哈爾濱工業大學（深圳）
HARBIN INSTITUTE OF TECHNOLOGY

实验报告

开课学期：2023 春季

课程名称：计算机网络

实验名称：协议栈之 IP、ICMP、UDP 协议实现

学生班级：计算机 6 班

学生学号：200110619

学生姓名：梁鑫嵘

评阅教师：

报告成绩：

实验与创新实践教育中心制

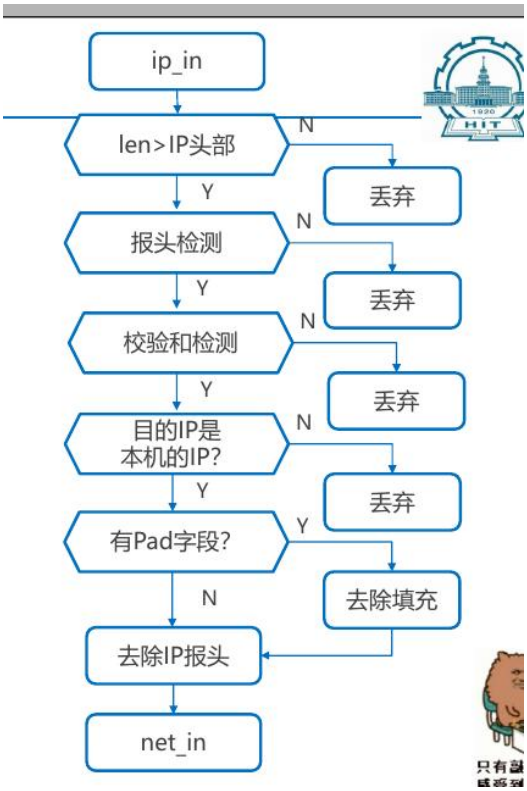
2023 年 3 月

一、实验详细设计

（注意不要完全照搬实验指导书上的内容，请根据你自己的设计方案来填写图文并茂地描述实验实现的所有功能和详细的设计方案及实验过程中的特色部分。）

1. IP 协议详细设计

ip_in 设计：



ip_in 参考了实验指导书中的流程图进行设计。

1. 丢弃长度小于 IP 头的数据包

```
// check package length
if (buf->len < sizeof(ip_hdr_t)) {
  Log("ip: package too short");
  return;
}
```

2. 丢弃长度小于 IP 头中的记录的包

```
ip_hdr_t *p = (ip_hdr_t *) buf->data;
if (buf->len < p->hdr_len << 2) {
  Log("ip: package shorter than header expected");
  return;
}
```

3. 丢弃协议不匹配的包：版本协议 4、hdr_len<5、DF 设置了但是过长的包

```
// check version, support ipv4 only
if (p->version != IP_VERSION_4) {
  Log("ip: invalid version %d", p->version);
  return;
}
// check header length
if (p->hdr_len < 5) {
  Log("ip: invalid header length %d", p->hdr_len);
  return;
}
// check DF bit
if (p->flags_fragment16 & IP_DO_NOT_FRAGMENT && buf->len >
ETHERNET_MAX_TRANSPORT_UNIT) {
  Log("ip: DF bit set, but it is a large frame");
  icmp_unreachable(buf, p->src_ip, ICMP_CODE_PROTOCOL_UNREACH);
  return;
}
```

4. 仅接收目标 IP 为自己 IP 的包

```
// check ip destination
if (memcmp(p->dst_ip, net_if_ip, NET_IP_LEN) != 0) {
  Log("ip: destination is %s, not mine", iptos(p->dst_ip));
  return;
}
```

5. 计算校验和

```
// checksum
uint16_t checksum_expected = p->hdr_checksum16;
p->hdr_checksum16 = 0;
uint16_t checksum_actual = checksum16((uint16_t *) buf->data,
sizeof(ip_hdr_t));
if (checksum_expected != checksum_actual) {
  Log("ip: checksum failed! expected: %x, actual: %x", checksum_expected,
checksum_actual);
  return;
}
p->hdr_checksum16 = checksum_expected;
```

6. 去除可能的 Padding 部分

```
uint16_t total_len = swap16(p->total_len16);
Dbg("ip: before remove padding, len=%zu, total_len16=%d", buf->len,
total_len);
```

```
// removing paddings
buf_remove_padding(buf, buf->len - total_len);
Dbg("ip: after remove padding, len=%zu", buf->len);
```

7. 去除 IP 包头

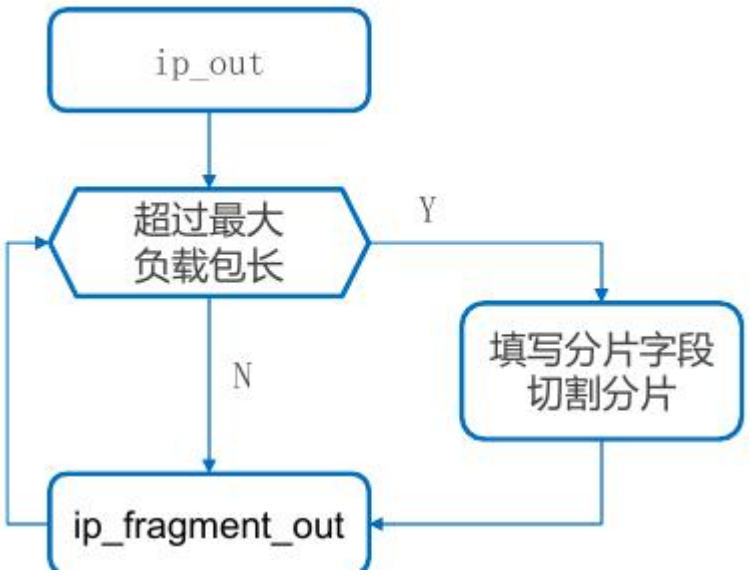
```
// remove ip header
buf_remove_header(buf, sizeof(ip_hdr_t));
```

