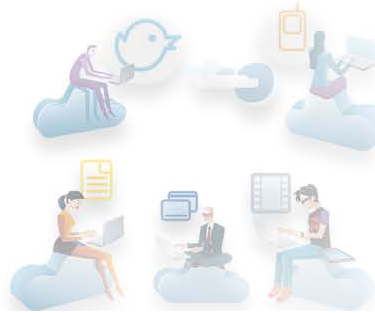
第三章 数据链路层

第 3 讲　　基 本 协 议

东南大学仪器科学与工程学院

主讲：陈熙源



基本协议

无限制单工
单工停等协议
有噪声信道的单工协议

协议定义

数据结构：

Boolean、Seq_nr、Packet、Frame_kind 和 Frame。

① **Boolean**：一个**枚举类型**，可以取值 **true 和 false** 。

② **Seq_nr**：一个**小整数**，用来**对帧进行编号**，**以便区分不同的帧。**这些序列号**从0开始，一直到（含）max_seq**。所以每个需要

③ 需要用到该数的协议都要定义它。

Packet：**同一台机器上网络层和数据链路层之间**，或者网络层对等体之间的信息交换单元

基本协议

```
/* Wait for an event to happen; return its type in event. */
void wait_for_event(event_type *event);

/* Fetch a packet from the network layer for transmission on the channel. */
void from_network_layer(packet *p);

/* Deliver information from an inbound frame to the network layer. */
void to_network_layer(packet *p);

/* Go get an inbound frame from the physical layer and copy it to r. */
void from_physical_layer(frame *r);

/* Pass the frame to the physical layer for transmission. */
void to_physical_layer(frame *s);

/* Start the clock running and enable the timeout event. */
void start_timer(seq_nr k);

/* Stop the clock and disable the timeout event. */
void stop_timer(seq_nr k);

/* Start an auxiliary timer and enable the ack_timeout event. */
void start_ack_timer(void);

/* Stop the auxiliary timer and disable the ack_timeout event. */
void stop_ack_timer(void);

/* Allow the network layer to cause a network_layer_ready event. */
void enable_network_layer(void);

/* Forbid the network layer from causing a network_layer_ready event. */
void disable_network_layer(void);

/* Macro inc is expanded in-line: Increment k circularly. */
#define inc(k) if (k < MAX_SEQ) k = k + 1; else k = 0
```

1.Wait_for_event是一个严格的循环，它等待有事情发生。

2.过程 **to_network_layer**和 **from_network_layer**是数据链路层用于向网络层传递分组，或者从网络层接受分组的。

注意：

from_physical_layer和 to_physical_layer在数据链路层和物理层之间传递帧。

基本协议

无限制的单工协议

1. 数据只能单向传输

传输方和接受方的网络层总是处于准备就绪的状态

2. 协议包括两个单独的过程：一个发送过程、一个接受过程

发送过程：在源机器的数据链路层上运行

接收过程：在目标机器的数据链路层上运行

这里没有用到序列号和确认，所以不需要MAX_SEQ。

基本协议

无限制的单工协议

两个单独的过程：

一个发送过程

无限的while循环，它尽可能快速的把数据送到线路上。

循环体包含三个动作：从网络层获取一个分组，利用变量s构造一个往外发的帧，然后通过物理层发送该帧。

一个接受过程

等待有事情发生，这里唯一可能的事件是**未损坏帧的到来。**

最终帧到达，过程wait_for_event返回，其中event=frame_arrival。

基本协议

单工的停-等协议

要处理的问题：

如何避免发送方用超过接收方处理能力的大量数据来淹没接收方。

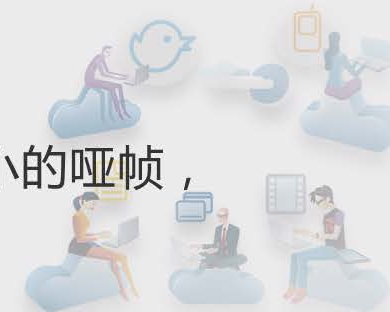
分析：

假设在接收方的硬件内没有自动缓存和排队机制的话，则发送方必须等到原来的帧被 from_physical_layer 取走以后才能发送新的帧，从而避免新的帧覆盖掉原来的帧。

解决方案:

让接收方提供反馈信息给发送方。

接收方将一个分组传递给网络层后，它给发送方送回一个小的哑帧，实际上这是给发送方一个许可，允许它发送下一帧。



基本协议

单工的停-等协议

发送方送出一帧，然后先等待一个确认，再继续发送，这样的协议称为**停-等协议**。

```
/* Protocol 2 (stop-and-wait) also provides for a one-directional flow of data from
   sender to receiver. The communication channel is once again assumed to be error
   free, as in protocol 1. However, this time, the receiver has only a finite buffer
   capacity and a finite processing speed, so the protocol must explicitly prevent
   the sender from flooding the receiver with data faster than it can be handled. */

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 */
  }
}
```

基本协议

有噪声信道的单工协议

当发送方和接受方的数据链路层在等待状态的时候，两者都有一个变量记录下有关的值。

发送方在next_frame_to_send中记录下了下一个要发送的帧的序列号

接受方在frame_expected中记录了下一个期望的序列号

每个协议在进入无限循环之前都有一个简短的初始化阶段。

作 业

在协议3中，当发送方的定时器正在运行的时候，它还有可能启动定时器吗？如果可能的话，请问这种情况是如何发生的？如果不可能的话，请问为什么这是不可能的。