

网络路由实验报告

姓名：钟赞 学号：2016K8009915009

实验内容

实验内容一

- 基于已有代码框架，实现路由器生成和处理mOSPF Hello/LSU消息的相关操作，构建一致性链路状态数据库

实验内容二

- 基于实验一，实现路由器计算路由表项的相关操作

实验步骤

实验内容一

发送mOSPF Hello消息

- 每个结点周期性地发送mOSPF Hello消息，宣告自己的存在，消息包括节点ID, 端口的子网掩码。
 - 用 `mospf_init_hdr` 函数封装mOSPF包头，`mospf_init_hello` 函数封装mOSPF Hello包。
 - 目的IP地址为224.0.0.5，目的MAC地址为01:00:5E:00:00:05。
 - 用 `iface_send_packet` 函数发包。

接收mOSPF Hello消息

- 收到mOSPF Hello消息后，进行解析：
 - 如果发送该消息的节点不在邻居列表中，添加至邻居列表。
 - 如果已存在，更新其达到时间。
- 更新邻居表项时需要加锁。
- 添加新邻居表项后，由于邻居列表结构发生变动，所以需要发送mOSPF LSU消息。

发送mOSPF LSU消息

- 当结点的邻居列表发生变动（添加老化删除）时，或超过lsu interval (30秒)未发送过链路状态信息时，向每个邻居节点发送链路状态信息。
 - 计算各个接口的邻居数量之和，申请长度为邻居数量之和的LSA表项，填充各表项。
 - 此过程需要加锁。
 - 当端口没有相邻路由器（例如r1-eth0, r4-eth2）时，仍然建立一个LSA条目，令邻居节点ID为0，IP为0.0.0.0。
 - 每个接口生成mOSPF LSU包，发送给该节点的所有邻居节点：
 - 每次生成链路状态信息时序列号(sequence number)加1
 - 目的IP地址为邻居节点相应端口的IP地址，目的MAC地址为该端口的MAC地址

接收mOSPF LSU消息

- 如果之前未收到结点rid的链路状态信息，则把rid对应的项插入到链路状态数据库中。
- 如果之前收到过结点rid的链路状态信息，且该信息的序列号更大，则把rid对应的项更新为LSU包中的条目。此过程需要加锁。
- TTL减1，如果TTL值大于0，则向除该端口以外的端口转发该消息。

实验内容二

计算最短路径

- 用Dijkstra算法计算源结点到其它结点的最短路径和相应前一跳节点
- 数据结构

```
#define MAX_NUM 255
int graph_matrix[MAX_NUM][MAX_NUM]; // adjacency matrix of graph
int visited[MAX_NUM]; // if node[i] is visited
int prev[MAX_NUM]; // last hop in route from node[0] to node[i]
int hops[MAX_NUM]; // shortest hops from node[0] to node[i]
u32 id_order[MAX_NUM]; // id of each node
int noden; // number of nodes
```

生成路由表

- 由于未设置路由表老化操作，先清空已有的路由表；
- 按照路径长度从小到大依次遍历每个结点，利用dijkstra算法找打最近的结点
- 计算结点0到结点i的下一跳节点：

查找从源节点到该节点的下一跳节点：递归寻找prev[i]，直到找到第一跳结点j；遍历邻居表项，找到结点j的网关，对结点i对应的每个网络，填充路由表。

实验结果

实验内容一

- 运行实验
 - 运行网络拓扑(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
- 实验结果


```
"Node: h1"
root@zy-VB:11-mospf# ping 10.0.6.22
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=0.282 ms
64 bytes from 10.0.6.22: icmp_seq=1 ttl=61 time=1.60 ms (DUP!)
64 bytes from 10.0.6.22: icmp_seq=2 ttl=61 time=0.150 ms
64 bytes from 10.0.6.22: icmp_seq=2 ttl=61 time=0.176 ms (DUP!)
64 bytes from 10.0.6.22: icmp_seq=3 ttl=61 time=0.167 ms
64 bytes from 10.0.6.22: icmp_seq=3 ttl=61 time=0.198 ms (DUP!)
^C
--- 10.0.6.22 ping statistics ---
3 packets transmitted, 3 received, +3 duplicates, 0% packet loss, time 2029ms
rtt min/avg/max/mdev = 0.150/0.429/1.603/0.526 ms
root@zy-VB:11-mospf# traceroute 10.0.6.22
traceroute to 10.0.6.22 (10.0.6.22), 64 hops max
 1  10.0.1.1  0.037ms  0.098ms  0.058ms
 2  10.0.3.3  0.101ms  0.116ms  0.074ms
 3  10.0.5.4  0.109ms  0.094ms  0.104ms
 4  10.0.6.22  0.126ms  0.101ms  0.098ms
```

可见h1和h2可以正常连通，路径为h1->r1->r3->r4->h2。

- 运行 `mininet > link r3 r4 down`，关掉r3到r4的链路，等一段时间后，再次用h1去traceroute节点h2

```
"Node: h1"
root@zy-VB:11-mospf# traceroute 10.0.6.22
traceroute to 10.0.6.22 (10.0.6.22), 64 hops max
 1  10.0.1.1  0.037ms  0.098ms  0.058ms
 2  10.0.3.3  0.101ms  0.116ms  0.074ms
 3  10.0.5.4  0.109ms  0.094ms  0.104ms
 4  10.0.6.22  0.126ms  0.101ms  0.098ms
root@zy-VB:11-mospf# ping 10.0.6.22
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=1.26 ms
64 bytes from 10.0.6.22: icmp_seq=1 ttl=61 time=1.46 ms (DUP!)
64 bytes from 10.0.6.22: icmp_seq=2 ttl=61 time=0.138 ms
64 bytes from 10.0.6.22: icmp_seq=2 ttl=61 time=0.424 ms (DUP!)
^C
--- 10.0.6.22 ping statistics ---
2 packets transmitted, 2 received, +2 duplicates, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.138/0.823/1.465/0.555 ms
root@zy-VB:11-mospf# traceroute 10.0.6.22
traceroute to 10.0.6.22 (10.0.6.22), 64 hops max
 1  10.0.1.1  0.037ms  0.121ms  0.061ms
 2  10.0.2.2  0.182ms  0.076ms  0.071ms
 3  10.0.4.4  0.208ms  1.004ms  1.301ms
 4  10.0.6.22  3.925ms  0.982ms  1.922ms
```

可见h1和h2可以仍然正常连通，路径为h1->r1->r2->r4->h2。