Virtual Private Network

N8.1. What are the main differences between SSH tunnel and VPN tunnel?

N8.2. To log into Syracuse University's network, Bob needs to use a TLS-based VPN. After he has established a VPN tunnel between his machine and Syracuse University's network (`128.230.0.0/16`), he checks the routing table on his computer. Here is what he sees:

```
-------------------------------------------------------------
  Network
Destination    Netmask          Gateway         Interface
-------------------------------------------------------------
    0.0.0.0    0.0.0.0          192.168.0.1     192.168.0.13
  127.0.0.0    255.0.0.0        On-link         127.0.0.1
  127.0.0.1    255.255.255.255  On-link         127.0.0.1
128.230.0.0    255.255.0.0      128.230.153.48  128.230.153.80
128.230.153.12 255.255.255.255  192.168.0.1     192.168.0.13
128.230.153.80 255.255.255.255  On-link         128.230.153.80
192.168.0.0    255.255.255.0    On-link         192.168.0.13
192.168.0.13   255.255.255.255  On-link         192.168.0.13
192.168.0.255  255.255.255.255  On-link         192.168.0.13
```

From the above routing information, please answer the following questions (you need to explain your answer).

(a) What is the IP address of the TUN interface on Bob's machine?

(b) What is the IP address of Syracuse University's VPN server?

(c) What is the computer's real IP address, i.e., the IP address assigned to the machine's physical network interface card?

(d) Assume that Bob is behind a firewall that blocks him from accessing a web site (assume that the IP address of the web site is 1.2.3.4). Please describe how Bob can use Syracuse University's VPN to bypass the firewall. If changes need to be made to this routing table, please show exactly what changes Bob needs to make to achieve the goal.

N8.3. In Figure 1, Machine X has established a VPN with Machine Y, which is a VPN server connected to the private network `10.0.20.0/24`. With the VPN, a user on Machine X can now access machines on the `10.0.20.0/24` network. The user runs the following command on Machine X: `"telnet 10.0.20.100"`. Figure 1 shows the packet flow triggered by this command. Please answer the following questions:

(a) What is the relationship between packets ❶ and ❷?

(b) What is the relationship between packets ❸ and ❹?

(c) What is the source IP and destination IP of packets ❶, ❷, ❸, and ❹?

(d) What routing entries are needed on Machine X?

(e) What routing entries are needed on Machine Y?

(f) What routing entries are needed on Machine `10.0.20.100`?

(g) If we break the VPN tunnel, what is going to happen to the `telnet` connection? Is it going to be broken? After a few seconds, we reconnect the VPN tunnel between X and Y, what is going to happen?

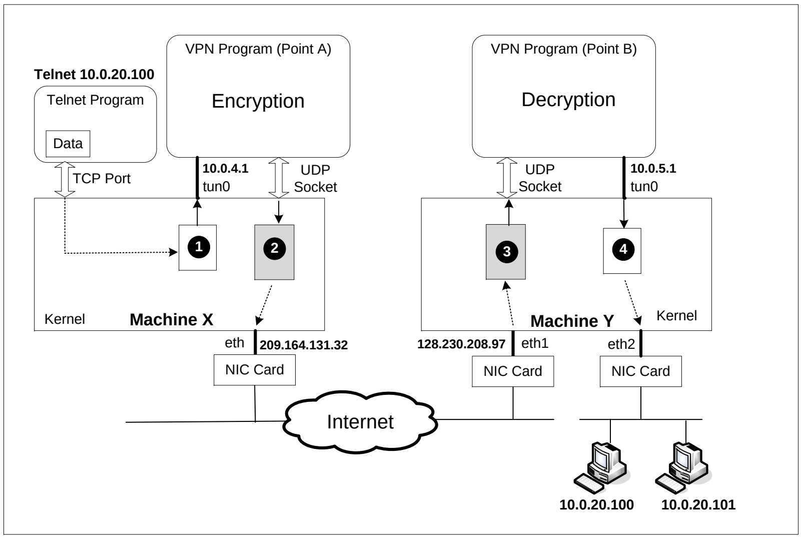

Figure 1: Packet flow over VPN (for Problem N8.3.)

N8.4. A VPN allows Host U on a private network `192.168.60.0/24` to communicate with Host V on another private network `192.168.80.0/24`. See Figure 2 for the VPN setup. Please describe the following:

(a) What routing entries need to be added to Host U, VPN Client, VPN server, and Host V? You don't need to write down the actual command, but you need to describe those routing entries.

(b) When Host V receives a packet from Host U, what is the source IP address of the packet?

(c) When VPN server receives a packet from Host U to Host V, via the VPN tunnel, what is the source and destination IP addresses of the packet?

(d) After the VPN tunnel is set up, when we ping Host V from Host U, please describe in details how the ICMP echo request packets get to Host V from Host U, and how the ICMP echo reply packets get back to Host U.

N8.5. When we use VPN to reach Facebook, which is blocked by our firewall, we route our Facebook-bound packets towards the TUN interface to reach the VPN server via the tunnel. The VPN server will route our packets towards Facebook (via the Internet).

