# 实验6 · 生成树机制实验

吴嘉皓

2015K8009915007

## 一、实验内容

1. 补全stp.c中的stp\_handle\_config\_packet(stp\_t \*stp, stp\_port\_t \*p, struct stp\_config \*config)函数，实现对给定拓扑的生成树机制；
2. 自己构造一个不少于6个节点，链路冗余度不小于2的拓扑，并使用1中完成的stp程序计算输出最小生成树拓扑；

## 二、实验流程

### （一）代码目录

2015K8009915007\_吴嘉皓\_06.tar.gz

├── 06-stp

│   ├── disable\_ipv6.sh

│   ├── disable\_offloading.sh

│   ├── dump\_output\_4.sh

│   ├── dump\_output\_6.sh

│   ├── four\_node\_ring.py

│   ├── four\_result.log

│   ├── include

│   │   ├── base.h

│   │   ├── ether.h

│   │   ├── hash.h

│   │   ├── headers.h

│   │   ├── list.h

│   │   ├── log.h

│   │   ├── packet.h

│   │   ├── stp.h

│   │   ├── stp\_proto.h

│   │   ├── stp\_timer.h

│   │   ├── types.h

│   │   └── utils.h

│   ├── main.c

│   ├── Makefile

│   ├── packet.c

│   ├── six\_node\_ring.py

│   ├── six\_result.log

│   ├── stp.c

│   ├── stp-reference

│   └── stp\_timer.c

├── README.txt

└── 实验6-生成树机制-实验报告.pdf

### （二）实验流程

 1. 按序在**06-stp**目录下输入如下命令：

然后，等待30s左右，在xterm的b1终端中运行 pkill -SIGTERM stp ；

接着在终端中运行， ./dump\_output\_4.sh ,并查看比对结果。

2. 按序在**06-stp**目录下输入如下命令：

然后，等待30s左右，在xterm的b1终端中运行 pkill -SIGTERM stp ；

接着在终端中运行， ./dump\_output\_6.sh ,并查看比对结果。

## 三、实验结果

### （一）四节点环路拓扑的文字输出结果

NODEb1 dumps:

INFO: this switch is root.

INFO: port id: 01, role: DESIGNATED.

INFO: designated ->root: 0101, ->switch: 0101, ->port: 01, ->cost: 0.

INFO: port id: 02, role: DESIGNATED.

INFO: designated ->root: 0101, ->switch: 0101, ->port: 02, ->cost: 0.

NODE b2 dumps:

INFO: non-root switch, desinated root: 0101, root path cost: 1.

INFO: port id: 01, role: ROOT.

INFO: designated ->root: 0101, ->switch: 0101, ->port: 01, ->cost: 0.

INFO: port id: 02, role: DESIGNATED.

INFO: designated ->root: 0101, ->switch: 0201, ->port: 02, ->cost: 1.

NODE b3 dumps:

INFO: non-root switch, desinated root: 0101, root path cost: 1.

INFO: port id: 01, role: ROOT.

INFO: designated ->root: 0101, ->switch: 0101, ->port: 02, ->cost: 0.

INFO: port id: 02, role: DESIGNATED.

INFO: designated ->root: 0101, ->switch: 0301, ->port: 02, ->cost: 1.

NODE b4 dumps:

INFO: non-root switch, desinated root: 0101, root path cost: 2.

INFO: port id: 01, role: ROOT.

INFO: designated ->root: 0101, ->switch: 0201, ->port: 02, ->cost: 1.

INFO: port id: 02, role: ALTERNATE.

INFO: designated ->root: 0101, ->switch: 0301, ->port: 02, ->cost: 1.

### （二）六节点环路拓扑的文字输出结果

NODE b1 dumps:

INFO: this switch is root.

INFO: port id: 01, role: DESIGNATED.

INFO: designated ->root: 0101, ->switch: 0101, ->port: 01, ->cost: 0.

INFO: port id: 02, role: DESIGNATED.

INFO: designated ->root: 0101, ->switch: 0101, ->port: 02, ->cost: 0.

NODE b2 dumps:

INFO: non-root switch, desinated root: 0101, root path cost: 1.

INFO: port id: 01, role: ROOT.

INFO: designated ->root: 0101, ->switch: 0101, ->port: 01, ->cost: 0.

INFO: port id: 02, role: DESIGNATED.

