Chapter 3 数据链路层

2020年5月9日 17:33

中山大學 SUN YAT-SEN UNIVERSITY

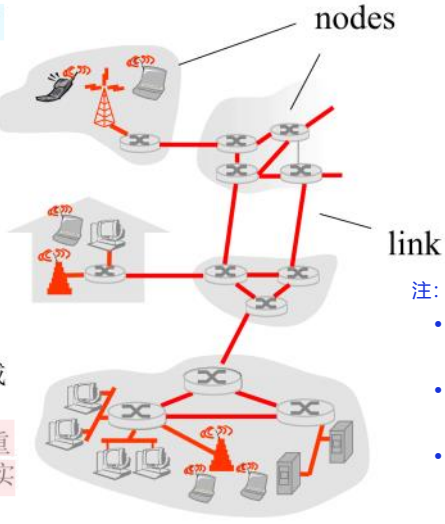
第三单元 数据链路层

- 概述
- 差错检测
- 可靠数据传输
- 停等协议
- 滑动窗口协议
- PPP协议

2019.8.10

概述

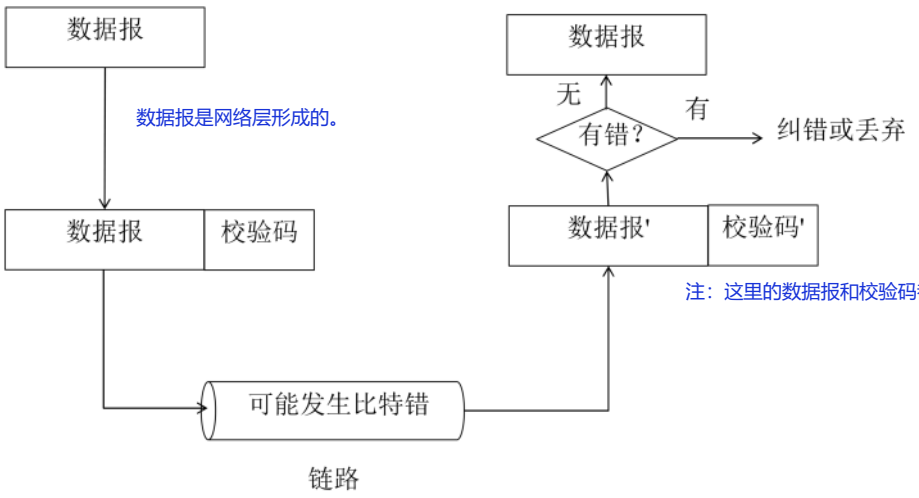
- **数据链路层**负责在直连网络或物理网络的节点之间传送数据包，可以把数据包从一个节点通过链路传送给相邻的另一个节点。
- 数据链路层（第2层）的数据包(packet)是数据帧(frame)
- 数据链路层的四个主要功能：
  - 形成帧 (framing)
  - 差错检测(error detect)：发现比特错或纠错
  - 差错控制(error control)：发现丢包、重复、错序、溢出（overflow）等错误，实现流控制(flow control)。
  - 介质访问控制(medium access control)：处理多路访问中的碰撞(collision)问题。



注：

- 物理层传送的是原始比特流，而数据链路层将它划分成一块块——形成帧。
- 重复：是算法导致的（直连网里面只有线路，不考虑路由的影响）
- 溢出：发送方（S方）发的太快，接收方（R方）来不及接收，发生丢包。

差错检测



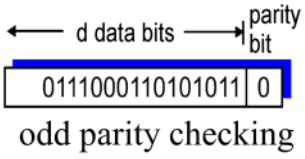
注：这里的数据报和校验码都有可能出错！

奇偶校验

一维奇偶检验

Single Bit Parity:

Detect single bit errors



odd parity checking

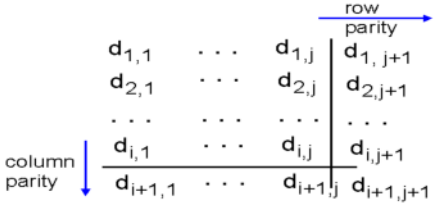
哪位有错？

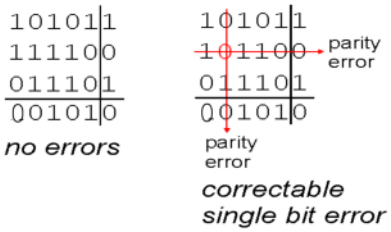
1 0 1 1 1
1 0 0 0 1
0 1 1 1 1
1 0 1 0 1
1 1 0 0 0

