CS144 Practice Midterm—Solutions
Fall 2010

Answer 1:

Assuming an IP header and TCP header take 20 bytes each, when sending 1459 bytes of the file, you will not be sending "full" data packets to the receiver.
This may cause Nagle's algorithm to interact badly with delayed acks:

--According to Nagle's, a sender only sends a partially empty packet when no partially empty packets remain unacknowledged.

--According to the TCP RFC, a receiver can delay sending an ack for up to 200 ms (if you do not receive another data packet in that time).

Answer 2:

RTT. If the routes being flapped between have different round trip times, route-flapping will affect TCP's round-trip time estimates. Incorrect round-trip estimates may make TCP very inefficient.

Answer 3:

Alyssa is assuming that the network is perfectly symmetric. It may be that her initial UDP packet experienced some delay (eg from queueing) that the response packet did not. In addition, it may be that her initial UDP packet and the response even took different routes through the network.

Answer 4:

One example is checksums over pseudoheaders

Answer 5:

One example is IP fragmentation.

Answer 6:

The FCC.

Answer 7:

Flows.

Answer 8:

- dropped packets: TCP detects packet drops and retransmits lost portions
- duplicated packets: TCP offers packet de-duplication
- reordered packets: TCP's sliding window mechanism buffers out of order packets and delivers packets in-order to the application
- senders sending data too fast: TCP endhosts advertise a receiver window for flow control
- unfairness: TCP has congestion control mechanisms to throttle down

Answer 9:

Part a:
The smallest Maximum Transmission Unit (MTU) value between Ida's office and her home is 1500 bytes. Her packet's payload size plus the headers (20+8=28 bytes) exceeds the MTU for some link on the route, so her packet is fragmented.

Part b:
1500 happens to be the MTU for Ethernet (without jumbo frames), but Ida would not necessarily get the same results. She could write a simple echo client/server to determine the minimum MTU on the route. The client would use IP's DF (Don't Fragment) flag and exponentially increase the payload size until packets started to be dropped.

Part c:
No. There's no retransmission in IP or UDP, so if a fragment is lost, all fragments are dropped. 4. 20 bytes. 1500*3 = 4500 bytes total - 4432 bytes of payload = 68 bytes of headers. One UDP header = 8 bytes. 68 - 8 = 60 bytes for three IP headers (3 fragments). Thus, 60/3 = 20 bytes.

Answer 10:

TIME_WAIT keeps the socket open to catch “wandering duplicates” and to retransmit the final ACKs. Without TIME_WAIT, something like the following might happen:

A sends a FIN
B sends an ACK
B sends a FIN
A sends an ACK. A closes the connection. A’s ACK is dropped.

In this case, B will retransmit the FIN and promptly receive a TCP RST (reset) because A no longer knows about the connection. This violates TCP’s reliable nature (a RST indicates data may have been lost, when that is not the case).

Answer 11:

100 Mbps * 100 ms * 2 = 2.50 MB

Answer 12:

Classful addressing was inefficient. CIDR permits finer control of IP address allocation.

Answer 13:

Part a:
Split horizon:
A advertises no routes.
B advertises routes: [cost,dst].
[.5,D],[1,c],[3,a]
C advertises no routes to B. And advertises to D:
[1,B],[4,a].

Part b:
Poison reverse:
A advertises routes to B
[inf,C],[inf,D]
B's advertisements remain unchanged.
C advertises to B:
[inf,A],[inf,D]
C advertises to D:
[1,B],[4,a]

Answer 14:

Part a:
Assuming no losses on the network, larger windows increase a TCP flow's bandwidth: a sender will send more packets per round trip time.

Part b:
In many implementations of TCP, the packet will be buffered. This makes it so that out-of-order packets do not need to be re-transmitted.

Part c:
Assume an adversary knows:
- A TCP receiver's IP address and port number;
- A TCP sender's IP address and port number; and
- Exactly which seqno a receiver is expecting

In this case, an adversary can "spoof" a TCP reset packet "from" the sender and to the receiver. When the packet arrives at the receiver, the receiver will close his/her end of the TCP connection.

Let's relax the requirement that the adversary knows the sequence number the TCP receiver is expecting. The adversary can guess blindly, sending reset packets with a series of sequence numbers, hoping that one of the sequence numbers will be in the range [seqno. receiver expecting, seqno. receiver expecting + window size). If it is in this range, the reset packet will be buffered on the receiver, eventually causing the receiver to reset its connection. The larger the window size, the more susceptible a connection would be to such an attack (see link below).

The above can be quite a problem: consider the BGP protocol, which relies on long-lived TCP sessions. (Whenever a TCP session is closed, a BGP router must rebuild and re-transmit large amounts of information.) See http://bandwidthco.com/whitepapers/netforensics/tcpip/TCP%20Reset%20Attacks.pdf for more information.

Answer 16:

1) Slow start's main purpose is to quickly get the window size up to the connection's capacity, so that the link's bandwidth is not wasted. As soon as a duplicate ACK is received, or the threshold size is reached, those are taken to be indicators that the window size is approaching capacity, and switches to congestion avoidance.

2) Congestion avoidance tries to detect when the link quality is degrading, usually due to congestion (too many users trying to send too many packets), and quickly decreases the window size to decrease usage of it. It halves the window size to play it safe. Ideally, if all the users on the link do this, the link will be able to recover, and all the users will be back in slow start, trying to gauge the link's capacity.