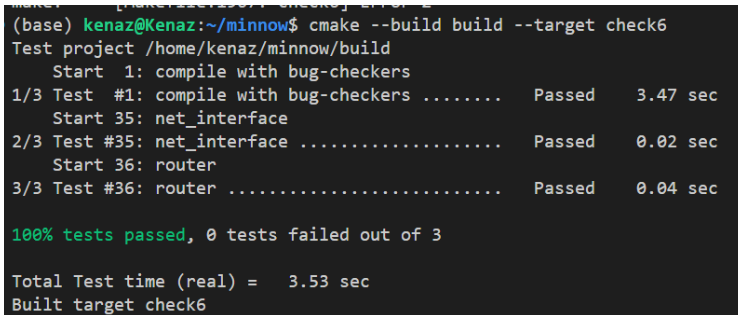
Lab6 实验报告

王卫东　221900332

2024 年 12 月 18 日

Here's the check6 test results.

一　Experimental Results



```
(base) kenaz@Kenaz:~/minnow$ cmake --build build --target check6
Test project /home/kenaz/minnow/build
    Start  1: compile with bug-checkers
1/3 Test  #1: compile with bug-checkers ........   Passed    3.47 sec
    Start 35: net_interface
2/3 Test #35: net_interface ...................   Passed    0.02 sec
    Start 36: router
3/3 Test #36: router ..........................   Passed    0.04 sec

100% tests passed, 0 tests failed out of 3

Total Test time (real) =   3.53 sec
Built target check6
```

图 1: passing lab6 tests

二　Implementation

Add a new class RouteEntry to store the routing table entry.In the Router class,add a vector<RouteEntry> routing_table_ to store the routing table entries.

```
class RouteEntry
{
public:
  const uint32_t route_prefix;
  const uint8_t prefix_length;
  const std::optional<Address> next_hop;
```

```
    const size_t interface_num;
  RouteEntry( uint32_t rp,
          uint8_t pl,
          std::optional<Address> nh,
          size_t i )
    : route_prefix( rp )
    , prefix_length( pl )
    , next_hop( nh )
    , interface_num( i )
  {}
};
```

Here is the Implementation of the route function.

```
// Go through all the interfaces, and route every incoming datagram to its proper
      outgoing interface.
void Router::route()
{
  // Your code here.
  for ( auto& interface : _interfaces ) {
    auto& queue = interface->datagrams_received();
    while ( !queue.empty() ) {
      auto dgram = queue.front();
      if ( dgram.header.ttl <= 1 )
        return;
      // Longest prefix match
      uint32_t dest_ip = dgram.header.dst;
      int match_id = -1;
      int match_len = -1;
      for ( size_t i = 0; i < _routes.size(); i++ ) {
        auto route = _routes[i];
        uint32_t route_prefix = route.route_prefix;
        uint8_t prefix_length = route.prefix_length;
        uint32_t mask = ( prefix_length == 0 ) ? 0 : numeric_limits<int>::min() >> (
              prefix_length - 1 );
        if ( ( dest_ip & mask ) == route_prefix ) {
          if ( prefix_length > match_len ) {
            match_len = prefix_length;
            match_id = i;
          }
        }
      }
      if ( match_id == -1 )
        return;
      dgram.header.ttl--;
      dgram.header.compute_checksum();
      auto next_hop = _routes[match_id].next_hop;
```

```
        auto interface_num = _routes[match_id].interface_num;
        if ( next_hop.has_value() ) {
          _interfaces[interface_num]->send_datagram( dgram, next_hop.value() );
        } else {
          _interfaces[interface_num]->send_datagram( dgram, Address::from_ipv4_numeric
               ( dest_ip ) );
        }
        queue.pop();
        // Your code here.
      }
    }
  }
```

三 Challenge

Keep aware that when we change the TTL of the datagram, we should recompute the checksum of the datagram header.