[概述 3](#_Toc506814705)

[链路层 4](#_Toc506814706)

[以太网帧 4](#_Toc506814707)

[PPP帧 5](#_Toc506814708)

[ARP请求 6](#_Toc506814709)

[网路层 7](#_Toc506814710)

[IP 7](#_Toc506814711)

[传输层 10](#_Toc506814712)

[UDP 10](#_Toc506814713)

[TCP 11](#_Toc506814714)

[IGMP 13](#_Toc506814715)

[ICMP网络不可达 14](#_Toc506814716)

[ICMP主机不可达 15](#_Toc506814717)

[ICMP协议不可达 16](#_Toc506814718)

[ICMP端口不可达 18](#_Toc506814719)

[ICMP源站选路失败 19](#_Toc506814720)

[ICMP目的网络不认识 20](#_Toc506814721)

[ICMP目的主机不认识 21](#_Toc506814722)

[ICMP源主机被隔离 22](#_Toc506814723)

[ICMP目的网络被强制禁止 23](#_Toc506814724)

[ICMP目的主机被强制禁止 24](#_Toc506814725)

[ICMP由于服务类型TOS，网络不可达 25](#_Toc506814726)

[ICMP由于服务类型TOS，主机不可达 26](#_Toc506814727)

[ICMP由于过滤，通信被强制禁止 27](#_Toc506814728)

[ICMP主机越权 28](#_Toc506814729)

[ICMP优先权中止生效 29](#_Toc506814730)

[ICMP源站抑制 30](#_Toc506814731)

[ICMP重定向差错 31](#_Toc506814732)

[ICMP网络重定向 33](#_Toc506814733)

[ICMP主机重定向 34](#_Toc506814734)

[ICMP服务类型和网络重定向 35](#_Toc506814735)

[ICMP服务类型和主机重定向 36](#_Toc506814736)

[ICMP请求回显 37](#_Toc506814737)

[ICMP路由器通告报文 38](#_Toc506814738)

[ICMP路由器请求报文 40](#_Toc506814739)

[ICMP参数问题 41](#_Toc506814740)

[ICMP时间戳请求 43](#_Toc506814741)

[ICMP时间戳应答 44](#_Toc506814742)

[ICMP信息请求（作废不用） 45](#_Toc506814743)

[ICMP信息应答（作废不用） 46](#_Toc506814744)

[ICMP地址掩码请求 47](#_Toc506814745)

[ICMP地址掩码应答 48](#_Toc506814746)

[应用层 53](#_Toc506814747)

[DNS请求报文 53](#_Toc506814748)

[DNS回复报文 55](#_Toc506814749)

[FTP命令报文 56](#_Toc506814750)

[FTP应答报文 57](#_Toc506814751)

[SMTP报文 58](#_Toc506814752)

[HTTP请求报文 59](#_Toc506814753)

[HTTP响应报文 60](#_Toc506814754)

[DHCP报文 62](#_Toc506814755)

[RIP请求报文 63](#_Toc506814756)

[RIP应答报文 64](#_Toc506814757)

[RIP轮询报文 65](#_Toc506814758)

[RIP轮询表项报文 66](#_Toc506814759)

[SNMP报文 67](#_Toc506814760)

[TELNET报文 68](#_Toc506814761)

# 概述

表示次序：左高右低

传输次序：big endian(高→低)

# 链路层

## 以太网帧

|  |  |  |  |
| --- | --- | --- | --- |
| 首部 | dst MAC | 6 |  |
| src MAC | 6 |  |
| 协议 | 2 | 08 00→IP， |
|  |  |  |
| IP报文 | 46~1500B | | |
| 尾部 | FCS | 4 | CRC检验序列 |
|  |  | | |
|  |  | | |

两个连续的标志字段→空帧

连续的两帧之间只需要用一个标志字段

转义：

扫描信息字段比特流，查看输入的数据，把其中的每一个

0x7E→0x7D5E

0x7E→0x7D5D

0x00~0x1F→0x7D20~0x7D3F

执行字节填充后，数据变大，但只是增加的传输量，接收方接收到数据后会执行相反的变换，恢复出原来的数据

零比特填充：

11111→111110

## PPP帧

|  |  |  |  |
| --- | --- | --- | --- |
| 首部 | 标志F | 1 | 7E |
| 地址A | 1 | FF |
| 控制C | 1 | 03 |
| 协议 | 2 | 00 21→IP， |
| IP报文 | 0~1500B | | |
| 尾部 | FCS | 2 | CRC检验序列 |
| 标志F | 1 | 7E |

## HDLC帧

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 首部 | 标志F | 1 | 7E | |
| 地址A | 1 | 非平衡方式：从站地址  平衡方式：应答站地址 | |
| 控制C | 1 | 信息帧I | 0xxx xxxx  传输数据信息，或使用捎带技术对数据进行确认 |
| 监督帧S | 10xx xxxx  用于流量控制和差错控制，执行对信息帧的确认、请求重发和请求暂停发送等 |
| 无编号帧U | 11xx xxxx  用于提供对链路的建立，拆除等多种控制功能 |
| 协议 | 2 | 08 00→IP， | |
| IP报文 | 0~1500B | | | |
| 尾部 | FCS | 2 | CRC检验序列 | |
| 标志F | 1 | 7E | |

透明传输：零比特填充

发送：数据中出现5个连续1时，立即在后面填入一个0，

接收：先找到F字段以确定帧的边界，接着再对比特流进行扫描，每当发现5个连续1，就将其后的一个0删除，以还原成原来的比特流。

## ARP请求

|  |  |  |  |
| --- | --- | --- | --- |
| 首部 | dst MAC | 6 |  |
| src MAC | 6 |  |
| 协议 | 2 | 08 00→IP，81 37→Novell IPX |
| 请求 | 硬件类型 | 2 |  |
| 协议类型 | 2 |  |
| 硬件地址长度 | 1 |  |
| 协议地址长度 | 1 |  |
| op | 2 |  |
| 发送端以太网地址 | 6 |  |
| 发送端IP地址 | 4 |  |
| 目的以太网地址 | 6 |  |
| 目的IP地址 | 4 |  |
| 尾部 | FCS | 4 | CRC检验序列 |

