## ARP协议实现

### 实验内容与任务

本次实验的主要内容ARP协议实现。本次实验包含基础任务和拓展任务两部分，具体任务要求如下：

* **基础任务**：编写程序，完善TCP/IP协议栈的ARP协议部分。围绕ARP的初始化、无回报ARP的生成、ARP的输入处理，以及ARP的超时重新请求几个部分完成。并且保证完成ARP协议的完整实现。
* **拓展任务（必做）：**在基础任务实现的ARP协议实现基础上，完成如下任务：
  + ARP多个表项的实现；
  + IP层的输入输出处理。

### 实验过程及要求（限300字）

本组实验的具体过程及对应要求如下：

* + 1. 实验开始前准备工作：在实验开始前，学员需要掌握C语言编程基础，理解TCP/IP协议栈的工作原理，尤其是ARP协议的功能和作用。同时，熟悉MAC地址与IP地址的转换原理，了解网络设备如何通过ARP请求与响应进行地址解析。
    2. 实验过程中：按照实验要求，完成ARP协议的实现。具体步骤包括：具体而言，构造ARP请求和响应报文，实现报文格式的编码与解析。发送ARP请求，构建并广播ARP请求，获取目标设备的MAC地址。处理ARP响应，在收到响应后，提取并记录目标IP与MAC地址的映射。管理ARP缓存，设计缓存机制，存储IP-MAC映射，并实现超时处理机制。实现 IP 层的输入处理、校验和的处理和 IP 包的输出。
    3. 实验结束后：总结ARP协议的实现过程，详细描述报文格式、缓存管理和通信流程，并根据实验要求撰写实验报告，分析实验结果。

### 相关知识及背景

地址解析协议（ARP）是TCP/IP协议栈中连接网络层和数据链路层的关键协议，负责在局域网内将IP地址解析为物理网络的MAC地址。ARP的存在使得主机能够根据目标IP地址获取对应的MAC地址，从而保证数据帧在以太网中的正确传输。作为网络通信的基础性协议，ARP的性能和可靠性直接影响网络的效率和安全性。

在现代网络环境中，ARP协议广泛应用于各种网络设备和系统，如计算机、路由器和交换机等。深入理解ARP的工作原理，有助于掌握网络设备之间的通信机制，理解数据在网络中的传输过程。特别是对于网络工程和网络安全领域，从协议层面了解ARP，有助于识别和防范诸如ARP欺骗等网络攻击，提高网络的安全防护能力。

通过本次实验，学员将亲自动手实现ARP协议的核心功能，包括ARP请求与响应的构建与解析、ARP缓存表的管理等。这不仅加深了对TCP/IP协议栈的理解，也培养了实际编程和问题解决的能力。掌握ARP协议的实现，对后续学习更复杂的网络协议（如IP、ICMP、TCP和UDP）以及从事网络相关工作都有重要的意义。

### 教学目的

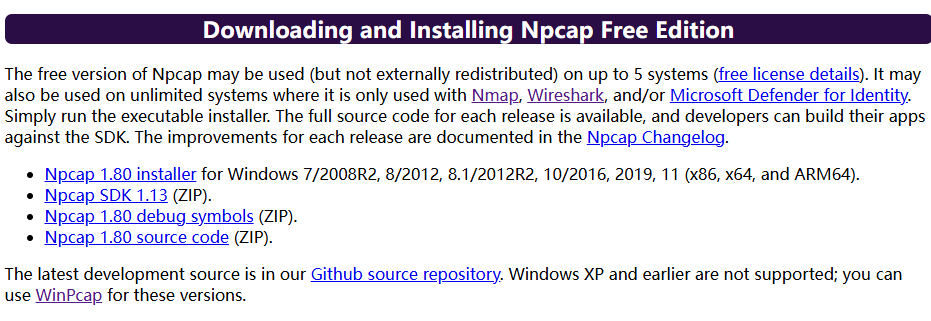
本组实验的主要目的如下：

* + 1. 理解ARP协议的原理，掌握利用C语言编程实现ARP协议的方法；
    2. 理解TCP/IP协议栈中地址解析问题所涉及的关键技术，如数据编码、数据表示和地址映射等；
    3. 通过逐步实现ARP协议的实验，培养学员分析和解决实际问题的能力，加深对网络通信机制的理解。

### 实验教学与指导

* 1. 环境配置

首先，安装wireshark工具。先打开Wireshark的 [官网](https://www.wireshark.org/" \o "官网) 点击Get Acquainted菜单找到Download，进行下载。在最新版本中选择下载Windows 64位的安装包，单击Windows Installer(64-bit) 下载，注意安装Npcap以及WinPcap。



图形用户界面, 应用程序

描述已自动生成

安装VScode，并完成C语言环境配置。VScode官网：<https://code.visualstudio.com/>

图形用户界面, 文本, 应用程序

描述已自动生成

安装MinGW，网址为<https://sourceforge.net/projects/mingw-w64/>。

解压完成后，配置环境变量，参考：<https://blog.csdn.net/2303_79329831/article/details/136400289>。

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

在C语言环境配置完成后，进入到虚拟机环境设置。

我们选用虚拟机软件：VMware。官网：<https://www.vmware.com/products/desktop-hypervisor/workstation-and-fusion>。



下载完成后，我们下载Windows镜像，参考博客：<https://blog.csdn.net/Cappuccino_jay/article/details/125536014?spm=1001.2014.3001.5502>。

