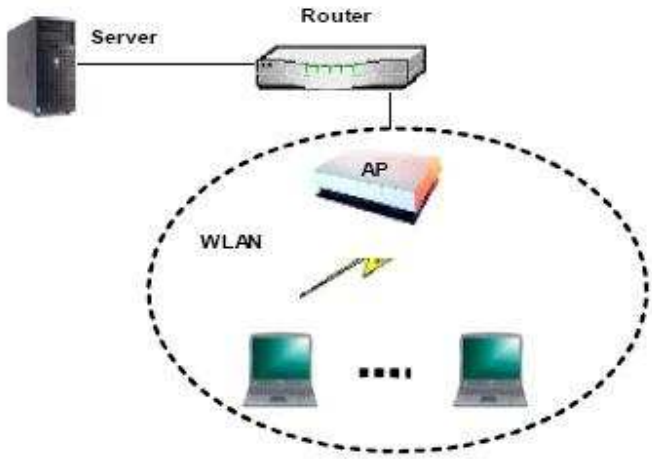
Examen de Introducción a las Redes de Computadoras y Comunicación de Datos (ref: sirc0503.doc) 28 de febrero de 2005

***Atención**: para todos los ejercicios, suponga 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).*

Ejercicio 1

En una red TCP/IP hay un host móvil inalámbrico conectado a una Estación Base (AP) que desea acceder a un servidor (conectado a la red cableada). Dicha conexión inalámbrica (a través de la WLAN – Wireless LAN) tiene una tasa de pérdida de paquetes muy alta, comparada con la de la red cableada.



Se pide:

a) Describa cual es el principal problema que ocurre con la tasa efectiva de transmisión, y considerando el comportamiento del TCP respecto a la ventana de congestión, entendiendo que el protocolo considerará las pérdidas como congestión en la red, y no como característica de la red WLAN. Suponga además que puede modificar los parámetros del TCP, que se ejecuta sobre la red WLAN.

b) Escribir en un lenguaje de alto nivel una solución que funciona en la Estación Base (AP), que resuelva el problema planteado, para el flujo en el sentido *server -> host móvil inalámbrico*.

c) ¿Que ideas propone para resolver el problema del flujo TCP en el sentido *host móvil inalámbrico -> server*?

Solución:

a) Describa cual es el principal problema que ocurre con la tasa efectiva de transmisión, y considerando el comportamiento del TCP respecto a la ventana de congestión, entendiendo que el protocolo considerará las pérdidas como congestión en la red, y no como característica de la red WLAN. Suponga además que puede modificar los parámetros del TCP, que se ejecuta sobre la red WLAN.

El problema que ocurre en TCP sobre una red inalámbrica con las características descriptas, es que al perder paquetes el emisor considera que se debe a congestionamiento en la red, y disminuye la velocidad de envío (disminuyendo la ventana de envío del TCP).

Esto si se repite en forma sistemática provocará que la conexión se vuelva más lenta, sin reflejar la capacidad de transmisión de la red.

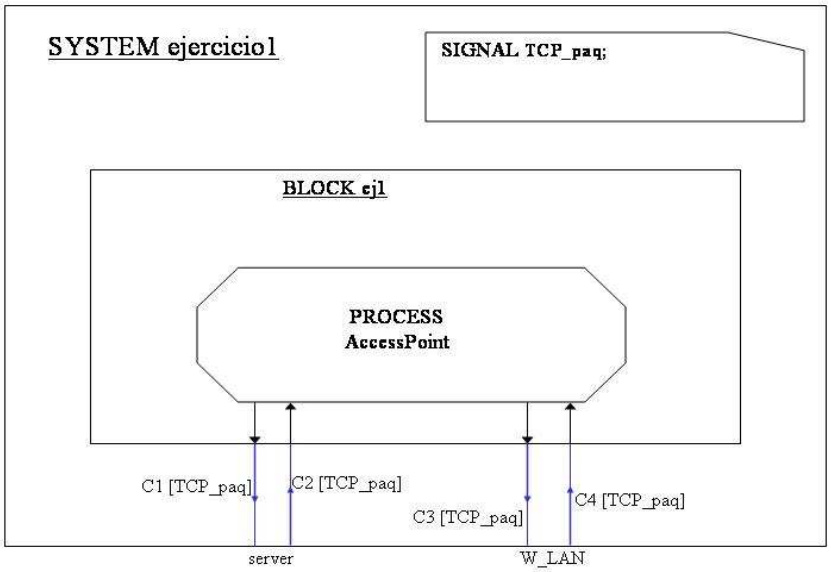
b) Escribir en un lenguaje de alto nivel una solución que funciona en la Estación Base (AP), que resuelva el problema planteado, para el flujo en el sentido *server -> host móvil inalámbrico*.

