



中山大學
SUN YAT-SEN UNIVERSITY

课程作业报告

Virtual Routing

学 院 名 称 : 数据科学与计算机学院

专业 (班级) : 16 级 计科 2 班

学 生 姓 名 : 张涵隽 徐原 石邢越

任 课 老 师 : 谢逸

时 间 : 2018 年 05 月 14 日

一 . 实验要求

实现两种路由类型：

A . 自组织路由

- 1.为小组成员间的计算机构建一个网络连接拓扑图
- 2.根据虚拟拓扑图在计算机之间建立连接并且定义链路费用
- 3.每一台计算机同时扮演客户端和路由器
- 4.每一台计算机实时交换并且更新路由信息
- 5.计算机之间可以相互传递信息

B . 中心化路由 （基于自组织路由）

- 1.基本要求同自组织路由
- 2.建立控制端，控制端决定并分发路由策略(路由表)给每个成员

实现两种算法类型：

A . 距离向量路由选择算法 (DV)

B . 链路状态路由选择算法 (LS)

(两种算法分别运用于自治路由，中心化路由采用 LS 算法。当计算机之间传递数据时寻求最优路径，完成数据传送)

二. 实验设计

2.1 数据结构设计

2.1.1 路由器信息格式

IP 地址
子网掩码
端口号
名字

2.1.2 控制器信息格式

IP 地址
子网掩码
端口
路由配置文件文件名

2.1.3 数据包定义

类型
源地址
目的地址
内容

2.1.4 转发表定义

目的地址
下一跳地址

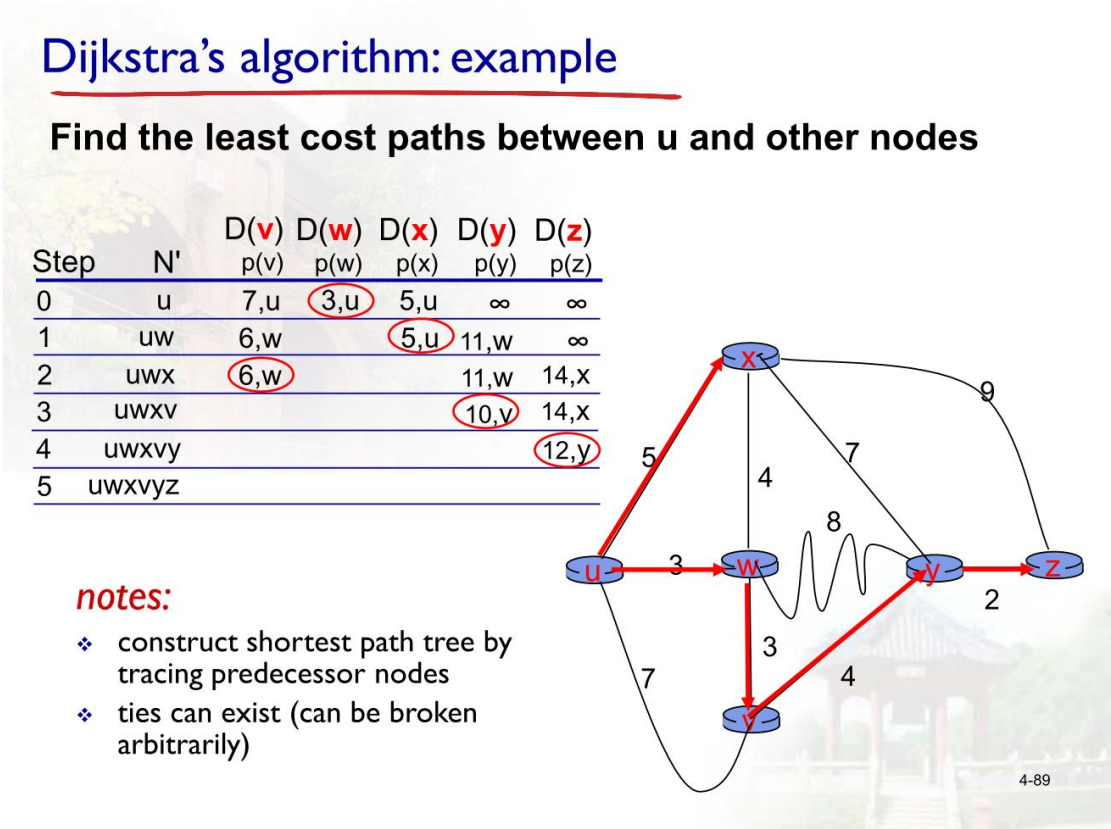
2.2 链路状态路由选择算法（LS）设计

