

# EI313 Lab5

---

## EI313 Lab5

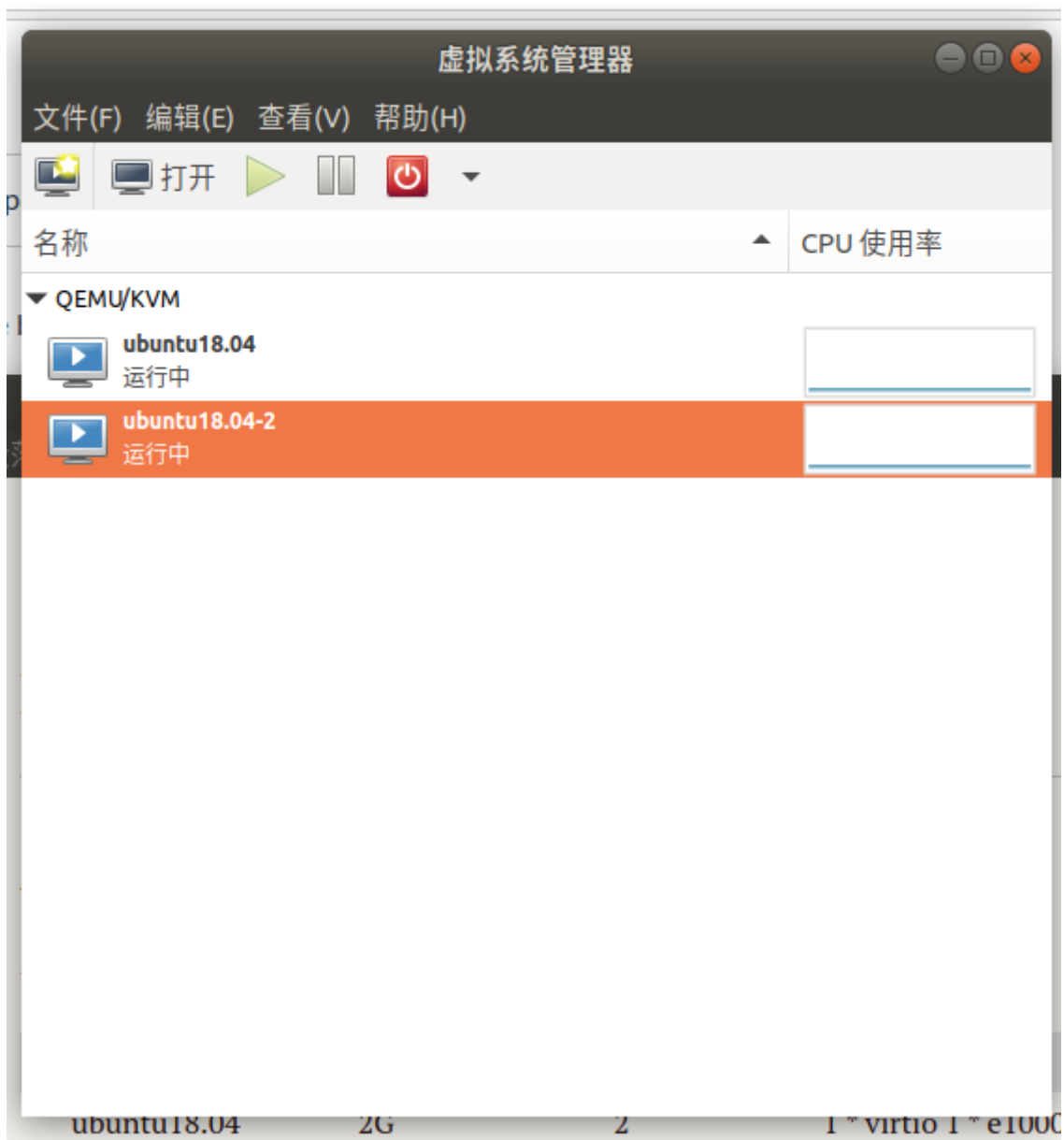
- 1、创建虚拟机
- 2、编译和安装DPDK
  - 2.1、安装编译DPDK所需要的依赖库
  - 2.2、编译DPDK
- 3、绑定网卡
  - 3.1、在host上给虚拟机创建新网卡
  - 3.2、将网卡与DPDK绑定
- 4、测试DPDK
  - 4.1、安装并编译pktgen-dpdk
  - 4.2、运行l2fwd
  - 4.3、运行pktgen-dpdk
- 5、实验结果及分析
  - 5.1、单张virtio网卡
  - 5.2、1张e1000网卡
  - 5.3、2张e1000网卡两个端口
  - 5.4、实验结果总结
- 6、总结

## 1、创建虚拟机

---

利用virt-manger创建两个虚拟机，两个虚拟机的参数如下：

名称	内存大小	CPU个数	网卡参数
ubuntu18.04	2G	2	1 * virtio 1 * e1000
ubuntu18.04-2	2G	2	1 * virtio 2 * e1000



## 2、编译和安装DPDK

### 2.1、安装编译DPDK所需要的依赖库

```
sudo su
apt install build-essential
apt install python3 python3-pip
apt install python3-pyelftools
apt install libnuma-dev
pip3 install meson(这里要用pip3来安装而不是apt install，否则会出现meson版本不够高而编译报错)
pip3 install ninja
```

### 2.2、编译DPDK

DPDK官网上下载最新的DPDK压缩包

```
tar xf dpdk.tar.gz
cd dpdk
sudo su
meson build
cd build
ninja
ninja install
ldconfig
```

