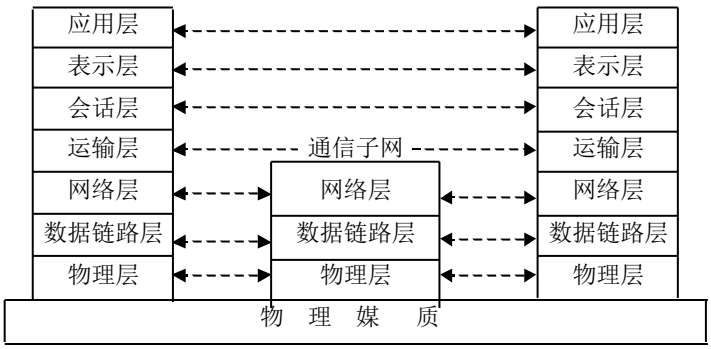
第一章：PPP 协议简介

PPP 协议是数据链路层协议，因此我们应该对数据链路层有简单的了解。数据链路层在 OSI 七层模型中位于最底层物理层之上，网络层之下（如图 1.1）。它一方面从物理层的 SAP（服务访问点）得到物理层的服务，主要是信号的编码和译码、为进行同步用的前同步码的产生和去除、比特的传输和接收等；另一方面也通过本层的 SAP 向网络层提供服务，主要是数据链路的建立和释放帧的封装与拆卸、差错控制等。



OSI 参考模型

图 1.1

数据路层主要讨论在数据链路上帧流的传输问题。这一层协议的内容包括：帧的格式，帧的类型，比特填充技术，数据链路的建立和终止信息流量控制，差错控制，向物理层报告一个不可恢复的错误等。这一层协议的目的是保障在相邻的站与节点或节点与节点之间正确地、有次序、有节奏地传输数据帧。常见的数据链路协议有两类：一是面向字符的传输控制规程，如基本型传输控制规程（BSC）；另一类是面向比特的传输控制规程，如高级数据链路控制规程（HDLC）。主要是后一类。

用户接入 Internet，在传送数据时都需要有数据链路层协议，其中最为广泛的是串行线路网际协议（SLIP）和点对点协议（PPP）。SLIP（Serial Line Internet Protocol）意为串行线路 Internet 协议。它是通过直接连接和用调制解调器连接的 TCP / IP。由于 SLIP 具有仅支持 IP 等缺点，主要用于低速（不超过 19.2kbit/s）的交互性业务，它并未成为 Internet 的标准协议。为了改进 SLIP，人们制订了点对点 PPP（Point-to-Point Protocol）协议。PPP 协议用于实现与 SLIP 一样的目的和作用，它在实现其作用的方式上比 SLIP 要优越得多。PPP 连接协议包括出错检测和纠正，以及分组验证，这是一个安全性特征，它能确保接收的数据分组确实来自于发送者。这些特性合起来使得通过电话线可以建立更为安全的连接。PPP 是一种被认可的 Internet 标准协议，所以目前得到最广泛的开发支持。

PPP 点到点协议是为在同等单元之间传输数据包这样的简单链路设计的链路层协议。这种链路提供全双工操作，并按照顺序传递数据包。设计目的主要是用来通过拨号或专线方式建立点对点连接发送数据，使其成为各种主机、网桥和路由器之间简单连接的一种共通的解决方案。

其功能有以下几点:

（1）PPP 具有动态分配 IP 地址的能力；

（2）PPP 支持多种网络协议，比如 TCP/IP、NetBEUI、NWLINK 等；
（3）PPP 具有错误检测以及纠错能力，支持数据压缩；
（4）PPP 具有身份验证功能。

PPP 协议中提供了一整套方案来解决链路建立、维护、拆除、上层协议协商、认证等问题。PPP 协议包含这样几个部分：链路控制协议 LCP（Link Control Protocol）；网络控制协议 NCP（Network Control Protocol）；认证协议，最常用的包括口令验证协议 PAP（Password Authentication Protocol）和挑战握手验证协议 CHAP（Challenge-Handshake Authentication Protocol）。LCP 负责创建，维护或终止一次物理连接。NCP 是一族协议，负责解决物理连接上运行什么网络协议，以及解决上层网络协议发生的问题。

第二章：PPP 封装

PPP 协议为串行链路上传输的数据报定义了一种封装方法,它基于高层数据链路控制（HDLC）标准。PPP 数据帧的格式如表格所示:

标志	地址	控制	协议	数据（<1500 字节）	FCS	标志
0x7E	0xff	0x03				0x7E
1Byte	1Byte	1Byte	2 Byte		2 Byte	1 Byte

（1）PPP 的帧格式前 3 个字段固定为：0x7E、0xFF 和 0x03。
（2）PPP 帧的长度都是整数个字节。
（3）若封装在 PPP 帧中的数据出现字节 0x7E，则用 2 字节序列 0x7D、0x5E 取代； 若出现字节 0x7D，则用 2 字节序列 0x7D、0x5D 取代；
（4）若信息字段中出现 ASCII 码的控制字符(即小于 0x20 的字符)，则在该字符前面要加入一个 0x7D 字节，并且要将该字符转为与 0x20 进行与操作后的结果。这样做的目的是防止这些表面上的 ASCII 码控制符(在这里实际上已不是控制符了)被错误地解释为控制符。
（5）协议字段由两个字节组成。字段中第八位必须是 0，最后一位必须为 1。链路若收到不符合这些规则的帧，必须被视为带有不被承认的协议。
在范围"0x0***"到"0x3***"内的协议字段，标志着特殊数据包的网络层协议。
在范围"0x8***" 到"0xb***"内的协议字段，标志着数据包属于网络控制协议（NCP）。
在范围"0x4***"到"0x7***"内的协议字段，用于没有相关 NCP 的低通信量协议。
在范围"0xc***"到"0xf***"内的协议字段，标志着使用链路层控制协议（LCP）的包。
以下的值作为保留：
0xC021 链路控制协议 LCP(Link control protocol)
0xC023 密码认证协议 PAP(Password authentication protocol)
0xC025 链路品质报告 Link Quality Report
0xC223 挑战握手验证协议 CHAP(Challenge handshake authentication protocol)
0x8021 IP 控制协议 IPCP(Internet protocol control protocol)
0x0021 Internet protocol（IP）
0x0001 填料协议 (Padding Protocol)
0x0003~0x001F reserved (transparency inefficient)保留（透明度效率低的）
0x007D reserved (Control Escape)保留（控制逃逸）

0x00CF	reserved (PPP NLPID)保留(PPP NLPID)
0x00FF	reserved (compression inefficient)保留(压缩效率低的)
0x8001~0x801F	未使用
0x807D	未使用
0x80CF	未使用
0x80FF	未使用

(6) FCS 字段为整个帧的循环冗余校验码，用来检测传输中可能出现的数据错误。(计算范围为 PPP 帧扣去帧头尾标志-7E 两个字节的范围)

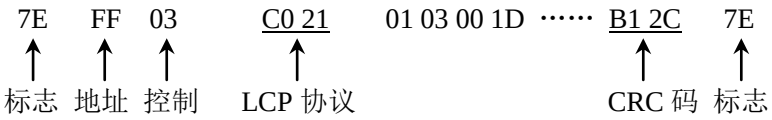
(7) 数据字段是零或更多的字节。数据字段的最大长度，包含填料但不包含协议字段，术语叫做最大接收单元（MRU），默认值是 1500 字节。若经过协商同意，也可以使用其它的值作为 MRU。在传输的时候，信息字段会被填充若干字节以达到 MRU。每个协议负责根据实际信息的大小确定填料的字节数。

(8) 即使使用所有的帧头字段，PPP 协议帧也只需要 8 个字节就可以形成封装。如果在低速链路上或者带宽需要付费的情况下，PPP 协议允许只使用最基本的字段，将帧头的开销压缩到 2 或 4 个字节的长度，这就是所谓的 PPP 帧头压缩。

为了加深认识，我们看一下下面这段截取的 PPP 报文：

7E FF 7D 23 C0 21 7D 21 7D 23 7D 20 7D 3D …… B1 2C 7E

根据以上规则，首先要将报文段中转意的字符转化回来。即将 7D 后的 5D、5E 转为 7D、7E,其余的字符都减去一个 0x20。例如以上报文就转为：



这个数据包就是下一章我们要说的 LCP 链路控制协议包。

第三章：LCP 协议

§3-1：LCP 数据包

LCP 链路控制协议，用于 HDLC 的上层，用于协商适合于数据链路的选项，如指出链路的一边同意接收的最大数据报大小即最大接收单元（MRU）。探测链路回路和其它普通的配置错误，以及终止链路。

一个 LCP 包被封装在 PPP 数据域中，该 PPP 协议域表示为 0xC021（链路控制协议）。LCP 包的格式如下：

代码	标识符	长度	数据
1Byte	1 Byte	2 Byte	

（1）代码

代码域确定 LCP 包的种类,不同的包有不同的格式。

（2）标识符

标识符域在匹配请求和回复中有用。当带有无效标识符域的包被接收时候，该包将不影响 LCP 自动机制，将被静静的丢弃。

（3）长度

长度域指出 LCP 包的长度，包括代码，标识符，长度和数据域。该长度必须不超过链路的 MRU。长度域以外的字节被当作填料而忽略处理。

（4）数据

数据域是零或多个八位字节，由长度域声明。数据域的格式由代码域决定。

LCP 包有 3 类：

1.链路配置包，用于建立和配置链路：Configure-Request（匹配请求），Configure-Ack（匹配正确应答），Configure-Nak（匹配不应答），和 Configure-Reject（匹配拒绝）。

2.链路结束包被用于结束一个链路：Terminate-Request（终止请求） 和 Terminate-Ack（终止应答）。

3.链路维修包被用于管理和调试一个链路：Code-Reject（代码拒绝）, Protocol-Reject（协议拒绝）, Echo-Request（回波请求）, Echo-Reply（回波应答），和 Discard-Request（抛弃请求）。

下面依次介绍不同种类的 LCP 包的格式：

<1> Configure-Request（匹配请求）

一个操作想要打开一个连接必须传送一个 Configure-Request 包，Configure-Request 包的格式如下:

代码	标识符	长度	选项
0x01	1 Byte	2 Byte	

标识符

用来标志不同的 Configure-Request 包。只要选项域的内容改变，并且只要收到先前有效的 Configure-Request 包，标识符域必须被改变，以保证其唯一性，仅仅在重发时，标识符域可以保持不变。

选项

选项域是长度的变量，并包含零个或多个发送方需要协商的配置选项的列表。列出的全部配置选项总是被同时协商。

选项域的格式如下：

类型	长度	数据
1Byte	1Byte	

类型

类型域用于指出配置选项的类型，不同的值代表不同的配置项目。以下值保留：

0x00 RESERVED（保留）

0x01 Maximum-Receive-Unit（最大-接收-单元）

0x02 Async-Control-Character-Map(异步-控制-字符-映射)

0x03 Authentication-Protocol（鉴定-协议）

0x04 Quality-Protocol（质量-协议）

0x05 Magic-Number

0x07 Protocol-Field-Compression（协议-域-压缩）

0x08 Address-and-Control-Field-Compression（地址-和-控制-域-压缩）

长度

长度域指出该配置选项（包括类型、长度和数据域）的长度。

数据

数据域是零个或者更多的八位字节，并且包含配置选项的特定详细信息，不同的配置项目含有不同的信息。当数据域长度超过长度域所指出的长度时，整个配置包将被静静的丢弃。

以下介绍不同的配置选项的格式：

- Maximum-Receive-Unit（最大-接收-单元）

该配置选项通知链路另一端可以接收多大的包。默认值是 1500 个字节。该选项用于指出一个最大的容量，而不一定使用这个最大容量。例如，当一个 2048 个字节的 Maximum-Receive-Unit（MRU）被协商，一端不需要用 2048 个字节发送每个包，而可以发送较小的包。MRU 配置选项格式如下：

类型	长度	最大接收单元
0x01	0x04	2Byte

- Async-Control-Character-Map(异步-控制-字符-映射)

该配置选项用于协商在异步链路中透明传输控制字符的方法。异步链路的每端支持两个异步控制字符映射。接收 ACCM 是 32 位的，而发送 ACCM 可以多达 256 位，也就是说，链路两端有四个不同的 ACCM，每端两个。