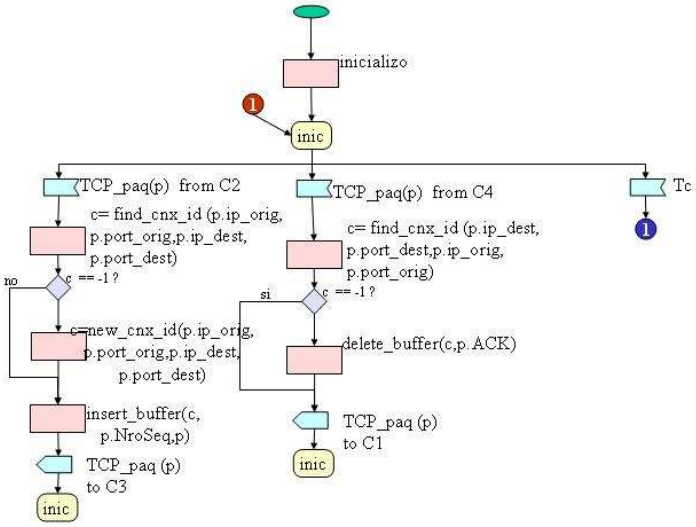
Se debe tomar cual es el Nro_de_Secuencia de los paquetes que se reciben desde el *server*, y si transcurrido cierto time_out, no se ha recibido el ACK del mismo, debe reiterarse el envío. Para esto se almacenan en un buffer todos los paquetes recibido.

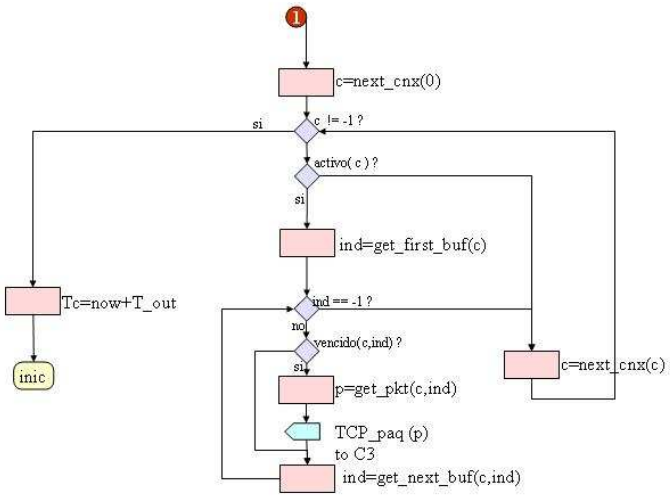
Se utilizará una estructura, donde se almacena un puntero al segmento recibido desde el *server*, y contiene además un time_stamp de la hora de recibido, y de su Nro_de_Secuencia. Al recibir un paquete desde el *host movil inalámbrico*, se eliminarán del buffer los segmentos que figuran como aceptados.

Para la resolución del problema utilizo una estructura que permita procesar lo expresado en los párrafos anteriores. El procedimiento a seguir es el siguiente:

- Si se recibe un paquete del server, se guarda en el buffer, incluyendo el tiempo de recibido, y actualizando el tiempo de llegada del último paquete, y se envía al host móvil
- Si se recibe un paquete del host móvil, se toma su ACK, se eliminan del buffer los paquetes recibidos ya aceptados, y se envía al server.
- Cada cierto tiempo, se procesa todo el buffer, re-enviando los paquetes de los cuales no se ha recibido ACK, y eliminando las conexiones de las cuales hace tiempo no se reciben paquetes.




