《现代交换原理》实验报告

实验名称 ＿＿＿＿＿MPLS 编程实验＿＿＿＿＿

班　　级 ＿＿＿＿＿＿2017211305＿＿＿＿＿＿

姓　　名 ＿＿＿＿＿＿＿＿于海鑫＿＿＿＿＿＿＿＿

指导教师 ＿＿＿＿＿＿＿＿丁玉荣＿＿＿＿＿＿＿＿

一、 实验目的

加强学生对 MPLS 交换中标记请求、标记分配与分发、标记分组转发的理解。

二、 实验设计

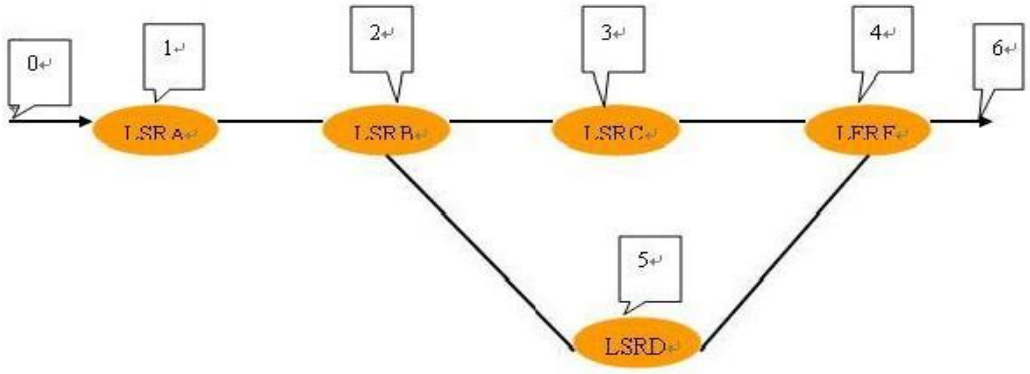
多协议标记交换 MPLS（Multiple Protocol Labeled Switching）技术是将第二层交换和第三层路由结合起来的一种 L2/L3 集成数据传输技术。MPLS 是一项面向连接的交换技术，因此有建立连接的过程。各个 MPLS 设备运行 路由协议，在标记分发协议 LDP 的控制下根据计算得到的路由在相邻的路由器进行标记分配和分发，从而通过标记的拼接建立起从网络入口到出口的标记交换路径 LSP。

在数据转发过程中，入口标记路由器 LER 根据数据流的属性比如网络层目的地址等将分组映射到某一转发等价类 FEC，并为分组绑定标记。核心标记交换路由器 LSR 只需根据分组中所携带的标记进行转发即可。出口标记路由器 LER 弹出标记，根据分组的网络层目的地址将分组转发到下一跳。MPLS 节点（MPLS 标记交换路由器 LSR 或 MPLS 边缘路由器 LER）均要创建和维护传统的路由表和标记信息库 LIB。

路由表记录记录路由信息，用于转发网络层分组和标记分发从而建立标记交换路径。LIB 记录了本地节点分配的标记与从邻接 MPLS 节点收到的标记之间的映射关系，用于标记分组的转发。

MPLS 技术的核心实质在于：（1）网络中分组基于标记的转发（2）LDP 协议控制下的进行标记分发从而建立标记交换路径 LSP。

实验网络的拓扑结构（节点分布示意图）：



三、 实验主要数据结构

所需要的头文件："mplsconstant.h"

其中的主要数据结构为：

```
//发送的请求信息包数据结构
struct ReqType
{
    int iFirstNode;    //请求信息包的源节点
    int iEndNode;      //请求信息包的目的节点
    double ipaddress; //请求信息包包含的网络层目的 IP 地址前缀（例如 197.42）
};

//路由表表项的数据结构
struct routertype
{
    double ipaddress; //网络层目的地址前缀
    int nexthop;       //下一跳节点
    int lasthop;       //上一跳节点
    int inpoint;       //入端口号
    int outpoint;      //出端口号
};

//标记信息表表项的数据结构
```

```
struct libtype
{
    double ipaddress; //网络层目的地址前缀
    int inpoint;      //入端口号
    int outpoint;     //出端口号
    int inlabel;      //入标记值
    int outlabel;     //出标记值
};

//发送的标记信息包数据结构

struct LabelPack
{
    int iFirstNode; //源节点号
    int iEndNode;   //目的节点号
    int labelvalue; //标签值
};

struct funcusedtype
{
    struct libtype libinfo;      //包含的标记信息表项
    struct LabelPack labelinfo; //包含的标记信息包数据结构
};

//发送的标记分组信息包类型

struct LabelledDataPack
{
    int iFirstNode;                 //源节点号
    int iEndNode;                   //目的节点号
    struct MessageType DataInfo; //包含的标记分组类型信息
};

//标记分组类型
struct MessageType
{
    double ipaddress; //网络层目的地址前缀
    int labelvalue;   //输出标签值
};
```

