第 3 章 数据链路层

本章将从差错产生的原因与差错控制方法入手，讨论基于点-点通信线路的数据链路层的基本概念与服务功能，以及典型的数据链路层协议。

本章教学要求

- 理解：数据传输过程中差错产生的原因与性质。
- 掌握：误码率的定义与差错控制方法。
- 掌握：数据链路层的基本概念。
- 了解：数据链路层协议的分类方法。
- 掌握：典型的数据链路层协议——PPP 基本工作原理。

3.1 差错产生的原因与差错控制方法

3.1.1 设计数据链路层的原因

在讨论数据链路层的基本概念与协议前，需要讨论一个问题，那就是为什么要设计数据链路层？这个问题可以从以下三个方面来回答。

(1) 物理线路由传输介质与通信设备组成。在物理线路上传输数据信号是存在差错的。误码率是指二进制比特在数据传输过程中被传错的概率。在实际物理线路的传输过程中，人们需要进行大量测试，求出各种物理线路的平均误码率，或者给出某些特殊情况下的平均误码率。测试结果表明：电话线路的传输速率在 300～2400bps 时，平均误码率在 10^{-4}～10^{-6}之间；传输速率在 4800～9600bps 时，平均误码率在 10^{-2}～10^{-4}之间。由于计算机网络对数据通信的要求是平均误码率必须低于 10^{-9}，因此普通电话线路不采取差错控制措施就不能满足计算机网络的要求。

(2) 设计数据链路层的主要目的是在有差错的物理线路的基础上，采取差错检测、差错控制与流量控制等方法，将有差错的物理线路改进成无差错的数据链路，向网络层提供高质量的数据传输服务。

(3) 从参考模型的角度来看，物理层以上的各层都有改善数据传输质量的责任，数据链路层是最重要的一层。

3.1.2 差错产生的原因和差错类型

我们将通过物理线路传输之后接收数据与发送数据不一致的现象称为传输差错(简称差错)。差错的产生是不可避免的,我们的任务是分析差错产生的原因与类型,研究检查是否出现差错以及如何纠正差错的差错控制方法。

差错的产生过程如图 3-1 所示。其中,图 3-1(a)表示的是数据通过通信信道的过程;图 3-1(b)表示的是数据传输过程中噪声的影响。

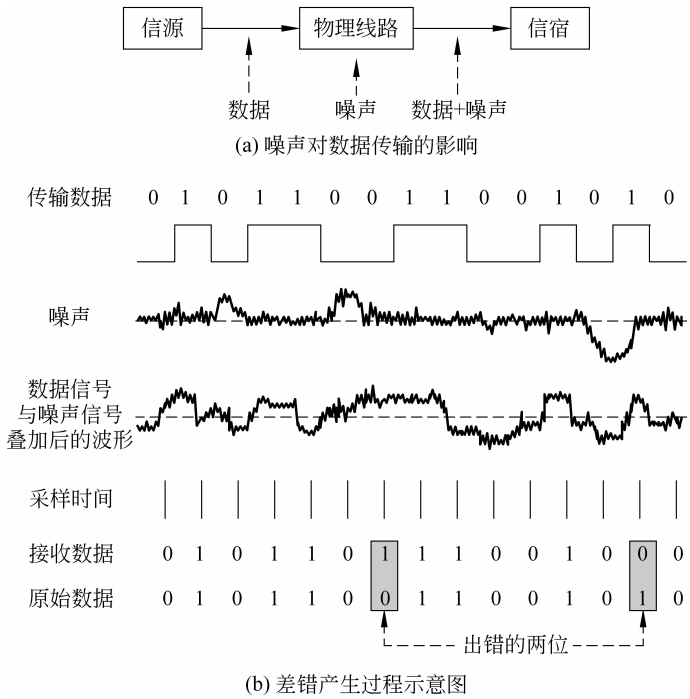


图 3-1 差错的产生过程

当数据信号从发送端出发经过物理线路时,由于物理线路存在着噪声,因此数据信号通过物理线路传输到接收端时,接收信号必然是数据信号与噪声信号电平的叠加。在接收端接收电路在取样时对叠加后的信号进行判断,以确定数据的 0、1 值。如果噪声对信号叠加的结果在电平判决时引起错误,这时就会产生传输数据的错误。

物理线路的噪声分为两类:热噪声和冲击噪声。其中,热噪声是由传输介质导体的电子热运动产生的。热噪声的特点是:时刻存在,幅度较小,强度与频率无关,但是频谱很宽。热噪声是一种随机的噪声,由热噪声引起的差错是一种随机差错。

冲击噪声是由外界电磁干扰引起的。与热噪声相比,冲击噪声的幅度比较大,它是引起传输差错的主要原因。冲击噪声持续时间与数据传输中每比特的发送时间相比可能较长,因此冲击噪声引起的相邻多个数据位出错呈突发性。冲击噪声引起的传输差错是一种突发差错。引起突发差错比特位的长度称为突发长度。通信过程中产生的传输差错是由随机差错与突发差错共同构成的。

3.1.3 误码率的定义

误码率是指二进制比特在数据传输系统中被传错的概率,它在数值上近似等于 $P_e = N_e/N$。其中,N 为传输的二进制比特总数,N_e 为被传错的比特数。

在理解误码率的定义时,需要注意以下几个问题。

(1) 误码率是衡量数据传输系统正常工作状态下传输可靠性的参数。数据信号在物理线路传输过程中一定会因为噪声、干扰等原因出现错误,传输错误是正常并且是不可避免的,但是一定要控制在一个允许的范围内。

(2) 对于一个实际的数据传输系统,不能笼统地说误码率越低越好,要根据实际传输要求提出误码率要求。在数据传输速率确定后,要求传输系统的误码率越低,则传输系统的设备就会越复杂,相应造价也就越高。

(3) 对于实际数据传输系统,如果传输的不是二进制数,需要折合成二进制数来计算。

(4) 差错的出现具有随机性,在实际测量一个数据传输系统时,只有被测量的传输二进制位数越大,才会越接近真实的误码率值。

3.1.4 检错码与纠错码

在计算机通信中,研究检测与纠正比特流传输错误的方法称为差错控制。差错控制的目的是减少物理线路的传输错误,目前还不可能做到检测和校正所有的差错。人们在设计差错控制方法时提出以下两种策略。

(1) 第一种策略是采用纠错码。纠错码为每个传输单元加上足够多的冗余信息,以便接收端能够发现,并能够自动纠正传输差错。

(2) 第二种策略采用检错码。检错码为每个传输单元加上一定的冗余信息,接收端可以根据这些冗余信息发现传输差错,但是不能确定是哪一位或哪些位出错,并且自己不能够自动纠正传输差错。

纠错码方法虽然有优越之处,但是实现起来困难,在一般的通信场合不易采用。检错码方法虽然需要通过重传机制达到纠错目的,但是工作原理简单,实现起来容易,因此得到了广泛的使用。

3.1.5 循环冗余编码工作原理

常用的检错码主要有奇偶校验码和循环冗余编码。奇偶校验码是一种最常见的检错码,它分为垂直奇偶校验、水平奇偶校验与水平垂直奇偶校验(即方阵码)。奇偶校验方法简单,但检错能力差,一般只用于通信要求较低的环境。目前,循环冗余编码(cyclic redundancy code,CRC)是应用最广泛的检错码编码方法,它具有检错能力强与实现容易的特点。

1. CRC 的基本工作原理

CRC 检错方法的工作原理可以从发送端与接收端两个方面进行描述。

(1) 发送端将发送数据比特序列当作一个多项式 $f(x)$,用双方预先约定的生成多项式 $G(x)$去除,求得一个余数多项式 $R(x)$。将余数多项式加到数据多项式之后,一起发送到接收端。

