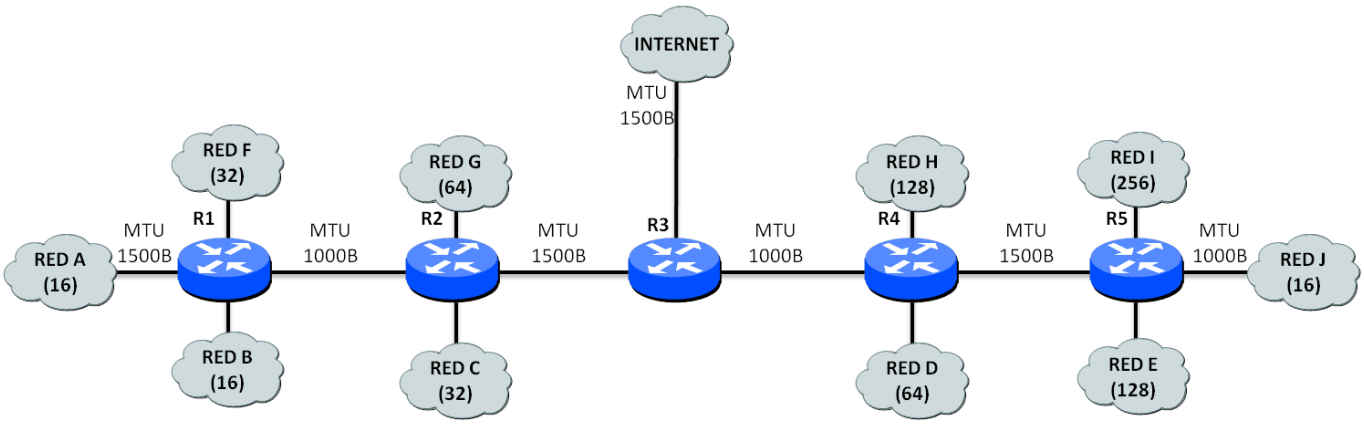
TRANSMISIÓN DE DATOS Y REDES DE COMPUTADORES II

Examen de Teoría[1]

Septiembre de 2012

APELLIDOS, NOMBRE: **GRUPO:**

1. (*2.5 puntos*) La red de una empresa presenta la siguiente topología:



El número de PCs conectados a estas redes está indicado entre paréntesis en el dibujo. La empresa tiene contratada con el ISP el rango de direcciones 32.32.32.0/22. Además, el *router* que da acceso a Internet (R3) tiene asignada la dirección 11.11.11.11/24.

a) Asigne las direcciones IP contratadas y máscaras a cada una de las redes presentes, a los PCs y a las interfaces de los *routers* tal que se minimicen las tablas de encaminamiento. ¿Es suficiente el rango de direcciones IP contratado? En caso contrario, indique por qué.

b) Escriba las tablas de encaminamiento de los *routers* R1 a R5, de manera que todos los equipos queden interconectados y puedan comunicarse hacia Internet.

c) Suponga que un equipo en la red A quiere mandar un *ping* a un equipo de la red J, enviando 1480 bytes en la petición (sin incluir la cabecera de nivel IP). Suponga que la respuesta tiene el mismo tamaño. Explique cómo funciona el comando *ping* (protocolos, mensajes, etcétera). Indique los ***paquetes*** IP que se envían por la red, desde que se generan por el emisor hasta que se reciben por el receptor, indicando el tamaño tanto de la cabecera IP como de los datos IP.

2. (*2.5 puntos*) Ahora, siguiendo con la topología del ejercicio 1, un equipo de la red B quiere descargarse un fichero de 14.600 bytes desde un servidor en la red I usando FTP.

a) Indique los comandos y respuestas del protocolo FTP utilizados para iniciar la descarga del fichero, y qué puertos se utilizarán en cada mensaje.

b) Suponga que el tiempo de transmisión y que la probabilidad de error en los paquetes son despreciables. El tiempo de procesamiento de cada paquete IP en los *routers* es 5 ms, más 1 ms por cada fragmentación realizada. Suponga que no se producen *timeouts* a nivel TCP y que el umbral para comenzar la fase de prevención de congestión es suficientemente elevado. ¿Cuánto tiempo tarda el cliente FTP en descargarse el fichero?

c) Suponga ahora que utiliza protocolo UDP en lugar de TCP. ¿Cuánto tiempo tardaría ahora en realizarse dicha transmisión?