inicializo
1
inic
TCP_paq(p) from C2
TCP_paq(p) from C4
Tc
c= find_cnx_id (p.ip_orig,
p.port_orig,p.ip_dest,
p.port_dest)
c == -1 ?
no
c=new_cnx_id(p.ip_orig,
p.port_orig,p.ip_dest,
p.port_dest)
insert_buffer(c,
p.NroSeq,p)
TCP_paq (p)
to C3
inic
c= find_cnx_id (p.ip_dest,
p.port_dest,p.ip_orig,
p.port_orig)
si
c == -1 ?
delete_buffer(c,p.ACK)
TCP_paq (p)
to C1
inic
1


1
c=next_cnx(0)
si
c != -1 ?
activo(c) ?
si
ind=get_first_buf(c)
Tc=now+T_out
inic
ind == -1 ?
no
vencido(c,ind) ?
si
p=get_pkt(c,ind)
TCP_paq (p)
to C3
ind=get_next_buf(c,ind)
c=next_cnx(c)

Funciones Auxiliares Utilizadas:

```
Type indice: struct of{
        port_orig: long;
        IP_orig: IP_ADDRESS;
        port_dest: long;
        IP_dest: IP_ADDRESS;
        }

Type buffer_cnxs: array [0..MAX_BUFF] of{
        time_received: timestamp;
        nroSeq: long;
        packet: char *;
        }

Type tabla_cnxs: array [0..MAX_CNX] of{
        cnx_ID: indice;
        last_time_received: timestamp;
        last_nroSeq: long;
        last_ACK: long;
        first_buff_id: 0..MAX_BUFF;
        last_buff_id: 0..MAX_BUFF;
        buffer_cnx: *buffer_cnxs;
        }


function find_cnx_id(p_org: long;IP_org: IP_ADDRESS;p_dst: long;IP_dst: IP_ADDRESS):0..MAX_CNX{
        encontre:=FALSE;
        ind:=( (p_org,IP_org,p_dst,IP_dst);
        resultado=-1;
        c:=0;
        while (not encontre && c <= MAX_CNX ){
                if (tabla_cnxs[c]. cnx_ID == ind ){
                        encontre=TRUE;
                        resultado:=c;
                }
                c++;
        }
        return (resultado);
}


function new_cnx_id(p_org: long;IP_org: IP_ADDRESS;p_dst: long;IP_dst: IP_ADDRESS):0..MAX_CN{
        // Devuelve el próximo lugar vacío en la tabla de conexiones
        encontre:=FALSE;
        ind:=( (p_org,IP_org,p_dst,IP_dst);
        c:=0;
        while (not encontre && c <= MAX_CNX ){
                if (tabla_cnxs[c]. cnx_ID == NULL ){
                        encontre=TRUE;
                        tabla_cnxs[c]. cnx_ID = ind;
                        resultado:=c;
                }
                c++;
        }
        return (resultado);
}
```

```
procedure insert_buffer(c:0..MAX_CNX,p.NroSeq:long,p char *){
        // almacena un paquete el la estructura de buffers
        // incluyendo su número de secuencia.
        if tabla_cnxs[c].first_buff_id <> tabla_cnxs[c].last_buff_id {
                // Hay espacio para guardar el paquete
                // Guardo el paquete en la estructura de los buffers
                tabla_cnxs[c].buffer_cnx[last_buff_id].packet=p;
                tabla_cnxs[c].buffer_cnx[last_buff_id].time_received=time;
                tabla_cnxs[c].buffer_cnx[last_buff_id].nroSeq=p.NroSeq;
                tabla_cnxs[c].last_buff_id=(tabla_cnxs[c].last_buff_id++)mod MAX_BUFF;
                tabla_cnxs[c].last_nroSeq=p.NroSeq;
                tabla_cnxs[c].last_time_received=time;
        }
}


procedure delete_buffer(c:0..MAX_CNX,p.ACK:long){
        // Borra los elementos del buffer hasta cierto ACK,
        // el motivo en general es la llegada del ACK correspondiente.
        while tabla_cnxs[c].buffer_cnx[first_buff_id].nroSeq < p.ACK &&
              tabla_cnxs[c].first_buff_id <> tabla_cnxs[c].last_buff_id {
                // elimino buffers ya aceptados en el presente ACK
                tabla_cnxs[c].first_buff_id=(tabla_cnxs[c].first_buff_id++)mod MAX_BUFF
        }
        tabla_cnxs[c].last_ACK=p.ACK
}



function next_cnx(c:0..MAX_CNX):-1..MAX_CNX{
        // devuelve la próxima conexión utilizada en la tabla de conexiones tabla_cnxs
        encontre:=FALSE;
        resultado:=-1;
        while (not encontre && c <= MAX_CNX ){
                if (tabla_cnxs[c]. cnx_ID == NULL ){
                        encontre:=TRUE;
                        resultado:=c;
                }
                c++;
        }
        return (resultado);
}


function get_first_buf(c:0..MAX_CNX):-1..MAX_BUFF{
        // devuelve el primer buffer todavía activo
        if (tabla_cnxs[c].first_buff_id <> tabla_cnxs[c].last_buff_id )
                return (tabla_cnxs[c].first_buff_id);
        else
                return -1;
}


function activo(c:0..MAX_CNX):boolean{
        // Verifica que la conexión este en uso,
        // para esto utiliza el tiempo de llegada del último paquete.
        // En caso de estar inactiva elimina la conexión.
        if (time - tabla_cnxs[c].last_time_received < Time_OUT_CNX){
                return TRUE;
        else{
                tabla_cnxs[c]. cnx_ID = NULL;
                return FALSE;
        }
}
```

```
function vencido(c:0..MAX_CNX,ind:0..MAX_BUFF):boolean{
        // Verifica el tiempo del paquete en el buffer
        // En caso de estar vencido, se retransmitirá.
        if (time - tabla_cnxs[c].buffer_cnx[ind].time_received > Time_OUT)
                return TRUE;
        else
                return FALSE;
}


function get_pkt(c:0..MAX_CNX,ind:0..MAX_BUFF):char *{
        // devuelve paquete a retransmitir
        return (tabla_cnxs[c].buffer_cnx[ind].packet);
}


function get_next_buf(c:0..MAX_CNX,ind:0..MAX_BUFF):-1..MAX_BUFF{
        // devuelve el primer buffer todavía activo
        if ( (ind + 1) <> tabla_cnxs[c].last_buff_id )
                return (ind + 1);
        else
                return -1;
}
```

