网络路由实验

孙佳钰 2015K8009929051

2018 年 11 月 29 日

1 实验内容

OSPF 主要工作为使每个节点都向外通告自己的链路状态信息，从而使网络中每个路由器都有完整的网络拓扑，然后自行计算生成到每个节点的最短路径。本次实验需要实现简化版的 OSPF 协议，内容有：

- 发送及处理向相邻节点广播自己的 hello 消息。
- 对发送过 hello 消息的相邻节点信息进行老化操作。
- 发送及处理向所有节点广播自己已知的相邻节点的链路状态的 lsu 消息。

2 实验流程

由于代码太多，故报告中没有加入完整代码，完整代码可见附件。

2.1 hello 消息的发送与处理

1. 遍历端口，向每个端口都发送组好的 hello 消息包。

```c
char *packet = malloc(size * sizeof(char));
memset(packet, 0, size);
struct ether_header *ehdr = (struct ether_header*)packet;
ehdr->ether_dhost[5] = 0x05;
ehdr->ether_dhost[2] = 0x5e;
ehdr->ether_dhost[0] = 0x01;
memcpy(ehdr->ether_shost, iface->mac, ETH_ALEN);
ehdr->ether_type = htons(ETH_P_IP);
struct iphdr *ihdr = packet_to_ip_hdr(packet);
ip_init_hdr(ihdr, iface->ip, MOSPF_ALLSPFRouters, size - ETHER_HDR_SIZE, \
     IPPROTO_MOSPF);
```

```
struct mospf_hdr *mhdr = (struct mospf_hdr *)((char *)ihdr + \
    IP_BASE_HDR_SIZE);
struct mospf_hello *mhhdr = (struct mospf_hello*)((char*)mhdr + \
    MOSPF_HDR_SIZE);
mospf_init_hello(mhhdr, iface->mask);
mospf_init_hdr(mhdr, MOSPF_TYPE_HELLO, MOSPF_HDR_SIZE + MOSPF_HELLO_SIZE, \
    instance->router_id, instance->area_id);
mhdr->checksum = mospf_checksum(mhdr);
iface_send_packet(iface, packet, size);
```

2. 在收到数据包的端口的邻居列表中查找该包的路由 id，找到则更新存活时间；未找到则增加该相邻节点。

```
list_for_each_entry(nbr, &iface->nbr_list, list) {
    if (nbr->nbr_id == ntohl(mhdr->rid)) {
        find = 1;
        nbr->alive = 0;
        break;
    }
}
if (0 == find) {
    fprintf(stdout, "DEBUG: receive new hello packet.\n");
    struct mospf_hello *mhhdr = (struct mospf_hello*)((char*)mhdr + \
        MOSPF_HDR_SIZE);
    mospf_nbr_t *new = (mospf_nbr_t*)malloc(sizeof(mospf_nbr_t));
    new->nbr_id = ntohl(mhdr->rid);
    new->nbr_ip = ntohl(ihdr->saddr);
    new->nbr_mask = ntohl(mhhdr->mask);
    new->alive = 0;
    list_add_tail(&new->list, &iface->nbr_list);
    iface->num_nbr++;
}
```

2.2 相邻节点信息老化

对每个端口的邻居列表都进行遍历，将其存活时间 +1，如果超过了 $3 * helloint$ 则将该邻居的信息删除。

```
list_for_each_entry(iface, &instance->iface_list, list) {
    mospf_nbr_t *nbr = NULL, *q;
    list_for_each_entry_safe(nbr, q, &iface->nbr_list, list) {
        nbr->alive ++;
        if (nbr->alive > 3 * iface->helloint) {
            list_delete_entry(&nbr->list);
            free(nbr);
        }
    }
}
```