说明： 在链路状态算法中，网络拓扑的所有的链路费用都是已知的，也

就是说可以用作 LS 算法的输入。该实验中通过让每个结点向网络中所有其他结点广播链路状态分组来完成, 其中每个链路状态分组包含它所连接的链路的特征和费用。

核心算法：Dijkstra 算法

Dijkstra 算法的思路见课件截图（图 1）。



（图 1 Dijkstra's Algorithm 思路）

Dijkstra 算法的伪代码如下：

D(v): 到算法的本次迭代，从源结点到目的结点 V 的最低费用路径的费用

P(v): 从源到 v 沿着当前最低费用路径的前一结点（v 的邻居）

N' : 结点子集, 如果从源到 v 的最低费用路径已确知, v 在 N' 中

Initialization:

$N' = \{u\}$

For all nodes v

If v is a neighbor of u

Then $D(v) = c(u, v)$

Else $D(v) = \text{infinity}$

Loop

Find w not in N' such that $D(w)$ is a minimum

Add w to N'

Update $D(v)$ for each neighbor v of w and not in N' :

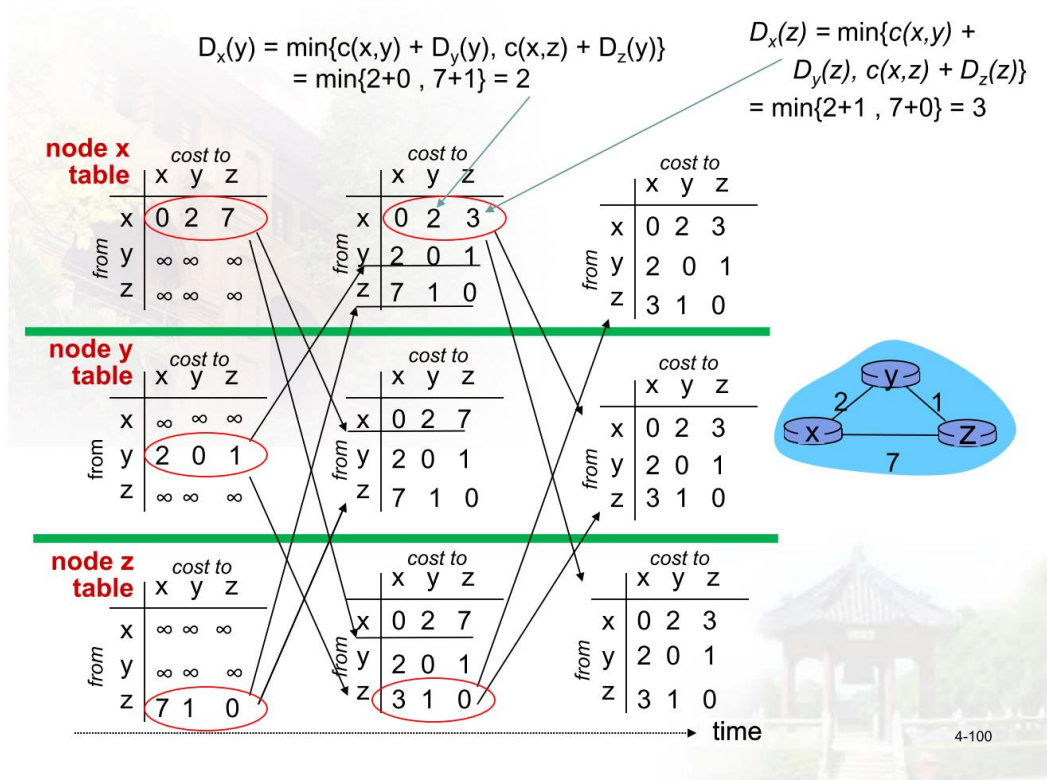
$D(v) = \min(D(v), D(w) + c(w, v))$

Until $N' = N$

2.3 距离向量路由选择算法

说明:

DV 算法思路见课件截图 (图 2)。



(图 2 DV 算法思路)

基本思想是，每个结点 x 以 $D_x(y)$ 开始，对在 N 中的所有结点，估计从它自己到结点 y 的最低费用路径的费用。令 $D_x(y)=[D_x(y):y \in N]$ 是结点 x 的距离向量，该向量是从 x 到在 N 中的所有其他结点 y 的费用估计的向量。使用 DV 算法，每个结点 x 维护下列路由选择信息：

- 对于每个邻居 v , 从 x 到直接相连邻居 v 的费用为 $c(x,v)$
- 结点 x 的距离向量，即 $D_x=[D_x(y):y \in N]$, 包含了 x 到 N 中所有目的地 y 的费用的估计值
- 它的每个邻居的距离向量，即对 x 的每个邻居 v , 有 $D_v=[D_v(y):y \in N]$

算法伪代码如下：

Initialization:

for all destinations y in N : $D_x(y) = c(x, y)$

for each neighbor w : $D_w(y) = ?$ For all destinations y in N

for each neighbor w

Send distance vector $D_x = [D_x(y) : y \text{ in } N]$ to w

Loop

wait(until I see a link cost change to some neighbor w or until I
receive a distance vector from some neighbor w)

for each y in N : $D_x(y) = \min_v \{c(x, v) + D_v(y)\}$

If $D_x(y)$ changed for any destination y

Send distance vector $D_x = [D_x(y) : y \text{ in } N]$ to all neighbors

forever

2.4 自组织路由

2.4.1 数据传输

实验采用 UDP 传输协议，其中使用 UDP 数据包监听线程判断某结点是否离线（虽然实际上的 OSPF 协议报文直接由 IP 承载而不涉及传输层的 TCP/UDP，但因为目前有限的网络编程水平，我们也选择用 UDP 完成类似的协议）。每台计算机选用两个端口，分别用于发送和接收数据。基于套接字 socket 编程传输数据（利用 project1 相关实现）。

2.4.2 路由算法

分别采用 LS 和 DV 算法（见本报告 2.3 小节详述）。

2.4.3 路由选择协议

2.4.3.1 基于 LS 算法实现 OSPF 协议

- (1) 向本自治系统中所有路由器发送信息，使用的方法是洪泛法。
- (2) 发送的信息就是与本路由器相邻的所有路由器的链路状态，但这只是路由器所知道的部分信息。“链路状态”就是说明本路由器都和哪些路由器相邻，以及该链路的链路消费。
- (3) 只有当链路状态发生变化时，路由器才用洪泛法向所有路由器发送此信息。

2.4.3.2 基于 DV 算法实现 RIP 协议

路由信息协议 RIP (Routing Information Protocol) 是一种分布式的、基于距离向量的路由选择协议。RIP 协议要求网络中的每一个路由器都要维护从它自己到其他每一个目的网络的距离记录。

- (1) 仅和相邻路由器交换信息。
- (2) 当网络拓扑发生变化时，即当有节点上线或者下线时，路由器及时通告自己的变更信息。实验中通告间隔设定为 0.5s。每隔 0.5s 就从相邻的路由器获取路由表。

2.4.4 超时检测/宕机处理

采用 UDP 数据包监听线程。链路监视，用以向邻居发送包、将设定秒数内没有收到包的邻居设为离线。邻居的回复由上面的 UDP 监听线程接收和处理。

2.5 中心化路由

数据传输、协议选择同前（不再赘述）

路由算法选择 LS 算法（dijkstra）

基于前者实现：中心控制

2.5.1 中心控制的改变：

- （1）增添 controller 端，系统内部不再自治，中心决定数据传输路径。
- （2）中心化路由分别采取 LS 和 DV 实时更新路由表以及寻求路径。
- （3）控制端的控制列表（其他端）的数据结构储存为：

