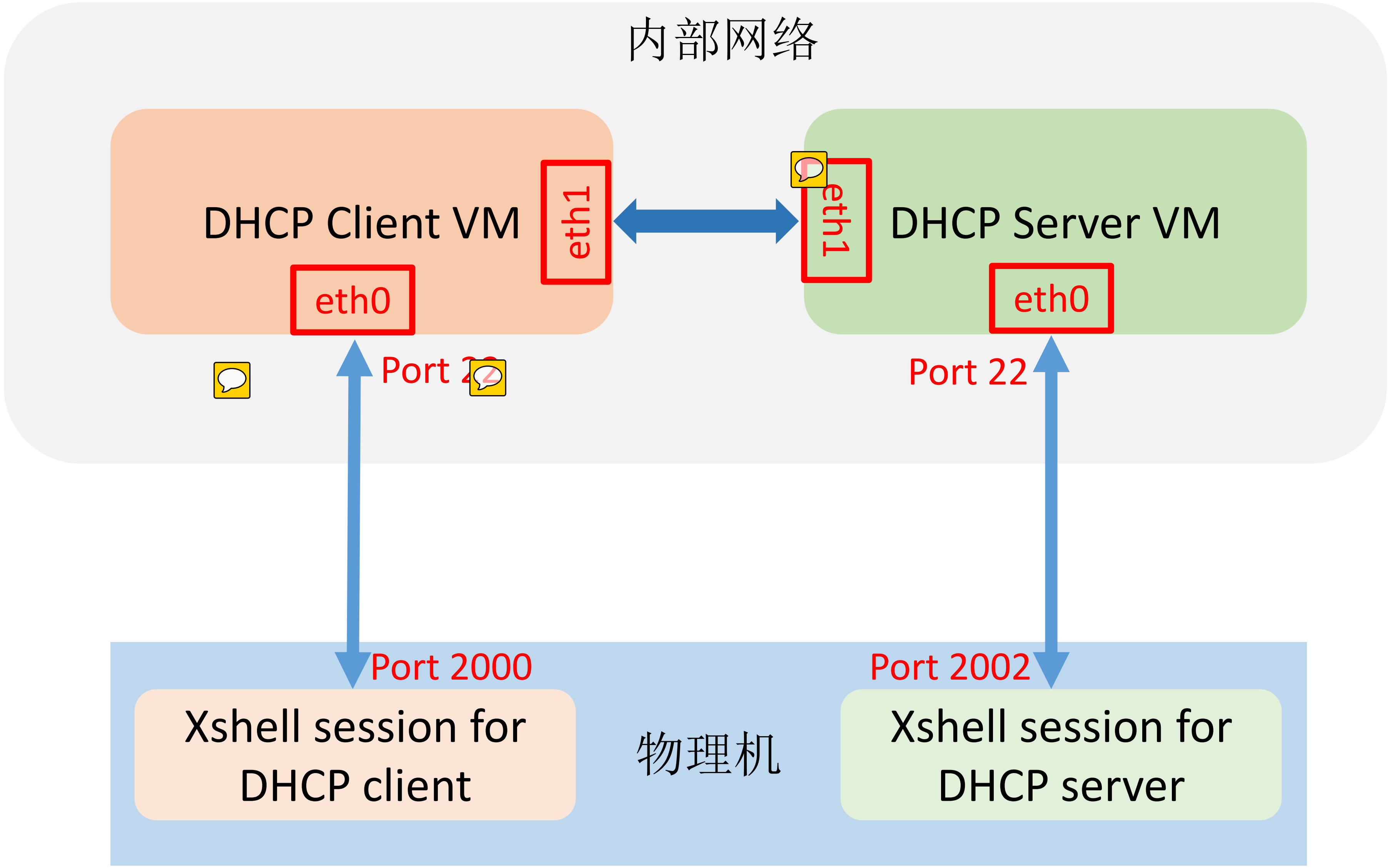
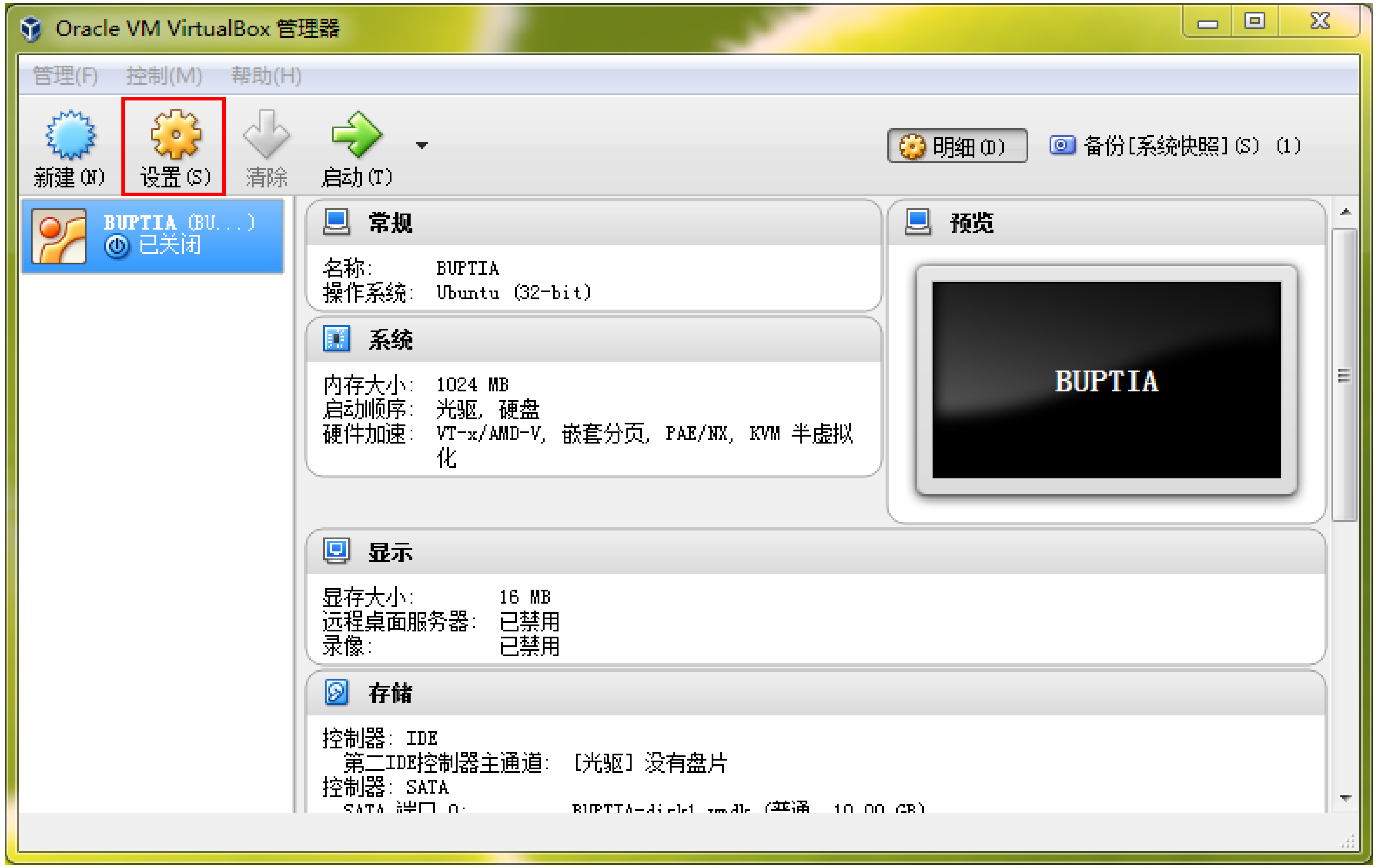
DHCP Project Preparation

