socket编程

0 虚拟网卡设置

例:

`/etc/sysconfig/network-scripts/ifcfg-ens32`

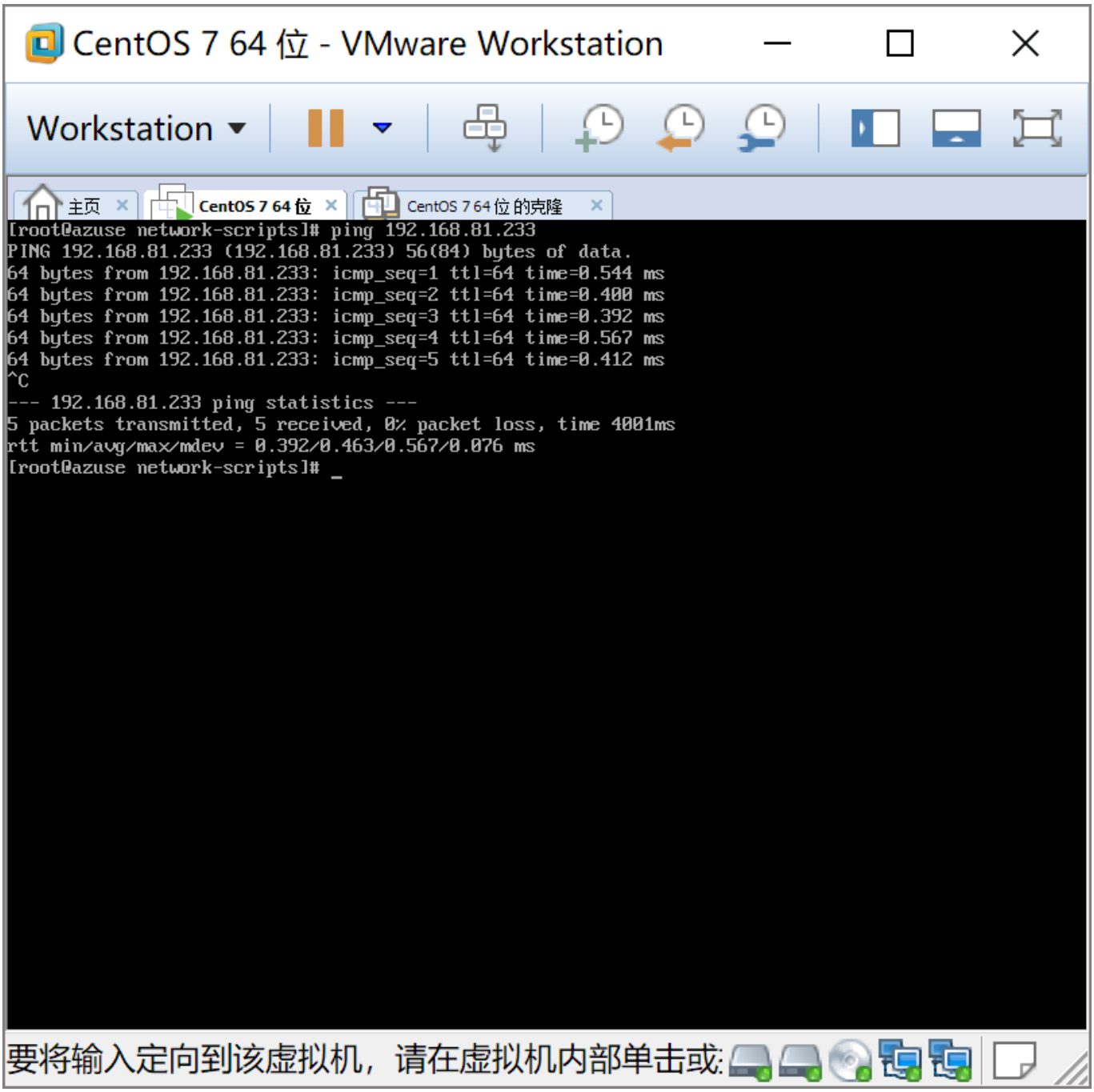
```
TYPE=Ethernet
PROXY_METHOD=none
BROWSER_ONLY=no
BOOTPROTO=static
DEFROUTE=yes
IPV4_FAILURE_FATAL=yes
IPV6INIT=no
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
IPV6_ADDR_GEN_MODE=stable-privacy
NAME=ens32
UUID=26cf553c-793d-4d7a-89f6-a3dad05f5054
DEVICE=ens32
ONBOOT=yes
IPADDR=192.168.80.230
NETMASK=255.255.255.0
PREFIX=24
GATEWAY=192.168.80.2
DNS1=192.168.80.2
IPV6_PRIVACY=no
HARDDR=00:0c:29:da:30:2d
ZONE=public
```

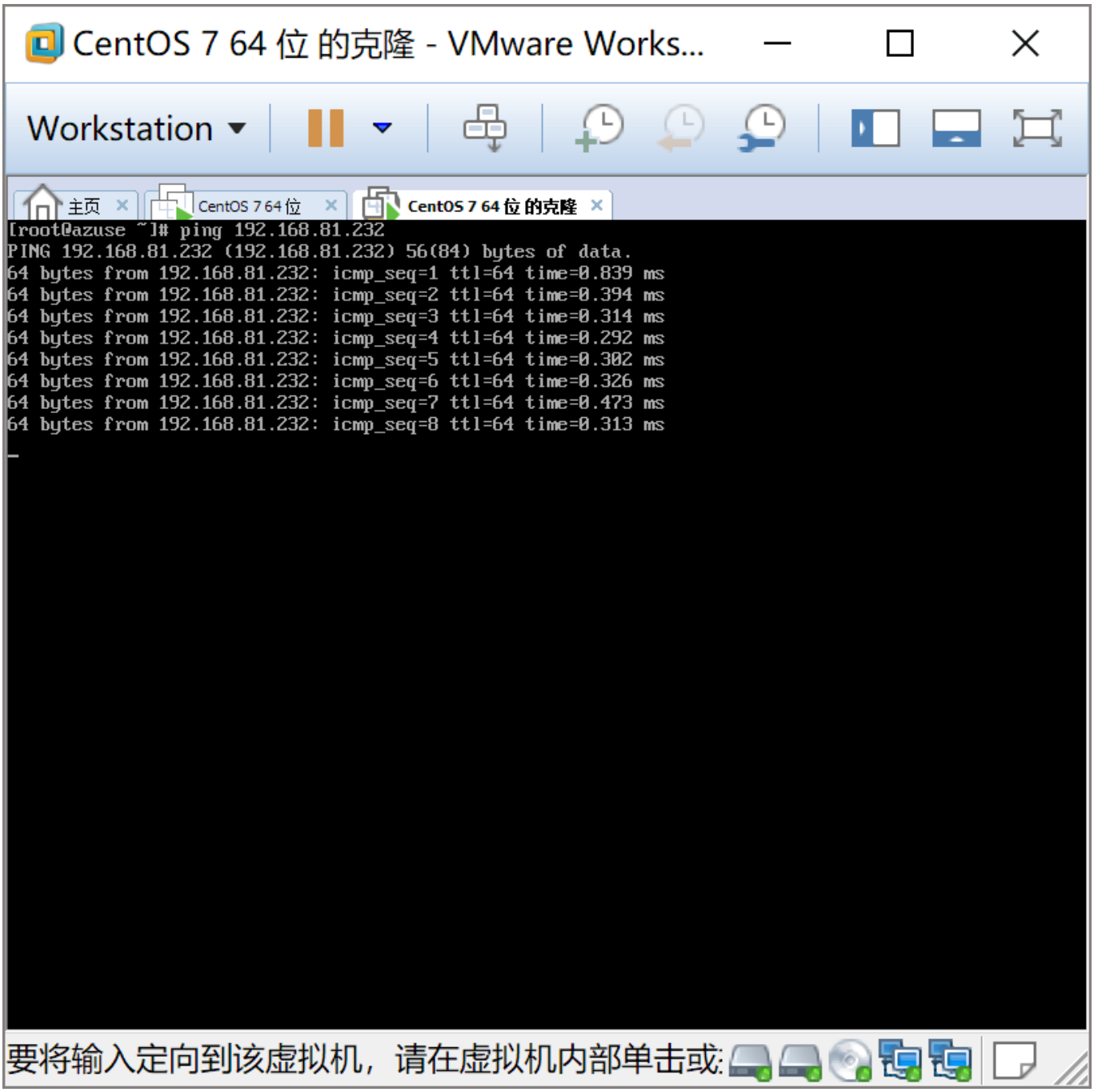
`/etc/sysconfig/network-scripts/ifcfg-ens32:1`

```
DEVICE=ens32:1
BOOTPROTO=static
IPADDR=192.168.81.232
NETMASK=255.255.255.0
ONBOOT=yes
```

`systemctl restart network`使配置生效

(将vm1的ip2设置为192.168.81.232 vm2的ip2设置为192.168.81.233)


CentOS 7 64 位 - VMware Workstation
Workstation
主页
CentOS 7 64 位
CentOS 7 64 位的克隆
[root@azuse network-scripts]# ping 192.168.81.233
PING 192.168.81.233 (192.168.81.233) 56(84) bytes of data.
64 bytes from 192.168.81.233: icmp_seq=1 ttl=64 time=0.544 ms
64 bytes from 192.168.81.233: icmp_seq=2 ttl=64 time=0.400 ms
64 bytes from 192.168.81.233: icmp_seq=3 ttl=64 time=0.392 ms
64 bytes from 192.168.81.233: icmp_seq=4 ttl=64 time=0.567 ms
64 bytes from 192.168.81.233: icmp_seq=5 ttl=64 time=0.412 ms
^C
--- 192.168.81.233 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4001ms
rtt min/avg/max/mdev = 0.392/0.463/0.567/0.076 ms
[root@azuse network-scripts]# _
要将输入定向到该虚拟机，请在虚拟机内部单击或



2 tcp_server1 tcp_client1

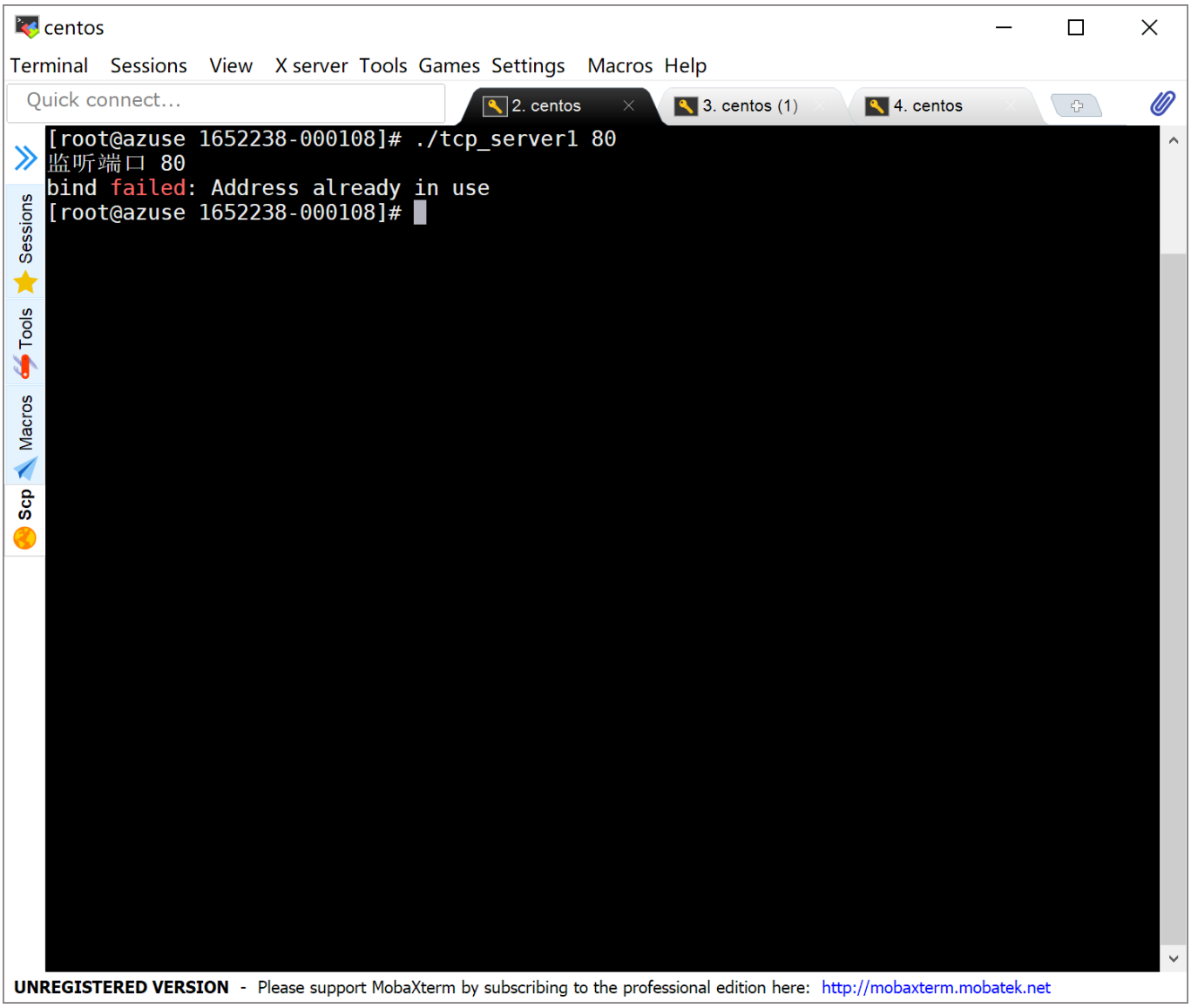
测试程序tcp_server1监听端口通过参数传入

先stop并disable firewalld

```
        address.sin_family = AF_INET;
    address.sin_addr.s_addr = INADDR_ANY;
    address.sin_port = htons( atoi(argv[1]) );

    if (bind(server_fd, (struct sockaddr *)&address,
                                sizeof(address))<0)
    {
        perror("bind failed");
        exit(EXIT_FAILURE);
```

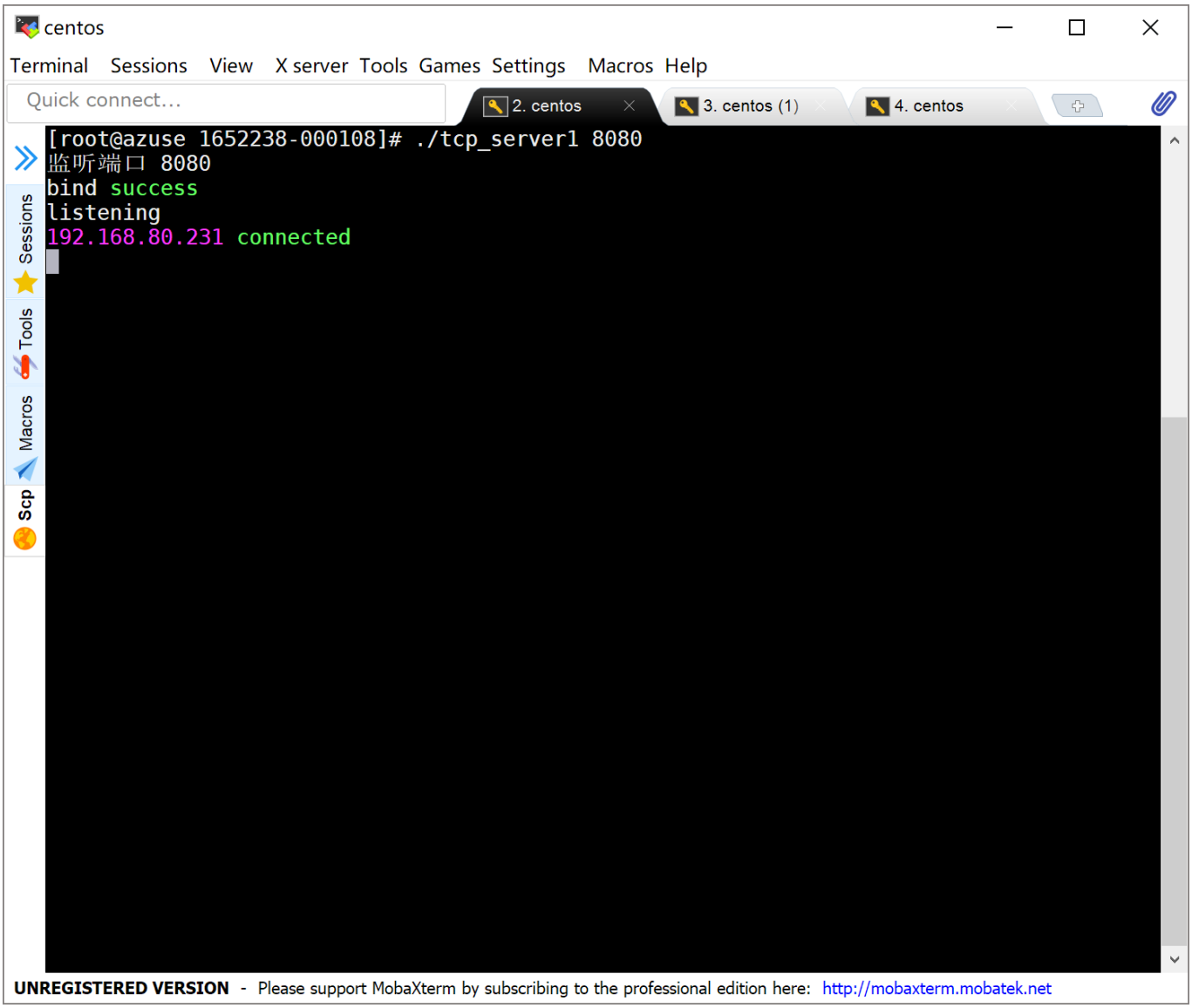
```
    }
```

如果端口已被占用，则会在bind函数上出错

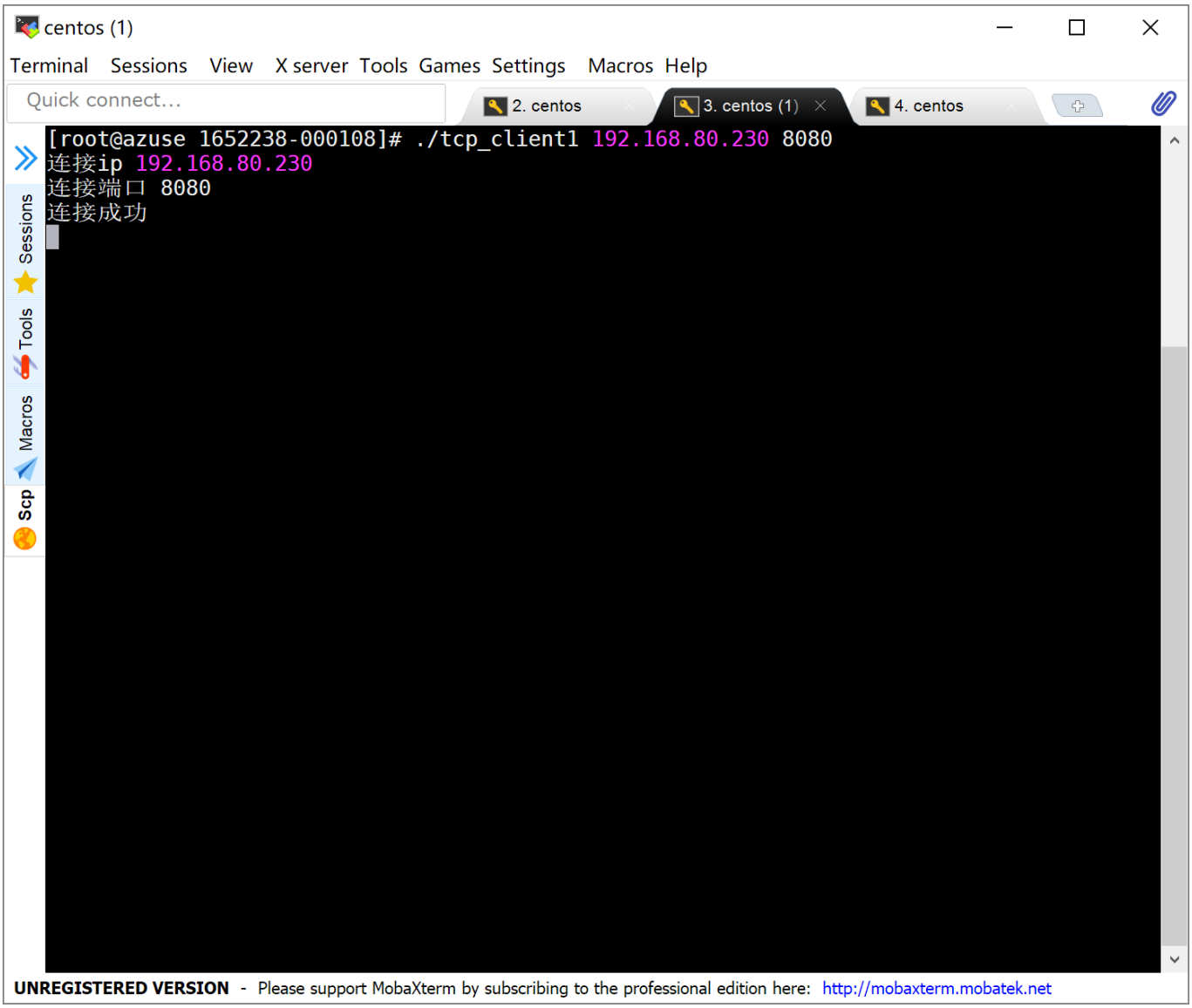


连接成功并进入recv状态

server进入recv状态：



client进入recv状态：



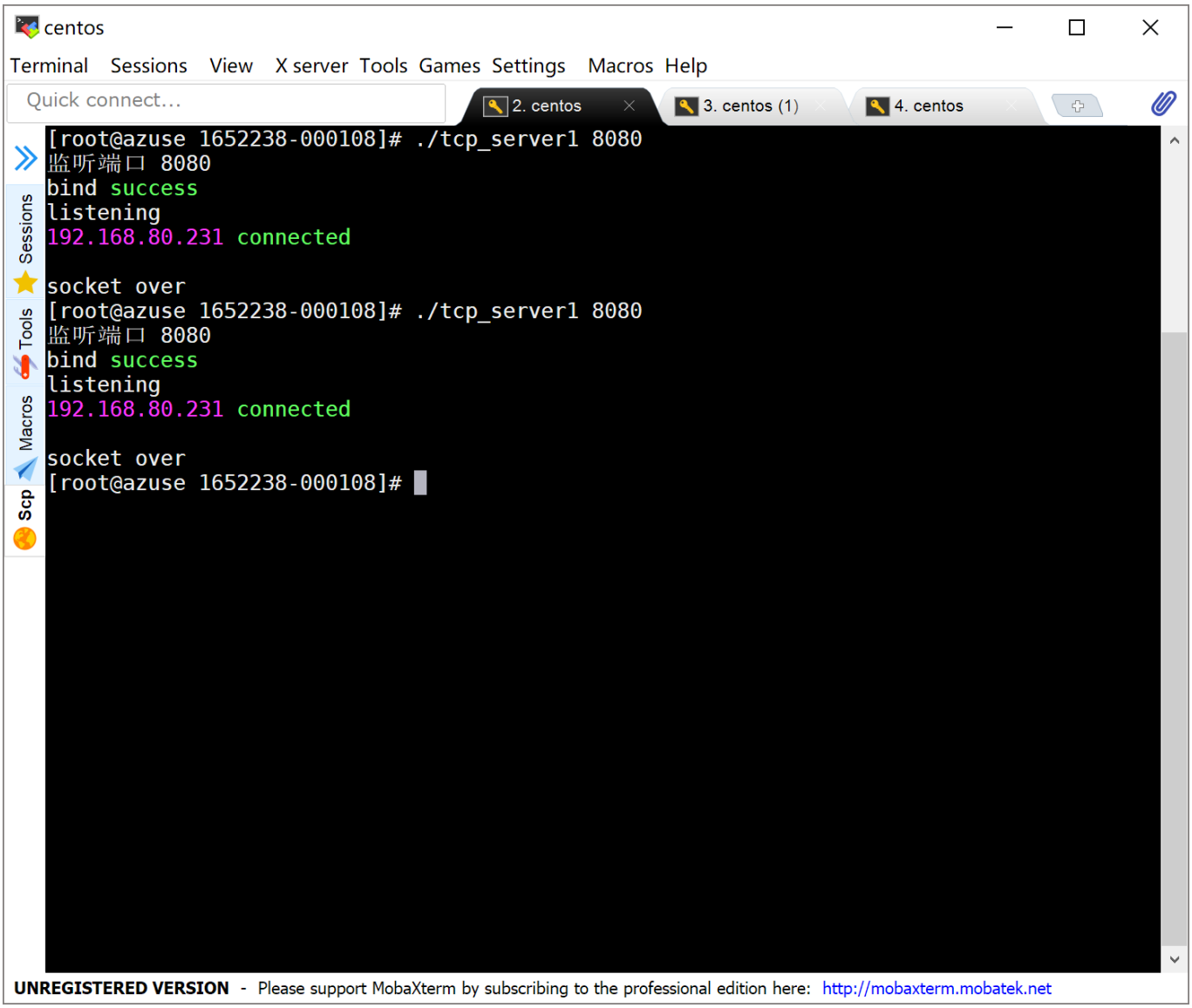
此时用ctrl+c中断client端，server端能否侦测到连接已中断?

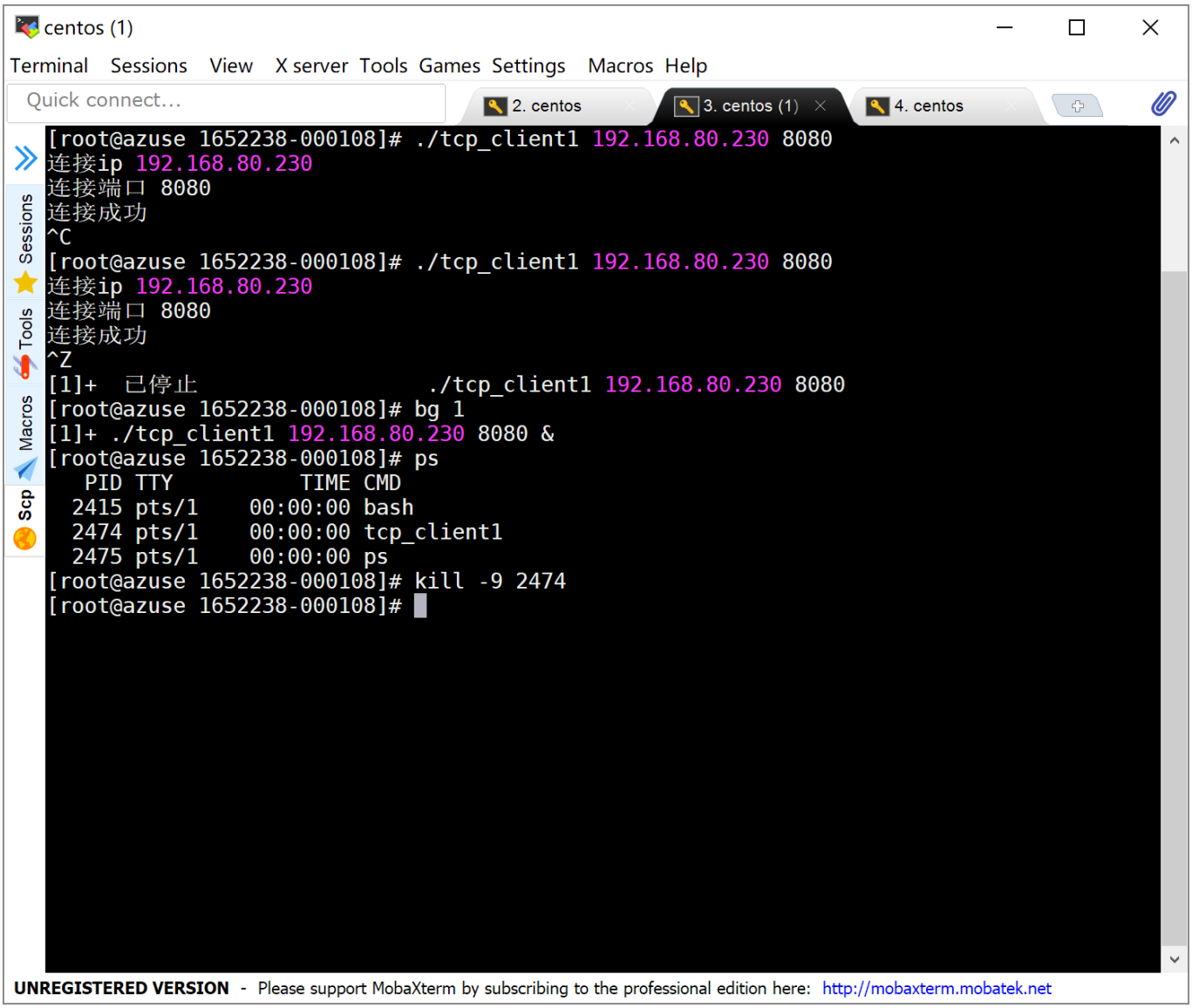
可以

此时用kill -9杀死client端，server端能否检测到连接已中断?

也可以

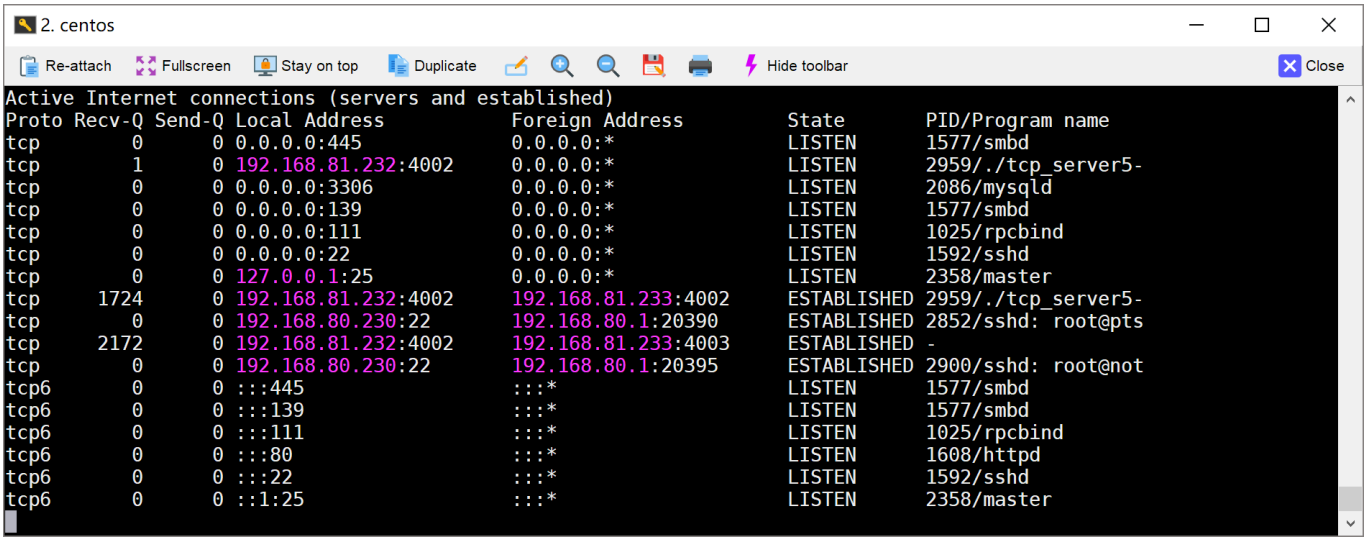
图：第一次位ctrl+c中断，第二次为先运行程序然后再ctrl+z bg1将程序放入后台（此时socket没有中断），再ps用kill -9杀死线程，socket中断。


centos
Terminal Sessions View X server Tools Games Settings Macros Help
Quick connect...
2. centos
3. centos (1)
4. centos
Sessions
Tools
Macros
Scp
[root@azuse 1652238-000108]# ./tcp_server1 8080
监听端口 8080
bind success
listening
192.168.80.231 connected
socket over
[root@azuse 1652238-000108]# ./tcp_server1 8080
监听端口 8080
bind success
listening
192.168.80.231 connected
socket over
[root@azuse 1652238-000108]#
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: http://mobaxterm.mobatek.net



双方链接成功后，再启动一个新的tcp_client1来连接server，会是什么情况?

