# Ping项目文档说明与关键资料

## Ping原理

Ping的 本质是ICMP回送请求报文和回送应答报文。

Ping是检测网络连通性的常用工具，同时也能够收集其他相关信息。用户可以在Ping命令中指定不同参数，如ICMP报文长度、发送的ICMP报文个数、等待回复响应的超时时间等，设备根据配置的参数来构造并发送ICMP报文，进行Ping测试。

### 1.1 Ping常用的配置参数说明如下：

1. -a source-ip-address指定发送ICMP ECHO-REQUEST报文的源IP地址。如果不指定源IP地址，将采用出接口的IP地址作为ICMP ECHO-REQUEST报文发送的源地址。

2. -c count指定发送ICMP ECHO-REQUEST报文次数。

3. -t ttl 设置TTL(Time To Live)为指定的值。该字段指定IP包被路由器丢弃之前允许通过的最大网段数, 缺省值是255。

4. -i interval 设定间隔几秒发送一个ping包，默认一秒ping一次；

5. -v 使ping处于verbose方式，它要ping命令除了打印ECHO-RESPONSE数据包之外，还打印其它所有返回的ICMP数据包；若指定-v则显示所有接收ICMP消息

Ping命令的输出信息中包括目的地址、ICMP报文长度、序号、TTL值、以及往返时间。序号是包含在Echo回复消息（Type=0）中的可变参数字段，TTL和往返时间包含在消息的IP头中。

### ping操作问答

#### Ping中的报文往返时间是如何计算的？

ping程序在ICMP报文数据中存放发送请求时的时间值，在未来接收到该报文的回显时，用该时间减去回显报文中之前保存过的时间，得到的就是往返时间值了。

#### ping一个网址和ping一个IP地址有什么不同？

ping一个网址，我们的主机会首先通过DNS先去寻找网址对应的IP地址，再继续ping这个IP地址。ping一个网址一开始会有一点点延迟，是因为我们的主机需要寻找对应的IP地址，假如我们的DNS缓存中没有该网站。首个报文有所延迟还可能是因为ARP协议需要寻址

#### ping一个环回网址和ping主机的IP地址有什么不同？

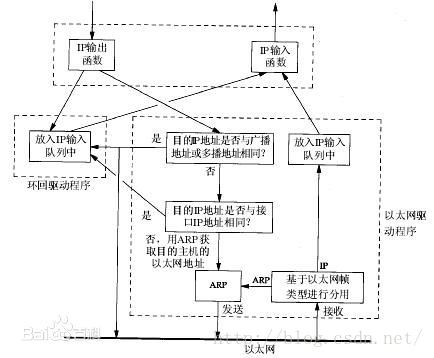


图 1‑1

考虑这么一个情况，主机的IP地址是192.168.1.230，而我们都知道127.0.0.1属于一个环回地址。ping这两个地址会有什么不同？

　　首先我们要了解一下，ICMP报文是在IP数据报内部的，ICMP需要借住IP的选路才能在互联网中工作。当ping程序向某个地址发送ICMP报文时，会先把ICMP报文打包成一个IP数据报，再通过IP数据报去进行转发。

如图1-1所示，ping 127.0.0.1的数据报在IP层中就会被放置到环回驱动程序中，随后就会被放回IP输入队列中，而192.168.1.230的报文会按照一般的IP地址数据下放到以太网驱动程序中去，并且由其中一段代码做判断，最后才回放到IP输入队列中。两个过程的区别在于，一个ping程序的数据会下放到以太网驱动程序，一个则直接回送。在网络层会把环回驱动程序和以太网驱动程序看作不同的链路层。

## ICMP协议要点

ICMP(Internet control message protocol, 网络控制信息协议)，它是一个工作在网络层(第三层)的协议，用于在IP主机、路由器之间传递控制消息。这些控制消息虽然不是用户数据，但是，对于保证用户数据的正确传输有着重要的作用，并且能够收集网络信息和诊断排错等。