二维奇偶检验

Two Dimensional Bit Parity:

Detect and correct single bit errors





二维奇偶检验的方法：

- 将数据分成n段，每段占1行。
- 每一列、每一行分别进行奇、偶检验
- 在接收方检测每一行、每一列符不符合奇偶检验的规定。如果不符合，则相交的那一位就是出错的那位。
- 注：这只能确定出一位错！！
- 可以检出所有的错误，但不能纠正所有的错误。
- 只能确定奇数位的出错，无法确定偶数位的出错。

检验和

(Checksum)

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

反码：1011 010100110011

10000110 10000111 00000100 01000100 11000000 00000000 10110101 00110011

数据　　校验和

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。

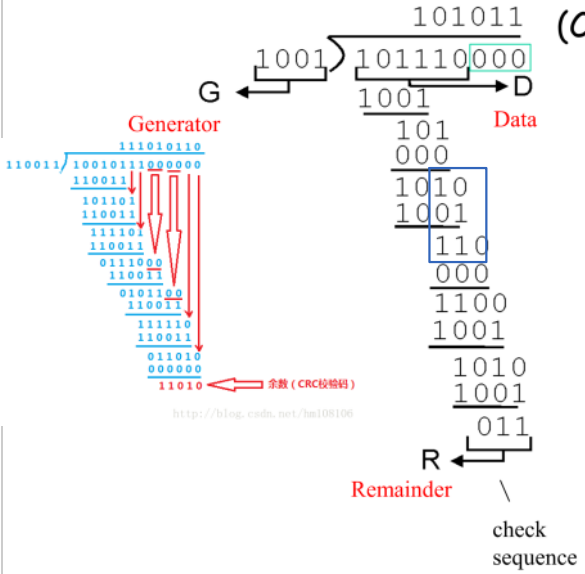
检验和的使用方法：

- 如果有进位，则对这个数+1.
- 也可以把加法都算完之后再将前面的进位整体加到后面，但是这样reg的位数就需要更多的位。

循环冗余校验码

https://blog.csdn.net/qq_25315595/article/details/82461500 CRC原理

(Cyclic Redundancy Check, CRC)



最后结果：101110 011

数据　校验码

如果传输过程中没有出现比特错，接收方用相同的除数去除数据加CRC校验码，余数应该为0。

采用模2除法：做减法时没有借位，类似于按位异或。

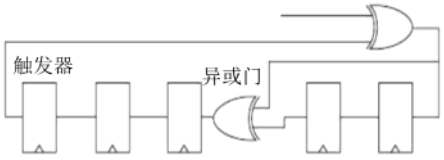
CRC原理：

- 先在数据的后端加上n个0（假设除数有n+1位）
- 除数是S\R双方事先商量好的
- 此处的除法是模二除法！所以不是借位，而是异或！！
- 且不一定是被除数比除数大才可以！！
  - 只要位数相同就可以按位异或了！！

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%。

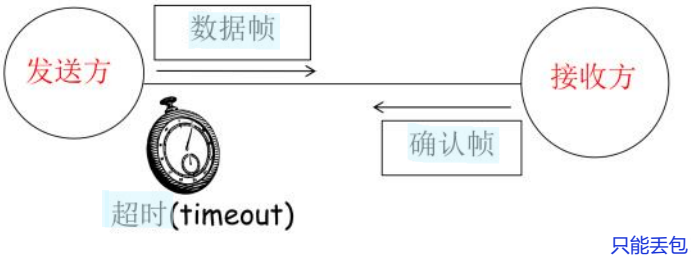
检错率极高！！



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

可靠数据传输

(Reliable Data Transfer)



只能丢包!

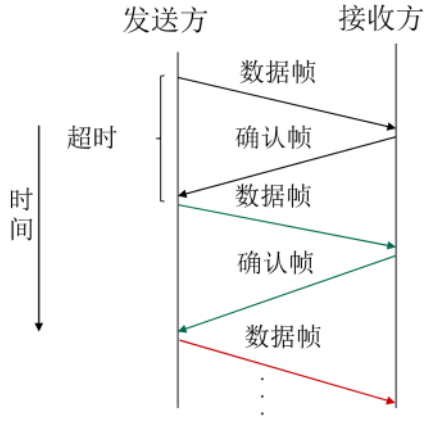
如果收到的帧出现错误又不能纠正，该帧将被丢弃，即出现丢包错误(loss)。自动重发请求(Automatic Repeat reQuest，ARQ)通过确认的方式重传丢失的包，它的方法是每发送一帧都启动一个超时定时器，如果它的确认帧在其超时时间内到达就删除该定时器，否则，将重传该帧并同时重启其定时器。

