



中山大學  
SUN YAT-SEN UNIVERSITY

虚拟路由  
项目报告  
模拟网络层路由

学院：数据科学与计算机学院

专业：计算机科学与技术

年级：**2016** 级

组长（学号）：王锡准（**16337236**）

组员（学号）：杨陈泽（**16337271**）

组员（学号）：肖遥（**16337258**）

# 目录

|          |                   |           |
|----------|-------------------|-----------|
| <b>1</b> | <b>项目介绍</b>       | <b>3</b>  |
| <b>2</b> | <b>基本 RIP 协议</b>  | <b>3</b>  |
| 2.1      | 协议概述              | 3         |
| 2.2      | 算法                | 3         |
| 2.3      | 协议特性              | 3         |
| 2.3.1    | 协议流程              | 3         |
| 2.4      | 报文格式              | 4         |
| 2.5      | 协议扩展              | 6         |
| 2.6      | 协议实现              | 6         |
| 2.7      | 结果                | 6         |
| 2.8      | 总结                | 7         |
| <b>3</b> | <b>基本 OSPF 协议</b> | <b>7</b>  |
| 3.1      | 协议概述              | 7         |
| 3.2      | 算法                | 8         |
| 3.2.1    | 路由计算算法            | 8         |
| 3.2.2    | 广播算法              | 8         |
| 3.3      | 协议特性              | 8         |
| 3.3.1    | 协议流程              | 8         |
| 3.4      | 报文格式              | 8         |
| 3.5      | 协议扩展              | 10        |
| 3.6      | 协议实现              | 10        |
| 3.7      | 结果                | 10        |
| 3.8      | 总结                | 11        |
| <b>4</b> | <b>基本中心化路由协议</b>  | <b>11</b> |
| 4.1      | 协议概述              | 11        |
| 4.2      | 算法                | 12        |
| 4.3      | 协议特性              | 12        |
| 4.3.1    | 协议流程              | 12        |
| 4.4      | 报文格式              | 12        |
| 4.5      | 协议扩展              | 13        |
| 4.6      | 协议实现              | 13        |
| 4.7      | 结果                | 13        |
| 4.8      | 总结                | 15        |

---

|          |                       |           |
|----------|-----------------------|-----------|
| <b>5</b> | <b>安装与部署</b>          | <b>16</b> |
| <b>6</b> | <b>项目管理记录</b>         | <b>16</b> |
| 6.1      | 管理概述 . . . . .        | 16        |
| 6.2      | 阶段管理 . . . . .        | 16        |
| 6.3      | 人员分工 . . . . .        | 16        |
| <b>7</b> | <b>总结</b>             | <b>16</b> |
| 7.1      | 关于在应用层模拟网络层 . . . . . | 16        |
| <b>A</b> | <b>参考文献</b>           | <b>17</b> |

## 1 项目介绍

模拟网络层路由项目包含三个部分：

1. 基本 RIP 协议
2. 基本 OSPF 协议
3. 基本中心化路由协议实现

在其中的每一个部分中，都实现了基本的路由功能以及对于基本算法缺点和不足的改进，比如，在基本 RIP 协议中，使用 `split-horizon`、`poison-inverse`，`holddown` 方法和 `triggered update` 方法缓解了无穷计数问题；在基本 OSPF 协议中，实现了多条等费用路径的路由选择，并且解决了路由循环问题和改善了路由震荡问题；在中心化路由协议中使用 `Twisted` 异步编程框架以减轻服务器进程的资源消耗；等等。项目中 3 个部分使用的都是 UDP 协议以传输路由信息，使用 UDP 也节约了网络资源。

