第3章

数据链路层

本章将会学习网络的第 2 层——数据链路层的设计原则。主要学习数据链路层中实现两台相邻机器可靠、有效的通信而涉及的一些算法。

这个问题起来很简单，机器 A 将数据放在线路上，由机器 B 完成接收。但实际上通信线路有时会出错，而且数据传输速率不高，同时在一位数据的发送时刻和接收时刻之间存在一定的传输延迟。这些限制严重影响了传输效率，通信过程的协议必须考虑这些问题。

这些协议就是本章节的主题，我们将先介绍数据链路层的设计要点，然后考虑出现错误的原因，以及如何检测和纠正这些错误来研究数据链路层的协议。然后，将由浅入深地学习几种协议，每一层次的协议都涉及了数据链路层中更多更复杂的问题。最后，将推导不同协议的模型以及正确性，同时给出协议用例。

data link

3.1 数据链路层的基本概念

网络中相邻节点在数据链路层中传输数据时，需要用到数据链路层的一些功能，这些功能包括：

(1) 向网络层提供一个定义好的服务接口；
(2) 处理传输错误；
(3) 调节数据流，确保慢速的接收方不会被快速的发送方淹没。

packet

为了实现这些功能，数据链路层从网络层获取分组信息，然后将分组封装到帧(frame)中以便传输。每一帧包含一个帧头，一个有效载荷域(用于存放分组)和一个帧尾，如图 3.1 所示，帧管理构成了数据链路层工作的核心，在本章将详细讨论这些功能。

虽然本章讨论的是数据链路层和数据链路协议，但是，我们在本章中将要学习的许多原理，如错误控制和流控制，同样适用于传输层和其他的协议。实际上，在许多网络中，这些功能只出现在数据链路层以上的各层中，而没有出现在数据链路层。但它们的原理都是一致的，而且在数据链路层中，它们通常表现出最为简单和单纯的形式，因此在本章细致地学习这些原理。

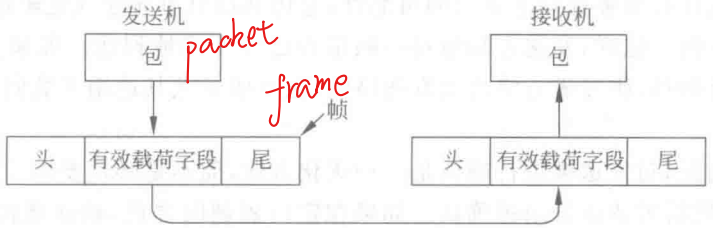

图 3.1 分组和帧之间的关系

1. 为网络层提供的服务

数据链路层为网络层提供的最主要的服务，是将数据从源机器的网络层传输到目标机器的网络层。源机器通过网络层中的一个进程，将一些数据位交给数据链路层，要求传输到目标机器。数据链路层的任务是将这些数据位传输给目标机器，然后再将这些数据进一步交给目标机器的网络层，如图 3.2(a)所示。实际的传输过程沿着图 3.2(b)所示的路径，但很容易让我们产生两个数据链路层在用一个协议通信的联想，因此本章中主要研究图 3.2(a)所示的模型。

