Redes de Computadoras

Segundo Parcial – 27 de noviembre de 2015

(ref: solprc20151127.odt)

Instrucciones

- **Indique su nombre completo y número de cédula en cada hoja.**
- **Numere todas las hojas e indique en la primera la cantidad total de hojas que entrega.**
- **Escriba las hojas de un solo lado y utilice una caligrafía claramente legible.**
- **Comience cada pregunta en una hoja nueva.**
- **Sólo se responderán dudas de letra. No se responderán dudas de ningún tipo durante los últimos 30 minutos de la prueba.**
- **La prueba es individual y sin material. Apague su teléfono celular mientras esté en el salón de la prueba.**
- **Duración: 2 horas. Culminadas las 2 horas, el alumno no podrá modificar de ninguna forma las hojas.**
- **Justifique todas sus respuestas.**

Pregunta 1 (4 puntos)

1. Describa el problema de conteo infinito, utilizando un ejemplo sencillo.
2. ¿En qué tipo de protocolo se presenta este problema?

Solución

Los algoritmos del tipo Bellman-Ford presentan problemas del tipo conteo a infinito (count-to-infinity). La causa central del problema parten de que A le dice a B que hay algún camino a cierto destino, no hay forma que B sepa que B es parte de dicho camino. Se puede ejemplificar con una red conectada como: A-B-C-D…..., y la métrica a utilizar sea el número de hops. Supongamos que se desconecta A. B no recibe actualización desde A, pero si recibe una actualización desde C, quién no sabe que A se bajó. C piensa que está a 2 saltos de distancia desde C (C -> B -> A), lo que es falso. Como B no sabe que es camino es a través de si mismo, actualiza su tabla con el nuevo valor: "B -> A = 2 + 1". Luego, B actualiza su información a C, y dado que C es alcanzable a través de B (desde el punto de vista de C), C decide actualizar su tabla a: "C -> A = 3 + 1". Esto se propaga lentamente a través de la red hasta llegar a infinito.

Los protocolos que son suceptibles a este problema son aquellos del tipo de Bellman-Ford o Distance-vector.

Pregunta 2 (3 puntos)

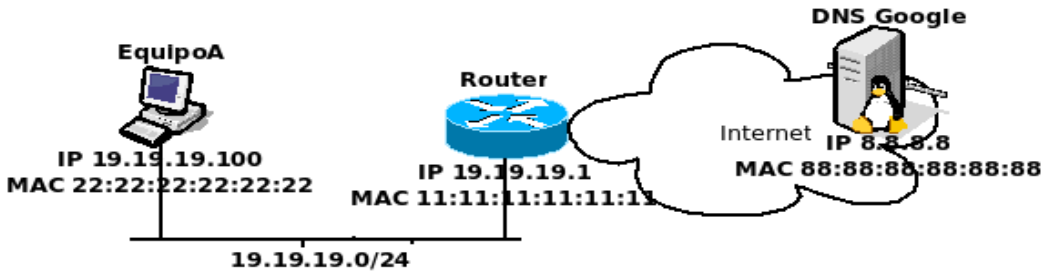
Describa los atributos AS_PATH y LOCAL_PREF (preferencia local) de BGP, explicando que rol tienen en el algoritmo de selección de ruta.

Solución

Cuando existe más de una ruta para un mismo prefijo de red, BGP aplica secuencialmente reglas de selección hasta quedarse con una ruta. En primer lugar se aplica el atributo LOCAL_PREF, que es configurado por el administrador del AS, típicamente en los routers de borde, y toma precedencia la ruta con mayor valor de LOCAL_PREF. Si luego de este primer paso queda más de una ruta, se aplica el atributo AS_PATH, que es la lista de Sistemas Autónomos que se deben atravesar para llegar al AS que contiene el prefijo, y en este sentido se le denomina también "path vector", por analogía a "distance vector", ya que da una noción de distancia en "saltos AS".

Pregunta 3 (5 puntos)

El host *EquipoA* que se encuentra en la LAN 19.19.19.0/24 como muestra la figura, y su conectividad con Internet es a través del router.



El equipo desea realizar una consulta DNS al servidor *DNS de Google*, ubicado en Internet. Para ésto debe enviar un datagrama UDP al puerto 53 de dicho servidor (IP:8.8.8.8).

Se pide:

a) Indique la dirección IP origen y destino del datagrama IP utilizado para realizar la consulta.

b) Suponiendo que la tabla ARP de EquipoA se encuentra vacía, indique el intercambio de tramas ARP necesario para enviar el datagrama de consulta DNS incluyendo el contenido de los diferentes campos del mismo. Asuma que el formato de la trama ARP es: `[MAC Origen, IP Origen, MAC Destino, IP Destino, tipo]`.

Solución

a) La consulta al DNS de Google, se realiza con un paquete IP con UDP con dirección IP origen 19.19.19.100 y destino 8.8.8.8.

b) Para su envío, el host equipoA analiza primero si la IP destino 8.8.8.8 se encuentra en su misma LAN (19.19.19.0/24), lo cual no se cumple. Por lo tanto, el datagrama deberá enviarse al default gateway que lo comunica con Internet, el router 19.19.19.1.

Luego, el equipoA realizará una solicitud ARP para averiguar la dirección Ethernet del default gateway de la red con IP 19.19.19.1.

