生成树机制实验

2020年10月25日

蔡润泽

本实验 `Github` 地址

实验内容

1、基于已有代码，实现生成树运行机制，对于给定拓扑 `four_node_ring.py` ，计算输出相应状态下的最小生成树拓扑

2、自己构造一个不少于7个节点，冗余链路不少于2条的拓扑，使用stp程序计算输出最小生成树拓扑

设计思路

已有的代码框机已经解决了STP通信等问题，本设计需要解决的问题为处理收到的包，并根据收到的对于端口的config，与本端口和本端口对应节点的config进行比较，来更新本端口和本节点的config。

代码实现的部分在代码框架中即需要实现 `stp_handle_config_packet` 函数。

本设计的对应代码如下：

```
static void stp_handle_config_packet(stp_t *stp, stp_port_t *p,
        struct stp_config *config) {

    if (recv_has_higher_pirority(p, config)) {
        update_port_config(p, config);
        update_root(stp, config);
        update_other_ports(stp);
        if(!stp_is_root_switch(stp)) {
            stp_stop_timer(&stp->hello_timer);
            stp_send_config(stp);
        }
    }
}
```

意思是，本端口处理收到 `stp_config` 时：

- 首先比较本端口和发送端口的优先级。
- 倘若优先级更高则进行以下处理：
    - 更新本端口的config

    - 更新本端口对应的本节点的config
    - 更新本节点其他端口的config
    - 若本节点不再是根节点，则停止本节点的计时器，并将config通过指定节点发出

上述代码中涉及到的子函数代码及其解释

`int recv_has_higher_pirority(stp_port_t * p, struct stp_config *config)`

该函数的作用是比较本端口和发送端口的config优先级，如果本端口优先级更高返回`0`,否则返回`1`。

具体的代码实现如下：

```
int recv_has_higher_pirority(stp_port_t * p, struct stp_config *config) {
    if (p->designated_root != ntohll(config->root_id)) {
        if(p->designated_root < ntohll(config->root_id)) {
            return 0;
        } else {
            return 1;
        }
    } else if (p->designated_cost != ntohl(config->root_path_cost)) {
        if (p->designated_cost < ntohl(config->root_path_cost)) {
            return 0;
        } else {
            return 1;
        }
    } else if (p->designated_switch != ntohll(config->switch_id)) {
        if (p->designated_switch < ntohll(config->switch_id)) {
            return 0;
        } else {
            return 1;
        }
    } else if (p->designated_port != ntohs(config->port_id)) {
        if (p->designated_port < ntohs(config->port_id)) {
            return 0;
        } else {
            return 1;
        }
    } else {
        return 1;
    }
}
```

上述代码的逻辑是，首先比较节点ID，其次比较到根节点的开销，再比较上一跳节点的ID，最后比较端口ID。所有的比较均为小的值优先级更高。

值得注意的是，由于接收到的包来自于网络抓包，因此收到的包需要先进行字节序的转换才能与本端口的config进行比较。

`void update_port_config(stp_port_t *p, struct stp_config *config)`

该函数的作用是根据接收端口发送config的值，来更新本端口的config。(同时本端口会变为非指定端口)

具体的代码实现如下：

```
void update_port_config(stp_port_t *p, struct stp_config *config) {
    p->designated_root = ntohll(config->root_id);
    p->designated_switch = ntohll(config->switch_id);
    p->designated_port = ntohs(config->port_id);
    p->designated_cost = ntohl(config->root_path_cost);
}
```