2017.05

Part 1: Developing Environment Configuration


内部网络
DHCP Client VM
eth1
eth0
Port 22
DHCP Server VM
eth1
eth0
Port 22
Port 2000
Xshell session for DHCP client
物理机
Port 2002
Xshell session for DHCP server

1. DHCP Client的虚拟机



常规

名称: BUPTIA
操作系统: Ubuntu (32-bit)

系统

内存大小: 1024 MB
启动顺序: 光驱, 硬盘
硬件加速: VT-x/AMD-V, 嵌套分页, PAE/NX, KVM 半虚拟化

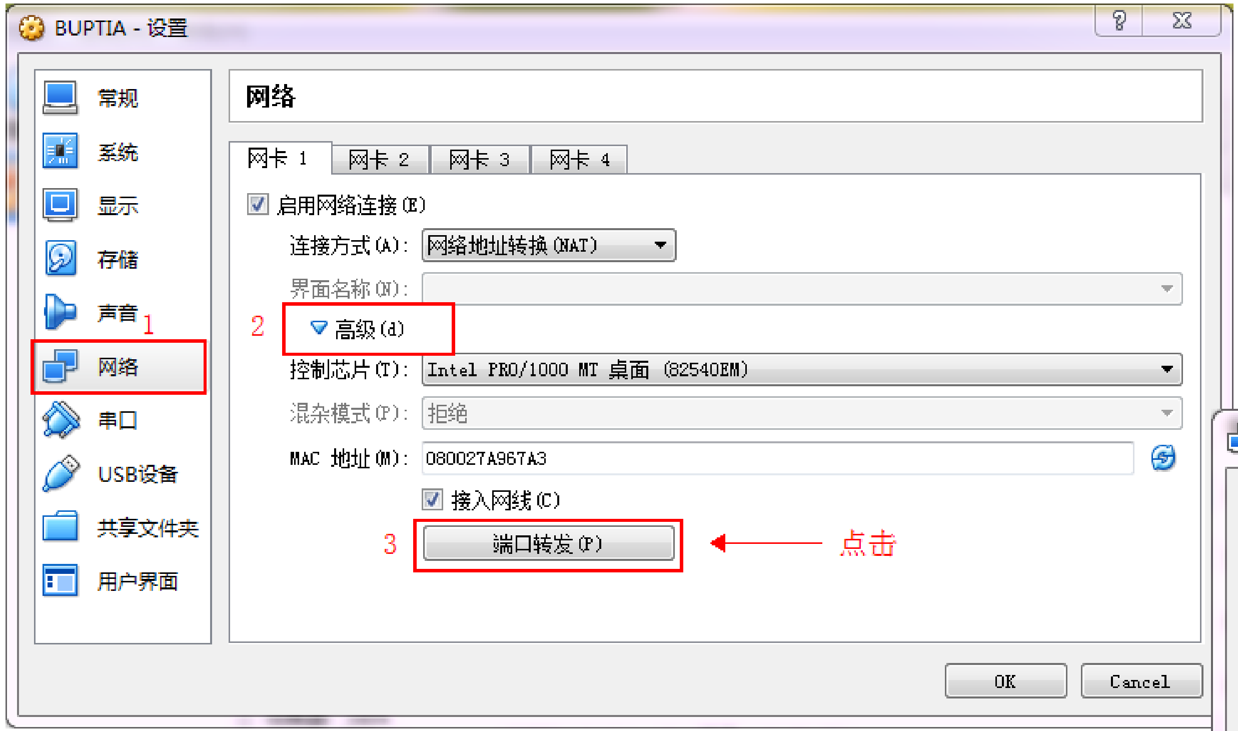
显示

显存大小: 16 MB
远程桌面服务器: 已禁用
录像: 已禁用

存储

控制器: IDE
第二IDE控制器主通道: [光驱] 没有盘片
控制器: SATA

1.1 设置“网卡1”


