Show All Answers Hide All Answers

6.02 Practice Problems: Reliable Data Transport Protocols

Problem 1. Consider the following chain topology:

A ---- B ----- C ---- D ---- E

A is sending packets to E using a reliable transport protocol. Each link above can transmit one packet per second. There are no queues or other sources of delays at the nodes (except the transmission delay of course).

A. What is the RTT between A and E?

Hide Answer

8 seconds

B. What is the throughput of a stop-and-wait protocol at A in the absence of any losses at the nodes?

Hide Answer

1/8 pkts/s

C. If A decides to run a sliding window protocol, what is the optimum window size it must use? What is the throughput achieved when using this optimum window size?

Hide Answer

optimum window = 8 pkts, optimum throughput = 1 pkt/s

D. Suppose A is running a sliding window protocol with a window size of four. In the absence of any losses, what is the throughput at E? What is the utilization of link B-C?

Hide Answer

throughput=0.5 pkts/s, utilization=0.5

E. Consider a sliding window protocol running at the optimum window size found in part 3 above. Suppose nodes in the network get infected by a virus that causes them to drop packets when odd sequence numbers. The sliding window protocol starts numbering packets from sequence number 1. Assume that the sender uses a timeout of 40 seconds. The receiver buffers out-of-order packets until it can deliver them in order to the application. What is the number of packets in this buffer 35 seconds after the sender starts sending the first packet?

Hide Answer

Answer: 7.

With a window size of 8, the sender sends out packets 1--8 in the first 8 seconds. But it gets back only 4 ACKs because packets 1,3,5,7 are dropped. Therefore, the sender transmits 4 more packets (9--12) in the next 8 seconds, 2 packets (13--14) in the next 8 seconds, and 1 (sequence number 15) packet in the next 8 seconds. Note that 32 seconds have elapsed so far. Now the sender gets no more ACKs because

packet 15 is dropped, and it stalls till the first packet times out at time step 40. Therefore, at time 35, the sender would have transmitted 15 packets, 7 of which would have reached the receiver. But because all of these packets are out of order, the receiver's buffer would have 7 packets.

Problem 2. Ben Bitdiddle implements a reliable data transport protocol intended to provide "exactly once" semantics. Each packet has an 8-bit incrementing sequence number, starting at 0. As the connection progresses, the sender "wraps around" the sequence number once it reaches 255, going back to 0 and incrementing it for successive packets. Each packet size is S = 1000 bytes long (including all packet headers).

Suppose the link capacity between sender and receiver is C = 1 Mbyte per second and the round-trip time is R = 100 milliseconds.

A. What is the highest throughput achievable if Ben's implementation is stop-and-wait?

Hide Answer

The highest throughput for stop-and-wait is (1000 bytes)/(100ms) = 10 Kbytes/s.

B. To improve performance, Ben implements a sliding window protocol. Assuming no packet losses, what should Ben set the window size to in order to saturate the link capacity?

Hide Answer

Set the window size to the bandwidth-delay product of the link, 1 Mbyte/s * 0.1 s = 100 Kbytes.

C. Ben runs his protocol on increasingly higher-speed bottleneck links. At a certain link speed, he finds that his implementation stops working properly. Can you explain what might be happening? What threshold link speed causes this protocol to stop functioning properly?

Hide Answer

Sequence number wraparound causes his protocol to stop functioning properly. When this happens, two packets with different content but the same sequence number are in-flight at once, and so an ack for the firstt spuriously acknowledges the second as well, possibly causing data loss in Ben's "reliable" protocol. This happens when

(255 packets)(1000 bytes/1 packet) = (C bytes/s)(0.1 s)

Therefore C = 2.55 Mbytes/s.

Problem 3. A sender S and receiver R are connected over a network that has k links that can each lose packets. Link i has a packet loss rate of p_i in one direction (on the path from S to R) and q_i in the other (on the path from R to S). Assume that each packet on a link is received or lost independent of other packets, and that each packet's loss probability is the same as any other's (i.e., the random process causing packet losses is indepedent and identically distributed).

A. Suppose that the probability that a data packet does not reach R when sent by S is p and the probability that an ACK packet sent by R does not reach S is q. Write expressions for p and q in terms of the p_i's and q_i's.

Hide Answer

$p = 1 - (1 - p\_1)(1 - p\_2)\ldots(1 - p\_k)$ and
$q = 1 - (1 - q\_1)(1 - q\_2)\ldots(1 - q\_k)$

B. If all p's are equal to some value $\alpha << 1$ (much smaller than 1), then what is p (defined above) approximately equal to?

