UNIVERSIDAD TECNOLÓGICA DE MÉXICO

Campus Cuitláhuac

POSGRADO EN REDES DE COMPUTADORAS

Periodo 00−3

DESARROLLO DE PROTOCOLOS DE COMUNICACIÓN

TAREA 2

Martes, 24 de Noviembre de 2000

Rlogin

Se utiliza para abrir una sesión en una máquina remota en su forma más simple que usted puede pulsar la máquina del rlogin que procurará a la conexión adentro a la máquina remota usando el mismo nombre del utilizador que usted tiene en la máquina local. Si su nombre del utilizador es diferente en la máquina remota entonces usted tiene que utilizar la máquina del rlogin − l username.

RCP

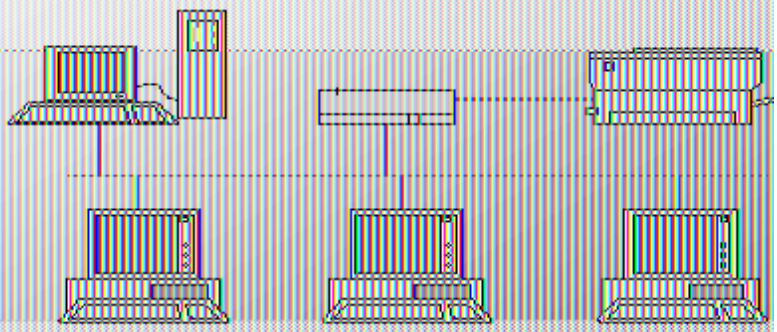
El RCP se utiliza para copiar ficheros entre las máquinas, la sintaxis es similar a machine:file del RCP del cp o el comando del RCP machine:file copiar un fichero de la máquina local a una máquina remota o copiar un fichero alejado a la máquina local. Si su nombre del usuario es diferente en una de las máquinas entonces puedes utilizar el comando user@machine:file del RCP.

RSH

Se utiliza para ejecutar un programa en una máquina remota, si el programa tiene argumentos entonces tienes que poner sentencias ej. la máquina ' ls − l ' del rsh. si su username es diferente en la máquina remota entonces rsh − l username del programa de la máquina destino.

Redes de Comput

Capa de Enlace de D



Protocolo Simplex

- Situación 1: (con suposiciones..)
  - Comunicación Simplex
  - Capas de Red están siempre "listas"
  - Tiempo de Procesamiento despreciable
  - Buffer de tamaño infinito
  - Capa física no tiene problemas ni pierde m
  - No se enumeran los paquetes

Protocolo IRREAL !!

Protocolo Simplex irreal

```
typedef enum {frame_arrival} event_type;
#include "protocol.h"

void sender1(void)
{
  frame s;                          /* buffer for an outboun
  packet buffer;                    /* buffer for an outboun

  while (true) {
      from_network_layer(&buffer);          /* go get ser
      s.info = buffer;                      /* copy it into s for tran
      to_physical_layer(&s);                /* send it on its way */
                                            /* Tomorrow, and tomo
                                               Creeps in this petty p
                                               To the last syllable of
                                                    - Macbeth, V, v */
  }
}

void receiver1(void)
{
  frame r;
  event_type event;                 /* filled in by wait, but n

  while (true) {
      wait_for_event(&event);       /* only possibility is fran
      from_physical_layer(&r);      /* go get the inbound fr
      to_network_layer(&r.info);                /* pass the d
  }
}
```


Protocolo Simplex

- Situación 2:
  - Canal libre de errores (existirá?)
  - Buffer de tamaño limitado
    - TX no debe saturar al RX
    - Se usa Control de Flujo
    - RX acusa recibo de cada marco
    - Es un protocolo Half duplex.

