**CISE Department, University of Florida**
**CNT 5106C, Spring 2015**
**Midterm Exam**

| **Name (Last, First):** | **SID:** | **Email:** |
| --- | --- | --- |
| | | |

---

**1. True or False Questions (20 points):**
Mark each statement true (T) or false (F).
Grading Policy for this problem: Each wrong answer faces 2-point deduction. Not answering a question is considered a wrong answer. The total reduction will not exceed 20 points.

(1) Network protocols must be implemented in software since they are too complex to be implemented in hardware. F

(2) In a peer-to-peer programming paradigm, each peer is neither a client nor a server. F

(3) Communications at the TCP level is host-to-host only; Communications at the IP level is router-to-router only. F

(4) Multiple application processes can share the same TCP connection for sending data. F

(5) Two UDP packets with the same destination IP address and destination port number will be delivered to the same socket. T

(6) A UDP receiver performs the checksum operation to verify whether a packet is corrupted. T

(7) TCP estimates the round trip time so that it can provide delay guarantee. F

(8) Internet Explorer from Microsoft is an example of an application-layer protocol. F

(9) Persistent HTTP reduces unnecessary round trip times. T

(10) The benefits of using web proxy servers include reduction of response time and bandwidth saving. T

(11) The DNS hierarchy contains root servers, top level domain servers, and local servers. F

(12) A pipelined reliable data transfer protocol can improve throughput. T

**2. Short Questions (32 points; 4 points each)**
Please answer each question no more than several sentences. Be brief and only discuss the essence. Each question is worth 3 points.

(1) Describe at least four types of services the transport layer provides.

Multiplexing/demultiplexing, reliability control, bandwidth guarantee, congestion control, flow control.

(2) How does the web browser get redirected to a new web page after the page is moved?

HTTP reply contains a status code “Moved Permanently” with the new location.

(3) What is delayed ACK in TCP?

When an in-order segment arrives at the receiver and there is no ACK pending, the receiver will wait for 500 ms before sending the ACK. However, if the next in-order segment arrives at the receiver within that 500 ms, the receiver immediately sends an ACK.

(4) What is TCP flow control and how is it done?

Flow control ensures that the sender does not overwhelm the receiver (in TCP’s case, does not overflow receiver’s receive buffer). In TCP, the receiver advocates the free buffer space at the receiver in the “receive window” field of the TCP segments in the reverse direction, for instance, in the ACK packets.

(5) Briefly describe the key difference between the Go-Back-N and Selective Repeat reliable transfer protocols.

Go-Back-N: The timer is set on the first outstanding packet (yet-to-be-acknowledged packet). If the timer expires, the time packet and all subsequent outstanding packets are retransmitted.

Selective Repeat: There is a separate timer for each outstanding packet. The sender only retransmits the packet that whose timer expires.

(6) In TCP's round trip time estimation algorithm,

$$\text{EstimatedRTT} = (1- \alpha)*\text{EstimatedRTT} + \alpha*\text{SampleRTT}$$

Describe the tradeoffs for using a large or small $\alpha$ value.

If the value is large, the estimate is more responsive to the new RTT sample. This is beneficial if the underlying RTT has changed suddenly but permanently. However, it is less capable in smoothing out short-term fluctuations of the RTT. A smaller value has the opposite effect.

(7) What's wrong if TCP's timeout interval is set to be too short or too long, respectively?

Too short: There will be premature timeouts. In this case, packets that are delayed may be declared as being lost. The sender will retransmit them, causing unnecessary retransmissions. This is a waste of bandwidth and may cause network congestion.

Too long: Lost packets are not detected quickly enough. This may cause the sender to wait, stalling the retransmissions and possibly the subsequent transmissions. This usually leads to poor throughput.

(8) Why does packet disordering in the network cause problems to TCP Fast Retransmit?

Fast retransmit: The sender retransmits after receiving three duplicated ACKs. This is deemed a faster response than timeout-based loss detection and retransmission. Packet disordering may trigger duplicated ACKs. The sender may infer that packets are lost and retransmit the packets when in fact the packets are merely disordered.

**3. Problem (15 points)**.

Recall that rdt 3.0 is the alternating-bit protocol. The assumption is that no packet can overtake an earlier packet in either direction. But, packets in either direction can be corrupted or dropped. The state machines of the sender and receiver are given in the end (the receiver's state machine is the same as that of rdt 2.2).

(a) **(4 points)** Please give a sample path of events for the following: When the sender is in the "Wait for ACK0" state, an ACK1 comes.