图形用户界面, 应用程序

描述已自动生成

虚拟机windows环境配置参考：<https://blog.csdn.net/ewii12567/article/details/139804188>。

在虚拟机Windows安装完成后，我们对虚拟机的网络设置进行一些配置，在VMware位置，点击编辑-虚拟网络编辑器，按如图设置：

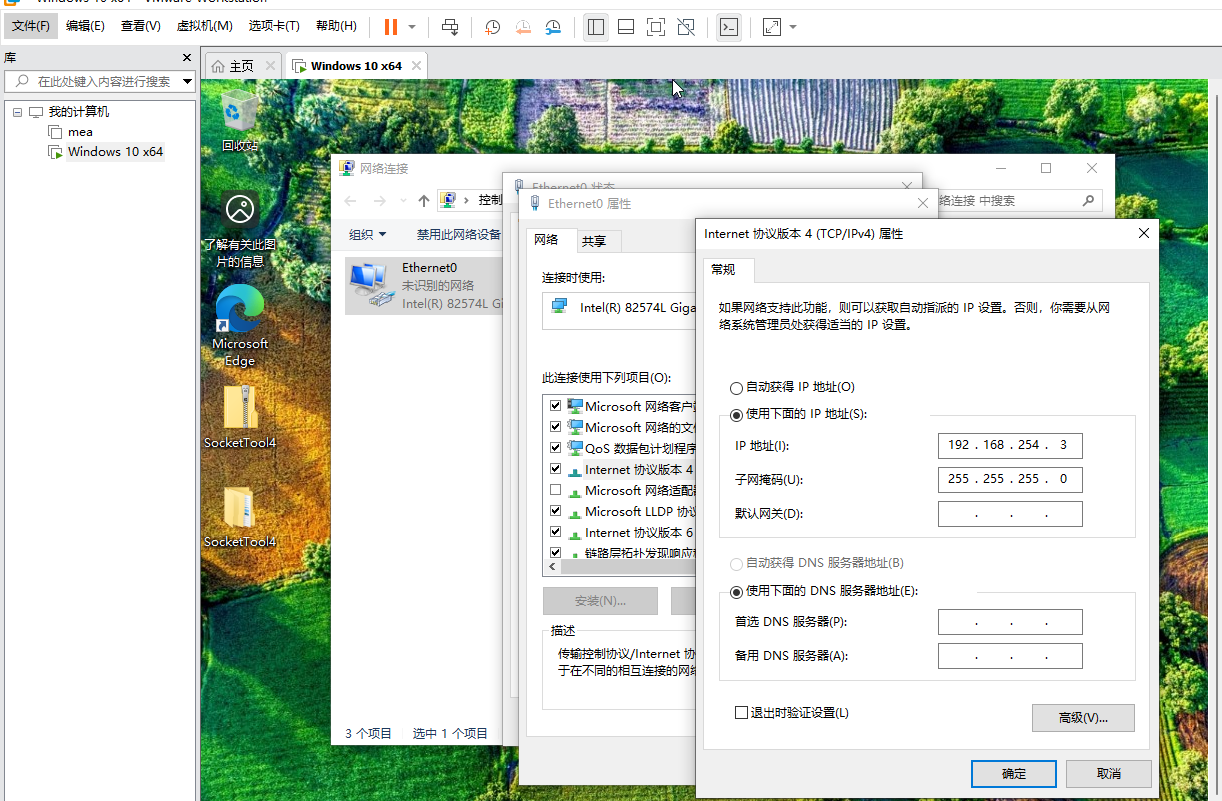


图形用户界面, 文本

描述已自动生成

文本

低可信度描述已自动生成

然后我们对虚拟机内部网络环境配置，网络连接-Ethernet0-属性-Internet版本4：

随后，我们更改防火墙属性，选择更改防火墙，进入例外卡片，勾选文件和打印机共享，最后确定。

**此处应当验收能实现主机与虚拟机的Ping通信。**

接下来，我们安装Cmake，官网：<https://cmake.org/download/>。

Vscode Cmake环境配置参考：<https://blog.csdn.net/weixin_46248871/article/details/137500744>



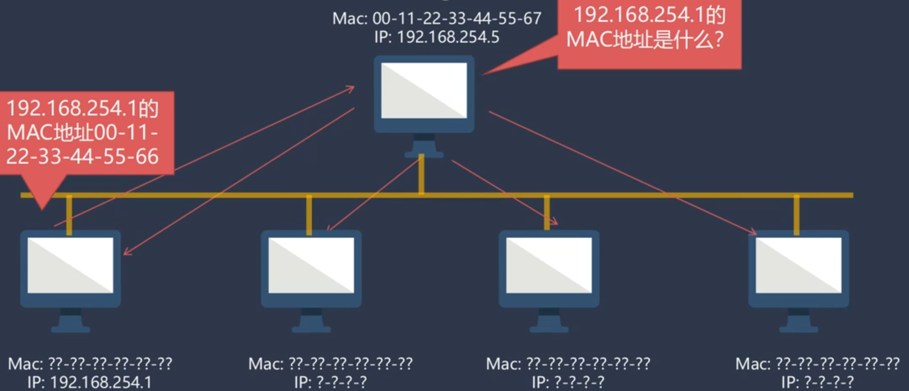
在此处同学们应当实现适配对本机的Cmakelist.txt配置。

其中各部分安装记得安装Vscode插件。至此，环境配置完成。

2）ARP的初始化

ARP是TCP/IP协议栈中连接网络层（IP）和数据链路层（以太网）的关键协议。它的主要功能是在局域网内将已知的IP地址解析为对应的MAC地址，从而保证数据帧能够在以太网上正确传输。

具体来说，当主机需要向某个IP地址发送数据时，会首先查找ARP缓存表，看看是否已有对应的MAC地址。如果没有，主机就会发送ARP请求广播，询问“谁拥有这个IP地址？”拥有该IP地址的设备收到请求后，会发送ARP响应，告知其MAC地址。主机收到响应后，就能将IP地址与MAC地址进行映射，并将其存入ARP缓存表中。



为避免每次都发送请求，便实现了一个映射表，保存之前的查询结果。

表格

描述已自动生成

但实际上要考虑两个问题：1.网卡的数量不固定，同一网络可能有一台或多台计算机；每台计算机可能有1张或多张网卡；计算机可能随时上线或下线，甚至永久退出。

1. IP ->MAC映射并不固定。同一IP可能先后分配给不同的网卡，例如，某机器下线，另一台机器上线，DHCP分配同一IP给不同网卡。

因此，我们的表项需要实现以下三项：1.动态添加新表项。2.删除无效的旧表项。3.表项无效或者错误的检查。

我们需要定义以下结构来实现这个表项：

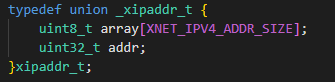
图形用户界面

描述已自动生成

其中 当前状态部分需要实现以下三种：有效/无效/请求中。超时/剩余生存时间需要实现以下内容：超过一定时间重新请求，避免无效/错误表项长期存在。在本次实验中，需要实现多个表项。

我们从给定的start文件的xnet\_tiny.h文件开始：

定义IP地址的结构：



并补充宏定义：#define XNET\_IPV4\_ADDR\_SIZE

接着，实现上述的ARP表项：

文本

描述已自动生成

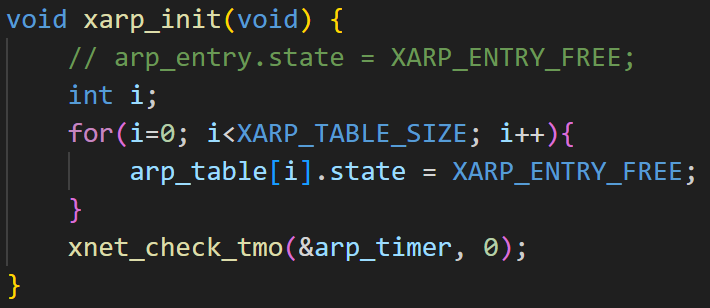
实现初始化函数：

void xarp\_init(void);

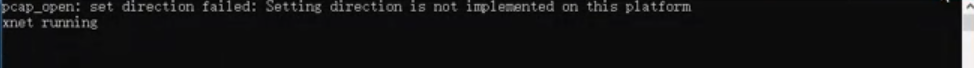
同时在xnet\_tiny.c实现调用，完成表的初始化和全局的静态定义：

并补充宏定义：#define XARP\_TABLE\_SIZE ，设置ARP表项最大数量

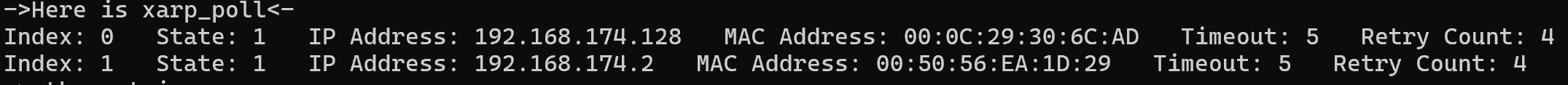
static xarp\_entry\_t arp\_table[XARP\_TABLE\_SIZE];



**此处编译完成的exe文件，应当有如下运行结果：**



ARP 多表项不便于直接验证，可以在程序中添加了额外的输出，用于验证正确性，在每次轮询时输出信息，如下图所示。



3）无回报ARP的生成

在这一节，同学们补充完成ARP空缺部分。

同样的，系统在启用网卡时，首先向网络上发送广播包，告诉所有人自己的IP地址和MAC地址，而其他地址与该系统通信时会先查询内部的ARP表在进行通信，这样就简化了最开始提到的通讯过程。

打开wireshark，启动虚拟机可以发现，启用网卡时，系统会向网络发送无回报ARP包，如下：



这个广播就是通知所有人我使用的这个IP为192.168.254.3，MAC地址就是前面部分。

我们在xnet\_tiny.h继续实现，我们首先定义接口函数，参数为请求的IP地址：

xnet\_err\_t xarp\_make\_request(const xipaddr\_t \* ipaddr);

