Solución Examen – 18 de febrero de 2019
(ref: serc20190218.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 (8 puntos)

Considere el *Chequeo de Redundancia Cíclica* (CRC) utilizado en la capa de enlace Ethernet.

a)¿Qué función cumple? ¿Cómo está implementado?

b) TCP y UDP contienen un campo Checksum. Fundamente su necesidad dada la existencia de CRC en Ethernet.

Solución:

a) Permite la verificación de la correctitud de los datos recibidos por la red. Su implementación se basa en el envío e un campo adicional que permite chequear que lo recibido, no se ha modificado.
La metodología utilizada se basa en el cálculo del resto de una división de polinomios. Éste resto es enviado como campo adicional, y al recibir los datos se realiza la división y se chequea que el resto coincida.
La certeza de ésta metodología es muy grande, y además se basa en la elección de polinomios que permitan detectar los errores más comunes ocurridos.

b) El checksum de TCP y UDP, se implementa sobre los cabezales y datos (adicionando un pseudo-dato que contiene las IP origen y destino). Como puede verse la información presentada es redundante.
El justificativo puede fundamentarse en que el modelo de capas no tiene certeza de haber utilizado en capa 2 Ethernet. Esto genera que no pueda asegurar que siempre se ha implementado el CRC de los datos, sino simplemente para las conexiones *end to end* realizadas sobre Ethernet.
Además es importante aclarar que el objetivo de los CRC es diferente, dado que el de capa 2 intenta verificar la transmisión entre dos nodos adyacentes, mientras que el de capa 4 entre dos nodos finales.

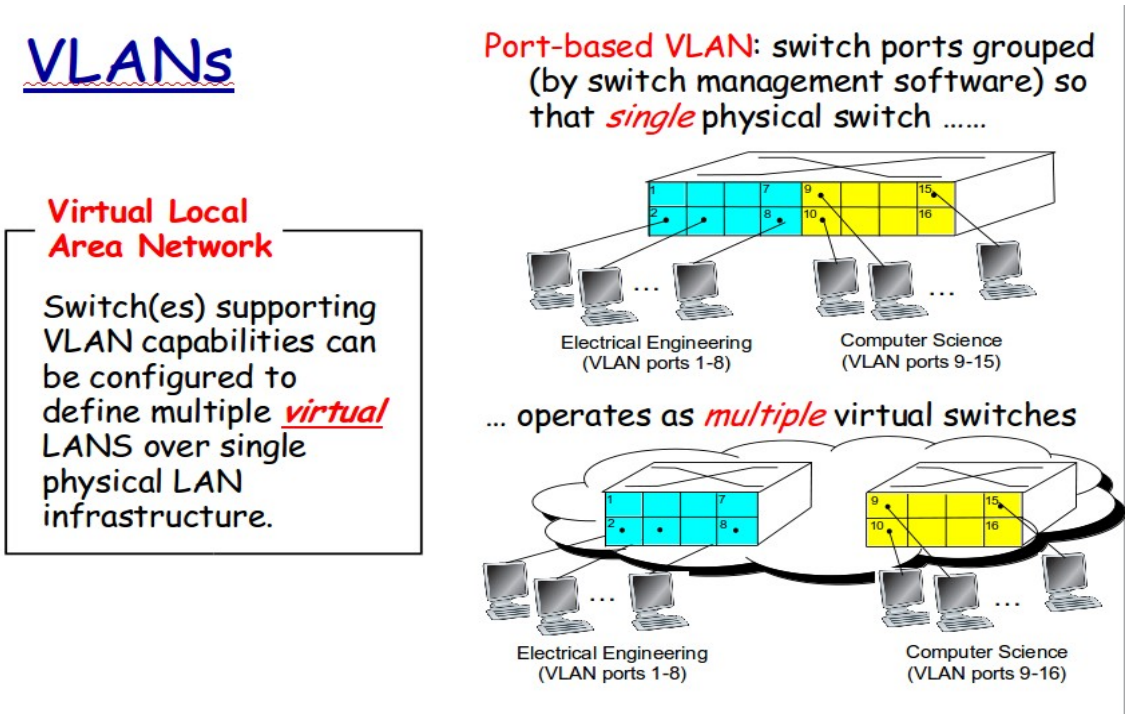
Pregunta 2 (8 puntos)

Defina el concepto de *Virtual Local Area Network* (VLAN). Presente un ejemplo que demuestre su utilidad. ¿Cómo se denomina y cómo funciona la interconexión de VLANs en switches separados?