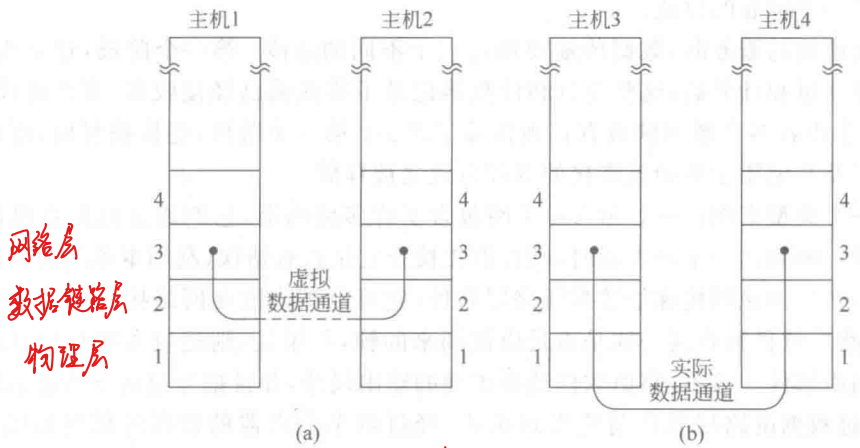

图 3.2 数据链路层的通信过程

(a) 虚拟通信过程；(b) 实际通信过程

数据链路层的设计目标是提供各种服务，实际提供的服务随系统的不同而改变，但是一般情况下，会提供以下三种可能的服务：

(1) 无确认的无连接服务；

(2) 有确认的无连接服务；

(3) 有确认的面向连接服务。

(1) 无确认的无连接服务是指源机器事先不建立逻辑连接，向目标机器发送独立帧，目标机器不对这些帧进行确认，也不用释放逻辑连接，因此数据链路层无法检测和修复噪声等原因引起的帧丢失现象。第一种服务适合错误率很低的情况，可以将恢复的任务交给上面的层来完成。同时因为数据传输过程中，数据迟到比数据损坏问题更严重，因此这种服务也适用于实时通信，绝大多数 LAN 在数据链路层上都是使用无确认的无连接服务。

(2)有确认的无连接服务提高了通信的可靠性，它仍然没有事先建立逻辑连接，但会单独确认所发送的每一帧。这样，发送方知道每一帧是否已经正确地到达。如果有一帧在指定的时间间隔内没有到达，则发送方将再次发送该帧，这类服务尤其适用于类似无线系统的不可靠信道。

必须强调的是，对发送帧进行确认是一种优化方式，而不是必须要求。网络层总是可以发送一个分组，然后等待该分组被确认。如果在定时器超时之前，确认还没有到来，则发送端仅需再次发送整个报文即可。这种策略的麻烦之处在于，硬件条件限制了帧的长度，但网络层的分组没有这样的限制。如果一个普通的分组被分装到 10 帧中，则 20% 的帧将会丢失，为了发送这个分组，可能需要花很长的时间。如果每个帧单独确认和重传，则整个分组很快就会发送过去。在光纤这样的可靠信道上，数据链路协议的额外开销不是必要的，但是在无线信道上，由于它们内在的不可靠性，这种开销是非常值得的。

数据链路层能够向网络层提供的最复杂的服务是面向连接的服务。源机器和目标机器在传输数据之前会先建立一个连接，该连接上发送的每一帧都被编号，数据链路层保证每一帧都按正确的顺序被接收到一次。在无连接服务中的情况则与此相反，你可以想象得到，如果确认报文丢失了，则一个分组可能会发送多次，也会接收多次，因此，面向连接的服务为网络层进程提供了一个可靠的位流。

当使用面向连接的服务时，数据传输要经过三个不同的阶段：第一个阶段，建立连接，双方初始化各种变量和计数器，这些变量和计数器记录了哪些帧已经接收到，哪些还没有；第二个阶段，一个或者多个数据帧被真正地传输了出去；第三个阶段，连接被释放，所有的变量、缓冲区，以及其他用于维护该连接的资源也随之被释放。

这里列举一个典型案例：一个 WAN 子网包含了许多路由器，它们通过点到点电话线连接起来。当某一帧到达一个路由器时，硬件首先检查它是否有错误(利用本章后面我们将要学习的技术)，然后将该帧传递给数据链路层软件(它可能被内嵌在网络接口板的一个芯片中)。数据链路层软件检查这一帧是否是应该到来的帧，如果是，则把包含在有效载荷域中的分组交给路由软件。接着，路由软件选择正确的输出线路，并且把分组向下传递给数据链路层软件，通过数据链路层软件将它发送出去，经过两个路由器的数据流情况如图 3.3 所示。

路由代码希望所有的工作都能正确地完成，即在每一条点到点线路上建立起可靠的、有序的连接，不要总是出现分组丢失的情况。如图中的虚线框所示，数据链路协议使得不可靠的通信线路看起来至少比原来更好。另一方面，尽管在每一个路由器中显示了多份数据链路层软件的副本，但实际上，只有一份数据链路层软件，它负责处理所有的线路，每条线路有不同的表和数据结构。

2. 成帧

为了向网络层提供服务，数据链路层必须使用物理层提供给它的服务。物理层的任务是接受一个原始的位流，并试图将它递交给目标机器，但不能保证位流的正确性。接收到的位的数量可能少于、等于或者多于发送的位的数量，值也可能不同，数据链路层会完成检测和纠正错误的工作。

