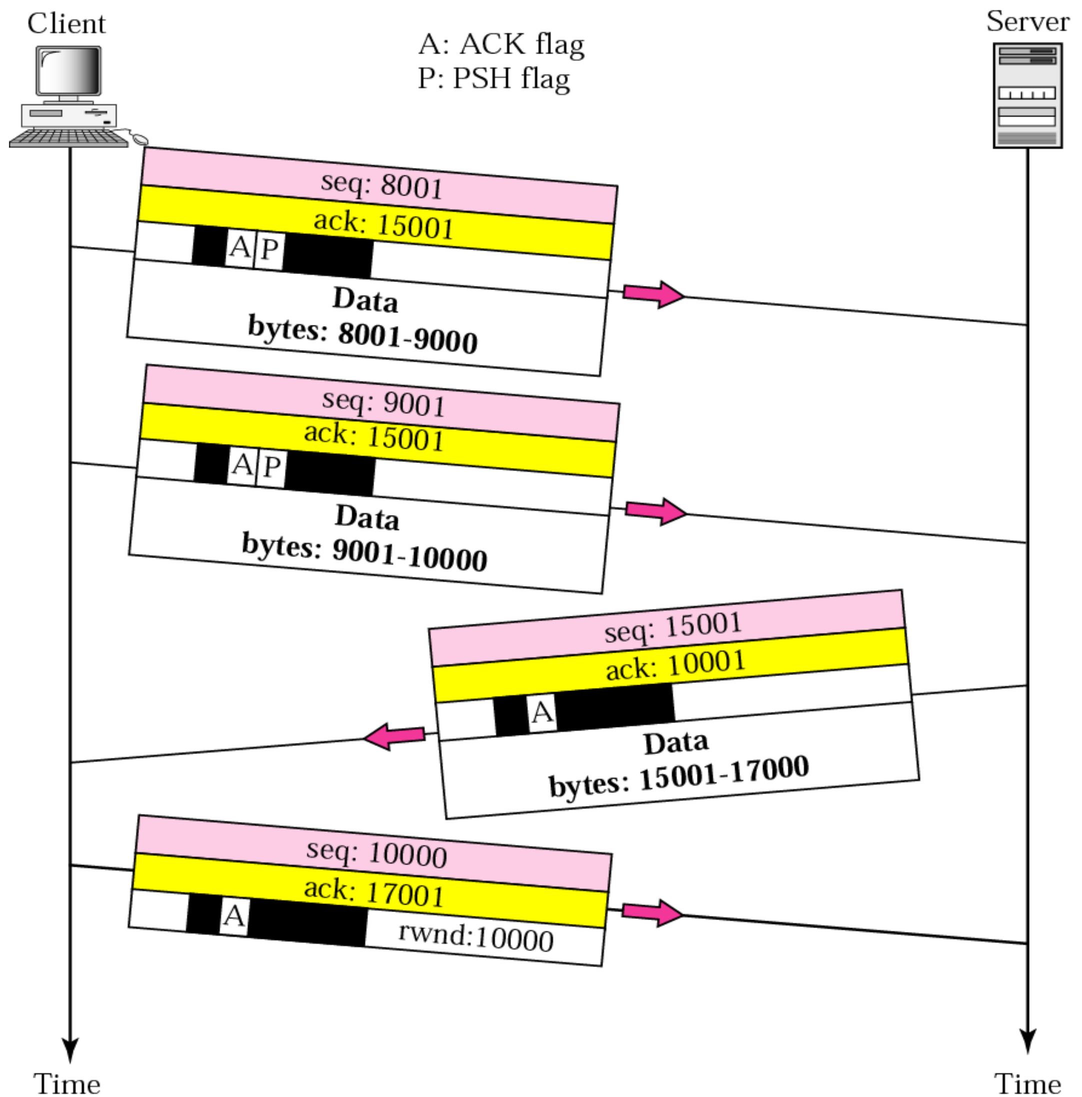
