Name _____ Student No. _____

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

Total Marks: **30**

[10] 1. Consider an application that transmits data at a steady rate — for example, the sender generates an N-bit unit of data every k time units, where k is *small* and *fixed*. Also, when this application starts, it will continue running for a relatively long period of time. Answer the following two questions providing brief justifications:

(a) What is more appropriate for this application: a *packet-switched network* or a *circuit-switched network*?

(b) Suppose that a packet-switched network is used, and the only traffic is generated by this kind of application. Furthermore, assume that the sum of the application data rates is less than the capacities of each and every link. Is some form of congestion control needed?

No marks for an answer without justification.

Solution: (a) A circuit-switched network would be well suited to the application, because the application involves long sessions with predictable smooth bandwidth requirements. Since the transmission rate is known and not "bursty," bandwidth can be reserved for each application session without significant waste. In addition, the overhead costs of setting up and tearing down connections are amortized over the lengthy duration of a typical application session.

(b) In the worst case, all the applications simultaneously transmit over one or more network links. However, since each link has sufficient bandwidth to handle the sum of all of the applications' data rates, no congestion (very little queuing) will occur. Given such generous link capacities, the network does not need congestion control mechanisms.

[10] 2. Consider distributing an F-bit file to N peers using a client-server architecture. Assume a model where the server can simultaneously transmit to multiple peers, transmitting to each peer at different rates, as long as the combined rate does not exceed u_S.

(a) Suppose that $u_S/N \leq d_{\min}$. Specify a distribution scheme that has a distribution time of NF/u_S.

(b) Suppose that $u_S/N \geq d_{\min}$. Specify a distribution scheme that has a distribution time of $F/d_{\min}$.

(c) Conclude that the minimum distribution time is in general given by

$$D_{\text{C}-\text{S}} = \max\left\{NF/u_S, F/d_{\min}\right\}.$$

(*Hint*: You may use what we showed in class: $D_{\text{C}-\text{S}} \geq \max\{NF/u_S, F/d_{\min}\}$.)

Solution: (a) Consider a distribution scheme in which the server sends the file to each client, simultaneously, at a rate of u_S/N. Note that this rate is less than each of the client's download rate, since by assumption $u_S/N \leq d_{\min}$. Thus each client can also receive at rate u_S/N. Since each client receives at rate u_S/N, the time for each client to receive the entire file is $F/(u_S/N) = NF/u_s$. Since all the clients receive the file in NF/u_S, the overall distribution time is also NF/u_S.

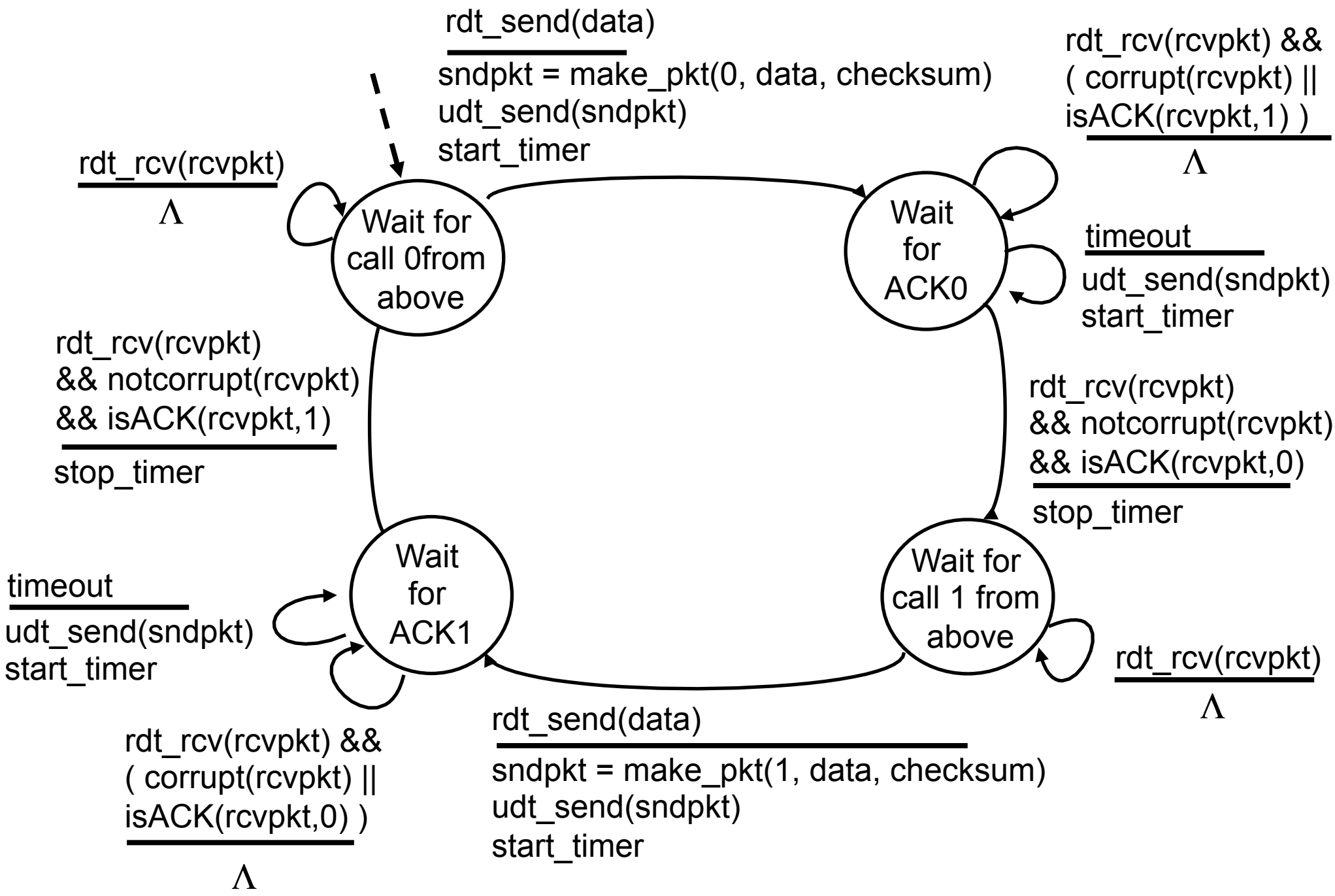
(b) Consider a distribution scheme in which the server sends the file to each client, in parallel, at a rate of $d_{\min}$. Note that the aggregate rate, $N \cdot d_{\min}$, is less than the server's link rate us, since by assumption $u_S/N \geq d_{\min}$. Since each client receives at rate $d_{\min}$, the time for each client to receive the entire file is $F/d_{\min}$. Since all the clients receive the file in this time, the overall distribution time is also $F/d_{\min}$.