在此步，我们应当定义接口函数：

具体包格式参考给的RFC826\_ARP文档。定义ARP包结构，补充图中空缺内容

文本

描述已自动生成

设置本机协议栈虚拟网卡IP地址(在此处补充完整)：



在xnet\_tiny.c文件中创建变量和本机广播地址(在此处补充完整)：屏幕上有字

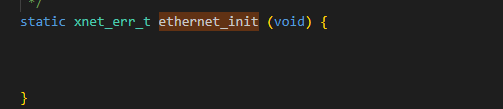
描述已自动生成

创建请求函数，具体功能 \*产生一个ARP请求，请求网络指定ip地址的机器发回一个ARP响应：

文本

描述已自动生成

当网卡启用后，会主动的向网络上发送一个ARP广播包，完善网卡初始化部分：



**此处编译完成的文件启动时，应在wireshark抓到无回报报文，wireshark中IP地址显示应为程序中设置的IP地址。**

4）ARP输入处理

当收到ARP包时， 将进行以下几步：1.包检查：检查数据包格式是否正确，如果有错误，则需要进行丢弃。2.请求包的响应：如果接收到ARP请求包，则需要进行返回响应。3.ARP表项更新：不管接收的是请求包还是响应包，都用其中的源MAC、MAC地址更新ARP表项，以方便后续的通信。

下面继续完成代码部分，我们进入xnet\_tiny.h文件，我们实现负责输入处理的函数xarp\_in,我们在xnet.h创建函数：



为了处理如下的ARP包结构，我们应当移除ARP包头部分，再对ARP输入包进行处理：图形用户界面, 网站

中度可信度描述已自动生成

为了实现此目的，我们修改xnet\_tiny.c的ethernet\_in函数，其中packet为待处理的包：

文本

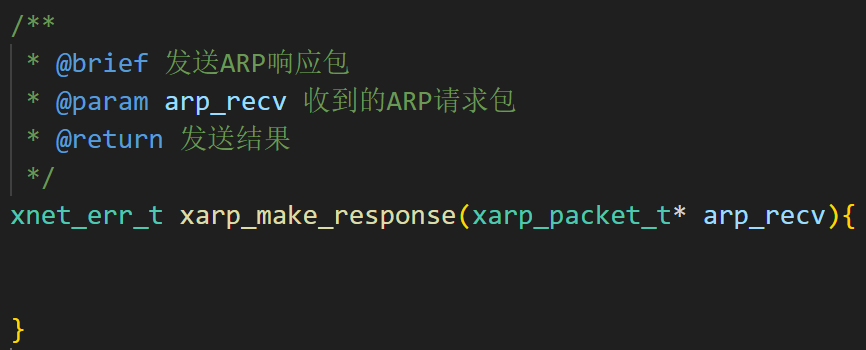
描述已自动生成

现在我们继续实现xarp\_in函数并补充宏定义：

文本

描述已自动生成

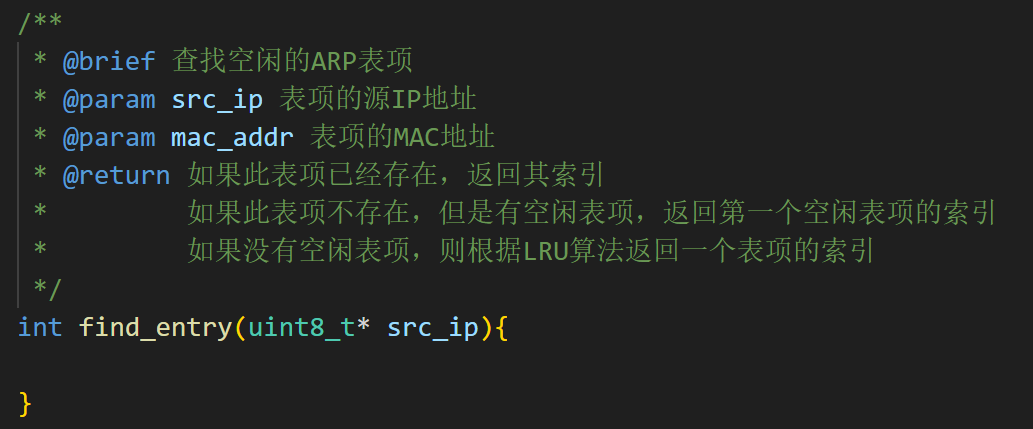
在这个函数中，如果我们收到了请求，那么我们则需要对指定的IP地址的机器进行一个响应，也就是xarp\_make\_response函数，在这个函数中应当做到：1.生成ARP响应。2.生成结果。请完成以下函数补充：

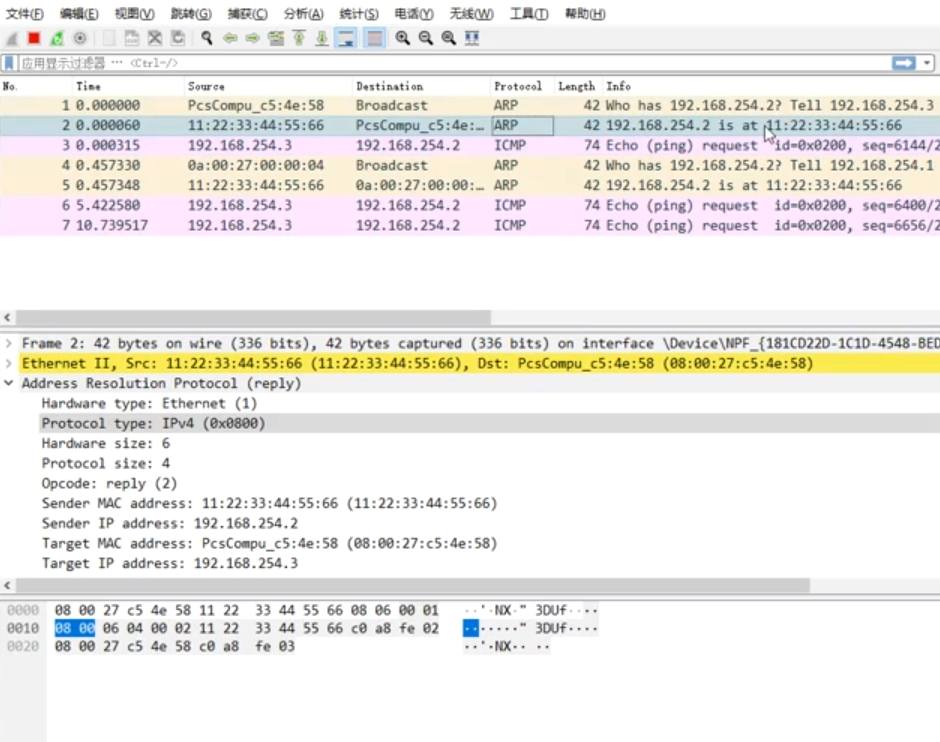


在除了实现响应，我们还需要对我们的ARP表进行更新，现在我们实现函数update\_arp\_table函数，其中src\_ip为源IP地址，mac\_addr为对应的mac地址：



在多表项情况下，更新ARP表项需要先根据 IP 地址在表项中寻找是否原本就存在根据原本存在、不存在但是有空闲表项、不存在但是不存在空闲表项，三种情况进行分别处理。在此处需要编写find\_entry函数查找空闲的ARP表项。



**在这一步，最终实现后，同学们应当实现以下现象，用虚拟机ping实体机，在wireshark能捕获到自己编写的协议栈的响应，如下图：**

5）ARP的超时重新请求

在某些情况下，ARP表项不应当在原有状态一直保持有效状态，如碰到以下情况：计算机下线、网卡变化。在这些情况下ARP表还是保持原有的情况，我们需要能够检测到这种情况，并对这些表项进行更新或者删除。