首部：

1~ 6，xx xx xx xx xx xx ：dst MAC，

7~12，xx xx xx xx xx xx：src MAC，

13~14：帧类型，08 00→IP、 、

请求：

15~16： ，

17~18

19，

20，

21~22，

23~28，

29~32，

33~38，

39~42，

尾部：

1~4：FCS(CRC)，

# 网路层

## IP

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 首部 | 固定  部分  20B | 版本 | 4bit | 4或6 | | |
| 首部长度 | 4bit | 范围5~F ，单位4B，得首部范围15B~60B | | |
| 应用层服  务类型TOS | 1 | 10→最小时延；08→最大吞吐量；  04→最小费用；02→最高可靠性；00→一般服务 | | |
| 报文长度 | 2 | 单位B | | |
| 标识 | 2 | 主机对发送的IP报文计数，溢出回零  每一份IP报文拥有唯一标识，分片→同标识 | | |
| 标志 | 3bit | 0 | 保留 | |
| DF | DF=0允许分片；  DF=1本报文不允许被分片 | |
| MF | MF=0后面没有分片，本报文是最后一片  MF=1后面还有分片 | |
| 片偏移 | 13bit | 本分片第一个字节在原分组数据部分位置，  起始0，单位8B | | |
| 生存时间  TTL | 1 | 本报文在网络中可通过的路由器个数的最大值单位hop，每经过一个路由器TTL减1，  收到TTL=0的IP报文：丢弃并向源主机报错 | | |
| 传输层协议 | 1 | 01→ICMP，02→IGMP，06→TCP，17→UDP | | |
| 检验和 | 2 | 发送主机计算 | | CheckSum=00 00  IP报文首部以2B分组，相加，取反  结果放入检验和字段 |
| 接收主机验证 | | IP报文首部以2B分组，相加，取反  结果FF FF→正确，否则错误(丢弃报错) |
| src IP | 4 |  | | |
| dst IP | 4 |  | | |
| 选项 | 每项都是4B的倍数，不足右补0  (1)安全和处理限制（用于军事领域，详细内容参见RFC 1108[Kent 1991]）  (2)记录路径（让每个路由器都记下它的IP地址，见7 . 3节）  (3)时间戳（让每个路由器都记下它的I P地址和时间，见7 . 4节）  (4)宽松的源站选路（为数据报指定一系列必须经过的I P地址，见8 . 5节）  (5)严格的源站选路（与宽松的源站选路类似，但是要求只能经过指定的这些地址，不能经过其他的地址）。 | | | | |
| 数据 | 可选项，TCP数据报  数据部分长度必须是8B的倍数  我们将在18章中看到在一个连接建立和一个连接终止时，双方交换的报文段 | | | | | |

TOS建议：

最小时延：Telnet、Rlogin、FTP控制、SMTP命令、DNS的UDP查询：

最大吞吐量：FTP数据、FTP任意块数据、SMTP数据阶段、DNS的区域传输

最小费用：用户网络新闻NNTP是唯一要求最小费用的应用

最高可靠性：ICMP的任何IGP、SNMP、

一般服务：DNS的TCP查询、ICMP差错、ICMP查询、BOOTP

# 传输层

## UDP

|  |  |  |  |
| --- | --- | --- | --- |
| 首部 | src port | 2 | 报文发送端进程端口号 |
| dst port | 2 | 报文接收端进程端口号 |
| 报文长度 | 2 | 单位B |
| 检验和 | 2 | CheckSum=00 00→发送端没有计算检验和。如果发送端没有计算检验和而接收端检测到检验和有差错，那么UDP数据报就要被悄悄地丢弃。不产生任何差错报文（当IP层检测到IP首部检验和有差错时也这样做） |
| 数据 | 应用层报文，可选项 | | | |

## TCP

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 固定  部分 | src port | 2 | 报文发送端进程端口号 | | | | |
| dst port | 2 | 报文接收端进程端口号 | | | | |
| seq | 4 | 发送端对发送的每一个数据字节进行计数，溢出回零  seq=本报文第一个数据字节的序号（有数据）  seq=将要发送的下一字节的序号（无数据）  两端都可作为发送端(全双工)，对各自发送的字节独立计数 | | | | |
| ack | 4 | 接收端对接收的每一个数据字节进行确认  ack=已成功收到的字节最大序号加+1  ack=期望收到的下一个数据字节的序号 | | | | |
| 首部长度 | 1 | 首部长度 | | | | 范围5~F ，单位4B，得首部范围15B~60B |
| 0 | | | | 保留 |
| tags | 1 | 00 | 保留 | | | |
| URG | URG=1：本报文数据段含有紧急数据，UP有效 | | | |
| ACK | ACK=1：ack有效 | | | |
| PSH | 收到PUSH报文，立即递交报文数据给服务进程而不等待 | | | |
| RST | 重建连接 | | | |
| SYN | SYN =1：本报文用于建立连接 | | | |
| FIN | FIN=1：本报文用于终止连接 | | | |
| rwnd | 2 | 本主机接收窗口大小，单位:B | | | | |
| 检验和 | 2 | 发送端  计算 | | | 左补12B伪首部：srcIP+dstIP+协议(06)+TCP报文长度  右补0填充至数据段为偶数字节  CheckSum=00 00  (TCP+数据段)以2B分组，相加，取反  结果放入检验和字段 | |
| 接收端  验证 | | | 左补12B伪首部：srcIP+dstIP+协议(06)+TCP报文长度  右补0填充至数据段为偶数字节  TCP报文以2B分组，相加，取反  结果=FFFF→正确 | |
| 紧急指针  up | 2 | 在数据报文中使用到，用于表示紧急数据报文  seq+up=紧急数据最后一个字节的序号 | | | | |
| 选项 | 选项1 | type | | | 02→MMS，03→Window Scale，04→SACK Permitted  08→时间戳， | | |
| 选项长度 | | | type+选项长度+内容 | | |
| 内容 | | | 有些选项有内容，有些则没有 | | |
| n\*NOP | | | 01 01…，每个选项占存都是4B的倍数，不足右补NOP | | |
| 选项2 | …… | | | | | |

|  |  |
| --- | --- |
| MMS | 02 04 [MMS,2B] ，接收一个报文包含的应用数据的最大字节数，两端建立连接时在SYN报文中通告各自的MSS，在不导致分片的前提下，MMS越大越好，以提高网络利用率以太网MSS值最大1460B，默认值-536B，BSD-1024B，目的地址非本地时使用默认值  MMS：最大报文段长度，相对TCP连接固定，发送缓存中的数据达到MMS个字节时，讲组装成一个TCP发送  TCP报文的总长度≤65535B  TCP报文的总长度≤建立连接时对方给出的MMS+首部长度 |
| Window Scale | 03 03 0x，x=0~E，接收窗口从16bit增加为(16+x)bit |
| SACK Permitted | 04 02 |
| TIME | 08 0A [时间戳,4B] [回显应答,4B] |

## IGMP

## ICMP网络不可达

void ICMP\_NetUnreachable() //网络不可达

{

/\*

1~2，03 00

3~4，检验和

\*/

}

## ICMP主机不可达

void ICMP\_HostUnreachable() //主机不可达

{

/\*

1~2，03 01

3~4，检验和

\*/

}

## ICMP协议不可达

void ICMP\_ProtocolUnreachable() //协议不可达

{

/\*

1~2，03 02

3~4，检验和

\*/

}

/\*

ICMP目的不可到达报文中的一种，以此来看一看ICMP差错报文中所附加的信息。使用UDP（见第11章）来查看它。

UDP的规则之一是，如果收到一份UDP数据报而目的端口与某个正在使用的进程不相符，那么UDP返回一个ICMP不可达报文。

一个ICMP端口不可达差错是立刻返回的

但是，TFTP客户程序看上去似乎忽略了这个ICMP报文，而在5秒钟之后又发送了另一份UDP数据报（第5行）。在客户程序放弃之前重发了三次。

包含在UDP首部中的内容是源端口号和目的端口号。就是由于目的端口号才导致产生了ICMP端口不可达的差错报文

接收ICMP的系统可以根据源端口号（2924）来把差错报文与某个特定的用户进程相关联（在本例中是TFTP客户程序）。

导致差错的数据报中的IP首部要被送回的原因是因为IP首部中包含了协议字段，使得ICMP可以知道如何解释后面的8个字节（在本例中是UDP首部）。

如果我们来查看TCP首部（图17-2）,可以发现源端口和目的端口被包含在TCP首部的前8个字节中。

当ICMP报文返回时，为什么TFTP客户程序还要继续重发请求呢？

这是由于网络编程中的一个因素，即BSD系统不把从插口(socket)接收到的ICMP报文中的UDP数据通知用户进程，除非该进程已经发送了一个connect命令给该插口。

