Shivam Patel

COMP 343

Dordal

Assignment 4-Chap 6: 1,2,3,4,5,6,8

1. Sketch a ladder diagram for stop-and-wait if Data[3] is lost the first time it is sent. Continue the diagram to the point where Data[4] is successfully transmitted. Assume an RTT of 1 second, **no sender timeout** (but the sender retransmits on duplicate), and a *receiver* timeout of 2 seconds.

Sender Receiver

Data[3]

Data[3]

ACK[3]

Data[4]

ACK[4]

1. Suppose a stop-and-wait receiver has an implementation flaw. When Data[1] arrives, ACK[1] and ACK[2] are sent, separated by a brief interval; after that, the receiver transmits ACK[N+1] when Data[N] arrives, rather than the correct ACK[N].
2. Assume the sender responds to each ACK as soon as it arrives. Explain why the sender will not be able to detect this receiver problem until after it has sent its final packet, Data[M], and receives an unexpected ACK[M+1]. Hint: draw a diagram.

Since the all the data sent is returned with an ACK, even though it is before the data sent, the sender doesn’t recognize the error until the last ACK comes back for a data[n+1] which was never sent.

Sender Receiver

Data[1]

ACK[1]

ACK[2] Data[2]

Data[3]

ACK[3]

ACK[4] Data[4]

ACK[5]

1. Is there anything the transmitter can do to detect this receiver problem earlier?

Not necessarily unless the sender pauses all data being sent

1. Create a table as in [6.3.1   Simple fixed-window-size analysis](http://intronetworks.cs.luc.edu/current/html/slidingwindows.html#fixed-window-size) for the original

A───R1───R2───R3───R4───B network with winsize = 8. As in the text examples, assume 1 packet/sec bandwidth delay for the R1⟶R2, R2⟶R3, R3⟶R4 and R4⟶B links. The A–R link and all reverse links (from B to A) are infinitely fast. Carry out the table for 10 seconds.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Time** | **A sends** | **R1 queues** | **R1 sends** | **R2 sends** | **R3 sends** | **R4 sends** | **B ACK’s** |
| 0 | 1,2,3,4,5,6,7,8 | 2,3,4,5,6,7,8 | 1 |  |  |  |  |
| 1 |  | 3,4,5,6,7,8 | 2 | 1 |  |  |  |
| 2 |  | 4,5,6,7,8 | 3 | 2 | 1 |  |  |
| 3 |  | 5,6,7,8 | 4 | 3 | 2 | 1 |  |
| 4 | 9 | 6,7,8,9 | 5 | 4 | 3 | 2 | 1 |
| 5 | 10 | 7,8,9,10 | 6 | 5 | 4 | 3 | 2 |
| 6 | 11 | 8,9,10,11 | 7 | 6 | 5 | 4 | 3 |
| 7 | 12 | 9,10,11,12 | 8 | 7 | 6 | 5 | 4 |
| 8 | 13 | 10,11,12,13 | 9 | 8 | 7 | 6 | 5 |
| 9 | 14 | 11,12,13,14 | 10 | 9 | 8 | 7 | 6 |
| 10 | 15 | 12,13,14,15 | 11 | 10 | 9 | 8 | 7 |

1. Create a table as in [6.3.1   Simple fixed-window-size analysis](http://intronetworks.cs.luc.edu/current/html/slidingwindows.html#fixed-window-size) for a network

A───R1───R2───B. The A–R1 ink is infinitely fast; the R1–R2 and R2–B each have a 1-second**propagation** delay, in each direction, and zero *bandwidth* delay (that is, one packet takes 1.0 sec to travel from R1 to R2; two packets also take 1.0 sec to travel from R1 to R2). Assume winsize=6. Carry out the table for 8 seconds. Note that with zero bandwidth delay, multiple packets sent together will remain together until the destination.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Time** | **A sends** | **R1 queues** | **R1 sends** | **R2 sends** | **B ACK’s** |
| 0 | 1,2,3,4,5,6 | 1,2,3,4,5,6 |  |  |  |
| 1 |  |  | 1,2,3,4,5,6 |  |  |
| 2 |  |  |  | 1,2,3,4,5,6 |  |
| 3 |  |  |  |  | 1,2,3,4,5,6 |
| 4 | 7 | 7 |  |  |  |
| 5 | 8 | 8 | 7 |  |  |
| 6 | 9 | 9 | 8 | 7 |  |
| 7 | 10 | 10 | 9 | 8 | 7 |
| 8 | 11 | 11 | 10 | 9 | 8 |

1. Suppose RTTnoLoad = 4 seconds and the bottleneck bandwidth is 1 packet every 2 seconds.
2. What window size is needed to remain just at the knee of congestion?