项目主页见<https://github.com/Leo-xh/Virtual-Routing>。

## 2 基本 RIP 协议

### 2.1 协议概述

本项目中实现的 RIP 协议采用了 RIPv1 和 RIPv2 的协议规定的一部分，是一个基于 DV 算法的网络层路由协议。其主要算法是 Bellman-Ford 算法，路由费用以路由跳数作为度量，运行该协议的每个路由器能够计算出到其余路由器的下一跳路由以及路由费用。定义了请求报文和响应报文两种报文格式。为了缓解无穷计数问题，使用 `split-horizon`、`poison-inverse`，`holddown` 方法和 `triggered update` 方法。同时使用了 `flush` 计时器进一步缓解无穷计数问题。在协议的扩展中，实现了该协议下适用的 `traceRoute` 功能。

### 2.2 算法

### 2.3 协议特性

#### 2.3.1 协议流程

本项目中实现的 RIP 协议主要包含的流程如下：

- 路由器在进入区域时，向所有邻居发送 `request` 报文，邻居则返回一个 `response` 报文，里面含有邻居的距离矢量表。
- 该路由器会定时地向所有的邻居发送 `response` 报文，即就是发送自己的距离矢量表。

- 收到一个 **response** 报文意味着要维持该邻居的状态；而如果在一段时间内没有收到某个邻居路由器的 **response** 报文，则认为该路由器故障或离开该区域。
- 当收到其它邻居路由器的距离矢量表时，根据该信息更新自己的距离矢量表；当自己的距离矢量更新时，向所有邻居发送自己新的距离矢量表。

## 2.4 报文格式

本实验中使用的普通报文普通数据报文、**request** 报文、**response** 报文、TraceRoute 报文、Echo 报文如下：

普通报文：

|                     |
|---------------------|
| Command             |
| source address      |
| destination address |
| payload             |

**request** 报文：

|                |           |                |
|----------------|-----------|----------------|
| Command        | Version   | Routing domain |
| Source address |           |                |
| Address family | Route tag |                |
| address        |           |                |
| Next hop       |           |                |
| metric         |           |                |

**Response** 报文：

|                         |           |                |
|-------------------------|-----------|----------------|
| Command                 | Version   | Routing domain |
| Source address          |           |                |
| Address family          | Route tag |                |
| address                 |           |                |
| Next hop                |           |                |
| metric                  |           |                |
| repeat of last 17 bytes |           |                |
| ...                     |           |                |

**TraceRoute** 报文：

|                     |
|---------------------|
| Command             |
| source address      |
| destination address |
| count               |

Echo 报文：

|                     |
|---------------------|
| Command             |
| source address      |
| destination address |
| local address       |

报文项解释：

1. Command (1 Byte)：指明这条报文的类型。
  - 0：普通报文。
  - 1：请求报文。
  - 2：响应（包含路由表）报文。
  - 3：TraceRoute 报文。
  - 4：Echo 报文。
2. Version (1 Byte)：RIP 协议的版本。
3. Routing domain (2 Bytes)：指明这是 RIP 中的哪一步。
4. Source address (6 Bytes)：指明发送方的源地址和监听端口。
5. Address family (2 Bytes)：使用什么作为地址，使用 IP 该项为 2。
6. Route tag (2 Bytes)：本项目中不用。
7. address (6 Bytes)：ip 和端口号。
8. Next hop (6 Bytes)：下一跳路由的 ip 和端口。
9. metric (1 Bytes)：代价度量（跳数）。
10. Local address (6 Bytes)：指明 Echo 报文发送者的源地址和监听端口。
11. Count(1 Bytes)：路由跳数。

## 2.5 协议扩展

### 1. 网络环路

路由器可能收到自己发送的路由信息，因此造成网络环路。

为了避免该情况，采用了：水平分割、毒性逆转、抑制定时器和触发更新机制。

### 2. 无穷计数

采用 flush 计时器，在检测到一个邻居可能失效时不立即将其从数据库中删除，等待一个间隔后再删除，目的是通告其它邻居该失效邻居不可达。

### 3. TraceRoute

## 2.6 协议实现

本 RIP 协议基本由 Python 实现，主要利用的是 socket 库进行套接字编程并采用的是 UDP 协议、struct 库进行数据的打包和解包、threading 库管理线程和定时器。