Protocolo Simplex v2.0

```
typedef enum {frame_arrival} event_type;
#include "protocol.h"

void sender2(void)
{
  frame s;                    /* buffer for an outbound frame */
  packet buffer;              /* buffer for an outbound packet */
  event_type event;           /* frame_arrival is the only possibility */

  while (true) {
      from_network_layer(&buffer);          /* go get something to send */
      s.info = buffer;        /* copy it into s for transmission */
      to_physical_layer(&s);  /* bye bye little frame */
      wait_for_event(&event); /* do not proceed until given the go ahead */
  }
}

void receiver2(void)
{
  frame r, s;                 /* buffers for frames */
  event_type event;           /* frame_arrival is the only possibility */
  while (true) {
      wait_for_event(&event); /* only possibility is frame_arrival */
      from_physical_layer(&r); /* go get the inbound frame */
      to_network_layer(&r.info);/* pass the data to the network layer */
      to_physical_layer(&s);  /* send a dummy frame to awaken sender */
  }
}
```

Redes de Computadores
Capa de Enlace de Datos

Protocolo Simplex STOP & W

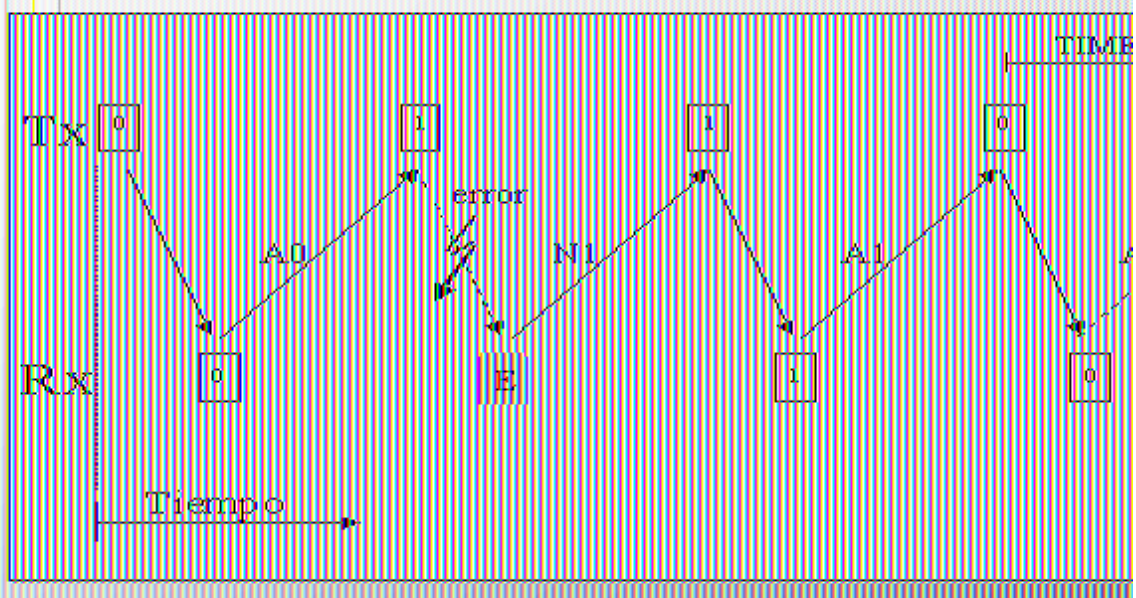
- Situación 3:
  - Idea básica: asegurarse de que cada
    transmitido es recibido correctament
    transmitir el siguiente.
  - Canal posee ruido
  - buffer limitado

Protocolo Simplex STOP & W

- Situación 3:
  - Cómo sabe el TX que el receptor recibió u
    - N-ack (Negative Awknowledge)
      - y si el marco no "alcanza" a llegar?
    - Temporizador
      - y si el acuse de recibo se perdió, se repetir
  - Solución: **Se enumeran los marcos !** ⟹
  - Basta enumeración módulo 2 (0 y 1)