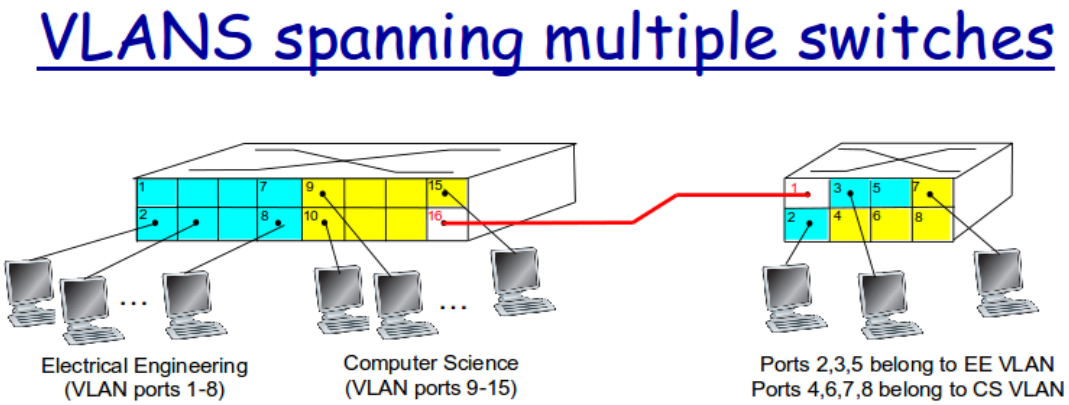
Solución:

Una VLAN, es un método para crear redes lógicas independientes dentro de una misma red física. Varias VLAN pueden coexistir en un único conmutador físico o en una única red física.

De esta forma, se puede disponer de varias VLANs dentro de un mismo switch. Podría decirse que cada una de estas redes agrupa los equipos de un determinado segmento de red. Crear estas particiones tiene unas ventajas bastante claras a la hora de administrar una red.



Para la interconexión de VLANs en dos switches separados se utiliza un enlace TRUNK. Estos enlaces tienen la capacidad de transmitir la VLAN a la que pertenece el frame, agregando en el protocolo 802.1Q un campo para ser especificado. De ésta forma el intercambio de frames contiene la información para saber a que VLAN debe ser transmitido el frame.



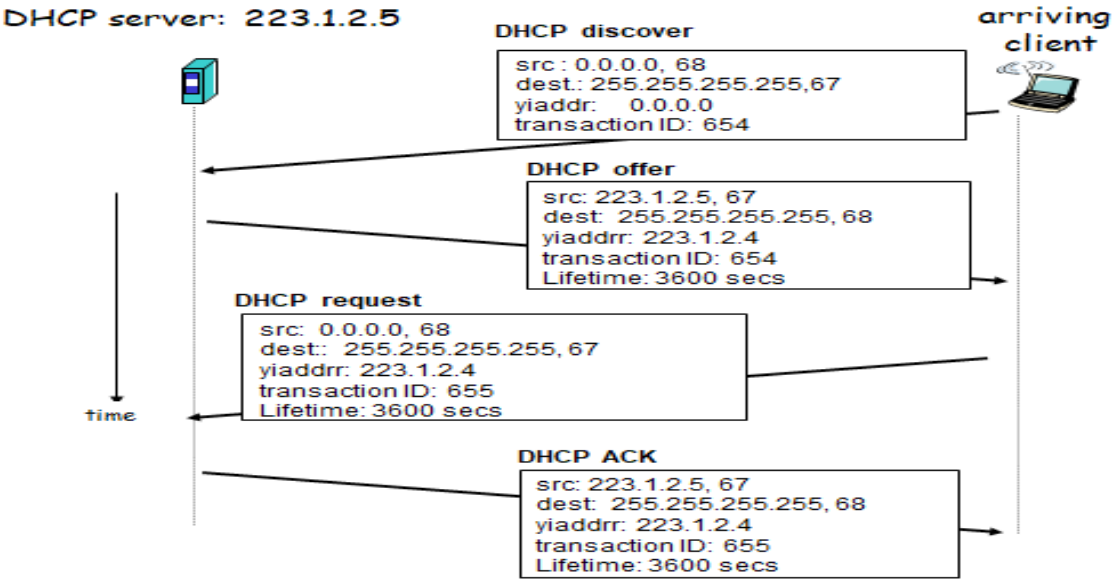
- *trunk port:* carries frames between VLANS defined over multiple physical switches
  - frames forwarded within VLAN between switches can't be vanilla 802.1 frames (must carry VLAN ID info)
  - 802.1q protocol adds/removed additional header fields for frames forwarded between trunk ports

