

socket编程

0 虚拟网卡设置

例：

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

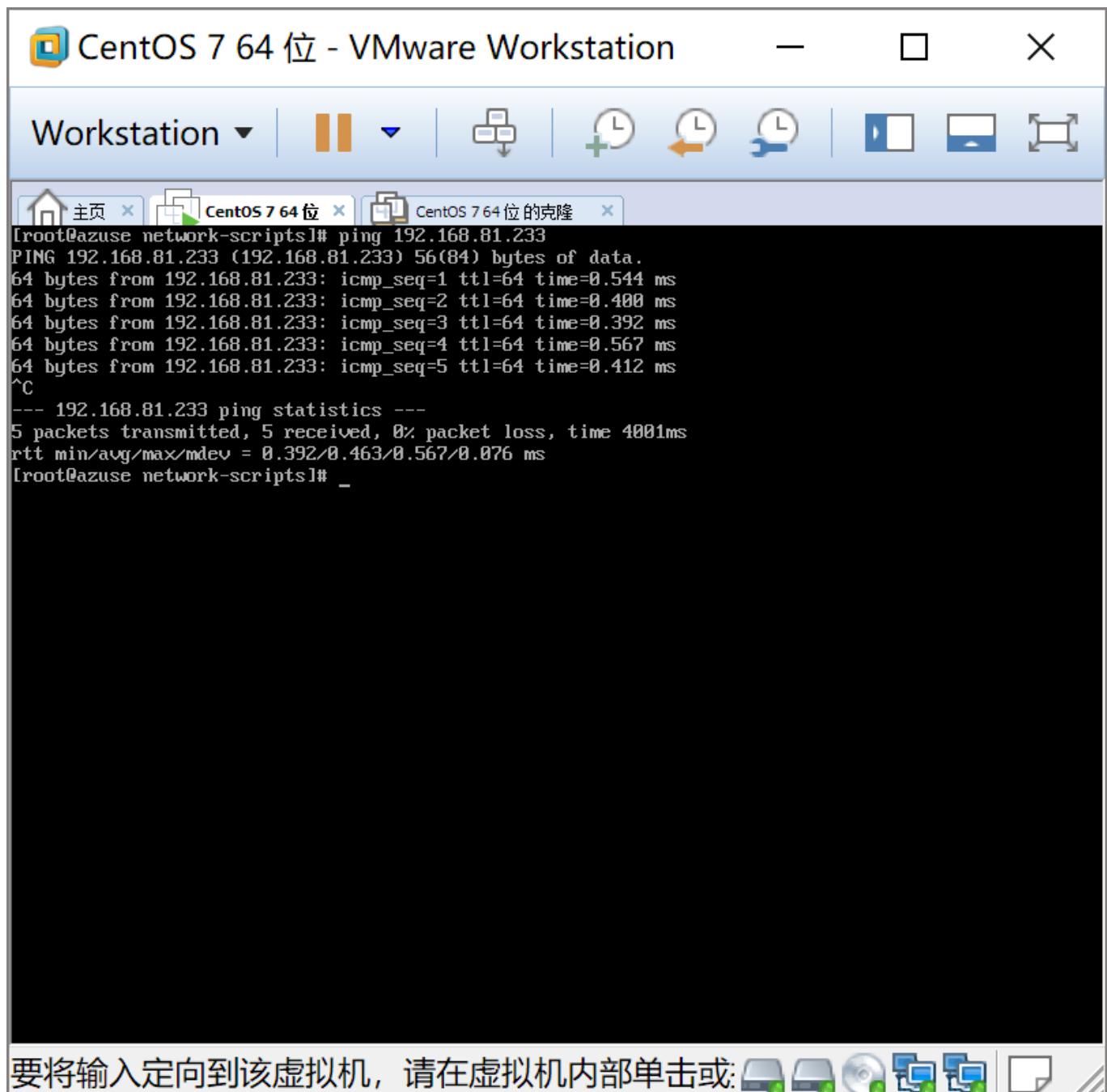
```
TYPE=Ethernet
PROXY_METHOD=none
BROWSER_ONLY=no
BOOTPROTO=static
DEFROUTE=yes
IPV4_FAILURE_FATAL=yes
IPV6INIT=no
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
IPV6_ADDR_GEN_MODE=stable-privacy
NAME=ens32
UUID=26cf553c-793d-4d7a-89f6-a3dad05f5054
DEVICE=ens32
ONBOOT=yes
IPADDR=192.168.80.230
NETMASK=255.255.255.0
PREFIX=24
GATEWAY=192.168.80.2
DNS1=192.168.80.2
IPV6_PRIVACY=no
HARDDDR=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)



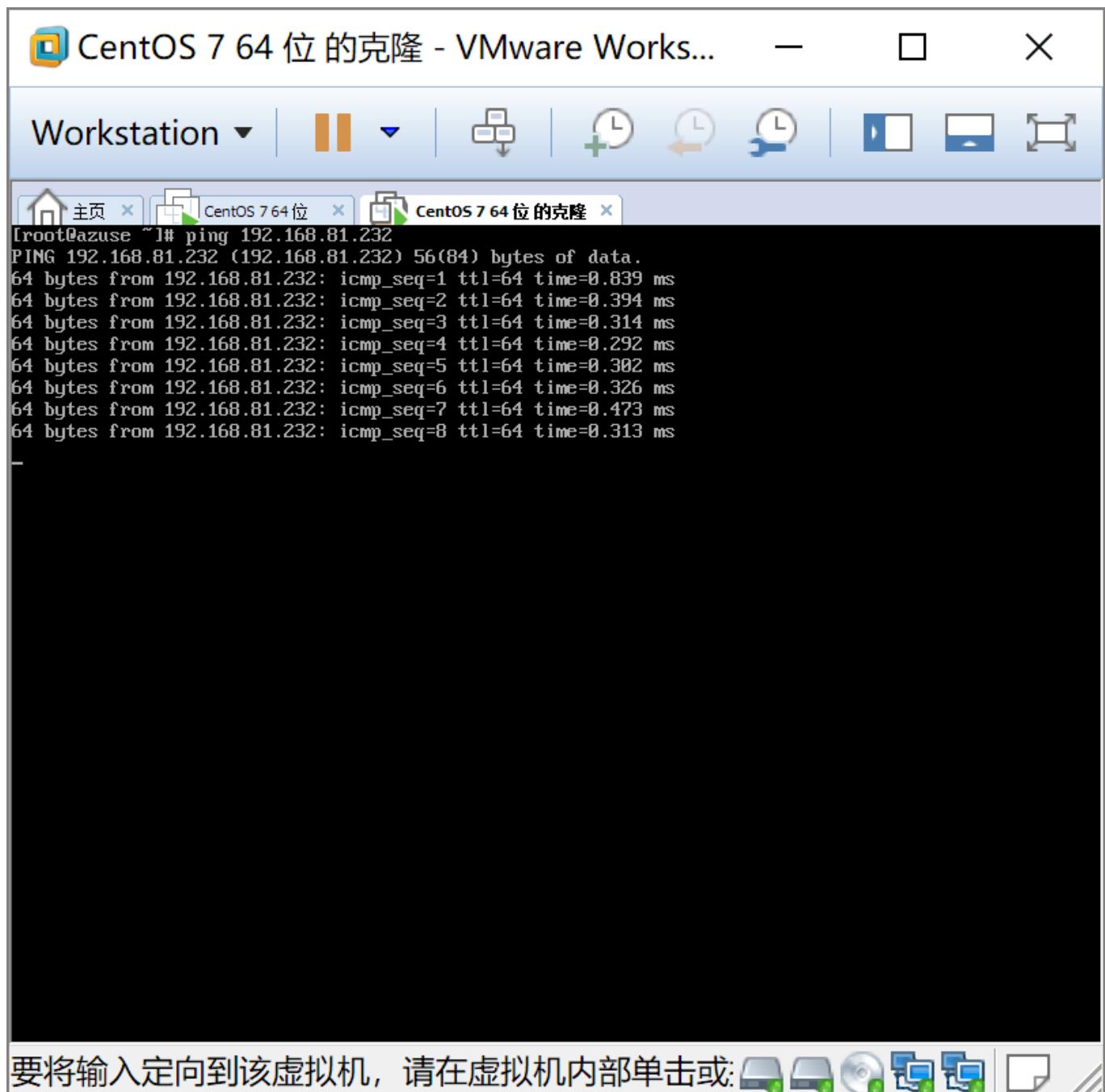
CentOS 7 64 位 - VMware Workstation

Workstation | II | + | - | X

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);
```

```
}
```

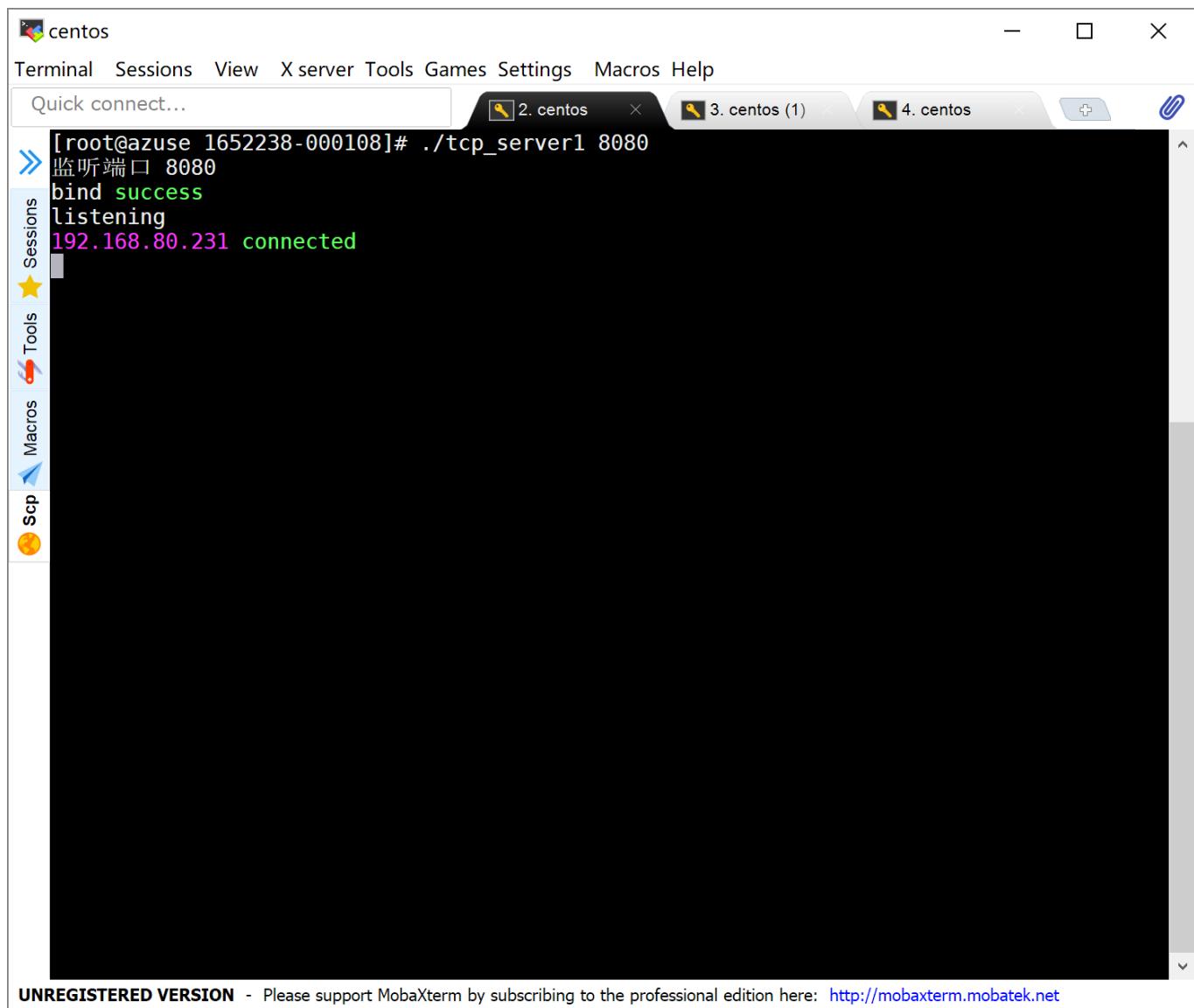
如果端口已被占用，则会在bind函数上出错

```
[root@azuse 1652238-000108]# ./tcp_server1 80
>>> 监听端口 80
bind failed: Address already in use
[root@azuse 1652238-000108]#
```

The screenshot shows a terminal window titled "centos" in the MobaXterm interface. The terminal menu bar includes Terminal, Sessions, View, X server, Tools, Games, Settings, Macros, and Help. A "Quick connect..." search bar is at the top. Below it are tabs for sessions: 2. centos, 3. centos (1), and 4. centos. On the left, there's a sidebar with icons for Sessions, Tools, Macros, and Scp. The main terminal area displays a command-line session where the user runs the ./tcp_server1 80 command. The output shows an error message: "bind failed: Address already in use". At the bottom of the terminal window, a banner reads: "UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <http://mobaxterm.mobatek.net>".

