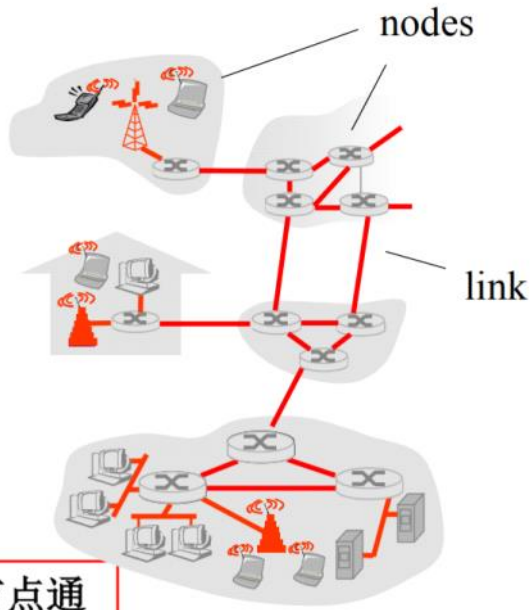
链路层

2019年3月19日　22:15

概述

一些术语:

- 主机和路由器是节点(nodes)
- 连接相邻节点的通道是链路(links)
  - 有线链路(wired links)
  - 无线链路(wireless links)
  - 局域网(LANs)
- 第2层的数据包(packet)是帧(frame)



数据链路层负责把数据包从一个节点通过链路（直连网络或物理网络）传给相邻的另一个节点。

功能

- 形成帧 (framing)
- 差错检测(error detect)：比特错，纠错
- 差错控制(error control)：丢包、重复、错序。
  流控制(flow control)
- 介质访问控制(medium access control)：多路访问，碰撞(collision)