第二个tcp_client1的connect函数返回正常，能发送数据，但是server无法接收
此时服务端的netstat显示链接为established，但是操作此链接的pid为空



连接成功后再启动新的server，绑定相同端口

1 不加reuseaddr bind失败，端口已在使用

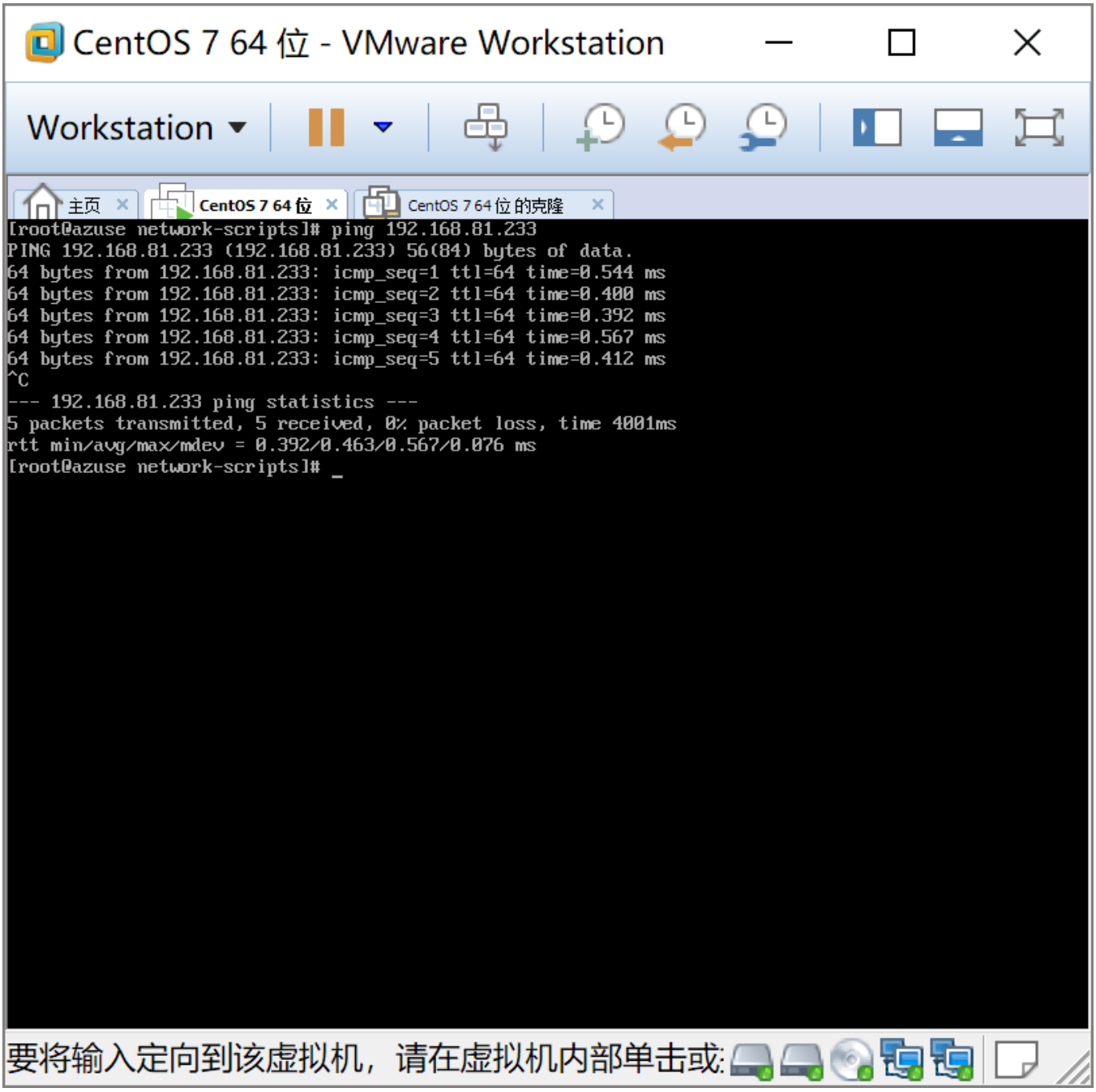
2 加reuseaddr reuseport

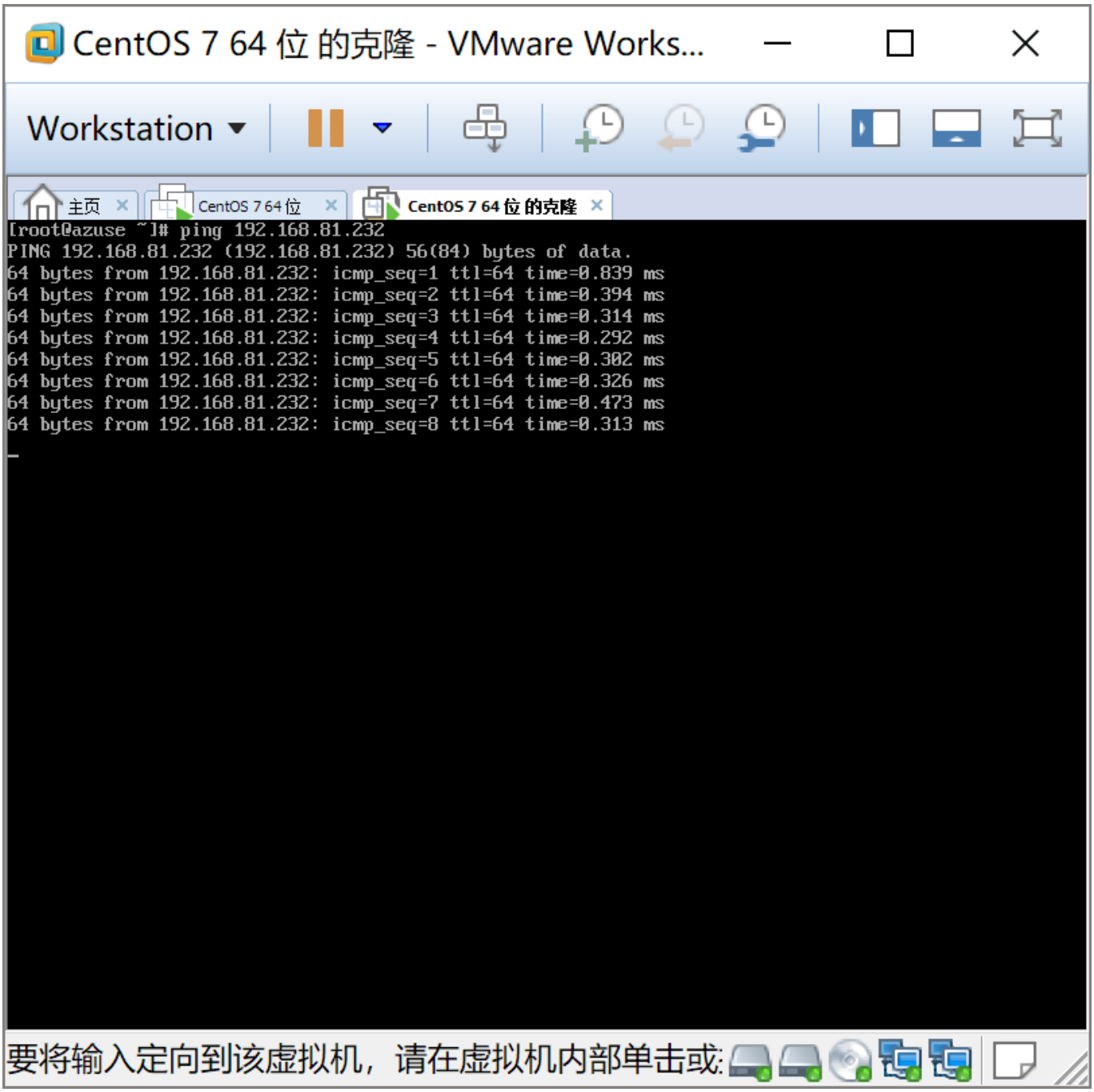
```
setsockopt(server_fd, SOL_SOCKET, SO_REUSEADDR | SO_REUSEPORT, &opt, sizeof(opt))
```

两个虚拟机ip设置与虚拟网卡不同的网段之后，能否ping通?

能，因为虚拟机只有ens32这一个网卡，所以所有流量都送192.168.80.1，而电脑的虚拟网卡收到发送到192.168.81.233的，由于虚拟网卡并没有发往192.168.81.0网段的路由转发规则，所以会广播，广播收到vm2得网卡响应，记录下这条路由表，之后就可以把所有发到192.168.81.233的包发往vm2了

(将vm1的ip2设置为192.168.81.232 vm2的ip2设置为192.168.81.233)



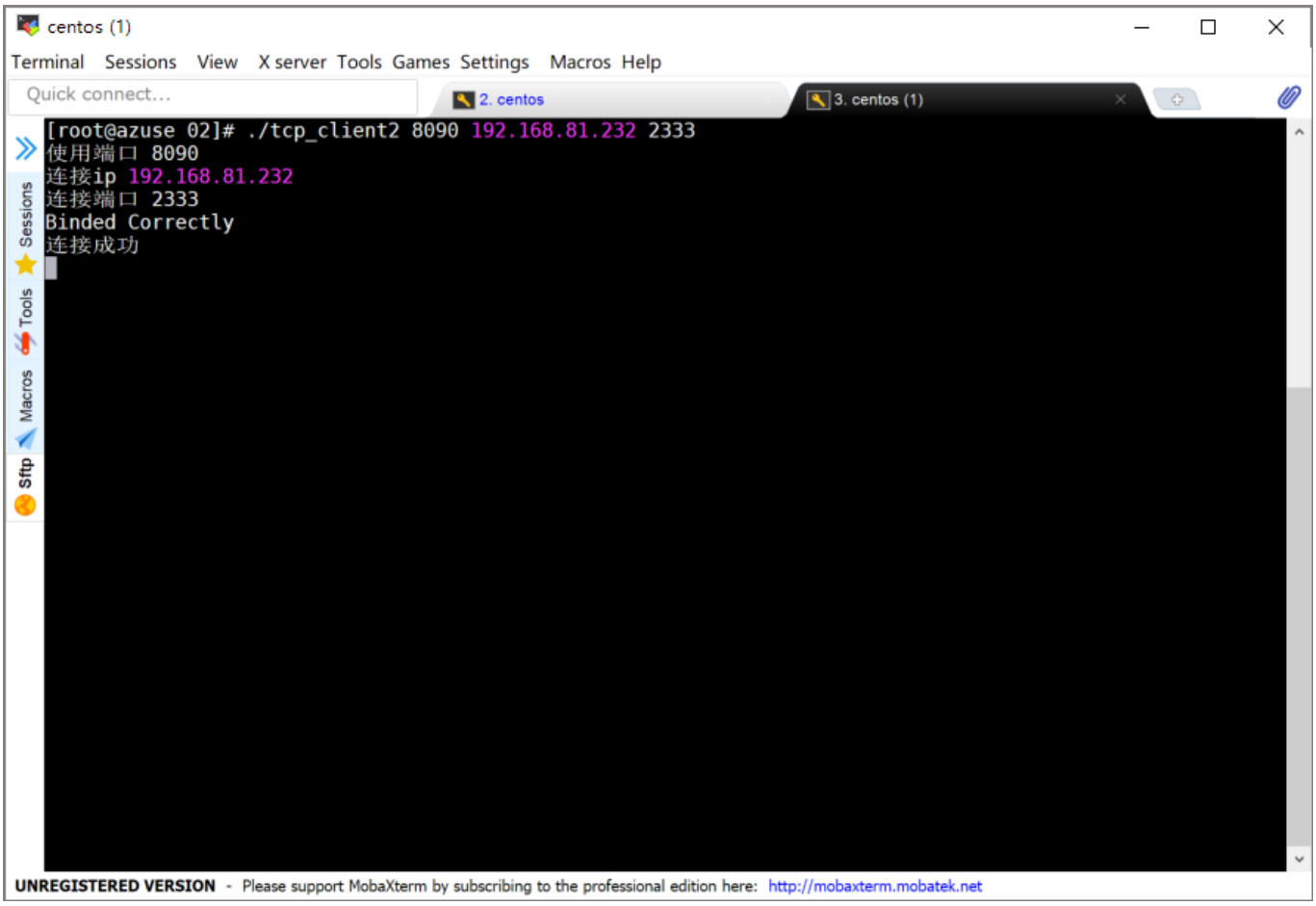


3 设置客户端端口号

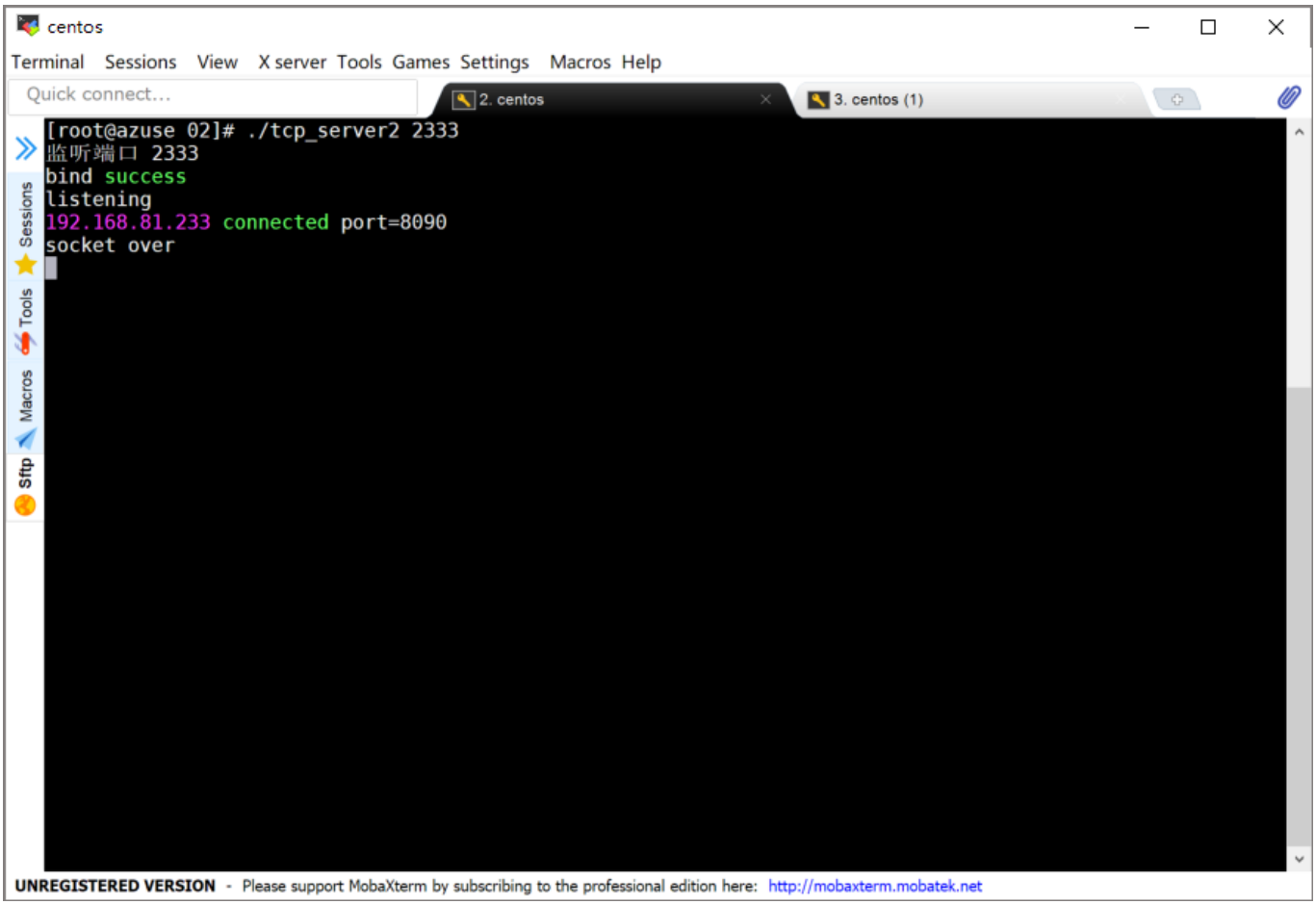
默认客户端使用的端口号是随机的，但是通过给客户端设置bind函数，我们可以绑定客户端的端口号。

```
    int port = atoi(argv[1]);
        my_addr.sin_family = AF_INET;
        my_addr.sin_addr.s_addr = inet_addr("192.168.81.233");
        my_addr.sin_port = htons(port);

        if (bind(sock, (struct sockaddr *)&my_addr, sizeof(struct sockaddr_in)) ==
0)
                printf("Binded Correctly\n");
        else
                printf("Unable to bind\n");
```

使用效果： 客户端


centos (1)
Terminal Sessions View X server Tools Games Settings Macros Help
Quick connect...
2. centos
3. centos (1)
[root@azuse 02]# ./tcp_client2 8090 192.168.81.232 2333
使用端口 8090
连接ip 192.168.81.232
连接端口 2333
Binded Correctly
连接成功
Sessions
Tools
Macros
Sftp
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: http://mobaxterm.mobatek.net

服务端：



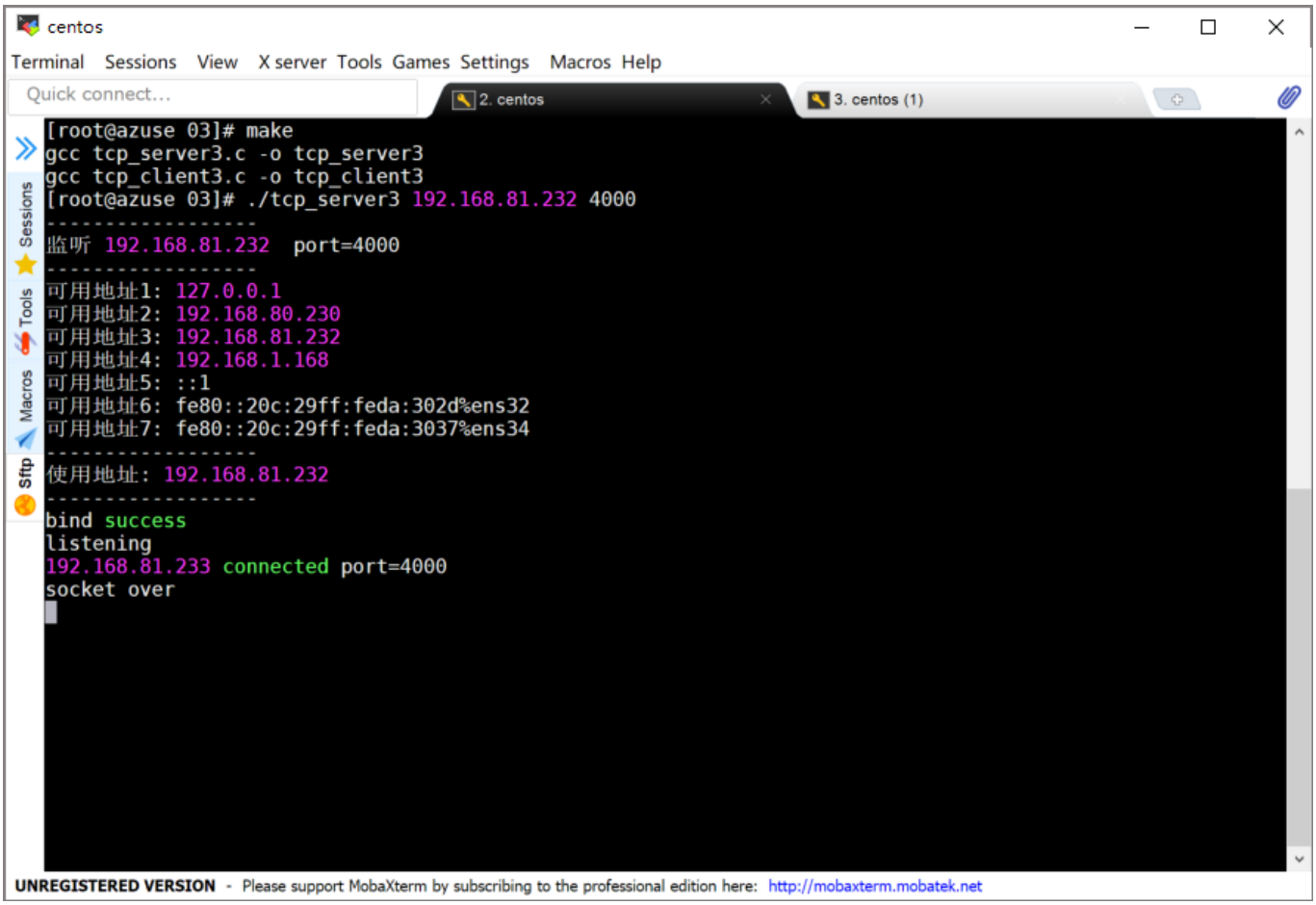
4 列出所有网卡ip，并且只绑定指定的ip

使用`getifaddress(&ifaddr)`来获得所有网卡信息

```
void listInterfaces(int *interface_num, char interface_address[100][NI_MAXHOST])
{
        struct ifaddrs *ifaddr, *ifa;
        int family, s, n;
        char host[NI_MAXHOST];

        if (getifaddrs(&ifaddr) == -1)
        {
                perror("getifaddrs");
                exit(EXIT_FAILURE);
        }

        /* Walk through linked list, maintaining head pointer so we
              can free list later */

        for (ifa = ifaddr, n = 0; ifa != NULL; ifa = ifa->ifa_next, n++)
        {
                if (ifa->ifa_addr == NULL)
                        continue;

                family = ifa->ifa_addr->sa_family;

                /* Display interface name and family (including symbolic
```

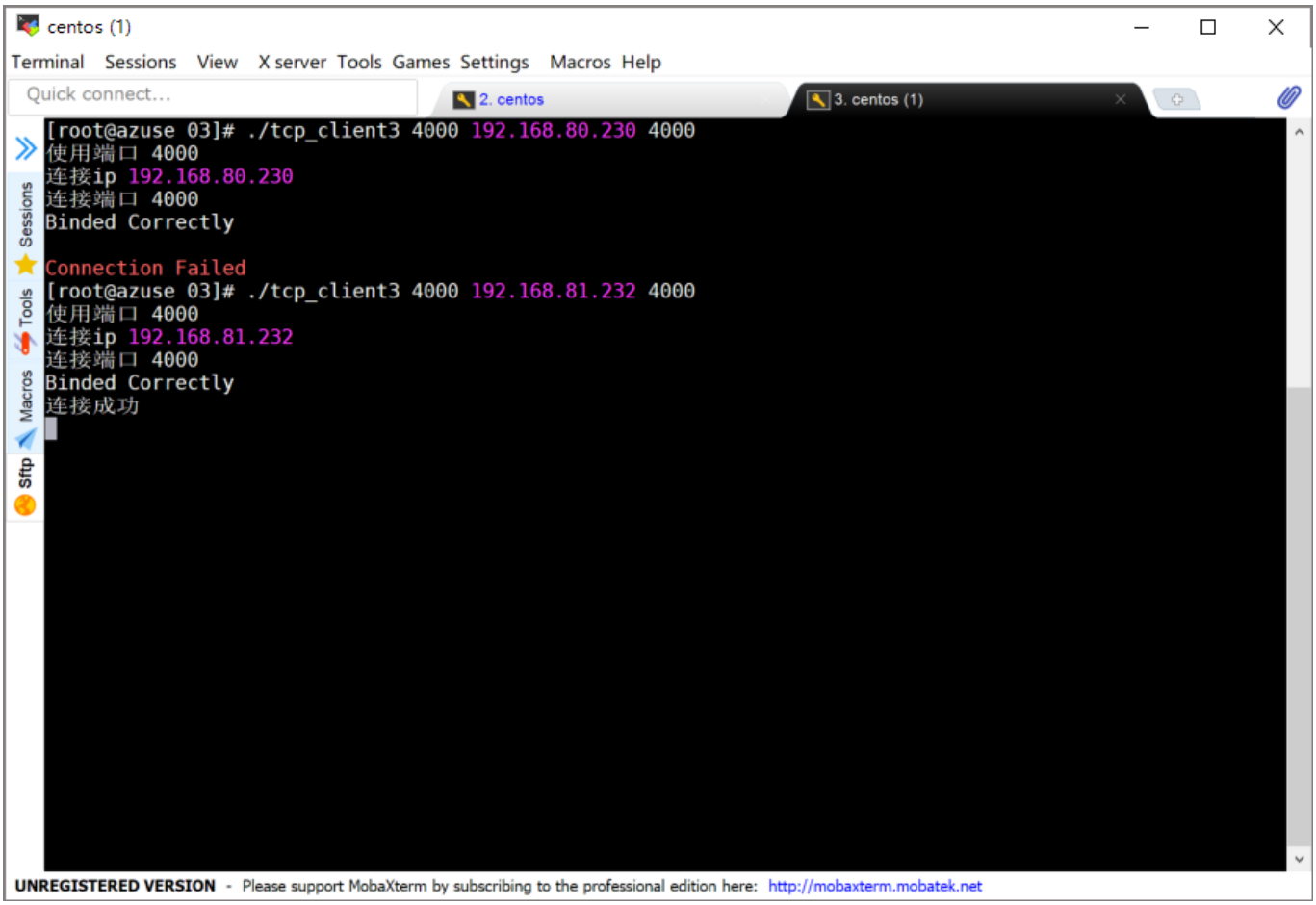
```
                  form of the latter for the common families) */

                // printf("%-8s %s (%d)\n",
                //         ifa->ifa_name,
                //         (family == AF_PACKET) ? "AF_PACKET" : (family ==
AF_INET) ? "AF_INET" : (family == AF_INET6) ? "AF_INET6" : "???",
                //         family);

                /* For an AF_INET* interface address, display the address */

                if (family == AF_INET || family == AF_INET6)
                {
                        s = getnameinfo(ifa->ifa_addr,
                                                (family == AF_INET) ?
sizeof(struct sockaddr_in) : sizeof(struct sockaddr_in6),
                                                host, NI_MAXHOST,
                                                NULL, 0, NI_NUMERICHOST);
                        if (s != 0)
                        {
                                //printf("getnameinfo() failed: %s\n",
gai_strerror(s));
                                exit(EXIT_FAILURE);
                        }

                        //printf("\t\taddress: <%s>\n", host);
                        strcpy(interface_address[*interface_num], host);
                        *interface_num = *interface_num + 1;
                        printf("可用地址%d: %s\n", *interface_num, host);
                }
                else if (family == AF_PACKET && ifa->ifa_data != NULL)
                {
                        struct rtnl_link_stats *stats = ifa->ifa_data;

                        // printf("\t\ttx_packets = %10u; rx_packets = %10u\n"
                        //         "\t\ttx_bytes   = %10u; rx_bytes   = %10u\n",
                        //         stats->tx_packets, stats->rx_packets,
                        //         stats->tx_bytes, stats->rx_bytes);
                }
        }

        freeifaddrs(ifaddr);
}
```

使用效果:



如果客户端连接未绑定的网卡会怎么样?

会连不上



5 tcp_server4编写, read/recv区别, write/send区别

tcp_server4-1的read部分

```
        while (1)
        {
                if ((new_socket = accept(server_fd, (struct sockaddr *)&address,
                                                                (socklen_t
*)&addrlen)) < 0)
                {
                        perror("accept failed");
                        continue;
                }
                else
                {
                        printf("%u.%u.%u.%u connected port=%d\n",
address.sin_addr.s_addr & 0x000000FF, (address.sin_addr.s_addr & 0x0000FF00) >> 8,
(address.sin_addr.s_addr & 0x00FF0000) >> 16, (address.sin_addr.s_addr &
0xFF000000) >> 24, ntohs(address.sin_port));
                }
                printf("reading...\n");
                read(new_socket, buffer, sizeof(buffer));
                printf("message read: %s\n", buffer);

                printf("socket over\n");
        }
```

tcp_client4-1-1write部分，一次性write20字节

```
strcpy(sendbuffer, "20byte length send");
write(sock, sendbuffer, sizeof(sendbuffer));
printf("message write: %s\n",sendbuffer);
```

tcp_client4-1-2write部分，循环write2字节

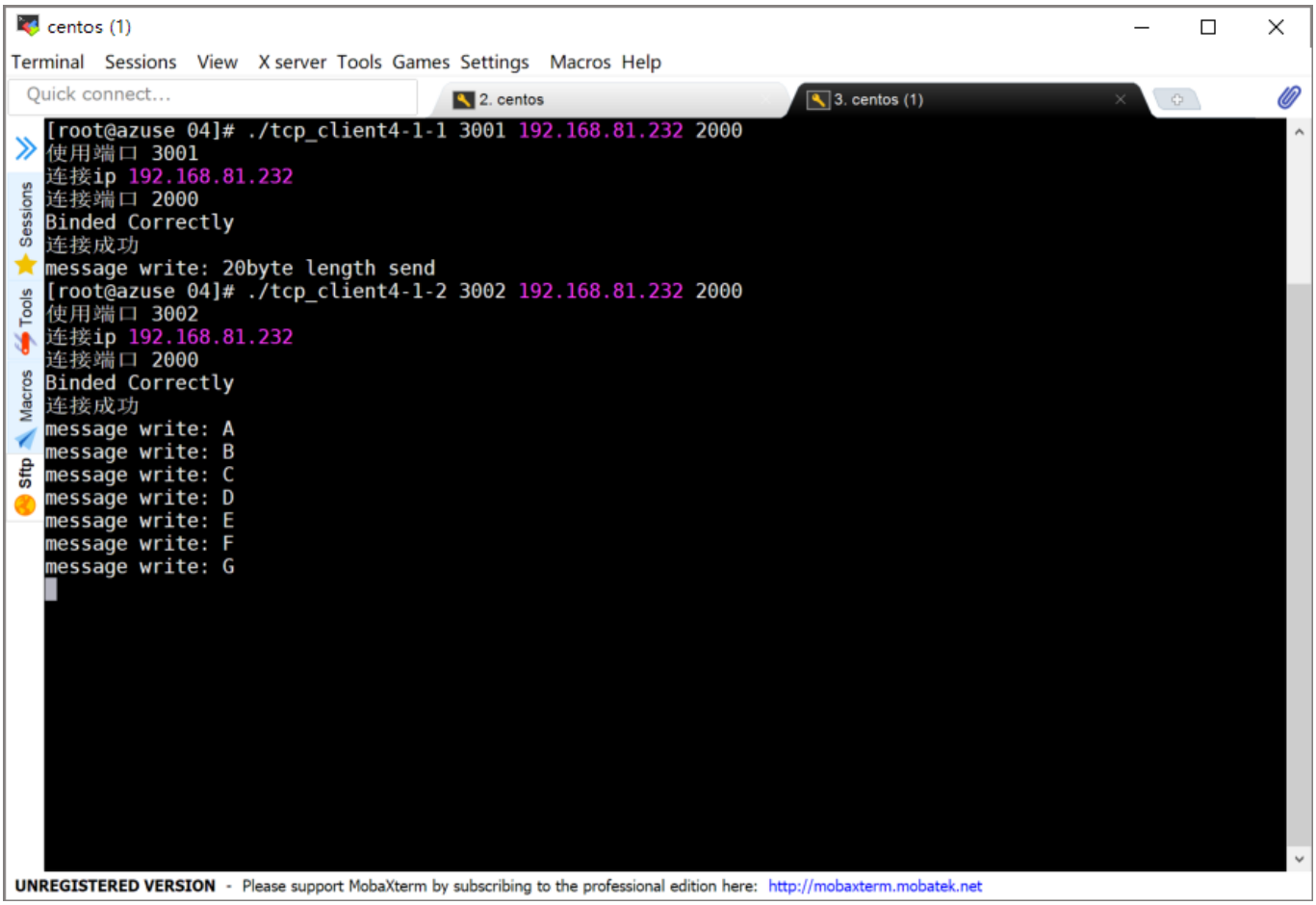
```
while (1)
        {
                write(sock, sendbuffer, sizeof(sendbuffer));
                printf("message write: %s\n", sendbuffer);

                sendbuffer[0]++;
                sleep(2);
        }
```