对于异步链路，默认的接收 ACCM 是 0xffffffff。默认的发送 ACCM 是 0xffffffff 加上控制逃逸字符和标记序列本身，再加上将要异步发出的而被标记的字符。对于其它类型的连接，默认值是 0，因为对其它类型的连接而言，没有必要进行映射。默认的字符（除了 0x20 外）都能够在所有已知的通信设备上透明传送。

然而，通常不需要映射所有的控制字符，甚至不需要映射任何控制字符。配置选项用来通知双方哪些控制字符在发送时需要映射。一方可能仍然在映射的格式中传送其它字符。此时发送方应该采取措施使接收方在接收时忽略这些字符。

下面所示为异步控制字符映射配置选项格式:

类型 0x02 ACCM	长度 0x06	ACCM 4Bytes

ACCM 域表示控制字符的设置。每个编号的位对应一个相同数值的八位数。ACCM 的最后一位编号为 0，对应 ASCII 字符 NUL.如果某位上置零，表示对应值的控制字符不被映射。如果位上置一，表示对应值的控制字符必须继续被映射。例如，如果第 19 位置零，那么对应 ASCII 控制字符 19 要直接以 0x13 发送；反之，要以 0x7D33 发送。

- Authentication-Protocol（认证-协议）

Authentication-Protocol 配置选项格式如下:

类型 0x03	长度 1 Byte	认证协议 2 Byte	数据

长度域的值大于或等于 4。认证协议域是两个八位字节，指出认证阶段想要使用的认证协议，如：

0xC023　　密码验证协议
0xC223　　挑战握手验证协议

数据域是零或多个八位字节，包含由具体配置协议项目决定的附加数据。

- Quality-Protocol（质量-协议）

该配置选项用于协商链路质量监测协议，Quality-Protocol 配置选项格式如下：

类型 0x04	长度 1Byte	质量协议 2Byte	数据

长度值大于或等于 4。质量协议域指出链路想要使用的质量监测协议，如：

0xC025　　链路质量报告

数据域是零或者多个八位字节，包含由具体协议项目决定的附加数据。

- Magic-Number

魔数设置主要是用来监测网络中是否有自环现象。如果Configure-Request包的发送方反复收到和自己发送包中的魔数相同的Configure-Request包，则认为网络中存在自环。但由于可能通信双方恰好都选择了相同的Magic-Number，造成双方或一方误认为有自环存在，因此要有性能良好的随机数函数来产生Magic-Number。协议中建议通信双方分别选择各自独特的随机数种子，以尽量减少误判。

Magic-Number 配置选项格式如下：

类型	长度	Magic-Number
0x05	0x06	4Byte

- Protocol-Field-Compression（协议域压缩）

Protocol-Field-Compression 配置选项格式如下:

类型	长度
0x07	0x02

该配置选项用于协商 PPP 协议域的压缩。PPP 协议域可以被压缩进一个与原来两字节协议域有明显区别的单字节形态。该配置选项被发送来通知另一端能接收这种压缩的协议域。只有在该配置选项协商后，被压缩的协议域才能被传送。在协商成功后，链路必须具有同时接受双字节和单字节协议域的 PPP 包的能力，即不区别两者。当发送任何 LCP 数据包时不允许压缩协议域，这一规则保证 LCP 包的明确识别。当一个协议域被压缩，数据链路层 FCS 域在被压缩的帧中计算，而不是最初的未压缩的帧。

- Address-and-Control-Field-Compression（地址和控制域压缩）

Address-and-Control-Field-Compression（ACFC）配置选项格式如下：

类型	长度
0x08	0x02

该配置选项用于协商数据链路层地址和控制域的压缩。由于数据链路层的地址和控制域是常量，所以易于压缩。该配置选项被发送来通知另一端能接收压缩的地址和控制域。如果当 ACFC 未被协商时接收到一个压缩了的帧，可以静静的丢弃该帧。当发送任何 LCP 包时，地址和信息域必须不被压缩。这一规则保证了能明确识别 LCP 包。当地址和控制域被压缩时，数据链路层 FCS 域在被压缩的帧中计算，而不是最初的未压缩帧。

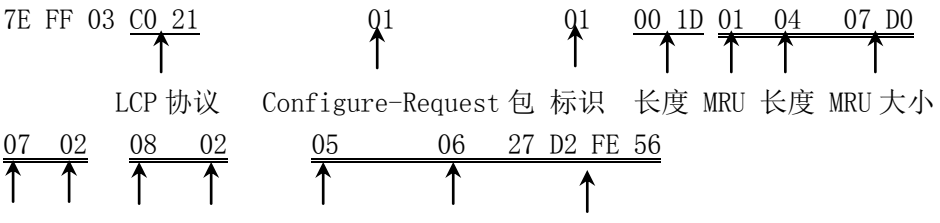
为了加深对 LCP 数据包的认识，我们看下面一段截取的报文：

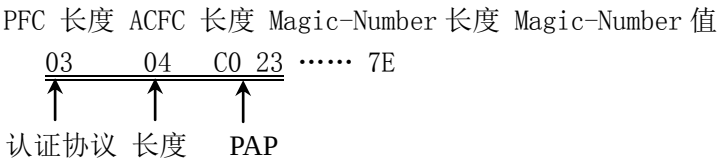
7E FF 7D 23 C0 21 7D 21 7D 21 7D 20 7D 3D 7D 21 7D 24 7D 27 D0 7D 27 7D 22 7D 28 7D 22 7D 25 7D 26 27 D2 FE 56 7D 23 7D 25 C0 23 …… 7E

转化结果如下：

7E FF 03 C0 21 01 01 00 1D 01 04 07 D0 07 02 08 02 05 06 27 D2 FE 56 03 05 C0 23 …… 7E

分析结果如下：





<2> Configure-Ack

Configure-Ack 包格式如下：

代码	标识符	长度	选项
0x02	1Byte	2Byte	

各域的意义与 Configure-Request 包相同。

Configure-Request 包的使用方法如下所述：

如果在收到的 Configure-Request 包中每一个配置选项及其所有的值都是能接受的，那么必须传送一个 Configure-Ack 包。Configure-Ack 包中的配置选项必须是接受的 Configure-Request 包中的配置选项的拷贝，且标识符域必须与 Configure-Request 包相同以保证匹配。

接到的与以上要求不符的包将被静静的丢弃。

<3> Configure-Nak

Configure-Nak 包格式如下：

代码	标识符	长度	选项
0x03	1Byte	2Byte	

各项的意义与 Configure-Request 相同。

Configure-Nak 包的使用方法如下所述：

如果收到的 Configure-Request 包中的配置项目是可接受的，也是自己愿意协商的，但是配置项目当中的一些值不能被接受，那么就必须传送一个 Configure-Nak 包。其中选项域仅由收到的 Configure-Request 包中不可接受的配置选项所填充。

如果在对端列出的配置项目之外还有新的项目要求配置，则 Configure-Nak 也可以发送新的配置项目和值，以提醒对端将其列入 Congigure-Request 包中，作为下一次发送的请求项目。

Configure-Nak 包中的标识符域必须匹配应答的 Configure-Request 包。

接到的与以上要求不符的包将被静静的丢弃。

<4 >Configure-Reject

Configure-Reject 包的格式如下：

代码	标识符	长度	选项
0x04	1Byte	2Byte	

各项的设置与 Configure-Request 包相同。

Configure-Reject 包的使用方法如下所述：

如果收到的 Configure-Request 包中一些配置项目是不可辨认的或者不被协商所接受（由网络管理员配置的），则必须传送一个 Configure-Reject 包。选项域仅由接收到的 Configure-Request 包中不可接受的配置项目填充，且必须不被重定义或修改，所有可识别的和通过协商的配置项目被过滤出 Configure-Reject 包。

在 Configure-Reject 包中，标识符域必须匹配应答的 Configure-Request 包。

一个新的 Configure-Request 包再发送的时候，必须不包含任何 Configure-Reject 中列出的配置项目。

接到的与以上要求不符的包将被静静的丢弃。

<5>Terminate-Request 和 Terminate-Ack

Terminate-Request 和 Terminate-Ack 包格式如下：

代码	标识符	长度	数据
1Byte	1Byte	2Byte	

代码域

0x05　Terminate-Request

0x06　Terminate-Ack

标识符域

传送 Terminate-Request 包过程中，无论何时数据域的内容发生了改变或接收到对前一个请求有效的应答，标识符域必须改变。对于重新传送，标识符域可以保持不变。

传送 Terminate-Ack 包过程中，接收到的 Terminate-Request 包的标识符域被拷贝到要发送的 Terminate-Ack 包的标识符域。

长度域

长度域指出该配置选项（包括类型、长度和数据域）的长度。

数据域

数据域为零或多个八位字节，是长度值的变量，包含发送方使用的未解释的数据。该数据可以由任何二进制值组成。

使用方法如下所述：

链路的一端在接收到一个 Terminate-Request 包时，必须传送一个 Terminate-Ack 包。当接收到一个未被引出的 Terminate-Ack 包时表示对端在关闭或停止状态，或者需要另外再商议。

Terminate-Request 包和 Terminate-Ack 包提供了一个关闭连接的机制。链路的一端如果想要关闭一个连接应该先传送一个 Terminate-Request 包，并且应该不断发送，直到收到 Terminate-Ack 包；或者收到链路低层已经关闭的指示；或者已经接收到了足够多的 Terminate-Request 包，以至于有充分的理由关闭。

<6> Code-Reject

在接收到一个带有未知代码的 LCP 包时，则必须传送一个 Code-Reject 包报告回未知代码的发送方。当接收到一个 Code-Reject 包，则应该报告问题并结束连接。

Code-Reject 包格式如下：

代码 0x07	标识符 1Byte	长度 2Byte	被拒绝包

对于每次 Code-Reject 包的发送，标识符域必须被改变。

被拒绝包域包含被拒绝的 LCP 包的拷贝。它由数据域开始，并且不包括任何数据链路层的头或者 FCS。被拒绝包域有时必须被缩短来与对端指定的 MRU 匹配。

<7>Protocol-Reject

Protocol-Reject 包格式如下：

代码 0x08	标识符 1Byte	长度 2Byte	被拒绝的协议 2Byte	被拒绝的信息

对每一次 Protocol-Reject 包的发送，标识符域必须被改变. 被拒绝的协议域包含被拒绝的 PPP 协议域。被拒绝的信息域包含被拒绝的包的拷贝，由数据域开始，不包含任何数据链路层头或 FCS。被拒绝的信息域必须削减来适应对端制定的 MRU。

使用方法如下所述：

当接收到一个带有未知协议域的 PPP 包时，表示对端试图使用一个不支持的协议。这通常发生在对端试图配置一个新的协议时。此时如果 LCP 处于打开状态，就必须传送一个 Protocol-Reject 包回应对端。对端在接收到 Protocol-Reject 包后，必须尽早停止发送被指出的协议包。

Protocol-Reject 包只能在 LCP 的打开状态下发送，在其它的状态下接收到的包应该被静静的丢弃。

<8> Echo-Request 和 Echo-Reply

Echo-Request 和 Echo-Reply 包格式如下:

代码 1Byte	标识符 1Byte	长度 2Byte	Magic-Number 4Byte	数据

代码

0x09 为 Echo-Request；

0x0A 为 Echo-Reply

长度域

长度域指出该配置选项（包括类型、长度和数据域）的长度。

传输 Echo-Reques 包过程中，无论何时数据域内容改变，并且无论何时接收到前一个请求的有效响应，标识符域都必须被改变。对于重新传送标识符可以保持不变。

传输 Echo-Reply 包过程中，Echo-Reply 包的标识符域是回复的 Echo-Request 包的标识符域的拷贝。

Magic-Number 域用于检测链路是否处于自环状态。在 Magic-Number 配置项目被成功的协商之前，Magic-Number 必须被以零传送。

数据域是零或者多个八位字节，包含发送方使用的未解释的数据。该数据可以由任何二进制值组成。

Echo-Request 包 和 Echo-Reply 包的使用方法如下所述：

Echo-Request 包 和 Echo-Reply 包用于检测数据链路层链路双方是否正常运转。这对调试、链路质量检测、测试和其它众多功能有帮助。

Echo-Request 和 Echo-Reply 包仅在 LCP 的打开状态下才能发送，在其它状态下接收到的 Echo-Request 和 Echo-Reply 包应该被静静的丢弃。在 LCP 的打开状态下，接收到一个 Echo-Request 包时，必须传送一个 Echo-Reply 包。

