

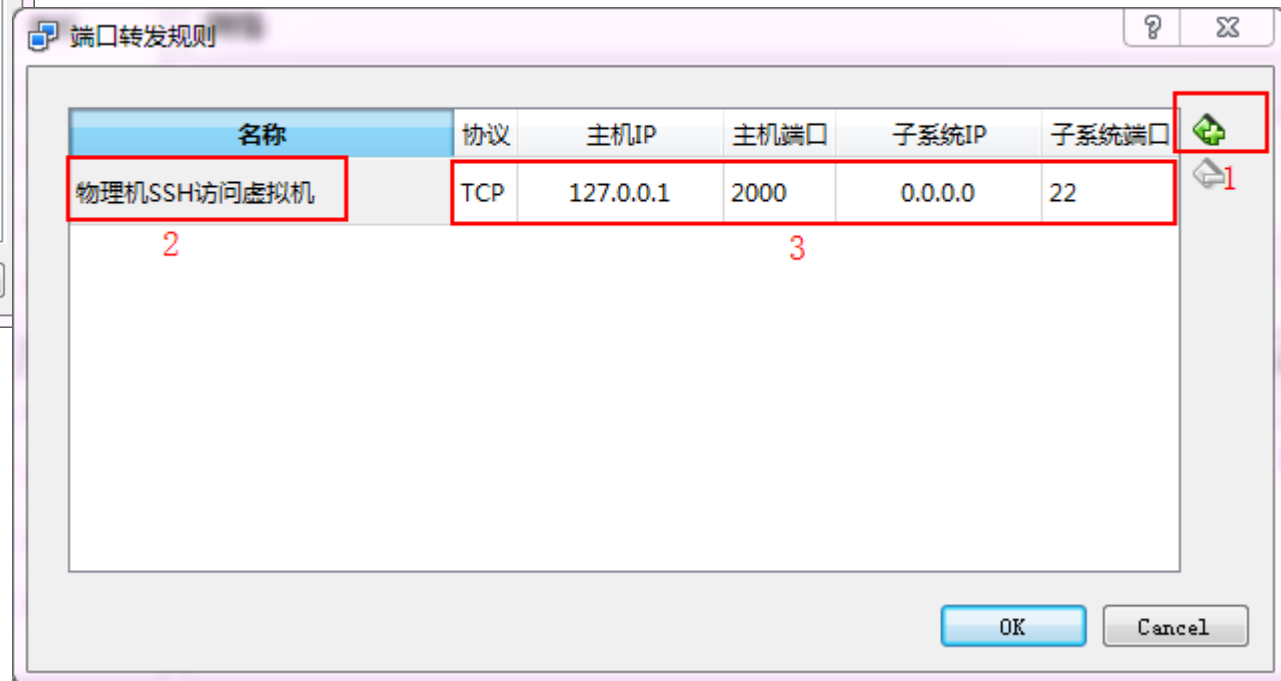
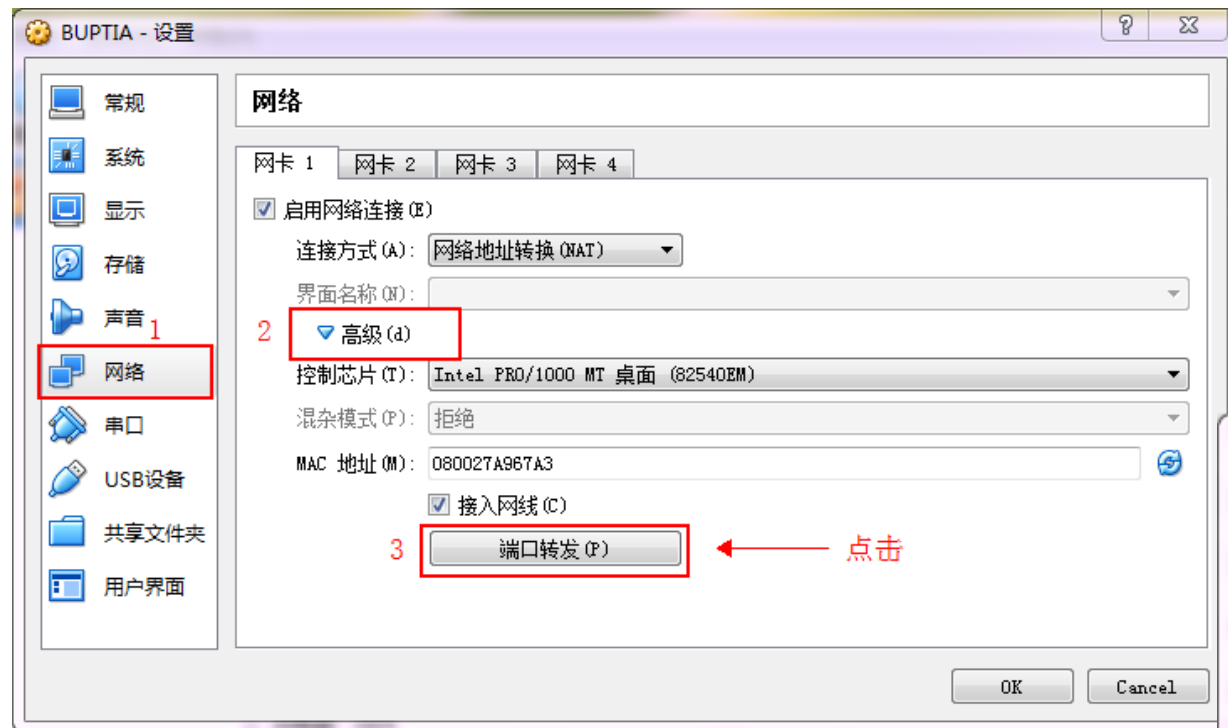
DHCP Project Preparation