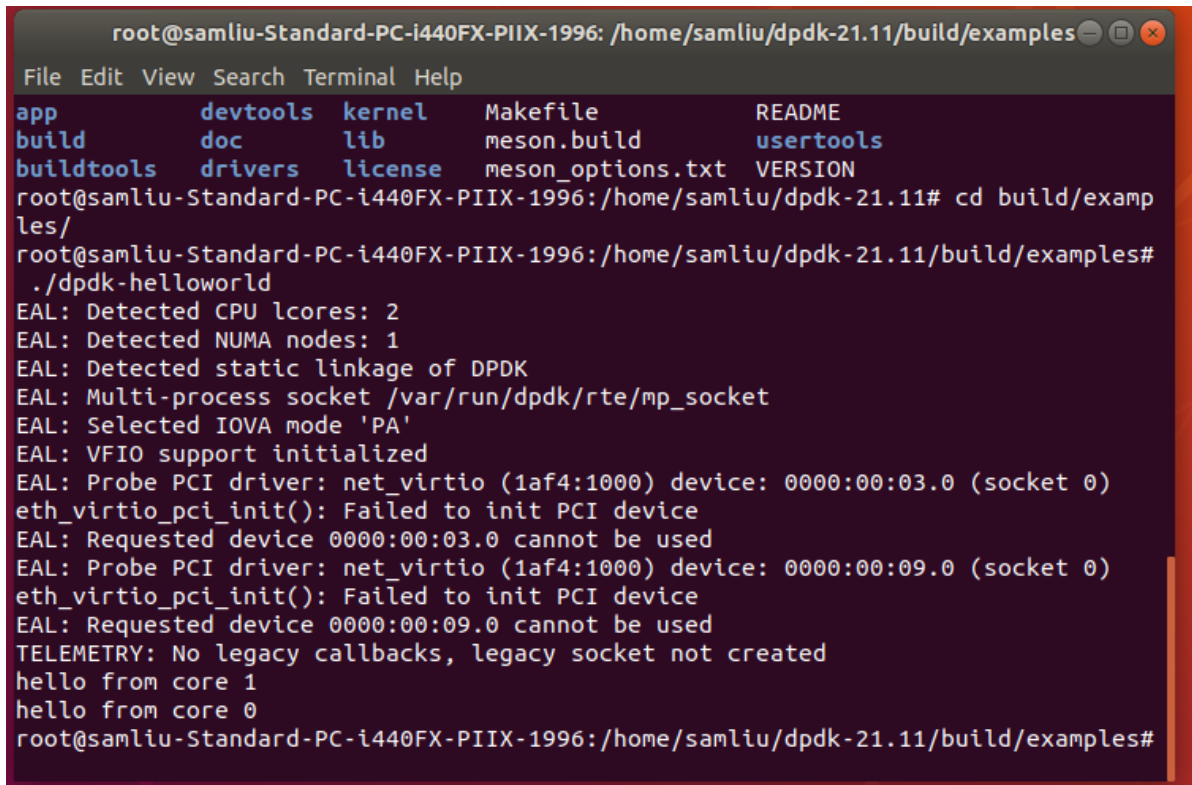
再编译所有DPDK的example应用

```
cd build
meson configure -Dexamples=all
ninja
```

在运行example文件前，需要给虚拟机配置大页，否则会出现报错

```
sudo su
echo 1024 > /sys/kernel/mm/hugepages/hugepages-2048kB/nr_hugepages
cd build/example
./dpdk-helloworld
```

helloworld输出如下，证明DPDK已经编译成功。

A terminal window titled 'root@samliu-Standard-PC-i440FX-PIIX-1996: /home/samliu/dpdk-21.11/build/examples' showing the execution of 'dpdk-helloworld'. The output includes system information (CPU cores, NUMA nodes), DPDK initialization logs (static linkage, multi-process socket, IOVA mode, VFIO support), and two 'hello' messages from core 1 and core 0. The terminal also displays a directory listing of the build directory.

```
root@samliu-Standard-PC-i440FX-PIIX-1996: /home/samliu/dpdk-21.11/build/examples
File Edit View Search Terminal Help
app      devtools kernel  Makefile  README
build    doc      lib     meson.build usertools
buildtools drivers license meson_options.txt VERSION
root@samliu-Standard-PC-i440FX-PIIX-1996:/home/samliu/dpdk-21.11# cd build/examples/
root@samliu-Standard-PC-i440FX-PIIX-1996:/home/samliu/dpdk-21.11/build/examples# ./dpdk-helloworld
EAL: Detected CPU lcores: 2
EAL: Detected NUMA nodes: 1
EAL: Detected static linkage of DPDK
EAL: Multi-process socket /var/run/dpdk/rte/mp_socket
EAL: Selected IOVA mode 'PA'
EAL: VFIO support initialized
EAL: Probe PCI driver: net_virtio (1af4:1000) device: 0000:00:03.0 (socket 0)
eth_virtio_pci_init(): Failed to init PCI device
EAL: Requested device 0000:00:03.0 cannot be used
EAL: Probe PCI driver: net_virtio (1af4:1000) device: 0000:00:09.0 (socket 0)
eth_virtio_pci_init(): Failed to init PCI device
EAL: Requested device 0000:00:09.0 cannot be used
TELEMETRY: No legacy callbacks, legacy socket not created
hello from core 1
hello from core 0
root@samliu-Standard-PC-i440FX-PIIX-1996:/home/samliu/dpdk-21.11/build/examples#
```