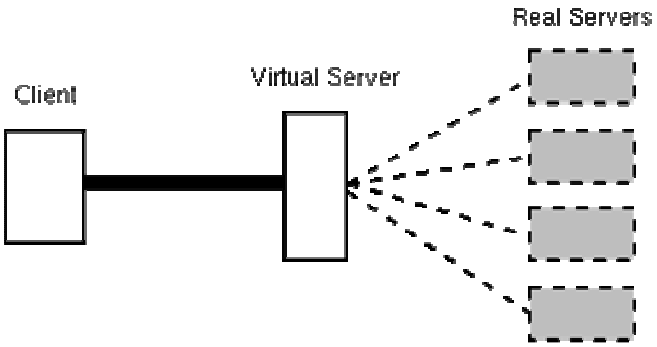
c) ¿Que ideas propone para resolver el problema del flujo TCP en el sentido *host móvil inalámbrico -> server*?

Para las posibles soluciones en el sentido *host móvil inalámbrico -> server*, se debe tener en cuenta que no se puede utilizar una técnica de almacenar en buffers los paquetes llegados, ya que la pérdida en este caso corresponde al primer tramo de la conexión, por lo que el Access Point no recibiría nada, y no estaría en capacidad de almacenar en buffers.

Las soluciones se plantean por la modificación del TCP, utilizando políticas de retransmisión selectiva (selective repeat), que permitan seguir almacenando los datos recibidos, pero que permitan informar al origen que existen paquetes faltantes en el flujo.

Ejercicio 2

Se desea implementar un servicio virtual de WEB, el que redirige las solicitudes de los clientes a diferentes servidores (Real Servers), y que busca que para cierto cliente se redirija al mismo Real Server (como se ve en la figura).



