

计算机网络协议编程实验指导书

1 IPv4 协议收发实验

1.1 实验目的

IPv4 协议是互联网的核心协议，它保证了网络节点（包括网络设备和主机）在网络层能够按照标准协议互相通信。IPv4 地址唯一标识了网络节点。在我们日常使用的计算机的主机协议栈中，IPv4 协议必不可少，它能够接收网络中传送给本机的分组，同时也能根据上层协议的要求将报文封装为 IPv4 分组发送出去。

本实验通过设计实现主机协议栈中的 IPv4 协议，让学生深入了解网络层协议的基本原理，学习 IPv4 协议基本的分组接收和发送流程。

另外，通过本实验，学生可以初步接触互联网协议栈的结构和计算机网络实验系统，为后面进行更为深入复杂的实验奠定良好的基础。

1.2 实验要求

根据计算机网络实验系统所提供的上下层接口函数和协议中分组收发的主要流程，独立设计实现一个简单的 IPv4 分组收发模块。要求实现的主要功能包括：

1) IPv4 分组的基本接收处理；

2) IPv4 分组的封装发送；

注：不要求实现 IPv4 协议中的选项和分片处理功能

1.3 实验内容

1) 实现 IPv4 分组的基本接收处理功能

对于接收到的 IPv4 分组，检查目的地址是否为本地地址，并检查 IPv4 分组头部中其它字段的合法性。提交正确的分组给上层协议继续处理，丢弃错误的分组并说明错误类型。

2) 实现 IPv4 分组的封装发送

根据上层协议所提供的参数，封装 IPv4 分组，调用系统提供的发送接口函数将分组发送出去。

1.4 实验帮助

在主机协议栈中，IPv4 协议主要承担辨别和标识源 IPv4 地址和目的 IPv4 地址的功能，一方面接收处理发送给自己的分组，另一方面根据应用需求填写目的地址并将上层报文封装发送。IPv4 地址可以在网络中唯一标识一台主机，因而在相互通信时填写在 IPv4 分组头部中的 IPv4 地址就起到了标识源主机和目的主机的作用。在后面 IPv4 分组的转发实验中，我们还将深入学习 IPv4 地址在分组转发过程中对选择转发路径的重要作用。

在两个主机端系统通信的环境中，网络的拓扑可以简化为两台主机直接相连，中间的具体连接方式可以抽象为一条简单的链路，如图 1.1 所示。IPv4 分组收发实验就是要在实验系统客户端的开发平台上，实现 IPv4 分组的接收和发送功能。

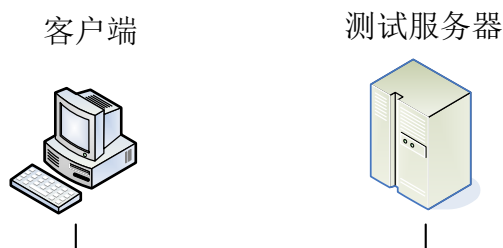


图 1.1 实验环境网络拓扑结构

1.4.1 处理流程

客户端接收到测试服务器发送来的 IPv4 分组后，调用接收接口函数 `stud_ip_recv()`（图 1.2 中接口函数 1）。学生需要在这个函数中实现 IPv4 分组接收处理的功能。接收处理完成后，调用接口函数 `ip_SendtoUp()` 将需要上层协议进一步处理的信息提交给上层协议（图 1.2 中接口函数 2）；或者调用函数 `ip_DiscardPkt()` 丢弃有错误的分组并报告错误类型（图 1.2 中函数 5）。

在上层协议需要发送分组时，会调用发送接口函数 `stud_ip_Upsend()`（图 1.2 中接口函数 3）。学生需要在这个函数中实现 IPv4 分组封装发送的功能。根据所传参数完成 IPv4 分组的封装，之后调用接口函数 `ip_SendtoLower()` 把分组交给下层完成发送（图 1.2 中接口函数 4）。