<9> Discard-Request

Discard-Request 包格式如下：

代码	标识符	长度	Magic-Number	数据
0x0B	1Byte	2Byte	4Byte	

每个 Discard-Request 包发送时标识符域必须改变。

长度域指出该配置选项（包括类型、长度和数据域）的长度。

Magic-Number 域对检测链路是否处于自环状态有帮助。在 Magic-Number 配置选项被成功的协商之前，Magic-Number 必须以零传送。

数据域是零或者多个八位字节，包含发送方使用的未解释的数据。该数据可以由任何二进制值组成。

Discard-Request 包使用方法如下所述：

Discard-Request 包用于测试远端链路接收器。有助于调试、执行测试和其它众多功能。Discard-Request 包仅在 LCP 的打开状态下才能发送。接收器必须静静的丢弃任何收到的 Discard-Request 包。

§3-2：LCP 定时器和计数器

• 重启定时器

重启定时器被用于计算 Configure-Request 和 Terminate-Request 包的传输时间。重启定时器到时产生一个超时事件，并且通知 Configure-Request 或 Terminate-Request 包重新传送。重启定时器必须是可配置的，缺省为三秒。

应该根据链路的速度调节重启定时器。缺省值被指定为低速（2,400～9,600 bps），高交换等待时间链路（典型电话线）。更高的速度链路或低交换等待时间的链路应该相对有更快的再次传输。重启定时器可以从最初的小一些的值开始增加到配置的最终值，以此来代替恒定值。每一个小于最终值的连续值应该至少是前一个值的两倍。初始值应该对包的大小来说足够大，是以线路速率传输往返时间的两倍，并且至少附加 100 毫秒来允许对端处理要响应的包。一些电路又加了 200 毫秒的附加迟延。

·重启计数器

Max-Terminate

必须有一个 Terminate-Requests 的重启计数器。Max-Terminate 显示 Terminate-Request 包发送后对端可能不会应答的没收到的 Terminate-Ack 包的个数。Max-Terminate 必须是可配置的，缺省为两次传输。

Max-Configure

Configure-Requests 采用一个类似的计数器。Max-Configure 显示 Configure-Request 包发送后对端没应答而没收到的 Configure-Ack，Configure-Nak 或 Configure-Reject 包的个数。Max-Configure 必须是可配置的，缺省为十次传输。

Max-Failure

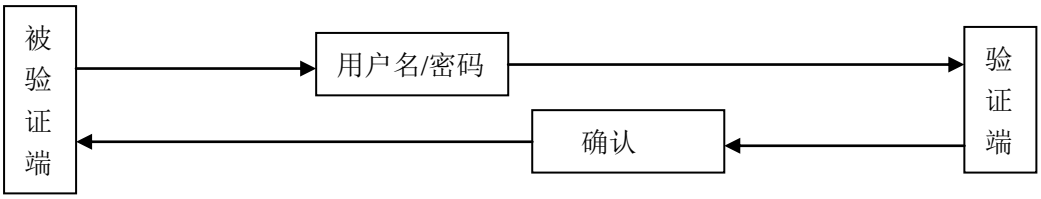
Configure-Nak 采用一个相关的计数器。Max-Failure 显示 Configure-Nak 包发送后 Configure-Ack 包发送前的 Configure-Nak 包的个数。任何更进一步的用于对端请求的选项被转换到 Configure-Reject 包，并且不再附加局部要求选项。Max-Failure 必须是可配置的，缺省为五次传输。

第四章：PAP 协议:

密码验证协议（PAP）提供了一种简单的方法，可以使链路双方使用 2 次握手建立身份验证。在链路建立完成后，链路一端不停地发送用户名/密码对给验证者，一直到验证被响应或者连接被终止为止。

PAP 不是一个健壮的身份验证方法。密码在电路上是明文发送的，并且对回送或者重复验证和错误攻击没有保护措施。对端控制着尝试的频率和时间。这个验证方法最适合用在使用有效的明文密码模拟登陆远程主机的情况。这个方法向普通用户提供了一种相似安全级别登陆远程主机。但要注意减少在 PPP 链路上传输明文密码和避免在整个网络上发送明文密

码。



PAP 两次握手认证
图 4.1

- 配置选项格式

下面是 PAP 验证协议配置选项的格式：

类型	长度	协议
0x03	0x04	0xC023

- PAP 包格式

一个 PAP 包是完全封装在 PPP 帧（协议域是 0xC023）的信息域中的。下面是 PAP 包格式的总结:

PPP 帧头	PPP 协议域	代码	标识符	长度	数据
0x 7E FF 03	0xC023	1Byte	1Byte	2Byte	

代码域代表 PAP 包的类型。PAP 代码分配如下：
0x01　　Authenticate-Request（认证请求）
0x02　　Authenticate-Ack（认证应答）
0x03　　Authenticate-Nak（认证无应答）

标识符用于匹配请求和响应。
长度域代表 PAP 包的长度，包括代码域，标识符和数据域。超出长度域指定的字节被认为是数据链路层的填料，在接收端应该忽略掉。
数据域是零个或多个字节。数据域的格式由代码域决定。

以下介绍不同 PAP 包的格式：

<1>Authenticate-Request

下面是 Authenticate-Request 包格式：

代码	标识符	长度	用户名长	用户名	密码长	密码
0x01	1Byte	2Byte	1Byte		1Byte	

标识符用于匹配请求和回应。每次发送一个 Authenticate-Request 包，标识符域必须改变。用户名长域代表用户名域的长度。用户名域是零个或多个字节，代表被验证端的名字。密码长域代表密码域的长度。密码域是零个或者多个字节，是用来验证的密码。

Authenticate-Request 包使用方法如下所述：

Authenticate-Request 包用来启动 PAP。在验证阶段链路被验证端必须传输代码域为 1 的 Authenticate-Request 包。并且 Authenticate-Request 包必须不停地被发送，直到接收到一个有效的响应包或者重试计数器超时。

验证端期待对端发送一个 Authenticate-Request 包。一旦接收到 Authenticate-Request 包，必须返回某种验证响应。由于 Authenticate-Request 包可能会丢失，所以在完成验证阶段后验证者必须允许接收重复的 Authenticate-Request 包。在另外的阶段接到的任何 Authenticate-Request 包必须被静静地抛弃。

<2>Authenticate-Ack 和 Authenticate-Nak

如果在接收的 Authenticate-Request 包中的用户名/密码对是可识别的和可接受的，则验证者必须发送一个 Authenticate-Ack 包。 如果在接收到的 Authenticate-Request 包中的用户名/密码对是不可识别的或不可接受的，则验证者必须发送一个 Authenticate-Nak 包，并且应该终止链路。

下面是 Authenticate-Ack 包和 Authenticate-Nak 包格式：

代码 1Byte	标识符 1Byte	长度 2Byte	信息长度 1Byte	信息

代码

0x02 Authenticate-Ack;

0x03 Authenticate-Nak

标识符域用于匹配请求和应答。此域必须从引起这次回应的 Authenticate-Request 包标识符域复制过来的。

信息长度域代表信息域的长度。

信息域是零个或者多个字节，并且它的内容依靠于实现者。它是可读的，不允许影响协议的操作。

为了加深理解，看下面这段 PAP 报文：

S:7E FF 03 C0 23 01 02 00 0A 03 48 48 48 01 51 C3 CB 7E

PAP Request ID 长度 用户名长 用户名 密码长 密码

发送端发出验证请求；

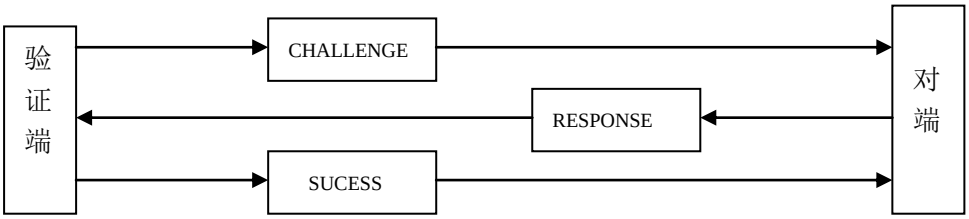
R:7E FF 03 C0 23 02 02 00 05 00 46 1E 7E

PAP ACK ID 长度 信息长度

接收端通过验证。

第五章：CHAP 协议

CHAP 使用 3 次握手周期性的验证对端。在链路建立初始化时这样做，也可以在链路建立后任何时间重复验证。在链路建立完成后，验证者向对端发送一个“challenge”（挑战）信息。对端使用“one-way-hash”（单向哈希）函数配合密钥（secret）计算出的值响应这个挑战信息。验证者使用自己计算的 hash 值校验响应值。如果两个值匹配，则验证是成功的，否则连接应该终止。



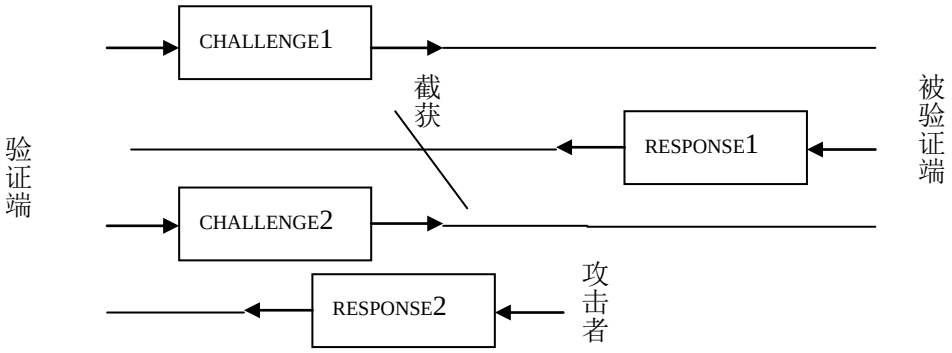
CHAP 三次握手认证

图 5.1

CHAP 通过使用递增的标识符和可变的挑战值防止回送攻击。使用重复挑战的目的是验证者控制着挑战的频率和时间。这种验证方法依靠的是只有验证双方知道的秘钥。秘钥不在链路上传播。

CHAP 要求秘钥是明文形式的。为了避免在网络的其它链路上发送秘钥，建议配置一台中心服务器统一管理秘钥，秘钥应该以可逆转的加密形式发送到服务器，在中心服务器上检查 challenge 和 respone 值，而不是在每一个网络访问服务器上检查。CHAP 算法要求秘钥的长度必须至少一个字节。比较好的秘钥的长度应该至少是选择的哈希算法的哈希值的长度（对于 MD5 是 16 个字节）。这样保证了秘钥足够防止穷尽搜索攻击。选择单向哈希算法使得要想从已知的 challenge 和 response 值得出秘钥的计算是不可行的。

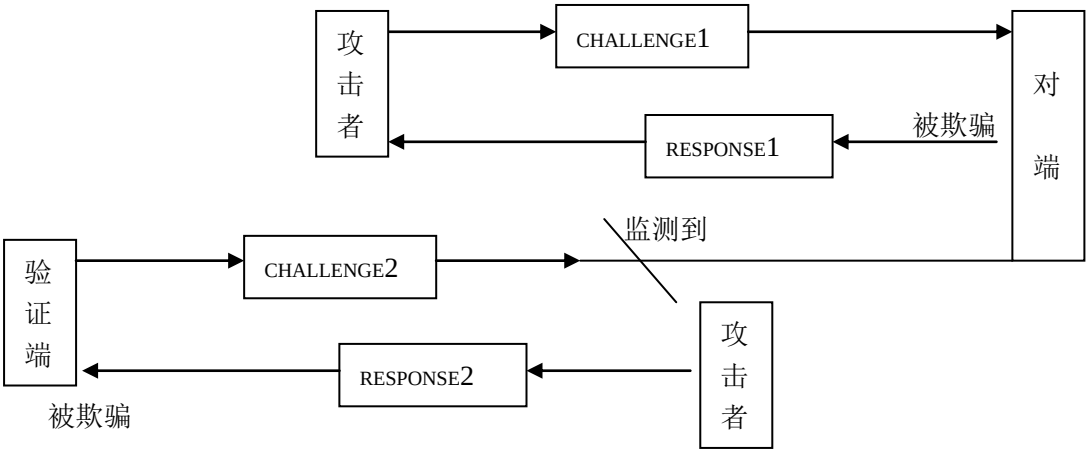
Challenge 值应该符合两个标准：唯一性和不可预测性。每一个 Challenge 值应该是唯一的，因为使用与相同秘钥联系的 Challenge 值的副本可能让攻击者利用前一个截获的响应包响应（如图 5.2）。由于希望可以在不同区域服务器中使用相同的秘钥验证，Challenge 应该具有全局临时唯一性。每一个 challenge 值也应该是不可预测的，否则攻击者欺骗对端响应一个可预测的未来 challenge，然后用这个响应伪装成对端欺骗验证者（如图 5.3）。尽管 CHAP 协议不能够防止实时的窃听攻击，但是使用唯一的和不可预测的 challenge 可以防止一定范围的能动攻击。