8. 调用 net_in，如果返回值不正常，发送 ICMP protocal unreachable

```
if (net_in(buf, p->protocol, p->src_ip) < 0) {
  Log("ip: in, unrecognized protocol %d, send icmp protocol unreachable",
p->protocol);
  buf_add_header(buf, sizeof(ip_hdr_t));
  icmp_unreachable(buf, p->src_ip, ICMP_CODE_PROTOCOL_UNREACH);
}
```

ip_out 设计：

同样借鉴了指导书中的流程图




```
const size_t ip_max_length = ETHERNET_MAX_TRANSPORT_UNIT - sizeof(ip_hdr_t);
if (buf->len > ip_max_length) {
    // 分片后利用 ip_fragment_out 进行分片输出（见下）
} else {
    // 直接利用 ip_fragment_out 输出
  Dbg("ip: handle small package(%zu bytes)", buf->len);
  ip_fragment_out(buf, ip, protocol, ip_id++, 0, 0);
}
```

其中的分片方法，是通过不断备份还原 IP 头和 buf 指针数据的方式实现在同一个 buf 中完成分片，尽量减少数据拷贝过程。

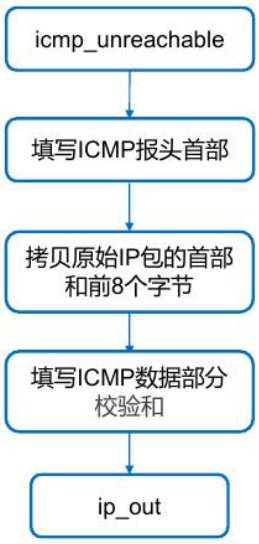
```
Log("ip: handle large package(%zu bytes)", buf->len);
```

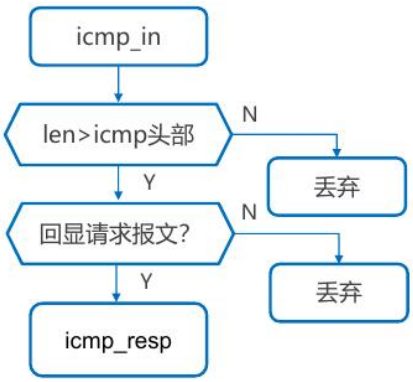
```
// split this package to multy packages, backup buf data
size_t original_len = buf->len;
uint8_t *original_data = buf->data;
buf->len = ip_max_length;
size_t offset = 0;
bool done = false;
uint8_t backup[sizeof(ip_hdr_t)];
// FIXME: this is a tricky way to handle large package
//  which reduce data copy but require lower layers to see package as immutable
while (!done) {
  if (offset + ip_max_length <= original_len) {
    // backup data in header area
    uint8_t *data_now = buf->data;
    memcpy(backup, data_now, sizeof(ip_hdr_t));
    ip_fragment_out(buf, ip, protocol, ip_id, offset, 1);
    // restore backup data
    memcpy(data_now, backup, sizeof(ip_hdr_t));
    offset += ip_max_length;
    buf->data = data_now + ip_max_length;
    // buf->len was changed in ip_fragment_out
    buf->len = ip_max_length;
  } else {
```

```
    buf->len = original_len - offset;
    // last len may be zero
    if (buf->len) ip_fragment_out(buf, ip, protocol, ip_id++, offset, 0);
    done = true;
  }
}
// restore this buf
buf->data = original_data;
buf->len = original_len;
```

2. ICMP 协议详细设计

ICMP 协议的设计参考自实验参考书中的流程图。






```
typedef enum icmp_code
{
    ICMP_CODE_PROTOCOL_UNREACH = 2, // 协议不可达
    ICMP_CODE_PORT_UNREACH = 3      // 端口不可达
} icmp_code_t;
```

```
void icmp_in(buf_t *buf, uint8_t *src_ip) {
  Log("icmp: in from %s", iptos(src_ip));
  // check package length
```

```
  if (buf->len < sizeof(icmp_hdr_t)) return;
  // if it's an echo request, send an echo reply
  icmp_hdr_t *icmp_hdr = (icmp_hdr_t *) buf->data;
  if (icmp_hdr->type == ICMP_TYPE_ECHO_REQUEST) {
    Log("icmp: ping recv from %s, send ping reply", iptos(src_ip));
    icmp_resp(buf, src_ip);
  }
}
void icmp_unreachable(buf_t *recv_buf, uint8_t *src_ip, icmp_code_t code) {
  Log("icmp: unreachable, send to %s, code=%d", iptos(src_ip), code);
  buf_init(&txbuf, sizeof(icmp_hdr_t) + sizeof(ip_hdr_t) + 8);
  icmp_hdr_t *p = (icmp_hdr_t *) txbuf.data;
  p->type = ICMP_TYPE_UNREACH;
  p->code = code;
  p->checksum16 = 0;
  p->id16 = 0;
  p->seq16 = 0;
  memcpy(txbuf.data + sizeof(icmp_hdr_t), recv_buf->data, sizeof(ip_hdr_t) + 8);
  p->checksum16 = checksum16((uint16_t *) txbuf.data, txbuf.len);
  ip_out(&txbuf, src_ip, NET_PROTOCOL_ICMP);
}
```