连接成功并进入recv状态

server进入recv状态：



```
[root@azuse 1652238-000108]# ./tcp_server1 8080
>>> 监听端口 8080
bind success
listening
192.168.80.231 connected
```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <http://mobaxterm.mobatek.net>

client进入recv状态：

```
[root@azuse 1652238-000108]# ./tcp_client1 192.168.80.230 8080
连接ip 192.168.80.230
连接端口 8080
连接成功
```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <http://mobaxterm.mobatek.net>

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。

```
[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>

```
[root@azuse 1652238-000108]# ./tcp_client1 192.168.80.230 8080
连接ip 192.168.80.230
连接端口 8080
连接成功
^C
[root@azuse 1652238-000108]# ./tcp_client1 192.168.80.230 8080
★ 连接ip 192.168.80.230
连接端口 8080
连接成功
^Z
[1]+  已停止                  ./tcp_client1 192.168.80.230 8080
[root@azuse 1652238-000108]# bg 1
[1]+ ./tcp_client1 192.168.80.230 8080 &
[root@azuse 1652238-000108]# ps
  PID TTY      TIME CMD
 2415 pts/1    00:00:00 bash
 2474 pts/1    00:00:00 tcp_client1
 2475 pts/1    00:00:00 ps
[root@azuse 1652238-000108]# kill -9 2474
[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

Proto	Recv-Q	Send-Q	Local Address	Foreign Address	State	PID/Program name
tcp	0	0	0.0.0.0:445	0.0.0.*	LISTEN	1577/smbd
tcp	1	0	192.168.81.232:4002	0.0.0.*	LISTEN	2959/.tcp_server5-
tcp	0	0	0.0.0.0:3306	0.0.0.*	LISTEN	2086/mysqld
tcp	0	0	0.0.0.0:139	0.0.0.*	LISTEN	1577/smbd
tcp	0	0	0.0.0.0:111	0.0.0.*	LISTEN	1025/rpcbind
tcp	0	0	0.0.0.0:22	0.0.0.*	LISTEN	1592/sshd
tcp	0	0	127.0.0.1:25	0.0.0.*	LISTEN	2358/master
tcp	1724	0	192.168.81.232:4002	192.168.81.233:4002	ESTABLISHED	2959/.tcp_server5-
tcp	0	0	192.168.80.230:22	192.168.80.1:20390	ESTABLISHED	2852/sshd: root@pts
tcp	2172	0	192.168.81.232:4002	192.168.81.233:4003	ESTABLISHED	-
tcp	0	0	192.168.80.230:22	192.168.80.1:20395	ESTABLISHED	2900/sshd: root@not
tcp6	0	0	:::445	::*	LISTEN	1577/smbd
tcp6	0	0	:::139	::*	LISTEN	1577/smbd
tcp6	0	0	:::111	::*	LISTEN	1025/rpcbind
tcp6	0	0	:::80	::*	LISTEN	1608/httpd
tcp6	0	0	:::22	::*	LISTEN	1592/sshd
tcp6	0	0	:::125	::*	LISTEN	2358/master

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

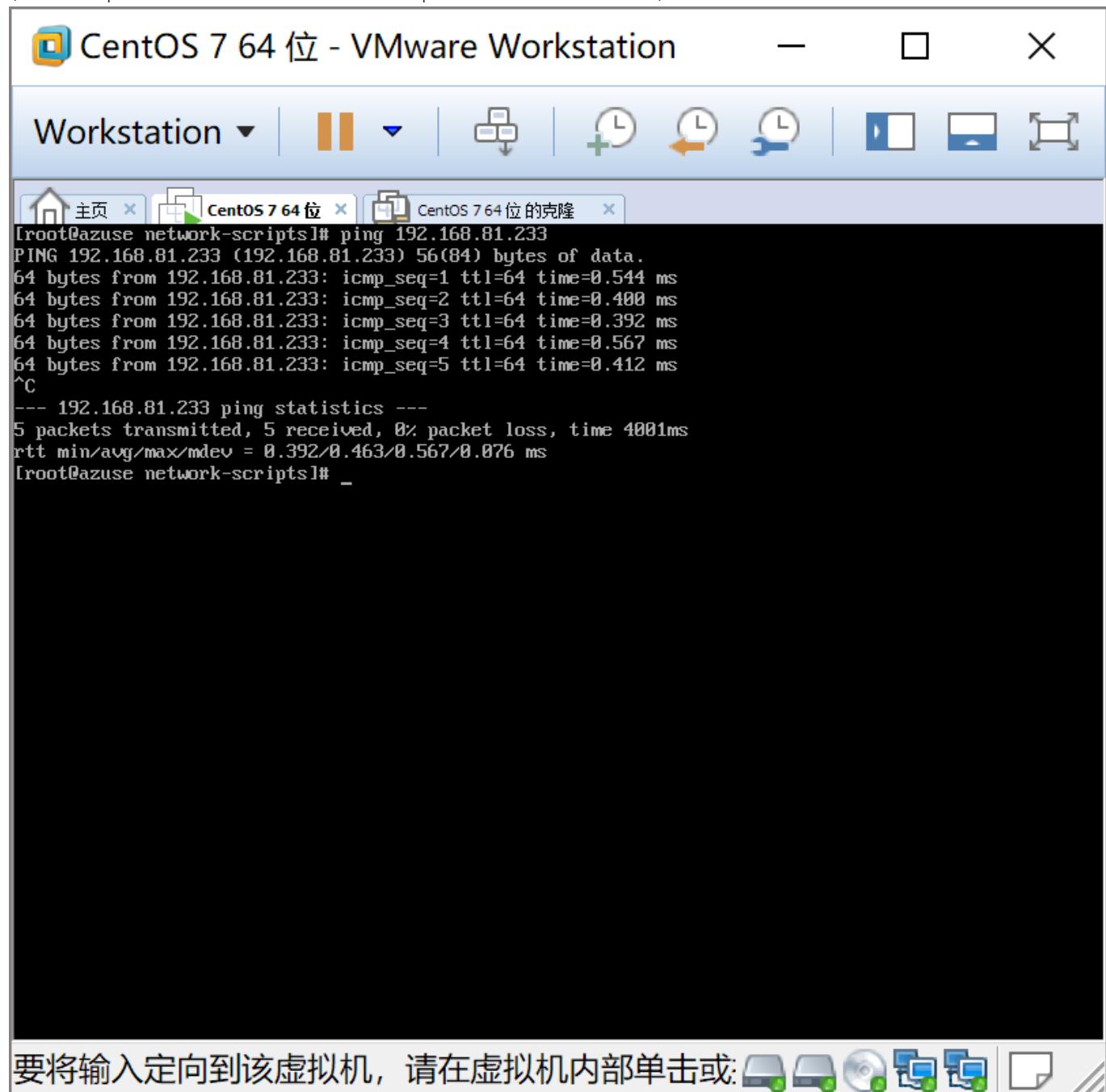
2 加reuseaddr reuseport

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

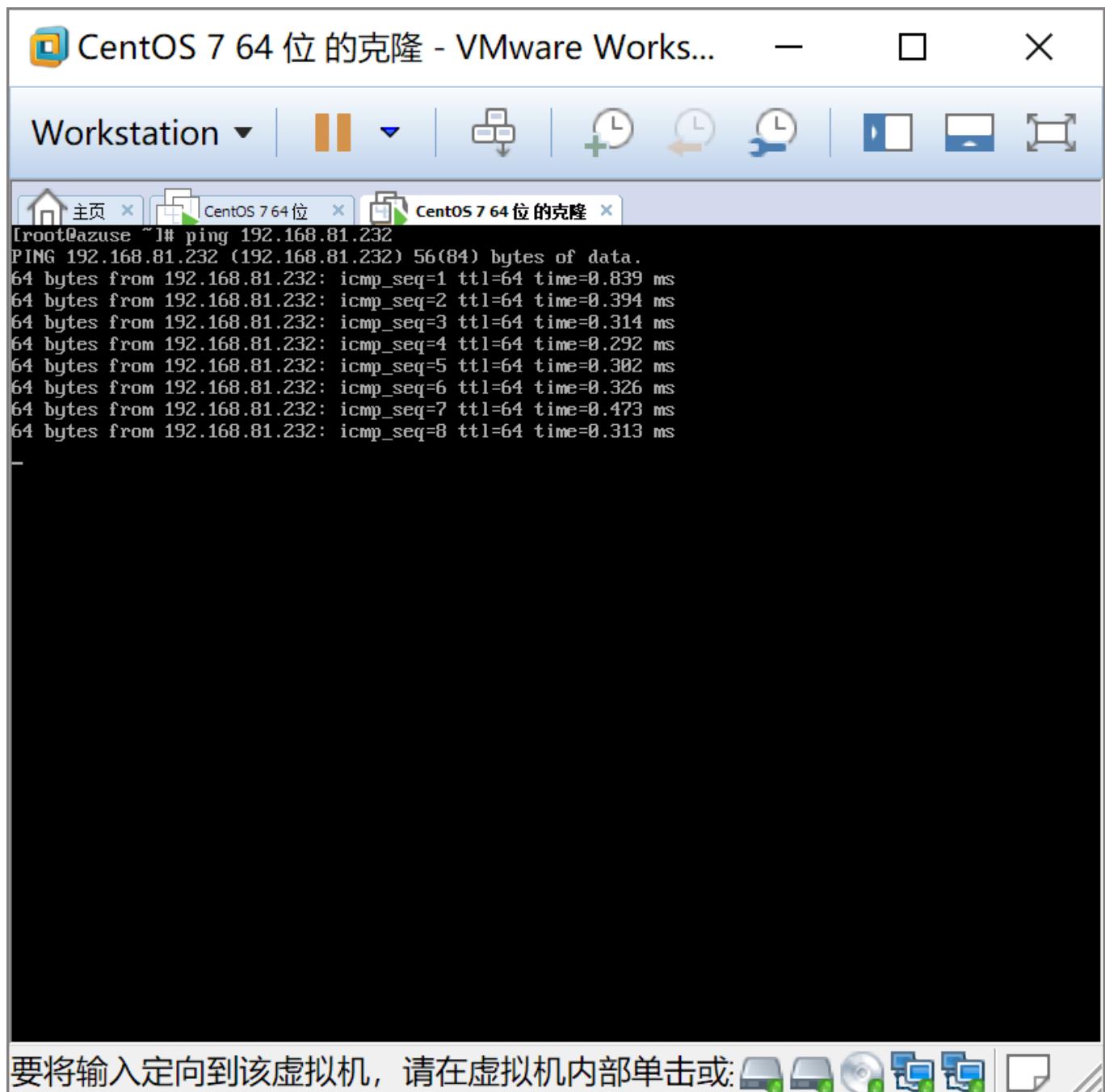
两个虚拟机ip设置与虚拟网卡不同的网段之后，能否ping通？

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)



要将输入定向到该虚拟机，请在虚拟机内部单击或:



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

```
[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
```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <http://mobaxterm.mobatek.net>

服务端：

```
[root@azuse 02]# ./tcp_server2 2333
监听端口 2333
bind success
listening
192.168.81.233 connected port=8090
socket over
```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <http://mobaxterm.mobatek.net>

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->if_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->if_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\tnumber: <%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->if_data != NULL)
        {
            struct rtnl_link_stats *stats = ifa->if_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：

The screenshot shows a MobaXterm window titled "centos" with three tabs open. The left sidebar includes icons for Sessions, Tools, Macros, and Sftp. The "Sessions" icon is highlighted. The first tab, "2. centos", shows the command "make" being run, followed by "gcc" compilation of two source files into executables "tcp_server3" and "tcp_client3". The second tab, "3. centos (1)", shows the server being started with "./tcp_server3 192.168.81.232 4000", indicating it's listening on port 4000. The third tab is empty. A vertical scroll bar is visible on the right side of the terminal window.

```
[root@azuse 03]# make
gcc tcp_server3.c -o tcp_server3
gcc tcp_client3.c -o tcp_client3
[root@azuse 03]# ./tcp_server3 192.168.81.232 4000
监听 192.168.81.232 port=4000
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:fed:a302d%ens3
可用地址7: fe80::20c:29ff:fed:a3037%ens34
使用地址: 192.168.81.232
bind success
listening
192.168.81.233 connected port=4000
socket over
```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <http://mobaxterm.mobatek.net>

如果客户端连接未绑定的网卡会怎么样？

会连不上

```
[root@azuse 03]# ./tcp_client3 4000 192.168.80.230 4000
使用端口 4000
连接ip 192.168.80.230
连接端口 4000
Binded Correctly

★ Connection Failed
[root@azuse 03]# ./tcp_client3 4000 192.168.81.232 4000
使用端口 4000
连接ip 192.168.81.232
连接端口 4000
Binded Correctly
连接成功
```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <http://mobaxterm.mobatek.net>

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
```

```
[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：

```
[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:fed:a302d%sens32
可用地址7: fe80::20c:29ff:fed:a3037%sens34
-----
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

```
[root@azuse 04]# ./tcp_client4-2-1 4001 192.168.81.232 2000
使用端口 4001
连接ip 192.168.81.232
连接端口 2000
Binded Correctly
连接成功
message sent: 20byte length send
[root@azuse 04]# ./tcp_client4-2-2 4002 192.168.81.232 2000
使用端口 4002
连接ip 192.168.81.232
连接端口 2000
Binded Correctly
连接成功
message sent: A
message sent: B
message sent: C
message sent: D
message sent: E
```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <http://mobaxterm.mobatek.net>

```
[root@azuse 04]# ./tcp_server4-2 192.168.81.232 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:fed:302d%ens32
可用地址7: fe80::20c:29ff:fed:3037%ens34
-----
使用地址: 192.168.81.232
-----
bind success
listening
192.168.81.233 connected port=4001
receiving...
message received: 20byte length send
socket over
192.168.81.233 connected port=4002
receiving...
message received: A
socket over
```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <http://mobaxterm.mobatek.net>

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");
}

