lab2 DPDK

Q1: What's the purpose of using hugepage?

网络传输是I/O密集型的时使用的连续空间很大，如果用正常的4K页来翻译，会需要很多个PTE，产生多次缺页中断，而使用更大的页，可以减少PTE数量，减少缺页中断的次数，并减少地址翻译过程的级数。缺页中断对性能的影响。同时TLB的命中率会提升很大。

Q2: Take examples/helloworld as an example, describe the execution flow of DPDK programs?

1. `rte_eal_init(argc,argv)` 将程序的参数传入init函数，对Environment Abstraction Layer(EAL)进行初始化
2. 通过下述方式让所有slaves执行lcore_hello函数。

```
RTE_LCORE_FOREACH_SLAVE(lcore_id) {
  rte_eal_remote_launch(lcore_hello,NULL,lcore_id);
}
```

3. 父节点自己执行lcore_hello函数
4. 通过 `rte_eal_mp_wait_lcore()` 这个函数等待其他结点结束执行

2,3两步可以简化为如下

```
rte_eal_mp_remote_launch(lcore_hello, NULL, CALL_MASTER);
```

Q3: Read the codes of examples/skeleton, describe DPDK APIs related to sending and receiving packets.

1. `rte_pktmbuf_pool_create` 创建mempool 用于保存message buffer(rte_mbuf)
2. port initialization
   - `rte_eth_rx_queue_setup` 函数创建receive queue
   - `rte_eth_tx_queue_setup()` 函数创建transmit queue
   - `rte_eth_dev_start(port)` 函数开启以太网端口
   - `rte_eth_promiscuous_enable(port)` 函数以混杂模式启动receive
3. 发送和接收

- `rte_eth_rx_burst` 接受一堆数据包到buf
- `rte_eth_tx_burst` 发送一堆数据包到buf

Q4: Describe the data structure of `rte_mbuf`

用来缓存数据包，通过rte_pktmbuf_alloc函数分配，保存在rte_mempool中,。

`rte_eth_rx_burst`, `rte_eth_tx_burst` 都需要通过ret_mbuf结构来发送接收数据包。

利用rte_pktmbuf_prepend,rte_pktmbuf_append函数设置空间，并利用rte_pktmbuf_mtod_offset来获取指向特定offset的指针

