计算机网络研讨课实验报告

冯吕 2015K8009929049

2018 年 6 月 1 日

实验题目

网络路由实验二

实验内容

在本次实验中，需要基于上一次实验生成一致性链路数据库基础上，计算路由器表项。
计算路由器表项时，使用 *Dijkstra* 算法。

实验流程

在计算路由器表项时，分如下三步进行：

- 将上一实验得到的一致状态链路数据库抽象成拓扑图；
- 利用生成的拓扑图通过 *Dijkstra* 算法计算最短路径；
- 根据最短路径生成路由表项；

构建拓扑图

构建网络拓扑图时，使用一个二维数组存储边，使用一个一维数组存储节点。
构建网络拓扑图的函数为：*caculate_graph*

```
1  static int caculate_graph(){
2          int node_num =  0;
3          int x, y;
4          mospf_db_entry_t *db_pos, *db_q;
5          init_Graph();
6          Node[node_num++] = instance->router_id;
7          if (list_empty(&mospf_db)){
8                  printf("Emmty Database.\n");
9                  return node_num;
10         }
11         list_for_each_entry_safe(db_pos, db_q, &mospf_db, list){
12                 Node[node_num++] = db_pos->rid;
13         }
14         list_for_each_entry_safe(db_pos, db_q, &mospf_db, list){
```

```
15                  for(int i = 0; i != db_pos->nadv; ++i){
16                          x = get_Node(db_pos->array[i].rid, node_num);
17                          y = get_Node(db_pos->rid, node_num);
18                          Graph[x][y] = Graph[y][x] = 1;
19                  }
20          }
21          return node_num;
22  }
```

在构建拓扑图时，首先遍历数据库获取所有节点，然后遍历数据库的每一个 *entry* 中的每一个 *array*, 查找节点，构建边。

计算最短路径

构建好拓扑图之后，即可利用 *Dijkstra* 算法计算最短路径。

计算最短路径的函数为 *Dijkstra*:

```
1   static void Dijkstra(int Node_num){
2           for (int i = 0; i < Node_num; ++i){
3                   Visited[i] = 0;
4                   Dist[i] = Graph[0][i];
5                   if (Dist[i] == MAX_INT || !Dist[i])
6                     Prev[i] = -1;
7                   else
8                     Prev[i] = 0;
9           }
10
11          Dist[0] = 0;
12          Visited[0] = 1;
13          int u = 0;
14
15          for (int i = 0; i != Node_num -1; ++i){
16                  u = Min_dist(Node_num);
17                  Visited[u] = 1;
18                  for(int v = 0; v != Node_num; ++v){
19                          if (Visited[v] == 0 &&
20                            Graph[u][v] > 0 &&
21                            Dist[u] != MAX_INT &&
22                            Dist[u] + Graph[u][v] < Dist[v]){
23                                  Dist[v] = Dist[u] + Graph[u][v];
24                                  Prev[v] = u;
25                          }
26                  }
27          }
28  }
```

生成路由器表项

之后，利用计算好的最短路径来生成路由器表项目。

生成路由器表项的线程为 *caculate_rtable_thread*, 其为整个生成路由器表项的线程，其中调用了上面的两个函数。

```
void *caculate_rtable_thread(void *param){
        while(1){
                sleep(10);
                pthread_mutex_lock(&mospf_lock);
                int Node_num = caculate_graph();
                Dijkstra(Node_num);
                u32 Dist_rank[Node_num];
                for(int i = 0; i != Node_num; ++i)
                  Dist_rank[i] = i;
                for (int i = 0; i != Node_num -1; ++i)
                  for (int j = 0; i != Node_num - i -1; ++j){
                          if (Dist[j] > Dist[j+1]){
                                  swap(&Dist_rank[j], &Dist_rank[j+1]);
                                  swap(&Dist[j] , &Dist[j+1]);
                          }
                  }
                mospf_db_entry_t *db_pos, *db_q;
                u32 gw, dest;
                int hop = -1;
                iface_info_t **iface_addr = NULL;
                for (int i = 0; i != Node_num; ++i){
                if(Prev[Dist_rank[i]] != -1 && !list_empty(&mospf_db)){
                    list_for_each_entry_safe(db_pos, db_q, &mospf_db, list){
                for (int j = 0; j != db_pos->nadv; ++j){
                if (!is_in_rtable(db_pos->array[j].subnet)){
                          dest = db_pos->array[j].subnet;
                          hop = get_Node(db_pos->rid , Node_num);
                          while (Prev[hop])
                            hop = Prev[hop];
                          get_iface_and_gw(Node[hop], iface_addr , &gw);
                          add_rt_entry(new_rt_entry(dest ,
                          (*iface_addr)->mask,gw, *iface_addr));
                                                  }
                                          }
                                  }
                          }
                }
                printf("====RTable====\n");
                print_rtable();
                pthread_mutex_unlock(&mospf_lock);
```

```
41          }
42  }
```

根据最短路径生成路由器表项时，按照路径长度从小到大依次遍历每个节点，对于节点端口对应的每个网络，如果该网络对应的路由未被计算过，查找从源节点到该节点的下一跳节点，确定下一跳网关地址、源节点的转发端口。

数据库老化

此外，还有一个线程进行数据库的老化，在 *mospf_db_entry_t* 结构中增加一项 *add_time* 记录表项的添加时间，如果添加时间超过 35，则将其从数据库中清除：

```
1  void *checking_database_thread(void *param){
2          time_t now;
3          rt_entry_t *rt_pos, *rt_q;
4          mospf_db_entry_t *db_pos, *db_q;
5          while(1){
6                  sleep(1);
7                  pthread_mutex_lock(&mospf_lock);
8                  if (!list_empty(&mospf_db)){
9                          now = time(NULL);
10                         list_for_each_entry_safe(db_pos, db_q,
11                         &mospf_db, list){
12                                 if (now - db_pos->add_time >= 35){
13                                         list_for_each_entry_safe(rt_pos,
14                                         rt_q, &rtable, list){
15                                                 if (rt_pos->gw != 0)
16                                                   remove_rt_entry(rt_pos);
17                                         }
18                                         list_delete_entry(&db_pos->list);
19                                         free(db_pos);
20                                 }
21                         }
22                 }
23                 pthread_mutex_unlock(&mospf_lock);
24         }
25         return NULL;
26 }
```

实验结果

运行实验，在路由器节点运行 *mospfd*，一段时间后，能够生成路由器表项，在 $h1$ 上进行 *traceroute* 能够到达 $h2$：

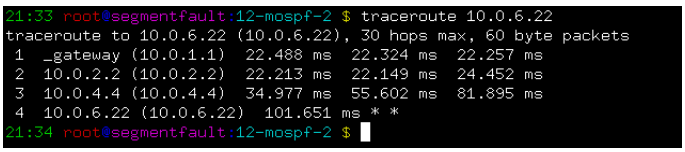

图 1: 运行截图

结果分析

实验结果正确。