```

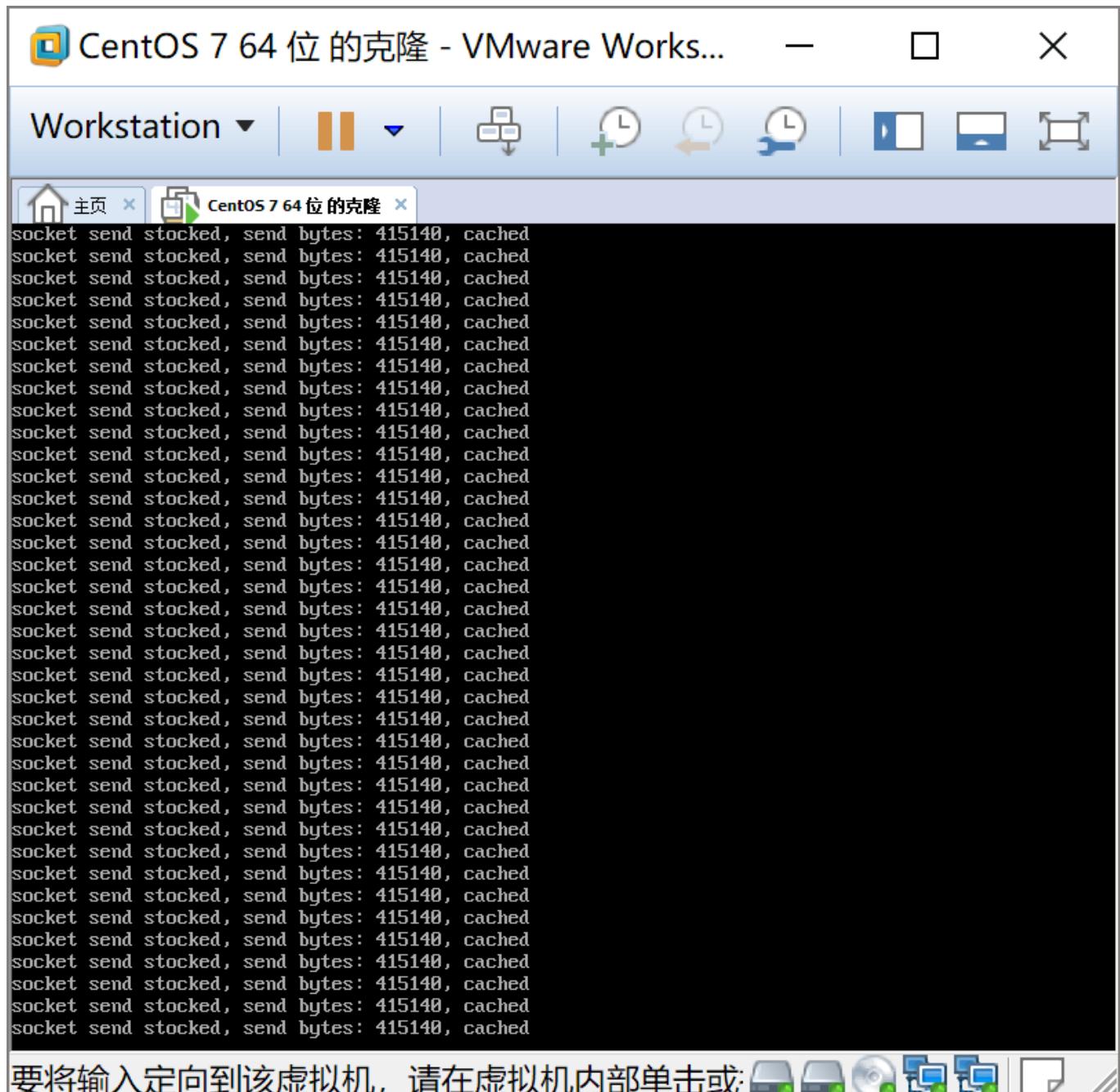
```
else
{
    long int *sendbytes_p = (long int *)shmat(shmid, NULL, 0);
    long int sendbytes_cached;
    while (1)
    {
        sendbytes_cached = *sendbytes_p;
        sleep(0.5);
        if (*sendbytes_p == sendbytes_cached)
        {
            //printf("socket send stocked, send bytes: %ld,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

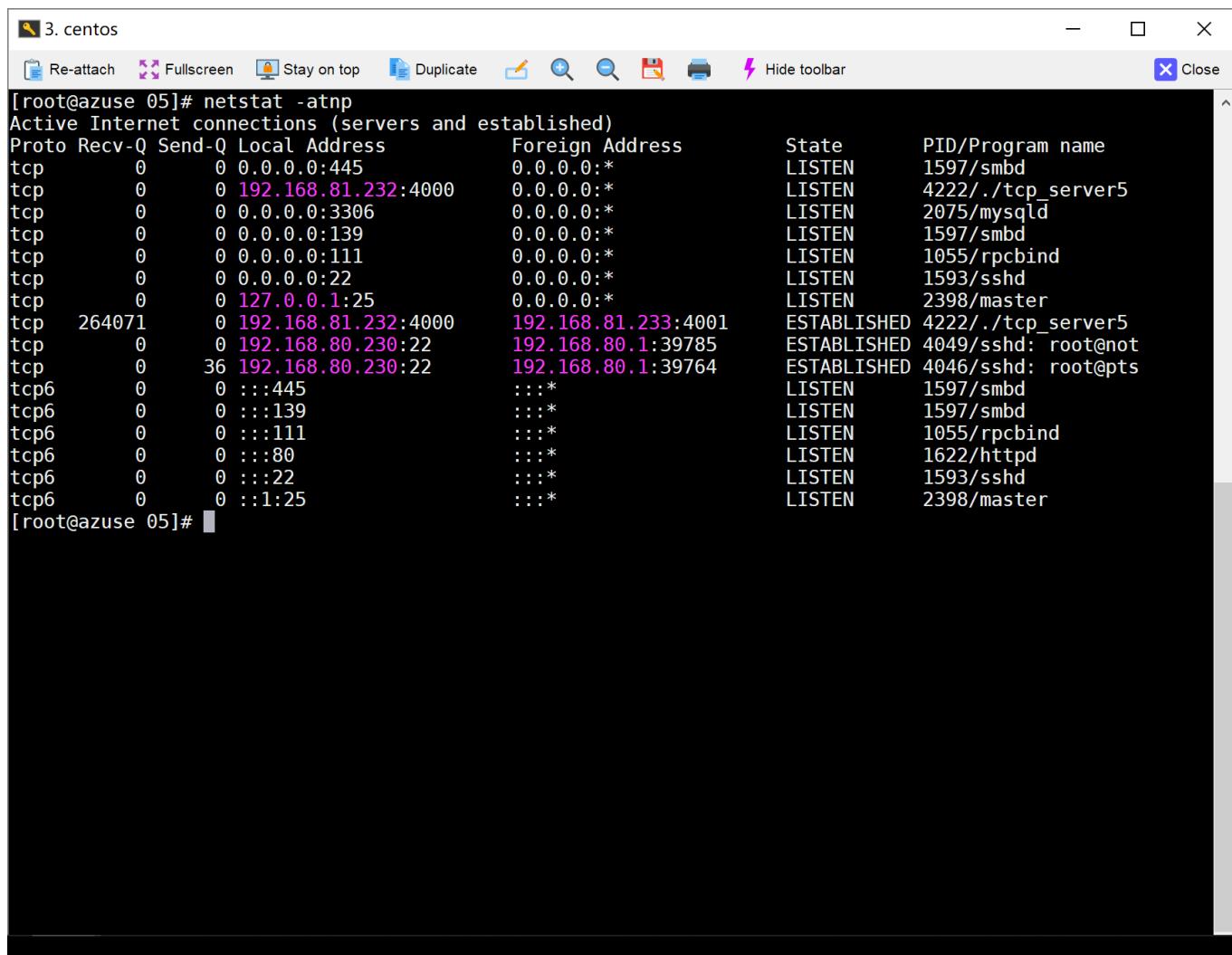


The screenshot shows a terminal window titled 'centos (1)' running on MobaXterm. The window contains the command output of 'netstat -atnp'. The output lists active Internet connections, including servers and established connections. The columns in the table are: Proto, Recv-Q, Send-Q, Local Address, Foreign Address, State, and PID/Program name. The output shows various services like smbd, mysqld, rpcbind, sshd, and httpd.

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	:::125	:::*	LISTEN	1779/master

[root@azuse 05]#

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <http://mobaxterm.mobatek.net>

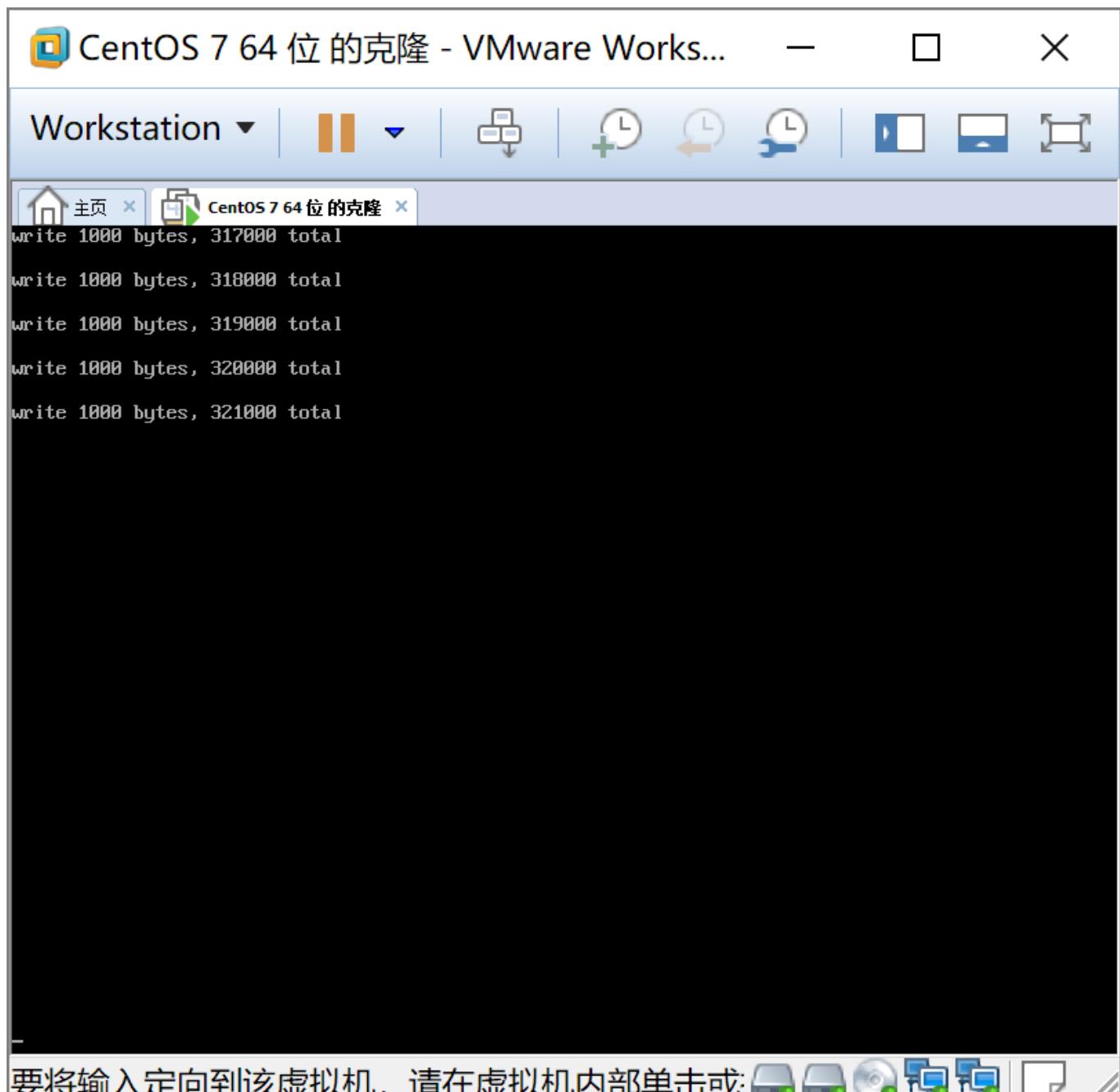


The screenshot shows a terminal window titled "3. centos". The command "netstat -atnp" is run, displaying active Internet connections. The output includes columns for Proto, Recv-Q, Send-Q, Local Address, Foreign Address, State, PID/Program name, and a timestamp. Key entries include:

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	192.168.81.232:4000	0.0.0.0:*	LISTEN	4222./tcp_server5
tcp	0	0	0.0.0.0:3306	0.0.0.0:*	LISTEN	2075/mysql
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	264071	0	192.168.81.232:4000	192.168.81.233:4001	ESTABLISHED	4222./tcp_server5
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
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

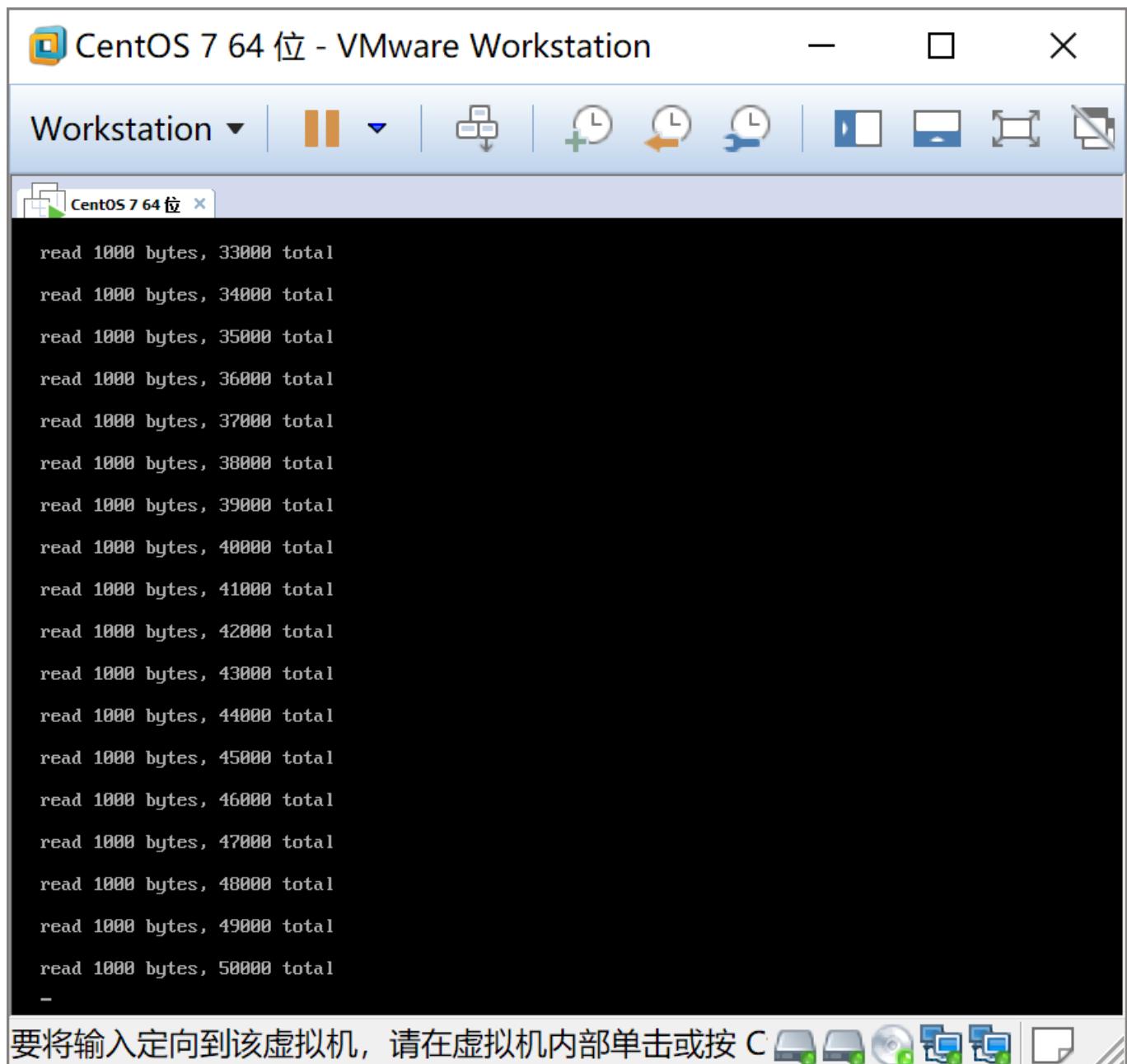
手动测试server读取多少自己后client会解除阻塞

每次一个getchar()写1000个字节，写到大约32w字节后阻塞



要将输入定向到该虚拟机，请在虚拟机内部单击或:

server端读取大约45千字节后，client解除阻塞