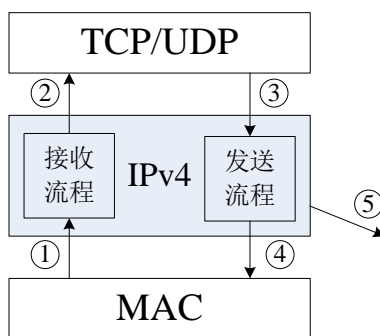


图 1.2 实验接口函数示意图

1.4.1.1 接收流程

在接口函数 `stud_ip_recv()` 中，需要完成下列处理步骤（仅供参考）：

- 1) 检查接收到的 IPv4 分组头部的字段，包括版本号（Version）、头长度（IP Head length）、生存时间（Time to live）以及头校验和（Header checksum）字段。对于出错的分组调用 `ip_DiscardPkt()` 丢弃，并说明错误类型。
- 2) 检查 IPv4 分组是否应该由本机接收。如果分组的目的地地址是本机地址或广播地址，则说明此分组是发送给本机的；否则调用 `ip_DiscardPkt()` 丢弃，并说明错误类型。
- 3) 如果 IPV4 分组应该由本机接收，则提取得到上层协议类型，调用 `ip_SendtoUp()` 接口函数，交给系统进行后续接收处理。

1.4.1.2 发送流程

在接口函数 `stud_ip_Upsend()` 中，需要完成下列处理步骤（仅供参考）：

- 1) 根据所传参数（如数据大小），来确定分配的存储空间的大小并申请分组的存储空间。
- 2) 按照 IPv4 协议标准填写 IPv4 分组头部各字段，标识符（Identification）字段可以使用一个随机数来填写。（注意：部分字段内容需要转换成网络字节序）

- 3) 完成 IPv4 分组的封装后，调用 ip_SendtoLower()接口函数完成后续的发送处理工作，最终将分组发送到网络中。

1.4.2 IPv4 分组头部格式

IPv4 分组头部格式如图 1.3 所示。

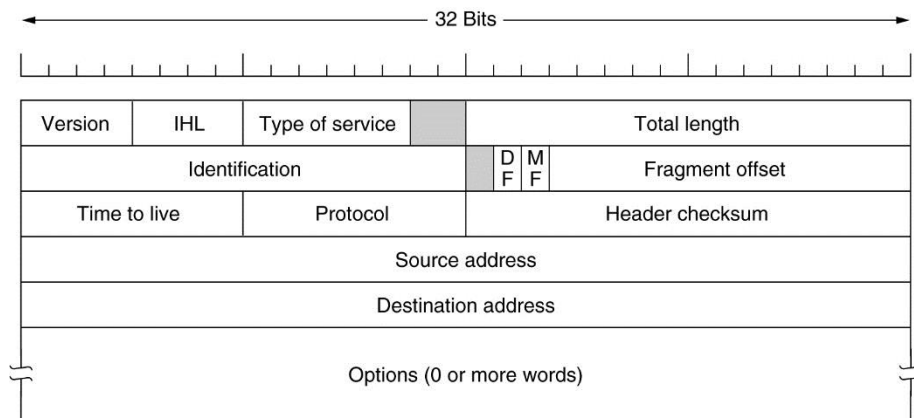


图 1.3 IPv4 分组头部格式

1.4.3 接口函数说明

1.4.3.1 需要实现的接口函数

1) 接收接口

```
int stud_ip_recv(char * pBuffer, unsigned short length)
```

参数：

pBuffer: 指向接收缓冲区的指针，指向 IPv4 分组头部

length: IPv4 分组长度

返回值：

0: 成功接收 IP 分组并交给上层处理

1: IP 分组接收失败

2) 发送接口

```
int stud_ip_Upsend(char* pBuffer, unsigned short len, unsigned int srcAddr, unsigned int dstAddr, byte protocol, byte ttl)
```

参数：

pBuffer: 指向发送缓冲区的指针，指向 IPv4 上层协议数据头部

len: IPv4 上层协议数据长度

srcAddr: 源 IPv4 地址

dstAddr: 目的 IPv4 地址

protocol: IPv4 上层协议号

ttl: 生存时间 (Time To Live)

返回值：

0: 成功发送 IP 分组

1: 发送 IP 分组失败

1.4.3.2 系统提供的接口函数

1) 丢弃分组

```
void ip_DiscardPkt(char * pBuffer, int type)
```

参数：

pBuffer: 指向被丢弃分组的指针

type: 分组被丢弃的原因, 可取以下值:

STUD_IP_TEST_CHECKSUM_ERROR	IP 校验和出错
STUD_IP_TEST_TTL_ERROR	TTL 值出错
STUD_IP_TEST_VERSION_ERROR	IP 版本号错
STUD_IP_TEST_HEADLEN_ERROR	头部长度错
STUD_IP_TEST_DESTINATION_ERROR	目的地址错

2) 发送分组

void ip_SendtoLower(char *pBuffer ,int length)

参数:

pBuffer: 指向待发送的 IPv4 分组头部的指针

length: 待发送的 IPv4 分组长度

3) 上层接收

void ip_SendtoUp(char *pBuffer, int length)