图 2‑1

如图2-1所示，ICMP报文主要有两大功能：查询报文和差错报文。

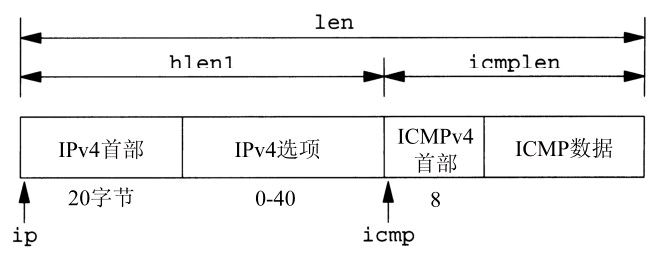
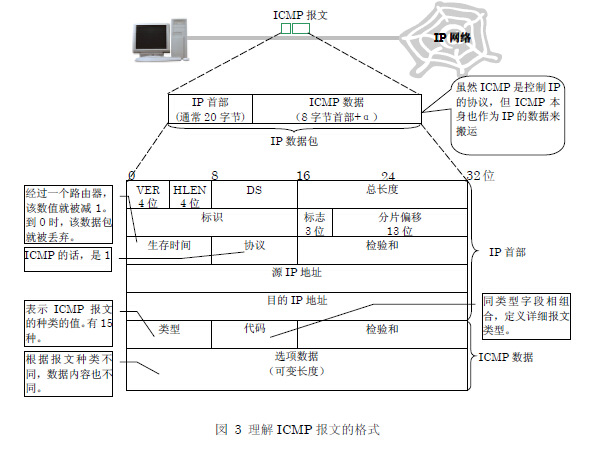


图 2‑2 处理ICMPv4应答涉及的首部、指针和长度

如图2-2，IP网络传送过程中，内容。



如图3，用来传送ICMP 报文的IP 数据包上实际上有不少字段。但是实际上与ICMP 协议相关的只有7 个子段。

1）协议；2）源IP 地址；3）目的IP 地址；4）生存时间；这四个包含在IP 首部的字段。

5）类型；6）代码；7）选项数据；这三个包含在ICMP数据部分的字段。

这里面，1)协议字段值是1。2)和3)是用来交流ICMP 报文的地址信息，没有特殊意义。对于理解ICMP 本身，重要的是5)，6)，7)三个字段。这里面的可以称为核心的重要字段是**5)类型，6)代码**这两个字段。所有ICMP 用来交流错误通知和信息询问的报文，都是由类型和代码的组合来表示的。RFC 定义了15种类型。“报文不可到达”这样的错误通知和“回送请求”这样的信息查询是由类型字段来区分的。ICMP报文由类型来表达它的大概意义，需要传递细小的信息时由代码来分类。进一步，需要向对方传送数据的时候，用7）选项数据字段来放置。