(c) Recall that in class we showed that

$$D_{\text{C}-\text{S}} \geq \max\{NF/u_S, F/d_{\min}\}. \tag{1}$$

We now consider the two schemes given above. If, as in (a), $u_S/N \leq d_{\min}$, then from inequality (1) we know that $D_{\text{C}-\text{S}} \leq NF/u_S$. On the other hand, if, as in (b), $u_S/N \geq d_{\min}$, then from (1) we have $D_{\text{C}-\text{S}} \leq F/d_{\min}$.

This gives us that $D_{\text{C}-\text{S}} \leq \max\{NF/u_S, F/d_{\min}\}$, since it is always the case that either $u_S/N \leq d_{\min}$ or $u_S/N \geq d_{\min}$. Hence, strict equality follows.


rdt_send(data)
sndpkt = make_pkt(0, data, checksum)
udt_send(sndpkt)
start_timer
rdt_rcv(rcvpkt) &&
(corrupt(rcvpkt) ||
isACK(rcvpkt,1))
Λ
rdt_rcv(rcvpkt)
Λ
Wait for call 0from above
Wait for ACK0
timeout
udt_send(sndpkt)
start_timer
rdt_rcv(rcvpkt)
&& notcorrupt(rcvpkt)
&& isACK(rcvpkt,1)
stop_timer
rdt_rcv(rcvpkt)
&& notcorrupt(rcvpkt)
&& isACK(rcvpkt,0)
stop_timer
Wait for ACK1
Wait for call 1 from above
timeout
udt_send(sndpkt)
start_timer
rdt_rcv(rcvpkt)
Λ
rdt_rcv(rcvpkt) &&
(corrupt(rcvpkt) ||
isACK(rcvpkt,0))
Λ
rdt_send(data)
sndpkt = make_pkt(1, data, checksum)
udt_send(sndpkt)
start_timer

[10] 3. Consider RDT3.0 given on the previous page (the sender side). Base on this, give the time-line for the following situation:

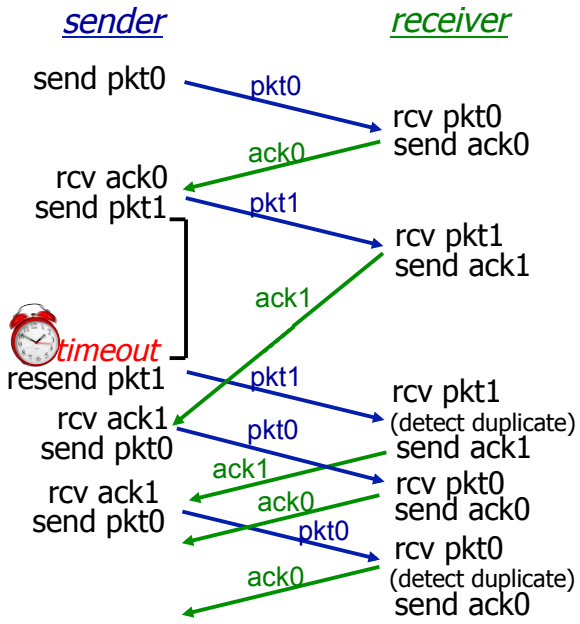
- pkt 0 is sent
- pkt 0 is acknowledged
- pkt 1 is sent
- pkt 1 is timed-out
- but pkt 1 is received and acknowledge (albeit late)

In light of your premature timeout time-line, explain the purpose of having

"`isACK(rcvpkt, 1)` do nothing"

in the "Wait for ACK 0" state. In particular, explain why it isn't necessarily more efficient to retransmit rather than do nothing.

Solution: Here is the time-line for the premature timeout:



This time-line illustrates the purpose of the "do nothing" scheme in the case of a `isACK(rcvpkt,1)` in the "Wait for ACK 0" state. The reason is simply this: if the acknowledgment to pkt 1 arrives late (i.e., after a time-out has already occurred), that means that pkt 1 was already resent, and there will be second acknowledgment. This second acknowledgment will arrive when the sender is already in the next phase (i.e., expecting an ACK 0) and hence requires no action. In fact, resending in this case would mean sending unnecessary packets.