ARP（Address Resolution Protocol）的超时重新请求机制旨在确保网络设备能够正确解析和更新ARP表项，维护设备之间的通信。ARP协议使用ARP表（ARP Cache）来存储IP地址和MAC地址之间的映射关系。但是，为了避免ARP表中存储的信息变得过时，ARP表项都有一个有效期（通常称为超时时间）。超时时间过后，ARP表项会被标记为失效并从表中删除。

下面我们来实现这个机制，具体到xarp\_poll函数，在这个函数中，我们会每隔一段时间就检查一次ARP表，根据表项的状态进行处理，在时间部分我们设定了一个tmo字段，随着时间会不断减少，当时间减少到0时，我们会重新发送ARP请求，同时我们会将表项的状态切换到正在解析的状态，正在解析的状态同时设置另一个超时，当我们重新发送请求后，在指定的时间段是否收到响应。

具体的，我们从port\_pcap.c部分继续编写：

首先，我们添加这部分新用到的库：

文本, 徽标

描述已自动生成

获取自程序启动以来，过去了多长时间：



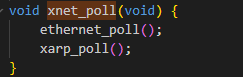
我们在xnet\_tiny.h补充定义：



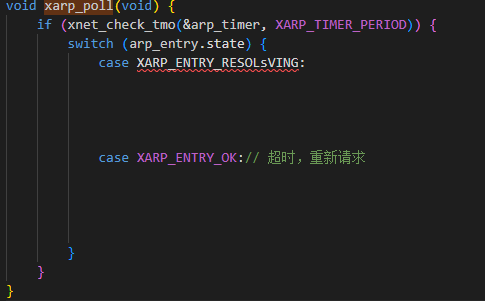
下面实现ARP查询函数，现在xnet\_tiny.h实现函数声明：



我们继续在xnet\_tiny.c主文件完成我们的查询函数xnet\_poll:



接下来实现xarp\_poll函数，查询ARP表项是否超时，超时则重新请求（对此函数补充完整），其中check\_tmo函数用于检查是否超时，并在后面部分实现：

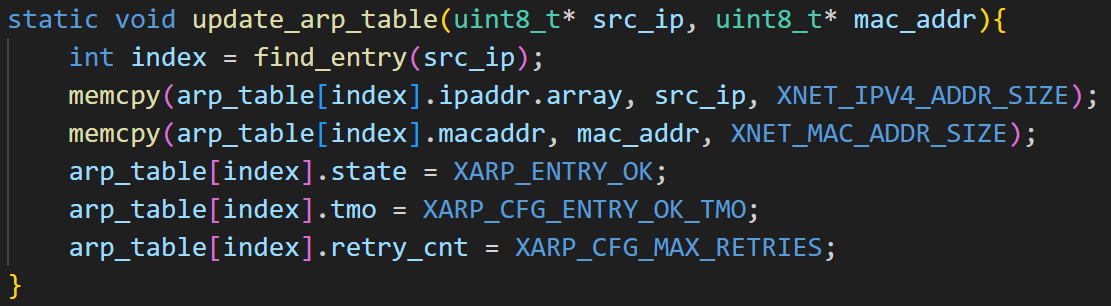


同时在xnet\_tiny.h实现补充xarp\_entry\_t 静态全局变量的定义：

文本

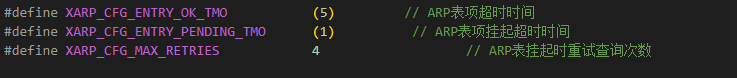
描述已自动生成

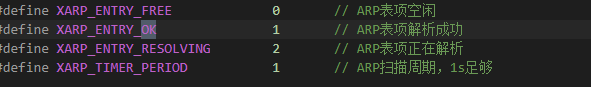
并在xnet\_tiny.c文件中对更新ARP表项时对超时机制进行设置：



其中src\_ip 为源IP地址 ，mac\_addr 为对应的mac地址。

同时在.h头里面添加我们新实现的宏：





下面实现poll函数中使用到的xnet\_check\_tmo，这个函数是用来检查是否超时：

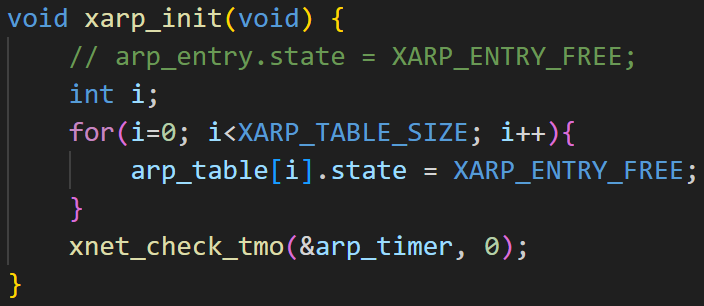
文本

描述已自动生成

并在.c文件中设置我们的一些新的变量：



并在ARP初始化部分实现调用：



**至此，编译程序我们应当能在wireshark看到每隔设定的时间重发一次ARP请求来更新表项。**

表格

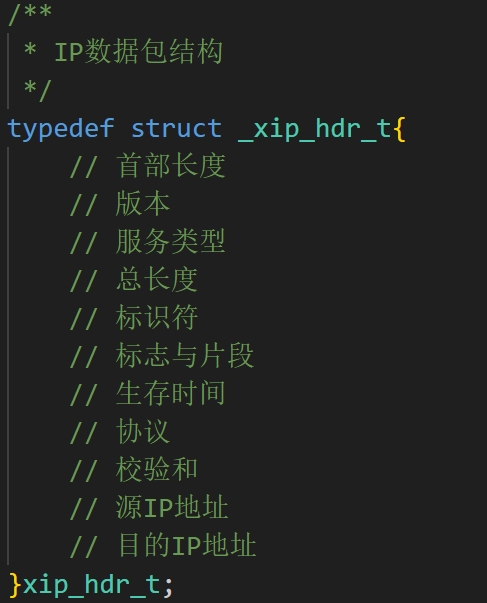
描述已自动生成

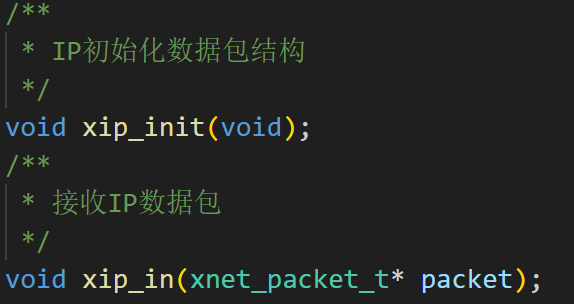
6）IP协议的输入输出处理

接下来我们进行实验的拓展实现，我们要实现虚拟机能够 ping 通我们的代码

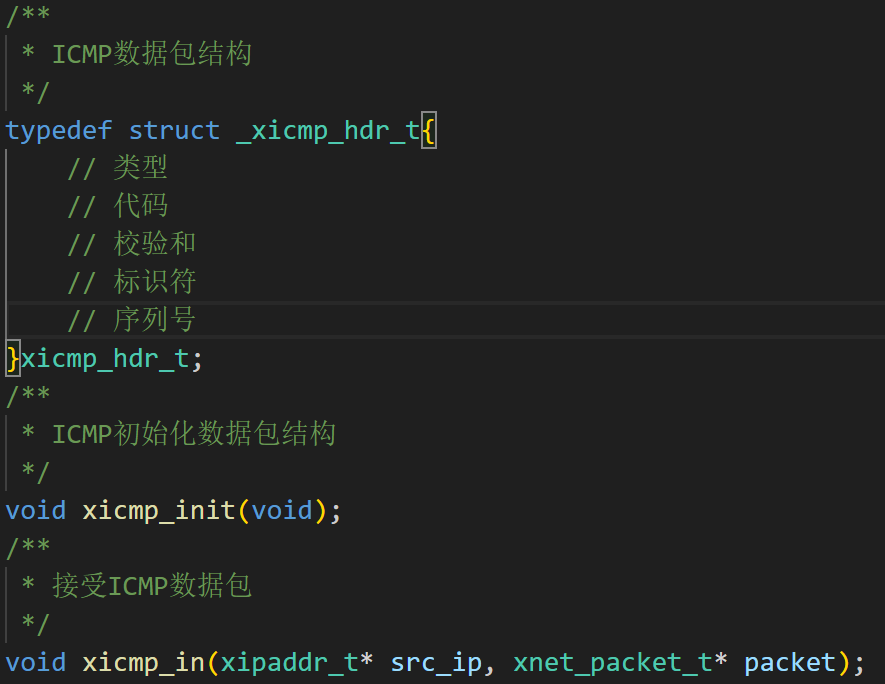
中所设置的 IP 地址，即需要实现 ICMP 层协议。要实现 ICMP 层，必须要实现IP 层。这是因为 ICMP 是网络层的一个协议，它依赖于 IP 协议来传输其消息。整体的实验思路与上述 ARP 协议的实现类似，基本就是在上述已经完成的协议上再进行相关工作的嵌套即可。