INFO: designated ->root: 0101, ->switch: 0201, ->port: 02, ->cost: 1.

NODE b3 dumps:

INFO: non-root switch, desinated root: 0101, root path cost: 1.

INFO: port id: 01, role: ROOT.

INFO: designated ->root: 0101, ->switch: 0101, ->port: 02, ->cost: 0.

INFO: port id: 02, role: DESIGNATED.

INFO: designated ->root: 0101, ->switch: 0301, ->port: 02, ->cost: 1.

INFO: port id: 03, role: DESIGNATED.

INFO: designated ->root: 0101, ->switch: 0301, ->port: 03, ->cost: 1.

NODE b4 dumps:

INFO: non-root switch, desinated root: 0101, root path cost: 2.

INFO: port id: 01, role: ROOT.

INFO: designated ->root: 0101, ->switch: 0201, ->port: 02, ->cost: 1.

INFO: port id: 02, role: ALTERNATE.

INFO: designated ->root: 0101, ->switch: 0301, ->port: 02, ->cost: 1.

INFO: port id: 03, role: DESIGNATED.

INFO: designated ->root: 0101, ->switch: 0401, ->port: 03, ->cost: 2.

NODE b5 dumps:

INFO: non-root switch, desinated root: 0101, root path cost: 2.

INFO: port id: 01, role: ROOT.

INFO: designated ->root: 0101, ->switch: 0301, ->port: 03, ->cost: 1.

INFO: port id: 02, role: DESIGNATED.

INFO: designated ->root: 0101, ->switch: 0501, ->port: 02, ->cost: 2.

NODE b6 dumps:

INFO: non-root switch, desinated root: 0101, root path cost: 3.

INFO: port id: 01, role: ROOT.

INFO: designated ->root: 0101, ->switch: 0401, ->port: 03, ->cost: 2.

INFO: port id: 02, role: ALTERNATE.

INFO: designated ->root: 0101, ->switch: 0501, ->port: 02, ->cost: 2.