2017.05

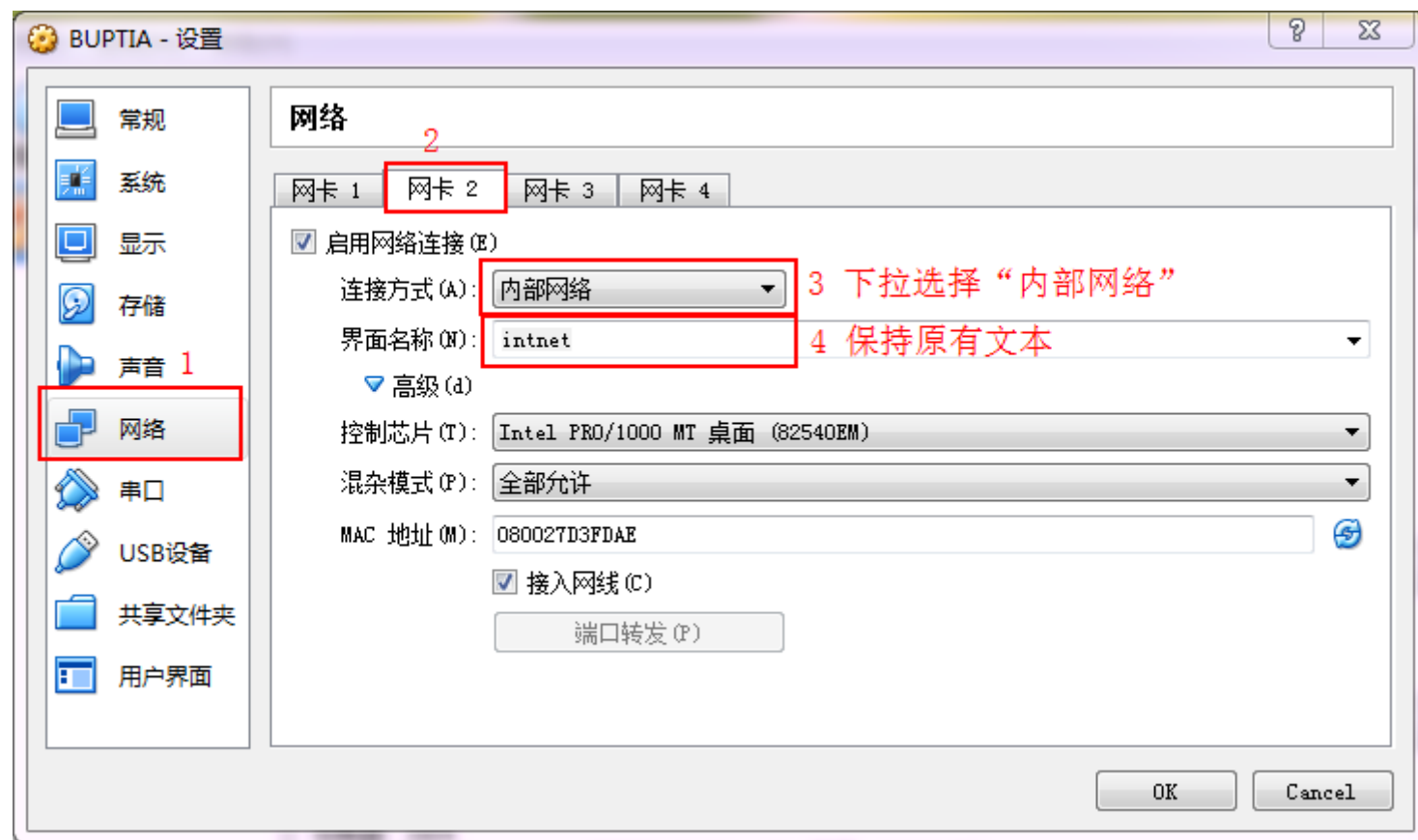
1. DHCP Client的虚拟机