Hide Answer

$p = 1 - (1 - \&alpha)^k \approx 1 - (1 - k\alpha) = k\alpha$

C. Suppose S and R use a stop-and-wait protocol to communicate. What is the expected number of transmissions of a packet before S can send the next packet in sequence? Write your answer in terms of p and q (both defined above).

Hide Answer

The probability of a packet reception from S to R is 1-p and the probability of an ACK reaching S given that R sent an ACK is 1-q. The sender moves from sequence number k to k + 1 if the packet reaches and the ACK arrives. That happens with probability (1-p)(1-q). The expected number of transmissions for such an event is therefore equal to 1/((1-p)(1-q)).

Problem 4. Consider a 40 kbit/s network link connecting the earth to the moon. The moon is about 1.5 light-seconds from earth.

A. 1 Kbyte packets are sent over this link using a stop-and-wait protocol for reliable delivery, what data transfer rate can be achieved? What is the utilization of the link?

Hide Answer

Stop-and-wait sends 1 packet per round-trip-time so the data transfer rate is 1 Kbyte/3 seconds = 333 bytes/s = 2.6 Kbit/s. The utilization is 2.6/40 = 6.5%.

The estimate above omits the transmission time of the packet. If we include the transmission time (8 kbit/(40 kbit/s) = 0.2 s), the result is 1 kbyte/3.2 seconds = 312 bytes/s = 2.5 Kbits/s.

B. If a sliding-window protocol is used instead, what is the smallest window size that achieves the maximum data rate? Assume that error are infrequent. Assume that the window size is set to achieve the maximum data transfer rate.

Hide Answer

Achieving full rate requires a send window of at least

bandwidth-delay product = 5 packets/s * 3 s = 15 packets.

C. Consider a sliding-window protocol for this link with a window size of 10 packets. If the receiver has a buffer for only 30 undelivered packets (the receiver discards packets it has no room for, and sends no ACK for discarded packets), how bits of sequence number are needed?

Hide Answer

The window size determines the number of unacknowledged packets the transmitter will send before

stalling, but there can be arbitrarily many acknowledged but undelivered (because of one lost packet) packets at the receiver. But if only 30 packets are held at the receiver, after which it stops acknowledging packets except the one it's waiting for, the total number of packets in transit or sitting in the receivers buffer is at most 40.

So a 6-bit sequence number will be sufficent to ensure that all unack'ed and undelivered packets have a unique sequence number (avoiding the sequence number wrap-around problem).

Problem 5. Consider a best-effort network with variable delays and losses. Here, Louis Reasoner suggests that the receiver does not need to send the sequence number in the ACK in a correctly implemented stop-and-wait protocol, where the sender sends packet `k+1` only after the ACK for packet `k` is received. Explain whether he is correct or not.

Hide Answer

(Not surprisingly,) Louis is wrong. Imagine that the sender sends packet `k` and then retransmits `k`. However, the original transmission and the retransmission get through to the receiver. The receiver sends an ACK for `k` when it gets the original transmission, and in response the sender sends packet `k+1`. Now, when the sender gets an ACK, it cannot tell whether the ACK was for packet `k` (the retransmission), or for packet `k+1`!

Problem 6. Consider a sliding window protocol between a sender and a receiver. The receiver should deliver packets reliably and in order to its application.

The sender correctly maintains the following state variables:

`unacked_pkts` -- the buffer of unacknowledged packets

`first_unacked` -- the lowest unacked sequence number (undefined if all packets have been acked)

`last_unacked` -- the highest unacked sequence number (undefined if all packets have been acked)

`last_sent` -- the highest sequence number sent so far (whether acknowledged or not)

If the receiver gets a packet that is strictly larger than the next one in sequence, it adds the packet to a buffer if not already present. We want to ensure that the size of this buffer of packets awaiting delivery *never exceeds* a value $W \geq 0$. Write down the check(s) that the sender should perform before sending a new packet in terms of the variables mentioned above that ensure the desired property.

Hide Answer

The largest sequence number that a receiver could have *possibly* received is `last_sent`. The size of the receiver buffer can become as large as `last_sent - first_unacked`. One might think that we need to add 1 to this quantity, but observe that the only reason any packets get added to the buffer is when some packet is lost (i.e., at least one of the packets in the sender's unacked buffer must have been lost).