3. UDP 协议详细设计

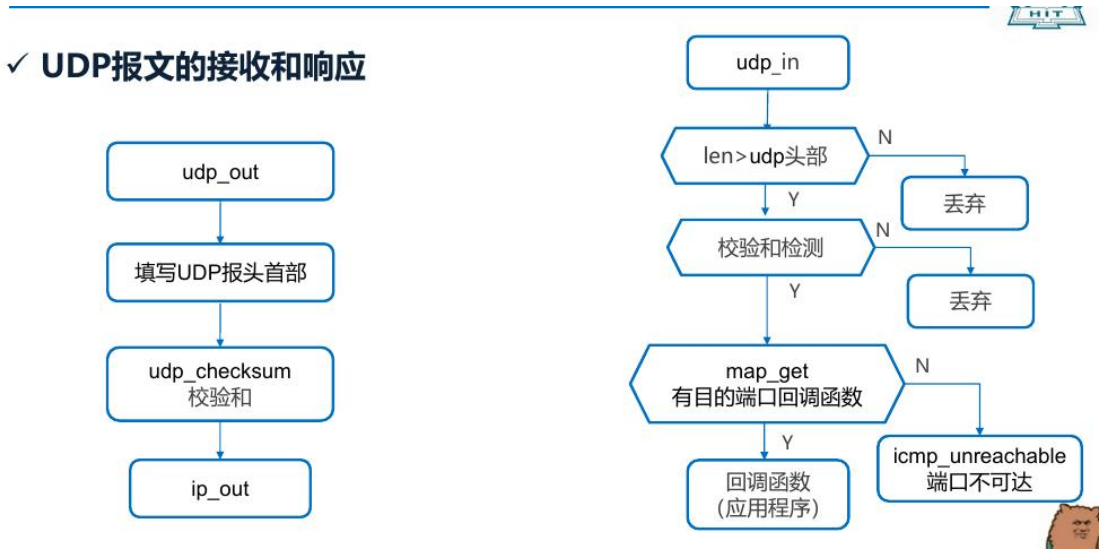
UDP 校验和计算：

需要在 UDP 包头前添加一个 UDP 伪头部，用于生成校验和。为了不损坏上一层 IP 层的头部信息，需要对对应 IP 头数据进行备份和还原。


```
// peso-header area in buf is mutable, must backup-restore it
uint8_t backup_ip_header[sizeof(ip_hdr_t)];
memcpy(backup_ip_header, buf->data - sizeof(ip_hdr_t), sizeof(ip_hdr_t));
ip_hdr_t *ip_header = (ip_hdr_t *) backup_ip_header;
uint16_t udp_length = buf->len;
// generate peso-header
buf_add_header(buf, sizeof(udp_peso_hdr_t));
udp_peso_hdr_t *peso = (udp_peso_hdr_t *) buf->data;
memcpy(peso->src_ip, src_ip, NET_IP_LEN);
memcpy(peso->dst_ip, dst_ip, NET_IP_LEN);
```

```
peso->placeholder = 0;

peso->protocol = ip_header->protocol;

peso->total_len16 = swap16(udp_length);

bool has_one_padding = false;

if (buf->len & 1) {

  Log("udp: checksum, data odd length (%zu), add 1 byte padding",

      buf->len - sizeof(udp_peso_hdr_t) - sizeof(udp_hdr_t));

  Assert(buf_add_padding(buf, 1) == 0, "Cannot add buf padding");

  has_one_padding = true;

}

// calculate checksum

uint16_t checksum = checksum16((uint16_t *) buf->data, buf->len);

if (has_one_padding) buf_remove_padding(buf, 1);

// restore backup

buf_remove_header(buf, sizeof(udp_peso_hdr_t));

memcpy(buf->data - sizeof(ip_hdr_t), backup_ip_header, sizeof(ip_hdr_t));

return checksum;
```