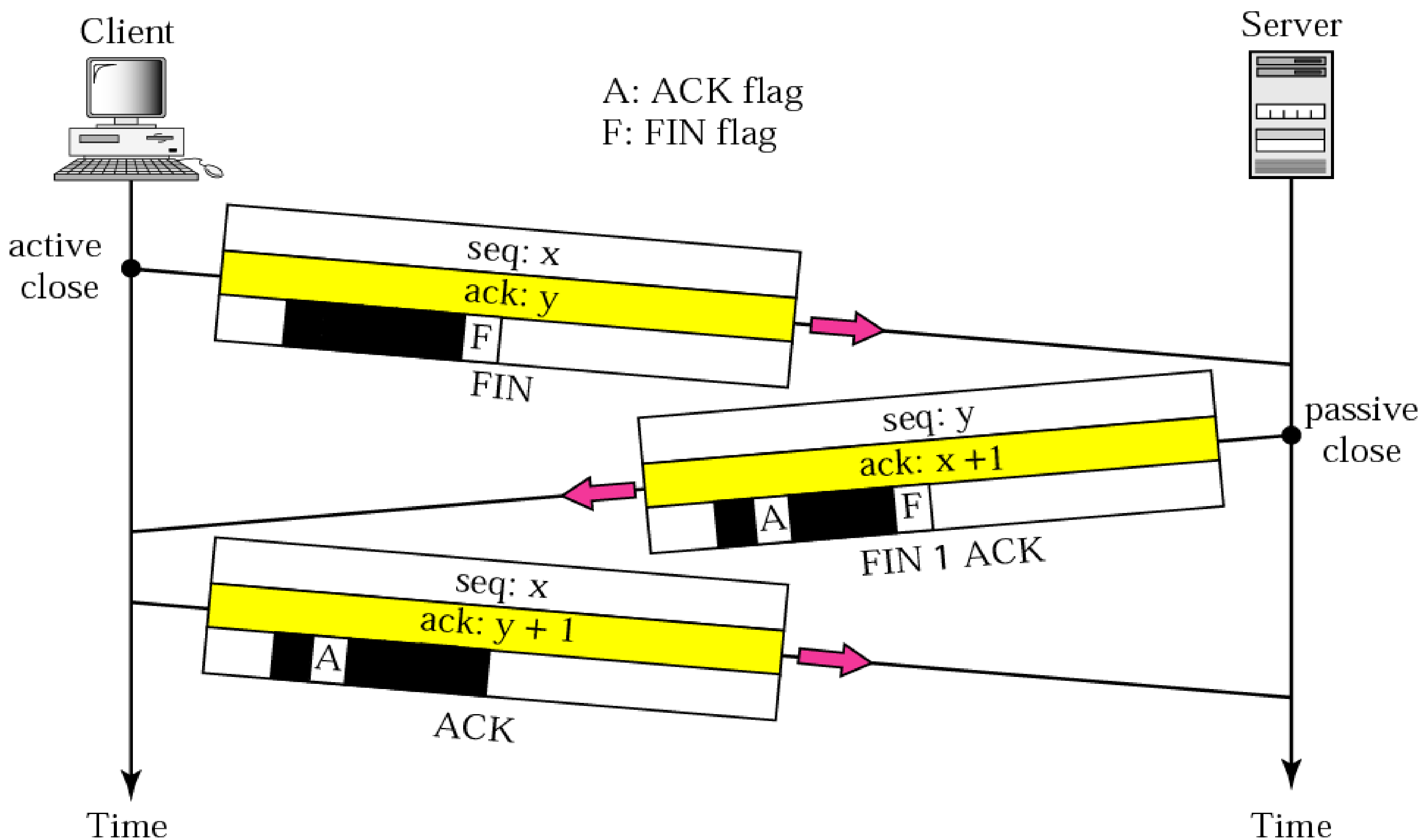
Figure 12.10 ***Data transfer***



