Giao thức cửa sổ trượt (Sliding windows)

Bởi:

unknown

Giao thức của sổ trượt (Sliding windows)

Vấn đề truyền tải thông tin theo hai chiều (Duplex)

Chúng ta muốn việc truyền tải thông tin giữa hai bên giao tiếp diễn ra một cách đồng thời theo hai chiều hơn là chỉ một chiều để khai thác tối đa khả năng của kênh truyền.

Để thực hiện được điều này, chúng ta thực sử dụng chế độ truyền tải hai chiều, gọi là song công (Duplex). Nguyên tắc thực hiện như sau:

Vẫn thực hiện việc truyền tải khung, tuy nhiên ta có phân biệt thành các loại khung: dữ liệu (data), báo nhận ACK (acknowledgement), và báo không nhận NACK(Not Acknowledgement) trong trường xác định loại (Type) của khung.

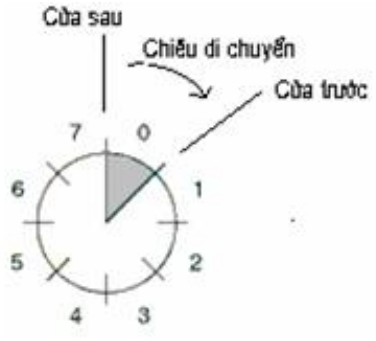
Khi một bên nào đó truyền tin, nó có thể kết hợp đưa thông tin báo cho bên kia biết tình trạng của gói tin mà nó đã nhận trước đó. Ta gọi là kỹ thuật **piggyback**.

Giới thiệu về giao thức cửa sổ trượt

Thay vì chỉ truyền đi một khung tại một thời điểm (simplex), giao thức cửa sổ trượt cho phép bên gởi có thể gởi đi nhiều khung.

Giao thức này sử dụng một cửa sổ để cho phép bên gởi theo dõi các khung mà nó được phép gởi đi và các khung mà nó đang chờ báo nhận, gọi là cửa sổ gởi (Sending Windows). Một cửa sổ khác để bên nhận theo dõi các khung mà nó được phép nhận, gọi là cửa sổ nhận (Receiving Windows).

Cấu trúc của cửa sổ được mô tả như sau:

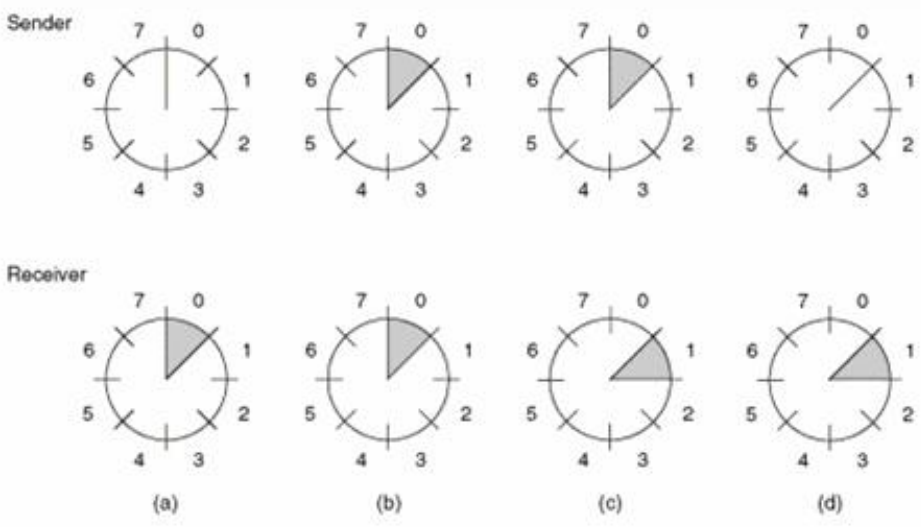


Cấu trúc cửa sổ trượt

- Phần tô đen là phạm vi của cửa sổ gồm có **cửa trước** và **cửa sau** cùng di chuyển theo một chiều.
- Kích thước của cửa sổ là chiều của cung giới hạn từ cửa sau đến cửa trước.
- Kích thước của cửa sổ có thể thay đổi. Khi cửa trước di chuyển, cửa sổ được mở rộng ra. Ngược lại khi cửa sau di chuyển, kích thước của cửa sổ bị thu hẹp lại và nó làm cho cửa sổ thay đổi vị trí, trượt / quay quanh một tâm của vòng tròn.
- Kích thước nhỏ nhất của cửa số là 0, khi đó cửa trước và cửa sau nằm cùng một vị trí. Giả sử, có $n=2^k$ vị trí cho các cửa, khi đó kích thước tối đa của cửa sổ là n-1 (không là n để phân biệt với kích thước là 0).
- Giả sử ta dùng k bit để đánh số thứ tự cho các khung. Ta sẽ có 2^k khung, đánh số từ 0 đến 2^k-1. Khi đó cửa sổ trượt sẽ được chia thành 2^k vị trí tương ứng với 2^k khung.
- Đối với cửa sổ gởi, các vị trí nằm trong cửa sổ trượt biểu hiện số thứ tự của các khung mà bên gởi đang chờ bên nhận báo nhận. Phần bên ngoài cửa sổ là các khung có thể gởi tiếp. Tuy nhiên phải đảm bảo rằng, cửa sổ gởi không được vượt quá kích thước tối đa của cửa sổ.
- Đối với bên nhận, các vị trí nằm trong cửa sổ biểu hiện số thứ tự các khung mà nó đang sẵn sàng chờ nhận.
- Kích thước tối đa của cửa sổ biểu thị dung lượng bộ nhớ đệm của bên nhận có thể lưu tạm thời các gói tin nhận được trước khi xử lý chúng. Giả sử bên nhận có một vùng bộ nhớ đệm có khả năng lưu trữ 4 khung nhận được. Khi đó, kích thước tối đa của cửa sổ sẽ là 4.

Hoạt động của cửa sổ trượt

Ví dụ sau mô tả hoạt động của cửa sổ trượt với kích thước cửa sổ là 1, sử dụng 3 bits để đánh số thứ tự khung (từ 0 đến 7).



Hoạt động của cửa sổ trượt

Khởi đầu, Hình (a):

Bên gởi: chưa gởi khung nào nên kích thước của cửa sổ là 0.

Bên nhận đang chờ nhận khung 0, kích thước cửa sổ là 1

Bên gởi gởi khung số 0: Nó kiểm tra kích thước của cửa số trượt là 0, nhỏ hơn kích thước tối đa nên nó được phép gởi. Cửa trước của cửa sổ gởi di chuyển lên một bước chứa giá trị 0 là số thứ tự của khung báo nhận bên gởi đang chờ. Kích thước cửa sổ trượt lúc này là 1, đạt đến kích thước tối đa nên nó không được phép gởi thêm khung nữa (Hình b).

Bên nhận nhận được khung 0: nó kiểm tra và nhận thấy khung không có lỗi. Nó gởi khung báo nhận số 0 về cho bên nhận. Đồng thời cửa sau của nó di chuyển để loại khung số 0 ra khỏi cửa sổ trượt. Cửa trước cũng di chuyển để mở rộng kích thước cửa sổ đến giá trị tối đa. Lúc này cửa sổ nhận chứa khung số 1 là khung mà nó đang chờ nhận tiếp (Hình c).

