```
/*Protocolo 3. Permite el flujo unidireccional de datos por un canal no confiable*/

#define MAX_SEQ 1                       /*El numero de secuencia será 0 o 1*/
typedef enum {frame_arrival, cksum_err, timeout} event_type;
#include “protocol.h”

void sender3(void)
{
 seq_nr next_frame_to_send;             /*numero de secuencia de la siguiente trama de salida*/
 frame s;
 packet buffer;
 event_type event;

 next_frame_to_send = 0;
 from_network_layer(&buffer);
 while(true){
  s.info = buffer;
  s.seq = next_frame_to_send;
  to_physical_layer(&s);
  start_timer(s.seq);
  wait_for_event(&event);
  if(event == frame_arrival);{
   from_network_layer(&buffer);
   inc(next_frame_to_send);
  }
 }

}


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

 frame_expected = 0;
 while(true){
  wait_for_event(&event);
  if(event == frame_arrival){
   from_physical_layer(&r);
   if(r.seq == frame_expected){
    to_network_layer(&r.info);
    inc(frame_expected);
   }
   to_physical_layer(&s);
  }
 }

}
```

```
/*Protocolo 4 (de ventana corrediza) es bidireccional y más robusto que el protocolo 3*/

#define MAX_SEQ 1
typedef enum {frame_arrival, cksum_err, timeout} event_type;
#include “protocol.h”

void protocol4(void)
{
 seq_nr next_frame_to_send;          /*solo 0 o 1*/
 seq_nr frame_expected               /*solo 0 o 1*/
 frame r, s;
 packet buffer;
 event_type event;

 next_frame_to_send = 0;
 frame_expected = 0;
 from_network_layer(&buffer);
 s.info = buffer;
 s.seq = next_frame_to_send;
 s.ack = 1 – frame_expected;
 to_physical_layer(&s);
 start_timer(s.seq);
 while(true){
  wait_for_event(&event);
  if(event == frame_arrival);{
   from_physical_layer(&r);
   if(r.seq == frame_expected){
     to_network_layer(&r.info);
     inc(frame_expected);
   }
   if (r.ack == next_frame_to_send) {
     from_network_layer(&buffer);
     inc(next_frame_to_send);
   }
  }
  s.info = buffer;
  s.seq = next_frame_to_send;
  s.ack = 1 – frame_expected;
  to_physical_layer(&s);
  start_timer(s.seq);
 }
}
```