Redes de Computadoras

Examen – 16 de diciembre de 2019
(ref: erc20191216.odt)

Instrucciones

- **Indique su nombre completo y número de cédula en cada hoja.**
- **Numere todas las hojas e indique la cantidad total de hojas que entrega en la primera.**
- **Escriba las hojas de un solo lado y utilice una caligrafía claramente legible.**
- **Comience cada pregunta teórica y cada ejercicio en una hoja nueva.**
- **Solo se responderán dudas de letra. No se responderán dudas de ningún tipo los últimos 30 minutos del examen.**
- **El examen es individual y sin material. Apague su teléfono celular mientras esté en el salón del examen.**
- **Es obligatorio responder correctamente al menos 15 puntos en las preguntas teóricas y 20 de los problemas prácticos. Los puntos ganados en el curso se suman a los puntos de teórico.**
- **El puntaje mínimo de aprobación es de 60 puntos.**
- **Para todos los ejercicios, si es necesario, puede suponer 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).**
- **Justifique todas sus respuestas.**
- **Duración: 3 horas. Culminadas las 3 horas el alumno no podrá modificar las hojas a entregar de ninguna forma.**

Preguntas Teóricas

Pregunta 1 (10 puntos)

En el servicio de DNS se encuentran configurados, entre otros, los siguientes registros:

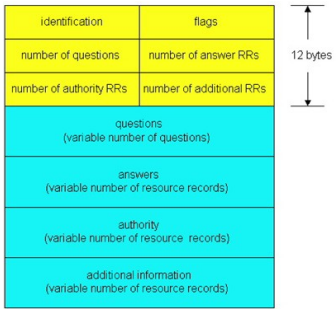
```
cert.br          IN          NS          ns.dns.br
cert.br          IN          NS          ns2.dns.br
ns.dns.br        IN          A           200.160.0.5
ns.dns.br        IN          AAAA        2001:12ff:0:a20::5
ns2.dns.br       IN          A           200.192.232.53
```

Desde una computadora se realiza una consulta recursiva a un servidor no autoritativo por el servidor de nombres del dominio `cert.br`.

a) Muestre el formato y contenido del mensaje *query* enviado.
b) Muestre el formato y contenido del mensaje *reply* recibido.

Solución

El formato de los mensajes query y reply (o query response) es el mismo, el que se muestra a continuación.



a)
Lleva un Identificador (Transaction ID) para asociar el mensaje reply al query, generado por quien crea el mensaje query.
Flags
Existe una flag (1 bit) que indica el tipo de mensaje: si está en 0 es una query (este caso).
Otra flag (1 bit) indica si se pretende que sea una query recursiva.
Lleva un Identificador (Transaction ID) para asociar el mensaje reply al query.
Los valores de los campos "contadores" ("number of...") son los siguientes:
Questions: 1
Answers: 0
Authority: 0
Additional: 0
En el campo Questions el contenido es
cert.br: type NS, class IN

No existen campos Answers, Authority ni Additional.

b)
Lleva el Identificador copiado del mensaje query que lo motiva
Flags
Flag (1 bit) en 1 indicando que es un reply (query response)
Otra flag (1 bit) indica si se pretende que sea una query recursiva.
Otra flag (1 bit) indica si se dispone de servicio recursivo.
Los valores de los campos "contadores" ("number of...") son los siguientes:
Questions: 1
Answers: 2
Authority: 0
Additional: 3
En el campo Questions el contenido es
cert.br: type NS, class IN
En el campo Answers el contenido es
cert.br: type NS, class IN, ns ns.dns.br
cert.br: type NS, class IN, ns ns2.dns.br
No existe campo Authority.
En el campo Additional el contenido es
ns.dns.br: type A, class IN, addr 200.160.0.5
ns.dns.br: type AAAA, class IN, addr 2001:12ff:0:a20::5
ns2.dns.br: type A, class IN, addr 200.192.232.53

Pregunta 2 (10 puntos)

