Solución del Examen - 20 de febrero de 2009 (ref: sirc0903.doc)

Pregunta 1 (8 puntos)

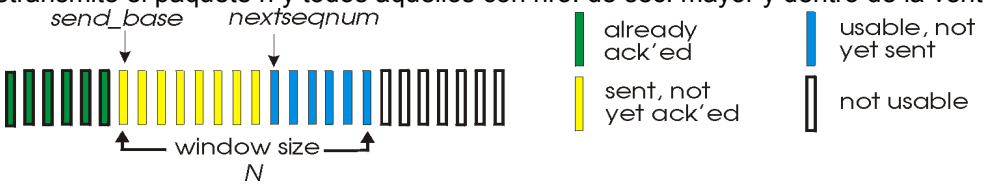
Explique el principio de funcionamiento de los protocolos que operan bajo el esquema Go-Back-N.

(Extraído del material de referencia de las clases teóricas, clase 2 del capítulo 3 del libro de referencia, Capa de Transporte)

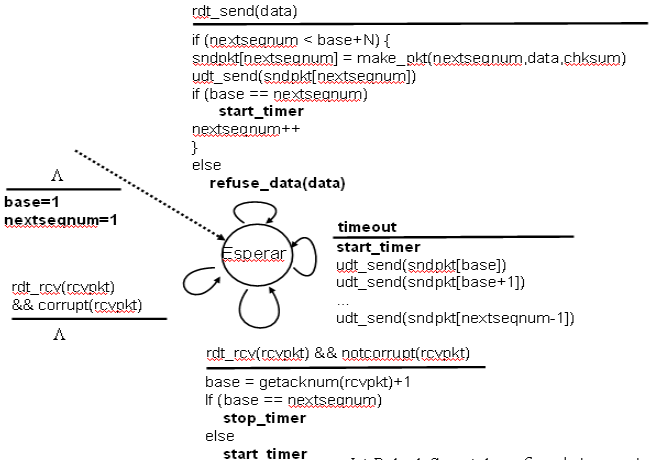
Go-back-N: titulares
El transmisor puede tener hasta N paquetes no-ACKed en el pipeline
El receptor sólo envía ACKs acumulativos
No envía el ACK de un paquete si hay un hueco
El transmisor tiene un timer para el paquete más viejo no-ACKed
Si el timer vence, retransmite todos los paquetes no-ACKed

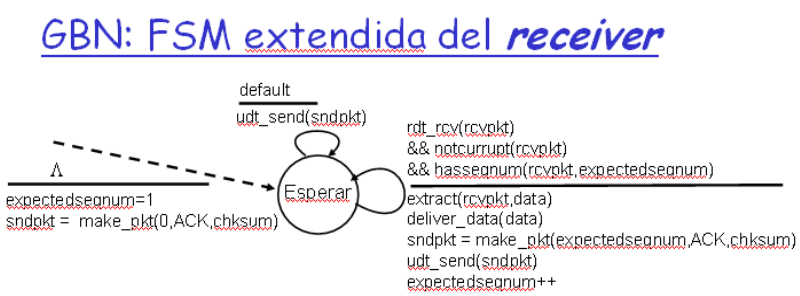
Emisor:

Número de secuencia de k bits en el encabezado del paquete
"window" permitida de hasta N paquetes consecutivos no-ACKed
Si la ventana no está llena, se puede seguir transmitiendo
ACK(n): ACKs de todos los paquetes hasta él, incluyendo el nro. de sec. n -- "ACK acumulativo"
podría recibir ACKs duplicados (ver receptor)
timer único: para el paquete más viejo aún no-ACKed
timeout(n): retransmite el paquete n y todos aquellos con nro. de sec. mayor y dentro de la ventana

