Lab 11: networking

1950787 杨鑫

实验目的：

- 了解计算机网络的一些知识
- 编写传输和接受数据包的函数
- 了解网络通信的大致结构和所需要的一些寄存器，数据结构等信息

实验步骤：

首先切换分支至 net branch，以获取本次实验的内容

同时清理文件，以获得纯净的初始文件系统

```
git checkout net
make clean
```

a. Job

实验目的：

您将使用称为 E1000 的网络设备来处理网络通信。对于 xv6（以及您编写的驱动程序），E1000 看起来像是连接到真实以太网局域网 (LAN) 的真实硬件。实际上，您的驱动程序将与之通信的 E1000 是由 qemu 提供的仿真，连接到同样由 qemu 仿真的 LAN。在这个模拟 LAN 上，xv6（“guest”）的 IP 地址为 10.0.2.15。 Qemu 还安排运行 qemu 的计算机出现在 IP 地址为 10.0.2.2 的 LAN 上。当 xv6 使用 E1000 向 10.0.2.2 发送数据包时，qemu 会将数据包传送到您正在运行 qemu（“主机”）的（真实）计算机上的适当应用程序。

您将使用 QEMU 的“用户模式网络堆栈”。 QEMU 的文档在这里有更多关于用户模式堆栈的信息。我们更新了 Makefile 以启用 QEMU 的用户模式网络堆栈和 E1000 网卡。

Makefile 将 QEMU 配置为将所有传入和传出的数据包记录到您实验室目录中的文件 packet.pcap 中。查看这些记录以确认 xv6 正在传输和接收您期望的数据包可能会有所帮助。

您的工作是在 kernel/e1000.c 中完成 e1000_transmit() 和 e1000_recv()，以便驱动程序可以传输和接收数据包。当 make Grade 表示您的解决方案通过了所有测试时，您就完成了。

实验分析：

由于本人还未学习计算机网络及其相关知识，所以在本次实验中遇到许多名词和概念都不太清楚，但是好在实验指导说明的十分详细，所以本人就尝试跟着实验指导完成了所需要的一些代码。

大致要求是要实现数据包的传输和接受功能，这里要读取相关的寄存器以获取所期待的信息；并且要修改一些数据结构以完成相应功能。

实验核心代码：

e1000.c

```
int
e1000_transmit(struct mbuf *m)
{
  //
  // Your code here.
  //
  // the mbuf contains an ethernet frame; program it into
  // the TX descriptor ring so that the e1000 sends it. Stash
  // a pointer so that it can be freed after sending.
  //
  acquire(&e1000_lock);
  uint32 tail = regs[E1000_TDT];
  if (!(tx_ring[tail].status & E1000_TXD_STAT_DD))
    return -1;

  if (tx_mbufs[tail])
    mbuffree(tx_mbufs[tail]);
  tx_mbufs[tail] = m;
  tx_ring[tail].addr = (uint64)m->head;
  tx_ring[tail].length = m->len;
  tx_ring[tail].cmd = E1000_TXD_CMD_EOP | E1000_TXD_CMD_RS;

  regs[E1000_TDT] = (tail + 1) % TX_RING_SIZE;
  release(&e1000_lock);
  return 0;
}

static void
e1000_recv(void)
{
  //
  // Your code here.
  //
  // Check for packets that have arrived from the e1000
  // Create and deliver an mbuf for each packet (using net_rx()).
  //
  uint32 tail = regs[E1000_RDT];
  int i = (tail + 1) % RX_RING_SIZE;

  while(rx_ring[i].status & E1000_RXD_STAT_DD){
    rx_mbufs[i]->len = rx_ring[i].length;
    net_rx(rx_mbufs[i]);
    rx_mbufs[i] = mbufalloc(0);
    rx_ring[i].addr = (uint64)rx_mbufs[i]->head;
    rx_ring[i].status = 0;
    i = (i + 1) % RX_RING_SIZE;
  }
  regs[E1000_RDT] = (i - 1 + RX_RING_SIZE) % RX_RING_SIZE;
}
```