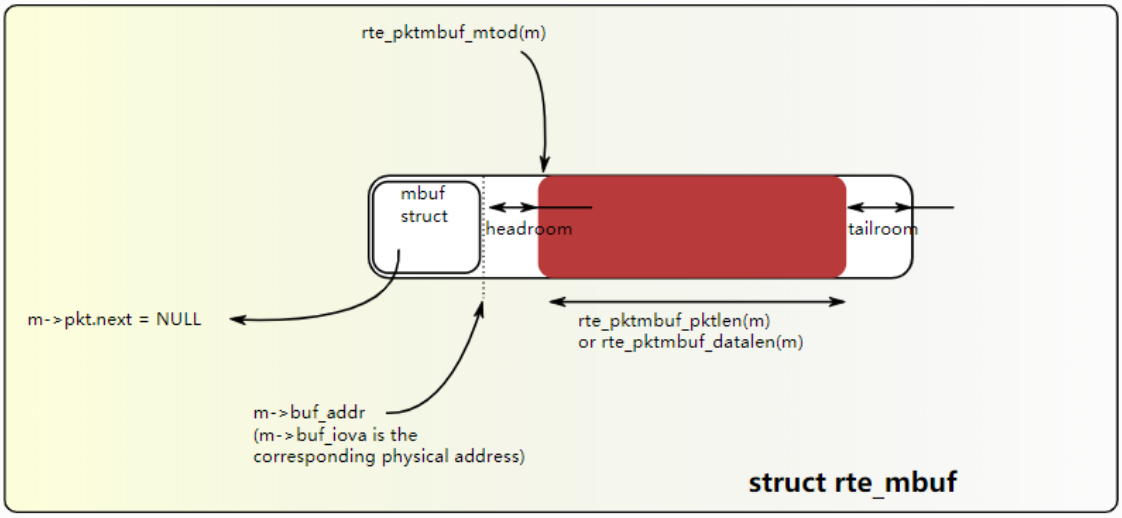


Fig. 9.1 An mbuf with One Segment

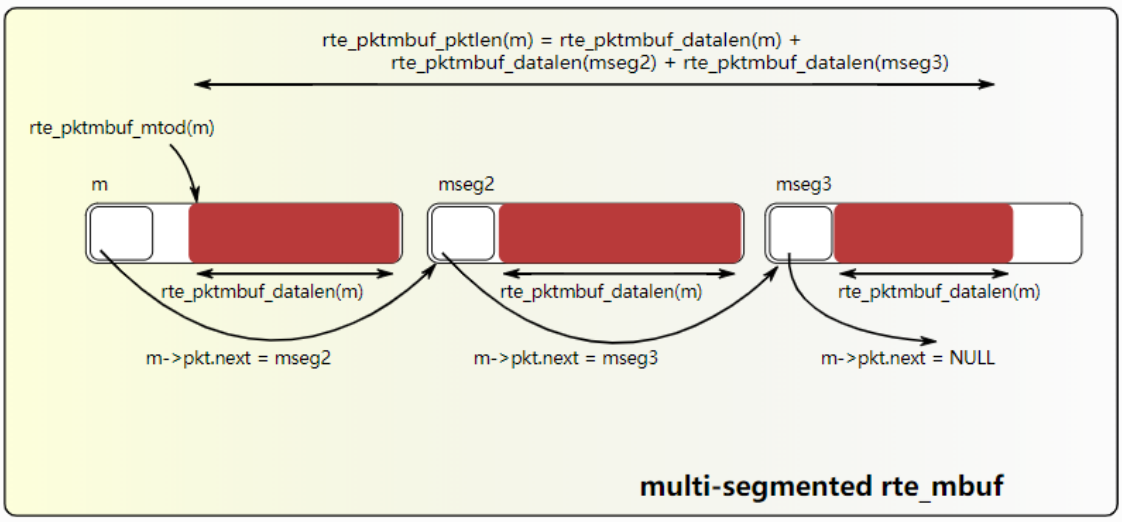
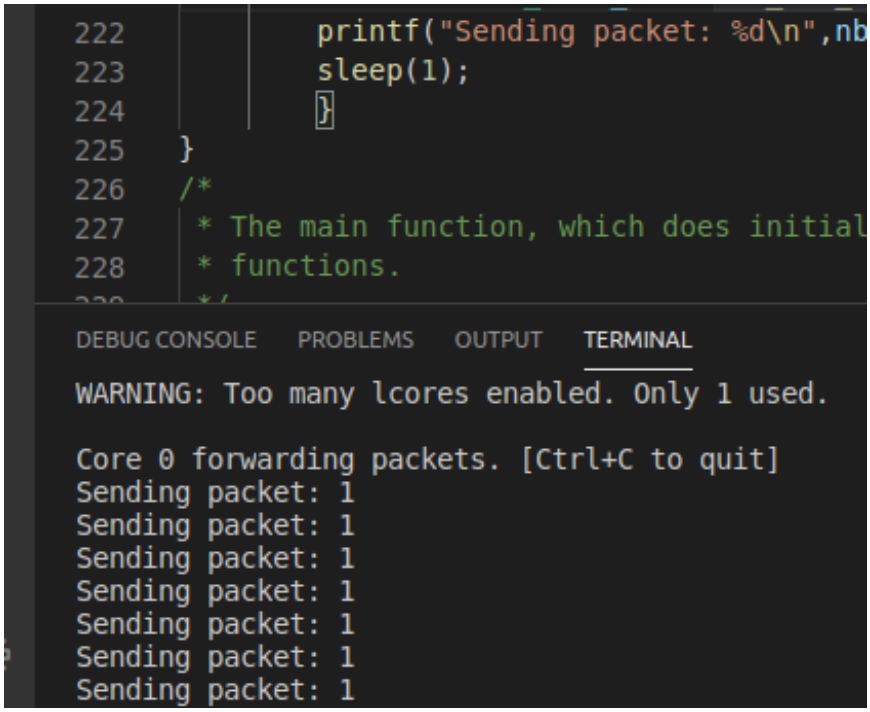