## 3、绑定网卡

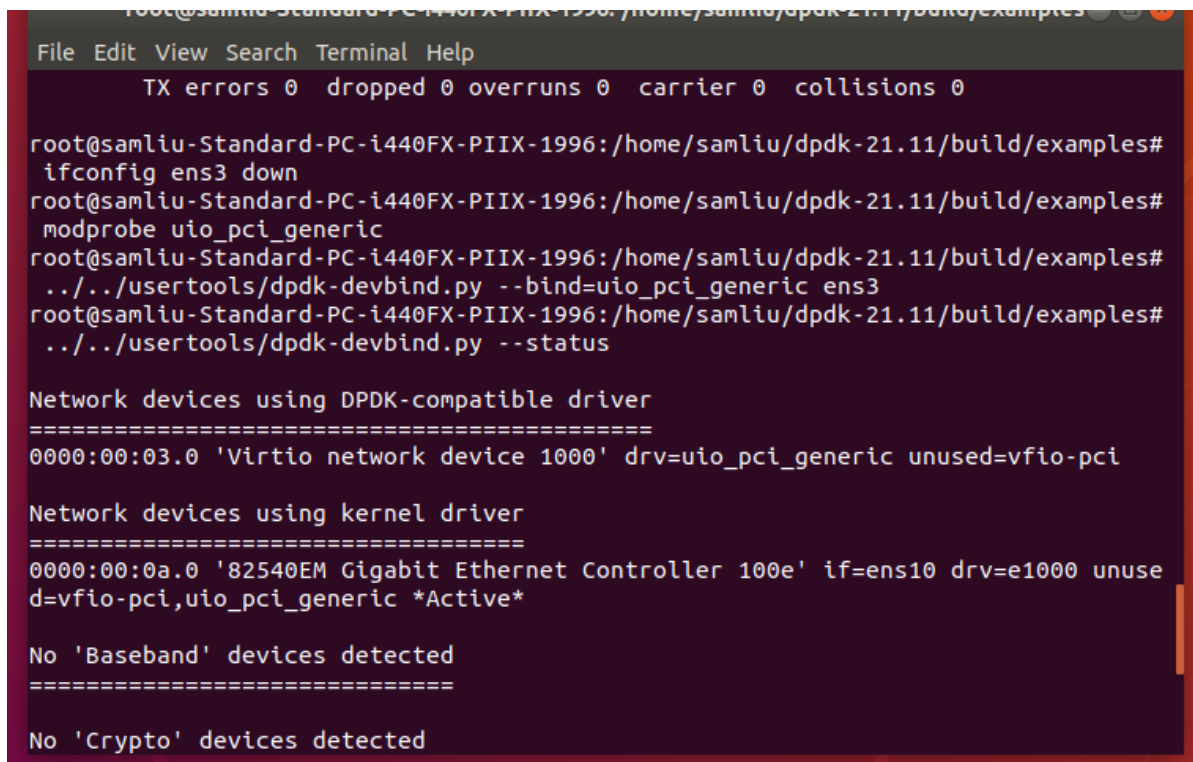
### 3.1、在host上给虚拟机创建新网卡

```
sudo su
virsh attach-interface [VM name] --type bridge --source virbr0 --model [e1000 or virtio]
加载到配置文件
virsh attach-interface [VM name] --type bridge --source virbr0 --model [e1000 or virtio] --config
virsh dumpxml [VM name] >/etc/libvirt/qemu/snale.xml
virsh define /etc/libvirt/qemu/snale.xml
```

由此虚拟机上就会被创建出新的网卡。

## 3.2、将网卡与DPDK绑定

- 首先安装net-tools, 用ifconfig查看网卡配置。
- ifconfig [网卡名称] down 将网卡解绑
- 加载uio\_pci\_generic功能: sudo modprobe uio\_pci\_generic
- 绑定网卡到uio\_pci\_generic下: sudo usertools/dpdk-devbind.py --bind=uio\_pci\_generic [网卡名称]
- 最后查看虚拟机网卡设备状态: sudo usertools/dpdk-devbind.py --status



```
File Edit View Search Terminal Help
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@samliu-Standard-PC-i440FX-PIIX-1996:/home/samliu/dpdk-21.11/build/examples#
ifconfig ens3 down
root@samliu-Standard-PC-i440FX-PIIX-1996:/home/samliu/dpdk-21.11/build/examples#
modprobe uio_pci_generic
root@samliu-Standard-PC-i440FX-PIIX-1996:/home/samliu/dpdk-21.11/build/examples#
../../usertools/dpdk-devbind.py --bind=uio_pci_generic ens3
root@samliu-Standard-PC-i440FX-PIIX-1996:/home/samliu/dpdk-21.11/build/examples#
../../usertools/dpdk-devbind.py --status

Network devices using DPDK-compatible driver
=====
0000:00:03.0 'Virtio network device 1000' drv=uio_pci_generic unused=vfio-pci

Network devices using kernel driver
=====
0000:00:0a.0 '82540EM Gigabit Ethernet Controller 100e' if=ens10 drv=e1000 unused=vfio-pci,uio_pci_generic *Active*

No 'Baseband' devices detected
=====

No 'Crypto' devices detected
```

如上图所示, 成功在虚拟机上将virtio类型的网卡与DPDK相绑定。

## 4、测试DPDK

### 4.1、安装并编译pktgen-dpdk

pktgen-dpdk并不是dpdk压缩包内有的examples, 因此需要从github上面git clone下来, 再使用meson+ninja编译。

```
git clone git://dpdk.org/apps/pktgen-dpdk
cd pktgen-dpdk
sudo su
apt install cmake
apt install -y libnuma-dev
apt install -y libpcap-dev
apt install pkg-config
meson build
cd build
ninja install
```