(2) 接收端用同样的生成多项式 $G(x)$去除接收到的数据多项式 $f'(x)$,得到计算余数多项式 $R'(x)$。如果计算余数多项式 $R'(x)$与接收余数多项式 $R(x)$相同,表示传输无差错;否则,表示传输有差错。出现差错,通知发送端重传数据,直至正确为止。

图 3-2 给出了 CRC 检错方法的工作原理示意图。

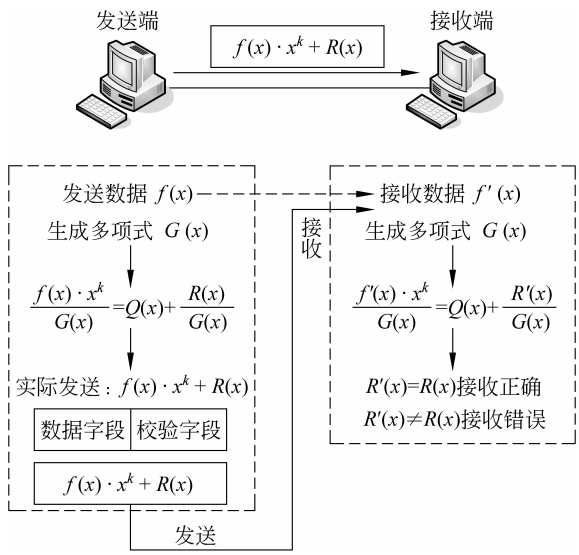


图 3-2 CRC 校验工作原理示意图

CRC 生成多项式 $G(x)$由协议来规定,$G(x)$的结构及检错效果是经过严格的数学分析与实验后确定的。目前,已有多种生成多项式列入国际标准,例如:

CRC-12 $G(x)=x^{12}+x^{11}+x^3+x^2+x+1$

CRC-16 $G(x)=x^{16}+x^{15}+x^2+1$

CRC-CCITT $G(x)=x^{16}+x^{12}+x^5+1$

CRC-32 $G(x)=x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$

2. CRC 校验的工作过程

CRC 校验的工作过程如下。

(1) 发送端发送数据多项式 $f(x)\cdot x^k$,其中 k 为生成多项式的最高幂值。对于二进制乘法来说,$f(x)\cdot x^k$ 的意义是将发送数据比特序列左移 k 位,用来放入余数。

(2) 将 $f(x)\cdot x^k$ 除以生成多项式 $G(x)$,得

$$f(x)\cdot x^k/G(x)=Q(x)+R(x)/G(x)$$

其中,式中 $R(x)$为余数多项式。

(3) 将 $f(x)\cdot x^k+R(x)$作为整体,发送到接收端。

(4) 接收端对接收到的数据多项式 $f'(x)$采用同样的运算,即

$$f'(x)\cdot x^k/G(x)=Q(x)+R'(x)/G(x)$$

求得计算余数多项式 $R'(x)$。

(5) 如果计算余数多项式 $R'(x)$等于接收余数多项式 $R(x)$,表示发送过程中没有出现

差错;如果计算余数多项式 $R'(x)$ 不等于接收余数多项式 $R(x)$,表示发送过程中出现了差错。

3. CRC 检错方法的举例

实际的 CRC 校验码生成是采用二进制的模二算法(即减法不借位、加法不进位)计算出来的,这是一种异或操作。下面通过一些例子来进一步解释 CRC 的基本工作原理。

(1) 需要注意的问题

在用模二算法生成 CRC 校验码时,需要注意以下问题。

① 以 CRC-12 为例,$G(x)=x^{12}+x^{11}+x^3+x^2+x+1$,可以写为

$$G(x)=1\times x^{12}+1\times x^{11}+0\times x^{10}+0\times x^9+0\times x^8+0\times x^7+0\times x^6+0\times x^5+0\times x^4+1\times x^3+1\times x^2+1\times x+1\times x^0$$

尽管 CRC-12 的最高位是 x^{12},$k=12$。而实际上用二进制表示时,它的位数 $N=13$,也就是说用二进制表示 $G(x)$ 应该是:1100000001111。$k=13-1=12$。

② 如果在例子中给出生成多项式比特序列为 11001,那么写成生成多项式应该为

$$G(x)=1\times x^4+1\times x^3+0\times x^2+0\times x^1+1\times x^0$$

生成多项式的 $N=5$,$k=5-1=4$。

(2) 举例

下面举一个例子来具体说明 CRC 校验码的生成过程。

① 发送数据比特序列为 110011(6 比特)。

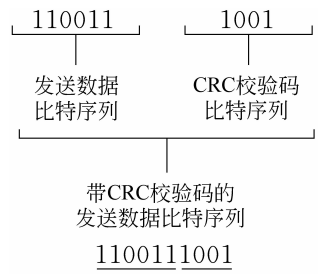
② 生成多项式比特序列为 11001($N=5$,$k=4$)。

③ 将发送数据比特序列乘以 2^4,那么产生的乘积应为 1100110000。

④ 将乘积用生成多项式比特序列去除,按模二算法求得余数比特序列为 1001。

```
                    100001   ←— Q(x)
G(x) —► 11001 √1100110000   ←— f(x)·x^k
                11001
                -----
                 10000
                 11001
                 -----
                  1001   ←— R(x)
```

⑤ 将余数比特序列加到乘积中得:



⑥ 如果在数据传输过程中没有发生错误,接收端收到的带有 CRC 校验码的数据比特序列一定能被相同的生成多项式整除,即:

$$
\begin{array}{r}
100001 \\
11001\ \overline{)\,1100111001} \\
11001 \\
\hline
11001 \\
11001 \\
\hline
0
\end{array}
$$

在实际应用中，CRC 校验码的生成与校验过程可以用软件或硬件来实现。目前，很多超大规模集成电路芯片可以实现复杂的 CRC 校验功能。

4. CRC 的检错能力

CRC 校验码的检错能力很强，它除了能够检查出离散错外，还能够检查出突发错。突发错是指在接收的二进制比特流中突然出现连续的几位或更多位数的错误。CRC 校验码具有以下检错能力。

(1) 能够检查出全部离散的 1 位错。

(2) 能够检查出全部离散的 2 位错。

(3) 能够检查出全部奇数位错。

(4) 能够检查出全部长度小于或等于 k 位的突发错。

(5) 能以$[1-(1/2)^{k-1}]$的概率检查出长度为 $k+1$ 位的突发错。

如果 $k=16$，CRC 校验码能够检查出小于或等于 16 位的所有突发错，并能以 $1-(1/2)^{16-1}\approx 99.997\%$的概率检查出长度为 17 位的突发错，漏检概率为 0.003%。

3.1.6 差错控制机制

接收端通过检错码检查数据帧是否出错，一旦发现错误，通常采用反馈重发(automatic request for repeat，ARQ)方法来纠正。图 3-3 给出了反馈重发纠错过程示意图。

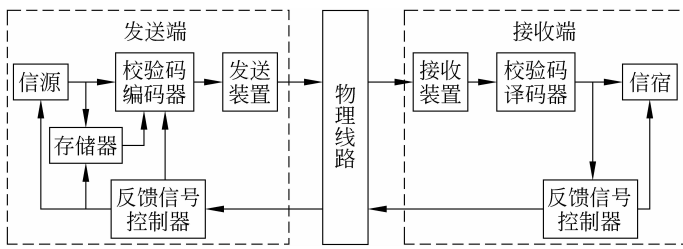


图 3-3 反馈重发纠错过程示意图

(1) 发送端将数据经过校验码编码器产生校验字段，并将校验字段与数据一起通过物理线路发送到接收端。为了适应反馈重发的需要，发送端在存储器中保留发送数据的副本。

(2) 接收端通过校验码译码器判断数据传输中是否出错。如果数据传输正确，接收端通过反馈信号控制器向发送端发送“传输正确”(ACK)信息。发送端的反馈信号控制器收到 ACK 信息后，将不再保留发送数据的副本。如果数据传输不正确，接收端向发送端发送“传输错误”(NAK)信息。