```
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
read 1000 bytes, 39000 total
read 1000 bytes, 40000 total
read 1000 bytes, 41000 total
read 1000 bytes, 42000 total
read 1000 bytes, 43000 total
read 1000 bytes, 44000 total
read 1000 bytes, 45000 total
read 1000 bytes, 46000 total
read 1000 bytes, 47000 total
read 1000 bytes, 48000 total
read 1000 bytes, 49000 total
read 1000 bytes, 50000 total
-
-
要将输入定向到该虚拟机，请在虚拟机内部单击或按 C
```

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

```
[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

```

3. centos
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.*              LISTEN    1597/smbd
tcp     0      0 192.168.81.232:4001       0.0.0.*              LISTEN    4247/.tcp_server5-
tcp     0      0 0.0.0.0:3306             0.0.0.*              LISTEN    2075/mysql
tcp     0      0 0.0.0.0:139              0.0.0.*              LISTEN    1597/smbd
tcp     0      0 0.0.0.0:111              0.0.0.*              LISTEN    1055/rpcbind
tcp     0      0 0.0.0.0:22               0.0.0.*              LISTEN    1593/sshd
tcp     0      0 127.0.0.1:25             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   232080  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 ::*:125                ::*:                LISTEN    2398/master

2. centos (1)
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.*              LISTEN    1373/smbd
tcp     0      0 0.0.0.0:3306             0.0.0.*              LISTEN    1755/mysql
tcp     0      0 0.0.0.0:139              0.0.0.*              LISTEN    1373/smbd
tcp     0      0 0.0.0.0:111              0.0.0.*              LISTEN    898/rpcbind
tcp     0      0 0.0.0.0:22               0.0.0.*              LISTEN    1382/sshd
tcp     0      0 127.0.0.1:25             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 221264  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      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 ::*:125                ::*:                LISTEN    1779/master

```

netstat可以带哪些参数？说明什么？

-a 列出所有端口

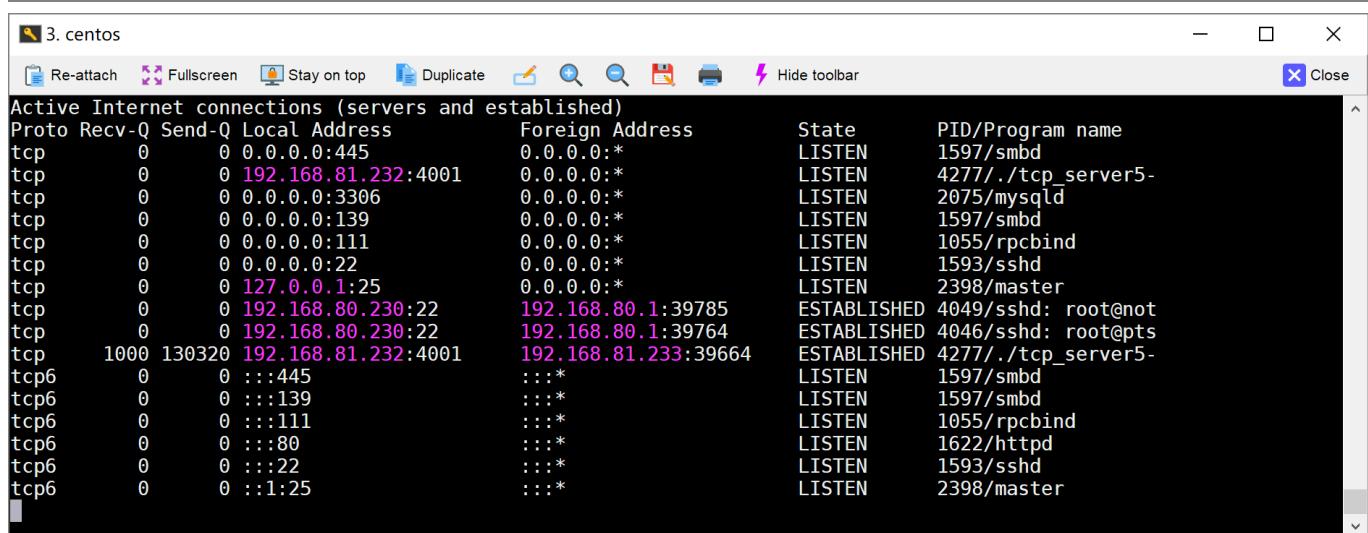
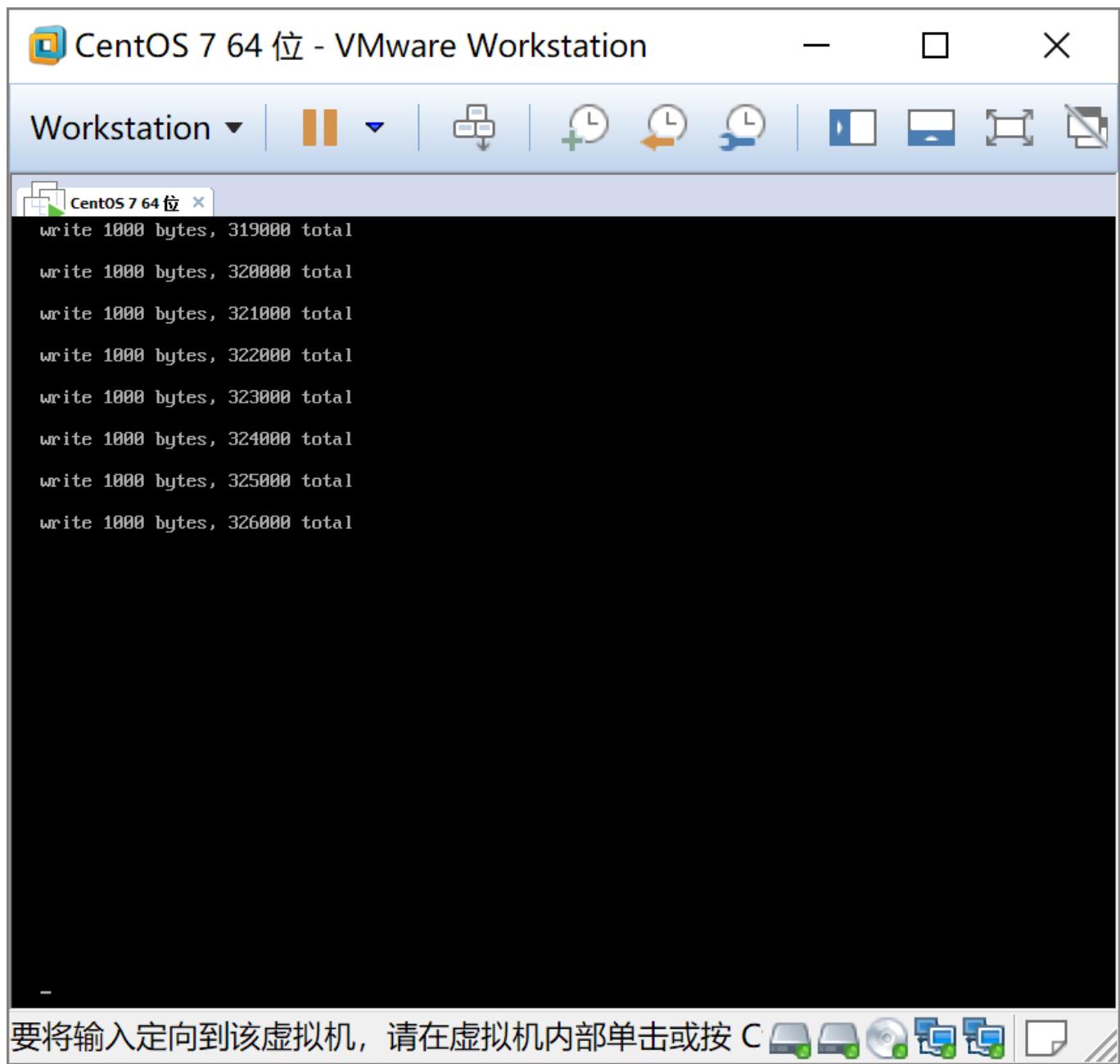
-t 只显示tcp -u 只显示udp -x 只显示unix端口 -p 显示进程和进程名称 -n 不显示别名，全部显示数字 -c 持续输出

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

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	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

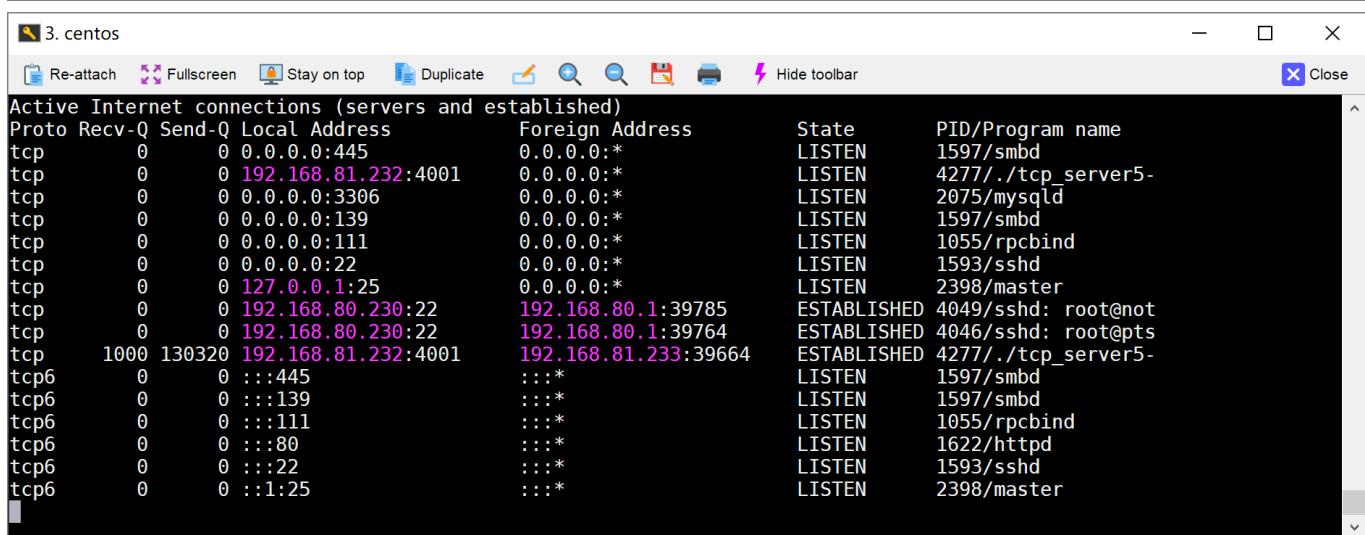
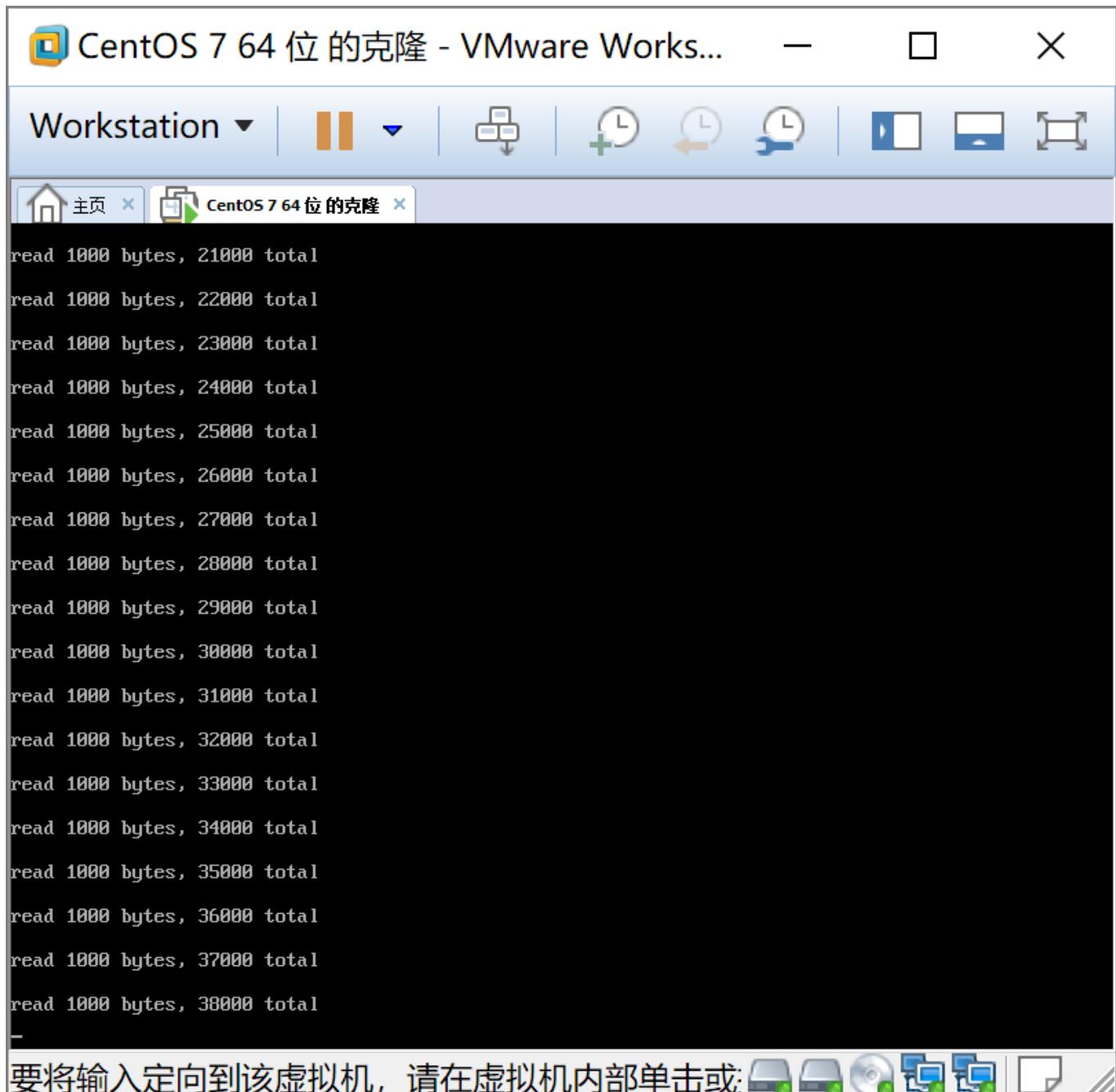
测试程序tcp_server2与tcp_client2，双方角色互换，server写，client读

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



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	196384	0	192.168.81.233:39664	192.168.81.232:4001	ESTABLISHED	2496/. ./tcp_client5-
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	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

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



2. centos (1)						
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	196384	0	192.168.81.233:39664	192.168.81.232:4001	ESTABLISHED	2496/. ./tcp_client5
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	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