如此编译完毕后就可以开始运行pktgen-dpdk了。

## 4.2、运行l2fwd

在运行l2fwd之前要先打开巨页，否则会出现报错。由于在之前已经编译了DPDK所有的examples文件，因此可以直接在build/examples目录下找到l2fwd文件。

l2fwd命令使用方法如下：

```
./dpdk-l2fwd -l 0-1 -n 2 -- -p 0x1 -q 2 -T 10
抽象化为：
./dpdk-l2fwd [EAL options] -- -p PORTMASK [-q NQ -T t]
```

- EAL options
  - -l 0-1 表示使用0和1两个CPU
  - -n 2 表示每个CPU使用两个处理队列
- -p PORTMASK: portmask为端口数量掩码，因为只绑定了一张网卡，因此掩码为0x1，表示只有第一个端口处于激活状态。
- -q NQ: NQ表示分配个每个逻辑内核的收发队列数量
- -T t: t表示打印统计数据到屏幕上的时间间隔

如果直接运行l2fwd，我发现在还没有开始运行pktgen时，就会有收发包的情况，因此为了避免这种情况，关闭混杂模式，作出如下修改：将光标处上一句注释掉，设置ret=0，重新编译。

```
main.c
~/dpdk-21.11/examples/l2fwd
Save
/* Disable type parsing (needed for promiscuous mode) */
portid);

/* Start device */
ret = rte_eth_dev_start(portid);
if (ret < 0)
    rte_exit(EXIT_FAILURE, "rte_eth_dev_start:err=%d, port=%u\n",
              ret, portid);

printf("done: \n");
if (promiscuous_on) {
    //ret = rte_eth_promiscuous_enable(portid);
    ret=0;
    if (ret != 0)
        rte_exit(EXIT_FAILURE,
                  "rte_eth_promiscuous_enable:err=%s, port=%u\n",
                  rte_strerror(-ret), portid);
}

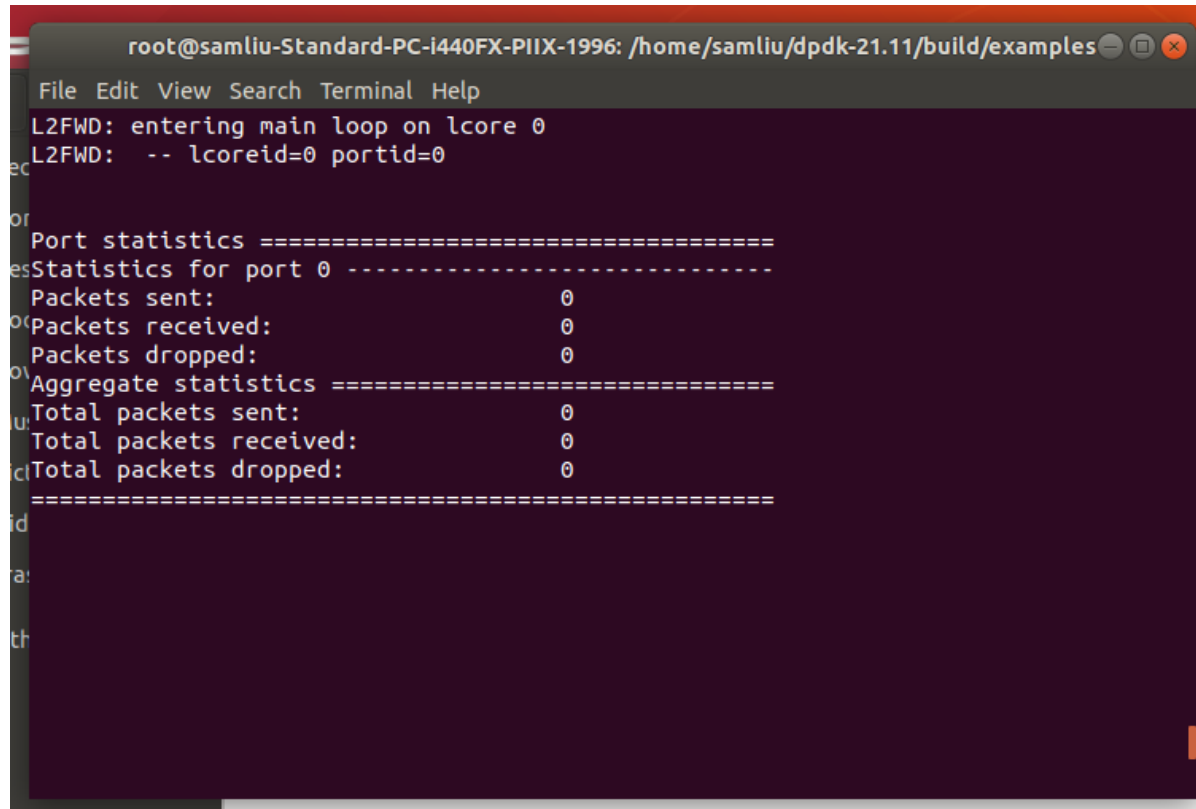
printf("Port %u, MAC address: " RTE_ETHER_ADDR_PRT_FMT "\n\n",
       portid,
       RTE_ETHER_ADDR_BYTES(&l2fwd_ports_eth_addr[portid]));

/* initialize port stats */
memset(&port_statistics, 0, sizeof(port_statistics));
}

if (!nb_ports_available) {
    rte_exit(EXIT_FAILURE,
              "All available ports are disabled. Please set portmask.\n");
}

check_all_ports_link_status(l2fwd_enabled_port_mask);
```