2.3 lsu 消息的发送与处理

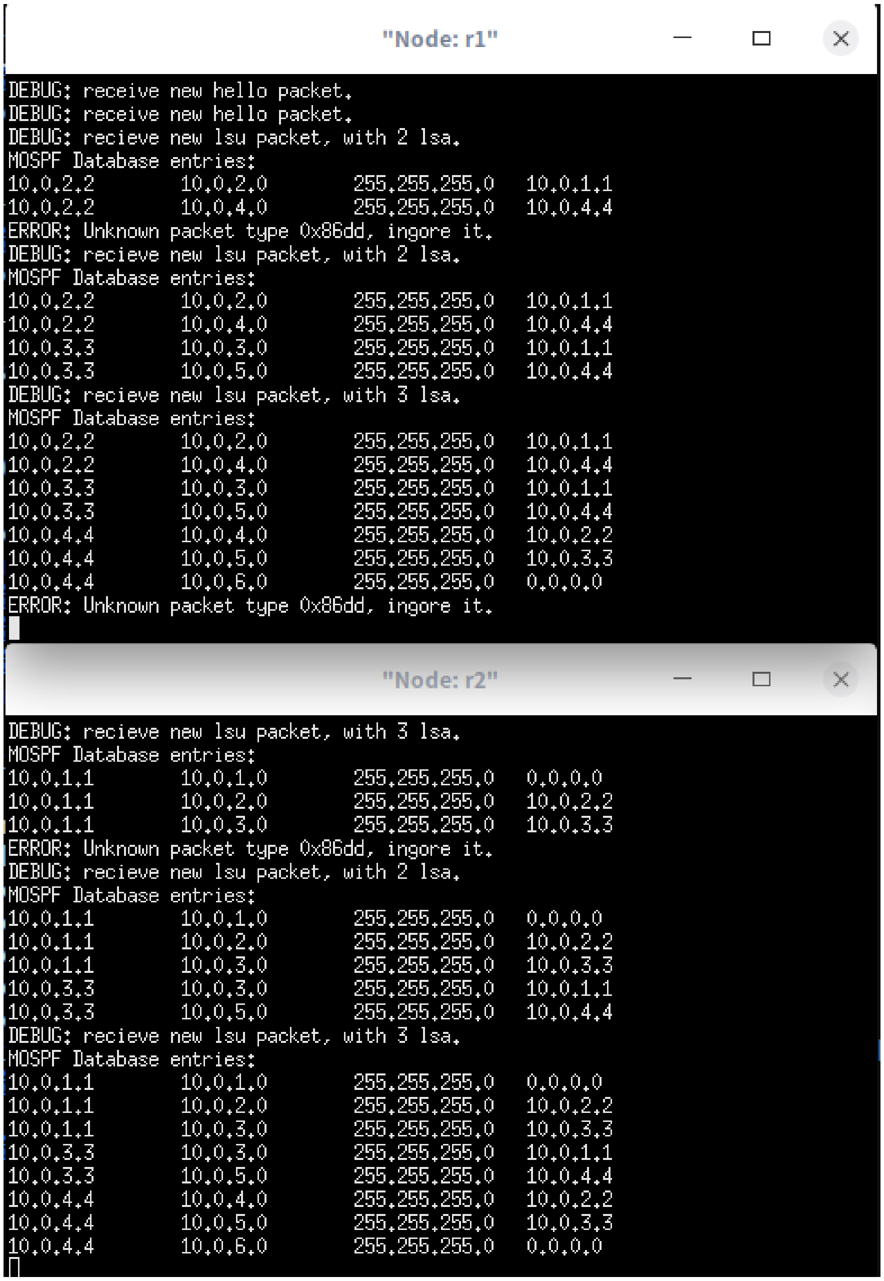
1. 首先生成 lsa 消息数组：遍历端口，若此端口没有邻居，则将该端口的信息作为一个 lsa；若此端口有邻居，则将所有邻居依次加入 lsa 数组。然后再次遍历端口，对每个端口都发送生成的数据包。

```
list_for_each_entry(iface, &instance->iface_list, list) {
    if (0 == iface->num_nbr) {
        lsarr[i].subnet = htonl(iface->ip & iface->mask);
        lsarr[i].mask = htonl(iface->mask);
        lsarr[i].rid = htonl(0);
        i++;
    } else {
        mospf_nbr_t *nbr = NULL;
            list_for_each_entry(nbr, &iface->nbr_list, list) {
            lsarr[i].subnet = htonl(nbr->nbr_ip & nbr->nbr_mask);
            lsarr[i].mask = htonl(nbr->nbr_mask);
            lsarr[i].rid = htonl(nbr->nbr_id);
            i++;
        }
    }
}
list_for_each_entry(iface, &instance->iface_list, list) {
    if (iface->num_nbr) {
        mospf_nbr_t *nbr = NULL;
        list_for_each_entry(nbr, &iface->nbr_list, list) {
            int size = _size + num * MOSPF_LSA_SIZE;
            char *packet = (char*)malloc(size * sizeof(char));
            memset(packet, 0, size);
            ...
            ip_send_packet(packet, size);
        }
    }
}
```