成帧：每个帧由一个数据字段和若干首部字段组成，网络层数据包就在数据字段中

MAC协议：用于协调多个节点的帧的传输

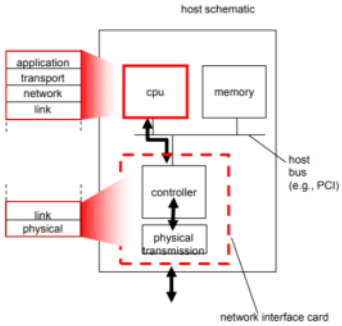
- 链路层的实现：

  链路层的主体部分是在网络适配器（network adapter）中实现的，网络适配器有时也成为网卡

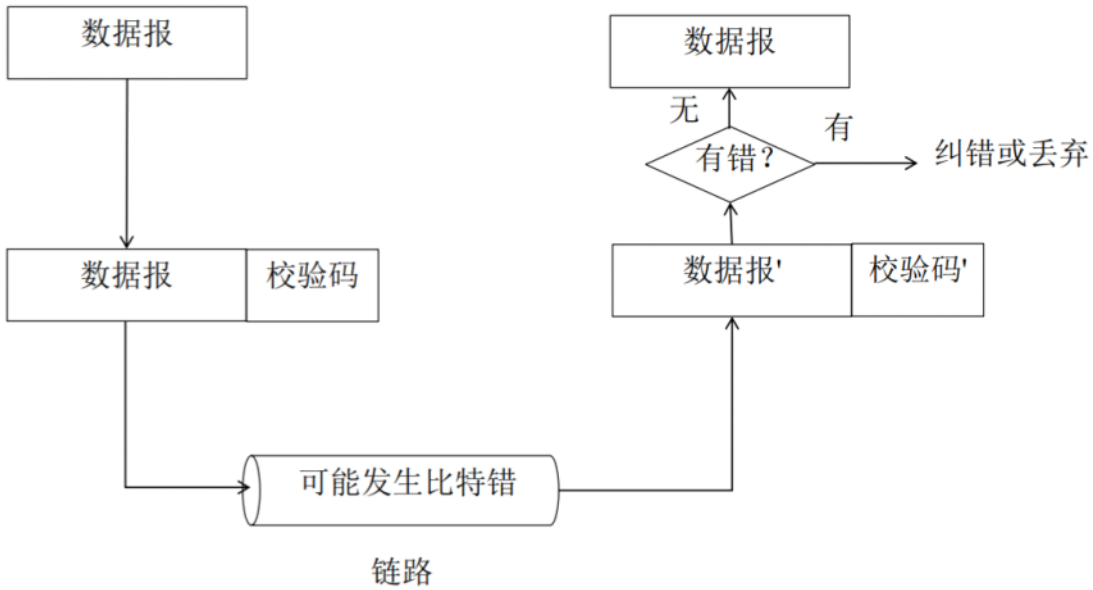
链路层的实现

网卡

- 链路层主要在网络接口卡(*network interface card*，NIC)及其驱动程序 上实现。路由器是在接口模块上实现。



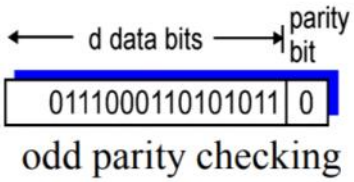
差错检测



奇偶校验

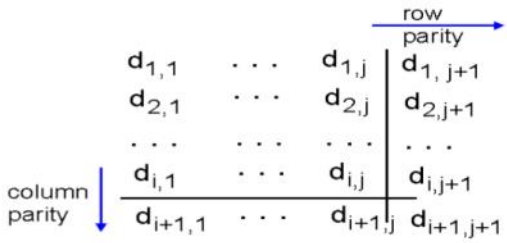
Single Bit Parity:

Detect single bit errors



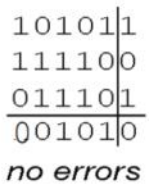
Two Dimensional Bit Parity:

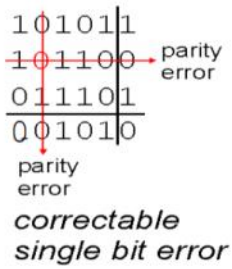
Detect and correct single bit errors



哪位有错？

1 0 1 1 1
1 0 0 0 1
0 1 1 1 1
1 0 1 0 1
1 1 0 0 0





检验和

(Checksum)

```
  10000110 10000111
+ 00000100 01000100
------------------
  10001010 11001011
+ 11000000 00000000
------------------
 1 01001010 11001011
+                  1
------------------
  01001010 11001100
```

反码：1011 010100110011

由于需要使用加法器实现算法，校验和一般不用于数据链路层，而是更高层，例如，IP层和传输层。

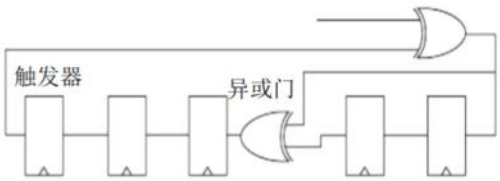
10000110 10000111 00000100 01000100 11000000 00000000 10110101 00110011

数据 校验和

每十六位相加，最高位进位原数去掉最高位的进位再加1，计算完后的值取反就是校验和

链路层常用CRC检验，因为容易用硬件实现，速度快，检错率很高。

CRC-CCITT($x^{16}+x^{12}+x^5+1$)的错误检测能力：可以检出所有随机奇数位错误和双位错，可以检出所有长度小于等于16位的突发错；对于长度等于17位的突发错误，检错率为99.9969%；长度大于等于18位的突发错误，检错率为99.9985%。



$G(X)=X^5+X^2+1$

现今的计算机网络中广泛应用的差错检测技术基于**循环冗余检测**（Cyclic Redundancy Check，CRC）**编码**。CRC 编码也称为**多项式编码**（polynomial code），因为该编码能够将要发送的比特串看作为系数是 0 和 1 一个多项式，对比特串的操作被解释为多项式算术。

