Samantha Rack
CSE 30264
Homework 7

P1 (a) An application might want to receive such an ICMP message if it is performing its own verification for the data it is sending across the network. If data that a host is sending is important, then the application will want to verify that the datagram was received by the correct host and on the correct port. UDP does not do this error checking, as there are no guarantees for packet delivery.

(b) On Linux, the application would have to use a raw socket (instead of UDP) to receive the ICMP packets.

(c) Because UDP is "spray and pray", there is no acknowledgment process and therefore no acknowledgment packets. If ICMP would respond to port P, then UDP would have to be able to expect these kinds of packets. When UDP receives a packet, it looks for the port number to which it will forward the packet. The application running on that port would have to be expecting ICMP packets each time it reads data. Additionally, this does not align with the protocol specification in that it does not do error checking.

P2 netstat gives information about the current connections, routing tables, and other information about the network. The closing connections spend 120s in the TIME_WAIT state.

P3 (a) Advertise Window:

DBP = .100s * 1Gbps * (1 byte / 8 bits) = 1.25E7 bytes in flight

$\log_2(1.25E7)$ = **27 bits**

Sequence Number:

(1E9 bits / s) * 30 s = 30E9 bits can be sent in time of maximum segment lifetime

30E9 bits * (byte / 8 bits) = 3.75E9 bytes

$\log_2(3.75E9)$ = **32 bits**

(b) The sequence number is initialized to a random value (to avoid multiple reincarnations of a host, port to be interpreted as the same), and is updated as more bytes are sent. This is less certain because the initial value is random. The advertise window is determined based on how much buffer space is available on the receiving host.

P4 TCP offers error correction as part of the protocol. Therefore, even if a link is unreliable, the packets will eventually be delivered. Running applications on the two ends of the link and counting how many packets were received and sent would always be equal since the protocol guarantees that every packet will be delivered only once.

P5 (a) Sequence Number – 32 bits = 2^32 numbers
2^32 bytes * (8 bits / byte) * (s / 1E9 bits) = **34.35s**

(b) 2^32 / 1000 * 34.35s = **1.47E8 s**

P6

```
uint_32 largerSeqNum (uint_32 x, uint_32 y, uint_32 initSeqNum) {
        if ( x < initSeqNum) x += initSeqNum;
        if ( y < initSeqNum) y += initSeqNum;
        if ( x > y) return x;
        return y;
}
```

P7 When a FIN arrives with a sequence number other than NextByteExpected and the sequence number is within the receive window, the host will buffer this packet until the rest of the packets before it are received. If the sequence number is not within the receive window, the packet is discarded and will be resent by the sender.

P8 This acknowledgement does need to send the acknowledgement of x+1 for its sequence number. Say we have host A which receives a data packet from host B (with sequence number y – amount of data sent). Host A send an acknowledgment with y to host B to indicate that this data was received. Say, then that host B sends a FIN packet to host A. If this packet is not received, but then host A has data to send to host B, host A will send this data with acknowledgement y again. If y+1 is not used for acknowledgment with FIN and SYN, then host B will assume that the FIN packet was received, though host A never got it.