3. (*2 puntos*) Un protocolo de reto-respuesta…

3.1. ¿Qué es y para qué sirve?

3.2. Suponiendo la existencia de una clave secreta compartida ponga un ejemplo de mensajes intercambiados.

3.3. Identifique sus posibles debilidades.

3.4. ¿Sería posible realizarlo si dispusiera de certificados digitales? En su caso ¿cómo?

[1] Esta prueba supone el 70% de la calificación final de la asignatura.

Ejercicio 1

Rango de direcciones IP contratado: 32.32.32.0/22 → 1024 direcciones, de 32.32.32.0 hasta 32.32.35.255

a) Asignación de direcciones IP a PCs y routers

Para que resulte más sencillo, comenzamos por las redes con más equipos.

Red I → 256 PCs + 1 router + dir. red + dir. difusión = 259 direcciones → máscara /23 → 512 direcciones → red 32.32.32.0/23 → desde 32.32.32.0 (dir. subred) hasta 32.32.33.255 (dir. difusión)

Red E → 128 PCs + 1 router + dir. red + dir. difusión = 131 direcciones → máscara /24 → 256 direcciones → red 32.32.34.0/24 → desde 32.32.34.0 (dir. subred) hasta 32.32.34.255 (dir. difusión)

Red H → idem, red 32.32.35.0/24

De esta forma, ya no quedan más direcciones a asignar. Al haber usado las máscaras más restrictivas posibles, esto significa que el rango no es suficiente para las redes que había que direccionar. A pesar de que hay 752 PCs a los que asignar direcciones, no hay suficientes direcciones ya que hay que seguir el direccionamiento jerárquico mediante máscaras.

Como no dan más direcciones públicas, para poder continuar con el ejercicio se seguirá utilizando direcciones IP privadas. La siguiente es una posible asignación, habiendo otras también válidas:

Red D → 64 + 1 + 1 + 1 = 67 direcciones → máscara /25 → 128 direcciones → red 192.168.0.0/25

Red G → idem → máscara /25 → red 192.168.0.128/25

Red C → 32 + 1 + 1 + 1 = 35 → máscara /26 → red 192.168.1.0/26

Red F → idem → máscara /26 → red 192.168.1.64/26

Red B → 16 + 1 + 1 + 1 = 19 → máscara /27 → red 192.168.1.128/27

Red A → idem → máscara /27 → red 192.168.1.160/27

Red J → idem → máscara /27 → red 192.168.1.192/27

El interfaz de cada router conectado a estas redes podría tener la primera dirección IP del rango comentado. Por ejemplo, el router R5 podría tener la dirección 192.168.1.193 en el interfaz hacia la red J.

Faltan por asignar las redes existentes entre los routers. Estas redes tienen 1 dirección del router izquierdo, 1 dirección del router derecho, una dirección de subred y una dirección de difusión. Es decir, en total 4 direcciones → máscara /30. Una posible asignación, continuando por el rango anterior sería la siguiente:

Red R1-R2: 192.168.1.224/30 → e.g. 192.168.1.225 para R1, 192.168.1.226 para R2

Red R2-R3: 192.168.1.228/30

Red R3-R4: 192.168.1.232/30

Red R3-R4: 192.168.1.236/30

Red R4-R5: 192.168.1.240/30

El router R3 ya se dice en el enunciado que tiene la dirección IP 11.11.11.11/24 en el interfaz hacia Internet.