Bên gởi nhận được khung báo nhận số 0: Vì đây là khung báo hiệu bên nhận đã nhận tốt nên cửa sau của cửa sổ gởi di chuyển để loại khung số 0 ra khỏi cửa sổ gởi. Lúc này cửa sổ gởi có kích thước là 0, bên gởi có quyền gởi tiếp khung (Hình d)

Như vậy khi kích thước của cửa sổ trượt là 1, ta có giao thức stop-and-wait.

Cài đặt giao thức cửa sổ trượt kích thước 1 bit (A One-Bit Sliding Window Protocol)

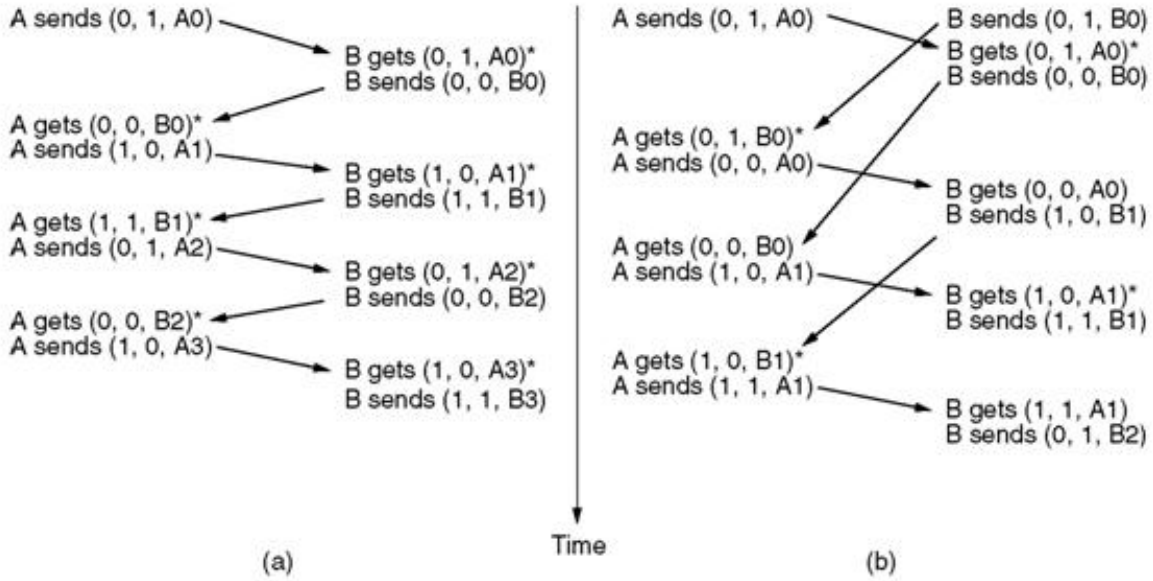
```
/* Protocol 4 (sliding window) is bidirectional. */
#define MAX_SEQ 1                                   /* Kích thước cửa sổ là 1 */
typedef enum {frame_arrival, cksum_err, timeout} event_type;
#include "protocol.h"
void protocol4 (void)
{
  seq_nr next_frame_to_send;                        /* Số thứ tự của khung gởi đi kế tiếp */
  seq_nr frame_expected;                            /* Số thứ tự của khung báo nhận đang chờ nhận */
  frame r, s;                                       /* Khung nhận và khung gởi */
  packet buffer;                                    /* Gói tin chờ gởi */
  event_type event;
  next_frame_to_send = 0;                           /* Khởi động số thứ tự khung gởi */
  frame_expected = 0;                               /* Khởi động số thứ tự khung báo nhận chờ nhận */
  from_network_layer(&buffer);                      /* Nhận gói tin từ tầng mạng để gởi đi */
  s.info = buffer;                                  /* Đưa gói tin dữ liệu vào khung để gởi */
  s.seq = next_frame_to_send;                       /* Đặt số thứ tự cho khung */
  s.ack = 1 – frame_expected;                       /* Đặt số thứ tự báo nhận vào khung */
  to_physical_layer(&s);                            /* Đưa khung xuống tầng vật lý để gởi */
  start_timer(s.seq);                               /* Khởi động bộ đếm thời gian */
  while (true) {
      wait_for_event(&event);                       /* Chờ sự kiện Khung đến, Khung bị lỗi, quá thời gian */
      if (event == frame_arrival) {                 /* Một khung đến không bị lỗi */
          from_physical_layer(&r);                  /* Nhận khung từ tầng vật lý */
          if (r.seq == frame_expected) {            /* Kiểm tra có phải là khung đang chờ nhận không */
              to_network_layer(&r.info);            /* Lấy gói tin ra khỏi khung và chuyển lên tầng mạng */
              inc(frame_expected);                  /* Tăng số thứ tự của khung chờ nhận kế tiếp */
          }
          if (r.ack == next_frame_to_send) {        /* Nếu bên kia đã báo nhận khung vừa gởi */
              stop_timer(r.ack);                    /* Xóa bộ đếm thời gian */
              from_network_layer(&buffer);          /* Nhận gói tin kế tiếp từ tầng mạng để gởi đi */
              inc(next_frame_to_send);              /* Tăng số thứ tự của khung kế tiếp */
          }
      }
      s.info = buffer;                              /* Đưa gói tin vào khung để gởi */
      s.seq = next_frame_to_send;                   /* Đặt số thứ tự cho khung gởi */
      s.ack = 1 – frame_expected;                   /* Đặt số thứ tự khung báo nhận */
      to_physical_layer(&s);                        /* Đưa khung xuống tầng vật lý để gởi */
      start_timer(s.seq);                           /* Khởi động bộ đếm thời gian */
  }
}
```

Ví dụ về 2 kịch bản của giao thức trên

(a): Việc gởi nhận diễn ra bình thường theo đúng tuần tự

(b): Việc gởi nhận diễn ra theo một trình tự bất kỳ

Ký hiệu **A send (seq, ack, packet number)** để chỉ rằng A gởi B một khung có số thứ tự là **seq**, đồng thời báo cho B biết A đã nhận được tốt khung có số thứ tự **ack** của B gởi sang. Khung chứa gói tin thứ **packet number**. Dấu * biểu thị rằng khung tốt, và gói tin được lấy ra khỏi khung để chuyển cho tầng mạng.



Kịch bản giao thức cửa sổ trượt với kích thước là 1

Vấn đề điều khiển lỗi (Error Control)

Vấn đề kế tiếp cần phải quan tâm là bên nhận sẽ làm gì nếu khung bị lỗi.