(3) 发送端的反馈信号控制器收到 NAK 信息后，将根据保留数据的副本重新进行发

送,直到正确接收为止。协议规定了最大重发次数。如果超过协议规定的最大重发次数,接收端仍然不能正确接收,那么发送端停止重传,并将向高层协议报告传输出错信息。

3.2 数据链路层的基本概念

3.2.1 链路与数据链路

链路(link)与数据链路(data link)的含义是不同的,图 3-4 给出了链路与数据链路关系的示意图。

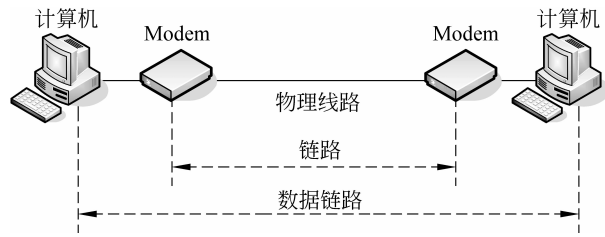


图 3-4 物理线路与数据链路的关系

理解链路与数据链路的区别与联系,需要注意以下问题。

(1) 链路是由物理线路与通信设备构成的。物理线路可以是有线的,也可以是无线的。物理线路连接相邻的两个结点,中间没有其他的交换结点或转接结点。

图 3-4 给出了一个简单的例子,那就是连接收发双方的物理线路是电话线。由于电话线是用来传输模拟语音信号的,在电话线上传输计算机产生的数字信号就必须使用调制解调器 Modem,实现数字信号与模拟信号之间的转换。收发双方的物理层通过电话线与 Modem 完成比特流的传输。因此,电话线与 Modem 就构成了连接收发双方物理层,实现比特流传输的链路。

(2) 在没有采取差错控制机制的链路上传输比特流是会出错的。在计算机网络中,设计数据链路层的目的就是为了发现和纠正链路在传输过程中可能出现的差错,使有差错的链路变成无差错的数据链路。数据链路是由实现数据链路层协议的硬件、软件与链路组成。

3.2.2 数据链路层的主要功能

数据链路层主要包括以下几个方面的功能。

1. 数据链路管理

当收发双方开始进行通信时,发送端需要确认接收端已经做好了接收的准备。为了做到这一点,收发双方必须事先交换必要的信息,建立数据链路连接;在数据传输过程中要维护数据链路;当通信结束时,要释放数据链路。数据链路层链路管理功能包括数据链路的建立、维护与释放。

2. 帧同步

数据链路层传输数据单元是帧。物理层的比特流是封装在帧中传输的。帧同步是指接收端应该能够从收到的比特流中正确地判断出一帧的开始位与结束位。

3. 流量控制

发送端发送数据超过物理线路的传输能力或超出接收端的帧接收能力时，就会造成链路拥塞。为了防止出现链路拥塞，数据链路层必须具有流量控制功能。

4. 差错控制

为了发现和纠正物理线路传输差错，使有差错的物理线路变成无差错的数据链路，数据链路层必须具有差错控制功能。

5. 透明传输

当传输的数据帧数据字段中出现某些特定的控制字符的二进制代码序列时就必须采取适当的措施，使接收端不至于将数据中的系统代码误认为是控制字符。数据链路层必须保证帧数据字段可以传输任意的二进制比特序列，即需要保证帧传输的“透明性”问题。

6. 寻址

在点-多点链路连接的情况下，数据链路层要保证每一帧都能传送到正确的接收端，因此数据链路层必须具备寻址的功能。

3.2.3 数据链路层与网络层、物理层的关系

1. 数据链路层与网络层的关系

数据链路层在 OSI 参考模型中处于网络层与物理层之间。网络层的主要功能是为联网计算机之间的通信寻找一条最佳的传输路径。如图 3-5 所示，如果 Internet 中的主机 A 要向主机 B 传送数据，那么主机 A 的网络层就会启动路由选择算法，找出一条到达主机 B 的传输路径。这条传输路径要经过路由器 1、路由器 2、路由器 3 的多跳才能够到达主机 B。这条传输路径是由多段链路组成。

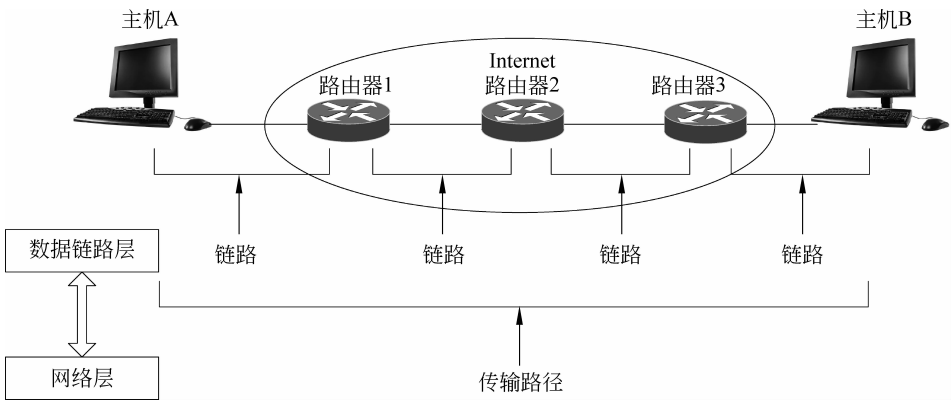


图 3-5　数据链路层与网络层关系示意图

理解数据链路层与网络层的关系需要注意以下两点。

（1）网络层路由选择算法找出的传输路径一般都是由多段链路组成。如果数据链路层要能够保证网络层的数据经过每一段链路传输时都不会出现差错，那么网络层数据经过多段链路传输也就不会出错。因此，数据链路层就是为保证网络层数据传输的正确性提供服务的。

（2）由于数据链路层的存在，网络层不需要知道物理层具体采用了哪种传输介质与通

信设备;是采用模拟通信方法,还是采用数字通信方法;使用的是有线信道,还是无线信道。只要接口关系与功能不变,物理层所采用的传输介质与通信设备的变化,对网络层不会产生影响。因此,数据链路层可以为网络层屏蔽物理层传输技术的差异性提供服务。

2. 数据链路层与物理层关系

数据链路连接建立在物理线路连接上。在物理层完成物理线路连接并提供比特流传输能力的基础上,数据链路层才能够传输数据链路层协议数据单元——帧。数据链路层协议软件控制数据链路的建立、帧传输与释放过程,同时通过流量控制与差错控制功能来保证数据在数据链路上的正确传输,为网络层提供服务。