UDP 包的发送和接收：




```
void udp_in(buf_t *buf, uint8_t *src_ip) {
  // check package length
  if (buf->len < sizeof(udp_hdr_t)) {
    Log("udp: too short package! len(%zu) < udp_header_size(%llu)", buf->len,
sizeof(udp_hdr_t));
    return;
  }
  uint8_t src_ip_copy[NET_IP_LEN];
  memcpy(src_ip_copy, src_ip, sizeof(src_ip_copy));
  udp_hdr_t *p = (udp_hdr_t *) buf->data;
  uint16_t total_len = swap16(p->total_len16);
  if (buf->len < total_len) {
    Log("udp: too short package! len(%zu) < total_len(%d)", buf->len, total_len);
    return;
  }
```

```
  uint16_t dst_port = swap16(p->dst_port16);
  if (dst_port != 60000) {
    Dbg("udp: ignored port %d", dst_port);
    return;
  } else {
    Log("udp: recv target port package");
  }
  uint16_t checksum_expected = p->checksum16;
  if (checksum_expected == 0) {
    Log("udp: ignore checksum");
  } else {
    p->checksum16 = 0;
    uint16_t checksum_actual = udp_checksum(buf, src_ip_copy, net_if_ip);
    if (checksum_expected != checksum_actual) {
      Log("udp: checksum error! expected=%x, actual=%x", checksum_expected,
checksum_actual);
      return;
    }
    p->checksum16 = checksum_expected;
  }
  // check port handler
  udp_handler_t *handler = (udp_handler_t *) map_get(&udp_table, &dst_port);
```

```
  if (handler) {
    Log("udp: successfully call handler for port %d: %p", dst_port, handler);
    (*handler)(buf->data + sizeof(udp_hdr_t), buf->len - sizeof(udp_hdr_t),
src_ip_copy, swap16(p->src_port16));
  } else {
    Log("udp: no handler for port %d!", swap16(p->dst_port16));
  }
}
void udp_out(buf_t *buf, uint16_t src_port, uint8_t *dst_ip, uint16_t dst_port)
{
  // add udp header
  buf_add_header(buf, sizeof(udp_hdr_t));
  udp_hdr_t *p = (udp_hdr_t *) buf->data;
  p->src_port16 = swap16(src_port);
  p->dst_port16 = swap16(dst_port);
  p->total_len16 = swap16(buf->len);
  p->checksum16 = 0;
  p->checksum16 = udp_checksum(buf, net_if_ip, dst_ip);
  // send to ip layer
  ip_out(buf, dst_ip, NET_PROTOCOL_UDP);
}
```

二、实验结果截图及分析

（对你自己实验的测试结果进行评价）

1. IP 协议实验结果及分析

4: [/home/chiro/programs/net-lab/testing/ip_test.c:40 main] Test begin.

4: [/home/chiro/programs/net-lab/testing/ip_test.c:58 main] Feeding input 01

4: [/home/chiro/programs/net-lab/src/arp.c:50 arp_print] ===ARP TABLE BEGIN===

4: [/home/chiro/programs/net-lab/src/arp.c:52 arp_print] ===ARP TABLE END ===

4: [/home/chiro/programs/net-lab/src/arp.c:168 arp_out] arp: 192.168.163.10 not found, see if there is a pending request...

4: [/home/chiro/programs/net-lab/src/arp.c:176 arp_out] arp: 192.168.163.10 was added to arp buffer, and a request was sent

4: [/home/chiro/programs/net-lab/testing/ip_test.c:58 main] Feeding input 02

4: [/home/chiro/programs/net-lab/src/arp.c:117 arp_in] arp in: arp package from mac 21-32-43-54-65-06; sender ip=192.168.163.10, sender mac=21-32-43-54-65-06, target ip=192.168.163.10, target mac=21-32-43-54-65-06, hw_type=1, opcode=2

4: [/home/chiro/programs/net-lab/src/arp.c:122 arp_in] arp in: this is a arp reply

4: [/home/chiro/programs/net-lab/src/arp.c:50 arp_print] ===ARP TABLE BEGIN===

4: [/home/chiro/programs/net-lab/src/arp.c:42 arp_entry_print] 192.168.163.10 | 21-32-43-54-65-06 | 2023-05-04 07:55:36

4: [/home/chiro/programs/net-lab/src/arp.c:52 arp_print] ===ARP TABLE END ===

4: [/home/chiro/programs/net-lab/src/arp.c:128 arp_in] arp in: re-send the pending packet

4: [/home/chiro/programs/net-lab/testing/ip_test.c:58 main] Feeding input 03

4: [/home/chiro/programs/net-lab/testing/ip_test.c:58 main] Feeding input 04

4: [/home/chiro/programs/net-lab/src/ip.c:53 ip_in] ip: checksum failed! expected: ac20, actual: ac29

4: [/home/chiro/programs/net-lab/testing/ip_test.c:58 main] Feeding input 05

4: [/home/chiro/programs/net-lab/testing/faker/udp.c:56 udp_in] udp: in, src_ip=192.168.163.10, buf len=109

4: [/home/chiro/programs/net-lab/testing/ip_test.c:58 main] Feeding input 06

4: [/home/chiro/programs/net-lab/testing/ip_test.c:58 main] Feeding input 07

4: [/home/chiro/programs/net-lab/testing/faker/udp.c:56 udp_in] udp: in, src_ip=192.168.163.10, buf len=116

4: [/home/chiro/programs/net-lab/testing/ip_test.c:58 main] Feeding input 08

4: [/home/chiro/programs/net-lab/testing/ip_test.c:58 main] Feeding input 09

4: [/home/chiro/programs/net-lab/src/arp.c:117 arp_in] arp in: arp package from mac 01-12-23-34-45-56; sender ip=192.168.163.110, sender mac=01-12-23-34-45-56, target ip=192.168.163.110, target mac=01-12-23-34-45-56, hw_type=1, opcode=2

4: [/home/chiro/programs/net-lab/src/arp.c:122 arp_in] arp in: this is a arp reply

4: [/home/chiro/programs/net-lab/src/arp.c:50 arp_print] ===ARP TABLE BEGIN===

4: [/home/chiro/programs/net-lab/src/arp.c:42 arp_entry_print] 192.168.163.10 | 21-32-43-54-65-06 | 2023-05-04 07:55:36