## ICMP实现ping命令

Ping命令用来在IP层次上调查与指定机器是否连通，调查数据包往复需要多少时间。为了实现这个功能，ping命令使用了两个ICMP报文。

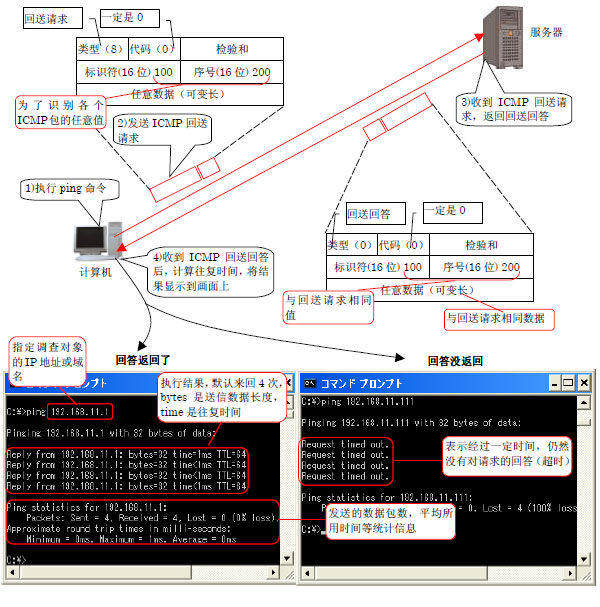


图 3-1

### 3.1 向目标服务器发送回送请求

首先，向目标服务器发送回送请求(类型是8，代码是0)报文。在这个回送请求报文里，除了类型和代码字段，还被追加了标识符和序号字段。标识符和序号字段分别是16位的字段。Ping命令在发送回送请求报文时，在这两个字段里填入任意的值。对于标识符，应用程序执行期间送出的所有报文里填入相同的值(一般为当前进程pid)。对于序号，每送出一个报文数值就增加1,。而且，回送请求的选项数据部分用来装任意数据。这个任意数据用来调整ping的数据交流包的大小。

### 3.2 鹦鹉学舌一样返回回送回答

计算机送出的回送请求到达目标服务器后，服务器回答这一请求，向送信方发送回送请求(类型是0，代码是0).这个ICMP回送回答报文在IP层来看，与被送来的回送请求报文基本上一样。不同的是，源和目标IP地址字段被交换了，类型字段里填入了表示会送回答的0.也就是说，从送信方来看，自己送出的ICMP报文从目标服务器那里像鹦鹉学舌般原样返回了。

送信方的计算机可以通过收到回送回答报文，来确认目标服务器在工作着。进一步，记住发送回送请求报文的时间，与接收到回送回答报文的时间一比较，就能计算出报文一去一回所需要的时间。但是，收到的回送回答报文里写的只是类型和代码的话，发送方计算机将无法判断它是否是自己发出去请求的回答。因此，前面说到的标识符和序号字段就有它的意义了。将这两个值与回送回答报文中的相同字段值进行比较，送信方计算机就能够简单地检测回送回答是否正确了。执行ping命令而调查的结果没有问题的话，就将目标服务器的IP地址、数据大小、往复花费的时间打印到屏幕上。

### 3.3 Ping命令不能确定与对方连通的原因

1. 目标服务器不存在；
2. 花在数据包交流上的时间太长，ping命令认为超时；
3. 目标服务器不回答ping命令。

如果是2，通过ping命令的选项来延长到超时的等待时间，就能正确显示结果。

如果是1或3，仅凭ping命令的结果就不能判断是哪一方的原因，所以，ping命令不一定能绝对判断对方是否存在。

## Ping 程序实现

Ping程序的操作非常简单，往某个IP地址发送一个ICMP回显请求，该节点则以一个ICMP回显应答。

构成Ping程序的各个函数及调用关系的概貌。

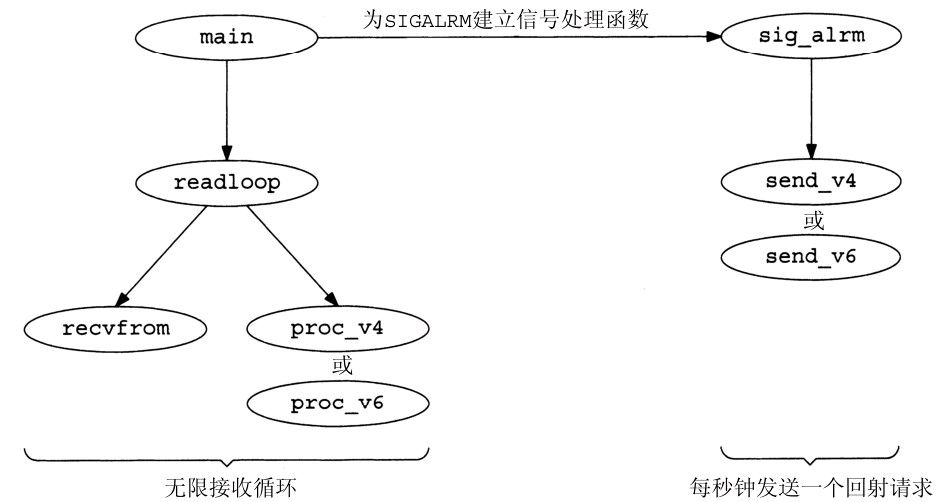


图 4‑1 Ping程序中各个函数的概貌

如图4-1所示，Ping程序分为两大部分，一部分在一个原始套接字上读入收到的每个分组，显示ICMP回显应答；另一部分每隔一秒钟发送一个ICMP回显请求。第二部分，由SIGALRM信号每秒钟驱动一次。