参数:

pBuffer: 指向要上交的上层协议报文头部的指针

length: 上交报文长度

4) 获取本机 IPv4 地址

unsigned int getIpv4Address()

参数: 无

2 IPv4 协议转发实验

2.1 实验目的

通过前面的实验，我们已经深入了解了 IPv4 协议的分组接收和发送处理流程。本实验需要将实验模块的角色定位从通信两端的主机转移到作为中间节点的路由器上，在 IPv4 分组收发处理的基础上，实现分组的路由转发功能。

网络层协议最为关注的是如何将 IPv4 分组从源主机通过网络送达目的主机，这个任务就是由路由器中的 IPv4 协议模块所承担。路由器根据自身所获得的路由信息，将收到的 IPv4 分组转发给正确的下一跳路由器。如此逐跳地对分组进行转发，直至该分组抵达目的主机。IPv4 分组转发是路由器最为重要的功能。

本实验设计模拟实现路由器中的 IPv4 协议，可以在原有 IPv4 分组收发实验的基础上，增加 IPv4 分组的转发功能。对网络的观察视角由主机转移到路由器中，了解路由器是如何为分组选择路由，并逐跳地将分组发送到目的主机。本实验中也会初步接触路由表这一重要的数据结构，认识路由器是如何根据路由表对分组进行转发的。

2.2 实验要求

在前面 IPv4 分组收发实验的基础上，增加分组转发功能。具体来说，对于每一个到达本机的 IPv4 分组，根据其目的 IPv4 地址决定分组的处理行为，对该分组进行如下的几类操作：

- 1) 向上层协议上交目的地址为本机地址的分组；
- 2) 根据路由查找结果，丢弃查不到路由的分组；
- 3) 根据路由查找结果，向相应接口转发不是本机接收的分组。

2.3 实验内容

实验内容主要包括：

- 1) 设计路由表数据结构。

设计路由表所采用的数据结构。要求能够根据目的 IPv4 地址来确定分组处理行为（转发情况下需获得下一跳的 IPv4 地址）。路由表的数据结构和查找算法会极大的影响路由器的转发性能，有兴趣的同学可以深入思考和探索。

- 2) IPv4 分组的接收和发送。

对前面实验（IPv4 协议收发实验）中所完成的代码进行修改，在路由器协议栈的 IPv4 模块中能够正确完成分组的接收和发送处理。具体要求不做改变，参见“IPv4 协议收发实验”。

- 3) IPv4 分组的转发。

对于需要转发的分组进行处理，获得下一跳的 IP 地址，然后调用发送接口函数做进一步处理。

2.4 实验帮助

分组转发是路由器最重要的功能。分组转发的依据是路由信息，以此将目的地址不同的分组发送到相应的接口上，逐跳转发，并最终到达目的主机。本实验要求按照路由器协议栈的 IPv4 协议功能进行设计实现，接收处理所有收到的分组（而不只是目的地址为本机地址的分组），并根据分组的 IPV4 目的地址结合相关的路由信息，对分组进行转发、接收或丢弃操作。

实验的主要流程和系统接口函数与前面“IPv4 协议收发实验”基本相同。在下层接收接口函数 `Stud_fwd_deal()` 中（图 2.1 中接口函数 1），实现分组接收处理。主要功能是根据分组中目的 IPv4 地址结合对应的路由信息对分组进行处理。分组需要上交，则调用接口函数 `Fwd_LocalRcv()`（图 2.1 中接口函数 2）；需要丢弃，则调用函数 `Fwd_DiscardPkt()`（图 2.1 中函数 5）；需要转发，则进行转发操作。转发操作的实现要点包括，TTL 值减 1，然后重新计算头校验和，最后调用发送接口函数 `Fwd_SendtoLower()`（图 2.1 中接口函数 4）将分组发送出去。注意，接口函数 `Fwd_SendtoLower()` 比前面实验增加了一个参数 `pNxtHopAddr`，要求在调用时传入下一跳的 IPv4 地址，此地址是通过查找路由表得到的。

另外，本实验增加了一个路由表配置的接口（图 2.1 中函数 6），要求能够根据系统所给信息来设定本机路由表。实验中只需要简单地设置静态路由信息，以作为分组接收和发送处理的判断依据，而路由信息的动态获取和交互，在有关路由协议的实验（RIP 协议）中会重点涉及。