图 3-6 给出了数据链路层与物理层协议工作过程示意图。理解数据链路层与物理层关系,需要注意以下问题。

(1) 主机 A 与主机 B 之间要传输数据,首先要建立物理线路连接。

(2) 建立物理线路连接后才能传输比特流;能够传输比特流,才能传输数据链路层的控制帧;控制帧通过协商来建立数据链路。

(3) 建立数据链路之后才能进入数据帧传输阶段。数据链路层协议使用控制帧,来协调和控制数据帧的传输过程,保证帧传输的可靠性。

(4) 当数据帧传输结束时,数据链路层协议通过控制帧来协商释放数据链路。

(5) 数据链路释放后,物理线路连接还应该存在,最后才是释放物理线路连接。

(6) 释放物理线路连接之后,主机 A 与主机 B 之间的通信关系才完全解除。

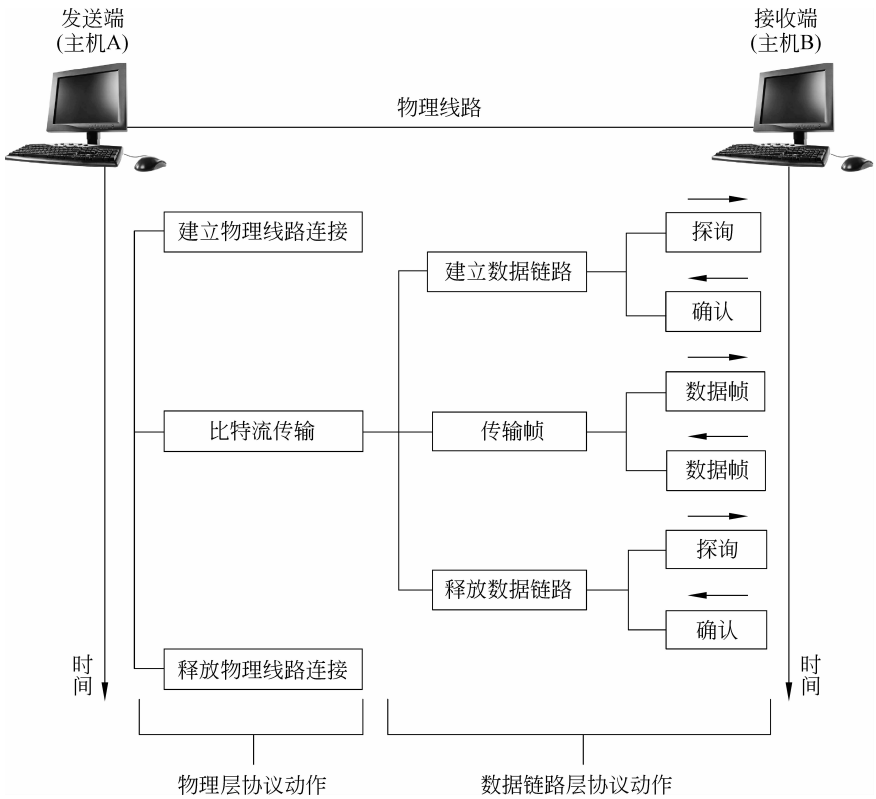


图 3-6 数据链路层与物理层关系示意图

从以上讨论中可以看出：数据链路层是在物理层比特流传输功能的基础上，为网络层提供服务。

3.3 数据链路层协议的演变与发展

实现数据链路层的功能就需要制定相应的数据链路层协议。数据链路层协议可以分为两类：面向字符型协议与面向比特型协议。首先可以从分析典型的面向字符型(Binary Synchronous Communication，BSC)协议入手，诠释数据链路层协议自然的发展与演变过程。

3.3.1 面向字符型数据链路层协议的特点

1. 控制字符编码规则

最早出现的数据链路层协议是面向字符型的协议。读者对 ACSII 码很熟悉。如果让读者设计一种数据链路层协议时，自然会想到：能不能利用 ACSII 码中的一些字符来实现数据链路控制功能呢？因为 ACSII 码早已经定义了一些控制字符。表 3-1 给出了部分控制字符意义及编码。

表 3-1 部分控制字符意义及编码

控 制 字 符	控制字符意义	二进制数
ENQ(Enquiry)	询问字符	00001011
ACK(Acknowledge)	肯定应答	00001101
NAK(Negative Acknowledge)	否定应答	00101010
SOH(Start of Head)	报头开始	00000010
EOT(End of Transmission)	传输结束	00001000
SYN(Synchronous)	同步字符	00101100
STX(Start of Text)	正文开始	00000100
ETX(End of Text)	正文结束	00000111
DLE(Data Link Escape)	转义字符	00011010

从 ASCII 码的控制字符编码表中可以查出，ENQ 的二进制编码是 0000101。需要注意的是，在面向字符型数据链路层协议中，控制字符长度是 8 位。ASCII 码只给出了低 7 位的编码，第 8 位根据编码的奇偶校验来确定。如果采用奇校验，那么 8 位控制字符 ENQ 的二进制编码是 00001011；ACK 的二进制编码是 00001101；正文传输开始 STX 的二进制编码为 00000100；正文传输结束 ETX 的二进制编码为 00000111。

2. 数据链路层协议工作阶段划分

接下来借鉴日常生活中人与人打电话的过程，设计数据链路层协议的工作流程。如果读者认真地分析面向字符型数据链路层(BSC)协议的工作流程，就一定会认同这个观点。

图 3-7 给出了面向字符型数据链路层协议的工作流程示意图。图中虚线表示控制字符

的传输,实线表示数据帧的传输。

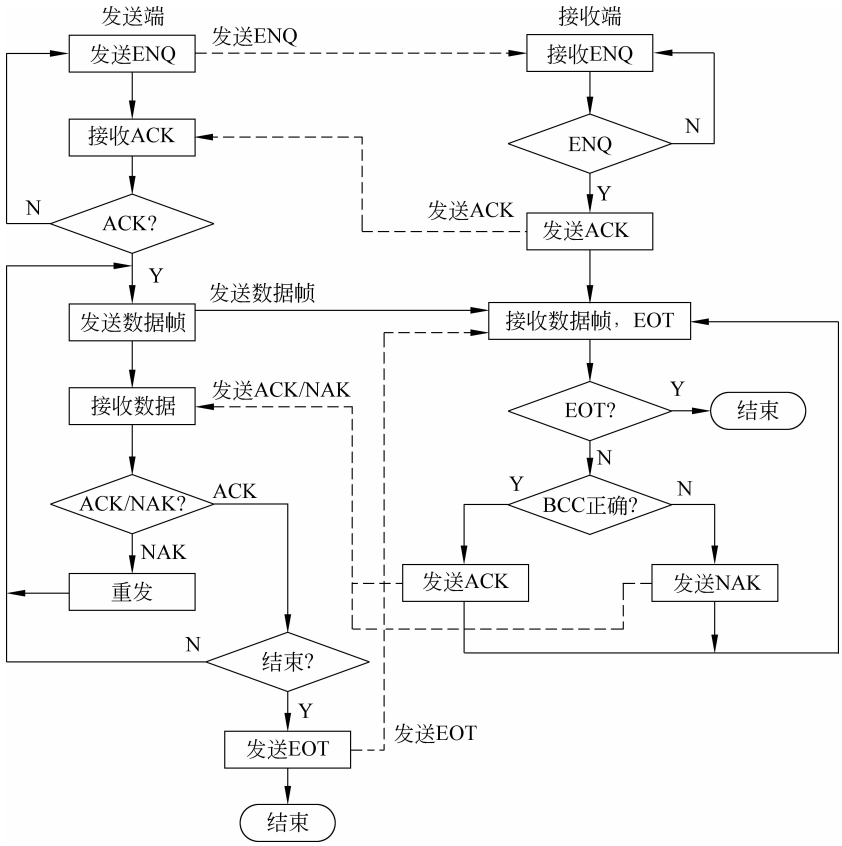


图 3-7　面向字符型数据链路层协议工作流程示意图

从图 3-7 中可以看出,数据链路层协议工作分为三个阶段。

(1) 第一阶段：建立数据链路连接

发送端与接收端开始数据传输之前,发送端通过发送控制字符 ENQ 询问接收端;接收端同意建立数据链路连接,则通过回送控制字符 ACK 确认数据链路连接,并进入准备接收帧状态。这就像人与人打电话时拨通电话,确认双方身份的过程。