Se pretende lograr que el servicio sea transparente para el cliente.

Para esto se cuenta con las siguientes herramientas aplicables en la búsqueda de la solución:

- un servidor WEB podrá incluir la siguiente línea en el texto devuelto, que le permite definir una variable y asignarle un valor. Deberá incluir::

  `Set-Cookie``: NAME_VAR=VALUE_VAR; path=/path_de_url; expires=fecha_expiracion`

  Al recibir el browser esta línea (por ejemplo llamando http://web_url/path_de_url/archivo1_url), almacena en una tabla interna un registro con la siguiente información:

  - Nombre= NAME_VAR
  - Contenido= VALUE_VAR
  - Host= web_url
  - Path= path_de_url
  - Expira= fecha_expiracion

  y llegada la fecha de expiración, el registro será borrado.

- Posteriormente, cada vez que un cliente realice en el browser (con el registro ya almacenado) la solicitud de un archivo por ejemplo http://web_url/path_de_url/archivo2_url, se incluye en forma transparente la siguiente línea en el texto:

  `Cookie``: NAME_VAR=VALUE_VAR`

Se pide:

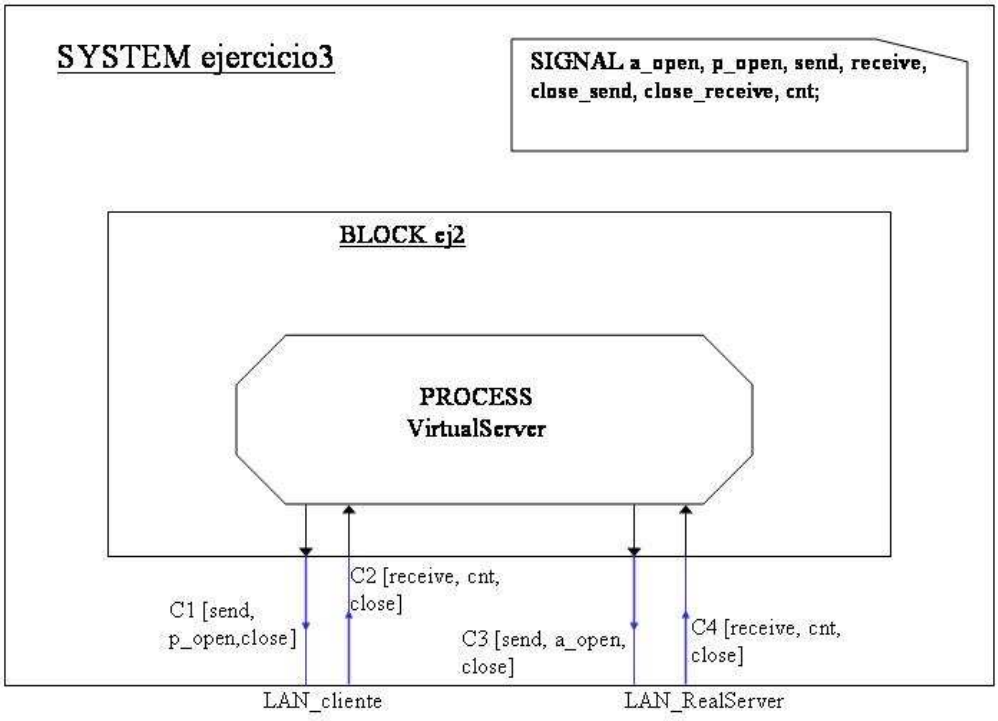
Escribir en un lenguaje de alto nivel, un proceso que se ejecuta en el Virtual Server, que permite redirigir las solicitudes llegada siempre al mismo Real Server. El proceso podrá modificar el texto devuelto por la consulta realizada a un Real Server, para entregarse al cliente.

