

计算机网络实验报告

05-switching

雷正宇 2016K8009909005

实验内容

实现对数据结构mac_port_map的所有操作，以及数据包的转发和广播操作

```
1  iface_info_t *lookup_port(u8 mac[ETH_ALEN]);
2  void insert_mac_port(u8 mac[ETH_ALEN], iface_info_t *iface);
3  int sweep_aged_mac_port_entry();
4  void broadcast_packet(iface_info_t *iface, const char *packet, int len);
5  void handle_packet(iface_info_t *iface, char *packet, int len);
```

使用iperf和给定的拓扑进行实验，对比交换机转发与集线器广播的性能

实验流程

实现查找函数

```
1  iface_info_t *lookup_port(u8 mac[ETH_ALEN])
2  {
3      int len = sizeof(u8) * ETH_ALEN;
4      iface_info_t *ans = NULL;
5      pthread_mutex_lock(&mac_port_map.lock);
6      mac_port_entry_t *mac_pos = NULL;
7      list_for_each_entry(mac_pos, &mac_port_map.hash_table[hash8((char *)mac,
8      len)], list) {
9          if (memcmp(mac, mac_pos->mac, len) == 0) {
10             mac_pos->visited = time(0);
11             ans = mac_pos->iface;
12             break;
13         }
14     }
15     pthread_mutex_unlock(&mac_port_map.lock);
16     return ans;
17 }
```

这个函数要实现的功能是在 mac_port_map.hash_table 中通过 mac 地址查询对应端口。按照上述实现，最终该函数的功能是：如果在哈希中找到了对应的项，就将 visited 更新为当前时间，并把结果返回；否则返回空指针。

实现插入函数

```
1 void insert_mac_port(u8 mac[ETH_ALEN], iface_info_t *iface)
2 {
3     mac_port_entry_t *new_port = malloc(sizeof(mac_port_entry_t));
4     new_port->iface = iface;
5     memcpy(new_port->mac, mac, sizeof(u8) * ETH_ALEN);
6     new_port->visited = time(0);
7
8     pthread_mutex_lock(&mac_port_map.lock);
9     list_add_tail(&new_port->list, &mac_port_map.hash_table[hash8((char
10 *)mac, sizeof(u8) * ETH_ALEN)]);
11     pthread_mutex_unlock(&mac_port_map.lock);
12 }
```

该函数主要是在交换机学习时使用，功能是将一个 <mac, 端口> 键值对插入 mac_port_map.hash_table 中。由于 mac_port_map.hash_table 的每一项都是使用的 Linux 风格链表，所以直接使用 list_add_tail 函数就可以简单地完成插入功能。

实现老化函数

```
1 int sweep_aged_mac_port_entry()
2 {
3     mac_port_entry_t *pos, *q;
4     time_t now = time(NULL);
5     int count = 0;
6
7     pthread_mutex_lock(&mac_port_map.lock);
8     for (int i = 0; i < HASH_8BITS; i++) {
9         list_for_each_entry_safe(pos, q, &mac_port_map.hash_table[i], list) {
10             if (now - pos->visited >= MAC_PORT_TIMEOUT) {
11                 list_delete_entry(&pos->list);
12                 free(pos);
13                 count++;
14             }
15         }
16     }
17     pthread_mutex_unlock(&mac_port_map.lock);
18
19     return count;
20 }
```

这个函数每次被调用后，会检查 mac_port_map.hash_table 中的所有键值对，如果发现其中有超过 30 秒没有被访问到的项（visited 距离现在时间超过 30）就会将它从链表中删除。最后返回被删除节点的数量。

实现广播函数

```
1 void broadcast_packet(iface_info_t *iface, char *packet, int len)
2 {
3     iface_info_t *pos = NULL;
4     list_for_each_entry(pos, &instance->iface_list, list) {
5         if (pos != iface)
6             iface_send_packet(pos, packet, len);
7     }
8 }
```

广播函数直接用上周实验的代码即可。

实现包处理逻辑

```
1 void handle_packet(iface_info_t *iface, char *packet, int len)
2 {
3     struct ether_header *eh = (struct ether_header *)packet;
4
5     if (lookup_port(eh->ether_shost) == NULL) {
6         insert_mac_port(eh->ether_shost, iface);
7     }
8
9     iface_info_t *find_result = lookup_port(eh->ether_dhost);
10    if (find_result != NULL) {
11        iface_send_packet(find_result, packet, len);
12    } else {
13        broadcast_packet(iface, packet, len);
14    }
15 }
```

主要逻辑为：当收到一个包时，在哈希中对其源地址进行检索。检索过程中，如果找到了对应的项，就会将其更新；否则就进行学习——把源地址和端口的键值对插入哈希中。

然后在哈希中对目的地址进行检索，如果找到了对应的端口，就向这个端口发送包，检索过程中会对这一项进行更新；否则就广播这个包。

修改启动脚本

启动脚本仅需要把原脚本中的

```
1 s1.cmd('./switch-reference &')
```

替换为自己制作的 switch 即可：

```
1 s1.cmd('./switch &')
```

实验结果和分析

以下是本次试验中使用 switch 得到的 iperf 测试结果：

```
root@ubuntu:/mnt/hgfs/[...]/[...]/[...]/05-switching# iperf -c 10.0.0.2 -t 30
-----
Client connecting to 10.0.0.2, TCP port 5001
TCP window size: 85.3 KByte (default)
-----
[ 13] local 10.0.0.1 port 40352 connected with 10.0.0.2 port 5001
[ ID] Interval      Transfer    Bandwidth
[ 13]  0.0-30.0 sec  34.5 MBytes  9.65 Mbits/sec
```

对比上周使用 hub 广播的测试结果：

```
"Node: h2"
root@ubuntu:/mnt/hgfs/[...]/[...]/[...]/04-broadcast# iperf -c 10.0.0.1 -t 30
-----
Client connecting to 10.0.0.1, TCP port 5001
TCP window size: 85.3 KByte (default)
-----
[ 13] local 10.0.0.2 port 58762 connected with 10.0.0.1 port 5001
[ ID] Interval      Transfer    Bandwidth
[ 13]  0.0-30.1 sec  33.9 MBytes  9.43 Mbits/sec
root@ubuntu:/mnt/hgfs/[...]/[...]/[...]/04-broadcast#
```

可以看出，在带宽都为 10 Mbps 的情况下，switch 的效率略高于 hub.