实现方法实现了一个 RIP 协议类，处理内部流程和为外部调用留出了收发数据、查询路由路径等接口。

## 2.7 结果

测试所用的拓扑图以及路由结果如下图1和2所示。

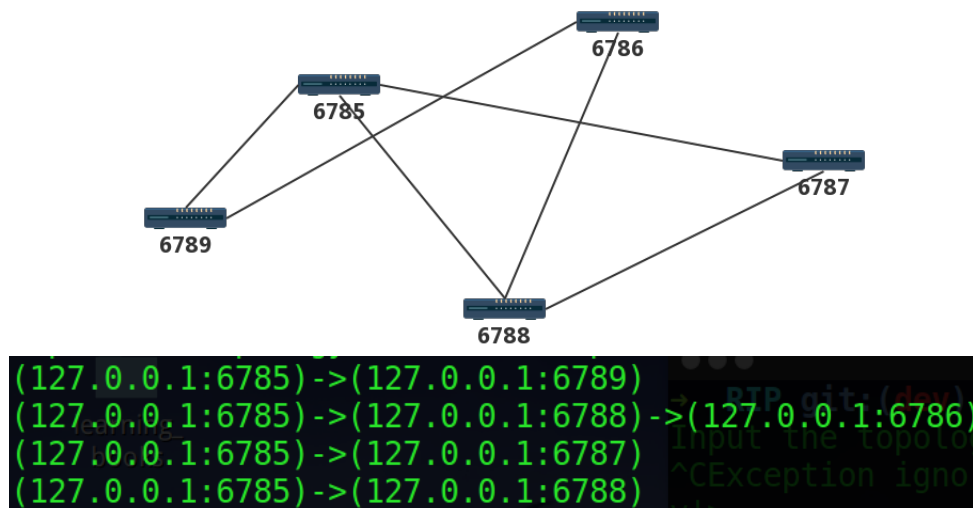


图 1: RIP 测试样例一

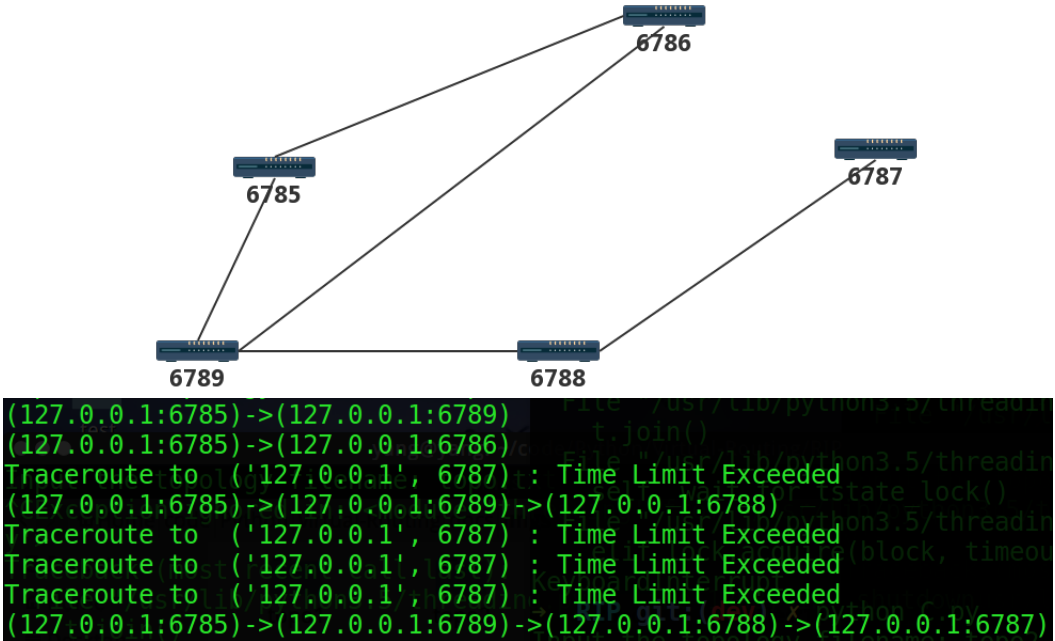


图 2: RIP 测试样例二

成功找出了其中的跳数最小的路径。

2.8 总结

3 基本 OSPF 协议

3.1 协议概述

本项目中实现的 OSPF 协议是一个基于链路状态的路由协议，运行该协议的每个路由器能够根据收到的所有链路信息计算出一个最短路径树，构造对应的转发表。其链路的权重的赋值取决于带宽，算法使用的是 Dijkstra 算法和 RPF 广播算法，定义了 Hello 和 LS（link-state）两种报文格式。实现了多条等费用路径的路由选择，并且解决了路由循环问题和改善了路由震荡问题。在协议的扩展中，实现了该协议下适用的 traceRoute 功能。



## 3.2 算法

### 3.2.1 路由计算算法

### 3.2.2 广播算法

使用了 RPF (Reverse Path First) 算法, 收到一个广播数据包的路由器会向来路以外的路由器转发该广播数据包当且仅当来路位于广播源和当前路由器的任一条最短路径上。

## 3.3 协议特性

### 3.3.1 协议流程

本项目中实现的 OSPF 协议主要包含的流程如下:

- 路由器在进入区域时, 向所有邻居发送 hello 报文 (该报文用于向邻居声明自己的存在), 再向该区域内的路由器广播自己的链路信息。
- 该路由器会定时地发送 hello 报文和广播链路信息。
- 收到一个 hello 报文意味着要维持该邻居的状态; 而如果在一段时间内没有收到某个邻居路由器的 hello 报文, 则认为该路由器故障或离开该区域。
- 当自身的链路信息发生改变时, 广播链路信息; 当收到其它路由器的链路信息时, 根据该信息更新自己的关于该区域的链路信息数据库。

## 3.4 报文格式

该协议中包含的 Hello 报文、LS 报文、traceRoute 报文、Echo 报文、普通数据报文格式如下:

Hello 报文。

|                |
|----------------|
| Command        |
| Source address |
| Metric         |

LSU 报文 (广播报文)。