Fig. 9.2 An mbuf with Three Segments

Part2

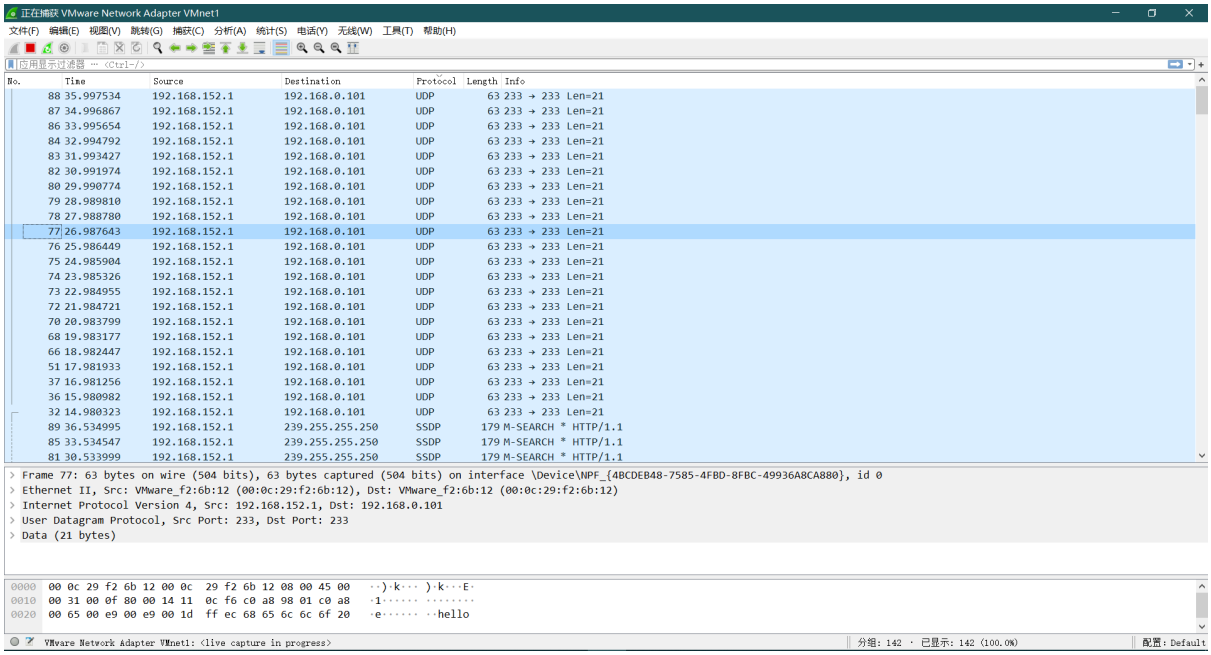
编写

- 根据ether_hdr, ipv4_hdr, udp_hdr(似乎最新版本的dpdk加上前缀rte_)的结构逐一将对应包的信息填入。利用rte_pktmbuf_mtod_offset，rte_pktmbuf_mtod函数获取mbuf不同位置的指针，并通过指针修改包的内容。填写时注意每一项都需要改成网络传输的big endian。注意各部分的length要计算正确。
- 在basicfwd的基础上，修改其lcore_main函数。通过rte_pktmbuf_prepend预先设定长度，并利用刚才写的函数构建udp包
- 利用rte_eth_tx_burst，循环发送刚刚构建好的udp包



识别

- 通过ifconfig找到网卡所对应的ip
- 根据虚拟机的配置找到网卡所对应的网络名：VMnet1
- 运行发包程序，并在host的wireshark上观察拦截到的包。
- 确认包的各部分信息完整正确。


