第十一章 UDP：用户数据报协议

11.1 UDP介绍

UDP是一个简单的面向数据报的运输层协议：进程的每个输出操作都正好产生一个UDP数据报，并组装成一份待发送的IP数据包。

这与面向流字符的协议不同，如TCP，应用程序产生的全体数据与真正发送的单个IP数据报可能没什么联系

UDP不提供可靠性：他把应用程序传给IP层的数据发送出来，但是并不保证他们能到达目的地。

应用程序必须关心IP数据报的长度。如果它超过网络MTU，那么就要对IP数据报进行分片。

UDP三大经典运用

(1)查询类：DNS

没有TCP三次握手过程，快

多个DNS同时查询

(2)数据传输：TFTP

停止等待协议，慢

适合于无盘工作站

(3)语音视频流

支持广播和组播

支持丢包，保障效率

11.2 UDP首部

 端口号：接受和发送进程

TCP端口号和UDP端口号是相互独立的。(rsh和syslog=514)

尽管相互独立，如果TCP和UDP同时提供某种知名服务，两个协议通常选择相同的端口号，这纯粹是为了使用方便，而不是协议本身的要求。(DNS)

UDP长度字段指的是UDP首部和UDP数据的长度。该字段最小为8字节。

11.3 UDP校验和



1. UDP检验和覆盖UDP首部和UDP数据

2. IP首部的检验和，它只覆盖IP首部

3. UDP的检验和是可选的，而TCP的检验和是必需的

4. IP计算检验和和UDP计算检验和之间存在不同的地方。首先，UDP数据报的长度可以为基数，但是检验和算法是把若干个16bit字相加。解决方法是必要时在最后增加填充字节0，这是为了检验和的计算。(也就是说，可能增加的填充字节不被床送)

5. UDP数据报和TCP段都包含一个12字节的伪首部，它是为了计算检验和而设置的。伪首部包含IP首部一些字段。其目的是让UDP两次检查数据是否已经正确到达目的地。

(哈希和校验和不一样，检验和是顺序改变检验和不变，哈希是位置改变哈希值改变)

11.4 IP分片

IP(加上首部)把MTU与数据报长度进行比较。

如果需要则进行分片。分片可以发生在原始发送端主机上，也可以发生在中间路由器上。

把一份IP数据报分片以后，只有到达目的地才进行重新组装。(FR fragment)

重新组装由目的端的IP层来完成，其目的是使分片和重新组装过程对运输层(TCP和UDP)是透明的。

已经分片过的数据报有可能会再次进行分片。(可能不止一次)



当IP数据报被分片后，每一片都成为一个分组，具有自己的IP首部，并在选择路由时与其他分组独立。这样，当数据报的这些分片到达目的端时有可能会失序，但是在IP首部中有足够的信息让接收端正确组装这些数据报片。(标识位和片偏移量)

尽管IP分片过程看起来是透明的，但有一点让人不想使用它：即使只丢一片数据也要重传整个数据。

IP层本身没有超时重传的机制，由更高层来负责超时和重传(TCP有超时和重传机制，但UDP没有。一些UDP应用程序本身也执行超时和重传)。当来自TCP报文段的某一片丢失后，TCP在超时后会重发整个TCP报文段，该报文段对应于一份IP数据报。没有办法只重传数据报中的一个数据报片。

如果对数据报分片的中间路由，而不是起始端系统，那么起始端系统无法知道数据报是如何分片的。就这个原因，经常要避免分片。

试验一：ICMP分片

|  |
| --- |
|  |
| （a）拓扑图 |
|  |
| （b）捕捉报 |
|  |
| （c）分片1(Flag:0x01,Total length:500,Fragment offset 0) |
|  |
| （d）分片2(Flag:0x01,Total length:500,Fragment offset 480) |
|  |
| （e）分片3(Flag:0x01,Total length:500,Fragment offset 960) |
|  |
| （e）ICMP回复 |

(a)R3(fa0/0 192.168.1.2)：设置默认路由(ip route 0.0.0.0 0.0.0.0 192.168.0.2)

R2(fa0/0 192.168.0. 1；fa1/0 198.168.1.1)：

R1(fa0/0 192.168.0. 2)：设置默认路由(ip route 0.0.0.0 0.0.0.0 192.168.0.2)

吞吐量(int fa1/0 ;ip mtu 500)

R1 Ping 192.168.1.2 size 1000 repeat 1(这里发送的1000包含IP首部和UDP首部)

(b-d) 分片1-3(ICMP协议报文存在于分片的主片)

(e)ICMP回复

试验二：UDP分片(类似于ICMP)

|  |  |
| --- | --- |
|  |  |
| （a）PC0->server（TFTP） | （b）Router0(分片1) |
|  |  |
| （c）Router0(分片2) | （d）Router0(分片3) |
|  | |

3PC、1Router和1hub

(a)server(fa0/0 192.168.1.2)：

R2(fa0/0 192.168.0. 1；fa1/0 198.168.1.1)：

PC0(fa0/0 192.168.0. 2)：

吞吐量(int fa1/0 ;ip mtu 500)

R1 Ping 192.168.1.2 size 1000 repeat 1(这里的1000仅仅指的是数据)

(b-d) 分片1-3(UDP协议报文存在于分片的主片)