Challenge2=Challenge1　　Response2=Response1

注：图中不考虑 Challenge 与 Response 的代码域

图 5.2



Challenge1 为攻击者预测的；Challenge2=Challenge1； Response2=Response1

注：图中不考虑 Challenge 与 Response 的代码域

图 5.3

• 认证步骤：

1. 链路建立阶段结束之后，认证者向对端发送“挑战”消息。
2. 对端用经过单向哈希函数计算出来的值做应答。
3. 认证者根据它自己的预期哈希值的计算来检查应答，如果值匹配，认证得到承认；否则，连接应该终止。
4. 经过一定的随机间隔，认证者发送一个新的挑战给对端，重复 1 到 3。

• 配置选项格式

协商 CHAP 认证协议的配置选项格式如下图所示：

类型	长度	认证协议	算法
0x03	0x05	0xC223	1Byte

算法域代表所使用的单向哈希算法。CHAP 算法域当前的值分配如下：

0-4 unused(保留)

5 MD5 算法

• CHAP 包格式

CHAP 数据包封装在 PPP 数据链路层帧的数据域中，PPP 的协议字段指示 0xC223，CHAP

包格式如下图所示：

PPP 帧头 7E FF 03	PPP 协议域 0xC223	代码 1Byte	标识符 1Byte	长度 2Byte	数据

代码字段指示 CHAP 包的类型，分配如下：

1　　挑战　Challenge
2　　应答　Response
3　　成功　Success
4　　失败　Failure

标识符字段辅助匹配挑战、应答和响应。

长度字段指示 CHAP 包的长度，包括代码、长度和数据字段。超出长度的字节应该视为数据链路层的填充，接收方应该忽略。

数据字段是零个或多个字节，数据字段格式由代码字段确定。

以下具体介绍不同的 CHAP 包：

①挑战和应答包

挑战包和应答包的格式如下:

代码 1Byte	标识符 1Byte	长度 2Byte	值长度 1Byte	值	名字

代码
0x01　　挑战 Challenge;
0x02　　应答 Response.

标识符字段一字节，每次发送挑战都必须改变标识符字段的值。应答标识必须与响应的挑战标识字段相同。

长度字段指示 CHAP 包的长度，包括代码、长度和数据字段。超出长度的字节应该视为数据链路层的填充，接收方应该忽略。

值长度指示“值”字段的长度。

值字段即发送的挑战或应答值。挑战值是可变字节流。挑战值必须在每次发送挑战时都要改变。挑战值的具体长度由产生它的方法而定，独立于所用的哈希算法。应答值由标识串接密钥串接挑战值然后经过哈希运算得出，其长度由所用的哈希算法决定。

名字字段一个或多个字节，代表传输数据包的系统的标识，字段内容没有限制。名字不应该是 NULL 或者 CR/LF 终止符。其长度由长度字段确定。因为 CHAP 可以验证许多不同的系统，所以名字域的内容可以用作在秘密数据库查询秘密的关键字。也可以在每个系统上支持更多的 Name/Secret 对。

挑战包和应答包的用法如下所述：

挑战包是 CHAP 的开始。在发送挑战包后，其它的挑战包必须在有效的应答包成功接收之后或者在可配置的重试计数器计满后发送。为了确保连接没有被更改，挑战包也可以在 NLP 阶段的任何时候发送。对端应该随时为认证阶段和 NLP 阶段的挑战做好准备，任何时候收到挑战包，都必须传送应答包。认证者无论何时收到应答包，都必须把应答值和自己计算的预期值比较。基于这种比较，认证者必须发送成功（Success）或者失败（Failure）CHAP 包。

由于成功包可能丢失，认证者在 NLP 阶段中必须允许重复的应答包。如果“失败包”丢失，认证者终止了链路，那么可以由 LCP 的“终止请求”和“终止应答”来指示认证失败。

应答包的代码域必须与应答的挑战包的代码域相同（消息部分可能不相同）。在任何其他阶段收到的任何应答包必须静静的丢弃。

② 成功与失败包

如果接收到的应答值等于预期值，那么认证者必须传送成功包。如果接收到的应答值不等于预期值，那么认证者必须传送失败包，并且应该终止链路。

成功和失败包格式如下所示:

代码 1Byte	标识符 1Byte	长度 2Byte	信息

代码：
0x03　　成功 Success
0x04　　失败 Failure

标识字段用于辅助匹配应答和响应，从响应的标识字段拷贝得来。
消息字段是一个或者多个字节，其内容与具体实现无关，最好可以直接阅读，但是不得影响协议操作，推荐使用可显示的 ASCII 字符（从 32 到 126）。其长度由长度字段确定。

为加深理解，我们看以下这段 CHAP 协商过程：

R:7E C2 23　01　01　00 19　14　BF 79 42 12 E1 A6 5B F6 71 C2 E3 CA 70 CD D9
　CHAP　挑战 ID　长度 值长度　值
8C DE C7 3F 3E DA 58 7E
接收端发出挑战值；

S:7E C2 23　02　01 00 18　10　BF 56 B6 AF 9A CA 15 B9 25 78 62 18 7F 9A B7 D3
　CHAP 应答 ID 长度 值长度　值
　68 6C 79　BF D2　7E
　NAME=hly

发送端应答；

R:7E C2 23　01　02 00 19　14　BB B0 14 DE 08 88 57 6E C3 4F 0D 0A EC ED 05
CHAP 挑战 ID 长度 值长度 值
2C 59 86 AA BC　93 67　7E
接收端发出挑战值；

S:7E C2 23 02　02　00 18　10　BB 74 AE 1B 09 CC 7A F6 7C F1 04 19 4C 7D 5E E3
CHAP 应答 ID 长度 值长度 值
68 6C 79　B3 34　7E
NAME=hly
发送端应答；

R:7E C2 23　03　02　00 04 A6 53 7E
CHAP 成功 ID　长度
接收端通过验证。

第六章：PPP 链路质量监控（LQM）

数据通信链路是很少完美的。由于各种原因（例如线路噪音，设备失败，缓存溢出等等），链路上的包可能被丢掉或者被损坏。有时候，有必要决定什么时候链路丢数据，丢包频率。例如，路由器可能要暂时允许另一个路由器取得优先权。一个连接可能选择断开或切换到另一个替换的链路。决定数据丢失的过程被称为“链路质量监控”（Link Quality Monitoring）。

有很多不同的方法测量链路质量，并且有更多的路径去实现这些方法。链路质量监控被分为一个机制（mechanism）和一个策略（policy）。PPP 通过定义链路质量报告包（Link-Quality-Report Packet）和它的详细处理过程，为链路质量监控详细说明了监控机制。PPP 没有具体说明链路质量监控策略——如何断定链路质量或者当链路不充分时该怎么做。这个被留做一个实现决策，并且在链路的各端可能是有差别的。用各种各样的方法去实现这一决策是被允许甚至鼓励的。链路质量监控机制说明书保证了使用不同策略的两个实现可以实现通信和进行内部操作。

为了允许实现灵活的策略，PPP 链路质量监控机制以包、字节和链路质量报告为单位测量数据丢失率。每种测量方法被分别用来测量链路的每半部分，包括内部和外部。所有的测量方法被通知给链路的两端，以致链路的每一端能够为它的输入和输出链路实现自己的链路质量策略。

- 计数器

每种链路质量监控的实现靠计数器记录着发送和成功接受到的包和字节的数目，并且定期的用链路质量报告包把这个信息发送给对端。

这些计数器类似于序列号。它们一直增加，记录通过外部链路的包和字节的数目。通过

比较连续的 LQR 中的数值，LQR 接受者可以通过计算包和字节的差值来知道链路成功通信的流量。

LQR 使用由 SNMP MIB-II[2]定义的接口计数器。当 LCP 进入建立阶段时，这些计数器并不初始化为任何值。另外，LQR 要求实现下面三个无符号的，单调递增的计数器，它们符合 SNMP MIB 计数器要求的类型和大小：

1．OutLQRs

OutLQRs 是一个 32 位的计数器。每发送一个 LQR 包，它递增 1。在 LCP 进入建立阶段时，这个计数器必须置零，并且在 LCP 离开终止阶段前不得被重置。这个计数器值在插入 LQR 包前增 1。

2．InLQRs

InLQRs 是一个 32 位的计数器。每接收一个 LQR 包，它递增 1。在 LCP 进入建立阶段时，这个计数器必须置零，并且在 LCP 离开终止阶段前不得被重置。这个计数器值在插入 LQR 包前增 1。

3．InGoodOctes

InGoodOctes 是一个 32 位的计数器。它每次增加每个正确接收的数据链路层包的字节数。这个计数器在 LCP 进入建立阶段时可以被初始化为任何值。但是直到 LCP 离开终止阶段前，不能被重置。

• 处理过程

PPP 链路质量监控机制可以用一个“逻辑过程”模型表示。如下图（图 6.1）所示，在每个双向链路的每一端有五个逻辑过程。

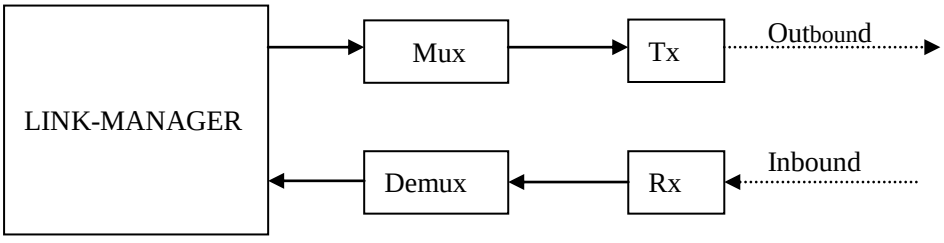


图 6.1

链路管理器(link-manager)：

链路管理器传输和接收 LQR，和实现所期待的链路质量策略。LQR 包以恒定的速率传输。这个速率是由 LCP 质量协议配置选项协商得到的。

Mux：

Mux 把来自各个协议（例如 LCP，IP 等等）的多元包处理成一个单独的，连续的，有

优先级的包流。LQR 包必须被赋予可能的最高优先级，以保证链路质量信息及时被传输。

Tx：

Tx 过程维护着 MIB 计数器 ifOutUniPackets 和 ifOutOctets,和内部计数器 OutLQRs，它用来测量在输出链路上传输的数据量。当 Tx 处理 LQR 包时，它把这些计数器的值插入到包中的 PeerOut 域。Tx 过程必须跟在 Mux 过程后面以确保这些包以在链路上传输的顺序计数。

Rx：

Rx 过程维护着 MIB 计数器 ifInUniPackets,ifInDiscards,ifInErrors 和 ifInOctets,和内部计数器 InLQRs 和 InGoodOctets，它用来测量在输入链路上接收的数据量。当 Tx 处理 LQR 包时，它把这些计数器的值插入到包中的 SaveIn 域。

Demux：

Demux 过程为各种协议分解多元包。Demux 过程必须跟在 Rx 过程后面以确保这些包以在链路上接收的顺序计数。

• 配置选项格式

链路一端在接收 LQR 前应该准备接收质量协议配置选项（Quality-Protocol Configuration Option）。但是，协商并不被要求。仅在链路一端希望确保对端传输不同于其它质量协议的 LQR，或者防止对端维护自己的计时器，或者在 LQR 传输间建立最大的时间间隔时，协商才是必须的。