对于数据链路层，一般的做法是将位流分解成离散的帧，并计算每一帧的校验和(本章后面将讨论校验和算法)。当一帧到达目标机器时，重新计算校验和。如果接收帧的校验和

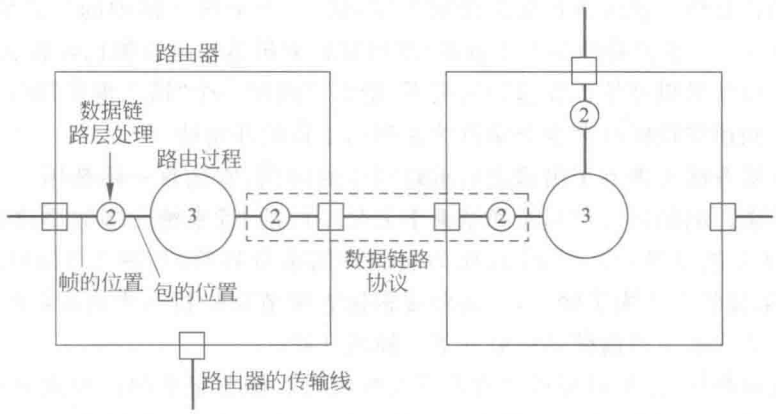

图 3.3 数据链路协议的位置

发生变化，数据链路层会知道传输过程中出现错误，它就会采取措施来处理错误（比如丢掉坏帧，可能还会送回一个错误报告）。

将原始的位流分解到离散的帧中，看似容易做起来却有些困难，有一种成帧的办法是在帧之间插入时间间隙（time gap），就好像在普通正文的英文单词之间插入空格一样。然而，网络一般不会对时间的正确性做任何保证，所以传输时有些间隙会被挤掉或插入其他间隙。

依靠时间来标识帧的起始和结束位置不太可靠，因此有必要设计其他的成帧方法。本节将讨论 4 种方法：

（1）字符计数法；

（2）含字节填充的分界符法；

（3）含位填充的分界标志法；

（4）物理层编码违例法。

1)字符计数法

第一种成帧方法利用头部中的一个域来指示帧中的字符数，当目标端的数据链路层看到这个字符计数值时，它知道后面跟着多少字符，因此也就知道了该帧的结束处在哪里。这项技术如图 3.4(a)所示，其中四帧的大小分别为 5、5、8 和 8 个字符。

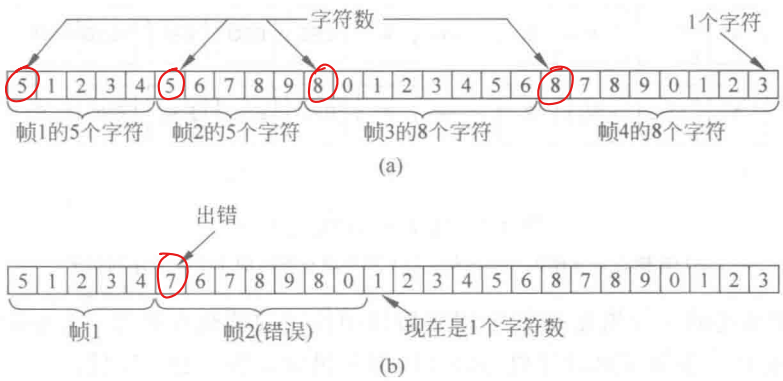

图 3.4 一个字符流的字符计数法成帧

(a) 无差错；(b) 有一个差错

该算法的计数值可能因为传输问题而出错，例如，如果第 2 帧中的计数值 5 变成了 7，如图 3.4(b)所示，由于校验和是不正确的，所以目标方虽然知道该帧已经被损坏，但它仍然无法知道下一帧从哪里开始。在这种情况下，给源方送回一个“请求重传”帧也无济于事，因为目标方并不知道应该跳过多少个字符才能到达重传的开始处。

第二种成帧方法考虑到了出错之后重新同步的问题，它让每一帧都用一些特殊的字节作为开始和结束。旧的协议中起始和结束字节是不同的，近来绝大多数协议倾向于使用相同的字节，称为标志字节(flag byte)，作为起始和结束分界符，如图 3.5(a)中的 FLAG 所示。这样，如果接收方丢失了同步，只需要搜索标志字节就能找到当前帧的结束位置，两个连续的标志字节代表了当前帧的结束和下一帧的开始。

传输二进制数据(比如目标程序或者浮点数值时)，标志字节的位模式出现在数据中时有时会出现很严重的问题，这种位模式会干扰帧的分界。解决这个问题的一种方法是，发送方的数据链路层在每个标志字节的前面插入一个特殊的转义字节(ESC)。接收端的数据链路层在将数据送给网络层之前删除掉转义字节。这种技术称为字节填充(byte stuffing)或者字符填充(character staffing)。这样只要看标志字节前面有没有转义字节，就可以区分成帧与数据中出现的标志字节。