BUPTIA - 设置

常规
系统
显示
存储
声音
网络
串口
USB设备
共享文件夹
用户界面

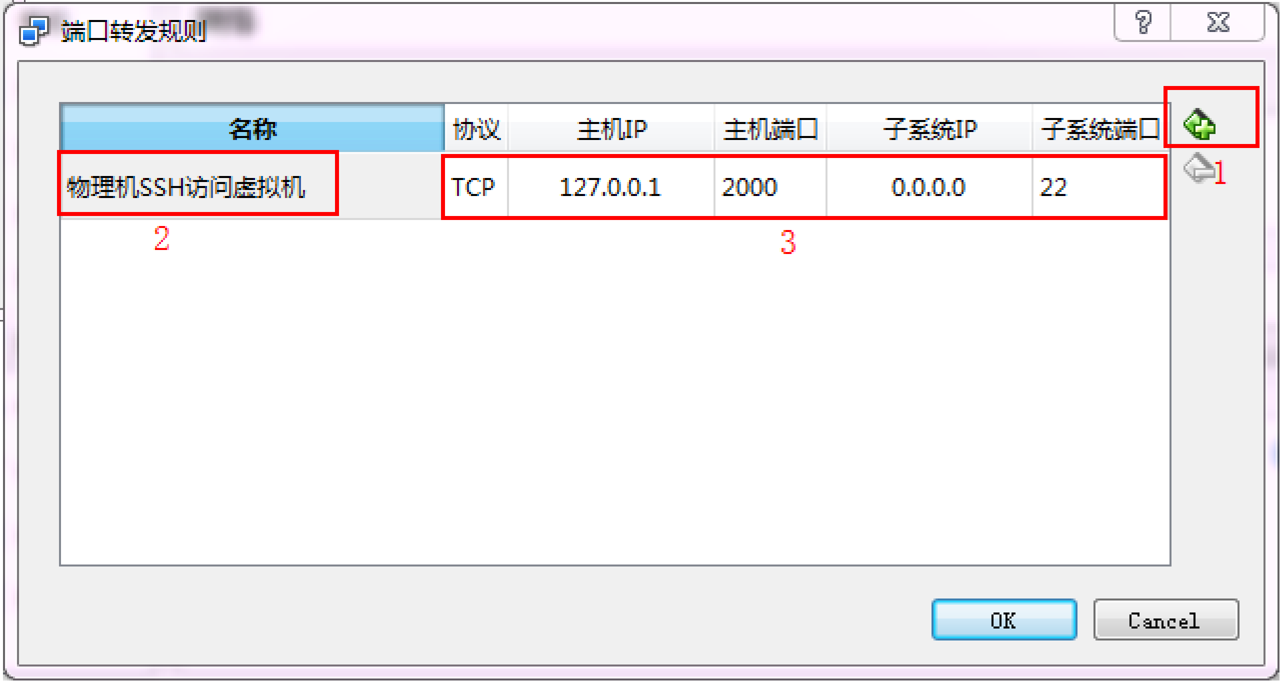
网络

网卡 1 | 网卡 2 | 网卡 3 | 网卡 4

☑ 启用网络连接(E)
连接方式(A): 网络地址转换(NAT)
界面名称(N):
▽ 高级(d)
控制芯片(T): Intel PRO/1000 MT 桌面 (82540EM)
混杂模式(P): 拒绝
MAC 地址(M): 080027A967A3
☑ 接入网线(C)
端口转发(P)