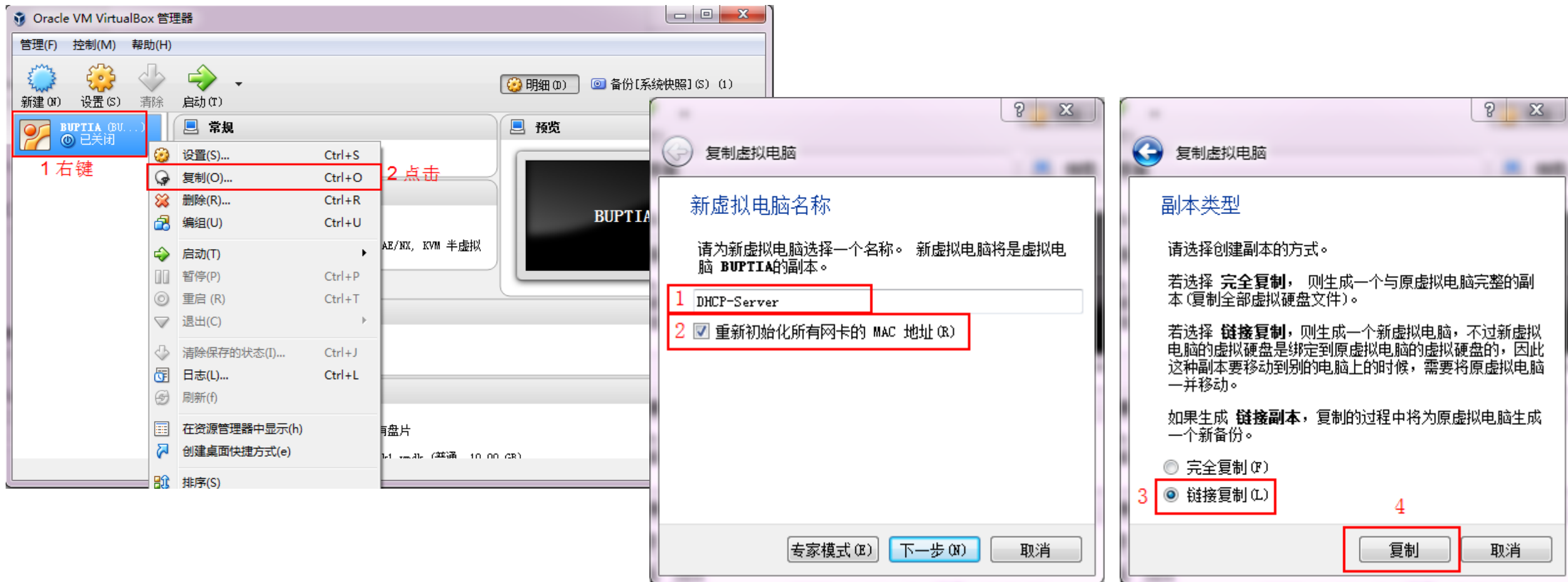
1.1 设置“网卡1”