标准的BSDTFTP客户程序并不发送connect命令，因此它永远也不会收到ICMP差错报文的通知。

这里需要注意的另一点是TFTP客户程序所采用的不太好的超时重传算法。

它只是假定5秒是足够的，因此每隔5秒就重传一次，总共需要25秒钟的时间。

在后面我们将看到TCP有一个较好的超时重发算法。TFTP客户程序所采用的超时重传算法已被RFC所禁用。

AIX3.2.2采用一种指数退避方法来设置超时值，分别在0、5、15和35秒时重发报文，这正是所推荐的方法。

最后需要指出的是，ICMP报文是在发送UDP数据报3.5ms后返回的，这与第7章我们所看到的Ping应答的往返时间差不多。

当路由器收到一份IP数据报文但又不能转发时，就要发送一份ICMP“主机不可达”差错报文。

可以很容易发现，在我们的网络上把接在路由器SUn上的拨号SLIP链路断开，然后试图通过该SLIP链路发送分组给任何指定SUn为默认路由器的主机。

较老版本的BSD产生一个主机不可达或者网络不可达差错，这取决于目的端是否

处于一个局域子网上。

4.4BSD只产生主机不可达差错。我们在上一节通过在路由器SUn上运行ETSTAt命令可以看到，当接通SLIP链路启动时就要在路由表中增加一项使用SLIP链路的表项，

而当断开SLIP链路时则删除该表项。这说明当SLIP链路断开时， SUn的路由表中就没有默认项了。

但是我们不想改变网络上其他主机的路由表，即同时删除它们的默认路由。相反，对于SUn不能转发的分组，

我们对它产生的ICMP主机不可达差错报文进行计数。在主机SVR4上运行PIg程序就可以看到这一点，它在拨号SLIP链路的另一端（拨号链路已被断开）：

当路由器SUn发现找不到能到达主机GEMIi的路由时，它就响应一个主机不可达的回显请求报文。

如果把SLIP链路接到ITEREt上，然后试图PIg一个与ITEREt没有连接的IP地址，那么应该会产生差错。

但令人感兴趣的是，我们可以看到在返回差错报文之前，分组要在ITEREt上传送多远：

从图8- 5可以看出，在发现该IP地址是无效的之前，该分组已通过了6个路由器。

只有当它到达SFET骨干网的边界时才检测到差错。这说明， 6个路由器之所以能转发分组是因为路由表中有默认项。

只有当分组到达SFET骨干网时，路由器才能知道每个连接到ITEREt上的每个网络的信息。这说明许多路由器只能在局部范围内工作。

参考文献[Ford, Rekhter, anDBrau1993]定义了顶层选路域（ top-leveLroutinGdomain），由它来维护大多数ITEREt网站的路由信息，而不使用默认路由。

他们指出，在ITEREt上存在5个这样的顶层选路域： SFET主干网、商业互联网交换（ CommerciaLInterneTExchange:CIX）、ASA科学互联网（ NASASciencEInternet: NSI）、SPRITLIk以及欧洲IP主干网（EBOE）。

\*/

## ICMP端口不可达

void ICMP\_PortUnreachable() //端口不可达

{

/\*

1~2，03 03

3~4，检验和

5~8，00 00 00 00

9~12：生成该差错报文的数据报IP首部(包括选项)+原始IP数据报中数据的前8字节

\*/

}

void ICMP\_Fragmentation\_needed\_but\_no\_D() //需要进行分片但设置了不分片比特

{

/\*

1~2，03 04

3~4，检验和

需要分片但设置了不分片比特，路径MTU发现机制（2.9节）却允许路由器把外出接口的MTU填在这个32bit字的低16bit中。我们在11.6节中给出了一个这种差错的例子。

