IPV6 转发实验报告

一、实验目的

本实验主要帮助理解 IPV6 的分组转发过程，实现路由器中的 IPV6 协议模块。本实验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 IPv6 分组选择路由，并逐跳地将 IPv6 分组转发到目的端的。

二、实验要求

本实验要求在前面 IPv6 分组收发实验的基础上，增加分组转发功能。对于每一个到达本机的 IPv6 分组，根据其目的 IPv6 地址查找本机的路由表，对该分组进行如下的几类操作：<1>丢弃查不到路由的分组；<2>向上层协议上交目的地址为本机地址的分组；<3>根据路由查找结果，向相应接口转发路由匹配成功的分组。

为完成本实验的实验要求，需要完成以下三项内容：

（1）设计路由表数据结构

要求能够根据 IPv6 地址来确定分组处理行为，转发情况下需获得下一跳的 IPv6 地址。

（2）IPv6 分组的接收和发送

对 IPv6 收发实验中完成的代码进行修改，在路由器协议栈的 IPv6 模块中能够正确完成分组的接收和发送处理。

（3）IPv6 分组的转发

对于需要转发的分组进行处理，获得下一跳的 IPv6 地址，然后调用发送接口函数进一步处理。

三、实验接口

（1）需要实现的接口函数说明：

本实验中需要实现的接口函数包括路由初始化函数、路由添加函数和转发处理函数。

1）路由初始化函数：

```
void stud_ipv6_Route_Init()
```

说明：本函数用于对路由表进行初始化，在系统初始化的时候将调用该函数，对路由表数据结构进行初始化操作。

2）路由添加函数：

```
void stud_ipv6_route_add(stud_ipv6_route_msg *proute)
```

参数：

proute：需要添加的路由，其数据结构 stud_ipv6_route_msg 的定义如下：

```
typedef     std_route_msg
{
    ipv6_addr    dest;
    UINT32    masklen;
    ipv6_addr     nexthop;
}stud_ipv6_route_msg;
```

说明：系统在配置路由表时需要调用此接口。本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。

3）转发处理函数：

int stud_ipv6_fwd_deal(char * pBuffer, int length)

参数：

pBuffer：指向所接收到的 IPv6 分组。

length：为 IPv6 分组的长度。

返回值：

0 为成功，-1 为失败。

说明：本函数是 IPv6 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 IPv6 分组的合法性检查，因此本函数应该考虑实现如下功能：

<1>判定是否为发给本机的分组，如果是，则调用 ipv6_fwd_LocalRcv().

<2>按照最长匹配原则查找路由表，获取下一跳 IPv6 地址。查找失败，则调用 ipv6_fwd_DiscardPkt().

<3>查找成功，则调用 ipv6_fwd_SendtoLower()，完成分组发送。

<4>转发过程中注意对 Hop Limit 的处理。

四、实验实现

```
#include "sysinclude.h"
#include <iostream>
#include <bitset>
#include <list>
using namespace std;
extern void ipv6_fwd_DiscardPkt(char *pBuffer, int type);
extern void ipv6_fwd_SendtoLower(char *pBuffer, int length, ipv6_addr *nexthop);
extern void getIpv6Address(ipv6_addr *pAddr);
extern void ipv6_fwd_LocalRcv(char *pBuffer, int length);

list<stud_ipv6_route_msg> rv6;

void stud_ipv6_Route_Init()
{
    rv6.clear();
    return;
}

void stud_ipv6_route_add(stud_ipv6_route_msg *proute)
{
    rv6.push_back(*proute);
    return;
}

int stud_ipv6_fwd_deal(char *pBuffer, int length)
{
```

```
ipv6_addr temp;
ipv6_addr* pAddr = &temp;
getIpv6Address(pAddr);


unsigned int* mypAddr = (unsigned int *) pAddr;
unsigned int* destpAddr = (unsigned int *) (pBuffer+24);
UINT8 *ttl = (UINT8 *) (pBuffer + 7);
if (*ttl < 100)
{
    ipv6_fwd_DiscardPkt(pBuffer, STUD_IPV6_FORWARD_TEST_HOPLIMIT_ERROR);
    return -1;
}
bitset<128> myIP;
bitset<128> destIP;

destIP = bitset<128>(destpAddr[0]) << 96 | bitset<128>(destpAddr[1]) << 64 | bitset<128>
(destpAddr[2]) << 32 | bitset<128> (destpAddr[3]);
myIP = bitset<128>(mypAddr[0]) << 96 | bitset<128>(mypAddr[1]) << 64 | bitset<128>
(mypAddr[2]) << 32 | bitset<128> (mypAddr[3]);


if (destIP == myIP)
{
    ipv6_fwd_LocalRcv(pBuffer, length);
    return 0;
}


int MAXLEN = 0; list<stud_ipv6_route_msg>::iterator pos = rv6.end();
for (list<stud_ipv6_route_msg>::iterator iter = rv6.begin(); iter != rv6.end(); ++iter)
{

    if (iter->masklen < MAXLEN) continue;

    unsigned int *routeAddr = (unsigned int *)&(iter->dest);
    bitset<128> nipA((unsigned long)routeAddr[0]); bitset<128> nipB((unsigned
long)routeAddr[1]); bitset<128> nipC((unsigned long)routeAddr[2]); bitset<128> nipD((unsigned
long)routeAddr[3]);
    bitset<128> nextIP = nipA << 96 | nipB << 64 | nipC << 32 | nipD;

    int j = 0;
    for (j = 0; j < iter->masklen; ++j)
    {
```

```
                if (nextIP[j] != destIP[j]) break;
            }
            if (j == iter->masklen)
            {
                MAXLEN = iter->masklen;
                    pos = iter;
            }
        }
        if (pos == rv6.end())
        {
            ipv6_fwd_DiscardPkt(pBuffer, STUD_IPV6_FORWARD_TEST_NOROUTE);
            return -1;
        }
        *ttl -= 1;
        if (*ttl == 1)
        {
            ipv6_fwd_SendtoLower(pBuffer, length, &(((IPv6Head *)(pBuffer))->destAddr));
            return 0;
        }
        if (*ttl > 1)
            ipv6_fwd_SendtoLower(pBuffer, length, &(pos->nexthop));
        return 0;
}
```

五、实验心得与体会

通过本次实验，我更加清晰地认识了网络原理课程中的 IP 包转发的过程，对最长匹配原则也有了一个清晰的理解。在实验中还有一些细节问题不太清楚，感谢助教的热心帮助。