Concluyendo, para utilizar direcciones IP públicas tendrían que haber dado adicionalmente una red pública con máscara /23 (512 direcciones, para englobar 192.168.0.0/23 en la asignación anterior). Por ejemplo, podría haber sido el rango consecutivo al anterior 32.32.36.0/23.

b) Tablas de encaminamiento de los routers R1 a R5

Cada router ha de incluir siempre las redes a las que está directamente conectado. Por ejemplo, el router R1 tendrá una entrada con conexión directa a las redes 192.168.1.160/27 (red A), 192.168.1.128/27 (red B), 192.168.1.64/26 (red F) y a 192.168.1.224/30 (red con el router R2). Esto hay que escribirlo para todos los routers, pero aquí se comenta para el primero para no extendernos.

Además, hay que añadir las entradas necesarias para llegar al resto de redes y hacia Internet. Una posibilidad es poner una ruta hacia cada red (agrupando si se puede, aunque no es necesario porque no se pide en el enunciado) y una ruta por defecto hacia el router R3 que es la pasarela hacia Internet. Hay que tener en cuenta que, dependiendo del router, muchas rutas pasan por el mismo router, por lo que se podrán agrupar. Por ejemplo, para llegar a las redes C, D, E, G, H, I y J desde el router R1 hay que pasar necesariamente por el router R2 → se pueden agrupar. No se pide en el enunciado que se agrupen, pero simplifica bastante la tarea.

Esto quedaría así:

Tabla del router R1

Red destino	Máscara	Sig. salto	Interfaz
conexiones directas (ya comentadas)			
default	0.0.0.0	192.168.1.226 (router R2)	Eth_R1_R2

Tabla del router R2

Red destino	Máscara	Sig. salto	Interfaz
conexiones directas (ya comentadas)			
192.168.1.128 (redes A y B agrupadas)	/26	192.168.1.225 (router R1)	Eth_R2_R1
192.168.1.64 (no agrupable con A y F)	/26	192.168.1.225 (router R1)	Eth_R2_R1
Default	0.0.0.0	192.168.1.230 (router R3)	Eth_R2_R3

Tabla del router R3

Red destino	Máscara	Sig. salto	Interfaz
conexiones directas (ya comentadas)			
32.32.32.0/22 (redes E, I, H agrupadas)	/22	192.168.1.234 (router R4)	Eth_R3_R4
192.168.0.0 (red D)	/25	192.168.1.234 (router R4)	Eth_R3_R4
192.168.1.192 (red J)	/27	192.168.1.234 (router R4)	Eth_R3_R4
192.168.0.128 (red G)	/23	192.168.1.2.229 (router R2)	Eth_R3_R2
192.168.1.128 (redes A y B agrupadas)	/26	192.168.1.2.229 (router R2)	Eth_R3_R2
192.168.1.0 (redes C y F agrupadas)	/25	192.168.1.2.229 (router R2)	Eth_R3_R2
default	0.0.0.0	IP_pasarela_operador	Eth_R3_Internet

Tabla del router R4

Red destino	Máscara	Sig. salto	Interfaz
conexiones directas (ya comentadas)			
32.32.32.0 (red I)	/23	192.168.1.242 (router R5)	Eth_R4_R5
32.32.34.0 (red E)	/24	192.168.1.242 (router R5)	Eth_R4_R5
192.168.1.192 (red J)	/27	192.168.1.242 (router R5)	Eth_R4_R5
default	0.0.0.0	192.168.1.237 (router R3)	Eth_R4_R3

Tabla del router R5

Red destino	Máscara	Sig. salto	Interfaz
conexiones directas (ya comentadas)			
default	0.0.0.0	192.168.1.241 (router R4)	Eth_R5_R4

Seguramente otra asignación de direcciones habría posibilitado agrupar más rutas, especialmente en el router R3. Pero como no se pedía en el enunciado, se ha optado por una asignación lo más sencilla posible para evitar errores innecesarios.

c) Ping desde un equipo en la red A hacia un equipo en la red J

El comando ping manda un mensaje ICMP echo-request, al que el otro extremo responde con un mensaje ICMP echo-reply. Al ser mensajes ICMP, van directamente encapsulados en un paquete IP, sin necesidad de otros niveles como e.g. transporte (por lo que no utilizan "puertos").

