# Jos Lab 6: Network Driver

**Li Xinyu 515030910292**

# Part A: Initialization and transmitting packets

> **Exercise 1**
> 修改`kern/trap.c`中的函数`trap_dispatch()`，在条件`case IRQ_OFFSET+IRQ_TIMER`下增加调用`time_tick()`；
> 修改`kern/syscall.c`，在函数`sys_time_msec()`中返回`time_msec()`，并在函数`syscall`的swtich语句下添加一条`case SYS_time_msec: ret = sys_time_msec();`。

### PCI Interface

> **Exercise 3**
> 创建函数`e1000_attach(struct pci_func *f)`，调用`pci_func_enable(f)`，根据手册5.2节，82540EM的Vender ID是0x8086，Device ID是0x100e，在pci_attach_vendor中{0,0,0}添加一条entry：`{ E1000_VENDOR_ID, E1000_DEVICE_ID, &e1000_attach}`。

## Memory-mapped I/O

> **Exercise 4**
> 在函数`e1000_attach`，调用`boot_map_region(kern_pgdir, E1000_BAR0, f->reg_size[0], f->reg_base[0], PTE_W|PTE_PCD|PTE_PWT)`为E1000的BAR 0在KSTACKTOP之上创建内存映射区域。

## C Structures

> **Exercise 5**
> 申明变量`volatile void* e1000`用来访问内存映射区域，`struct tx_desc tx_queue[E1000_NTXDESC] __attribute__((aligned(16)));`和`struct tx_pkt tx_pkt_buf[E1000_NTXDESC];`用来存放transmit descriptor和packet，然后根据手册初始化Transmit Descriptor Register：TDBAH、TDBAL、TDLEN、TDH、TDT、TCTL、TIPG，以及`tx_queue`。相关常量的宏定义从e1000_hw.h中得到。

-

> 

> **Exercise 6**
> 创建 `e1000_transmit(const char* data, uint32_t len)`，如果 `len > E1000_TX_PKT_LEN` 则返回 `-E_INVAL`，否则根据E1000_TDT寄存器得到transmit descriptor queue中的尾节点，如果 `tx_queue[cur].status` 为 `E1000_TXD_STAT_DD` 则队列已满返回 `-E_TXQ_FULL`。将 `data` 拷贝到当前descriptor的pkt中 `tx_pkt_buf[cur].pkt`，然后更新 `tx_queue[cur]` 的属性，最后将E1000_TDT寄存器更新为队列中的下一个。

-

> **Exercise 7**
> 在 `inc/lib.h`， `inc/syscall.h`， `lib/syscall.h` 中添加系统调用 `sys_net_transmit`，在 `kern/syscall.c` 中添加函数 `sys_net_try_transmit(const char* data, uint32_t len)` 调用 `e1000_transmit(data, len)`，在函数 `syscall` 的switch语句中添加一条 `case SYS_net_try_transmit: ret = sys_net_try_transmit((char *)a1, a2);`。

## Transmitting Packets: Network Server

> **Exercise 8**
> 修改函数 `output()`，使用死循环 `while(1)` 进行接收和发送，调用 `sys_ipc_recv(&nsipcbuf)` 接收network server的packet，如果 `thisenv->env_ipc_value == NSREQ_OUTPUT` 则调用 `sys_net_try_transmit(nsipcbuf.pkt.jp_data, nsipcbuf.pkt.jp_len))` 将packet发送到device driver，返回 `-E_TXQ_FULL` 时进行重试。

# Part B: Receiving packets and the web server

## Receiving Packets

> **Exercise 10**
> 和**Exercise 5**类似， `struct rcv_desc rcv_queue[E1000_NRCVDESC] __attribute__((aligned(16)));` 和 `struct rcv_pkt rcv_pkt_buf[E1000_NRCVDESC];` 用来存放receive descriptor和packet，然后根据手册初始化Receive Descriptor Register：RDBAH、RDBAL、RDLEN、RDH、RDT、RCTL，以及 `tx_queue`，对RAL和RAH的初始化使用硬编码的MAC地址。

-

> **Exercise 11**
> 创建 `e1000_receive(char* data)`，根据E1000_TDT寄存器得到transmit descriptor

queue中的尾节点，然后使用尾节点的下一个来存放packet，若 `tx_queue[cur].status` 为 `E1000_RXD_STAT_DD` 则队列已空返回 `-E_RCVQ_EMPTY` 。将当前 descriptor `rcv_pkt_buf[cur].pkt` 拷贝到 `data` ，然后更新 `rcv_queue[cur]` 的属性以及E1000_RDT寄存器。

## Receiving Packets: Network Server

**Exercise 12**
修改函数 `input()` ，使用死循环 `while(1)` 进行接收和发送，调用 `sys_page_alloc(0, &nsipcbuf, PTE_W|PTE_U|PTE_P)` 为 `nsipcbuf` 分配一个页用于接收device driver的packet，然后调用 `sys_net_receive(nsipcbuf.pkt.jp_data)` 接收packet，当返回值小于0时重复接收并调用 `sys_yeild()` ，最后调用 `ipc_send(ns_envid, NSREQ_INPUT, &nsipcbuf, PTE_U|PTE_P)` 将packet发送到network server。

## The Web Server

**Exercise 13**
修改函数 `send_data()` ，首先调用 `fstat(fd, &stat)` 检查文件大小，然后调用 `readn(fd, buf, stat.st_size)` ，最后调用 `write(req->sock, buf, stat.st_size)` 将文件内容通过socket发送给client。
修改函数 `send_file()` ，调用 `fd = open(req->url, O_RDONLY)` 打开文件，若 `fd < 0` 则 `send_error(req, 404)` ，然后调用 `fstat(fd, &stat)` ，如果 `stat.st_isdir` 为真则 `send_error(req, 404)` ，否则将 `file_size` 赋值为 `stat.st_size` 。

# Challenge

**Read MAC addres from the EEPROM**
创建函数 `e1000_read_eeprom(uint8_t addr)` ，使用EEPROM Read Register读取MAC地址，根据手册5.3.1和13.4.4中的描述，将EERD.START(最低位bit)置为1，将EERD.ADDR(第8-15bits)置为要读取的word的位置，然后进入循环等待直到EERD.DONE(第4bits)为1，然后读取EERD.DATA(第16-31bits)中2bytes内容。
在函数 `e1000_attach()` 中调用 `e1000_read_eeprom` 三次读取6bytes的MAC地址，存到字节数组 `e1000_mac` 中，然后初始化E1000_RAL和E1000_RAH寄存器。
在 `inc/lib.h` ， `inc/syscall.h` ， `lib/syscall.h` 中添加系统调用 `sys_net_mac` ，在 `kern/syscall.c` 中添加函数 `sys_net_mac(uint8_t* mac)` 将 `e1000_mac` 拷贝到参数 `mac` 中，在函数 `syscall` 的switch语句中添加一条 `case SYS_net_mac: ret = sys_net_mac((uint8_t *)a1)` 。
修改 `jif.c` 中的函数 `low_level_init()` ，调用 `sys_net_mac((uint8_t*)netif->hwaddr)` 初始化MAC地址取代硬编码。