OK Cancel

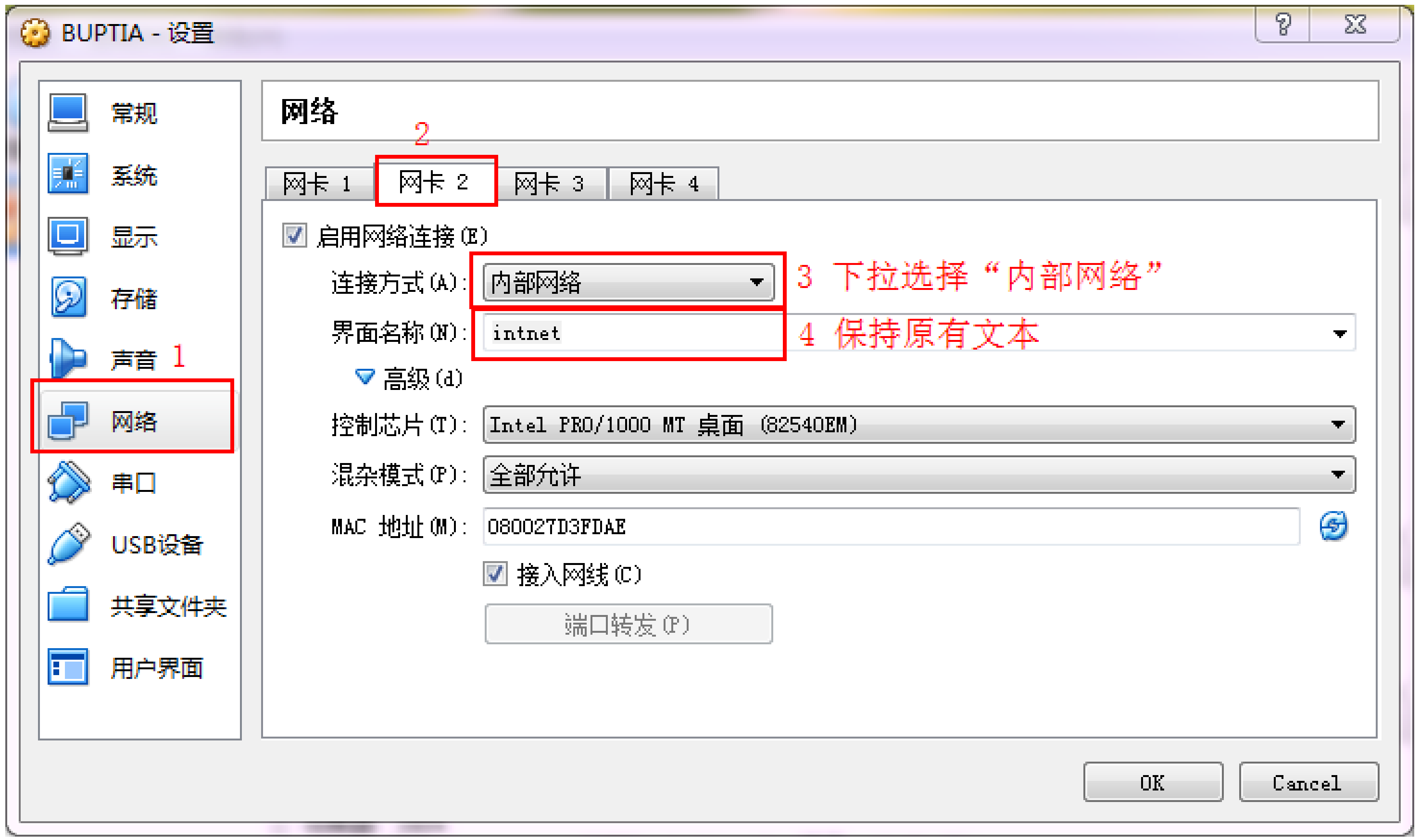
从主机的2000端口连虚拟机的22号端口


端口转发规则

名称	协议	主机IP	主机端口	子系统IP	子系统端口
物理机SSH访问虚拟机	TCP	127.0.0.1	2000	0.0.0.0	22

OK Cancel

1.2设置“网卡2”


BUPTIA - 设置

常规
系统
显示
存储
声音 1
网络
串口
USB设备
共享文件夹
用户界面

网络

2

网卡 1 | 网卡 2 | 网卡 3 | 网卡 4

☑ 启用网络连接(E)

连接方式(A): 内部网络 — 3 下拉选择“内部网络”

界面名称(N): intnet — 4 保持原有文本

▽ 高级(d)

控制芯片(T): Intel PRO/1000 MT 桌面 (82540EM)

混杂模式(P): 全部允许

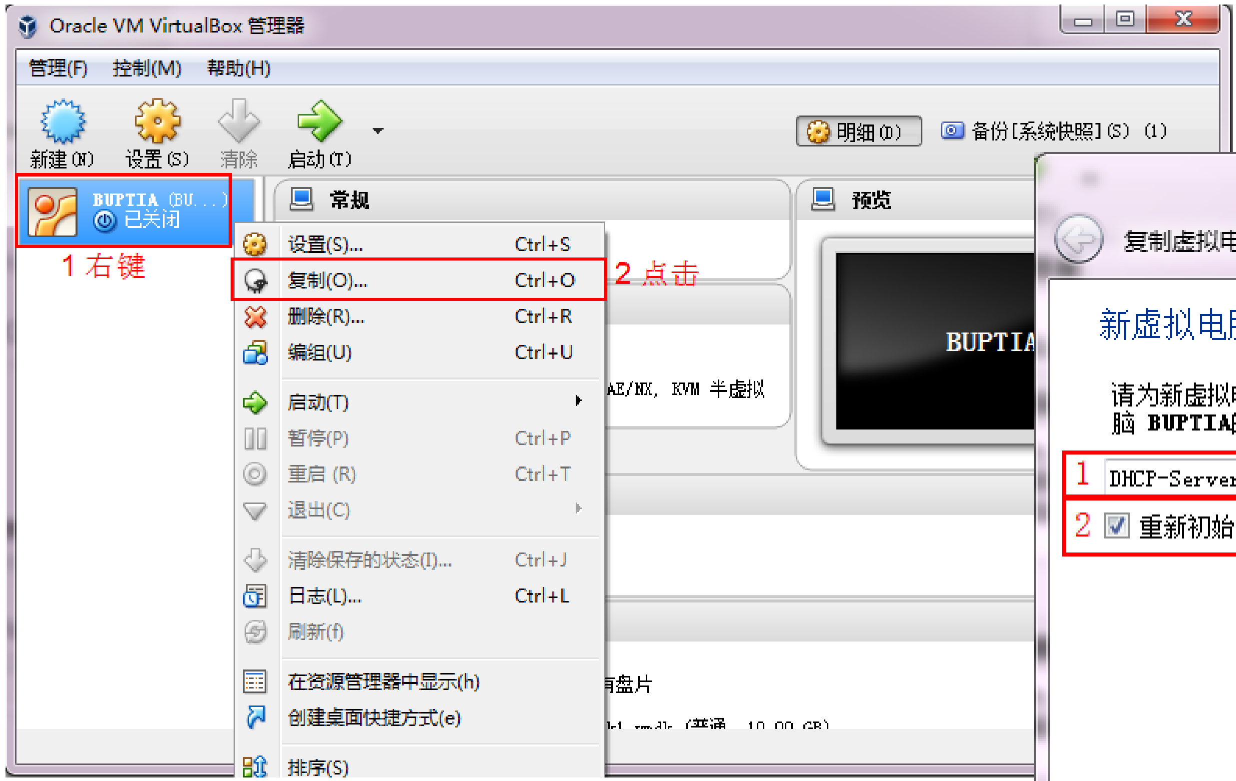
MAC 地址(M): 080027D3FDAE

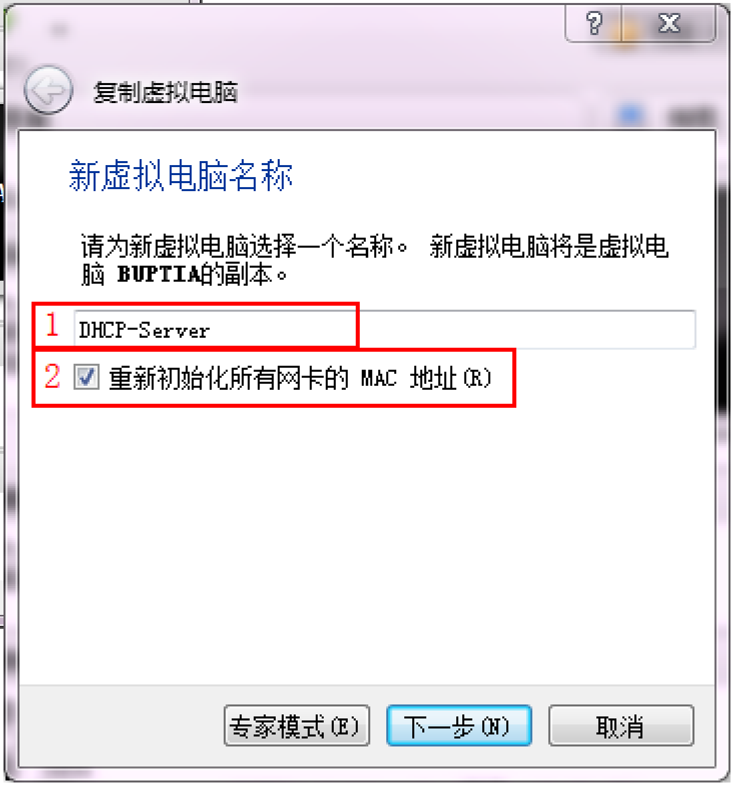
☑ 接入网线(C)