下面是协商 LQR 的质量协议配置选项格式：

类型	长度	质量协议	报告周期
0x04	0x08	0xC025	4Bytes

报告周期：

报告周期域（the Reporting-Period field）表明了在包传输间的最大时间间隔（以 1/100 秒计算）。对端可以以此商议更快速率的传输包。

此值为零表明对端不需要维护计时器。作为替代，对端一旦接收 LQR 立即产生一个 LQR。当链路一端已经发送或者将要发送一个 Configrue-Request 包，且包中的报告周期字段为零值时，报告周期为非零值的对端应发一个 Configure-Nak 包。

• 包格式

LQR 包被封装在 PPP 数据链路层帧的数据域中，此帧的协议域为 0xC025（LQR）。下面是 LQR 包格式的总结。每个字段的域名相对于包的接收者：

Magic-Number	LastOutLQRs	LastOutPackets	LastOutOctets	PeerInLQRs	PeerInPackets
4Bytes	4Bytes	4Bytes	4Bytes	4Bytes	4Bytes

PeerInDiscards 4Bytes	PeerInErrors 4Bytes	PeerInOctets 4Bytes	PeerOutLQRs 4Bytes	PeerOutPackets 4Bytes	PeerOutOctets 4Bytes

下面的各个域实际上并不经过输入链路传输。相反，它们逻辑上被接收者的 Rx 过程加到包上。

SaveInLQRs 4Bytes	SaveInPackets 4Bytes	SaveInDiscards 4Bytes	SaveInErrors 4Bytes	SaveInOctets 4Bytes

Magic-Number：

魔数字段用于辅助检测链路自环。魔数必须以零值传输并且在接收端被忽略，除非有配置选项设置。

LastOutLQRs：

LastOutLQRs 是从最近接收的 PeerOutLQRs 复制过来的。

LastOutPackets：

LastOutPackets 是从最近接收的 PeerOutPackets 复制过来的。

LastOutOctets：

LastOutOctets 是从最近接收的 PeerOutOctets 复制过来的。

PeerInLQRs：

PeerInLQRs 是从最近接收的 SaveInLQRs 复制过来的。

无论何时发现 PeerInLQRs 域为零，LastOut 域是不确定的，并且 PeerIn 域包含对端的初始化值。

PeerInPackets：

PeerInPackets 是从最近接收的 SaveInPackets 复制过来的。

PeerInDiscards：

PeerInDiscards 是从最近接收的 SaveInDiscards 复制过来的。

PeerInErrors：

PeerInErrors 是从最近接收的 SaveInErrors 复制过来的。

PeerInOctets：

PeerInOctets 是从最近接收的 SaveInOctets 复制过来的。

PeerOutLQRs：

PeerOutLQRs 是从接收的 OutLQRs 复制过来的。这个数必须包含此 LQR。

PeerOutPackets：

PeerOutPackets 是从当前的 MIB ifOutUniPackets 和 ifOutNUniPackets 复制过来的。这个数必须包含此 LQR。

PeerOutOctets：

PeerOutOctets 是从当前的 MIB ifOutOctets 复制过来的。这个数必须包含此 LQR。

SaveInLQRs：

SaveInLQRs 是从接收的 InLQRs 复制过来的。这个数必须包含此 LQR。

SaveInPackets：

SaveInPackets 是从当前接收的 MIB ifInUniPackets 和 ifInNUniPackets 复制过来的。这个数必须包含此 LQR。

SaveInDiscards：

SaveInDiscards 是从当前接收的 MIBifInDiscards 复制过来的。这个数必须包含此 LQR。

SaveInErrors：

SaveInErrors 是从当前接收的 MIB ifInErrors 复制过来的。这个数必须包含此 LQR。

SaveInOctets：

SaveInOctets 是从当前接收的 InGoodOctets 复制过来的。这个数必须包含此 LQR。

注意 InGoodOctets 和 MIB ifInOctetes 计数器不一样，因为 InGoodOctets 不包括被丢掉的或者有错的包中的字节数。

·报告传输

当 PPP 链路控制协议进入打开状态，链路质量监控过程可以开始发送 LQR。如果接收到 LQR 的 Protocol-Reject 包，LQM 过程必须终止发送 LQR 包。

一般说来，当链路的 LQR 计时器超时时就发送 LQR。如果没有使用 LQR 计数器，则一旦收到进入的 LQR 就产生一个 LQR。协商过程确保至少链路的一方使用 LQR 计时器。

另外，无论何时接收到两个连续的具有相同的 PeerInLQRs 值的 LQR，只产生一个 LQR。这表明其中一个 LQR 已经丢失过，或者接收方以低于对端的速率发送 LQR，或者对端加速 LQR 产生。无论何时 LQR 被发送，LQR 计时器必须重新启动。

·计算

每当从输入链路接收到 LQR 包，Link-Manager 就比较相关的域。用当前 LQR 的各个域值减去前一个 LQR 的各个域值就可以得到绝对的变化值，这样链路的两端可以看到变化了。

如果接收的 PeerInLQRs 域为零，则 LastOut 域是不确定的，并且 PeerIn 域包含对端的初始化值。这时不作任何计算。

下面的计数器达到最大值后会变成 0。必须注意这点，保证此时能够计算出正确的变化值。

LastOutLQRs。域可以直接和 PeerInLQRs 域比较来决定丢失了多少 outbound 的 LQR。

LastOutLQRs。域可以直接和 OutLQRs 计数器比较来决定有多少 outbound 的 LQR 仍在传递中。

PeerInPackets 的变化可以和 LastOutPackets 的变化比较来决定输出链路上丢失包的数目。

PeerInOctets 的变化可以和 LastOutOctets 的变化比较来决定输出链路上丢失 Octets 的数目。

SaveInPackets 的变化可以和 PeerOutPackets 的变化比较来决定输入链路上丢失包的数目。

SaveInOctets 的变化可以和 PeerPackets 的变化比较来决定输入链路上丢失 Octets 的数目。

PeerInDiscards 和 PeerInErrors 可以用来决定是否包丢失是因为对端的拥塞而不是链路失败。

·失败检测

当链路在链路的两个方向上正常工作时，LQR 是多余的。传输 LQR 的最大时间间隔应该选择为最小限度干涉传输的值。

当在任一个方向上存在可测量的数据丢失时，如果全部吞吐量是充分的，则这种条件是不足够解释链路丢失的。除了能够测量到丢失率的峰值，快速发送 LQR 是没有什么结果的。必须选择一个长时间间隔以足够保证有一个好的数据平滑影响，相应的短的时间间隔可以确保快速响应失败。如果链路输入正常，输出情况糟糕，则输入的 LQR 会表明在链路的输出方向上存在高丢失率。快速发送 LQR 是没有帮助的，因为它们可能会在到达对端的链路上丢失的。如果链路输出正常，但输入情况糟糕，则输入 LQR 将会频繁丢失。在这种情况下，应该以更快的速率发送 LQR。这主要依靠对端做的信息策略决策。对端也可以在相应情况中发送 LQR（复制 PeerInLQRs 域），这样某些 LQR 也许能成功到达。

如果 LQR 在期待的时间内没有到达，或者接收的 LQR 表明链路情况真的很糟，则至少发送一个额外的 LQR。算法决策需要至少两个往返时间间隔。由于链路高负载或者输出 LQR 丢失，这个丢失率可能是暂时的。

第七章：IPCP 协议：

IP 控制协议（IPCP）负责配置 PPP 链路的两端上 IP 协议传输。IPCP 使用和链路控制协议（LCP）同样的包交换机制。IPCP 包在 PPP 协议达到网络层的协议阶段以前不被交换。在本阶段达到之前收到的 IPCP 包应该被静静地抛弃。

IP 控制协议除下列情形外，是和链路控制协议 LCP 同样的东西：

1.协议域

IPCP 包是被压缩在协议域类型为 0x8021（IP 控制协议）的 PPP 帧的数据字段中的。

2.代码域

只有代码 1 到 7（Configure-Request，Configure-Ack，Configure-Nak，Configure-Reject，Terminate-Request，Terminate-Ack 和 Code-Reject）被使用。其他的编码应该被处理为未经承认并且应该导致 Code-Rejects 结果。

3.超时

IPCP 包在 PPP 协议达到网络层的协议阶段才被交换。在等待认证和链路质量检测完成的 Configue-Ack 或者其它应答之前，将不执行超时命令。只有在用户干预或者配置的超时时间间隔之后，超时才被执行。

• IPCP 包

在任何 IP 包可以被传输之前，PPP 协议必须达到网络层协议阶段，IP 控制协议必为打开状态。在 PPP 协议链路上面被传送的 IP 包的最大的长度和 PPP 协议的数据字段的最大的长度相等。大的 IP 数据报必须分片传输。

IPCP 帧格式如下：

PPP 帧头 7E FF 03	PPP 协议域 0x8021	代码 1Byte	标识 1Byte	长度 2Byte	配置选项

• 配置可选项

配置选项格式如下：

类型 1Byte	长度 1Byte	数据

类型域的值分配如下：

0x01　　IP-地址（Addresses）
0x02　　IP-压缩协议
0x03　　IP-地址（Address）

长度域的值表示配置选项的长度，包括类型、长度、数据字段。数据域的值根据类型域而不同。

下面分别介绍不同类型的配置选项：

(1) IP-地址（IP-Addresses）

IP-地址（IP-Addresses）配置选项的使用已经被反对了。它难以在全部使用本选项的例子中保证协商收敛。建议用 IP-地址（IP-Address）配置选项代替本选项，并且它的使用是首选的。

(2)IP-压缩协议

本配置选项用于协商特定的压缩协议。在默认情况下，压缩不使用。

IP-压缩协议配置可选项格式如下：

类型 0x02	长度 1Byte	IP-压缩协议 2Byte	数据

长度值大于等于 4。

IP-压缩协议域的值分配如下：

值	协议
0x002d	Van Jacobson Compressed TCP/IP(TCP/IP 报头压缩)

后面会专门介绍 Van Jacobson TCP/IP 报头压缩。

数据区是零或者由特别压缩协议决定的更多的八位字节附加数据。

（3） IP-地址（IP-Address）

本配置选项提供协商在链路一端上使用的 IP 地址的方法。它允许 Configure-Request 的发送方声明要求哪个 IP 地址，或者请求对端提供信息。对端可以通过 Configure-Nak 提供有效的 IP 地址信息。

IP-地址（IP-Address）的配置可选项格式如下：

类型	长度	IP 地址（cont）
0x03	0x06	4Byte

IP 地址即 4 字节 IP 地址。如果是发送者发出的 Configure-Request 包，即表示要求的本端地址。如果全部 4 个八位字节都是零值，它表示请求对端提供 IP-地址信息。 缺省 IP 地址不被分配。

• Van Jacobson TCP/IP 报头压缩

Van Jacobson TCP/IP 头部压缩算法的基本想法就是维持一定的连接信息，这样双方交互的包文就只需要标识其属于那个连接而不需要携带完整头部了。这个算法可以将 40 个字节的 TCP 和 IP 头部压缩成 3 个字节,这可以显著的改善低速链路的通信。

IP-压缩协议配置可选项被用来表示接收压缩包的能力。如果要求双向压缩，链路的每一端都必须独立地请求配置本可选项。

Van Jacobson TCP/IP 报头压缩 IP-压缩协议配置选项的格式如下:

类型	长度	IP-压缩协议	最大时间片标识	压缩时间片标志
0x02	0x06	0x002D	1Byte	1Byte

最大时间片标识域表示时间片标识的最大值。实际可以取到的时间片标志比最大时间片标识域的值大 1，因为它可以取从零到最大时间片标识域的值。

压缩时间片标志域是表示时间片标识域是否可以被压缩。

0 时间片标识符不必被压缩。

1 时间片标识符可以被压缩。

当传送 IP 包的时候，PPP 协议域是对下列值的置值：

0x0021 典型 IP（IP 协议不是 TCP，或分割包，或不能压缩）

0x002d 压缩 TCP（TCP/IP 报头被压缩报头替换）

0x002f　未压缩 TCP（IP 协议域被时间片标识符替换）

下面是截取的一段 IPCP 协商报文：

S:7E 80 21　01　05 00 10　02　06　00 2D　0F　01
　IPCP REQUEST ID 长度 压缩协议 长度 TCP 报头压缩 :最大时间片 时间片可压
　03　06　00 00 00 00　64 54　7E
IP 地址 长度 请求对端分配
发送请求；

