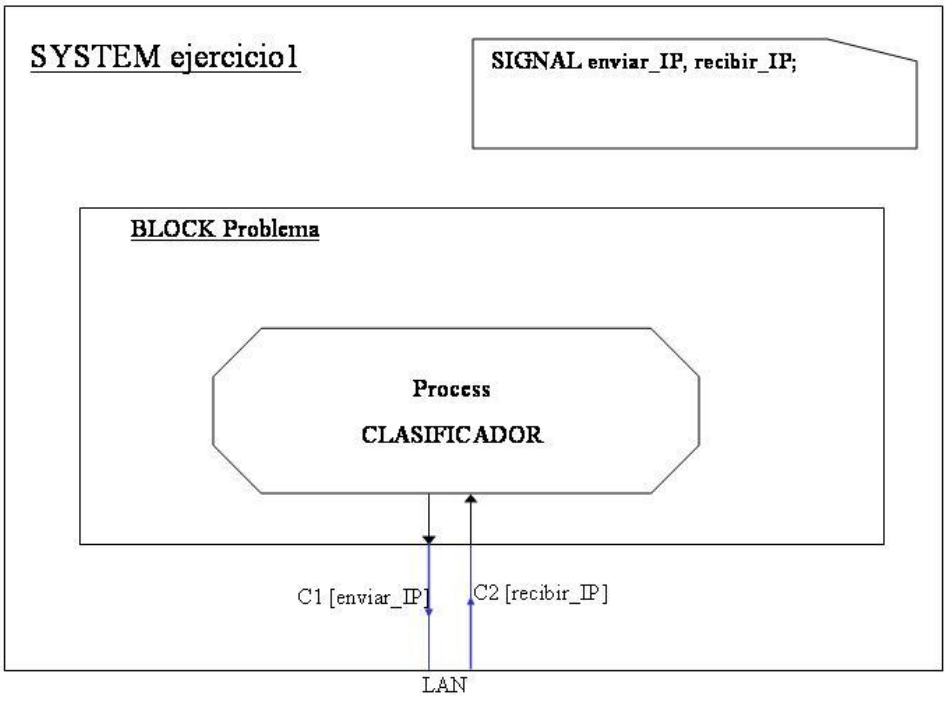
Es obvio, que si se enruta por el enlace GOLD, el usuario percibirá que funciona mejor, y en caso de enlutarse por el enlace DEFAULT, el mismo se comportará según el estado de saturación de los enlaces.

En principio puede afirmarse, que el tráfico catalogado como UNKNOWN, no pertenece a aplicaciones conocidas (o cuyo origen se desconoce). Es por esto que para el ISP, lo más razonable será utilizar el enlace más económico, DEFAULT.

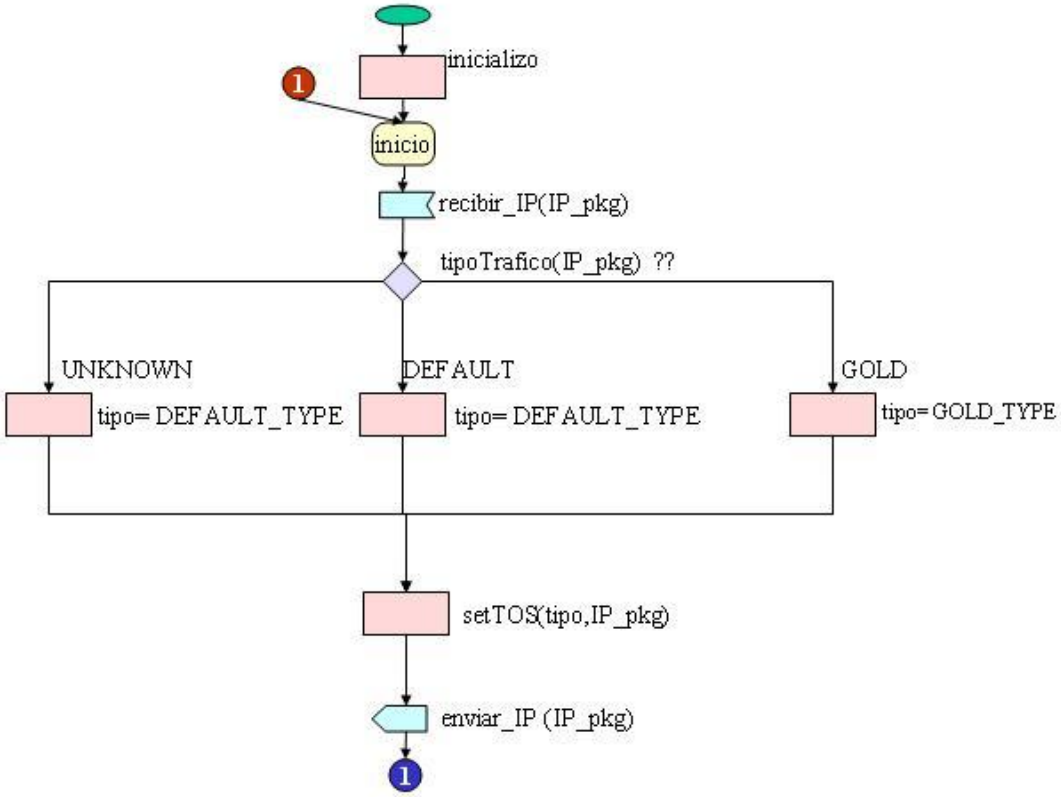
La asignación al enlace GOLD, no implica un error, sino que económicamente no sería razonable para el ISP. Podrían existir otros motivos, como puede ser sobresalir a la competencia, que podrían llevar a tomar ésta decisión.