Pregunta 3 (8 puntos)

¿Qué función cumple el protocolo *Dynamic Host Configuration Protocol* (DHCP) en IPv4? Describa en detalle los mensajes y cómo son usados por el protocolo, indicando cabeceras, payload y direcciones de red.

Solución:

El protocolo DHCP, Dinamic Host Configuration Protocol, es una de las formas disponibles para que un nodo conectado a una red IP obtenga una dirección IPv4 desde un servidor DHCP, el que gestiona un pool de direcciones IP a tales efectos.
Las dos opciones para que a un nodo se le asigne una dirección IP son: que se configure estáticamente ("hardcoded") en la propia configuración de red del nodo, o que la obtenga al unirse a la red, por ejemplo, mediante DHCP. Con este protocolo logramos mayor flexibilidad en el uso del rango de direcciones IP disponibles para numerar los nodos de una red, pudiendo reutilizarlas, especialmente en redes donde los nodos no requieren estar siempre conectados y con la misma dirección IP, como sería el caso de los servidores. Adicionalmente, DHCP puede entregar más información, como por ejemplo, nombre y dirección IP de un servidor DNS.



Los mensajes intercambiados son los siguientes:

- El equipo cliente solicita la configuración enviando un mensaje de difusión de tipo "*DHCP DISCOVER*".
- Cada servidor DHCP responde con un mensaje de tipo "*DHCP OFFER*" que contiene una dirección IP y otras opciones de configuración. Este mensaje se envía al cliente por unicast si es posible o por broadcast en otro caso (RFC 1531 - http://tools.ietf.org/html/rfc1531).
- El cliente acepta los parámetros ofrecidos por alguno de los servidores respondiendo con un mensaje de difusión de tipo "*DHCP REQUEST*" indicando en él el identificador de servidor que ha elegido. El mensaje *DHCPREQUEST* debe ser enviado a todos los servidores que recibieron el *DHCPDISCOVER* para que puedan reutilizar la dirección que habían ofrecido. También se usa *DHCPREQUEST* para extender en el tiempo la validez de una dirección IP.
- El servidor elegido envía un mensaje de confirmación de tipo "*DHCP ACK*" indicando que aprueba esa concesión y confirmando la configuración, así como indicando el tiempo máximo en que la dirección IP es válida.

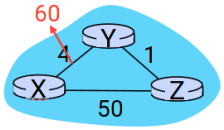
Pregunta 4 (8 puntos)

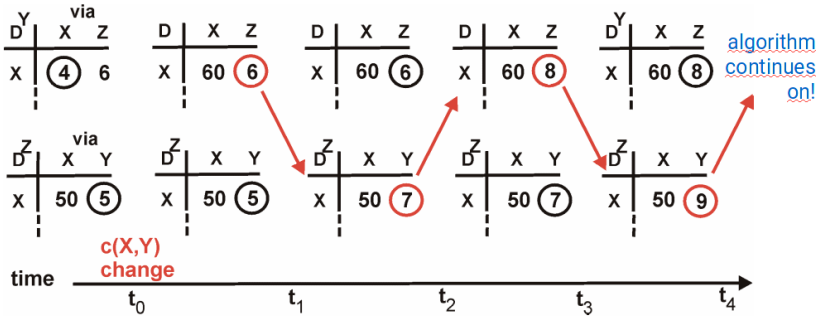
Explique de qué se trata el problema de conteo infinito en los algoritmos de ruteo del tipo *Distance Vector*. Indique que soluciones se plantean. Presente un ejemplo que muestre el problema y posteriormente, su solución.

Solución:

Distance Vector: link cost changes

Link cost changes:
- good news travels fast
- bad news travels slow - "count to infinity" problem!