测试效果

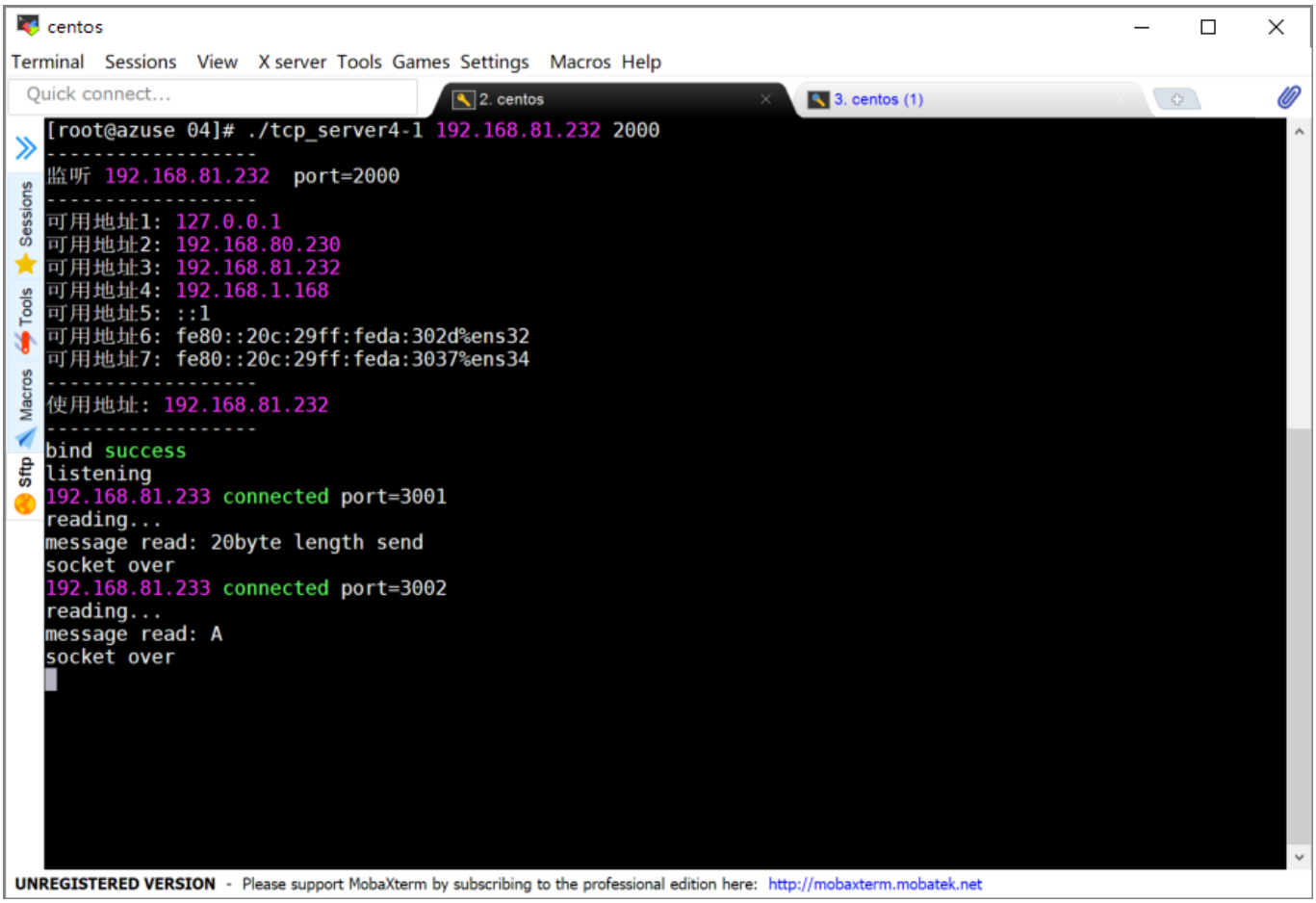
使用write一次发送20字节：服务端能正常接收所有数据。
循环write一次发送2字节：服务端只能接收第一次发送的数据。

客户端：

```
./tcp_client4-1-1 3001 192.168.81.232 2000
./tcp_client4-1-2 3002 192.168.81.232 2000
```


centos (1)
Terminal Sessions View X server Tools Games Settings Macros Help
Quick connect...
2. centos
3. centos (1)
[root@azuse 04]# ./tcp_client4-1-1 3001 192.168.81.232 2000
使用端口 3001
连接ip 192.168.81.232
连接端口 2000
Binded Correctly
连接成功
message write: 20byte length send
[root@azuse 04]# ./tcp_client4-1-2 3002 192.168.81.232 2000
使用端口 3002
连接ip 192.168.81.232
连接端口 2000
Binded Correctly
连接成功
message write: A
message write: B
message write: C
message write: D
message write: E
message write: F
message write: G
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: http://mobaxterm.mobatek.net

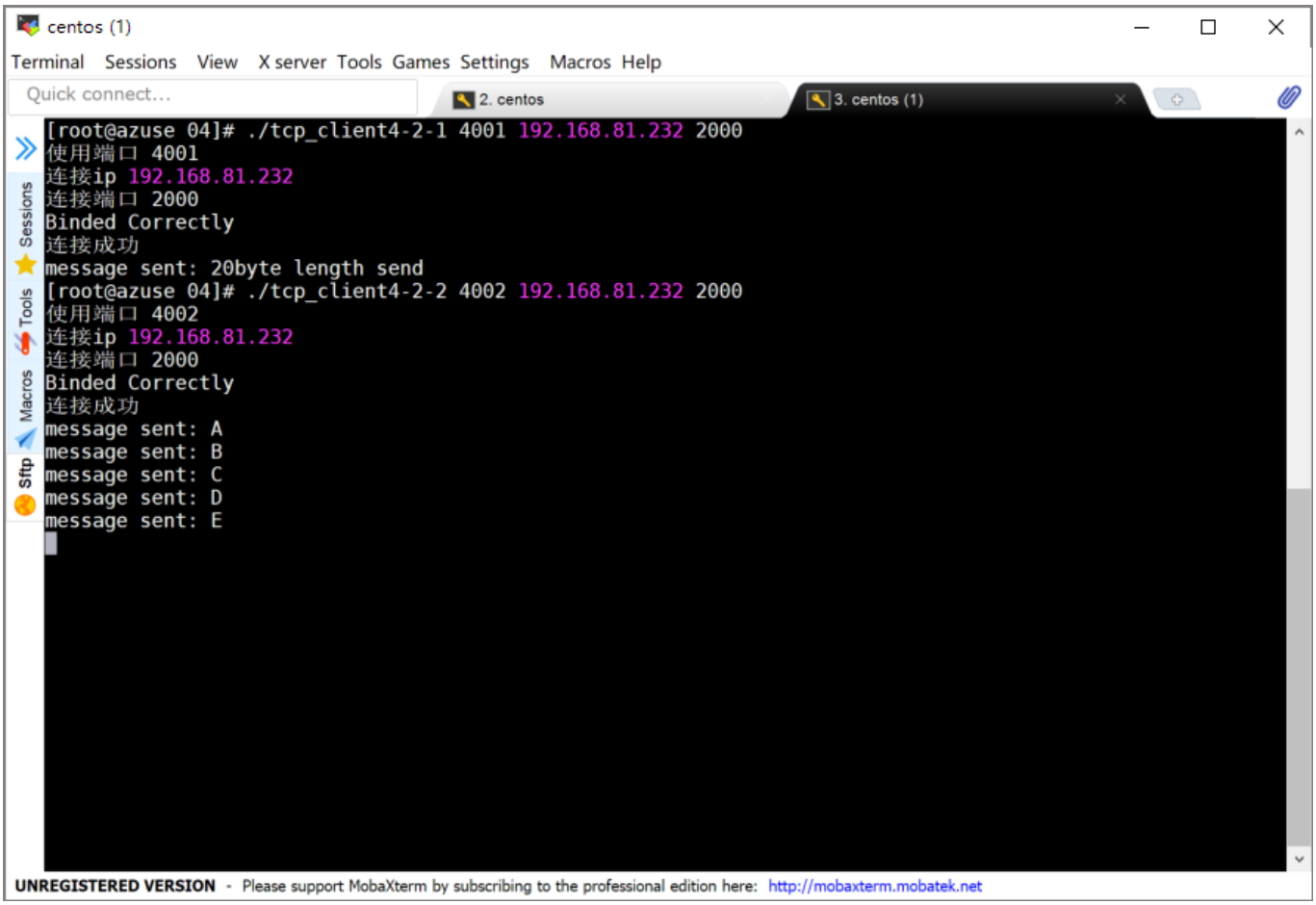
服务端:

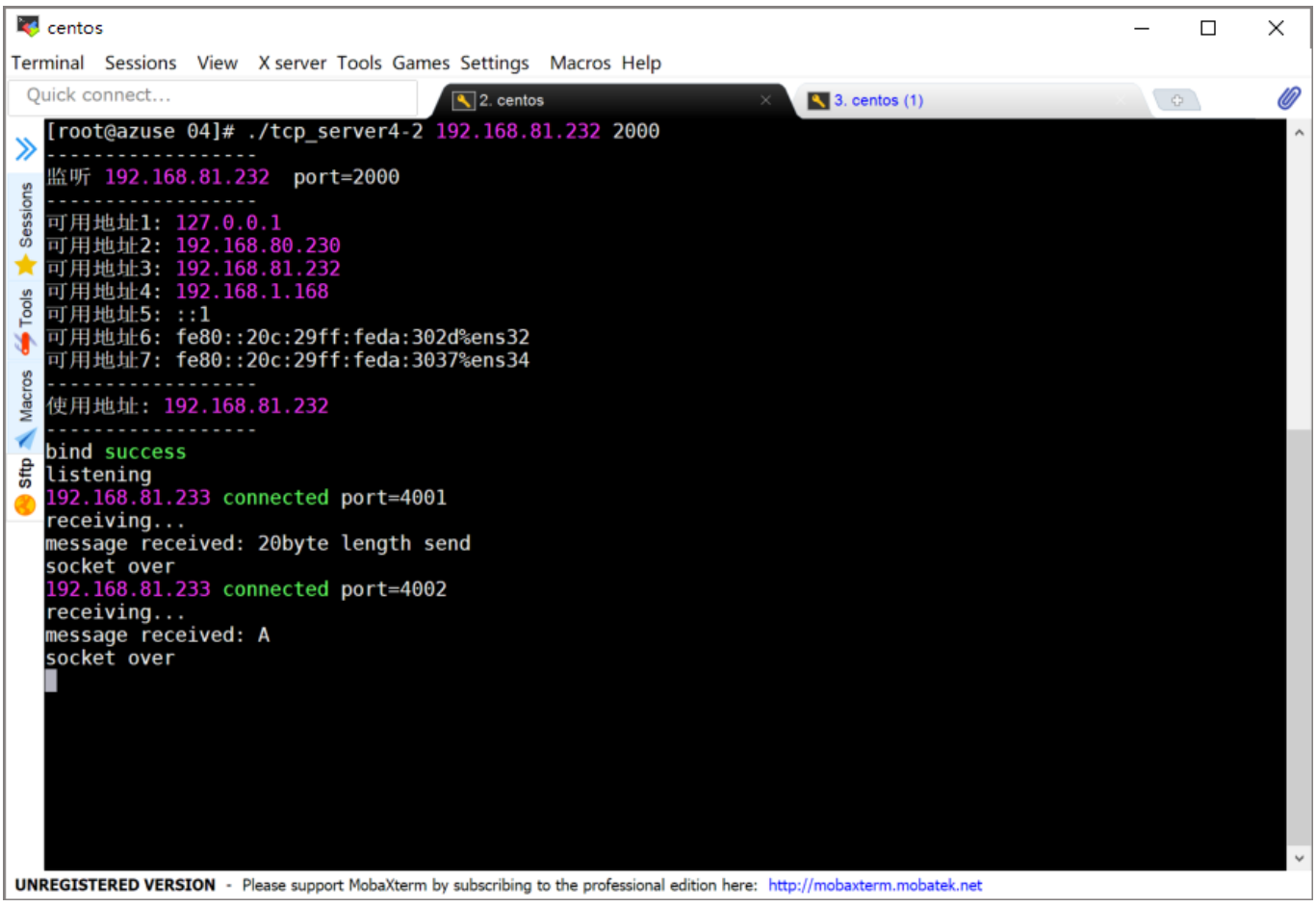

centos
Terminal Sessions View X server Tools Games Settings Macros Help
Quick connect...
2. centos
3. centos (1)
[root@azuse 04]# ./tcp_server4-1 192.168.81.232 2000
监听 192.168.81.232 port=2000
可用地址1: 127.0.0.1
可用地址2: 192.168.80.230
可用地址3: 192.168.81.232
可用地址4: 192.168.1.168
可用地址5: ::1
可用地址6: fe80::20c:29ff:feda:302d%ens32
可用地址7: fe80::20c:29ff:feda:3037%ens34
使用地址: 192.168.81.232
bind success
listening
192.168.81.233 connected port=3001
reading...
message read: 20byte length send
socket over
192.168.81.233 connected port=3002
reading...
message read: A
socket over
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: http://mobaxterm.mobatek.net

tcp_server4-2,将read换成recv，write换成send

使用效果相同，tcp_client4-2-2还是只能收到最初两个字节。
recv与send函数比read/write多一个实参`flag`，flag有两个选项`MSG_OOB`和`MSG_DONTROUTE`，不使用时应传入`0`。

> flags
> | MSG_DONTROUTE | 不查找表 |
> | MSG_OOB | 接受或者发送带外数据 |
> | MSG_PEEK | 查看数据,并不从系统缓冲区移走数据 |
> | MSG_WAITALL | 等待所有数据





6 client不断向server写数据测试阻塞状态

tcp_server5-1接收部分

```
        while (1)
        {
                if ((new_socket = accept(server_fd, (struct sockaddr *)&address,
                                                                (socklen_t
*)&addrlen)) < 0)
                {
                        perror("accept failed");
                        continue;
                }
                else
                {
                        printf("%u.%u.%u.%u connected port=%d\n",
address.sin_addr.s_addr & 0x000000FF, (address.sin_addr.s_addr & 0x0000FF00) >> 8,
(address.sin_addr.s_addr & 0x00FF0000) >> 16, (address.sin_addr.s_addr &
0xFF000000) >> 24, ntohs(address.sin_port));
                }
                printf("waiting for recv, press anykey to start");
                getchar();
                long int readbytes = 0;
                while (1)
                {
                        int tmp;
                        tmp = read(new_socket, buffer, sizeof(buffer));
                        readbytes += tmp;
                        printf("read %d bytes, %d total\n", tmp, readbytes);
                        getchar();
                }

                printf("socket over\n");
        }
```

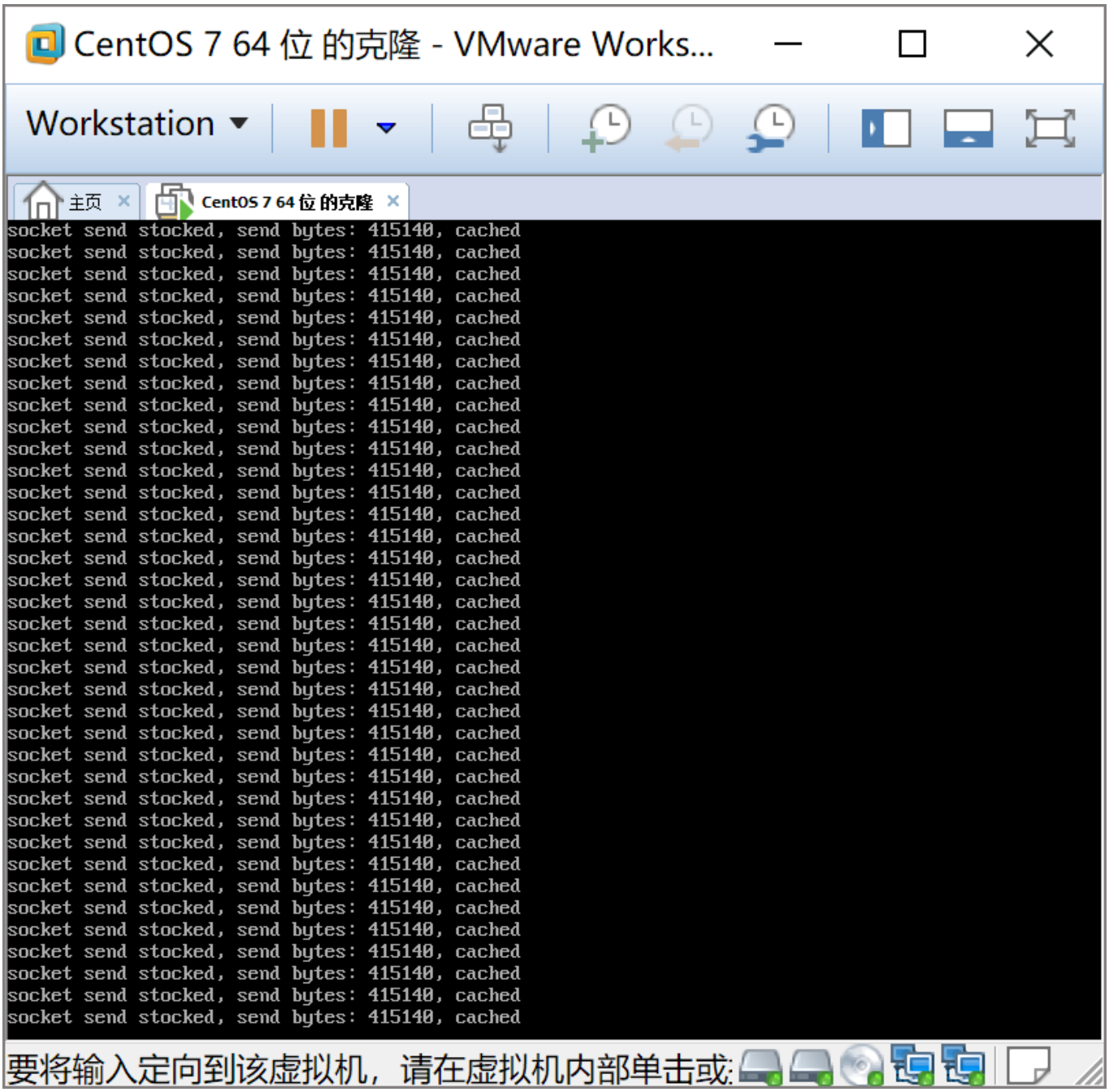
tcp_client5-1写部分，子进程进行socket通信，父进程检测子进程阻塞状态。

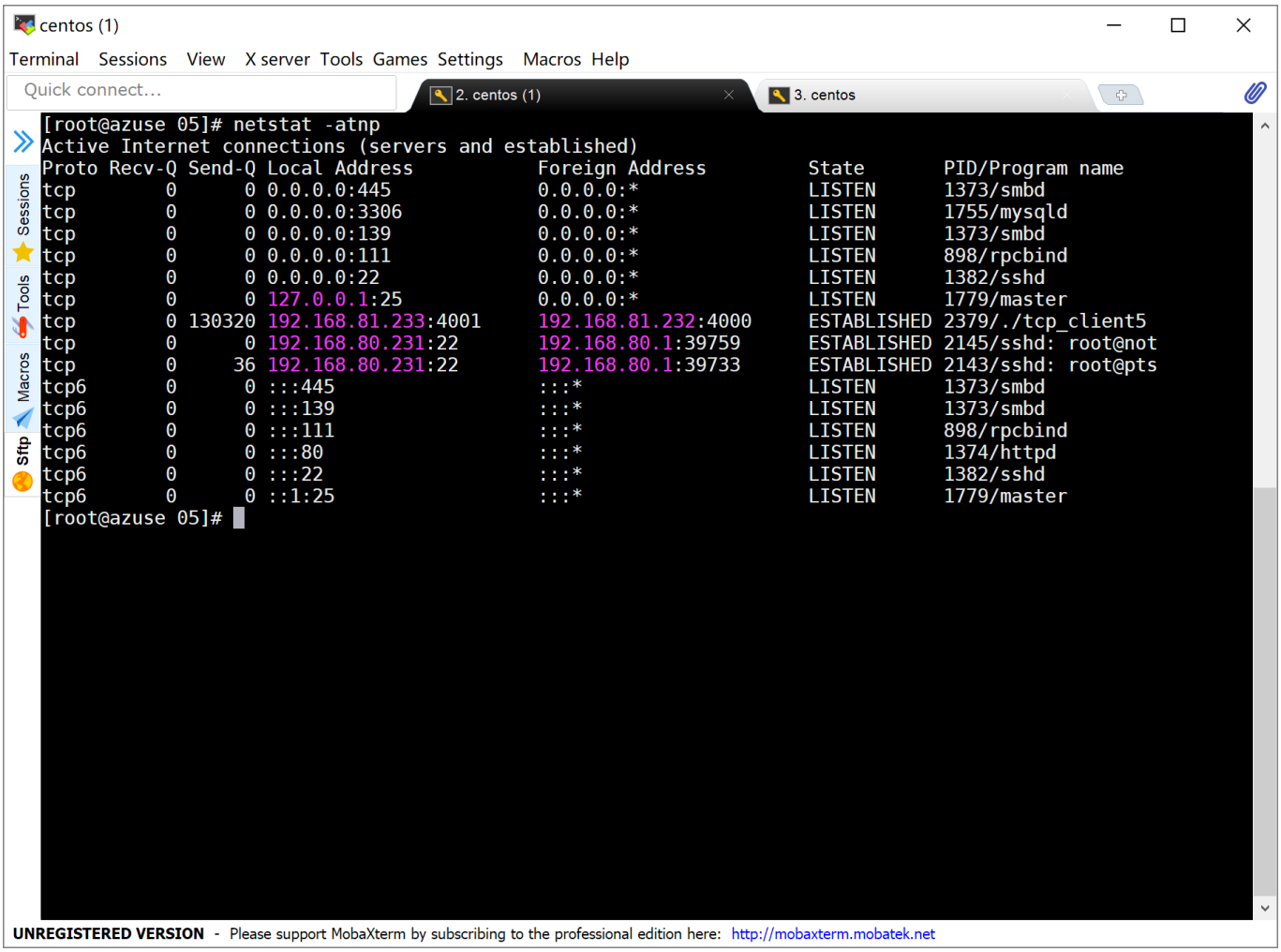
```
        int pid = fork();

        if (pid == 0)
        {
                prctl(PR_SET_PDEATHSIG, SIGHUP);
                long int *sendbytes_p = (long int *)shmat(shmid, NULL, 0);
                while (1)
                {
                        int tmp = write(sock, send_buffer, sizeof(send_buffer));
                        *sendbytes_p += tmp;
                        printf("write %d bytes, %ld total\n", tmp, *sendbytes_p);
                        getchar();
                        //sleep(1);
                }
                printf("socket over\n");
        }
```

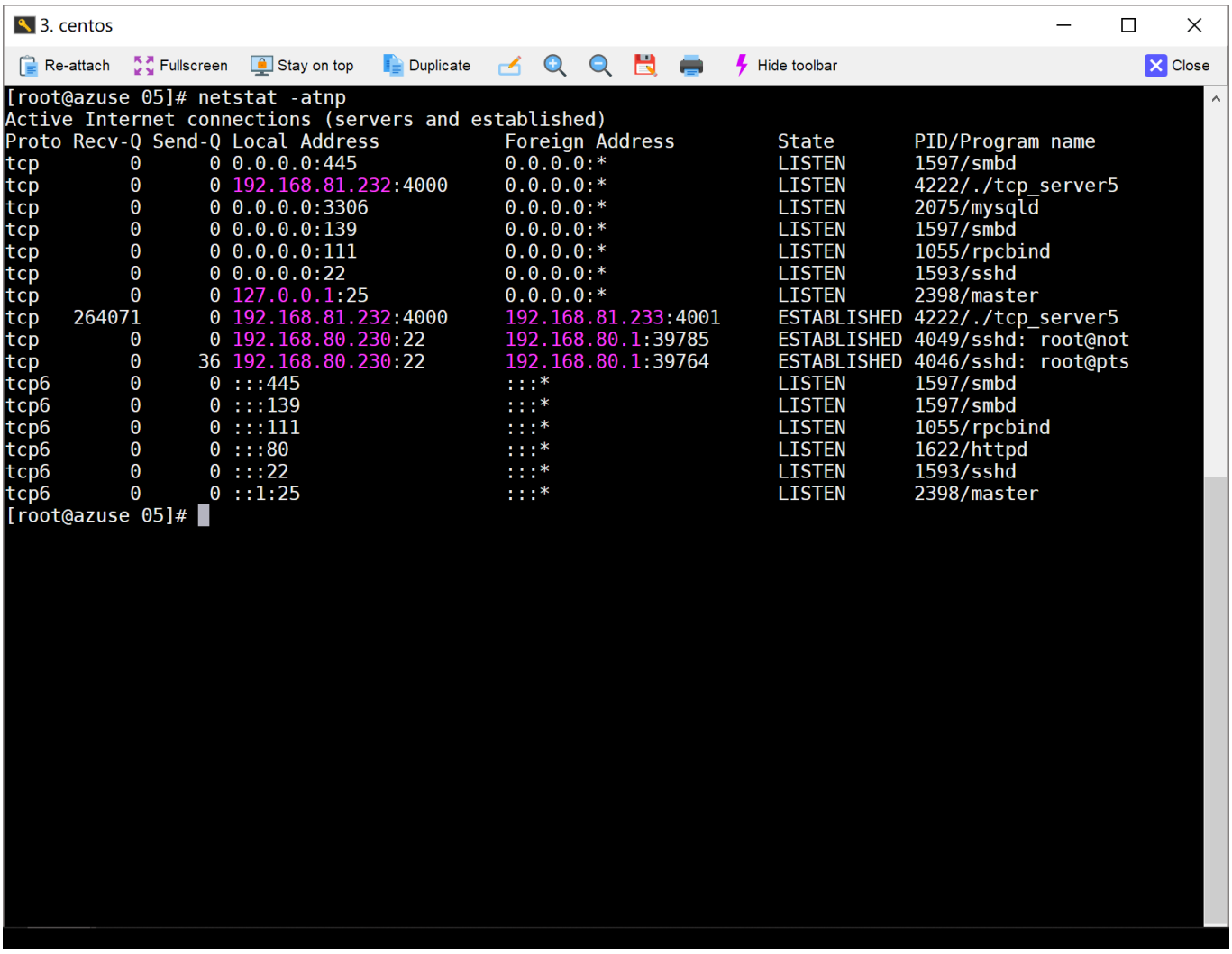
```
        else
        {
                long int *sendbytes_p = (long int *)shmat(shmid, NULL, 0);
                long int sendbytes_cached;
                while (1)
                {
                        sendbytes_cached = *sendbytes_p;
                        sleep(0.5);
                        if (*sendbytes_p == sendbytes_cached)
                        {
                                //printf("socket send stocked, send bytes: %ld,
cached\n", *sendbytes_p, sendbytes_cached);
                        }
                }
        }
```

写入多少字节后write函数阻塞

当子进程写入415140字节后，主进程检测到子进程阻塞。
此时client的senq为130320字节：
此时server的recq为264701字节：

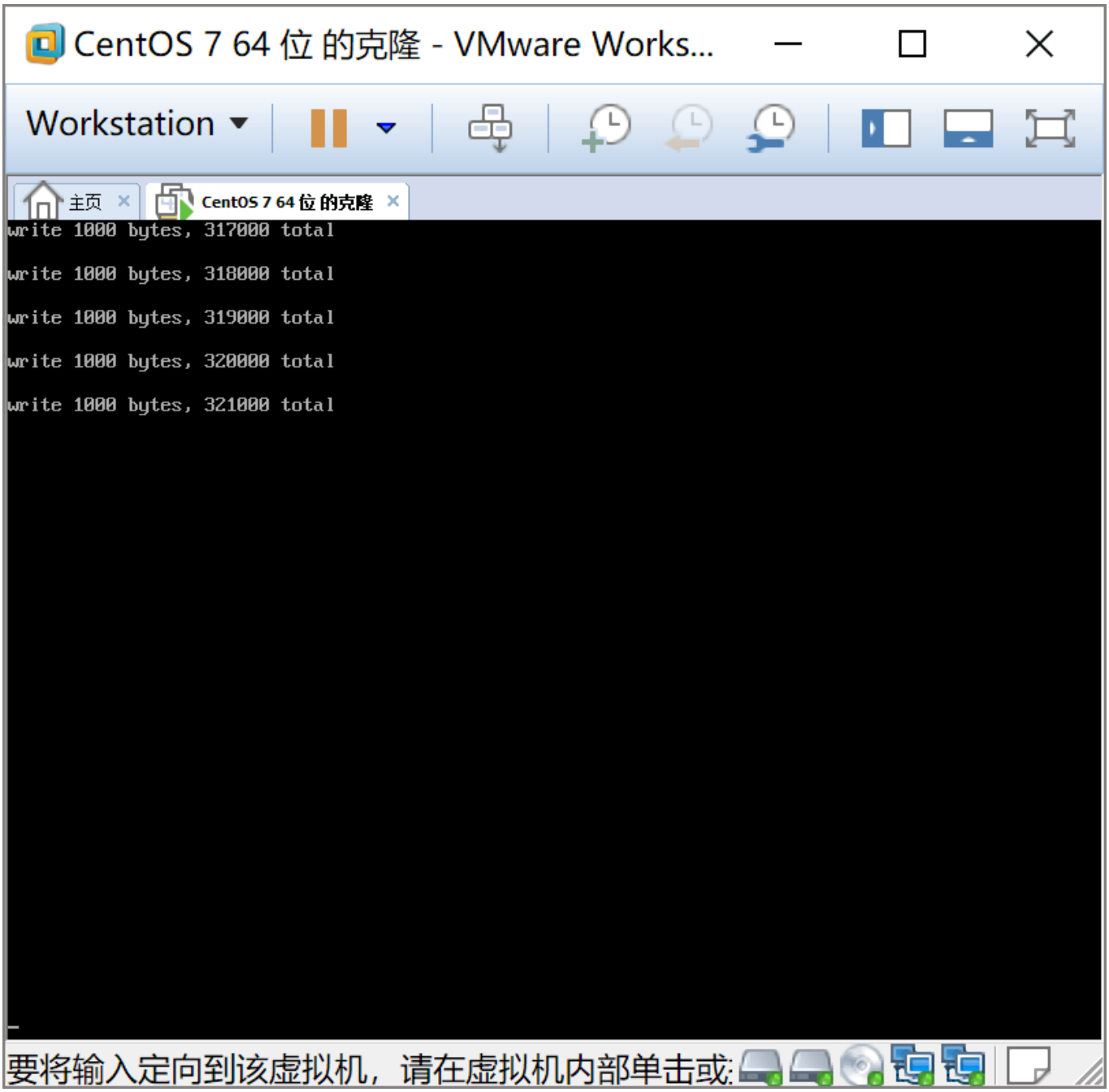

CentOS 7 64 位 的克隆 - VMware Works...
Workstation
主页
CentOS 7 64 位 的克隆
socket send stocked, send bytes: 415140, cached
要将输入定向到该虚拟机，请在虚拟机内部单击或


centos (1)
Terminal Sessions View X server Tools Games Settings Macros Help
Quick connect...
2. centos (1)
3. centos
Sessions
Tools
Macros
Sftp
[root@azuse 05]# netstat -atnp
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address Foreign Address State PID/Program name
tcp 0 0 0.0.0.0:445 0.0.0.0:* LISTEN 1373/smbd
tcp 0 0 0.0.0.0:3306 0.0.0.0:* LISTEN 1755/mysqld
tcp 0 0 0.0.0.0:139 0.0.0.0:* LISTEN 1373/smbd
tcp 0 0 0.0.0.0:111 0.0.0.0:* LISTEN 898/rpcbind
tcp 0 0 0.0.0.0:22 0.0.0.0:* LISTEN 1382/sshd
tcp 0 0 127.0.0.1:25 0.0.0.0:* LISTEN 1779/master
tcp 0 130320 192.168.81.233:4001 192.168.81.232:4000 ESTABLISHED 2379/./tcp_client5
tcp 0 0 192.168.80.231:22 192.168.80.1:39759 ESTABLISHED 2145/sshd: root@not
tcp 0 36 192.168.80.231:22 192.168.80.1:39733 ESTABLISHED 2143/sshd: root@pts
tcp6 0 0 :::445 :::* LISTEN 1373/smbd
tcp6 0 0 :::139 :::* LISTEN 1373/smbd
tcp6 0 0 :::111 :::* LISTEN 898/rpcbind
tcp6 0 0 :::80 :::* LISTEN 1374/httpd
tcp6 0 0 :::22 :::* LISTEN 1382/sshd
tcp6 0 0 ::1:25 :::* LISTEN 1779/master
[root@azuse 05]#
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: http://mobaxterm.mobatek.net