|                        |
|------------------------|
| Command                |
| Source address         |
| Transmit address       |
| Destination address    |
| Metric                 |
| Repeat of last 8 bytes |
| ...                    |

#### TraceRoute 报文

|                     |
|---------------------|
| Command             |
| source address      |
| destination address |
| count               |

#### Echo 报文

|                     |
|---------------------|
| Command             |
| source address      |
| destination address |
| local address       |

普通报文的格式与在 RIP 中的格式相同。报文项解释：

(1) Command (1 Byte)：指明这条报文的类型。

0: 普通报文

1: hello

2: LSU (Link State Update)

3: traceRoute

4: Echo

(2) Source address (6 Bytes)：指明发送方的源地址和监听端口。

(3) Destination address (6 Bytes)：指明目的地的源地址和监听端口。

(4) Transmit address (6 Bytes)：指明广播中转发者的源地址和监听端口。

(5) Local address (6 Bytes)：指明 Echo 报文发送者的源地址和监听端口。

(6) Metric (2 Bytes): 代价度量。

(7) Count(1 Bytes): 路由跳数。

### 3.5 协议扩展

1. 路由循环
2. 路由震荡为了避免路由震荡，设置路由器广播其链路信息的时间随机化。
3. TraceRoute

### 3.6 协议实现

本 OSPF 协议基本由 Python 实现，主要利用的是 socket 进行套接字编程、struct 库进行数据的打包和解包、threading 库管理线程和定时器。