Redes de Computadores
Capa de Enlace de Datos

Protocolo Simplex STOP & W



http://elqui.dcsc.utfsm.cl

Protocolo Simplex STOP & WAIT

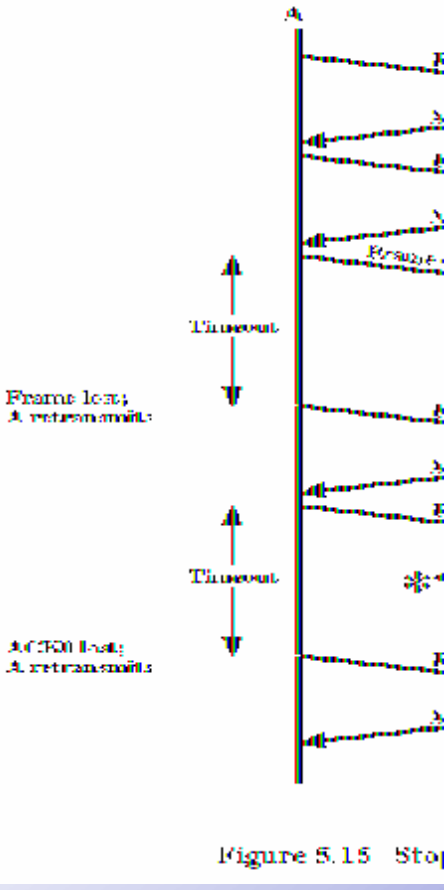


Figure 3.15 Sto

Redes de Computadores
Capa de Enlace de Datos

Protocolo Simplex STOP & WAIT

TX

```
#define MAX_SEQ 1                       /* must be 1 for protocol 3 */
typedef enum {frame_arrival, cksum_err, timeout} event_type;
#include "protocol.h"

void sender3(void)
{
  seq_nr next_frame_to_send;            /* sec number of next outgoing frame */
  frame s;                              /* scratch variable */
  packet buffer;                        /* buffer for an outbound packet */
  event_type event;

  next_frame_to_send = 0;               /* initialize outbound sequence numbers */
  from_network_layer(&buffer);          /* fetch first packet */
  while (true) {
      s.info = buffer;                  /* construct a frame for transmission */
      s.seq = next_frame_to_send;       /* insert sequence number in frame */
      to_physical_layer(&s);            /* send it on its way */
      start_timer(s.seq);               /* if answer takes too long, time out */
      wait_for_event(&event);           /* frame_arrival, cksum_err, timeout */
      if (event == frame_arrival) {
          from_physical_layer(&s);      /* get the acknowledgement */
          if (s.ack == next_frame_to_send) {
              from_network_layer(&buffer);/* get the next one to send */
              inc(next_frame_to_send); /* invert next_frame_to_send */
          }
      }
  }
}
```

RX Protocolo Simplex STOP & WAIT

```
void receiver3(void)
{
  seq_nr frame_expected;
  frame r, s;
  event_type event;

  frame_expected = 0;
  while (true) {
      wait_for_event(&event);                 /* possibilities: frame_arrival, cksum_err */
      if (event == frame_arrival) {           /* a valid frame has arrived. */
            from_physical_layer(&r);          /* go get the newly arrived frame */
            if (r.seq == frame_expected) {    /* this is what we have been waiting for. */
                  to_network_layer(&r.info); /* pass the data to the network layer */
                  inc(frame_expected);        /* next time expect the other sequence nr */
            }
            s.ack = 1 – frame_expected;       /* tell which frame is being acked */
            to_physical_layer(&s);            /* none of the fields are used */
      }
  }
}
```