运行l2fwd后得到结果如下图所示

A terminal window titled 'root@samliu-Standard-PC-i440FX-PIIX-1996: /home/samliu/dpdk-21.11/build/examples'. The terminal shows the output of the 'l2fwd' command. It starts with 'L2FWD: entering main loop on lcore 0' and 'L2FWD: -- lcoreid=0 portid=0'. Then it displays 'Port statistics' and 'Statistics for port 0' with a table of values: Packets sent: 0, Packets received: 0, Packets dropped: 0. Below that, it shows 'Aggregate statistics' with a table: Total packets sent: 0, Total packets received: 0, Total packets dropped: 0. The terminal has a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'.

```
root@samliu-Standard-PC-i440FX-PIIX-1996: /home/samliu/dpdk-21.11/build/examples
File Edit View Search Terminal Help
L2FWD: entering main loop on lcore 0
L2FWD: -- lcoreid=0 portid=0

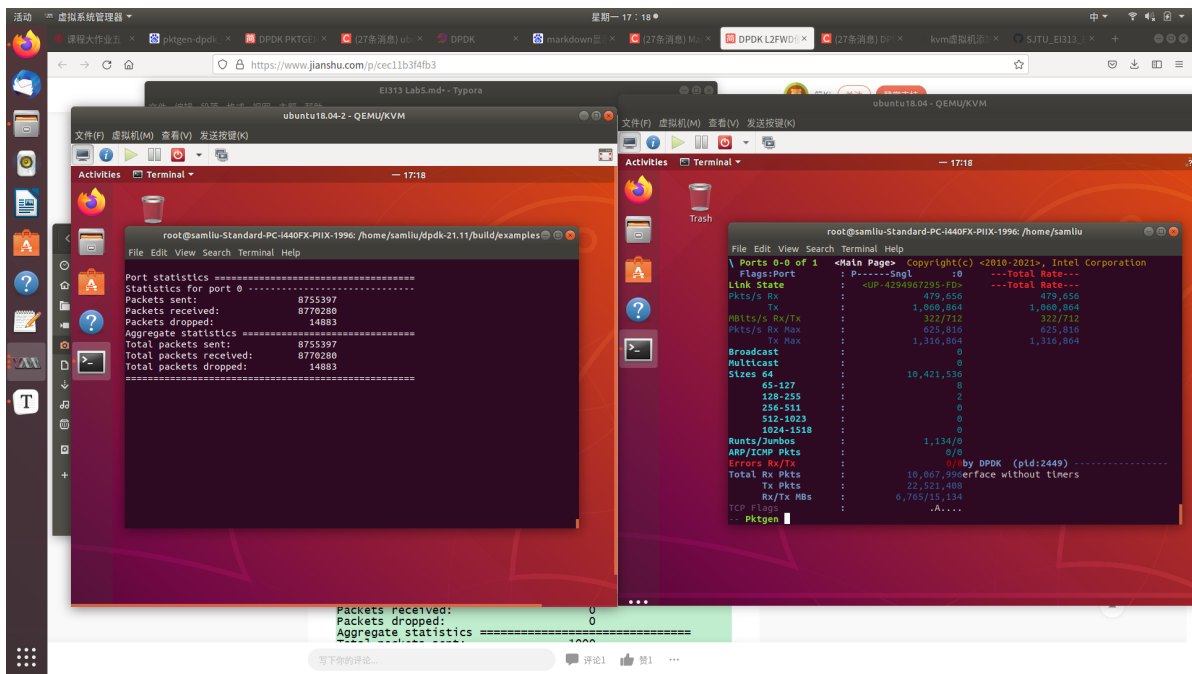
Port statistics =====
Statistics for port 0 -----
Packets sent:                0
Packets received:            0
Packets dropped:              0
Aggregate statistics =====
Total packets sent:          0
Total packets received:      0
Total packets dropped:        0
=====
```

### 4.3、运行pktgen-dpdk

```
首先进入pktgen-dpdk目录下
sudo su
pktgen -l 0-1 -n 2 -- -P -m "1.0" -T
“-P 启动混杂模式 -m表示cpu到端口的映射”
#进入pktgen CLI
set 0 dst mac 52:54:00:D0:30:E6 设置目标端口的MAC地址
set 0 size 64 设置每个packet的大小
start 0
```