`void update_root(stp_t *stp, struct stp_config *config)`

该函数的作用是更新节点的状态。

具体的代码实现如下：

```
void update_root(stp_t *stp, struct stp_config *config) {
    stp_port_t * non_designated_ports[STP_MAX_PORTS];
    int num_non_ports = 0;
    for (int i =0 ; i < stp->nports; i++) {
        if(!stp_port_is_designated(&stp->ports[i])) {
            non_designated_ports[num_non_ports] = &stp->ports[i];
            num_non_ports ++;
        }
    }
    int judge = 0;
    if (num_non_ports == 1) {
        stp->root_port = non_designated_ports[0];
    } else {
        for(int i = 0; i < num_non_ports -1; i++) {
            if (judge == 0) {
                if(!compare_ports_pirority(non_designated_ports[i], non_designated_ports[
                    stp->root_port = non_designated_ports[i];
                    judge = 1;
                }
            } else {
                if(!compare_ports_pirority(non_designated_ports[i], stp->root_port)) {
                    stp->root_port = non_designated_ports[i];
                }
            }
        }
    }
    if (stp->root_port == NULL) {
        stp->designated_root = stp->switch_id;
        stp->root_path_cost = 0;
    } else {
        stp->designated_root = stp->root_port->designated_root;
        stp->root_path_cost = stp->root_port->designated_cost + stp->root_port->path_cost
    }
}
```

上述代码的逻辑为：

- 遍历所有端口，满足如下条件的为根端口

    - 该端口是非指定端口
    - 该端口的优先级要高于所有剩余非指定端口(①)
- 如果不存在根端口，则该节点为根节点
- 否则，选择通过root_port连接到根节点，更新节点状态为：
    - `stp->root_port = root_port`
    - `stp->designate_root = root_port->designated_root`
    - `stp->root_path_cost = root_port->designated_cost + root_port->path_cost`

其中，上述代码段用到了 `compare_ports_pirority` 函数来比较本节点中不同端口的优先级，代码实现与 `recv_has_higher_pirority` 函数类似，唯一的不同是不需要进行字节序的转换。

`void update_other_ports(stp_t *stp)`】

该函数的作用是更新节点中其他端口的状态。

```
void update_other_ports(stp_t *stp) {
    for (int i =0 ; i < stp->nports; i++) {
        if(stp_port_is_designated(&stp->ports[i])) {
            stp->ports[i].designated_root = stp->designated_root;
            stp->ports[i].designated_cost = stp->root_path_cost;
        }
    }
}
```