那么我们首先实现 IP 层的输入处理、校验和的处理和 IP 包的输出。在xnet\_tiny.h文件中，定义IP数据包的结构并完成初始化，补充图中空缺内容

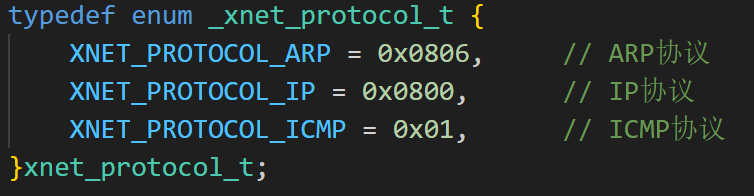




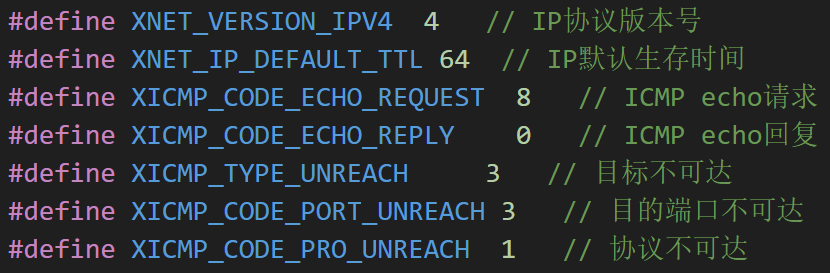
按照同样的思路，定义ICMP报文结构体及属性，并完成初始化



在协议类型中添加ICMP协议类型：

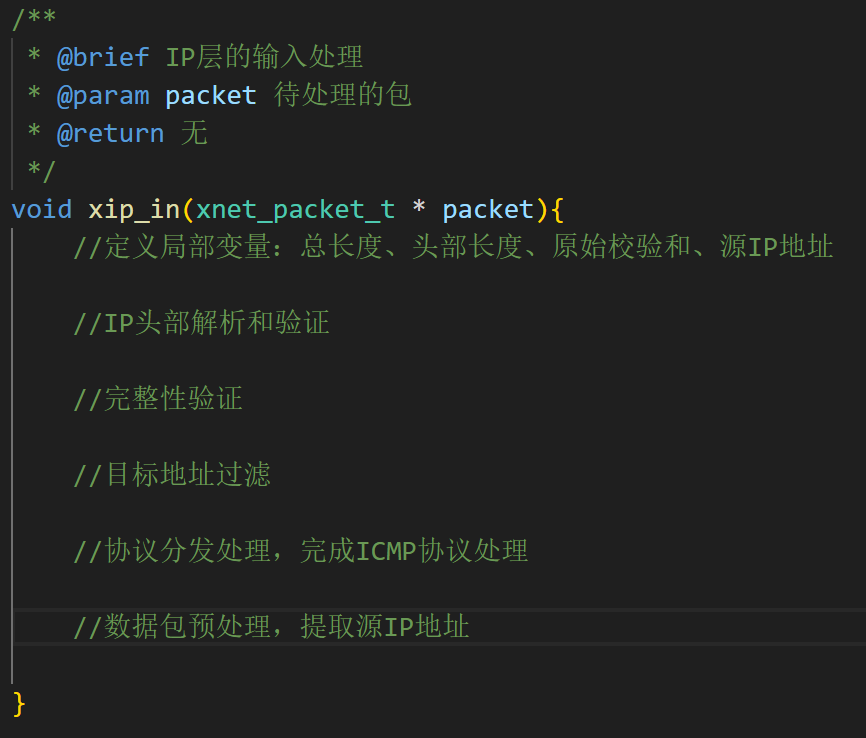


然后补充IP协议版本号，TTL值以及ICMP协议类型和目标不可达错误，

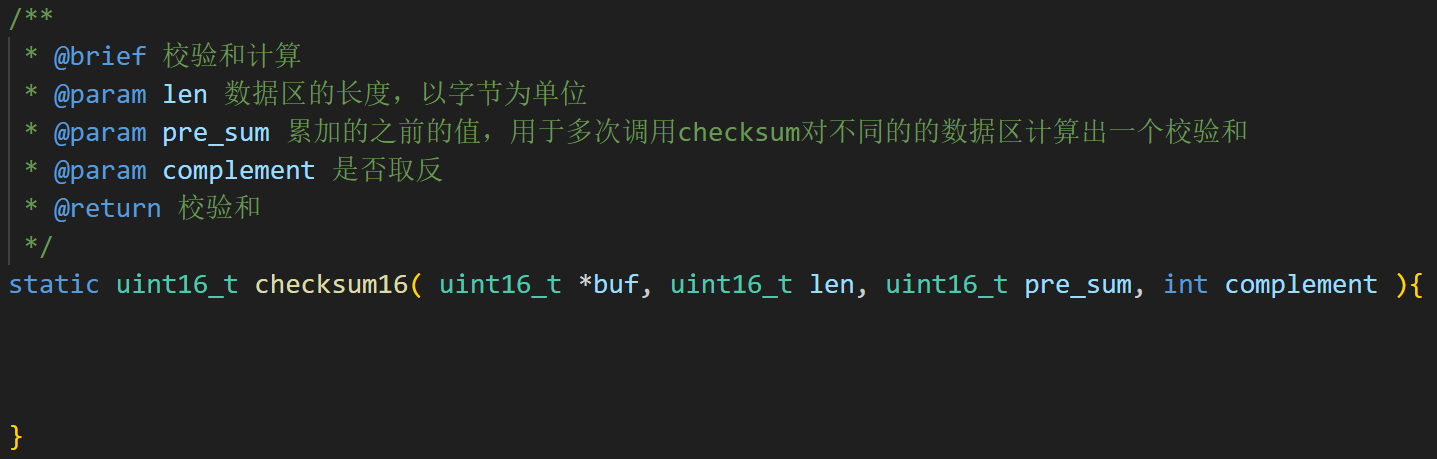


然后在xnet\_tiny.c文件中实现上述函数：

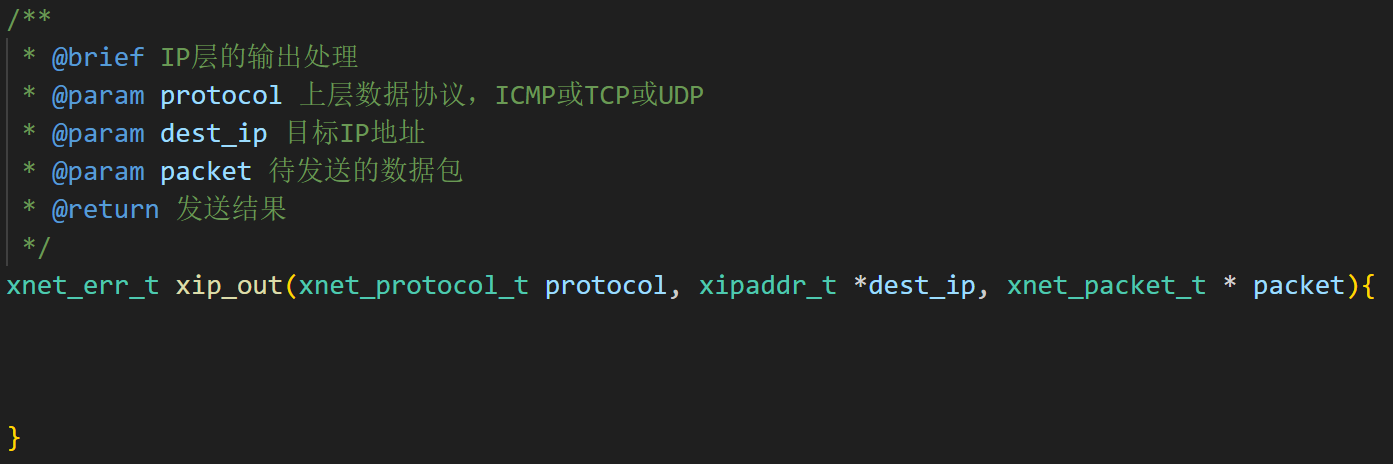
创建xip\_in函数用于处理收到的IPv4数据包。主要实现对IP数据包的必要常规检查，包括版本号要求检查、长度要求检查、校验和要求检查等，并且要求只处理目标IP 为自己的数据包，其他广播之类的 IP 全部丢掉：



在此之前需要实现校验和的处理函数checksum16，IP 层的校验和是 IP 协议头部中的一个字段，用于检测 IP 数据包在传输过程中是否出现错误。