## 四、结果分析

### （一）四节点拓扑结果分析

* 四节点环路拓扑结构如下：

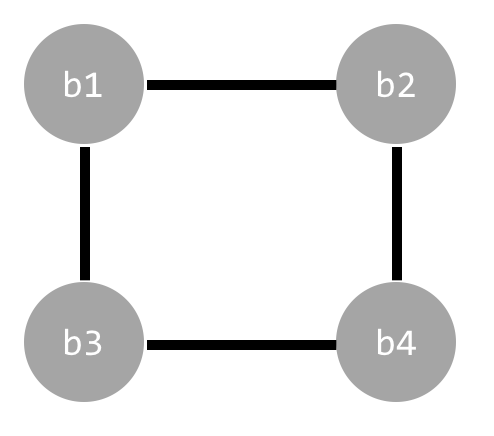


图 1 四节点的环路拓扑

* 依据【三、实验结果】中dump\_output\_4的输出，整理生成树拓扑如下：

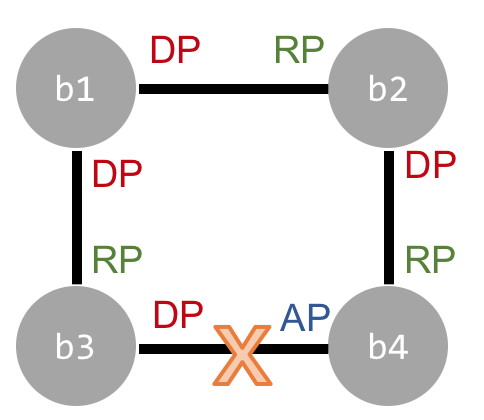


图 2 四节点环路的生成树拓扑

### （二）六节点拓扑结果分析

* 六节点环路拓扑结构如下：

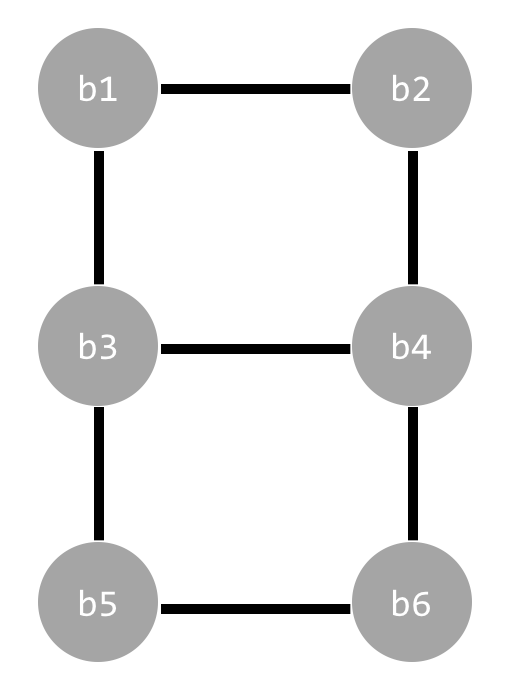


图 3 链路冗余度为2的六节点环路拓扑

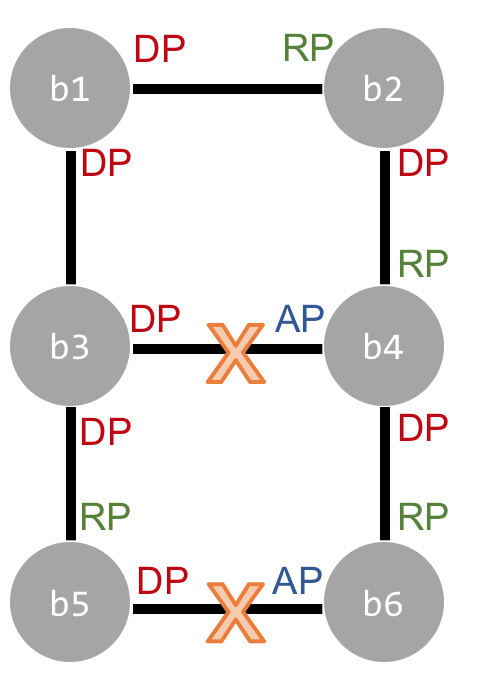
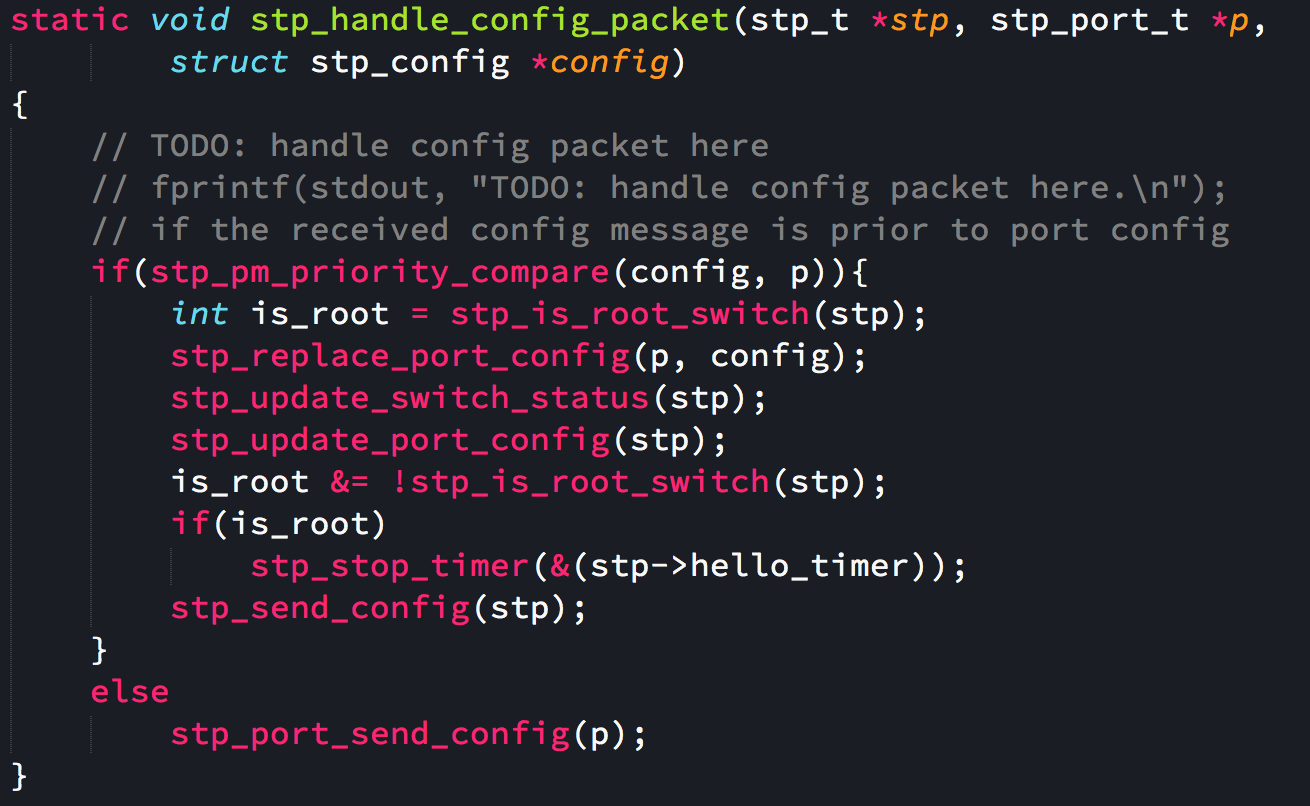
* 依据【三、实验结果】中dump\_output\_4的输出，整理生成树拓扑如下：