De esta forma, a los 1480 bytes hay que añadirles una cabecera IP que típicamente es de 20 bytes, haciendo un total de 1500 bytes. Como la MTU (Maximum Transmission Unit) se refiere al nivel IP, no habrá que realizar fragmentación en redes con una MTU igual o superior a 1500 bytes, pero sí en redes con una MTU inferior.

Veamos qué ocurre con los paquetes enviados:

Equipo en red A → genera un datagrama IP de 1500 bytes (1480 bytes del paquete ICMP y 20 bytes de la cabecera IP), que se puede transmitir en la MTU de 1500 bytes de la red A hasta el router R1 → aquí hay que hacer fragmentación, ya que la MTU entre los routers R1 y R2 es de 1000 bytes:

- Primer fragmento: 1000 bytes en total → 20 bytes para la cabecera, 980 bytes en el campo de datos del paquete IP (de los 1480 bytes iniciales del mensaje ICMP) → quedan 1480 – 980 = 500 bytes de datos para el siguiente fragmento.
- Segundo fragmento: 500 bytes (los que sobran del fragmento anterior) de datos, 20 bytes de cabecera IP → ya no quedan más datos, por lo que no hay más fragmentos.

Estos paquetes IP se transmiten sin problema por el resto de las redes hasta llegar al equipo en la red J, ya que no hay ninguna red entre routers ni la red J que tengan una MTU inferior a 1000 bytes.

Una vez que llega el mensaje ICMP echo-request, el equipo de la red J manda un mensaje ICMP echo-reply. Suponiendo que también tiene el mismo tamaño (1480 bytes), en la red J habría que realizar una fragmentación igual a la ya comentada (primer fragmento con 980+20 bytes, segundo fragmento con 500+20 bytes). Estos fragmentos se transmitirían sin necesidad de más fragmentación a través del resto de redes (R5→R4→R3→R2→R1→equipo en red A) ya que ninguna tiene una MTU inferior a 1000 bytes.

Ejercicio 2

Un equipo de la red B quiere descargarse un fichero de 14600 bytes desde un servidor FTP en la red I.

a) Comandos y respuestas del protocolo FTP utilizados, y puertos.

Puertos 21 (mensajes de control) y 20 (envío de ficheros) si se usa FTP en modo activo. Los mensajes típicos serían USER, PASSWORD, RETRIEVE y otros como CWD, LIST, … y LOGOUT. Véase el tema 4 para una descripción más exacta (que habría que incluir en el examen).

b) Tiempo en transmitir el fichero (considerando TCP y los datos en el enunciado)

El servidor FTP en la red I utilizará paquetes IP de 1500 bytes, ya que es la MTU de esa red. Como la cabecera IP es de 20 bytes y la cabecera TCP también es de 20 bytes, eso significa que cada paquete enviará 1460 bytes (1500 – 20 – 20) de datos del fichero a transmitir. Por lo tanto, habrá 10 datagramas IP (14600 bytes / 1460 bytes de datos por datagrama). En cuanto esos paquetes lleguen al router R4 se fragmentarán de forma similar a lo visto en el ejercicio anterior. Hay que recordar que la fragmentación se hace a nivel IP, por lo que se fragmenta es el segmento TCP de 1480 bytes (1460 bytes de datos y 20 de cabecera). O sea, que la cabecera IP sí se repite (cambiando los campos offset y more fragments) en cada paquete IP, pero la cabecera TCP no se repite (va en la parte de datos del paquete IP). Así:

- Primer fragmento: 980 bytes de datos a nivel IP + cabecera IP de 20 bytes. Los 980 bytes incluyen la cabecera TCP (20 bytes) y 960 bytes de datos del fichero.
- Segundo fragmento: quedan por transmitir 1480 – 980 = 500 bytes del fragmento TCP, más 20 bytes de la cabecera IP.