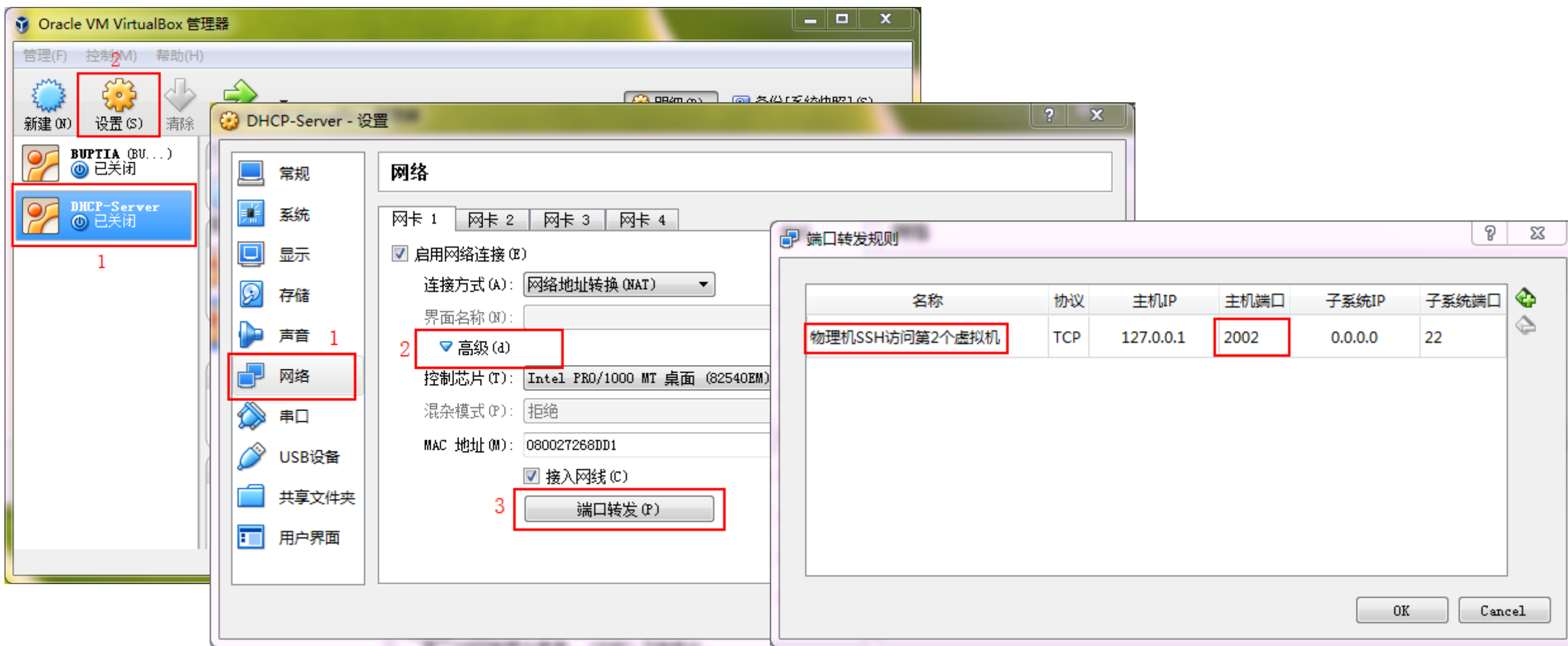
1.2 设置“网卡2”



