

（深圳）

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Eth、ARP协议实现

学生班级： 计科 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

# 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*

1. Eth协议详细设计

Ethernet部分逻辑相对比较简单，主要逻辑如下：

*void* ethernet\_in(*buf\_t* \**buf*) {  
 *ether\_hdr\_t* \*hdr = (*ether\_hdr\_t* \*) *buf*->data;  
 *uint16\_t* length\_type = swap16(hdr->protocol16);  
 *// if is broadcast or is for me, handle it（接收指向本机的 MAC 或者广播的 MAC）  
 if* (memcmp(hdr->dst, net\_if\_mac, **NET\_MAC\_LEN**) == 0 ||  
 memcmp(hdr->dst, net\_broadcast\_mac, **NET\_MAC\_LEN**) == 0) {  
 **Dbg**("ethernet: package for me, dst=%s, src=%s, length/type=%x(%d)", mactos(buf->data),  
 mactos(hdr->src), length\_type, length\_type);  
 *if* (46 <= length\_type && length\_type <= 1500) {  
 *// this is a length field（这里不处理将 length\_type 指定为 length 的特殊的包）* **Err**("ethernet: unknown protocol! length = %d", length\_type);  
 } *else if* (length\_type >= 0x0600) {  
 *// this is a type field（移除头部，做下一层处理）* buf\_remove\_header(*buf*, *sizeof*(*ether\_hdr\_t*));  
 net\_in(*buf*, length\_type, hdr->src);  
 } *else* {  
 **Log**("ethernet: invalid length/type field, drop this packet");  
 *return*;  
 }  
 } *else* {  
 **Log**("ethernet: package not for me, dst=%s, src=%s", mactos(*buf*->data), mactos(*buf*->data + NET\_MAC\_LEN));  
 }  
}

*void* ethernet\_out(*buf\_t* \**buf*, *const uint8\_t* \**mac*, *net\_protocol\_t protocol*) {  
 *protocol* = swap16(*protocol*);  
 **Dbg**("ethernet: out, mac=%s, protocol=%x, payload size=%zu", mactos(mac), protocol, buf->len);  
 *// if smaller than 46, pad it  
 if* (*buf*->len < 46) {  
 buf\_add\_padding(*buf*, 46 - *buf*->len);  
 }  
 buf\_add\_header(*buf*, *sizeof*(*ether\_hdr\_t*));  
 *ether\_hdr\_t* \*hdr = (*ether\_hdr\_t* \*) *buf*->data;  
 memcpy(hdr->src, net\_if\_mac, **NET\_MAC\_LEN**);  
 memcpy(hdr->dst, *mac*, **NET\_MAC\_LEN**);  
 hdr->protocol16 = *protocol*;  
 driver\_send(*buf*);  
}

接收时：判断目标MAC是否含本机MAC，如果是而且protocol合法则接收进入下一层

发送时：如果包过短则添加paddings，然后填写eth头并发出

1. ARP协议详细设计

AR协议本质是一个用于同步多个网络节点之间的IP-MAC对应关系表的协议。

主要接收逻辑：

1. 包长度、协议、目的地验证
2. 如果是一个ARP\_REPLY，则写入arp\_table并发送缓存中目标地址的所有网络包
3. 如果是一个请求自己的ARP\_REQUEST，则对目标MAC发送ARP\_REPLY相应