如果转义字节也出现在数据中间，那么该怎么办呢？同样用一个转义字节来填充。因此，任何单个转义字节一定是转义序列的一部分，而两个转义字节则代表了数据中自然出现的一个转义字节。图 3.5(b)显示了一些例子，在这些例子中，去掉填充之后被递交给网络层的字节序列与原始的字节序列完全一致。

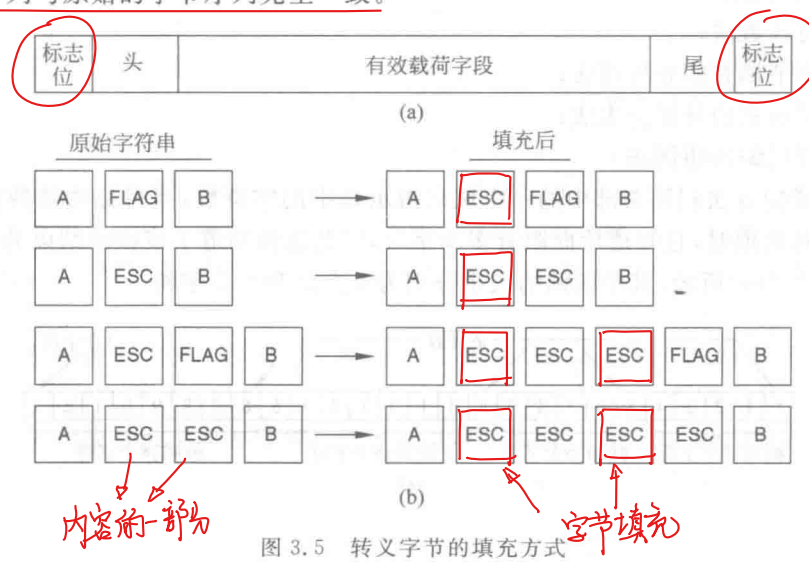

图 3.5 转义字节的填充方式

(a) 有标志字节作为分界的帧；(b) 字节填充前后的 4 个字节序列例子

图 3.5 中描述的字节填充方案是 PPP 协议中使用的填充方案的一个略微简化的形式，而 PPP 协议则是大多数家庭计算机与因特网服务供应商进行通信所使用的协议。

这种成帧方法的一个主要缺点是，它严重依赖 8 位字符模式。但有些字符并不是 8 位。例如，UNICODE 使用的是 16 位字符。随着网络的发展，在成帧机制中内含字符码长度的缺点越来越明显，因此需要新的技术以便允许任意长度的字符。

(3)零标志法

新的技术应当允许数据帧和其中每个字符都包含任意长度的位，它的工作方式如下所述：每一帧的开始和结束都有一个特殊的位模式 01111101，实际上就是一个标志字节。当发送方的数据链路层碰到数据中 5 个连续的位“1”时，它自动在输出位流中填充一个位“0”。这种位填充(bit stuffing)机制与字节填充机制非常相似，在字节填充机制中，当发送方看到数据中的标志字节时，它就在其前面填充一个转义字节，然后再送到输出字符流中。

当接收方看到 5 个连续的输入位“1”，并且后面是位“0”时，它自功去掉(即删除)该“0”位。就好像字节填充过程对于两方计算机中的网络层完全透明一样，位填充过程也对网络层完全透明。如果用户数据包含了标志模式 01111110，则该标志当作 011111010 来传输，但是存储在接收方内存中的是 01111110。图 3.6 给出了位填充的一个例子。

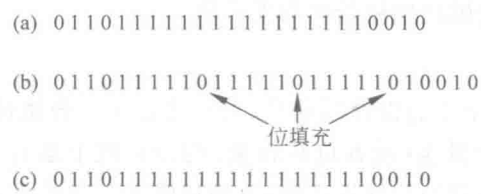

图 3.6 位填充 bit stuffing

标志序列只可能出现在帧的边界上，永远不可能出现在数据中。在位填充机制中，通过标志模式可以明确地识别出两帧之间的边界。因此，如果接收方失去了帧同步，它只需在输入流中扫描标志序列即可，

最后一种成帧方法只适用于那些“物理介质上的编码方法中包含冗余信息”的网络，例如，有些 LAN 用 2 个物理位来编码 1 位数据。通常，“1”位是“高-低”电平对，而“0”位是“低-高”电平对。这种方案意味着每一个数据位都有一个中间电平跃变，这使得接收方很容易定位到位的边界上。“高-高”和“低-低”这两种组合并不用于数据，但是在某些协议中用于帧的分界。

