# 实验报告

孔静 2014K8009929022

December 3, 2017

## Contents

## 1 实验题目

- 网络路由实验二

## 2 实验内容

- 基于已有框架代码，实现计算路由器路由表项的相关操作
  - Hello/LSU 消息的相关操作
- 运行实验
  - 运行网络拓扑 (topo/mospf_topo.py)
  - 在各个路由器节点上执行 disable_arp.sh, disable_icmp.sh, disable_ip_forward.sh)，禁止协议栈的相应功能
  - 运行./mospfd，使得各个节点生成一致的链路状态数据库
  - 等待一段时间后，每个节点生成完整的路由表项
  - 在节点 h1 上 ping/traceroute 节点 h2

## 3 实验流程

### 3.1 文件列表

```
lab11
 ├─homework.pdf
 └─11-mospf
    ├─mospf_daemon.c
    ├─mospf_database.c
    └─...
```

## 3.2 相关代码

- 补充通知没有 hello 的邻居节点

```
list_for_each_entry(iface, &(instance->iface_list), list){
  if(list_empty(&(iface->nbr_list))){
    lsa[number].subnet = htonl(iface->ip);
    lsa[number].mask = htonl(iface->mask);
    lsa[number].rid = htonl(0);
    number++;
  }else{
    list_for_each_entry(nbr, &(iface->nbr_list), list){
      lsa[number].subnet = htonl(nbr->nbr_ip);
      lsa[number].mask = htonl(nbr->nbr_mask);
      lsa[number].rid = htonl(nbr->nbr_id);
      number++;
    }
  }
}
```

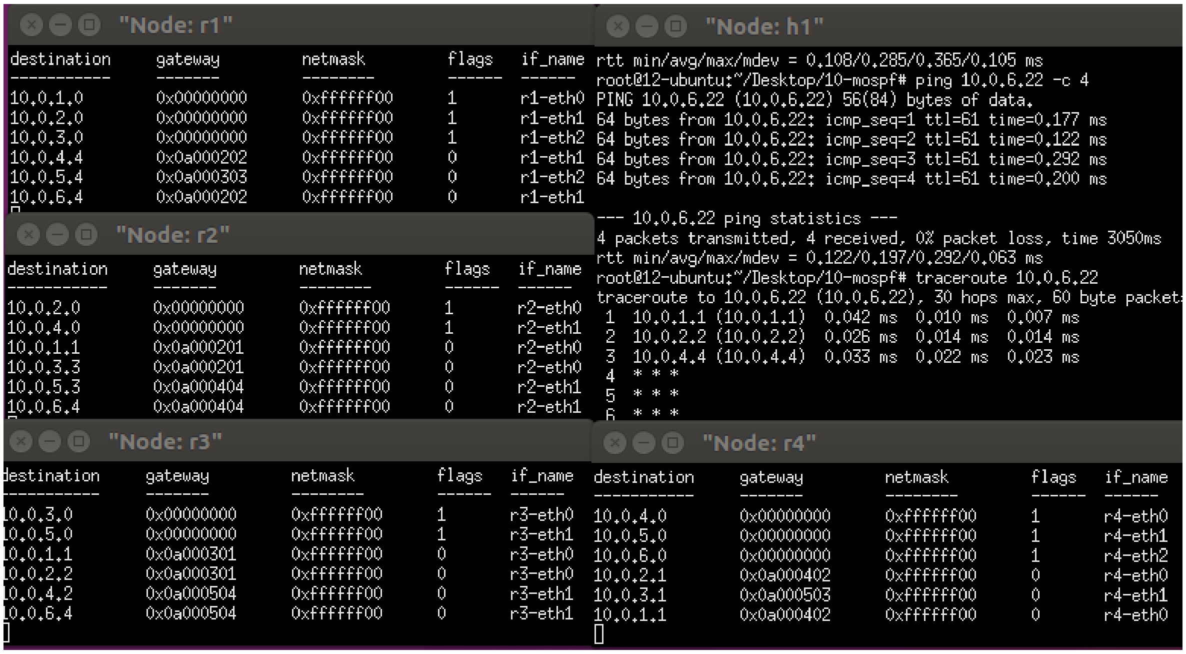
- 广度优先搜索

```
void update_rtable()
{
    clear_rtable();
    init_rtable();
    struct queue_t *head, *current, *prev;
    head = (struct queue_t *)malloc(sizeof(struct queue_t));
    init_list_head((struct list_head *)head);

    char *visited = (char *)malloc(MAX_NODE_NUM * sizeof(char));
    char *rid_visited = (char *)malloc(MAX_NODE_NUM * sizeof(char));
    memset(visited, 0, MAX_NODE_NUM * sizeof(char));
    memset(rid_visited, 0, MAX_NODE_NUM * sizeof(char));

    iface_info_t *iface;
    mospf_nbr_t *nbr;
    list_for_each_entry(iface, &instance->iface_list, list){
        list_for_each_entry(nbr, &iface->nbr_list, list){
            prev = (struct queue_t *)malloc(sizeof(struct queue_t));
            prev->rid = nbr->nbr_id;
            prev->gw = nbr->nbr_ip;
            prev->iface = iface;
            list_add_tail((struct list_head *)prev, (struct list_head *)head);
        }
        visited[hash(iface->mask & iface->ip)] = 1;
    }
    struct mospf_lsa *lsa;
    mospf_db_entry_t *db;
    current = (struct queue_t *)((struct list_head *)head)->next;
    while(current != head){
        list_for_each_entry(db, &mospf_db, list){
            if(db->rid == current->rid){
                lsa = db->array;
                for (int i = 0; i < db->nadv; i++, lsa++){
                    if (rid_visited[hash(lsa->rid)] == 0){
                        prev = (struct queue_t *)malloc(sizeof(struct queue_t));
                        prev->rid = lsa->rid;
                        prev->gw = current->gw;
                        prev->iface = current->iface;
                        list_add_tail((struct list_head *)prev, (struct list_head *)head);
                        rid_visited[hash(lsa->rid)] = 1;
                    }
                    if (visited[hash(lsa->subnet)] == 0){
                        rt_entry_t *rt_entry;
                        rt_entry = (rt_entry_t *)malloc(sizeof(rt_entry_t));
                        init_list_head(&rt_entry->list);
                        rt_entry->dest = lsa->subnet;
                        rt_entry->gw = current->gw;
                        rt_entry->mask = lsa->mask;
                        rt_entry->iface = current->iface;
                        rt_entry->flags = 0;
                        memcpy(rt_entry->if_name, current->iface->name, 16);
                        add_rt_entry(rt_entry);
                        visited[hash(lsa->subnet)] = 1;
                    }
                }
                break;
            }
        }
        prev = current;
        current = (struct queue_t *)((struct list_head *)current)->next;
        free(prev);
    }
    free(head);
    free(visited);
    free(rid_visited);
    print_rtable();
}
```