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sendq=1448, recvq=1724, sendq+recvq=3172

```
[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:

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/mysqlld
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	:::125	:::*	LISTEN	1779/master

```
[root@azuse 05]# ./tcp_server5-3 192.168.81.232 4002
-----
listen 192.168.81.232 port=4002
-----
INTERFACE_ADDRESS: 1: 127.0.0.1
INTERFACE_ADDRESS: 2: 192.168.80.230
INTERFACE_ADDRESS: 3: 192.168.81.232
INTERFACE_ADDRESS: 4: 192.168.1.168
INTERFACE_ADDRESS: 5: ::1
INTERFACE_ADDRESS: 6: fe80::20c:29ff:fed:a302d%ens32
INTERFACE_ADDRESS: 7: fe80::20c:29ff:fed:a3037%ens34
-----
listen address: 192.168.81.232
-----
RCVBUF SET: 4000
bind success
listening
192.168.81.233 connected port=4002
waiting for recv, press anykey to start
```

要将输入定向到该虚拟机，请在虚拟机内部单击或按 C

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	192.168.81.232:4002	0.0.0.0:*	LISTEN	4585/.tcp_server5-
tcp	0	0	0.0.0.0:3306	0.0.0.0:*	LISTEN	2075/mysqlld
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	1724	0	192.168.81.232:4002	192.168.81.233:4002	ESTABLISHED	4585/.tcp_server5-
tcp	0	0	192.168.80.230:22	192.168.80.1:39764	ESTABLISHED	4046/sshd: root@pts
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	:::125	:::*	LISTEN	2398/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：

```
[root@azuse 06]# ./tcp_server6-1 4003 1000 900
监听 0.0.0.0 port=4003
每次读1000字节，每次写900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
```

centos (1)

Terminal Sessions View X server Tools Games Settings Macros Help

Quick connect... os (1) 6. centos 7. centos (1)

```
[root@azuse 06]# ./tcp_client6-3 192.168.81.232 4003 800 700
> 连接ip 192.168.81.232
连接端口 4003
一次读800 一次写700
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>

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

死锁：

```
[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
```

```
[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]#
```

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：

The screenshot shows the MobaXterm interface with multiple sessions listed in the top bar: 'centos (1)', 'ios (1)', '6. centos', and '7. centos (1)'. The main window displays session '2. centos' which contains a log of file operations:

```
read 1000 bytes, 5620000 total
> write 1000 bytes, 5620000 total
read 1000 bytes, 5621000 total
write 1000 bytes, 5621000 total
read 1000 bytes, 5622000 total
write 1000 bytes, 5622000 total
read 1000 bytes, 5623000 total
write 1000 bytes, 5623000 total
read 1000 bytes, 5624000 total
write 1000 bytes, 5624000 total
read 1000 bytes, 5625000 total
write 1000 bytes, 5625000 total
read 1000 bytes, 5626000 total
write 1000 bytes, 5626000 total
read 1000 bytes, 5627000 total
write 1000 bytes, 5627000 total
read 1000 bytes, 5628000 total
write 1000 bytes, 5628000 total
read 1000 bytes, 5629000 total
write 1000 bytes, 5629000 total
read 1000 bytes, 5630000 total
write 1000 bytes, 5630000 total
read 1000 bytes, 5631000 total
```

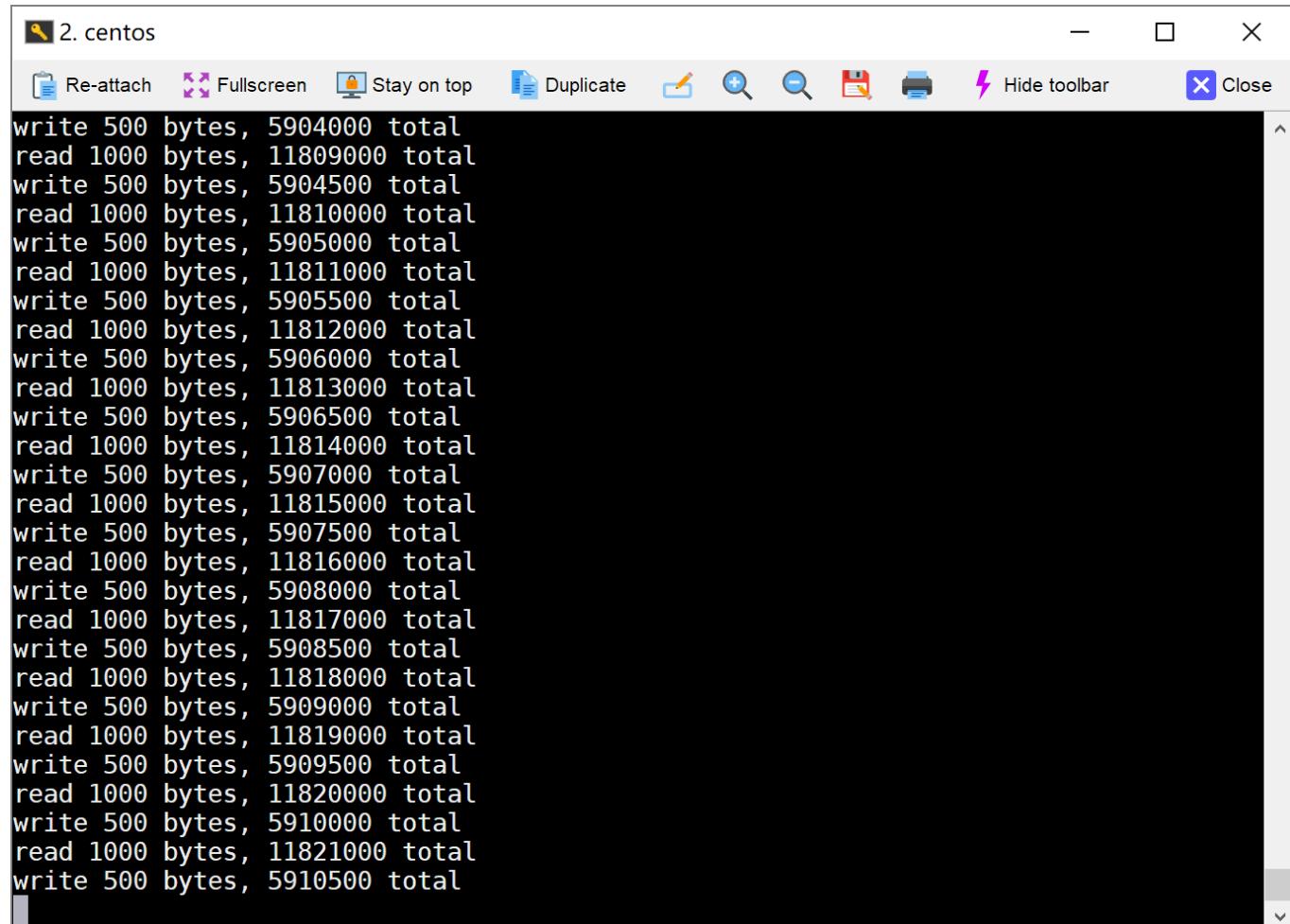
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <http://mobaxterm.mobatek.net>

The screenshot shows the MobaXterm interface with session '2. centos' selected. The toolbar at the top includes icons for Re-attach, Fullscreen, Stay on top, Duplicate, Edit, Find, Copy, Paste, Hide toolbar, and Close. The main window displays a log of file operations:

```
write 1000 bytes, 4652000 total
read 1000 bytes, 4653000 total
write 1000 bytes, 4653000 total
read 1000 bytes, 4654000 total
write 1000 bytes, 4654000 total
read 1000 bytes, 4655000 total
write 1000 bytes, 4655000 total
read 1000 bytes, 4656000 total
write 1000 bytes, 4656000 total
read 1000 bytes, 4657000 total
write 1000 bytes, 4657000 total
read 1000 bytes, 4658000 total
write 1000 bytes, 4658000 total
read 1000 bytes, 4659000 total
write 1000 bytes, 4659000 total
read 1000 bytes, 4660000 total
write 1000 bytes, 4660000 total
read 1000 bytes, 4661000 total
write 1000 bytes, 4661000 total
read 1000 bytes, 4662000 total
write 1000 bytes, 4662000 total
read 1000 bytes, 4663000 total
write 1000 bytes, 4663000 total
read 1000 bytes, 4664000 total
write 1000 bytes, 4664000 total
read 1000 bytes, 4665000 total
write 1000 bytes, 4665000 total
```

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

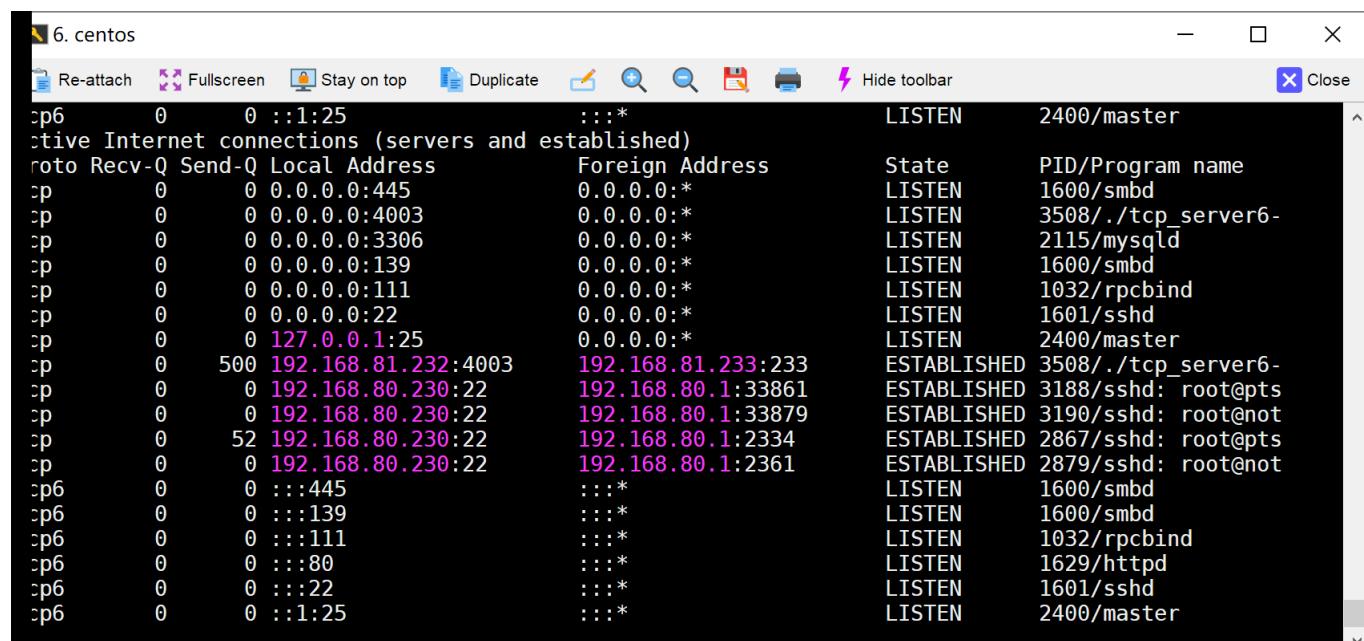
正常读写，不过server有500字节的sendq, client有1000字节的sendq:



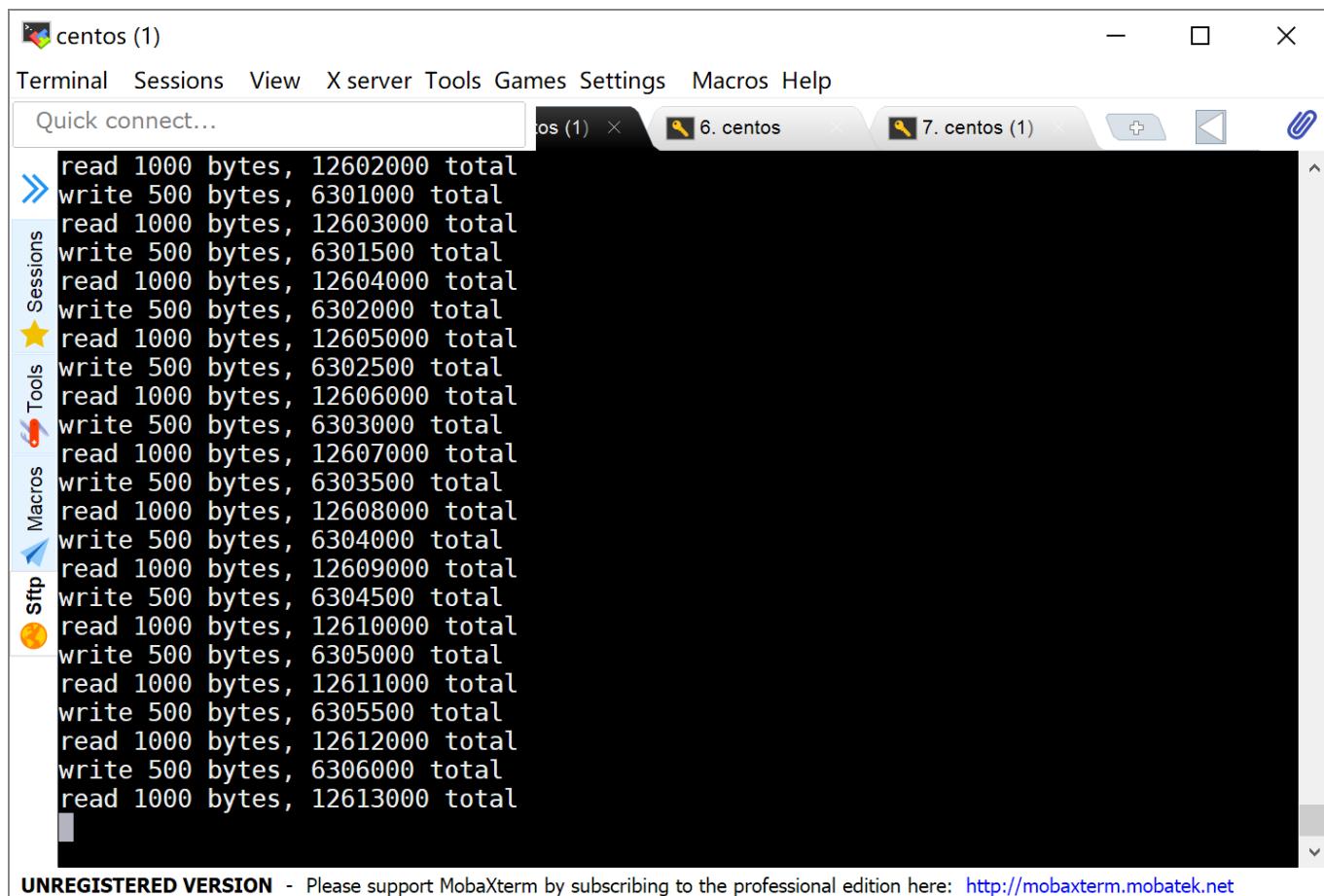
```

write 500 bytes, 5904000 total
read 1000 bytes, 11809000 total
write 500 bytes, 5904500 total
read 1000 bytes, 11810000 total
write 500 bytes, 5905000 total
read 1000 bytes, 11811000 total
write 500 bytes, 5905500 total
read 1000 bytes, 11812000 total
write 500 bytes, 5906000 total
read 1000 bytes, 11813000 total
write 500 bytes, 5906500 total
read 1000 bytes, 11814000 total
write 500 bytes, 5907000 total
read 1000 bytes, 11815000 total
write 500 bytes, 5907500 total
read 1000 bytes, 11816000 total
write 500 bytes, 5908000 total
read 1000 bytes, 11817000 total
write 500 bytes, 5908500 total
read 1000 bytes, 11818000 total
write 500 bytes, 5909000 total
read 1000 bytes, 11819000 total
write 500 bytes, 5909500 total
read 1000 bytes, 11820000 total
write 500 bytes, 5910000 total
read 1000 bytes, 11821000 total
write 500 bytes, 5910500 total