*void* arp\_in(*buf\_t* \**buf*, *uint8\_t* \**src\_mac*) {  
 **Dbg**("arp: in from %s", mactos(src\_mac));  
 *// check package length  
 if* (*buf*->len < *sizeof*(*arp\_pkt\_t*)) {  
 **Log**("arp: invalid package length");  
 *return*;  
 }  
 *// check package  
 arp\_pkt\_t* \*p = (*arp\_pkt\_t* \*) *buf*->data;  
 *if* (p->pro\_type16 == **constswap16**(NET\_PROTOCOL\_IP) &&  
 p->hw\_len == **NET\_MAC\_LEN** && p->pro\_len == **NET\_IP\_LEN**) {  
 *// handle arp reply* **Log**("arp in: arp package from mac %s; sender ip=%s, sender mac=%s, target ip=%s, target mac=%s, hw\_type=%d, opcode=%d",  
 mactos(*src\_mac*), iptos(p->sender\_ip), mactos(p->sender\_mac), iptos(p->target\_ip),  
 mactos(p->target\_mac), swap16(p->hw\_type16), swap16(p->opcode16));  
 *if* (p->opcode16 == **constswap16**(ARP\_REPLY) && memcmp(p->target\_ip, net\_if\_ip, **NET\_IP\_LEN**) == 0 &&  
 memcmp(p->target\_mac, net\_if\_mac, **NET\_MAC\_LEN**) == 0) {  
 **Log**("arp in: this is a arp reply");  
 map\_set(&arp\_table, p->sender\_ip, p->sender\_mac);  
 arp\_print();  
 *// flush pending buffer  
 queue\_t* \*pending\_queue = (*queue\_t* \*) map\_get(&arp\_buf, p->sender\_ip);  
 *if* (pending\_queue) {  
 **Log**("arp in: re-send the pending packet");  
 *buf\_t* \*queued\_buf;  
 *while* ((queued\_buf = queue\_pop(pending\_queue)) != **NULL**)  
 ethernet\_out(queued\_buf, p->sender\_mac, NET\_PROTOCOL\_IP);  
 *// remove this item in pending buffer* map\_delete(&arp\_buf, p->sender\_ip);  
 *// items in this queue was constructed by malloc, so can free them  
 // but queue struct is map data, cannot free* queue\_free\_data(pending\_queue, **true**);  
 }  
 } *else* {  
 *// handle arp request  
 if* (p->opcode16 == **constswap16**(ARP\_REQUEST) && memcmp(p->target\_ip, net\_if\_ip, **NET\_IP\_LEN**) == 0) {  
 **Log**("arp in: this is a arp request, from ip=%s, mac=%s", iptos(p->sender\_ip), mactos(p->sender\_mac));  
 *// update arp table* map\_set(&arp\_table, p->sender\_ip, p->sender\_mac);  
 *// send reply* arp\_resp(p->sender\_ip, p->sender\_mac);  
 }  
 }  
 } *else* {  
 **Log**("arp in: invalid package! pro\_type=%x, target ip=%s, target mac=%s", swap16(p->pro\_type16), iptos(p->target\_ip),  
 mactos(p->target\_mac));  
 }  
}