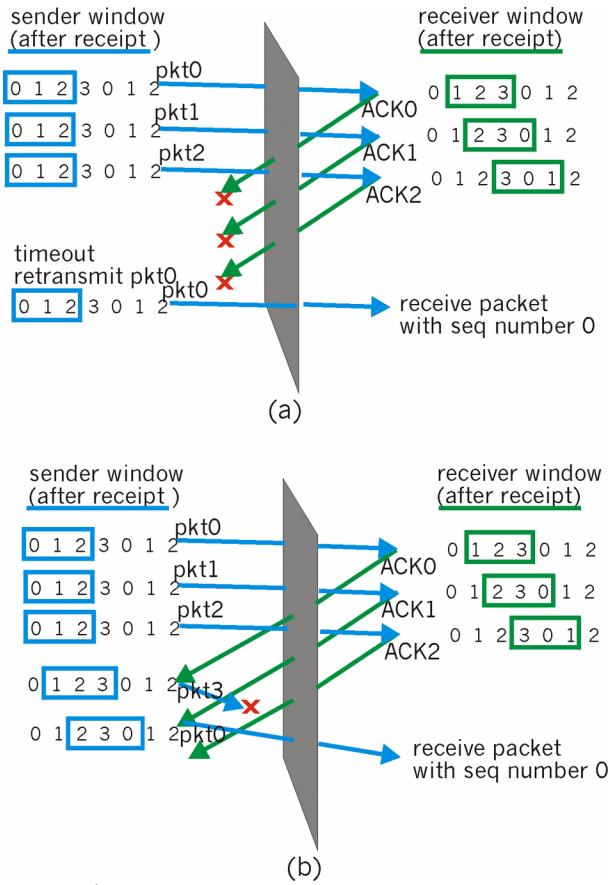
a) Explique las diferencias entre protocolos de transporte stop-and-wait y en pipeline.
b) ¿Por qué son necesario los segundos?
c) De un ejemplo del problema de tener tamaños de ventana muy parecidos al rango de números de secuencia en protocolos de tipo Selective Repeat.

Solución:

a) En un protocolo stop-and-wait, el emisor espera por el reconocimiento de un paquete para empezar la transmisión de un nuevo paquete.
En el caso de los protocolos de pipeline, el emisor puede enviar varios paquetes sin esperar a los mensajes de reconocimiento. Para esto es necesario definir una ventana de transmisión que definirá cuantos paquetes se pueden transmitir "simultáneamente" sin haber recibido un reconocimiento.

b) Los protocolos de pipeline permiten realizar un mejor aprovechamiento de los recursos disponibles. En un protocolo stop-and-wait, por cada transmisión de un paquete el emisor debe esperar por el reconocimiento generando una cantidad importante de tiempo ocioso. En el caso del pipeline, el emisor está una mayor cantidad de tiempo transmitiendo y haciendo uso del canal.

c) Para el caso de un protocolo de pipeline, una opción es utilizar selective repeat para el reconocimiento de los paquetes. Esto implica que el receptor genera un paquete de reconocimiento por cada paquete recibido. Del lado del emisor, solo se reenvían paquetes que no fueron reconocidos luego de un cierto tiempo.
Imaginemos un escenario con una ventana de transmisión de 3 paquetes y número de secuencia del 0 al 3. Se ilustra a continuación un posible problema:



En la imagen (a) se puede observar que el receptor no detecta que el emisor está enviando un mensaje duplicado y considera que es un nuevo paquete.
Ver que no puede diferenciarlo del caso (b) donde es un nuevo paquete con secuencia 0.

Pregunta 3 (10 puntos)

Utilizando las primitivas de la API de sockets del curso, implemente una aplicación que espera por una conexión TCP en el puerto 1234 y recibe un mensaje de largo desconocido que finaliza con un salto de línea. Describa la funcionalidad de cada primitiva utilizada.

Solución:

```
function main ()
   master = socket.tcp() //Crea un socket TCP

   master.bind (*, 1234)
   //Asocia el socket a una interfaz (en esta caso todas) y un puerto local

   server = master.listen()
   //Convierte el socket en un socket server capaz de recibir conexiones.

   client, err = server.accept()
   //Espera a que se establezca una conexión.
   //Devuelve un socket cliente TCP conectado. Es una operación bloqueante.

   buffer = []
   do
      data, err = client.receive()
      //Realiza una lectura en un socket conectado.
      //Devuelve la información disponible en el stream hasta el momento.
      buffer += data
   until (find(data, '\n') || err)
```

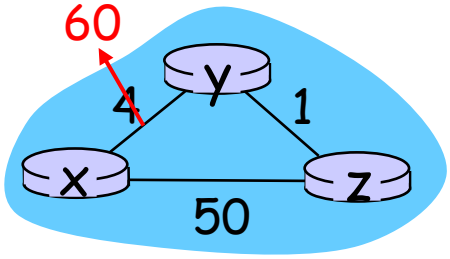
```
    server.close()
    client.close()
    // Cierra los sockets finalizando la conexión TCP

    return data

end function
```

Pregunta 4 (10 puntos)

a) Identifique la causa del problema de conteo infinito en los algoritmos Vector-Distancia. Describa que es lo que sucede con los vectores distancia de los nodos involucrados.
b) ¿Por qué es imposible que se de este problema en algoritmos de Estado del Enlace (*Link-State*)?