端口转发(P)

OK Cancel

2. DHCP server的虚拟机



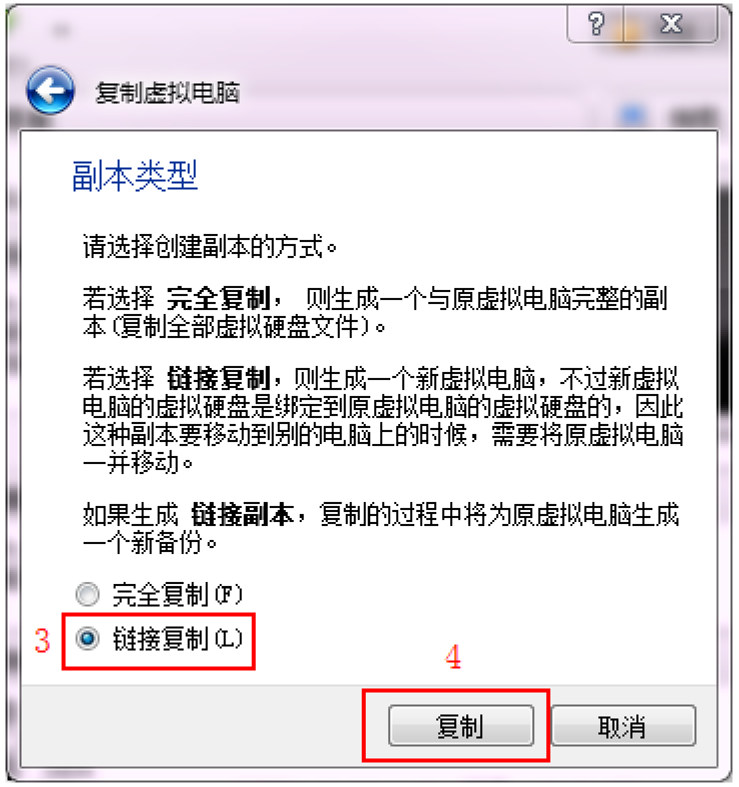

复制虚拟电脑

新虚拟电脑名称

请为新虚拟电脑选择一个名称。 新虚拟电脑将是虚拟电脑 **BUPTIA**的副本。

1 DHCP-Server

2 [✓] 重新初始化所有网卡的 MAC 地址(R)


复制虚拟电脑

副本类型

请选择创建副本的方式。

若选择 **完全复制**， 则生成一个与原虚拟电脑完整的副本(复制全部虚拟硬盘文件)。

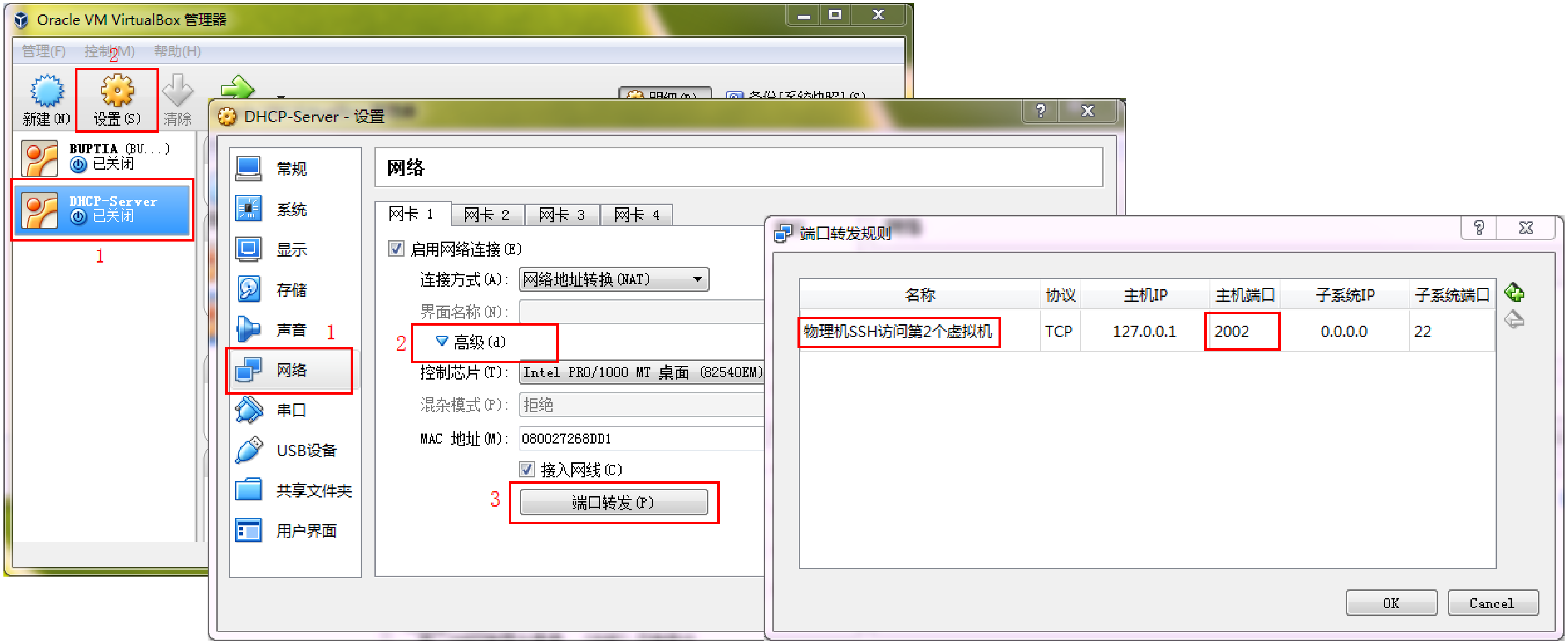
若选择 **链接复制**，则生成一个新虚拟电脑，不过新虚拟电脑的虚拟硬盘是绑定到原虚拟电脑的虚拟硬盘的，因此这种副本要移动到别的电脑上的时候，需要将原虚拟电脑一并移动。