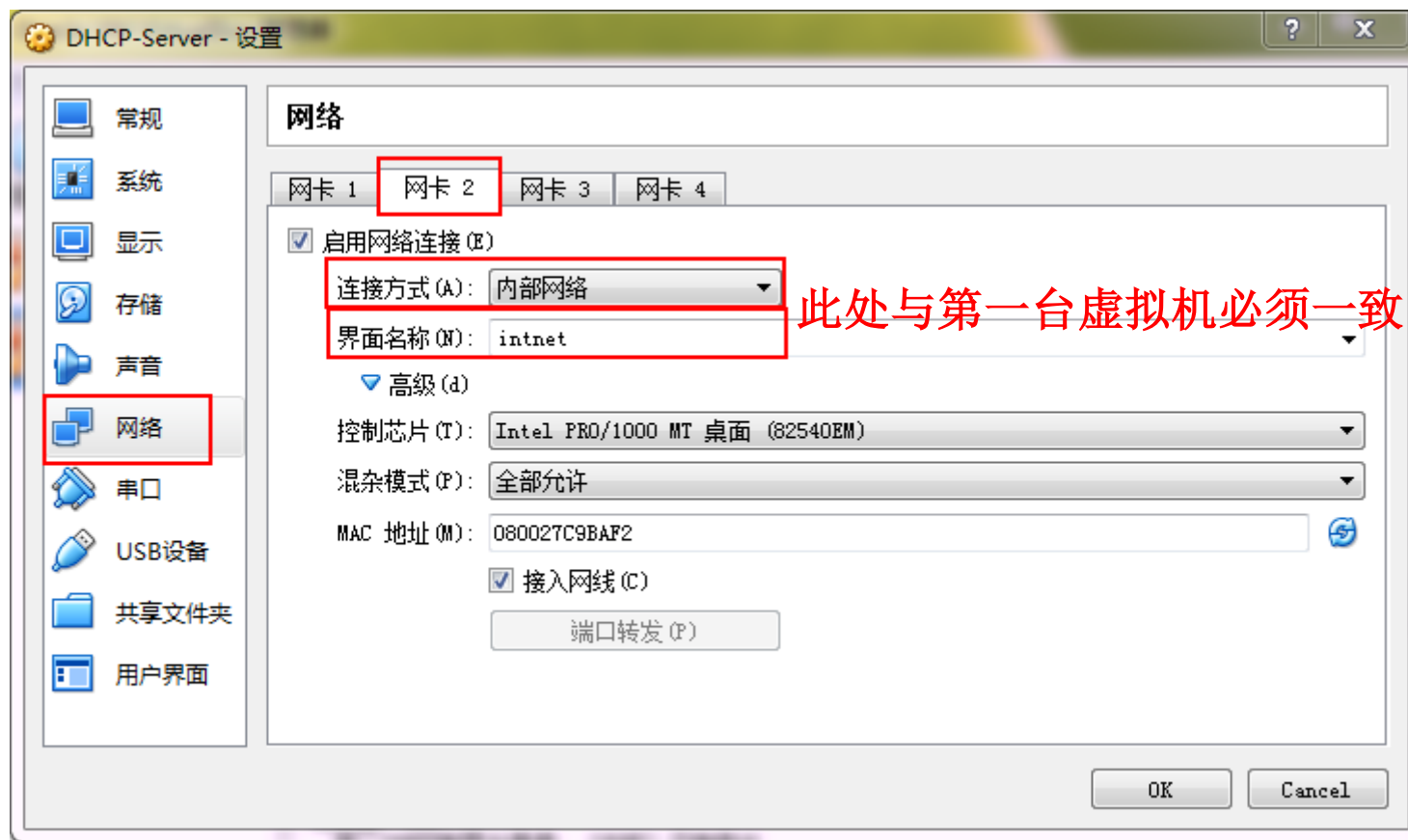
2. DHCP server的虚拟机



2.1 设置“网卡1”