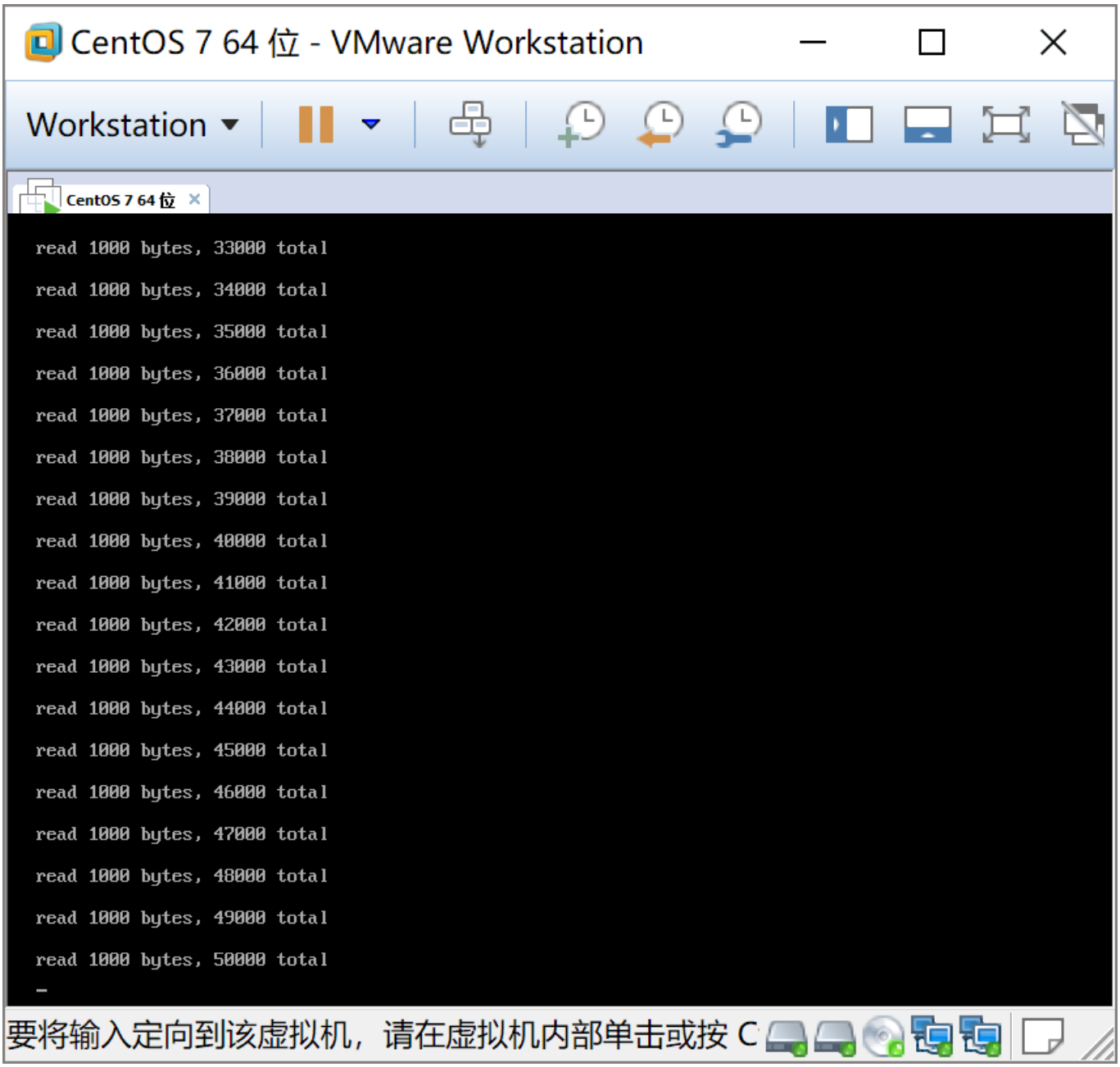


手动测试server读取多少自己后client会解除阻塞

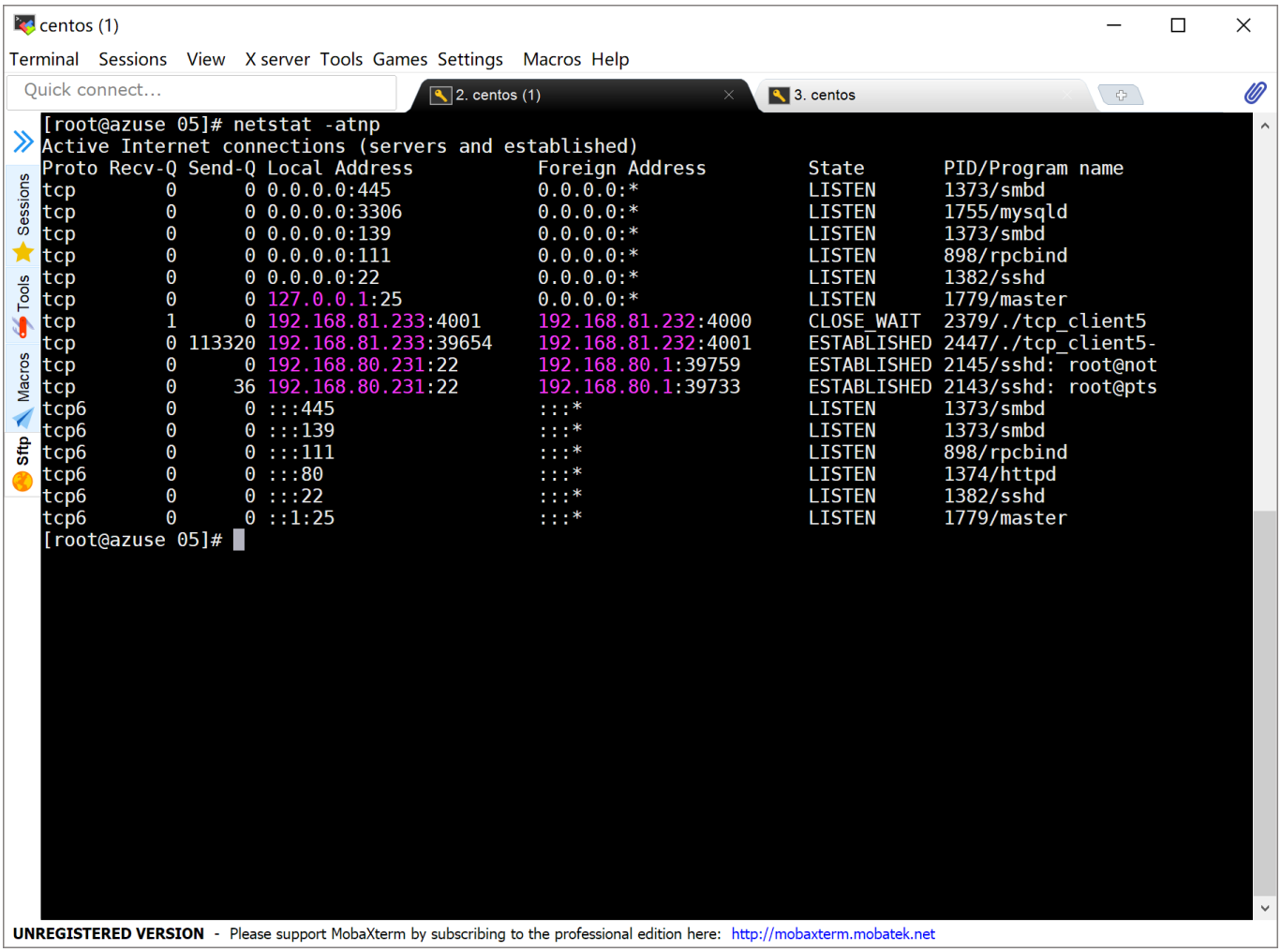
每次一个getchar()写1000个字节，写到大约32w字节后阻塞

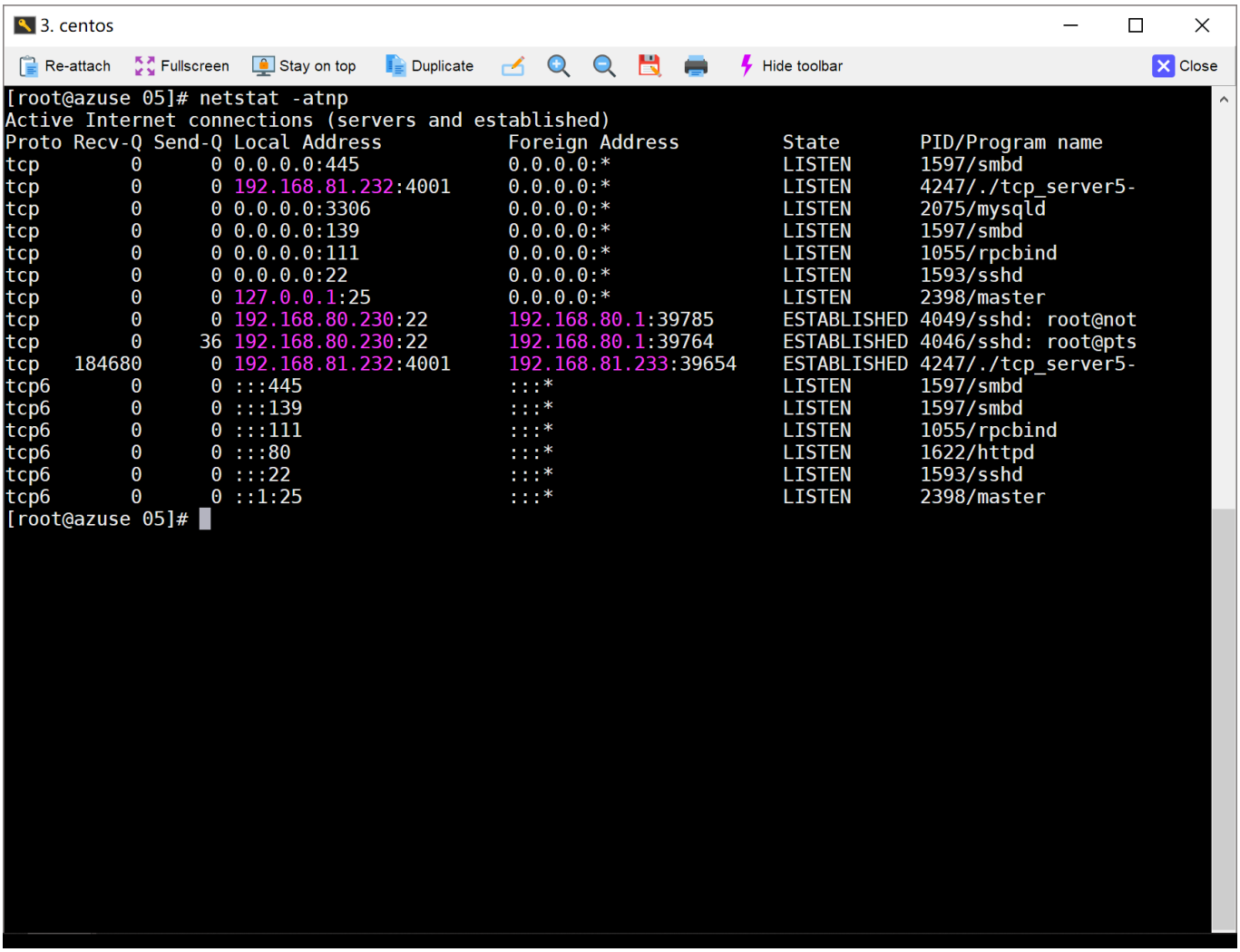


server端读取大约45千字节后，client解除阻塞

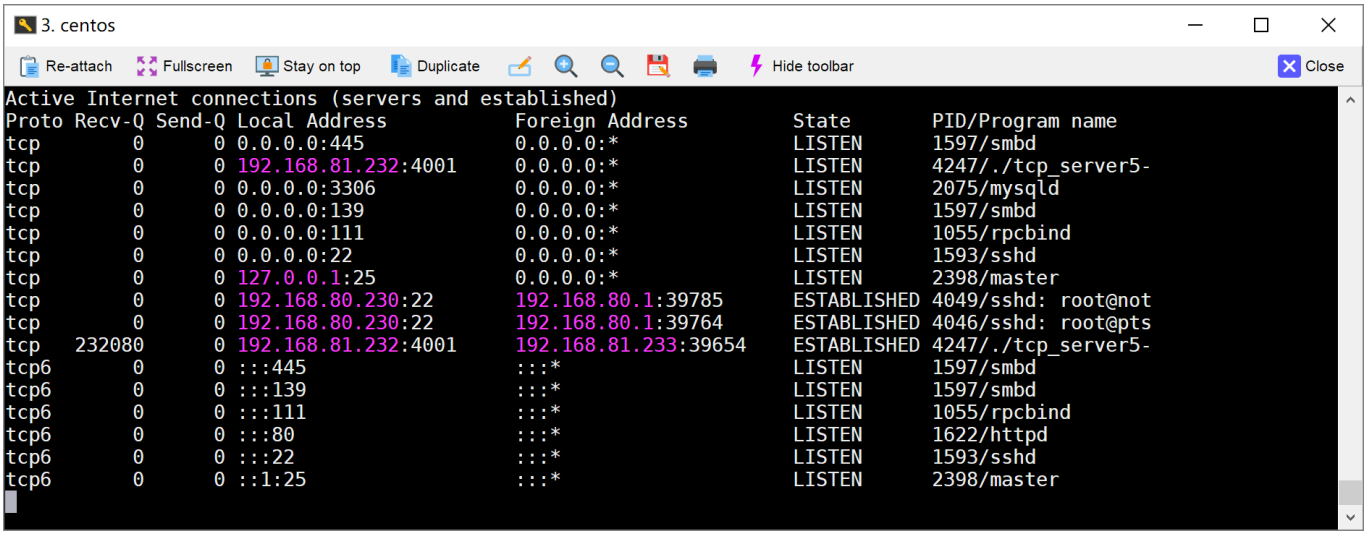


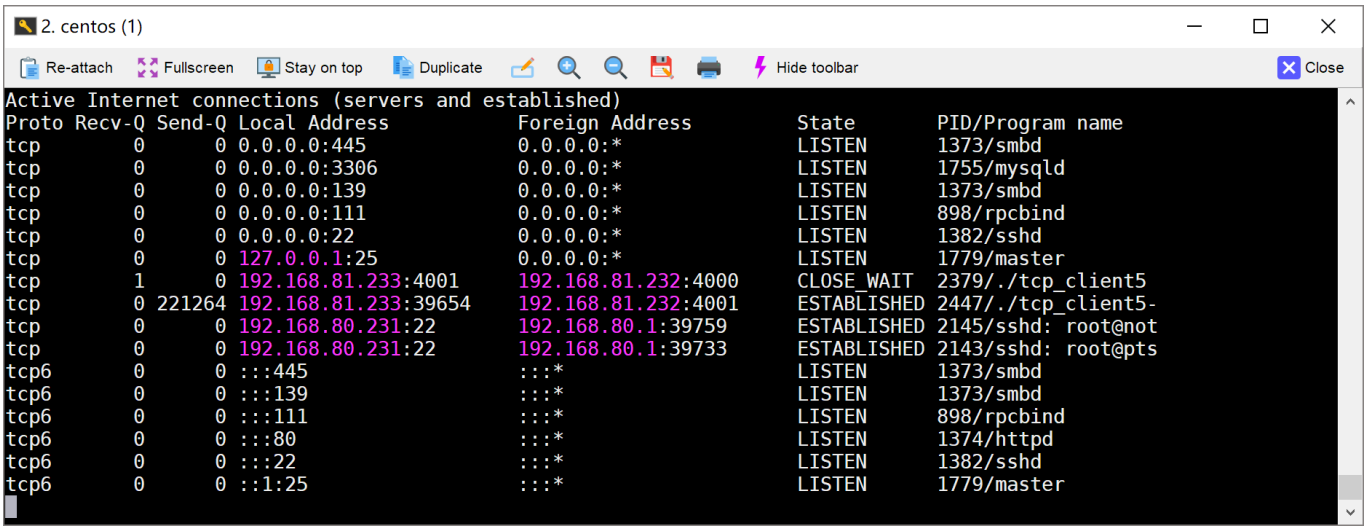
此时的sendq=113320, recvq=184680， sendq+recvq=298000：


centos (1)
Terminal Sessions View X server Tools Games Settings Macros Help
Quick connect...
2. centos (1)
3. centos
Sessions
Tools
Macros
Sftp
[root@azuse 05]# netstat -atnp
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address Foreign Address State PID/Program name
tcp 0 0 0.0.0.0:445 0.0.0.0:* LISTEN 1373/smbd
tcp 0 0 0.0.0.0:3306 0.0.0.0:* LISTEN 1755/mysqld
tcp 0 0 0.0.0.0:139 0.0.0.0:* LISTEN 1373/smbd
tcp 0 0 0.0.0.0:111 0.0.0.0:* LISTEN 898/rpcbind
tcp 0 0 0.0.0.0:22 0.0.0.0:* LISTEN 1382/sshd
tcp 0 0 127.0.0.1:25 0.0.0.0:* LISTEN 1779/master
tcp 1 0 192.168.81.233:4001 192.168.81.232:4000 CLOSE_WAIT 2379/./tcp_client5
tcp 0 113320 192.168.81.233:39654 192.168.81.232:4001 ESTABLISHED 2447/./tcp_client5-
tcp 0 0 192.168.80.231:22 192.168.80.1:39759 ESTABLISHED 2145/sshd: root@not
tcp 0 36 192.168.80.231:22 192.168.80.1:39733 ESTABLISHED 2143/sshd: root@pts
tcp6 0 0 :::445 :::* LISTEN 1373/smbd
tcp6 0 0 :::139 :::* LISTEN 1373/smbd
tcp6 0 0 :::111 :::* LISTEN 898/rpcbind
tcp6 0 0 :::80 :::* LISTEN 1374/httpd
tcp6 0 0 :::22 :::* LISTEN 1382/sshd
tcp6 0 0 ::1:25 :::* LISTEN 1779/master
[root@azuse 05]#
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: http://mobaxterm.mobatek.net


3. centos
Re-attach Fullscreen Stay on top Duplicate Hide toolbar Close
[root@azuse 05]# netstat -atnp
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address Foreign Address State PID/Program name
tcp 0 0 0.0.0.0:445 0.0.0.0:* LISTEN 1597/smbd
tcp 0 0 192.168.81.232:4001 0.0.0.0:* LISTEN 4247/./tcp_server5-
tcp 0 0 0.0.0.0:3306 0.0.0.0:* LISTEN 2075/mysqld
tcp 0 0 0.0.0.0:139 0.0.0.0:* LISTEN 1597/smbd
tcp 0 0 0.0.0.0:111 0.0.0.0:* LISTEN 1055/rpcbind
tcp 0 0 0.0.0.0:22 0.0.0.0:* LISTEN 1593/sshd
tcp 0 0 127.0.0.1:25 0.0.0.0:* LISTEN 2398/master
tcp 0 0 192.168.80.230:22 192.168.80.1:39785 ESTABLISHED 4049/sshd: root@not
tcp 0 36 192.168.80.230:22 192.168.80.1:39764 ESTABLISHED 4046/sshd: root@pts
tcp 184680 0 192.168.81.232:4001 192.168.81.233:39654 ESTABLISHED 4247/./tcp_server5-
tcp6 0 0 :::445 :::* LISTEN 1597/smbd
tcp6 0 0 :::139 :::* LISTEN 1597/smbd
tcp6 0 0 :::111 :::* LISTEN 1055/rpcbind
tcp6 0 0 :::80 :::* LISTEN 1622/httpd
tcp6 0 0 :::22 :::* LISTEN 1593/sshd
tcp6 0 0 ::1:25 :::* LISTEN 2398/master
[root@azuse 05]#

重复测试后发现，sendq与recvq的最大值在220000和230000左右

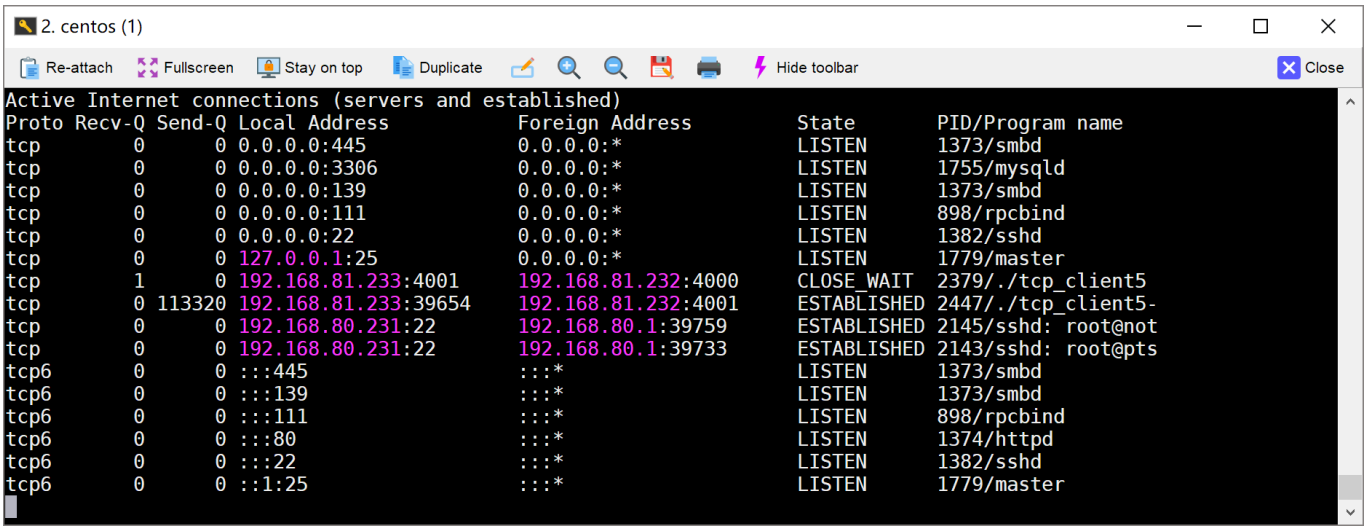




netstat可以带哪些参数？说明什么？

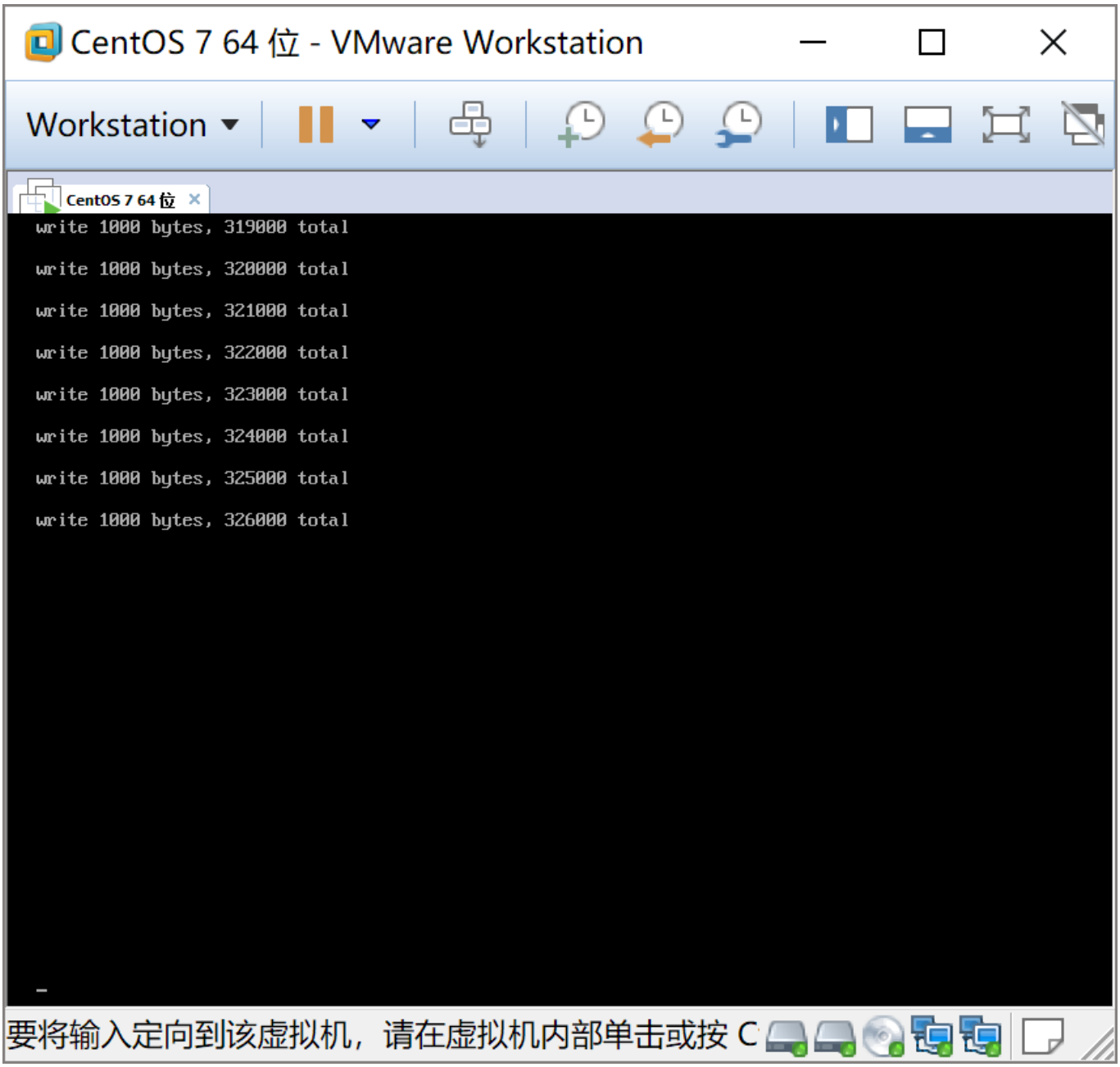
`-a` 列出所有端口
`-t` 只显示tcp `-u` 只显示udp `-x` 只显示unix端口 `-p` 显示进程和进程名称 `-n` 不显示别名，全部显示数字 `-c` 持续输出

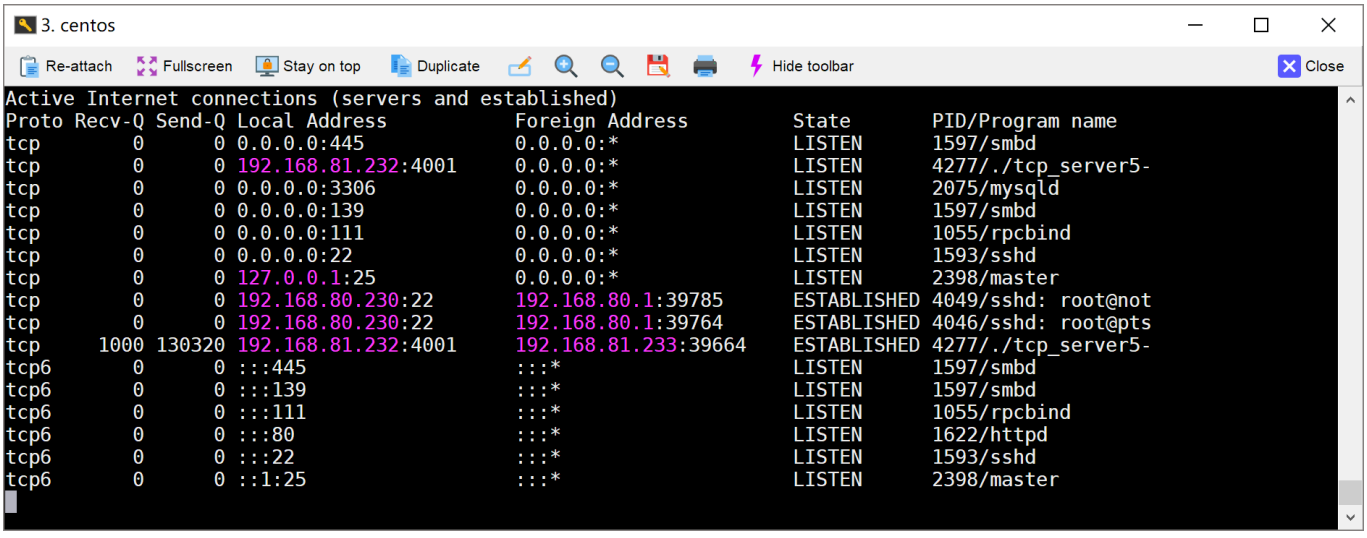
综上我检查sendq和recvq的netstat命令为：`netstat -atnpc`

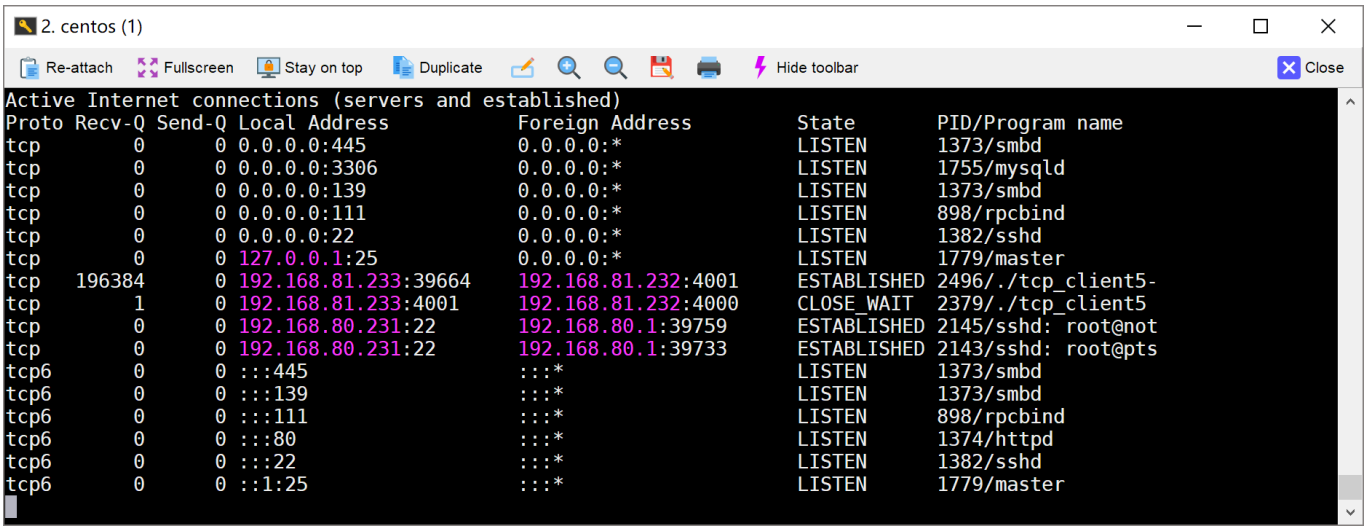


测试程序tcp_server2与tcp_client2，双方角色互换，server写，client读

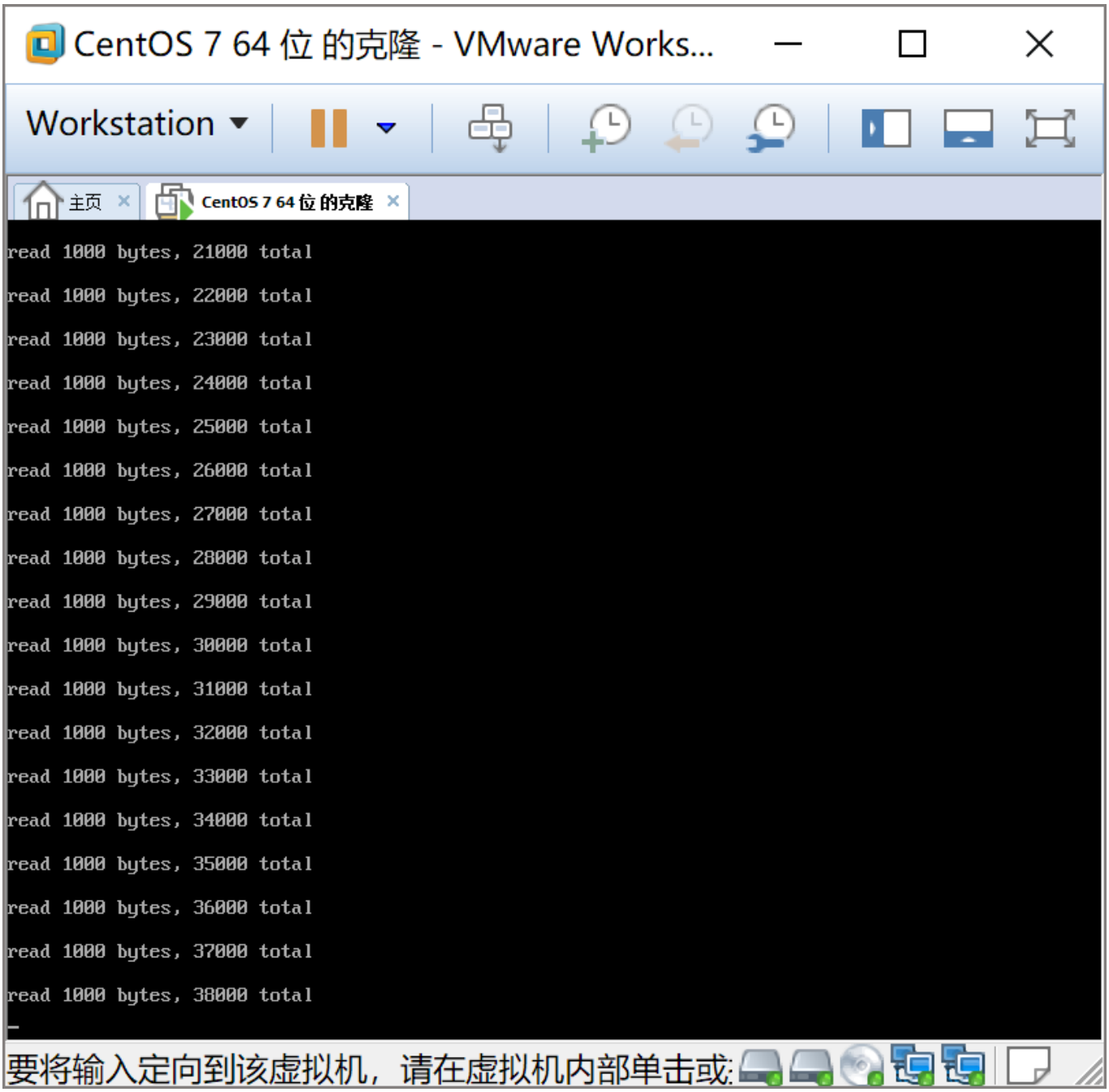
服务端不停的写入，发现到326000字节左右阻塞了，此时sendq=130320, recvq=196384