Solución:

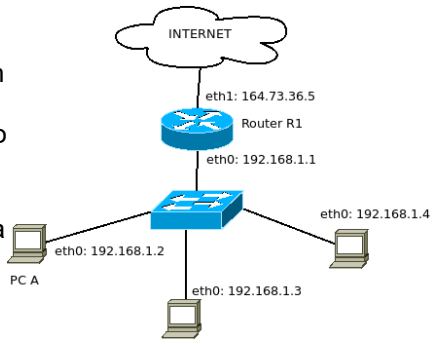
a) El problema de conteo a infinito sucede debido a que cada nodo solo tiene información parcial del estado de la red. Cada nodo recibe de sus vecinos un vector de distancias que informa el costo del camino para alcanzar los demás nodos de la red pero no recibe información del camino ni de los costos intermedios del camino.
Esto genera que un nodo pueda calcular sus costos con información desactualizada y propagar esta información a sus vecinos.
En este contexto, imaginemos el caso de la figura.



Antes del cambio de costo, Y le había informado a Z que alcanza a X con costo 4 y Z le había informado a Y que alcanza a X con costo 5.
Cuando el nodo Y detecta un aumento del costo de uno de sus enlaces, informa a sus vecinos su nuevo vector de distancias donde alcanza a X con costo 6 (por Z). X no puede saber que el camino que Z le informó pasa por él y por lo tanto por el enlace que aumentó de costo
A su vez, Z tiene en su vector que alcanza a X con costo 5 (lo que le informó Y originalmente mas 1), por lo que re calcula su vector y ahora alcanza a X con costo 7. Difunde esto a sus vecions. Z no puede saber que ese camino pasa por el enlace que aumentó de costo.
Esto genera un intercambio de los vectores de Y y Z, donde por cada iteración se aumenta en 1 el costo hasta detectar que el mejor camino es por el enlace de costo 50.

b) Esto no puede suceder en los algoritmos de Estado de Enlace porque cada nodo conoce toda la topología de la red en base al *flooding* de los mensajes *link-state*, y a partir de esta información calcula los caminos de costo mínimo.

Problema 1 (30 puntos)



Se cuenta con una red LAN privada que tiene acceso a Internet mediante un router que implementa NAT como se muestra en la figura.
Todas las PCs de la red LAN tiene configurada la `IP 192.168.1.1` como *default gateway* y como servidor DNS.

Considere el escenario donde la PC A realiza una solicitud HTTP para descargar la página web `www.fing.edu.uy/index.html`.

Se pide:

a) Enumere y describa todos los mensajes de capa 2, 3 y 4 que atraviesan la interfaz eth0 de la PC A. Especifique claramente direcciones, puertos y la función de cada mensaje.
Considere que no existe información de ningún tipo cacheada al ejecutar la solicitud.
Se sugiere utilizar una tabla con la siguiente información (cuando corresponda):
Tipo de mensaje – Cabezal de capa 2 – Cabezal de capa 3 – Payload

b) Explique el funcionamiento del mecanismo NAT y muestre los mensajes (con direcciones y puertos) que atraviesan las interfaces eth0 y eth1 del router cuando se trasmiten la solicitud y respuesta HTTP.

Solución:

Tipo de mensaje	**Cabezal capa 2**	**Cabezal capa 3**	**Payload**
ARP Request generado por la PC A	MAC orig: MAC_A_eth0 MAC dest: FF:FF:FF:FF:FF:FF	-	sender_mac: MAC_A sender_IP: 192.168.1.2 target_mac: 00:00:00:00:00:00 target_IP: 192.168.1.1
ARP Response generado por R1	MAC orig: MAC_R1_etho MAC dest: MAC_A_eth0	-	sender_mac: MAC_R1_eth0 sender_IP: 192.168.1.1 target_mac: MAC_A target_IP: 192.168.1.2
DNS Query	MAC orig: MAC_A_eth0 MAC dest: MAC_R1_etho	IP orig: 192.168.1.2 IP dest: 192.168.1.1	Datagrama UDP con: Puerto orig: 12345 Puerto dest: 53 en cuyo payload viaja la consulta DNS de tipo A por `www.fing.edu.uy`
DNS Reply	MAC orig: MAC_R1_etho MAC dest: MAC_A_eth0	IP orig: 192.168.1.1 IP dest: 192.168.1.2	Datagrama UDP con: Puerto orig: 53 Puerto dest: 1234 en cuyo payload viaja la respuesta DNS de la consulta anterior
Inicio de conexión TCP	MAC orig: MAC_A_eth0 MAC dest: MAC_R1_etho	IP orig: 192.168.1.2 IP dest: IP_fing	Segmento TCP con: Puerto orig: 12345 Puerto dest: 80 con la bandera SYN encendida.
Respuesta de conexión TCP	MAC orig: MAC_R1_etho MAC dest: MAC_A_eth0	IP orig: IP_fing IP dest: 192.168.1.2	Segmento TCP con: Puerto orig: 80 Puerto dest: 12345 con las banderas SYN y ACK encendidas.
Ack de conexión TCP	MAC orig: MAC_A_eth0 MAC dest: MAC_R1_etho	IP orig: 192.168.1.2 IP dest: IP_fing	Segmento TCP con: Puerto orig: 12345 Puerto dest: 80 con la bandera ACK encendida.
Pedido HTTP GET	MAC orig: MAC_A_eth0 MAC dest: MAC_R1_etho	IP orig: 192.168.1.2 IP dest: IP_fing	Segmento TCP con: Puerto orig: 12345 Puerto dest: 80 en cuyo payload viaja la solicitud HTTP: `GET /index.html HTTP/1.1` (puede estar incluido en el segmento TCP ACK anterior)

Respuesta HTTP	MAC orig: MAC_R1_etho MAC dest: MAC_A_eth0	IP orig: IP_fing IP dest: 192.168.1.2	Segmento TCP con: Puerto orig: 80 Puerto dest: 12345 en cuyo payload viaja la respuesta HTTP: HTTP/1.1 200 OK y la página HTML
Reconocimiento del segmento TCP anterior	MAC orig: MAC_A_eth0 MAC dest: MAC_R1_etho	IP orig: 192.168.1.2 IP dest: IP_fing	Segmento TCP con: Puerto orig: 12345 Puerto dest: 80 con la bandera ACK encendida.
Fin de conexión TCP desde cliente	MAC orig: MAC_A_eth0 MAC dest: MAC_R1_etho	IP orig: 192.168.1.2 IP dest: IP_fing	Segmento TCP con: Puerto orig: 12345 Puerto dest: 80 con la bandera FIN encendida.
Respuesta de fin de conexión TCP	MAC orig: MAC_R1_etho MAC dest: MAC_A_eth0	IP orig: IP_fing IP dest: 192.168.1.2	Segmento TCP con: Puerto orig: 80 Puerto dest: 12345 con las banderas FIN y ACK encendidas.
Fin de conexión TCP desde servidor	MAC orig: MAC_R1_etho MAC dest: MAC_A_eth0	IP orig: IP_fing IP dest: 192.168.1.2	Segmento TCP con: Puerto orig: 80 Puerto dest: 12345 con la bandera FIN encendida.
Respuesta de fin de conexión TCP	MAC orig: MAC_A_eth0 MAC dest: MAC_R1_etho	IP orig: 192.168.1.2 IP dest: IP_fing	Segmento TCP con: Puerto orig: 12345 Puerto dest: 80 con las banderas FIN y ACK encendidas.

Parte b

Debido a que no es posible utilizar el espacio de direcciones privado que tiene asignada la LAN para comunicarse hacia Internet, el mecanismo NAT realiza una traducción de estas direcciones privadas en una dirección pública.
Debido a que debe traducir varias direcciones privadas en una única dirección pública hacia afuera de la red LAN, debe utilizar una tabla de traducciones que incluye los número de puerto para realizar este mapeo.

En nuestro escenario, el router recibe por su interfaz eth0 un paquete IP con:
IP orig: 192.168.1.2
Puerto orig: 12345
IP dest: IP_fing
Puerto dest: 80

Al recibir el paquete, el router NAT genera un nuevo número de puerto origen (por ej. 5001) para el paquete, sustituye la dirección IP de origen por su dirección IP en la red pública: 164.73.36.5 y sustituye el número de puerto origen original (12345) por el nuevo número de puerto (5001).
Finalmente almacena el siguiente mapeo en su tabla de traducciones:

Red pública	Red privada
164.73.36.5, 5001	192.168.1.2, 12345

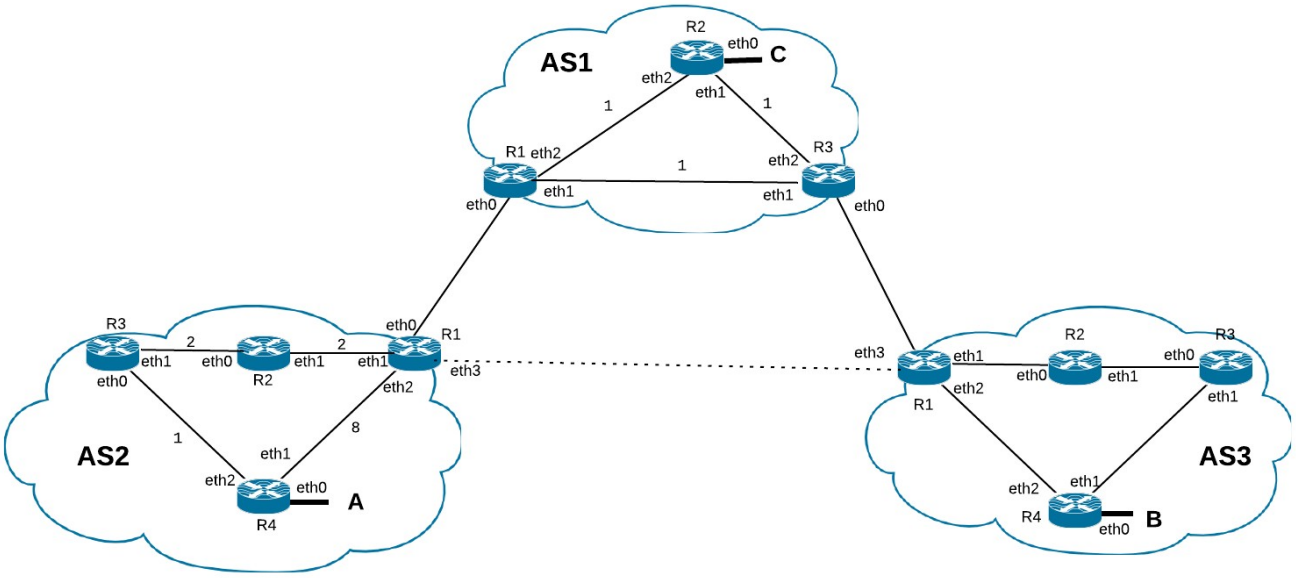
Por lo tanto, desde la interfaz eth1 se transmite a la red un paquete IP con los siguientes datos:
IP orig: 164.73.36.5
Puerto orig: 5001
IP dest: IP_fing
Puerto dest: 80

Cuando el router NAT recibe la respuesta del servidor web, recibe un paquete IP con los siguientes datos:
IP orig: IP_fing
Puerto orig: 80

```
IP dest: 164.73.36.5
Puerto dest: 5001

Luego, el router busca en su tabla de traducciones utilizando la dirección IP de destino y el
puerto de destino y obtiene la dirección IP y puerto necesarios para hacer el cambio
nuevamente.
Por lo tanto, el router NAT modifica esta vez la dirección IP y puerto de destino y se obtiene
el siguiente paquete que será transmitido por la interfaz eth0 hacia la red privada:
IP orig: IP_fing
Puerto orig: 80
IP dest: 192.168.1.2
Puerto dest: 12345
```

Problema 2 (30 puntos)

Considere la red mostrada a continuación. Suponga que los sistemas autónomos AS1 y AS2 están ejecutando OSPF como protocolo de enrutamiento interno, mientras que el AS3 está ejecutando RIP. Suponga además que se utilizan sesiones eBGP y iBGP para el protocolo de enrutamiento entre sistemas autónomos, e inicialmente no existe enlace físico entre AS2 y AS3.



Se conoce que el AS2 tiene disponible el prefijo 121.0.0.0/24, el AS3 122.0.0.0/16 y el AS1 el 123.0.0.0/25.