端数量
端地址
上一次发送包的时间
在线状态
端名称
邻居列表
转发表

三. 代码说明

3.1 自治路由（LS）

文件名	作用
DataSeture	定义数据结构
packetSender	发送 UDP 数据包
router_ospf_A	路由器 A
router_ospf_B	路由器 B
router_ospf_C	路由器 C
router_ospf_D	路由器 D
router_ospf_E	路由器 E
utils	初始化和检查配置信息

3.2 自治路由（DV）

文件名	作用
DataSeture	定义 DV 需要的数据结构
RIP	采取 DV 算法完成 RIP 模拟
Untils	初始化和检查配置信息

3.3 中心化路由（LS）

文件名	作用
DataSeture	定义数据结构
packetSender	发送 UDP 数据包
controller	中心控制器
router_ospf	路由器（A,B,C,D,E 五个路由器都使用这份代码）
utils	初始化和检查配置信息

四．实验结果

4.1 自治 LS 实验结果

(1) 输入主机名，主机开始运行，并周期性（30s）地广播链路状态信息（以 A 路由为例）

```
Please input the name of this router(must match a file in configuration folder)
A
Router A is running... Local Address is 192.168.199.211/24:3984
2018-05-15 16:48:59
broadcasting link state...
2018-05-15 16:49:29
broadcasting link state...
█
```

(2) 接收 OSPF 数据包，并更新自己的路由表（依次新建 A、B、C、D、E 主机）

```
2018-05-15 16:52:18
Receive Link-state packet from B 192.168.199.211/24:3985
Running dijkstra for incoming link-state packet from Router B 192.168.199.211/24:3985
inf inf inf inf inf
inf inf inf inf inf
inf inf inf inf inf
inf inf inf inf inf
inf inf inf inf inf

current source node is A
num_of_online_slaves: 2
Get Forward Table:
Get Forward Table:
```

B 主机与 A 主机没有直接相连，且 A 与 B 相连链路上不是所有路由器都在线，

所以 A 主机的路由表没有变化。

```
2018-05-15 16:52:23
Receive Link-state packet from C 192.168.199.211/24:3986
Running dijkstra for incoming link-state packet from Router C 192.168.199.211/24:3986
inf inf 3 inf inf
inf inf inf inf inf
3 inf inf inf inf
inf inf inf inf inf
inf inf inf inf inf

current source node is A
num_of_online_slaves: 3
Get Forward Table:
1 next_hop: C des: C
```

```
2018-05-15 16:52:25
Receive Link-state packet from D 192.168.199.211/24:3987
Running dijkstra for incoming link-state packet from Router D 192.168.199.211/24:3987
inf inf 3 inf inf
inf inf inf 3 inf
3 inf inf 2 inf
inf 3 2 inf inf
inf inf inf inf inf

current source node is A
num_of_online_slaves: 4
Get Forward Table:
1 next_hop: C des: B
2 next_hop: C des: C
3 next_hop: C des: D
```

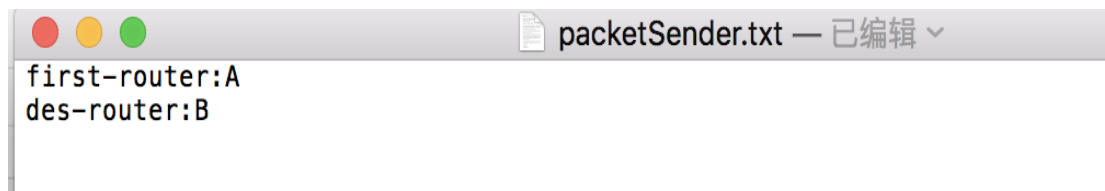
D 上线后，A 与 B 有联通的链路（即 A->C->D->B），所以 A 的转发表中出现目的为 B 的条目。

```
2018-05-15 16:52:28
Receive Link-state packet from E 192.168.199.211/24:3988
Running dijkstra for incoming link-state packet from Router E 192.168.199.211/24:3988
inf inf 3 inf 1
inf inf inf 3 2
3 inf inf 2 inf
inf 3 2 inf inf
1 2 inf inf inf

current source node is A
num_of_online_slaves: 5
Get Forward Table:
1 next_hop: E des: B
2 next_hop: C des: C
3 next_hop: C des: D
4 next_hop: E des: E
```