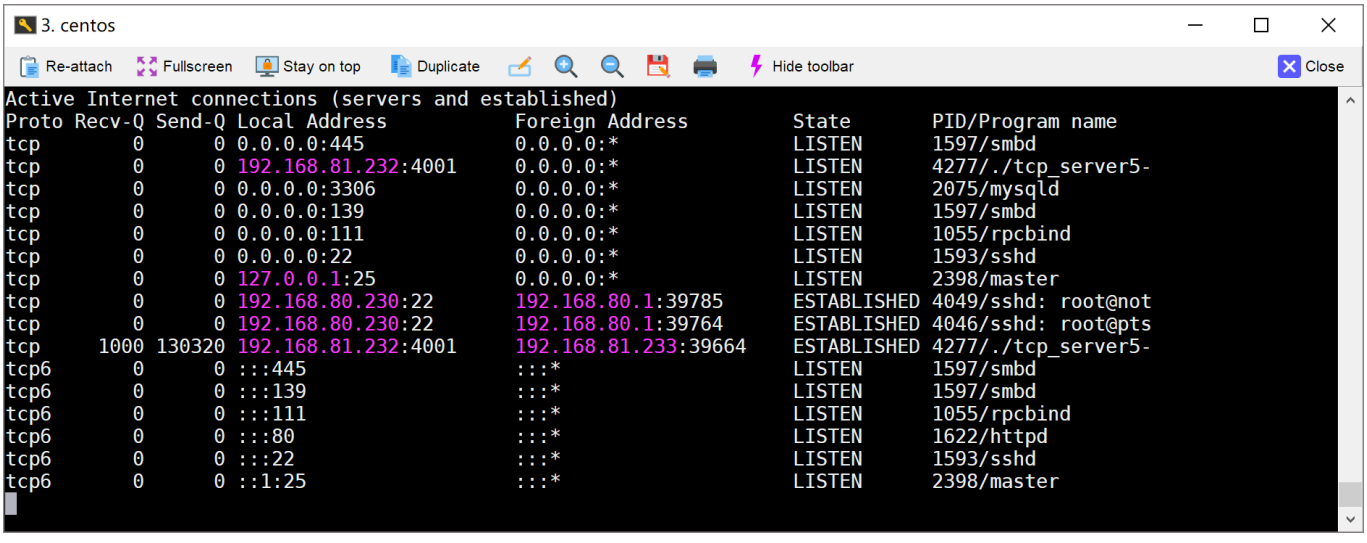

CentOS 7 64 位 - VMware Workstation
Workstation
CentOS 7 64 位
write 1000 bytes, 319000 total
write 1000 bytes, 320000 total
write 1000 bytes, 321000 total
write 1000 bytes, 322000 total
write 1000 bytes, 323000 total
write 1000 bytes, 324000 total
write 1000 bytes, 325000 total
write 1000 bytes, 326000 total
要将输入定向到该虚拟机，请在虚拟机内部单击或按 C

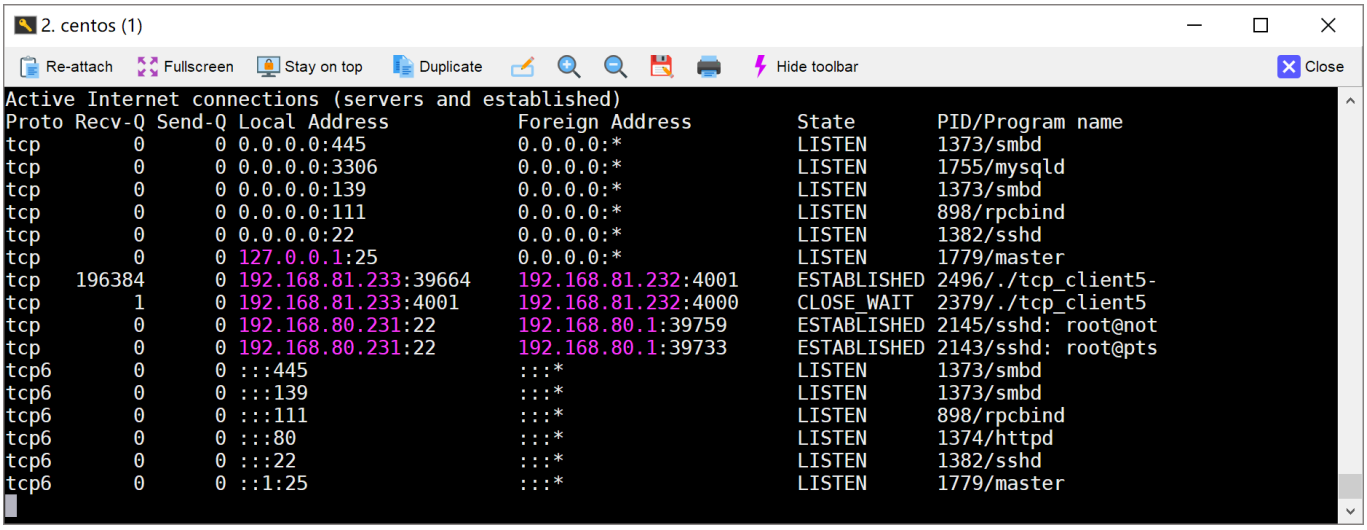

3. centos
Re-attach
Fullscreen
Stay on top
Duplicate
Hide toolbar
Close
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address Foreign Address State PID/Program name
tcp 0 0 0.0.0.0:445 0.0.0.0:* LISTEN 1597/smbd
tcp 0 0 192.168.81.232:4001 0.0.0.0:* LISTEN 4277/./tcp_server5-
tcp 0 0 0.0.0.0:3306 0.0.0.0:* LISTEN 2075/mysqld
tcp 0 0 0.0.0.0:139 0.0.0.0:* LISTEN 1597/smbd
tcp 0 0 0.0.0.0:111 0.0.0.0:* LISTEN 1055/rpcbind
tcp 0 0 0.0.0.0:22 0.0.0.0:* LISTEN 1593/sshd
tcp 0 0 127.0.0.1:25 0.0.0.0:* LISTEN 2398/master
tcp 0 0 192.168.80.230:22 192.168.80.1:39785 ESTABLISHED 4049/sshd: root@not
tcp 0 0 192.168.80.230:22 192.168.80.1:39764 ESTABLISHED 4046/sshd: root@pts
tcp 1000 130320 192.168.81.232:4001 192.168.81.233:39664 ESTABLISHED 4277/./tcp_server5-
tcp6 0 0 :::445 :::* LISTEN 1597/smbd
tcp6 0 0 :::139 :::* LISTEN 1597/smbd
tcp6 0 0 :::111 :::* LISTEN 1055/rpcbind
tcp6 0 0 :::80 :::* LISTEN 1622/httpd
tcp6 0 0 :::22 :::* LISTEN 1593/sshd
tcp6 0 0 ::1:25 :::* LISTEN 2398/master



客户端开始read，发现read到35000左右，服务端的阻塞解除了，此时sendq=109768, recvq=201232


CentOS 7 64 位 的克隆 - VMware Works...
Workstation
主页
CentOS 7 64 位 的克隆
read 1000 bytes, 21000 total
read 1000 bytes, 22000 total
read 1000 bytes, 23000 total
read 1000 bytes, 24000 total
read 1000 bytes, 25000 total
read 1000 bytes, 26000 total
read 1000 bytes, 27000 total
read 1000 bytes, 28000 total
read 1000 bytes, 29000 total
read 1000 bytes, 30000 total
read 1000 bytes, 31000 total
read 1000 bytes, 32000 total
read 1000 bytes, 33000 total
read 1000 bytes, 34000 total
read 1000 bytes, 35000 total
read 1000 bytes, 36000 total
read 1000 bytes, 37000 total
read 1000 bytes, 38000 total
要将输入定向到该虚拟机，请在虚拟机内部单击或


3. centos
Re-attach
Fullscreen
Stay on top
Duplicate
Hide toolbar
Close
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address Foreign Address State PID/Program name
tcp 0 0 0.0.0.0:445 0.0.0.0:* LISTEN 1597/smbd
tcp 0 0 192.168.81.232:4001 0.0.0.0:* LISTEN 4277/./tcp_server5-
tcp 0 0 0.0.0.0:3306 0.0.0.0:* LISTEN 2075/mysqld
tcp 0 0 0.0.0.0:139 0.0.0.0:* LISTEN 1597/smbd
tcp 0 0 0.0.0.0:111 0.0.0.0:* LISTEN 1055/rpcbind
tcp 0 0 0.0.0.0:22 0.0.0.0:* LISTEN 1593/sshd
tcp 0 0 127.0.0.1:25 0.0.0.0:* LISTEN 2398/master
tcp 0 0 192.168.80.230:22 192.168.80.1:39785 ESTABLISHED 4049/sshd: root@not
tcp 0 0 192.168.80.230:22 192.168.80.1:39764 ESTABLISHED 4046/sshd: root@pts
tcp 1000 130320 192.168.81.232:4001 192.168.81.233:39664 ESTABLISHED 4277/./tcp_server5-
tcp6 0 0 :::445 :::* LISTEN 1597/smbd
tcp6 0 0 :::139 :::* LISTEN 1597/smbd
tcp6 0 0 :::111 :::* LISTEN 1055/rpcbind
tcp6 0 0 :::80 :::* LISTEN 1622/httpd
tcp6 0 0 :::22 :::* LISTEN 1593/sshd
tcp6 0 0 ::1:25 :::* LISTEN 2398/master



> SendQ 和 RecvQ 缓冲队列，这两个缓冲区的容量在具体实现时会受一定的限制，虽然它们使用的实际内存大小会动态地增长和收缩，但还是需要一个硬性的限制，以防止行为异常的程序所控制的单一 TCP 连接将系统的内存全部消耗。正式由于缓冲区的容量有限，它们可能会被填满，事实也正是如此，如果与 TCP 的流量控制机制结合使用，则可能导致一种形式的死锁。

一旦 RecvQ 已满，TCP 流控制机制就会产生作用（使用流控制机制的目的是为了保证发送者不会传输太多数据，从而超出了接收系统的处理能力），它将阻止传输发送端主机的 SendQ 中的任何数据，直到接收者调用输入流的 read()方法将 RecvQ 中的数据移除一部分到 Delivered 中，从而腾出了空间。发送端可以持续地写出数据，直到 SendQ 队列被填满，如果 SendQ 队列已满时调用输出流的 write()方法，则会阻塞等待，直到有一些字节被传输到 RecvQ 队列中，如果此时 RecvQ 队列也被填满了，所有的操作都将停止，直到接收端调用了输入流的 read()方法将一些字节传输到了 Delivered 队列中。

tcp_server3 改变tcp收发缓冲区大小，观察过程

用setsockopt的SO_RCVBUF和SO_SNDBUF设置收发缓冲区大小，用getsockopt检查设置是否生效：
server端设置RCVBUF:

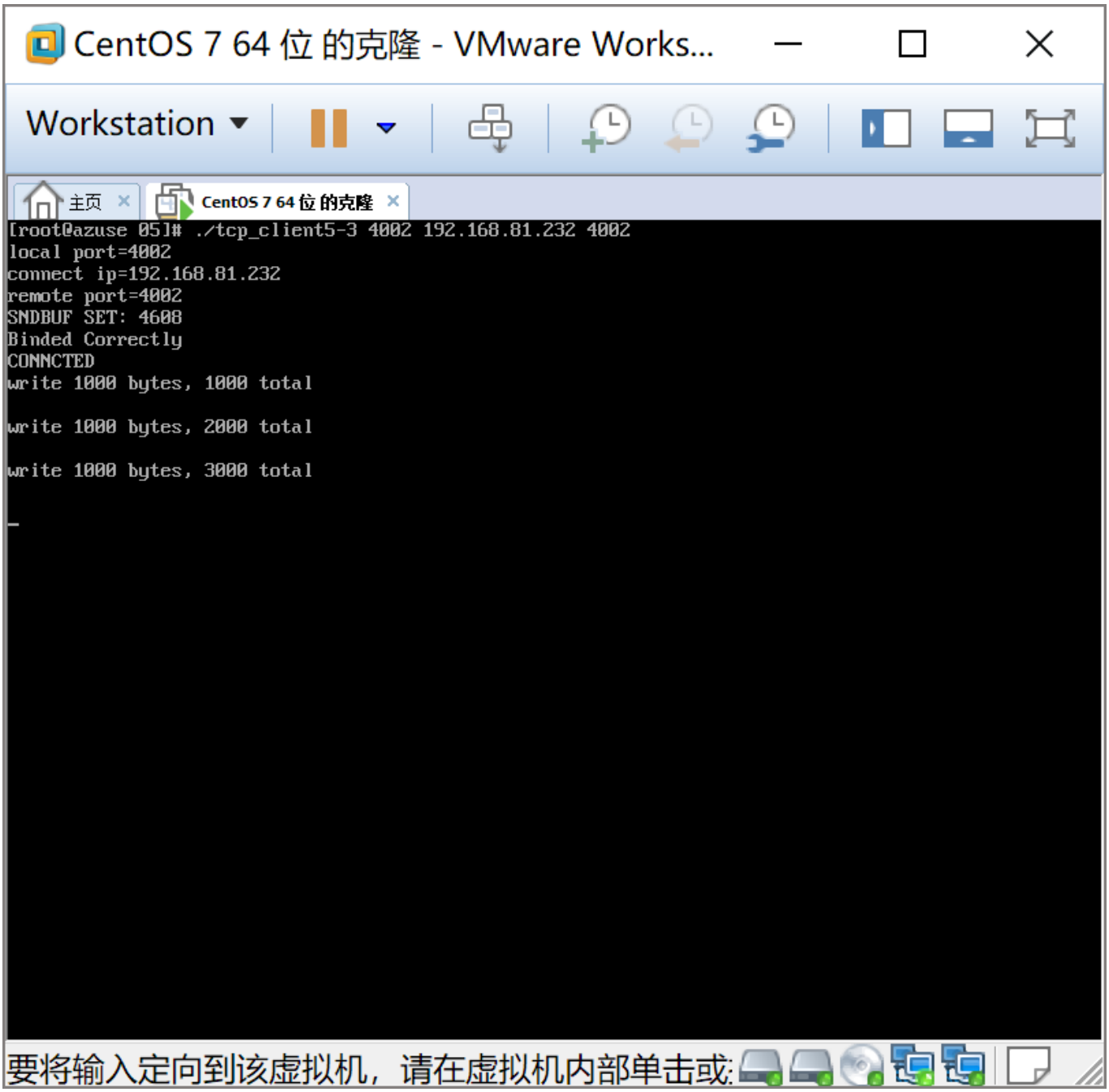
```
        int opt = 2000;
        if (setsockopt(sock, SOL_SOCKET, SO_SNDBUF, &opt, sizeof(opt)))
        {
                perror("setsockopt");
                exit(EXIT_FAILURE);
        }
        else
        {
                int rc;
                int optval;
                int optlen;
                struct linger l;
                optlen = sizeof(int);
                rc = getsockopt(sock, SOL_SOCKET, SO_SNDBUF, (char *)&optval,
&optlen);
                if (rc == 0)
                {
                        if (optlen == sizeof(int))
                        {
```

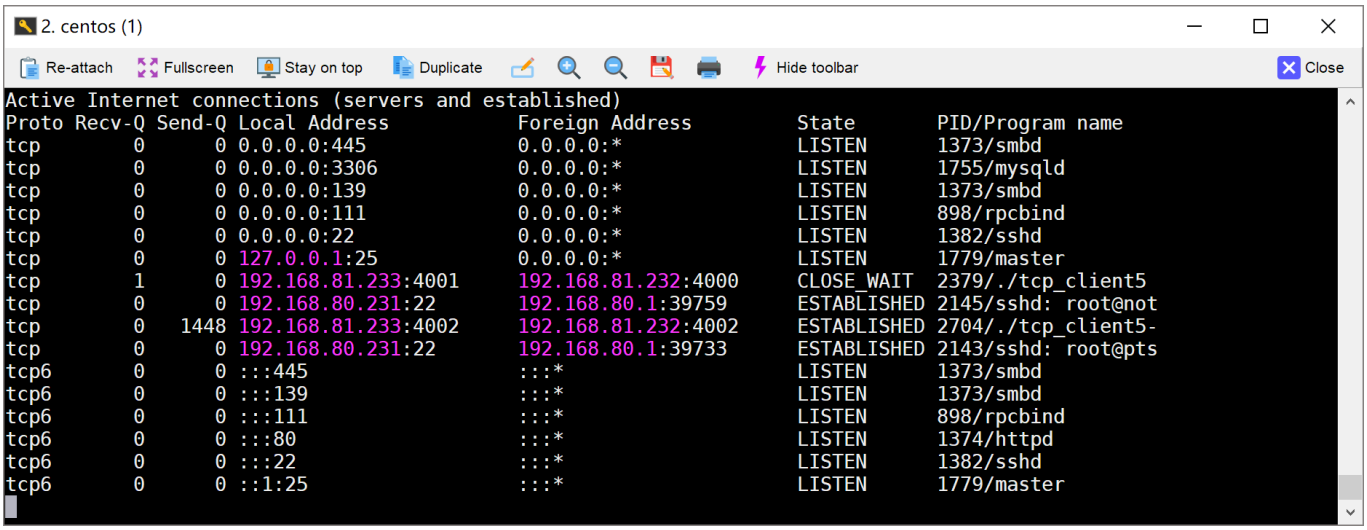
```
                                printf("SNDBUF SET: %d\n", optval);
                        }
                }
        }
```

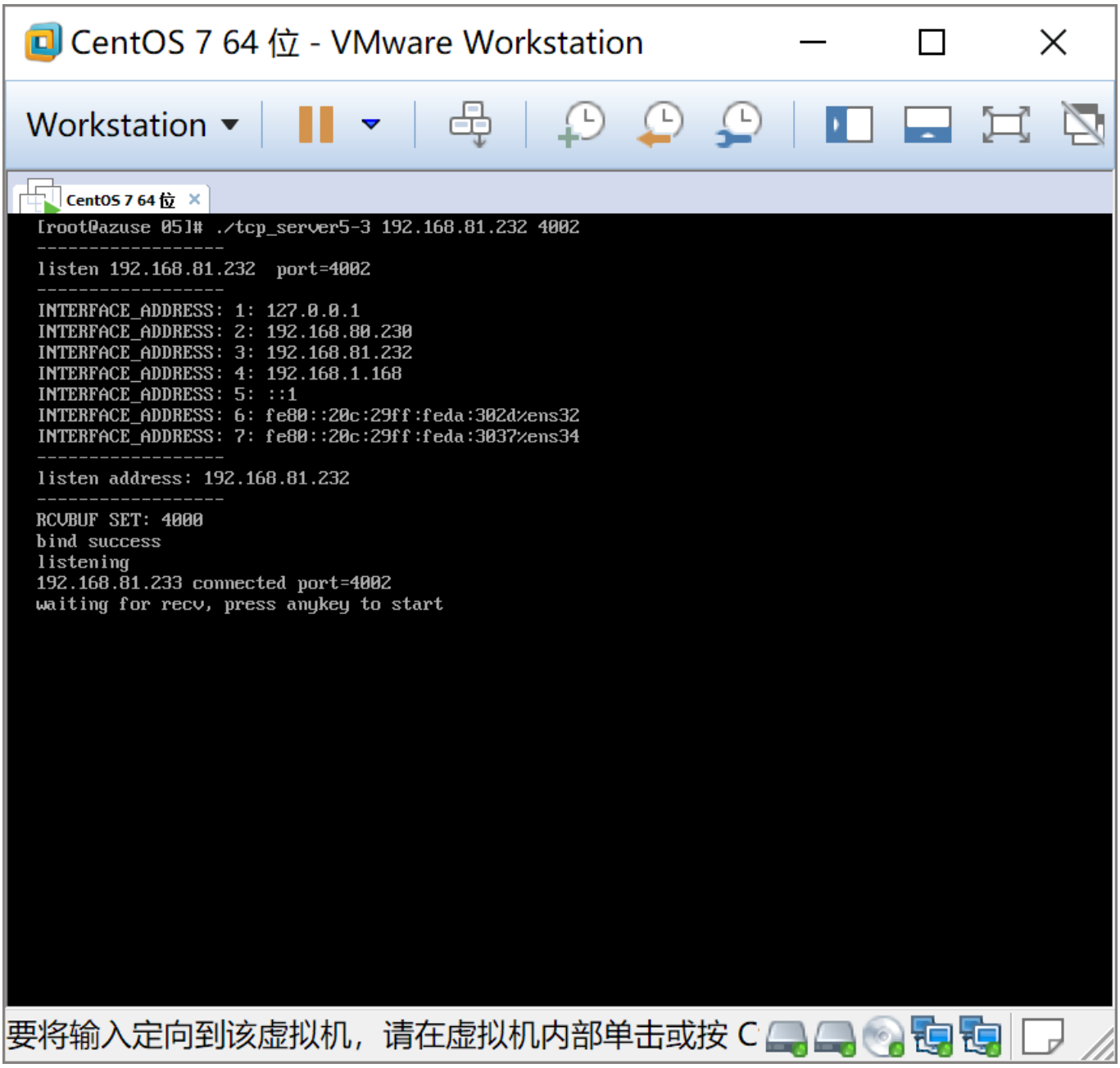
client端设置SNDBUF：

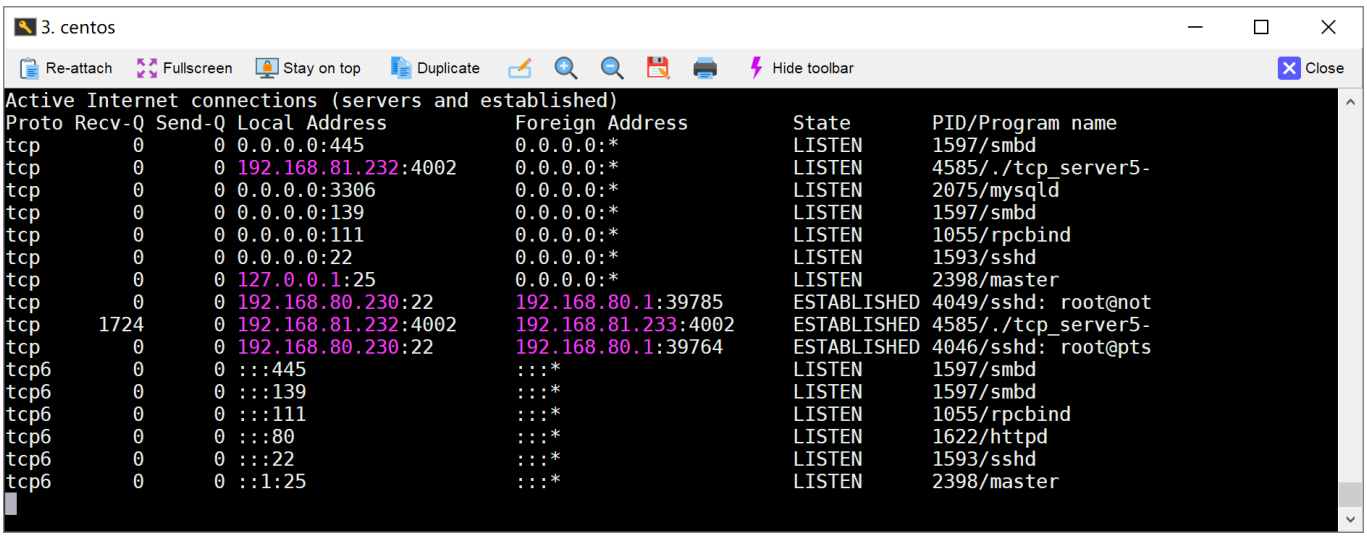
```
        int opt = 2000;
        if (setsockopt(sock, SOL_SOCKET, SO_SNDBUF, &opt, sizeof(opt)))
        {
                perror("setsockopt");
                exit(EXIT_FAILURE);
        }
        else
        {
                int rc;
                int optval;
                int optlen;
                struct linger l;
                optlen = sizeof(int);
                rc = getsockopt(sock, SOL_SOCKET, SO_SNDBUF, (char *)&optval,
&optlen);
                if (rc == 0)
                {
                        if (optlen == sizeof(int))
                        {
                                printf("SNDBUF SET: %d\n", optval);
                        }
                }
        }
```

设置收发缓冲区大小都是2000，但是用getsockopt获得实际大小其实一直是翻倍的，设置完后客户端可以正常写完前3000字节，当写入之后1000字节时阻塞，此时senq=1448, recvq=1724, sendq+recvq=3172


CentOS 7 64 位 的克隆 - VMware Works...
Workstation
主页
CentOS 7 64 位 的克隆
[root@azuse 05]# ./tcp_client5-3 4002 192.168.81.232 4002
local port=4002
connect ip=192.168.81.232
remote port=4002
SNDBUF SET: 4608
Binded Correctly
CONNCTED
write 1000 bytes, 1000 total
write 1000 bytes, 2000 total
write 1000 bytes, 3000 total
要将输入定向到该虚拟机，请在虚拟机内部单击或


2. centos (1)
Re-attach
Fullscreen
Stay on top
Duplicate
Hide toolbar
Close
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address Foreign Address State PID/Program name
tcp 0 0 0.0.0.0:445 0.0.0.0:* LISTEN 1373/smbd
tcp 0 0 0.0.0.0:3306 0.0.0.0:* LISTEN 1755/mysqld
tcp 0 0 0.0.0.0:139 0.0.0.0:* LISTEN 1373/smbd
tcp 0 0 0.0.0.0:111 0.0.0.0:* LISTEN 898/rpcbind
tcp 0 0 0.0.0.0:22 0.0.0.0:* LISTEN 1382/sshd
tcp 0 0 127.0.0.1:25 0.0.0.0:* LISTEN 1779/master
tcp 1 0 192.168.81.233:4001 192.168.81.232:4000 CLOSE_WAIT 2379/./tcp_client5
tcp 0 0 192.168.80.231:22 192.168.80.1:39759 ESTABLISHED 2145/sshd: root@not
tcp 0 1448 192.168.81.233:4002 192.168.81.232:4002 ESTABLISHED 2704/./tcp_client5-
tcp 0 0 192.168.80.231:22 192.168.80.1:39733 ESTABLISHED 2143/sshd: root@pts
tcp6 0 0 :::445 :::* LISTEN 1373/smbd
tcp6 0 0 :::139 :::* LISTEN 1373/smbd
tcp6 0 0 :::111 :::* LISTEN 898/rpcbind
tcp6 0 0 :::80 :::* LISTEN 1374/httpd
tcp6 0 0 :::22 :::* LISTEN 1382/sshd
tcp6 0 0 ::1:25 :::* LISTEN 1779/master





7 双方同时read/write测试

tcp6-1 server和client都先read

tcp_server6-1死循环部分：

```
                while(1)
                {
                        int tmp;
                        tmp = read(new_socket, readbuffer, readlength);
                        if(tmp > 0)
                        {
                                readbytes += tmp;
                                printf("read %d bytes, %d total\n", tmp,
readbytes);

                        }
                        else if(tmp == 0)
                                printf("read fail, connection closed\n");
                        else
                                printf("read error\n");
                        getchar();
                        tmp = write(new_socket, writebuffer, writelength);
                        if(tmp > 0)
                        {
                                writebytes += tmp;
                                printf("write %d bytes, %d total\n", tmp,
writebytes);

                        }
                        else
                                printf("write error\n");
                        getchar();
                }
```

tcp_client6-1死循环部分：

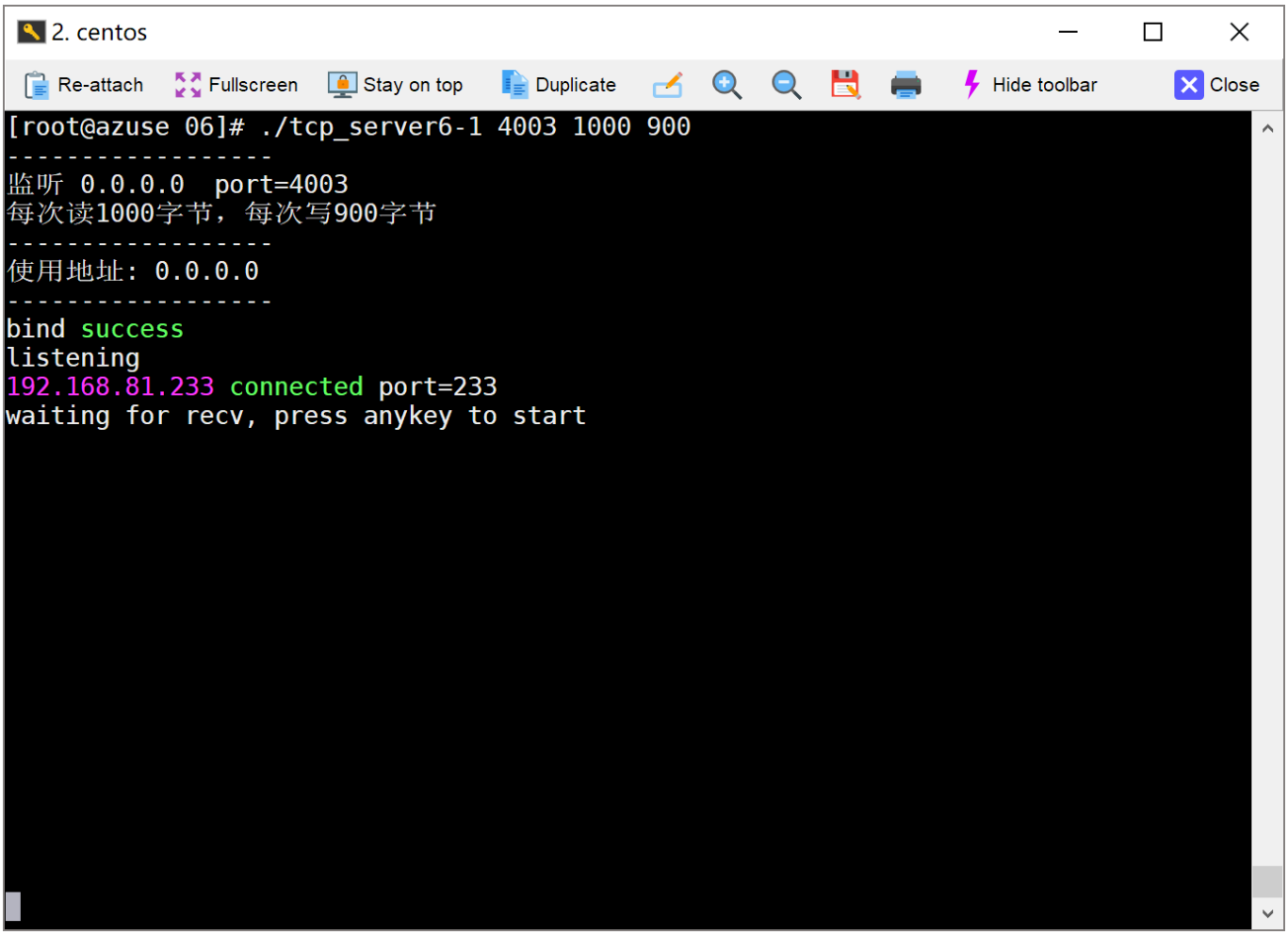
```
                while (1)
                {
                        int tmp;
                        tmp = read(sock, read_buffer, readlength);
                        if (tmp > 0)
                        {
                                *readbytes_p += tmp;
                                printf("read %d bytes, %d total\n", tmp,
*readbytes_p);
                        }
                        else if (tmp == 0)
                                printf("read fail, connection closed\n");
                        else
                                printf("read error\n");
                        getchar();
                        tmp = write(sock, send_buffer, writelength);
                        if (tmp > 0)
```