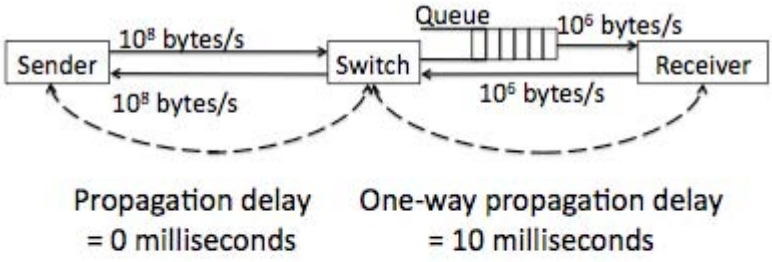
We also need to handle the case when all the packets sent by the sender have been acknowledged -- clearly, in this case, the sender should be able to send data.

Hence, if the sender sends a new packet only if

```
if len(unacked_packets) == 0 or last_sent - first_unacked < W
```

the desired requirement is satisfied.

Problem 7. Ben decides to use the sliding window transport protocol we studied in 6.02 and implemented in the pset on the network below. The receiver sends end-to-end ACKs to the sender. The switch in the middle simply forwards packets in best-effort fashion.



Max queue size = 30 packets
Packet size = 1000 bytes
ACK size = 40 bytes
Initial sender window size = 10 packets

A. The sender's window size is 10 packets. Selecting the best answer from the choices below, at what approximate rate (in packets per second) will the protocol deliver a multi-gigabyte file from the sender to the receiver? Assume that there is no other traffic in the network and packets can only be lost because the queues overflow.
   a. Between 900 and 1000.
   b. Between 450 and 500.
   c. Between 225 and 250.
   d. Depends on the timeout value used.

Hide Answer

Choice b. The RTT, which is the time taken for a single packet to reach the receiver and the ACK to return, is about 20 milliseconds plus the transmission time, which is about 1 millisecond (1000 bytes at a rate of 1 Megabyte/s). Hence, the throughput is 10 packets / 21 milliseconds = 476 packets per second. If one ignored the transmission time, which is perfectly fine given the set of choices, one would estimate the throughput to be about 500 packets per second.

B. You would like to double the throughput of this sliding window transport protocol running on the network shown on the previous page. To do so, you can apply one of the following techniques alone:
   a. Double the window size.
   b. Halve the propagation time of the links.
   c. Double the speed of the link between the Switch and Receiver.

   For each of the following sender window sizes, list which of the above techniques, if any, can approximately double the throughput. If no technique does the job, answer "None". There might be more than one answer for each window size, in which case you should list them all. Note that each technique works in isolation. Explain your answers.

   1. W = 10.

Hide Answer

A and B.

When W = 10, the throughput is about 476 packets/s. If we double the window size, throughput would double to 952 packet/s. If we reduce the propagation time of the links, throughput would roughly double as well. The new throughput would still be smaller than the bottleneck capacity of 1000 packets/s.

2. W = 50.

Hide Answer

C.

When W = 50, throughput is already 1000 packets/s. At this stage, doubling the window or halving the RTT does not increase the throughput. If we double the speed of the link between the Switch and Receiver, the bottleneck becomes 2000 packets/s. A window size of 50 packets over an RTT of 20 or 21 milliseconds has a throughput of more than 2000 packet/s. Hence, doubling the bottleneck link speed will double the throughput when W = 50 packets. With a queue size of 30 packets and a window size of 50, the initial window of packets sent back-to-back would indeed cause the queue to overflow. However, that doesn't cause the throughput to drop in the steady state, so for a long data transfer, the throughput will be as described above.

3. W = 30.

Hide Answer

None.

When W = 30, throughput is already 1000 packets/s. Now, if we double the window or halve the RTT, the throughput won't change. An interesting situation occurs when we double the link speed, because the bottleneck link would now be capable of delivering 2000 packets/s. But our window size is 30 and RTT about 20 milliseconds, giving a throughput of about 1500 packets/s (or if we use 21 milliseconds, we get 1428 packet/s). That's an improvement of about 50%, far from the doubling we wanted. None of the techniques work.

Problem 8. Consider the sliding window protocol described in lecture and implemented in the pset. The receiver sends "ACK k" when it receives a packet with sequence number k. Denote the window size by W. The sender's packets start with sequence number 1. Which of the following is true of a correct implementation of this protocol over a best-effort network?

