Các giao thức điều khiển lỗi

Bởi:

unknown

Một số giao thức điều khiển lỗi (Error Control)

Phần kế tiếp chúng ta xem xét một số giao thức cơ bản được sử dụng nhiều trong việc điều khiển lỗi. Các giao thức này được xây dựng dựa trên các giả định sau:

- Chúng ta có máy tính A muốn gởi dữ liệu cho máy tính B.
- Luôn luôn có đủ dữ liệu cho máy A gởi đi
- Các giao diện giao tiếp với tầng mạng và tầng vật lý đã được định nghĩa chuẩn.
- Bên nhận thông thường thực hiện việc chờ đợi một sự kiện nào đó phát sinh bằng cách gọi hàm wait_for_event().

Các giao thức được trình bày dưới dạng các chương trình viết bằng ngôn ngữ c. Chúng sử dụng các định nghĩa trong tập tin **protocol.h** có nội dung như sau:

```
#define MAX_PKT 1024                                  /* Kích thước tối đa của một gói tin */

typedef enum {false, true} boolean;                   /* Kiểu luận lý */
typedef unsigned int seq_nr;                          /* Số thứ tự của khung gởi hoặc khung báo nhận*/
typedef struct {unsigned char data[MAX_PKT];} packet; /* Định nghĩa kiểu của gói tin */
typedef enum {data, ack, nak} frame_kind;             /* Các loại khung */
typedef struct {                                      /* Kiểu dữ liệu của khung: */
  frame_kind kind;                                          //Loại khung
  seq_nr seq;                                               //Số thứ tự của khung gởi đi
  seq_nr ack;                                               //Số thức tự của khung muốn báo nhận
  packet info;                                              //Thông tin gởi nhận,
} frame;                                                    // là gói tin nhận của tầng mạng

/* Chờ một sự kiện xuất hiện; trả về kiểu của sự kiện */
void wait_for_event(event_type *event);
/* Nạp gói tin nhận được từ tầng mạng vào khung để gởi đi */
void from_network_layer(packet *p);
/* Chuyển dữ liệu từ khung nhận được cho tầng mạng */
void to_network_layer(packet *p);
/* Nhận khung đến từ tầng vật lý và lưu nó vào khung r */
void from_physical_layer(frame *r);
/* Chuyển một khung xuống tầng vật lý để truyền đi */
void to_physical_layer(frame *s);
/* Khởi động đồng hồ và bật sự kiện quá thời hạn cho khung thứ k đang gởi đi */
void start_timer(seq_nr k);
/* Dừng đồng hồ và tắt sự  kiện quá thời hạn cho khung thứ k đang gởi đi */
void stop_timer(seq_nr k);
/* Khởi động đồng hồ phụ và bật sự  kiện quá thời hạn cho khung phản hồi*/
void start_ack_timer(void);
/* Dừng đồng hồ phụ và tắt sự  kiện quá thời hạn cho khung phản hồi*/
void stop_ack_timer(void);
/* Cho phép tầng mạng tạo sự kiện tầng mạng đã sẵn sàng */
void enable_network_layer(void);
/* Cấm tầng mạng tạo sự kiện tầng mạng đã sẵn sàng */
void disable_network_layer(void);
/* Macro để tăng giá trị K theo kiểu quay vòng */
#define inc(k) if (k < MAX_SEQ) k = k + 1; else k = 0
```

Giao thức truyền đơn công không ràng buộc (Unrestricted Simplex Protocol)

Protocol 1 (utopia) được dùng cho việc truyền tải thông tin theo một chiều từ người gởi sang người nhận. Kênh truyền được giả định là không có lỗi và bên nhận được giả định rằng có thể xử lý được hết tất cả các thông tin gởi đến một cách nhanh chóng. Chính vì thế mà bên gởi chỉ đơn thuần thực hiện một vòng lặp đưa dữ liệu lên đường truyền với tốc độ nhanh nhất có thể.

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

void sender1(void)
{
  frame s;                          /* Vùng đệm để chứa khung gởi đi */
  packet buffer;                    /*  Vùng đệm để chứa gói tin gởi đi */

  while (true) {
      from_network_layer(&buffer); /* Nhận gói tin từ tầng mạng để gởi đi */
      s.info = buffer;              /* Đưa gói tin vào khung để gởi đi */
      to_physical_layer(&s);        /* Gởi khung xuống tầng vật lý để gởi lên đường truyền */
  }                              /



}

void receiver1(void)
{
  frame r;
  event_type event;