确认帧(Acknowledgement Frame)是一个控制帧，接收方把它发给发送方用来表示已经收到了它的数据帧。

后面讲的停等协议和滑动窗口协议都是ARQ协议。 注：这里讲的协议只是一个算法，不是真正意义上的协议。

停等协议（stop-and-wait）

SWS=1。



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

只有收到前一个数据帧的确认才可以发送下一个数据帧。

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？

以下分析3种错误：

- 数据帧丢失：（接收方不知道）->超时->重传->确认
- 确认帧丢失：->超时->重传->确认
  - 此时R方接收到2个相同的包，则R方会识别并且丢弃
- 超时：确认帧在到达S方时已超时->重传->R方再发确认帧
  - 由于S方不知道重传原因，所以只要R方收到了数据帧，则一定要发回确认帧！！否则S方会一直不停重传！！
- 可以在帧的前面和后面加“帧头+帧尾”，并在帧头中加一个序号，对应的数据帧和确认帧的序号相同。
  - 只有link层有帧头帧尾，其它层只有帧头！

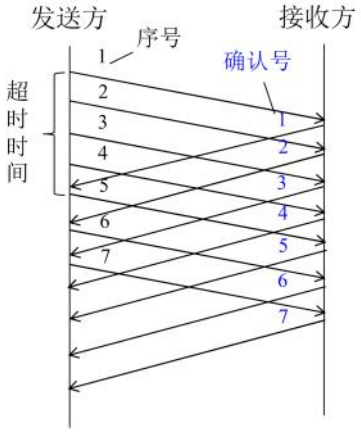
这里还有几个问题：第二次计时的时候收到第一次的确认帧要怎么办？序号个数要多少个（个数=2^序号位数）才够？传播延迟太大怎么办？（不等确认帧可以吗？）

- 在滑动窗口协议中，很好的解决了这些问题：发送窗口可不等确认帧连续发；传播延迟也相对应减少。

滑动窗口协议

(Sliding Window)

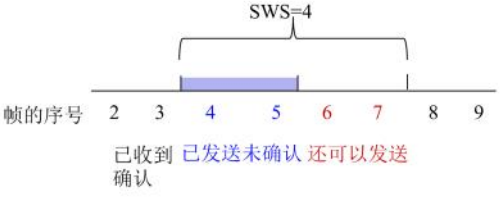
滑动窗口协议是一种ARQ协议，它不需要等待已发送的数据帧的确认帧回来就可以连续发送多个帧，其个数由发送窗口来控制。



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)。

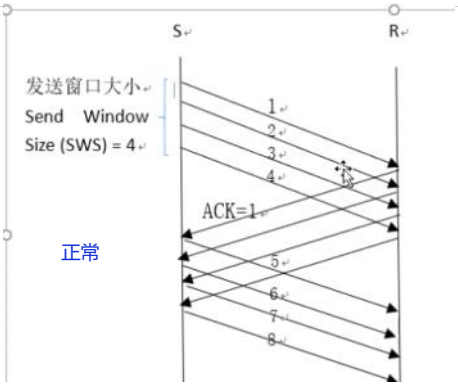
发送窗口大小(Sending Window Size, SWS)表示发送窗口的大小，也是发送缓冲区的大小。

如果接收方发送的确认帧(Acknowledgement Frame)的确认号为ACK，表示序号为ACK以及之前的数据帧已经全部收到并已交给上层协议。

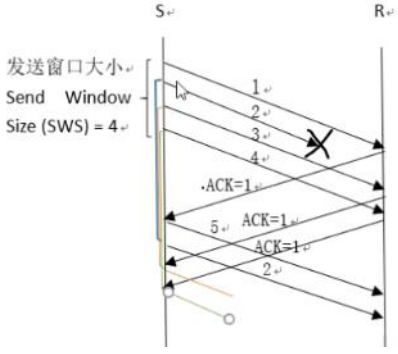


注：

- 此处假设未收到确认时可连续发送4个数据帧（也就是发送窗口/缓冲窗口的大小是4）
- 每个小窗口的大小是按照最长帧来开的，且只放一个帧。
- 缓冲区是为了重传做准备！
- 假设现在data=2丢失了，那么确认帧ACK=2不会发送，那么收到data=3,data=4的时候，R方发回去的确认号ACK=1，则S方收到ACK=1对应的确认帧时会发送data=5，但是后面就不会。等待第二帧超时之后就重传第二帧。
- 如果超时的时间设置的过长，则出错要等很久才会发现。效率太低！

