Examen Redes de Computadoras

Ángel Iván Gladín García
Emiliano Galeana Araujo

Facultad de ciencias, UNAM

Fecha de entrega: Lunes 30 de marzo de 2020

1 Parte teórica

1.1 Memoria fotográfica

1.1.1 Edificio

Se decidió utilizar el edificio Poniente (P) de la facultad de Ciencias de la UNAM.

1.1.2 Descripción

El edificio cuenta con 4 pisos que son: Planta Baja, Primer Piso, Segundo Piso y el Sótano.

Acometida Es la parte de la instalación eléctrica que se construye desde las redes de distribución, hasta las instalaciones del usuario.

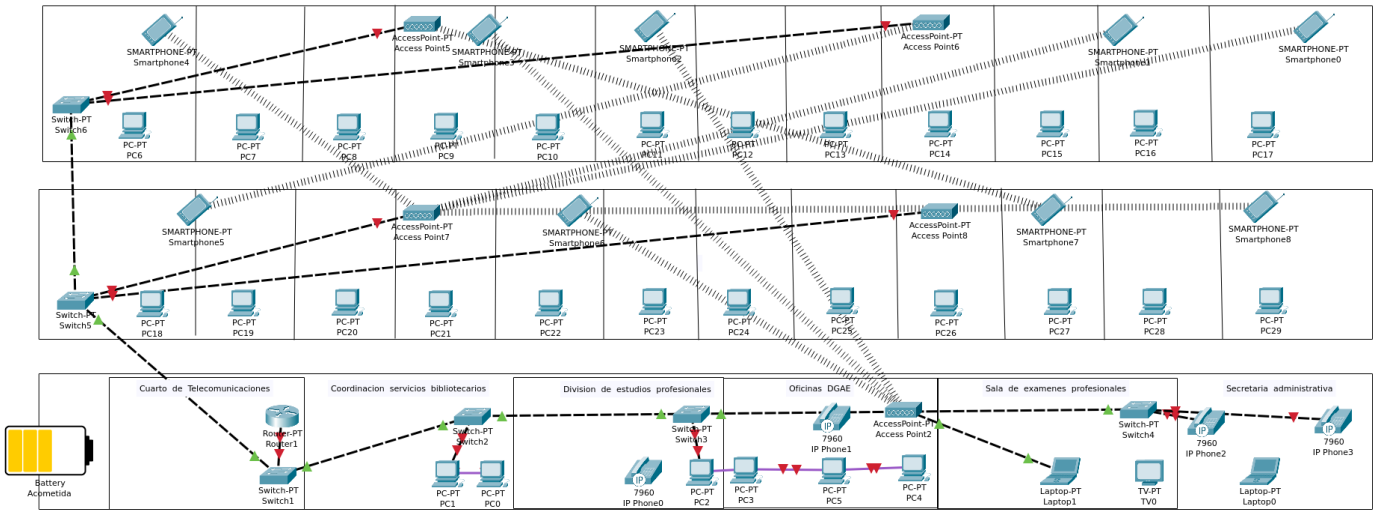
Cuartos de telecomunicaciones Área que es utilizada para el uso exclusivo de equipo asociado con el sistema de cableado de telecomunicaciones.

Racks Racks de cada cuarto (Incluyendo Patch Panels) Estructura que permite sostener un dispositivo tecnológico. Tenemos en el cuarto de telecomunicaciones, y en los acces points de cada piso.

Switches Switches de acceso, distribución y core. Representan el perímetro de la red, por dónde entra o sale el tráfico de la red en cuestión.

Ruteador Dispositivo empleado a la hora de la interconexión de una red de ordenadores.

El siguiente diagrama representa al edificio P, con algunas modificaciones para poder hacer ejemplos más claros.



1.1.3 Inventario

Debido a que este un ejemplo meramente de demostración, es importante aclarar que la siguiente tabla no fue hecha con datos reales.