  while (true) {
      wait_for_event(&event);       /* Chờ sự kiện, chỉ xuất hiện khi khung đến */
      from_physical_layer(&r);      /* Nhận khung từ tầng vật lý */
      to_network_layer(&r.info);    /* Lấy thông tin ra khỏi khung và gởi lên tầng mạng */
  }
}
```

Giao thức truyền đơn công dừng và chờ (Simplex Stop-and-wait Protocol)

Giao thức Stop-and-wait cũng được thiết kế cho các cuộc truyền tải thông tin một chiều từ người gởi sang người nhận. Kênh truyền tải thông tin một lần nữa cũng được giả định rằng không có lỗi như giao thức Unrestricted Simplex Protocol. Tuy nhiên, trong trường hợp này, bên nhận chỉ có một vùng lưu trữ có khả năng hạn chế và một tốc độ xử lý giới hạn, vì thế giao thức phải được thiết kế dự phòng cho trường hợp dữ liệu máy gởi đến nhanh làm tràn vùng lưu trữ thông tin của bên nhận.

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

void sender2(void)
{
 frame s;                          /* Vùng đệm để chứa khung gởi đi */
 packet buffer;                    /* Vùng đệm để chứa gói tin gởi đi */
 event_type event;                 /* Sự kiện báo hiệu khung đến */

 while (true) {
     from_network_layer(&buffer);  /* Nhận gói tin từ tầng mạng để gởi đi */
     s.info = buffer;              /* Đưa gói tin vào khung để gởi đi */
     to_physical_layer(&s);        /* Gởi khung xuống tầng vật lý để gởi lên đường truyền */
     wait_for_event(&event);       /* Chờ sự kiện đến của khung báo nhận gởi về từ bên gởi*/
 }
}

void receiver2(void)
{
 frame r, s;
 event_type event;
 while (true) {
     wait_for_event(&event);       /* Chờ sự kiện, chỉ xuất hiện khi khung đến */
     from_physical_layer(&r);      /* Nhận khung từ tầng vật lý */
     to_network_layer(&r.info);    /* Lấy thông tin ra khỏi khung và gởi lên tầng mạng */
     to_physical_layer(&s);        /* Gởi khung báo nhận sang bên gởi */
 }
}
```

Giao thức truyền đơn công cho kênh truyền có nhiễu (Simplex Protocol for Noisy Channel)

Giả sử ta bỏ đi giả thuyết kênh truyền không có lỗi. Trong trường hợp này, với các kỹ thuật xử lý lỗi (Parity check, CRC), bên nhận có thể phát hiện ra được các khung bị lỗi. Tuy nhiên, điều gì sẽ xảy ra nếu khung gởi đi bị mất, không đến được nơi nhận. Khi đó sẽ dẫn đến tình trạng như sau:

- Người gởi không biết được khung có đến nơi nhận tốt hay không.
- Giải pháp là yêu cầu người nhận gởi các khung báo nhận thông báo về tình hình các khung bị lỗi.
- Các khung báo nhận có thể bị mất.

- Giải pháp: Mỗi khi gởi một khung đi, Bên gởi sẽ thiết lập một bộ đếm thời gian. Nếu sau một khoảng thời gian qui định mà không nhận được khung báo nhận, bên gởi sẽ gởi lại các khung không được báo nhận
- Bên nhận không phân biệt được các khung trùng lắp do bên gởi gởi lại.
- Giải pháp: Mỗi khung sẽ có một số thứ tự để phân biệt lẫn nhau. Số thứ tự này sẽ được tăng dần cho đến một giá trị cực đại sau đó lại quay về giá trị 0. Trong ví dụ sau, số thứ tự có giá trị cực đại là 1. Như vậy ta chỉ sử dụng 2 giá trị là 0 và 1 để đánh số thứ tự cho khung.

```
/* Protocol 3 (par) allows unidirectional data flow over an unreliable channel. */
#define MAX_SEQ 1                                   /* Giá trị tối đa của số thứ tự khung là 1 */
typedef enum {frame_arrival, cksum_err, timeout} event_type;
#include "protocol.h"

void sender3(void)
{
  seq_nr next_frame_to_send;                        /* Số thứ tự của gói tin của lần gởi kế tiếp */
  frame s;                                          /* Khung để gởi dữ liệu đi */
  packet buffer;                                    /* Vùng lưu trữ cho gói tin gởi */
  event_type event;

  next_frame_to_send = 0;                           /* Khởi động số thứ tự cho khung gởi */
  from_network_layer(&buffer);                      /* Nhận gói tin đầu tiên từ tầng mạng để gởi đi */
  while (true) {
      s.info = buffer;                              /* Xây dựng khung để gởi đi */
      s.seq = next_frame_to_send;                   /* Đánh số thứ tự cho khung */
      to_physical_layer(&s);                        /* Gởi khung xuống tầng vật lý để truyền đi */
      start_timer(s.seq);                           /* Nếu khung báo nhận đến chậm, tạo sự kiện time-out */
      wait_for_event(&event);                       /* Chờ một sự kiện xảy ra*/
      if (event == frame_arrival) {                 /* Nếu là sự kiện khung đến */
            from_physical_layer(&s);                        /* - Nhận khung báo nhận */
            if (s.ack == next_frame_to_send) {              /* - Nếu đúng là khung báo nhận của khung gởi */
                  stop_timer(s.ack);                              /* - Dừng đồng hồ */
                  from_network_layer(&buffer);                    /* - Nhận gói tin kế tiếp từ tầng mạng */
                  inc(next_frame_to_send);                        /* - Tăng số thứ tự của khung gởi kế */
            }
      }
  }
}

void receiver3(void)
{
  seq_nr frame_expected;                    /* Số thứ tự của gói tin chờ nhận kế tiếp */
  frame r, s;                               /* Khung nhận và khung báo nhận */
  event_type event;

  frame_expected = 0;                       /* Khởi động số thứ tự cho khung nhận */
  while (true) {
      wait_for_event(&event);               /* Chờ một sự kiện xảy ra*/
      if (event == frame_arrival) {         /* Nếu là sự kiện khung đến */
            from_physical_layer(&r);                /* - Nhận khung dữ liệu từ tầng vật lý */
            if (r.seq == frame  expected) {         /* - Nếu đúng là khung đang chờ */
                  to_network_layer(&r.info);              /* - Gởi dữ liệu nhận được lên tầng mạng */
                  inc(frame_expected);                    /* - Tăng số thứ tự của khung nhận kế */
            }
```