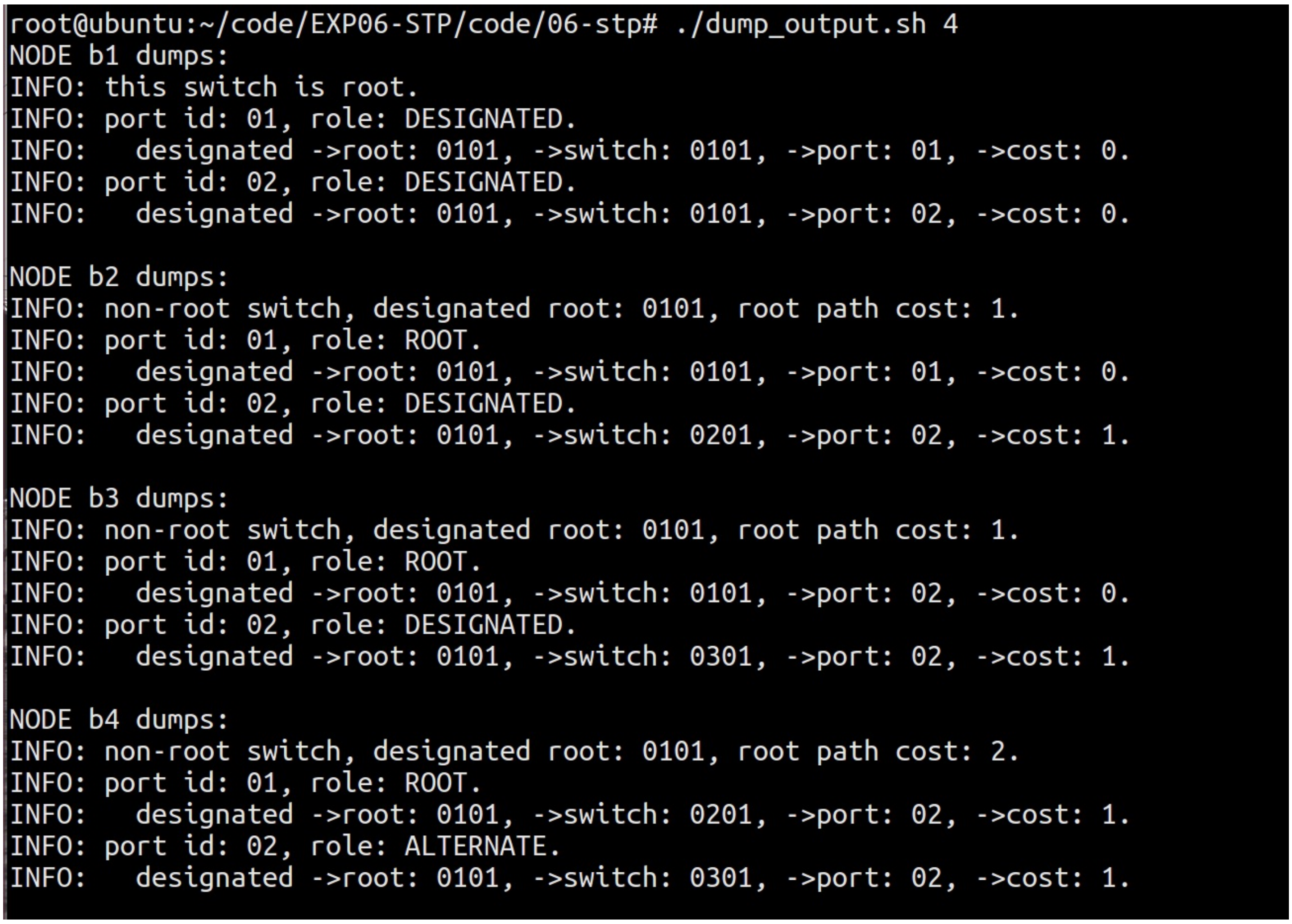
对于所有指定端口，更新其认为的根节点和路径开销。

另外，如果一个端口为非指定端口，且其Config较网段内其他端口优先级更高，那么该端口成为指定端口。（本功能无需在此函数中添加额外代码即可实现）

结果验证

利用网络拓扑进行四节点测试

本设计的测试结果如下：

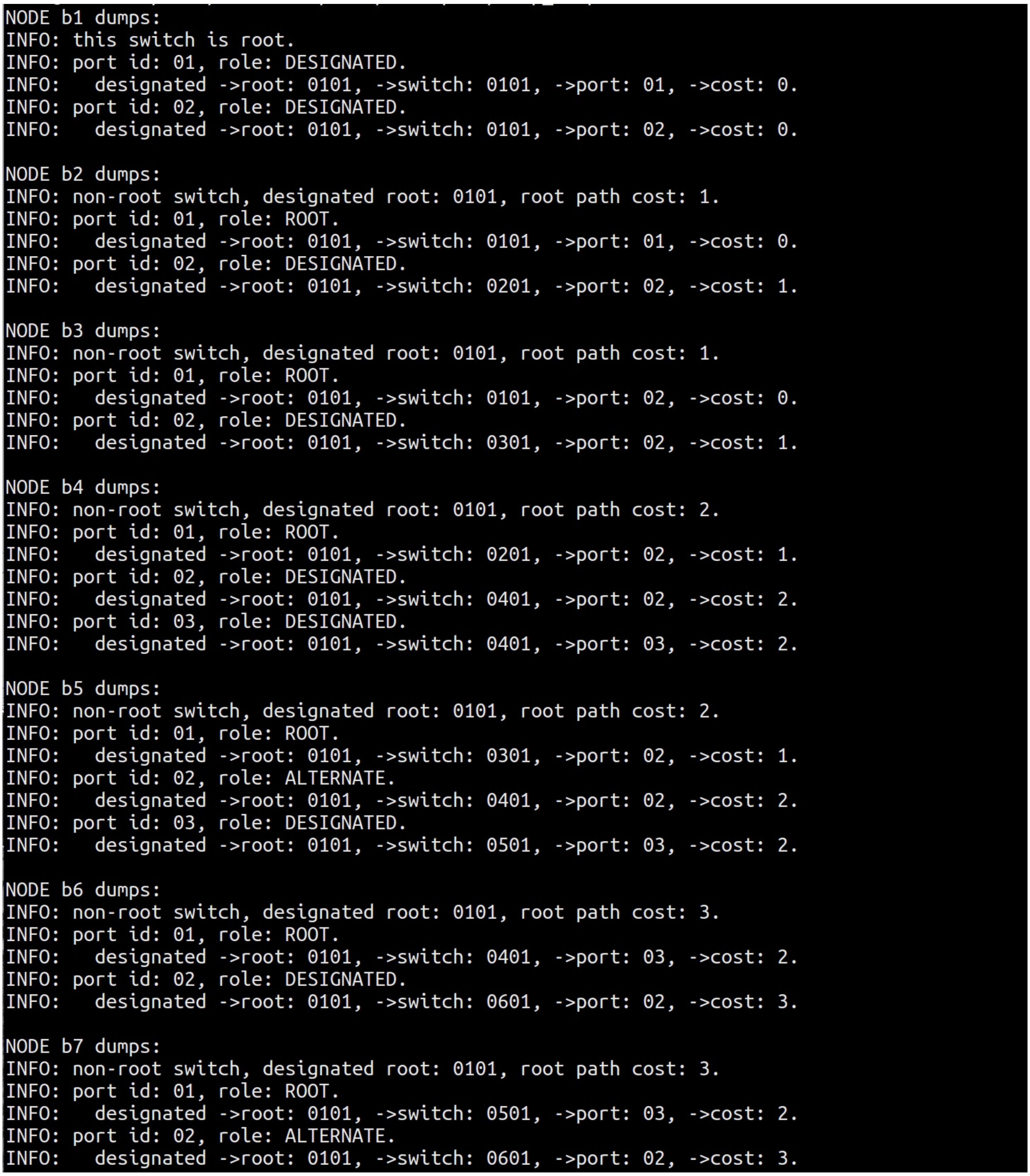


```
root@ubuntu:~/code/EXP06-STP/code/06-stp# ./dump_output.sh 4
NODE b1 dumps:
INFO: this switch is root.
INFO: port id: 01, role: DESIGNATED.
INFO:   designated ->root: 0101, ->switch: 0101, ->port: 01, ->cost: 0.
INFO: port id: 02, role: DESIGNATED.
INFO:   designated ->root: 0101, ->switch: 0101, ->port: 02, ->cost: 0.

NODE b2 dumps:
INFO: non-root switch, designated 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, designated 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, designated 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.
```

