计算机网络研讨课实验报告

冯吕 2015K8009929049

2018 年 4 月 10 日

实验题目

交换机学习实验

实验内容

在本次实验中，需要实现交换机的转发。在广播网络中, 转发节点将每个数据包从所有其他端口广播出去，而在交换网络中，交换机将收到的数据包沿着目的主机方向转发。

实现交换机的关键思想是:

> 当交换机从某端口收到源 mac 地址为 X 的数据包时，那么，将数据包从该端口转出一定可以到达目的 mac 地址为 X 的目的主机。

交换机需要实现如下操作:

- 查询操作: 每收到一个数据包, 根据目的 MAC 地址查询相应转发条目: 如果查询到对应条目, 则根据相应转发端口转发数据包, 并更新访问时间; 否则, 广播该数据包。
- 插入操作: 每收到一个数据包, 如果其源 MAC 地址不在转发表中, 则将该地址与入端口的映射关系写入转发表。
- 老化操作: 每秒钟运行一次老化操作, 删除超过 30 秒未访问的转发条目。

另外，转发表的老化操作与其他操作独立运行：通过多线程和互斥操作实现。

实验具体需要实现的内容是：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，对应的函数如下:

```
1 iface_info_t *lookup_port(u8 mac[ETH_ALEN]);
2
3 void insert_mac_port(u8 mac[ETH_ALEN], iface_info_t *iface);
4
5 int sweep_aged_mac_port_entry();
6
7 void broadcast_packet(iface_info_t *iface, const char *packet, int len);
8
9 void handle_packet(iface_info_t *iface, char *packet, int len);
```

以及使用 $iperf$ 和给定的拓扑进行实验, 对比交换机转发与集线器广播 (Lab 4) 的性能。

实验流程

由实验内容知，该实现五个函数，下面一次讲解每个函数的实现。

函数 *lookup_port*

函数实现代码如下：

```
iface_info_t *lookup_port(u8 mac[ETH_ALEN], int search_des)
{
        pthread_mutex_lock(&mac_port_map.lock);
        // count key value
        u8 key = hash8(mac, ETH_ALEN);
        //find port
        mac_port_entry_t *entry = mac_port_map.hash_table[key];
        if (entry){
                //update visited time
                if (search_des){
                        entry->visited = time(NULL);
                }
                pthread_mutex_unlock(&mac_port_map.lock);
                return entry->iface;
        }
        pthread_mutex_unlock(&mac_port_map.lock);
        return NULL;
}
```

该函数通过 *mac* 地址来查询 *hash* 表，查找是否存在对应的转发条目。首先，计算对应的 *hash key*，然后查看对应转发条目。由于我在实现 *hand_package* 时需要进行两次查询，一次查询目的 *mac* 地址对应的转发条目，此时，如果存在，需要更新访问时间，一次查询源 *mac* 地址对应的转发条目，此时，不用更新访问时间，因此，我对该函数多加了一个参数以标志查询目的。

函数 *insert_mac_port*

函数实现代码如下：

```
void insert_mac_port(u8 mac[ETH_ALEN], iface_info_t *iface)
{
        u8 key = hash8(mac, ETH_ALEN);
        //find port
        pthread_mutex_lock(&mac_port_map.lock);
        mac_port_entry_t **entry = &mac_port_map.hash_table[key];
        if (!*entry){
                //malloc new entry
                *entry = (mac_port_entry_t*)malloc(sizeof(mac_port_entry_t));
                (*entry)->next = NULL;
                memcpy((*entry)->mac, mac, ETH_ALEN);
                (*entry)->iface = iface;
```

```
13          }
14          (*entry)->visited = time (NULL);
15          mac_port_entry_t *tmp = (*entry)->next;
16          mac_port_entry_t *pre = NULL;
17          // insert entry into link list
18          while (tmp){
19                  pre = tmp;
20                  tmp = tmp->next;
21          }
22          if ( pre ){
23                  pre->next = *entry;
24                  (*entry)->next = NULL;
25          }
26          pthread_mutex_unlock(&mac_port_map.lock);
27  }
```

当包的源 *mac* 地址不在转发表中，需要进行插入操作，插入时，需要先根据 *mac* 地址计算对应的 *hash key*，然后将转发条目存到对应的位置。

函数 *sweep_aged_mac_port_entry*

函数实现代码如下：

```
1   int sweep_aged_mac_port_entry()
2   {
3           pthread_mutex_lock(&mac_port_map.lock);
4           int n = 0;
5           mac_port_entry_t *entry = NULL;
6           time_t now = time(NULL);
7           // scan all port
8           for (int i = 0; i < HASH_8BITS; i++){
9                   entry = mac_port_map.hash_table[i];
10                  if ( !entry ){
11                          continue;
12                  }
13                  // judge whether TIMEOUT or not
14                  if ( entry->visited - now > MAC_PORT_TIMEOUT ){
15                          /*printf ("in if \n");*/
16                          n++;
17                          mac_port_entry_t *tmp = entry->next;
18                          // free all entry
19                          while (tmp){
20                                  entry->next = tmp->next;
21                                  free(tmp);
22                                  tmp = entry->next;
23                          }
```

```
24                              free(entry);
25                      }
26              }
27              pthread_mutex_unlock(&mac_port_map.lock);
28              return n;
29      }
```

该函数是用来删除老化的转发条目，因此，顺序浏览转发表，判断是否超时，即上次访问时间减去当前时间是否大于 30s，如果超时，则将该转发条目从表中删除，并返回总共删除的转发条目数。