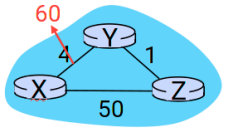


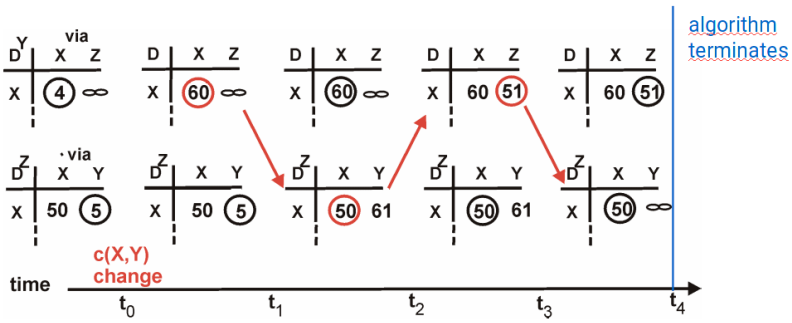


Network Layer 4-27

Distance Vector: poisoned reverse

If Z routes through Y to get to X :
- Z tells Y its (Z's) distance to X is infinite (so Y won't route to X via Z)
- will this completely solve count to infinity problem?





Network Layer 4-28

Pregunta 5 (8 puntos)

a) Presente las diferencias del direccionamiento de IPv6 respecto al de IPv4.
b) Explique como es implementada la funcionalidad de ARP en IPv6. Para la descripción deberá explicar qué protocolos están involucrados y los mensajes intercambiados, indicando cabeceras, payload y direcciones de red.

Solución:

a)Las direcciones de red de IPv6 poseen 128 bits de longitud, mientras que las de IPv4 poseen 32 bits. La notación utilizada en IPv4 codifica los octetos en su representación decimal, los agrupa de a 8 bits y los separa por puntos, mientras que la notación utilizada por IPv6 codifica los octetos en su representación hexadecimal y separa parejas de éstos con el carácter ":" eliminando ceros a la izquierda dentro de dichas parejas, así como bloques completos de ceros representados por "::".
Los tipos de direcciones soportados por IPv4 son Unicast (identifica una única interfaz de red en la red), Multicast (identifica un grupo de interfases en la red) y Broadcast (identifica todas las interfases de una red). En IPv6 se soportan Unicast y Multicast, con la misma semántica; Broadcast se considera un caso particular de Multicast (en el que todas las interfases de cierta red son miembros) y se agrega Anycast, donde se identifica una interfaz (cualquiera) de entre un grupo de éstas.

b) El protocolo que implementa una funcionalidad equivalente al ARP en IPv6 es implementado como parte del Neighbor Discovery Protocol (ND), que provee servicios adicionales a la simple resolución del mapeo IP→MAC. El equivalente a ARP es implementado solamente con el intercambio de mensajes ND Solicitation y ND Response, equivalentes a ARP request y ARP Response.
Cada nodo en la red (ya sea un host o un router) presenta una tabla de vecinos (neighbors). Ésta tabla contiene registros con los siguientes campos: IP address, MAC address; interface y TTL (Time to live), para los nodos de su LAN.

< IP address; MAC address; Interface; TTL>

El TTL indica después de que tiempo es eliminado el registro. La Interface es necesaria para las direcciones de tipo link-local, que al tener todas el mismo prefijo en todas las interfases, necesista de ésta para desambiguar la tabla.

Ejemplo de funcionamiento:

- Cuando el equipo A desea enviar un datagrama al equipo B, y la dirección MAC de B no se encuentra en la tabla de vecinos, debe realizar los siguientes pasos
- Envía un mensaje ICMPv6 de tipo Neighbor Solicitation, que en la cabecera IPv6 lleva las direcciones: propia en el origen y en el campo destino, la dirección de múlticast de todos los nodos del segmento ff02::1 (siendo precisos, ff02::1:ff concatenado con los 3 últimos bytes de la dirección que estamos averiguando). A nivel de la cabecera Ethernet, se coloca como origen la direccón MAC asociada a la dirección por donde se envía el paquete y la dirección MAC correspondiente a la direccíon de multicast, (siendo precisos, la MAC 33:33?: concatenado con los 3 últimos bytes de la IPv6 que estamos averiguando). En el payload del mensaje ICMPv6 se coloca la dirección IPv6 que estamos averiguado, así como, la MAC origen. Este mensaje es recibido por todos los nodos de la LAN.
- Cuando B recibe el paquete Neighbor Solicitation, lo contesta con un Neighbor Advertisement, que en la cabecera IPv6 lleva las direciones IPv6 de ambos equipos, en la cabecera Ethernet van las MAC correspondientes de ambos. El Payload IPv6 es de tipo ICMPv6, indicando la TargetAddress igual a la IPv6 Origen. Este mensaje es enviado en forma unicast