(2) 第二阶段：帧传输

发送端软件按照协议规定的帧结构,将网络层传送来的 IP 分组数据封装成帧,再通过物理层发送到接收端。接收端通过校验字段 BCC 来判断帧传输是否出错。如果出错则丢弃该帧,并向发送端发送控制字符 NAK,要求发送端重发;如果正确,则向发送端发送 ACK 确认信息,并取出帧中数据字段,传送到网络层。这就像人与人打电话时,一方和另一方讲话的过程。

(3) 第三阶段：释放数据链路连接

发送端发送完数据之后,发送传输结束 EOT 控制字符给接收端;接收端同意释放数据链路连接,也要通过回送控制字符 ACK 来表示同意释放数据链路连接。这样,一次数据链路连接结束。这就像准备结束一次通话,双方协商的过程。

需要注意的是：BSC 协议规定发送端发出 ENQ 询问字符之后,必须等待接收端发回

ACK 确认字符之后才能够进入发送帧状态。当发送端发送一帧，必须等待接收端收到 ACK 或 NAK 确认字符之后，才能进入下一帧的发送。这种通信方式称为停止等待工作方式。停止等待工作方式的特点是：协议效率低，通信线路的利用率低。

3. 帧封装

面向字符型数据链路层协议中，帧可以分为两类：一类是控制帧，另一类是数据帧。控制帧一般只包含长度为 1B 的控制字符，用于数据链路的建立、释放以及帧封装与帧传输状态控制。数据帧与控制字符的结构是不相同的。网络层的 IP 分组放在帧的数据字段中，数据链路层软件需要根据协议对帧格式的规定，为 IP 分组数据加上帧头和帧尾，封装成数据帧。简化的 BSC 协议帧封装过程示意如图 3-8 所示。

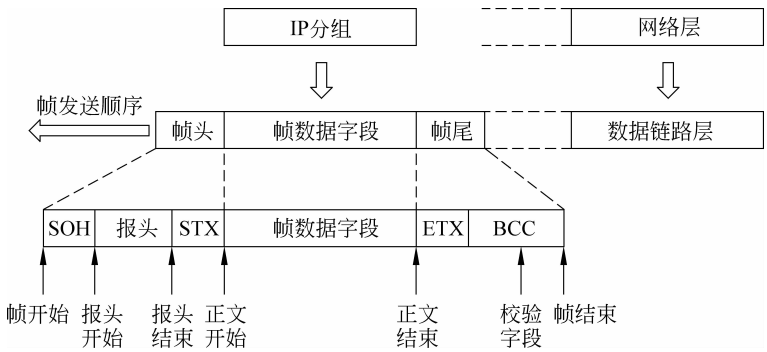


图 3-8　帧封装过程示意图

在 BSC 协议中，帧头是由 1B 的报头开始(SOH)控制字符、用户定义的报头字段以及 1B 的正文开始(STX)控制字符组成。紧接在 STX 字符之后的是帧数据字段。帧数据字段之后是正文结束(ETX)控制字符；ETX 字符之后是帧校验(BCC)字段。控制字符 ETX 与帧校验字段 BCC 组成了帧尾。

接收端软件收到 SOH 与 STX 字符之后，判断出在这两个控制字符之间的数据是用户定义的报头字段；正文开始(STX)控制字符与正文结束(ETX)控制字符之间是帧数据字段。校验(BCC)字段的数据是用来计算包含控制字符 SOH 到 ETX 之间的数据，在传输过程中是否出错。

每一种数据链路层协议都要规定帧数据字段的最大长度，即最大传输单元(maximum transfer unit，MTU)。如果网络层 IP 分组的数据大于帧数据字段的最大长度，那么就需要将网络层传送的数据分成多个传输单元，封装在多个帧中传送。

4. 数据传输的透明性问题

可以用帧数据字段定界符 STX 与 ETX 为例来说明这个问题。控制字符正文开始(STX)标志着帧数据字段传输的开始，控制字符正文结束(ETX)标志着一帧数据字段传输的结束。由于数据字段传输的开始与结束都是用专门指定的控制字符来标识的，那么专门用于表示控制字符正文开始(STX)的二进制数为 00000100 与表示正文结束(ETX)的二进制数为 00000111 在帧数据字段中就不允许出现。如图 3-9 所示，如果在数据字段中出现了 00000111 的二进制编码时，协议软件就可能错误地认为“收到控制字符 ETX，帧数据字段传输结束”，造成部分帧数据丢失，导致帧传输失败。

帧数据字段封装的是高层数据,这些数据包括文本、语音、图形、图像或视频等多种类型。这些数据中各种二进制数的组合都会存在,因此限定数据字段中不出现专用的控制字符 ETX(00000111)二进制编码是不可能的。如果将通信协议可以传输的任意比特组合数据的特性定义为"透明传输",那么面向字符型数据链路层协议存在着传输"不透明"的问题。

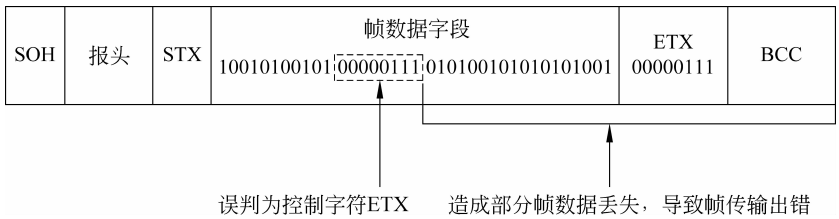


图 3-9　数据链路层协议存在传输"透明性"问题示意图

解决的方法是:定义一个转义字符 DLE。发送端软件检查出帧数据字段出现与正文结束(ETX)相同的二进制数 00000111 时,就在它的前面加入一个转义字符 DLE(00011010)。接收端软件检查出 00011010 00000111 的比特组合时,就认为前 8 比特 00011010 是转义字符,后 8 比特 00000111 不是正文结束(ETX)控制字符,输出添加的转义字符。

当然,定义专用的转义字符 DLE(00011010)同样会带来传输不透明的问题。解决的部分同样是在发送端的帧数据字段出现与转义字符 DLE 相同的 00011010 比特组合时,再添加一个转义字符 DLE,变成 00011010 00011010 比特组合。接收端软件检查出 00011010 00011010 比特组合时,就认为前 8 比特 00011010 是添加的转义字符,必须删去。

通过以上的讨论,可以看出面向字符型协议有三个明显的缺点:

(1) 用专用的控制字符实现数据链路管理,效率低。控制帧与数据帧格式不同。不同类型计算机的控制字符可能不同,同时每增加一种功能就需要定义一个新的控制字符,协议功能扩展性差。

(2) 不能实现数据的"透明传输"。

(3) 停止等待工作方式的协议效率低,通信线路利用率低。

针对这些缺点,人们研究了面向比特型数据链路层协议。最有影响的是高级数据链路控制(High-level Data Link Control,HDLC)协议。

3.3.2　面向比特型数据链路层协议的特点

掌握计算机基础知识的读者自然就会想到:既然用"字符"去控制数据链路层工作的协议效率低,那么用"比特"去控制数据链路层工作时效率会不会提高呢?研究人员正是按照这个基本的思路,开始研究面向比特型数据链路层协议的。1974 年 IBM 公司研究了数据链路层面向比特型的 SDLC 协议。1993 年,ISO 在 SDLC 的基础上修改成作为国际标准 ISO 3309 的 HDLC 协议。