\*/

}

## ICMP源站选路失败

void ICMP\_Source\_routing\_failed() //源站选路失败

{

/\*

1~2，03 05

3~4，检验和

\*/

}

## ICMP目的网络不认识

void ICMP\_DstNetUnknown() //目的网络不认识

{

/\*

1~2，03 06

3~4，检验和

\*/

}

## ICMP目的主机不认识

void ICMP\_DstHostUnknown() //目的主机不认识

{

/\*

1~2，03 07

3~4，检验和

\*/

}

## ICMP源主机被隔离

void ICMP\_SrcHostIsolated() //源主机被隔离（作废不用）

{

/\*

1~2，03 08

3~4，检验和

\*/

}

## ICMP目的网络被强制禁止

void ICMP\_DstNetProhibited() //目的网络被强制禁止

{

/\*

1~2，03 09

3~4，检验和

\*/

}

## ICMP目的主机被强制禁止

void ICMP\_DstHostProhibited() //目的主机被强制禁止

{

/\*

1~2，03 0A

3~4，检验和

\*/

}

## ICMP由于服务类型TOS，网络不可达

void ICMP\_NetUnreachableForTOS() //由于服务类型TOS，网络不可达

{

/\*

1~2，03 0B

3~4，检验和

\*/

}

## ICMP由于服务类型TOS，主机不可达

void ICMP\_HostUnreachableForTOS() //由于服务类型TOS，主机不可达

{

/\*

1~2，03 0C

3~4，检验和

\*/

}

## ICMP由于过滤，通信被强制禁止

void ICMP\_CommunicationProhibitedForFiltering() //由于过滤，通信被强制禁止

{

/\*

1~2，03 0D

3~4，检验和

\*/

}

## ICMP主机越权

void ICMP\_HostPrecedenceViolation() //主机越权

{

/\*

1~2，03 0E

3~4，检验和

\*/

}

## ICMP优先权中止生效

void ICMP\_PrecedenceCutoffEffective() //优先权中止生效

{

/\*

1~2，03 0F

3~4，检验和

\*/

}

## ICMP源站抑制

void ICMP\_SourceQuench() //源站抑制

{

/\*

1~2，04 00

3~4，检验和

\*/

}

/\*

## ICMP重定向差错

当IP数据报文应该被发送到另一个路由器时，收到数据报文的路由器就要发送ICMP重定向差错报文给IP数据报文的发送端。

这在概念上是很简单的，正如图9- 3所示的那样。只有当主机可以选择路由器发送分组的情况下，我们才可能看到ICMP重定向报文（回忆我们在图7- 6中看过的例子）。

1) 我们假定主机发送一份IP数据报文给R1。这种选路决策经常发生，因为R1是该主机的默认路由。

2) R1收到数据报文并且检查它的路由表，发现R2是发送该数据报文的下一站。

当它把数据报文发送给R2时，R1检测到它正在发送的接口与数据报文到达接口是相同的（即主机和两个路由器所在的LAN）。

这样就给路由器发送重定向报文给原始发送端提供了线索。

3) R1发送一份ICMP重定向报文给主机，告诉它以后把数据报文发送给R2而不是R1。

图9-3ICMP重定向的例子

重定向一般用来让具有很少选路信息的主机逐渐建立更完善的路由表。

主机启动时路由表中可以只有一个默认表项（在图9- 3所示的例子中，为R1或R2）。

一旦默认路由发生差错，默认路由器将通知它进行重定向，并允许主机对路由表作相应的改动。

ICMP重定向允许TCP/ IP主机在进行选路时不需要具备智能特性，而把所有的智能特性放在路由器端。

显然，在我们的例子中，R1和R2必须知道有关相连网络的更多拓扑结构的信息，但是连在LAN上的所有主机在启动时只需一个默认路由，通过接收重定向报文来逐步学习。

ICMP重定向报文的接收者必须查看三个IP地址：

( 1)导致重定向的IP地址（即ICMP重定向报文的数据位于IP数据报文的首部）；

( 2)发送重定向报文的路由器的IP地址（包含重定向信息的IP数据报文中的源地址；

( 3)应该采用的路由器IP地址（在ICMP报文中的4~ 7字节）。

关于ICMP重定向报文有很多规则。首先，重定向报文只能由路由器生成，而不能由主机生成。

另外，重定向报文是为主机而不是为路由器使用的。假定路由器和其他一些路由器共同参与某一种选路协议，则该协议就能消除重定向的需要

（这意味着在图9- 1中的路由表应该消除或者能被选路守护程序修改，或者能被重定向报文修改，但不能同时被二者修改）。

在4. 4BSD系统中，当主机作为路由器使用时，要进行下列检查。在生成ICMP重定向报文之前这些条件都要满足。

1) 出接口必须等于入接口。

2) 用于向外传送数据报文的路由不能被ICMP重定向报文创建或修改过，而且不能是路由器的默认路由。

3) 数据报文不能用源站选路来转发。

4) 内核必须配置成可以发送重定向报文。

内核变量取名为IPSEDREDIRECTs或其他类似的名字（参见附录E）。

大多数当前的系统（例如BSD、SunOS4.1.x、SolariS2.X及AIX3.2.2）在默认条件下都设置该

变量，使系统可以发送重定向报文。其他系统如SVR4则关闭了该项功能。

另外，一台4. 4BSD主机收到ICMP重定向报文后，在修改路由表之前要作一些检查。这是

为了防止路由器或主机的误操作，以及恶意用户的破坏，导致错误地修改系统路由表。

1) 新的路由器必须直接与网络相连接。

2) 重定向报文必须来自当前到目的地所选择的路由器。

3) 重定向报文不能让主机本身作为路由器。

4) 被修改的路由必须是一个间接路由。

关于重定向最后要指出的是，路由器应该发送的只是对主机的重定向（代码1或3，如图

9- 5所示），而不是对网络的重定向。子网的存在使得难于准确指明何时应发送对网络的重定

向而不是对主机的重定向。只当路由器发送了错误的类型时，一些主机才把收到的对网络的

重定向当作对主机的重定向来处理。

\*/

## ICMP网络重定向

void ICMP\_RedirectNet() //网络重定向

{

/\*

1~2，05 00

3~4，检验和

5~8，应该使用的路由器IP地址

IP首部(包括选项)+原始IP数据报中的数据前8字节

\*/

}

## ICMP主机重定向

void ICMP\_RedirectHost() //主机重定向

{

/\*

1~2，05 01

3~4，检验和

5~8，应该使用的路由器IP地址

IP首部(包括选项)+原始IP数据报中的数据前8字节

\*/

}

## ICMP服务类型和网络重定向

void ICMP\_Redirect() //服务类型和网络重定向

{

/\*

1~2，05 10→服务类型和网络重定向

3~4，检验和

5~8，应该使用的路由器IP地址

IP首部(包括选项)+原始IP数据报中的数据前8字节

\*/

}

## ICMP服务类型和主机重定向

void ICMP\_Redirect() //服务类型和主机重定向

{

/\*

1~2，05 11→服务类型和主机重定向

3~4，检验和

5~8，应该使用的路由器IP地址

IP首部(包括选项)+原始IP数据报中的数据前8字节

\*/

}

## ICMP请求回显

void ICMP\_Echo() //请求回显

{

/\*

1~2，08 00

3~4，检验和

\*/

}

## ICMP路由器通告报文

void ICMP\_Announce() //路由器通告报文

{

/\*

1~2，09 00

3~4，检验和

5，地址数

6，地址项长度

7~8，生存时间

9~12，路由器地址1

13~16，优先级1

17~20，路由器地址2

21~24，优先级2

……

\*/

}

/\*

在本章前面已提到过一种初始化路由表的方法，即在配置文件中指定静态路由。这种方

法经常用来设置默认路由。另一种新的方法是利用ICMP路由器通告和请求报文。

一般认为，主机在引导以后要广播或多播传送一份路由器请求报文。一台或更多台路由

器响应一份路由器通告报文。另外，路由器定期地广播或多播传送它们的路由器通告报文，

允许每个正在监听的主机相应地更新它们的路由表。

RFC1256[DeerinG1991]确定了这两种ICMP报文的格式。ICMP路由器请求报文的格式如

图9- 6所示。ICMP路由器通告报文的格式如图9- 7所示。

路由器在一份报文中可以通告多个地址。地址数指的是报文中所含的地址数。地址项大