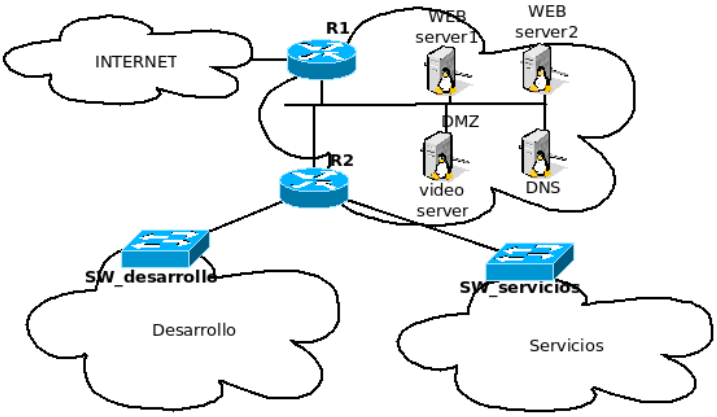
El equipo A graba el par IP.MAC en la tabla ARP hasta que pase el TTL

Problemas Prácticos

Problema 1 (30 puntos)

Una empresa que desarrolla y provee servicios web para Internet, tiene contratada su conexión a Internet a través de un proveedor, quien suministra el router (R1) y le asignó la red pública 190.1.1.0/29 para su uso.

La empresa cuenta con tres subredes: *DMZ*, donde se encuentran los equipos que brindan servicios al público en Internet, *Servicios* donde se alojan los recursos que permiten resolver servicios brindados en la DMZ y *Desarrollo*, subred utilizada para el desarrollo de los servicios brindados.



La red debe cumplir las siguientes condiciones:

- *Servicios* tiene 20 equipos (contiene los equipos que proveen funcionalidades a los servicios web brindados),
- *Desarrollo* tiene 63 puestos de trabajo, debe tener acceso a Internet, pero no deben ser accesibles desde el mundo
- *DMZ* con 4 equipos (2 servidores web, un equipo servidor de video y un equipo para DNS). Estos equipos deben tener conectividad con la red de *Servicios*.

Se pide:

a) Asigne direcciones IP para todos las redes de la empresa, utilizando direcciones IP privadas de ser necesario y asignando los menores prefijos disponibles a cada red.

b) Indique las tablas de forwarding de los routers R1 y R2, el equipo *web_server1*, un equipo de la red *Servicios* y uno de *Desarrollo*, cumpliendo las condiciones presentadas en la letra.

c) Especifique los paquetes que se transmiten en la red al realizar un *ping* desde un equipo de la red de *Desarrollo* a la IP del equipo *WEB_server1*. Enumere mensajes, direcciones y payload de capa 2, 3 y 4. Considere que no existe información de ningún tipo cacheada al ejecutar el comando *ping*.

Solución:

a)
LAN pública disponible 190.1.1.0/29, que equivale a 6 direcciones IP
R1: 190.1.1.1
DNS: 190.1.1.2
web_server1: 190.1.1.3
web_server2: 190.1.1.4
video_server: 190.1.1.5
R2: 190.1.1.6

LAN Servicio: 20 hosts.
Se necesitan además: 1 dir IP para el R2, dirección de red y broadcast, 23 direcciones IP en total → prefijo /27 ($2^5 = 32$).

LAN Desarrollo: 63 hosts.
Se necesitan además: 1 dir IP para el R2, dirección de red y broadcast, 66 direcciones IP en total → prefijo /25 ($2^7 = 128$).

Dado que las subredes Servicios y Desarrollo se comunican con "el exterior", se propone utilizar redes privadas y dar conectividad mediante NAT.