CRC 编码操作如下。考虑 d 比特的数据 D，发送结点要将它发送给接收结点。发送方和接收方首先必须协商一个 $r+1$ 比特模式，称为**生成多项式**（generator），我们将其表示为 G。我们将要求 G 的最高有效位的比特（最左边）是 1。CRC 编码的关键思想如图 5-6 所示。对于一个给定的数据段 D，发送方要选择 r 个附加比特 R，并将它们附加到 D 上，使得得到的 $d+r$ 比特模式（被解释为一个二进制数）用模 2 算术恰好能被 G 整除（即没有余数）。用 CRC 进行差错检测的过程因此很简单：接收方用 G 去除接收到的 $d+r$ 比特。如果余数为非零，接收方知道出现了差错；否则认为数据正确而被接收。

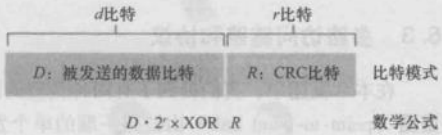


图 5-6 CRC

现在我们回到发送方怎样计算 R 这个关键问题上来。前面讲过，我们要求出 R 使得对于 n 有：

$$D \cdot 2^r \text{ XOR } R = nG$$

也就是说，我们要选择 R 使得 G 能够除以 $D \times 2^r$XOR R 而没有余数。如果我们对上述等式的两边都用 R 异或（即用模 2 加，而没有进位），我们得到

$$D \cdot 2^r = nG \text{ XOR } R$$

这个等式告诉我们，如果我们用 G 来除 $D \cdot 2^r$，余数值刚好是 R。换句话说，我们可以这样计算 R：

$$R = \text{remainder}\frac{D \cdot 2^r}{G}$$

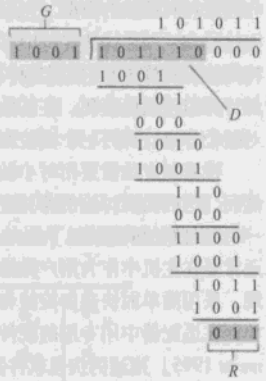


图 5-7 一个简单的 CRC 计算

图 5-7 举例说明了在 $D=101110$，$d=6$，$G=1001$ 和 $r=3$ 的情况下的计算过程。在这种情况下传输的 9 个比特是 101110011。你应该自行检查一下这些计算，并核对一下 $D \cdot 2^r = 101011 \cdot G$ XOR R 的确成立。

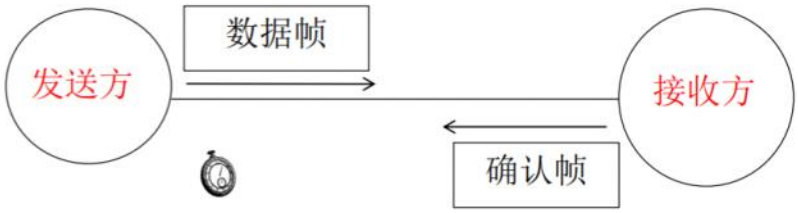
国际标准已经定义了 8、12、16 和 32 比特生成多项式 G。CRC-32 32 比特的标准被多种链路级 IEEE 协议采用，使用的一个生成多项式是：

$$G_{\text{CRC-32}} = 100000100110000010001110110110111$$

每个 CRC 标准都能检测小于 $r+1$ 比特的突发差错。（这意味着所有连续的 r 比特或者更少的差错都可以检测到。）此外，在适当的假设下，长度大于 $r+1$ 比特的突发差错以概率 $1-0.5^r$ 被检测到。每个 CRC 标准也都能检测任何奇数个比特差错。有关 CRC 检测实现的讨论可参见［Williams 1993］。CRC 编码甚至更强的编码所依据的理论超出了本书的范围。教科书［Schwartz 1980］对这个主题提供了很好的介绍。

可靠数据传输

(Reliable Data Transfer)