图 4上述六节点环路的生成树拓扑

### （二）代码实现分析

* 主函数 stp\_handle\_config\_packet()



先比较config消息和本地端口的优先级（stp\_pm\_priority\_compare()），

如果接收到的config消息优先级高，说明该网段通过对方端口连接根节点代价更小，则进行如下操作：

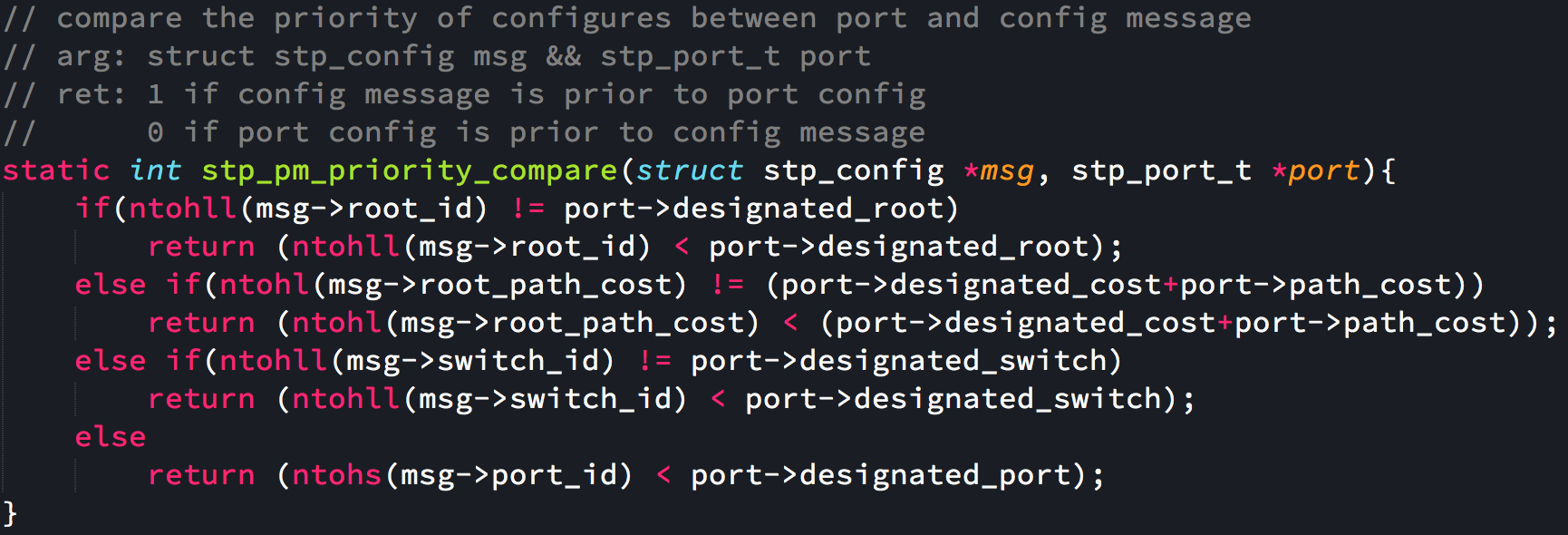
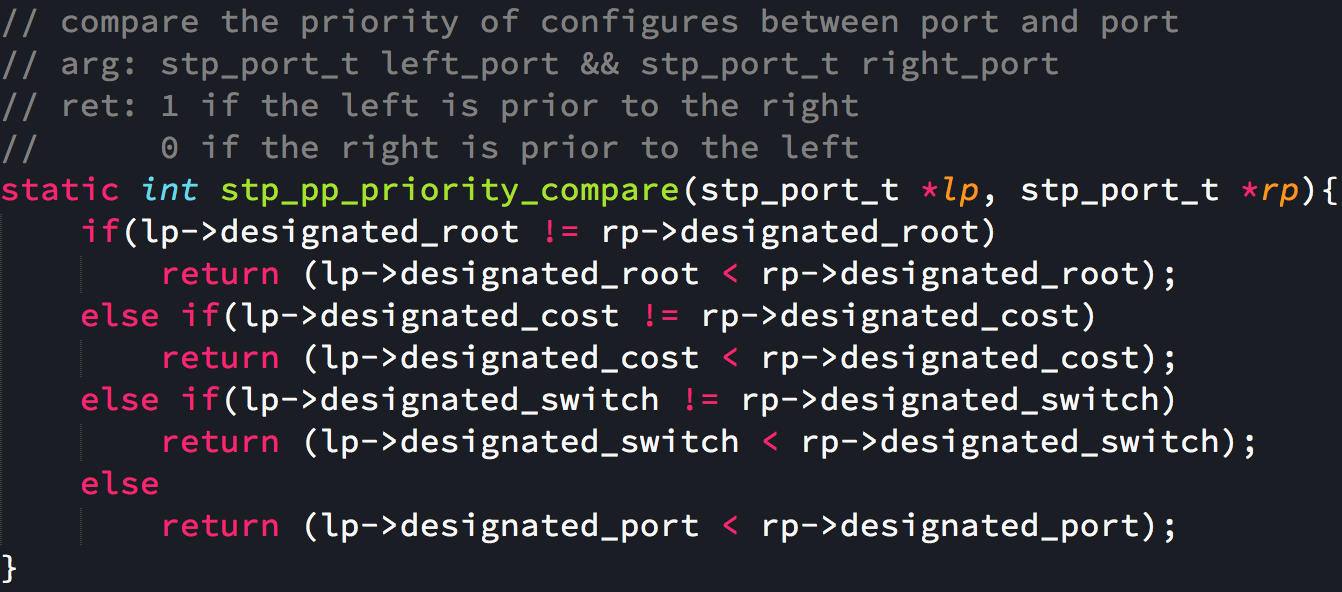
1. 将本端口的Config替换为收到的Config消息（stp\_replace\_port\_config(p, config)）
2. 更新节点状态(stp\_update\_switch\_status(stp))
3. 更新剩余端口的Config(stp\_update\_port\_config(stp))
4. 如果节点由根节点变为非根节点，停止hello定时器
5. 将更新后的Config从每个指定端口转发出去

