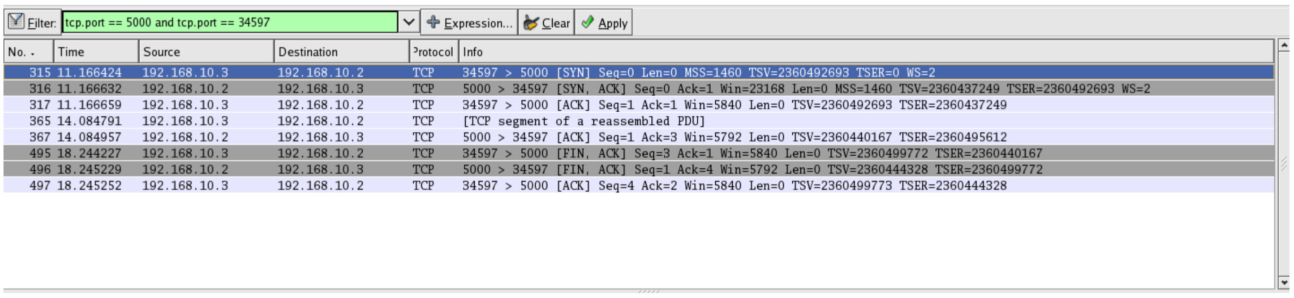
CSE678 Homework1

Xinyu Li
li.1659@buckeyemail.osu.edu

January 12, 2012

Question 2



No.	Time	Source	Destination	Protocol	Info
315	11.166424	192.168.10.3	192.168.10.2	TCP	34597 > 5000 [SYN] Seq=0 Len=0 MSS=1460 TSV=2360492693 TSER=0 WS=2
316	11.166632	192.168.10.2	192.168.10.3	TCP	5000 > 34597 [SYN, ACK] Seq=0 Ack=1 Win=23168 Len=0 MSS=1460 TSV=2360437249 TSER=2360492693 WS=2
317	11.166659	192.168.10.3	192.168.10.2	TCP	34597 > 5000 [ACK] Seq=1 Ack=1 Win=5840 Len=0 TSV=2360492693 TSER=2360437249
365	14.084791	192.168.10.3	192.168.10.2	TCP	[TCP segment of a reassembled PDU]
367	14.084957	192.168.10.2	192.168.10.3	TCP	5000 > 34597 [ACK] Seq=1 Ack=3 Win=5792 Len=0 TSV=2360440167 TSER=2360495612
495	18.244227	192.168.10.3	192.168.10.2	TCP	34597 > 5000 [FIN, ACK] Seq=3 Ack=1 Win=5840 Len=0 TSV=2360499772 TSER=2360440167
496	18.245229	192.168.10.2	192.168.10.3	TCP	5000 > 34597 [FIN, ACK] Seq=1 Ack=4 Win=5792 Len=0 TSV=2360444328 TSER=2360499772
497	18.245252	192.168.10.3	192.168.10.2	TCP	34597 > 5000 [ACK] Seq=4 Ack=2 Win=5840 Len=0 TSV=2360499773 TSER=2360444328

Figure 1: Capture packet printout

- No. 315 is SYN packet from sender, the sequence Number is 0
- No. 316 is SYN, ACK packet from receiver, the sequence Number is 0, acknowledgement number is 1
- No. 317 is ACK packet from the sender, the sequence Number is 1, acknowledgement number is 1
- No. 365 is data packet from the sender.
- No. 367 is ACK packet from the receiver, the sequence Number is 1, acknowledgement number is 3
- No. 495 is FIN, ACK packet from the sender, the sequence Number is 3, acknowledgement number is 1
- No. 496 is FIN, ACK packet from the receiver, the sequence number is 1, acknowledgement number is 4
- No. 497 is ACK packet from the sender, the sequence number is 4, acknowledgement number is 2