小指的是每个路由器地址32bit字的数目，始终为2。生存期指的是通告地址有效的时间（秒

数）。

接下来是一对或多对IP地址和优先级。IP地址必须是发送路由器的某个地址。优先级是

一个有符号的32bit整数，指出该IP地址作为默认路由器地址的优先等级，这是与子网上的其

他路由器相比较而言的。值越大说明优先级越高。优先级为0X80000000说明对应的地址不能

作为默认路由器地址使用，尽管它也包含中通告报文中。优先级的默认值一般为0。

9.6.1路由器操作

当路由器启动时，它定期在所有广播或多播传送接口上发送通告报文。准确地说，这些

通告报文不是定期发送的，而是随机传送的，以减小与子网上其他路由器发生冲突的概率。

一般每两次通告间隔450秒和600秒。一份给定的通告报文默认生命周期是30分钟。

使用生命周期域的另一个时机是当路由器上的某个接口被关闭时。在这种情况下，路由

器可以在该接口上发送最后一份通告报文，并把生命周期值设为0。

除了定期发送主动提供的通告报文以外，路由器还要监听来自主机的请求报文，并发送

路由器通告报文以响应这些请求报文。

如果子网上有多台路由器，由系统管理员为每个路由器设置优先等级。例如，主默认路

由器就要比备份路由器具有更高的优先级。

9.6.2主机操作

主机在引导期间一般发送三份路由器请求报文，每三秒钟发送一次。一旦接收到一个有

效的通告报文，就停止发送请求报文。

主机也监听来自相邻路由器的请求报文。这些通告报文可以改变主机的默认路由器。另

外，如果没有接收到来自当前默认路由器的通告报文，那么默认路由器会超时。

只要有一般的默认路由器，该路由器就会每隔10分钟发送通告报文，报文的生命周期是

30分钟。这说明主机的默认表项是不会超时的，即使错过一份或两份通告报文。

9.6.3实现

路由器发现报文一般由用户进程（守护程序）创建和处理。这样，在图9- 1中就有另一个修改路由表的程序，尽管它只增加或删除默认表项。

守护程序必须把它配置成一台路由器或

主机来使用。

这两种ICMP报文是新加的，不是所有的系统都支持它们。在我们的网络中，只有

SolariS2.x支持这两种报文（I. RDISc守护程序）。尽管RFC建议尽可能用IP多播传送，

但是路由器发现还可以利用广播报文来实现。

\*/

## ICMP路由器请求报文

void ICMP\_Request() //路由器请求报文

{

/\*

1~2，0A 00

3~4，检验和

\*/

}

void ICMP\_TTLExhaustInCommunication() //传输期间生存时间为0

{

/\*

1~2，0B 00

3~4，检验和

\*/

}

void ICMP\_TTLExhaustInDatagramAssem() //在数据报组装期间生存时间为0

{

/\*

1~2，0B 01

3~4，检验和

\*/

}

## ICMP参数问题

void ICMP\_ParameterProblem() //参数问题

{

/\*

1~2，0C 00

3~4，检验和

\*/

}

/\*

ICMP时间戳请求：允许系统向另一个系统查询当前的时间。

返回的建议值是自午夜开始计算的毫秒数，协调的统一时间(UTC)，调用者必须通过其他方法获知当时的日期。

由于时间戳的值是自午夜开始计算的毫秒数，即UTC，因此它们的值始终小于86400000(24×60×60×1000)。

往返时间rtt=它收到应答时的时间值-发送请求时的时间值

difference=接收时间戳值-发起时间戳值

如果我们相信RTT的值，并且相信RTT的一半用于请求报文的传输，另一半用于应答报文的传输，

那么为了使本机时钟与查询主机的时钟一致，本机时钟需要进行调整，调整值是difference减去RTT的一半。

它表明，当系统返回一个非标准时间戳值时（不是自午夜开始计算的毫秒数，UTC），它就用32bit时间戳中的高位来表示。

不能计算发起时间戳和接收时间戳之间的时间差，因为它们的单位不一致。

另一种方法

1)在1.12节中描述了日期服务程序和时间服务程序。前者是以人们可读的格式返回当前的时间和日期，是一行ASCII字符。可以用telnet命令来验证这个服务：

另一方面，时间服务程序返回的是一个32bit的二制进数值，表示自UTC，1900年1月1日午夜起算的秒数。

这个程序是以秒为单位提供的日期和时间（前面我们提过的rdate命令使用的是TCP时间服务程序）。

2)严格的计时器使用网络时间协议（NTP），该协议在RFC1305中给出了描述[Mills1992]。这个协议采用先进的技术来保证LAN或WAN上的一组系统的时钟误差在毫秒级以内。对计算机精确时间感兴趣的读者应该阅读这份RFC文档。

3)开放软件基金会（OSF）的分布式计算环境（DCE）定义了分布式时间服务（DTS），它也提供计算机之间的时钟同步。文献[Rosenberg,KenneyandFisher1992]提供了该服务的其他细节描述。

4)伯克利大学的Unix系统提供守护程序timed(8)，来同步局域网上的系统时钟。不像NTP和DTS，timed不在广域网范围内工作

\*/

## ICMP时间戳请求

void ICMP\_TimestampRequest() //时间戳请求

{

/\*

1~2，0D 00

3~4，检验和

5~6，标识符：由发送端任意设定，在应答中被返回，发送端就可以把应答与请求进行匹配

7~8，序列号：由发送端任意设定，在应答中被返回，发送端就可以把应答与请求进行匹配

9~12， orig：发起时间戳：

13~16， recv：接收时间戳：

17~20， xmit：发送时间戳：用于让发送方分别计算发送请求的时间和发送应答的时间，

\*/

}

## ICMP时间戳应答

void ICMP\_TimestampReply() //时间戳应答

{

/\*

1~2，0E 00

3~4，检验和

5~6，标识符：由发送端任意设定，在应答中被返回，发送端就可以把应答与请求进行匹配

7~8，序列号：由发送端任意设定，在应答中被返回，发送端就可以把应答与请求进行匹配

9~12， orig：发起时间戳：

13~16， recv：接收时间戳：

17~20， xmit：发送时间戳，xmit=recv

\*/

}

## ICMP信息请求（作废不用）

void ICMP\_InformationRequest() //信息请求（作废不用）

{

/\*

1~2，0F 00

3~4，检验和

\*/

}

## ICMP信息应答（作废不用）

void ICMP\_InformationReply() //信息应答（作废不用）

{

/\*

1~2，10 00

3~4，检验和

\*/

}

/\*

ICMP地址掩码请求：

用于无盘系统在引导过程中获取自己的子网掩码

一般发往广播地址

ICMP地址掩码应答

一般应答地址必须是单播地址，除非请求端的源IP地址是0.0.0.0。

除非系统是地址掩码的授权代理，否则它不能发送地址掩码应答（为了成为授权代理，它必须进行特殊配置，以发送这些应答。参见附录E）。

但是，大多数主机在收到请求时都发送一个应答，甚至有一些主机还发送差错的应答。

ICMP地址掩码应答必须是收到请求接口的子网掩码（这是因为多接口主机每个接口有不同的子网掩码）

\*/

## ICMP地址掩码请求

void ICMP\_AddressMaskRequest() //地址掩码请求

{

/\*

1~2，11 00

3~4，检验和

5~6，标识符：由发送端任意设定，在应答中被返回，发送端就可以把应答与请求进行匹配

7~8，序列号：由发送端任意设定，在应答中被返回，发送端就可以把应答与请求进行匹配

9~12，子网掩码：

\*/

}

## ICMP地址掩码应答

void ICMP\_AddressMaskReply() //地址掩码应答

{

/\*

1~2，12 00

3~4，检验和

5~6，标识符：由发送端任意设定，在应答中被返回，发送端就可以把应答与请求进行匹配

7~8，序列号：由发送端任意设定，在应答中被返回，发送端就可以把应答与请求进行匹配

9~12，子网掩码

\*/

}

/\*

ICMP

终点不可达

源点抑制

时间超过

参数问题

改变路由

\*/

/\*

重定向报文

/\*

ICMP重定向报文的接收者必须查看三个IP地址：

(1)导致重定向的IP地址（即ICMP重定向报文的数据位于IP数据报的首部）；

(2)发送重定向报文的路由器的IP地址（包含重定向信息的IP数据报中的源地址；

(3)应该采用的路由器IP地址（在ICMP报文中的4~7字节）。

1.重定向报文只能由路由器生成，不能由主机生成。

2.重定向报文是为主机而不是为路由器使用的。假定路由器和其他一些路由器共同参与某一种选路协议，则该协议就能消除重定向的需要（这意味着在图9-1中的路由表应该消除或者能被选路守护程序修改，或者能被重定向报文修改，但不能同时被二者修改）。