正在捕获 VMware Network Adapter VMnet1
文件(F) 编辑(E) 视图(V) 跳转(G) 捕获(C) 分析(A) 统计(S) 电话(Y) 无线(W) 工具(T) 帮助(H)
应用显示过滤器 … <Ctrl-/>
No. Time Source Destination Protocol Length Info
88 35.997534 192.168.152.1 192.168.0.101 UDP 63 233 → 233 Len=21
87 34.996867 192.168.152.1 192.168.0.101 UDP 63 233 → 233 Len=21
86 33.995654 192.168.152.1 192.168.0.101 UDP 63 233 → 233 Len=21
84 32.994792 192.168.152.1 192.168.0.101 UDP 63 233 → 233 Len=21
83 31.993427 192.168.152.1 192.168.0.101 UDP 63 233 → 233 Len=21
82 30.991974 192.168.152.1 192.168.0.101 UDP 63 233 → 233 Len=21
80 29.990774 192.168.152.1 192.168.0.101 UDP 63 233 → 233 Len=21
79 28.989810 192.168.152.1 192.168.0.101 UDP 63 233 → 233 Len=21
78 27.988780 192.168.152.1 192.168.0.101 UDP 63 233 → 233 Len=21
77 26.987643 192.168.152.1 192.168.0.101 UDP 63 233 → 233 Len=21
76 25.986449 192.168.152.1 192.168.0.101 UDP 63 233 → 233 Len=21
75 24.985904 192.168.152.1 192.168.0.101 UDP 63 233 → 233 Len=21
74 23.985326 192.168.152.1 192.168.0.101 UDP 63 233 → 233 Len=21
73 22.984955 192.168.152.1 192.168.0.101 UDP 63 233 → 233 Len=21
72 21.984721 192.168.152.1 192.168.0.101 UDP 63 233 → 233 Len=21
70 20.983799 192.168.152.1 192.168.0.101 UDP 63 233 → 233 Len=21
68 19.983177 192.168.152.1 192.168.0.101 UDP 63 233 → 233 Len=21
66 18.982447 192.168.152.1 192.168.0.101 UDP 63 233 → 233 Len=21
51 17.981933 192.168.152.1 192.168.0.101 UDP 63 233 → 233 Len=21
37 16.981256 192.168.152.1 192.168.0.101 UDP 63 233 → 233 Len=21
36 15.980982 192.168.152.1 192.168.0.101 UDP 63 233 → 233 Len=21
32 14.980323 192.168.152.1 192.168.0.101 UDP 63 233 → 233 Len=21
89 36.534995 192.168.152.1 239.255.255.250 SSDP 179 M-SEARCH * HTTP/1.1
85 33.534547 192.168.152.1 239.255.255.250 SSDP 179 M-SEARCH * HTTP/1.1
81 30.533999 192.168.152.1 239.255.255.250 SSDP 179 M-SEARCH * HTTP/1.1
Frame 77: 63 bytes on wire (504 bits), 63 bytes captured (504 bits) on interface \Device\NPF_{4BCDEB48-7585-4FBD-8FBC-49936A8CA880}, id 0
Ethernet II, Src: VMware_f2:6b:12 (00:0c:29:f2:6b:12), Dst: VMware_f2:6b:12 (00:0c:29:f2:6b:12)
Internet Protocol Version 4, Src: 192.168.152.1, Dst: 192.168.0.101
User Datagram Protocol, Src Port: 233, Dst Port: 233
Data (21 bytes)
0000 00 0c 29 f2 6b 12 00 0c 29 f2 6b 12 08 00 45 00
0010 00 31 00 0f 80 00 14 11 0c f6 c0 a8 98 01 c0 a8
0020 00 65 00 e9 00 e9 00 1d ff ec 68 65 6c 6c 6f 20 e hello
VMware Network Adapter VMnet1: <live capture in progress>
分组: 142 · 已显示: 142 (100.0%)
配置: Default