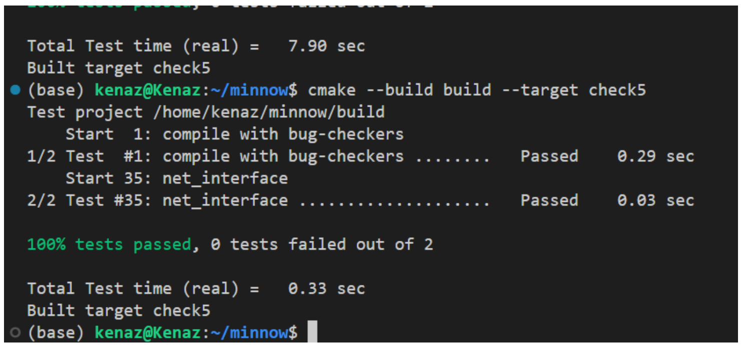
Lab5 实验报告

王卫东　221900332

2024 年 11 月 29 日

Here’s the check5 test results.

一　Experimental Results



```
Total Test time (real) =   7.90 sec
Built target check5
(base) kenaz@Kenaz:~/minnow$ cmake --build build --target check5
Test project /home/kenaz/minnow/build
    Start  1: compile with bug-checkers
1/2 Test  #1: compile with bug-checkers ........   Passed    0.29 sec
    Start 35: net_interface
2/2 Test #35: net_interface ...................   Passed    0.03 sec

100% tests passed, 0 tests failed out of 2

Total Test time (real) =   0.33 sec
Built target check5
(base) kenaz@Kenaz:~/minnow$
```

图 1: passing lab5 tests

二　Implementation

I used cur_time to record the current time, and used arp_life_ to record the lifetime of the sent arp message. When the lifetime of the arp table is exceeded, the arp table will be deleted. The wait_queue_ is used to store the frames that are waiting for the arp table to be updated. When the arp table is updated, the frames in the wait_queue_ will be sent out.

```
private:
  // Datagrams that have been received
  std::queue<InternetDatagram> datagrams_received_ {};
```

```
    // map:生命周期, multimap: 地址映射
    std::unordered_map<uint32_t, std::pair<uint32_t, EthernetAddress>> arp_table_
         {};
    std::unordered_map<uint32_t, std::pair<uint32_t, EthernetFrame>> arp_life_ {};
    std::multimap<uint32_t, EthernetFrame> wait_queue_ {};
    uint32_t cur_time { 0 };
```

Here is the Implementation of the send_message function.

```
  void NetworkInterface::send_datagram( const InternetDatagram& dgram, const
       Address& next_hop )
  {
    EthernetFrame ethernetframe;
    ethernetframe.header.src = ethernet_address_;
    if ( !arp_table_.contains( next_hop.ipv4_numeric() ) ) {
      if ( arp_life_.contains( next_hop.ipv4_numeric() ) ) {
      return; // 发送还在lifetime内的ARP请求
    }
    ethernetframe.header.type = EthernetHeader::TYPE_ARP;
    ethernetframe.header.dst = ETHERNET_BROADCAST;
    ARPMessage arpmessage;
    arpmessage.opcode = ARPMessage::OPCODE_REQUEST;
    arpmessage.sender_ethernet_address = ethernet_address_;
    arpmessage.sender_ip_address = ip_address_.ipv4_numeric();
    arpmessage.target_ip_address = next_hop.ipv4_numeric();
    ethernetframe.payload = serialize( arpmessage );
    arp_life_.emplace( arpmessage.target_ip_address, make_pair( cur_time,
         ethernetframe ) );
    EthernetFrame waitframe{
        .header = {
          .dst = ETHERNET_BROADCAST,
          .src = ethernet_address_,
          .type = EthernetHeader::TYPE_IPv4,
        },
        .payload = serialize(dgram),
      };
    wait_queue_.emplace( next_hop.ipv4_numeric(), waitframe );
    // datagrams_received_.emplace(dgram);
    transmit( ethernetframe );
    return;
    }
    ethernetframe.header.type = EthernetHeader::TYPE_IPv4;
    ethernetframe.header.dst = arp_table_[next_hop.ipv4_numeric()].second;
    ethernetframe.payload = serialize( dgram );
    transmit( ethernetframe );
  }
```

In the frame_recv function,when receiving arp message,I need to update the
arp table and try to send the corresponding frames in the wait_queue.(Implementation omitted here)

三　Challenge