The setup used to implement part 1 and part 2 of PA3

1. A machine C with a physical interface eth0 : IP address 10.129.5.195. Machines behind this C machine or C gateway are simulated using virtual interfaces. These run clients
   a. eth0:1 with IP 192.168.7.41
   b. eth0:2 with IP 192.168.7.42
   c. eth0:3 with IP 192.168.7.43

2. A machine S with a physical ethernet interface eth0 : IP address 10.129.5.194. Machines behind this S machine or S gateway are simulated using virtual interfaces. These run servers.
   a. eth0:1 with IP 192.168.6.41
   b. eth0:2 with IP 192.168.6.42
   c. eth0:3 with IP 192.168.6.43

3. Both machines are connected physically using a ethernet cable.

4. pclient.c file given was used beyond C on virtual interfaces to create and run clients. While server1 of PA1 was modified to run from particular IP and used to create server on machines beyond S.
Part 1:

IPTABLE rules are written at gateway machines C and S to connect machines beyond them on either sides. I have written 2 rule on machines C behind which client machines run and have written 1 rule on machine S behind which server is started. I have written the rules for server running on port 5000.

2 Rules on client side:

One rule to NAT source address of all traffic from client machines IP range 192.168.7.0/24 to IP of client machine C with IP 10.129.5.195 so that it can send the packets to S machine 10.129.5.194 as C and S are connected. MASQUERADE or SNAT can be used for this and I have used SNAT to change the source address here.

Second rule is to NAT the destination address of the packet coming from clients running behind C. Based on which server they are trying to connect to, the destination IP:port map is decided. This is done using DNAT. For connecting to server on 192.168.6.41, the DNAT does change of destination address at C as S IP address and a particular port which is predetermined.

1 Rule at server side:
One rule at destination side or S machine behind which server machines are run is to route the packets that have been NATed at C and come to particular port on S machine which need to be forwarded to respective server to which the client intends to connect to. Again DNAT is used to filter incoming packets on S on particular ports to be forwarded to machines behind S. Along with these rules:

a. A route is defined for all traffic from clients behind C to route through C. This need not be done if a default gateway is defined. Otherwise, define the C machine as gateway to the machines connected behind C. Same is to be done at machines running servers (behind S). For all machines behind S, gateway needs to be defined as S.

b. Configure your C, S machines to be in forwarding/router mode. This can be done by setting /proc/sys/net/ipv4/ip_forward value as 1.

Analysis of PART1 implementation in wireshark:

TCP Connection is established from C i.e 10.129.5.195 to S 10.129.5.194 whenever a client running behind C tries to connect to a server running behind S. Connecting 3 clients behind c to 3 servers behind S shows 3 new connections in the capture.
At client

Also I had used 10001, 10002, 10003 as ports at S to connect to 3 servers in and defined the same in iptable rules. Screen shot below shows the same
Part 2

In this part, 2 tun devices are created on either sides i.e S and on C.

Any traffic from machines running clients and destined to IP range 192.168.6.0/24 (where servers run) is routed through tun device at C.

Similarly any traffic from server running machines destined to IP range of clients behind C i.e. 192.168.7.0/24 range is routed through tun device at S.

Then, A client c program tunc.c is run at C and tuns.c is run at S. The C programs take care of writing from tun device to eth0 and reading from eth0 to tun device at either ends.

The tunc and tuns codes are kept running and then we can communicate from clients behind C to server S by running servers and clients.

The screen shots explain that only one TCP connection is established between S and C.

All other connections are tunneled through this. I have captured tcp dump at both S, C as well as tun devices created at the end.

At tun devices tcp connections are established from a client behind C to server behind S for every client-server pair trying to communicate. Whereas there is only one TCP connection from S to C, which is persistent till the tunnel runs.

ScreenShot of TCP dump captured at S:

![Screen Shot of TCP dump](image_url)

Shows only one TCP connection running between 10.129.5.194 and 10.129.5.195 happens when a tunnel is created.
Show TCP connection established for every client server connection.
192.168.7.41 --> 192.168.6.41:5000
192.168.7.42 --> 192.168.6.42:5000
192.168.7.43 --> 192.168.6.43:5000

Comparison of Solutions in part1 and part2:

The solution in part 2 is more elegant and flexible compared to the solution in part1. The solution in part 1 is more rigid as iptables rules are bound to particular IP and port.

Also each connection of client-server eats up free ports of C and S in part1 where as it would use just one tcp connection in case of tunneling using solution in part2.

Since solution in part2 uses just one TCP connection , it may also affect the performance if more client-servers tunnel through gateway C-S.