### 4.1 ping.h头文件

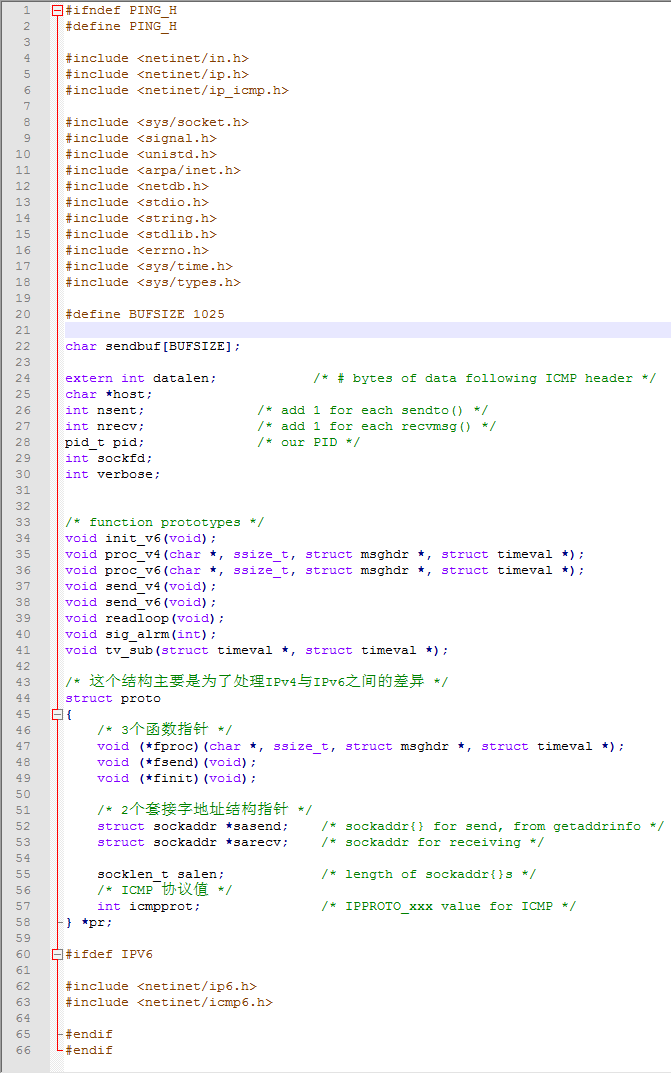


图 4‑2 ping.h头文件

如图4-2所示， 1~41包含基本的IPv4和ICMPv4头文件，定义一些全局变量以及各个函数的原型。44~58定义了proto结构处理IPv4与IPv6之间的差异，这个结构包含3个函数指针、2个套接字地址结构指针、2个套接字地址结构的大小以及ICMP的协议值。全局指针变量pr将指向为IPv4或IPV6初始化的某个proto结构。剩余的其他部分包含定义IPv6和ICMPv6结构和常值的2个头文件(RFC 3542[Stevens et al. 2003])。

### main函数



图 4‑4 main函数

如图4-3所示，4~10为IPv4和IPv6分别定义一个proto结构。其中套接字地址结构指针成员均初始化为空指针，因为暂时还不清楚最终使用的是IPv4还是IPv6 .

