Networking 实验报告

专　业：软　件　工　程

指　导　老　师：王　冬　青

年　级：大　学　二　年　级

学　号：2153061

姓　名：谢　嘉　麒

1. networking (hard)

1.1.实验目的

本实验要求使用名为E1000的网络设备来处理网络通信。需要完成kernel/e1000.c文件下的e1000_transmit()和e1000_recv()两个函数。

1.2.实验步骤

1.2.1.实验准备

根据提示，我们需要参考E1000软件开发手册的以下几个章节：

（1）第2节：提供整个设备的概述；

（2）第3.2节：提供数据包接收的概述；

（3）第3.3节：提供数据包发送的概述；

（4）第13节：提供E1000使用寄存器的概述；

（5）第14节：有助于理解初始化代码；

实验提到需要理解初始化代码，因此关注kernel/e1000.c文件下的e1000_init()函数。结合实验手册不难看出，初始化时给出两个环形缓冲区tx_ring和rx_ring，这两个缓冲区将作为数据包的存放地点，并在更新后视为数据包发送。

1.2.2.e1000_transmit()的实现

根据1.2.1.的分析，只需要将发送的数据包放入环形缓冲区，设置好参数并更新管理缓冲区的寄存器，网卡的硬件即可自动将数据包发送出去。即在ring中找到下一个空余位置，将包放进去等待传输即可。

函数具体实现思路如下：

（1）获取E1000的锁，防止多个进程同时访问；

（2）获取发送队列的当前描述符索引，即将regs[E1000_TDT]存放在idx中；

（3）定义struct tx_desc类型的指针desc，获取当前描述符的指针&tx_ring][idx]；

（4）检查当前描述符状态是否为已完成，不是则释放E1000的锁并返回-1，表示发送队列溢出；

（5）检查当前索引tx_mbufs，若有值则说明该位置缓冲区使用过，需要使用mbuffree(tx_mbufs[idx])释放；

（6）使用以下命令配置当前描述符

```
desc->addr = (uint64)m->head; // 设置数据包的物理地址
desc->length = m->len; // 设置数据包的长度
desc->cmd = E1000_TXD_CMD_RS | E1000_TXD_CMD_EOP; // 设置命令，RS表示报告状态，EOP表示数据包结束
tx_mbufs[idx] = m;// 在tx_mbufs数组中记录该描述符对应的mbuf
```

（7）更新发送队列的描述符索引，即 regs[E1000_TDT] = (idx + 1) % TX_RING_SIZE;

（8）使用__sync_synchronize();同步内存，使更新对其他CPU可见；

（9）释放E1000的锁并返回0；

1.2.3.e1000_recv()的实现

此函数需要遍历ring，将新的包交付网络上层的协议/应用。根据xv6实验手册，拷贝过程无需实现，调用net_rx()即可。当网卡硬件产生一次中断后，中断处理过程e1000_intr会执行regs[E1000_ICR] = 0xffffffff来说明已完成中断中所有数据报的处理，需要将rx_ring所有内容进行拷贝。

函数具体实现思路如下：

（1）获取接收队列的下一个描述符索引 (regs[E1000_RDT] + 1) % RX_RING_SIZE；

（2）获取当前描述符的指针 &rx_ring[idx]并存入struct rx_desc* 类型的指针变量desc中；

（3）当描述符状态为未完成时，开启以下循环；

（4）循环中，获取E1000的锁，防止多个进程同时访问；

（5）获取接受队列对应的mbuf，并将收到的数据包用mbufput(buf, desc->length)函数放入mbuf；

（6）为下一个描述符分配新的mbuf，即 rx_mbufs[idx] = mbufalloc(0)；若分配失败，触发panic；

（7）设置新的描述符信息如下

```
desc->addr = (uint64) rx_mbufs[idx]->head; // 设置下一个描述符的物理地址
desc->status = 0; // 重置描述符的状态
regs[E1000_RDT] = idx; // 更新接收队列的描述符索引
```

（8）利用 __sync_synchronize()同步内存，确保更新对其他CPU可见；

（9）释放E1000的锁；

（10）利用net_rx(buf)将mbuf传递给网络层进行进一步的处理；

（11）获取下一个描述符索引和下一个描述符指针如下

```
idx = (regs[E1000_RDT] + 1) % RX_RING_SIZE; // 获取下一个描述符索引
desc = &rx_ring[idx]; // 获取下一个描述符的指针
```

1.2.4.编译与检测

完成以上步骤后，执行make server指令可以看到qemu宿主机的服务器正常运行：

```
e1000_transmit: called mbuf=0x0000000087ec0000
e1000_transmit: called mbuf=0x0000000087ec1000
e1000_transmit: called mbuf=0x0000000087ec2000
e1000_transmit: called mbuf=0x0000000087ec3000
e1000_transmit: called mbuf=0x0000000087ec4000
e1000_transmit: called mbuf=0x0000000087ee4000
e1000_transmit: called mbuf=0x0000000087ee2000
e1000_transmit: called mbuf=0x0000000087ef4000
OK
testing multi-process pings: e1000_transmit: called mbuf=0x0
0
e1000_transmit: called mbuf=0x0000000087e98000
e1000_transmit: called mbuf=0x0000000087e97000
e1000_transmit: called mbuf=0x0000000087e4c000
e1000_transmit: called mbuf=0x0000000087e4d000
e1000_transmit: called mbuf=0x0000000087e4b000
e1000_transmit: called mbuf=0x0000000087e4a000
e1000_transmit: called mbuf=0x0000000087e49000
e1000_transmit: called mbuf=0x0000000087e48000
e1000_transmit: called mbuf=0x0000000087e4f000
OK
```

执行make grade也可以看到测试通过：

```
== Test running nettests ==
$ make qemu-gdb
Timeout! (32.5s)
== Test   nettest: ping ==
  nettest: ping: OK
== Test   nettest: single process ==
  nettest: single process: OK
== Test   nettest: multi-process ==
  nettest: multi-process: OK
```

1.3.实验中遇到的问题和解决方法

本次实验需要对网卡及其驱动程序有一定的了解，同时由于实验需要自主实现的部分并不复杂，故完成比较顺利。但需要注意一点，即网卡本身是仅有一份的独占资源，因此在进行操作时需要加锁。同时当缓冲区满的时候，需要加入__sync_synchronize()，起到内存屏障的作用，确保更新对其他CPU可见，保证对内存的操作严格按照指定的顺序进行。

1.4.实验心得

通过本次实验，笔者对xv6网卡及其驱动程序有了一定的了解，同时理解了E1000网卡驱动这个使用DMA的经典例子。同时在准备和编写网卡驱动程序的过程中，笔者也对编写驱动程序的一般步骤进行总结，从理解数据结构，创建内核并初始化，编写功能几个方面逐一进行学习和掌握。