Estos fragmentos ya no se vuelven a fragmentar durante su transmisión, ya que ninguna red tiene una MTU inferior a 1000 bytes.

Resumiendo: cada datagrama IP sufre una fragmentación en el router R4. Así, el tiempo de ida será 1 ms por esa fragmentación más 5 ms de procesamiento en cada router. Como hay 5 routers entre los equipos, ese tiempo de ida será 5*5+1=26 ms. NOTA: nos han dicho que el retardo por los enlaces es despreciable (velocidad de transmisión muy alta).

Las confirmaciones a nivel TCP (mensajes ACK) sólo incluyen las cabeceras IP y TCP (40 bytes) y no datos (se podría hacer piggybacking, pero el cliente FTP no tiene datos que enviar al servidor). Por tanto, no sufren fragmentación (40 < 1000 bytes). De esta manera, tardarán 5*5 ms = 25 ms en su transmisión entre el cliente y el servidor FTP.

Igual ocurre con los mensajes iniciales para crear la conexión TCP (que tardarían 25 ms + 25 ms + 25 ms = 75 ms), e igual para el cierre de conexión (75 ms). No se incluye en el resto del enunciado para centrarnos en el control de congestión, pero sí habría que añadirlos.

Para ver el tiempo total de la transmisión del fichero hay que considerar el comportamiento del control de congestión en TCP. Para simplificar, como es una red cableada podemos suponer que la probabilidad de error será muy baja. Igualmente, como no tenemos más datos, podemos suponer que la red está descargada y no habrá descarte de paquetes en los routers, por lo que no se perderán ni paquetes ni sus confirmaciones.

Tampoco nos dicen cuál es el tamaño inicial de la ventana de congestión. Podemos suponer 1 ó 2, que son los valores típicos. Ni nos dicen cuál es el tamaño del umbral para pasar a la fase de prevención de congestión. Como se envían pocos paquetes, podemos suponer que no se sale de la fase de inicio lento.

Suponemos que no hay problemas por el control de flujo (receptor con buffer suficientemente grande, con capacidad suficiente para procesar los paquetes y poco cargado en el momento de la transmisión del fichero).

NOTA: cualquier otra suposición razonable sería válida (e.g. suponer que el umbral es de 5 segmentos), aunque complicaría la resolución del ejercicio innecesariamente.

No se tienen en cuenta los mensajes previos (comandos FTP) porque el enunciado dice cuánto se tarda en transmitirse el fichero, no los mensajes anteriores.

<u>Suponiendo que la ventana inicial vale 1 segmento:</u>

t=0	CWD = 1; Servidor → envía 1 segmento que tarda 26 ms en llegar al cliente
t=26+500ms	Como no llega un segundo segmento TCP, el cliente espera 500 ms antes de enviar la confirmación (ACK)
t=526+25ms	Llega la confirmación → CWD = valor antiguo + nº segmentos confirmados en el ACK = 1+1=2 → ahora el servidor manda dos segmentos.
t=551+26+5ms	Llegan los dos segmentos al cliente. Como llegan 2, no hay que esperar 500 ms (véase la generación de ACKs en el tema 3), y se envía un ACK confirmando esos dos segmentos inmediatamente. Los 5 ms son porque el segundo paquete irá por el router anterior, por lo que cuando llegue el primer paquete faltan 5 ms (tiempo de procesamiento en el último router) para que llegue el segundo paquete.
t=583+25ms	Llega el ACK. CWD=2+2=4 → ahora el servidor manda 4 segmentos
t=608+26+3*5ms	Llegan los 4 segmentos al cliente. Éste manda dos ACKs (cada uno confirmando dos segmentos). 3*5 ms es porque los últimos 3 paquetes llegan en 5, 10 y 15 ms después que el primer paquete.
t=649+25+5ms	Llegan los 2 ACKs confirmando 4 segmentos en total → CWD=4+4=8 → el servidor podría mandar 8 segmentos, pero sólo quedan 3 para transmitir el fichero entero. Por lo tanto, manda esos 3 segmentos.
t=679+26+2*5ms	Llegan los tres segmentos. Los dos primeros generan un ACK inmediatamente, el tercero genera un ACK dentro de 500 ms (no llega un segundo segmento que haga que el ACK se envíe inmediatamente). AQUÍ EL CLIENTE HABRÍA RECIBIDO EL FICHERO.
t=715+25ms	Llega el ACK de los dos primeros segmentos del paso anterior → CWD = 8+2 = 10
t=740+500+25 ms	Llega el segundo ACK, que tuvo que esperar 500 ms en el cliente antes de generarse. Éste confirma un único segmento → CWD = 10+1 = 11. FIN DE LA TRANSMISIÓN.

