CS144：计算机网络导论 2025年秋季

实验检查点 5: 深入协议栈（网络接口）

截止日期：11月9日（周日）晚上11:59

合作政策： 与检查点0相同。请勿查看其他学生的代码或这些作业旧版本的解答。请在你的报告中完全披露任何合作者或任何灰色地带——坦诚是最好的策略。

0 概述

在本周的检查点中，你将深入协议栈并实现一个网络接口：连接在世界各地旅行的互联网数据报与在一跳之间旅行的链路层以太网帧之间的桥梁。此组件可以"安装"在你之前实验中实现的TCP/IP实现的"下方"，但它也将在不同的环境中使用：当你在检查点6中构建路由器时，它将在网络接口之间路由数据报。图1展示了网络接口如何适应这两种环境。

在之前的实验中，你编写了一个TCP实现，可以与任何其他使用TCP的计算机交换TCP段。这些段实际上是如何传递给对端的TCP实现的？正如我们讨论过的，有几种选择：

> `TCP-in-UDP-in-IP`。 TCP段可以携带在用户数据报的有效载荷中。在正常（用户空间）环境中工作时，这是最容易实现的：Linux提供了一个接口（"数据报套接字"，UDPSocket），允许应用程序仅提供用户数据报的有效载荷和目标地址，内核负责构建UDP头部、IP头部和以太网头部，然后将数据包发送到适当的下一跳。内核确保每个套接字都有独占的本地和远程地址及端口号组合，由于内核是编写这些到UDP和IP头部的那个，它可以保证不同应用程序之间的隔离。
>
> `TCP-in-IP`。 在常见用法中，TCP段几乎总是直接放置在互联网数据报内部，在IP和TCP头部之间没有UDP头部。这就是人们所说的"TCP/IP"。这实现起来稍微困难一些。Linux提供了一个称为TUN设备的接口，允许应用程序提供整个互联网数据报，内核负责其余部分（编写以太网头部，实际通过物理以太网卡发送等）。但现在应用程序必须自己构建完整的IP头部，而不仅仅是有效载荷。
>
> `TCP-in-IP-in-Ethernet`。 在上述方法中，我们仍然依赖Linux内核处理网络栈的一部分。每次你的代码向TUN设备写入IP数据报时，Linux都必须构建一个适当的链路层（以太网）帧，将IP数据报作为其有效载荷。这意味着Linux必须弄清楚下一跳的以太网目的地址，给定下一跳的IP地址。如果它还不知道这个映射，Linux会广播一个查询，询问"谁声称拥有以下IP地址？你的以太网地址是什么？"并等待响应。

这些功能由网络接口执行：一个将出站IP数据报转换为链路层（例如以太网）帧的组件，反之亦然。（在真实系统中，网络接口通常有名称如eth0、eth1、wlan0等。）在本周的实验中，你将实现一个网络接口，并将其放在你的TCP/IP栈的最底层。你的代码将产生原始以太网帧，这些帧将通过称为TAP设备的接口移交给Linux——类似于TUN设备，但更低级，因为它交换原始链路层帧而不是IP数据报。

大部分工作在于查找（和缓存）每个下一跳IP地址的以太网地址。用于此的协议称为地址解析协议，即ARP。

我们为你提供了单元测试来检验你的网络接口。在检查点6中，你将在TCP环境之外使用相同的网络接口，作为IP路由器的一部分。

1 开始

> 确保已提交所有解决方案。请不要修改src目录顶层之外的任何文件，也不要修改webget.cc。否则你可能无法合并检查点5的起始代码。
>
> 在实验作业仓库内，运行 `git fetch --all` 以获取实验作业的最新版本。
>
> 通过运行 `git merge origin/check5-startercode` 下载检查点5的起始代码。（如果你已将"origin"远程重命名为其他名称，你可能需要在此处使用不同的名称，例如 `git merge`

upstream/check5-startercode。）

确保构建系统已正确设置：`cmake -S . -B build`

编译源代码：`cmake --build build`

打开并开始编辑 `writeups/check5.md` 文件。这是你实验报告的模板，将包含在你的提交中。

提醒：请在工作时在本地Git仓库中进行频繁的小提交。如果你需要帮助确保做对了，请向同学或教学人员寻求帮助。你可以使用git log命令查看你的Git历史。

2 检查点5：地址解析协议

你在本实验中的主要任务是实现 `NetworkInterface` 的三个主要方法（在 `network_interface.cc` 文件中），维护从IP地址到以太网地址的映射。映射是一个缓存，或"软状态"：NetworkInterface将其保留以提高效率，但如果必须从头开始，映射将自然重新生成而不会导致问题。

```
void NetworkInterface::send_datagram(InternetDatagram dgram, const Address &next;_hop);
```