2.2 检查“网卡2”



3. 修改网络配置

- 启动两个虚拟机，按照以下方式修改网络配置，然后保存配置，重启虚拟机

- Windows

- 打开Xshell，输入 ssh [student@127.0.0.1](#) 2000，进入dhcp client虚拟机
 - sudo vim /etc/network/interfaces，修改eth1的配置为
- 新增Xshell选项卡，输入 ssh [student@127.0.0.1](#) 2002，进入dhcp server虚拟机
 - sudo vim /etc/network/interfaces，修改eth1的配置为

```
auto eth1
iface eth1 inet static
address 0.0.0.0
netmask 0.0.0.0
```

- Mac

- 打开Terminal，输入ssh [student@127.0.0.1](#) -p 2000，进入dhcp client虚拟机
 - sudo vim /etc/network/interfaces，修改eth1的配置为
- 新增Terminal，输入ssh [student@127.0.0.1](#) -p 2002，进入dhcp server虚拟机
 - sudo vim /etc/network/interfaces，修改eth1的配置为

```
auto eth1
iface eth1 inet static
address 192.168.0.1
netmask 255.255.255.0
```


4. 增加iptables规则

- 在Linux系统中，当数据包的目标地址为255.255.255.255广播地址，源地址必须设置为0.0.0.0，数据包才能被处理
- 由于普通的socket函数不能设置源地址为0.0.0.0，所以需要利用iptables工具对广播数据包修改源地址
- 请分别在两个虚拟机的终端模拟器内执行：
 - `sudo iptables -t nat -A POSTROUTING -d 255.255.255.255 -o eth1 -j SNAT --to-source 0.0.0.0`
 - `sudo iptables-save > /etc/network/iptables.rules`
 - `sudo vim /etc/rc.local`, 修改为以下2句
iptables-restore < /etc/network/iptables.rules
exit 0

```
sudo sh -c "iptables-save > /etc/network/iptables.rules"
```

5. Test

- In xShell or Terminal of DHCP-Server VM
 - `sudo wireshark`
- In xShell or Terminal of DHCP-Client VM
 - `sudo dhclient eth1`
- DHCP DISCOVER should display in Wireshark:

Capturing from eth1 [Wireshark 1.10.6 (v1.10.6 from master-1.10)]

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help

Filter: Expression... Clear Apply Save

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	0.0.0.0	255.255.255.255	DHCP	342	DHCP Discover - Transaction ID 0xba3a8117
2	2.914356000	0.0.0.0	255.255.255.255	DHCP	342	DHCP Discover - Transaction ID 0xba3a8117
3	10.994629000	0.0.0.0	255.255.255.255	DHCP	342	DHCP Discover - Transaction ID 0xba3a8117
4	23.949323000	0.0.0.0	255.255.255.255	DHCP	342	DHCP Discover - Transaction ID 0xba3a8117
5	41.294355000	0.0.0.0	255.255.255.255	DHCP	342	DHCP Discover - Transaction ID 0xba3a8117
6	50.441325000	0.0.0.0	255.255.255.255	DHCP	342	DHCP Discover - Transaction ID 0xba3a8117
7	65.741077000	0.0.0.0	255.255.255.255	DHCP	342	DHCP Discover - Transaction ID 0xba3a8117
8	84.319062000	0.0.0.0	255.255.255.255	DHCP	342	DHCP Discover - Transaction ID 0xba3a8117
9	01.600118000	0.0.0.0	255.255.255.255	DHCP	342	DHCP Discover - Transaction ID 0xba3a8117

Environment of
Network is OK!