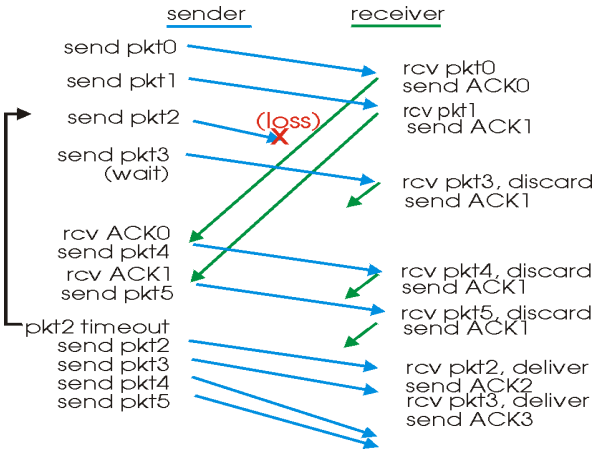


GBN: FSM extendida del *sender*





- ACK-solamente: siempre enviar ACK para el paquete correctamente recibido y en orden con el mayor nro. de sec.
  - Se podrían generar ACKs duplicados
  - Solamente necesitamos recordar expectedseqnum
- Paquetes fuera de orden:
  - Descartar (no almacenar en *buffer*) -> sin buffer en el receptor
  - Re-ACK paquetes con el mayor nro. de sec. en orden



Pregunta 2 (10 puntos)

a) *Explique el concepto de enrutamiento jerárquico.*
b) *Describa un protocolo de enrutamiento interno (IGP) que utiliza jerarquías*
c) *¿En qué consiste el mecanismo de "hot-potato routing"?*

Ver clases de teórico de Capa de Red

Pregunta 3 (6 puntos)

Explique el concepto VLAN (p.e. 802.1q) y comente sobre su utilidad.

Es una tecnología que nos permite tener más de un dominio de broadcast en una misma red switcheada. Se trata de una mejora que se puede incorporar a la red se manera transparente para los usuarios finales ya que la configuración corre por cuenta de los administradores de la misma.
La utilidad radica en que se puede tener "varias redes" LAN soportadas por el mismo equipamiento y donde la pertenencia de cada puerto a cada red se puede ajustar bajo demanda. Al poder distinguir qué puertos de un switch pertenecen a qué VLAN, indirectamente es una mejora respecto a seguridad, en particular en lo

que refiere a tráfico no unicast y también en el caso de falta de determinadas entradas en las tablas que asocian dirección MAC con puerto.

Pregunta 4 (10 puntos)

Considere la arquitectura de Servicios Diferenciados.

a) *Describa las funciones que debe realizar un "edge router" y un "core router", y qué campo de IPv4 se utiliza para marcar los tipos de servicio.*
b) *Describa el mecanismo de forwarding "Per Hop Behaviour".*

Titulares:

a) El "edge router" debe clasificar y marcar el tráfico que ingresa a la red. El "core router" debe encaminar el tráfico de acuerdo a las marcas definidas en el ingreso, y al PHB definido para cada clase de tráfico, usando el campo TOS de IPv4.
b) El PHB son las políticas de tratamiento del tráfico de acuerdo a las marcas realizadas en el borde de la red. Por ejemplo prioridades, % del ancho de banda, etc.

Ver clases de teórico de "Multimedia Networking".

Pregunta 5 (6 puntos)

a) *Indique en qué consisten las tecnologías de transmisión "full duplex" y "half duplex".*
b) *Considerando la respuesta a anterior, especifique y justifique cual es la modalidad utilizada por las siguientes tecnologías:*
   Ethernet (familia de protocolos IEEE 802.3).
   Bluetooth (protocolo IEEE 802.15).

a) Full duplex: la tecnología permite que se transmitan y reciban datos simultáneamente.
Half duplex: la tecnología permite que se transmita y reciban datos pero no de manera simultánea.

b) Ethernet: tecnología que surgió como half duplex pero que en su evolución (FastEthernet, Gigabit Ethernet, 10 Gigabit Ethernet) ha incorporado la posibilidad de trabajar en modo full duplex (lo usual en estos tiempos).
Bluetooth: Formalmente es half duplex. Como trabaja con Time Division Duplex donde el maestro y los esclavos se alternan en el uso del medio utilizando ranuras de tiempo de 625 us a veces, con poca rigurosidad, se indica que es full duplex.