Los campos del frame ARP de *request* son los siguientes:

Tipo: ARP_request (1).
MAC Origen : Dirección ethernet del origen (22:22:22:22:22:22)
IP Origen: Dirección Ipv4 del origen (19.19.19.100)
MAC Destino: Dirección ethernet del destino, que es desconocida y se desea averiguar, por lo que se coloca 0 (00:00:00:00:00:00)
IP Destino: Dirección Ipv4 del nodo buscado (19.19.19.1)

Nota: este mensaje ARP está encapsulado en una trama ethernet cuya dirección MAC de origen es 22:22:22:22:22:22, y de destino FF:FF:FF:FF:FF:FF.

Este ARP request genera una respuesta del router, indicando su dirección ethernet:

Los campos del frame ARP de reply son los siguientes:

Tipo: ARP_reply (2).
MAC Origen : Dirección ethernet del origen (11:11:11:11:11:11).
IP Origen: Dirección Ipv4 del origen (19.19.19.1).
MAC Destino: Dirección ethernet del destino (22:22:22:22:22:22).
IP Destino: Dirección Ipv4 del nodo destino (19.19.19.100).

Nota: este mensaje ARP está encapsulado en una trama ethernet cuya dirección MAC de origen es 11:11:11:11:11:11, y de destino 22:22:22:22:22:22.

Nota adicional: no se conoce la topología ni las tecnologías de enlaces utilizados fuera de la LAN, y por lo tanto no es posible conocer que pasa a nivel de capa 2 fuera de la LAN. En particular, es incorrecto enviar solicitudes de ARP al DNS.

Problema 1 (10 puntos)

Implemente la función void `doNAT(ip_packet & pkt)`, que realiza la traducción de direcciones en un router doméstico.

Asuma que la función es ejecutada únicamente cuando debe realizarse la traducción de direcciones, que el único payload que pueden contener los datagramas son segmentos TCP o UDP, que no debe ocuparse de borrar entradas de la tabla, y que para la red local se utiliza el prefijo 10.10.0.0/24

Defina la estructura de la tabla NAT, las operaciones necesarias para acceder o agregar entradas en ella, y las que necesite para operar sobre los cabezales de un datagrama ip o su payload.

Solución

```
typedef int16_t port_number;
typedef int32_t ip_addr;

typedef struct {
	ip_addr lanside_ip;
	port_number lanside_port;
	ip_addr wanside_ip;
	ip_addr wanside_port;
} natEntry;

natEntry* natTable;

void doNAT(ip_packet &pkt) {
	ip_addr srcIP = getSrcIpAddr(pkt);
	if (is_local(srcIP)) {
		port_number srcPort = getPayloadSrcPort(pkt);
		port_number wanPort = getWANSidePort(natTable, srcIP, srcPort);
		if (wanPort == null) { //No hay entrada en la tabla NAT, genero una
			wanPort = getAvailableWANPort(natTable);
			addNATEntry(natTable, srcIP, srcPort, wanPort);
		}
		setPayloadSrcPort(pkt, wanPort);
		calculateAndSetPayloadChecksum(pkt, getProtocolField(pkt));
		setWANSideSrcIP(pkt);
		calculateAndSetChecksum(pkt);
	} else {	//IP origen remota
		port_number wanPort = getPayloadSrcPort(pkt);
		port_number dstPort = getLANSideOrigPort(natTable, wanPort);
		if (dstPort != null) {
			ip_addr dstIP = getLANSideIP(natTable, wanPort);
			setPayloadDstPort(pkt, dstPort);
			calculateAndSetPayloadChecksum(pkt, getProtocolField(pkt));
			setDstIP(pkt, dstIP);
```

```
                    calculateAndSetChecksum(pkt);
                }
        }
}

bool is_local(ip_addr) {
        return (ip_addr & 0xFFFFFF00) ^ 0x0A0A0000 == 0
}
```

Funciones auxiliares utilizadas:

`getPayloadSrcPort(ip_packet)`	Obtiene el número de puerto origen en el segmento TCP o UDP
`getWANSidePort(natEntry*, ip_addr, port_number)`	Busca y devuelve el número de puerto, o null si no hay entrada en la tabla
`getAvailableWANPort(natEntry*)`	Devuelve un número de puerto que no esté utilizado en la tabla
`addNATEntry(natEntry*, ip_addr, port_number, port_number)`	Agrega una entrada a la tabla NAT. Asumo que la función conoce la IP de la interfaz externa del router
`setPayloadSrcPort(ip_packet&, port_number)`	Setea el puerto origen en el cabezal TCP o UDP
`calculateAndSetPayloadChecksum(ip_packet&, char)`	Calcula y setea el checksum en el cabezal UDP o TCP, dependiendo del campo de protocolo
`getProtocolField(ip_packet)`	Devuelve el campo de protocolo del cabezal IP
`setWANSideSrcIP(ip_packet&)`	Coloca el número de IP de la interfaz externa del router como origen en el cabezal IP
`calculateAndSetChecksum(ip_packet&)`	Calcula y setea el checksum del cabezal IP
`getLANSideOrigPort(natEntry*, port_number)`	Obtiene el puerto origen antes de traducir, o null si no encuentra una entrada
`getLANSideIP(natEntry*, port_number)`	Obtiene IP original del paquete almacenada en la entrada de la tabla NAT correspondiente
`setPayloadDstPort(ip_packet&, port_number)`	Setea puerto destino en cabezal TCP o UDP
`setDstIP(ip_packet&, ip_addr)`	Setea IP destino