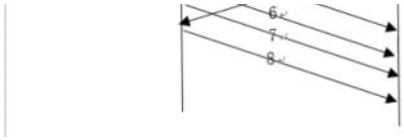


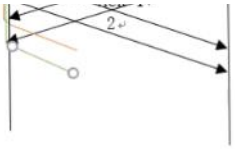
正常



注：在异常的情况：

- 只有超时之后才会重传（收到ACK=1并不会实现重传）
- 这样后面3、4都会超时，5也会超时（因为2的确认帧没回来之前，所有的帧都不会确认）。所以也就是2后面所有已经发送过的帧全部重传。
- 超时通常会设置为略大于一个往返时间。过长的话纠错时间会过长，效率低





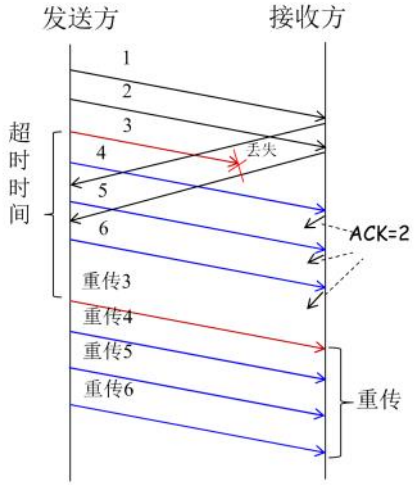
- 超时通常会设置为略大于一个往返时间。过长的话纠错时间会过长，效率低。
- 这个情况下，GO-BACK N很少出错。

回退N协议

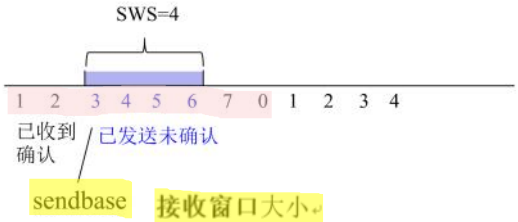
(go back N)

极少出错!

回退N协议是一种滑动窗口协议，出现超时的时候将重传所有已发送且未收到确认的帧。



序号可以循环使用：



接收窗口大小

Receive Window Size (RWS) = 1

1-1

2-2

注：

- Go back N的缓冲区大小为1，所以在2数据帧丢失的情况下，那个3、4、5传过来之后会丢弃，S方需要重传。
- 由于RWS和接收缓冲区的大小为1，所以加入第二帧丢失的时候，接收窗口为2-2，那么3、4、5到的时候就不会收下来，因为不在范围内。
- 数据链路层：为上层服务，为了可靠，要保持次序，只有前面的帧传给了上层，后面的帧才会被传上去。
- 接收窗口和接收缓冲区是不一样的，通常认为接收窗口接受了然后转给缓冲区。

Q：现在在回退N协议中，假设S方的待发送序列为3456701234，并且除5所有的帧都收到了（包括重传帧），那么R方收到的帧的序号顺序是？（包括收下但丢掉的帧）

A：34 670 56701234（由于5丢失，那么34在收到确认帧之后会发送70，只要记住在外面的帧的个数一定等于SWS，所以有5670在外面，而5丢失，所以会导致5670重传，也就是34 670（收到但会被丢弃）56701234，即3467056701234.

Q：3丢失，接收方在发送ACK=2的时候，如果顺便告诉S方已经收到ACK=4,5,6，可行吗?

A：可行！选择性确认。

Q：如果SWS=1,带宽是2Mbps。那么SWS=4的时候?

A：8Mbps。一个往返时间，可以发4帧。SWS实现流量控制。

想要改进？ - 改RWS=4

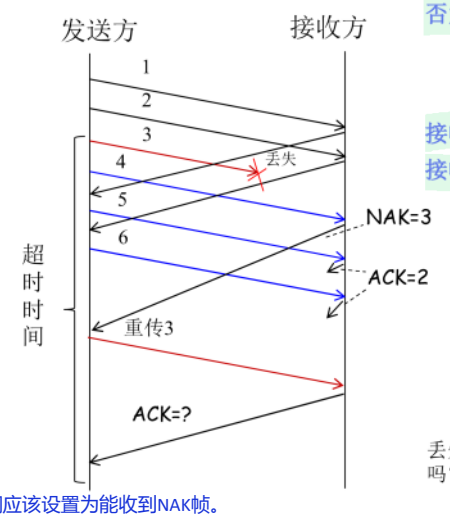
- 将超时时间变大！

选择性重传

(Selective Repeat)