Giải pháp đơn giản là truyền lại tất cả các khung bắt đầu từ khung thứ N bị

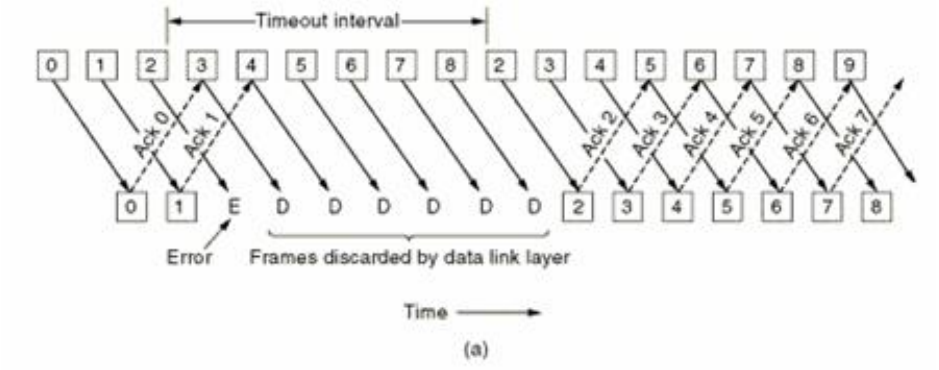
lỗi. Nếu có những khung khác được nhận trong khoảng thời gian này thì chúng đều bị bỏ qua. Đây gọi là giao thức **Go-Back-N.**

Giải pháp thứ hai là chỉ truyền lại những khung bị lỗi, và chờ đến khi nó được gởi lại trước khi tiếp tục việc gởi tin, gọi là giao thức **Selective Repeat.**

Giao thức Go-Back-N

Giao thức Go-Back-N thì rất đơn giản. Khi một khung bị lỗi. Bên nhận bỏ qua khung. Vì không một báo nhận nào gởi về cho bên nhận nên sự kiện quá thời gian xảy ra, bên gởi phải gởi lại khung bị lỗi và toàn bộ các khung phía sau nó.

Ví dụ:



Giao thức Go-Back-N

Trong ví dụ trên, bên nhận phát hiện ra khung số 2 bị lỗi nó bỏ qua các khung sau đó (3,4,5,6,7,8), chỉ chờ nhận lại khung số 2. Phía bên gởi chờ báo nhận từ bên nhận cho đến khi quá thời gian, nó sẽ thực hiện gởi lại các khung 2, 3, 4, 5, 6, ...

Đoạn chương trình sau cài đặt giao thức Go-Back-N

/* Giao thức này cho phép nhiều khung được gởi đi. Bên gởi có thể gởi trước đến MAX_SEQ khung mà không cần chờ một báo nhận. Điểm lưu ý khác là tầng mạng không phải luôn luôn có dữ liệu sẵn sàng để gởi. Khi nào có dữ liệu để gởi, tầng mạng sẽ sinh ra một sự kiện network-layer- ready.*/

```
#define MAX_SEQ 7                       /* Kích thước lớn nhất của cửa sổ trượt, phải là 2^k-1*/
typedef enum {frame_arrival, cksum_err, timeout, network_layer_ready} event_type;
#include "protocol.h"

static boolean between(seq_nr a, seq_nr b, seq_nr c)
{
   /*              True nếu a<=b<c                    */
  if (((a <= b) && (b < c)) || ((c < a) && (a <= b)) || ((b < c) && (c < a)))
      return(true);
   else
      return(false);
}

static void send_data(seq_nr frame_nr, seq_nr frame_expected, packet buffer[ ])
{
/*              Tạo khung gởi gói tin đi                    */
  frame s;                              /* Khung để gởi gói tin đi */

  s.info = buffer[frame_nr];            /* Đưa gói tin vào khung */
  s.seq = frame_nr;                     /* Đặt số thứ tự cho khung gởi*/
  s.ack = (frame_expected + MAX_SEQ) % (MAX_SEQ + 1/* Đặt số thứ tự cho khung cần báo nhận
  to_physical_layer(&s);                /* Gởi khung xuống tầng vật lý để truyền đi */
  start_timer(frame_nr);                /* Khởi động bộ đếm thời gian cho khung gởi đi*/
}

void protocol5(void)
{
  seq_nr next_frame_to_send;        /* Số thứ tự cho khung gởi kế tiếp */
  seq_nr ack_expected;              /* Khung lâu nhất chưa được báo nhận */
  seq_nr frame_expected;            /* Khung chờ nhận kế tiếp /
  frame r;                          /* Khung */
  packet buffer[MAX_SEQ + 1];       /* Vùng bộ nhớ đệm cho các khung gởi đi */
  seq_nr nbuffered;                 /* Số lượng bộ nhớ đệm đang được dùng */
  seq_nr i;                         /* Chỉ số mảng của vùng nhớ đệm */
  event_type event;

  enable_network_layer();           /* Cho phép tầng mạng tạo sự kiện network_layer_ready */
  ack_expected = 0;                 /* Đặt báo nhận đầu tiên chờ nhận là 0 */
  next_frame_to_send = 0;           /* Khung đầu tiên gởi đi là 0 */
  frame_expected = 0;               /* Khung chờ nhận đầu tiên là 0*/
  nbuffered = 0;                    /* Chưa có dữ liệu trong vùng nhớ đệm */
```

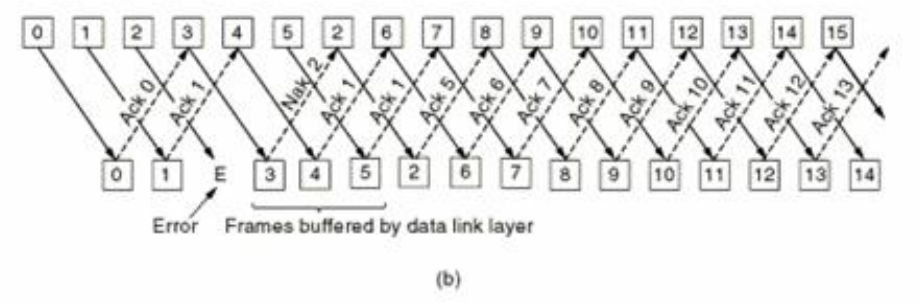
```
while (true) {
  wait_for_event(&event);              /* Chờ 1 trong 4 sự kiện liệt kê ở trên xảy ra */

  switch(event) {
    case network_layer_ready:      /* Tầng mạng có một gói tin cần gởi đi */
         /* Chấp nhận, lưu và truyền đi một khung mới */
         from_network_layer(&buffer[next_frame_to_send]);/* Nhận khung từ tầng vật lý*/
         nbuffered = nbuffered + 1; /* Tăng kích thước cửa sổ gởi */
         send_data(next_frame_to_send, frame_expected, buffer);/* Gởi khung đi */
         inc(next_frame_to_send); /* Di chuyển cửa trước của cửa sổ gởi */
         break;

    case frame_arrival:              /* Một khung dữ liệu hay điều khiển vừa đến */
         from_physical_layer(&r);  /* Nhận khung từ tầng vật lý */

         if (r.seq == frame_expected) {
             /* Là khung đang được chờ đợi */
              to_network_layer(&r.info)/* Chuyển gói tin lên tầng mạng */
              inc(frame_expected);   /*Di chuyển cửa sau của cửa sổ nhận */
          }

       /* Nếu là Ack thứ n, sẽ không quan tâm đến các ACK n-1, n-2, ... */
         while (between(ack_expected, r.ack, next_frame_to_send)) {
       /* Xử lý các báo nhận */
              nbuffered = nbuffered —1;/* Giảm kích thước cửa sổ gởi */
              stop_timer(ack_expected);/* Khung đã đến,  xóa bộ đếm thời gian */
              inc(ack_expected);/* Tăng số thứ tự khung chờ nhận kế tiếp */
         }
         break;

    case cksum_err: break;           /* Khung nhận bị lỗi, bỏ qua */

    case timeout:   /* Quá thời gian, truyền lại tất cả các khung đang chờ báo nhận */
         next_frame_to_send = ack_expected;  /* Bắt đầu truyền lại */
         for (i = 1; i <= nbuffered; i++) {
              send_data(next_frame_to_send, frame_expected, buffer);/* Truyền lại */
              inc(next_frame_to_send); /* Chuẩn bị để truyền khung kế tiếp */
         }

  }

  if (nbuffered < MAX_SEQ)           /* Vùng đệm còn khả năng chứa gói tin ? */
         enable_network_layer();
  else
         disable_network_layer();
 }
}
```