R:7E 80 21　04　05　00 0A 02 06 00 2D 0F 01　63 E1　7E
　IPCP REJECT ID　长度　拷贝上文
接收端拒绝压缩；

S:7E 80 21　01　06　00 0A 03 06 00 00 00 00　6A 5F　7E
　IPCP REQUEST ID　长度 请求对端分配 IP
发送请求对端分配 IP;

R:7E 80 21　03　06 00 0A　03 06 0A C8 13 06　15 2E　7E
　IPCP　NAK ID 长度　分配 IP:0A.C8.13.06
接收端建议对端请求分配 IP:0A.C8.13.06

S:7E 80 21　01　07 00 0A 03 06 0A C8 13 06　17 6B　7E
　IPCP REQUEST ID 长度　IP 地址
发送请求，申请 IP:0A.C8.13.06

R:7E 80 21　02　07 00 0A 03 06 0A C8 13 06　24 B0　7E
　IPCP　ACK ID 长度　拷贝上文
接收方同意 IP 申请。

第八章：压缩控制协议 CCP

在叙述压缩控制协议 CCP 之前，我们有必要对和 PPP 相关的压缩进行总结，这样有助于清楚的了解各压缩所处的位置。

和 PPP 相关的压缩分为两类, 一类是 PPP 帧头结构的压缩, 一类是 PPP 所承载的上层协议报文的压缩。

PPP 帧头结构的压缩主要是指 PPP 帧头部 Address、Control 和 Protocol 字段的压缩, 关于这些压缩的协商发生在 PPP 链路建立阶段。由于 PPP 头部的 Address 和 Control 字段都是固定不变的, 所以为了节省带宽就可以压缩这两个字段。PPP 会在链路建立阶段用 Address-and-Control-Field-Compression 选项协商是否压缩这两个字段。由于典型的承载在 PPP 之上的协议的协议号都小于 2 5 6（0x0100），所以用于协议号的 16 位二进制数就可

以压缩为 8 位。PPP 也是在 LCP 阶段用 Protocol-Field-Compression 选项协商是否进行协议字段的压缩。

对于 PPP 所承载上层协议报文的压缩，由于是对上层报文的压缩，所以对于压缩技术的协商必然发生在网络阶段，也就是链路建立、认证阶段之后。对于 PPP 上层报文而言，我们这里只讨论 TCP/IP 报文。对于 TCP/IP 报文而言，又有两种压缩的方法：一是 TCP/IP 报文首部的压缩，一是 TCP/IP 整个报文或者说 PPP 数据部分的压缩。对于首部压缩而言，目前标准的压缩算法是前面一章介绍的 Van Jacobson TCP/IP 头部压缩算法，这个算法使用与否的协商发生在 IPCP 协商阶段,是作为 IPCP 的一个参数: IP-Compression-Protocol 协商的。对于 PPP 数据部分的压缩技术的协商,由于数据报文的压缩技术种类比较多，而且事实上不基于任何网络协议，所以 IETF 特别规范了一个专门的协议:CCP(PPP Compression Control Protocol)用于协商具体压缩算法、压缩算法的具体参数。

下面就开始介绍本章的内容压缩控制协议 CCP：

压缩控制协议（CCP）负责在 PPP 链路上的两端配置并协商采用哪种压缩算法。并且用可靠的方式来标志压缩和解压缩机制的失败。CCP 采用和 LCP 相同的分组交换自动机制。直到 PPP 已经到达网络层协议的阶段,CCP 包之间才交换。在未达到这个阶段时所接收到的 CCP 包将被静静地丢弃。

CCP 与 LCP 基本相同，主要有以下不同：

（1）协议域

每个 CCP 包被封装在 PPP 的信息域里。此时 PPP 协议域的内容为 0x80FD（表明是 CCP）。当单个链路的数据压缩算法用在到达同一目的地的多重链路上时， PPP 协议域的内容为 0x80FB（表明是单链路压缩控制协议）。

（2）代码域

除了代码 1 到 7(Configue-Request,Configue-Ack,Configue-Nak,Configue-Reject，Terminate-Request，Terminate-Ack 和 Code-Reject） 外,CCP 中还定义了代码 14 和 15（Reset-Request 和 Reset-Ack）,其它被认为是不可识别的代码则做 Code-Reject 处理。

（3）超时设定

直到 PPP 达到网络层协议阶段时，CCP 包之间才交换。在等待认证和链路质量检测完成的 Configue-Ack 或者其它应答之前，将不执行超时命令。只有在用户干预或者配置的超时时间间隔之后，超时才被执行。

在任何压缩过的数据包被传送之前，PPP 必须处在网络层协议阶段，且 CCP 必须处在开始状态。一个或多个被压缩的数据包被封装在 PPP 的信息域里，此时 PPP 协议域的内容为 0x00FD（CCP）。每个压缩算法可能采用不同的机制来表明在一个 PPP 帧中包含多个未压缩的数据包。

当采用多重 PPP 链路到达同一目的地时，有两种利用数据压缩算法的方法。第一种方法是在多条链路上发送包之前压缩数据，将多条链路视为一条。第二种方法是把每条链路看作

是单独的分离的连接，可能进行压缩，也可能不进行压缩。在第二种情况下，PPP 协议域的内容为 0x00FB（单链路压缩控制协议）。

在每个方向每次只能采用一种主要的算法，并且在发送第一个压缩帧前进行协商。此时压缩数据包的协议域将表明是数据帧被压缩而不是数据帧和算法一同被压缩。

在 PPP 链路上传输压缩数据包的最大长度与封装的 PPP 包的数据域的最大长度一样。对于压缩后增大的数据包（如果采用某种压缩算法后，由于一些原因导致信息的长度增加）将按标准的数据包格式不压缩进行发送，或者如果压缩算法支持的话 ，可以分解成多个数据包进行发送。每个压缩算法必须提供一种确定其传输数据是否可靠的方法，或者使用可靠的传输机制，如 LAPB。

• CCP 包

CCP 包的格式和一些基本的功能已经在 LCP 中定义过了，下面着重介绍 CCP 代码域中规定的新的 Reset-Request 和 Reset-Ack 的取值：

Reset-Request 和 Reset-Ack 数据包格式如下所示：

代码	标识	长度	数据域
1Byte	1Byte	2Bytes	

代码域

0x0D Reset-Request

0x0F Reset-Ack

标识符

在传输时，标识符域的内容随着数据域内容的变化而变化，无论何时接收到的有效应答都应与之前发送的请求相对应。重传时，标识符保持不变。接收到 Reset-Request 数据包后，将其标识符域的内容拷贝到 Reset-Ack 数据包中的相应域。

数据域

数据域有零个或多个字节,可包含连续的数据供发送者使用。数据可能由二进制值组成，其长度为零到对方的 MRU（最大接受单元）减 4。

其用法如下所述：

为了提供一种指示一条压缩链路上某个方向的解压缩失败的机制，同时又不影响其它方向上传输的数据，CCP 中包含了 Reset-Request 和 Reset-Ack 的定义。通过定期的传递一个随机值,对解压缩的数据执行 CRC 校验,或者应用其它机制来确定是否发生了解压缩失败。在所有的压缩算法中，强烈推荐那些能够保证在解压缩的数据有效后，再将其传递到系统中其它部分的机制。

为了表明解压缩失败，CCP 应传递一个 Reset-Request 包,数据域中包含需要的数据。一旦 Reset-Request 包发送后，发送端将会丢弃任何接收到的压缩数据包，并且在收到一个有效的 Reset-Ack 包前继续发送带有相同标识符的 Reset-Request 包。

当接收到 Reset-Request 包后，要传递的压缩数据包将被置为初始状态。这包括清空压缩字典，重置哈希值，或者采取其它的机制。之后，一个 Reset-Ack 包被传递，其标识符域的值与应答的 Reset-Request 的标识符域的值相同，数据域中包含着需要的数据。当接收到 Reset-Ack 后，正被接收到的需被解压缩的数据包被置为初始状态。这包括清空压缩字典，重置哈希值，或者采用其它的机制。因为每个管道有不止一个的 Reset-Ack 包，所以要找到与当前指定的标识符相匹配的 Reset-Ack 包相对应的要被解压缩的数据包，并将其重置为初始状态。

• CCP 的配置选项

CCP 的配置选项允许协商压缩算法及其参数。CCP 采用 LCP 中定义的配置选项的格式，并包含一组独立的选项。

在 CCP 中，配置选项指定的算法是接收端希望接收到的能够解压缩发送端送来的数据的算法。因此系统希望能够提供一些可接受的算法，然后协商采用。也有可能没有协商出任何压缩算法，此时，就不采用任何压缩算法，链路在没有压缩算法应用的情况下继续运作。如果链路的可靠性已经被单独协商，则此链路可继续使用，直到 LCP 再次被协商。

LCP 协商选项的方法将被应用。如果某个选项不被识别，则发送 Configue-Reject 数据包，如果发送端执行的所有压缩协议都被接收端发送 Configue-Reject 数据包拒绝，则在链路的这个方向上不采用任何压缩算法。如果一个选项被承认，但是由于其在请求中的值（或者可选的参数不在请求中）不被接受，此时，接收端将发送带有修改后认为合适的选项的 Configue-Nak 数据包。Configue-Nak 数据包必须包含可被接受的那些选项。之后，一个新的 Configue-Request 数据包应当被发送，并带有一个可取的选项，此选项是在 Configue-Nak 数据包中指定的。

以下是 CCP 选项类型域中的值：

CCP 选项	压缩类型
0x00	OUI
0x01	Predictor type 1
0x02	Predictor type 2
0x03	Puddle Jumper
0x04-0f	unassigned
0x10	Hewlett-Packard PPC
0x11	Stac Electronics LZS
0x12	Microsoft PPC
0x13	Gandalf FZA
0x14	V.42bis compression
0x15	BSD LZW Compress（BSD 压缩协议）
0xff	Reserved

未利用的值 0x04-0f 准备分配给那些不需要版权费用可免费得到的压缩算法。

（1）Proprietary Compression OUI

这个配置选项提供了一种协商使用私有压缩协议的方法。因为第一个匹配的压缩算法会被采用，所以最好在普通的选项被采用之前，首先传输任何可知的 OUI 压缩选项。在接受这个选项之前，结构中唯一的标识符确定的私有压缩算法是否能够解压缩要得到证实，并且任何供应商指定的具体的协商值能被充分理解。

Proprietary Compression OUI 配置选项大致的格式如下所示：

类型 0x00	长度 1Byte	IEEE OUI 3Bytes	子类型 1Byte	值

长度域
值大于等于 6。

IEEE OUI
供应商所提供的结构中唯一的标识符，是由 IEEE 802 分配给供应商的以太网物理地址最重要的三个字节。它确定的选项是被指定的供应商私有的。

子类型
这个域明确到每个 OUI，并且为那个 OUI 指定了压缩类型。对这个域并没有进行标准化。每个 OUI 执行它们自己的值。

值
值域有零个或多个字节，并且包含供应商提供的压缩协议指定的附加数据。

（2）其它的压缩类型

这些配置选项提供一种协商使用已定义的公开压缩算法，其中包含了许多的压缩算法。各压缩类型配置选项的大致格式如下所示：

类型 1Byte	长度 1Byte	值

类型域
从 0x01 到 0xFD。
长度域
值大于等于 2。
值域
值域有零个或多个字节，并且包含由此压缩协议指定的附加数据。

下面具体介绍一种压缩协议:

BSD LZW Compress（BSD 压缩协议）

在 PPP 链路上传输的 BSD 压缩数据报的最大长度等于 PPP 包数据字段的最大长度。PPP 协议编号在 0x0000 到 0x3FFF 之间的包，除了 0x00FD 和 0x00FB 的编号之外都将被压缩，而其他的 PPP 包则一般不经压缩就被发送出去。而且控制包为了保证健壮性，它们将不被压缩。