6. Hints about broadcast

- Set socket: (#include <sys/socket.h>)

```
int setsockopt(SOCKET s, int level, int optname, const char* optval, int optlen);
```

0:success
-1:error

s(套接字): 指向
一个打开的套接
口描述字

level:(级别):
指定选项代码
的类型, 包含
SOL_SOCKET和
IPPROTO_TCP
(这里选用基
本套接口
SOL_SOCKET)

optname(选项
名): 选项名称。
指明要设置的
选项, 包括
SO_BROADCAST
允许发送广播
数据等

optval(选项值):
是一个指向变
量的指针类型,
指向存放选项
值的缓冲区

optlen(选项长度):
optval 的大小

- eg: allow socket to broadcast

```
setsockopt(sock,SOL_SOCKET,SO_BROADCAST,&i,len);
```

- 网络接口信息结构体struct ifreq (#include <net/if.h>)

```
struct ifreq {
    char ifr_name[IFNAMSIZ]; /* Interface name */
    union {
        struct sockaddr ifr_addr; /* address */
        struct sockaddr ifr_dstaddr; /* other end of p-p lnk */
        struct sockaddr ifr_broadaddr; /* broadcast address */
        struct sockaddr ifr_netmask; /* interface net mask */
        struct sockaddr ifr_hwaddr; /* MAC address */
        short ifr_flags; /* flags */
        int ifr_ifindex;
        int ifr_metric; /* metric */
        int ifr_mtu; /* mtu */
        struct ifmap ifr_map; /* device map */
        char ifr_slave[IFNAMSIZ]; /* slave device */
        char ifr_newname[IFNAMSIZ]; /* New name */
        char *ifr_data; /* for use by interface */
    };
};
```

- eg: declare an ifreq struct to store eth1 interface information

```
struct ifreq if_eth1;
strcpy(if_eth1.ifr_name, "eth1");
```

- Example: Allow a socket to broadcast and bind the socket to interface eth1

```
#include <sys/socket.h> /* for setsockopt() */
#include <net/if.h> /* for ifreq */
int i=1;
struct ifreq if_eth1;
strcpy(if_eth1.ifr_name, "eth1");
socklen_t len = sizeof(i);
/* Allow socket to broadcast */
setsockopt(sock, SOL_SOCKET, SO_BROADCAST, &i, len);
/* Set socket to interface eth1 */
if (setsockopt(sock, SOL_SOCKET, SO_BINDTODEVICE, (char *)&if_eth1, sizeof(if_eth1)) < 0) {
    printf("bind socket to eth1 error\n");
}
```

- Set address:
set local address to 0.0.0.0 and broadcast address to 255.255.255.255
- Example:

Server:

Client:

```
/*Zero out structure*/
memset(&servAddr, 0, sizeof(servAddr));
/* Internet addr family */
servAddr.sin_family = AF_INET;
/* Server port */
servAddr.sin_port = htons(serverPort);
/*Server IP address 0.0.0.0*/
servAddr.sin_addr.s_addr = htonl(INADDR_ANY);
if ((bind(sock, (struct sockaddr *)&servAddr, sizeof(servAddr))) < 0){
    printf("bind() failed.\n");
}
/*Client IP address 255.255.255.255*/
clntAddr.sin_addr.s_addr = inet_addr(broadcastIP);
```

```
/*Zero out structure*/
memset(&clntAddr, 0, sizeof(clntAddr));
/* Internet addr family */
clntAddr.sin_family = AF_INET;
/* Client port */
clntAddr.sin_port = htons(clientPort);
/*Client IP address 0.0.0.0*/
clntAddr.sin_addr.s_addr = htonl(INADDR_ANY);
if ((bind(sock, (struct sockaddr *)&clntAddr, sizeof(clntAddr))) < 0){
    printf("bind() failed.\n");
}
/*Server IP address 255.255.255.255*/
servAddr.sin_addr.s_addr = inet_addr(broadcastIP);
```

End