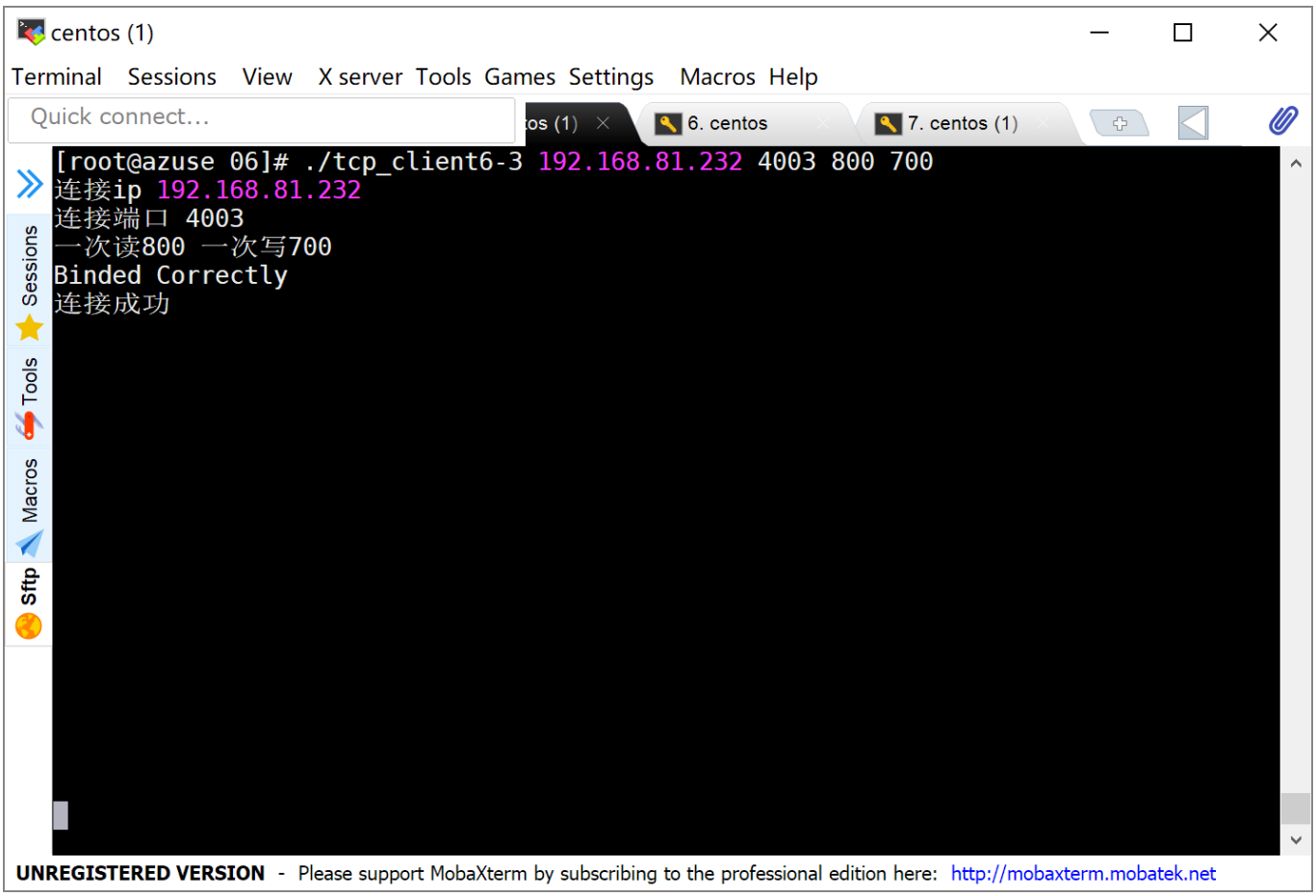
```
                        {
                                *sendbytes_p += tmp;
                                printf("write %d bytes, %d total\n", tmp,
*sendbytes_p);
                        }
                        else
                                printf("write error\n");
                        getchar();
                        //sleep(1);
                }
```

server开始运行后，启动client，正常连接，但无法正常发送数据。client端进入read阻塞等待来自server的信息，server也进入read阻塞等待来自client的信息，形成死锁。
当一方进程终止后，另一方read返回0代表连接关闭，阻塞解除。

server每次read1000字节 write900字节，client每次read800字节 write700字节

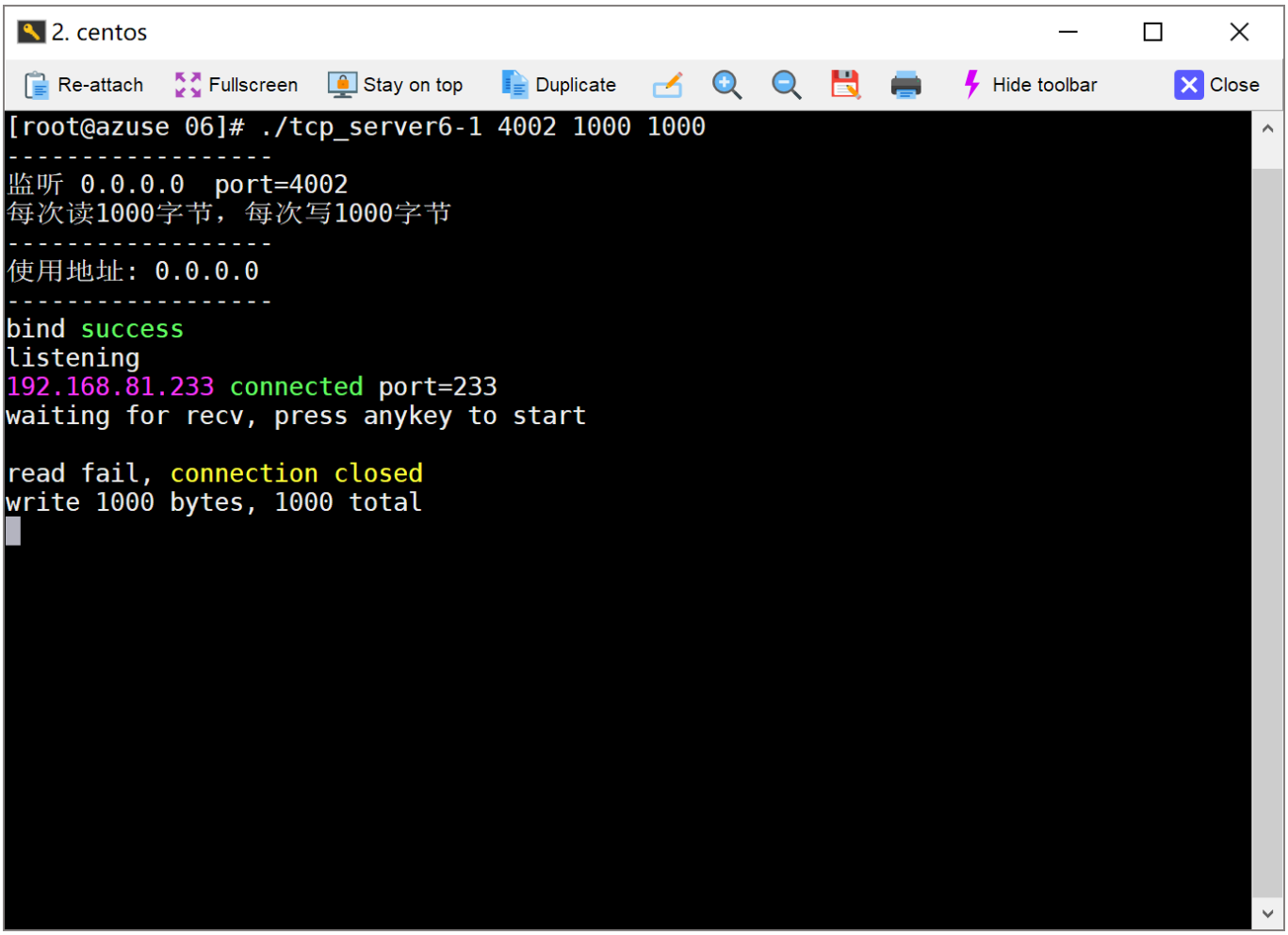
死锁：





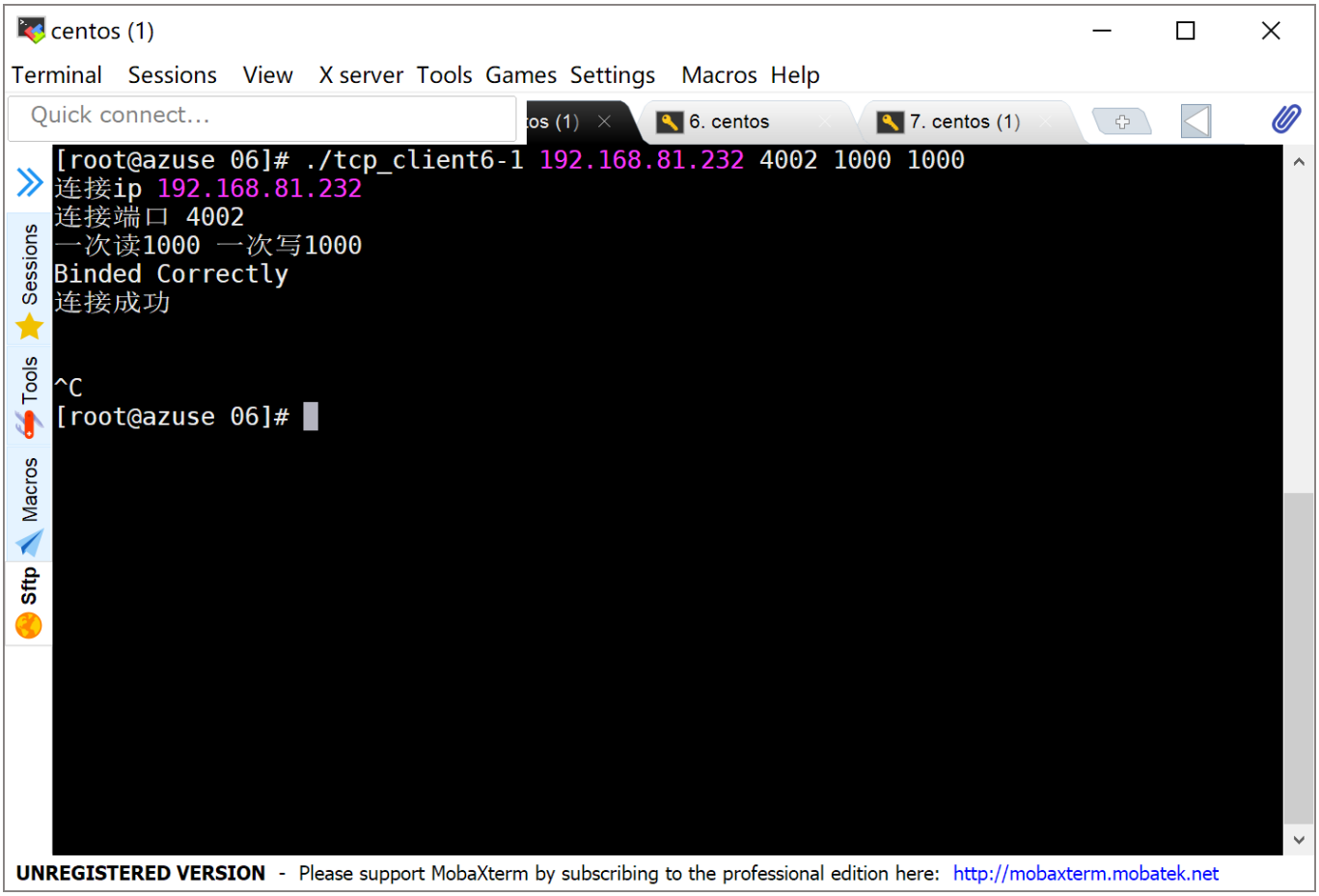
server每次write/read 1000字节，client每次write/read 1000字节

死锁:


2. centos
Re-attach
Fullscreen
Stay on top
Duplicate
Hide toolbar
Close
[root@azuse 06]# ./tcp_server6-1 4002 1000 1000

监听 0.0.0.0 port=4002
每次读1000字节，每次写1000字节

使用地址: 0.0.0.0
----
bind success
listening
192.168.81.233 connected port=233
waiting for recv, press anykey to start
read fail, connection closed
write 1000 bytes, 1000 total

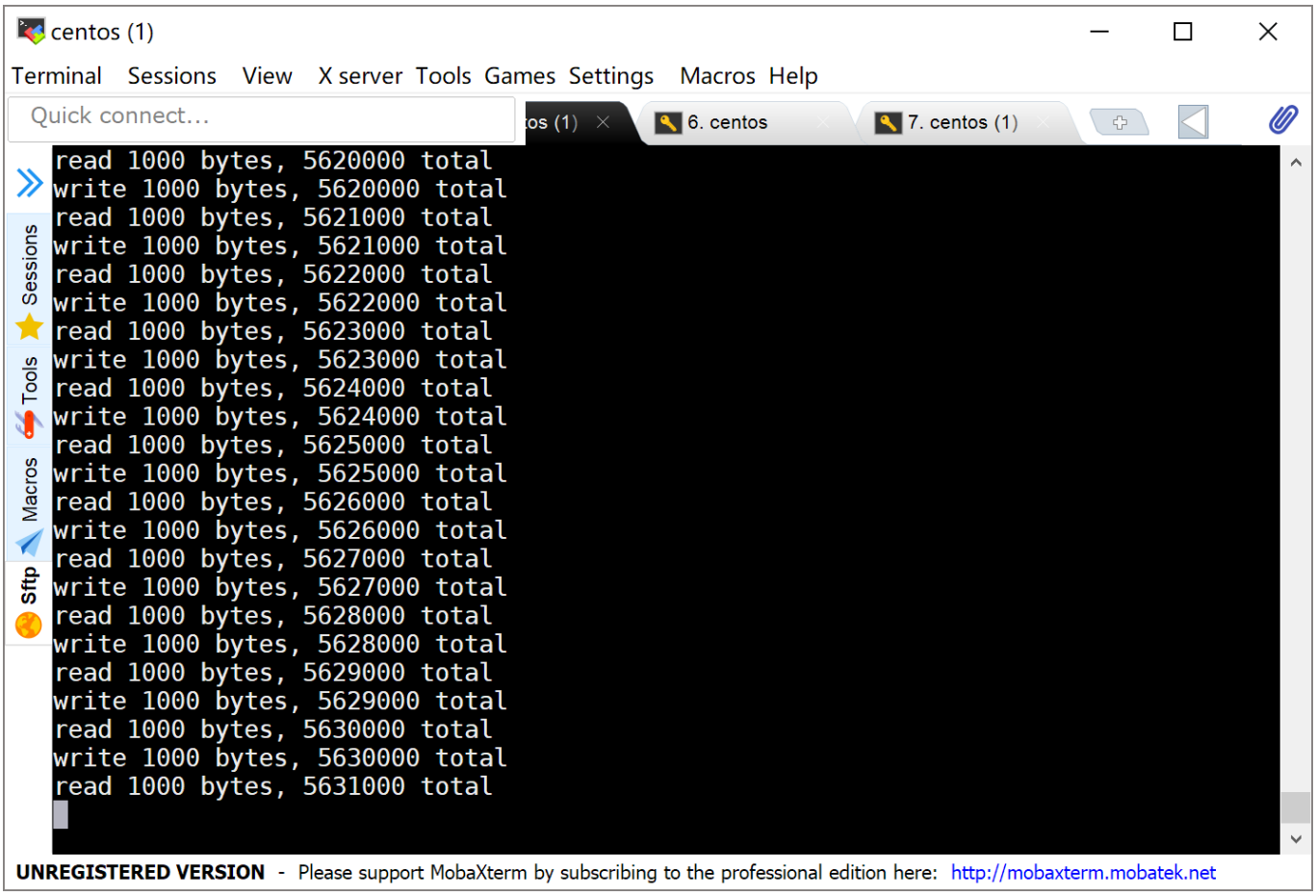

centos (1)
Terminal Sessions View X server Tools Games Settings Macros Help
Quick connect...
6. centos
7. centos (1)
[root@azuse 06]# ./tcp_client6-1 192.168.81.232 4002 1000 1000
连接ip 192.168.81.232
连接端口 4002
一次读1000 一次写1000
Binded Correctly
连接成功
^C
[root@azuse 06]#
Sessions
Tools
Macros
Sftp
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: http://mobaxterm.mobatek.net

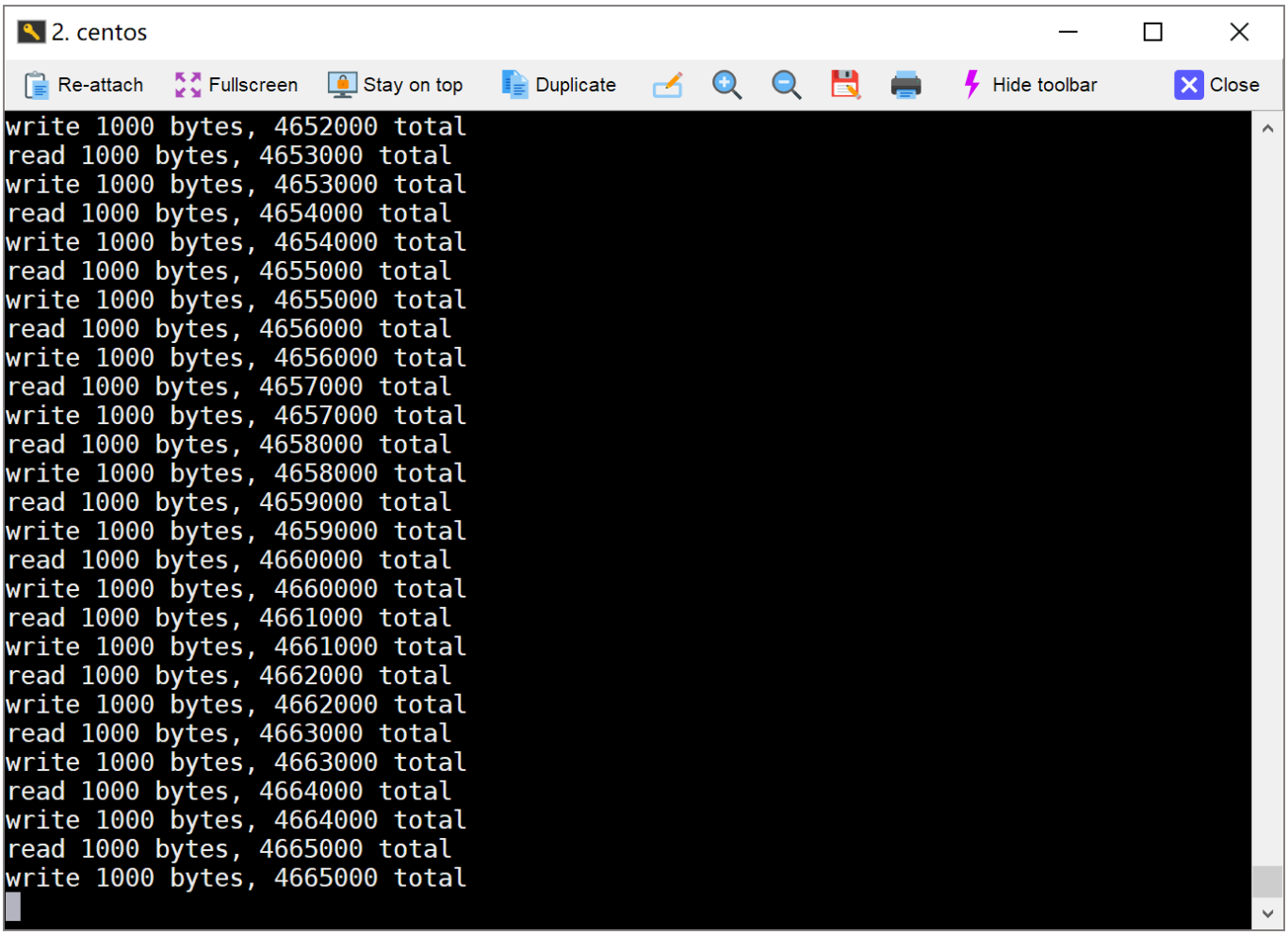
tcp6-2 server先write，client先write

调换上题中write与read的位置，发先双方可以不停的write/read直到手动终止。
因为write的数据可以先放到recvq中等待程序read，不会造成程序阻塞，程序继续运行到read就可以正常读出数据

server每次write/read 1000字节，client每次write/read 1000字节

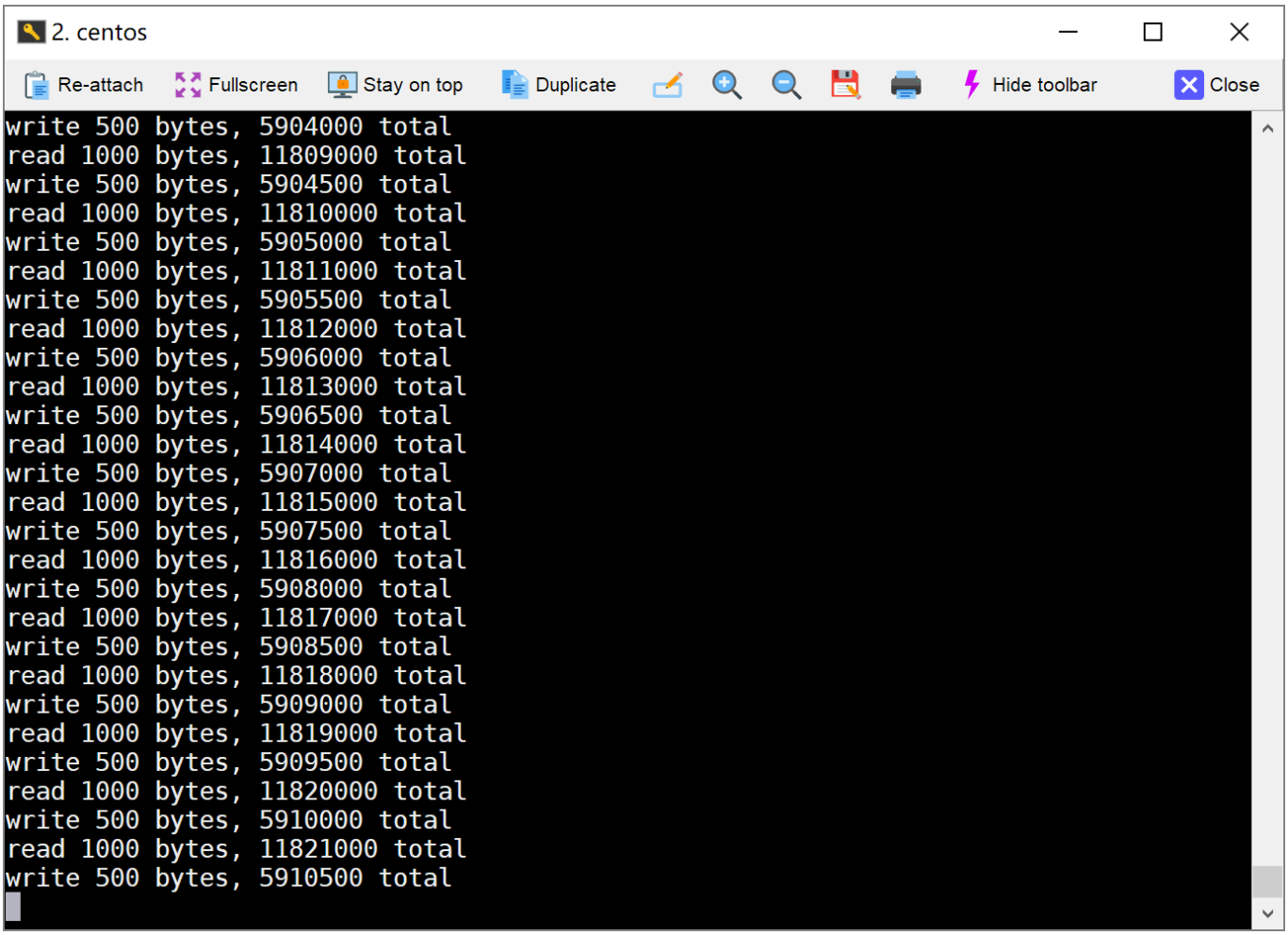
正常读写：

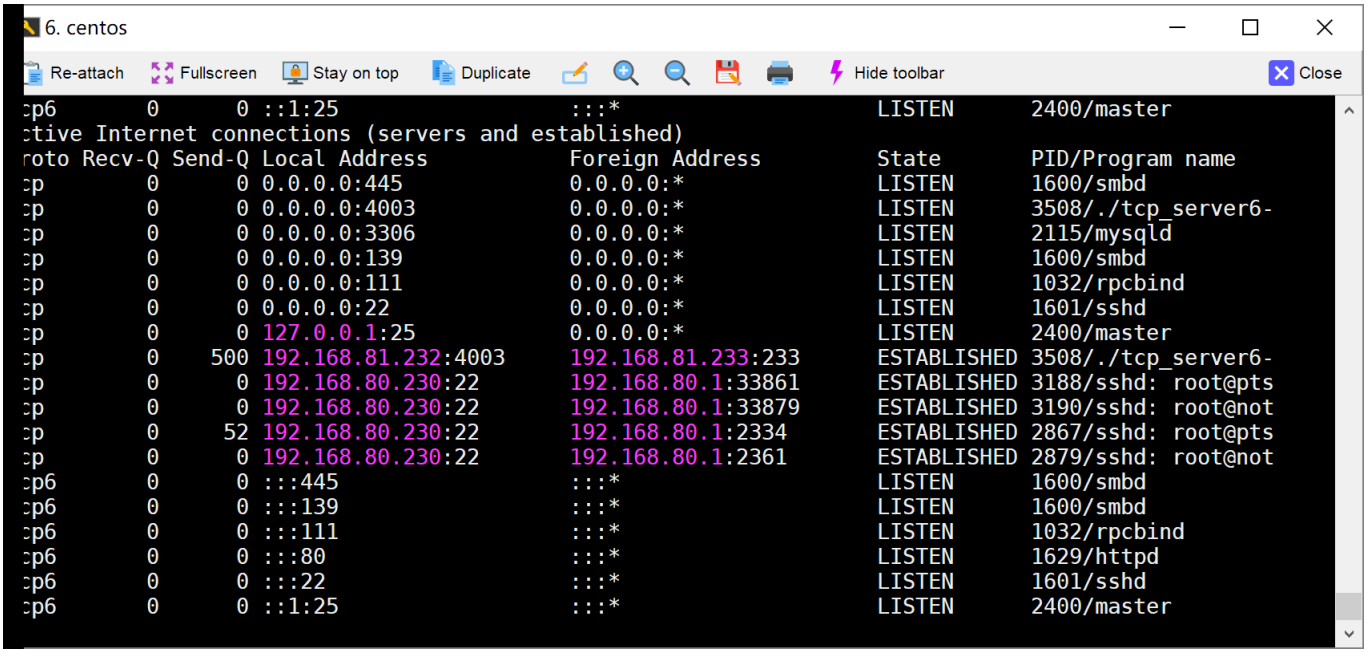


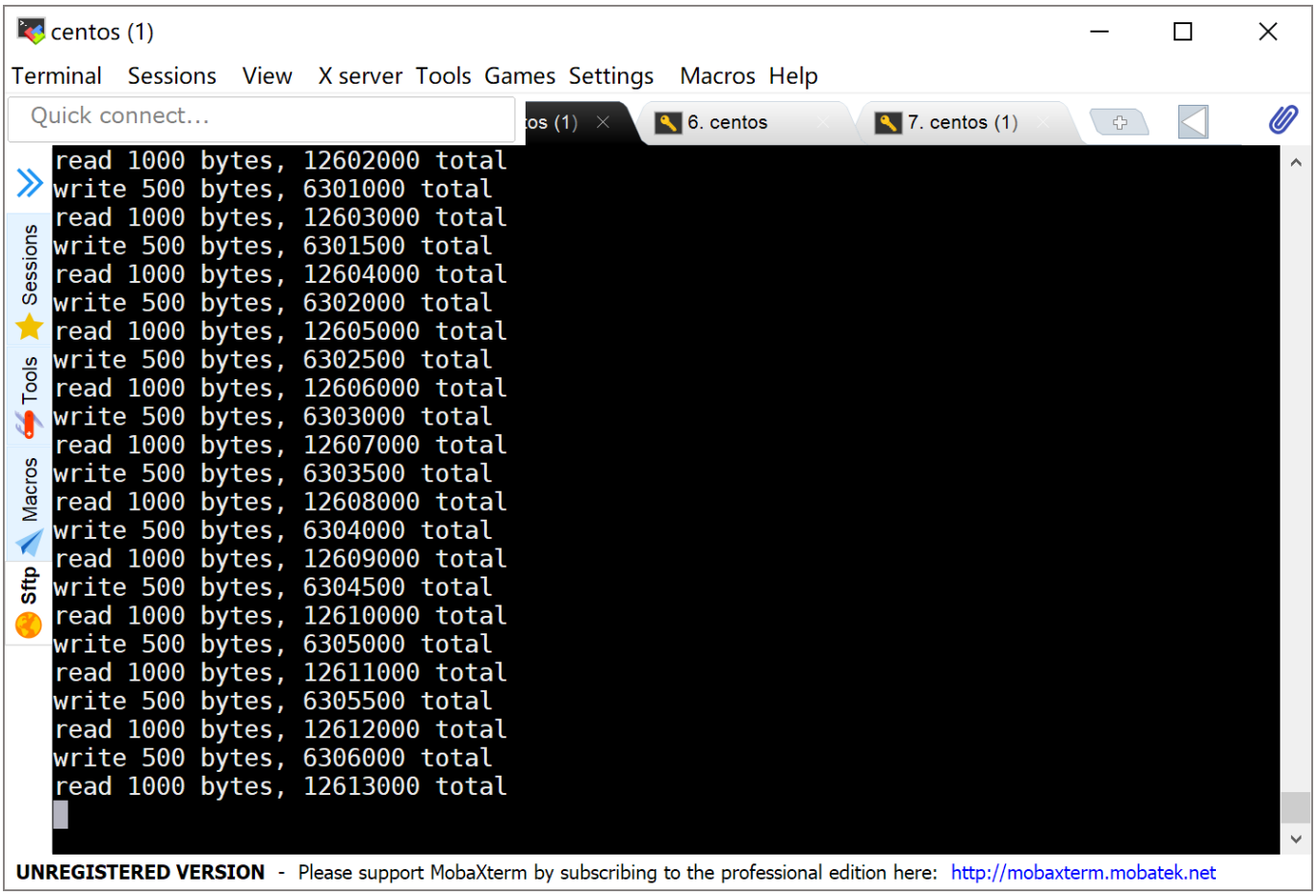


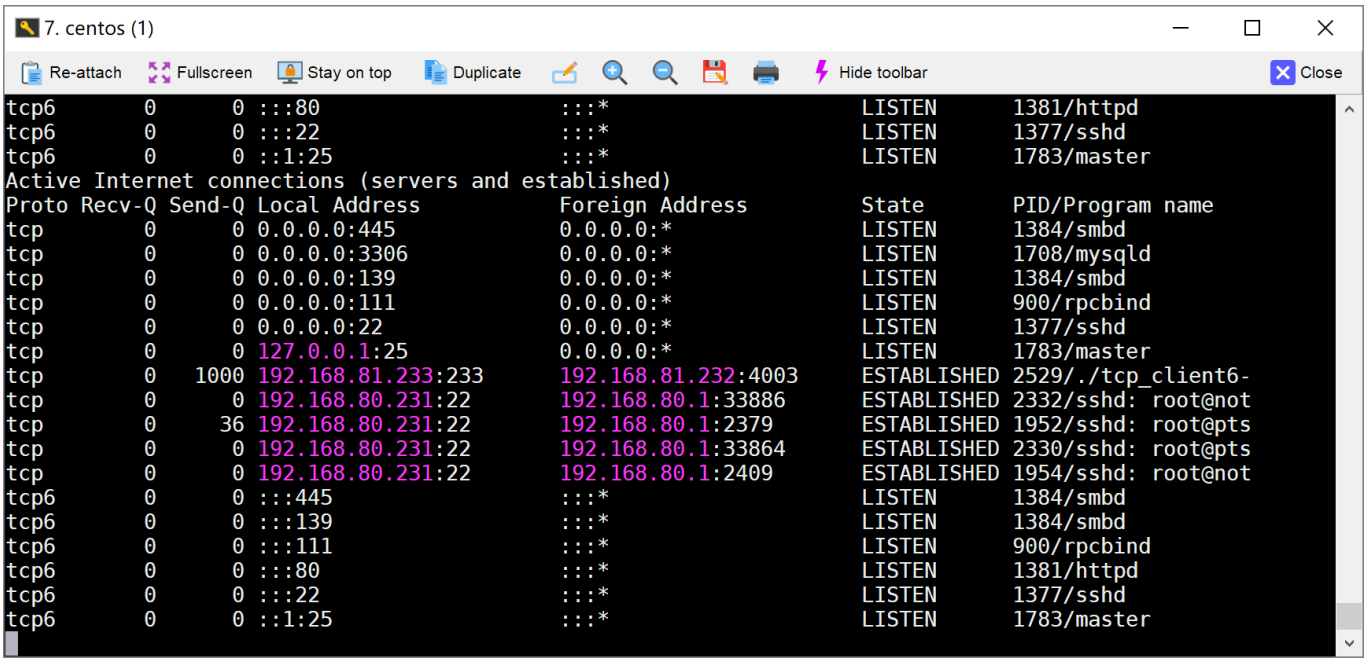
server每次read1000字节write500字节，client每次write1000字节，read500字节