注意此时要设置巨页否则会无法运行。

运行后可以看到两个虚拟机窗口的运行结果如下所示：

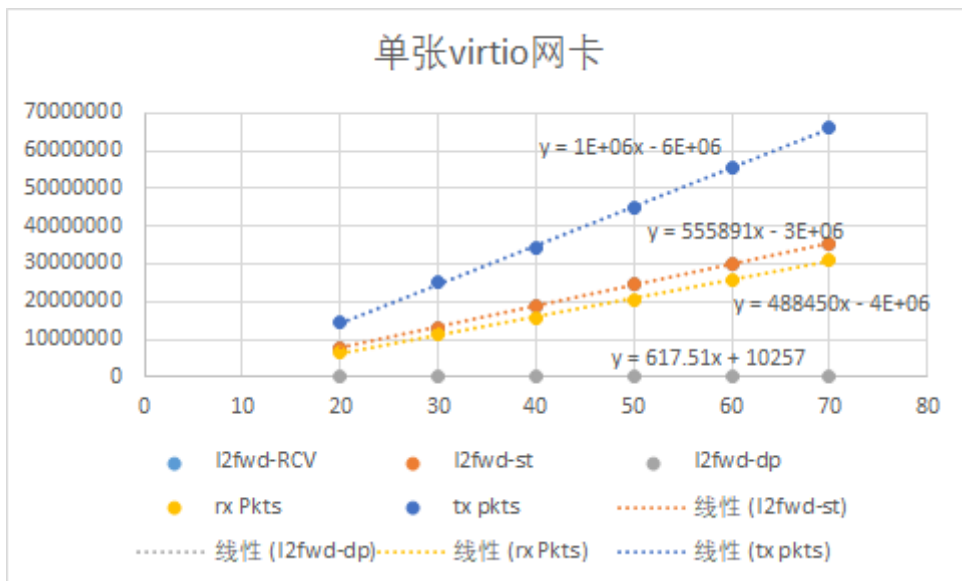


## 5、实验结果及分析

### 5.1、单张virtio网卡

在单virtio卡收发且一个packets大小为64KB时：

时间 (S)	I2fwd-RCV	I2fwd-st	I2fwd-dp	rx Pkts	tx pkts
20	7783040	7762310	20730	6393324	14536448
30	13064326	13035378	28948	11198196	25300352
40	18752873	18716408	36465	15701008	34475072
50	24432668	24389649	43019	20643516	44643584
60	30119905	30072175	47730	25815244	55260800
70	35369362	35317986	51376	30826100	66247808



图片的斜率表示收发速率。

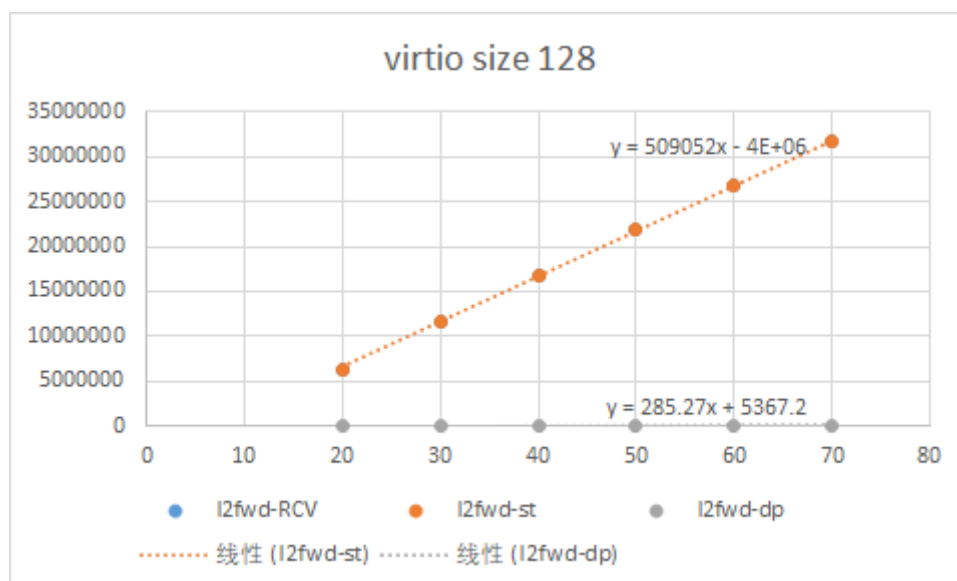
从数据中我们可以发现，l2fwd每秒能够接受约556000个数据包，其中丢失约620个数据包，其余被成功转发，丢包率约为0.1%。此处有丢包现象的发生，主要原因在于l2fwd处只有一个端口进行收发操作，因此存在自收发机制，导致丢包现象产生，因此考虑到使用e1000网卡，即可解决丢包的问题。

对比pktgen发送的包的速度和l2fwd所接收的包的速度，可以看出并不是所有发送的包都能够被接收到，丢包率为45%左右。究其原因可能是发送出去的数据包触发了局域网的拥塞控制或出现了传输差错而被丢弃。网卡在接受数据包时，需要维护一个缓冲区队列，即收发队列，当超过缓冲区的上限使，后续进入的包就会被丢弃，本实验使用2个收发队列，因此依旧存在丢包的可能。

而l2fwd转发回数据包的速率和pktgen接收的速率类似，因为造成拥塞的数据包已经被遗弃了，因此理论上来说再发送回来的数据包就不会再引发大量拥塞，因此二者速率相近。

将packets大小设置为128KB时，获得的实验结果如下：

时间 (S)	l2fwd-RCV	l2fwd-st	l2fwd-dp
20	6349991	6340577	9414
30	11661762	11646364	15398
40	16814303	16796489	17814
50	21933549	21913657	19892
60	26934272	26912749	21253
70	31816383	31790920	25454