## 4 实验结果



"Node: r1"

```
destination     gateway         netmask         flags   if_name
-----------     -------         --------        ------  -------
10.0.1.0        0x00000000      0xffffff00      1       r1-eth0
10.0.2.0        0x00000000      0xffffff00      1       r1-eth1
10.0.3.0        0x00000000      0xffffff00      1       r1-eth2
10.0.4.4        0x0a000202      0xffffff00      0       r1-eth1
10.0.5.4        0x0a000303      0xffffff00      0       r1-eth2
10.0.6.4        0x0a000202      0xffffff00      0       r1-eth1
```

"Node: r2"

```
destination     gateway         netmask         flags   if_name
-----------     -------         --------        ------  -------
10.0.2.0        0x00000000      0xffffff00      1       r2-eth0
10.0.4.0        0x00000000      0xffffff00      1       r2-eth1
10.0.1.1        0x0a000201      0xffffff00      0       r2-eth0
10.0.3.3        0x0a000201      0xffffff00      0       r2-eth0
10.0.5.3        0x0a000404      0xffffff00      0       r2-eth1
10.0.6.4        0x0a000404      0xffffff00      0       r2-eth1
```

"Node: h1"

```
rtt min/avg/max/mdev = 0.108/0.285/0.365/0.105 ms
root@12-ubuntu:~/Desktop/10-mospf# ping 10.0.6.22 -c 4
PING 10.0.6.22 (10.0.6.22) 56(84) bytes of data.
64 bytes from 10.0.6.22: icmp_seq=1 ttl=61 time=0.177 ms
64 bytes from 10.0.6.22: icmp_seq=2 ttl=61 time=0.122 ms
64 bytes from 10.0.6.22: icmp_seq=3 ttl=61 time=0.292 ms
64 bytes from 10.0.6.22: icmp_seq=4 ttl=61 time=0.200 ms

--- 10.0.6.22 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3050ms
rtt min/avg/max/mdev = 0.122/0.197/0.292/0.063 ms
root@12-ubuntu:~/Desktop/10-mospf# traceroute 10.0.6.22
traceroute to 10.0.6.22 (10.0.6.22), 30 hops max, 60 byte packet
 1  10.0.1.1 (10.0.1.1)  0.042 ms  0.010 ms  0.007 ms
 2  10.0.2.2 (10.0.2.2)  0.026 ms  0.014 ms  0.014 ms
 3  10.0.4.4 (10.0.4.4)  0.033 ms  0.022 ms  0.023 ms
 4  * * *
 5  * * *
 6  * * *
```

"Node: r3"

```
destination     gateway         netmask         flags   if_name
-----------     -------         --------        ------  -------
10.0.3.0        0x00000000      0xffffff00      1       r3-eth0
10.0.5.0        0x00000000      0xffffff00      1       r3-eth1
10.0.1.1        0x0a000301      0xffffff00      0       r3-eth0
10.0.2.2        0x0a000301      0xffffff00      0       r3-eth0
10.0.4.2        0x0a000504      0xffffff00      0       r3-eth1
10.0.6.4        0x0a000504      0xffffff00      0       r3-eth1
```

"Node: r4"

```
destination     gateway         netmask         flags   if_name
-----------     -------         --------        ------  -------
10.0.4.0        0x00000000      0xffffff00      1       r4-eth0
10.0.5.0        0x00000000      0xffffff00      1       r4-eth1
10.0.6.0        0x00000000      0xffffff00      1       r4-eth2
10.0.2.1        0x0a000402      0xffffff00      0       r4-eth0
10.0.3.1        0x0a000503      0xffffff00      0       r4-eth1
10.0.1.1        0x0a000402      0xffffff00      0       r4-eth0
```

## 5 结果分析

- 数据库建立拓扑后，用广度优先搜索获得最短路径