如果生成 **链接副本**，复制的过程中将为原虚拟电脑生成一个新备份。

○ 完全复制(F)

3 ● 链接复制(L)

2.1 设置“网卡1”



启用网络连接(E): ✓

连接方式(A): 网络地址转换(NAT)

界面名称(N):

控制芯片(T): Intel PRO/1000 MT 桌面 (82540EM)

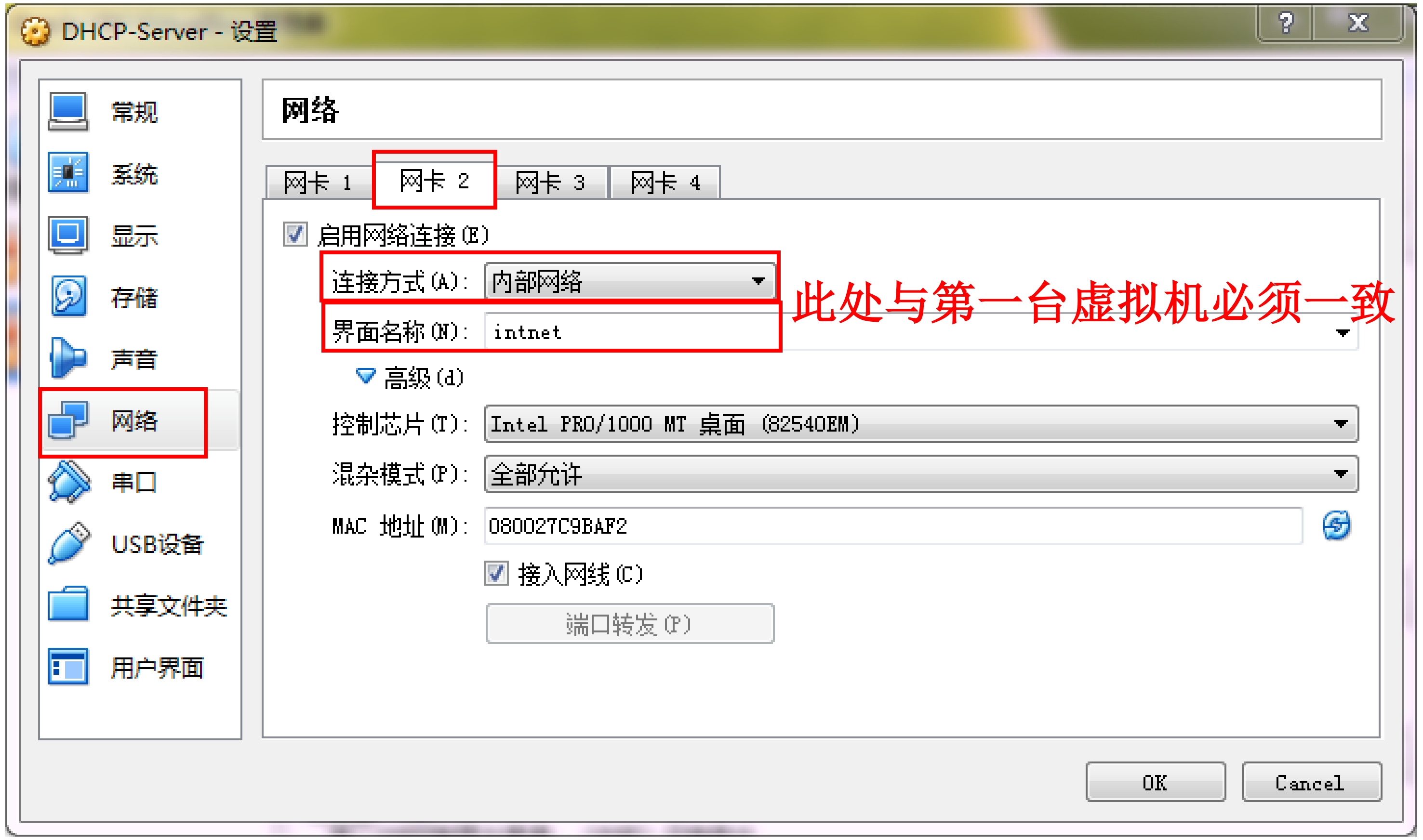
混杂模式(P): 拒绝

MAC 地址(M): 080027268DD1

接入网线(C): ✓

名称	协议	主机IP	主机端口	子系统IP	子系统端口
物理机SSH访问第2个虚拟机	TCP	127.0.0.1	2002	0.0.0.0	22

2.2 检查“网卡2”


DHCP-Server - 设置

网络

☑ 启用网络连接(E)

连接方式(A): 内部网络

界面名称(N): intnet

此处与第一台虚拟机必须一致

▽ 高级(d)

控制芯片(T): Intel PRO/1000 MT 桌面 (82540EM)

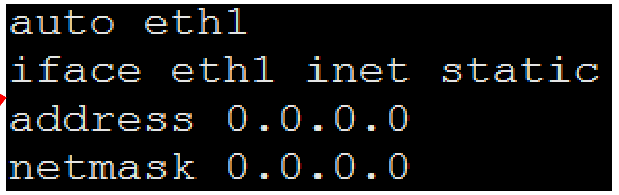
混杂模式(P): 全部允许

MAC 地址(M): 080027C9BAF2

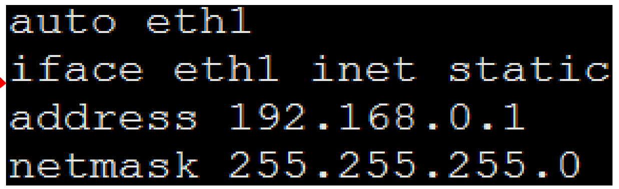
☑ 接入网线(C)