主要发送逻辑：

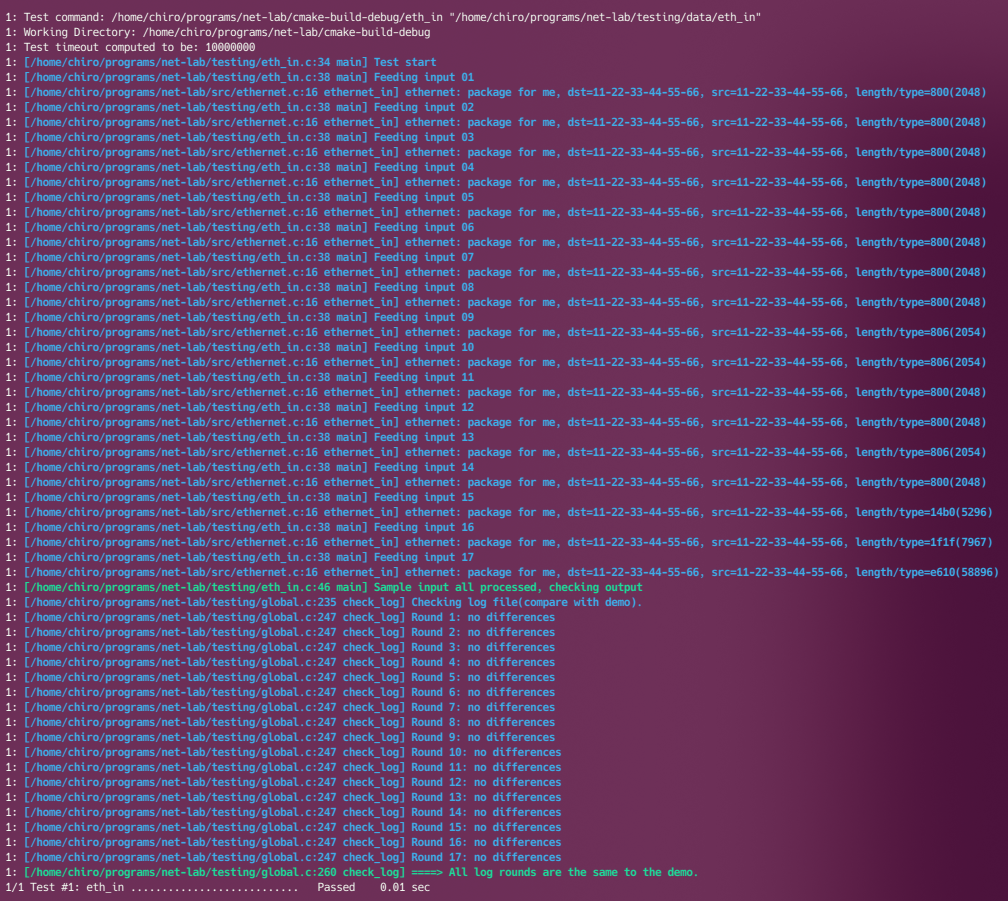
1. 如果目标IP不在arp\_table中而且arp请求未发送，就先缓存包并发出arp请求
2. 如果目标IP在arp\_table中，则直接发送到对应MAC
3. 在系统初始化时需要向外发送arp通告协议帮助网络中的其他节点立即更新arp表

*void* arp\_out(*buf\_t* \**buf*, *uint8\_t* \**ip*) {  
 *if* (*ip*[0] == 0) **Err**("arp: out to %s", iptos(*ip*));  
 *else* **Dbg**("arp: out to %s", iptos(ip));  
 *// find mac in arp table  
 uint8\_t* \*mac = (*uint8\_t* \*) map\_get(&arp\_table, *ip*);  
 *if* (!mac) {  
 arp\_print();  
 **Log**("arp: %s not found, see if there is a pending request...", iptos(*ip*));  
 *queue\_t* \*pending\_queue = (*queue\_t* \*) map\_get(&arp\_buf, *ip*);  
 *if* (pending\_queue) {  
 **Log**("arp: a pending request queue found, push this request to queue");  
 *buf\_t* \*copy = malloc(*sizeof*(*buf\_t*));  
 buf\_copy(copy, *buf*, 0);  
 queue\_push(pending\_queue, copy);  
 } *else* {  
 **Log**("arp: %s was added to arp buffer, and a request was sent", iptos(*ip*));  
 *// add to pending buffer  
 buf\_t* \*copy = malloc(*sizeof*(*buf\_t*));  
 buf\_copy(copy, *buf*, 0);  
 *queue\_t* \*q = queue\_new(copy);  
 map\_set(&arp\_buf, *ip*, q);  
 *// queue indexes was copied to map data, free the queue struct* free(q);  
 *// not found, send a request* arp\_req(*ip*);  
 }  
 } *else* {  
 *// found, send the packet* ethernet\_out(*buf*, mac, NET\_PROTOCOL\_IP);  
 }  
}