3.出接口必须等于入接口。

4.用于向外传送数据报的路由不能被ICMP重定向报文创建或修改过，而且不能是路由器的默认路由。

5.数据报不能用源站选路来转发。

6.内核必须配置成可以发送重定向报文。

内核变量取名为ipsendredirects或其他类似的名字（参见附录E）。大多数系统在默认条件下都设置该变量，使系统可以发送重定向报文。

另外，一台4.4BSD主机收到ICMP重定向报文后，在修改路由表之前要作一些检查。为防止路由器或主机的误操作，以及恶意用户的破坏，导致错误地修改系统路由表。

1)新的路由器必须直接与网络相连接。

2)重定向报文必须来自当前到目的地所选择的路由器。

3)重定向报文不能让主机本身作为路由器。

4)被修改的路由必须是一个间接路由。

路由器应该发送的只是对主机的重定向（代码1或3，），而不是对网络的重定向。子网的存在使得难于准确指明何时应发送对网络的重定向而不是对主机的重定向。

只当路由器发送了错误的类型时，一些主机才把收到的对网络的重定向当作对主机的重定向来处理。

发生ICMP不可达差错的另一种情况是，当路由器收到一份需要分片的数据报，而在IP首

部又设置了不分片（DF）的标志比特。如果某个程序需要判断到达目的端的路途中最小MTU

是多少—称作路径MTU发现机制（2.9节），那么这个差错就可以被该程序使用。

这种情况下的ICMP不可达差错报文格式如图11-9所示。这里的格式与图6-10不同，因为

在第2个32bit字中，16~31bit可以提供下一站的MTU，而不再是0。如果路由器没有提供这种新的ICMP差错报文格式，那么下一站的MTU就设为0。

新版的路由器需求RFC[Almquist1993]声明，在发生这种ICMP不可达差错时，路

由器必须生成这种新格式的报文。

例子

关于分片作者曾经遇到过一个问题，ICMP差错试图判断从路由器netb到主机sun之间的

拨号SLIP链路的MTU。我们知道从sun到netb的链路的MTU：当SLIP被安装到主机sun时，

这是SLIP配置过程中的一部分，加上在3.9节中已经通过netstat命令观察过。现在，我们想

从另一个方向来判断它的MTU（在第25章，将讨论如何用SNMP来判断）。在点到点的链路中，

不要求两个方向的MTU为相同值。

所采用的技术是在主机solaris上运行ping程序到主机bsdi，增加数据分组长度，直

到看见进入的分组被分片为止。如图11-10所示。在主机sun上运行tcpdump，观察SLIP链路，看什么时候发生分片。开始没有观察到分

片，一切都很正常直到ping分组的数据长度从500增加到600字节。可以看到接收到的回显请求（仍然没有分片），但不见回显应答。

为了跟踪下去，也在主机bsdi上运行tcpdump，观察它接收和发送的报文。输出如图

11-11所示。首先，每行中的标记（DF）说明在IP首部中设置了不分片比特。这意味着Solaris2.2一

般把不分片比特置1，作为实现路径MTU发现机制的一部分。

第1行显示的是回显请求通过路由器netb到达sun主机，没有进行分片，并设置了DF比

特，因此我们知道还没有达到netb的SLIPMTU。

接下来，在第2行注意到DF标志被复制到回显应答报文中。这就带来了问题。回显应答与

回显请求报文长度相同（超过600字节），但是sun外出的SLIP接口MTU为552。因此回显应

答需要进行分片，但是DF标志比特又被设置了。这样，sun就产生一个ICMP不可达差错报文

返回给bsdi（报文在bsdi处被丢弃）。

这就是我们在主机solaris上没有看到任何回显应答的原因。这些应答永远不能通过

sun。分组的路径如图11-12所示。最后，在图11-11中的第3行和第6行中，mtu=0表示主机sun没有在ICMP不可达报文中返

回出口MTU值，如图11-9所示（在25.9节中，将重新回到这个问题，用SNMP判断netb上的

SLIP接口MTU值为1500）。

我们同样也可以使用UDP产生ICMP“源站抑制(sourcequench)”差错。当一个系统（路

由器或主机）接收数据报的速度比其处理速度快时，可能产生这个差错。注意限定词“可能”。

即使一个系统已经没有缓存并丢弃数据报，也不要求它一定要发送源站抑制报文。

图11-18给出了ICMP源站抑制差错报文的格式。有一个很好的方案可以在我们的测试网

络里产生该差错报文。可以从bsdi通过必须经过拨号SLIP链路的以太网，将数据报发送给路

由器sun。由于SLIP链路的速度大约只有以太网的千分之一，因此，我们很容易就可以使其缓

存用完。下面的命令行从主机bsdi通过路由器sun发送100个1024字节长数据报给solaris。

我们将数据报发送给标准的丢弃服务，这样，这些数据报将被忽略：

bsdi%sock-u-i-w1024-n100solarisdiscard

图11-19给出了与此命令行相对应的tcpdump输出结果

在这个输出结果中，删除了很多行，这只是一个模型。接收前26个数据报时未发生差错；我们只给出了第一个数据报的结果。然而，从第27个数据报开始，每发送一份数据报，

就会接收到一份源站抑制差错报文。总共有26+（74×2）=174行输出结果。从2.10节的并行线吞吐率计算结果可以知道，以9600b/s速率传送1024字节数据报只需要

1秒时间（由于从sun到netb的SLIP链路的MTU为552字节，因此在我们的例子中，20+8+

1024字节数据报将进行分片，因此，其时间会稍长一些）。但是我们可以从图11-19的时间中

看出，sun路由器在不到1秒时间内就处理完所有的100个数据报，而这时，第一份数据报还

未通过SLIP链路。因此我们用完其缓存就不足不奇了。

尽管RFC1009[BradenandPostel1987]要求路由器在没有缓存时产生源站抑制差

错报文，但是新的RouterRequirementsRFC[Almquist1993]对此作了修改，提出路由

器不应该产生源站抑制差错报文。由于源站抑制要消耗网络带宽，且对于拥塞来说是

一种无效而不公平的调整，因此现在人们对于源站抑制差错的态度是不支持的。

在本例中，还需要指出的是，sock程序要么没有接收到源站抑制差错报文，要么接收到

却将它们忽略了。结果是如果采用UDP协议，那么BSD实现通常忽略其接收到的源站抑制报

文（正如我们在21.10节所讨论的那样，TCP接受源站抑制差错报文，并将放慢在该连接上的

数据传输速度）。其部分原因在于，在接收到源站抑制差错报文时，导致源站抑制的进程可能

已经中止了。实际上，如果使用Unix的time程序来测定sock程序所运行的时间，其结果是它

只运行了大约0.5秒时间。但是从图11-19中可以看到，在发送第一份数据报过后0.71秒才接收

到一些源站抑制，而此时该进程已经中止。其原因是我们的程序写入了100个数据报然后中止

了。但是所有的100个数据报都已发送出去—有一些数据报在输出队列中。

这个例子重申了UDP是一个非可靠的协议，它说明了端到端的流量控制。尽管sock程序

成功地将100个数据报写入其网络，但只有26个数据报真正发送到了目的端。其他74个数据报可能被中间路由器丢弃。除非在应用程序中建立一些应答机制，否则发送端并不知道接收端