4: [/home/chiro/programs/net-lab/src/arp.c:42 arp_entry_print] 192.168.163.110 | 01-12-23-34-45-56 | 2023-05-04 07:55:36

4: [/home/chiro/programs/net-lab/src/arp.c:52 arp_print] ===ARP TABLE END ===

4: [/home/chiro/programs/net-lab/testing/ip_test.c:58 main] Feeding input 10

4: [/home/chiro/programs/net-lab/src/arp.c:117 arp_in] arp in: arp package from mac 1A-94-F0-3C-49-AA; sender ip=192.168.163.2, sender mac=1A-94-F0-3C-49-AA, target ip=192.168.163.2, target mac=1A-94-F0-3C-49-AA, hw_type=1, opcode=1

4: [/home/chiro/programs/net-lab/src/arp.c:141 arp_in] arp in: this is a arp request, from ip=192.168.163.2, mac=1A-94-F0-3C-49-AA

4: [/home/chiro/programs/net-lab/testing/ip_test.c:58 main] Feeding input 11

4: [/home/chiro/programs/net-lab/testing/ip_test.c:58 main] Feeding input 12

4: [/home/chiro/programs/net-lab/src/ip.c:65 ip_in] ip: in, unrecognized protocol 6, send icmp protocol unreachable

4: [/home/chiro/programs/net-lab/testing/ip_test.c:58 main] Feeding input 13

4: [/home/chiro/programs/net-lab/testing/ip_test.c:58 main] Feeding input 14

4: [/home/chiro/programs/net-lab/testing/ip_test.c:58 main] Feeding input 15

4: [/home/chiro/programs/net-lab/src/ip.c:65 ip_in] ip: in, unrecognized protocol 6, send icmp protocol unreachable

4: [/home/chiro/programs/net-lab/testing/ip_test.c:81 main] Sample input all processed, checking output

4: [/home/chiro/programs/net-lab/testing/global.c:235 check_log] Checking log file(compare with demo).

4: [/home/chiro/programs/net-lab/testing/global.c:247 check_log] Round 1: no differences

4: [/home/chiro/programs/net-lab/testing/global.c:247 check_log] Round 2: no differences

4: [/home/chiro/programs/net-lab/testing/global.c:247 check_log] Round 3: no differences

4: [/home/chiro/programs/net-lab/testing/global.c:247 check_log] Round 4: no differences

4: [/home/chiro/programs/net-lab/testing/global.c:247 check_log] Round 5: no differences

4: [/home/chiro/programs/net-lab/testing/global.c:247 check_log] Round 6: no differences

4: [/home/chiro/programs/net-lab/testing/global.c:247 check_log] Round 7: no differences

4: [/home/chiro/programs/net-lab/testing/global.c:247 check_log] Round 8: no differences

4: [/home/chiro/programs/net-lab/testing/global.c:247 check_log] Round 9: no differences

4: [/home/chiro/programs/net-lab/testing/global.c:247 check_log] Round 10: no differences

4: [/home/chiro/programs/net-lab/testing/global.c:247 check_log] Round 11: no differences

4: [/home/chiro/programs/net-lab/testing/global.c:247 check_log] Round 12: no differences

4: [/home/chiro/programs/net-lab/testing/global.c:247 check_log] Round 13: no differences

4: [/home/chiro/programs/net-lab/testing/global.c:247 check_log] Round 14: no differences

4: [/home/chiro/programs/net-lab/testing/global.c:247 check_log] Round 15: no differences

4: [/home/chiro/programs/net-lab/testing/global.c:260 check_log] ====> All log rounds are the same to the demo.

4: [/home/chiro/programs/net-lab/testing/global.c:268 check_pcap] Checking pcap output file(compare with demo).

4: [/home/chiro/programs/net-lab/testing/global.c:342 check_pcap] Packet 1: no differences

4: [/home/chiro/programs/net-lab/testing/global.c:342 check_pcap] Packet 2: no differences

4: [/home/chiro/programs/net-lab/testing/global.c:342 check_pcap] Packet 3: no differences

4: [/home/chiro/programs/net-lab/testing/global.c:342 check_pcap] Packet 4: no differences

4: [/home/chiro/programs/net-lab/testing/global.c:342 check_pcap] Packet 5: no differences

4: [/home/chiro/programs/net-lab/testing/global.c:342 check_pcap] Packet 6: no differences

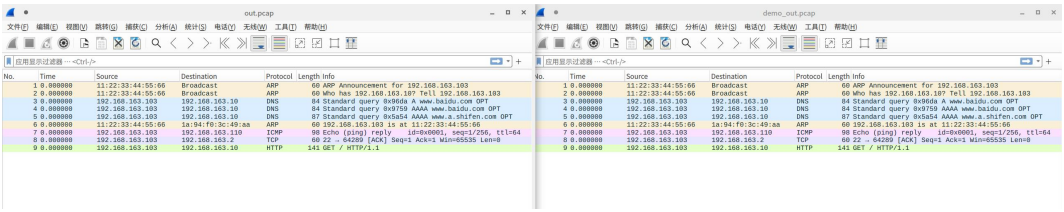
4: [/home/chiro/programs/net-lab/testing/global.c:342 check_pcap] Packet 7: no differences

4: [/home/chiro/programs/net-lab/testing/global.c:342 check_pcap] Packet 8: no differences

4: [/home/chiro/programs/net-lab/testing/global.c:342 check_pcap] Packet 9: no differences

4: [/home/chiro/programs/net-lab/testing/global.c:309 check_pcap] ====> All packets are the same to the demo.