```

Local Address	Foreign Address	State	PID/Program name
0 ::1:25	::*	LISTEN	2400/master
0 0.0.0.0:445	0.0.0.0:*	LISTEN	1600/smbd
0 0.0.0.0:4003	0.0.0.0:*	LISTEN	3508./tcp_server6-
0 0.0.0.0:3306	0.0.0.0:*	LISTEN	2115/mysqld
0 0.0.0.0:139	0.0.0.0:*	LISTEN	1600/smbd
0 0.0.0.0:111	0.0.0.0:*	LISTEN	1032/rpcbind
0 0.0.0.0:22	0.0.0.0:*	LISTEN	1601/sshd
0 127.0.0.1:25	0.0.0.0:*	LISTEN	2400/master
0 500 192.168.81.232:4003	192.168.81.233:233	ESTABLISHED	3508./tcp_server6-
0 0 192.168.80.230:22	192.168.80.1:33861	ESTABLISHED	3188/sshd: root@pts
0 0 192.168.80.230:22	192.168.80.1:33879	ESTABLISHED	3190/sshd: root@not
0 52 192.168.80.230:22	192.168.80.1:2334	ESTABLISHED	2867/sshd: root@pts
0 0 192.168.80.230:22	192.168.80.1:2361	ESTABLISHED	2879/sshd: root@not
0 0 ::1:445	::*	LISTEN	1600/smbd
0 0 ::1:139	::*	LISTEN	1600/smbd
0 0 ::1:111	::*	LISTEN	1032/rpcbind
0 0 ::1:80	::*	LISTEN	1629/httpd
0 0 ::1:22	::*	LISTEN	1601/sshd
0 0 ::1:25	::*	LISTEN	2400/master

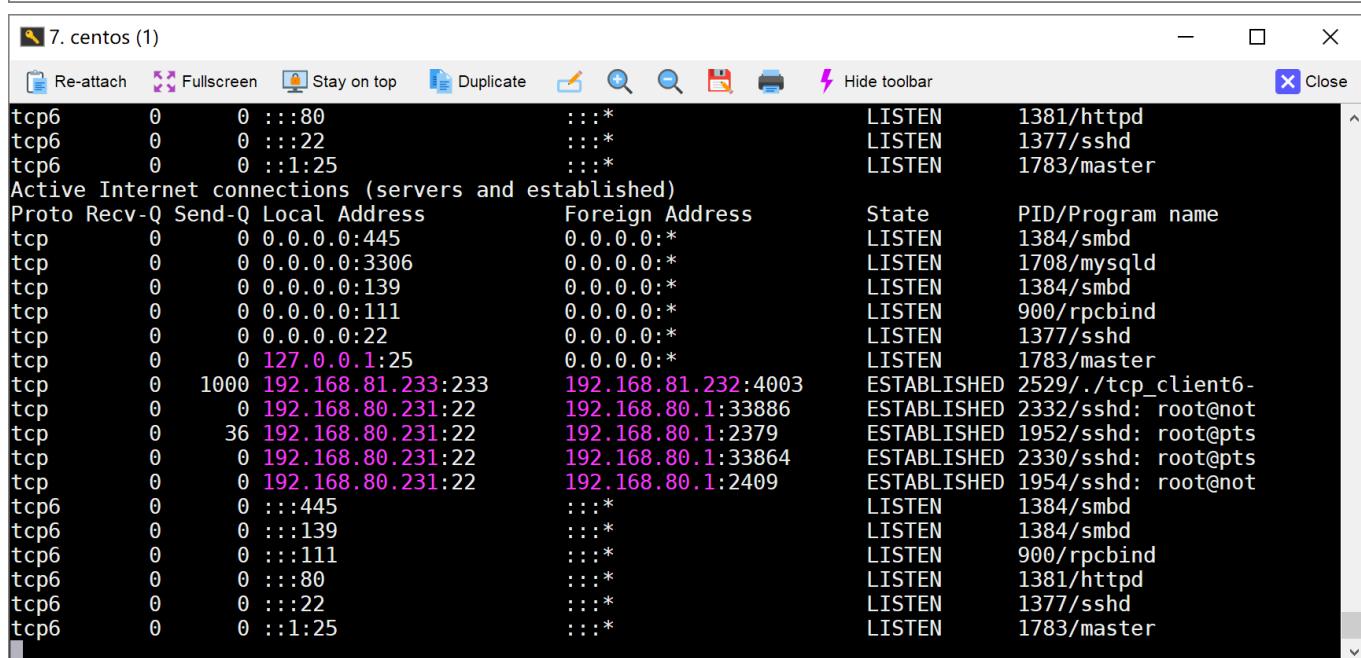


```

read 1000 bytes, 12602000 total
> write 500 bytes, 6301000 total
read 1000 bytes, 12603000 total
write 500 bytes, 6301500 total
read 1000 bytes, 12604000 total
write 500 bytes, 6302000 total
read 1000 bytes, 12605000 total
write 500 bytes, 6302500 total
read 1000 bytes, 12606000 total
write 500 bytes, 6303000 total
read 1000 bytes, 12607000 total
write 500 bytes, 6303500 total
read 1000 bytes, 12608000 total
write 500 bytes, 6304000 total
read 1000 bytes, 12609000 total
write 500 bytes, 6304500 total
read 1000 bytes, 12610000 total
write 500 bytes, 6305000 total
read 1000 bytes, 12611000 total
write 500 bytes, 6305500 total
read 1000 bytes, 12612000 total
write 500 bytes, 6306000 total
read 1000 bytes, 12613000 total

```

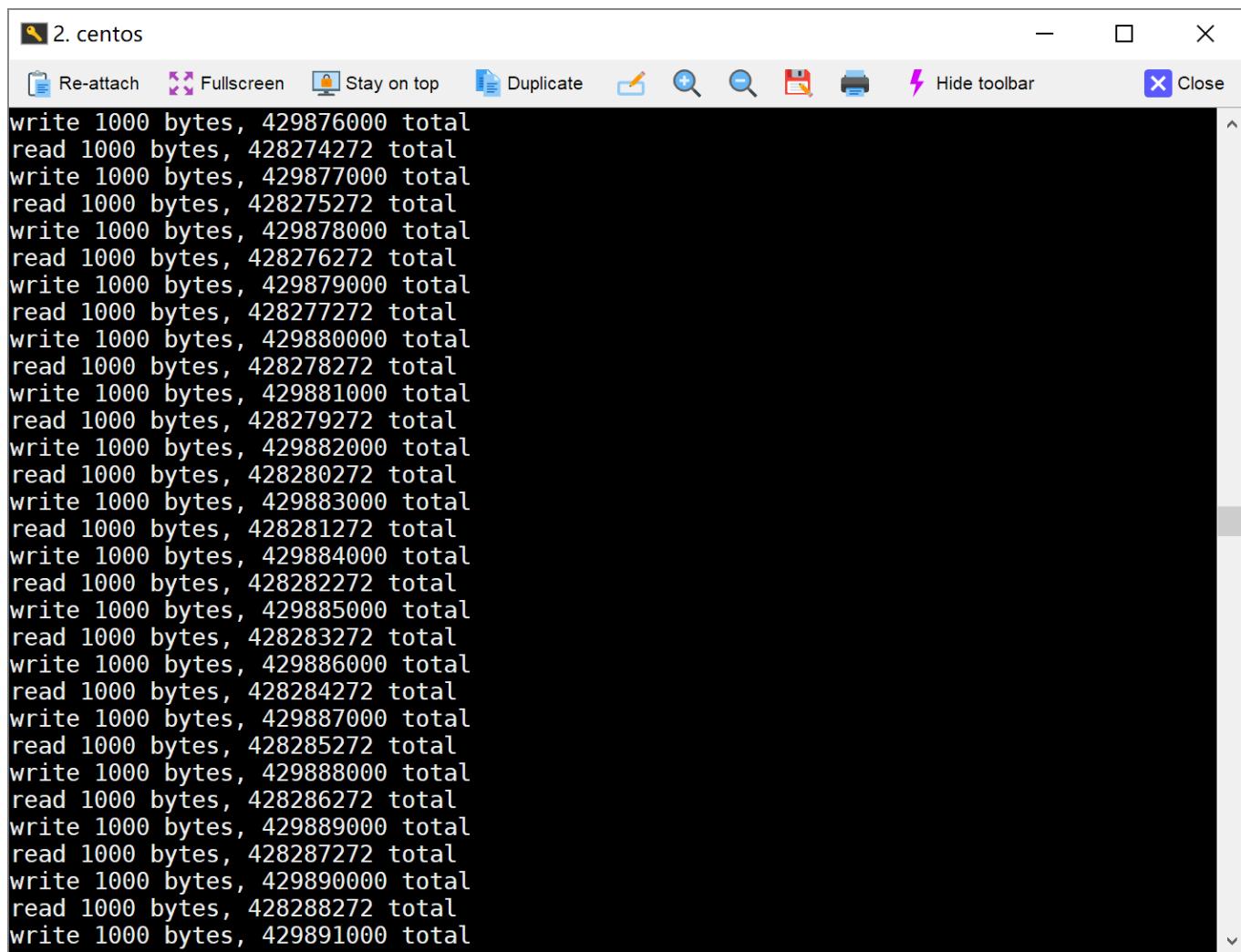
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <http://mobaxterm.mobatek.net>



Proto	Recv-Q	Send-Q	Local Address	Foreign Address	State	PID/Program name
tcp6	0	0	:::80	:::*	LISTEN	1381/httpd
tcp6	0	0	:::22	:::*	LISTEN	1377/sshd
tcp6	0	0	:::1:25	:::*	LISTEN	1783/master
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	1384/smbd
tcp	0	0	0.0.0.0:3306	0.0.0.0:*	LISTEN	1708/mysqld
tcp	0	0	0.0.0.0:139	0.0.0.0:*	LISTEN	1384/smbd
tcp	0	0	0.0.0.0:111	0.0.0.0:*	LISTEN	900/rpcbind
tcp	0	0	0.0.0.0:22	0.0.0.0:*	LISTEN	1377/sshd
tcp	0	0	127.0.0.1:25	0.0.0.0:*	LISTEN	1783/master
tcp	0	1000	192.168.81.233:233	192.168.81.232:4003	ESTABLISHED	2529/.tcp_client6-
tcp	0	0	192.168.80.231:22	192.168.80.1:33886	ESTABLISHED	2332/sshd: root@not
tcp	0	36	192.168.80.231:22	192.168.80.1:2379	ESTABLISHED	1952/sshd: root@pts
tcp	0	0	192.168.80.231:22	192.168.80.1:33864	ESTABLISHED	2330/sshd: root@pts
tcp	0	0	192.168.80.231:22	192.168.80.1:2409	ESTABLISHED	1954/sshd: root@not
tcp6	0	0	:::445	:::*	LISTEN	1384/smbd
tcp6	0	0	:::139	:::*	LISTEN	1384/smbd
tcp6	0	0	:::111	:::*	LISTEN	900/rpcbind
tcp6	0	0	:::80	:::*	LISTEN	1381/httpd
tcp6	0	0	:::22	:::*	LISTEN	1377/sshd
tcp6	0	0	:::1:25	:::*	LISTEN	1783/master

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。



The screenshot shows a terminal window titled "2. centos". The window contains a log of network operations, likely from a client application. The log consists of many lines of text, each starting with either "write" or "read" followed by "1000 bytes" and a timestamp. The timestamps show a regular pattern of activity, with "write" operations occurring every 1000ms and "read" operations occurring every 1000ms. The log ends with a final "write" operation at the bottom.

```
write 1000 bytes, 429876000 total
read 1000 bytes, 428274272 total
write 1000 bytes, 429877000 total
read 1000 bytes, 428275272 total
write 1000 bytes, 429878000 total
read 1000 bytes, 428276272 total
write 1000 bytes, 429879000 total
read 1000 bytes, 428277272 total
write 1000 bytes, 429880000 total
read 1000 bytes, 428278272 total
write 1000 bytes, 429881000 total
read 1000 bytes, 428279272 total
write 1000 bytes, 429882000 total
read 1000 bytes, 428280272 total
write 1000 bytes, 429883000 total
read 1000 bytes, 428281272 total
write 1000 bytes, 429884000 total
read 1000 bytes, 428282272 total
write 1000 bytes, 429885000 total
read 1000 bytes, 428283272 total
write 1000 bytes, 429886000 total
read 1000 bytes, 428284272 total
write 1000 bytes, 429887000 total
read 1000 bytes, 428285272 total
write 1000 bytes, 429888000 total
read 1000 bytes, 428286272 total
write 1000 bytes, 429889000 total
read 1000 bytes, 428287272 total
write 1000 bytes, 429890000 total
read 1000 bytes, 428288272 total
write 1000 bytes, 429891000 total
```

client端每次可以稳定读写700比特。

```

>> write 700 bytes, 420448736 total
>> read 700 bytes, 422853200 total
>> write 700 bytes, 420449436 total
>> read 700 bytes, 422853900 total
>> write 700 bytes, 420450136 total
>> read 700 bytes, 422854600 total
>> write 700 bytes, 420450836 total
>> read 700 bytes, 422855300 total
>> write 700 bytes, 420451536 total
>> read 700 bytes, 422856000 total
>> write 700 bytes, 420452236 total
>> read 700 bytes, 422856700 total
>> write 700 bytes, 420452936 total
>> read 700 bytes, 422857400 total
>> write 700 bytes, 420453636 total
>> read 700 bytes, 422858100 total
>> write 700 bytes, 420454336 total
>> read 700 bytes, 422858800 total
>> write 700 bytes, 420455036 total
>> read 700 bytes, 422859500 total
>> write 700 bytes, 420455736 total
>> read 700 bytes, 422860200 total
>> write 700 bytes, 420456436 total
>> read 700 bytes, 422860900 total
>> write 700 bytes, 420457136 total
>> read 700 bytes, 422861600 total
>> write 700 bytes, 420457836 total
>> read 700 bytes, 422862300 total
>> write 700 bytes, 420458536 total
>> read 700 bytes, 422863000 total
>> write 700 bytes, 420459236 total