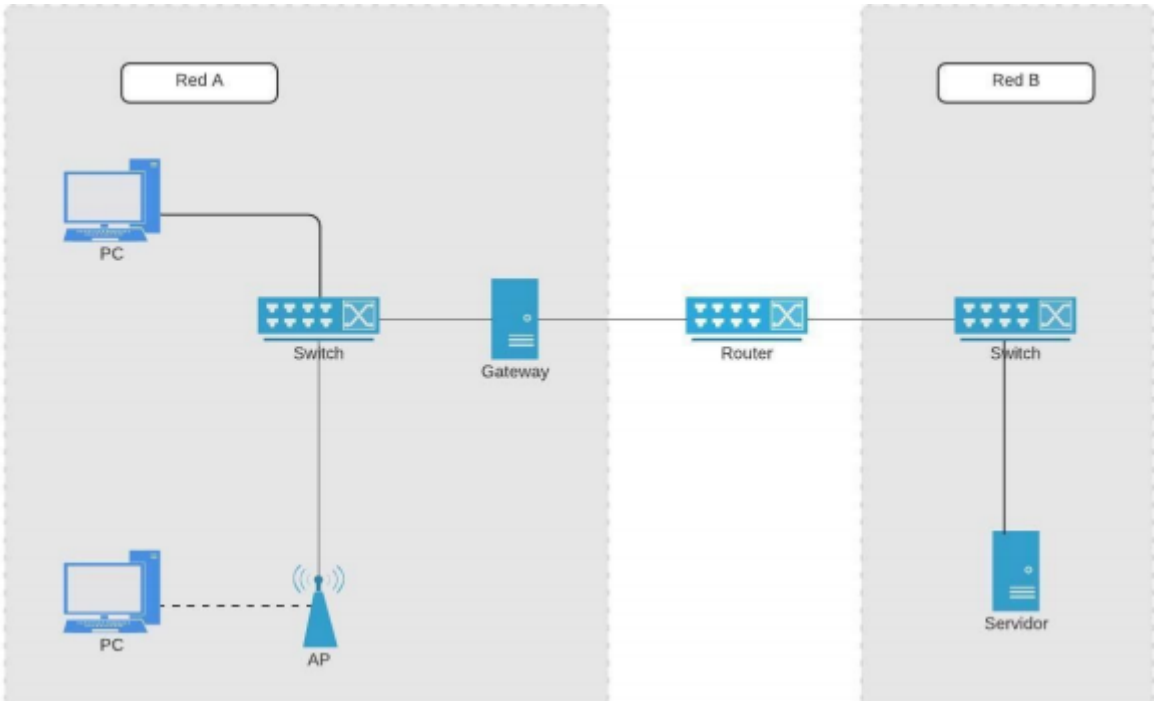
0	1	0	0	1	0	0	0
1	1	0	0	0	0	0	0
1	0	0	0	1	0	0	0
1	1	0	0	0	0	0	0
0	1	0	0	1	0	0	0

Table 1: Inventario .

1.1.4 Tabla de rutas

2 Parte práctica

2.1 Considerar una red de la siguiente forma



Llevar a cabo lo siguiente:

- Asignar direcciones de un *segmento privado de clase B* a los equipos de la Red A.

  Recordemos los rangos de direcciones para utilizar con redes privadas:

  - Clase A: 10.0.0.0 a 10.255.255.255
  - Clase B: 172.16.0.0 a 172.31.255.255
  - Clase C: 192.168.0.0 a 192.168.255.255

  Recordemos también lo que hace el comando `ifcfg: simplistic script which replaces ifconfig IP management`, es una manera más simple de usar `ifconfig`, que nos permite cambiar las direcciones.

  La interfaz que vamos a cambiar es `eth1`.

  Primero, necesitamos quitar la IP que tiene asignada en este momento. con el siguiente comando `ifcfg eth1 inet 0.0.0.0 down`. Una vez hecho esto, nuestra interfaz está abajo y podemos asginar otra dirección única como sigue: `ifcfg eht1 inet xxx.x.x.x netmask xxx.xxx.xxx.x up`, con esto, solo bastaría agregar las direcciones que se quieran. Hay que hacer este pifcfg eth1 inet 0.0.0.0 down. procedimiento para todos los equipos de la Red A.

  Como son pocos los equipos, podemos hacerlo de uno por uno, pero si fueran varios podríamos hacer uso de `DHCP`.

- Establecer la dirección del *gateway* en la Red A.

  `ip default-gateway xxx.xx.xx.x`, donde xxx.xx.xx.x es la dirección que se quiere establecer.

- Indica la configuración de red que se debe usar en los equipos de la Red A (*dirección de red, gateway, máscara de subred* y otros que consideres necesarios).
- Hacer uso de un *segmento de red público de clase C* en los equipos de la Red B.

  Al igual que en inciso anterior, esta es la clase que nos interesa Clase C: 192.168.0.0 a 192.168.255.255, y usando el mismo comando (`ifcfg`) y la misma idea, procederíamos a cambiar el segmento desactivando la interfaz que se requiera, asignando el nuevo segmento, y finalmente levantarlo (`up`).
- Establecer la dirección del *gateway* en la Red B.