Ejemplo de Protocolo de Ve

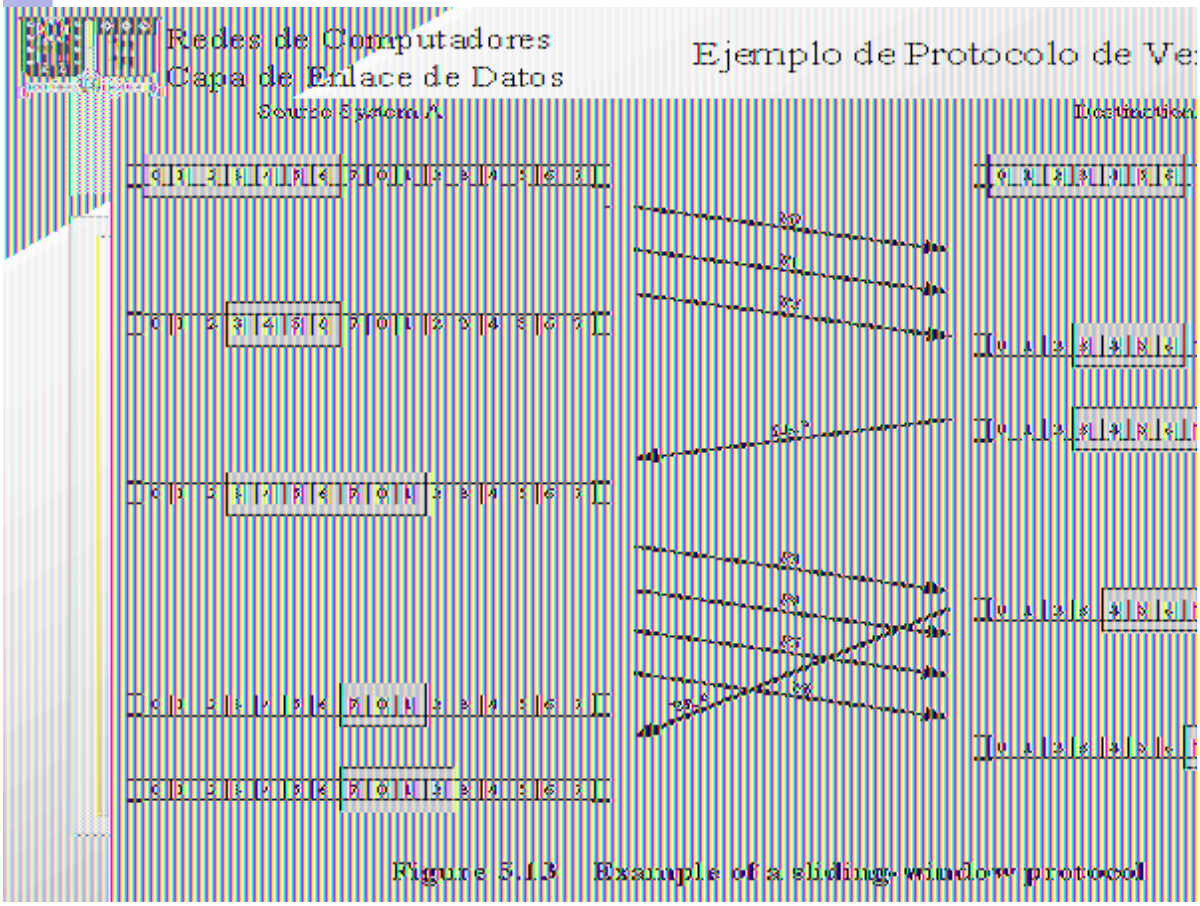
