Solución del Examen - 2 de febrero de 2009 (ref: sirc0902.doc)

Preguntas Teóricas

Pregunta 1 (9 puntos)

Se desea proteger a nivel de transporte una comunicación *http*, logrando así una comunicación que generalmente se identifica como *https*.

a) Mencione el protocolo descrito durante el curso que puede cumplir con dicho objetivo.
b) Indique qué clave contiene el certificado que el servidor le envía al cliente, y explique para qué se utiliza dicha clave.
c) ¿Qué clave/s se utiliza/n para proteger los datos que viajan en el sentido servidor → cliente?
d) Mencione otros componentes del certificado, y qué elemento adicional se necesita en el cliente para asegurar la identidad del servidor.

Pregunta 2 (10 puntos)

a) En un contexto unicast, describa los procesos de forwarding y enrutamiento, y explique como se relacionan.
b) En un contexto broadcast, describa un proceso de flooding, y explique como se puede transformar en flooding controlado.

Pregunta 3 (5 puntos)

Indique el significado de FDMA y TDMA. Explique el principio de operación de cada uno.

Pregunta 4 (9 puntos)

a) Describa la conmutación de paquetes basada en:
   i. Datagramas
   ii. Circuitos virtuales.
b) Explique ventajas y desventajas de cada opción desde el punto de vista de la asignación de recursos en la red.

Pregunta 5 (7 puntos)

Describa el mecanismo de control de flujo de TCP.

Problemas Prácticos

Problema 1 (30 puntos)

Una solución frecuentemente utilizada para brindar seguridad a servicios legados sobre TCP se basa en *encriptar* su tráfico, sin la necesidad de reprogramarlos. La solución general del problema se basa en utilizar un proceso de tipo adaptador que ejecute un protocolo criptográfico para resolver los aspectos de seguridad y realizar el transporte de los segmentos TCP en forma segura. El servicio legado que se desea ejecutar únicamente recibe conexiones entrantes, y nunca intenta activamente conectarse con un equipo externo.

La forma de publicar los nuevos servicios usualmente se basa en:

- Configurar el servicio legado para que éste atienda en la interfaz *loopback* del equipo en un puerto que denominaremos **`LOOPBACK_PORT`**.
- En el puerto conocido **`SERVICE_PORT`** de la interfaz pública donde habitualmente atendía el servicio legado, comenzará a atender el adaptador criptográfico.
- Cuando un cliente establece una conexión TCP, el adaptador creará una nueva instancia concurrente que atenderá la conexión. Esta instancia establece otra conexión al servicio legado en el **`LOOPBACK_PORT`**.
- Cada una de las instancias concurrentes, cuando reciben un segmento desde el cliente, lo *desencriptan* y transmiten por el *socket* al servidor legado. Si el segmento se recibe desde el servidor, es *encriptado* y transmitido al cliente.

Se pide:

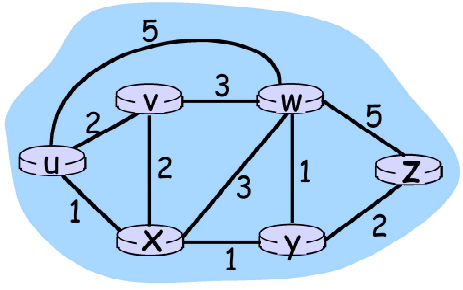
a) Describa los componentes principales de la solución, así como, las consideraciones a realizar para mantener el estado de las conexiones.
b) Implemente, utilizando un lenguaje de alto nivel, el servicio anteriormente descripto.
c) ¿Qué problema tiene esta solución en caso de que el funcionamiento del servicio implique intentar activamente una conexión hacia los clientes?
d) Describa posibles mecanismos para evitar o minimizar los efectos de ataques del tipo SYN FLOODING en esta solución.

Asuma disponible dos funciones: **`encriptar()`** y **`desencriptar()`**, que reciben un segmento TCP y lo *encriptan* o *desencriptan* respectivamente. Dichas funciones hacen el manejo criptográfico de forma transparente.