靠NAK来决定重传，而不是靠超时。

选择性重传协议通过发送NAK帧要求发送方单独重传丢失的帧。

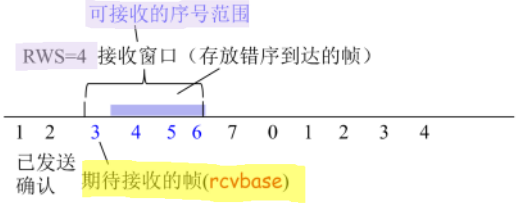


时间应该设置为能收到NAK帧。

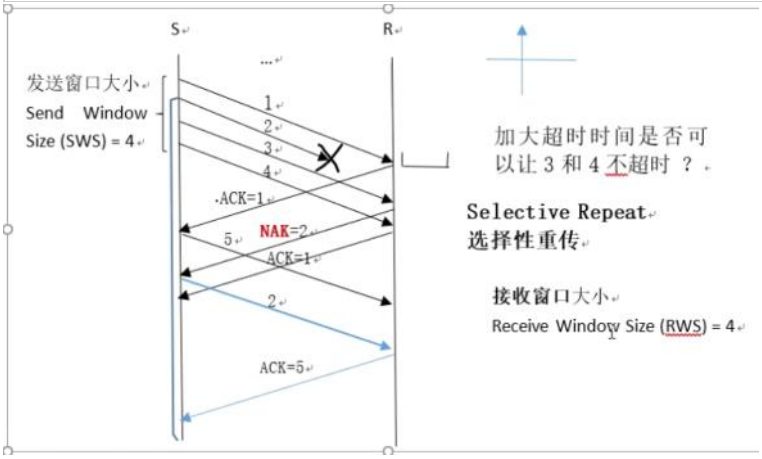
否定性确认帧(Negative Acknowledgement,NAK)用于表示这一帧之前的数据帧全部收到并已交给上层协议，要求发送方重传这一帧。每个帧只发送一次NAK。

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？ 如何设置超时时间？



注：

- 比如说这里的3号数据帧丢失，那么此时由于4到了3不到，则会认为3号丢失，发送NAK=3。（后面的仍然发送ACK=2)
- 当重传3号数据帧到达之后，则会将3、4、5、6都发送给上层。然后发送ACK=6。（也可以传一个发送一个确认帧，则ACK=3\4\5\6都会发送）
- 如果NAK丢失，不会出错，因为还有超时重传机制。
- NAK=2和ACK=1并不是等价的。因为
  - 前者表示”2丢失了但是前面的都收到并交给上层了“（此时这个NAK的数值是期待收到的帧），
  - 后者表示”1前面的都收到“。
- 当SWS=1的时候，就变成了停等协议。
  - 除了超时时间比较长之外，就是停等协议。
- 当RWS=1的时候，就变成了go back N.
  - 在窗口之外的帧不发NAK，那么会通过超时重传。
- 如果丢失3、4，则应该是发NAK=3和NAK=4。

Q：为什么在RWS=SWS=4时，序号一定要是8个？（0-7）

A：因为假设是7个（0-6）时，那么在前4个帧1234被R方接受并传给上层但是他们的确认帧丢失的情况下，R方的接收窗口的序号可接受范围已经变成了5601（ACK=4)。而由于S方没有收到确认帧，（此时也没有NAK）超时后会重发1234，而此时1号数据帧来到R方，会由于在接收窗口的可接受范围内而被重新接受，这样就发生了错误（本来不应该接受），234会因不在窗口范围内被丢弃。但是如果是8个帧，则R方接受范围就会变成5670，则不会发生错误。

所以，最少序号范围>=SWS+RWS.

