计算机网络编程
实验报告

班级：07111707

组长：1120171189 崔程远

成员：1120172149 吴沁璇

1120172153 张澈

1120172163 王晓媛

1120172736 张鉴昊

1120172765 曾煜瑾

1120173326 曾紫飞

北京理工大学

计算机学院

2020 年 5 月

第五章 实验 5　路由表查找程序

1. 实验目的

加深对根据 IP 地址查找路由表进行路由转发的掌握和理解。

2. 实验内容

根据输入的 IP 数据报的目的地址，查找路由表，得到该数据报的下一跳。在多行匹配时选最长前缀匹配作为下一跳。路由表的数据结构采用简单的线性表—结构数组，IP 地址和子网掩码内部存储二进制值，不要存储字符串。

配置文件关键要点：

路由表结构为：地址/前缀 下一跳。其中 0/0 代表默认路由，即没有匹配时转发的地址

RoutingTable=

135.46.56.0/22　Interface0

136.46.60.0/22　Interface1

192.53.40.0/23　Router1

0/0　Router2

程序运行屏幕输出要点：

屏幕提示输入 IP 数据报的目的 IP 地址

显示当前路由表的信息

显示每行匹配与否的计算情况

输出该数据报的下一跳

3. 实验原理

路由器主要完成两个功能：一是分组转发，二是路由计算。前者处理通过路由表的数据流，关键操作是转发表查询、转发及相关的队列管理金和任务调度等；后者通过和其他路由器进行基于路由协议的交互，完成路由表的计算。

转发是从路由表得出的，其表项和路由表项有直接的对应关系。但转发表的格式和路由表的格式不同，其结构应使查找过程最优化，而路由表则需对网络拓扑变化的计算最优化。转发表中含有一个分组将要发往的目的地址，以及分组的下一跳，即下一步接收者的目的地址，实际为 MAC 地址。为了减少转发表的重复项目，可以使用一个默认路由代替所有具有相同“下一跳”的项目，并将默认路由设置得比其他项目的优先级低。

当数据包到达时，路由器会查看该数据包的目标地址，并检查它属于哪个子网。具体做法是：路由器把数据包的目标地址与每个子网的掩码进行 AND 操作，看结果是否对应于某个前缀。当可能有多个具有不同前缀的表项得到匹配，在这种情况下，使用具有最长前缀的表项。

4. 实验环境

语言	集成开发环境	编译器
C++	Visual Studio 2017	gcc version 4.8.1
Java	Eclipse 2019	java version "1.8.0_65"
Python	Pycharm 2017	Python 3.7.0

5. 实验步骤

题目要求 IP 地址和子网掩码内部存储二进制值，不要存储字符串，所以只能用 int 数组保存 01 二进制，之后转换成十进制整数进行 AND 操作。三份代码的结构和输出完全一致，均采用面向对象的构造方法，以 C++代码为例进行分析：

• 定义全局变量

```
private:
    int IPAddress[n][4]; //IP 地址的二进制整数分段
    int subnetMask[n][4]; //子网掩码的二进制整数分段
    string nextHop[n] = { "Interface0", "Interface1", "Router1", "Router2" }; //
下一跳
    int IPSubAddress[n][4] = { { 135, 46, 56, 0 }, { 136, 46, 60, 0 }, { 192, 53,
40, 0 }, { 0, 0, 0, 0 } }; //IP 地址的十进制整数分段
    int prefix[n] = { 22, 22, 23, 0 }; //前缀
    string subIPAddress[4]; //目的 IP 地址的字符串分段
```

• 初始化 IPAddress 和 subnetMask

```
void Initialize()
{
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < 4; j++)
        {
            IPAddress[i][j] = atoi(getBinaryString(IPSubAddress[i][j]).c_str());
        }
        string s = "";
        for (int j = 0; j < prefix[i]; j++)
        {
            s += "1";
        }
        for (int j = prefix[i]; j < 32; j++)
        {
            s += "0";
        }
        for (int j = 0; j < 4; j++)
        {
            subnetMask[i][j] = atoi(s.substr(j * 8, 8).c_str());
        }
    }
}
```

• 打印路由表

```
void printRoutingTable()
    {
        cout << "Routing Table: " << endl;
        printf("IPAddress/Prefix\tNextHop\n");
        for (int i = 0; i < n; i++)
        {
            if (i == n - 1)
            {
                printf("0/0\t\t\t");
            }
```

```
            else
            {
                printf("%d", IPSubAddress[i][0]);
                for (int j = 1; j < 4; j++)
                {
                    printf(".%d", IPSubAddress[i][j]);
                }
                printf("/%d\t\t", prefix[i]);
            }
            cout << nextHop[i] << endl;
        }
        cout << endl;
    }
```