E 上线后，A 到 B 的最短路发生变化（即 A->E->B），所以 A 的转发表中以 B 为目的的条目中，下一跳更新成 E。

(3) 根据路由表选择最短路，转发数据（从 A 向 B 发送一个数据包）



```
packetSender.txt — 已编辑 v
first-router:A
des-router:B
```

A 发送报文“i am payload”给 B，下一跳是 E：

```
current source node is A
Get Forward Table:
1 next_hop: E des: B
2 next_hop: E des: E
Get Forward Table:
1 next_hop: E des: B
2 next_hop: E des: E
2018-05-15 21:08:01
Received(forwarding) Normal Packet, src= 127.0.0.1/24:53230 des= 192.168.199.211/24:3985 payload= i am payload
Forwarding to ForwardTableEntry:(des_addr = 192.168.199.211/24:3985 next_router_addr= 127.0.0.1/24:3988)
```

E 收到 A 发的报文，转发到目的主机 B，下一跳是 B：

```
current source node is E
Get Forward Table:
1 next_hop: A des: A
2 next_hop: B des: B
Get Forward Table:
1 next_hop: A des: A
2 next_hop: B des: B
2018-05-15 21:08:01
Received(forwarding) Normal Packet, src= 127.0.0.1/24:53230 des= 192.168.199.211/24:3985 payload= i am payload
Forwarding to ForwardTableEntry:(des_addr = 192.168.199.211/24:3985 next_router_addr= 192.168.199.211/24:3985)
```

经过 E 转发后，B 接收到 A 发的报文：

```
current source node is B
Get Forward Table:
1 next_hop: E des: A
2 next_hop: E des: E
Get Forward Table:
Get Forward Table:
1 next_hop: E des: A
2 next_hop: E des: E
Get Forward Table:
1 next_hop: E des: A
2 next_hop: E des: E
2018-05-15 21:08:01
Received(forwarding) Normal Packet, src= 127.0.0.1/24:53230 des= 192.168.199.211
/24:3985 payload= i am payload
**Local host is the destination of this packet.
```

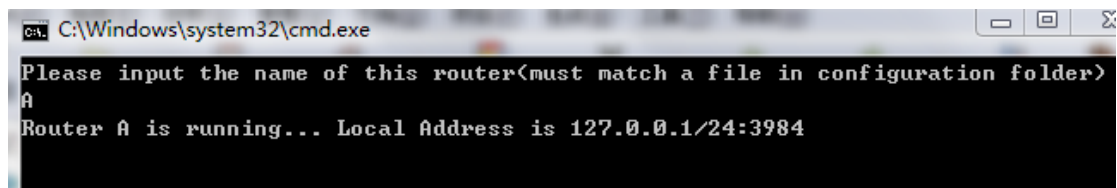
4) 其中某一主机宕机

主机 C 关闭，同时向子网中其他主机发送 offline 消息。

```
current source node is A
Get Forward Table:
1 next_hop: C des: C
Get Forward Table:
1 next_hop: C des: C
Neighbor C is offline.
```

4.2 自治 DV 实验结果

(1) 一开始注册五个端（命名）



五个节点都登陆后，显示最终 DV 算法下的路由信息(不再变更):

A:

```
1 next_hop: C des: C
2 next_hop: C des: D
3 next_hop: E des: B
4 next_hop: E des: E
```

B:

```
1 next_hop: D des: D
2 next_hop: D des: C
3 next_hop: E des: A
4 next_hop: E des: E
```

C:

```
1 next_hop: A des: A
2 next_hop: D des: D
3 next_hop: D des: B
4 next_hop: A des: E
```

D:

```
1 next_hop: B des: B
2 next_hop: C des: C
3 next_hop: C des: A
4 next_hop: B des: E
```

E:

```
1 next_hop: B des: B
2 next_hop: B des: D
3 next_hop: A des: C
4 next_hop: A des: A
```

(具体的邻居登录下线变化可参见 demo)

当有节点加入或者退出时

链路表随之一直更新 (30s)

由于时延 30s, demo 更容易查看变更 (具体可参见 demo)

