

07-router-report

Author: 苗屹松, ID: 2014K8009907018, Date: 2017-11-09

1. Overview:

这次实验做完很开心，因为两方面，一方面是深入理解了ARP机制，另一方面是改了很多bug，虽然花了不少时间debug，但改好了还是很开心。

2. I Give Credit to:

武老师，花了很多时间讲解ARP原理中我本来不懂的地方，以及告诉我ICMP包的bug.

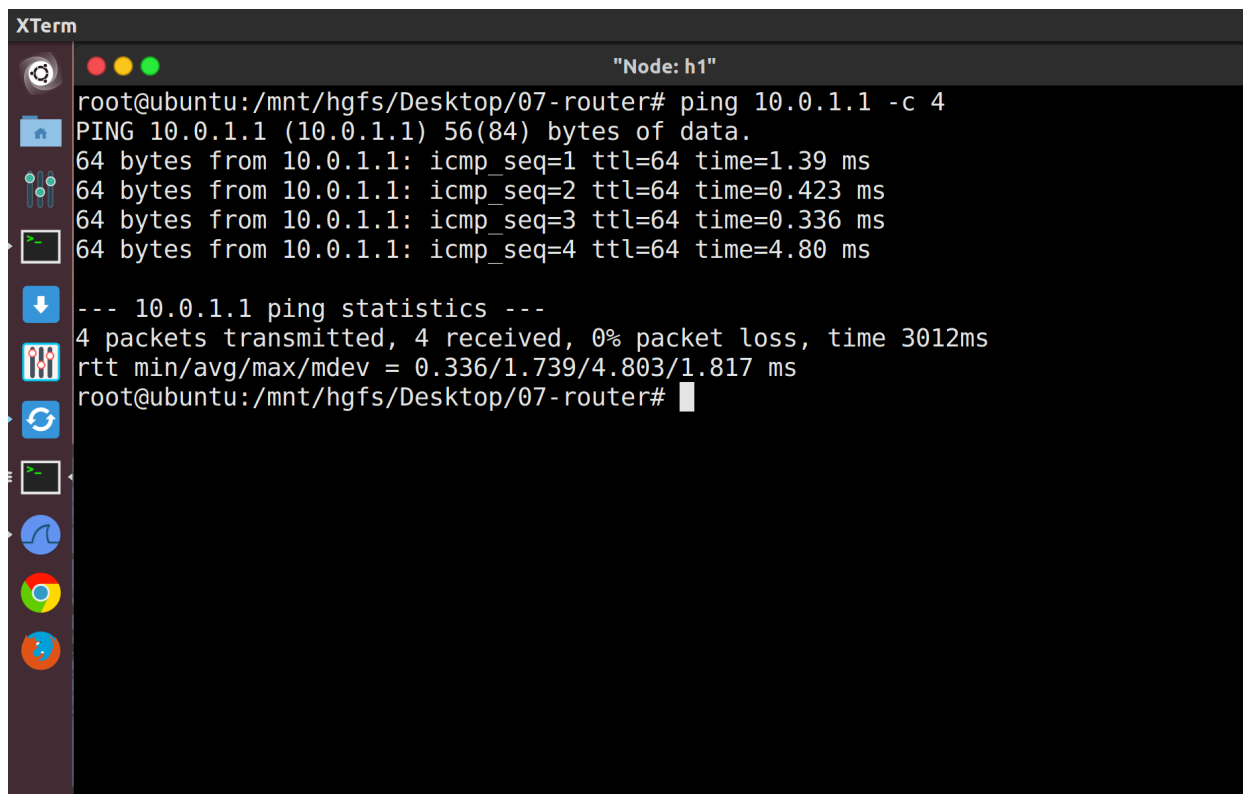
方言歌：告诉我用memcpy直接把新的IP/ICMP信息复制进char*，并多使用宏定义的常数，可以减少bug

林海涛：给我讲解了ICMP中checksum函数的len该怎么算，数值上我确实算对了，但程序还是有错。11月9日老师发现了真理：checksum的对象是 sending_packet指针，我把指针放错地方了！

3. 实验结果:

五个测试都成功了!!!

(1)h1 ping r1-eth0



```
XTerm
"Node: h1"
root@ubuntu:/mnt/hgfs/Desktop/07-router# ping 10.0.1.1 -c 4
PING 10.0.1.1 (10.0.1.1) 56(84) bytes of data.
64 bytes from 10.0.1.1: icmp_seq=1 ttl=64 time=1.39 ms
64 bytes from 10.0.1.1: icmp_seq=2 ttl=64 time=0.423 ms
64 bytes from 10.0.1.1: icmp_seq=3 ttl=64 time=0.336 ms
64 bytes from 10.0.1.1: icmp_seq=4 ttl=64 time=4.80 ms

--- 10.0.1.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3012ms
rtt min/avg/max/mdev = 0.336/1.739/4.803/1.817 ms
root@ubuntu:/mnt/hgfs/Desktop/07-router#
```