Cài đặt giao thức Go-Back-N

Giao thức Selective Repeat

Trong giao thức này, khung bị lỗi bị bỏ đi, nhưng các khung nhận tốt sau đó đều được lưu lại tạm thời trong vùng nhớ đệm. Khi quá thời gian, bên gởi chỉ gởi lại khung cũ nhất chưa được báo nhận. Nếu khung này đến nơi chính xác, bên nhận có thể chuyển lên tầng mạng tất cả các khung đã được lưu vào bộ nhớ đệm theo đúng thứ tự.

Trong giao thức này, bên nhận sử dụng khung **Báo không nhận** NAK (Negative Acknowledge) khi phát hiện ra khung bị lỗi, ví dụ lỗi CRC, sai thứ tự gói tin. NAK sẽ được gởi về bên nhận trước khi sự kiện quá thời gian báo nhận của khung bị lỗi xảy ra. Nhờ đó tăng được hiệu xuất truyền tin.



Giao thức Selective Repeat với cửa sổ trượt lớn hơn 1

Trong ví dụ trên các khung 0, 1 được nhận tốt và đã được báo nhận, còn khung số 2 thì bị lỗi trên đường truyền. Khi khung số 3 đến, tầng liên kết dữ liệu phát hiện lỗi về số thứ tự khung chờ nhận, vì thế nó gởi khung NAK cho khung số 2 và lưu tạm thời khung số 3 vào vùng nhớ đệm. Tương tự, các khung 4 và 5 cũng được lưu lại mà chưa chuyển lên tầng mạng (vì phải chờ nhận khung số 2).

Khi khung NAK 2 đến bên gởi, nó truyền lại ngay khung số 2.

Khi khung số 2 đến bên nhận, nó đã có đủ các khung 2,3,4,5 theo đúng thứ tự vì thế nó chuyển 4 khung này lên tầng mạng theo một thứ tự đúng đắn. Đồng thời bên nhận gởi về bên gởi khung ACK 5 để báo rằng đã nhận tốt đến khung số 5.

Trong trường hợp khung NAK2 bị mất, không đến được bên gởi, thì sự kiện quá thời gian sẽ xảy ra. Khi đó bên gởi cũng chỉ gởi lại khung số 2 mà thôi.

/* Giao thức này chấp nhận các khung đến không đúng thứ tự, nhưng chúng lại được chuyển lên tầng mạng theo một thứ tự đúng đắn. Với mỗi khung gởi đi, sẽ có một bộ đếm thời gian đi kèm. Khi quá thời hạn chỉ khung tương ứng được gởi lại, thay vì gởi lại tất cả các khung như giao thức Go-Back-N */

```
#define MAX_SEQ 7                                   /*  Số thứ tự khung lớn nhất*/
#define NR_BUFS ((MAX_SEQ + 1)/2)                   /* Kích thước tối đa cửa sổ gởi và  nhận
typedef enum {frame_arrival, cksum_err, timeout, network_layer_ready, ack_timeout} event_type;
#include "protocol.h"
boolean no_nak = true;                              /* Chưa gởi khung NAK*/
seq_nr oldest_frame = MAX- SEQ + 1;                 /* Khung cũ nhất đã gởi */

static boolean between(seq_nr a, seq_nr b, seq_nr c)
{
/*          Kiểm tra b có là giá trị giữa a và c không                      */
 return ((a <= b) && (b < c)) II ((c < a) && (a <= b)) II ((b < c) && (c < a));
}

static void send_frame(frame_kind fk, seq_nr frame_nr, seq_nr frame_expected, packet buffer[ ])
{
/* Tạo và gởi khung dữ liệu / báo nhận / báo lỗi */
 frame s;                                           /* Khung */

 s.kind = fk;                                       /* Kiểu khung: data, ack, nak */
 if (fk == data) s.info = buffer[frame_nr % NR_BUFS];
 s.seq = frame_nr;                                  /* Đặt số thứ tự cho khung dữ liệu gởi đi */
 s.ack = (frame_expected + MAX_SEQ) % (MAX_SEQ + 1);
 if (fk == nak) no_nak = false;                     /* Chỉ gởi 1 NAK cho một khung */
 to_physical_layer(&s);                             /* Gởi khung đi */
 if (fk == data) start_timer(frame_nr % NR_BUFS)/* Khởi động bộ đếm thời gian cho khung dữ liệu gởi */
 stop_ack_timer();                                   /* Ngừng bộ đếm thời gian chờ báo nhận*/
}
void protocol6(void)
{
 seq_nr ack_expected;                               /* Khung chờ được báo nhận */
 seq_nr next_frame_to_send;                         /* Khung kế tiếp gởi đi */
 seq_nr frame_expected;                             /* Khung đang chờ nhận */
 seq_nr too_far;                                    /* Khung kế tiếp chưa được nhận */
 int i;                                             /* Chỉ số vùng nhớ tạm */
 frame r;                                           /* Khung nhận và gởi */
 packet out_buf[NR_BUFS];                           /* Vùng đệm cho dữ liệu gởi */
 packet in_buf[NR_BUFS];                            /* Vùng đệm cho dữ liệu nhận */
 boolean arrived[NR_BUFS];                          /* Theo dõi sử dụng vùng đệm dữ liệu nhận */
 seq_nr nbuffered;                                  /* Số lượng vùng đệm gởi đang được sử dụng */
 event_type event;

 enable_network_layer();                            /* Gán giá trị khởi động */
 ack_expected = 0;                                                          /
 next_frame_to_send = 0;
 frame_expected = 0;
 too_far = NR_BUFS;
 nbuffered = 0;
 for (i = 0; i < NR_BUFS; i++) arrived[i] = false;    /* Khởi tạo các vùng đệm nhận rỗng */
```