• 配置选项格式

CCP BSD 压缩配置选项用于在链路上协商 BSD 压缩的使用。在默认或者最终未达成一致的情况下，是不使用压缩的。

下面是 BSD 压缩配置选项格式的概要：

类型	长度	版本	字典
0x15	0x03	001	5bits

字典域用 5bits 表示所用到的代码的最大长度。它的范围是 9 到 16。

请注意压缩数据接收端必须和发送端的编码长度一样。接收端用大一些的字典或者长一点的编码是不实际的，因为两边的字典要在同时清除，如果数据不是可压缩的以至于传输非压缩包，接收端也不能收到 LZW“清除”编码。如果收到的 Configure-Request 指定一个比当前选择小一些的字典，最好接受，而不是用 Configure-Nak 信号要求指定一个大些的字典。

以下是 BSD 压缩包格式的概要：

PPP 协议域	序列	数据
2Bytes	2Bytes	

PPP 协议字段

如果 BSD Compress 压缩协议和 PPP 压缩控制协议能很好地沟通，那么协议字段的值就是 0x00FD 或者 0x00FB。当两边认同协议字段压缩（Protocol-Field-Compression）时，这个值就可能被压缩。

序列

序列号当字典被清除之后它便从 0 开始，每当收到一个包，也包括非压缩包，就加 1。65535 之后的顺序号是 0。换句话说，序列号通常自动循环。

序列号可以保证丢失或者颠倒包的次序不会引起两边数据库不同步。可以在一个压缩包解压之前检查包的序列号。当出现了一个意想不到的序列号的时候，字典必须用 CCP 的 Reset-Request 包或者 Configure-Request 包来重新同步。

数据

压缩的 PPP 封装包，由协议段和原始数据段组成，后面紧跟的是非压缩包。不管 PPP 协议字段压缩是否协商，协议字段在这个包被压缩之前必须在原有包里压缩。因而，如果原有协议数字小于 0x100，那么就会被压缩到一个字节。

压缩数据的格式在 “BSD 压缩算法”里有精确的描述，在这里就不详述了。

下面介绍 BSD 压缩协议的几个特点：

• 填充数据

如果有填充数据被附加在 BSD 压缩包里，就需要在事前对自解释的填充数据配置选项（Self-Describing-Padding Configuration Option）进行交流协商。如果没有填充数据，那么自解释填充数据（Self-Describing-Padding）也就不需要了。

• 可靠性与序列

BSD 压缩算法要求包在传递的时候要按照顺序传递。用 Reset- Request 和 Reset-Ack 压缩控制包对压缩控制协议的重新配置，可以指示发送端与接收端在同步过程中的信息流失情况。如果帧序列检查探测到损坏了的数据包，普通机制将废弃这些包。包的丢失和包顺序错位是通过每个包内的序列号检查出来的。

在对包进行解压之前必须要检查包的序列号。当发现解压错误的情况时，接收端不是传输一组 Reset- Request 包，而是通过传输一组新的 CCP 的 Reset-Request 包，强制 CCP 短时间退出打开状态。但是，前者比后者的开销要小。当接收端第一次碰到意料之外的序列号，它应该发送一个在压缩控制协议里定义好的 CCP 的 Reset-Request 包。发送端发送 Reset-Ack 包和接受端收到一个 Reset-Ack 包时，就必须将序列号置成零值并清除压缩字典，然后接着发送和接受压缩数据包。在检测到错误之后，接收端必须丢弃所有的压缩包直到收到一个 Reset-Ack 包。

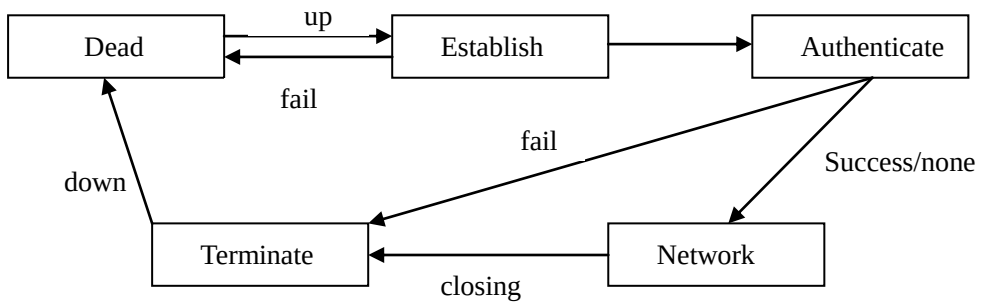
• 数据扩充

数据扩充即压缩后数据包反而变大了。当发现了数据扩充，PPP 包必须以未被压缩的格式发送出去。数据扩充量少于 3 个字节的包应该以未被压缩的格式发送出去，但是如果扩充结果没有超过链路的 MTU，也可以以压缩格式发送出去。如果接收的一个包的 PPP 协议号码在 0x0000 到 0x3FFF 之间(当然必须除掉 0xFD 和 0xFB),则可以假定这个包已经引起了扩充。如果压缩包比其原有格式还要大，那么，对于上级执行层来说，就有必要认为 PPP 链路应该有一个更小的 MTU，以保证压缩过的不可压缩包的大小不会大于协商好了的 PPP MTU 的大小。

第九章：PPP 链路

§9-1：PPP 链路的各阶段

在建立、维持和终止 PPP 链路的过程里，PPP 链路经过几个清楚的阶段，如框图所示：



PPP 链路阶段图

图 9.1

1．链路死亡阶段

链路一定开始并结束于这个阶段。当一个外部事件（例如载波侦听或网络管理员设定）指出物理层已经准备就绪时，PPP 将进入链路建立阶段。在这个阶段，LCP 自动机制将处于初始状态，向链路建立阶段的转换将给 LCP 自动机制一个 UP 事件信号。

2．链路建立阶段

LCP 用于交换配置信息包（Configure packets），建立连接。一旦一个配置成功信息包（Configure-Ack packet）被发送且被接收，就完成了交换，进入了 LCP 开启状态。所有的配置选项都假定使用默认值，除非被配置交换所改变。LCP 只配置不依赖于特别的网络层协议的配置选项。在网络层协议阶段，个别的网络层协议的配置由个别的网络控制协议（NCP）来处理。在这个阶段接收的任何非 LCP 包必须被静静的丢弃。收到 LCP Configure-Request 包能使链路从网络层协议阶段或者认证阶段返回到链路建立阶段。

3．认证阶段

在一些链路上，在允许网络层协议包交换之前，链路的一端可能需要对端去认证它。这并不是必须的。如果一次操作希望对端根据某一特定的认证协议来认证，那么它必须在链路建立阶段要求使用那个认证协议。应该尽可能在链路建立后立即进行认证。而且，链路质量检查可以同时发生。在一次操作中，禁止因为交换链路质量检查包而不确定地将认证向后推迟。在认证完成之前，禁止从认证阶段前进到网络层协议阶段。如果认证失败，认证者应该跃迁到链路终止阶段。

在这一阶段里，只有链路控制协议、认证协议，和链路质量监视协议的包是被允许的。在该阶段里接收到的其他的包必须被静静的丢弃。

一次操作中，仅仅是因为超时或者没有应答就造成认证的失败是不应该的。认证应该允许再次传输，只有在若干次的认证尝试失败以后，才不得已进入链路终止阶段。在操作中，哪一方拒绝了另一方的认证，哪一方就要负责开始链路终止阶段。

4．网络层协议阶段

一旦 PPP 完成了前面的阶段，每一个网络层协议（例如 IP，IPX，或 AppleTalk）必须被适当的网络控制协议（NCP）分别设定。每个 NCP 可以随时被打开和关闭。

因为一次操作最初可能需要浪费大量的时间用于链路质量检测，所以当等待对端设定 NCP 的时候，操作应该避免使用固定的延时。当一个 NCP 处于 Opened 状态时，PPP 将携带相应的网络层协议包。当相应的 NCP 不处于 Opened 状态时，接收到的网络层协议包，只有支持的协议才被静静的丢弃，任何不被支持的协议包必须在 Protocol-Reject 里返回。在这个阶段，链路通信量由 LCP，NCP，和网络层协议包的任意可能的联合组成。

5. 链路终止阶段

PPP 可以在任意时间终止链路。引起链路终止的原因很多：载波丢失、认证失败、链路质量失败、空闲周期定时器期满、或者管理员关闭链路。LCP 用交换 Terminate（终止）包的方法终止链路。当链路正被关闭时，PPP 通知网络层协议，以便他们可以采取正确的行动。交换 Terminate（终止）包之后，操作应该通知物理层断开，以便强制链路终止，尤其当认证失败时。Terminate-Request 包的发送者，在收到 Terminate-Ack 包后，或者在重启计数器期满后，应该断开连接；收到 Terminate-Request 的一方，应该等待对端去切断。在发出 Terminate-Request 后，至少也要经过一个重启时间，才允许断开。PPP 前进到链路死亡阶段。 在该阶段收到的任何非 LCP 包，必须被静静的丢弃。

LCP 关闭链路就足够了，不需要每一个 NCP 发送一个 Terminate 包。相反，一个 NCP 关闭却不足以引起 PPP 链路的终止，即使那个 NCP 是当前唯一一个处于 Opened 状态的 NCP。

§9-2：PPP 自动状态机制

要想对各个阶段的建立、终止有更深入的了解，就要研究 PPP 各阶段（如 LCP,NCP）的状态机制。而 PPP 各阶段的状态机制又是同一种机制，我们只要了解其中一种就可以了。以下就以 LCP 状态机制为例。

LCP 有限状态自动机由事件、动作和状态转换定义。事件包括接收外部命令（例如打开和关闭、重启定时器超时），和接收从对端来的包。动作包括启动重启定时器和向对端传输包。一些包类型--Configure-Nak、Configure-Rejects、Code-Rejects、Protocol-Reject、Echo-Request、Echo-Reply、Discard-Request--在自动机描述中不加以区分。从后文可知，这些包确实有着不同的功能。然而他们总是引起相同的转换。

事件	操作
Up = lower layer is Up	tlu = This-Layer-Up
Down = lower layer is Down	tld = This-Layer-Down
Open = administrative Open	tls = This-Layer-Started
Close= administrative Close	tlf = This-Layer-Finished
TO+ = Timeout with counter > 0	irc = Initialize-Restart-Count
TO- = Timeout with counter expired	zrc = Zero-Restart-Count
RCR+ = Receive-Configure-Request (Good)	scr = Send-Configure-Request
RCR- = Receive-Configure-Request (Bad)	

RCA = Receive-Configure-Ack	sca = Send-Configure-Ack
RCN = Receive-Configure-Nak/Rej	scn = Send-Configure-Nak/Rej
RTR = Receive-Terminate-Request	str = Send-Terminate-Request
RTA = Receive-Terminate-Ack	sta = Send-Terminate-Ack
RUC = Receive-Unknown-Code RXJ+ = Receive-Code-Reject (permitted) or Receive-Protocol-Reject RXJ- = Receive-Code-Reject (catastrophic) or Receive-Protocol-Reject	scj = Send-Code-Reject
RXR = Receive-Echo-Request or Receive-Echo-Reply or Receive-Discard-Request	ser = Send-Echo-Reply

全部的状态转换如下表。状态在水平轴，事件在垂直轴。状态转换和动作被表示成：动作/新状态的形式。多个动作用逗号分隔，无先后顺序。短划线（‘-’）代表无效的转换。

	State					
	0	1	2	3	4	5
Events	Initial	Starting	Closed	Stopped	Closing	Stopping
Up	2	irc,scr/6	-	-	-	-
Down	-	-	0	tls/1	0	1
Open	tls/1	1	irc,scr/6	3	5	5
Close	0	tlf/0	2	2	4	4
TO+	-	-	-	-	str/4	str/5
TO-	-	-	-	-	tlf/2	tlf/3
RCR+	-	-	sta/2	irc,scr,sca/8	4	5
RCR-	-	-	sta/2	irc,scr,scn/6	4	5
RCA	-	-	sta/2	sta/3	4	5
RCN	-	-	sta/2	sta/3	4	5
RTR	-	-	sta/2	sta/3	sta/4	sta/5
RTA	-	-	2	3	tlf/2	tlf/3
RUC	-	-	scj/2	scj/3	scj/4	scj/5
RXJ+	-	-	2	3	4	5
RXJ-	-	-	tlf/2	tlf/3	tlf/2	tlf/3
RXR	-	-	2	3	4	5