Problema 2 (30 puntos)

El algoritmo de enrutamiento de Dijsktra es *link state*, donde cada nodo conoce la topología de la red y el costo de cada enlace. Se pide:

a) Para la red de la figura, ejecute el algoritmo paso a paso en el nodo *u* hasta que logra establecer el camino más corto al resto de los nodos. Justifique detalladamente.
b) Para el nodo *u*, de la tabla de forwarding, y dibuje el *shortest-path tree* resultante de la aplicación del algoritmo de Dijsktra.



Solución

Nota: en algunas casos se copian transparencias del curso a modo de resumen de las respuestas.

Pregunta 1 (9 puntos)

a) SSL: Secure Sockets Layer. *Nota: en el curso se mencionó la existencia del protocolo TLS (Transport Layer Security), que es una estandarización del IETF de una versión modificada del SSL v3.0.*

b) El certificado digital es un documento que contiene la clave pública de la entidad "servidor". La utiliza el cliente para cifrar la *Master Secret Key* que ha creado y env iársela al servidor. *Nota 1: el certificado está firmado digitalmente con la clave privada de la CA emisora del mismo. Nota 2: con mayor rigurosidad, cifra la Pre Master Secret Key, pero ésto no fue detallado en el curso*.

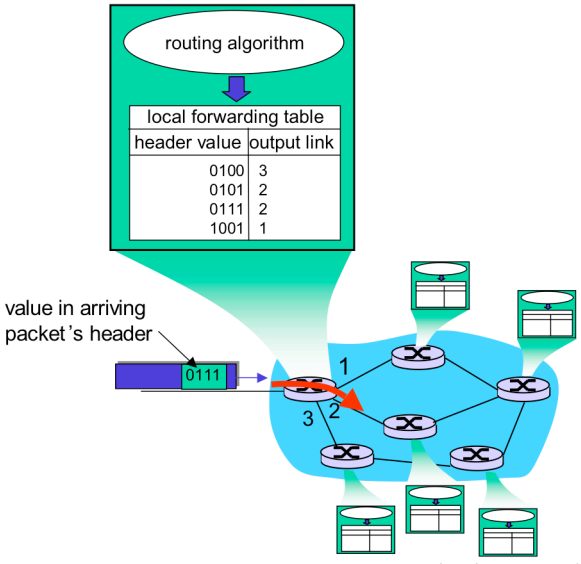
c) Son claves que se derivan a partir de la *Master Secret Key* que ambas partes conocen luego que el cliente se la envió al servidor. Ellas protegerán el tráfico en ambos sentidos, tanto en MAC (*Message Authentication Code*) como cifrado por lo tanto, serán cuatro claves en total, dos en cada sentido.

d) Versión, número de serie, algoritmos y parámetros utilizados para la firma, nombre del emisor, período de validez, nombre del sujeto, algoritmos, parámetros y clave del sujeto, extensiones (en la versión 3), firma del certificado. Para que el cliente pueda asegurar la identidad del servidor, el cliente debe tener la clave pública del servidor a los efectos de poder verificar la firma del certificado en cuestión. El elemento adicional es el certificado de la CA que firmó el certificate request del servidor.

Pregunta 2 (10 puntos)

a) En un contexto unicast, describa los procesos de forwarding y enrutamiento, y explique como se relacionan.

Interacción entre routing & forwarding



Int. Redes de Computadores-Capa de Red 4-70

El proceso de enrutamiento permite determinar el camino entre nodos de la red, típicamente mediante intercambio de información de topología y ejecución de un algoritmo en cada nodo para determinar el camino al resto de los nodos.
El proceso de forwarding consiste en copiar cada paquete entrante a un nodo al puerto de salida adecuado para que siga su camino en la red (o entregarlo a un proceso local si el destinatario es el propio nodo). La

tabla de forwarding, que asocia destinos con puertos (o interfaces) del nodo/router, se construye en base a la información de enrutamiento.

b) En un contexto broadcast, describa un proceso de flooding, y explique como se puede transformar en flooding controlado.