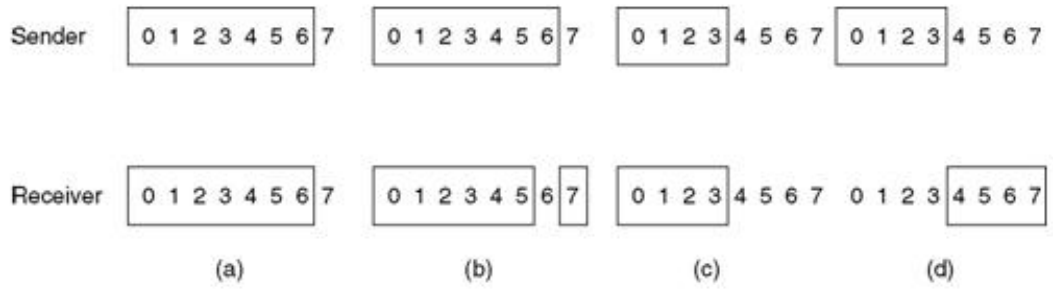
```
while (true) {
  wait_for_event(&event);                    /* Chờ 1 trong 5 sự kiện phát sinh */
  switch(event) {
    case network_layer_ready:                /* Nhận, lưu và truyền một khung mới */
         nbuffered = nbuffered + 1;          /* Mở rộng kích thước cửa sổ gởi */
         from_network_layer(&out_buf[next_frame_to_send % NR_BUFS]); /* Nhận gói tin từ tầng mạng */
         send_frame(data, next_frame_to_send, frame_expected, out_buf);/* Gởi khung đi */
         inc(next_frame_to_send);            /* Tăng số thứ tự khung gởi kế tiếp */
         break;

    case frame_arrival:                      /* Một khung dữ liệu hoặc điều khiển vừa đến */
         from_physical_layer(&r);            /* Nhận khung từ tầng vật lý */
         if (r.kind == data) {
             /* Là khung dữ liệu có lỗi thì gởi khung NAK, ngược lại tính giờ để gởi khung ACK  */
              if ((r.seq != frame_expected) && no_nak)
                 send_frame(nak, 0, frame_expected, out_buf); else start_ack_timer();
              if (between(frame_expected, r.seq, too_far) && (arrived[r.seq%NR_BUFS] == false)) {
             /* Là khung có thứ tự chấp nhận và chưa được lưu dữ liệu lại*/
                   arrived[r.seq % NR_BUFS] = true;   /* Đưa dữ liệu vào vùng đệm */
                   in_buf[r.seq % NR_BUFS] = r.info;  /* Đánh dấu vùng đệm đã sử dụng */
                   while (arrived[frame_expected % NR_BUFS]) {
             /* Chuyển tất cả các khung đang trong vùng đệm lên tầng mạng và mở rộng cửa sổ nhận */
                        to_network_layer(&in_buf[frame_expected % NR_BUFS]);
                        no_nak = true;
                        arrived[frame_expected % NR_BUFS] = false;
                        inc(frame_expected);  /* Di chuyển cửa sau cửa sổ nhận */
                        inc(too_far);         /* Di chuyển cửa trước cửa sổ nhận */
                        start_ack_timer();    /* Khởi động đồng hồ cho các báo nhận */
                   }
              }
         }
         if((r.kind==nak) && between(ack_expected,(r.ack+1)%(MAX_SEQ+1),next frame to send))
              send_frame(data, (r.ack+1) % (MAX_SEQ + 1), frame_expected, out_buf);

         while (between(ack_expected, r.ack, next frame to send)) {
              nbuffered = nbuffered— 1;        /* Giảm kích thước cửa sổ gởi */
              stop_timer(ack_expected % NR_BUFS);/* Khung đã đến nơi thật sự */
              inc(ack_expected);               /* Di chuyển cửa dưới của cửa sổ gởi */
         }
         break;
    case cksum_err:
         if (no_nak) send_frame(nak, 0, frame_expected, out_buf)/* Khung bị lỗi */
         break;
    case timeout:
         send_frame(data, oldest_frame, frame_expected, out_buf);/* Bên gởi xử lý chậm */
         break;
    case ack_timeout:
         send_frame(ack,0,frame_expected, out_buf);  /* Quá thời gian cho các báo nhận */
  }
  if (nbuffered < NR_BUFS) enable_network_layer(); else disable_network_layer();
 }
}
```

Cài đặt giao thức Selective-Repeat

Một số điểm cần lưu ý khi sử dụng cửa số trượt với kích thước lớn hơn 1:

Kích thước tối đa của cửa sổ gởi và nhận là bao nhiêu ?

Giả sử ta dùng 3 bit để đánh số cho khung. Như vậy bên gởi được phép gởi trước tối đa 7 khung trước khi chờ bên nhận gởi báo nhận về.



Giao thức cửa sổ trượt với kích thước là 7

- Lúc đầu bên gởi gởi đi 7 khung từ 0 đến 6, bên nhận đang sẵn sàng chờ nhận bất kỳ một khung nào có số thứ tự từ 0 đến 6 (Hình a).
- Tất cả các khung đến nơi không có lỗi, bên nhận gởi các báo nhận và chuyển cửa số nhận về vị trí sẵn sàng để nhận các khung 7,0,1,2,3,4 và 5 (Hình b).
- Tại thời điểm đó, đường truyền có sự cố làm cho tất cả các khung báo nhận đều mất. Quá thời gian, bên gởi gởi lại khung 0. Khi khung này đến bên nhận, nó kiểm tra xem khung có nằm trong cửa sổ nhận không. Điều không may mắn đã xảy ra: khung 0 nằm trong cửa sổ nhận mới (Hình b). Bên nhận nhận khung 0 xem như một khung mới hoàn toàn và chuyển khung 0 lên tầng mạng. Như vậy tầng mạng đã nhận 2 lần cùng một gói tin, tức giao thức vận hành sai.
- Tình trạng này có thể tránh được nếu ta đảm bảo rằng cửa sổ nhận mới không đè chồng lên cửa sổ trước đó. Điều này có thể thực hiện được nếu ta giới hạn kích thước tối đa của của sổ nhận bằng một nửa khoảng đánh số thứ tự của khung.
    - Ví dụ: Nếu dùng 3 bit để đánh số thứ tự khung từ 0 đến 7 thì kích thước tối đa cửa sổ nhận là (7-0+1)/2 =4.
    - Nếu dùng 4 bit để đánh số thứ tự khung từ 0 đến 15 thì kích thước tối đa cửa sổ nhậnt là (15-0+1)/2 =8.

Số lượng buffer để lưu khung là bao nhiêu?

Số lượng buffer chỉ cần bằng kích thước tối đa của cửa sổ nhận, không cần thiết phải bằng số lượng khung. Ví dụ: Nếu dùng 3 bit để đánh số thứ tự khung từ 0 đến 7 thì kích thước tối đa cửa sổ nhận là (7-0+1)/2 =4 và số lượng buffer cần thiết cũng là 4.

Khi nào gởi báo nhận cho một gói tin?

Ta thấy rằng, khi một khung đến, báo nhận của khung này sẽ không được gởi ngược về một cách tức thì. Thay vào đó, nó sẽ được gởi kèm trong khung dữ liệu kế tiếp của bên nhận. Nếu bên nhận không có dữ liệu để gởi đi, báo nhận sẽ bị giữ lại khá lâu. Chính vì thế, mỗi khi có khung đến thì bộ đếm thời gian start_ack_timer được khởi động. Nếu

trong suốt khoảng thời gian này không có một khung dữ nào cần gởi đi, thì sau đó một khung báo nhận riêng biệt sẽ được gởi đi. Bộ đếm thời gian start_ack_timer sinh ra sự kiện ack_timeout. Chúng ta cũng phải đảm bảo rằng, bộ đếm thời gian start_ack_timer thì ngắn hơn bộ đếm thời gian chờ báo nhận cho các khung dữ liệu.