Winsize = (1 packet/ 2seconds) x 4sec = 2

A winsize of **4** would be more optimal

1. Suppose winsize=6. How many packets are in the queue, at the steady state, and what is RTTactual?

Packets in Queue= 6 – 4 = **2 packets**

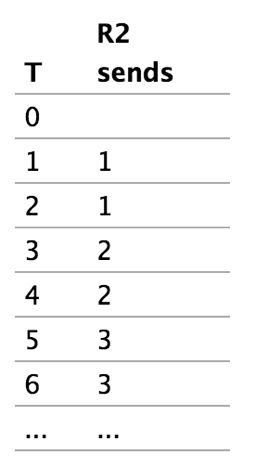
Steady State = queue\_usage = 6 packets x (1-(4 sec/12secs)) =

6 packets x (2/3 seconds) = **4 packets/sec**

RTTactual  = 6 packets / (1 packet/2sec) =**12 sec**

1. Create a table as in [6.3.1   Simple fixed-window-size analysis](http://intronetworks.cs.luc.edu/current/html/slidingwindows.html#fixed-window-size) for a network

A───R1───R2───R3───B. The A–R1 link is infinitely fast. The R1–R2 and R3–B links have a bandwidth delay of 1 packet/second with no additional propagation delay. The R2–R3 link has a bandwidth delay of 1 packet / 2 seconds, and no propagation delay. The reverse B⟶A direction (for ACKs) is infinitely fast. Assume winsize = 6. Carry out the table for 10 seconds. Note that in this exercise you will need to show the queue for both R1 and R2. (If you carry out the pattern at least partially until T=18, you can verify that RTTactual for packet 8 is as calculated in the previous exercise. You will need more than 10 packets, but fewer than 16; the use of hex labels A, B, C for packets 10, 11, 12 is a convenient notation.)

Hint: The column for “R2 sends” (or, more literally, “R2 is in the process of sending”) should look like this:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Time** | **A sends** | **R1 queues** | **R1 sends** | **R2 queues** | **R2 sends** | **R3 sends** | **B ACK’s** |
| 0 | 1,2,3,4,5,6 | 2,3,4,5,6 | 1 |  |  |  |  |
| 1 |  | 3,4,5,6 | 2 |  | 1 |  |  |
| 2 |  | 4,5,6 | 3 | 2 | 1 |  |  |
| 3 |  | 5,6 | 4 | 3 | 2 | 1 |  |
| 4 | 7 | 6,7 | 5 | 3,4 | 2 | 1 | 1 |
| 5 |  | 7 | 6 | 4,5 | 3 | 2 |  |
| 6 | 8 | 8 | 7 | 4,5,6 | 3 | 2 | 2 |
| 7 |  |  | 8 | 5,6, | 4 | 3 |  |
| 8 | 9 | 9 |  | 5,6,7,8 | 4 | 3 | 3 |
| 9 |  |  | 9 | 6,7,8,9 | 5 | 4 |  |
| 10 | 10 | 10 |  | 6,7,8,9 | 5 | 4 | 4 |

1. Suppose RTTnoLoad is 50 ms and the available bandwidth is 2,000 packets/sec. Sliding windows is used for transmission.
2. What window size is needed to remain just at the knee of congestion?

Winsize = (2000packets/sec) x (50ms) x (1sec/1000ms) = **100 packets**

1. If RTTactual rises to 60 ms (due to use of a larger winsize), how many packets

are in a queue at any one time?

Queuetime= 100 packets x (1-(50ms/60ms)) =

100 packets x (1/6sec) =

**16.7 packets/sec**

1. What value of winsize would lead to RTTactual = 60 ms?

Winsize=60ms x (1sec/1000ms) x (2000packet/sec) = **120 packets**

1. What value of winsize would make RTTactual rise to 100 ms?

Winsize = 100ms x (1sec/1000ms) x (2000packet/sec) = **200 packets**