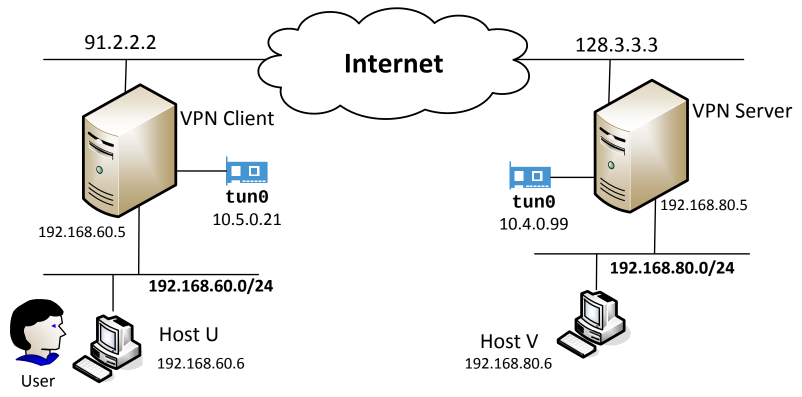

Figure 2: Figure for Problem N8.4.

When Facebook sends reply to us, will the packet be sent directly to us (i.e., without going through the tunnel), or to the VPN server (and then go through the tunnel)? Please explain why.

N8.6. A website in California only allows machines in California to access it. The way how it enforces the rule is to check whether a visitor's IP address is from California or not. You live in Syracuse, New York, and you desperately want to visit this website. Please describe how you can do it.

N8.7. The following code snippet creates a TUN interface. What are the possible names of this interface?

```
tun = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'abc%d', IFF_TUN | IFF_NO_PI)
ifname_bytes  = fcntl.ioctl(tun, TUNSETIFF, ifr)
```

N8.8. When we read from the TUN interface, the sample code in the book uses `os.read(tun, 2048)` to read at most `2048` bytes from the TUN interface. We know that an IP packet might be larger than `2048` bytes, so should we increate this number? If so, what should we increase this number to? The following shows the properties of the TUN interface.

```
$ ifconfig
tun0: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST>
    mtu 1500 inet 192.168.53.99
    netmask 255.255.255.0  destination 192.168.53.99
    ...
```

N8.9. When we read data from the TUN interface, which of the following is most likely the first byte of the data: (A) `0x12`, (B) `0x23`, (C) `0x34`, (D) `0x45`, (E) `0x56`?

N8.10. The `"ip route"` command shows the following results.

```
$ ip route
```

```
default via 10.0.5.1 dev enp0s3 proto dhcp
10.0.5.0/24 dev enp0s3 proto kernel scope link src 10.0.5.5
192.168.53.0/24 dev tun0 proto ... src 192.168.53.99
10.0.6.0/24 dev tun0 scope link
```

What will be the source IP of each packet if we send the packet to the following addresses, respectively.

1. `1.1.1.1`
2. `10.0.5.9`
3. `192.168.53.7`
4. `10.0.6.6`

N8.11. On the VPN server, why do we need to turn on the IP forwarding?

N8.12. After setting up the VPN, we run `tcpdump` to monitor the traffic on all the interfaces (`lo`, `tun0`, and `eth0`). We see the following packets. Please describe which interface each of the packets come from.

```
root@client:# tcpdump -n -i any
IP 10.0.53.99 > 10.0.8.5: ICMP echo request, id 94, seq 282    ①
IP 10.0.7.5.38018 > 10.0.7.11.9090: UDP                        ②
IP 10.0.7.11.9090 > 10.0.7.5.38018: UDP                        ③
IP 10.0.8.5 > 10.0.53.99: ICMP echo reply, id 94, seq 282      ④
```

N8.13. There is a TCP connection between A and B over a VPN. Due to a hardware failure, the VPN client is down for a few seconds, but it comes up quickly, and reconnects with the VPN server. Does this affect the TCP connection between A and B?

N8.14. When we sniff inside a docker container (which is not in the `host` mode), we can only capture the packets from/to itself; packets among other existing containers cannot be captured. What are the reasons?

N8.15. One of the differences between TAP and TUN interfaces is how they get packets. The TUN interface relies on routing, i.e., the kernel will route packets to the TUN interface. How does the TAP interface get packets?

N8.16. In the sample code provided by the book, the `select()` system call is used. Why is it used?

N8.17. A student implemented the following simple tun client program, which prints out the packets received on the TUN interface. Unfortunately, the student mistyped the value for the `IFF_NO_PI` flag: the correct value should be `0x1000`.

```
TUNSETIFF = 0x400454ca
IFF_TUN   = 0x0001
IFF_NO_PI = 0x100

tun = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'tun%d', IFF_TUN | IFF_NO_PI)
ifname_bytes  = fcntl.ioctl(tun, TUNSETIFF, ifr)
```

```
ifname = ifname_bytes.decode('UTF-8')[:16].strip("\x00")
os.system("ip addr add 10.0.53.99/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))

while True:
   packet = os.read(tun, 2048)
   if packet:
      pkt = IP(packet)
      print(pkt.summary())
```

When the student ran "`ping 10.0.53.1`" from the same machine, the program printed out strange results. Please study the purpose of the `IFF_NO_PI` flag using online resources, and then explain why such results were printed out.

```
root@aed05427c19d:/volumes# ./tun.py
Interface Name: tun0
64.1.81.52 > 10.0.53.99 udp frag:84 / Raw
64.1.80.77 > 10.0.53.99 248 frag:84 / Raw
64.1.80.43 > 10.0.53.99 26 frag:84 / Raw
64.1.79.139 > 10.0.53.99 186 frag:84 / Raw
```