4: [/home/chiro/programs/net-lab/testing/ip_test.c:98 main] For this test, log is only a reference. Your implementation is OK if your pcap file is the same to the demo pcap file.



测试中对 IP 协议的实现进行了检查，包括 IP checksum、protocal 不匹配等。经过检查，log、pcap 与目标一致。

检查 IP 分包：

5: [/home/chiro/programs/net-lab/testing/ip_frag_test.c:34 main] Feeding input.

5: [/home/chiro/programs/net-lab/src/ip.c:117 ip_out] ip: handle large package(5040 bytes)

5: [/home/chiro/programs/net-lab/testing/ip_frag_test.c:46 main] Comparing logs.

5: [/home/chiro/programs/net-lab/testing/ip_frag_test.c:69 main] Log file check passed

检查通过。

2. ICMP 协议实验结果及分析

6: [/home/chiro/programs/net-lab/testing/icmp_test.c:37 main] Test begin.

6: [/home/chiro/programs/net-lab/testing/icmp_test.c:54 main] Feeding input 01

6: [/home/chiro/programs/net-lab/src/arp.c:50 arp_print] ===ARP TABLE BEGIN===

6: [/home/chiro/programs/net-lab/src/arp.c:52 arp_print] ===ARP TABLE END ===

6: [/home/chiro/programs/net-lab/src/arp.c:168 arp_out] arp: 192.168.163.10 not found, see if there is a pending request...

6: [/home/chiro/programs/net-lab/src/arp.c:176 arp_out] arp: 192.168.163.10 was added to arp buffer, and a request was sent

6: [/home/chiro/programs/net-lab/testing/icmp_test.c:54 main] Feeding input 02

6: [/home/chiro/programs/net-lab/src/arp.c:117 arp_in] arp in: arp package from mac 21-32-43-54-65-06; sender ip=192.168.163.10, sender mac=21-32-43-54-65-06, target ip=192.168.163.10, target mac=21-32-43-54-65-06, hw_type=1, opcode=2

6: [/home/chiro/programs/net-lab/src/arp.c:122 arp_in] arp in: this is a arp reply

6: [/home/chiro/programs/net-lab/src/arp.c:50 arp_print] ===ARP TABLE BEGIN===

6: [/home/chiro/programs/net-lab/src/arp.c:42 arp_entry_print] 192.168.163.10 | 21-32-43-54-65-06 | 2023-05-04 08:02:30

6: [/home/chiro/programs/net-lab/src/arp.c:52 arp_print] ===ARP TABLE END ===

6: [/home/chiro/programs/net-lab/src/arp.c:128 arp_in] arp in: re-send the pending packet

6: [/home/chiro/programs/net-lab/testing/icmp_test.c:54 main] Feeding input 03

6: [/home/chiro/programs/net-lab/testing/icmp_test.c:54 main] Feeding input 04

6: [/home/chiro/programs/net-lab/src/ip.c:53 ip_in] ip: checksum failed! expected: ac20, actual: ac29

6: [/home/chiro/programs/net-lab/testing/icmp_test.c:54 main] Feeding input 05

6: [/home/chiro/programs/net-lab/testing/faker/udp.c:56 udp_in] udp: in, src_ip=192.168.163.10, buf len=109

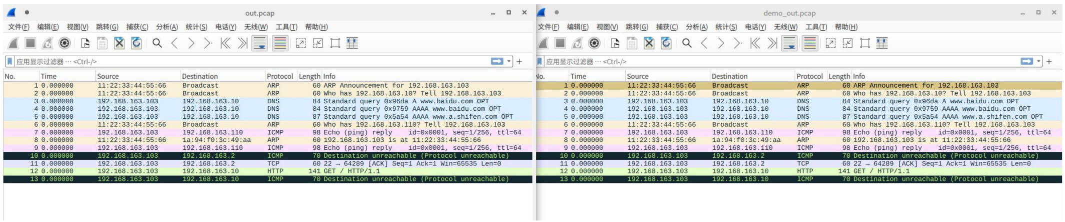
6: [/home/chiro/programs/net-lab/testing/icmp_test.c:54 main] Feeding input 06

6: [/home/chiro/programs/net-lab/testing/icmp_test.c:54 main] Feeding input 07

6: [/home/chiro/programs/net-lab/testing/faker/udp.c:56 udp_in] udp: in, src_ip=192.168.163.10,

