Router

2020011586 闵安娜

1.遇到的问题和解决方式

环境

运行虚拟机需要自动挂载共享目录 。关机前记得卸载共享目录：使用umount /mnt/hgfs卸载刚刚挂载的目录。 永久卸载就直接把/etc/fstab文件里面相关挂载记录删除即可。

比如移动硬盘，如果不卸载。可能之后的盘不一样，容易损失文件。

死锁

1. ping了一个不存在的ip之后，再做任何操作，路由器都接收不到数据包了：检查ICMP host unreachable和removeRequest，发现有锁。
2. periodicCheckArpRequestsAndCacheEntries() 。中调用了 std::lock_guard lock(m_mutex);，但是该函数在 ticker() 被掉了，出现了死锁。移除mutex lock即可。

其他

实现ICMP消息处理程序时得到segfault（和程序崩溃）：混淆了icmp类型和icmp代码。这是由于错误地计算了icmp头和icmp有效载荷造成的。

一些cpp细节

关于const和引用

- 枚举变量的输入输出一般都采用switch语句将其转换为字符或字符串；枚举类型数据的其他处理也往往应用switch语句，以保证程序的合法性和可读

性。不能直接将常量赋给枚举变量
- 为了保证安全，多用const &，但是这样就不能作为可选参数了，所以用重载函数。

2.对实验的建议与感想

- 采用不同的宏，让std::cerr分类输出，方便查错。就是在std::cerr前后分别加上ifdef 和 endif

3.额外的库

没有使用额外的库

4.设计思路

重点在于handlePacket 函数。

主要是对icmp/arp/ipv4各种包的接收、检查、发送区分开来 分别实现。

为了代码风格的良好，我使用了重载函数。其实也可以再写一些抽象类。但此次代码量不大显得琐碎遂作罢。

高层设计：

simple-router：该模块在一个接口接收数据包并进行处理。当一个数据包被接收时，以太网头被检查以确定它是一个IP数据包还是一个ARP数据包。如果它是一个ARP数据包，则使用arp处理函数来确定该ARP数据包是一个ARP回复还是请求。如果它是一个ARP请求，路由器为所需的接口制定一个ARP回复，如果它存在的话，并将数据包发回给请求者。如果该数据包是ARP回复，路由器将给定的MAC-IP映射添加到arp缓存中，然后发送任何等待该特定映射的数据包。

arp-cache:浏览每个ARP请求并检查它是否有效。如果一个请求没有被发送超过5次，它就是有效的。如果有效，则创建带有最新时间和跳数的ARP请求包。为了创建数据包，需要构建以太网和ARP头。最后，检查ARP缓存中的每个条目，如果其 "isValid "字段为真。如果不是，就把它从缓存中删除。

IPv4处理例程：在检查并确定数据包的类型是一个IP数据包后，IP处理程序将被调用来处理该数据包。handleIp程序从数据包中提取IP头并验证其校验和。如果校验和是有效的，该例程将继续处理该数据包，如果不是，它将直接忽略它。然后，该例程将继续递减TTL，重新计算校验和，并检查该数据包。如果它的TTL小于0，或者该数据包的目的地是路由器，该程序将检查该数据包是否携带ICMP有效载荷，并正确分配它。如果该例程发现该数据包需要被转发，它将调用查找例程，在路由表中找到下一跳的IP地址，并尝试将其转发到那里。如果没有，该请求将被排在队列中，以后再发送。

SimpleRouter：handlePacket：我们有两个函数可以帮助处理 ARP 和 IPv4 数据包。我们检查数据包并确定它是什么类型的数据包，然后再将其发送到要处理的函数。

handleArpPacket：我将它分成两个较小的函数来处理回复和请求。

handleArpReply：用于将数据包发送到下一跳 sendArpRequest：创建响应并发送

5.感谢

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！