A. Any new (i.e., previously unsent) packet with sequence number *greater than* W is sent by the sender if, and only if, a new (i.e., previously unseen) ACK arrives.

Hide Answer

True.

B. The sender will never send more than one packet between the receipt of one ACK and the next.

Hide Answer

False. The sender could time-out and retransmit.

C. The receiver can discard any new, out-of-order packet it receives after sending an ACK for it.

Hide Answer

False. The sender thinks the receiver has delivered this packet to the application.

D. Suppose that no packets or ACKs are lost and no packets are ever retransmitted. Then ACKs will arrive at the sender in non-decreasing order.

Hide Answer

False. Packets or ACKs could get reordered in the network.

E. The sender should retransmit any packet for which it receives a duplicate ACK (i.e., an ACK it has received earlier).

Hide Answer

False. Duplicate ACKs can be ignored by the sender.

Problem 9. In his haste in writing the code for the exponential weighted moving average (EWMA) to estimate the smoothed round-trip time, `srtt`, Ben Bitdiddle writes

```
srtt = alpha * r + alpha * srtt
```

where `r` is the round-trip time (RTT) sample, and `0 < alpha < 1`.

For what values of alpha does this buggy EWMA over-estimate the intended `srtt`? You may answer this question assuming any convenient non-zero sequence of RTT samples, r.

Hide Answer

This buggy EWMA over-estimates the true `srtt` when alpha > 0.5. The true `srtt` is equal to

```
alpha * r + (1-alpha) * srtt
```

So

```
alpha * r + alpha * srtt > alpha * r + (1-alpha) * srtt
```

implies `alpha > 0.5`.

Problem 10. A sender S and receiver R communicate reliably over a series of links using a sliding window protocol with some window size, W packets. The path between S and R has one bottleneck link (i.e., one link whose rate bounds the throughput that can be achieved), whose data rate is C packets/second. When the window size is W, the queue at the bottleneck link is always full, with Q data packets in it. The round trip time (RTT) of the connection between S and R during this data transfer with window size W is T seconds. There are no packet or ACK losses in this case, and there are no other connections sharing this path.

A. Write an expression for W in terms of the other parameters specified above.

Hide Answer

W = C*T. Note that some students may interpret T as the RTT without any queueing. That's wrong, but

we ought to still give them credit as long as they have consistently made this mistake in all the parts. With this interpretation, W = CT + Q.

B. We would like to reduce the window size from W and still achieve high utilization. What is the minimum window size, Wmin, which will achieve 100% utilization of the bottleneck link? Express your answer as a function of C, T, and Q.

Hide Answer

Clearly, W = Wmin + Q, where Wmin is the smallest window size that gives 100% utilization. A smaller window than that would keep the network idle some fraction of the time. Hence, Wmin = C*T - Q.

With the flawed interpretation of T, Wmin = C*T.

C. Now suppose the sender starts with a window size set to Wmin. If all these packets get ac- knowledged and no packet losses occur in the window, the sender increases the window size by 1. The sender keeps increasing the window size in this fashion until it reaches a window size that causes a packet loss to occur. What is the smallest window size at which the sender observes a packet loss caused by the bottleneck queue overflowing? Assume that no ACKs are lost.

Hide Answer

Packets start getting dropped when the window size is W+1, i.e., when it is equal to C*T+1.

With the flawed interpretation of T, the window size at which packets start being dropped is W+1 = C*T + Q + 1.

Problem 11. A sender A and a receiver B communicate using the stop-and-wait protocol studied in 6.02. There are n links on the path between A and B, each with a data rate of R bits per second. The size of a data packet is S bits and the size of an ACK is K bits. Each link has a physical distance of D meters and the speed of signal propagation over each link is c meters per second. The total processing time experienced by a data packet and its ACK is T_p seconds. ACKs traverse the same links as data packets, except in the opposite direction on each link (the propagation time and data rate are the same in both directions of a link). There is no queueing delay in this network. Each link has a packet loss probability of p, with packets being lost independently.

What are the following four quantities in terms of the given parameters?

A. Transmission time for a data packet on one link between A and B.

Hide Answer

S/R. Each data packet has size S bits, and the speed of the link is R bits per second.

B. Propagation time for a data packet across n links between A and B.

Hide Answer

nD/C . Total distance to be travelled is nD since each link has length D meters, and there are n such links. The propagation speed is C meters/second.