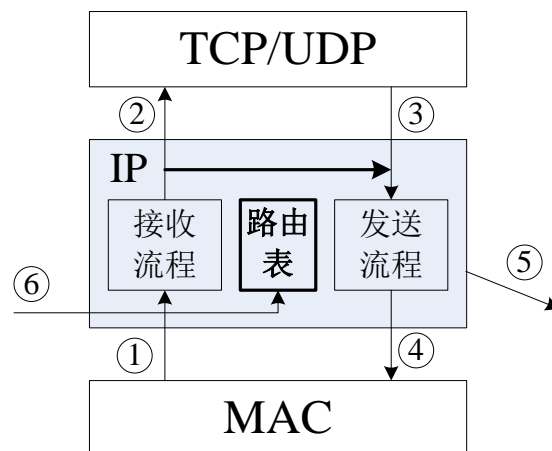


图 2.1 IPv4 分组转发实验接口示意图

与前面 IPv4 协议收发实验不同的是，在本实验中分组接收和发送过程中都需要引入路由表的查找步骤。路由器的主要任务是进行分组转发，它所接收的多数分组都是需要进行转发的，而不像主机协议栈中 IPv4 模块只接收发送给本机的分组；另外，路由器也要接收处理发送给本机的一些分组，如路由协议的分组（RIP 实验中会涉及到）、ICMP 分组等。如何确定对各种分组的处理操作类型，就需要根据分组的 IPV4 目的地址结合路由信息进行判断。

一般而言，路由信息包括地址段、距离、下一跳地址、操作类型等。在接收到 IPv4 分组后，要通过其目的地址匹配地址段来判断是否为本机地址，如果是则本机接收；如果不是，则通过其目的地址段查找路由表信息，从而得到进一步的操作类型，转发情况下还要获得下一跳的 IPv4 地址。发送 IPv4 分组时，也要拿目的地址来查找路由表，得到下一跳的 IPv4 地址，然后调用发送接口函数做进一步处理。在前面实验中，发送流程中没有查找路由表来确定下一跳地址的步骤，这项工作由系统来完成了，在本实验中则作为实验内容要求学生实现。需要进一步说明的是，在转发路径中，本路由器可能是路径上的最后一跳，可以直接转发给目的主机，此时下一跳的地址就是 IPv4 分组的目的地址；而非最后一跳的情况下，下一跳的地址是从对应的路由信息中获取的。因此，在路由表中转发类型要区分最后一跳和非最后一跳的情况。

路由表数据结构的设计是非常重要的，会极大地影响路由表的查找速度，进而影响路由器的分组转发性能。本实验中虽然不会涉及大量分组的处理问题，但良好且高效的数据结构无疑会为后面的实验奠定良好的基础。链表结构是最简单的，但效率比较低；树型结构的查找效率会提高很多，但组织和维护有些复杂，可以作为提高的要求。具体数据结构的设计，可以在实践中进一步深入研究。

2.4.1 路由表维护

需要完成下列分组接收处理步骤：

- 1) `stud_Route_Init()`函数中，对路由表进行初始化。
- 2) `stud_route_add()`函数中，完成路由的增加。

2.4.2 转发处理流程

在 `stud_fwd_deal()`函数中，需要完成下列分组接收处理步骤：

- 1) 查找路由表。根据相应路由表项的类型来确定下一步操作，错误分组调用函数 `fwd_DiscardPkt()`进行丢弃，上交分组调用接口函数 `fwd_LocalRcv()`提交给上层协议继续处理，转发分组进行转发处理。注意，转发分组还要从路由表项中获取下一跳的 IPv4 地址。
- 2) 转发处理流程。对 IPv4 头部中的 TTL 字段减 1，重新计算校验和，然后调用下层接口 `fwd_SendtoLower()`进行发送处理。

2.4.3 实验接口函数

本小节列出了实验时会用到的各种接口函数，其中有一些函数是需要学生来完成的，有一些函数则是已经实现好了，供学生在实验时直接使用的。表 2.1 列出了所有的接口函数及其简要说明，更详细的说明在后面描述。