Lo que en ningún caso es admisible es el descartar el tráfico UNKNOWN. Que no se encuentre registrado, no implica que no deba entregarse, además que el ISP debe resolver la conectividad con Internet, para cualquier aplicación, no solamente para las registradas.

c) Especifique en un lenguaje de alto nivel, un procedimiento que impelente lo especificado, a nivel de Capa 3 (Red).

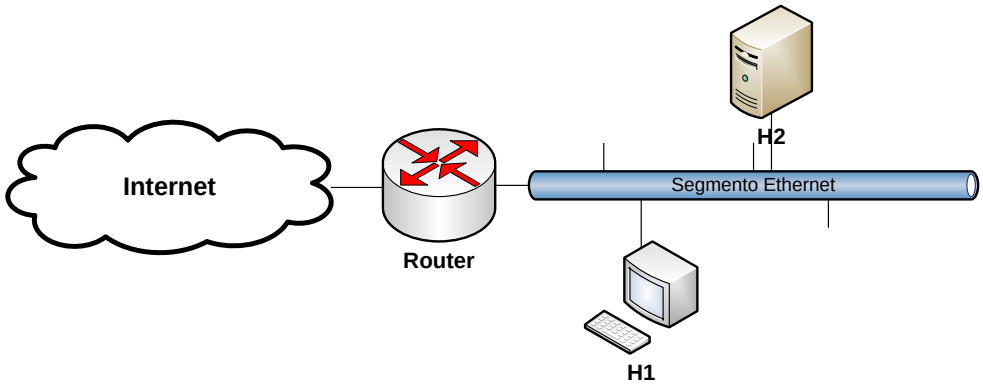


Process CLASIFICADOR



Problema 2 (30 puntos)

a) Ejemplifique el funcionamiento del protocolo ARP (*Address Resolution Protocol*), ejemplificando en una red compuesta por dos hosts (H1 y H2), una LAN Ethernet y un router que la conecta a Internet.



Para ello, discuta los casos:

   a. H1 envía un paquete IP a H2
   b. H1 envía un paquete IP a un sitio en internet, por ejemplo a la dirección IP del servidor *www.google.com*.

b) Diseñe un mecanismo para que el host H1 y el host H2 puedan compartir una dirección IP. El objetivo es *proveer redundancia a un servidor web*, por lo que se deben cumplir los siguientes objetivos:

   a. Uno de los hosts debe ser el "primario" y el otro el secundario para la IP compartida
   b. En caso de falla del primario, el "secundario" debe tomar el control. Cuando el primario se reinicia, el secundario debe devolver el control al primario.
   c. La solución debe estar basada únicamente en los hosts (el router no ayuda en nada) y debe estar basada en las respuestas de ARP que los hosts H1 y H2 devuelven. Tener en cuenta que H1 y H2 tienen direcciones IP propias además de la dirección IP compartida.
   d. Se puede asumir que cuando todo el sistema arranca, H1 es el primario y H2 es el secundario.