  Igual que en el inciso anterior, hacemos uso del siguiente comando: `ip default-gateway xxx.xx.xx.x`.
- Indica la configuración de red que se debe usar en el servidor de la Red B (*dirección de red, gateway, máscara de subred* y otros que consideres necesarios).
- Asignar una *dirección de red* a las interfaces correspondientes del *router*, de acuerdo a las redes conectadas.
- Incluir en el diagrama un *DNS* en una red independiente, haciendo uso de un segmento de *red de clase C*, con direcciones públicas.
- Considerando que todos los clientes y servidores involucrados son equipos Linux, indica la forma en la que se debe establecer la configuración de *red estática* y la configuración necesaria para llevar a cabo la *resolución de nombres* en el sistema (archivos y comandos necesarios).
- Por cuestiones de simplicidad, asignar a cada dispositivo una *dirección física de 4 números hexadecimales*.
- Explicar con detalle el procedimiento para llevar a cabo la conexión entre una computadora de la Red A al servidor de la Red B, considerando lo siguiente:
  - Se debe establecer el nombre del servidor con base en tu nombre y apellidos.

    jose-luis.torres.local o juan.camacho.local
  - El servidor de la Red B cuenta con una aplicación de "servidor Web" el cual solamente acepta peticiones mediante *HTTP* version 1.1 en el puerto 54321.
  - El cliente establecerá una conexión para obtener el documento /paginas /directorio.html.
  - El cliente debe hacer uso de un puerto, fuera del rango establecido para los "puertos bien conocidos" y "puertos registrados", para llevar a cabo la conexión.
  - El *gateway* de la Red A implementa *NAT*, haciendo uso de una dirección de un segmento de *clase C*.
  - Se debe incicar detalladamente los passos que se siguen en el cliente de la Red A que hacen uso de una conexión con un *medio guiado*, para establecer la conexión con el servidor, incluyendo los *bloques de datos* que se general al pasar por cada una de las capas de TCP/IP. Se debe mostrar la forma en la que se aplica el "encapsulamiento" (En las clases se presentaron ejemplos de esto)
  - En la *Capa de Enlace* se debe considerar el uso de "relleno de bits" para llevar a cabo el *envío de tramas*.
  - Se debe mostrar la forma en la que se "desencapsulan" los paquetes al llegar al *router* y la forma en la que se vuelven "encapsular" para su reenvío.

2.2 Explicar cuáles son las características que debe tener un código para permitir llevar a cabo *detección* o *corrección* de errores

Recordemos los tipos de errores que podemos tener:

- De bit: Ocurre cuando únicamente un bit de una unidad de datos cambia. (Por ejemplo, cambia de 0 a 1 o viceversa).
- De ráfaga: Este error ocurre cuando dos o más bits de la unidad de datos han cambiado. En estos errores no necesariamente tenemos cambios en bits consecutivos.

Para detectar estos errores, podemos hacer uso de la *redundancia*, que, consiste en enviar dos veces cada unidad de datos, así podemos hacer una comparación bit a bit entre ambos datos y detectar si hay errores. Para aplicar esta técnica, podemos añadir al flujo de datos un grupo pequeño de bits al final de cada unidad, a estos los llamamos bits redundantes y los descartams una vez que se compruebe la integridad de la transmisión.

Para la correción de errores, podemos hacerlo de dos maneras, si existe un error, podemos pedir al emisor que retransmita la información, la segunda opción es que el receptor los corrija.

En este contexto, responde las siguientes preguntas:

- ¿Qué características debe tener un código para permitir la detección de errores de un máximo de n bits?

  Como se explicó antes, podríamos mandar la información dos veces y compararla, y así, detectar errores. Pero la más usada, como también se mencionó, es el uso de los bits de redundancia. Lo que se requiere del código para permitir la detección es que tenga n bits más que nos ayuden a verificar si existe un error. Esto haría el mensaje más largo, pero más sencillo el detectar si existe o no un error.

- ¿Qué características debe tener un código para permitir la corrección de errores de un máximo de n bits?

  Para poder llevar a cabo la correción de errores, (Hablando en errores de un bit) necesitamos saber cuál fue el bit que se invirtió y usamos los bits de redundancia. ¿Cuántos bits de redundancia son necesarios?