实验评分：

按照实验要求对本次实验的所有小实验进行评分，结果如图：

```
== Test running nettests ==
$ make qemu-gdb
(3.9s)
== Test   nettest: ping ==
  nettest: ping: OK
== Test   nettest: single process ==
  nettest: single process: OK
== Test   nettest: multi-process ==
  nettest: multi-process: OK
== Test   nettest: DNS ==
  nettest: DNS: OK
== Test time ==
time: OK
Score: 100/100
```

通过了所有测试

问题以及解决办法：

a.cmd 标志如何设置

需要查看 Intel 文档以获取 cmd 所代表的相应的含义：

7	6	5	4	3	2	1	0
IDE	VLE	DEXT	RSV RPS[a]	RS	IC	IFCS	EOP

a. **82544GC/EI** only.

TDESC.CMD	Description
IDE (bit 7)	Interrupt Delay Enable When set, activates the transmit interrupt delay timer. The Ethernet controller loads a countdown register when it writes back a transmit descriptor that has RS and IDE set. The value loaded comes from the IDV field of the Interrupt Delay (TIDV) register. When the count reaches 0, a transmit interrupt occurs if transmit descriptor write-back interrupts (IMS.TXDW) are enabled. Hardware always loads the transmit interrupt counter whenever it processes a descriptor with IDE set even if it is already counting down due to a previous descriptor. If hardware encounters a descriptor that has RS set, but not IDE, it generates an interrupt immediately after writing back the descriptor. The interrupt delay timer is cleared.
VLE (bit 6)	VLAN Packet Enable When set, indicates that the packet is a VLAN packet and the Ethernet controller should add the VLAN Ethertype and an 802.1q VLAN tag to the packet. The Ethertype field comes from the VET register and the VLAN tag comes from the special field of the TX descriptor. The hardware inserts the FCS/CRC field in that case. When cleared, the Ethernet controller sends a generic Ethernet packet. The IFCS controls the insertion of the FCS field in that case. In order to have this capability CTRL.VME bit should also be set, otherwise VLE capability is ignored. VLE is valid only when EOP is set.
DEXT (bit 5)	Extension (0b for legacy mode). Should be written with 0b for future compatibility.
RPS RSV (bit 4)	Report Packet Sent When set, the **82544GC/EI** defers writing the DD bit in the status byte (DESC.STATUS) until the packet has been sent, or transmission results in an error such as excessive collisions. It is used is cases where the software must know that the packet has been sent, and not just loaded to the transmit FIFO. The **82544GC/EI** might continue to prefetch data from descriptors logically after the one with RPS set, but does not advance the descriptor head pointer or write back any other descriptor until it sent the packet with the RPS set. RPS is valid only when EOP is set. This bit is reserved and should be programmed to 0b for all Ethernet controllers except the **82544GC/EI**.
RS (bit 3)	Report Status When set, the Ethernet controller needs to report the status information. This ability may be used by software that does in-memory checks of the transmit descriptors to determine which ones are done and packets have been buffered in the transmit FIFO. Software does it by looking at the descriptor status byte and checking the Descriptor Done (DD) bit.

TDESC.CMD	Description
IC (bit 2)	Insert Checksum When set, the Ethernet controller needs to insert a checksum at the offset indicated by the CSO field. The checksum calculations are performed for the entire packet starting at the byte indicated by the CCS field. IC is ignored if CSO and CCS are out of the packet range. This occurs when (CSS ≥ length) OR (CSO ≥ length - 1). IC is valid only when EOP is set.
IFCS (bit 1)	Insert FCS Controls the insertion of the FCS/CRC field in normal Ethernet packets. IFCS is valid only when EOP is set.
EOP (bit 0)	End Of Packet When set, indicates the last descriptor making up the packet. One or many descriptors can be used to form a packet.

这里，需要使用 EOP 和 RS 两位：EOP 表示该 buffer 含有一个完整的 packet，该处是这个 packet 的最后一个描述符；RS 告诉网卡在发送完成后，设置 status 中的 E1000_TXD_STAT_DD 位，表示发送完成。

实验心得：

1. 略微了解了一些计算机网络的知识
2. 实现了数据包的传输和接受功能
3. 了解了相关的寄存器和数据结构组织