Name _____ Student No. _____

No aids allowed. Answer all questions on test paper. Use backs of sheets if necessary.

Total Marks: **40**

[10] 1. The article *Improving performance on the Internet*, by Tom Leighton, includes a discussion of the so called "Fat File Paradox." Explain briefly what that is.

Solution: Given that data packets can traverse networks at close to the speed of light, why does it take so long for a big file to cross the country, even if the network is *not* congested? The reason for this "paradox" is the following: because of the way the underlying network protocols work, *latency* and *throughput* are directly coupled. TCP, for example, allows only small amounts of data to be sent at a time (the TCP "sliding window") before having to pause and wait for acknowledgment from the receiving end. This means that throughput is effectively throttled by network round-trip time (latency), which can become the bottleneck for file download speeds.

[10] 2. (a) Give two reasons why a centralized DNS (Domain Name Service) would not work.

(b) DNS is *hierarchical* and *distributed*; explain briefly what that means and what are the main levels of the hierarchy..

(c) What is the difference between *recursive* and *iterative* DNS queries?

(d) What would be the output of the following command:

```
nslookup -type=CNAME www.cas.mcmaster.ca
```

(If you do not know the exact output, explain at least what would be the expected output.)

Solution: (a) A single point of failure, traffic volume, it cannot be close geographically to 'everywhere', does not scale (maintenanc, etc.). (b) It is a hierarchy of servers, with *Root DNS Servers* at the top, *Top Level Domains* in the middle (for .com, .org, .edu, countries, etc.), and *Authoritative DNS servers* in the bottom. Queries start at the bottom of the hierarchy, and travel up as needed. The system is a vast world-wide distributed database. (c) Roughly speaking, recursive queries are handled by the hierarchy, with forwarding of the query until an answer is found, which is then forwarded to the host that initiated the query. In iterative quering, each DNS server returns the address of what it believes is the next DNS server that the initial host has to query to get the answer; this is repeated until the answer is found. (d) It returns the Canonical Name for the given address:

```
www.cas.mcmaster.ca     canonical name = ritchie.cas.mcmaster.ca.
```

[10] 3. Indicate the window, the `base` and `nextseqnum` variables; you want to indicate the value of these three parameters *immediatelly following* the sender event (as you can see demonstrated in the first row).

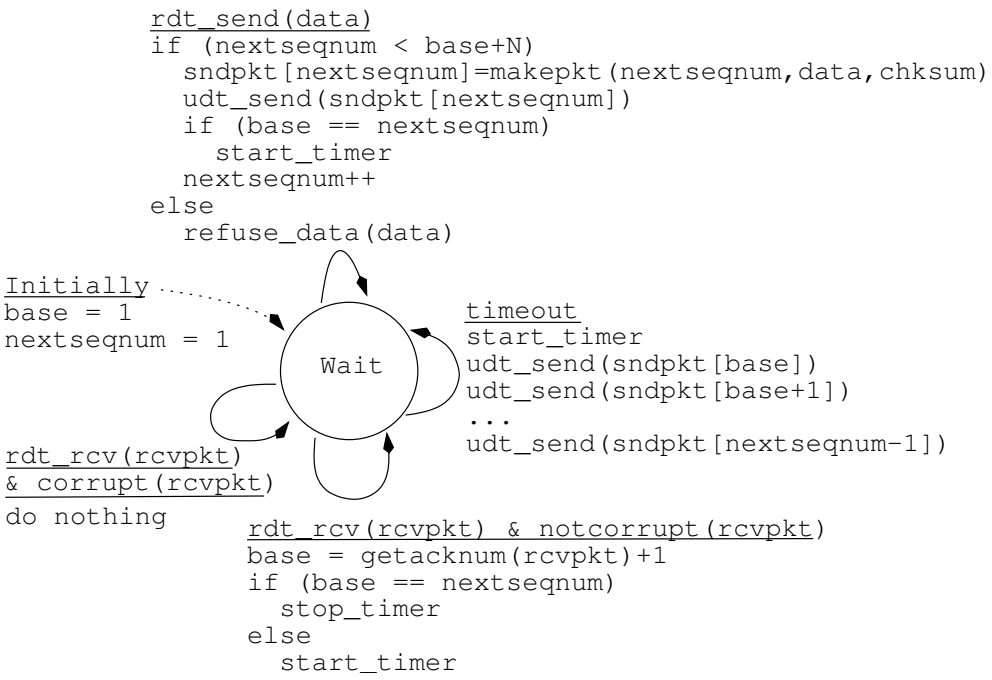
The next page contains the programs for the Go-Back-N sender and receiver.