Problema 1 (30 puntos)

a) La estructura es global a todos los hilos, por lo que debe ser mutuo excluida al momento de actualizarla. Las constantes no se actualizan

```
//-------------------------------------------------------
// DEFINICIÓN DE CONSTANTES Y TIPOS
//-------------------------------------------------------
#define      UNREACHABLE   -1
#define      Nmax                10

#define      TIME_OUT            10
#define      RETRAY       3

Typedef struct {
      char   *host_repos;
      int    port
      int    response_time;
} TRepo;

Repos = array [0..(Nmax-1)] of TRepo;
```

b)

```
// Consideramos que la estructura ya tiene cargados los repositorios

// Función main que realiza ambos pasos:
//     1 – actualiza la tabla de tiempos
//     2 – realiza el update
main () {

pthread_t thread_id[Nmax];

       // Lanza la ejecución de los hilos
       for (int i=0; i<Nmax;i++) {
         pthread_create(&thread_id[i],NULL,test_velocidades(),(void *) i);
       }

         // Espera a que todos terminen para comenzar el update
         for(int j=0; j<Nmax;j++) {
                  pthread_join( thread_id[j], NULL);
         }

       // Realiza el update
       get_and_install_update()

}
// FIN MAIN


//----------------------------------------
// Código ejecutado por cada thread
// Toma como parámetro el indice para identificar el
// repositorio que debo testear.
void *test_velocidades (void *idexPtr) {

bool  termino=false;
int   start_time = time();


// Me conecto con el repositorio
socket socketServidor = create_socket();
socketServidor.connect(MisRepos[(int*)indexPtr]. host_repos,
                       MisRepos[(int*)indexPtr]. port);

// Hago el pedido HTTP
send(socketServidor, “GET /test/test_speed.gz”);

// Espero hasta completar la bajada o salir por inactividad
while (!termino) {

    // Espero eventos en el socket
    select (socketServidor, TIME_OUT) {

        case socketServidor:
            if (segment == null) {
                // el servidor cerró el socket.
                // La funcion que sigue hace esto, pero con semáforos
                //MisRepos[(int*)indexPtr]. response_time= start_time-time();
                  actualizoTiempo(start_time-time());
                  termino=true;
            } else {
                // Consumo otra parte del archivo.
                // no es necesario guardarlo.
                char[] segmento = receive(socketServidor);
            }
            Break;
        Timeout: // Marco el repositorio inalcanzable
            actualizoTiempo(UNREACHABLE);
            termino=true;
            break;
     }
}
```

```
// Función para instalar el update
void get_and_install_update() {

bool updated=false;
bool timedout=false;

// Mientras no haya hecho el update y estoy dentro del
// número de retransmisiones posibles.
for (int i = 0; (i< RETRAY) && (!updated); i++) {

    socket socketServidor = create_socket();
    // obtenerReposHostPorVelocidad(i) y obtenerReposPortPorVelocidad(i)
    // me dan el host y puerto del repositorio i-ésimo según la velocidad.
    socketServidor.connect(obtenerReposHostPorVelocidad(i),
                           obtenerReposPortPorVelocidad(i));

    // Pido el archivo de update
    send(socketServidor, "GET /update/new.bin");

    // Mientras no logré hacer el update y no salí por timeout
    while (!updated and !timedout) {

        select (socketServidor, TIME_OUT) {

            case socketServidor:
                if (segment == null) { // el servidor cerró el socket
                    // La funcion que sigue hace esto, pero con semáforos
                    //MisRepos[(int*)indexPtr]. response_time=
                    //                                    start_time-time();
                    updateReposResponseTime((int*)indexPtr,start_time-time())
                    updated=true;
                } else {
                    // Consumo otra parte del archivo
                    char[] segmento = receive(socketServidor);
                    // lo agrego al archivo .bin en mi filesystem
                    appendToFile(ARCH_BIN,segmento);
                }
                Break;
            Timeout:
                timedout=true;
                break;
       } // Fin select
    } // Fin while
} // Fin for

// Si completé la bajada del archivo hago la instalación
If (updated) {
    installUpdate(ARCH_BIN); // Hace la instalación en el SO
}

} // Fin  get_and_install_update
```