是否收到了这些数据

\*/

# 应用层

## DNS请求报文

|  |  |  |  |
| --- | --- | --- | --- |
| 首部 | 1~2 | ID：由客户端设置，用于确定query与response是否匹配 | |
| 3~4 | QR | 0→query，1→response |
| opcode(4) | 0000：标准查询，  0001：反向查询，  0002：服务器状态请求 |
| AA | 1→ 授权回答，该名字服务器是授权于该域的 |
| TC | 1→ 可截断回复，表示回复报文总长度超过512B，只返回前512B |
| RD | 1→ 递归查询，0→迭代查询 |
| RA | 1→ 递归回复，0→迭代回复 |
| 000 | 保留 |
| rcode(4) | 0000：无差错  0001：式差错  0010：题在域名服务器上  0011：参照问题  0100：询类型不支持  0101：管理上被禁止 |
| 5~6 | Question Count  查询报文问题数通常是1，下面3项则均为0；应答报文回答数至少是1，剩下的两项可以是0或非0 | |
| 7~8 | Answer RR Count | |
| 9~10 | Authority RR Count | |
| 11~12 | Additional RR Count | |
| Question | 域名 |  | |
| type |  | |
| class |  | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Answer  RR | 域名 | 该RR中的数据对应的名字 | | |
| 无压缩 | 不定长，ASCII码，无填充字节，  num(00~3F)+字符串+……+num+字符串+00  如：'3www447993com' | |
| 压缩 | 定长2B，11+offset(14bit)，查询名=DNS报文第[offset+1]个字节到结尾(00) | |
| 混合 | num(00~3F)+字符串+……+num+字符串+11+offset(14bit) | |
| type |  |  | data格式 |
| 0001 | A：IPv4 | 4字节的IP，如：  C0 A8 01 01(192.168.1.1) |
| 0002 | NS：域名服务器 |  |
| 0005 | CNAME：规范名称 |  |
| 0006 | SOA：区开始标记 |  |
| 000B | WKS：熟知服务 |  |
| 000C | PTR：指针记录 |  |
| 000D | HINFO：主机信息 |  |
| 000F | MX：邮件交换记录 |  |
| 001C | AAAA：IPv6 |  |
| 00FC | AXFR：对区域转换的请求 |  |
| 00FF | ANY：对所有记录的请求 |  |
| class | 0001→IN(互联网) | | |
| TTL | 客户程序保留该RR的秒数，通常值为2天 | | |
| Length | 数据长度,单位B | | |
| data | 格式依赖于type | | |
| Authority  RR |  | | | |
| Additional RR |  | | | |

## DNS回复报文

注意DNS回复报文包含查询问题，即报文中的Question字段和查询报文中的Question字段相同

## FTP命令报文

FTP命令：3~4个大写ASCII符，有些带选项参数。

ABOR：放弃先前的F T P命令和数据传输

LIST：列表显示文件或目录

PASS：服务器上的口令

PORT：客户端I P地址（n1 .n2 .n3 .n4）和端口（ n5×2 5 6 +n6）

QUIT：从服务器注销

RETR：检索（取）一个文件

STOR：存储（放）一个文件

SYST：服务器返回系统类型

TYPE：说明文件类型：A表示A S C I I码，I表示图像

USER：服务器上用户名

## FTP应答报文

3个ASCII码数字+报文选项。

软件系统需要根据数字代码来决定如何应答，而选项串是面向人工处理的。

由于客户通常都要输出数字应答和报文串，一个可交互的用户可以通过阅读报文串（而不必记忆所有数字回答代码的含义）来确定应答的含义。

第1位:

1：肯定预备应答。它仅仅是在发送另一个命令前期待另一个应答时启动

2：肯定完成应答。一个新命令可以发送

3：肯定中介应答。该命令已被接受，但另一个命令必须被发送

4：暂态否定完成应答。请求的动作没有发生，但差错状态是暂时的，所以命令可以过后再发

5：永久性否定完成应答。命令不被接受，并且不再重试

第2位:

0：语法错误

1： 信息

2：连接。应答指控制或数据连接

3：鉴别和记帐。应答用于注册或记帐命令

4：未指明

5：文件系统状态

第3位数字给出差错报文的附加含义。例如，这里是一些典型的应答，都带有一个可能的报文串。

• 125 数据连接已经打开；传输开始。

• 200 就绪命令。

• 214 帮助报文（面向用户）。

• 331 用户名就绪，要求输入口令。

• 425 不能打开数据连接。

• 452 错写文件。

• 500 语法错误（未认可的命令）。

• 501 语法错误（无效参数）。

• 502 未实现的M O D E (方式命令)类型。

通常每个F T P命令都产生一行回答。例如， Q U I T命令可以产生如下应答：

221 Goodbye.

如果需要产生一条多行应答，第1行在3位数字应答代码之后包含一个连字号，而不是空格，最后一行包含相同的3位数字应答代码，后跟一个空格符

## SMTP报文

void SMTP()

{

/\*

SMTP命令：

|  |  |
| --- | --- |
| HELO |  |
| MAIL |  |
| RCPT |  |
| DATA |  |
| QUIT |  |
| RSET |  |
| VRFY | 使客户能够询问发送方以验证接收方地址，而无需向接收方发送邮件。通常是系统管理员在查找邮件交付差错时手工使用的。我们将在下一节中给出这方面的例子。 |
| NOOP | 除了强迫服务器响应一个O K应答码（2 0 0）外，不做任何事情。还有附加和可选命令。E X P N扩充邮件表，与V R F Y类似，通常是由系统管理员使用的。事实上，许多S e n d m a i l的版本都把这两者等价地处理。4.4BSD 中的S e n d m a i l版本8不再将两者等同处理。V R F Y不扩充别名也不接受.forward文件。 |
| TURN | 使客户和服务器交换角色，无需拆除T C P连接并建立新的连接就能以相反方向发送邮件（ S e n d m a i l不支持这个命令）。其他还有三个很少被实现的命令（ S E N D、S O M L和S A M L）取代M A I L命令。这三个命令允许邮件直接发送到客户终端（如果已注册）或发送到接收方的邮箱 |

## HTTP请求报文

请求读取URL标志的信息： GET http://www.xyz.edu.cn/dir/index.htm HTTP/1.1

请求读取URL标志的信息首部： HEAD http://www.xyz.edu.cn/dir/index.htm HTTP/1.1

给服务器添加信息： POST http://www.xyz.edu.cn/dir/index.htm HTTP/1.1

用于代理服务器： CONNECT http://www.xyz.edu.cn/dir/index.htm HTTP/1.1

请求选项信息： OPTION http://www.xyz.edu.cn/dir/index.htm HTTP/1.1

存储一个文档： PUT http://www.xyz.edu.cn/dir/index.htm HTTP/1.1

删除URL标志的资源： DELETE http://www.xyz.edu.cn/dir/index.htm HTTP/1.1

环回测试： TRACE http://www.xyz.edu.cn/dir/index.htm HTTP/1.1

## HTTP响应报文

HTTP/1.1 200 OK

HTTP/1.1 400 Bad Request

HTTP/1.1 404 Not Found

HTTP/1.1 301 Moved Permanently

Location: http//www.xyz.edu/ee/index.html

Content-type: image/gif

Content-type: text/html

Content-type: text/plain

Content-type: application/postscript

Content-length:

Last-modified:

Date:

Server:

//空行

实体主体

在最后一个响应报文首部的后面，服务器程序紧跟着图象后发送了一个空行(一个回车/换行对)

