Examen – 2 de febrero de 2010
(ref: sirc1002.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.**
- **Comience cada ejercicio en una hoja nueva.**
- **Sólo se contestarán dudas de letra. No se aceptarán dudas de ningún tipo los últimos 30 minutos del examen.**
- **El examen es individual y sin material.**
- **Es obligatorio responder correctamente al menos 15 puntos en las preguntas teóricas.**
- **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).**
- **Duración: 3 horas.**

Preguntas Teóricas

Pregunta 1 (8 puntos)

a) Defina qué es el DNS y cuál es su función.

El protocolo DNS (Domain Name System) es un protocolo de capa de aplicación que permite realizar la traducción de nombres de hosts a direcciones IP. También se denomina DNS a la base de datos jerárquica distribuida que almacena los mapeos entre nombres y direcciones IP.

b) Explique la diferencia entre las consultas iterativas y recursivas al DNS.

En una consulta iterativa un host le hace una consulta a su DNS local. El DNS local en ese momento deberá encargarse de resolver la consulta reenviándola a todos los servidores DNS que sean necesarios. Comienza por consultar la raíz, y cada DNS que consulte (que no tenga la respuesta) le indicará cuál es el servidor siguiente (más abajo en la jerarquía) al que debe preguntarle.

En una consulta recursiva el host le hace una consulta a su DNS local. El DNS local preguntará a un servidor raíz. Si el raíz no conoce la respuesta, él mismo le preguntará al servidor que corresponda que esté más abajo en la jerarquía. Si este servidor no conoce la respuesta, se la preguntará al siguiente que esté más abajo en la jerarquía. Luego de que obtienen la respuesta, se volverá hacia atrás en la recursión para devolver el resultado al que consultó originalmente.

Pregunta 2 (8 puntos)

a) ¿Cuáles son las funciones clave de la Capa de Red?

Forwarding: mover paquetes entre puertos de entrada y salida del router
Enrutamiento: determinar la ruta de los paquetes desde origen a destino. Esto se realiza por medio de algoritmos de enrutamiento.

b) ¿Qué objetivo cumple cada una de las funciones indicadas en la parte a), y cómo interactúan para resolver los cometidos de la Capa de Red?

El enrutamiento genera las tablas de forwarding locales. Posteriormente para los paquetes recibidos se aplican las reglas especificadas en las mencionadas tablas.

c) Describa brevemente las dos formas de prestar servicios de capa de red.

Pregunta 3 (8 puntos)

a) Explique el funcionamiento del mecanismo de chequeo por paridad en dos dimensiones utilizado para detección de errores en Capa de Enlace.

La paridad en dos dimensiones considera el flujo de datos como un flujo "bi dimensional", tomando el flujo como una matriz donde cada palabra es puesta en una fila, y palabras consecutivas en filas consecutivas. De esta forma el bit j-ésimo de cada palabra forma la columa j-ésima de toda la matriz.

A cada fila se le agrega un bit de paridad (calculado sobre la fila) y cada cierto numero de palabras se agrega una palabra de paridad calculada por columna.

b) ¿Permite realizar correcciones? ¿De cuántos bits? Justifique

La paridad en dos dimensiones permite:
• Corregir errores de un bit por palabra y por columna, ya que esto hace posible saber que bit está invertido a través de la intersección de la fila y columna que fallan la verifiación de paridad.
• En algunos casos permite detectar errores dobles.

Pregunta 4 (8 puntos)

a) Explique el concepto de Firma Digital y su uso en caso de mensajes de correo electrónico.

Técnica criptofráfica que busca lograr "en el mundo digital" objetivos similares a los que se persiguen con las firmas manuscritas tradicionales: verificable, no falsificable, no repudiable, integridad del mensaje.

La firma digital combina la criptografía asimétrica (pares de claves "pública-privada") con funciones de hash. A partir del mensaje a firmar m, primero se obtiene un hash de él H(m), al que se lo cifra con la clave privada del emisor del mismo Kse y el resultado Kse(H(m)), junto con el mensaje m, son enviados al receptor (observar que no se brinda servicio de confidencialidad). El receptor del mensaje utiliza la clave pública del emisor Kpe para obtener el hash Kpe(Kse(H(m))) y compararlo con el hash que él calcula a partir del mensaje recibido H(m) , dando por válida la firma si ambos hashes son iguales.

b) ¿MAC (Message Authetication Code) puede ser utilizado como técnica para firmar digitalmente? Justifique su respuesta.

No. MAC se basa en criptografía simétrica, por lo que su uso en firmas digitales no es posible ya que el principio de funcionamiento de éstas se basa en el cifrado asimétrico.

Pregunta 5 (8 puntos)

a) Defina el concepto de jitter y su diferencia respecto al retardo.