Solución:

a) El primer paso que cualquier stack IP realiza previamente a enviar (o reenviar en el caso de un router) un paquete es hacer una búsqueda en su tabla de enrutamiento. De esta búsqueda puede surgir que el destino esta en una misma subred que el host local o que es un destino remoto. Estudiamos los casos por separado:

   a. <u>Envio de un paquete IP desde H1 a H2</u>
   En este caso, como H1 y H2 estan en la misma subred (información obtenida de la tabla de enrutamiento), H1 envia un mensaje ARP a la dirección de broadcast de la Ethernet (FF:FF:FF:FF:FF:FF) preguntando por la MAC address que corresponde a la dirección IP de H2.

El host H2 responde con un paquete ARP dirigido a la MAC de H1 (esta respuesta ya no es un broadcast) diciendo que la IP de H2 corresponde con una cierta MAC.
Estos mensajes son conocidos como "*who-has*" y "*is-at*"
Finalmente entonces, H1 arma una trama Ethernet con MAC de destino la de H2 y origen la propia y como payload el paquete IP que se desea enviar.

b. Envio de un paquete IP desde H1 a *www.google.com*
En este caso, una vez realizada la búsqueda en DNS para obtener la IP de www.google.com, se hace la misma búsqueda en la tabla de enrutamiento. En este caso, el resultado dira que el destino es remoto, no esta en una subred local.
En este caso, debo enviarle el paquete al siguiente salto según lo indica la tabla de enrutamiento.
En este caso, el host H1 necesita la MAC del siguiente salto, por lo cual se realiza el mismo envio y espera de respuesta de "*who-has"* y "*is-at"*, pero esta vez la pregunta es por la MAC del siguiente salto.
Una vez conseguida la respuesta, H1 arma una trama Ethernet con MAC de origen la propia y de destino la del siguiente salto y como payload el paquete IP que tiene como IP destino la remota (google) y de origen la propia.

b) Para resolver este problema necesitamos dos procesos, uno que verifique la disponibilidad del primario y otro que en función de si soy primario o no, responda los pedidos de MAC "*who-has"* que lleguen para la MAC de la IP compartida.

Recordar que los "*who-has"* son dirigidos al broadcast, por lo cual estos llegan tanto a H1 como a H2.

Hipotesis tengo para usar:

- Una bandera booleana en memoria compartida llamada *soy_primario*, que vale **true** si actualmente soy primario y **false** en otro caso.
- Cuando los sistemas arrancan, uno está configurado para arrancar como primario y el otro para arrancar como secundario. Hay una bandera booleana compartida *arranco_primario* para indicar esto
- Primitivas TCP

La verificación se hace cada un cierto tiempo (no continuamente).

a. Proceso CHECK_SERVICE

```
Process CHECK_SERVICE

Var soy_primario
Var arranco_primario
Var my_ip /* dirección IP propia, no compartida */
Var remote_ip /* dirección IP del otro host, no la compartida */
Var T = 30s /* verifico temporariamente */

Begin

      /* si arranco primario, verifico que el secundario no este
         ya contestando */
      If ( arranco_primario ) then
            If ( tcp_connect (remote_ip, 80) == SUCCESS ) then
                  soy_primario = False
            End
      End
```

```
	While (true)
		If (not soy_primario) then
			If ( tcp_connect (remote_ip, 80) == FAIL ) then
				soy_primario = True
			End
		End
		sleep(T)
	End
End
```

b. Proceso RESPONDE_ARP

```
Process RESPONDE_ARP

Var soy_primario
Var arranco_primario
Var my_ip /* dirección IP propia, no compartida */
Var remote_ip /* dirección IP del otro host, no la compartida */
Var shared_ip   /* ip compartida */
Var my_mac /* mi mac address */
Var T = 30s /* verifico temporariamente */

Begin
	While(True)
		If (arp = recibo_trama_arp() )
			If arp.ip_buscada == my_ip then
				/* respondo arp normal */
				respondo_arp(my_mac, my_ip)
			End
			If arp.ip_buscada == shared_ip AND soy_primario then
				/* respondo arp normal, pero para la ip comp. */
				respondo_arp(my_mac, shared_ip)
			End
		End
	End
End
```