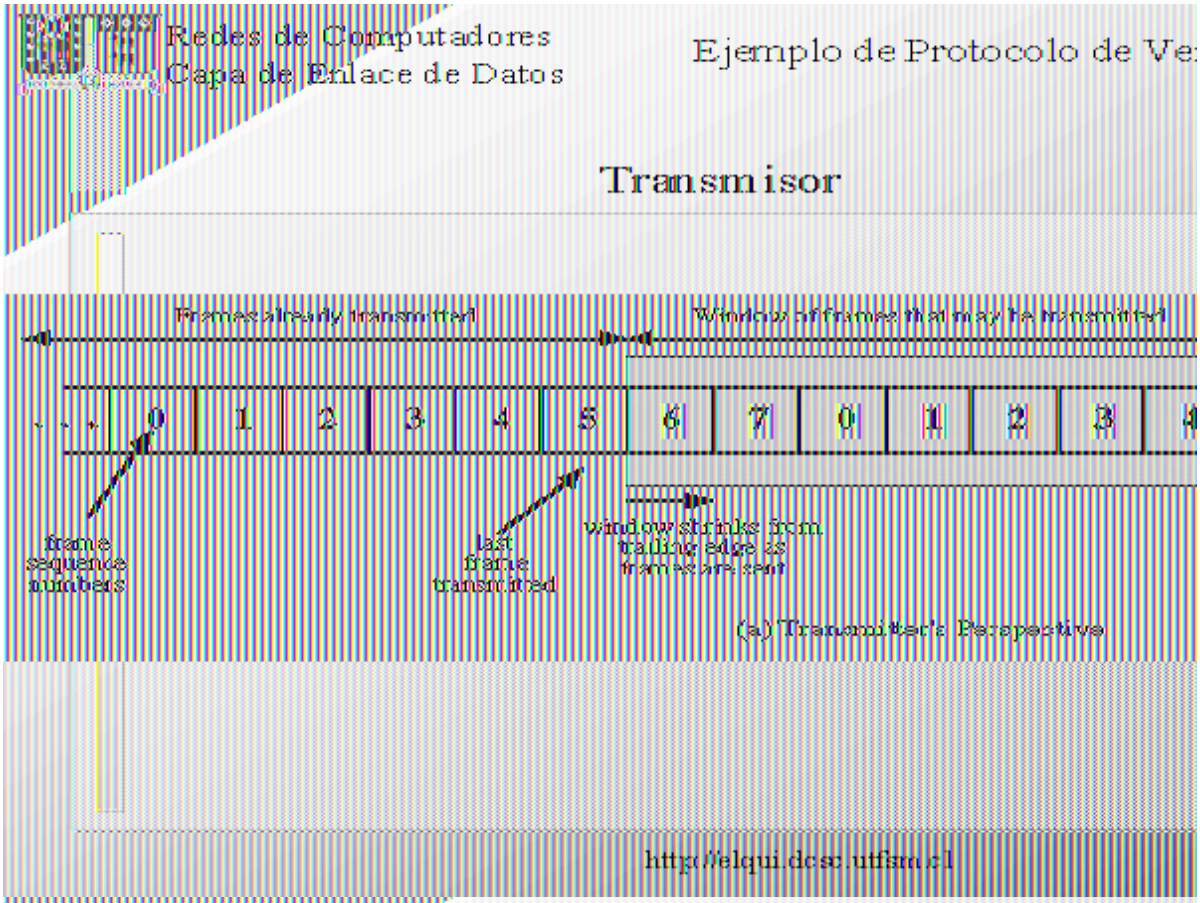
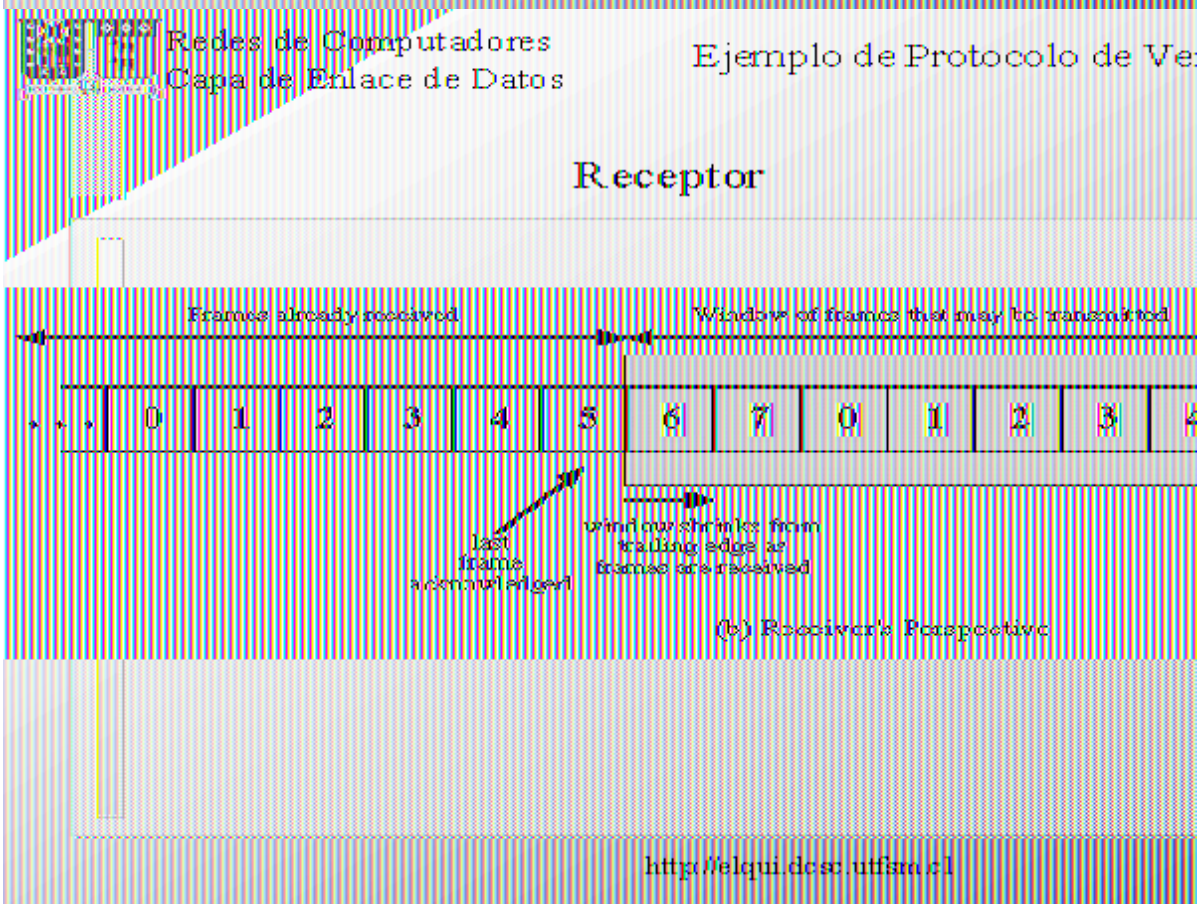


