生成树机制实验

孙佳钰 2015K8009929051

2018 年 10 月 17 日

1 实验内容

实现生成树机制中对收到的的 config 消息的处理算法。

2 实验流程

实验原理为生成树协议。所要实现的只有一个 *stp_handle_config_packet* 函数，参数为结点指针 *stp*，端口指针 *p*，收到的 config 消息指针 *config*。函数目的为将某端口收到的 config 信息与端口内储存的 config 信息进行比较，然后更新节点状态并转发新的 config 消息。以下为该函数的步骤。

1. 收到 config 消息后，将其与本端口 config 消息进行优先级比较。这里引入 *pri* 参数，为 1 时表示收到的 config 消息的优先级高，为 0 时表示该网段中本节点优先级高。

```
int pri = 0;
if (p->designated_root > ntohll(config->root_id))                pri = 1;
else if (p->designated_root < ntohll(config->root_id))           pri = 0;
else if (p->designated_cost > ntohl(config->root_path_cost))  pri = 1;
else if (p->designated_cost < ntohl(config->root_path_cost))  pri = 0;
else if (p->designated_switch > ntohll(config->switch_id))    pri = 1;
else if (p->designated_switch < ntohll(config->switch_id))    pri = 0;
else if (p->designated_port > ntohs(config->port_id))            pri = 1;
```

2. 如果收到的 config 优先级更高，说明该网段通过对方端口连接到根节点开销更小，首先要将本端口的 config 消息替换为收到的 config 消息，本端口为非指定端口。

```
if (pri == 1) {
    // Config received has higher priority.
    // Update Config for this port.
    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);
```

3. 然后更新节点状态。首先要遍历所有端口，找到根端口。若没有根端口，则该节点为根节点；否则，选择通过找到的根端口连接到根节点。根端口要满足：该端口是非指定端口，且优先级要高于所有剩余非指定端口；判断优先级的过程中基本复用了上面写过的判断过程。

```
    // Update Config for this node.
    // To find a root_port.
    int root = 0;
    int has_root = 1;
    for (int i = 0; i < stp->nports; i++) {
        if (!stp_port_is_designated(&(stp->ports[i]))) {
            root = i;
            break;
        }
        // There is no root_port.
        if (i == stp->nports - 1) has_root = 0;
    }
    for (int i = root + 1; i < stp->nports; i++) {
        if (stp_port_is_designated(&(stp->ports[i]))) continue;
        int pri = 1;
        if (stp->ports[i].designated_root > \
            stp->ports[root].designated_root)     pri = 0;
        else if (stp->ports[i].designated_root < \
            stp->ports[root].designated_root)     pri = 1;
        else if (stp->ports[i].designated_cost > \
            stp->ports[root].designated_cost)     pri = 0;
        else if (stp->ports[i].designated_cost < \
            stp->ports[root].designated_cost)     pri = 1;
        else if (stp->ports[i].designated_switch > \
            stp->ports[root].designated_switch)   pri = 0;
        else if (stp->ports[i].designated_switch < \
            stp->ports[root].designated_switch)   pri = 1;
        else if (stp->ports[i].designated_port > \
            stp->ports[root].designated_port)     pri = 0;
        if (pri) root = i;
    }
    if (!has_root) {
```

```
33          // This is root node.
34          stp->root_port = NULL;
35          stp->designated_root = stp->switch_id;
36          stp->root_path_cost = 0;
37      } else {
38          stp->root_port = &(stp->ports[root]);
39          stp->designated_root = stp->root_port->designated_root;
40          stp->root_path_cost = stp->root_port->designated_cost + \
41              stp->root_port->path_cost;
42      }
```

4. 然后更新各端口的 config。如果一个端口是非指定端口，且网段通过本节点到根节点的开销比通过对端节点小，那么该端口变成指定端口。然后对于所有指定端口，更新其认为的根节点和路径开销。

```
1     for (int i = 0; i < stp->nports; i++) {
2         if (stp_port_is_designated(&(stp->ports[i]))) {
3             stp->ports[i].designated_root = stp->designated_root;
4             stp->ports[i].designated_cost = stp->root_path_cost;
5         } else if (stp->root_path_cost < stp->ports[i].designated_cost) {
6             stp->ports[i].designated_switch = stp->switch_id;
7             stp->ports[i].designated_port = stp->ports[i].port_id;
8             stp->ports[i].designated_root = stp->designated_root;
9             stp->ports[i].designated_cost = stp->root_path_cost;
10        }
11    }
```

5. 如果节点由根节点变为非根节点，则停止 hello 计时器。代码实现为若该节点为非根节点，则停止 hello 计时器，可避免储存原节点信息。然后将更新后的 config 从每个指定端口转发出去。

```
1     if (!stp_is_root_switch(stp))
2         stp_stop_timer(&(stp->hello_timer));
3     for (int i = 0; i < stp->nports; i++)
4         if (stp_port_is_designated(&(stp->ports[i])))
5             stp_port_send_config(&(stp->ports[i]));
6 }
```

6. 若收到的 config 消息优先级比本端口低，则该端口为指定端口，发送 config 消息。

```
1 else stp_port_send_config(p);
```

3 实验结果及分析

```
sjy@sjy-PC:/media/sjy/新加卷/国科大/网络实验/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.
```

上图是采用代码包中 4 个节点的拓扑所得的结果。

```
sjy@sjy-PC:/media/sjy/新加卷/国科大/网络实验/06-stp$ sudo pkill -SIGTERM st
p
sjy@sjy-PC:/media/sjy/新加卷/国科大/网络实验/06-stp$ ./dump_output.sh 6 >re
s.txt
sjy@sjy-PC:/media/sjy/新加卷/国科大/网络实验/06-stp$ sudo pkill -SIGTERM st
p
sjy@sjy-PC:/media/sjy/新加卷/国科大/网络实验/06-stp$ ./dump_output.sh 6 >re
s_ref.txt
sjy@sjy-PC:/media/sjy/新加卷/国科大/网络实验/06-stp$ diff res.txt res_ref.t
xt
sjy@sjy-PC:/media/sjy/新加卷/国科大/网络实验/06-stp$
```

上图是采用自己所写的 6 个节点的拓扑所得的结果，具体拓扑可见所附压缩包中文件 *six_node_topo.py*。由于结果太长，判断是否正确也比较麻烦，故分别用所写代码生成的可执行文件和老师所给的 *stp_reference* 可执行文件来生成，最后判断两个结果是否相同。