4.3 中心化 LS 实验结果

(1) A 上线，告知 controller。此时拓扑图如下。



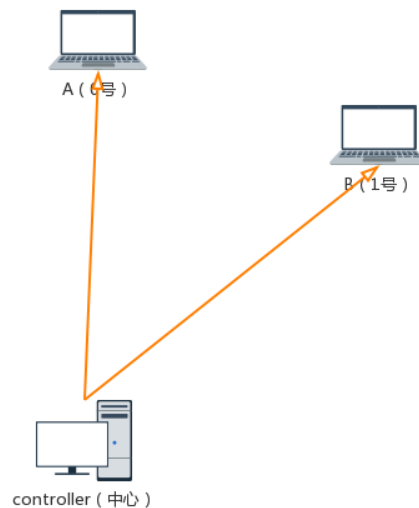
此时网络中只有一台主机/路由器，没有可以转发的路径，controller 和 A 显示的界面如下图。Least cost 表示此时 A 到其他主机的最短路径。

```
Receive OSPF neighbor list
Receive the packet from A 127.0.0.1/24:3984
Slave A is online.
Route the packet using dijkstra algorithm A 127.0.0.1/24:3984
inf inf inf inf inf
inf inf inf inf inf
inf inf inf inf inf
inf inf inf inf inf
inf inf inf inf inf

Source node in this step is A
The number of online slaves online: 1
Least cost: {0: 0, 1: inf, 2: inf, 3: inf, 4: inf}
```

```
Please name this router as a file's name in configuration folder
A
Router A is running... Local Address is 127.0.0.1/24:3984
Controller is online.
Get Forward Table:
```

(2) B 上线，告知 controller。此时拓扑图如下。



此时网络中有 2 台主机/路由器, 但它们之间不相连, 没有可以转发路径, controller 和 B 显示的界面如下图。Least cost 表示此时 B 到其他主机的最短路径。

```

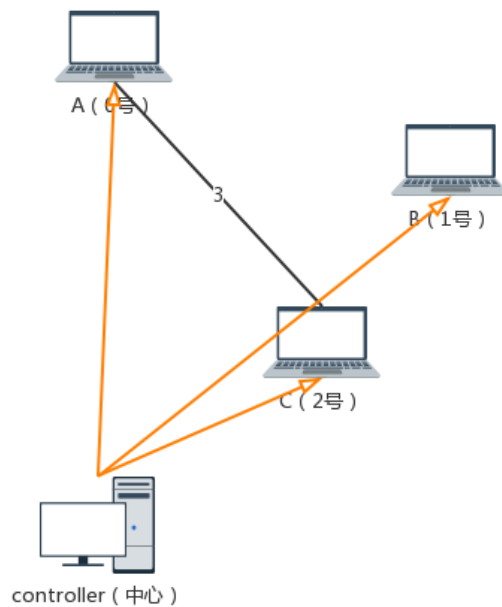
Receive OSPF neighbor list
Receive the packet from B 127.0.0.1/24:3985
Slave B is online.
Route the packet using dijkstra algorithm B 127.0.0.1/24:3985
inf inf inf inf inf
inf inf inf inf inf
inf inf inf inf inf
inf inf inf inf inf
inf inf inf inf inf

Source node in this step is A
The number of online slaves online: 2
len(N'): 1
Least cost: {0: 0, 1: inf, 2: inf, 3: inf, 4: inf}
Source node in this step is B
The number of online slaves online: 2
len(N'): 1
Least cost: {1: 0, 0: inf, 2: inf, 3: inf, 4: inf}
  
```

```

Please name this router as a file's name in configuration folder
B
Router B is running... Local Address is 127.0.0.1/24:3985
Controller is online.
Get Forward Table:
  
```

(3) C 上线, 告知 controller。此时拓扑图如下。



此时网络中有 3 台主机/路由器，A 和 C 之间相连，可以路由转发，B 和其他的主机/路由器时间没有连接。controller 和 C 显示的界面如下图。Least cost 表示此时 C 到其他主机的最短路径。

```

Receive OSPF neighbor list
Receive the packet from C 127.0.0.1/24:3986
Slave C is online.
Route the packet using dijkstra algorithm C 127.0.0.1/24:3986
inf inf 3 inf inf
inf inf inf inf inf
3 inf inf inf inf
inf inf inf inf inf
inf inf inf inf inf
  