实现方法实现了一个 OSPF 协议类，处理内部流程和为外部调用留出了收发数据、查询路由等接口。

### 3.7 结果

测试所用的拓扑图以及路由结果如下图3和4所示。

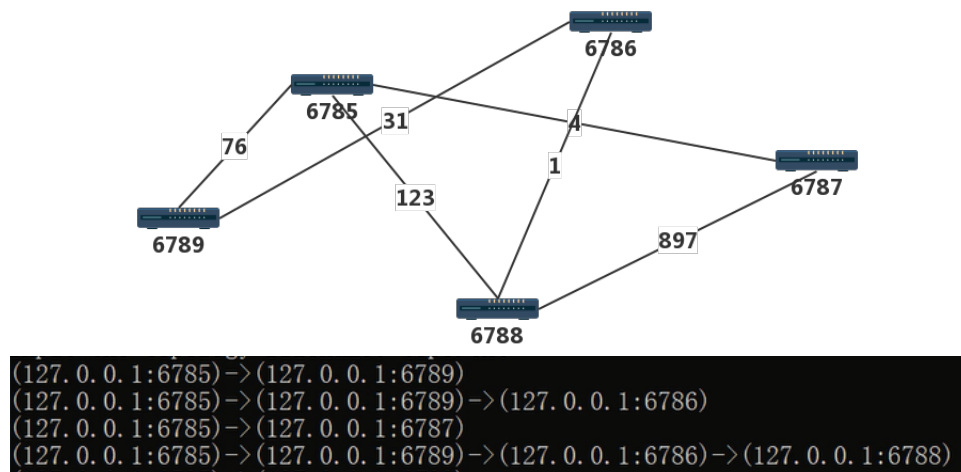


图 3: OSPF 测试样例一

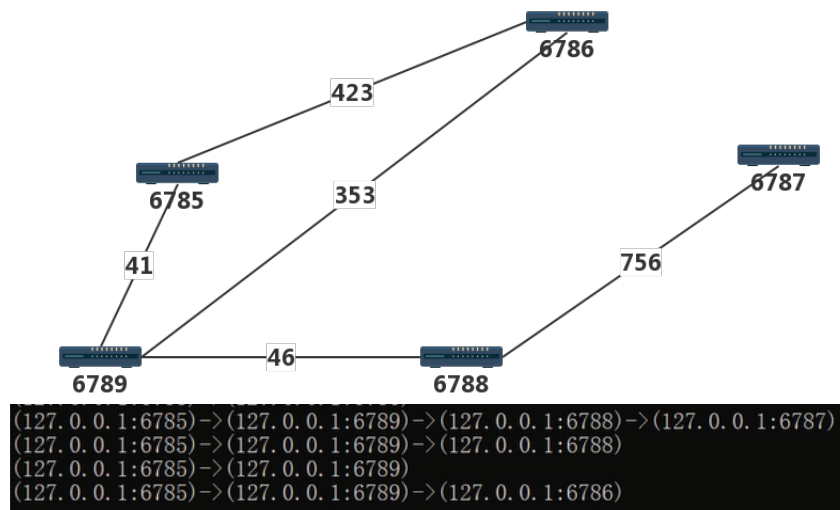


图 4: OSPF 测试样例二

成功找出了其中的最短路径。

### 3.8 总结

- 本项目中实现的基本 OSPF 协议淡化了其区域概念，简化了报文类型和路由器类型，同时也简化了路由器状态。
- 原生的 LS 算法会导致路由震荡和路由循环问题，而我们实现的 OSPF 协议改善甚至基本解决了这类问题。

## 4 基本中心化路由协议

### 4.1 协议概述

中心化路由协议是一个基于链路状态的网络层路由协议，使用的是 Dijkstra 算法，该协议中定义了一个服务器，负责接收区域内运行该协议的路由器的链路信息并且根据这些信息计算出整个区域内路由器的最短路径树，构造出每个路由器的转发表，并发送给对应路由器。并且在服务器中能够获取指定路由器对的最短路径。