Se pide:

a) Numere las subredes A, B y C de la manera más ajustada posible. Debe tener en cuenta que la subred A debe soportar hasta 100 interfaces y además considerar un crecimiento del 26%. La subred B hasta 192 interfaces y la subred C hasta 31 interfaces. Numere además las subredes intra-routers únicamente del AS1 con el prefijo 10.0.0.0/8 y las inter-AS con el 120.0.0.0/24.

b) Escriba la tabla de forwarding para los routers R1 y R4 del AS2. Debe indicar para cada entrada el protocolo con el que la misma fue aprendida (OSPF, RIP, iBGP, eBGP), complementando de ser necesario con los parámetros propios del protocolo.

c) Ahora suponga que existe un enlace físico entre AS2 y AS3 (mostrado mediante una línea de puntos en la figura). Simule la ejecución del comando traceroute desde un host situado en la subred B a uno en la subred A. Debe justificar e indicar para cada salto la métrica del protocolo de routing involucrado en la decisión. Puede utilizar una tabla con el formato: #HOP - IP - Decisión.

Solución:

Subred A: 100 ifs + red + broadcast + 26 de crecimiento = 128. Por lo tanto voy a necesitar un prefijo /25 que me permite tener 128 direcciones. Uso el prefijo 121.0.0.0/25.

Subred B: 192 ifs + red + broadcast = 194. Por lo tanto voy a necesitar un prefijo /24 que me permite tener 256 direcciones. Uso el prefijo 122.0.0.0/24.

Subred C: 31 ifs + red + broadcast = 33. Por lo tanto voy a necesitar un prefijo /26 que me permite tener 64 direcciones. Uso el prefijo 123.0.0.0/26.

Para las tres subredes intra-routers en el AS1 se debe particionar el prefijo 10.0.0.0/8. Cada una necesita 2 ip's para las interfaces del router + red + broadcast. Por lo tanto voy a necesitar un /30 para cada una:

R1-R2: 10.0.0.0/30
R1-R3: 10.0.0.4/30
R2-R3: 10.0.0.8/30

Para las inter-AS es el mismo razonamiento anterior:

AS1-AS2: 120.0.0.0/30
AS2-AS3: 120.0.0.4/30

También, opcionalmente, se numeran las cuatro subredes intra-routers en el AS2. Para esto se debe particionar el prefijo 10.0.0.0/8. Cada una necesita 2 ip's para las interfaces del router + red + broadcast. Por lo tanto voy a necesitar un /30 para cada una:

R1-R2: 10.0.0.0/30
R1-R4: 10.0.0.4/30
R2-R3: 10.0.0.8/30
R3-R4: 10.0.0.12/30

b)

Tabla de R1

Prefijo/Máscara	Next-Hop	Interfaz	Costo	Protocolo
121.0.0.0/25	10.0.0.2	eth1	5	OSPF
122.0.0.0/24	120.0.0.1	eth0	-	eBGP – AS-PATH: AS3-AS1
123.0.0.0/26	120.0.0.1	eth0	-	eBGP – AS-PATH: AS1
10.0.0.0/30	DC	eth1	-	Kernel
10.0.0.4/30	DC	eth2	-	Kernel
10.0.0.8/30	10.0.0.2	eth1	2	OSPF
10.0.0.12/30	10.0.0.2	eth1	4	OSPF
120.0.0.0/30	DC	eth0	-	Kernel

Se toma IP(AS1-R1-eth0) como 120.0.0.1 e IP(AS2-R2-eth1) como 10.0.0.2.

Tabla de R4

Prefijo/Máscara	Next-Hop	Interfaz	Costo	Protocolo
121.0.0.0/25	DC	eth0	-	Kernel
122.0.0.0/24	10.0.0.9	eth2	5	iBGP – AS-PATH: AS3-AS1
123.0.0.0/26	10.0.0.9	eth2	5	iBGP – AS-PATH: AS1
10.0.0.0/30	10.0.0.9	eth2	3	OSPF

10.0.0.4/30	DC	eth2	-	Kernel
10.0.0.8/30	10.0.0.9	eth2	1	OSPF
10.0.0.12/30	DC	eth1	-	Kernel

Se toma IP(AS2-R3-eth0) como 10.0.0.9.

c)

#Hop	**Ip**	**Decisión**
1	IP(AS3-R4-eth0)	default
2	IP(AS3-R1-eth2)	BGP gateway + RIP 1 hop
3	IP(AS2-R1-eth3)	BGP AS-PATH AS2-AS3 < AS2-AS1-AS3
4	IP(AS2-R2-eth1)	OSPF 2+2+1 < 8
5	IP(AS2-R3-eth1)	OSPF 2+1 < 10
6	IP(AS2-R4-eth2)	OSPF 1 < 12
7	IP(AS2-host)	DC

Obs: En realidad, las interfaces internas de los routers, numeradas con direcciones IP privadas, quedarían sin resuesta (* * *), ya que los routers de borde deberían filtrar paquetes originados en direcciones IP privadas.