- SWS和RWS大小关系：
  - SWS >= RWS

这应该是选择性确认。（不要看了！）

Selective repeat

sender

data from above: 发

- if next available seq # in window, send pkt

timeout(n): 超时

- resend pkt n, restart timer

收到ACK ACK(n) in [sendbase,sendbase+N-1]:

- mark pkt n as received
- if n smallest unACKed pkt, advance window base to next unACKed seq #

这里的ACK表示的是对应的包收到了，而不是之前所以的都收到了。

receiver

pkt n in [rcvbase, rcvbase+N-1]

buffer

- if n=rcvbase, (1) deliver buffered, in-order pkts to upper layer
  (2) advance window to the pkt x expected to receive,
  (3) rcvbase set to x
  (5) send ACK(rcvbase -1)
- if n!=rcvbase,
  send NAK(rcvbase-1)
  if not sent within some time
  or send ACK(rcvbase -1) ✓

pkt n not in [rcvbase,rcvbase-1]

- discard the pkt
- send ACK(rcvbase-1)

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。当发送方要重传帧的时候，不会发送这些帧。

如：data=3丢失，则当4、5、6帧到达的时候，R方发送ACK=2的时候，顺便把已收到4、5、6的信息告诉S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。此时超时时间也对应延长。

节省带宽！！如到达3、4、5、6，则最后只会发送确认帧ACK=6。

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。

因为其实是双方都在发送信息的。

延迟确认式捎带确认的效率很高！

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协议都采用了PPP协议进行封装。

以太网

PPP数据帧

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 0x7E	0xFF	0x03			CRC-16/CRC-32	01111110

- PPP协议采用HDLC协议的广播地址(0xFF)和无编号帧(0x03)。固定的
- 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$x^{16} + x^{12} + x^5 + 1$。 通过多项式实现，如17位为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。

注：

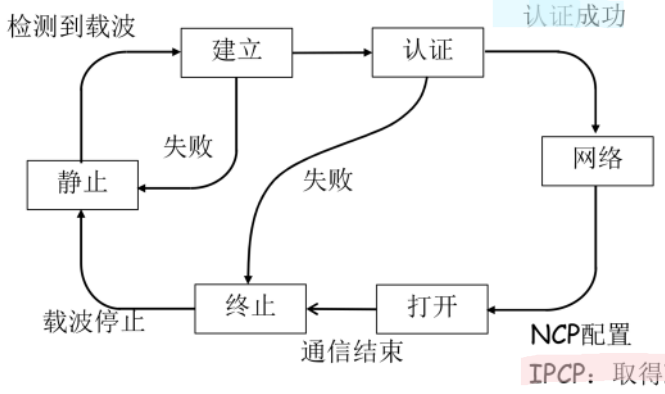
- 这里的“协议”和“校验码”都是要提前选好的。
- 字节填充法：有没有什么问题
  - A:有的！因为有可能数据段也会有0X7E。所以此处将两个表示分别替换为0x7D-5E和0x7D-5D。而数据段中，只有小于0x20的字节才加上值0x20和前面的0x7D.
  - 在以前，电话是以语音的形式传输的。而前0x20个字节是用来控制的，比较特殊。
- 为啥这里可以省略地址？
  - A：因为这里是点到点的网络，不需要地址。
- PPP只检错，不纠错。
- PPP不可靠，不可保证有序，错了就扔。
  - 有序是通过上层协议保证的：如TCP

PPP协议的状态图

提前协商的!

是否需要压缩协议？是否进行身份鉴别？协议是什么？

LCP配置　双方协商选项



Link Control Protocol(LCP)

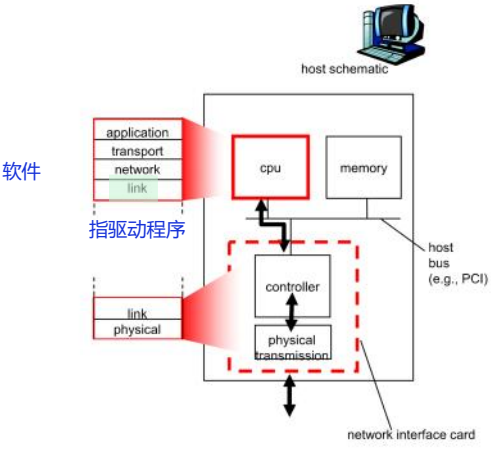
Network Control Protocol (NCP): IPCP(Internet)，IPXCP(Novell)

注：

- PPP协议用IPCP协议获取IP地址
- PPP协议使用LCP协议确定是否需要身份认证和确定链路层的参数。
- IPCP用于IP协议，IPXCP用于IPX协议。IP地址是IP协议的

链路层的实现

- 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。



软件

指驱动程序

注：驱动程序从网卡中通过中断&DMA取数据给OS。

总结

- 概述：数据链路层的功能是什么？
- 差错检测：奇偶校验、校验和、CRC校验码
- 可靠数据传输
- 停等协议
- 滑动窗口协议
- PPP协议