Sender -> Receiver: pkt 1
Receiver -> Sender: ACK1
Sender -> Receiver: pkt0 (But, pkt0 is corrupted.)
Receiver -> Sender: ACK1

(b) **(4 points)** Suppose ACK1 is received and is NOT corrupted while the sender is in the "Wait for ACK0" state. Should this event be interpreted as pkt0 is corrupted and therefore should be immediately retransmitted? Why and why not?

Yes. The only possibility is that pkt0 is corrupted based on receiver's state machine. Retransmit pkt0 should work and should improvement the throughput, since retransmission occurs faster.

(c) **(7 points)** Suppose while the sender is in the "Wait for ACK0" (or "Wait for ACK1") state, it receives a corrupted feedback packet. Would immediate retransmission of pkt0 (or pkt1, respectively) work? Also, consider what would happen if there are no packet losses but premature timeouts always occur. Consider how many times the *n*th packet is sent. Take the limit for *n* approaching infinity.

The protocol would still work, since a retransmission would be what would happen if the packet received with errors has actually been lost (and from the receiver standpoint, it never knows which of these events, if either, will occur).

To get at the more subtle issue behind this question, one has to allow for premature timeouts to occur. In this case, if each extra copy of the packet is ACKed and each received extra ACK causes another extra copy of the current packet to be sent, the number of times packet $n$ is sent will increase without bound as $n$ approaches infinity.