```

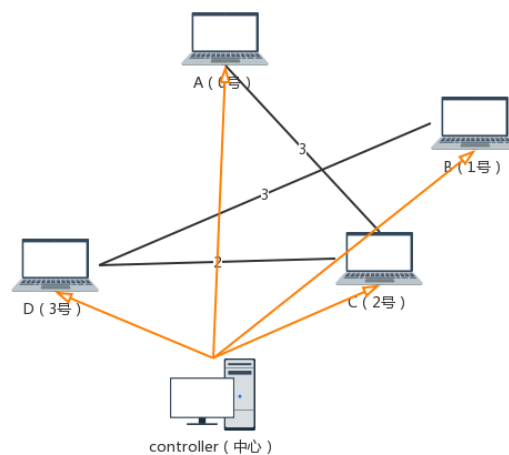
(中间以 Dijkstra 算法计算最短路径的过程省略)

```

Least cost: {2: 0, 0: 3, 1: inf, 3: inf, 4: inf}
  
```

```
C:\windows\system32
Please name this router as a file's name in configuration folder
C
Router C is running... Local Address is 127.0.0.1/24:3986
Neighbor A is online
Controller is online.
Get Forward Table:
1 next_hop: A des: A
Get Forward Table:
1 next_hop: A des: A
Get Forward Table:
1 next_hop: A des: A
```

(4) D 上线，告知 controller。此时拓扑图如下。



此时网络中有 4 台主机/路由器，有 3 条花费分别为 2，2，3 的边，controller 和 D 显示的界面如下图。Least cost 表示此时 D 到其他主机的最短路径。

```
Receive OSPF neighbor list
Receive the packet from D 127.0.0.1/24:3987
Slave D is online.
Route the packet using dijkstra algorithm D 127.0.0.1/24:3987
inf inf 3 inf inf
inf inf inf inf inf
3 inf inf inf inf
inf inf inf inf inf
inf inf inf inf inf
```

(中间以 Dijkstra 算法计算最短路径的过程省略)

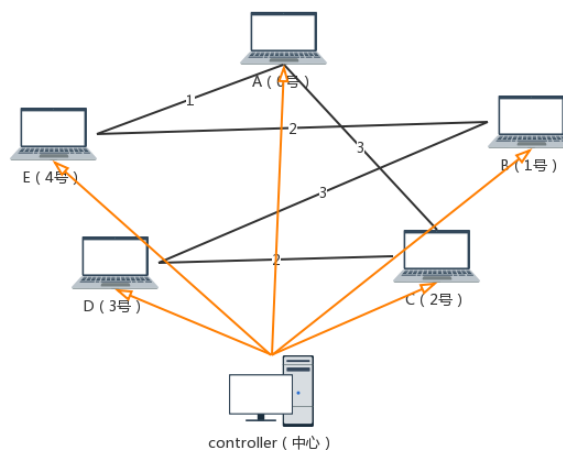
```
Least cost: {3: 0, 0: 5, 1: 3, 2: 2, 4: inf}
```

```

Please name this router as a file's name in configuration folder
D
Router D is running... Local Address is 127.0.0.1/24:3987
Controller is online.
Neighbor B is online
Neighbor C is online
gGet Forward Table:
Get Forward Table:
1 next_hop: B des: B
rGet Forward Table:
1 next_hop: C des: A
2 next_hop: B des: B
3 next_hop: C des: C
Get Forward Table:
1 next_hop: C des: A
2 next_hop: B des: B
3 next_hop: C des: C
Get Forward Table:
1 next_hop: C des: A
2 next_hop: B des: B
3 next_hop: C des: C

```

(5) E 上线，告知 controller。此时拓扑图如下。



此时网络中有 5 台主机/路由器，有 5 条花费分别为 1，2，2，2，3 的边，controller 和 E 显示的界面如下图。Least cost 表示此时 E 到其他主机的最短路径。

```

Receive OSPF neighbor list
Receive the packet from E 127.0.0.1/24:3988
Slave E is online.
Route the packet using dijkstra algorithm E 127.0.0.1/24:3988
inf inf 3 inf 1
inf inf inf 3 inf
3 inf inf 2 inf
inf 3 2 inf inf
1 inf inf inf inf