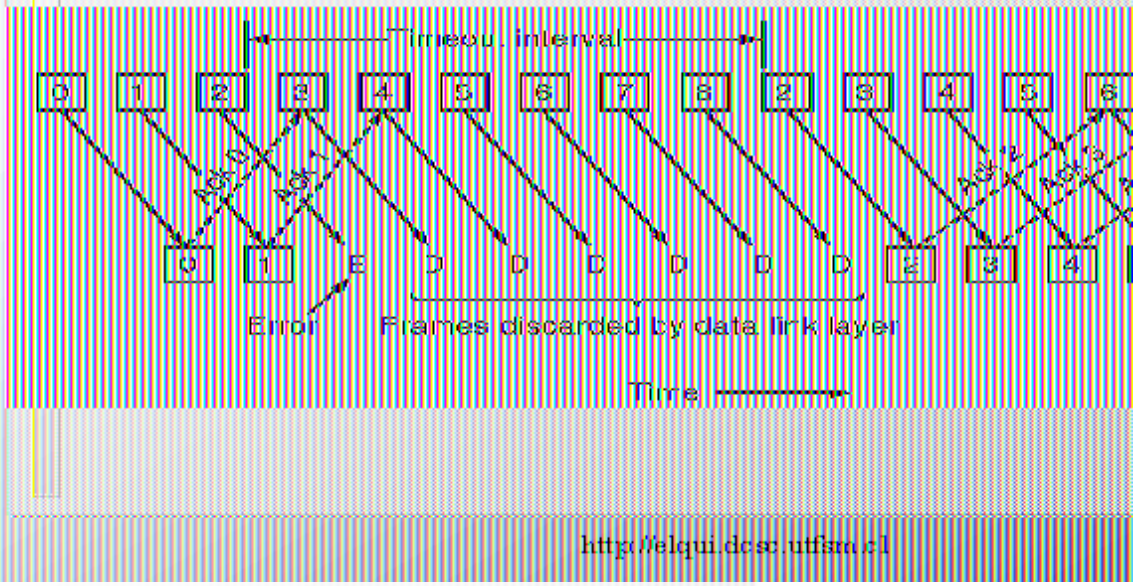
Figure 3.13 Example of a sliding-window protocol