buf len=116
6: [/home/chiro/programs/net-lab/testing/icmp_test.c:54 main] Feeding input 08
6: [/home/chiro/programs/net-lab/src/icmp.c:36 icmp_in] icmp: in from 192.168.163.110
6: [/home/chiro/programs/net-lab/src/icmp.c:42 icmp_in] icmp: ping recv from 192.168.163.110, send ping reply
6: [/home/chiro/programs/net-lab/src/icmp.c:13 icmp_resp] icmp: resp, req_buf len 64
6: [/home/chiro/programs/net-lab/src/arp.c:50 arp_print] ===ARP TABLE BEGIN===
6: [/home/chiro/programs/net-lab/src/arp.c:42 arp_entry_print] 192.168.163.10 | 21-32-43-54-65-06 | 2023-05-04 08:02:30
6: [/home/chiro/programs/net-lab/src/arp.c:52 arp_print] ===ARP TABLE END ===
6: [/home/chiro/programs/net-lab/src/arp.c:168 arp_out] arp: 192.168.163.110 not found, see if there is a pending request...
6: [/home/chiro/programs/net-lab/src/arp.c:176 arp_out] arp: 192.168.163.110 was added to arp buffer, and a request was sent
6: [/home/chiro/programs/net-lab/testing/icmp_test.c:54 main] Feeding input 09
6: [/home/chiro/programs/net-lab/src/arp.c:117 arp_in] arp in: arp package from mac 01-12-23-34-45-56; sender ip=192.168.163.110, sender mac=01-12-23-34-45-56, target ip=192.168.163.110, target mac=01-12-23-34-45-56, hw_type=1, opcode=2
6: [/home/chiro/programs/net-lab/src/arp.c:122 arp_in] arp in: this is a arp reply
6: [/home/chiro/programs/net-lab/src/arp.c:50 arp_print] ===ARP TABLE BEGIN===
6: [/home/chiro/programs/net-lab/src/arp.c:42 arp_entry_print] 192.168.163.10 | 21-32-43-54-65-06 | 2023-05-04 08:02:30
6: [/home/chiro/programs/net-lab/src/arp.c:42 arp_entry_print] 192.168.163.110 | 01-12-23-34-45-56 | 2023-05-04 08:02:30
6: [/home/chiro/programs/net-lab/src/arp.c:52 arp_print] ===ARP TABLE END ===
6: [/home/chiro/programs/net-lab/src/arp.c:128 arp_in] arp in: re-send the pending packet
6: [/home/chiro/programs/net-lab/testing/icmp_test.c:54 main] Feeding input 10
6: [/home/chiro/programs/net-lab/src/arp.c:117 arp_in] arp in: arp package from mac 1A-94-F0-3C-49-AA; sender ip=192.168.163.2, sender mac=1A-94-F0-3C-49-AA, target ip=192.168.163.2, target mac=1A-94-F0-3C-49-AA, hw_type=1, opcode=1
6: [/home/chiro/programs/net-lab/src/arp.c:141 arp_in] arp in: this is a arp request, from ip=192.168.163.2, mac=1A-94-F0-3C-49-AA
6: [/home/chiro/programs/net-lab/testing/icmp_test.c:54 main] Feeding input 11
6: [/home/chiro/programs/net-lab/testing/icmp_test.c:54 main] Feeding input 12
6: [/home/chiro/programs/net-lab/src/ip.c:65 ip_in] ip: in, unrecognized protocol 6, send icmp protocol unreachable
6: [/home/chiro/programs/net-lab/src/icmp.c:55 icmp_unreachable] icmp: unreachable, send to 192.168.163.2, code=2
6: [/home/chiro/programs/net-lab/testing/icmp_test.c:54 main] Feeding input 13
6: [/home/chiro/programs/net-lab/testing/icmp_test.c:54 main] Feeding input 14
6: [/home/chiro/programs/net-lab/testing/icmp_test.c:54 main] Feeding input 15
6: [/home/chiro/programs/net-lab/src/ip.c:65 ip_in] ip: in, unrecognized protocol 6, send icmp protocol unreachable
6: [/home/chiro/programs/net-lab/src/icmp.c:55 icmp_unreachable] icmp: unreachable, send to

192.168.163.10, code=2
6: [/home/chiro/programs/net-lab/testing/icmp_test.c:76 main] Sample input all processed, checking output
6: [/home/chiro/programs/net-lab/testing/global.c:235 check_log] Checking log file(compare with demo).
6: [/home/chiro/programs/net-lab/testing/global.c:247 check_log] Round 1: no differences
6: [/home/chiro/programs/net-lab/testing/global.c:247 check_log] Round 2: no differences
6: [/home/chiro/programs/net-lab/testing/global.c:247 check_log] Round 3: no differences
6: [/home/chiro/programs/net-lab/testing/global.c:247 check_log] Round 4: no differences
6: [/home/chiro/programs/net-lab/testing/global.c:247 check_log] Round 5: no differences
6: [/home/chiro/programs/net-lab/testing/global.c:247 check_log] Round 6: no differences
6: [/home/chiro/programs/net-lab/testing/global.c:247 check_log] Round 7: no differences
6: [/home/chiro/programs/net-lab/testing/global.c:247 check_log] Round 8: no differences
6: [/home/chiro/programs/net-lab/testing/global.c:247 check_log] Round 9: no differences
6: [/home/chiro/programs/net-lab/testing/global.c:247 check_log] Round 10: no differences
6: [/home/chiro/programs/net-lab/testing/global.c:247 check_log] Round 11: no differences
6: [/home/chiro/programs/net-lab/testing/global.c:247 check_log] Round 12: no differences
6: [/home/chiro/programs/net-lab/testing/global.c:247 check_log] Round 13: no differences
6: [/home/chiro/programs/net-lab/testing/global.c:247 check_log] Round 14: no differences
6: [/home/chiro/programs/net-lab/testing/global.c:247 check_log] Round 15: no differences
6: [/home/chiro/programs/net-lab/testing/global.c:260 check_log] ====> All log rounds are the same to the demo.
6: [/home/chiro/programs/net-lab/testing/global.c:268 check_pcap] Checking pcap output file(compare with demo).
6: [/home/chiro/programs/net-lab/testing/global.c:342 check_pcap] Packet 1: no differences
6: [/home/chiro/programs/net-lab/testing/global.c:342 check_pcap] Packet 2: no differences
6: [/home/chiro/programs/net-lab/testing/global.c:342 check_pcap] Packet 3: no differences
6: [/home/chiro/programs/net-lab/testing/global.c:342 check_pcap] Packet 4: no differences
6: [/home/chiro/programs/net-lab/testing/global.c:342 check_pcap] Packet 5: no differences
6: [/home/chiro/programs/net-lab/testing/global.c:342 check_pcap] Packet 6: no differences
6: [/home/chiro/programs/net-lab/testing/global.c:342 check_pcap] Packet 7: no differences
6: [/home/chiro/programs/net-lab/testing/global.c:342 check_pcap] Packet 8: no differences
6: [/home/chiro/programs/net-lab/testing/global.c:342 check_pcap] Packet 9: no differences
6: [/home/chiro/programs/net-lab/testing/global.c:342 check_pcap] Packet 10: no differences
6: [/home/chiro/programs/net-lab/testing/global.c:342 check_pcap] Packet 11: no differences
6: [/home/chiro/programs/net-lab/testing/global.c:342 check_pcap] Packet 12: no differences
6: [/home/chiro/programs/net-lab/testing/global.c:342 check_pcap] Packet 13: no differences
6: [/home/chiro/programs/net-lab/testing/global.c:309 check_pcap] ====> All packets are the same to the demo.
6: [/home/chiro/programs/net-lab/testing/icmp_test.c:93 main] For this test, log is only a reference. Your implementation is OK if your pcap file is the same to the demo pcap file.