11.5 IP分片：注意事项

在分片时，除最后一片外，其他每一片中的数据部分(除IP首部外的其余部分)必须是8字节的整数倍。

IP首部被复制到各个片中。但是，端口号在UDP首部，只能在第一片中被发现。

术语：IP数据报是指IP层端到端的传输单元(在分片之前和重新组装之后)，分组是指在IP层和链路层之间传送的数据单元。一个分组可以是一个完整的IP数据报，也可以是IP数据报的一个分片。

分组：在wireshark专有说法。Wireshark抓到实际传输的数据包，会进行额外的处理，如将分片包进行重组，并在最后一个分片包中显示重组的数据。所以此时的分组不一定是实际传输的数据。 (解释上面试验一与二的问题)

Ping 192.186.1.2 size repeat 1

IP + ICMP + Data = 1000字节

IP = 20字节 ICMP + Data = 980字节

第一片IP(20) + ICMP(8) + Data(472) = 500(8+472=480能被8整除)

剩余数据

980 – 8 – 472 = 500

第二片 IP(20) + Data(480) = 500

剩余数据

500 – 480 = 20

第三片 IP(20) + Data(20) + 垫片(6) = 46

Show run int fa0/0 显示端口信息

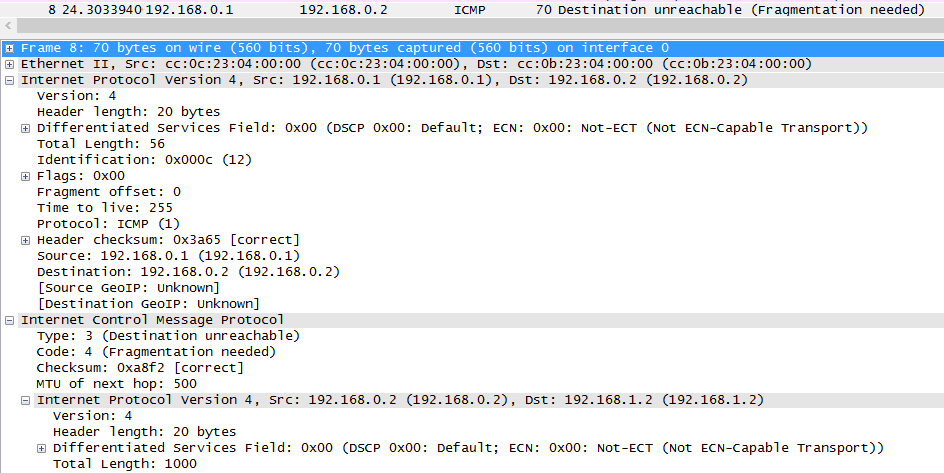
11.6 ICMP不可达差错(需要分片)

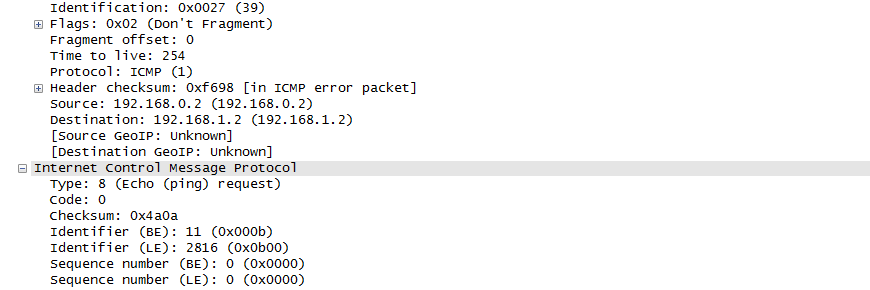
图 需要分片但又设置不分片标志比特时的ICMP不可达差错报文格式

下一站的MTU：未通过网络设备设置的MTU大小

实验三：ICMP不可达差错(需要分片)

在试验一的基础上，R1(Ping 192.168.1.2 size 1000 repeat 1 df-bit)，这样request和reply的包都是df-bit，ping不通也可能是回不通。





1. ICMP不可达差错(需要分片)

本实验中下一站mtu大小为500。

11.7 UDP和ARP之间的交互式用

当ARP缓存还是空的时候，UDP在被发送之前一定会发送一个ARP请求来获取目的主机的MAC地址，如果这个UDP的数据包足够大，IP层一定会对这个数据包分片，系统会让每一个分片都发送一个ARP查询，所有的分片都在等待，但是接受到第一个回应的时候，主机却只发送了最后一个数据片而抛弃其他(怪异的现象)。之后因为分片不能组装，接受主机将最后一个数据片也丢弃，然后发送组装超时ICMP报文(其实很多系统不产生这个差错)，以保证接收主机自己的接收端不会缓存那些永远不能完整组装的分片。

最大UDP数据报长度：

1. IP数据报的最大长度是65535字节。

2. 应用程序可能会受到其程序接口的限制，Socket API提供应用程序可以读写的最大UDP数据报的长度直接相关。

3. TCP/IP内核的实现。

4. 与端口和目的端口的实现有关。

数据报截断：由于IP能够发送或接收特定长度的数据报，并不意味着接收应用程序可以读取该长度的数据。因此，UDP编程接口允许应用程序指定每次返回的最大字节数。如果接收到的数据报长度大于应用程序所能处理的长度，发生的情况会取决于编程接口和实现。

ICMP源站抑制差错：当目标主机的处理速度赶不上数据报接收的速度，因为接受主机IP层缓存会被占满，所以主机就会发出一个ICMP源站抑制差错(源站收到ICMP是可能UDP数据报已经结束进程，但是TCP更愿意接受源站抑制信息，因为TCP丢包会重传)。