```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <http://mobaxterm.mobatek.net>

同时server的recvq大量积压:

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.*	LISTEN	1598/smbd
tcp	0	0	0.0.0.0:4002	0.0.0.*	LISTEN	3011/.tcp_server6-
tcp	0	0	0.0.0.0:3306	0.0.0.*	LISTEN	2066/mysqld
tcp	0	0	0.0.0.0:139	0.0.0.*	LISTEN	1598/smbd
tcp	0	0	0.0.0.0:111	0.0.0.*	LISTEN	1031/rpcbind
tcp	0	0	0.0.0.0:22	0.0.0.*	LISTEN	1608/sshd
tcp	0	0	127.0.0.1:25	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	1570352	0	192.168.81.232:4002	192.168.81.233:233	ESTABLISHED	3011/.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	:::125	::*	LISTEN	2362/master

client的sendq和recvq大量积压:

4. centos (1)						
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	1324864	1303200	192.168.81.233:233	192.168.81.232:4002	ESTABLISHED	2173/.tcp_client6-
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	0	192.168.80.231:22	192.168.80.1:32873	ESTABLISHED	1955/sshd: root@pts
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

tcp6-3 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。

The screenshot shows the MobaXterm interface with multiple sessions open. Session 1 (centos) is active and displays a terminal window with the following output:

```
read 1000 bytes, 445000 total
> write 1000 bytes, 445000 total
read 1000 bytes, 446000 total
write 1000 bytes, 446000 total
read 1000 bytes, 447000 total
write 1000 bytes, 447000 total
read 1000 bytes, 448000 total
write 1000 bytes, 448000 total
read 1000 bytes, 449000 total
write 1000 bytes, 449000 total
read 1000 bytes, 450000 total
write 1000 bytes, 450000 total
read 1000 bytes, 451000 total
write 1000 bytes, 451000 total
read 1000 bytes, 452000 total
write 1000 bytes, 452000 total
read 1000 bytes, 453000 total
write 1000 bytes, 453000 total
read 1000 bytes, 454000 total
write 1000 bytes, 454000 total
read 1000 bytes, 455000 total
write 1000 bytes, 455000 total
^C
[root@azuse 06]#
```

Session 2 (ios) is also visible in the background.

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <http://mobaxterm.mobatek.net>

The screenshot shows the MobaXterm interface with multiple sessions open. Session 2 (centos) is active and displays a terminal window with the following output:

```
read 1000 bytes, 443000 total
write 1000 bytes, 444000 total
read 1000 bytes, 444000 total
write 1000 bytes, 445000 total
read 1000 bytes, 445000 total
write 1000 bytes, 446000 total
read 1000 bytes, 446000 total
write 1000 bytes, 447000 total
read 1000 bytes, 447000 total
write 1000 bytes, 448000 total
read 1000 bytes, 448000 total
write 1000 bytes, 449000 total
read 1000 bytes, 449000 total
write 1000 bytes, 450000 total
read 1000 bytes, 450000 total
write 1000 bytes, 451000 total
read 1000 bytes, 451000 total
write 1000 bytes, 452000 total
read 1000 bytes, 452000 total
write 1000 bytes, 453000 total
read 1000 bytes, 453000 total
write 1000 bytes, 454000 total
read 1000 bytes, 454000 total
write 1000 bytes, 455000 total
read 1000 bytes, 455000 total
write 1000 bytes, 456000 total
read error
[root@azuse 06]#
```

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

双方正常通信：

The screenshot shows a terminal window titled "7. centos (1)" running on a Windows host. The terminal displays several lines of network traffic log output from a Linux system. The log entries show various read and write operations at the byte level, such as "read 1000 bytes, 5169000 total" and "write 500 bytes, 2584500 total". Below the log, a message reads: "UNREGISTERED VERSION - Please support MobaTerm by subscribing to the professional edition here: <http://mobaterm.mobatek.net>".

At the bottom of the terminal window, there is a table titled "Active Internet connections (servers and established)". It lists network connections with columns for Local Address, Foreign Address, Proto, Recv-Q, Send-Q, State, and PID/Program name. Some connections are established (ESTABLISHED), while others are listening (LISTEN). Examples include "tcp6 0 0 ::1:25 LISTEN 1381/httpd" and "tcp6 0 0 ::1:25 LISTEN 1783/master".

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
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 Xserver Tools Games Settings Macros Help
Quick connect...
Sessions
Tools
Macros
SFTP
2. centos
3. centos (1)
4. centos (+)
> 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

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <http://mobaxterm.mobatek.net>

Proto	Recv-Q	Send-Q	Local Address	Foreign Address	State	PID/Program name
tcp	0	0	0.0.0.0:445	0.0.0.*	LISTEN	1365/smbd
tcp	0	0	0.0.0.0:4003	0.0.0.*	LISTEN	3048/.tcp_server6-
tcp	0	0	0.0.0.0:3306	0.0.0.*	LISTEN	2066/mysql
tcp	0	0	0.0.0.0:139	0.0.0.*	LISTEN	1598/smbd
tcp	0	0	0.0.0.0:111	0.0.0.*	LISTEN	1031/rpcbind
tcp	0	0	0.0.0.0:22	0.0.0.*	LISTEN	1608/sshd
tcp	0	0	127.0.0.1:25	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	192.168.81.232:58094	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	:::125	::*	LISTEN	2362/master

Proto	Recv-Q	Send-Q	Local Address	Foreign Address	State	PID/Program name
tcp	0	0	0.0.0.0:445	0.0.0.*	LISTEN	1365/smbd
tcp	0	0	0.0.0.0:3306	0.0.0.*	LISTEN	1739/mysql
tcp	0	0	0.0.0.0:139	0.0.0.*	LISTEN	1365/smbd
tcp	0	0	0.0.0.0:111	0.0.0.*	LISTEN	887/rpcbind
tcp	0	0	0.0.0.0:22	0.0.0.*	LISTEN	1378/sshd
tcp	0	0	127.0.0.1:25	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	:::125	::*	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

The screenshot shows the MobaXterm interface with multiple sessions listed in the sidebar: Sessions (1), Tools, Macros, Sftp, and Session 1 (centos). The main window displays a terminal session titled 'centos (1)' with the following log output:

```
write 1000 bytes, 3401000 total
> read 1000 bytes, 3401000 total
write 1000 bytes, 3402000 total
read 1000 bytes, 3402000 total
write 1000 bytes, 3403000 total
read 1000 bytes, 3403000 total
write 1000 bytes, 3404000 total
read 1000 bytes, 3404000 total
write 1000 bytes, 3405000 total
read 1000 bytes, 3405000 total
write 1000 bytes, 3406000 total
read 1000 bytes, 3406000 total
write 1000 bytes, 3407000 total
read 1000 bytes, 3407000 total
write 1000 bytes, 3408000 total
read 1000 bytes, 3408000 total
write 1000 bytes, 3409000 total
read 1000 bytes, 3409000 total
write 1000 bytes, 3410000 total
read 1000 bytes, 3410000 total
write 1000 bytes, 3411000 total
read 1000 bytes, 3411000 total
write 1000 bytes, 3403000 total
```

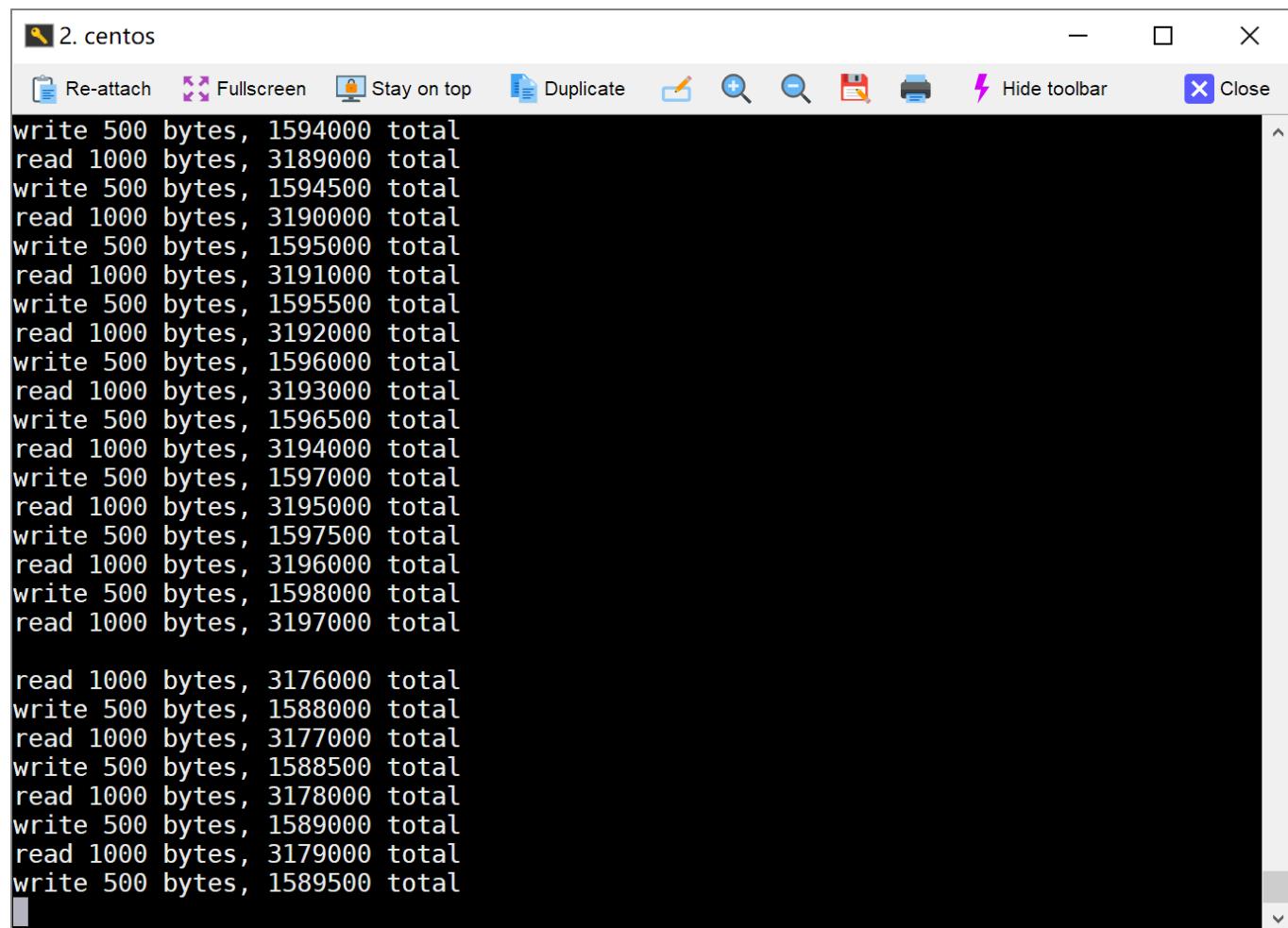
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <http://mobaxterm.mobatek.net>

The screenshot shows the MobaXterm interface with multiple sessions listed in the sidebar: Sessions (1), Tools, Macros, Sftp, and Session 2 (centos). The main window displays a terminal session titled '2. centos' with the following log output:

```
write 1000 bytes, 2130000 total
read 1000 bytes, 2131000 total
write 1000 bytes, 2131000 total
read 1000 bytes, 2132000 total
write 1000 bytes, 2132000 total
read 1000 bytes, 2133000 total
write 1000 bytes, 2133000 total
read 1000 bytes, 2134000 total
write 1000 bytes, 2134000 total
read 1000 bytes, 2135000 total
write 1000 bytes, 2135000 total
read 1000 bytes, 2136000 total
write 1000 bytes, 2136000 total
read 1000 bytes, 2137000 total
write 1000 bytes, 2137000 total
read 1000 bytes, 2138000 total
write 1000 bytes, 2138000 total
read 1000 bytes, 2139000 total
write 1000 bytes, 2139000 total
read 1000 bytes, 2140000 total
write 1000 bytes, 2140000 total
read 1000 bytes, 2141000 total
write 1000 bytes, 2141000 total
read 1000 bytes, 2142000 total
write 1000 bytes, 2142000 total
read 1000 bytes, 2143000 total
write 1000 bytes, 2143000 total
```