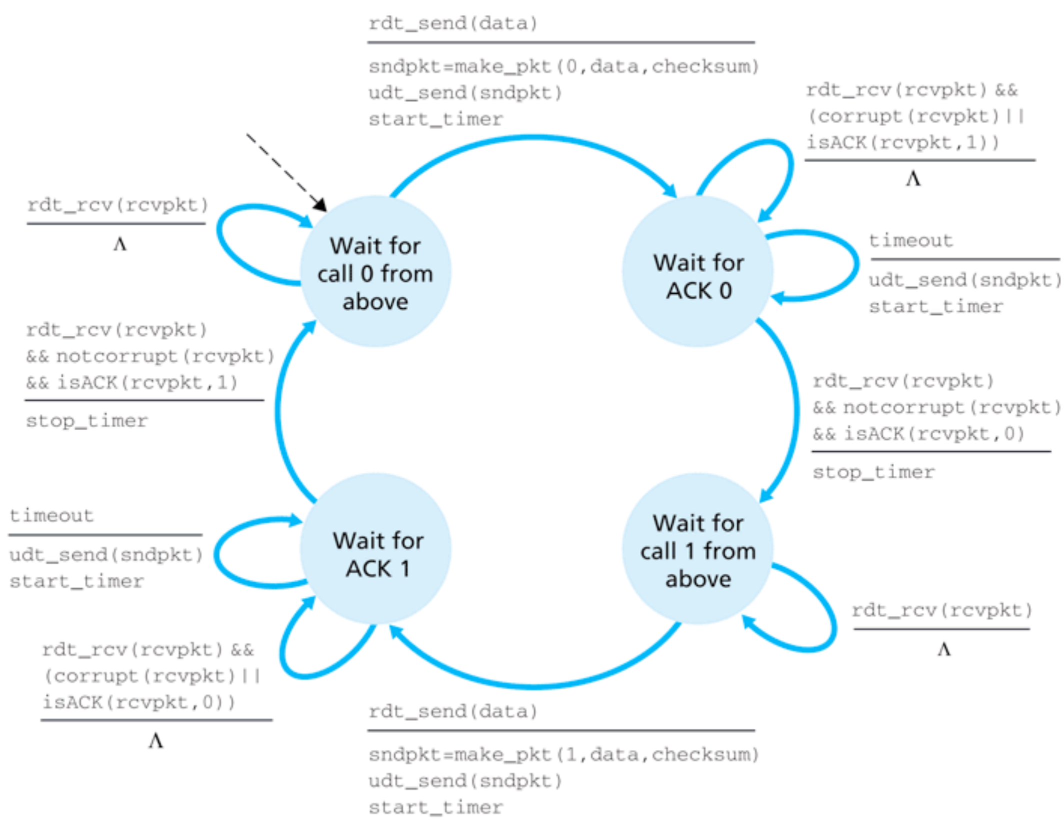



**Figure 3.15** ♦ rdt3.0 sender

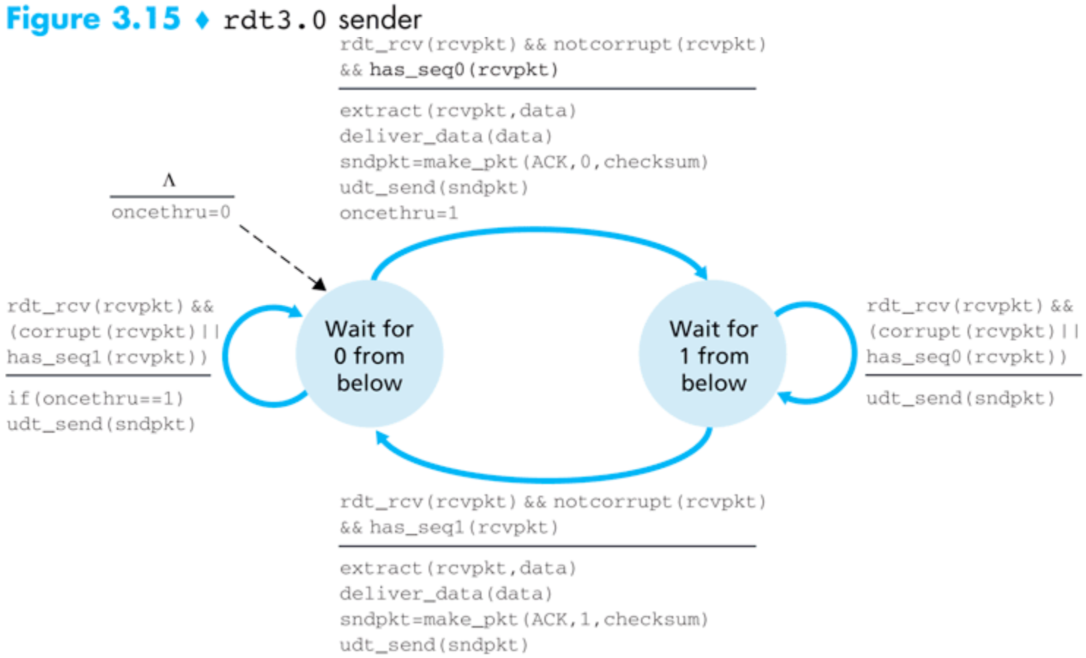


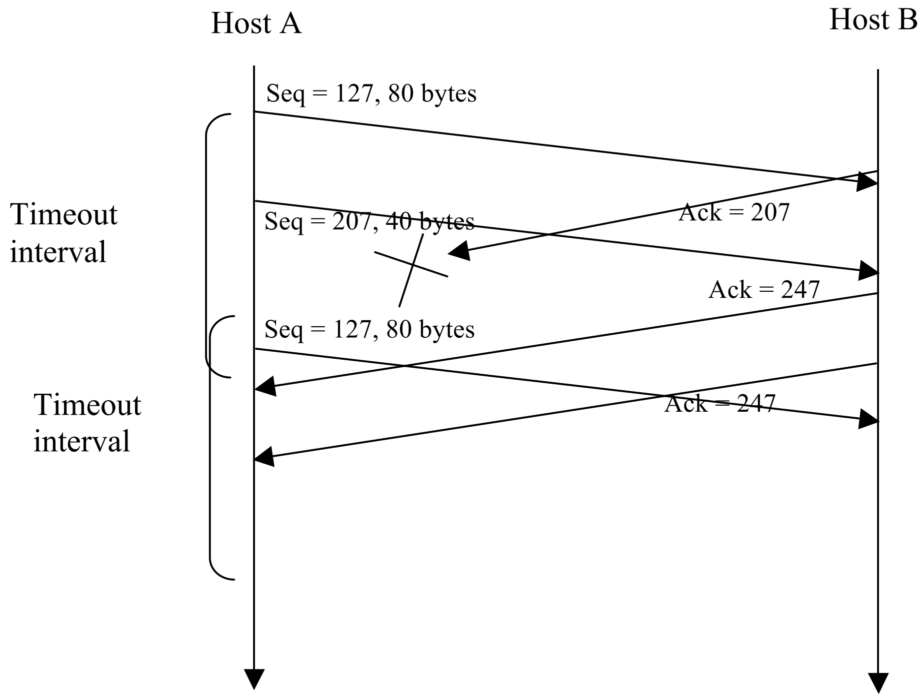

**Figure 3.14** ♦ rdt2.2 receiver

**4. Problem (13 points)** Host A and B are communicating over a TCP connection, and Host B has already received from A all bytes up through byte 126. Suppose Host A then sends two segments to Host B back-to-back. The first and second segments contain 80 and 40 bytes of data, respectively. In the first segment, the sequence number is 127, the source port number is 302, and the destination port number is 80. Assume Host B sends an acknowledgement whenever it receives a segment from Host A.