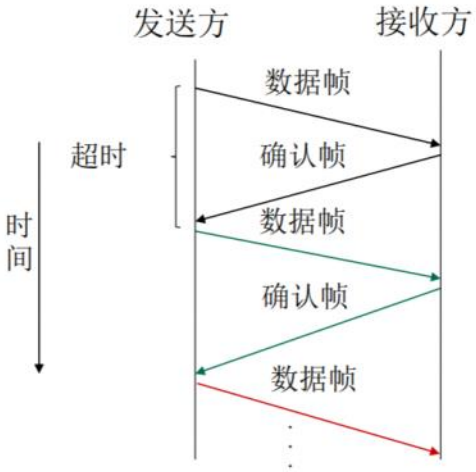
超时(timeout)

如果不纠错，出现比特错的帧将被丢弃，即出现丢包错误(loss)。

自动重发请求(Automatic Repeat reQuest，ARQ)，每发送一帧都启动一个超时定时器，哪个帧超时将重传该帧，并重启其定时器。后面讲的停等协议和滑动窗口协议都是ARQ协议。

确认帧(Acknowledgement Frame)是接收方发给发送方用来表示已经收到数据帧的一个控制帧。

停等协议（stop-and-wait）



只有收到前一个数据帧的确认才可以发送下一个数据帧。

例子: 1000Mbps 链路, 15 ms传播延迟, 1KB的帧，停等协议，该链路的吞吐量是多少？

帧长L=1KB

数据传输率R=1000Mbps

往返时间(RTT)=2*15ms=30ms

发送延迟(L/R)=1KB/1Gbps = 0.008ms

吞吐量(throughput)=L/(RTT+L/R)

=1KB/30.008ms

=0.266Mbps

循环使用 2个

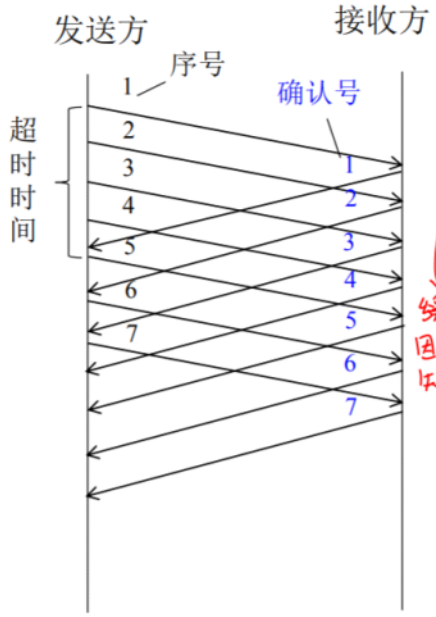
错误：数据帧丢失，确认帧丢失，超时

序号问题：至少需要几个序号？

区分重传的数据/确认

滑动窗口协议

(Sliding Window)