LAN Servicios: 192.168.1.0/27
Se propone la IP 192.168.1.1 para el router R2 (eth1)

LAN Desarrollo: 192.168.2.0/25
Se propone la IP 192.168.2.1 para el router R2 (eth2)

b)
R1:
se supone eth0 conectada a Internet y eth1 a la red DMZ.

Prefijo de Destino	Interfaz de salida	Next-Hop
190.1.1.0/29	eth1	Directamente conectado
0.0.0.0/0	eth0	IP_Proveedor

R2:
se supone eth0 conectada a DMZ y eth1 a la red Sevicios y eth2 conectada a Desarrollo.

Prefijo de Destino	Interfaz de salida	Next-Hop
190.1.1.0/29	eth0	Directamente conectado
0.0.0.0/0	eth0	190.1.1.1
192.168.1.0/27	eth1	Directamente conectado
192.168.2.0/25	eth2	Directamente conectado

Para el tráfico con destino las redes públicas, cursado en las interfaces eth1 y eth2 se implementa NAT, para permitir la salida a Internet. La dirección IP pública de salida para ambas redes es 190.1.1.6.

WEB_server1:

Prefijo de Destino	Interfaz de salida	Next-Hop
190.1.1.0/29	eth0	Directamente conectado
0.0.0.0/0	eth0	190.1.1.1
192.168.1.0/27	eth0	190.1.1.6

El equipo WEB_server1 debe tener conectividad con la red se Servicios. No se pide conectividad con la de desarrollo.

equipo_red_servicios:

Prefijo de Destino	Interfaz de salida	Next-Hop
192.168.1.0/27	eth0	Directamente conectado
0.0.0.0/0	eth0	192.168.1.1

equipo_red_desarrollo:

Prefijo de Destino	Interfaz de salida	Next-Hop
192.168.2.0/25	eth0	Directamente conectado
0.0.0.0/0	eth0	192.168.2.1

c)

No se presenta información de capa 4, dado que en el envío de un ping (ICMP echo request y reply) no interviene capa 4.
Es importante aclarar que como WEB_server1 no tiene ruta a desarrollo para que el ping funcione se debe realizar a través del NAT, de lo contrario nunca llegará la respuesta.

Tipo de mensaje	**Cabezal capa 2**	**Cabezal capa 3**	**payload**
ARP_request (realizados por equipo en desarrollo)	MAC_orig: MAC_des MAC_dest: ff:ff:ff:ff:ff:ff	-	sender_mac: MAC_des sender_IP: 192.168.2.X target_mac: 00:00:00:00:00:00 target_IP: 192.168.2.1
ARP_response (realizados por equipo R2)	MAC_orig: MAC_R2_eth2 MAC_dest: MAC_des	-	sender_mac: MAC_R2_eth2 sender_IP: 192.168.2.1 target_mac: MAC_des target_IP: 192.168.2.X
ICMP echo request	MAC_orig: MAC_des MAC_dest: MAC_R2_eth2	IP origen: 192.168.2.X IP destino: 190.1.1.3	Echo request
ARP_request (realizados por equipo R2)	MAC_orig: MAC_R2_eth0 MAC_dest: ff:ff:ff:ff:ff:ff	-	sender_mac: MAC_R2_eth0 sender_IP: 190.1.1.6 target_mac: 00:00:00:00:00:00 target_IP: 190.1.1.3
ARP_response (realizados por equipo WEB_server1)	MAC_orig: MAC_WSer1 MAC_dest: MAC_R2_eth0	-	sender_mac: MAC_WSer1 sender_IP: 190.1.1.3 target_mac: MAC_R2_eth0 target_IP: 190.1.1.6
ICMP echo request	MAC_orig: MAC_R2_eth0 MAC_dest: MAC_WSer1	IP origen: 190.1.1.6 IP destino: 190.1.1.3 (cambia IP por NAT)	Echo request
ICMP echo response	MAC_orig: MAC_WSer1 MAC_dest: MAC_R2_eth0	IP origen: 190.1.1.3 IP destino: 190.1.1.6 (cambia IP por NAT)	Echo response
ICMP echo response	MAC_orig: MAC_R2_eth2 MAC_dest: MAC_des	IP origen: 190.1.1.3 IP destino: 192.168.2.X	Echo response

