Computer Networks Lab6

黄嘉祺 221220108

1 Program Structure and Design

1.1 Implementing the Router

- 实验要求我们实现一个 `Router` 类型来支持进行路由表的相关更新，以获得其它主机的地址
- 具体来说，需要在 `src/router.cc` 中实现补完 `add_route` 和 `route` 两个成员函数
  - `add_route` 用于在转发表中增加一个表项，更新它
  - `route` 用于将每个接口的输入数据包转发到正确的输出接口
- 在 `src/router.hh` 中补充定义了一个结构体 `RouteEntry` 来表示转发表中的表项，同时定义了成员变量 `std::vector<RouteEntry> route_table_` 用于保存转发表

```
struct RouteEntry
{
  uint32_t route_prefix;
  uint8_t prefix_length;
  std::optional<Address> next_hop;
  size_t interface_num;
};
```

- `add_route` 函数的实现思路很简单，只需要将新的表项插入到转发表中即可

```
void Router::add_route( const uint32_t route_prefix,
                        const uint8_t prefix_length,
                        const optional<Address> next_hop,
                        const size_t interface_num )
{
  cerr << "DEBUG: adding route " << Address::from_ipv4_numeric( route_prefix ).ip() << "/"
       << static_cast<int>( prefix_length ) << " => " << ( next_hop.has_value() ? next_hop->ip() : "(direct)" )
       << " on interface " << interface_num << "\n";

  // Your code here.
  route_table_.push_back( { route_prefix, prefix_length, next_hop, interface_num } );
}
```

- `route` 函数的实现思路如下
  - 首先，遍历所有的网络接口 `_interfaces` 。对于每个接口，该函数会检查该接口接收到的数据报文队列 `datagrams_received()` 是否为空。如果不为空，它会从队列中取出一个数据报文 `dgram` 并将其从队列中移除。
  - 然后，获取数据报文的目标地址 `dgram.header.dst` 。如果数据报文的 `dgram.header.ttl` （生存时间）为 `0` ，它会输出一条错误信息并丢弃该数据报文。否则，它会将 `dgram.header.ttl` 减 `1` ，并重新计算数据报文的校验和。

- 接下来，它遍历路由表 `route_table_` 中的每一条路由。对于每一条路由，它会检查该路由的前缀长度是否小于当前的最大前缀长度 `max_prefix_length` ，如果是，则跳过该路由。然后，它会逐位比较目标地址 `dst` 和路由前缀 `route.route_prefix` ，如果在某一位上不匹配，则跳出循环。如果所有位都匹配，并且该路由的前缀长度大于或等于当前的最大前缀长度，则更新最大前缀长度 `max_prefix_length` 、下一跳地址 `next_hop` 和接口编号 `interface_num` 。
- 最后，如果没有找到匹配的路由（即 `max_prefix_length` 仍为 `-1` ），它会输出一条错误信息并丢弃该数据报文。如果找到了匹配的路由，它会根据 `next_hop` 是否有值来决定将数据报文发送到下一跳地址或直接发送到目标地址。

```
28  void Router::route()
29  {
30    // Your code here.
31    for ( auto& interface : _interfaces ) {
32      while ( !interface->datagrams_received().empty() ) {
33        int8_t max_prefix_length = -1;
34        auto interface_num = -1;
35        optional<Address> next_hop;
36        auto dgram = interface->datagrams_received().front();
37        interface->datagrams_received().pop();
38        auto dst = dgram.header.dst;
39        if ( dgram.header.ttl == 0 ) {
40          std::cerr << "TTL is 0, dropping packet\n";
41          continue;
42        }
43        dgram.header.ttl--;
44        if ( dgram.header.ttl == 0 ) {
45          std::cerr << "TTL is 0, dropping packet\n";
46          continue;
47        }
48        dgram.header.compute_checksum();
```

```
49        for ( auto& route : route_table_ ) {
50          if ( route.prefix_length < max_prefix_length )
51            continue;
52          int i = -1;
53          for ( i = 0; i < route.prefix_length; i++ ) {
54            if ( ( dst & ( 1 << ( 31 - i ) ) ) != ( route.route_prefix & ( 1 << ( 31 - i ) ) ) )
55              break;
56          }
57          if ( i == route.prefix_length && route.prefix_length >= max_prefix_length ) {
58            max_prefix_length = route.prefix_length;
59            next_hop = route.next_hop;
60            interface_num = route.interface_num;
61          }
62        }
63        if ( max_prefix_length == -1 ) {
64          std::cerr << "No route found, dropping packet\n";
65          continue;
66        }
67        if ( next_hop.has_value() ) {
68          _interfaces[interface_num]->send_datagram( dgram, next_hop.value() );
69        } else {
70          _interfaces[interface_num]->send_datagram( dgram, Address::from_ipv4_numeric( dgram.header.dst ) );
71        }
72      }
73    }
74  }
```

2 Implementation Challanges

- 在实现 `route` 函数的时候，一开始在进行最长前缀匹配的时候，我设置成在循环体里进行更新了，导致了错误的结果。后来我发现了这个问题，将更新操作放到了循环体外，就解决了这个问题

3 Remaining Bugs

目前我的代码已经通过了全部的测试，暂时未发现明显的bug

4 Experimental Results and Performance

- net_interface and router: Passed all 3 tests