2. 在链路状态数据库中查询收到的数据包的发送者 id，若查询到则比较序列号大小，若收到的数据包序列号大则根据收到的包更新数据库信息；若未查询到则增加一条数据库记录，记录该包信息。

```
list_for_each_entry(db_entry, &mospf_db, list) {
    if (db_entry->rid == ntohl(mhdr->rid)) {
        find = 1;
        if (db_entry->seq < ntohs(lsuhdr->seq)) {
            ……
        }
    }
}
if (0 == find) {
    fprintf(stdout, "DEBUG: recieve new lsu packet, with %d lsa.\n", num);
    mospf_db_entry_t *new = (mospf_db_entry_t*)malloc(sizeof \
        (mospf_db_entry_t));
    ……
}
list_add_tail(&new->list, &mospf_db);
if (--lsuhdr->ttl > 0) {
    iface_info_t *iface_t = NULL;
    list_for_each_entry(iface_t, &instance->iface_list, list) {
        if (iface_t->num_nbr && (iface->index != iface_t->index)) {
            char *_packet = (char*)malloc(len);
            memcpy(_packet, packet, len);
            ……
            mospf_nbr_t *nbr = NULL;
            list_for_each_entry(nbr, &iface_t->nbr_list, list) {
                if (nbr->nbr_id == ntohl(mhdr->rid)) continue;
                ……
                ip_send_packet(_packet, len);
            }
        }
    }
}
```

3 实验结果及分析



```
DEBUG: receive new hello packet.
DEBUG: receive new hello packet.
DEBUG: recieve new lsu packet, with 2 lsa.
MOSPF Database entries:
10.0.2.2        10.0.2.0        255.255.255.0   10.0.1.1
10.0.2.2        10.0.4.0        255.255.255.0   10.0.4.4
ERROR: Unknown packet type 0x86dd, ingore it.
DEBUG: recieve new lsu packet, with 2 lsa.
MOSPF Database entries:
10.0.2.2        10.0.2.0        255.255.255.0   10.0.1.1
10.0.2.2        10.0.4.0        255.255.255.0   10.0.4.4
10.0.3.3        10.0.3.0        255.255.255.0   10.0.1.1
10.0.3.3        10.0.5.0        255.255.255.0   10.0.4.4
DEBUG: recieve new lsu packet, with 3 lsa.
MOSPF Database entries:
10.0.2.2        10.0.2.0        255.255.255.0   10.0.1.1
10.0.2.2        10.0.4.0        255.255.255.0   10.0.4.4
10.0.3.3        10.0.3.0        255.255.255.0   10.0.1.1
10.0.3.3        10.0.5.0        255.255.255.0   10.0.4.4
10.0.4.4        10.0.4.0        255.255.255.0   10.0.2.2
10.0.4.4        10.0.5.0        255.255.255.0   10.0.3.3
10.0.4.4        10.0.6.0        255.255.255.0   0.0.0.0
ERROR: Unknown packet type 0x86dd, ingore it.
```

```
DEBUG: recieve new lsu packet, with 3 lsa.
MOSPF Database entries:
10.0.1.1        10.0.1.0        255.255.255.0   0.0.0.0
10.0.1.1        10.0.2.0        255.255.255.0   10.0.2.2
10.0.1.1        10.0.3.0        255.255.255.0   10.0.3.3
ERROR: Unknown packet type 0x86dd, ingore it.
DEBUG: recieve new lsu packet, with 2 lsa.
MOSPF Database entries:
10.0.1.1        10.0.1.0        255.255.255.0   0.0.0.0
10.0.1.1        10.0.2.0        255.255.255.0   10.0.2.2
10.0.1.1        10.0.3.0        255.255.255.0   10.0.3.3
10.0.3.3        10.0.3.0        255.255.255.0   10.0.1.1
10.0.3.3        10.0.5.0        255.255.255.0   10.0.4.4
DEBUG: recieve new lsu packet, with 3 lsa.
MOSPF Database entries:
10.0.1.1        10.0.1.0        255.255.255.0   0.0.0.0
10.0.1.1        10.0.2.0        255.255.255.0   10.0.2.2
10.0.1.1        10.0.3.0        255.255.255.0   10.0.3.3
10.0.3.3        10.0.3.0        255.255.255.0   10.0.1.1
10.0.3.3        10.0.5.0        255.255.255.0   10.0.4.4
10.0.4.4        10.0.4.0        255.255.255.0   10.0.2.2
10.0.4.4        10.0.5.0        255.255.255.0   10.0.3.3
10.0.4.4        10.0.6.0        255.255.255.0   0.0.0.0
```

仅截取 r1 和 r2 的结果，可以看出是正确的。