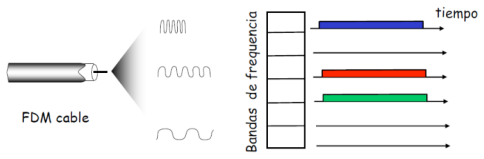
Duplicación en la red

- flooding: cuando un nodo recibe un paquete, envía copias a todos sus vecinos
  - Problemas: ciclos & "tormenta" de broadcasts
- flooding controlado: el nodo solo hace broadcast de un paquete si no lo ha enviado antes
  - el nodo debe llevar la cuenta de los paquetes enviados recientemente
  - o "reverse path forwarding" (RPF): solo envía un paquete si llegó por el camino más corto entre el nodo y la fuente
- spanning tree
  - ningún nodo recibe paquetes redundantes

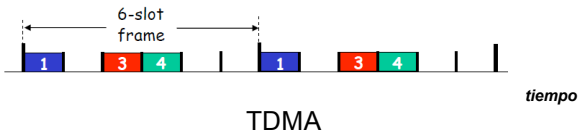
Int. Redes de Computadores-Capa de Red 4-129

Pregunta 3 (5 puntos)

FDMA: *Frecuency Division Multiple Access*.
TDMA: *Time Division Multiple Access*.
Ambas son estrategias de uso de un canal compartido. El primero se basa en que cada usuario del canal utiliza todo el tiempo una banda de frecuencias de todo el rango de frecuencias disponible y el segundo se basa en que cada usuario del canal utiliza todo el rango de frecuencias disponible durante ranuras de tiempo (*timeslots*) equiespaciadas. Ambos modelos son equitativos e implementan una estrategia estática de uso del canal.
Los siguientes diagramas (un ejemplo) ayudan a clarificar la descripción del párrafo anterior.



FDMA



TDMA

Pregunta 4 (9 puntos)

c) Describa la conmutación de paquetes basada en:
   i. Datagramas
   ii. Circuitos virtuales.

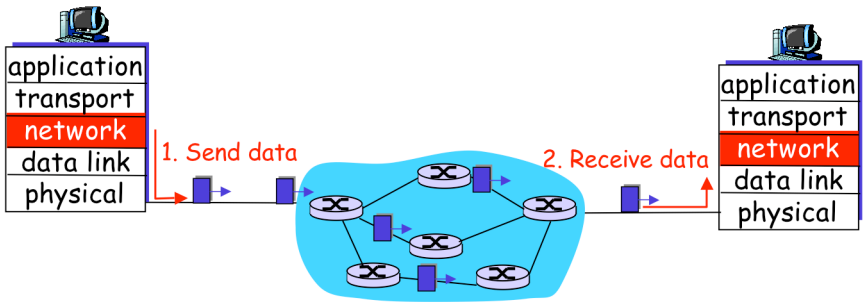
Servicios de Capa de Red orientados y no-orientados a conexión

- datagramas: servicio no-orientado a conexión
- circuitos virtuales (VC): servicio orientado a conexión
- análogo a los servicios de capa de transporte, pero:
  - servicio: host-a-host
  - no se puede elegir: la red provee uno u el otro
  - implementación: en el "core" de la red

Int. Redes de Computadores-Capa de Red 4-11

Redes de datagramas

- no existe procedimiento de establecimiento de conexi ón en capa de red
- routers: no mantienen estado de conexiones extremo a extremo
  - no existe en concepto de "conexión" a nivel de red
- los paquetes son encaminados utilizando la direcci ón de host destino
  - Los paquetes entre un mismo origen y destino pueden tomar caminos diferentes



Int. Redes de Computadores-Capa de Red 4-16

Circuitos Virtuales

"el camino de extremo a extremo se comporta como un circuito telefónico"
- tiene en cuenta parámetros de performance
- la red es responsable a lo largo del camino

- establecimiento de llamada antes del flujo de datos
- cada paquete tiene un identificador de VC (no la dirección del host de destino)
- *cada* router en el camino mantiene el estado de cada conexión
- se pueden *asignar* recursos de routers y enlaces (ancho de banda, buffers) para cada VC (recursos dedicados = servicio predecible)

Int. Redes de Computadores-Capa de Red 4-1

d) Explique ventajas y desventajas de cada opción desde el punto de vista de la asignación de recursos en la red.