• 将输入的 IP 地址分成四段

```
//赋值到subIPAddress数组
void Split(string s)
{
    int num = 0;
    int pos1 = 0;
    int pos2 = 0;
    while (true)
    {
        pos2 = s.find(".", pos1);
        if (pos2 == s.npos)
        {
            break;
        }
        subIPAddress[num] = s.substr(pos1, pos2 - pos1);
        pos1 = pos2 + 1;
        num++;
    }
    subIPAddress[num] = s.substr(pos1);
}
```

• 查询路由表

```
void Search(string s)
    {
        Split(s);
        int maxPrefixPos = n;
        int maxPrefixLen = -1;
        for (int i = 0; i < n; i++)
        {
            bool flag = true;
            for (int j = 0; j < 4; j++)
            {
                int value1 = strtol(to_string(subnetMask[i][j]).c_str(), NULL, 2);
                int value2 = strtol(subIPAddress[j].c_str(), NULL, 10);
                int value3 = strtol(to_string(IPAddress[i][j]).c_str(), NULL, 2);
                if ((value1 & value2) != value3)
```

```
                {
                    flag = false;
                    break;
                }
            }
            if (flag == false)
            {
                printf("Searching line %d, not matched.\n", i);
            }
            else if (flag == true)
            {
                printf("Searching line %d, matched successfully.\n", i);
                if (prefix[i] > maxPrefixLen)
                {
                    maxPrefixLen = prefix[i];
                    maxPrefixPos = i;
                }
            }
        }
        cout << "Next hop: " << nextHop[maxPrefixPos] << endl;
    }
```

- 主函数

```
int main()
{
    cout << "Please input Destination IPAddress of datagram as format
'num1.num2.num3.num4' such as '192.53.41.128'(no ''):" << endl;
    string s;
    cin >> s;
    cout << endl;
    SearchRoutingTable operation;
    operation.Initialize();
    operation.printRoutingTable();
    operation.Search(s);
    system("pause");
}
```

6. 实验结果

- C++

```
D:\desktop\C++\x64\Debug\C++.exe
Please input Destination IPAddress of datagram as format 'num1.num2.num3.num4'
192.53.41.28

Routing Table:
IPAddress/Prefix        NextHop
135.46.56.0/22          Interface0
136.46.60.0/22          Interface1
192.53.40.0/23          Router1
0/0                     Router2

Searching line 0, not matched.
Searching line 1, not matched.
Searching line 2, matched successfully.
Searching line 3, matched successfully.
Next hop: Router1
请按任意键继续. . .
```

• Java

```
Console  Debug Shell  Problems
<terminated> SearchRoutingTable [Java Application] E:\Program Software\java-2019-09\eclipse\jre\
Please input Destination IPAddress of datagram as format 'num1.num2.num3.num4'
192.53.41.128

Routing Table:
IPAddress/Prefix        NextHop
135.46.56.0/22          Interface0
136.46.60.0/22          Interface1
192.53.40.0/23          Router1
0/0                     Router2

Searching line 0, not matched.
Searching line 1, not matched.
Searching line 2, matched successfully.
Searching line 3, matched successfully.
Next hop: Router1
```

• Python

```
D:\desktop\Python>python SearchRoutingTable.py
Please input Destination IPAddress of datagram as format 'num1.num2.num3.num4'
192.53.41.128

Routing Table:
IPAddress/Prefix        NextHop
135.46.56.0/22          Interface0
136.46.60.0/22          Interface1
192.53.40.0/23          Router1
0/0                     Router2

Searching line 0, not matched.
Searching line 1, not matched.
Searching line 2, matched successfully.
Searching line 3, matched successfully.
Next hop: Router1
请按任意键继续. . .
```

7. 实验总结

这个实验也比较简单，首先计算出路由表每行的 IP 地址和子网掩码并保存好，之后根据用户输入的目的 IP 地址，将该地址与每个子网的掩码进行 AND 操作，看结果是否对应于某个前缀，如果四段均匹配，则成功匹配，如果出现多行匹配选择最长前缀匹配作为下一跳。然而这道题要求 IP 地址和子网掩码内部存储二进制值，这就麻烦了许多，要用 32 位 int 数组存储每个 IP 地址和子网掩码，我是将其分成四段，用二维 int 数组存储，这样更直观些，之后进行字符串和二进制整数、十进制整数的转换，需要调用相关函数。