En el caso de que la ventana inicial fuese de dos segmentos, estas esperas de 500 ms antes de generar los ACK no existirían:

t=0	CWD=2 → el servidor envía dos segmentos.
t=26+5ms	Llegan al cliente. Éste envía inmediatamente un ACK confirmando los dos segmentos.
t=31+25ms	Llega el ACK al cliente → CWD=2+2=4 → el servidor envía 4 segmentos
t=56+26+3*5ms	Llegan los segmentos al cliente, que envía 2 ACKs confirmando cada uno 2 segmentos
t=97+25+5 ms	Llegan los ACKs al servidor → CWD=4+2+2=8 → podría enviar hasta 8 segmentos, pero sólo quedan 4 segmentos para terminar el fichero. Envía esos 4 segmentos.
t=127+26+3*5 ms	Llegan los 4 segmentos al cliente. AQUÍ EL CLIENTE HABRÍA RECIBIDO EL FICHERO. El cliente manda inmediatamente 2 ACKs confirmando los 4 segmentos.
t=168+25 ms	Llegan los 2 ACKs al servidor, que aumentaría su ventana → CWD=8+4=12. FIN DE LA TRANSMISIÓN.

Como se observa, empezar con una ventana inicial igual a dos segmentos evita esperas innecesarias de 500 ms por la llegada en solitario de un único segmento. Máxime cuando esto, al principio, se debe al valor de la ventana y no a la falta de datos por transmitir.

c) Transmisión del fichero usando UDP

En el caso de UDP no hay control de congestión (ni de errores, ni de flujo). Así, lo único que se haría sería enviar 10 datagramas UDP de forma consecutiva. Igual que antes habría que fragmentar los paquetes IP.

Así, el tiempo de transmisión (no hay control de congestión) vendría dado por el tiempo de procesamiento y fragmentación de los 10 datagramas IP → como ya se comentó, el 2º paquete llega 5 ms después que el primero (procesamiento en el último router), y así sucesivamente. Así, el tiempo total será el del primer paquete (incluyendo procesamiento y fragmentación → 5*5 + 1 = 26 ms) más el tiempo de procesamiento del resto de paquetes en el último router (9 paquetes * 5 ms = 45 ms). Así, el tiempo total será 26 + 45 = 71 ms.

Lo IMPORTANTE es que en este caso NO se realiza CONTROL DE CONGESTIÓN. Más que el hecho de que haya que sumar el tiempo de los paquetes en el último router (aunque es cierto, pero tiene menos relación con lo visto en la asignatura por lo que se valoraba muy poco). Sí es importante que no hay que esperar a que el receptor reciba un mensaje para enviar el siguiente, entre otras cosas porque no sabemos cuándo se recibe (ni si se recibe; no hay confirmaciones).

NOTA: la cabecera típica UDP es de 8 bytes, por lo que cada datagrama UDP tendría 1480 – 8 = 1472. Así, habría 9 datagramas con 1472 bytes de datos, y un último con 14600 – 9*1472 = 1352 bytes. Pero también tendría que realizar fragmentación, por lo que el tiempo sería el ya comentado.