Question 3

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.10.64   *               255.255.255.192 U     0      0        0 eth0
192.168.10.0    *               255.255.255.192 U     0      0        0 eth2
192.168.10.128  router2.localdo 255.255.255.192 UG    0      0        0 eth0
169.254.0.0     *               255.255.0.0     U     0      0        0 eth2
```

Destination: The Destination network or host.
Gateway: The gateway address, if it is shows * means none.
Genmask: The netmask for the destination network. If the Genmask is 255.255.255.255 means that the destination is a host.
Flags: The Flags for the route. U means route is up. G means the Gateway.
Metric: The distance to the target, it is count by calculating hops.
Ref: Number of references to this route.
Use: Count of lookups for the route.
Iface: Interface to which packets for this route will be sent.
In this result, the first column are destination network, as we can see from the topolgy diagram from the testbed, 192.168.10.0 means the PrasunNet1 network, 192.168.10.64 means the PrasunNet2 network, 192.168.10.128 means the PrasunNet3 network, 169.254.0.0 means the link-local network, from 169.254.0.0 to 169.254.254.255, this network is reserved for link-local addressing in IPv4 by IETF, they are assigned to interfaces by host internal. The second column means the gateway that the packet need to go through to reach the destination network, from the topology diagram we can get that router can directly reach the PrasunNet1 and PrasunNet2, need pass router2 to reach PrasunNet3. The third column is the network mask for the destination network, the PrasunNet1, PrasunNet2 and PrasunNet3 are all the network which have 62 hosts in the network. The fourth column is the Flags, U means the route is up and the G means reach that network need use gateway. Metric is calculate the distance from the route1 to the destination network. The fifth column is the number of references to this route. The sixth column is used for count of lookups for the route. The seventh column is the interface being used which send the packet to the destination network, as we can see from the network topology diagram, the PrasunNet3 and PrasunNet2 are both use eth2 interface, the PrasunNet1 is use eth0 interface.

Question 4

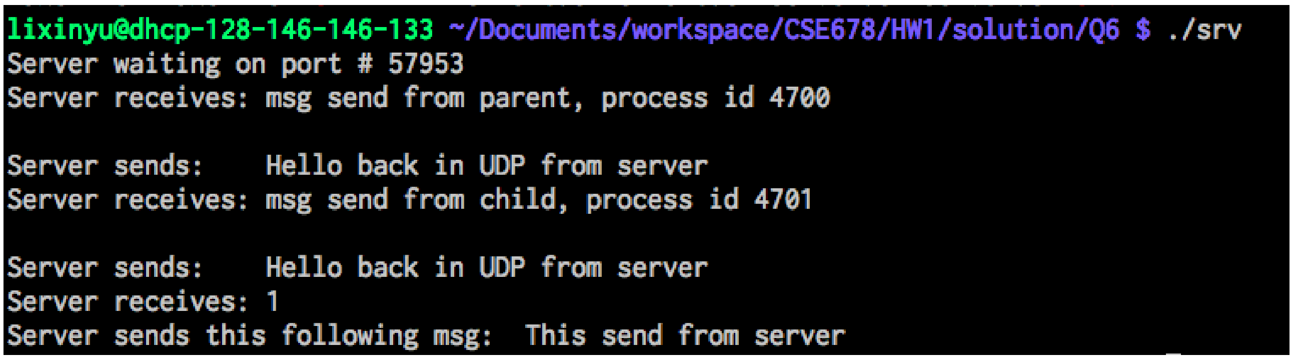
1. 4.
   The main program will create a process, called p1, at this time i = 0, then fork a new process called p2, after fork p2, i in both p1 and p2 are 1 now, then p1 fork a new process called p3, p2 fork a new process called p4, by now, all the process enter the while loop, so there totally 4 process created by this program.

2. 2^n.
   Because every time when i increase 1, every current process will fork a new process, that is means that every time i increase 1, the total number of process will multiply 2, so after n times, the total process will become 2^n.

Question 5

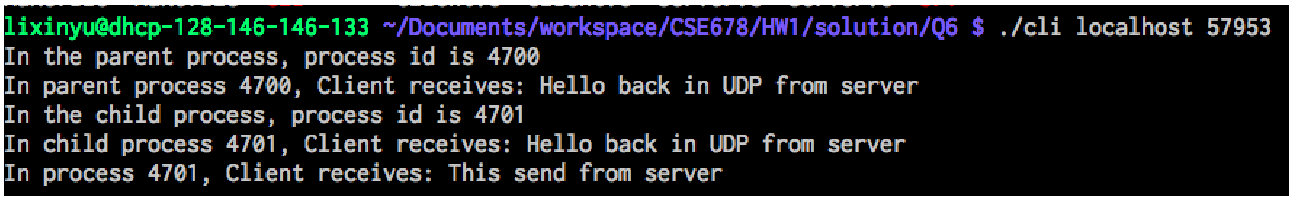
1. Yes, I can send data through the socket from both the process.
2. No, just one process can get the data, it all depends on the system schedule which process can receive the data.

I've write a code to verify my answers. The source code is in the attachment. The following is the screenshot which the porgram run on my laptop.



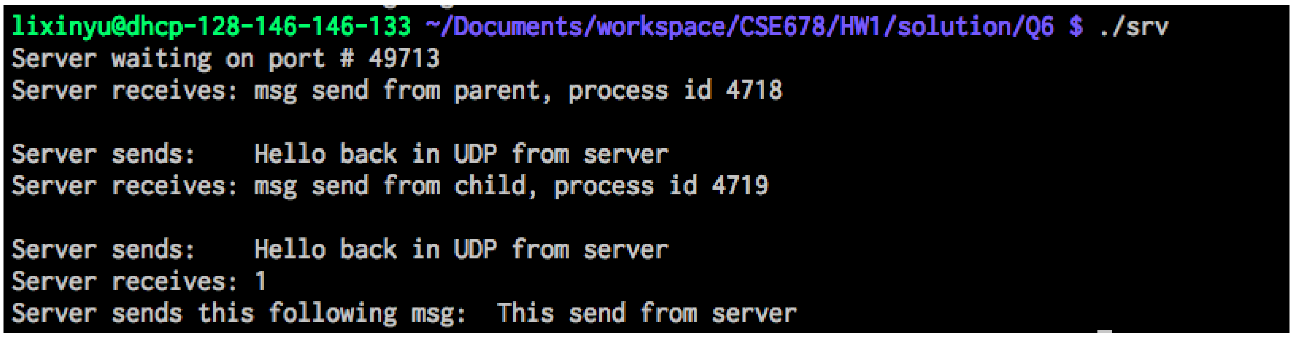
```
lixinyu@dhcp-128-146-146-133 ~/Documents/workspace/CSE678/HW1/solution/Q6 $ ./srv
Server waiting on port # 57953
Server receives: msg send from parent, process id 4700