C. Round-trip time (RTT) between A and B?. (The RTT is defined as the elapsed time between the start of transmission of a data packet and the completion of receipt of the ACK sent in response to the data packetâ€™s reception by the receiver.)

Hide Answer

```
nS/R + nK/r + 2nD/C + T_p
```

We need to consider the following times:

- Transmit data across n links: nS/R using result from part A.
- Transmit ACK across n links: nK/R also using result from part A.
- Propagate data across n links and ACKS across n links: 2nD/C
- Total time to process the data and the ACK: T_p

D. Probability that a data packet sent by A will reach B.

Hide Answer

$(1- p)^n$. Probability of loss in a link is p, so probability of no loss in a link is 1-p. Since link losses are independent, probability of no loss in n links is $(1-p)^n$. No loss in n links means the data gets from A to B successfuly.

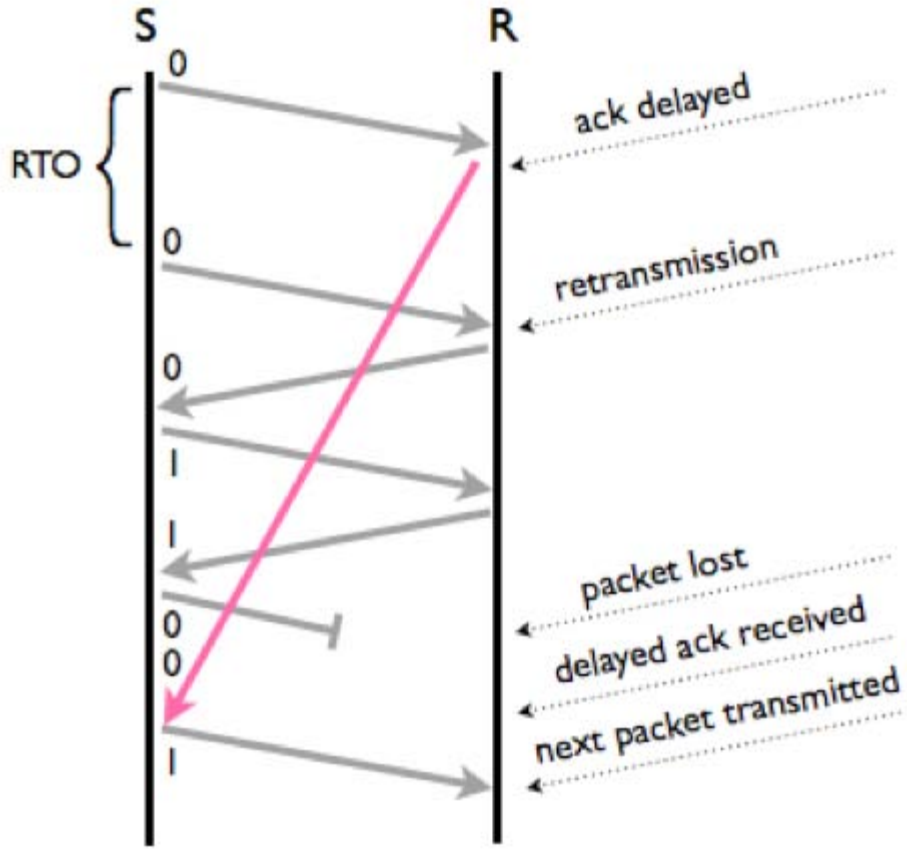
Problem 12.

Opt E. Miser implements the 6.02 stop-and-wait reliable transport protocol with one modification: being stingy, he replaces the sequence number field with a 1-bit field, deciding to reuse sequence numbers across data packets. The first data packet has sequence number 1, the second has number 0, the third has number 1, the fourth has number 0, and so on. Whenever the receiver gets a packet with sequence number s (= 0 or 1), it sends an ACK to the sender echoing s. The receiver delivers a data packet to the application if, and only if, its sequence number is different from the last one delivered, and upon delivery, updates the last sequence number delivered.

D. He runs this protocol over a best-effort network that can lose packets (with probability less than 1) or reorder them, and whose delays may be variable. Does the modified protocol always provide correct reliable, in-order delivery of a stream of packets?

Hide Answer

No. For example, see the picture below.



Problem 13. Consider a reliable transport connection using the 6.02 sliding window protocol on a network path whose RTT in the absence of queueing is RTTmin = 0.1 seconds. The connection's bottleneck link has a rate of C = 100 packets per second, and the queue in front of the bottleneck link has space for Q = 20 packets.