NOTA: aunque la velocidad de transmisión en este caso es muy superior (entre otras cosas porque se ha supuesto que las redes tienen una velocidad muy alta $\rightarrow\infty$), al no realizarse control de errores, control

de congestión, control de flujo, establecimiento y cierre de conexión… la transmisión no es fiable y puede tener errores.

Ejercicio 3

Un protocolo reto-respuesta…

a) ¿Qué es y para qué sirve?

Es un protocolo sencillo que permite autenticar a uno (o los dos) extremo(s) de la comunicación. Para evitar enviar la clave en texto plano a través de la red, uno de los extremos le manda un reto (número aleatorio) al otro extremo.

El otro extremo calcula una respuesta que es función del reto y de su clave, de forma que esa respuesta es la que se envía. La respuesta se realiza de forma que sea imposible obtener la clave aunque un posible atacante escuchara tanto el reto como la respuesta. Para ello se suelen utilizar funciones *hash* como por ejemplo MD5.

El extremo que envió el reto, que debe compartir la clave secreta con el otro extremo, calculará su respuesta a partir de la clave y del reto que envió. Si la respuesta calculada coincide con la respuesta enviada por el otro extremo, éste queda autenticado.

Para realizar una autenticación de los dos extremos basta con realizar el mismo proceso pero invirtiendo los papeles.

b) Suponiendo una clave secreta compartida, ponga un ejemplo de mensajes intercambiados.

El extremo 1 envía su nombre de usuario en una petición de inicio de autenticación:

Extremo 1 → inicio_autenticacion (usuario) → Extremo 2

El extremo 2 le envía un reto (texto o número aleatorio):

Extremo 1 ← envio_reto (reto) ← Extremo 2

El primer extremo calcula una respuesta, e.g. usando MD5 sobre la cadena "*usuario:clave:reto*" y se la envía al otro extremo:

Extremo 1 → envio_respuesta_reto (MD5("usuario:clave:reto")) → Extremo 2

El segundo extremo calcula a su vez la respuesta, y la compara con la recibida. Si todo está bien, manda un mensaje confirmando que la autenticación ha sido correcta. Si no, manda que ha sido incorrecta.

Extremo 2: calcula *x*=MD5("*usuario:clave:reto*"); recibe *y*=MD5("*usuario:clave:reto*"); compara *x* == *y*. Si son iguales, manda el siguiente mensaje:

Extremo 1 ← autenticación_correcta ← Extremo 2

Si *x* es distinto de *y*, manda el siguiente mensaje:

Extremo 1 ← autenticación_incorrecta ← Extremo 2

c) Identifique sus posibles debilidades

Es susceptible de ataques por repetición: aunque un atacante no conociese la clave, si ve el mismo reto puede conocer la respuesta. Esto se resuelve introduciendo *nonces*, es decir, algo que no se pueda

repetir. Por ejemplo, se podría utilizar un reto que tuviese una parte aleatoria y otra parte que dependiese de la fecha y la hora, de forma que en una fecha/hora posterior no se podría volver a dar.

De forma parecida, si la autenticación es bidireccional, se podría hacer un ataque por reflexión → le pido al otro extremo que me envíe la respuesta usando el mismo reto que él me envió. Esto se puede resolver usando conjuntos diferentes para los retos en cada extremo. También se podría considerar el uso de nonces.

Otro problema o debilidad es que el uso de reto-respuesta requiere que las claves secretas compartidas se hayan intercambiado antes. Esto tiene su complejidad si el número de usuarios es elevado, ya que hay que buscar un procedimiento seguro para el intercambio de claves. Algunos vistos en teoría son el intercambio Diffie-Hellman (con posible ataque de persona en medio), el centro de distribución de claves (KDC), o el uso de claves de sesión gracias a servidores de autenticación (por ejemplo usando el protocolo Kerberos).

d) ¿Sería posible realizarlo si dispusiera de certificados digitales? En su caso, ¿cómo?

Sí sería posible. Un certificado digital tiene la identidad de su dueño y su clave pública, todo ello firmado (cifrado con su clave privada) por una autoridad de certificación fiable (e.g. VeriSign o CERES).