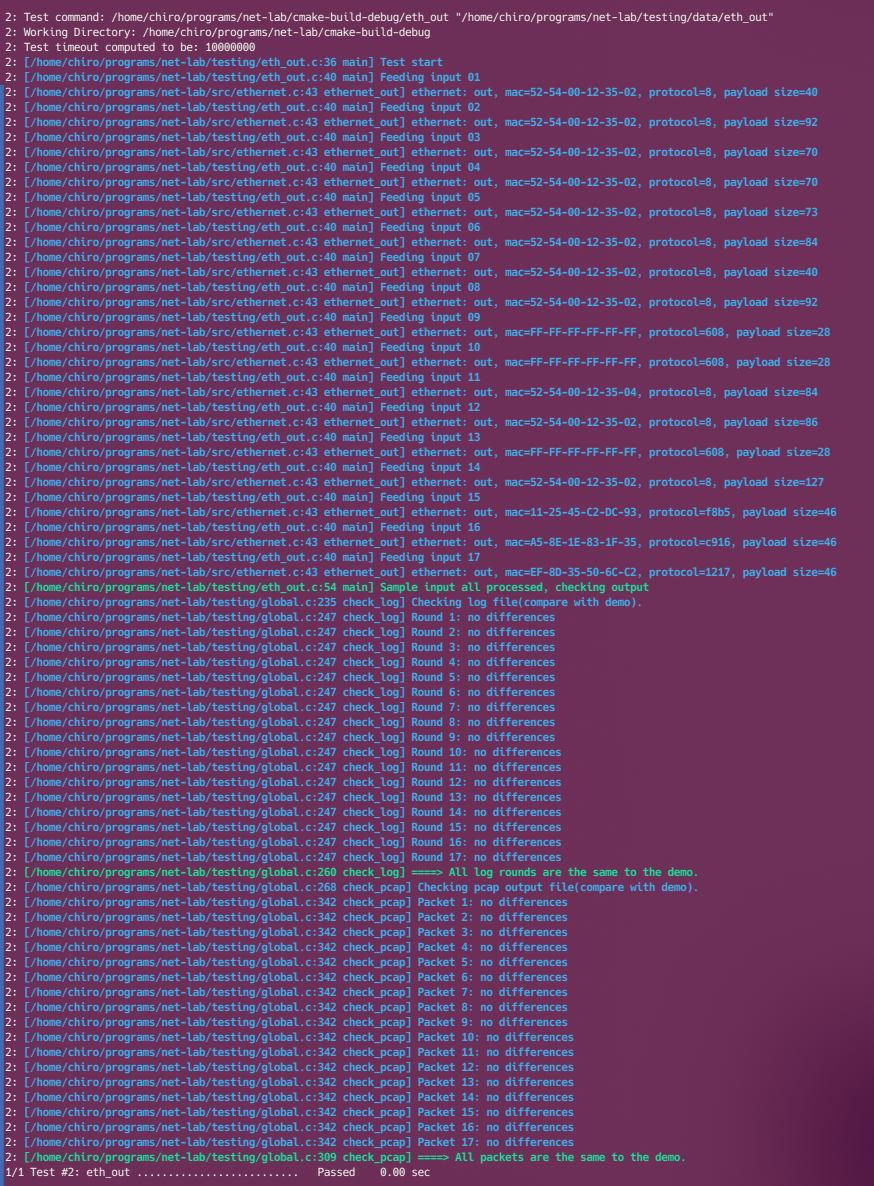
# 二、实验结果截图及分析

1. Eth协议实验结果及分析

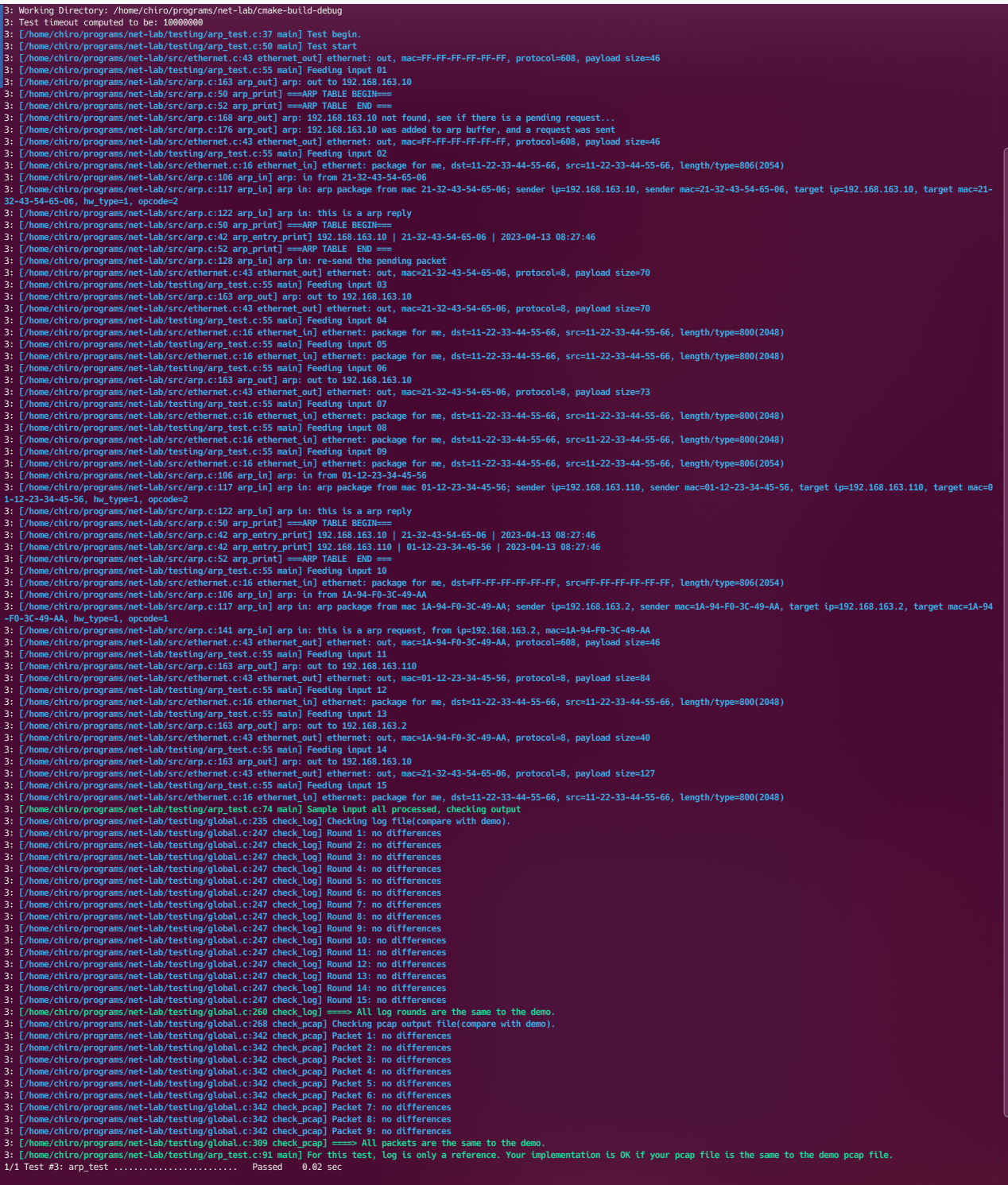
eth\_in：



eth\_out：



1. ARP协议实验结果及分析



1. 想要发送一个网络包，但是目标IP不在ARP表中，于是将包缓存到arp **buffer，并发出arp请求。**

**arp: 192.168.163.10 was added to arp buffer, and a request was sent**

1. 收到了arp请求的回复，更新arp table，发出buffer中的包

**arp in: this is a arp reply**   
**===ARP TABLE BEGIN===**   
**192.168.163.10 | 21-32-43-54-65-06 | 2023-04-13 08:27:46**   
**===ARP TABLE  END ===**   
**arp in: re-send the pending packet**   
**ethernet: out, mac=21-32-43-54-65-06, protocol=8, payload size=70**

实验中有两次以上过程的重复。