Giao thức HDLC (High-Level Data Link Control)

Giao thức điều khiển liên kết dữ liệu quan trong nhất là HDLC. Không phải vì nó được sử dụng rộng rãi mà nó còn là cơ sở cho nhiều giao thức điều khiển liên kết dữ liệu khác.

Các đặc tính của giao thức HDLC

Giao thức HDLC định nghĩa 3 loại máy trạm, hai cấu hình đường nối kết và 3 chế độ điều khiển truyền tải

Ba loại trạm trong HDLC

- Trạm chính (Primary Station): Có trách nhiệm điều khiển các thao thác về đường truyền. Các khung được gởi từ trạm chính gọi là lệnh (Command).
- Trạm phụ (Secondary Station): Hoạt động dưới sự kiểm soát của trạm chính. Khung gởi từ trạm phụ gọi là các trả lời. Trạm chính duy trì nhiều đường nối kết luận lý đến các trạm phụ trên đường truyền.
- Trạm hỗn hợp (Combined Station): Bao gồm đặc điểm của trạm chính và trạm phụ. Một trạm hỗn hợp có thể gởi đi các lệnh và các trả lời.

Hai cấu hình đường nối kết:

- Cấu hình không cân bằng (Unbalanced Configuration): Gồm một máy trạm chính (Primary Station) và nhiều máy trạm phụ (Secondary station) và hỗ trợ cả 2 chế độ truyền song công và bán song công
- Cấu hình cân bằng (Balanced Configuration): Bao gồm 2 máy trạm hỗn hợp, và hỗ trợ cả 2 chế độ truyền song công và bán song công.

Có 3 chế độ truyền tải là:

- Chế độ trả lời bình thường (NRM- Normal Response Mode), được sử dụng với cấu hình đường nối kết không cân bằng. Máy chính có thể khởi động một cuộc truyền tải dữ liệu về cho máy phụ. Nhưng máy phụ chỉ có thể thực hiện việc truyền dữ liệu cho máy chính như là những trả lời cho các yêu cầu của máy chính.
- Chế độ cân bằng bất đồng bộ (ABM - Asynchronous Response Mode): Được sử dụng với cấu hình nối kết cân bằng. Cả hai máy đều có quyền khởi động các cuộc truyền tải dữ liệu mà không cần sự cho phép của máy kia.

- Chế độ trả lời bất đồng bộ (ARM-Asynchronous Response Mode): Sử dụng cấu hình không cân bằng. Một máy phụ có thể khởi động một cuộc truyền tải và không cần sự cho phép tường minh của máy chính. Máy chính vẫn đảm trách vai trò bảo trì đường truyền bao gồm việc khởi động, phục hồi lỗi và xóa nối kết.

Chế độ NRM đòi hỏi phải có nhiều đường dây để nối một máy chính với nhiều thiết bị đầu cuối. Chế độ ABM được sử dụng nhiều nhất trong 3 chế độ, nó cho phép sử dụng hiệu quả đường truyền. Chế độ ARM thì ít được dùng đến.

Cấu trúc khung

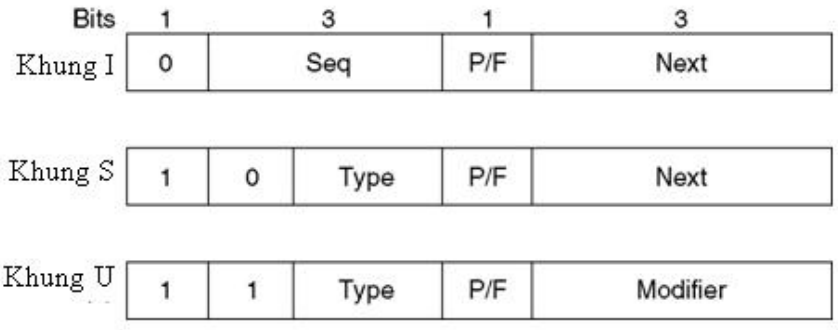
HDLC sử dụng chế độ truyền tải đồng bộ, các bits dữ liệu truyền đi được gói vào trong các khung và sử dụng một cấu trúc khung cho tất cả các loại dữ liệu cũng như thông tin điều khiển.

Khung trong giao thức HDLC có cấu trúc như sau:

Cấu trúc khung của HDLC

Flag (8 bit)	Là cờ dùng để xác định điểm bắt đầu và kết thúc của khung, giá trị nó là 01111110. HDLC sử dụng kỹ thuật bit độn để loại trừ sự xuất hiện của cờ trong dữ liệu.
Address (8 bit)	Vùng ghi địa chỉ để xác định máy phụ được phép truyền hay nhận khung.
Control (8bit)	Được dùng để xác định loại khung. Mỗi loại có thông tin điều khiển khác nhau. Có 3 loại khung: Thông tin (I), Điều khiển (S) và không đánh số (U).
Information(128-1024 bytes)	Vùng chứa dữ liệu cần truyền.
FCS (Frame Check Sequence- 8 bit)	Vùng chứa mã kiểm soát lỗi, dùng phương pháp đa thức **CRC-CCITT= X16 + X12 + X5 +1**

Giá trị 8 bit của trường control hình thành 3 loại khung như sau:



Cấu trúc trường điều khiển trong khung HDLC

Giao thức HDLC sử dụng một cửa sổ trượt với số thứ tự khung 3 bít. Trường seq trong khung I để chỉ số thứ tự của khung thông tin hiện tại. Trường Next để chỉ số thứ tự của khung thông tin mà bên gởi đang chờ nhận (thay vì là khung đã nhận tốt như giao thứ cửa sổ trượt đã giới thiệu ở phần trước).

Bit P/F có ý nghĩa là Poll/Final, tức chọn hoặc kết thúc. Khi máy tính chính mời một máy phụ truyền tin, thì bit này được đặt lên 1 có ý nghĩa là P (Poll, chọn). Ngược lại khi thông tin được truyền từ máy phụ lên máy chính thì nó được đặt xuống 0, để báo với máy chính rằng máy phụ hiện tại vẫn còn dữ liệu để gởi đi. Khi máy phụ gởi khung cuối cùng, bit này được đặt lên 1, có ý nghĩa là F (Final, kết thúc), để báo cho máy chính biết rằng nó đã hoàn thành việc truyền tải thông tin.

Khung S(Supervisory Frame)là khung điều khiển, dùng để kiểm soát lỗi và luồng dữ liệu trong quá trình truyền tin. Khung S có 4 kiểu được xác định bởi tổ hợp giá trị của 2 bit trong trường Type.