4.2 算法

4.3 协议特性

4.3.1 协议流程

- 1. 路由器加入某个区域后，向邻居发送 hello 报文，并将定期发送 hello 报文。
- 2. 路由器定期向服务器发送 LS 报文，服务器接收后计算对应信息并定时向路由器发送转发表。
- 3. 服务器和路由器都有失效机制，当应当发送信息的路由器在一段时间没有发送信息后将其置为失效。

4.4 报文格式

基本 Centralized 协议中包含的报文格式有 hello 报文、LS 报文、转发表报文三种报文格式。  
LS 报文：

|                        |
|------------------------|
| Command                |
| Source address         |
| Destination address    |
| Metric                 |
| Repeat of last 8 bytes |
| ...                    |

Forward Table 报文：

|                         |
|-------------------------|
| Command                 |
| Destination address     |
| nextHop                 |
| Repeat of last 12 bytes |
| ...                     |

Hello 报文：

|                |
|----------------|
| Command        |
| Source address |
| Metric         |

普通报文：

|                     |
|---------------------|
| Command             |
| source address      |
| destination address |
| payload             |

报文项解释：

1. Command (1 Byte)：指明这条报文的类型。
  - 0: 普通报文
  - 1: hello
  - 2: LS
  - 3: 转发表
2. Source address (6 Bytes)：指明发送方的源地址和监听端口。
3. Destination address (6 Bytes)：指明目的地的源地址和监听端口。
4. nextHop (6 Bytes)：下一跳 IP 地址和监听端口。
5. Metric (2 Bytes)：代价度量。

## 4.5 协议扩展

查找路径功能。在服务器中，实现了指定路由器对查找两者之间的最短路径的功能。

## 4.6 协议实现

1. 客户端程序使用 python 编写，主要使用的是 socket、threading 等库。
2. 使用 Twisted 异步编程框架编写服务器端。使用 Twisted 框架编写的服务器端具有低功耗、资源利用率高的特点，同时事件驱动的内在逻辑也符合该服务器的功能特性。