正常读写，不过server有500字节的sendq，client有1000字节的sendq：



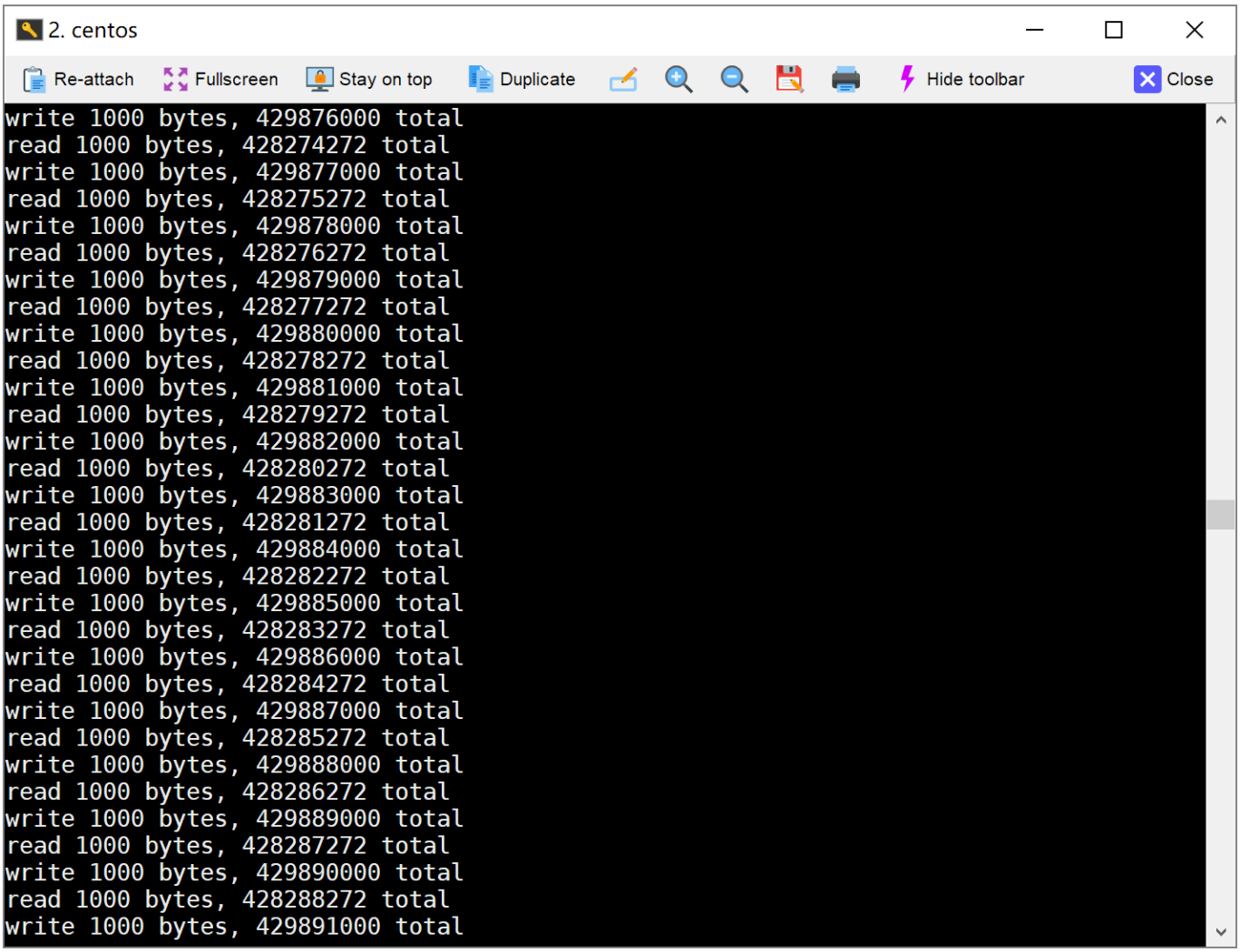




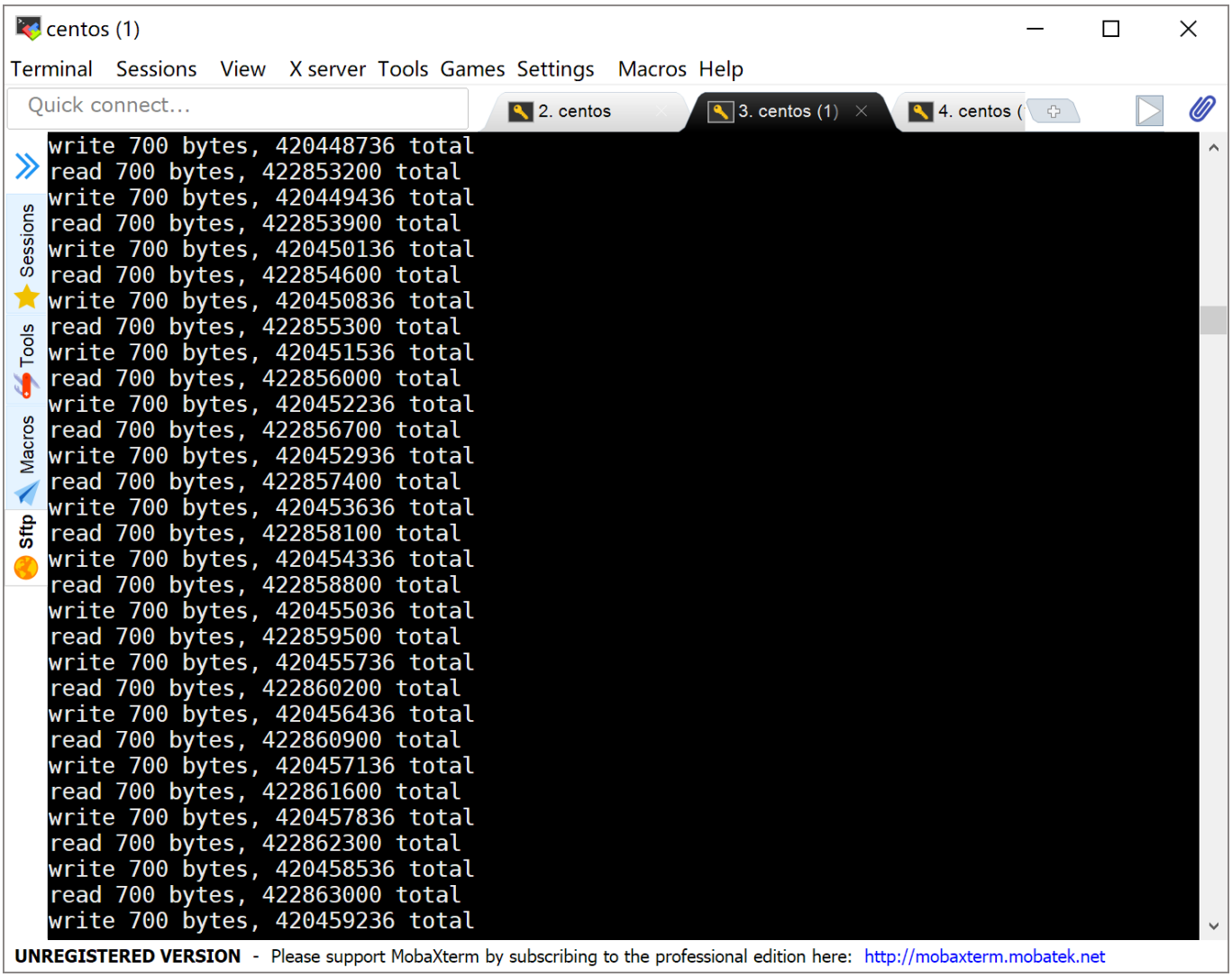


server每次read/write 1000字节，client每次read/write700字节

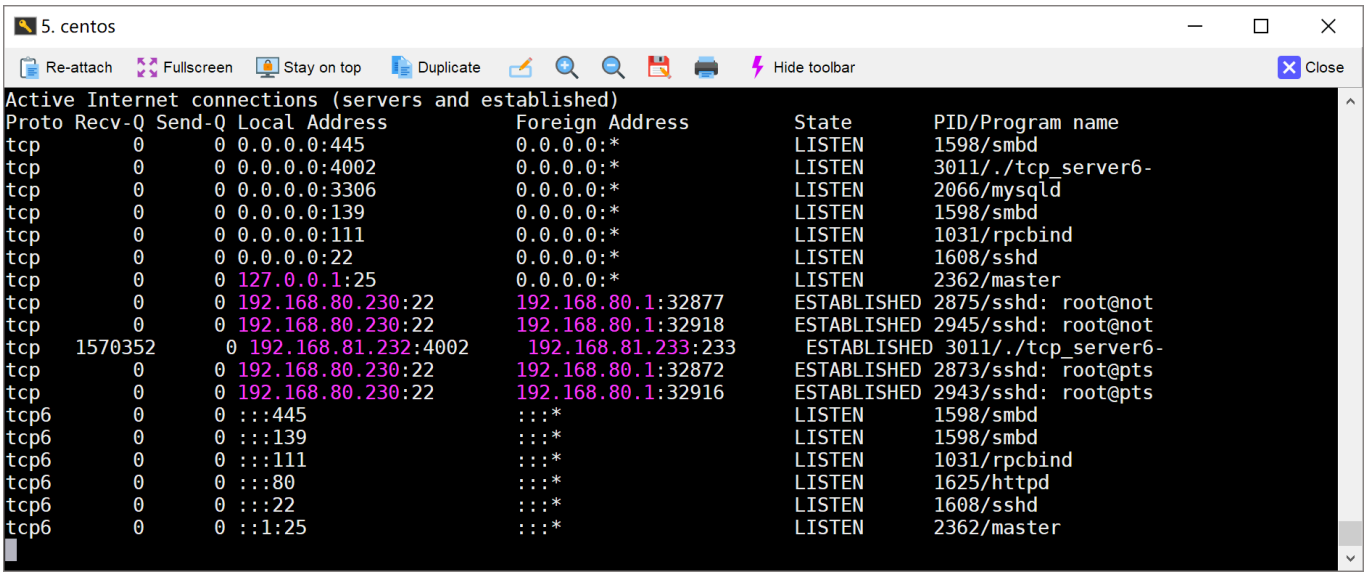
server端每次可以稳定读写1000比特。



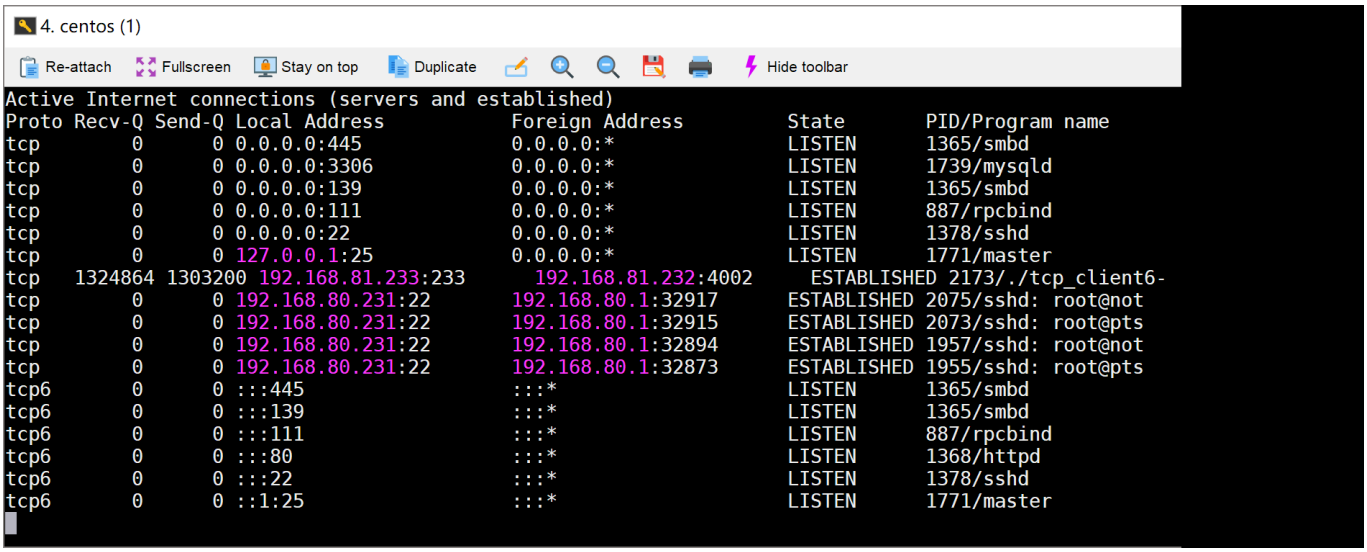
client端每次可以稳定读写700比特。



同时server的recvq大量积压：



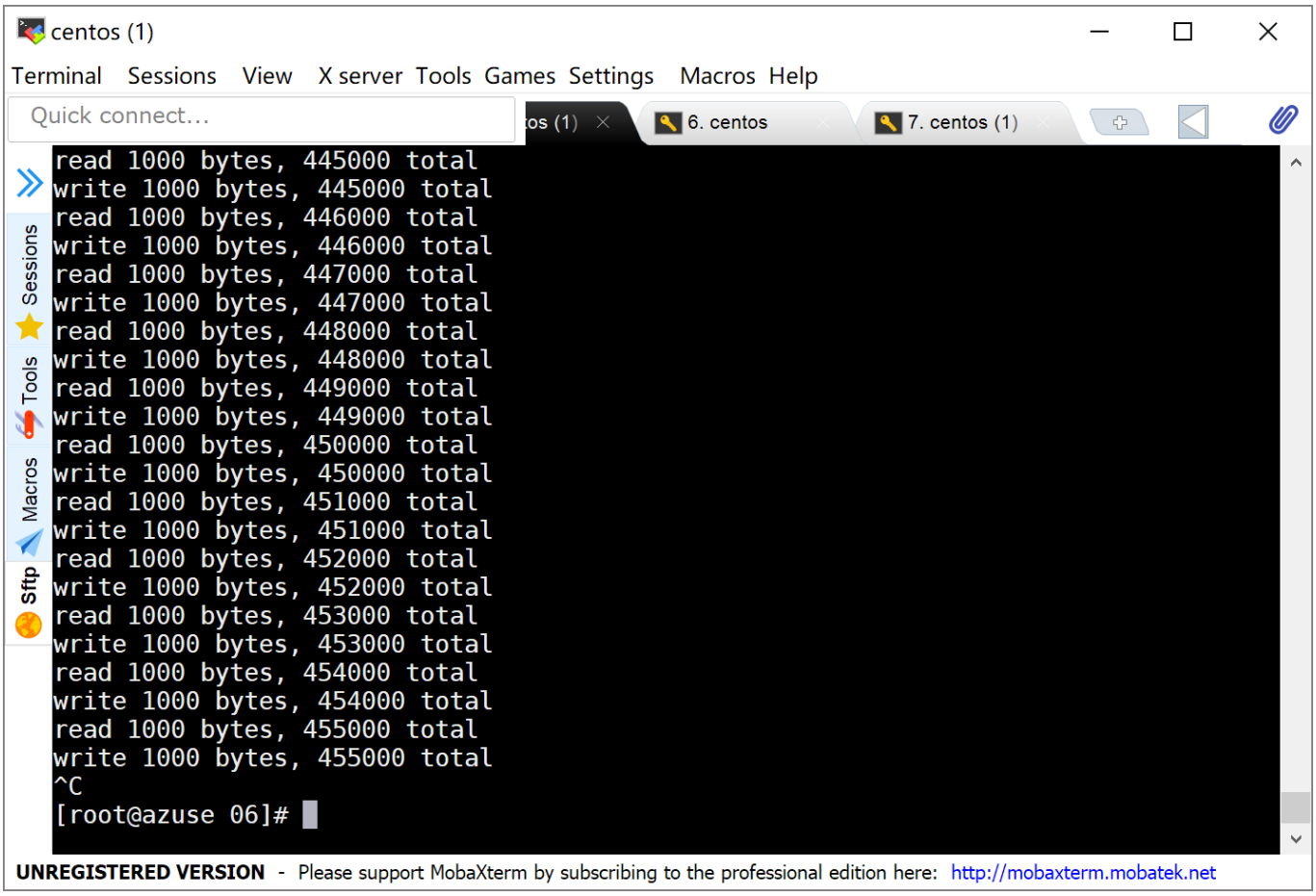
client的sendq和recvq大量积压：

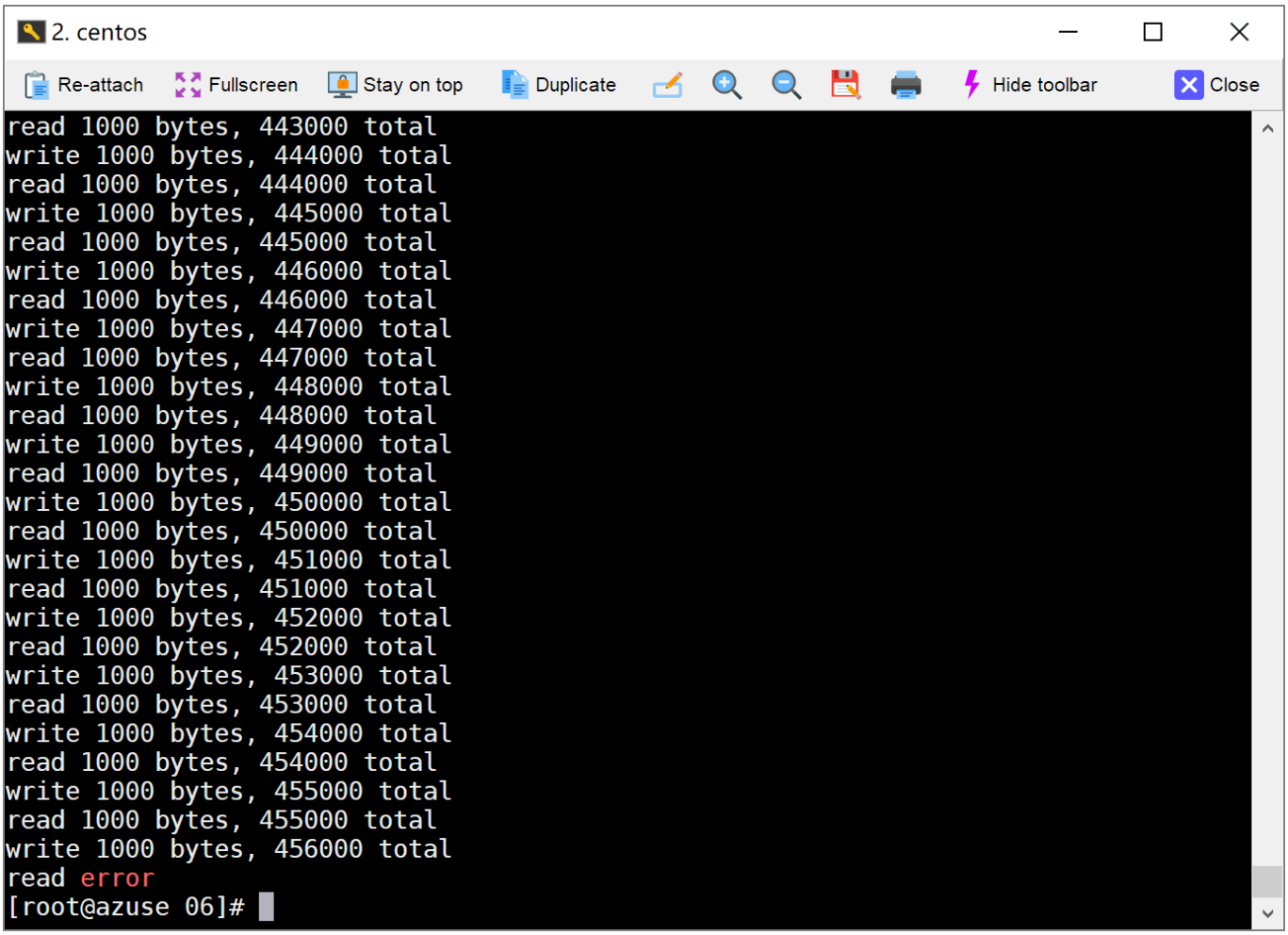


tcp6-3 server先write，client先read

server每次write/read 1000字节，client每次write/read 1000字节

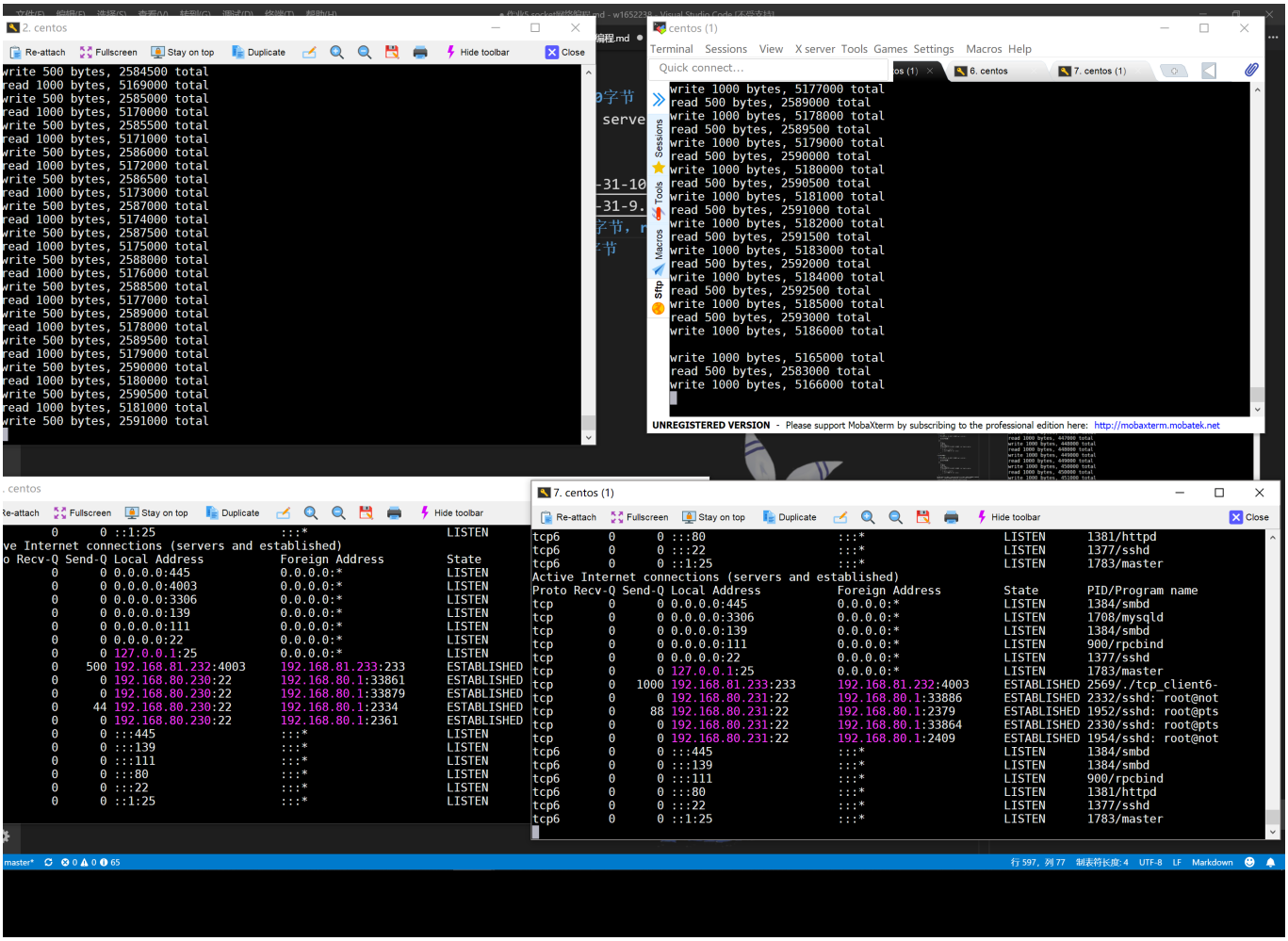
双方可以正常通信，client连上server后read阻塞等待server write，server在客户端连上后直接write再read阻塞等client write。如此循环往复。直到手动终止。





server每次read1000字节write500字节，client每次write1000字节，read500字节

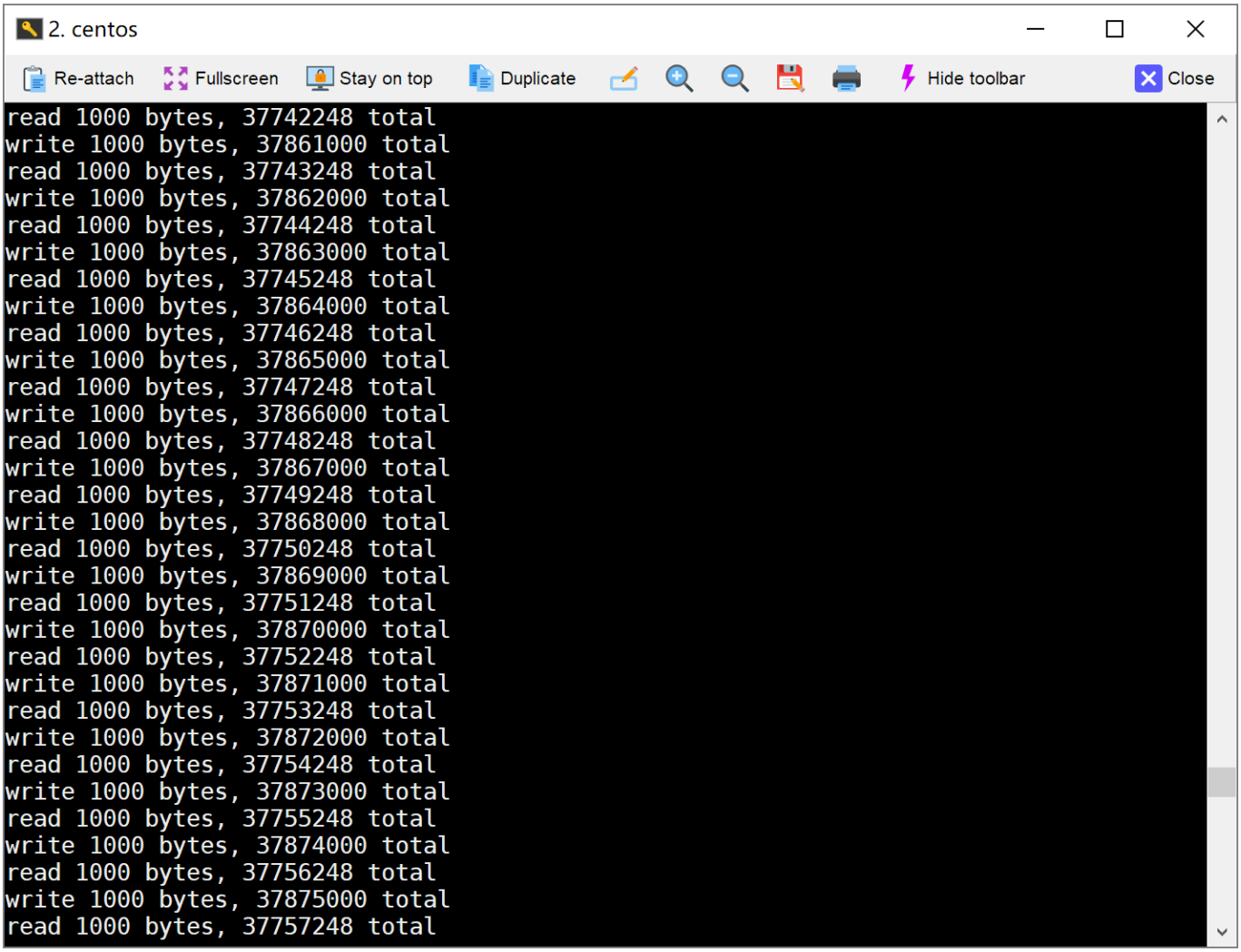
双方正常通信：

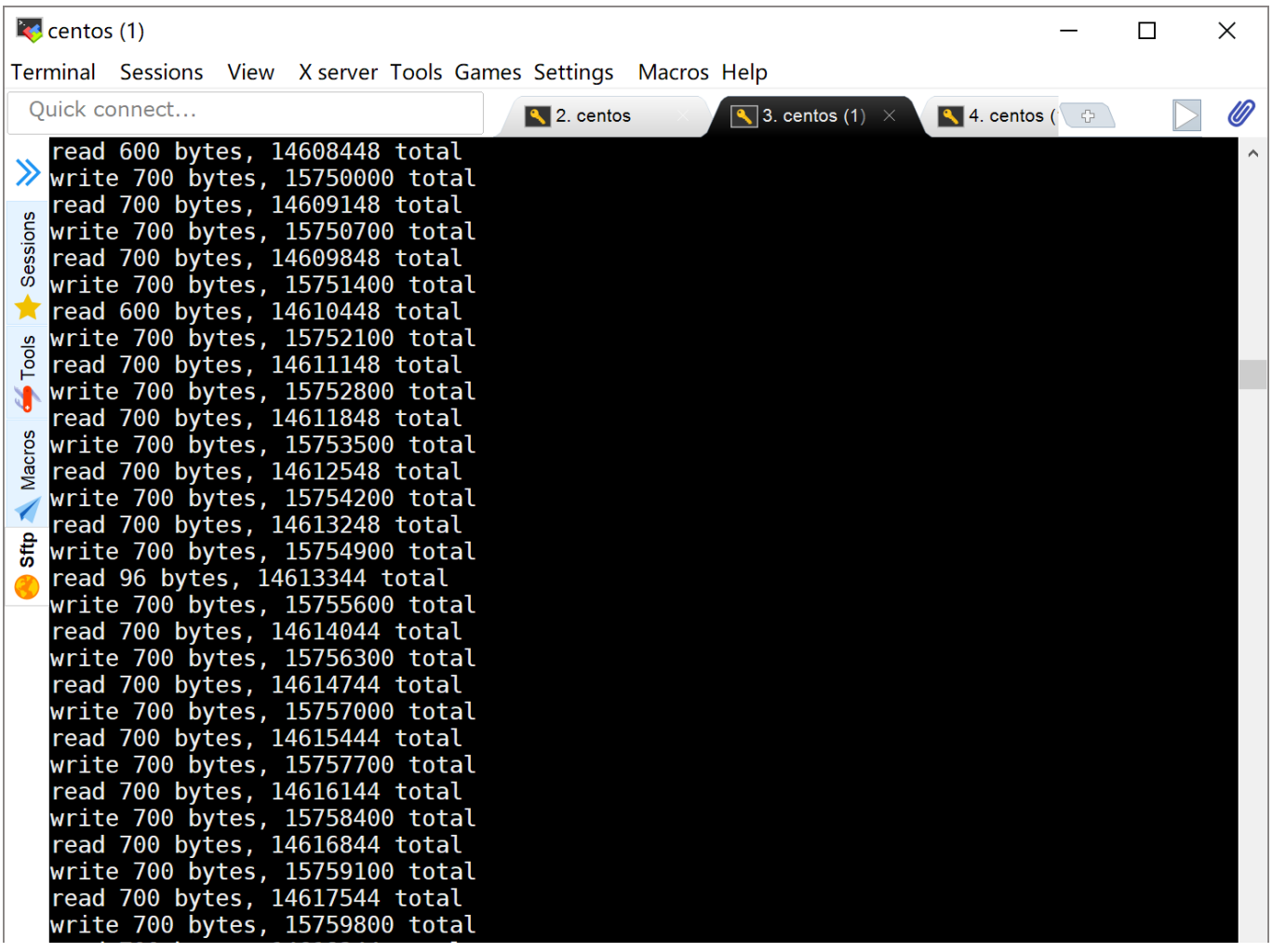


server每次read/write 1000字节，client每次read/write700字节

server每次可以read1000字节，write1000字节。
client每次可以write700字节， read700字节。有时read不足700字节。

sendq和recvq还是有大量积压。


2. centos
Re-attach
Fullscreen
Stay on top
Duplicate
Hide toolbar
Close
read 1000 bytes, 37742248 total
write 1000 bytes, 37861000 total
read 1000 bytes, 37743248 total
write 1000 bytes, 37862000 total
read 1000 bytes, 37744248 total
write 1000 bytes, 37863000 total
read 1000 bytes, 37745248 total
write 1000 bytes, 37864000 total
read 1000 bytes, 37746248 total
write 1000 bytes, 37865000 total
read 1000 bytes, 37747248 total
write 1000 bytes, 37866000 total
read 1000 bytes, 37748248 total
write 1000 bytes, 37867000 total
read 1000 bytes, 37749248 total
write 1000 bytes, 37868000 total
read 1000 bytes, 37750248 total
write 1000 bytes, 37869000 total
read 1000 bytes, 37751248 total
write 1000 bytes, 37870000 total
read 1000 bytes, 37752248 total
write 1000 bytes, 37871000 total
read 1000 bytes, 37753248 total
write 1000 bytes, 37872000 total
read 1000 bytes, 37754248 total
write 1000 bytes, 37873000 total
read 1000 bytes, 37755248 total
write 1000 bytes, 37874000 total
read 1000 bytes, 37756248 total
write 1000 bytes, 37875000 total
read 1000 bytes, 37757248 total


centos (1)
Terminal Sessions View X server Tools Games Settings Macros Help
Quick connect...
2. centos
3. centos (1)
4. centos (
Sessions
Tools
Macros
Sftp
read 600 bytes, 14608448 total
write 700 bytes, 15750000 total
read 700 bytes, 14609148 total
write 700 bytes, 15750700 total
read 700 bytes, 14609848 total
write 700 bytes, 15751400 total
read 600 bytes, 14610448 total
write 700 bytes, 15752100 total
read 700 bytes, 14611148 total
write 700 bytes, 15752800 total
read 700 bytes, 14611848 total
write 700 bytes, 15753500 total
read 700 bytes, 14612548 total
write 700 bytes, 15754200 total
read 700 bytes, 14613248 total
write 700 bytes, 15754900 total
read 96 bytes, 14613344 total
write 700 bytes, 15755600 total
read 700 bytes, 14614044 total
write 700 bytes, 15756300 total
read 700 bytes, 14614744 total
write 700 bytes, 15757000 total
read 700 bytes, 14615444 total
write 700 bytes, 15757700 total
read 700 bytes, 14616144 total
write 700 bytes, 15758400 total
read 700 bytes, 14616844 total
write 700 bytes, 15759100 total
read 700 bytes, 14617544 total
write 700 bytes, 15759800 total

```
read 700 bytes, 14618244 total
```