Básicamente las redes de datagramas son "de mejor esfuerzo", es decir que consumen los recursos disponibles, y en caso de congestión degradan el servicio por igual a todos los usuarios; aprovechan bien el multiplexado estadístico del tráfico en la red. Las redes de VC ofrecen garantía de QoS basadas en la reserva de recursos, y por lo tanto protegen a los usuarios de las situaciones de congestión, a costo de disponer de más recursos o rechazar conexiones (control de admisión).

Pregunta 5 (7 puntos)

El servicio de control de flujo busca una equiparación de velocidades, entre el envío de unidades de datos (en este caso, segmentos TCP) por parte del transmisor (en este caso la entidad TCP del transmisor) y el "drenado" del buffer del receptor (en este caso la entidad TCP del receptor) por parte de la aplicación del receptor que hace uso de dicha conexión TCP. En particular es una condición de trabajo impuesta por el receptor al emisor. Para ello el receptor anuncia el tamaño (cantidad de bytes) disponible actualmente en su buffer de recepción, a través del campo de 16 bits "Received Window" del encabezado TCP, en los segmentos que envía hacia el transmisor. El transmisor TCP se asegura de no tener más de "Received Window" bytes no-ACKed.

Problema 1 (30 puntos)

a) El proceso adaptador cuenta con un programa principal (función **`adaptador`**) y múltiples instancias de hilos que atienden solicitudes (función **`atiende`**).
La función **`adaptador`** abre el socket que atenderá en **`SERVICE_PORT`**, y realiza un loop infinito esperando por conexiones. Por cada conexión que llegue abre un nuevo hilo y le envía el socket que debe atender.
La función **`atiende`** toma como parámetro el socket que debe atender (**`socketCliente`**) y abre una nueva conexión con el servicio legado (**`socketServidor`**). Luego se queda haciendo un loop en el cual lee los datos disponibles para leer de uno u otro socket, los encripta o desencripta según corresponda y los envía a través del otro socket.

Cuando uno de los sockets se cierra, esto significa que uno de los lados realizó un close y que no estará enviando más datos en esa dirección. Sin embargo, el otro lado de la conexión podría seguir enviando datos durante un tiempo indefinido.
Para mantener este comportamiento con el adaptador en medio, se utilizan dos variables booleanas que indican si aún se pueden recibir datos del cliente (el socket del cliente aún no hizo close) y del servidor (el socket del servidor no hizo close). El proceso que atiende la conexión terminará su ejecución cuando la conexión de ambos lados se haya cerrado, o cuando ocurre un timeout de **MAX_TIME** sin ningún dato recibido de los sockets.

b)

```
void adaptador() {
   socket sock = create_socket();
   bind(sock, PUBLIC_IP, SERVICE_PORT);
   while (true) {
      // lo dejamos bloqueado hasta que llega una conexión
      socket nuevaConexion = accept(sock);
      // la conexión la atiende un hilo separado
      runThread(atiende(nuevaConexion));
   }
}

void atiende(Socket socketCliente) {
   socket socketServidor = create_socket();
   socketServidor.connect(LOCALHOST, LOOPBACK_PORT);
   bool finCliente = false;
   bool finServidor = false;
   while (!finCliente || !finServidor) {
      select (socketServidor, socketCliente, MAX_TIME) {
         case socketCliente:
            char[] segmento = receive(socketCliente)
            if (segment == null) { // el cliente cerró el socket
               finCliente = true;
               close(socketServidor); // no le enviaremos más datos al servidor
            } else {
               send(socketServidor, desencriptar(segmento));
            }
            break;
         case socketServidor:
            char[] segmento = receive(socketServidor);
            if (segment == null) { // el servidor cerró el socket
               finServidor = true;
               close(socketCliente); // no le enviaremos más datos al cliente
            } else {
               send(socketCliente, encriptar(segmento));
            }
            break;
         timeout:
            // cierro los dos
            if (!finServidor) {
                close(socketServidor);
                finServidor = true;
            }
            if (!finCliente) {
                close(socketCliente);
                finCliente = true;
            }
            break;
      }
   }
}
```