	State			
	6	7	8	9

Events	Req-Sent	Ack-Rcvd	Ack-Sent	Opened
Up	-	-	-	-
Down	1	1	1	tld/1
Open	6	7	8	9
Close	irc,str/4	irc,str/4	irc,str/4	tld,irc,str/4
TO+	scr/6	scr/6	scr/8	-
TO-	tlf/3	tlf/3	tlf/3	-
RCR+	sca/8	sca,tlu/9	sca/8	tld,scr,sca/8
RCR-	scn/6	scn/7	scn/6	tld,scr,scn/6
RCA	irc/7	scr/6	irc,tlu/9	tld,scr/6
RCN	irc,scr/6	scr/6	irc,scr/8	tld,scr/6
RTR	sta/6	sta/6	sta/6	tld,zrc,sta/5
RTA	6	6	8	tld,scr/6
RUC	scj/6	scj/7	scj/8	scj/9
RXJ+	6	6	8	9
RXJ-	tlf/3	tlf/3	tlf/3	tld,irc,str/5
RXR	6	7	8	ser/9

自动机里的状态转换和动作是由事件引起的。

具体说明以下动作：

This-Layer-Up（tlu）

该动作向上层表明已经进入 Opened 状态。典型的，该动作被 LCP 用于向 NCP，或者链路质量协议、认证协议发送 Up 事件信号。或者可以被一个 NCP 用于向网络层通知本层现在有效。

This-Layer-Down（tld）

该动作向上层表明已经离开 Opened 状态。典型地，该动作被 LCP 用于向一个 NCP、或认证协议、链路质量协议发送 Down 事件信号，或者可以被一个 NCP 用于向网络层通知本层已经不可再用。

This-Layer-Started 了（tls）

该动作向低层表明已经进入 Starting，并且低层对于链路是需要的。低层在有效时应该用一个 Up 事件响应。

This-Layer-Finished（tlf）

该动作向低层表明已经进入 Initial、Closed 或 Stopped 状态，并且低层对于链路不再需

要。低层在终止时应该用一个 Down 事件响应。典型地，该动作可以被 LCP 用于表示前进到链路死亡状态，或者可以被一个 NCP 用于向 LCP 指示没有其它的 NCP，链路将终止。

Initialize-Restart-Count（irc）

该动作对重启计数器设置适当的值（Max-Terminate 或 Max-Configure）。每次传输，包括第一次传输，计数器自减。另外，当重启计数器超时时重新设置计数器的值回初始值。

Zero-Restart-Count（zrc）

该动作对重启计数器清零。

具体说明以下事件：

Open：

该事件指出链路的通信量是可以管理的：即，网络管理者（人或程序）指出链路允许被 Opened。当这一事件发生，且链路不处于 Opened 状态时，自动机则试图给对端发送配置包。如果自动机不能开始配置（下层是 Down，或者前一个 Close 事件还没有结束），那么链路的建立将被自动的推迟。当收到一个 Terminate-Request，或者其它导致链路不可用的事件发生时，自动机将进入一个状态，在那里链路准备 re-open。无需额外的管理干涉。

经验表明：当用户想就链路进行重新谈判时，他们将额外的执行一条 Open 命令。这表明新的值将被协商。既然这不是 Open 事件的含义，那就暗示着在 Opened, Closing, Stopping 或 Stopped 状态，当执行一条 Open 用户命令时，先执行发生一个 Down 事件，紧接着一个 Up 事件。一定要注意不能有从另一个源发生的 Down 事件的干扰。紧接着 Down 事件的 Up 事件将引起一次有秩序的链路的再协商（通过先前进到的 Starting 状态，再进入到 Request-Sent 状态）。

Close：

该事件意味着链路没有通信量。即，网络管理者（人或程序）指示链路不允许被开放。当该事件发生且链路不处于 Closed 状态时，自动机试图终止连接。拒绝重新配置链路的尝试，直到一个新的 Open 事件发生。

当认证失败，链路应该被终止，以防止受到重复性的攻击和为其他用户服务。这可以通过模仿一个 Close 事件给 LCP，然后紧跟着一个 Open 事件来完成，既然链路在管理上是可被访问的。一定要注意不能有从另一个源发生的 Down 事件的干扰。紧接着 Open 事件的 Close 事件将引起一次有秩序的链路的终止（通过先前进到 Closing 状态，再进入到 Stopping 状态），This-Layer-Finished 动作能断开链路的连接。在 Stopped 或 Starting 状态，自动机等待下一次连接尝试。

Timeout (TO+,TO-)：

该事件表明重启定时器期满。重启定时器用于记录 Configure-Request 和 Terminate-Request 包的响应的时间。

TO+事件表明重启计数器持续大于零，它触发了相应的 Configure-Request 或 Terminate-Request 包的发送。

TO-事件表明重启计数器不大于零，不需要重新发送包了。

由状态转换表格可以知道：达到 Opened 状态，是 PPP 各阶段的目标。PPP 每个阶段只

有达到 Opened 状态，上层协议（阶段）才能开始工作（接到 UP 信号）。而如果下层协议要重新建立（接到 Down 信号），上层协议就必须终止。这也是在 NCP 阶段，如果接到 LCP 包，NCP 将终止并回到 LCP 阶段的原因。如果上层协议中断，却不足以关闭 PPP 链路，只有底层的关闭才可以实现 PPP 链路的终止。因此，关闭每一个 NCP 并不一定能关闭 PPP，只有 LCP 终止了，PPP 链路才被断开。

另外注意从 Req-sent 状态到 Opened 状态的过程。只有在链路两端都通过确认 Ack，才可能前进到 Opened 状态。如果在 Req-sent 状态时先收到了 Ack 包，触发 RCA 事件，进入到 Ack-Rcvd 状态，此时已经得到对端的确认，只需等待对端发来期望的 Request 包，就可以发送 Ack 包(本方的确认)并进入到 Opened 状态。如果在 Req-sent 状态时先收到了 Request 包，若不满意（RCR^-），就发送 Nak 包，原地等待，若满意（RCR^+）就发送 Ack 包前进到 Ack-Sent 状态（本方已确认），此时只要收到 Ack 包（对端确认）就可以进入到 Opened 状态。

§9-3：实例

以下是一段较完整的 PPP 协议运转过程的报文：

W:7E FF 03 C0 21 01 01 00 14 02 06 00 0A 00 00 05 06 48 43 D3 0A 07 02 08 02 2E 15 7E

C021:LCP Configure-Request:ID=1
ACCM:映射 ASCII17、19
Magic-Number:4843D30A
PFC
ACFC

写入配置请求；

R:7E FF 03 C0 21 02 01 00 14 02 06 00 0A 00 00 05 06 48 43 D3 0A 07 02 08 02 C5 7C 7E

C021:LCP Configure-Ack:ID=1
ACCM:映射 ASCII17、19
Magic-Number:4843D30A
PFC
ACFC

接收方同意请求；

R:7E FF 03 C0 21 01 03 00 1D 01 04 07 D0 02 06 00 0A 00 00 07 02
08 02 05 06 A7 A0 42 6F 03 05 C2 23 05 4B F2 7E

C021:LCP Configure-Request ID=3
MRU=07D0
ACCM:映射 ASCII17、19
PFC
ACFC

Magic-Number:A7A0426F
认证协议:CHAP MD5
接收方发送请求（有新的配置请求）；

W:7E FF 03 C0 21 03 03 00 08 03 04 C0 23 99 7F 7E

C021:LCP Configure-Nak:ID=3
认证协议：PAP
提醒接收方将认证协议改为 PAP;

R:7E FF 03 C0 21 01 05 00 1C 01 04 07 D0 02 06 00 0A 00 00 07 02 08 02 05 06 A7 A0 42 6F 03 04 C0 23 91 AB 7E

C021:LCP Configure-Request:ID=5
MRU=07D0
ACCM:映射 ASCII17、19
PFC
ACFC
Magic-Number:A7A0426F
认证协议:PAP
接收方重新发出配置请求（将认证协议改为 PAP）；

W:7E FF 03 C0 21 02 05 00 1C 01 04 07 D0 02 06 00 0A 00 00 07 02 08 02 05 06 A7 A0 42 6F 03 04 C0 23 A2 F1 7E

C021:LCP Configure-Ack:ID=5
MRU=07D0
ACCM:映射 ASCII17、19
PFC
ACFC
Magic-Number:A7A0426F
认证协议:PAP
同意配置请求；

W:7E FF 03 C0 21 09 00 00 08 48 43 D3 0A CF AE 7E

C021:LCP Echo-Request:ID=0
Magic-Number=48 43 D3 0A
发送回波请求，检测链路；

W:7E FF 03 C0 23 01 03 00 0F 03 48 4C 59 06 57 4F 52 4B 45 52 2E 87 7E

C023:PAP Request:ID=3

用户名 ID=48 4C 59

密码=57 4F 52 4B 45 52

发送用户名和密码，请求认证；

R:7E FF 03 C0 21 0A 00 00 08 A7 A0 42 6F 87 F0 7E

C021:LCP Echo-Reply:ID=0

Magic-Number=A7 A0 42 6F

接收端发送回波应答；

R:7E C0 23 02 03 00 05 00 8B 09 7E

C023:PAP Ack:ID=3

接收端通过验证；

W:7E FF 03 80 21 01 07 00 10 03 06 00 00 00 00 02 06 00 2D 0F 00 BE 0B 7E

8021:IPCP Configure-Request:ID=7

IP:00.00.00.00：请求对端分配

压缩 TCP;最大时间片标识 0F;时间片标识符不压缩

发送 IP 请求和压缩请求；

R:7E 80 21 04 07 00 0A 02 06 00 2D 0F 00 6E 85 7E

8021:IPCP Configure-Reject:ID=7

压缩 TCP;最大时间片标识 0F;时间片标识符不压缩

接收端不同意压缩；

W:7E FF 03 80 21 01 08 00 0A 03 06 00 00 00 00 24 1A 7E

8021:IPCP Configure-Request:ID=8

IP:00.00.00.00：请求对端分配

请求 IP 地址分配；

R:7E 80 21 01 01 00 0A 03 06 C0 A8 FE FE 48 CC 7E

8021:IPCP Configure-Request:ID=1

IP:C0.A8.FE.FE

W:7E FF 03 80 21 02 01 00 0A 03 06 C0 A8 FE FE 5F 56 7E

8021:IPCP Configure-Ack:ID=1

IP:C0.A8.FE.FE

W:7E FF 03 80 21 01 08 00 0A 03 06 00 00 00 00 24 1A 7E

8021:IPCP Configure-Request:ID=8
IP:00.00.00.00
继续发送 IP 请求；

R:7E 80 21 03 08 00 0A 03 06 0A 5F 15 DB E9 3A 7E

8021:IPCP Configure-Nak:ID=8
IP:0A.5F.15.DB
接收端建议采用 IP: 0A.5F.15.DB

W:7E FF 03 80 21 01 09 00 0A 03 06 0A 5F 15 DB 24 C1 7E

8021:IPCP Configure-Request:ID=9
IP:0A.5F.15.DB
采纳建议，发送 IP：0A.5F.15.DB 请求；

R:7E 80 21 02 09 00 0A 03 06 0A 5F 15 DB 33 5B 7E

8021:IPCP Configure-Ack:ID=9
IP:0A.5F.15.DB
接收端通过请求，分配 IP:0A.5F.15.DB；

……
TCP/IP 协议过程；

W:7E FF 03 C0 21 09 02 00 08 48 43 D3 0A 74 99 7E

C021:LCP Echo-Request:ID=2
Magic-Number=48 43 D3 0A
发送回波请求；

R:7E FF 03 C0 21 0A 02 00 08 A7 A0 42 6F 3C C7 7E

C021:LCP Echo-Request:ID=2
Magic-Number=A7 A0 42 6F
接收端回波应答；

W:7E FF 03 C0 21 05 02 00 10 55 73 65 72 65 71 75 65 73 74 53 33
7E

C021:LCP Terminate-Request:ID=2

发送链路中断请求；

R:7E FF 03 C0 21 06 02 00 10 55 73 65 72 65 71 75 65 73 74 72 A9 7E

C021:LCP Terminate-Ack:ID=2
接收端同意中断链路；

7E FF 03 C0 21 05 05 00 04 5C A4 7E

C021:LCP Terminate-Request:ID=5
接收端发出中断链路请求；

W:7E FF 03 C0 21 06 05 00 04 91 81 7E

C021:LCP Terminate-Ack:ID=5
同意中断链路。