可以看出该4节点的测试成功。

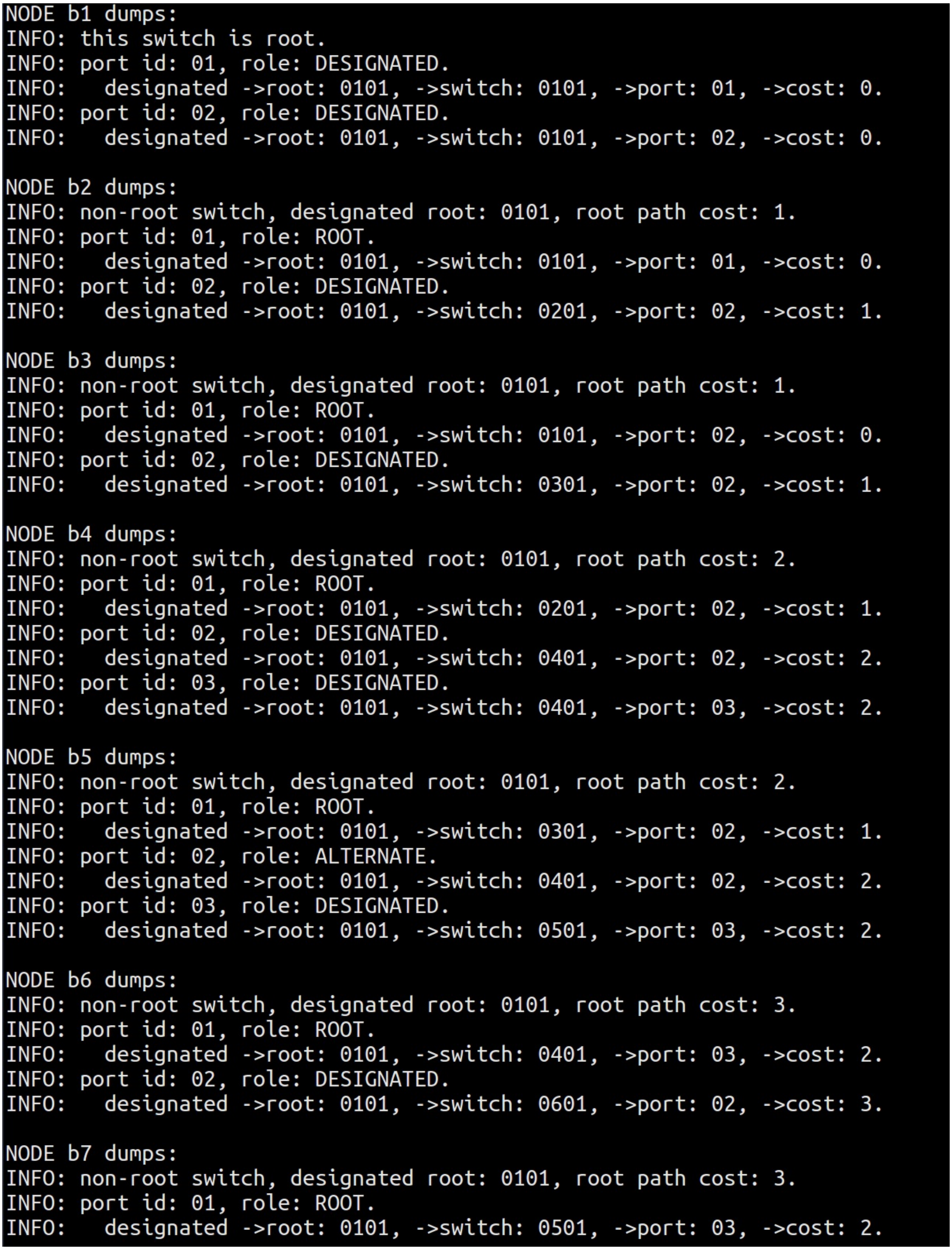
利用网络拓扑进行七节点测试

本设计的测试结果如下:



```
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, designated 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, designated 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, designated 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: DESIGNATED.
INFO:   designated ->root: 0101, ->switch: 0401, ->port: 02, ->cost: 2.
INFO: port id: 03, role: DESIGNATED.
INFO:   designated ->root: 0101, ->switch: 0401, ->port: 03, ->cost: 2.

NODE b5 dumps:
INFO: non-root switch, designated root: 0101, root path cost: 2.
INFO: port id: 01, role: ROOT.
INFO:   designated ->root: 0101, ->switch: 0301, ->port: 02, ->cost: 1.
INFO: port id: 02, role: ALTERNATE.
INFO:   designated ->root: 0101, ->switch: 0401, ->port: 02, ->cost: 2.
INFO: port id: 03, role: DESIGNATED.
INFO:   designated ->root: 0101, ->switch: 0501, ->port: 03, ->cost: 2.

NODE b6 dumps:
INFO: non-root switch, designated 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: DESIGNATED.
INFO:   designated ->root: 0101, ->switch: 0601, ->port: 02, ->cost: 3.

NODE b7 dumps:
INFO: non-root switch, designated root: 0101, root path cost: 3.
INFO: port id: 01, role: ROOT.
INFO:   designated ->root: 0101, ->switch: 0501, ->port: 03, ->cost: 2.
INFO: port id: 02, role: ALTERNATE.
INFO:   designated ->root: 0101, ->switch: 0601, ->port: 02, ->cost: 3.
```