El retardo es el tiempo que tarda un paquete desde que sale del emisor hasta que llega al receptor, mientras que el jitter es la variación de este retardo entre distintos paquetes de un mismo flujo. El retardo puede variar entre un paquete y otro porque (por ejemplo) los paquetes pueden tomar diferentes caminos de la fuente al destino. Un alto jitter puede implicar que paquetes que salieron más tarde de la fuente lleguen más temprano al destino (llegan en desorden).

Si la fuente y el destino están a varios saltos de distancia, y todos los paquetes del flujo siguieran el mismo camino pero no hubiera congestión, podríamos mantener el retardo constante (aunque sea un retardo de varios segundos) mientras que el jitter sería mínimo o nulo.

b) Analice las siguientes aplicaciones de acuerdo a qué tan sensibles son respecto del jitter, el retardo, y la pérdida de paquetes.

- Descarga confiable de archivos.
- Streaming almacenado.

Para la descarga confiable de archivos se necesita que lleguen todos los paquetes o el archivo será inválido, por lo que no es tolerante a pérdida de paquetes (hay que retransmitir). El retardo solo implicaría tener que esperar un poco más de tiempo para comenzar a descargar el archivo, por lo que en general no significará un problema. Lo mismo ocurre con el jitter, incluso si los paquetes llegan en desorden será responsabilidad de la capa de transporte esperar a que lleguen todos y entregarlos ordenadamente, pero esto solo implicará que se espere un poco más para la descarga.

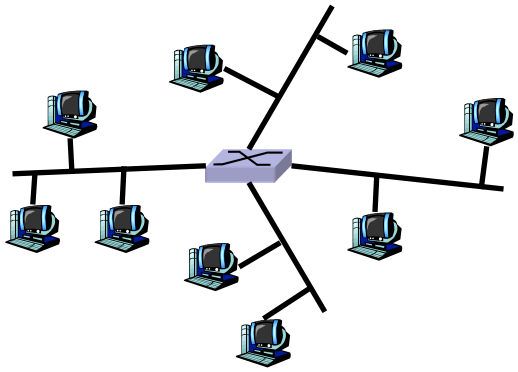
En el caso de streaming almacenado el usuario puede esperar un poco (unos cuantos segundos) a que comience la transmisión, pero una vez que comienza deseará que la misma no se corte y no tenga que esperar de nuevo. Estas transmisiones son por lo tanto tolerantes al retardo, y en cierta medida pueden tolerar el jitter en caso de que no sea muy excesivo. Si se pierde alguno de los paquetes de la transmisión el cliente puede saltearse la reproducción del mismo o interpolar paquetes cercanos. Siempre que no se pierdan demasiados fragmentos, la calidad de la reproducción no se deteriorará notoriamente.

Problemas Prácticos

Problema 1 (30 puntos)

Para implementar su red interna, una empresa cuenta con un conjunto de segmentos ethernet, los cuales desea interconectar mediante un switch de capa de enlace. El switch con el que cuentan es programable, y le han encargado a usted que implemente el programa que debe ejecutar para desempeñar la función correspondiente. La topología de la red se esboza en la siguiente figura:
El switch es del tipo store-and-forward y cuenta con `CANT_INTERFACES` interfaces, cada una conectada a un segmento ethernet. El programa que esté ejecutándose en el switch debe cumplir con los siguientes requisitos:

- Para poder conectar y desconectar hosts en los segmentos ethernet a voluntad sin realizar configuraciones extra, se debe implementar un mecanismo de self-learning.



- Los hosts no avisan si se desconectan de una ethernet y se conectan a otra. En estos casos es tolerable que se reenvíen las tramas a la ethernet antigua durante un tiempo (estas tramas se perderán), pero el tiempo debe limitarse a un máximo de `TIMEOUT`. En caso de que el host comience a enviar nuevas tramas desde su nueva ubicación, el switch deberá aprender inmediatamente la nueva ubicación y descartar la antigua.
- Debido a que esta red cuenta con un solo switch, puede considerarse que no habrá ciclos en la topología, por lo que no hay que preocuparse de que las tramas de un mismo host puedan llegar desde dos interfaces a la vez (esto solo ocurrirá en el caso mencionado en el punto anterior).
- No se pide que se realice control de CRC.

Se asume que se cuenta con un tipo de dato `trama` que contiene los campos habituales de una trama de capa de enlace. Se cuenta además con las siguientes primitivas:

```
void receive(out trama t, out int nro_interfaz)
void send(in trama t, in int nro_interfaz)
```

Se pide:

a) Indique cuáles de los campos de la trama de capa de enlace deberá utilizar el switch para su funcionamiento.

b) Defina la estructura de datos que utilizará el switch para almacenar la información sobre los hosts que se van conociendo.

c) Implemente el programa que ejecuta el switch.