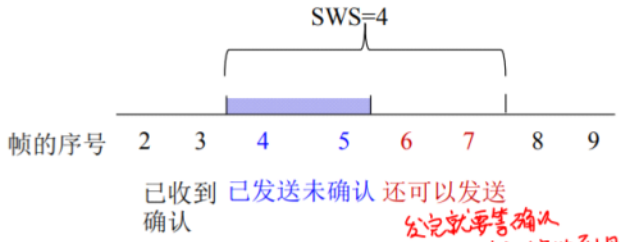


使用**滑动窗口协议**时不需要等待前面发送的帧的确认回来，就可以连续发送多个帧，其个数由发送窗口来控制。

发送窗口(Sending Window)是个连续发送数据帧的可用序号范围，主要用于流控制(flow control)。

发送窗口大小(Sending Window Size, SWS)表示发送窗口的大小，也是发送缓冲区的大小。

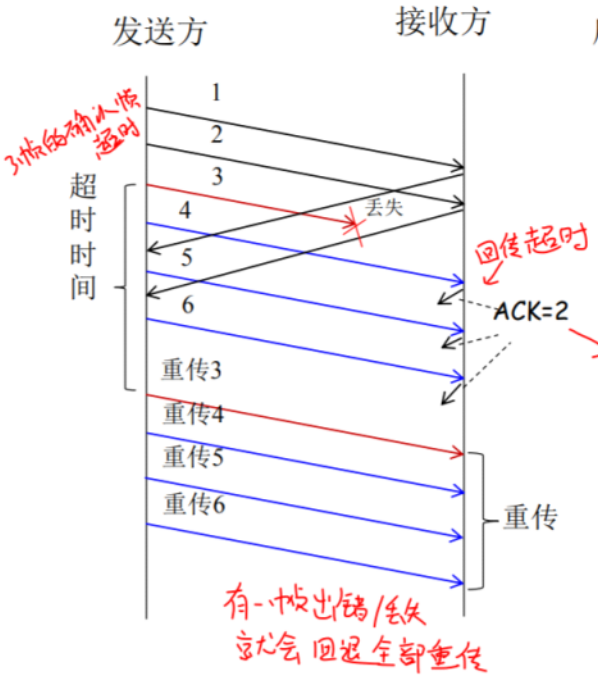
这里的确认帧(Acknowledgement Frame)为确认号为ACK的帧，用于接收方通知发送方其发送的序号为ACK的数据帧以及更早发送的数据帧已经全部收到并已交给上层协议。



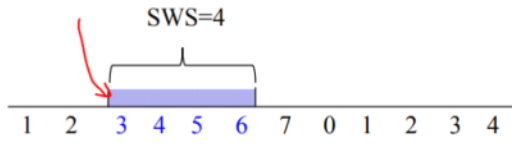
其中一种滑动窗口协议

回退N协议

(go back N)

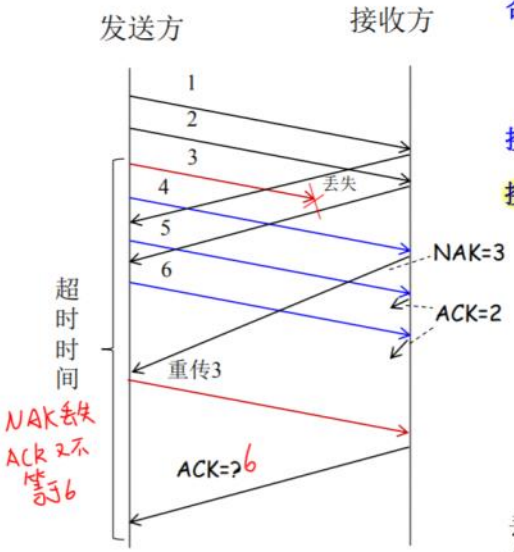


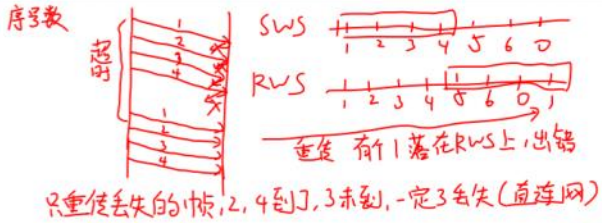
序号可以循环使用



选择性重传

(Selective Repeat)

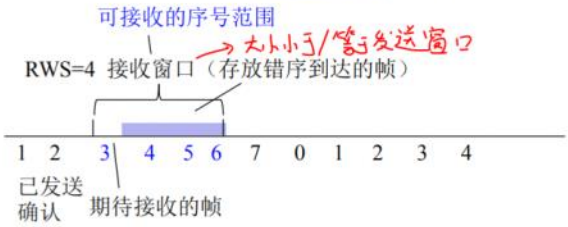




否定性确认帧(Negative Acknowledgement,NAK)用于表示这一帧之前的数据帧全部收到并已交给上层协议，要求发送方重传这一帧。每个帧只发送一次NAK。

接收窗口用于确定应该保存哪些帧，用序号范围表示。

接收窗口大小(Receiving Window Size, RWS)表示接收窗口的大小，也是接收缓冲区的大小。



丢失NAK是否会出错？丢失多个帧怎么办？没有后续帧会出错吗？ 如何确保不发生超时重传？

提高滑动窗口协议的效率

选择性确认(Selective Acknowledgement)