1. 数据链路配置和数据传输方式

针对面向字符型协议"停止等待"工作方式的缺点,HDLC 定义了数据链路的两种基本的配置方式:非平衡配置与平衡配置,以及两种帧传送方式:正常响应模式与异步响应模式。

(1) 数据链路配置

HDLC 定义了数据链路的非平衡配置与平衡配置两种基本的配置方式。非平衡配置

根据一组结点在通信过程中的地位，分主站与从站；由主站来控制数据链路的工作过程。主站发出命令；从站接受命令，发出响应，配合主站工作。

非平衡配置又可以分为两种类型：点对点方式和多点方式。在多点方式的链路中，主站与每个从站之间都要分别建立数据链路。

平衡配置的特点是链路两端的两个站都是复合站。复合站同时具有主站与从站的功能，因此每个复合站都可以发出命令与响应。

(2) 数据传输方式

非平衡配置可以有两种数据传送方式：正常响应模式与异步响应模式。

在正常响应模式中，主站可以随时向从站传输数据帧。从站只有在主站向它发送命令帧探询，从站响应后才可以向主站发送数据帧。

在异步响应模式中，主站和从站可以随时相互传输数据帧。从站不需要等待主站发出探询就可以发送数据帧。但是，主站仍然负责数据链路的初始化、链路的建立、释放与差错控制等功能。

平衡配置只有一种数据传送方式，就是异步平衡模式。每个复合站都可以平等地发起数据传输，而不需要得到对方复合站的许可。

2. HDLC 帧结构

针对面向字符型协议“控制帧与数据帧格式不同”的缺点，HDLC 协议定义了如图 3-10 所示的帧结构，其中用 8 位控制字段 C 的 b_0 与 b_1 位来标识三种不同类型的帧：信息(I)帧、监控(S)帧与无编号(U)帧。

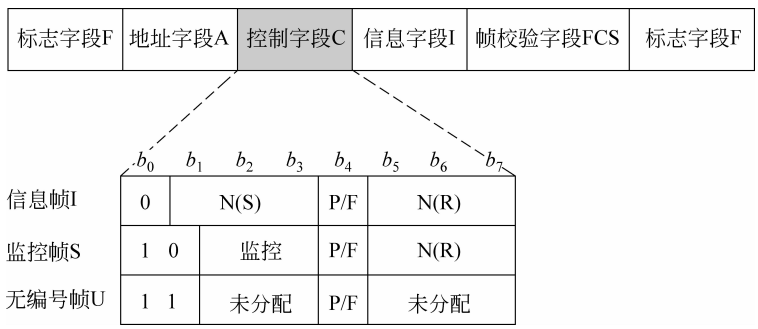


图 3-10　HDLC 帧结构

信息(I)帧用于传送网络层的 IP 分组数据。监控(S)帧用于协调发送端与接收端的信息帧传输、确认与流量控制。无编号(U)帧用于实现设置数据链路配置方式与数据传输方式等链路控制功能。三种帧的功能不同，但是帧格式相同。

HDLC 是通过在帧中规定特定的位的数值与出现的顺序来管理数据链路的工作过程的。例如，针对连续发送多个信息帧，在发送与接收过程中要对帧的顺序进行控制的需要，HDLC 用控制字段 C 的 $b_1 \sim b_3$、$b_5 \sim b_7$ 位，分别用于标识帧的发送序号 N(S)与接收序号 N(R)。因此，这类协议称为面向字符型协议。

3. 数据传输的透明性问题

物理层要解决比特流传输过程中的比特同步问题，数据链路层要解决帧同步问题。帧同步是指如何从接收到的比特流中正确判断一个帧开始和结束的位置。为此，HDLC 规定

在一个帧开头的第 1 个字节和结尾的最后 1 个字节的特殊标记。标志字段 F(flag)就是帧的开始与结束的标记。标志字段 F 为 011111110 特定的比特序列。接收端只要找到这两个标志字段,就可以确定这两个字段之间的比特流是一个帧的内容。

由于要规定一个特定字符 F 作为标志字段,那么帧的数据比特序列中就不能出现与标志字段 F 相同的比特序列,否则就会出现判断错误。也就是说,在帧开始与结束的两个 011111110 标志字段 F 之间的数据比特序列中,如果出现和标志字段 011111110 一样的比特组合时,就会误认为这个字节是帧结束标记。HDLC 也存在着同样的数据传输的透明性问题。

为了解决这个问题,HDLC 采用了 0 比特插入/删除方法。图 3-11 给出了 0 比特插入/删除方法的工作过程。

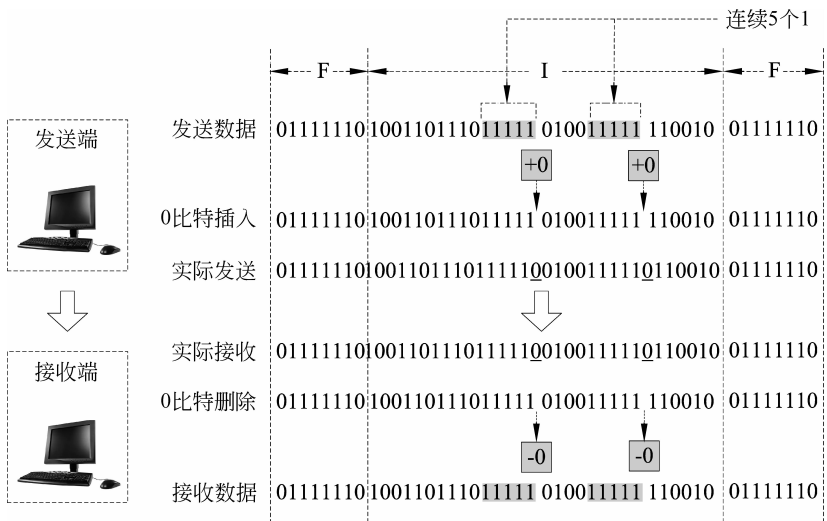


图 3-11　0 比特插入/删除方法的工作过程

0 比特插入/删除方法规定,发送端在两个标志字段为 F 之间的比特序列中,如果检查出连续的 5 个 1,不管它后面的比特位是 0 或 1,都增加 1 个 0 比特;在接收过程中,在两个标志字段为 F 之间的比特序列中检查出连续的 5 个 1 之后就删除 1 个 0 比特。在发送端经过 0 比特插入后的数据,就可以保证不会出现 6 个连续 1。在接收一个帧时,首先找到 F 字段以确定帧的起始边界,接着再对其中的比特序列进行检查,每当发现 5 个连续 1 时,就将这 5 个连续 1 后的 1 个 0 比特删除,以便将数据还原成原来的比特序列。这样就保证了在所传送的比特序列中,不管出现什么样的比特组合,也不至于引起帧边界的判断错误。采用 0 比特插入/删除方法后,帧的数据字段就可以传送任意组合的比特序列,实现了数据链路层的透明传输。

由于 HDLC 协议要针对非平衡与平衡配置、点对点方式和多点方式、正常响应与异步响应模式等情况,解决数据链路的初始化、链路的建立、释放以及差错控制、流量控制等问题,因此 HDLC 协议非常复杂。

随着通信线路质量的提高与光纤的广泛应用,简化数据链路层协议成为趋势。在实际的数据链路层软件的开发中,技术人员往往只需要遵循面向比特型数据链路层协议的基本

设计原则，根据具体的需求采用 HDLC 协议的子集，使得网络协议软件更加简洁，运行更加高效。经过多年的实践与比较，目前在数据链路层形成了两种应用广泛的协议标准。一种是用于 Internet 点-点链路的点-点协议（Point-to-Point Protocol，PPP），另一种是用于广播链路的 Ethernet 局域网 MAC 层协议。

