路由器

李祁 2017010256

一、所遇到的问题

1.1 在运行时出现了死锁现象

出现原因：我在函数`periodicCheckArpRequestsAndCacheEntries()`中调用了`std::lock_guard<std::mutex> lock(m_mutex);`，但是该函数在`ticker()`中就已经被调用了，所以导致出现死锁。

解决办法：在自己实现的函数中移除mutex lock。

1.2 在运行时出现segmentation fault

出现原因：在从packet中查找函数头的时候，使用了如下代码：

```
struct ethernet_hdr* hEther = (struct ethernet_hdr*)packet.data();
struct arp_hdr* hARP = (struct arp_hdr*)(hEther + sizeof(struct ethernet_hdr));
```

但hEther的大小大于8bit，导致偏移量超过了packet的长度，出现越界。

解决方法：正确的找到`hARP`的方法应该为：

```
hARP = (struct arp_hdr*)((uint8 *)hEther + sizeof(struct ethernet_hdr));
```

1.3 client未能正确收到Time Exceeded

出现原因：没有在`ICMP`的`data`段复制IP包头和前8字节的数据。

解决方法：添加上复制的代码

```
memcpy((uint8_t*)hReplyICMPT3->data, (uint8_t*)hIPv4, ICMP_DATA_SIZE);
```

即可。

二、建议

1.1 DEBUG信息

输出一些DEBUG信息可以帮助我们找到出现问题的根源，但是输出太多信息会导致大文件传输时出现非常多的IO，甚至会直接使得终端不响应。为了能方便的开启和关闭DEBUG信息，使用了如下宏定义：

```
// #define DEBUG
#ifdef DEBUG
#define CERR(msg) \
    { std::cerr << msg << std::endl; }
#else
#define CERR(msg)
#endif
```

在需要输出DEBUG信息的时候直接将`#define DEBUG`取消注释，非常的方便。

1.2 实现思路

这一次工作量的重点在于`simple-router.cpp`中的`handlePacket`函数。在实现其逻辑时，我尽量将不同类型数据包（ICMP、IPv4、ARP）的接收、检查、发送区分开来分别实现。在单独实现时感觉各个函数的逻辑比较清晰，基本只需要参考文档将其格式解析即可。

三、感想

这一次路由器的作业使得我对IPv4，ICMP，ARP有了比较清晰的认识，对路由器的运行机制有了更为深入的了解。感谢助教和老师对本次作业的精心设计！