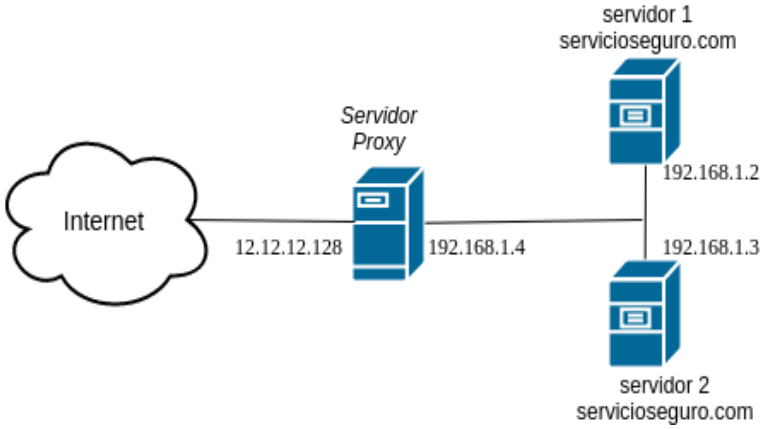
No se ha incluído la búsqueda de la IP en el DNS, elemento que debería realizarse previo al envío del *ping* para determinar la IP destino 190.1.1.3.

Problema 2 (30 puntos)

Sea la red mostrada en la figura, donde un equipo *servidor proxy reverso* recibe conexiones web desde Internet (puerto 80) y conecta con equipos internos a la red, configurados con direcciones IP privadas. En Internet, el DNS devuelve la IP pública del *proxy* (12.12.12.128) para el dominio www.servicioseguro.com.

El *servidor proxy reverso* debe recibir y procesar el acceso al recurso mencionado balanceando la carga entre los dos servidores web internos, redirigiendo los pedidos al servidor que corresponda. La estrategia de balanceo será *round-robin*.

El *servidor proxy reverso* además debe corroborar que los clientes no estén dentro de la lista de direcciones IPs "sospechosas", mitigando así el riesgo que los servidores puedan ser atacados.



Se pide:

Implemente en un lenguaje de alto nivel, utilizando las primitivas de la API de *sockets*, el programa que ejecuta el *servidor proxy reverso*. El *servidor proxy reverso* deberá ser capaz de atender múltiples clientes simultáneamente balanceando la carga entre los dos servidores de la LAN y denegar el acceso a clientes que intenten conectarse desde una dirección que se encuentre en la "lista negra". Esta lista puede asumirse conocida y consultarla con la operación `ip_prohibida(<IP_ADDRESS>)` que devuelve un booleano indicando si la IP pasada por parámetro está en la misma o no.

Solución:

```
a)
#define IP_1 = 192.168.1.2
#define IP_2 = 192.168.1.3

var next_server = 0 //global
mutex mtx

void secureProxy(){
    int master = socket.tcp();
    bind (master, 12.12.12.128, 80);
    serverSock = master.listen();
    while (true){
        clientSock, err = accept(serverSock); //espero conexiones
        thread.new (atenderCliente, clientSock); //nuevo hilo para el cliente
    }
    close (serverSock);
}
```

```
void atenderCliente (clientSock){
    ip, port = clientSock.getpeer();
    if (ip_prohibida(ip)){ //Se realiza en el hilo para no demorar a otros clientes.
        close (clientSock);
        return;
    }
    master = socket.tcp();
    // next_server se lee y escribe en todos los hilos que pasen el control de la ip
    mtx.lock()
    turno = next_server
    next_server = 1-next_server
    mtx.unlock()

    if ( turno == 0){
        proxyclientSock = master.connect(IP_1, 80);
    }
    else{
        proxyclientSock = master.connect(IP_2, 80);
    }

    //hilos para enviar datos y esperar respuesta de los servidores y del cliente
    thread.new (pushData, clientSock, proxyclientSock);
    thread.new (pushData, proxyclientSock, clientSock);
}

void pushData(Socket src, Socket dst){
    String data;
    while (true){
        data, err = src.receive();
        if (data, err == nil, 'closed')
            break;

        remain, err = dst.send(data)
        if (err == 'closed')
            break;
    }
    close (src);
    close (dst);
}
```