1. 标记请求实验要求函数：

```
extern "C" _declspec(dllexport) struct ReqType req_process(
int idnow, struct routertype routenow) {
    struct ReqType reqtemp;
    return reqtemp;
}
```

参数意义：

int idnow：当前的节点号；

struct routertype routenow：当前所指的路由表的表项；

函数要求：根据提供的当前节点号和路由表表项值产生标记请求包；

过程描述：

标记请求包的源节点号由当前节点号提供，目的节点号和 ip 地址前缀由当前所指的路由表表项的下一跳节点和 ip 地址前缀提供；

2：标记分配与分发实验：

```
extern "C" _declspec(dllexport) struct funcusedtype label_p
rocess(struct routertype routenow, int labelout, int idnow)
 {
    struct funcusedtype tempstruct;
    return tempstruct;
}
```

参数意义：

struct routertype routenow：当前所指的路由表表项；

int labelout：分配的输出标签号；

int idnow：当前的节点号；

函数要求：

该函数要求根据提供的路由表当前表项、分配的输出标签号和当前节点号，构造一 funcusedtype 信息包。注：各节点的输入标签可以自由选定，但必须是 1-9 的整数；

过程描述：

该 funcusedtype 信息包的 libinfo 部分可由当前的路由表表项、当前分配的标签号的有关部分构成； labelinfo 部分由当前节点号和当前的路由表表项的有关部分构成；

3．标记分组转发实验

```
extern "C" _declspec(dllexport) struct LabelledDataPack pac
k_process(struct routertype routenow, struct libtype libnow
, int idnow) {
    struct LabelledDataPack packtemp;
    return packtemp;
}
```

参数意义：

struct routertype routenow： 当前所指的路由表表项；

struct libtype libnow： 当前的标签信息表表项；

int idnow： 当前的节点号；

函数要求：

该函数要求根据提供的路由表表项、标签信息表表项和当前节点号，构造出一个标签数据信息包。

过程描述：

该标签信息包的源节点、目的节点、 IP 地址前缀和标签值均可由当前节点号、路由表表项和标签信息表表项构成；

四、 实验代码

A. 实验 1

```
#include "mplsconstant.h"

extern "C" _declspec(dllexport) struct ReqType req_process(
int idnow, struct routertype routenow) {
    struct ReqType reqtemp;
    reqtemp.iFirstNode = idnow;
    reqtemp.iEndNode = routenow.nexthop;
    reqtemp.ipaddress = routenow.ipaddress;
    return reqtemp;
}
```

B. 实验 2

```
#include "mplsconstant.h"

extern "C" _declspec(dllexport) struct funcusedtype label_p
rocess(struct routertype routenow, int labelout, int idnow)
 {
    struct funcusedtype tempstruct;
    tempstruct.libinfo.ipaddress = routenow.ipaddress;
    tempstruct.libinfo.inpoint = routenow.inpoint;
    tempstruct.libinfo.outpoint = routenow.outpoint;
    tempstruct.libinfo.inlabel = 7;
    tempstruct.libinfo.outlabel = labelout;
    tempstruct.labelinfo.iFirstNode = idnow;
    tempstruct.labelinfo.iEndNode = routenow.lasthop;
    tempstruct.labelinfo.labelvalue = tempstruct.libinfo.in
label;
    return tempstruct;
}
```

C. 实验 3

```
#include "mplsconstant.h"

extern "C" _declspec(dllexport) struct LabelledDataPack pac
k_process(struct routertype routenow, struct libtype libnow
, int idnow) {
    struct LabelledDataPack packtemp;
```

```
    packtemp.iFirstNode = idnow;
    packtemp.iEndNode = routenow.nexthop;
    packtemp.DataInfo.ipaddress = routenow.ipaddress;
    packtemp.DataInfo.labelvalue = libnow.outlabel;

    return packtemp;
}
```

五、 实验结果

实验结果符合预期，实验成功。

六、 实验心得

这次实验比较简单，代码实现基本就是赋值的操作，最重要的是增强了我对MPLS 原理和工作过程的理解 。