Sender Event	Sender Window	`base`	`nextseqnum`
send pkt 0	**[0 1 2 3]** 4 5 6 7 8	0	1
send pkt 1	0 1 2 3 4 5 6 7 8		
send pkt 2	0 1 2 3 4 5 6 7 8		
send pkt 3	0 1 2 3 4 5 6 7 8		
rcv ack 0, send pkt 4	0 1 2 3 4 5 6 7 8		
rcv ack 1, send pkt 5	0 1 2 3 4 5 6 7 8		
rcv ack 1	0 1 2 3 4 5 6 7 8		
rcv ack 1	0 1 2 3 4 5 6 7 8		
pkt 2 timeout	0 1 2 3 4 5 6 7 8		

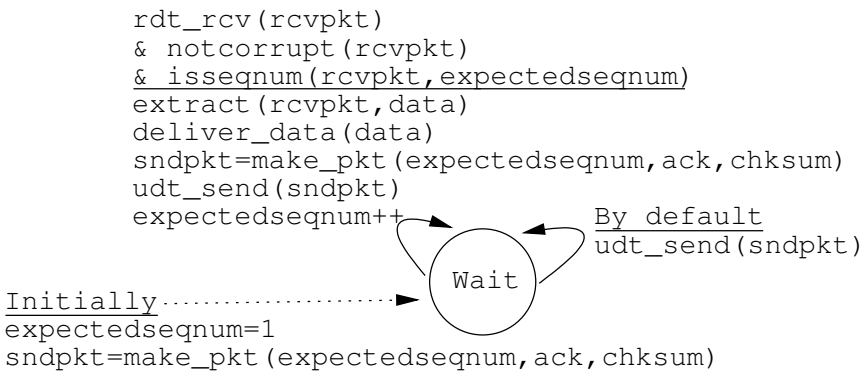
After the last event, "pkt 2 timeout" which packets are re-sent by sender?

Solution: The contents of the 1st row were given; in rows 2,3,4 the window consists of $[0, 3]$ and the base is always 0, while the next sequence number climbes, i.e., it has the number of the corresponding row. In row 5, the window is $[1, 4]$, the base is 1, and the next sequence number is 5. In the last 4 rows, the window is $[2, 5]$ and the base is 2, while the next sequence number is 6. After the last even all the packets in the window ($[2, 5]$) are retransmitted.

Go-Back-N Sender



Go-Back-N Receiver



[10] 4. TCP looks very much like Go-Back-N, but there is one major difference: it has the capability of buffering correctly received but out of order packets.

Another feature of TCP is that the sender performs a **fast retransmit** (described in RFC 5681) in the case that three duplicate ACKs are received (as in question 3) *before* that segment's timer expires.

Why do you think that designers of TCP chose to do a fast retransmit after three ACKs rather than after two?

Solution: Suppose packets 0, 1, and 2 are sent, and that packet 0 is received and ACKed. If packets 1 and 2 are reordered along the end-to-end-path (i.e., are received in the order 2, 1) then the receipt of packet 2 will generate a duplicate ack for 0 and would trigger a retransmission under a policy of waiting only for second duplicate ACK for retransmission. By waiting for a triple duplicate ACK, it must be the case that two packets after packet 0 are correctly received, while 1 was not received. The designers of the triple duplicate ACK scheme probably felt that waiting for two packets (rather than 1) was the right tradeoff between triggering a quick retransmission when needed, but not retransmitting prematurely in the face of packet reordering.