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

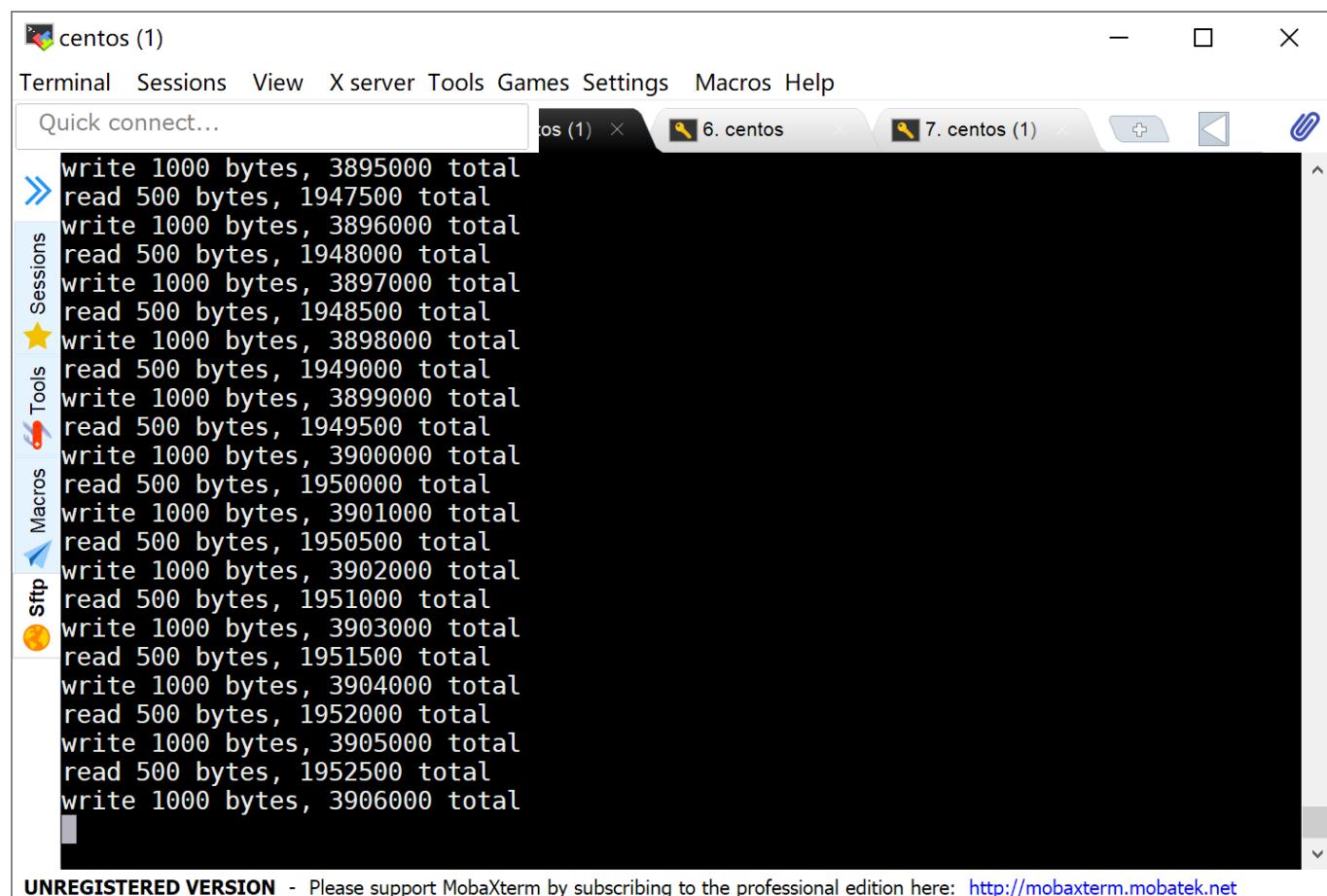
双方可以正常通信，无丢失



2. centos

write 500 bytes, 1594000 total
read 1000 bytes, 3189000 total
write 500 bytes, 1594500 total
read 1000 bytes, 3190000 total
write 500 bytes, 1595000 total
read 1000 bytes, 3191000 total
write 500 bytes, 1595500 total
read 1000 bytes, 3192000 total
write 500 bytes, 1596000 total
read 1000 bytes, 3193000 total
write 500 bytes, 1596500 total
read 1000 bytes, 3194000 total
write 500 bytes, 1597000 total
read 1000 bytes, 3195000 total
write 500 bytes, 1597500 total
read 1000 bytes, 3196000 total
write 500 bytes, 1598000 total
read 1000 bytes, 3197000 total

read 1000 bytes, 3176000 total
write 500 bytes, 1588000 total
read 1000 bytes, 3177000 total
write 500 bytes, 1588500 total
read 1000 bytes, 3178000 total
write 500 bytes, 1589000 total
read 1000 bytes, 3179000 total
write 500 bytes, 1589500 total



centos (1)

Terminal Sessions View X server Tools Games Settings Macros Help

Quick connect... 5 os (1) 6. centos 7. centos (1)

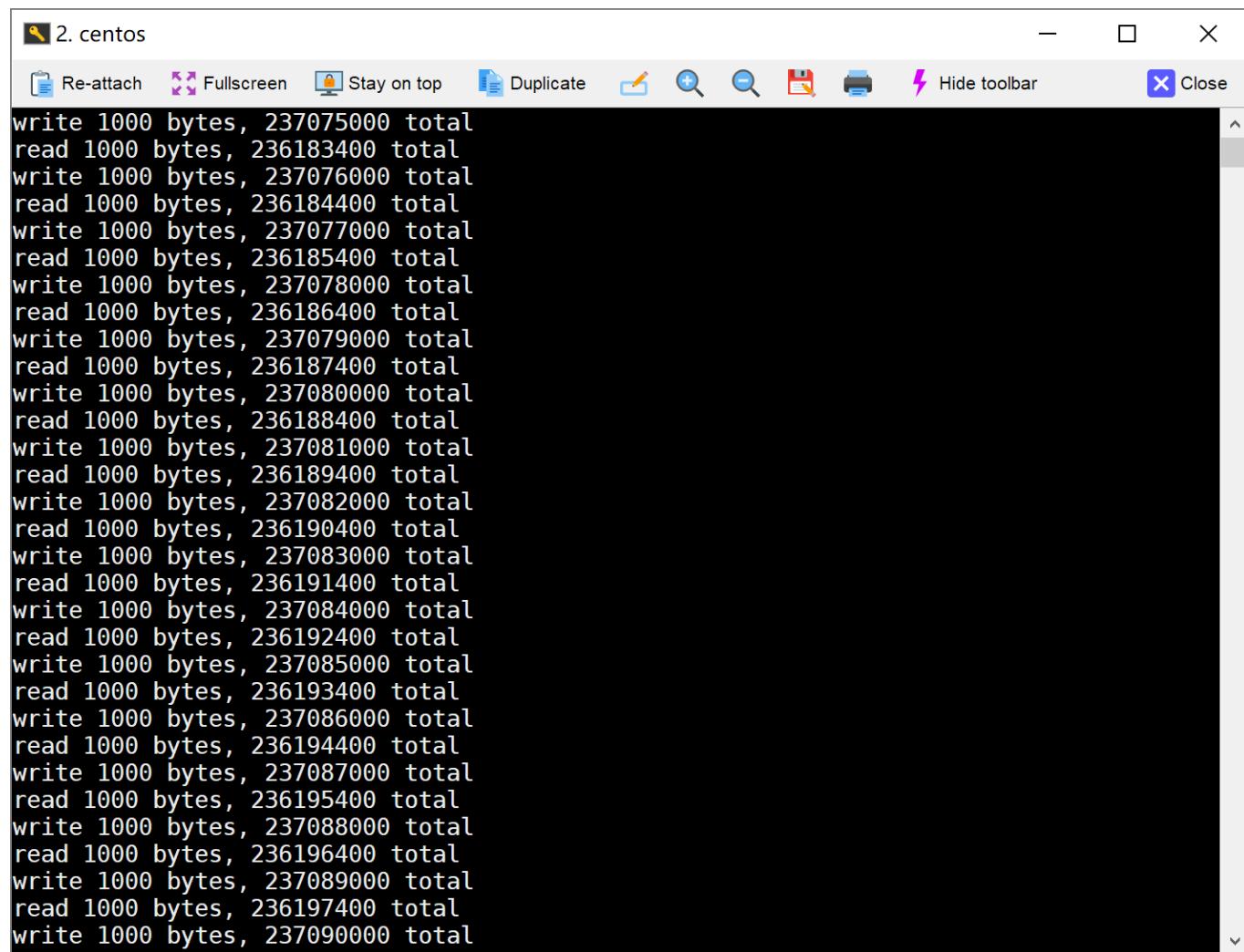
Sessions Tools Macros SFTP

write 1000 bytes, 3895000 total
read 500 bytes, 1947500 total
write 1000 bytes, 3896000 total
read 500 bytes, 1948000 total
write 1000 bytes, 3897000 total
read 500 bytes, 1948500 total
write 1000 bytes, 3898000 total
read 500 bytes, 1949000 total
write 1000 bytes, 3899000 total
read 500 bytes, 1949500 total
write 1000 bytes, 3900000 total
read 500 bytes, 1950000 total
write 1000 bytes, 3901000 total
read 500 bytes, 1950500 total
write 1000 bytes, 3902000 total
read 500 bytes, 1951000 total
write 1000 bytes, 3903000 total
read 500 bytes, 1951500 total
write 1000 bytes, 3904000 total
read 500 bytes, 1952000 total
write 1000 bytes, 3905000 total
read 500 bytes, 1952500 total
write 1000 bytes, 3906000 total

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <http://mobaxterm.mobatek.net>

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

server读写1000字节。



```
2. centos
write 1000 bytes, 237075000 total
read 1000 bytes, 236183400 total
write 1000 bytes, 237076000 total
read 1000 bytes, 236184400 total
write 1000 bytes, 237077000 total
read 1000 bytes, 236185400 total
write 1000 bytes, 237078000 total
read 1000 bytes, 236186400 total
write 1000 bytes, 237079000 total
read 1000 bytes, 236187400 total
write 1000 bytes, 237080000 total
read 1000 bytes, 236188400 total
write 1000 bytes, 237081000 total
read 1000 bytes, 236189400 total
write 1000 bytes, 237082000 total
read 1000 bytes, 236190400 total
write 1000 bytes, 237083000 total
read 1000 bytes, 236191400 total
write 1000 bytes, 237084000 total
read 1000 bytes, 236192400 total
write 1000 bytes, 237085000 total
read 1000 bytes, 236193400 total
write 1000 bytes, 237086000 total
read 1000 bytes, 236194400 total
write 1000 bytes, 237087000 total
read 1000 bytes, 236195400 total
write 1000 bytes, 237088000 total
read 1000 bytes, 236196400 total
write 1000 bytes, 237089000 total
read 1000 bytes, 236197400 total
write 1000 bytes, 237090000 total
```

client每次读写700字节。

The screenshot shows the MobaXterm interface with four sessions open: 1. centos (1), 2. centos, 3. centos (1), and 4. centos (1). The terminal window displays a series of network log entries from a socket. The log entries show repeated reads and writes of 700 bytes, with totals ranging from 412925948 to 412936448. The log entries are as follows:

```

> read 700 bytes, 412925948 total
> write 700 bytes, 416056900 total
> read 700 bytes, 412926648 total
> write 700 bytes, 416057600 total
> read 700 bytes, 412927348 total
> write 700 bytes, 416058300 total
> read 700 bytes, 412928048 total
> write 700 bytes, 416059000 total
> read 700 bytes, 412928748 total
> write 700 bytes, 416059700 total
> read 700 bytes, 412929448 total
> write 700 bytes, 416060400 total
> read 700 bytes, 412930148 total
> write 700 bytes, 416061100 total
> read 700 bytes, 412930848 total
> write 700 bytes, 416061800 total
> read 700 bytes, 412931548 total
> write 700 bytes, 416062500 total
> read 700 bytes, 412932248 total
> write 700 bytes, 416063200 total
> read 700 bytes, 412932948 total
> write 700 bytes, 416063900 total
> read 700 bytes, 412933648 total
> write 700 bytes, 416064600 total
> read 700 bytes, 412934348 total
> write 700 bytes, 416065300 total
> read 700 bytes, 412935048 total
> write 700 bytes, 416066000 total
> read 700 bytes, 412935748 total
> write 700 bytes, 416066700 total
> read 700 bytes, 412936448 total

```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <http://mobaxterm.mobatek.net>

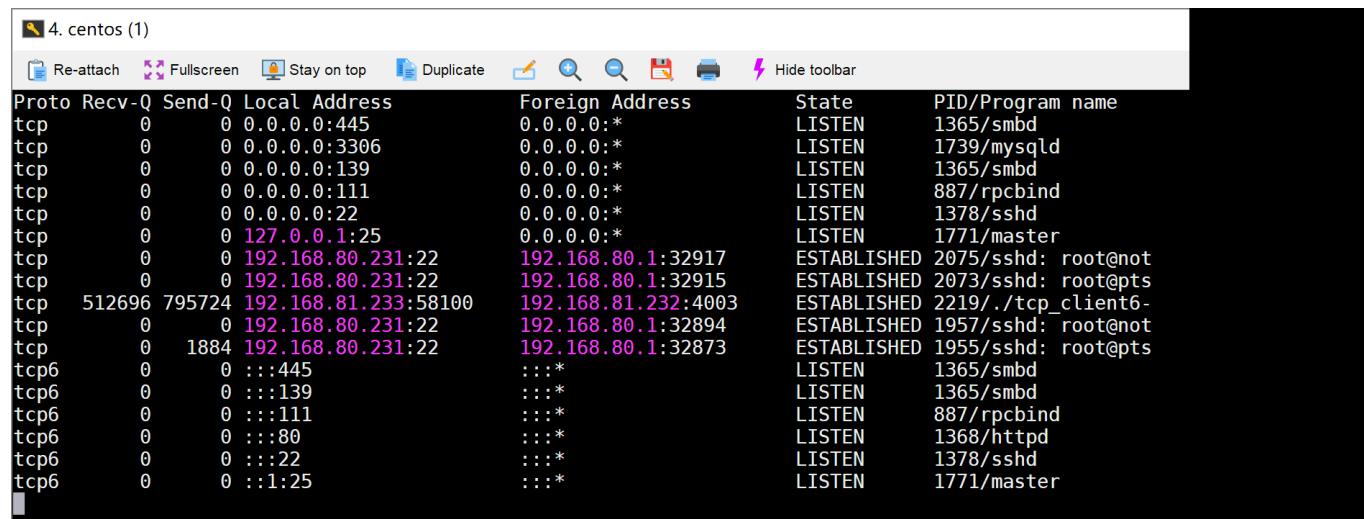
senq与recvq:

server:

The screenshot shows the MobaXterm interface with session 5. centos selected. The terminal window displays the output of the netstat -an command, listing various network connections and listening ports. The output includes columns for protocol, local address, foreign address, status, and process. Key entries include:

Protocol	Local Address	Foreign Address	Status	Process
tcp	0 0.0.0.0:445	0.0.0.0:*	LISTEN	1598/smbd
tcp	0 0.0.0.0:4003	0.0.0.0:*	LISTEN	3051/.tcp_server6-
tcp	0 0.0.0.0:3306	0.0.0.0:*	LISTEN	2066/mysqlld
tcp	0 0.0.0.0:139	0.0.0.0:*	LISTEN	1598/smbd
tcp	0 0.0.0.0:111	0.0.0.0:*	LISTEN	1031/rpcbind
tcp	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	2468668 968 192.168.81.232:4003	192.168.81.233:58100	ESTABLISHED	3051/.tcp_server6-
tcp	0 0 192.168.80.230:22	192.168.80.1:32918	ESTABLISHED	2945/sshd: root@not
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

client:



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	512696	795724	192.168.80.233:58100	192.168.81.232:4003	ESTABLISHED	2219/.tcp_client6-
tcp	0	0	192.168.80.231:22	192.168.80.1:32894	ESTABLISHED	1957/sshd: root@not
tcp	0	1884	192.168.80.231:22	192.168.80.1:32873	ESTABLISHED	1955/sshd: root@pts
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	:::125	:::*	LISTEN	1771/master

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。