  Supongamos que usamos r bits de redundancia, y que m es la cantidad de bits de nuestro mensaje, el mensaje a transmitir tiene una longitud de $m + r$ bits. Y se necesita que los r bits sean capaces de indicar todas las posibilidades de error de 1 bit, incluyendo el no error (Esto son $m+r+1$ posibilidades). Sabiendo esto, r debe ser tal que $2^r \geq m+r+1$.

  ¿Cómo corregir errores? R.W. Hamming desarrolló una técnica y se puede aplicar a unidades de datos de cualquier longitud usando la relación entre bits de datos y de redundancia. Lo que hace es colocar en ciertas posiciones entre 0 y la longitud del mensaje los bits de redundancia. El receptor recibe esta información, calcula valores para los bits de redundancia usando el mismo conjunto que el emisor (Mismas posiciones). El resultado nos dice si ha habido un error y en qué bit se ocasionó. De esta manera se puede solucionar.

  ¿Qué pasa con errores de ráfaga? Lo antes mencionado es muy útil cuando se trata de errores de un bit. Para solucionar los errores en ráfaga se puede usar una implementación de Hamming, pero el número de bits de redundancia es muy elevado.

- ¿Cuál es el número mínimo de mensajes que deben enviar las dos partes para garantizar una comunicación libre de errores?

Si queremos minizar, podría realizarse con n mensajes (El número de mensajes en el que partimos el mensaje original para su transmisión). De esta manera, si no hubiese errores, el emisor solo tendría que esperar cierto tiempo a que el transmisor envié un mensaje de que hubo un error; Y, en caso de no llegar, podría continuar mandando los siguientes mensajes.

El tiempo que definimos depende de cuánto tarde el receptor en verificar si existen o no errores en el mensaje recibido.

2.3 ¿Cuáles son las tres características que componen al World Wide Web?

- URI (Uniform Resource Identifier), que es un sustema universal para referenciar fuentes en la Web, así como páginas Web.
- HTTP (HyperText Transfer Protocol) especifica cómo el navegador y el servidor se comunican entre si.
- HTML (HyperText Markup Language) se usa para definir la estructura y el contenido de los documentos con hipertexto.

2.4 Explica el funcionamiento de CRC

Comprobación de Redundancia Cíclica o Control de Redundancia Cíclica, del inglés (Cyclic Redundancy Check), es una función diseñada para detectar cambios accidentales en datos de computadoras y es comunmente usada en redes y dispositivos de almacenamiento.

2.4.1 Funcionamiento

A cada bloque de datos le corresponde una secuencia fija de números binarios conocida como código CRC (Se calcula con una misma función para cada bloque). Ambos se envían o almacenan juntos. Cuando un bloque de datos es leído o recibido , dicha función es aplicada nuevamente al bloque, si el código CRC generado no coincide con el código CRC original, entonces significa que el bloque contiene un error. Eso hará que el dispositivo intente solucionar el error releyendo el bloque o requiriendo que sea enviado nuevamente. Si ambos códigos coinciden, entonces se asume que el bloque no contiene errore (Existe una remota posibilidad de que exista un error sin detectar).

2.4.2 Ejercicio

Considerar el siguiente polinomio:

CRC-16 = 11000000000000101 = $X^{16} + X^{15} + X^2 + 1$

1101011011

Calcula el resultado de aplicar este polinomio a la cadena de texto “Hola mundo”, mediante CRC. Incluye el procedimiento para obtener el resultado.

“Hola mundo” = `01001000 01101111 01101100 01100001 00100000 01101101 01110101 01101110 01100100 01101111`

Agregamos los bits redundantes... Como tenemos un polinomio de 17 bits, agregamos 16 bits.

`01001000 01101111 01101100 01100001 00100000 01101101 01110101 01101110 01100100 01101111 00000000 00000000`

Procedemos a hacer la división aydándonos con un *XOR*. No pondremos toda la división aquí, pero adjuntaremos una hoja de cálculo donde se puede revisar, sin embargo, vamos a notar los casos que consideramos importantes.

La primer fila representa los primeros ocho bits de nuestro mensaje, y la segunda representa los primeros ocho bits de nuestro polinomio. En la tercera anotamos el resultado de la aplicación de `XOR`.

Vemos que, como el primer bit de la aplicación es 1, no podemos avanzar, por lo que volvemos a realizar la aplicación. La cuarta línea representa los primeros ocho bits de nuestro polinomio. Y la quinta línea representa el resultado de la segunda aplicación del `XOR`.