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
- 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 0.0.0.0
netmask 0.0.0.0
```



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

4. 增加iptables规则

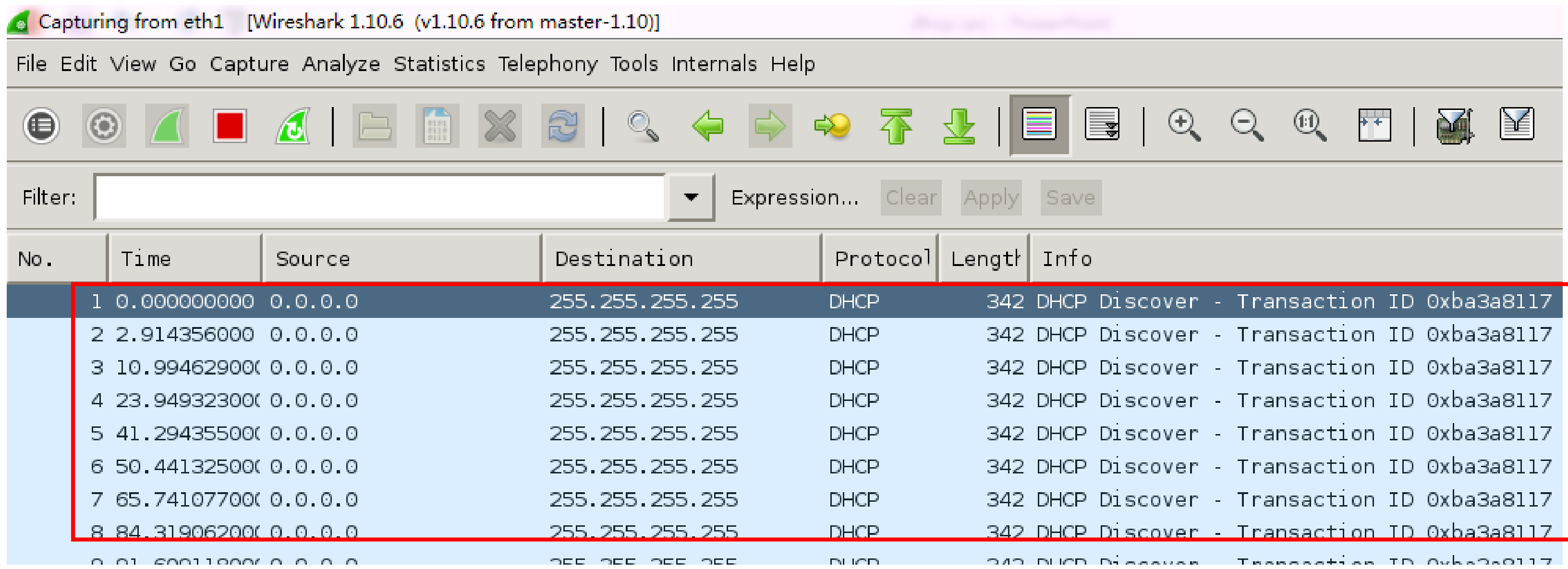
- 在Linux系统中， 当数据包的目标地址为255.255.255.255广播地址，源地址必须设置为0.0.0.0，数据包才能被处理
- 由于普通的socket函数不能设置源地址为0.0.0.0，所以需要利用iptables工具对广播数据包修改源地址
- 请分别在**两个**虚拟机的终端模拟器内执行：
  - sudo -i
  - iptables -t nat -A POSTROUTING -d 255.255.255.255 -o eth1 -j SNAT --to-source 0.0.0.0
  - iptables-save > /etc/network/iptables.rules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

    iptables-restore < /etc/network/iptables.rules

    exit 0

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 10.99462900( 0.0.0.0   255.255.255.255   DHCP        342 DHCP Discover - Transaction ID 0xba3a8117
    4 23.94932300( 0.0.0.0   255.255.255.255   DHCP        342 DHCP Discover - Transaction ID 0xba3a8117
    5 41.29435500( 0.0.0.0   255.255.255.255   DHCP        342 DHCP Discover - Transaction ID 0xba3a8117
    6 50.44132500( 0.0.0.0   255.255.255.255   DHCP        342 DHCP Discover - Transaction ID 0xba3a8117
    7 65.74107700( 0.0.0.0   255.255.255.255   DHCP        342 DHCP Discover - Transaction ID 0xba3a8117
    8 84.31906200( 0.0.0.0   255.255.255.255   DHCP        342 DHCP Discover - Transaction ID 0xba3a8117
    9 91.60911800( 0.0.0.0   255.255.255.255   DHCP        342 DHCP Discover - Transaction ID 0xba3a8117
```

Environment of Network is OK!