c)
No hay identificación o autenticidad del repositorio
No hay MD5 del archivo de update o del de test (es mas grave en el segundo caso)
Se puede achicar el tamaño del archivo de test, hacer con esto ser electo primero y luego ofrecer un .bin con código malicioso

d)
Usar certificados digitales para autenticar al repositorio, o tener un sitio oficial de los repositorios oficiales.
Generar hashes de manera de ver que el contenido no ha sido variado, y de alguna manera tener una autoridad extra que sea la que tiene el MD5, de lo contrario yo pongo el MD5 de mi programa malicioso e igual coincide (integridad).
Para el archivo de test se podrían tomar medidas similares.

Problema 2 (30 puntos)

a) i) Dada la especificación, para que un host de la RED A se conecte con el Web Server (IP 192.168.4.45), debe conectarse a una IP de la propia red A (IP 92.168.1.253). Todos los hosts de la RED A que se conecten con el Web Server, inicialmente (tabla de ARP vacia), enviarán un ARP Request (broadcast) preguntando por la dirección MAC asociada a la dirección IP 192.168.1.2, el router R le deberá responder al host correspondiente (ARP reply) con su MAC en la RED A (00:03:3:CB:BC:19), que es la asociada a la dirección IP 192.168.1.253.

En resumen, el router R deberá responder a todo ARP request (Red A) de la MAC asociada a la dirección IP 192.168.1.253, con un ARP reply conteniendo su dirección MAC de la red A (00:03:3:CB:BC:19)

ii) Cuando el Host A se comunica con el Web Server y la tabla de ARP está vacia, envia un ARP:

MAC origen: 00:15:60:AA:49:FA
MAC destino: FF:FF:FF:FF:FF:FF
Tipo de ARP: ARP Request preguntando por la MAC asociada a la dirección IP 192.168.1.253

El router R le responde al HOST A:

MAC origen: 00:03:3:CB:BC:19
MAC destino: 00:15:60:AA:49:FA
Tipo de ARP: ARP Reply respondiendo que la MAC asociada a la dirección IP 192.168.1.253 es 00:03:3:CB:BC:19

b) i) Para los paquetes originados en un host de la RED A y con destino la dirección IP 192.168.1.253, el Router R deberá cambiar en el encabezado IP, la dirección IP de destino, sustituyendo 192.168.1.253 por 192.168.4.45 (IP real del Web Server).

Para los paquetes originados en el Web Server (IP 192.168.4.45) y con destino a hosts de la RED A, el Router R deberá cambiar en el encabezado IP, la dirección IP de origen, sustituyendo 192.168.4.45 por 192.168.1.253.

ii) Cuando el router R recibe un paquete de datos orignado en el host A (dirección IP origen 192.168.1.20) y con destino la dirección IP 192.168.1.253, cambia en el encabezado IP, la dirección de IP destino, sustituyendo 192.168.1.253 por 192.168.4.45.

Cuando el router R recibe un paquete de datos orignado en el Web Server (dirección IP origen 192.168.4.45) y con destino la dirección IP 192.168.1.20 (host A), cambia en el encabezado IP, la dirección de IP origen, sustituyendo 192.168.4.45 por 192.168.1.253.