Redes de Computadores
Capa de Enlace de Datos
Ejemplo de Protocolo de Ve
Transmisor
Frames already transmitted
Window of frames that may be transmitted
0
1
2
3
4
5
6
7
0
1
2
3
frame sequence numbers
last frame transmitted
window shrinks from trailing edge as frames are sent
(a) Transmitter's Perspective
http://elqui.dcsc.utfsm.cl


Redes de Computadores
Capa de Enlace de Datos
Ejemplo de Protocolo de Ve
Receptor
Frames already received
Window of frames that may be transmitted
0
1
2
3
4
5
6
7
0
1
2
3
last frame acknowledged
window shrinks from trailing edge as frames are received
(b) Receiver's Perspective
http://elqui.dcsc.utfsm.cl

Go Back-N

- Ventana TX=n, Ventana RX=1



Go Back-N

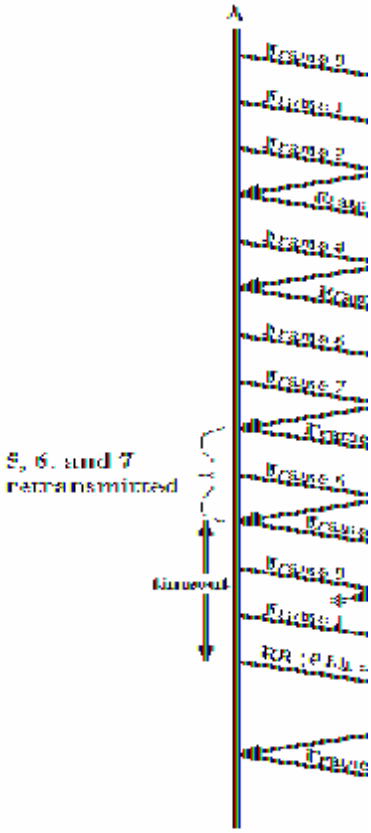
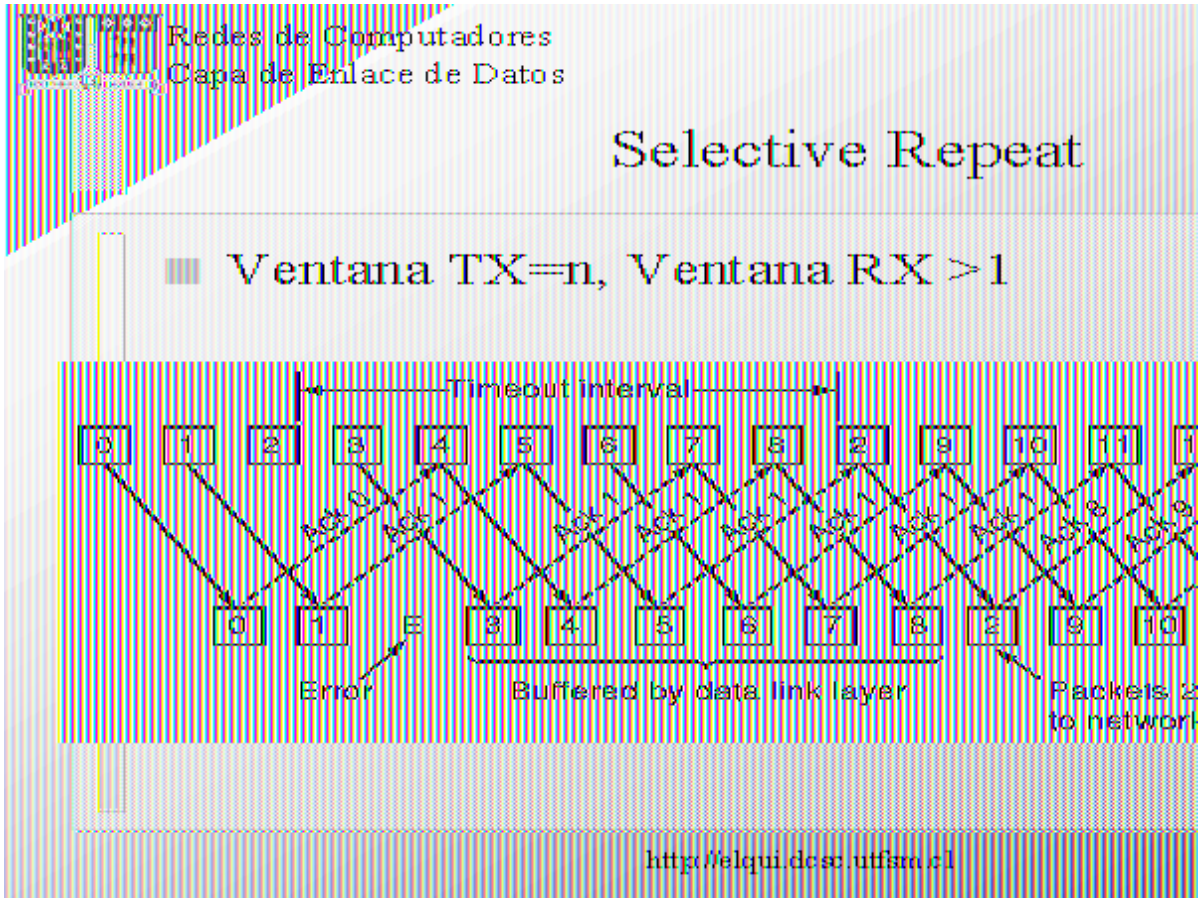


Figure 5.16



PROTOCOLO DE VENTANA CORREDIZA

Con este protocolo se tiene un Buffer en el cual se almacenan las copias de las tramas a enviar, esperando se envíe un ACK como respuesta.

La transmisión − recepción es consecutiva por lo que para cada elemento recibido se espera una confirmación, el tamaño de la trama se puede preestablecer o se negocia.