当调用者（例如你的TCPConnection或路由器）想要将出站互联网（IP）数据报发送到下一跳时调用此方法。你的接口的工作是将此数据报转换为以太网帧并（最终）发送它。

如果目的以太网地址已知，立即发送。创建一个以太网帧（类型 = `EthernetHeader::TYPE_IPv4`），将有效载荷设置为序列化后的数据报，并设置源和目的地址。

如果目的以太网地址未知，广播一个ARP请求以获取下一跳的以太网地址，并将IP数据报排队，以便在收到ARP回复后发送。

例外：你不希望用ARP请求淹没网络。如果网络接口在过去五秒内已经发送了关于相同IP地址的ARP请求，不要发送第二个请求——只需等待第一个请求的回复。同样，将数据报排队直到你获知目的以太网地址。

```
void NetworkInterface::recv_frame(const EthernetFrame &frame;);
```

当以太网帧从网络到达时调用此方法。代码应忽略任何不是发往本网络接口的帧（即以太网目的地址是广播地址或接口自己的以太网地址，存储在ethernet_address_成员变量中）。

如果入站帧是IPv4，使用 `helpers.hh` 中的free `parse()` 函数将有效载荷解析为 `InternetDatagram`，如果成功（即 `parse()` 函数返回true），将结果数据报推送到 `datagrams_received_` 队列。

如果入站帧是ARP，将有效载荷解析为 `ARPMessage`，如果成功，记住发送方IP地址和以太网地址之间的映射30秒。（从请求和回复中都学习映射。）此外，如果它是询问我们IP地址的ARP请求，发送适当的ARP回复。

```
void NetworkInterface::tick(const size_t ms_since_last_tick);
```

随着时间推移调用此方法。使任何已过期的IP到以太网映射过期。

你可以通过运行 `cmake --build build --target check5` 来测试你的实现。此测试不依赖你的TCP实现。

3 问答

预期需要多少代码？ 总体来说，我们预计实现（在network_interface.cc中）大约需要100-150行代码。

如何“发送”一个以太网帧？ 对其调用 `transmit()`。

我应该使用什么数据结构来记录下一跳IP地址和以太网地址之间的映射？ 由你决定！

如何将Address对象形式的IP地址转换为可以写入ARP消息的原始32位整数？ 使用 `Address::ipv4_numeric()` 方法。

如果NetworkInterface发送了ARP请求但从未收到回复怎么办？我应该在某个超时后重新发送吗？使用ICMP向原始发送方发出错误信号吗？ 在现实生活中，是的，这两件事都要做，但在本实验中不用担心。（在现实生活中，如果接口无法获得其ARP请求的回复，它最终会向原始发送方发送一个ICMP"主机不可达"消息。）

如果InternetDatagram正在排队等待获知下一跳的以太网地址，而该信息始终没有到来怎么办？我应该在某个超时后丢弃数据报吗？
是的，网络接口不应该在其愿意为同一IP地址发送另一个ARP请求的时间之外存储数据报。

如果在此PDF发布后有更多常见问题，我在哪里可以阅读？
请定期查看网站（https://cs144.github.io/lab_faq.html）和EdStem。

4 开发与调试建议

在 `network_interface.cc` 文件中实现NetworkInterface的公共接口。你可以在 `network_interface.hh` 中的NetworkInterface类中添加任何你喜欢的私有成员。

你可以用 `cmake --build build --target check5` 测试你的代码。

请重新阅读检查点0文档中关于"使用Git"的部分，并记住将代码保存在分发的Git仓库的main分支上。进行小的提交，使用好的提交消息来标识更改了什么以及为什么。

请努力使你的代码对评分的CA可读。使用合理且清晰的变量命名约定。使用注释来解释复杂或微妙的代码段。使用"防御性编程"——显式检查函数的前置条件或不变量，如果有任何错误则抛出异常。在你的设计中使用模块化。重复的代码块和巨大的函数会使你的代码难以理解。

5 提交

在你的提交中，请仅更改 `src` 目录中的 `.hh` 和 `.cc` 文件。在这些文件中，请随意添加必要的私有成员，但请不要更改任何类的公共接口。

在提交任何作业之前，请按顺序运行以下命令：

(a) 确保已将所有更改提交到Git仓库。记住：在编码时进行小的提交。

(b) `cmake --build build --target format`（规范化编码风格）

(c) `cmake --build build --target check5`（确保自动化测试通过）

(d) 可选：`cmake --build build --target tidy`

在 `writeups/check5.md` 中撰写报告。此文件应为大约20到50行的文档，每行不超过80个字符。报告应包含以下部分：

(a) 程序结构与设计。(b) 实现挑战。(c) 残余错误。

请同时填写完成作业所花的小时数和任何其他评论。

如果有任何问题，请尽快通知课程工作人员。祝你好运！