函数名	说明	是否需要学生实现
<code>stud_fwd_deal</code>	系统处理收到的 IP 分组的函数，当接收到一个 IP 分组的时候，实验系统会调用该函数进行处理	是
<code>stud_route_add</code>	向路由表添加路由的函数。系统将调用该函数向路由表添加一条 IPv4 路由。	是
<code>fwd_SendtoLower</code>	将封装完成的 IP 分组通过链路层发送出去的函数。	否
<code>fwd_LocalRcv</code>	将 IP 分组上交本机上层协议的函数，即当分组需要上交上层函数的时候调用本函数。	否
<code>fwd_DiscardPkt</code>	丢弃 IP 分组的函数。当需要丢弃一个 IP 分组的时候调用。	否
<code>stud_BestRoute</code>	根据目的 IPv4 地址查找路由表获得下一跳的 IPv4 地址	否
<code>stud_Route_Init</code>	路由表初始化函数，系统初始化的时候将调用此函数对路由表进行初始化操作。	否
<code>getIpv4Address</code>	获取本机的 IPv4 地址，用来判断分组地址和本机地址是否相同	否

表 2.1 函数接口表

1) `stud_fwd_deal()`

`int stud_fwd_deal(char * pBuffer, int length)`

参数：

`pBuffer`：指向接收到的 IPv4 分组头部

`length`：IPv4 分组的长度

返回值：

0 为成功, 1 为失败;

说明:

本函数是 IPv4 协议接收流程的下层接口函数, 实验系统从网络中接收到分组后会调用本函数。调用该函数之前已完成 IP 报文的合法性检查, 因此学生在本函数中应该实现如下功能:

- a. 判定是否为本机接收的分组, 如果是则调用 `fwd_LocalRcv()`;
- b. 按照最长匹配查找路由表获取下一跳, 查找失败则调用 `fwd_DiscardPkt()`;
- c. 调用 `fwd_SendtoLower()` 完成报文发送;
- d. 转发过程中注意 TTL 的处理及校验和的变化;

2) `fwd_LocalRcv()`

`void fwd_LocalRcv(char *pBuffer, int length)`

参数:

`pBuffer`: 指向分组的 IP 头

`length`: 表示分组的长度

说明:

本函数是 IPv4 协议接收流程的上层接口函数, 在对 IPv4 的分组完成解析处理之后, 如果分组的目的地址是本机的地址, 则调用本函数将正确分组提交上层相应协议模块进一步处理。

3) `fwd_SendtoLower()`

`void fwd_SendtoLower(char *pBuffer, int length, unsigned int nexthop)`

参数:

`pBuffer`: 指向所要发送的 IPv4 分组头部

`length`: 分组长度 (包括分组头部)

`nexthop`: 转发时下一跳的地址。

说明:

本函数是发送流程的下层接口函数, 在 IPv4 协议模块完成发送封装工作后调用该接口函数进行后续发送处理。其中, 后续的发送处理过程包括分片处理、IPv4 地址到 MAC 地址的映射 (ARP 协议)、封装成 MAC 帧等工作, 这部分内容不需要学生完成, 由实验系统提供支持。

4) `fwd_DiscardPkt()`

`void fwd_DiscardPkt(char *pBuffer, int type)`

参数:

`pBuffer`: 指向被丢弃的 IPV4 分组头部

`type`: 表示错误类型, 包括 TTL 错误和找不到路由两种错误, 定义如下:

`STUD_FORWARD_TEST_TTLERROR`

`STUD_FORWARD_TEST_NOROUTE`

说明:

本函数是丢弃分组的函数, 在接收流程中检查到错误时调用此函数将分组丢弃。

5) `stud_route_add()`

`void stud_route_add(stud_route_msg *proute)`

参数:

`proute`: 指向需要添加路由信息的结构体头部, 其数据结构 `stud_route_msg` 的定义如下:

```
typedef struct stud_route_msg
{
```



```
        unsigned int  dest;  
        unsigned int  masklen;  
        unsigned int  nexthop;  
    } stud_route_msg;
```

说明：

本函数为路由表配置接口，系统在配置路由表时需要调用此接口。此函数功能为向路由表中增加一个新的表项，将参数所传递的路由信息添加到路由表中。

6) stud_BestRoute ()

unsigned int stud_BestRoute(unsigned int dest)

参数：

dest: 目的 IPv4 地址

返回值：

转发时下一跳的地址

7) stud_Route_Init()

void stud_Route_Init()

参数：

无

说明：

本函数将在系统启动的时候被调用，学生可将初始化路由表的代码写在这里。

8) getIpv4Address()

UINT32 getIpv4Address()

说明：

本函数用于获取本机的 IPv4 地址，学生调用该函数即可返回本机的 IPv4 地址，可以用来判断 IPV4 分组是否为本机接收。

返回值：

本机 IPv4 地址

除了以上的函数以外，学生可根据需要自己编写一些实验需要的函数和数据结构，包括路由表的数据结构，对路由表的搜索、初始化等操作函数。