CS144：计算机网络导论 2025年秋季

实验检查点 6: 构建IP路由器

截止日期：11月16日（周日）晚上11:59

0 合作政策

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。

1 概述

在本周的实验检查点中，你将在现有的NetworkInterface之上实现一个IP路由器。路由器有多个网络接口，可以在任何一个接口上接收互联网数据报。路由器的工作是根据路由表转发它收到的数据报：一个规则列表，对于任何给定的数据报，告诉路由器：

从哪个接口发送出去

下一跳的IP地址

你的工作是实现一个路由器，能够为任何给定的数据报弄清楚这两件事。（你不需要实现生成路由表的算法，例如RIP、OSPF、BGP或SDN控制器——只需实现遵循路由表的算法。）

你的路由器实现将使用Minnow库和新的Router类，以及测试将在模拟网络中检查你路由器功能的测试。检查点6基于你在检查点5中实现的NetworkInterface，但不使用你之前实现的TCP栈。IP路由器不需要了解TCP、ARP或以太网（只需要IP）。我们预计你的实现大约需要30-60行代码。（scripts/lines-of-code工具在起始代码上打印"Router: 38 lines of code"，我们的示例解决方案为"89 lines of code"。）

2 开始

确保已提交检查点5的所有解决方案。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6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/check6-startercode` 下载检查点6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/check6-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/check6.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。

3 实现路由器

在本实验中，你将实现一个Router类，它可以：

跟踪路由表（转发规则或路由的列表），以及

转发它收到的每个数据报：到正确的下一跳，在正确的出站NetworkInterface上。

你的实现将添加到 `router.hh` 和 `router.cc` 框架文件中。在开始编码之前，请查看 `router.hh` 中新Router类的文档。

以下是你将实现的两个方法，以及我们对每个方法的期望：

`add_route`方法

```
void add_route(uint32_t route_prefix,
               uint8_t prefix_length,
               optional<Address> next_hop,
               size_t interface_num);
```

此方法向路由表添加一条路由。你需要在Router类中添加一个数据结构作为私有成员来存储此信息。此方法只需要保存路由以供以后使用。

路由的各部分是什么意思？

路由是一个"匹配-动作"规则：它告诉路由器，如果数据报发往特定网络（一系列IP地址），并且如果该路由被选为最具体的匹配路由，那么路由器应将数据报转发到特定接口上的特定下一跳。

匹配： 数据报是否发往此网络？route_prefix和prefix_length共同指定了一个IP地址范围（一个网络），可能包括数据报的目的地。route_prefix是一个32位数字IP地址。prefix_length是0到32之间的数字（含）；它告诉路由器route_prefix的最高多少位是有效的。例如，要表示到网络18.47.0.0/16的路由（这匹配前两个字节为18和47的任何32位IP地址），route_prefix将是305070080（18 x 2的24次方 + 47 x 2的16次方），prefix_length将是16。任何发往18.47.x.y的数据报都将匹配。

动作：如果路由匹配并被选中该怎么做。如果路由器直接连接到相关网络，next_hop将是空的optional。在这种情况下，下一跳就是数据报的目的地址。但如果路由器通过其他路由器连接到相关网络，next_hop将包含路径上下一个路由器的IP地址。interface_num给出路由器应使用来发送数据报到下一跳的NetworkInterface的索引。你可以用 `interface(interface_num)` 方法访问此接口。

`route`方法

```
void route();
```

这里是关键时刻。此方法需要将每个传入数据报路由到下一跳，通过适当的接口。它需要实现IP路由器的"最长前缀匹配"逻辑来找到要遵循的最佳路由。这意味着：

路由器搜索路由表以找到匹配数据报目的地址的路由。"匹配"意味着目的地址的最高prefix_length位与route_prefix的最高prefix_length位相同。

在匹配的路由中，路由器选择prefix_length值最大的路由。这就是最长前缀匹配路由。

如果没有路由匹配，路由器丢弃该数据报。

路由器递减数据报的TTL（生存时间）。如果TTL已经为零，或者递减后变为零，路由器应丢弃该数据报。

否则，路由器在适当的接口（`interface(interface_num)->send_datagram()`）上将修改后的数据报发送到适当的下一跳。

互联网设计中有一个美妙之处（或至少是一个成功的抽象）：路由器从不考虑TCP、ARP或以太网帧。路由器甚至不知道链路层是什么样的。路由器只考虑互联网数据报，只通过NetworkInterface抽象与链路层交互。当涉及诸如"链路层地址如何解析？"或"链路层是否有自己独立于IP的寻址方案？"或"链路层帧的格式是什么？"或"数据报有效载荷的含义是什么？"之类的问题时，路由器根本不在乎。

4 测试

你可以通过运行 `cmake --build build --target check6` 来测试你的实现。这将在特定模拟网络中测试你的路由器，如图2所示。

5 问答

我应该使用什么数据结构来记录路由表？ 由你决定！但请不要搞得太复杂。每个数据报需要O(N)工作量（其中N是路由表中的条目数）是完全可以接受的。如果你想做更高效的事情，我们鼓励你先获得一个可工作的实现，然后再优化，并仔细记录和注释你选择实现的任何内容。

如何将Address对象形式的IP地址转换为原始32位整数？ 使用 `Address::ipv4_numeric()` 方法。

如何将原始32位整数形式的IP地址转换为Address对象？ 使用 `Address::from_ipv4_numeric()` 方法。

如何比较一个32位IP地址的最高N位（其中`0 <= N <= 32`）与另一个32位IP地址的最高N位？ 这可能是本作业中"最棘手的"部分——把这个逻辑搞对。可能值得编写一个小的C++测试程序（一个简短的独立程序）或向Minnow添加一个测试，以验证你对相关C++运算符的理解并仔细检查你的逻辑。

请记住，在C和C++中，将32位整数移位32位可能会产生未定义行为。测试在sanitizer下运行你的代码，以尝试检测这种情况。你可以通过从minnow目录运行 `./build/tests/router` 直接运行路由器测试。

如果路由器没有到目的地的路由，或者TTL变为零，它不应该向数据报的源发送ICMP错误消息吗？ 在现实生活中，是的，那会很有帮助。但在本实验中不需要——丢弃数据报就足够了。（即使在现实世界中，也不是每个路由器都会在这些情况下向源发送ICMP消息。）

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。

6 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 check6`（确保自动化测试通过）

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

在 `writeups/check6.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，请尽快在实验课上通知课程工作人员，或在EdStem上发帖。