0	1	0	0	1	0	0	0
1	1	0	0	0	0	0	0
1	0	0	0	1	0	0	0
1	1	0	0	0	0	0	0
0	1	0	0	1	0	0	0

Table 2: Aplicación de XOR.

Como ya llegamos a tener un 0 como primer bit, (Recordemos que solo estamos usando los primeros ocho bits, en la hoja se encuentra con los 17 bits) procedemos a mover nuestro resultado hasta encontrar un 1, y aplicar el `XOR` con el polinomio. Como ahora nos sobra un bit, bajamos uno de nuestro mensaje.

Continuando con la aplicación, nos encontramos con el siguiente escenario. La primer línea es el segmento al que le hemos aplicado `XOR`, y la segunda línea, nuestro polinomio. Podemos ver que nuestra tercera línea empieza con muchos 0's, por lo que a la siguiente aplicación, vamos a tener que posicionar nuestro polinomio para que coincida con el primer uno que encontremos.

1	1	0	0	0	0	1	1
1	1	0	0	0	0	0	0
0	0	0	0	0	0	1	1

Table 3: Aplicación de XOR.

Después de hacer la aplicación a toda la cadena (mensaje y bits reduntantes), nuestro residuo (El CRC) es el siguiente `00011110 11101000`. Lo que sigue es adjuntar nuestro residuo con el mensaje original, esto quiere decir que nuestro mensaje qudaría de la siguiente manera: `01001000 01101111 01101100 01100001 00100000 01101101 01110101 01101110 01100100 01101111 10011011 10001101`. Este es el mensaje que se envía al receptor.

Lo siguiente es volver a hacer una división (Aplicación de `XOR`) siguiendo con el mismo polinomio y ahora usando el mensaje que recibe el receptor (Igual está en la hoja de cálculo esta división).

Haciendo la aplicación al polinomio, vemos que el residuo nos queda `00000000 00000000`. Lo que quiere decir que el mensaje se envió y recibió sin errores.

2.5 Extra

- Describe las potenciales *debilidades* del protocolo *DNS*, ¿Qué mecanismos de seguridad podrías implementar sobre este protocolo para mitigar dichas vulnerabilidades?

  Nos centramos en el concepto de que el *DNS* funciona como un directorio telefónico, donde nosotros ponemos el nombre de dominio y nos regresa la dirección IP.

La debilidad que creemos podría ser más importante, es que el protocolo al pedir por un nombre de dominio, nos regrese la dirección de un sitio malicioso o una dirección ajena a la que pedimos. Esto es importante, pues no se podrían resolver las consultas que se hagan. Y la manera que se nos ocurrió para mitigar esta vulnerabilidad sería que el protocolo pueda comparar el resultado con resultados anteriores o resultados de computadoras vecinas en nuestra red. De esta manera si algún protocolo regresa una dirección distinta a la de los demás, podríamos no tomarla en cuenta, e ir por la respuesta que la mayoría de computadoras efectuando el protocolo regresaron.

- Explicar la diferencia entre "codificar" y "cifrar". Incluir ejemplos de cada uno de ellos.

  Cuando hablamos de 'cifrar', nos referimos a ocultar la información basándonos en la sintáxis del mensaje. Podemos utilizar algoritmos para realizar esto, los cuales suelen utilizar una clave para transformar la estructura del mensaje.

  Si hablamos de 'codificar' se basa en alterar la semántica del mensaje, lo que está relacionado con el significado del mensaje.

  Ejemplos que podemos aplicar a lo anterior, podemos *cifrar* mensajes en aplicaciones de mensajería (Telegram, WhatsApp, etc). Y un ejemplo de *codificar* puede ser en la computadora con emojis, cuando representamos el emoji con caracteres, por ejemplo `U+1F601`, y podemos verlo en la terminal como un emoji.

References

[1] https://www.varonis.com/blog/what-is-dns/

https://www.welivesecurity.com/la-es/2016/12/07/codificacion-o-cifrado-diferencia/?fbclid=IwAR3R2RuHwb9grV7cKTuIXJOhtLbkw2AblOub1RrrMujh5nKns0tuqTv8fYE

https://www.cs.mcgill.ca/ rwest/wikispeedia/wpcd/wp/w/World_Wide_Web.htm

https://www.ecured.cu/Comprobaci

https://sites.google.com/site/sistemasdemultiplexado/arquitecturas-de-las-redes-de–comunicacin-caractersticas/8–deteccin-y-correccin-de-errores