否则，说明该网段通过本端口连接根节点代价更小，则进行以下操作：

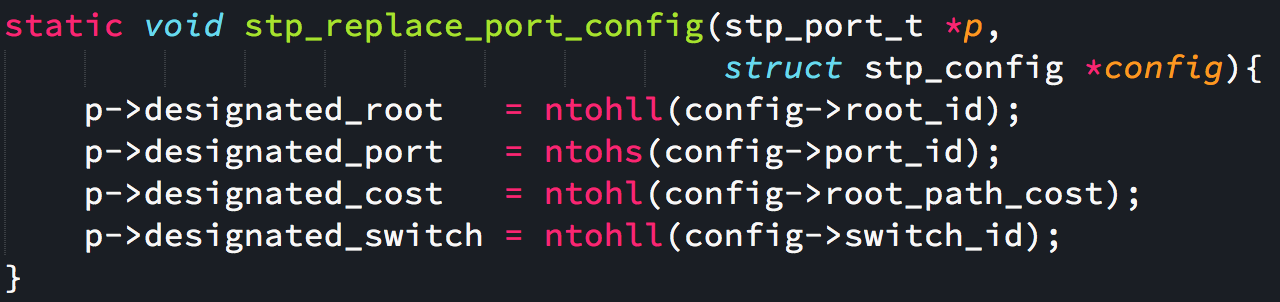
1. 从该端口发送Config消息

【注意】判断节点由根结点变为非根节点时，需要注意，由于在2）中更新节点状态时已经将该节点的switch\_id改成了另外一个更优先的节点。因此，判断节点由根结点变为非根节点的条件为：