12行把随同回显请求发送的可选数据量设置为56个字节，由此产生84字节的IPv4数据报(包括20字节IPv4首部和8字节ICMP首部)，或104字节的IPv6数据报，随同某个回显请求发送的任何数据必须在对应的回显应答中返送回来。在这个数据区中的前8个字节存放本回显请求发送时刻的时间戳，然后在收到对应的回显应答之时，使用返送回来的时间戳计算并显示RTT。

25~36为本程序唯一支持的命令行选项-v，它可使我们显示接受到的大多数ICMP消息（只显示属于本ping进程的ICMP回显应答）。建立SIGALRM信号处理函数，将看到该信号一经启动将每秒钟产生一次，导致每秒钟发送ICMP回显请求。

38~81在命令行参数中必须有一个主机名或IP地址数串，通过调用host\_serv函数来处理它，返回的addrinfo结构中含有的协议族：或为AF\_INET，或为AF\_INET6。据此初始化全局指针变量pr，让它指向正确的proto结构。**注意**：调用IN6\_IS\_ADDR\_V4MAPPED确认由host\_serv返回的IPv6地址不是一个IPv4映射的IPv6地址（因为这样的地址尽管是一个IPv6地址，发送给其主机的却是IPv4分组，可以直接改用IPv4地址）。把已由getaddrinfo函数分配的套接字地址结构用于发送，并另行分配一个同样大小的套接字地址结构用于接收。

84行调用readloop函数执行处理。

### readloop函数



图 4‑5 readloop函数

12~13创建一个合适协议的原始套接字。调用setuid把进程的有效用户ID设置为实际用户ID，适用于本程序的可执行文件具有setuid到root的属性且以普通用户执行的情形。

14~15如果所用协议又一个初始化函数，那就调用它。IPv6协议初始化函数已经给出，可以调用。

16~17设置套接字接受缓冲区的大小。把套接字接受缓冲区大小设置成61440(60\*1024),比默认设置大，这样可以防备用户对IPv4广播地址或某个多播地址执行ping，两者均可能产生大量的应答。套接字接收缓冲区设置得越大，它发生溢出的可能性就越小。

18行调用SIGALRM信号处理函数发送第一个分组，该函数除发送一个分组外，还调度下一个SIGALRM信号在1秒钟之后产生。

19~24设置将传递给recvmsg的msghdr结构及iovec结构中的恒定成员。

25~37读入所有ICMP消息的无限循环。本程序的主循环，它读入返回到原始ICMP套接字的每个分组。调用gettimeofday记录分组收取时刻，然后调用合适的协议函数(proc\_v4或proc\_v6)处理包含在该分组中的ICMP消息。

## 5 其他

### 5.1 setlocale(LC\_ALL, “”)函数

C 标准库的 locale 设定函数是 setlocale()，功能是用来配置地域的信息，设置当前程序使用的本地化信息。若 locale 为零(NULL)，则不会改变地域化配置，返回当前的地域值，若系统尚未实作则返回 false。

配置地域化信息函数

#include <locale.h>

char \*setlocale (int category, const char \* locale);

category: LC\_ALL 包括下面的全部选项都要。 LC\_COLLATE 配置字符串比较，PHP 目前尚未实作出来本项。 LC\_CTYPE 配置字符类别及转换。例如全变大写 strtoupper()。 LC\_MONETARY 配置金融货币，PHP 目前尚未实作。 LC\_NUMERIC 配置小数点后的位数。 LC\_TIME 配置时间日期格式，与 strftime() 合用。 **而参数 locale 若是空字符串 ""，则会使用系统环境变量的 locale** 。若 locale 为零(NULL)，则不会改变地域化配置，返回当前的地域值，若系统尚未实作则返回 false。

### recvmsg和sendmsg函数

#include <sys/socket.h>

ssize\_t recvmsg(int sockfd, struct msghdr \*msg, int flags);

ssize\_t sendmsg(int sockfd, struct msghdr \*msg, int flags);

返回：读入或写出字节数——成功；-1——出错

struct msghdr {

void \*msg\_name; /\* protocol address \*/

socklen\_t msg\_namelen; /\* size of protocol address \*/

struct iovec \*msg\_iov; /\* scatter/gather array \*/

int msg\_iovlen; /\* # elements in msg\_iov \*/

void \*msg\_control; /\* ancillary data (cmsghdr struct) \*/

socklen\_t msg\_controllen; /\* length of ancillary data \*/

int msg\_flags; /\* flags returned by recvmsg() \*/

};

#include <sys/uio.h>

struct iovec {

void \*iov\_base; /\* starting address of buffer \*/

size\_t iov\_len; /\* size of buffer \*/

}