3.4 点-点协议

3.4.1 PPP 协议的主要特点

PPP 协议广泛应用于广域网环境中主机-路由器、路由器-路由器连接以及家庭用户接入 Internet 之中，成为点-点线路中应用最多的数据链路层协议。1994 年公布的 RFC1661，以及 RFC1662、RFC1663 对 PPP 的帧结构、差错处理、IP 地址动态分配与身份认证等功能进行了详细的说明。图 3-12 以 ADSL 接入为例给出了 PPP 应用示意图。

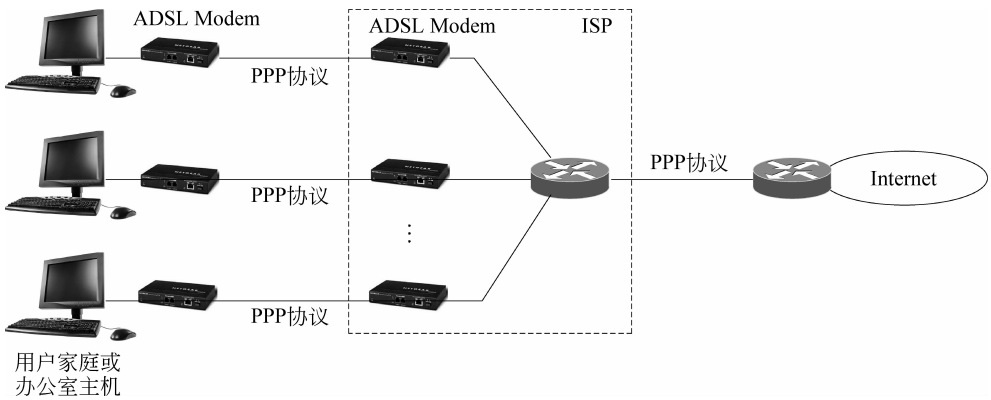


图 3-12　PPP 应用示意图

设计 PPP 协议的目标是简洁、高效与兼容。PPP 的特点主要表现在以下三个方面。

（1）在物理层只支持点-点线路连接，不支持点-多点连接；只支持全双工通信，不支持单工与半双工通信；支持异步通信或同步通信。

（2）PPP 协议只实现帧封装、传输、拆帧与校验功能；不使用帧序号，不提供流量控制功能。

（3）随着通信线路质量的提高与光纤的广泛应用，物理层误码率明显降低。同时在 TCP/IP 协议中，TCP 协议已经采取了一系列差错控制措施，因此简化数据链路层的功能已经具备条件。因此，PPP 协议只要求接收端进行 CRC 校验。判断帧传输正确，则接收该帧；如果 CRC 校验发现错误，则丢弃该帧。

理解 PPP 协议的特点还需要注意以下问题。

（1）PPP 数据链路协议与网络层协议的关系

掌握一种网络协议的特点，需要了解一些影响到协议研究的相关技术。PPP 协议标准是在 1994 年发布的，当时网络层 IP 协议仍然处于一个发展的阶段。这段时期另一种网络操作系统 NetWare 应用也很广泛，它的网络层使用的是专用的 IPX（Internet work Packet Exchange）协议。当时为了使 PPP 协议能够适应 IP 与 IPX 及其他网络层协议，PPP 协议

除了设计了链路控制协议(Link Control Protocol,LCP),还增加了网络控制协议(Network Control Protocol,NCP)。NCP协议用来建立和配置IP与IPX等不同的网络层协议。随着Internet的广泛应用,目前网络层基本上都在采用IP协议。因此,网络控制协议NCP就显得不重要了,这也为简化PPP协议编程提供了条件。

(2) PPP数据链路协议与Ethernet网的关系

随着宽带接入的主机越来越多地使用Ethernet协议,1999年公布了"运行在Ethernet上PPP协议"——PPPoE(PPP over Ethernet)协议(RFC2516文档)。PPPoE是将PPP帧封装在Ethernet帧中,使得多个PPP用户可以共享一条高带宽的Ethernet链路。目前,ADSL、Cable Modem、光纤FTTx宽带接入都使用PPPoE方式。

正是由于PPP协议具有简洁、高效,同时适应IP与Ethernet协议应用的特点,因此,PPP协议已经广泛应用于ADSL Modem电话线路接入、HFC传输网中Cable Modem同轴电缆接入、光纤接入,以及Internet路由器-路由器、主机-路由器的链路中。

3.4.2 PPP协议帧结构

1. PPP协议的帧类型

为了通过点-点的PPP链路进行通信,每个端主机首先发送LCP帧,以建立、配置和测试PPP数据链路。当PPP链路建立起来后,每个端主机发送NCP帧,以选择和配置网络层协议。当网络层协议配置好后,网络层的数据就可以通过PPP信息帧传输。因此,PPP协议的帧有三种类型:PPP信息帧、PPP链路控制帧与PPP网络控制帧。

2. PPP信息帧

图3-13给出了PPP信息帧格式示意图。PPP帧格式与HDLC帧格式类似,它由帧头、信息字段与帧尾三部分组成。

标志字段(7E)	地址字段(FF)	控制字段(03)	协议字段	信息字段	帧校验字段(FCS)	标志字段(7E)
1B	1B	1B	2B	≤1500B	2B	1B

图3-13 PPP信息帧格式

PPP信息帧头包括以下几个部分。

(1) 标志(flag)字段

标志字节长度为1个字节,用于表示帧的开始与结束。标志字节值为01111110,即0x7E。符号0x表示后面的字符用十六进制表示。

(2) 地址(address)字段

由于PPP协议只用于点-点链路,因此地址字段规定取值为11111111(即0xFF)。

(3) 控制(control)字段

控制字段长度为1个字节,规定取值为00000011(即0x03)。

(4) 协议(protocol)字段

协议字段长度为两个字节。对应三种类型帧的协议字段值分别是:

① 0x0021——表示PPP帧的信息字段是IP分组数据。

② 0xC021——表示信息字段是PPP的LCP数据帧。

③ 0x8021——表示信息字段是 PPP 帧的 NCP 数据帧。

(5) 信息字段

信息字段长度可变,最长为 1500B。

(6) 帧校验(FCS)字段

帧校验字段字段长度一般为两个字节,经过协商也可以是 4B。

3. PPP 协议的字节填充规则

需要注意的是:PPP 协议可以用于异步通信,也可以用于同步通信。

(1) PPP 协议用于异步通信

当 PPP 协议用于异步通信时,信息字段中不能出现与标志字段 0x7E 相同的值,这就是帧传输"透明性"问题。为了解决这个问题,RFC1662 定义转义字符 0x7D,并且使用字节填充。字节填充规则是:

① 在信息字段中出现的每一个 0x7E 字节,要转换成双字节 0x7D 0x5E。

② 在信息字段中出现的每一个 0x7D 字节,要转换成双字节 0x7D 0x5D。

③ 在信息字段中出现 ASCII 中控制字符(即数值小于 0x20)时,在该字符前加一个 0x7D 字节,同时改变该字节。例如,传输结束 ETX(0x03)转换后的双字节是 0x7D 0x31。

④ 由于在发送端进行字节填充,因此接收端需要检测并删除填充字节,还原成发送的值。

(2) PPP 协议用于同步通信

当 PPP 协议用于同步通信(如在 SONET/SDH 链路上使用)时,采用 0 比特插入/删除方法。

3.4.3 PPP 协议工作过程

图 3-14 给出 PPP 协议工作过程示意图。

理解 PPP 协议工作过程需要注意以下问题。

1. 链路静止与链路连接状态

在 PPP 链路起始之前与中止之后,用户计算机与 ISP 路由器之间并没有建立物理线路连接,这时候 PPP 处于链路静止(Link Dead)状态。当用户计算机(如通过 ADSL Modem)呼叫路由器时,双方建立物理层连接,PPP 进入链路连接(Link Establish)状态。