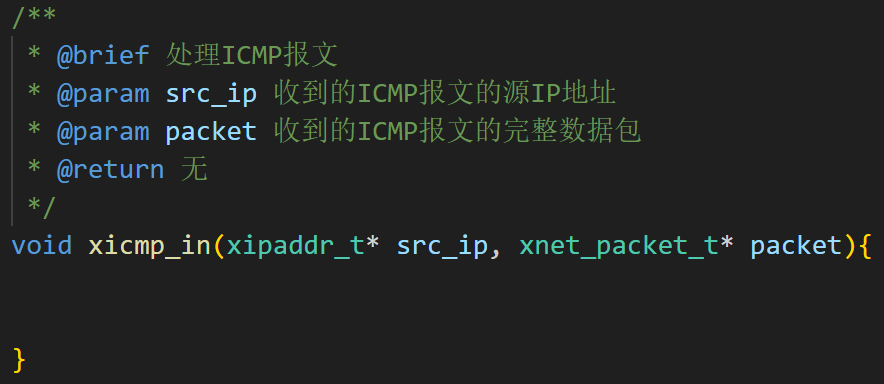
然后实现IP报的输出函数，与前文 ARP 包的输出实现非常相似，根据 IP包的特点进行相应的调整和修改即可，在此不再过多阐述。



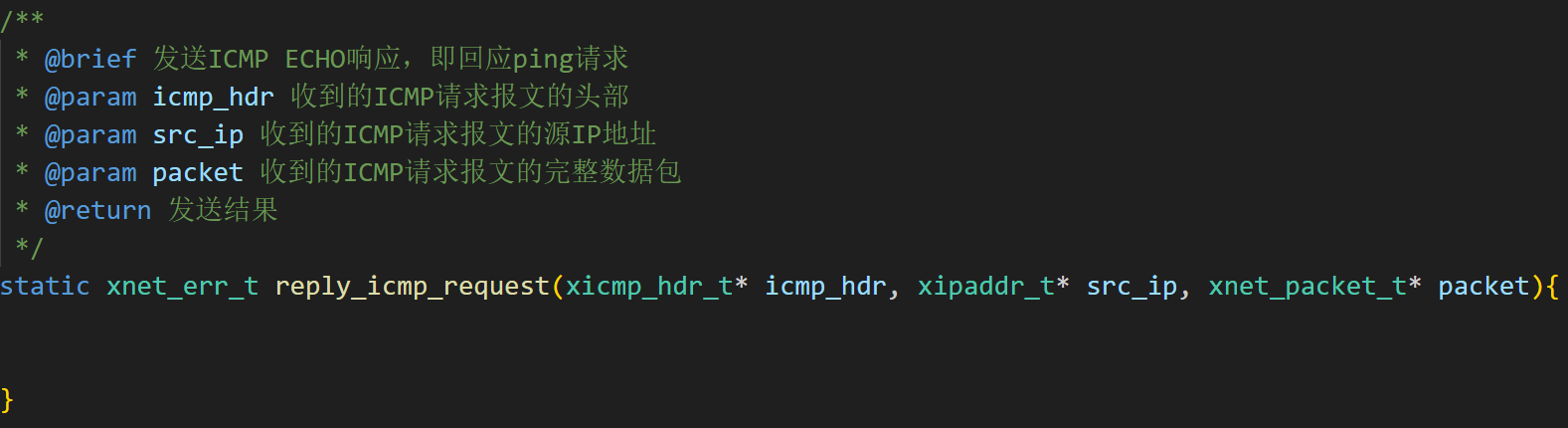


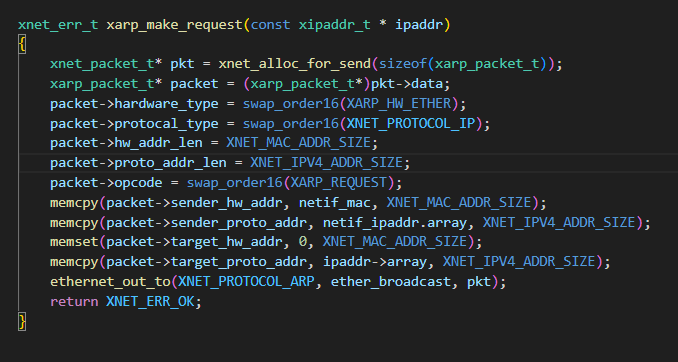
为了实现 ICMP 层的协议，我们需要完成 XNET\_PROTOCOL\_ICMP: 情况下的后续操作函数。其中首先调用 remove\_header()函数移除包头，之后将剩下的包传 给 ICMP 协 议 层 进 行 处 理 。

接下来我们继续实现ICMP包的输入处理函数xicmp\_in：



需要实现数据包的检查，数据包长度需要大于完整的ICMP头部长度，如果数据包检查正常，那么进行发送 ICMP ECHO 响应的 reply\_icmp\_request函数的调用，对其进行处理。

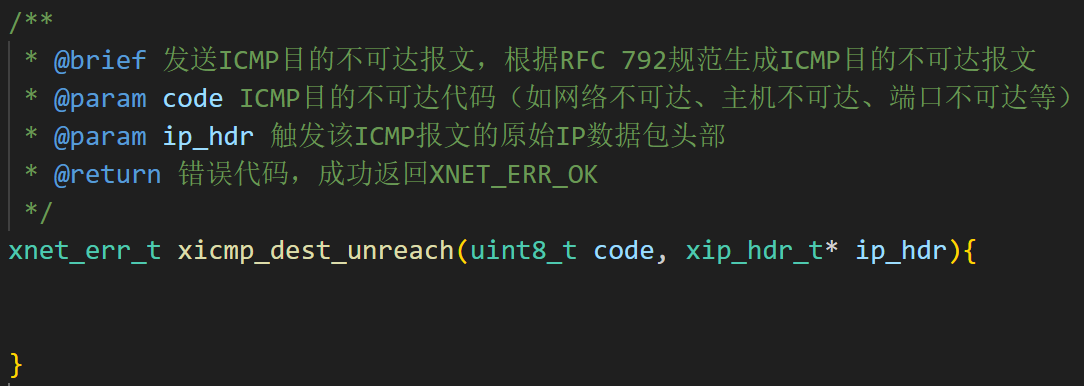




为了回复 ICMP 请求，首先需要分配一个新的回复数据包，用于承载回复信息，编写xnet\_alloc\_for\_send函数实现。在新的数据包中，需要构建一个 ICMP 头部，这个头部将包含回复类型、代码、请求的 ID 和序列号等信息。