关于成帧机制，还需说明一点，许多数据链路协议联合使用字符计数法和其他某一种方法，以达到更高的安全性要求。当一帧到达时，首先利用计数域定位到该帧的结束处。只有当这个位置上确实出现了正确的分界符，并且帧的校验和也正确时，该帧才被认为是有效的。否则，接收方在输入流中扫描下一个分界符。

3. 错误控制

对于可靠的、面向连接的服务，发送方只是不断地往外发送数据帧，而没有考虑到它们是否正确到达，对于无确认的无连接服务，仅仅这样是不够的。我们还需要考虑如何确保所有的帧最终都被递交给目标机器上的网络层，并且保持正确的顺序。

确保可靠递交的常用方法是向发送方提供一些有关线路另一端状况的反馈信息，通常情况下，协议要求接收方送回一些特殊的控制帧，在这些控制帧中，对于它所接收到的帧进行肯定的或者否定的确认。如果发送方收到了关于某一帧的肯定确认，那么它就知道这一帧已经安全地到达了；另一方面，否定的确认意味着传输过程中产生了错误，所以这一帧必须重传。

有时候由于硬件的问题，有的帧完全丢失了(比如噪声突发时)。在这种情况下，接收方根本不会有任何反应，因为它没有理由做出反应。由此可见，如果在一个协议中，发送方送

出了一帧之后就等待肯定的或者否定的确认，那么，若由于硬件故障的原因而丢失了某一帧的话，则发送方就将永远等待下去了。

可以通过在数据链路层中引入定时器来解决这个问题，当发送方送出一帧时，通常还要启动一个定时器。该定时器的过期时间应该设置得足够长，以保证该帧在正常情况下能够到达目标方，并且在目标方进行处理，然后再将确认送回到发送方。一般情况下，在定时器到期之前，该帧将正确地接收到，并且确认报文也会送回来，这时定时器被取消。

然而，如果原始的帧或者确认报文丢失了，则定时器将被触发，从而警告发送方有一个潜在的问题存在。可以重新发送该帧，但多次重复发送可能会导致接收方将两次或者多次接收同一帧，并且多次将它传递给网络层。为了避免发生这样的情形，有必要为送出去的帧分配序列号，这样接收方能够区别原始帧和重传帧。

4. 流控制 flow control

数据链路层中另一个重要的设计问题是，当发送方在一台快速（或者负载较轻）的计算机上运行，而接收方在一台慢速（或者负载较重）的计算机上运行时，接收速率小于发送速率，发送方以很高的速度持续地往外发送帧，直到接收方完全被淹没。即使传输过程不会出错，但到了某一个点上的时候，接收方也将无法再处理持续到来的帧，从这时开始就要丢弃一些帧了。

常用的办法有两种：第一种是基于反馈的流控制（feedback-based flow control），接收方给发送方送回信息，允许它发送更多的数据，或者至少要告诉发送方它的情况；第二种方法是基于速率的流控制（rate-basd flow control），使用这种方法的协议有一种内置的机制，它限制了发送方传输数据的速率，而无需利用接收方的反馈信息。在本章中，将学习基于反馈的流控制方案，数据链路层从来不使用基于速率的流控制方案。

基于反馈的流控制方案有许多种，但是绝大多数使用了同样的基本原理。通常在没有得到接收方许可（隐式或者显式许可）之前，禁止发送方往外发帧。例如，建立连接时，接收方提示发送若干帧之后就禁止发送，等待许可之后再继续发送。

3.2 检错与纠错

不可靠信道：纠错
可靠信道：检错

电话系统有三个部分：交换机、局间干线和本地回路。目前前两部分几乎已经完全数字化，只有本地回路仍然是模拟的双绞线，由于替换掉这些双绞线需要花费大量的资金，所以在接下去的几年中本地回路还是双绞线。虽然在数字部分很少发生传输错误，但是，在本地回路上错误还很常见。而且，无线通信正在普及，它的错误率比光纤干线要高出几个数量级。在接下去的很多年中，传输错误将一直伴随着我们，我们必须要知道该如何处理传输错误。

由于物理过程而产生的错误，在有些介质（比如无线电波）上常常是突发性的连续多位，而不是单个的。突发性的错误与孤立的、单个位的错误相比，既有优点也有缺点。假设数据块的大小为 1000 位，每一位的错误率是 0.001。如果错误是独立的，则大多数数据块将包含一个错误。然而，如果错误是突发性的，发生一次就是连续 100 位，则平均而言，在 100 个数据块中只有 1 个或者 2 个数据块受到影响，突发性错误比单独的错误更加难以纠正。

突发差错 ← 冲击噪声
随机差错 ← 热噪声