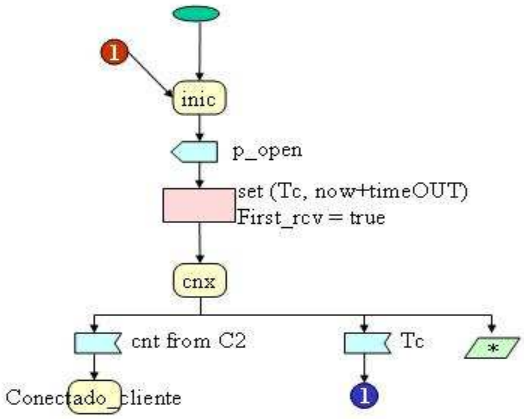
Solución:

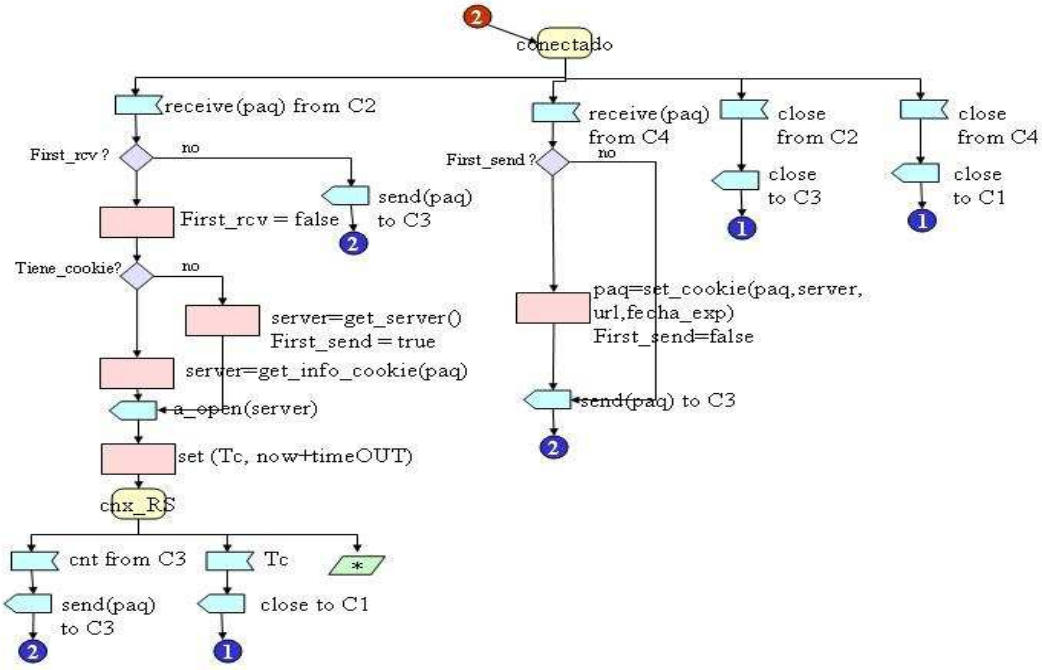
La idea propuesta es la siguiente, el Virtual_Server recibe conexiones de los clientes, y en la solución propuesta se muestra un proceso que atiende cada conexión. La idea planteada en caso de estar atendiendo en simultaneo varias conexiones, se comportará con un esquema de fork, que genera un proceso para cada atención.

Para la misma analiza si tiene un cookie seteado (el que en la solución planteada viene en el primer paquete), y se actúa de la siguiente manera:

```
Si tiene_cookie
      Redirijo la conexión al servidor especificado en la cookie
Sino
      Busco un servidor y redirijo la conexión
```




1
inic
p_open
set (Tc, now+timeOUT)
First_rcv = true
cnx
cnt from C2
Tc
*
Conectado_cliente
1