```

(中间以 Dijkstra 算法计算最短路径的过程省略)

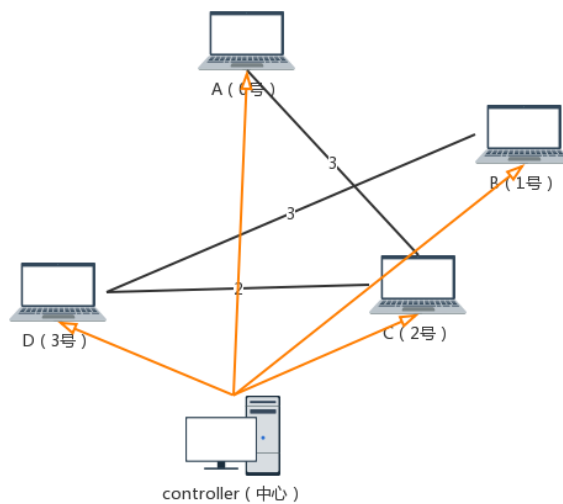
```
Least cost: {4: 0, 0: 1, 1: 2, 2: 4, 3: 5}
```

```

Router E is running... Local Address is 127.0.0.1/24:3988
Neighbor A is online
Controller is online.
Neighbor B is online
Get Forward Table:
1 next_hop: A des: A
2 next_hop: A des: B
3 next_hop: A des: C
4 next_hop: A des: D
Get Forward Table:
1 next_hop: A des: A
2 next_hop: A des: B
3 next_hop: A des: C
4 next_hop: A des: D
Get Forward Table:
1 next_hop: A des: A
2 next_hop: B des: B
3 next_hop: A des: C
4 next_hop: A des: D
Get Forward Table:
1 next_hop: A des: A
2 next_hop: B des: B
3 next_hop: A des: C
4 next_hop: A des: D

```

(6) E 下线，告知 controller。此时拓扑图如下。



此时网络中有 4 台主机/路由器，有 3 条花费分别为 2, 2, 3 的边，controller 和 E 的邻居 A、B 显示的界面如下图。

```

Slave E is offline.
receive OSFP neighbor list
Receive the packet from A 127.0.0.1/24:3984
Route the packet using dijkstra algorithm A 127.0.0.1/24:3984
inf inf 3 inf inf
inf inf inf 3 inf
3 inf inf 2 inf
inf 3 2 inf inf
inf inf inf inf inf
  
```

```

Neighbor E is online
Get Forward Table:
1 next_hop: E des: B
2 next_hop: C des: C
3 next_hop: C des: D
4 next_hop: E des: E
127.0.0.1/24:5001
Neighbour E is offline
Get Forward Table:
1 next_hop: C des: B
2 next_hop: C des: C
3 next_hop: C des: D
Get Forward Table:
1 next_hop: C des: B
2 next_hop: C des: C
3 next_hop: C des: D
  
```

```
Neighbor E is online
Get Forward Table:
1 next_hop: D des: A
2 next_hop: D des: C
3 next_hop: D des: D
4 next_hop: D des: E
Get Forward Table:
1 next_hop: E des: A
2 next_hop: D des: C
3 next_hop: D des: D
4 next_hop: E des: E
Get Forward Table:
1 next_hop: E des: A
2 next_hop: D des: C
3 next_hop: D des: D
4 next_hop: E des: E
Get Forward Table:
1 next_hop: E des: A
2 next_hop: D des: C
3 next_hop: D des: D
4 next_hop: E des: E
Get Forward Table:
1 next_hop: D des: A
2 next_hop: D des: C
3 next_hop: D des: D
127.0.0.1/24:5001
Neighbour E is offline
Get Forward Table:
1 next_hop: D des: A
2 next_hop: D des: C
3 next_hop: D des: D
```

五. 项目管理

小组分工合作见以下项目管理图，其中红色部分、紫色部分、绿色部分分别是张涵隽、徐原、石邢越同学分配的工作，蓝色部分为合作完成的部分。