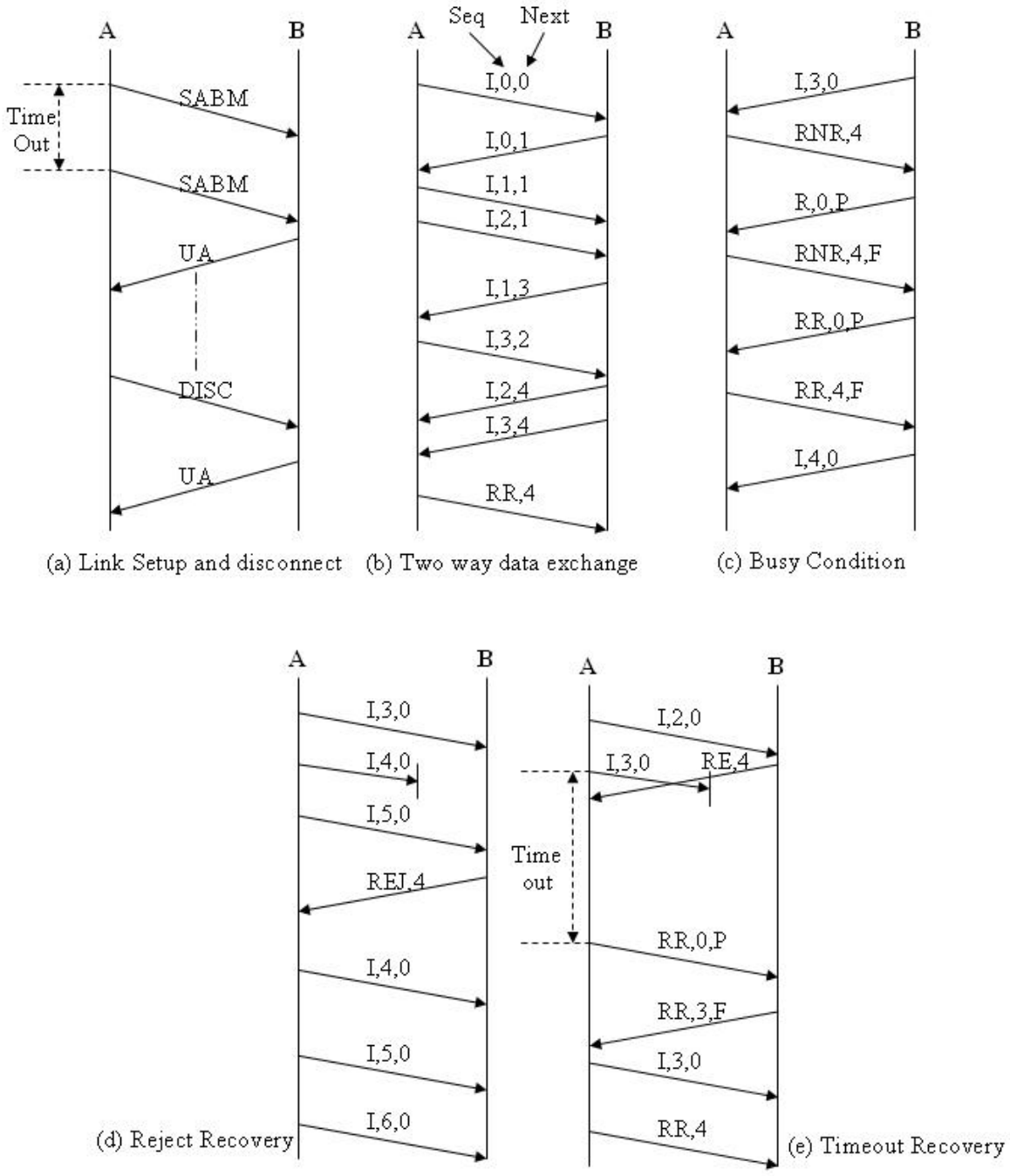
SS=00	RR (Receive Ready), là khung báo nhận, thông báo sẵn sàng nhận dữ liệu, đã nhận tốt đến khung Next-1, và đang đợi nhận khung Next. Được dùng đến khi không còn dữ liệu gởi từ chiều ngược lại để vừa làm báo nhận (figgyback)
SS=01	REJ (Reject): đây là một khung báo không nhận (negative acknowledge), yêu cầu gởi lại các khung, từ khung Next.
SS=10	RNR (Receive Not Ready): thông báo không sẵn sàng nhận tin, đã nhận đến đến khung thứ Next-1, chưa sẵn sàng nhận khung Next
SS=11	SREJ (Selective Reject): yêu cầu gởi lại một khung có số thức tự là Next

Khung U (Unnumbered Frame) thường được sử dụng cho mục đích điều khiển đường truyền, nhưng đôi khi cũng được dùng để gởi dữ liệu trong dịch vụ không nối kết. Các lệnh của khung U được mô tả như sau:

1111P100	Lệnh này dùng để thiết lập chế độ truyền tải SABM (Set Asynchronous Balanced Mode).
1100P001	Lệnh này dùng để thiết lập chế độ truyền tải SNRM (Set Normal Response Mode).
1111P000	Lệnh này dùng để thiết lập chế độ truyền tải SARM (Set Asynchronous Response Mode).
1100P010	Lệnh này để yêu cầu xóa nối kết DISC (Disconnect).
1100F110	UA (Unumbered Acknowledgment). Được dùng bởi các trạm phụ để báo với trạm chính rằng nó đã nhận và chấp nhận các lệnh loại U ở trên.
1100F001	CMDR/FRMR (Command Reject/Frame Reject). Được dùng bởi trạm phụ để báo rằng nó không chấp nhận một lệnh mà nó đã nhận chính xác.

Một vài kịch bản về giao thức HDLC

Kịch bản (a) mô tả các khung liên quan trong quá trình thiết lập và xóa nối kết. Đầu tiên một trong hai bên giao tiếp sẽ gởi khung SABM sang bên kia và thiết lập một bộ đếm thời gian. Bên phía còn lại khi nhận được khung SABM sẽ trả lời bằng khung UA. Bên yêu cầu nối kết khi nhận được khung UA sẽ xóa bỏ bộ đếm thời gian. Nối kết đã được hình thành và hai bên có thể truyền khung qua lại cho nhau. Nối kết sẽ xóa đi nếu một trong hai bên giao tiếp gởi khung DISC. Trong một trường hợp khác, nếu sau một khoảng thời gian trôi qua, bên yêu cầu nối kết không nhận được khung UA, nó sẽ cố gắng gởi lại khung SABM một số lần qui định. Nếu vẫn không nhận được khung UA, bên yêu cầu nối kết sẽ thông báo lỗi lên tầng cao hơn.



Một vài kịch bản của HDLC

Kịch bản (b) mô tả tiến trình trao đổi khung I giữa hai bên. Ta thấy rằng bên A gởi liên tiếp các khung (I,1,1 và I,2,1) mà không nhận được khung báo nhận thì số thứ tự của khung chờ nhận vẫn không thay đổi, trong trường hợp này là 1. Ngược lại khi bên B nhận liên tiếp các khung (I,1,1 và I,2,1) mà không gởi khung nào đi, thì khung chờ nhận kế tiếp của khung thông tin truyền đi phải là số kế tiếp của khung vừa nhận, là 3.