## 4.7 结果

测试结果如下图5和6所示：

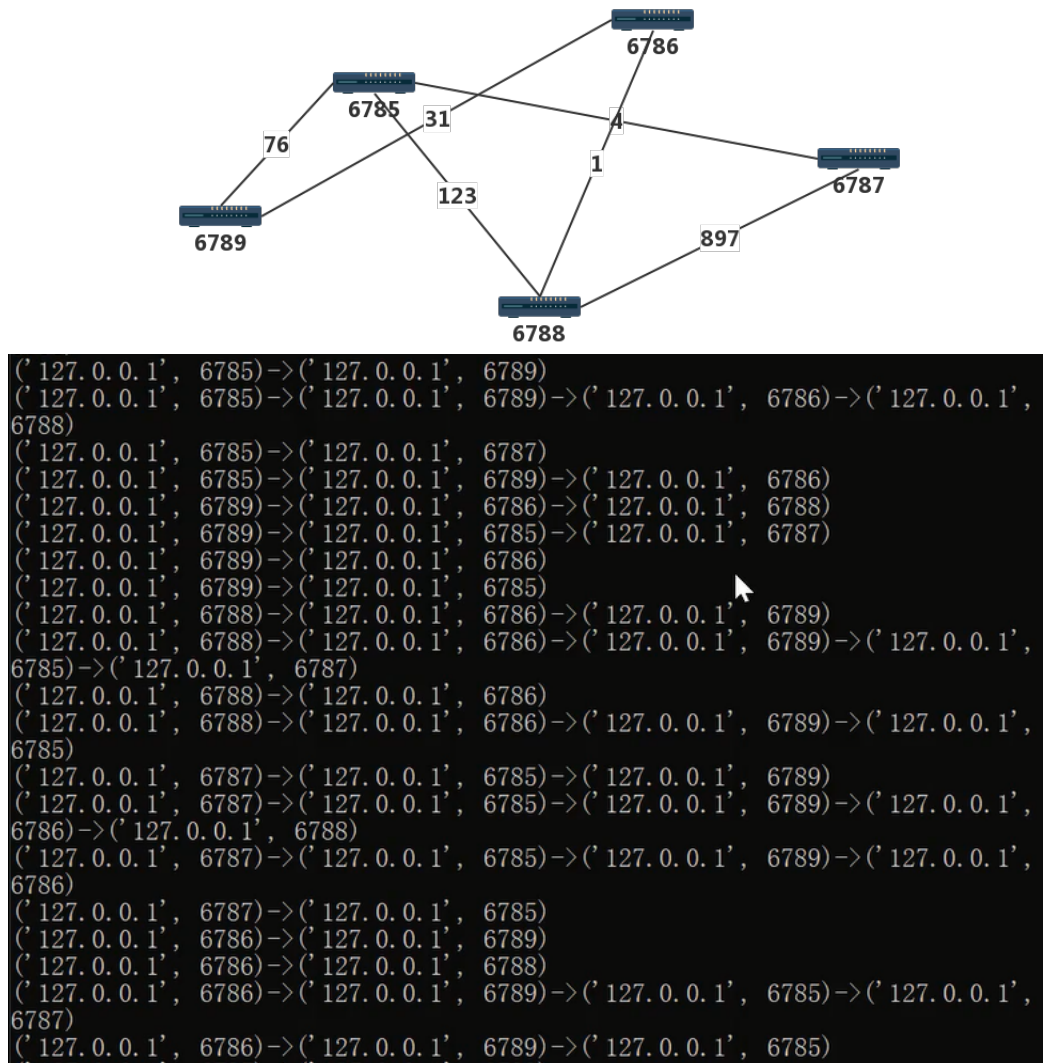


图 5: 中心化路由测试样例一

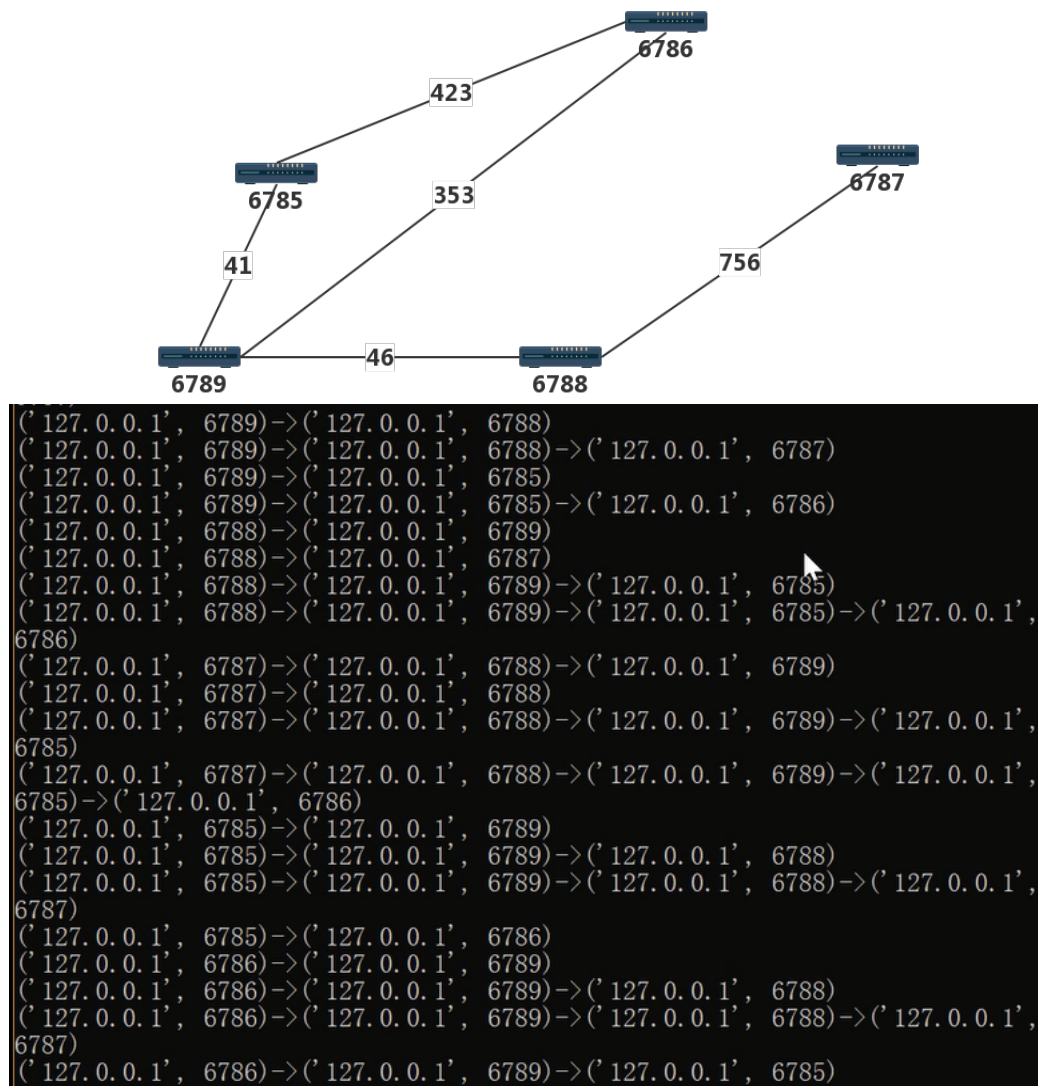


图 6: 中心化路由测试样例二

成功找出了全部最短路径。

## 4.8 总结

1. 由于在中心化路由协议中每个路由器的转发表都是由服务器计算并且同时几乎获得，即转发表基于同一张图结构，则不存在路由循环、路由震荡等问题。



## 5 安装与部署

## 6 项目管理记录

### 6.1 管理概述

该项目放在 github 上，见<https://github.com/Leo-xh/Virtual-Routing>，组内三人共同开发。

### 6.2 阶段管理

任务确定为设计并实现基本的 RIP 协议、OSPF 协议、中心化路由协议。任务确定前进行资料检索，总结原生 DV 和 LS 算法以及可能的解决方案，以及现存的协议版本。任务确定后检索实现和设计相关的资料。

### 6.3 人员分工

| 任务                | 主要参与者 |
|-------------------|-------|
| 协议设计              | 全体组员  |
| 代码框架搭建            | 王锡淮   |
| 算法设计以及实现          | 肖遥    |
| 通信功能              | 杨陈泽   |
| 线程、定时器以及数据结构的管理功能 | 王锡淮   |
| 调试                | 全体组员  |
| 项目打包以及文档编写        | 全体组员  |

## 7 总结

### 7.1 关于在应用层模拟网络层

## A 参考文献

1. RFC2328, Open Shortst Path First Protocol.
2. RFC2453, Route Information Protocol.
3. [https://en.wikipedia.org/wiki/Open\\_Shortest\\_Path\\_First](https://en.wikipedia.org/wiki/Open_Shortest_Path_First), Wikipedia for OSPF.
4. [https://en.wikipedia.org/wiki/Routing\\_Information\\_Protocol](https://en.wikipedia.org/wiki/Routing_Information_Protocol), Wikipedia for RIP.
5. [http://www.clnchina.com.cn/discussion\\_forum/2010/0721/8402.shtml](http://www.clnchina.com.cn/discussion_forum/2010/0721/8402.shtml), Four Timer in RIP.