(a) **(3 points)** In the second segment sent from Host A to B, what are the sequence number, source port number, and destination port number?
(b) **(3 points)** If the first segment arrives before the second segment, in the acknowledgement of the first arriving segment, what are the acknowledgement number, the source port number and the destination port number?
(c) **(3 points)** If the second segment arrives before the first segment, in the acknowledgement of the first arriving segment, what is the acknowledgement number?
(d) **(4 points)** Suppose the two segments sent by A arrive in order at B. The first acknowledgement is lost and the second acknowledgement arrives after the first timeout interval. Draw a timing diagram, showing these segments and all other segments and acknowledgements sent. (Assume there is no additional packet loss.) For each segment in your figure, provide the sequence number and the number of bytes of data; for each acknowledgement that you add, provide the acknowledgement number.

a) In the second segment from Host A to B, the sequence number is 207, source port number is 302 and destination port number is 80.
b) If the first segment arrives before the second, in the acknowledgement of the first arriving segment, the acknowledgement number is 207, the source port number is 80 and the destination port number is 302.
c) If the second segment arrives before the first segment, in the acknowledgement of the first arriving segment, the acknowledgement number is 127, indicating that it is still waiting for bytes 127 and onwards.

d)




**5. Problem (20 points)** Consider the DNS lookup process (see the figures below for iterative and recursive queries). We assume the local DNS server is close to the clients and there is no propagation time between a client and the local DNS server in either direction. For the rest of the communication legs, suppose the round-trip propagation time is 30ms for **each leg**.
Suppose the processing time of a DNS message by any server is 50ms. This includes the time to parse the message and the time to transmit a subsequent message as required by the protocol. We assume the local DNS server may get busy, and hence, DNS messages may have to wait in a FIFO (first-in-first-out) queue before they are processed. We assume that other servers are never busy, and hence, there is no queueing delay at those servers.

**Note:** The later parts of the question build on top of the earlier parts. If you cannot answer one part of the question, you should continue to try the later parts. In that case, please write down your assumptions if needed.

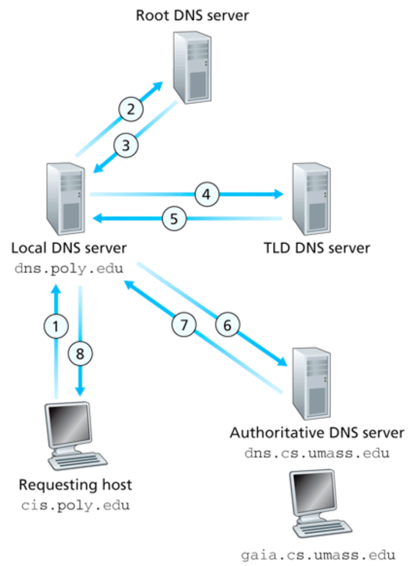



**Figure 2.21** ♦ Interaction of the various DNS servers

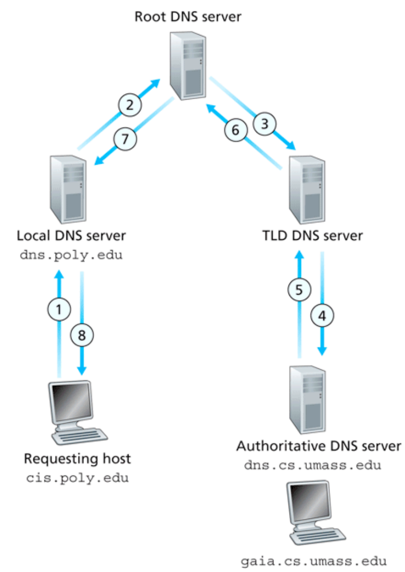



**Figure 2.22** ♦ Recursive queries in DNS

(a) **(2 points)** To resolve a DNS query from a client, how many messages need to be processed (i.e., received) by the local DNS server, including the original query message from the client? Consider both the iterative query and the recursive query. (Note that each received message must enter the FIFO queue at the local DNS server, and hence, contributes to the arrival rate of messages to the queue.)

(b) **(2 points)** How many messages per second from the FIFO queue can the local DNS server process?

(c) **(2 points)** Suppose the clients make 4 queries per second. What are the traffic intensities (load) to the FIFO queue in the iterative query and the recursive query? Recall: The traffic intensity/load to a queue is the ratio of the average arrival rate and the average service/processing rate.

(d) **(2 points)** On average, how many queries can the local DNS server resolve in each second in the iterative and recursive cases?