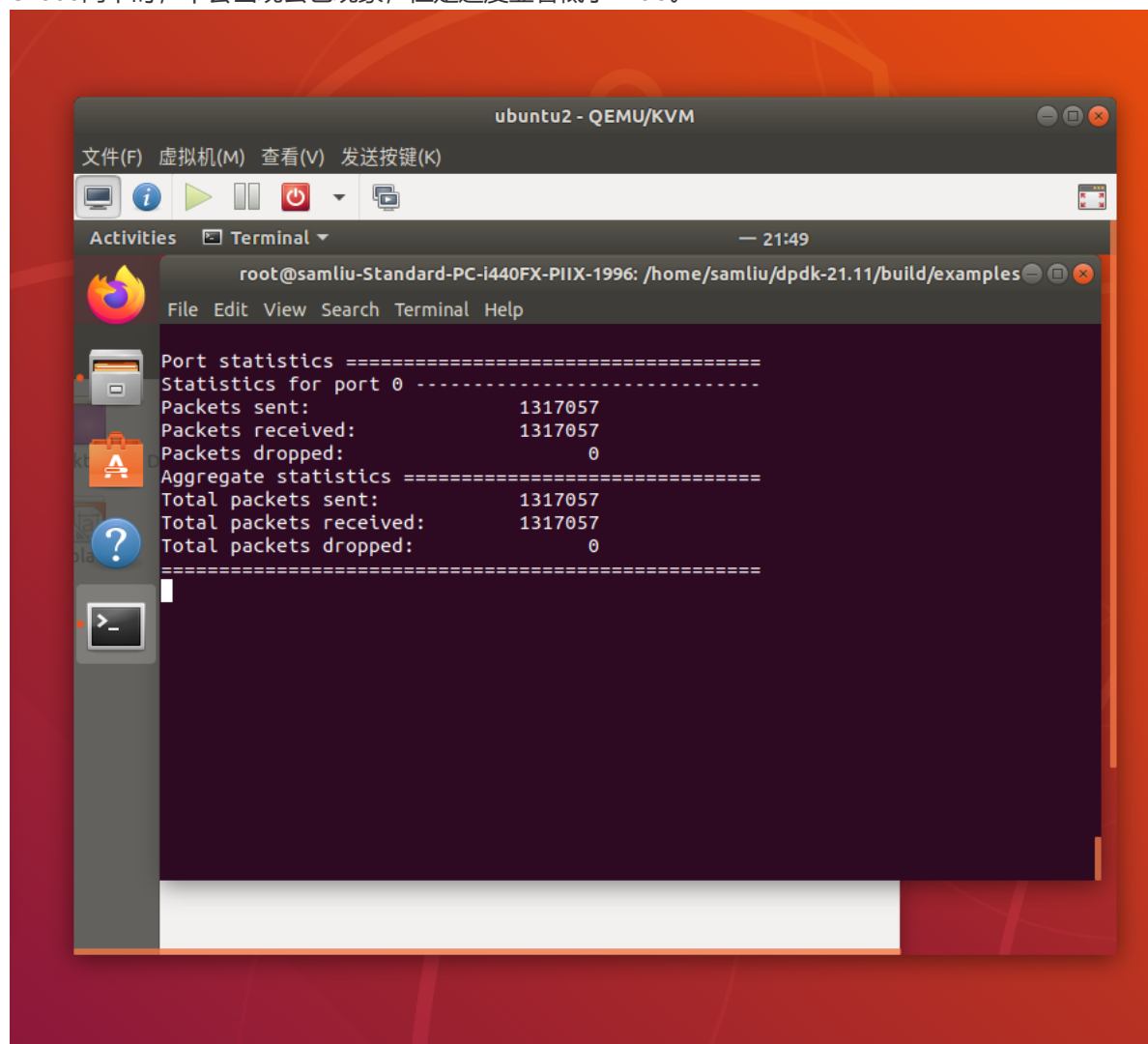


可以看出，此时丢包率相比起64KB时有所下降，通过计算斜率分析发现丢包率降为0.06%左右，原因可能是大的数据包不容易受到网络波动影响。

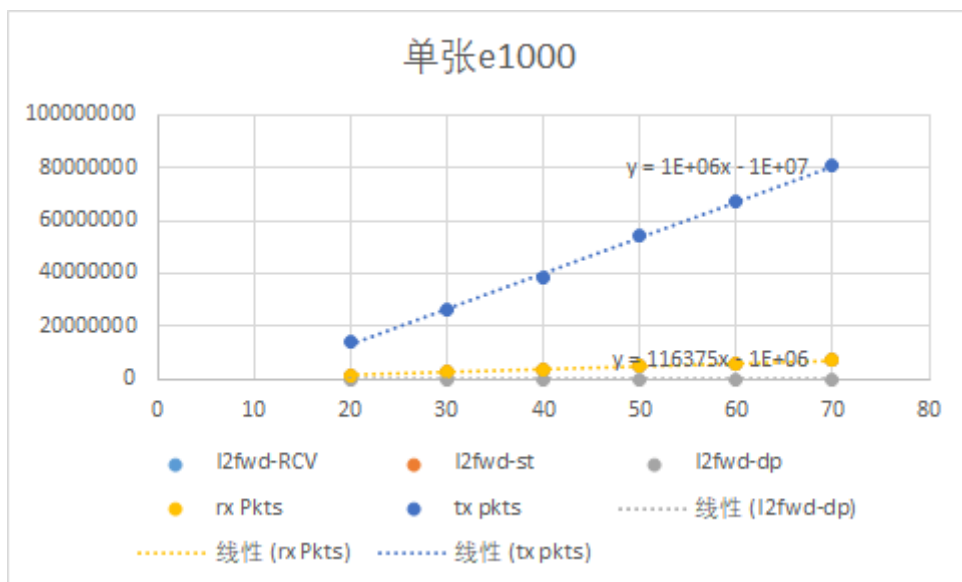
## 5.2、1张e1000网卡



使用e1000网卡的步骤与virtio是一致的，只需要将网卡的类别改一下即可，测试后发现，在使用1张e1000网卡时，不会出现丢包现象，但是速度显著低于virtio。