Así, los extremos podrían intercambiarse los certificados. Como se fían de la autoridad de certificación, se fían de su contenido. Si uno cifra la clave secreta, que quiere compartir con el otro, con la clave pública del otro, sólo éste puede descifrarla. Así, ya está resuelto el tema de compartir una clave y estar seguro de que es la clave compartida con el otro extremo.

A partir de aquí, se puede repetir el procedimiento explicado en el apartado b.

Nótese que aunque se puede realizar, no tiene ningún sentido ya que el uso de certificados ya me autentica al otro extremo, e.g. simplemente con que cifre con su clave privada los mensajes que me envía: 1) sólo el pudo cifrarlos ya que nadie más conoce su clave privada y 2) yo puedo descifrarlos porque tengo la clave pública que está en su certificado → queda autenticado. De esta forma, ya no haría falta el intercambio de reto-respuesta ya que estaban previamente autenticados.

Adenda al problema 1

En el apartado 1.a se pide que la asignación debe minimizar las tablas de encaminamiento. Se puede interpretar que este criterio es prioritario frente a hacer una asignación ajustada a la dimensión (número de hosts) en cada red. Si se adopta esta aproximación una asignación para minimizar sería la siguiente:

Como las IP públicas de las que disponemos son sólo 1024 (/22), no se pueden asignar todas las redes con IP públicas. A partir de aquí tenemos libertad para la asignación en el sentido de que se puede hacer todo privado o parte privada y parte pública. Visto desde R3 empecemos por las redes de la derecha: H,I, J E y D.

De mayor a menor como además de los hosts indicados cada red necesita 3 direcciones más, los bits de "host" que necesitamos para cada una de las redes es:

Red I (256+3) → 9 bits, Redes E y H (128+3) → 8 bits, Red D (64+3) → 7 bits, Red J (16+3) → 5 bits

Sean:

X la “red” correspondiente al enlace punto a punto entre los routers R4 y R5

Y la “red” entre R3 y R4. Z la red entre R3 y R2, finalmente T la red entre R1 y R2.

Bien, si lo prioritario es minimizar el número de entradas de encaminamiento, la asignación ha de ser tal que permita agruparlas desde fuera. Por ello empecemos por la más grande desde lo más adentro posible (parte de la derecha) y vamos asignando de tal manera que se vean como una sóla entrada desde el router de “fuera”, por ejemplo las redes I, J, E y X se ven como una sola (/21) des R4

1-7	8-15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
				0	0	0	0	0	RED A								
				0	0	0	0	1		RED J							
				0	0	0	0	1					RED E				
				0	0	0	1	1								RED X	
				0	0	1				RED H							
				0	1	0					RED D						
				0	1	1										RED Y	
				1	0	0	0	0				RED F					
				1	0	0	0	1					RED A				
				1	0	0	1	1					RED B				
				1	0	0	1	0								RED T	
				1	0	1					RED G						
				1	1	0						RED C					
				1	1	1										RED Z	

De esta forma todas las redes de la derecha de R3 necesitan sólo una entrada en su tabla, siempre y cuando se utilice una máscara /19, para la red de la izquierda se agrupan igualmente en una sola entrada /19.

En R4 además de los routing directos ý del routing por defecto a R3 se necesitan sólo 3 entradas:

/21 para la red H, /21 para la red D y /21 para las redes X, I, J, E.

Procedemos igualmente para las redes de la izquierda de R3.

Si se usan las direcciones públicas (hasta 1024) se podrían agrupar por ejemplo A, B, C F y G (las de la izquierda) y asignarlas de forma parecida pero entonces el número de entradas de encaminamiento ya no sería el mínimo. Con 1024 (/22) públicas hay para asignar todas las de la izquda.

1-7	8-15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
								0	0	0	0	RED F					
								0	0	0	1		RED A				
								0	0	1	1		RED B				
								0	0	1	0					RED T	
								0	1		RED G						
								1	0			RED C					
								1	1							RED Z	