更新节点状态前是根结点，并且跟新节点状态之后，变成了非根节点。

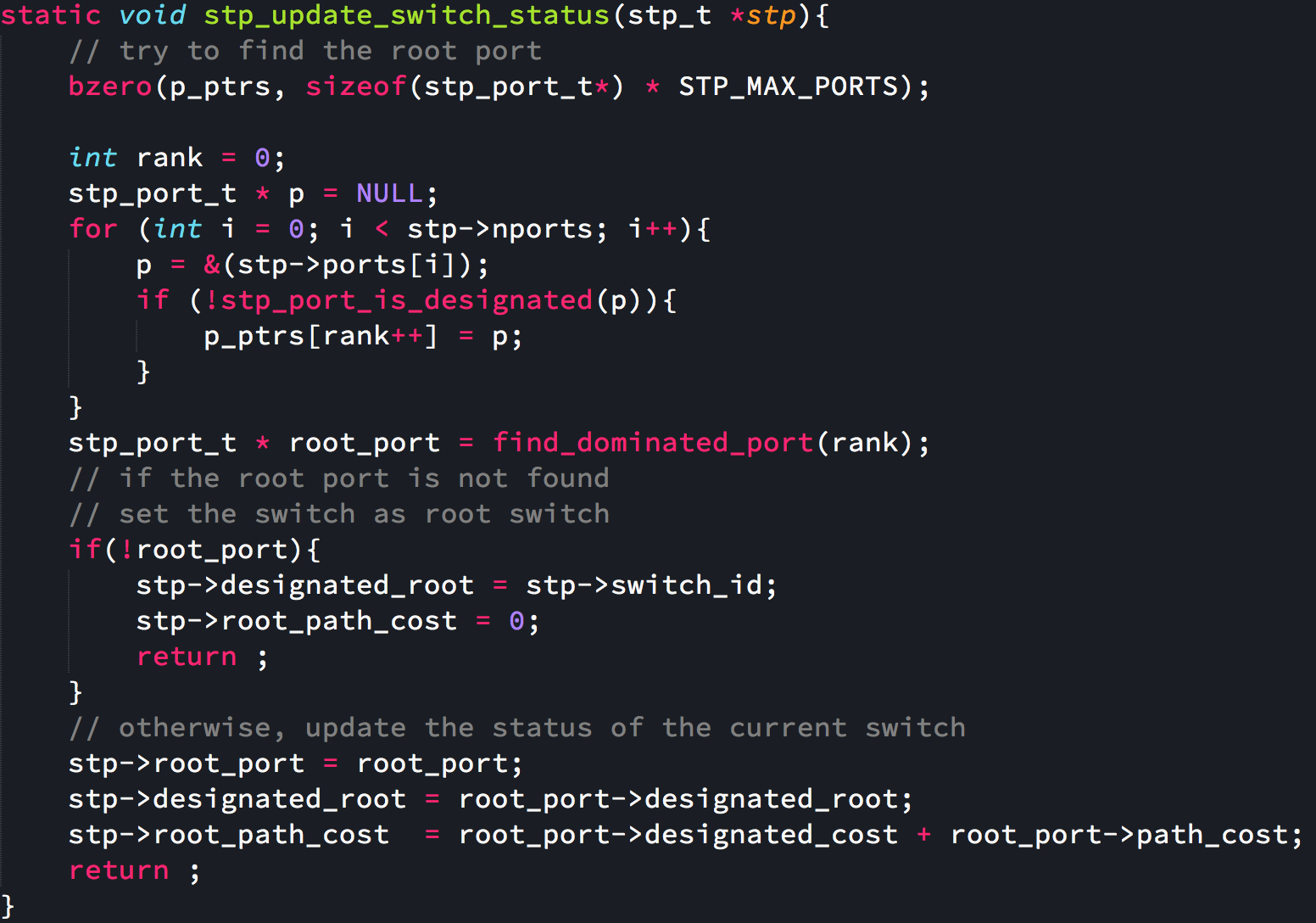
* stp\_pm\_priority\_compare() && stp\_pp\_priority\_compare()

依次比较认为的根结点ID、到根结点的代价、上一跳的节点以及上一跳的端口ID，来得到优先级。（如上图代码所示）比较时注意网络字节序的转换。

* stp\_replace\_port\_config()