(e) **(2 points)** What is the average (expected) queueing delay experienced by the messages (including the time spent waiting in the queue and the processing time)? This queueing delay is also known as the *sojourn time* at the queue.

Hint: We assume that the expected sojourn time at the queue, denoted by $W$, is give by: $E[W] = E[S]/(1-\rho)$, where $E[S]$ is the expected processing time, which is 50ms in this case, and $\rho$ is the traffic intensity/load.

(f) **(6 points)** What are the average query response times in the two cases: the iterative query and recursive query? Once a client makes a query, the response time is the time till the client receives the corresponding DNS reply.

(g) **(4 points)** Now consider caching. Suppose the local DNS server caches DNS records and suppose the cache hit ratio is 60%. What are the average response times for the iterative and recursive cases? Suppose the local clients still make 4 queries per second.

Answers:
(a) iterative query: 4 messages (query, three replies)
recursive query: 2 messages (query and reply)

more details:

iterative query:
For each query, there are four messages to be processed by the local DNS server. The first one is the query made by a client. The second one is the response message from the root server. The third one is the response message from the TLD server. The fourth one is the response message from the authoritative server.

recursive query:
For each query, there are two messages to be processed by the local DNS server. The first one is the query made by a client. The second one is the reply from the root server after the DNS query is resolved by the root server.

(b) 20 per second

(c) iterative: $\rho = 4 * 4 / 20 = 0.8$; recursive: $\rho = 4 * 2 / 20 = 0.4$

(d) iterative: 5; recursive: 10

(e)
iterative query: For each query, there are four messages entering the queue at the local DNS server.
$\rho = 0.8$; based on that, the expected sojourn time of a message: 50ms / (1 - 0.8) = 250ms.

recursive query: For each query, there are two messages entering the queue at the local DNS server.
$\rho = 0.4$
The expected sojourn time of a message in the queue: 50ms / (1 - 0.4) = 83.3ms

(f)
iterative query:

- the expected time till the second message enters the queue: 250ms + 30ms + 50ms (expected sojourn time of the first message + propagation time to and from the root server + processing time at the root server)
- additional time till the third message enters the queue: 250ms + 30ms + 50ms (expected sojourn time of the second message + propagation time to and from the TLD server + processing time at the TLD server)
- additional time till the fourth message enters the queue: 250ms + 30ms + 50ms

- the expected sojourn time of the fourth message: 250ms
- total delay (response time): 1240ms

recursive query:
total delay (response time) = 83.3ms + (30ms + 50ms) * 3 + 83.3ms = 406.7ms

- the first 30ms + 50ms in the above: the round-trip propagation time from the local DNS server to the root server and back + the processing time at the root server.
- the second 30ms + 50ms in the above: the round-trip propagation from the root server to the TLD server and back + the processing time at the TLD server.
- the third 30ms + 50ms in the above: the round-trip propagation time from the TLD server to the authoritative server and back + the processing time at the authoritative server.

(g)
iterative query:
hit case: For each query, one message enters the queue at the local DNS server.
miss case: For each query, four messages enter the queue.
average arrival rate of messages to the queue: 0.6 * 4 * 1 + 0.4 * 4 * 4 = 8.8 messages/second
traffic load: $\rho$ = 8.8 / 20 = 0.44
expected queueing delay for each message (sojourn time): 50ms / (1 - 0.44) = 89.3ms

expected response time in case of hit: 89.3ms
expected response time in case of miss: (89.3ms + 30ms + 50ms) * 3 + 89.3ms = 597.1ms
average response time: 0.6 * 89.3ms + 0.4 * 597.1ms = 292.4ms

recursive query:
hit case: For each query, one message enters the queue.
miss case: For each query, two messages enter the queue.
average arrival rate of messages to the queue: 0.6 * 4 * 1 + 0.4 * 4 * 2 = 5.6 messages/second
traffic load: $\rho$ = 5.6 / 10 = 0.28
expected queueing delay for each message (sojourn time): 50ms / (1 - 0.28) = 69.4ms

expected response time in case of hit: 69.4ms
expected response time in case of miss: 69.4ms + (30ms + 50ms) * 3 + 69.4ms = 378.9ms
average response time: 0.6 * 69.4ms + 0.4 * 378.9ms = 193.2ms

| **Problem** | **Score** | **Max. Score** |
|---|---|---|
| **1** | | 20 |
| **2** | | 32 |
| **3** | | 15 |
| **4** | | 13 |
| **5** | | 20 |
| **Total** | | 100 |