(2)h1 ping h2

```
XTerm
"Node: h1"
root@ubuntu:/mnt/hgfs/Desktop/07-router# ping 10.0.2.22 -c 4
PING 10.0.2.22 (10.0.2.22) 56(84) bytes of data.
64 bytes from 10.0.2.22: icmp_seq=1 ttl=63 time=0.354 ms
64 bytes from 10.0.2.22: icmp_seq=2 ttl=63 time=0.564 ms
64 bytes from 10.0.2.22: icmp_seq=3 ttl=63 time=0.201 ms
64 bytes from 10.0.2.22: icmp_seq=4 ttl=63 time=0.269 ms

--- 10.0.2.22 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3041ms
rtt min/avg/max/mdev = 0.201/0.347/0.564/0.136 ms
root@ubuntu:/mnt/hgfs/Desktop/07-router# ping 10.0.2.22 -c 4
PING 10.0.2.22 (10.0.2.22) 56(84) bytes of data.
64 bytes from 10.0.2.22: icmp_seq=1 ttl=63 time=0.203 ms
64 bytes from 10.0.2.22: icmp_seq=2 ttl=63 time=0.264 ms
64 bytes from 10.0.2.22: icmp_seq=3 ttl=63 time=0.224 ms
64 bytes from 10.0.2.22: icmp_seq=4 ttl=63 time=0.256 ms

--- 10.0.2.22 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3067ms
rtt min/avg/max/mdev = 0.203/0.236/0.264/0.030 ms
root@ubuntu:/mnt/hgfs/Desktop/07-router#
```

(3) h1 ping h3

```
XTerm
"Node: h1"
root@ubuntu:/mnt/hgfs/Desktop/07-router# ping 10.0.3.33 -c 4
PING 10.0.3.33 (10.0.3.33) 56(84) bytes of data.
64 bytes from 10.0.3.33: icmp_seq=1 ttl=63 time=0.175 ms
64 bytes from 10.0.3.33: icmp_seq=2 ttl=63 time=0.297 ms
64 bytes from 10.0.3.33: icmp_seq=3 ttl=63 time=0.485 ms
64 bytes from 10.0.3.33: icmp_seq=4 ttl=63 time=0.874 ms

--- 10.0.3.33 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3033ms
rtt min/avg/max/mdev = 0.175/0.457/0.874/0.265 ms
root@ubuntu:/mnt/hgfs/Desktop/07-router#
```

(4) h1 ping 10.0.2.11

```
XTerm
"Node: h1"
root@ubuntu:/mnt/hgfs/Desktop/07-router# ping 10.0.2.11 -c 4
PING 10.0.2.11 (10.0.2.11) 56(84) bytes of data.
From 10.0.2.11 icmp_seq=1 Destination Host Unreachable
From 10.0.2.11 icmp_seq=1 Destination Host Unreachable
From 10.0.2.11 icmp_seq=1 Destination Host Unreachable
From 10.0.2.11 icmp_seq=1 Destination Host Unreachable

--- 10.0.2.11 ping statistics ---
4 packets transmitted, 0 received, +4 errors, 100% packet loss, time 3055ms
pipe 4
root@ubuntu:/mnt/hgfs/Desktop/07-router#
```

(5)h1 ping 10.0.4.4

```
XTerm
"Node: h1"
root@ubuntu:/mnt/hgfs/Desktop/07-router# ping 10.0.4.4 -c 4
PING 10.0.4.4 (10.0.4.4) 56(84) bytes of data.
From 10.0.4.4 icmp_seq=1 Destination Net Unreachable
From 10.0.4.4 icmp_seq=2 Destination Net Unreachable
From 10.0.4.4 icmp_seq=3 Destination Net Unreachable
From 10.0.4.4 icmp_seq=4 Destination Net Unreachable

--- 10.0.4.4 ping statistics ---
4 packets transmitted, 0 received, +4 errors, 100% packet loss, time 3066ms
root@ubuntu:/mnt/hgfs/Desktop/07-router#
```

4. 踩过的坑：

因为老师的PPT把实验原理解释的非常详细，我就不复制粘贴了，这里说一下我踩过的坑。

(1) 理解ARP

一开始做的很慢，因为我没有理解ARP，老师给我讲解了 h1 ping h2的流程，我重现一次：

(a)h1想 Ping h2，这时，h1有一个Linux协议栈，会自动发arp请求给10.0.1.1(r1-eth0)，然后r1-eth0 arp_reply。

(b)然后这个ping包就在r1-eth0被转发。（注意，只有r1-eth*有forward功能）

(c)然后通过最长前缀匹配，找到转发端口为r1-eth1(10.0.2.1)。

(d)我们从r1-eth1想发送packet给h2(10.0.2.22)，可是发现没有h2的mac地址。

(e)这时r1-eth1会广播一个arp请求，只有h2能收到（原因？局域网）

(f)然后h2收到了arp请求，就arp_reply。添加arpcache

(g)然后r1-eth1就能发packet了，然后h2就收到了。

(h)这时，我猜测，h2收到PING包，根据Linux协议栈，会自动回复一个REPLY_PING包，目标是h1(10.0.1.11)，h2->eth1，最长前缀匹配找到转发端口：r1-eth0。

(i)这时在arpcache中没有10.0.1.11对应的mac地址呀，所以r1-eth0模仿刚刚r1-eth1的方法，广播arp请求，h1收到并arp_reply，然后r1-eth1就能找到arpcache中h1的mac地址了，发送REPLY_PING包发给h1，这样一次ping就完成了！

(2) Linux协议栈

实验一开始很慢的原因也是，我发现很多包是自动发送的，但我并没有让它发呀！老师告诉我这是因为Linux协议栈，我目前发现了3个功能！

(a)如果h*是一个ping包的起点，那么它会自动的向r1-eth*发送请求，

(b)如果h*收到了arp请求，它会自动回复！

(c)如果h*是ping包的终点，那么它回向起点自动回复一个arp！

(3)字节序转换

因为这个也产生很多bug

超过1byte的(u16, u32)都需要字节序转换

没超过1byte的(u8)不需要字节序转换

(4)指针的问题

ICMP包做checksum的时候，是对整个ICMP头部+数据 做的。

所以应该把指针设对位置！