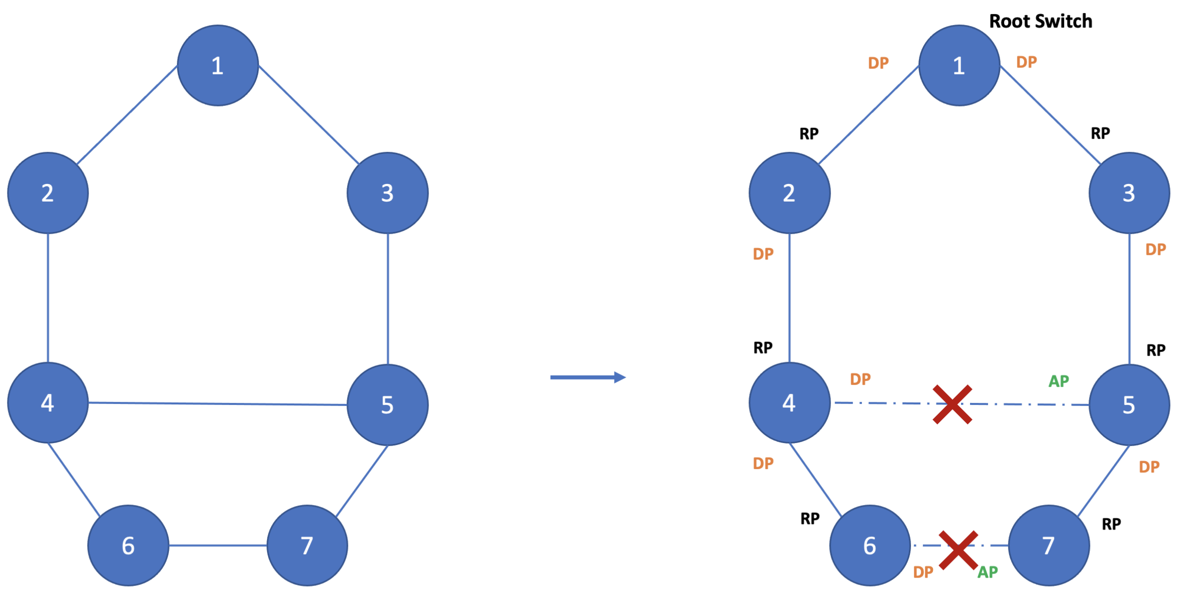
`reference`的结果如下：



```
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, designated 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, designated 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, designated 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: DESIGNATED.
INFO:   designated ->root: 0101, ->switch: 0401, ->port: 02, ->cost: 2.
INFO: port id: 03, role: DESIGNATED.
INFO:   designated ->root: 0101, ->switch: 0401, ->port: 03, ->cost: 2.

NODE b5 dumps:
INFO: non-root switch, designated root: 0101, root path cost: 2.
INFO: port id: 01, role: ROOT.
INFO:   designated ->root: 0101, ->switch: 0301, ->port: 02, ->cost: 1.
INFO: port id: 02, role: ALTERNATE.
INFO:   designated ->root: 0101, ->switch: 0401, ->port: 02, ->cost: 2.
INFO: port id: 03, role: DESIGNATED.
INFO:   designated ->root: 0101, ->switch: 0501, ->port: 03, ->cost: 2.

NODE b6 dumps:
INFO: non-root switch, designated 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: DESIGNATED.
INFO:   designated ->root: 0101, ->switch: 0601, ->port: 02, ->cost: 3.

NODE b7 dumps:
INFO: non-root switch, designated root: 0101, root path cost: 3.
INFO: port id: 01, role: ROOT.
INFO:   designated ->root: 0101, ->switch: 0501, ->port: 03, ->cost: 2.
```

根据比对，本设计的结果与reference的结果相同，即实验成功。

七节点STP生成的结果


1
2
3
4
5
6
7
Root Switch
DP
RP
AP