Solución

a) Se debe utilizar los campos MAC address de origen y destino de la trama ethernet.

b)

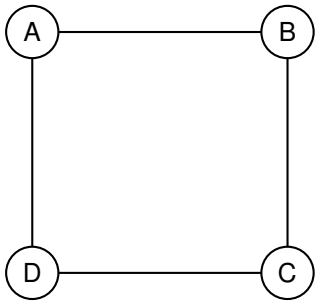
```
struct nodo_tabla {
	MAC host_dir;
	int nro_interfaz;
	int timestamp;
};
Map<MAC, nodo_tabla> tabla_switch;
```

c)

```
void main() {
	tabla_switch = new Map<MAC, nodo_tabla>();
	while (true) {
		trama t;
		int interfaz;
		receive(t, interfaz);

		// primero vemos a dónde lo reenviamos
		MAC destino = t.MAC_destino;
		if (tabla_switch.containsKey(destino)) {
			// tenemos una potencial ubicación del host destino
			nodo_tabla nodo = tabla_switch[destino];
			if (time() - nodo.timestamp >= TIMEOUT) {
				// la entrada de la tabla está vencida, la eliminamos
				tabla_switch.remove(destino);
				flood(t, interfaz);
			} else if (interfaz != nodo.nro_interfaz) {
				// reenvio solo si vino por una interfaz distinta de la
que voy
				// a mandar, si es la misma ya tiene que haberse recibido
				send(t, nodo.nro_interfaz);
			}
		} else {
			// no sabemos dónde mandarlo
			flood(t, interfaz);
		}

		// por último procesamos los datos del origen
		MAC origen = t.MAC_origen;
		if (tabla_switch.containsKey(origen)) {
			// ya lo teníamos en la tabla, actualizamos los datos
			nodo_table nodo = tabla_switch[origen];
			nodo.nro_interfaz = interfaz;
			nodo.timestamp = time();
		} else {
			// no lo teníamos, creamos uno nuevo
			nodo_table nodo = new nodo_table();
			nodo.host_dir = origen;
			nodo.nro_interfaz = interfaz;
			nodo.timestamp = time();
			tabla_switch[origen] = nodo;
		}
	}
}

void flood( trama t,  int interfaz_origen) {
	// realiza flooding controlado
	// mandando a todos menos a la interfaz de origen
	for (int i = 0; i < CANT_INTERFACES; i++) {
		if (i != interfaz_origen) {
			send(t, i);
		}
	}
}
```

Problema 2 (30 puntos)

Se considera una red de cuatro nodos A,B,C y D con la topología que se muestra en la figura; los enlaces tienen costo unitario.
Esta red ejecuta un protocolo de enrutamiento genérico de la familia "vector-distancia" (distance-vector).



Se pide:

a) Suponiendo la topología de la red estable en el tiempo y que los nodos son todos encendidos simultáneamente (es decir, se inician todos en el mismo estado con sus tablas de distancias vacías), deduzca las sucesivas tablas de distancias de A, B, C, D hasta llegar al estado estable de la red.

b) Explique qué sucede con las tablas de distancias si se cae el nodo D. Justifique incluyendo algunas iteraciones de las tablas de distancias de A. B y C.

Solución

a)

t=0									
en A					en B				
	A	B	C	D		A	B	C	D
A	0	1		1	A				
B					B	1	0	1	
D					C				
en D					en C				
	A	B	C	D		A	B	C	D
A					B				
C					C		1	0	1
D	1		1	0	D				

t=1									
en A					en B				
	A	B	C	D		A	B	C	D
A	0	1	2	1	A	0	1	inf	1
B	1	0	1	inf	B	1	0	1	2
D	1	inf	1	0	C	inf	1	0	1
en D					en C				
	A	B	C	D		A	B	C	D
A	0	1	inf	1	B	1	0	1	inf
C	inf	1	0	1	C	2	1	0	1
D	1	2	1	0	D	1	inf	1	0

t=2									
en A					en B				
	A	B	C	D		A	B	C	D
A	0	1	2	1	A	0	1	2	1
B	1	0	1	2	B	1	0	1	2
D	1	2	1	0	C	2	1	0	1
en D					en C				
	A	B	C	D		A	B	C	D
A	0	1	2	1	B	1	0	1	2
C	2	1	0	1	C	2	1	0	1
D	1	2	1	0	D	1	2	1	0

b)

Se trata del problema del conteo infinito. Cuando el nodo D cae, el nodo C detecta que la ruta mas corta a D es inaccesible y por lo tanto elije la segunda mas corta que es a través de B con distancia total 3. Lo mismo pasa con A. Luego B actualiza su tabla con las distancias a D en A y C y aumenta la suya a 4. el proceso se repite hasta infinito o la mayor distancia posible.