Assume that the sender uses a sliding window protocol with fixed window size. There is no other connection on the path.

A. If the size of the window is 8 packets, then what is the throughput of the connection?

Hide Answer

The bandwidth-delay product of the connection is 10 packets (bottleneck rate times the minimum RTT). With a window size of 8, queues will not yet have built up, so the throughput is 80 packets/second.

B. If the size of the window is 16 packets, then what is the throughput of the connection?

Hide Answer

The bandwidth-delay product of the network is 10 packets, so if W >= 10, there will be 10 packets in flight. With W=16, 6 of these packets will be in the queue. The queuing delay will be 6/100 = 0.06 seconds. Then RTT = RTTmin + queuing delay = .1 + .06 = 0.16 and the throughput is W/RTT = 16/.16 = 100 pkts/s.

C. What is the smallest window size for which the connection's RTT exceeds RTTmin?

Hide Answer

11 packets. The bandwidth-delay product is 10 packets. It's probably reasonable to accept an answer of 10 packets too.

Problem 14. TCP, the standard reliable transport protocol used on the Internet, uses a sliding window. Unlike the protocol studied in 6.02, however, the size of the TCP window is variable. The sender changes the size of the window as ACKs arrive from the receiver; it does not know the best window size to use a priori.

TCP uses a scheme called *slow start* at the beginning of a new connection. Slow start has three rules, R1, R2, and R3, listed below (TCP uses some other rules too, which we will ignore).

In the following rules for slow start, the sender's current window size is W and the last in-order ACK received by the sender is A. The first packet sent has sequence number 1.

R1. Initially, set W ← 1 and A ← 0.
R2. If an ACK arrives for packet A+1, then set W←W+1, and set A←A+1.
R3. When the sender retransmits a packet after a timeout, then set W←1.

Assume that all the other mechanisms are the same as the 6.02 sliding window protocol. Data packets may be lost because packet queues overflow, but assume that packets are not reordered by the network.

We run slow start on a network with RTTmin = 0.1 seconds, bottleneck link rate = 100 packets per second, and bottleneck queue = 20 packets.

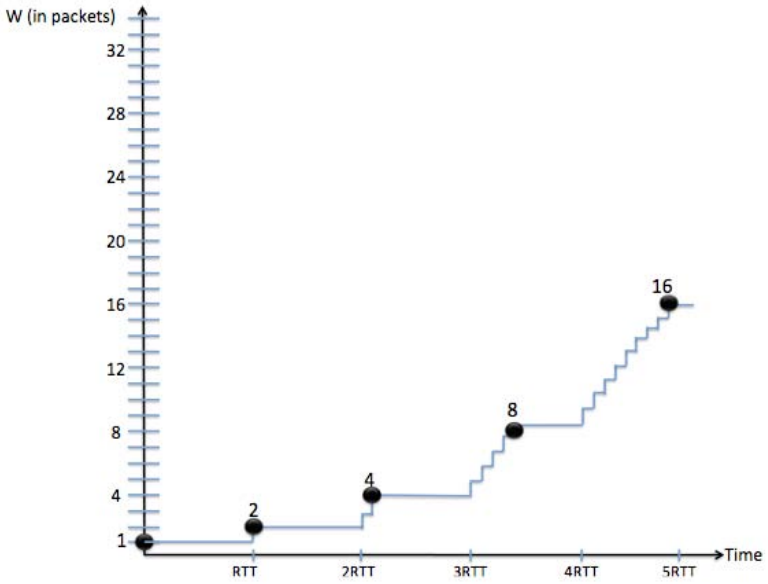
A. What is the smallest value of W at which the bottleneck queue overflows?

Hide Answer

The smallest W for which the queue overflows is 10 + 20 + 1 = 31 packets. The 10 is because that's the bandwidth-delay product; the 20 is the maximum size of the queue. And we need one more packet to cause an overflow.

B. Sketch W as a function of time for the first 5 RTTs of a connection. The X-axis marks time in terms of multiples of the connection's RTT. (Hint: Non-linear!)

Hide Answer


W (in packets)
32
28
24
20
16
12
8
4
1
2
4
8
16
Time
RTT
2RTT
3RTT
4RTT
5RTT

6.02 Introduction to EECS II: Digital Communication Systems

Fall 2012