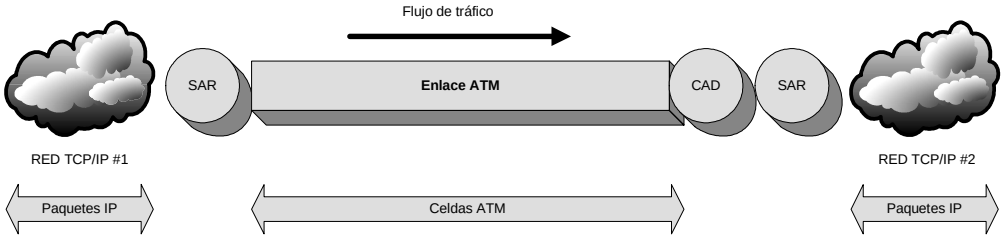

2
conectado
receive(paq) from C2
First_rcv ?
no
send(paq)
to C3
2
First_rcv = false
Tiene_cookie?
no
server=get_server()
First_send = true
server=get_info_cookie(paq)
a_open(server)
set (Tc, now+timeOUT)
cnx_RS
cnt from C3
Tc
*
send(paq)
to C3
2
close to C1
1
receive(paq)
from C4
First_send ?
no
paq=set_cookie(paq,server,
url,fecha_exp)
First_send=false
send(paq) to C3
2
close
from C2
close
to C3
1
close
from C4
close
to C1
1

Ejercicio 3

Se consideran dos redes TCP/IP que se comunican a través de un enlace WAN basado en tecnología ATM. Como se sabe, ATM transmite datos en paquetes de largo fijo (53 bytes, compuestos de 48 bytes de datos y 5 bytes de header) llamados *celdas*. A los efectos del presente ejercicio, lo que nos interesa saber de ATM es que en el enlace a nivel de capa 2 se inserta en ambos extremos dos procesos llamados SAR (*Segmentation and Reassembly*). El proceso SAR se encarga de tomar los paquetes IP a su ingreso, fragmentarlos en celdas de 48 bytes, agregarle los headers correspondientes y entregarlos a la capa física para su transmisión por el medio. El SAR del extremo de arribo realiza la función inversa.

A su vez, en el extremo de "llegada" de datos, se inserta otro proceso anterior al SAR, que llamaremos CAD (*Control de Admisión*), el cual se encarga de hacer cumplir el llamado *Contrato de Tráfico*. Este Contrato permite a un operador de telecomunicaciones vender ancho de banda de manera controlada y dinámica.

El modelo simplificado en un sentido de la red es entonces el siguiente:



El contrato de tráfico (simplificado en nuestro caso) se define en función de los siguientes parámetros:

- Un tiempo de medida, T_c (medido en segundos)
- Un ancho de banda comprometido, *SCR* (*Sustained Cell Rate*, medido en celdas/segundo)
- El ancho de banda **físico** del enlace, B_l (medido en bytes por segundo)

El Contrato dice entonces, que las celdas recibidas durante el tiempo de observación (T_c) que superen a las permitidas por el ancho de banda comprometido, serán descartadas. Observar que el SCR es un parámetro medido, el ancho de banda físico es B_l.

Se pide:

a) Proponer un algoritmo para realizar el CAD e implementarlo en alto nivel. Explicarlo relacionándolo con algoritmos similares presentes en la literatura de la materia.

b) Calcule cuanto tiempo demora aproximadamente en transmitirse un paquete IP de largo *L* para un cierto contrato de tráfico. ¿En que condiciones existe descarte de celdas?

c) Discuta el efecto que tiene el descarte de celdas para la comunicación TCP/IP propuesta, en particular sobre las conexiones TCP que puedan haber.

Solución:

a) Proponer un algoritmo para realizar el CAD e implementarlo en alto nivel. Explicarlo relacionándolo con algoritmos similares presentes en la literatura de la materia.

El procedimiento que resuelve lo pedido es el siguiente:

```
var
      Nc: integer;      // Nro de celdas recibidas, se reinicia cada Tc
Const
      SCR: integer;
      Tc: integer;
      Bl: integer;

Procedure CAD;
Begin
      setTimer (now() + Tc);
      while (trae) do
      begin
            if (celda_recibida) then
                  Nc:= Nc +1;
                  if (Nc <= SCR *Tc) then
                        toSAR();    // envía la celda al SAR
                  else
                        discard();  // descarto la celda y envío alguna cosa
                  end;
            end;
            if (timer expiró) then
                  Nc:=0;
                  setTimer(now() + Tc);
            end;
      end;
```