Trong kịch bản (c) máy A không thể xử lý kịp các khung do B gởi đến vì thế nó gởi khung RNR để yêu cầu B tạm dừng việc việc truyền tải. Bên B định kỳ gởi thăm dò bên

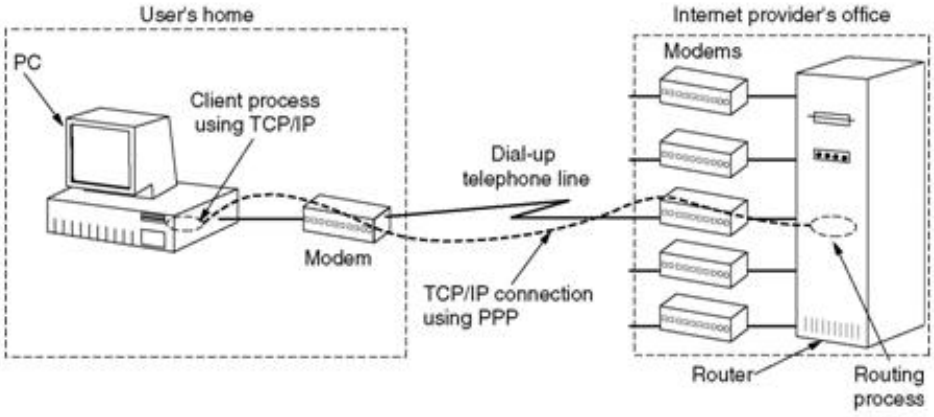
A bằng cách gởi khung RR với bit P được đặt lên 1. Nếu bên A vẫn chưa thể nhận thông tin từ bên B nó sẽ trả lời bằng khung RNR, ngược lại nếu A đã sẵn sàng thì nó sẽ trả lời bằng khung RR.

Trong kịch bản (d), bên A gởi sang B ba khung thông tin 3,4 và 5. Khung 4 bị mất hoàn toàn trên đường truyền. Khi bên B nhận được khung 5, nó sẽ bỏ qua khung này vì sai thứ tự khung. B gởi REJ với trường Next là 4 để yêu cầu A gởi lại tất cả các khung từ khung số 4.

Kịch bản (e) minh họa cách thức phục hồi lỗi dựa vào thời gian (timeout). Khung số 3 bị lỗi và do đó B bỏ nó. B không thể gởi khung REJ vì nó không thể xác định được đó có phải là khung I hay không. Bên A sau một khoảng thời gian trôi qua không thấy khung trả lời từ B, nó sẽ gởi khung RR với bit P=1 để kiểm tra trạng thái của bên kia. Bên B sẽ đáp lại bằng khung RR với trường Next là 3 để báo hiệu khung số 3 đã mất. Sau đó A sẽ truyền lại khung số 3.

Giao thức Điểm nối điểm (PPP- Point-to-Point Protocol)

PPP là một giao thức đặc biệt quan trọng trong mạng Internet. Nó cho phép truyền tải thông tin giữa các router trên mạng hay để cho phép nối các máy tính người dùng vào mạng của nhà cung cấp dịch vụ Internet (ISP).



Sơ đồ nối kết của giao thức PPP

Giao thức PPP được định nghĩa trong RFC (Request For Comments) 1661 và sau đó được mở rộng thêm bằng các RFC 1662, RFC 1663. PPP thực hiện chức năng phát hiện lỗi trên dữ liệu truyền, hỗ trợ nhiều giao thức vận hành trên nó, phân phối địa chỉ IP khi máy tính nối kết vào mạng, kiểm tra quyền đăng nhập và nhiều tính năng khác.

PPP cung cấp 3 đặc tính sau:

Định nghĩa một phương pháp định khung cùng với phương pháp phát hiện lỗi.

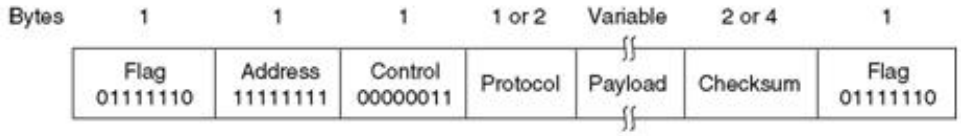
1. Giao thức điều khiển đường truyền cho phép thiết lập kênh giao tiếp, kiểm tra kênh, thỏa thuận về các thông số truyền tin và xóa kênh truyền khi không cần thiết nữa. Giao thức này được gọi là giao thức LCP (Link Control Protocol).
2. Có phương pháp thương lượng về các tùy chọn tầng mạng một cách độc lập với giao thức mạng được sử dụng. Phương pháp được chọn lựa NCP (Network Control Protocol) khác nhau cho mỗi giao thức mạng.

Để hiểu rõ về giao thức PPP, ta xét trường hợp quay số nối kết máy tính ở nhà vào mạng của một ISP.

Đầu tiên máy tính các nhân sẽ quay số thông qua modem đến router của ISP. Router sẽ tiếp nhận cuộc gọi và một nối kết vật lý được hình thành. Máy tính sẽ gởi một loạt các gói tin theo giao thức LCP trong một hoặc nhiều khung của giao thức PPP để thỏa thuận về các thông số mà PPP sẽ sử dụng.

Sau đó một loạt các gói tin của giao thức NCP sẽ được gởi đi để thực hiện cấu hình tầng mạng. Thông thường máy tính muốn sử dụng giao thức TCP/IP nên nó cần một địa chỉ IP. Giao thức NCP sẽ gán địa chỉ IP cho máy tính. Từ lúc này, máy tính đóng vai trò như một máy trên mạng Internet. Nó có thể gởi và nhận các gói tin của giao thức IP. Khi người dùng kết thúc, NCP xóa đi nối kết của tầng mạng và giải phóng địa chỉ IP của máy tính để sử dụng cho các máy tính khác nối vào sau đó. Giao thức LCP sẽ xóa nối kết của tầng liên kết dữ liệu. Và cuối cùng máy tính sẽ yêu cầu modem kết thúc cuộc gọi (Hang up) và giải phóng nối kết ở tầng vật lý.

Khung của giao thức PPP tương tự như khung của giao thức HDLC, tuy nhiên đây là khung theo kiểu hướng ký tự. Nó sử dụng kỹ thuật byte độn.



Cấu trúc khung của giao thức PPP

PPP sử dụng byte đặc biệt 01111110 để làm cờ đánh dấu điểm bắt đầu và kết thúc của khung.

Địa chỉ 11111111 để chỉ rằng tất cả các trạm đều nhận khung. Nhờ đó giao thức LCP không cần thiết phải đánh địa chỉ cho các trạm.

Trường Control có giá trị 00000011 để biểu thị rằng giao thức không sử dụng cơ chế báo nhận dựa trên số thứ tự của khung.

Trường Protocol để xác định phần gói tin được chứa đựng trong phần Payload được định nghĩa bởi giao thức mạng nào. Mỗi protocol đã được qui định một giá trị riêng. Bit đầu

tiên là 0 được sử dụng cho các giao thức mạng IP, IPX, OSI CLNP, XNS. Kích thước mặc định là 2 bytes, tuy nhiên giao thức LCP có thể thỏa thuận để sử dụng 1 byte.

Payload là nơi chứa gói tin với chiều dài khác nhau. Chiều dài tối đa mặc định là 1500 bytes, tuy nhiên LCP có thể thỏa thuận để thay đổi.

Cuối cùng là trường checksum dùng để kiểm tra lỗi trong khung.