5. centos

Re-attach | Fullscreen | Stay on top | Duplicate | Hide toolbar | Close

```
tcp        0      0 0.0.0.0:445             0.0.0.0:*               LISTEN      1598/smbd
tcp        0      0 0.0.0.0:4003            0.0.0.0:*               LISTEN      3048/./tcp_server6-
tcp        0      0 0.0.0.0:3306            0.0.0.0:*               LISTEN      2066/mysqld
tcp        0      0 0.0.0.0:139             0.0.0.0:*               LISTEN      1598/smbd
tcp        0      0 0.0.0.0:111             0.0.0.0:*               LISTEN      1031/rpcbind
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      1608/sshd
tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN      2362/master
tcp        0      0 192.168.80.230:22       192.168.80.1:32877      ESTABLISHED 2875/sshd: root@not
tcp        0      0 192.168.80.230:22       192.168.80.1:32918      ESTABLISHED 2945/sshd: root@not
tcp  2520320      0 192.168.81.232:4003     192.168.81.233:58094    ESTABLISHED 3048/./tcp_server6-
tcp        0      0 192.168.80.230:22       192.168.80.1:32872      ESTABLISHED 2873/sshd: root@pts
tcp        0      0 192.168.80.230:22       192.168.80.1:32916      ESTABLISHED 2943/sshd: root@pts
tcp6       0      0 :::445                  :::*                    LISTEN      1598/smbd
tcp6       0      0 :::139                  :::*                    LISTEN      1598/smbd
tcp6       0      0 :::111                  :::*                    LISTEN      1031/rpcbind
tcp6       0      0 :::80                   :::*                    LISTEN      1625/httpd
tcp6       0      0 :::22                   :::*                    LISTEN      1608/sshd
tcp6       0      0 ::1:25                  :::*                    LISTEN      2362/master
```

4. centos (1)

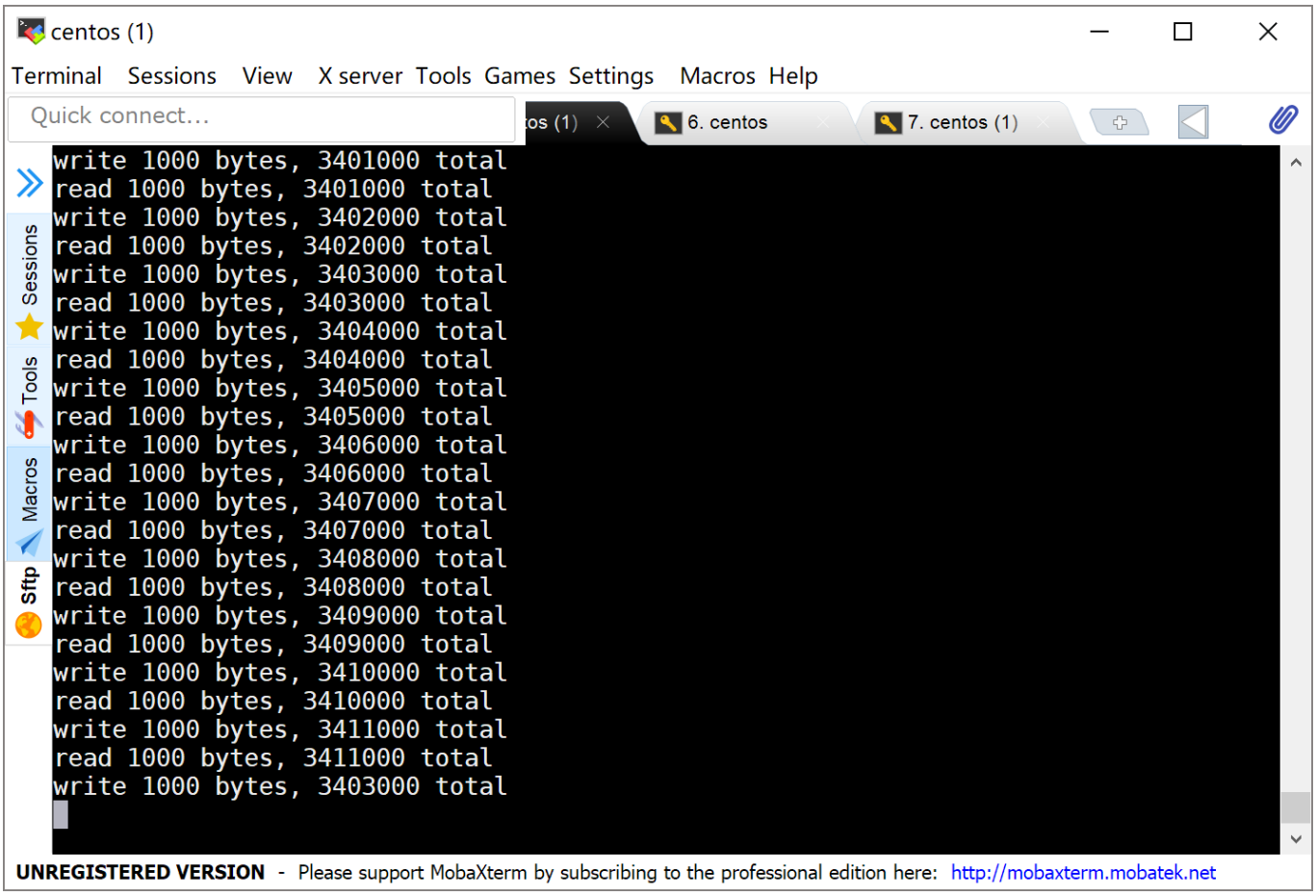
Re-attach | Fullscreen | Stay on top | Duplicate | Hide toolbar

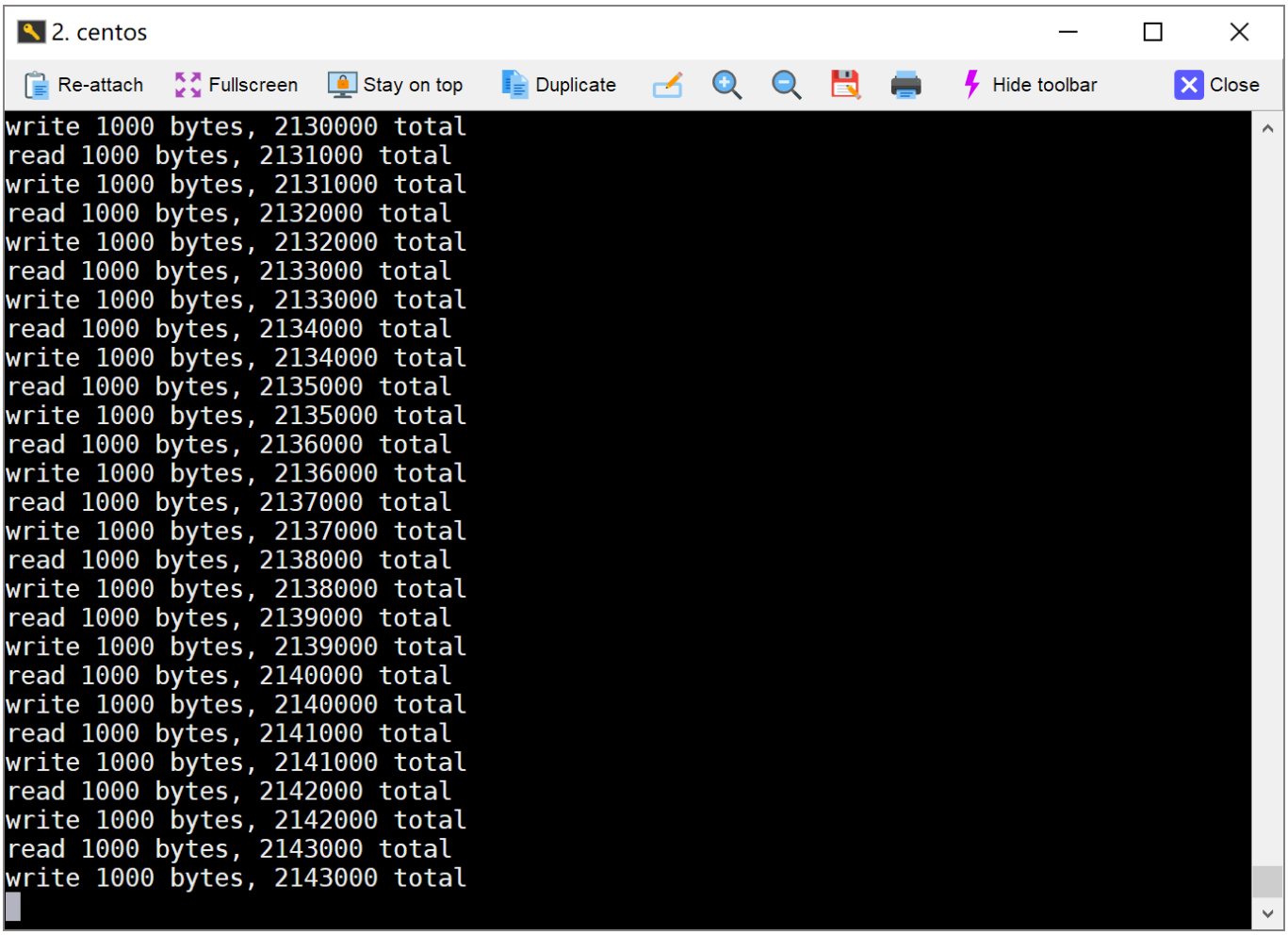
```
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:445             0.0.0.0:*               LISTEN      1365/smbd
tcp        0      0 0.0.0.0:3306            0.0.0.0:*               LISTEN      1739/mysqld
tcp        0      0 0.0.0.0:139             0.0.0.0:*               LISTEN      1365/smbd
tcp        0      0 0.0.0.0:111             0.0.0.0:*               LISTEN      887/rpcbind
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      1378/sshd
tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN      1771/master
tcp        0      0 192.168.80.231:22       192.168.80.1:32917      ESTABLISHED 2075/sshd: root@not
tcp        0      0 192.168.80.231:22       192.168.80.1:32915      ESTABLISHED 2073/sshd: root@pts
tcp        0      0 192.168.80.231:22       192.168.80.1:32894      ESTABLISHED 1957/sshd: root@not
tcp        0   1128 192.168.80.231:22       192.168.80.1:32873      ESTABLISHED 1955/sshd: root@pts
tcp    51744 1585736 192.168.81.233:58094    192.168.81.232:4003     ESTABLISHED 2215/./tcp_client6-
tcp6       0      0 :::445                  :::*                    LISTEN      1365/smbd
tcp6       0      0 :::139                  :::*                    LISTEN      1365/smbd
tcp6       0      0 :::111                  :::*                    LISTEN      887/rpcbind
tcp6       0      0 :::80                   :::*                    LISTEN      1368/httpd
tcp6       0      0 :::22                   :::*                    LISTEN      1378/sshd
tcp6       0      0 ::1:25                  :::*                    LISTEN      1771/master
```

tcp6-4 server先read，client先write

server每次write/read 1000字节，client每次write/read 1000字节

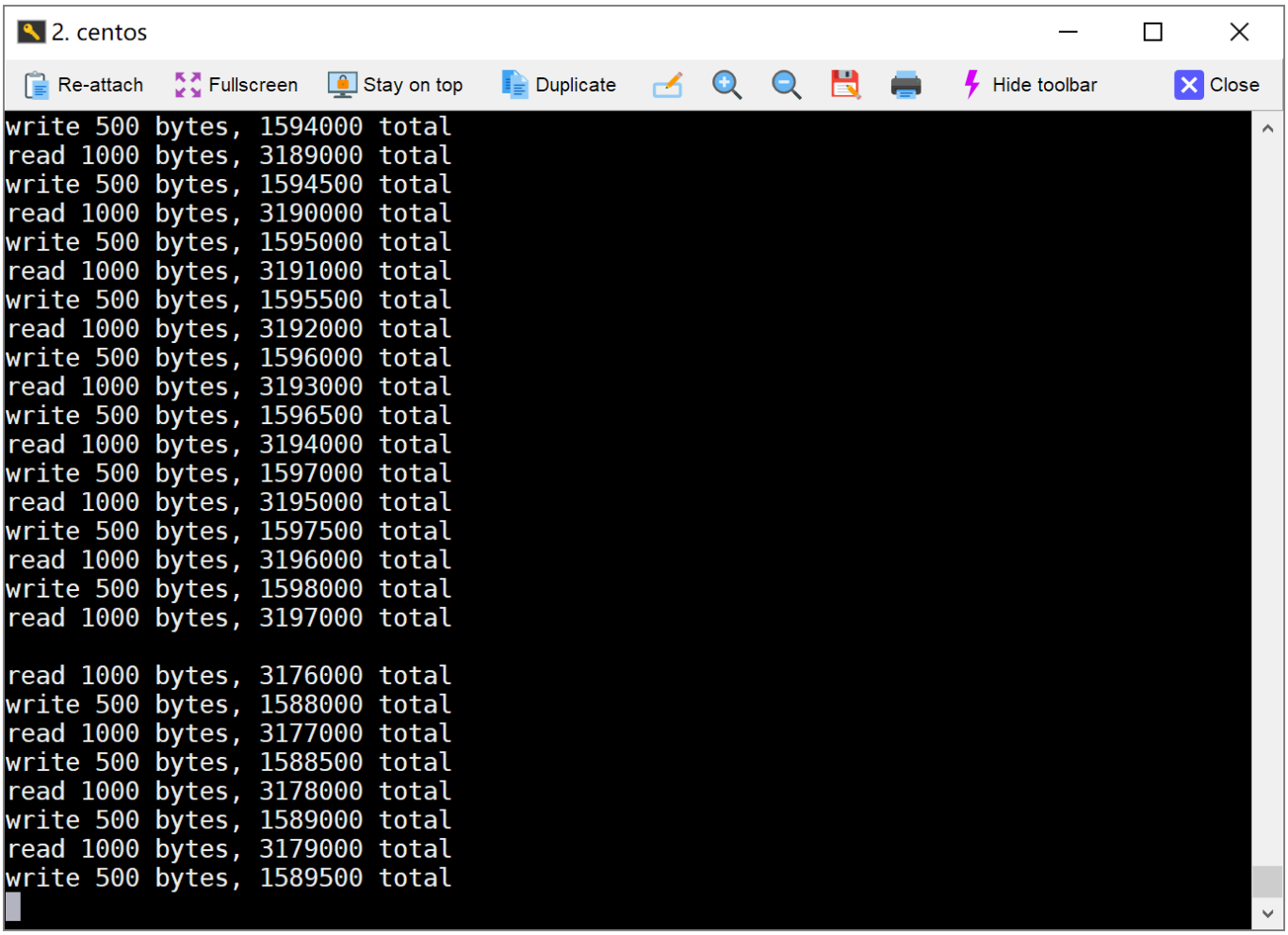
双方都可以每次读写1000字节，无丢失

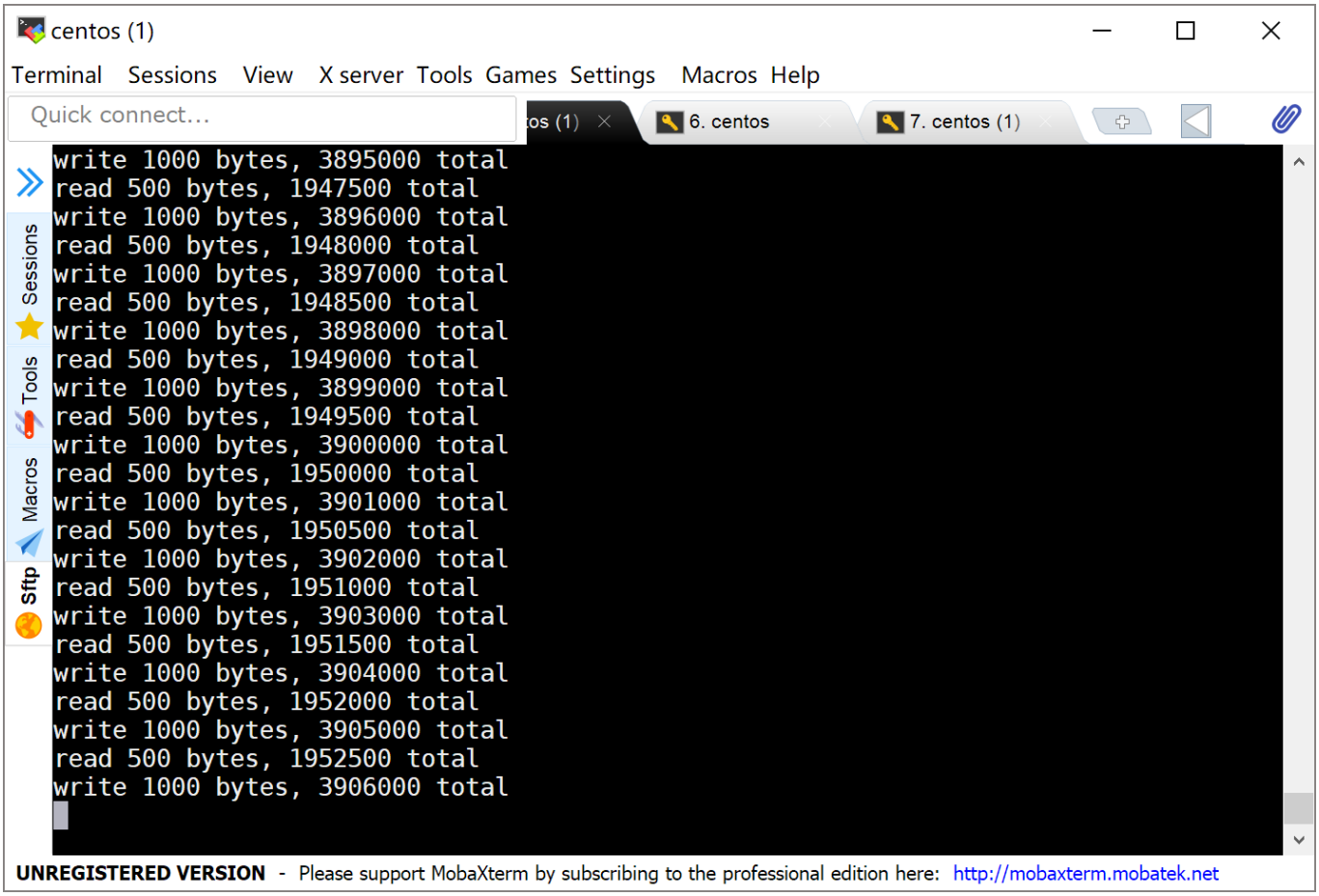




server每次read1000字节write500字节，client每次write1000字节，read500字节

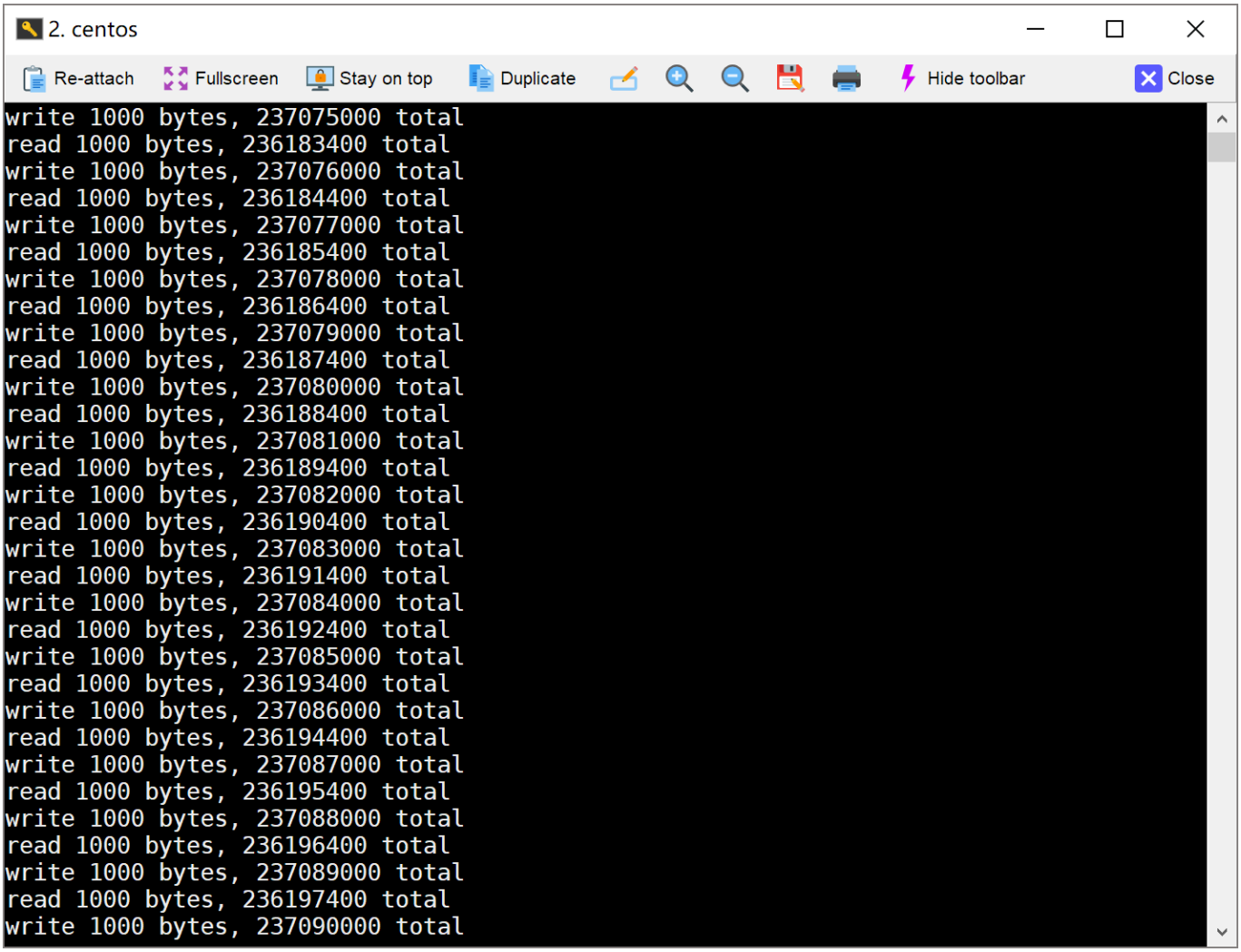
双方可以正常通信，无丢失



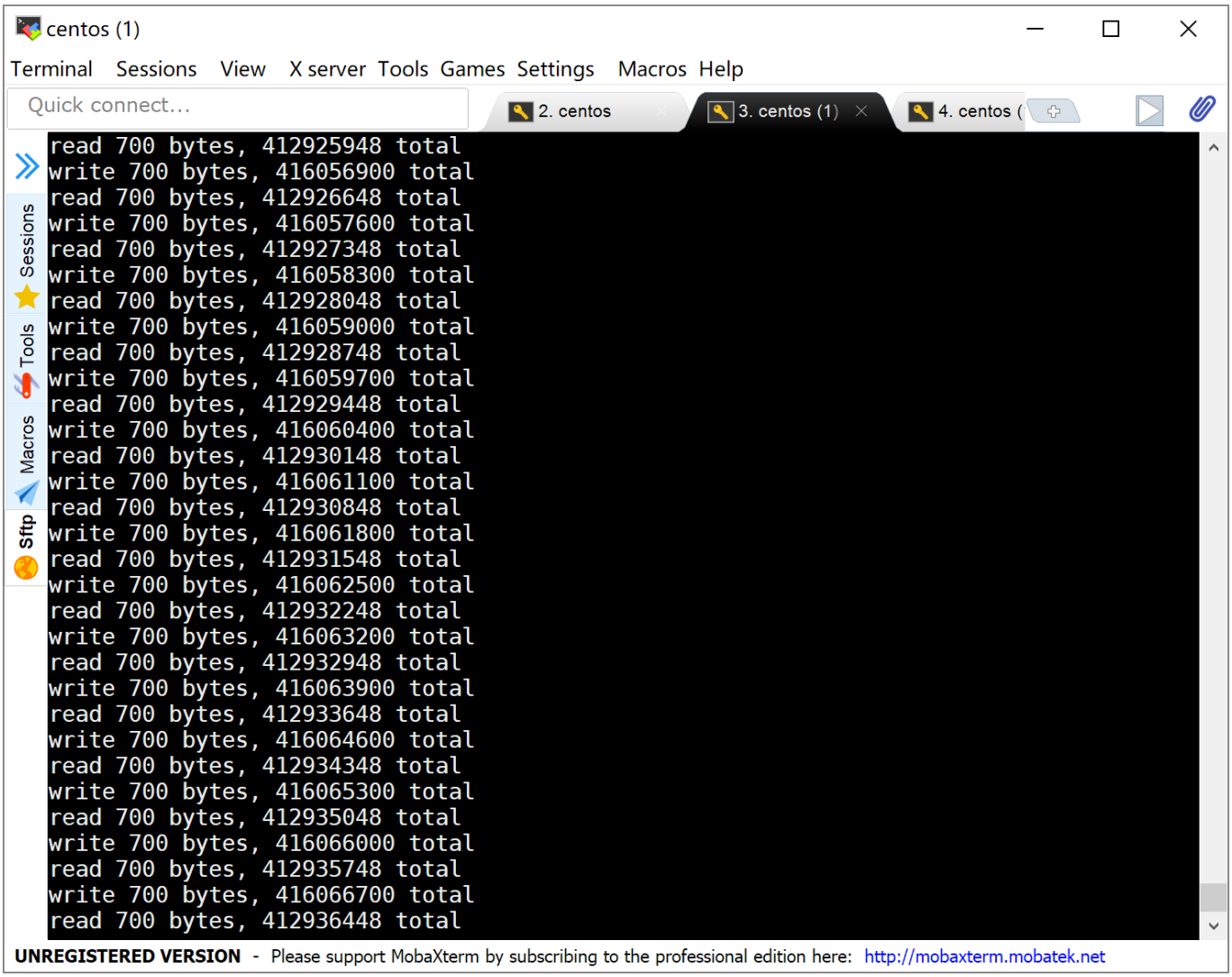


server每次read/write 1000字节，client每次read/write700字节

server读写1000字节。

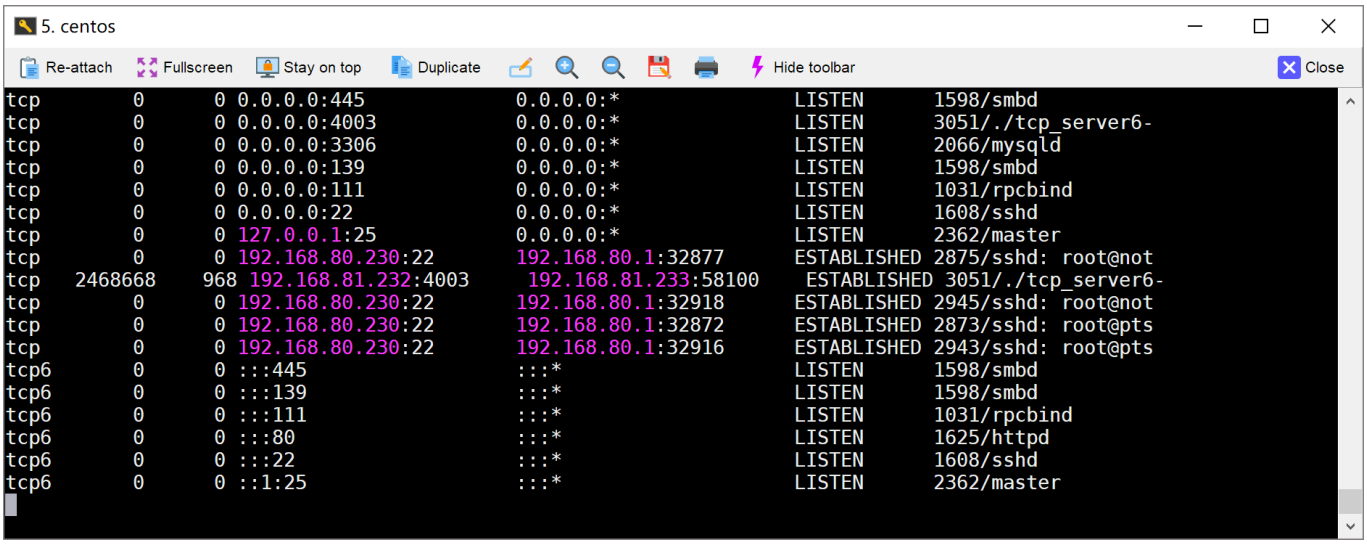


client每次读写700字节。

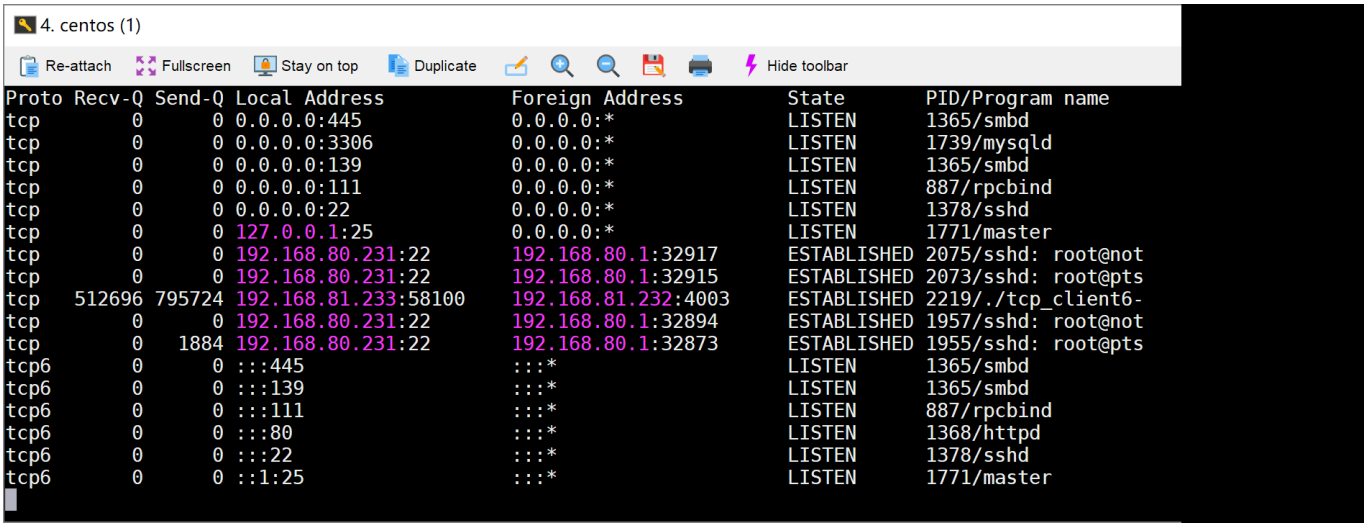


senq与recvq:

server:



client:



结论，不相等的read/write长度在高强度下会在sendq和recvq积压，为了防止recvq和sendq，每次read的长度最好和另一方write的长度对应。