Part 2: Hints about the Programs

6. Hints about broadcast

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

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

int: 0:success -1:error

SOCKET s: s(套接字): 指向一个打开的套接口描述字

int level: 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）

int optname: 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

const char* optval: 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

int optlen: optlen(选项长度)：optval 的大小

- E.g.: allow socket to broadcast

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

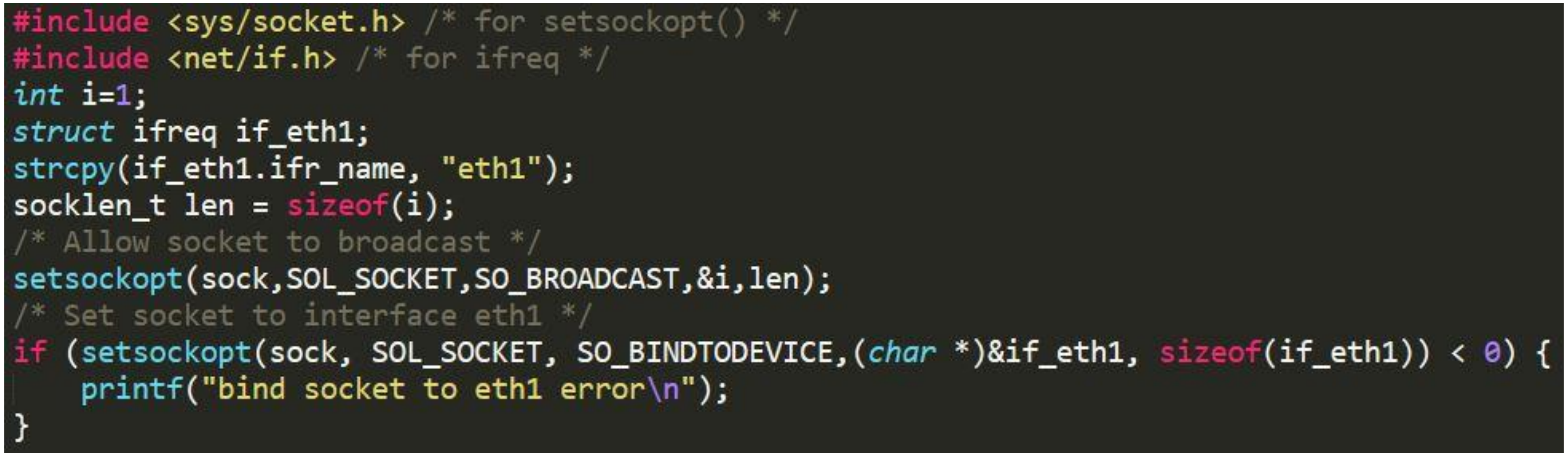
- 网络接口信息结构体struct ifreq (#include <net/if.h>)

```
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 */
        };
};
```

- E.g.: declare an ifreq struct to store eth1 interface information

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

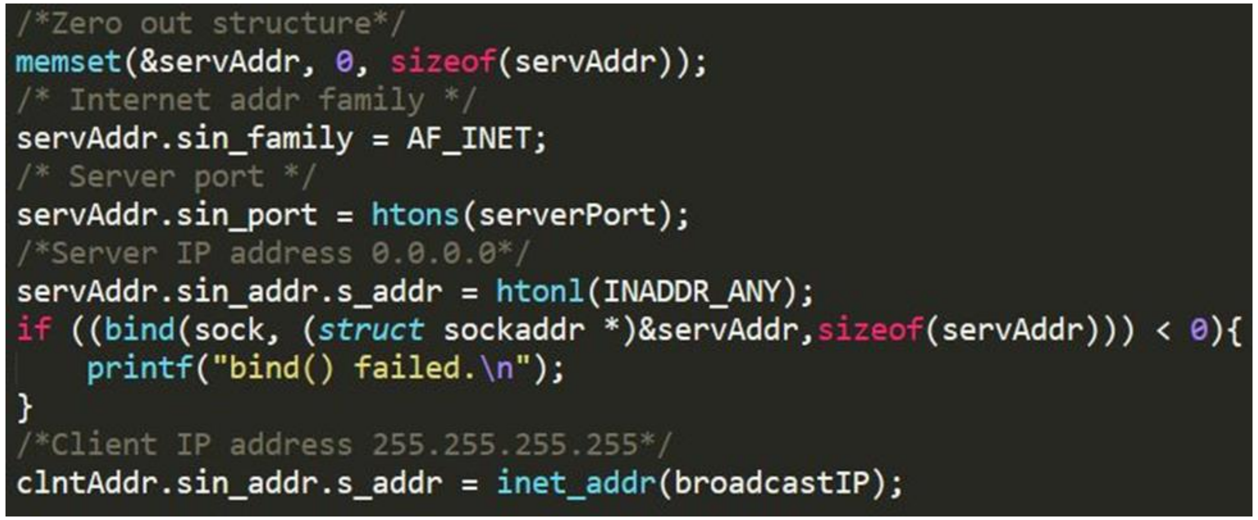
- Example: Allow a socket to broadcast and bind the socket to interface eth1



```c
#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");
}
```

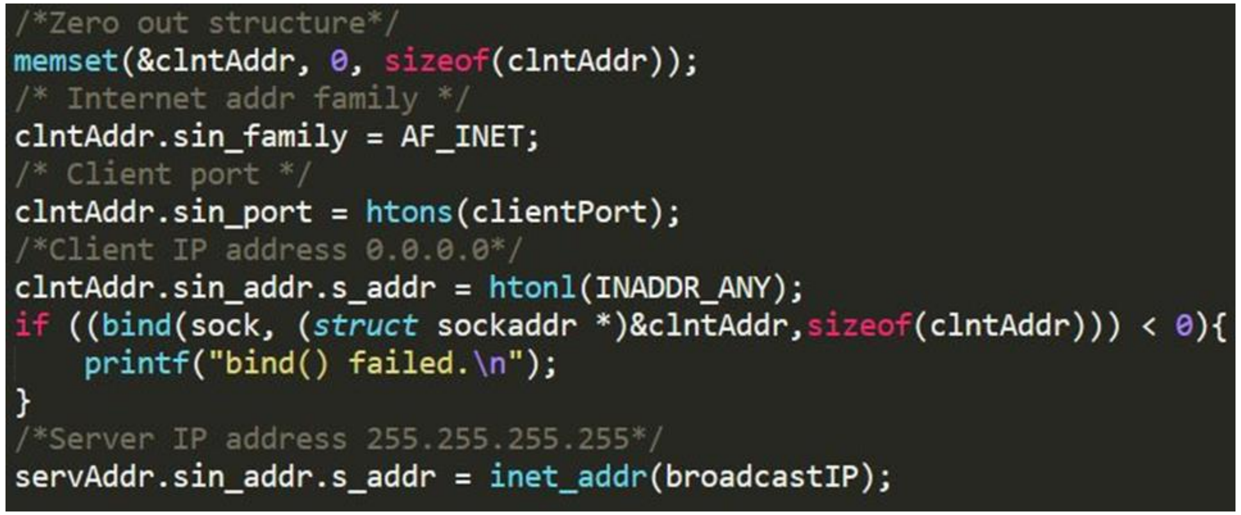
- Set address:
  set local address to 0.0.0.0 and broadcast address to 255.255.255.255
- Example:

Server:



```
/*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);
```

Client:



```
/*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