将原始 ICMP 请求中的数据部分复制到回复包中，通常这部分是请求者发送的数据，用于验证通信的完整性。调用checksum16() 函数计算 ICMP 回复包的校验和，确保数据包在传输过程中的完整性。

此外，还需要编写xicmp\_dest\_unreach 函数用于发送 ICMP（Internet Control Message Protocol）目的不可达消息，以通知源设备目标不可达。函数接受两个参数：不可达代码 code和原始 IP 头 ip\_hdr。

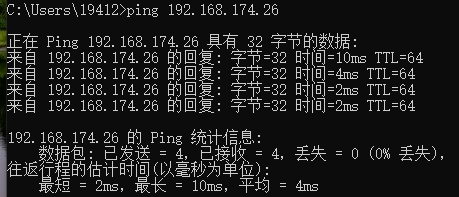


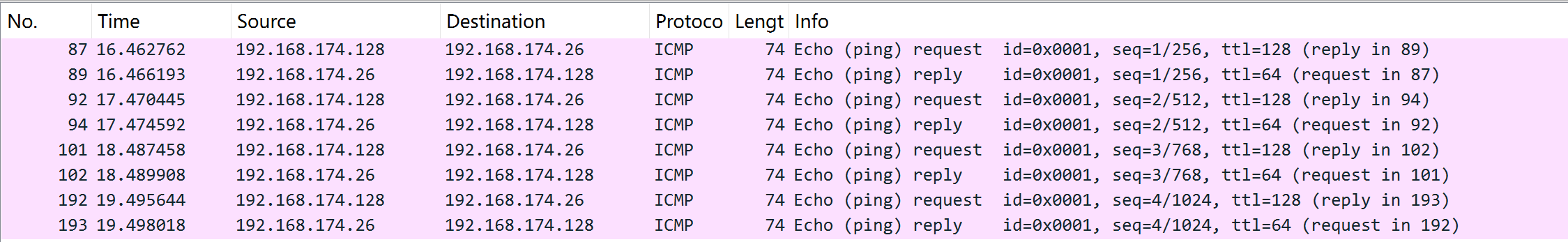
最终通过IP层发送，调用IP输出函数xip\_out()发送回复。

然后补充协议栈的初始化函数，将实现的 ip 与 icmp 初始化函数添加进去：



完成上述实现之后，同学们应当实现我们使用虚拟机可以 ping 通我们在代码中所设置的 IP，并且可以在 wireshark 中抓到相对应的报文。





### 实验原理及方案

ARP（地址解析协议）是TCP/IP协议族中用于将IP地址解析为MAC地址的重要协议。IP通信依赖于数据链路层的硬件地址（MAC地址），而ARP负责动态地将网络层的IP地址转换为对应的数据链路层MAC地址，从而实现设备间的通信。ARP协议的实现主要包括发送ARP请求、接收并处理ARP响应、更新ARP缓存、以及缓存超时机制。

本次实验围绕TCP/IP协议栈的ARP协议的实现这一问题展开，要求学员在理解ARP协议原理的基础上，深入探究ARP的技术细节，动手实现ARP协议的C或Python程序。

* 1. ARP的初始化

在一个典型的局域网中，设备通过IP地址进行网络层通信，但IP地址并不能直接用于数据链路层传输。以太网等数据链路层协议使用MAC地址进行通信，因此，发送设备需要将目标IP地址解析为MAC地址才能发送数据帧。

如果该设备的ARP缓存中没有目标设备的MAC地址映射，它会广播ARP请求，询问网络上哪个设备持有特定的IP地址。ARP请求是一个以太网层的广播包，发送到子网内所有设备，只有持有目标IP地址的设备才会进行响应。

ARP初始化的过程是设备发现并解析网络中其他设备的关键步骤。ARP请求包含源设备的IP地址和MAC地址，而目标设备通过ARP响应提供其对应的MAC地址。这个机制确保设备能够通过网络层（IP地址）和链路层（MAC地址）之间建立正确的映射关系。

* 1. 无回报ARP的生成

无回报ARP（Gratuitous ARP），又称为“主动ARP”或“自愿ARP”，是一种特殊的ARP操作。与典型的ARP请求不同，无回报ARP并不是为了解析目标设备的MAC地址，而是设备主动向网络发送广播ARP包，通常用于更新网络中的IP-MAC映射关系、检测IP地址冲突等。

无回报ARP是设备主动广播自身的IP地址和MAC地址，不带有显式的ARP请求和响应互动。其主要目的是通知网络中其他设备更新其ARP缓存表中的信息。这种情况下，设备并不期待其他设备回应。它是单向广播的，通常被用于下列几种情况：

更新网络中的ARP表：当设备的MAC地址或IP地址发生变动时，可以主动发送无回报ARP，以便通知网络中其他设备更新其ARP缓存。

IP冲突检测：设备在启动时，通过发送无回报ARP来检测是否有其他设备占用了相同的IP地址。如果另一台设备使用了相同的IP地址，它会回应此ARP广播，从而帮助设备检测到IP冲突。

负载均衡器和高可用性系统：当系统切换主备设备时，备设备通常会发送无回报ARP来通知网络中的所有节点其IP-MAC映射已经改变，避免继续向已下线的设备发送数据。

无回报ARP的生成过程如下：

1. 生成ARP广播包：设备在确定需要广播自身IP-MAC映射时，会生成一个ARP广播包。该包包含设备自身的IP地址和MAC地址，并且目标硬件地址设置为全0，因为无回报ARP并不是请求对方设备的MAC地址，而是向网络中的所有设备广播自身的信息。
2. 设置操作码为ARP请求：尽管无回报ARP是主动广播，但它在帧结构中被标记为ARP请求（操作码为1），这使得网络中的其他设备会将其视为一种信息广播，用于更新ARP缓存。
3. 发送广播：ARP广播包通过以太网层进行传输，目标MAC地址为FF:FF:FF:FF:FF:FF，即局域网内的所有设备都可以接收到此广播。
4. 网络中的设备处理：网络中所有收到此广播的设备会检查ARP包中的发送方IP地址和MAC地址，并将其更新到本地的ARP缓存表中。这样，即使该IP地址之前未出现在这些设备的ARP表中，它们也会记录并更新新的映射。
   1. ARP的输入处理

ARP（地址解析协议）的输入处理指的是设备在接收到ARP请求或响应时，如何对该ARP报文进行解析和处理，并据此更新设备的ARP缓存，或进一步采取必要的网络行为。ARP输入处理的核心任务是解析报文，更新ARP缓存，并根据报文类型采取不同的操作。

在这部分有以下步骤：

1. 接收ARP报文：设备通过网络接口接收到ARP报文，无论是广播还是单播形式。这些ARP报文可以是ARP请求、ARP响应，或者是无回报ARP。
2. 解析ARP报文：设备对ARP报文进行解析，提取其中的关键信息。
3. 检查报文有效性：设备检查ARP报文的有效性，包括检查硬件类型是否为以太网、协议类型是否为IPv4、操作码是否为合法的请求或响应。如果报文不符合ARP协议规定，设备将丢弃该报文。
4. 更新ARP缓存：根据ARP报文中的信息，设备更新自己的ARP缓存表。设备通常会把报文中的发送方IP地址和发送方MAC地址映射记录下来，以便将来进行快速的IP到MAC地址解析。
5. 据操作码进行处理：不同类型的ARP报文有不同的处理方式：如果接收到的是ARP请求，设备需要检查目标IP地址是否与自身的IP地址匹配，如果匹配，则需要发送一个ARP响应包，告知请求设备自己的MAC地址。如果接收到的是ARP响应，设备会根据响应包中的信息，更新或添加到ARP缓存表，并不再发送进一步的响应。如果接收到的是无回报ARP，设备会将报文中的IP-MAC映射记录下来，以更新其ARP缓存。
   1. ARP的超时重新请求机制