c) Por la utilización del servicio de encriptación, la totalidad de las conexiones entrantes tendrán origen en la propia máquina. El servicio legado no tendrá más información basada en el origen de la conexión, que puede afectar la utilización de estadísticas, restricciones de seguridad y otros usos normales de esta información.
Por otra parte, el carecer de esta información hace imposible el inicio de conexiones reversas en el caso más general. Es posible generar algún tipo de conexión reversa si el protocolo transportado transmite como parte de su mensajería las direcciones de red requeridas. Si bien parece una buena solución, esto destruye los conceptos de aislamiento de capas, por lo que, los protocolos bien diseñados no pueden ser asegurados de esta forma, y esta solución extiende la vida útil de protocolos con fallas conceptuales de diseño.
Como último comentario, también debería crearse una aplicación similar a la anterior que encripte el tráfico de esta nueva conexión a los efectos de poder proteger la totalidad del tráfico.

d) Los ataques del tipo SYN FLOODING buscan generar un desmedido consumo de recursos en los equipos atacados. Lo que se debe buscar para mitigar los efectos de estos ataques es detectar situaciones en las que el consumo de recursos sea peligroso, así como, detectar patrones de comportamiento que puedan considerarse sospechosos.
Para implementar la primera medida, se puede hacer que la aceptación de nuevas conexiones se haga en base a la disponibilidad de recursos. Esto puede hacerse tanto accediendo a información del equipo que brinda el servicio, como en base a estadísticas de conexiones establecidas y mediciones de performance de la aplicación.
Por otra parte, la detección de patrones puede tener en cuenta, entre otros, intentos de conexión de una dirección en intervalos de tiempo.
También pueden considerarse direcciones de red adyacentes o consecutivas, considerando la posibilidad de recibir un ataque distribuido, originado en base a todos los hosts de una red, con basa tasa de conexiones individuales, pero muy alta grupalmente.

Problema 2 (30 puntos)

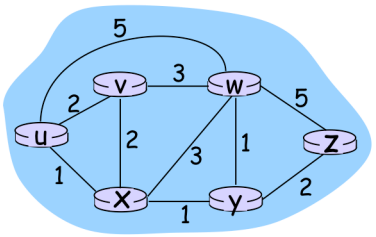
Parte a) Se dede describir el funcionamiento del algoritmo de Dijstra, en particular como evoluciona paso a paso para el caso presentado.

Algoritmo de Dijsktra

```
1  Initialization:
2   N' = {u}
3   for all nodes v
4     if v adjacent to u
5         then D(v) =c(u,v)
6     else D(v) = ∞
7
8  Loop
9    find w not in N such that D(w) is a minimum
10   add w to N'
11   update D(v) for all v adjacent to w and not in N':
12     D(v) = min( D(v), D(w) + c(w,v) )
13   /* new cost to v is either old cost to v or known
14    shortest path cost to w plus cost from w to v */
15 until all nodes in N'
```


Algoritmo de Dijkstra: ejemplo

Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					



Parte b)

Algoritmo de Dijkstra: ejemplo (cont)

"Shortest-path tree" resultante desde u:

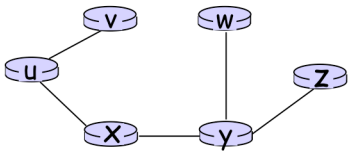


Tabla de forwarding resultante en u:

destino	enlace
v	(u,v)
x	(u,x)
y	(u,x)
w	(u,x)
z	(u,x)