《现代交换原理》实验报告

实验名称　MPLS 多协议标记交换

班　　级　2015211307

学　　号　2015211906

姓　　名　王　睿　嘉

指导教师　丁　玉　荣

实验 4 MPLS 多协议标记交换

一、实验目的

加强学生对 MPLS 交换中标记请求、标记分配与分发及标记分组转发的理解。

二、实验设计

多协议标记交换 MPLS 技术将第二层交换和第三层路由结合起来，从而实现 L2/L3 集成数据传输。

MPLS 是一种面向连接的交换技术，因此有建立连接的过程。

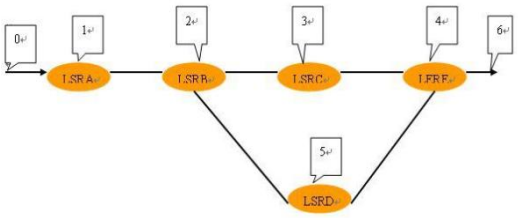
各 MPLS 设备运行路由协议。在标记分发协议 LDP 的控制下，根据计算得到的路由，在相邻路由器间进行标记分配和分发，从而通过标记的拼接建立起从网络入口到出口的标记交换路径 LSP。

在数据转发过程中，入口标记路由器 LER 根据数据流的属性，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 或边缘路由器 LER）均要创建和维护传统的路由表及标记信息库 LIB。

路由表记录路由信息。由于网络层分组的转发和标记的分发，建立起标记交换路径。LIB 则记录了本地节点分配的标记与邻接节点收到的标记间的映射关系，用于标记分组的转发。

MPLS 的核心实质在于：（1）网络中分组转发基于标记；（2）LDP 协议控制标记分发，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）如下：



三、主要数据结构

头文件："mplsconstant.h"

主要数据结构：

```
//请求信息包
struct ReqType{
    int iFirstNode;                    //源节点
```

```
    int iEndNode;                   //目的节点
    double ipaddress;               //网络层目的地址前缀（例如197.42）
};
//路由表表项
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
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
};
//标记信息包
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;         //标记信息表表项
    struct LabelPack labelinfo;     //标记信息包
};
//标记分组
struct MessageType {
    double ipaddress;               //网络层目的地址前缀
    int labelvalue;                 //输出标签值
};
//发送的标记分组信息包
struct LabelledDataPack{
    int iFirstNode;                 //源节点号
    int iEndNode;                   //目的节点号
    struct MessageType DataInfo;    //标记分组类型信息
};
```

标记请求函数：

```
struct ReqType req_process(int idnow, struct routertype routenow) {
    struct ReqType reqtemp;
```

```
    return reqtemp;
}
/*参数意义：
int idnow：当前节点号；
struct routertype routenow：当前路由表表项；
函数要求：根据提供的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地址前缀由当前路由表表项的下一跳节点和IP地址前缀提供
*/
```

标记分配与分发函数：

```
struct funcusedtype label_process(struct routertype routenow, int labelout, int idnow){
    struct funcusedtype tempstruct;
    return tempstruct;
}
/*参数意义：
struct routertype routenow：当前路由表表项；
int labelout：分配的输出标签号；
int idnow：当前节点号；
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funcusedtype信息包。注：各节点的输入标签可自由选定，但必须是1-9的整数；
过程描述：该funcusedtype信息包的libinfo部分可由当前路由表表项、当前分配的标签号的有关结构构成，labelinfo部分由当前节点号和当前的路由表表项的有关结构构成
*/
```

标签分组转发函数：

```
struct LabelledDataPack pack_process(struct routertype routenow, struct libtype libnow, int idnow) {
    struct LabelledDataPack packtemp;
    return packtemp;
}
/*参数意义：
struct routertype routenow：当前路由表表项；
struct libtype libnow：当前标签信息表表项；
int idnow：当前节点号；
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提供
*/
```

四、源代码

标记请求：

```
#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;
}
```

标记分配与分发：

```
#include "mplsconstant.h"
extern "C" _declspec(dllexport) struct funcusedtype label_process(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 = idnow % 9 + 1;          //1-9整数
    tempstruct.libinfo.outlabel = labelout;

    tempstruct.labelinfo.iFirstNode = idnow;
    tempstruct.labelinfo.iEndNode = routenow.lasthop;
    tempstruct.labelinfo.labelvalue = tempstruct.libinfo.inlabel;
    return tempstruct;
}
```

标记分组转发：

```
#include "mplsconstant.h"
extern "C" _declspec(dllexport) struct LabelledDataPack pack_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

编译运行后，仔细观察MPLS交换过程：

（1）请求标记信息包的内容都是REQ+194.27，沿源主机的边缘路由器一直到目的主机的边缘路由器；

（2）从目的主机的边缘路由器返回一个分配标记信息包。其中，边缘路由器不会使用这个程序；

（3）连接建立后，数据包按照标记好的路径从源主机转发到目的主机。在此过程中，路由进出标记和第二部分所建立的一致。

六、实验心得

在了解仿真环境主要数据结构的情况下，该实验的难度并不算很大。

实践出真知，本次MPLS多协议标记交换实验是对课堂和书本所学知识的补充。实际情形与已了解到的原理大体一致。通过自己动手、亲力亲为编写交换过程相应代码，并对结果进行分析，加深了对下游按需标记分配方式下，MPLS连接建立及标记分组转发过程的理解和记忆，收获颇丰。