时间 (S)	l2fwd-RCV	l2fwd-st	l2fwd-dp	rx Pkts	tx pkts
20	1317057	1317057	0	1287376	13987942
30	2532034	2532034	0	2386771	26035490
40	3728258	3728258	0	3459903	38159749
50	4933730	4933730	0	4792417	54619821
60	6122690	6122690	0	5916549	67054913
70	7352258	7352258	0	7049231	80698494



从结果可以看出，相比单张virtio收发而言，一张e1000的I2fwd的收发速度要明显低于单virtio，仅为virtio的20%左右。因为virtio支持半虚拟化，将部分不需要虚拟化的指令直接移交给硬件处理，可以显著提升速度；而e1000是一个纯QEMU的网卡，没有实现半虚拟化，因此KVM对他来说就无效，无法提供硬件虚拟化加速。并且从VM1到VM2之间的丢包现象更加明显，究其原因是收发速度的变慢而导致缓冲队列迅速被填满，导致大量包在从VM1到VM2的过程中丢失。

### 5.3、2张e1000网卡两个端口

此时 I2fwd的命令要改为下式，-p的掩码改为0x3表示用两个口收发：

```
./dpdk-l2fwd -l 0-1 -n 2 -- -p 0x3 -q 2 -T 10
```

在运行pktgen时，需要再规定src的mac地址，即：

```
set 0 src mac 52:54:00:A3:7D:29
```

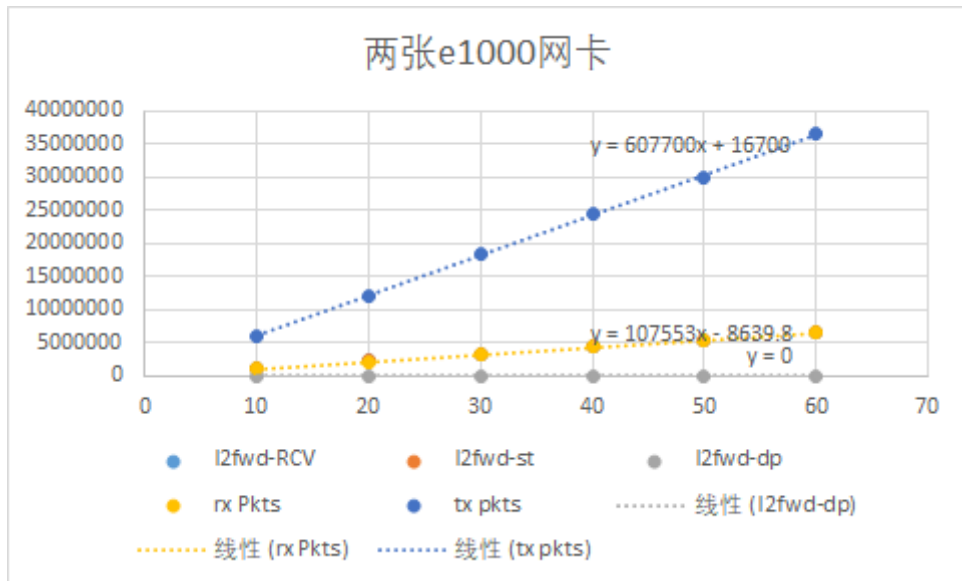
通过两个端口进行收发，就可以很好地解决I2fwd转发丢包的问题。

进入 examples/helloworld，进行如下配置：

```
export RTE_SDK=/home/Lianpeng/dpdk-19.11 (此处为DPDK的解压目录，改成自己的)
export RTE_TARGET=x86_64-native-linuxapp-gcc (改成自己的DPDK环境配置)
```

具体结果如下：

时间 (S)	l2fwd-RCV	l2fwd-st	l2fwd-dp	rx Pkts	tx pkts
10	1128258	1128258	0	1076771	6060992
20	2196607	2196607	0	2146456	12151936
30	3267920	3267920	0	3217380	18396544
40	4355742	4355742	0	4314179	24466112
50	5444980	5444980	0	5263719	29910912
60	6559398	6559398	0	6515752	36730688



从结果可以看出，双e1000的l2fwd收发速度略低于单张e1000，因为在两个端口间传输也需要耗费时间，且每次传输的报文头部变长了。但是从VM1到VM2的过程中，丢包率降低了。

## 5.4、实验结果总结

virtio网卡是一个基于qemu-kvm的网卡，支持半虚拟化，可以使用硬件来实现虚拟化的加速；而e1000网卡是一个完全基于qemu的网卡，不支持半虚拟化，无法使用硬件实现虚拟化的加速，因而理论上来说速度相比virtio要慢。

从实验结果来看：

- 单张virtio的优势在于l2fwd收发速度更快，且在pktgen从VM1到VM2发包的过程中丢包率更低；劣势在于l2fwd转发过程中存在丢包现象。
- 单张virtio下，当pktgen传输的包的大小变大时，l2fwd转发过程中的丢包率反而会变小。
- 单张e1000的优势在于l2fwd转发过程中不会丢包；劣势在于收发速度慢，且在pktgen从VM1到VM2的发包过程中丢包率更高。使用两张e1000在l2fwd端收发速率略低于单张e1000，但从VM1到VM2过程中的丢包率显著下降了。

## 6、总结

此次大作业核心在于二层转发，过程中我对计算机网络的知识有了更深层次的认识，在实验过程中发现了DPDK在不同条件下实现二层转发的优势和劣势，让我受益匪浅。感谢老师和助教一个学期以来的悉心指导，让我对qemu虚拟化技术和背后的网络有了深入的了解！