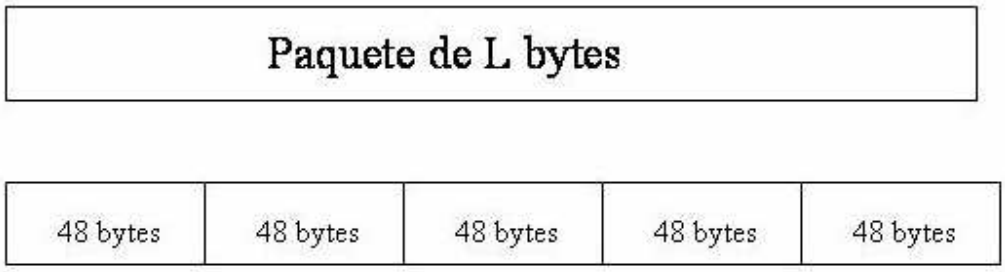
- Este algoritmo se asocia a los algoritmos de “token bucket” y “leaky bucket”. A diferencia de estos, no se acumula el crédito de un intervalo al siguiente

a) Calcule cuanto tiempo demora aproximadamente en transmitirse un paquete IP de largo *L* para un cierto contrato de tráfico. ¿En que condiciones existe descarte de celdas?

Un paquete de L bytes demora:

t_0= ([L/48] * 53) / Bl

- Ya un paquete de L bytes, y tiene 48 bytes/celda ➔ son [L/48] celdas,
- La velocidad de transferencia es Bl bytes/segundo ➔ cada celda tarda en enviarse 53 / Bl
- Por lo que tarda t_0 segundos



El descarte ocurre cuando dentro de un Tc se superan la cantidad de SCR * Tc celdas

¿Cuántas celdas pasan en Tc?

N_0 = Tc / (53 / Bl) =

Ya que:

- 53 / Bl es el tiempo de una celda
- Tc es el período que se quiere considerar.

Entonces:

N_0 = (Tc * Bl) / 53 =

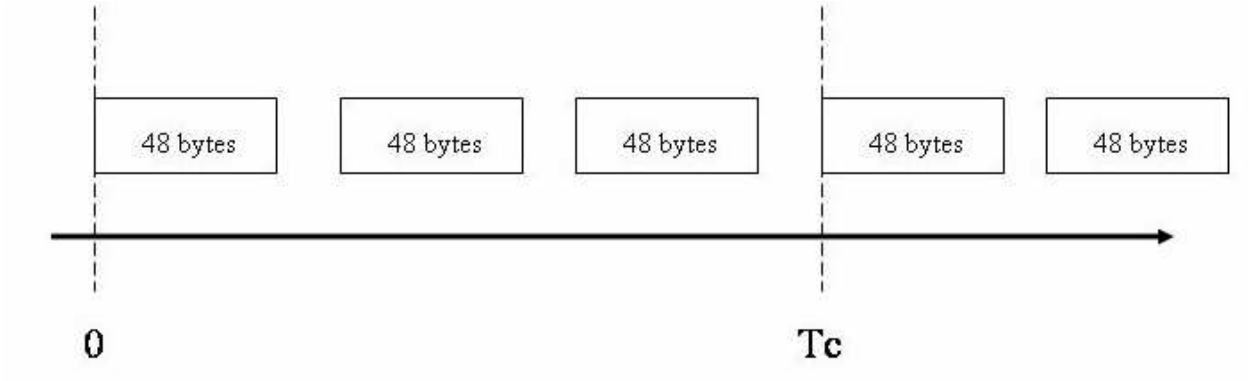
Por lo que descarte habrá cuando:

N_0 >= SCR * Tc

N_0 = (Tc * Bl) / 53 >= SCR * Tc

(~~Tc~~ * Bl) / 53 >= SCR * ~~Tc~~

Bl >= SCR * 53



b) Discuta el efecto que tiene el descarte de celdas para la comunicación TCP/IP propuesta, en particular sobre las conexiones TCP que puedan haber.

El descarte de una celda de un paquete IP, provoca el descarte de todo el paquete IP afectado. Esto afecta a las conexiones TCP, en el sentido de que se debe retransmitir paquetes para una misma conexión bajando la eficiencia de los mismos.

Hay que observar que aunque se descarte el paquete IP por una celda que ha sido descartada, las otras celdas fueron igualmente transmitidas, lo que indica que afecta negativamente en la eficiencia del enlace.