Server sends:    Hello back in UDP from server
Server receives: msg send from child, process id 4701

Server sends:    Hello back in UDP from server
Server receives: 1
Server sends this following msg:  This send from server
```

Figure 2: First time run, the Server output



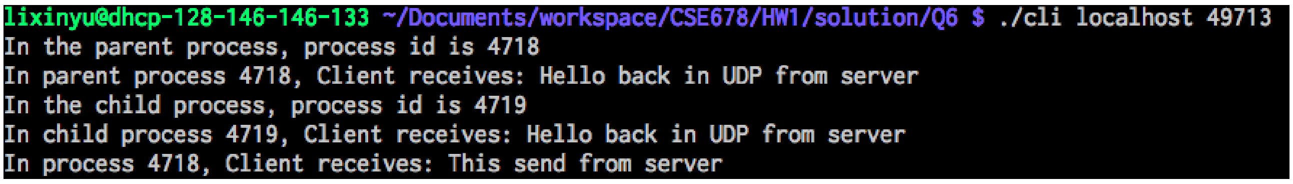
```
lixinyu@dhcp-128-146-146-133 ~/Documents/workspace/CSE678/HW1/solution/Q6 $ ./cli localhost 57953
In the parent process, process id is 4700
In parent process 4700, Client receives: Hello back in UDP from server
In the child process, process id is 4701
In child process 4701, Client receives: Hello back in UDP from server
In process 4701, Client receives: This send from server
```

Figure 3: First time run, the Client output



```
lixinyu@dhcp-128-146-146-133 ~/Documents/workspace/CSE678/HW1/solution/Q6 $ ./srv
Server waiting on port # 49713
Server receives: msg send from parent, process id 4718

Server sends:    Hello back in UDP from server
Server receives: msg send from child, process id 4719

Server sends:    Hello back in UDP from server
Server receives: 1
Server sends this following msg:  This send from server
```

Figure 4: Second time run, the Server output



```
lixinyu@dhcp-128-146-146-133 ~/Documents/workspace/CSE678/HW1/solution/Q6 $ ./cli localhost 49713
In the parent process, process id is 4718
In parent process 4718, Client receives: Hello back in UDP from server
In the child process, process id is 4719
In child process 4719, Client receives: Hello back in UDP from server
In process 4718, Client receives: This send from server
```

Figure 5: Second time run, the Client output

- From Figure 2 to 5, we can easily see that we can both send the data from both process.

- From Figure 3 and Figure 5, we can see that the in Figure 3, the client child process receive the data send from server, but in Figure 5, the client parent process receive the data send from the server. Which process should receive the data send from the sever, it is depends on the system schedule.