2. LCP 链路建立

PPP 帧的协议字段值为 C021H 表示链路控制帧。图 3-15 给出了 PPP 链路控制帧的格式。

(1) 链路配置协商阶段

用户计算机端 ADSL Modem 与路由器建立 PPP 链路连接时,首先向路由器发出 LCP 配置请求(configure-request)帧。LCP 配置请求帧的链路控制字段包含着特定的配置请求。LCP 配置请求包括链路最大帧长度、帧特定域的压缩、链路认证协议等。如果没有明确配置协商请求的,就采用默认值。链路最大帧长度的默认值是 1500B。在大部分情况下,用户将考虑协商双方在链路建立过程中保证安全的链路认证协议。PPP 的链路认证协议主要有两种,一是口令认证协议(Password Authentication Protocol,PAP);另一个是查询-握手认证协议(Challenge-Handshake Authentication Protocol,CHAP)。链路建立双方要

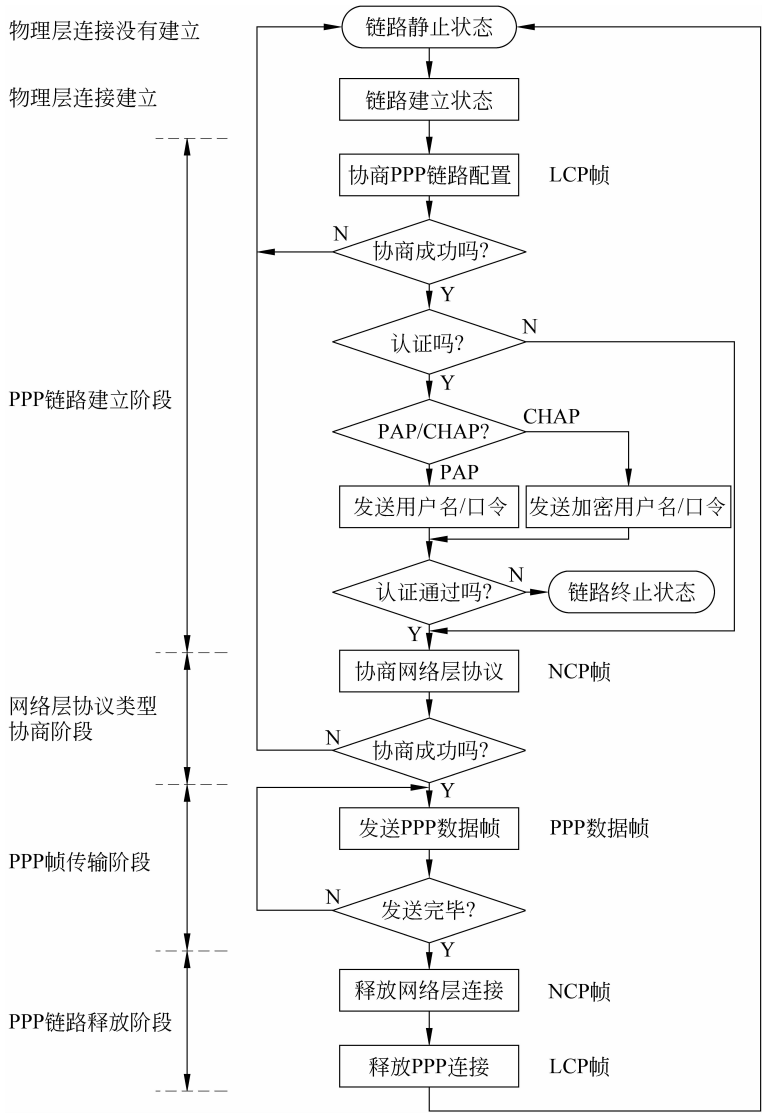


图 3-14　PPP 工作过程示意图

标志字段 (7E)	地址字段 (FF)	控制字段 (03)	协议字段 (C021)	链路控制数据	帧校验字段 (FCS)	标志字段 (7E)

图 3-15　LCP 帧格式

通过协商选择一种链路认证协议。

(2) 认证阶段

如果 LCP 配置请求协商成功,双方将根据共同选择的认证协议进入用户身份认证阶段。

PAP 认证比较简单,它通过明文的方式,由用户端向 ISP 路由器端发送用户名与口令,ISP 端与注册时存储的对应用户名与口令进行比较,如果相同则认为是合法用户。ISP 端

向用户端发出 LCP 确认帧。PAP 认证过程只需要经历两次握手，而且允许用户多次输入用户名与口令。PAP 用明文传输用户名与口令容易被窃听，允许多次输入用户名与口令容易遭到重放攻击，因此 PAP 是一种不安全的认证协议。图 3-16(a)给出了 PAP 认证过程示意图。

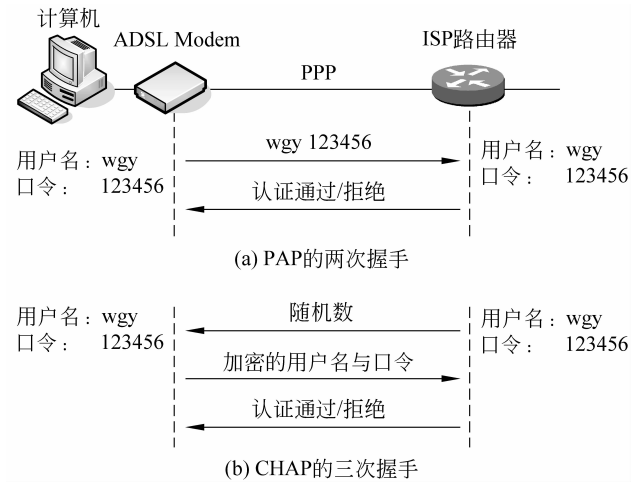


图 3-16　认证过程示意图

针对 PAP 存在的问题，CHAP 做了以下三点重要的改进。

① 用户端对发送的用户名与口令用 MD5 加密算法进行加密，在链路上传输的是加密的用户名与口令。

② CHAP 认证过程要通过三次握手。由用户端向 ISP 端发送有用户名而不包括口令的 LCP 帧，来启动 CHAP 认证过程。CHAP 认证的三次握手过程如下：

第一次握手，ISP 端向用户端发送一个查询应答 LCP 帧，帧中包括用于数字签名的随机数。

第二次握手，用户端向 ISP 端发送用随机数与 MD5 算法加密后的用户名与口令。

第三次握手，ISP 端用同样的随机数与 MD5 算法解密后，与存储的用户名与口令比较，相同则认为是合法用户，不同则认为是非法用户，ISP 端用 LCP 帧向用户端返回认证结果。图 3-16(b)给出了 CHAP 认证的三次握手过程示意图。

③ CHAP 认证在初始链路建立使用之后也需要周期性地多次进行。每次产生的随机数不同，这样做的目的是防止出现重发攻击。

如果用户身份认证失败，PPP 回到链路终止(link terminate)状态；如果认证通过，PPP 进入网络协议协商阶段。

3. NCP 帧与网络层协议协商阶段

PPP 帧的协议字段值为 8021H 表示网络控制帧。图 3-17 给出了 PPP 网络控制帧的格式。

标志字段 (7E)	地址字段 (FF)	控制字段 (03)	协议字段 (8021)	网络控制数据	帧校验字段 (FCS)	标志字段 (7E)

图 3-17　PPP 网络控制帧的格式