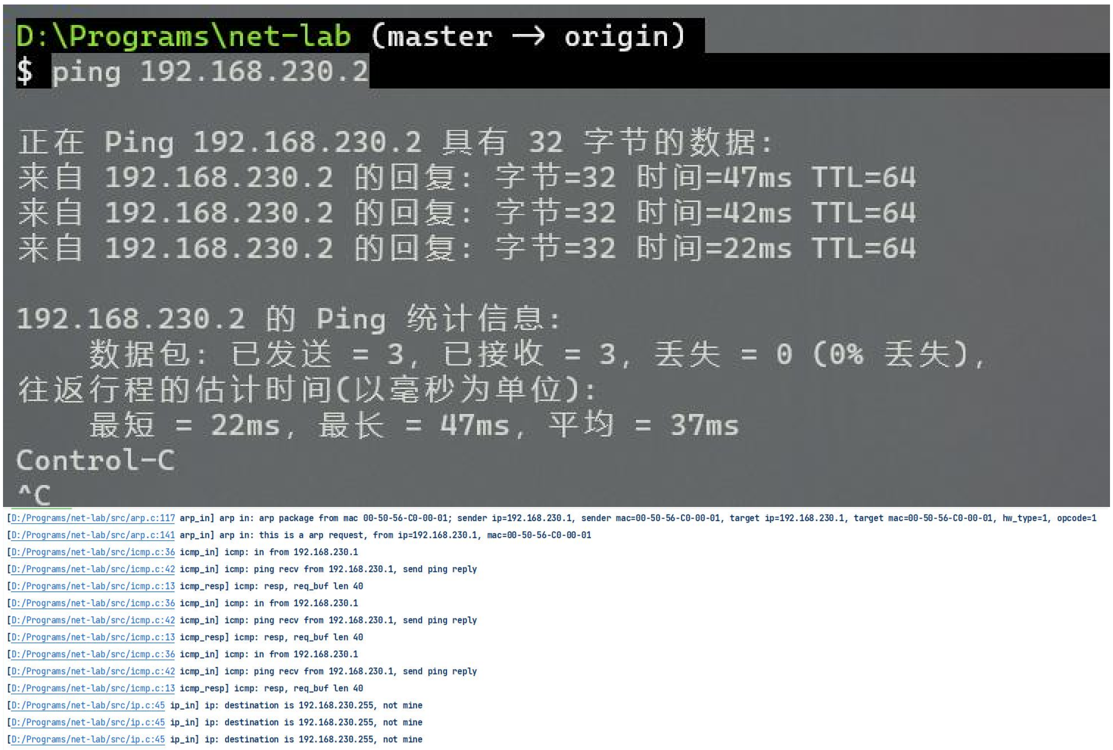
实验中主要检查了 IP 协议不匹配时发送的 icmp protocal unreachable 包。

3. UDP 协议实验结果及分析

（本小节还需要分析你自己用 Wireshark 抓包工具捕获到的相关报文（包含 UDP 和 ARP 报文），解析报文内容）

将 config.h 里的相关配置完成修改。

首先检查数据是否可以正常收发，先 ping 一下。



能够正常回应 ICMP ping。

UDP 收发正常。

三、实验中遇到的问题及解决方法

（包括设计过程中的错误及测试过程中遇到的问题）

有时候会忘记大小端转换，将对应数据转换过后就好了。

在 UDP 校验添加伪头部的时候，需要注意备份还原的 IP 头的位置，我的实现中曾经弄错了位置，比较难调试出来。

在我的 IP 分包实现中使用了比较特殊的方法在同一个 buf 中实现了分包，这在一些情况下可能不适用。

四、实验收获和建议

（实验过程中的收获、感受、问题、建议等）

挺好。