将本地端口的信息更新为接收到的config消息。

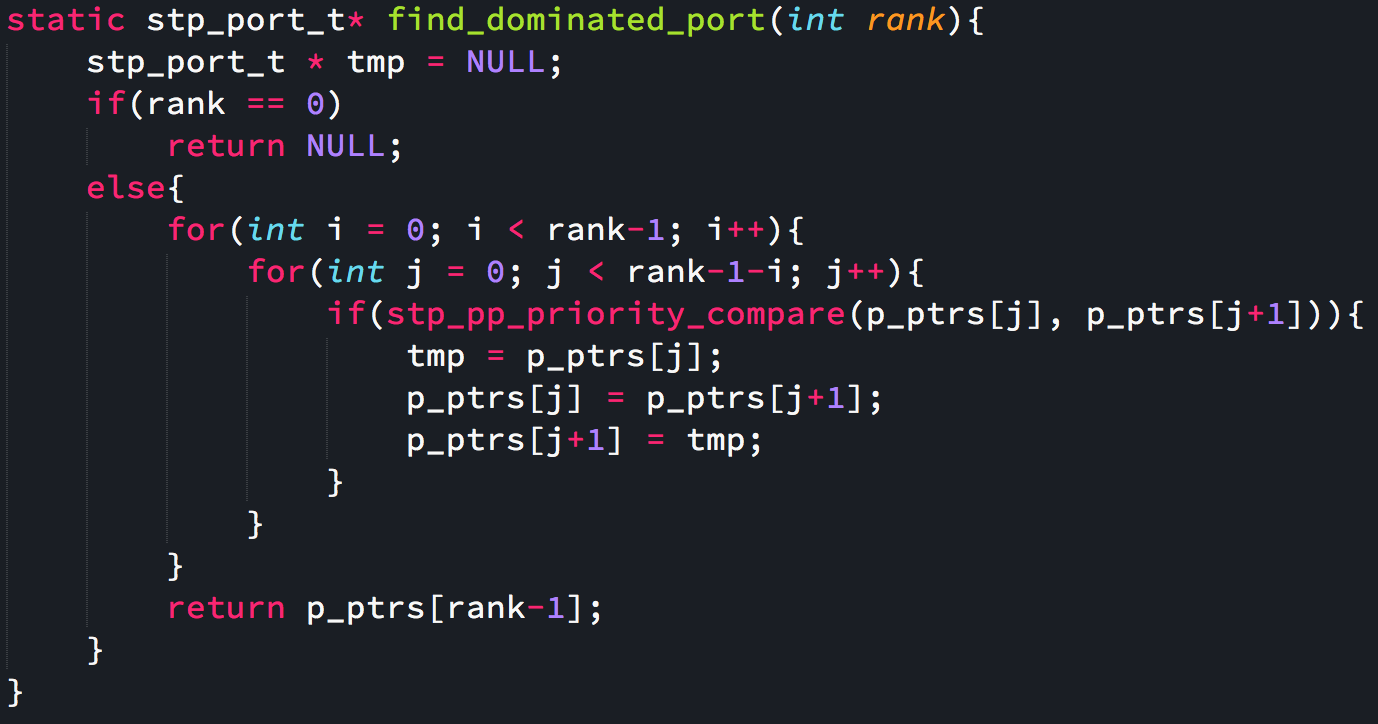
【注意】因为config消息是网络传输过来的包，因此整型数需要将网络字节序转化为本地字节序才能正确使用。

* stp\_update\_switch\_status()

1. 寻找根端口
   1. 找到所有非指定端口；
   2. 对所有非指定端口进行优先级排序，取优先级最高的作为根端口；

(find\_dominated\_port(int rank))

1. 如果没有根端口，则该节点为根结点；
2. 否则，（找到了根端口）设置节点的根端口、指定端口、路径代价。

* find\_dominated\_port()

对所有的非指定端口进行比较，返回指向优先级最高的端口的指针。

如果没有非指定端口，则返回空指针。

* stp\_update\_port\_config()

扫描所有端口，更新其信息：

1. 如果是指定端口，则只需要更新其指定端口信息和路径代价信息；
2. 如果是非指定端口，且其网段通过本节点到根节点的代价比通过对端节点的代价小，那么该端口成为指定端口，并更新该端口的所有信息；