函数 *broadcast_package*

函数实现代码如下：

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

广播函数则将包从所有其他端口转发出去：依次列出所有端口，然后判断是否为其他端口，如果是，则将包转发出去。

函数 *handle_package*

函数实现代码如下：

```
1  void handle_packet(iface_info_t *iface, char *packet, int len)
2  {
3          struct ether_header *eh = (struct ether_header *)packet;
4
5          iface_info_t *iface_t = lookup_port(eh->ether_dhost, 1);
6          if (iface_t){
7                  iface_send_packet(iface_t, packet, len);
8          }
9          else {
10                 broadcast_packet(iface, packet, len);
11         }
12         iface_t = lookup_port(eh->ether_shost, 0);
13         if (!iface_t){
14                 insert_mac_port(eh->ether_shost, iface);
15         }
16 }
```

该函数是转发机的包处理函数，当转发机收到一个包时，首先查询目的 *mac* 地址是否在转发表中，如果在，则从对应端口将包转发出去，如果不在，则进行广播；另外，还需要进行一次查询操作，查询源 *mac*

地址是否在转发表中，如果不在，则需要插入转发表，两次查询不同的是，前者如果查询到，则需要更新访问时间，而后者则不需要。

使用 $iperf$ 和给定拓扑结构进行新能测量

为了方便和广播性能进行对比，采用同样的测量方式：$h1$ 作为 $client$，$h2, h3$ 作为 $servers$ 以及 $h1$ 作为 $server$，$h2, h3$ 作为 $clients$。

实验结果

在拓扑结构中，$h1$ 和 $s1$ 之间的最大带宽为 $20MB/s$，$h2$ 和 $s1$ 以及 $h3$ 和 $s1$ 之间的最大带宽均为 $10MB/s$，测量结果如下：

- $h1\ server, h2/h3 clients$：同时向 $h1$ 发送包，测得 $h1$ 的带宽为 $19.12MB/s$，链路利用率为 95.6%，$h2, h3$ 带宽均为 $9.57MB/s$，利用率为 95.7%；
- $h1\ client, h2/h3 severs$：$h1$ 带宽分别为 $9.63MB/s$ 和 $9.64MB/s$，$h2, h3$ 带宽均为 $9.56MB/s$。此时，$h1$ 带宽之所以如此低，是因为受 $h2, h3$ 和 $s1$ 之间的链路的最大带宽限制。

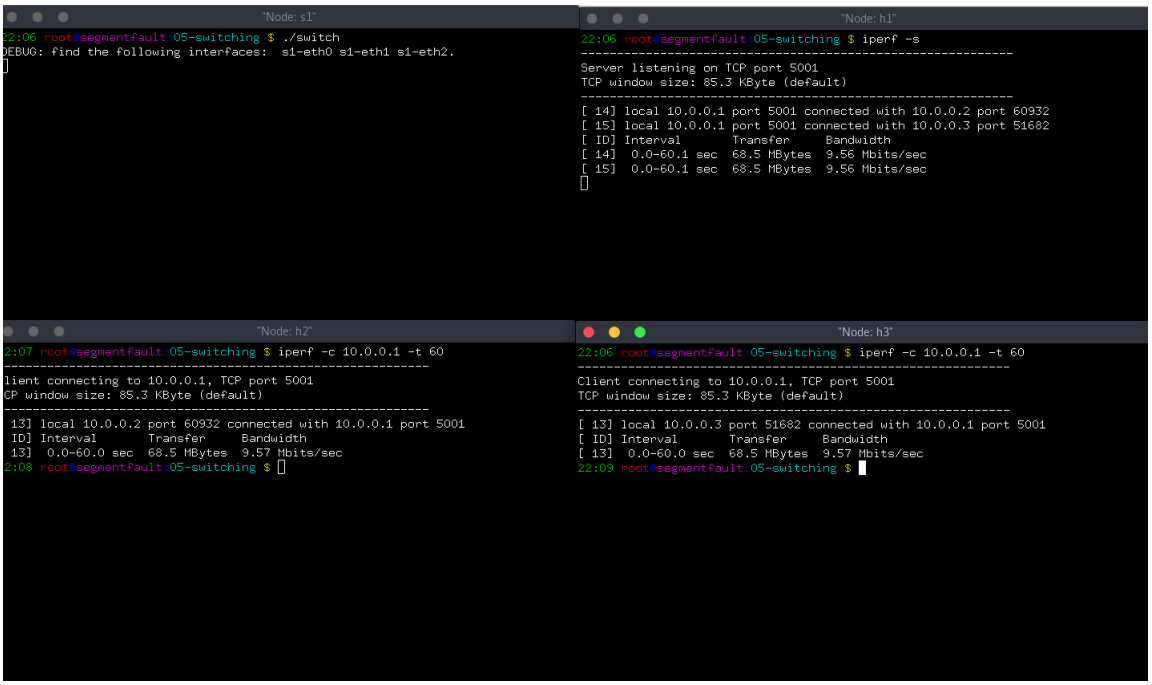

图 1: 使用 $iperf$ 进行性能测量

和广播性能对比：

- $h1\ server, h2/h3 clients$：同时向 $h1$ 发送包，测得 $h1$ 的带宽为 $17.96MB/s$，链路利用率为 89.8%，$h2$ 带宽均为 $9.16MB/s$，利用率为 91.6%，$h3$ 带宽为 $8.98MB/s$，链路利用率为 89.8%；
- $h1\ client, h2/h3 severs$：$h1$ 带宽分别为 $9.41MB/s$ 和 $9.43MB/s$，$h2, h3$ 带宽均为 $9.37MB/s$。

通过对比可知，转发机的链路利用率比广播网络要高。

结果分析

在交换机中，每一个包都有目的地址，因此，可以实现多个数据包的同时传输，而在广播网络中，同一时刻只能有一个数据包进行传输，还可能发生碰撞，因此，交换机的性能要比广播网络的性能好。