因为H T T P协议交换8 bit字节的数据，所以可以通过T C P连接发送二进制数据。

这点不同于其他的I n t e r n e t应用，特别是S M T P协议(卷1的第2 8章)，它通过T C P连接传输7 bit的A S C I I

字符，显式地将每字节的高位设置为0，阻止了二进制数据的交换。

第一条TCP连接是访问主页(GET/)，主页的HTML文档共涉及了7个GIF图象。

客户程序收到这个主页后，马上并行地打开4条T C P连接去获取前4个GIF图象。这是Netscape程序为了减少打开主页总时间的一种方法(

大多数Web客户程序并不像这样，而是只能一次下载一个图象。

并行连接数量可由用户来配置，默认是4个。当这些连接中有一条结束时，客户程序会立即打开一条新的连接来获取下一个图象，直到客户程序取得全部7个图 象。

8条TCP连接都由客户端程序发起，依次使用1114~1 121的8个客户端口号。

8条TCP连接都由服务器程序关闭。

把客户程序发送最初的SYN 看作连接的开始，客户程序收到服务器程序的FIN后发送FIN 认为是连接的结束

取得这个主页以及它所涉及的所有7个图象共需要约12秒的时间。

下一章的图1 4 - 2 2中给出了由客户程序发起的第一条连接(端口1 11 4 )的Tc p d u m p分组跟踪情况。

注意，端口号为1115，111 6，111 7的三条连接是在第一条连接(端口号为111 4 )结束之前建立的，

这是因为N e t s c a p e的客户程序在读到第一条连接上的文件结束标志以后，并在关闭第一条连接之前发起三条无阻塞的连接。

实际上，客户程序在收到FIN标志后约半秒钟才发出FIN分组同时使用多条T C P连接是否真的能减少交互式用户所需的处理时间呢？

为了测试这一点，我们在主机s u n 上运行N e t s c a p e客户程序(图1 - 1 3 )，还是来获取A d d i s o n - We s l e y的主页。

但这台主机是采用如今常用的方式连接I n t e r n e t，即通过拔号调制解调器以28.8 Kb/s的速度连接I n t e r n e t。

修改客户程序的首选文件，对客户程序最大的连接数从1至7都进行了测试。测试时关闭了客户程序的硬盘缓存功能。

在每一种最大连接数下客户程序均运行三次，取结果的平均值。

结果可以看出，从1到4，随着连接数增加，总时间在减少

但是如果用Tc p d u m p来跟踪这种交换，我们会发现，虽然用户可能把连接数设成超过4，但是程序的极限是4。

不管怎么说，超过4条连接后增加连接数对总时间即便有影响也是很小，不如从1 ~ 2，2 ~ 3，3 ~ 4那么明显。

图1 3 - 5所示的总时间比图1 3 - 6所示的最短时间( 1 0 . 2秒)要多约2秒，这是因为客户主机的显示硬件速度有差异。

图1 3 - 6所示的测试是客户程序运行在一台工作站上，而图1 3 - 5所示的测试客户程序运行在一台显示速度和运行速度均较慢的个人计算机上。

[Padmanabhan 1995]指出了多连接方法的两个问题：

1.这样做对其他协议不公平。例如，FTP协议获取多个文件时每次只能使用一条连接(不包括控制连接)。

2.当在一条连接上遇到拥塞并执行拥塞避免(在卷1的第2 1 . 6节中有描述)时，拥塞避免信息不会传递到其他连接上去。

对客户程序来说，同时对同一主机使用多条连接实际上使用的可能是同一条路径。

如果处于瓶颈的路由器因为拥塞而丢弃某条连接的分组，那么其他连接的分组通过该路由器时也同样可能会被丢弃。

客户程序同时使用多个连接带来的另一个问题是容易造成服务器程序未完成的连接队列溢出，

这样会使得客户主机重传它的S Y N分组而造成较大的时延。下一章我们讨论We b服务器时，

将在1 4 . 5节中详细讨论服务器程序的未完成连接队列

## DHCP报文

## RIP请求报文

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 客户端 | 路由器 | 路由守护程序 | 520 |  |
| 服务端 | 路由器 | 路由守护程序 | 520 |  |
| UDP | UDP虽然不保证可靠交付，但开销小，满足RIP的需求  因为UDP不可靠，所以必须不断地(周期性)和邻站交换信息才能使路由信息及时更新 | | | |
| IP |  | | | |
| 路由  守护  程序 | 在Unix系统中，大多数普通的守护程序都是路由程序和网关程序（术语DAEMOn指的是运行在后台的进程，它代表整个系统执行某些操作。DAEMON一般在系统引导时启动，在系统运行期间一直存在）。  在某个给定主机上运行何种路由协议，如何在相邻路由器上交换选路信息，以及选路协议是如何工作的，所有这些问题都是非常复杂的，其本身就可以用整本书来加以讨论（有兴趣的读者可以参考文献[Perlma1992]以获得更详细的信息）。  路由表经常被IP访问（在一个繁忙的主机上，一秒钟内可能要访问几百次），  但是它被路由守护程序更新的频度却要低得多（可能大约30秒种一次）。当接收到ICMP重定向报文时，路由表也要被更新，这一点我们将在9. 5节讨论ROUTe命令时加以介绍。在本章中，我们还将用ETSTAt命令来显示路由表。 | | | |

这种请求报文的命令字段为1，但地址系列字段设置为0，而度量字段设置为16。这是一种要求另一端完整路由表的特殊请求报文。

\*/

|  |  |  |
| --- | --- | --- |
| 标志 | 1 | 01，表示请求 |
| 版本 | 1 | 01 |
| 保留 | 2 | 00 00 |
| 协议 | 2 | 00 02→IP |
| 保留 | 2 | 00 00 |
| IP地址 | 4 |  |
| 保留 | 4 | 00 00 00 00 |
| 保留 | 4 | 00 00 00 00 |
| 度量 | 4 | 00 00 00 10 |

## RIP应答报文

路由部分：由路由信息构成，每个路由信息占20B ，一个RIP报文最多包含25个路由信息

最多包含25条路由，保证报文长度为20×25+4=504小于512B,

为发送整个路由表，经常需要多个RIP\_Reply报文。

\*/

void RIP\_Reply()

{

/\*

首部：

1，02，表示应答

2，01，版本

3~4，00 00

路由1：

5~6，00 00

7~8，00 00

9~12，dst IP：目标网络的IP地址

13~16，00 00 00 00：保留

17~20，00 00 00 00：保留

21~24，00 00 00 10：度量

路由2：

……

\*/

}

## RIP轮询报文

void RIP\_Polling()

{

/\*

首部：

1，05，表示轮询

2，01，版本

3~4，00 00

5~6，00 00

7~8，00 00

9~12，IP

13~16，00 00 00 00

17~20，00 00 00 00

21~24，00 00 00 10

\*/

}

## RIP轮询表项报文

void RIP\_PollingList()

{

/\*

首部：

1，06，表示轮询表项

2，01，版本

3~4，00 00

5~6，00 00

7~8，00 00

9~12，IP

13~16，00 00 00 00

17~20，00 00 00 00

21~24，00 00 00 10

\*/

}

## SNMP报文

## TELNET报文