接收方把已收到的帧的序号告诉发送方。

捎带确认(Piggybacking)

通信双方其实是全双工方式工作。接收方在发数据给对方时顺便把确认号也告诉对方。

延迟确认(Delayed Acknowledgement)

接收方收到一帧后并不立即发送确认帧，而是等待一段时间再发送。

PPP 协议

RFC 1557

- PPP协议(Point-to-Point Protocol) 是点到点网络的数据链路层协议。
- PPP协议是根据HDLC(High-Level Data Link Control)协议进行设计的，主要用于串行电缆(V.35)、电话线(MODEM) 等各种串行链路。
- PPP协议可以提供连接认证、传输加密和压缩功能。
- PPP协议可以为各种网络层协议提供服务：因特网的IP协议、Novell公司的IPX协议、苹果公司的AppleTalk协议。
- PPP协议的多链路捆绑技术可以通过将通信两端之间的多条通信链路捆绑成一条虚拟的链路而达到扩充链路可用带宽的目的。
- ADSL的PPPoE协议和VPN中的PPTP协议等采用了PPP协议进行封装。

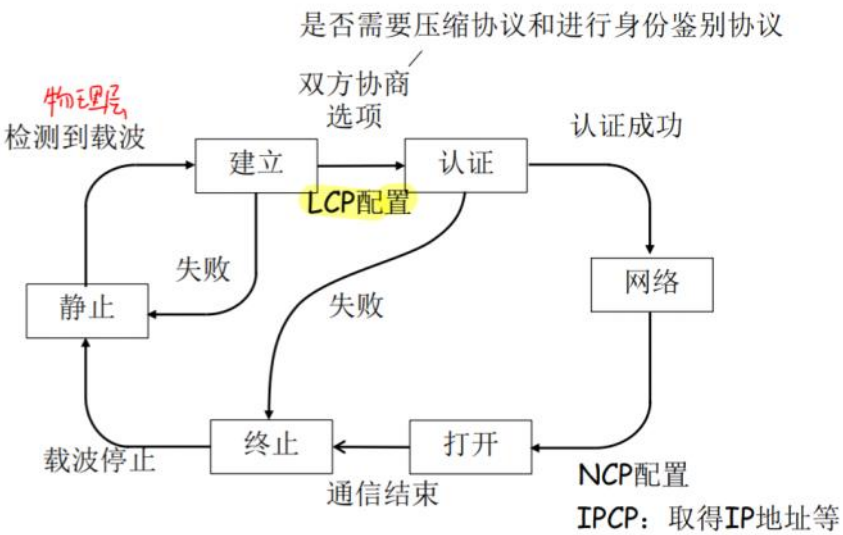
PPP数据帧

划分一帧

字节：1	1	1	1 或 2	≤1500	2 or 4	1
标志	地址	控制	协议	数据	校验码	标志
01111110	0xFF	0x03			CRC-16/CRC-32	01111110

- PPP协议采用HDLC的广播地址(0xFF)和无编号帧(0x03)。
- PPP协议采用字节填充法(Byte-stuffing): 信息字段出现的标志字节(0x7E)用0x7D-5E 替换，0x7D用0x7D-5D替换。所有小于0x20的字节(控制字节)都加上值0x20，并在前面加上转义字符0x7D，例如，0x01用0x7D-21替换。
- PPP协议的16位CRC校验码的除数为$x^{16} + x^{12} + x^5 + 1$。
- 协议字段(Protocol)指明上层协议，例如, 0x802B-IPX，0x0021-IP，0xC021-LCP，0x8021-IPCP等 。
- 可以省略地址和控制字节(头部压缩)。PPP协议还可以进行TCP压缩和数据压缩。
- PPP协议没有纠错功能，也没有流控制和确保有序的功能。

PPP协议的状态图



Link Control Protocol(LCP)
Network Control Protocol (NCP): IPCP(Internet)，IPXCP(Novell)