ARP（地址解析协议）的超时重新请求机制指的是设备在尝试解析某个IP地址到MAC地址时，若未能在设定的时间内收到响应，会采取的重发ARP请求的策略。这种机制旨在保证网络设备在通信中能够及时获取目标设备的MAC地址，并维持ARP缓存的准确性。

ARP缓存存储的是IP地址与MAC地址之间的映射关系。在通信过程中，网络设备通常会先查询ARP缓存以查找目标设备的MAC地址。如果缓存中存在该IP地址的记录，设备会直接使用缓存中的MAC地址进行通信；如果没有找到相应记录，设备会发出ARP请求，广播请求目标IP地址对应的MAC地址。

如果设备在发送ARP请求后，未能在指定的时间内收到ARP响应，它会认为该ARP请求失败。这时，设备会重新发送ARP请求，通常会进行一定次数的重发，以确保能够成功解析目标设备的MAC地址

步骤如下：1. 发送ARP请求：当设备需要与一个目标IP地址进行通信时，它首先会查询ARP缓存表。如果缓存中没有该IP地址的对应条目，设备会发出一个ARP请求，以广播方式在本地网络中请求目标设备的MAC地址。

2. 等待响应： 设备在发送ARP请求后，进入等待状态。在这个状态下，设备会等待目标设备的ARP响应，通常是几百毫秒到几秒的时间（具体取决于设备的实现和网络情况）。

3. 超时判断： 如果在规定的时间内设备没有收到目标设备的ARP响应，它会触发超时事件，认为该请求失败。此时，设备进入重新请求机制，准备发出新的ARP请求。

4. 重发ARP请求： 超时后，设备会重新发送ARP请求，再次请求目标IP地址的MAC地址。这个过程通常会重复数次，直到收到响应或者达到最大重试次数为止。设备之间的重试次数和重试间隔时间是预先设定的，例如，设备可能设置为重试3到5次，每次间隔1秒。

5. 处理无响应的情况： 如果设备在多次重发后，依然没有收到ARP响应，它将放弃请求，并在上层协议（如IP、TCP或UDP）层上返回错误，表明目标设备不可达。这通常会导致应用层超时或连接失败。设备可能会记录这一信息，并在稍后再尝试通信时重新发起ARP请求。

6. ARP缓存条目的超时： 即使设备成功解析了目标设备的MAC地址，ARP缓存中的条目也并非永久有效。每个ARP条目都有一个生存时间（TTL），通常为几分钟到几十分钟。超过这个时间后，如果该条目未被更新，设备会将其删除。当设备再次需要该目标设备的MAC地址时，会重新发出ARP请求。

* 1. IP层的输入输出处理

IP（Internet Protocol，网际协议）是TCP/IP协议族中最核心的网络层协议，负责在不同网络之间传输数据报文，为上层协议（如ICMP、TCP、UDP等）提供无连接的、尽力而为的传输服务。IP层的主要功能包括：数据报的封装与解封、路由与转发、分片与重组、以及差错检测等。

在本实验中，IP层的输入输出处理是ARP协议实验的拓展部分，旨在实现完整的网络层通信，使得上层的ICMP协议能够在虚拟网络环境中正确传输，从而实现虚拟机与本地实验程序之间的互通（如Ping通信）。

IP层输入输出的基本原理如下：

1. IP层输入处理原理

当设备从链路层接收到一个以太网帧后，如果帧类型字段表明其上层协议为IP，协议栈就会将其交由IP层进行输入处理。输入处理的主要步骤包括：

①协议头部解析与检查：IP层首先对数据报的首部进行解析，提取版本号、头部长度、总长度、协议类型、源地址和目的地址等关键字段。系统会验证版本号是否为IPv4、头部长度是否合法、总长度是否匹配、以及首部校验和是否正确。若发现校验和错误或长度不符，说明数据包在传输中被损坏，应直接丢弃。

②目的地址匹配与过滤：IP层只接收发送给本机IP地址的数据包。如果目的IP与本机IP不符（例如广播包或发往其他主机的单播包），协议栈将丢弃该包，以确保不会处理与本机无关的流量。

③去除IP头并上交上层协议：在完成校验与验证后，IP层去除包头，将净荷部分交给相应的上层协议处理。例如，如果协议类型字段为1（ICMP），则将数据交给ICMP模块进一步解析。

通过以上过程，IP输入处理实现了从链路层到网络层的正确过渡，保证上层协议接收到的数据是完整且有效的。

1. IP层输出处理原理

IP层的输出处理主要负责根据上层协议生成的数据，封装成合法的IP数据包并发送给链路层。其主要步骤包括：

①IP头部构建：IP层在发送数据前，需要构造完整的IP头部，包括版本号、首部长度、服务类型（TOS）、总长度、标识、标志位、片偏移、TTL（生存时间）、协议类型、头部校验和、源IP地址和目的IP地址等字段。

其中TTL字段用于限制数据包在网络中转发的跳数，防止路由循环；协议字段标识上层协议类型（如ICMP为1，TCP为6，UDP为17）。

②校验和计算：IP协议使用16位校验和来检测IP头部在传输过程中的错误。发送方在生成数据报时，需调用checksum16()函数计算校验和并填入头部；接收方在输入处理时重新计算校验和以验证数据完整性。

③调用ARP解析并下发至链路层：在生成完整的IP包后，IP层通过ARP模块查找目标IP地址对应的MAC地址。如果ARP缓存中没有该映射，IP层会触发ARP请求以解析MAC地址；待解析完成后，再将完整的IP数据报交由以太网层封装成以太网帧发送。

### 实验报告要求

本组实验的报告要求包含：实验目的和要求、实验内容和原理、操作方法与实验步骤、实验结果与分析、以及实验源代码等。其中实验结果与分析为实验报告的重点。具体实验报告模板参见附件一。

### 考核要求与方法

本实验成绩由现场验收成绩和实验报告成绩两部分构成，均按百分制计分，各占总成绩的70% 和30%。现场验收成绩由教员/教辅根据实验内容完成情况、以及对所完成内容的理解和掌握情况综合评分。其中，现场验收成绩各部分分值如表1所示。具体而言，表格中分数为各个部分最大分值，具体得分在0到最大分值之间，由教员/教辅根据实验完成情况综合评分。

表1 实验现场验收各部分分值表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 基础任务 | | | | 扩展任务 |
| 环境的基础配置 | ARP的初始化 | 无回报ARP的生成 | ARP的输入处理 | ARP的超时重新请求 | 多个表项的实现及IP协议的输入输出处理 |
| 10分 | 20分 | 20分 | 20分 | 20分 | 10分 |

此外，本实验需要验需要提交一份实验报告和编写的代码文件。报告内容除了实验原理、实验步骤等常规部分，还应当包括以下三个部分：

* 代码说明；
* 代码运行情况截图；
* 实验结果分析以及实验心得体会总结。

具体而言，实验报告评分标准如下：

1. 实验报告格式规范整洁（20分）；
2. 实验报告中详细记录了实验过程，在实验中所遇到的问题以及解决方法（40分）；
3. 实验报告中仔细分析了实验结果，并能提出自己的改进措施（40分）；
4. **实验报告中存在抄袭或雷同内容，实验报告记0分**。