Note:

The FIN segment consumes one sequence number if it does not carry data.

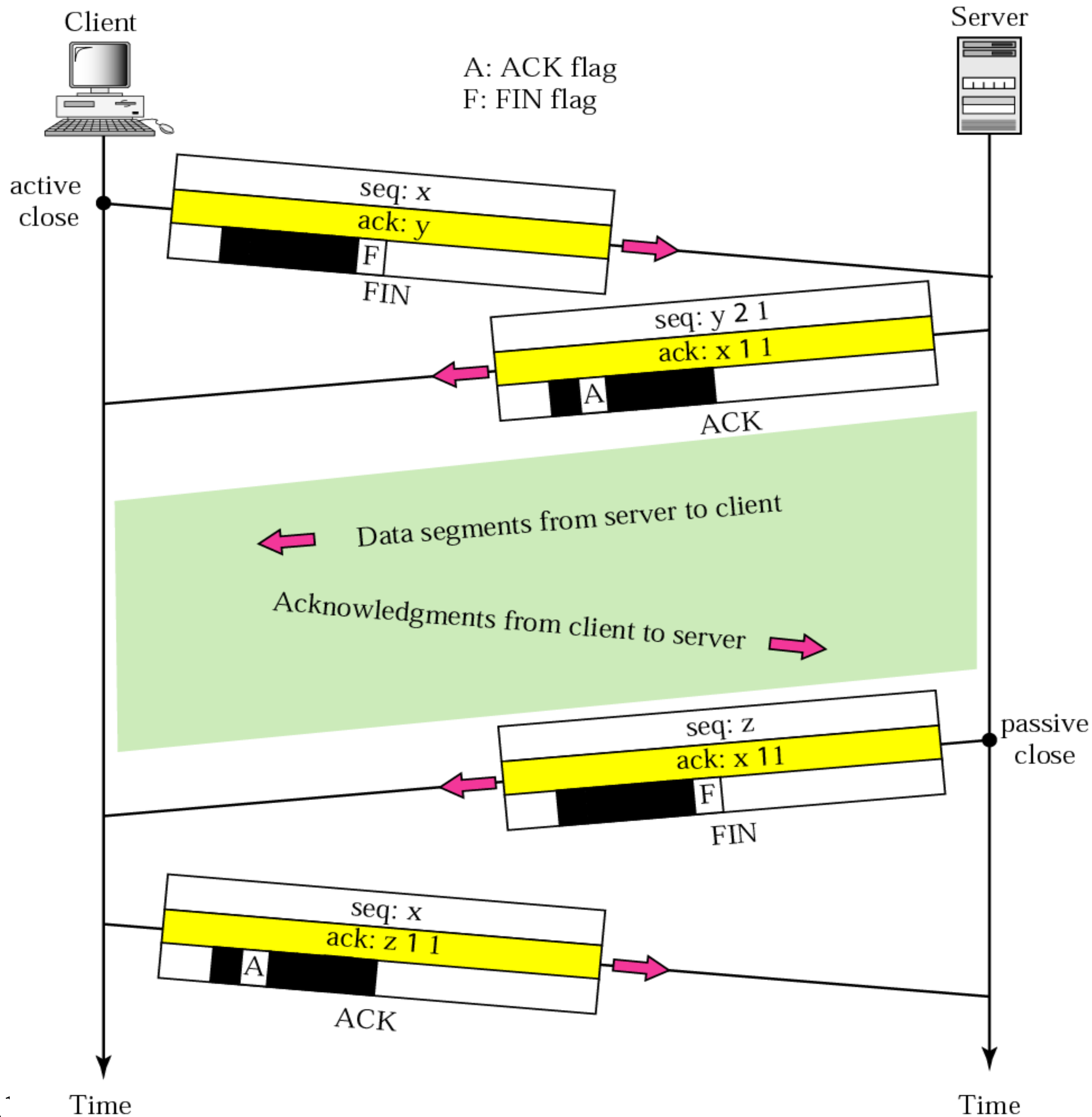
Figure 12.11 *Connection termination using three-way handshaking*



Note:

The FIN + ACK segment consumes one sequence number if it does not carry data.

Figure 12.12 *Half-close*



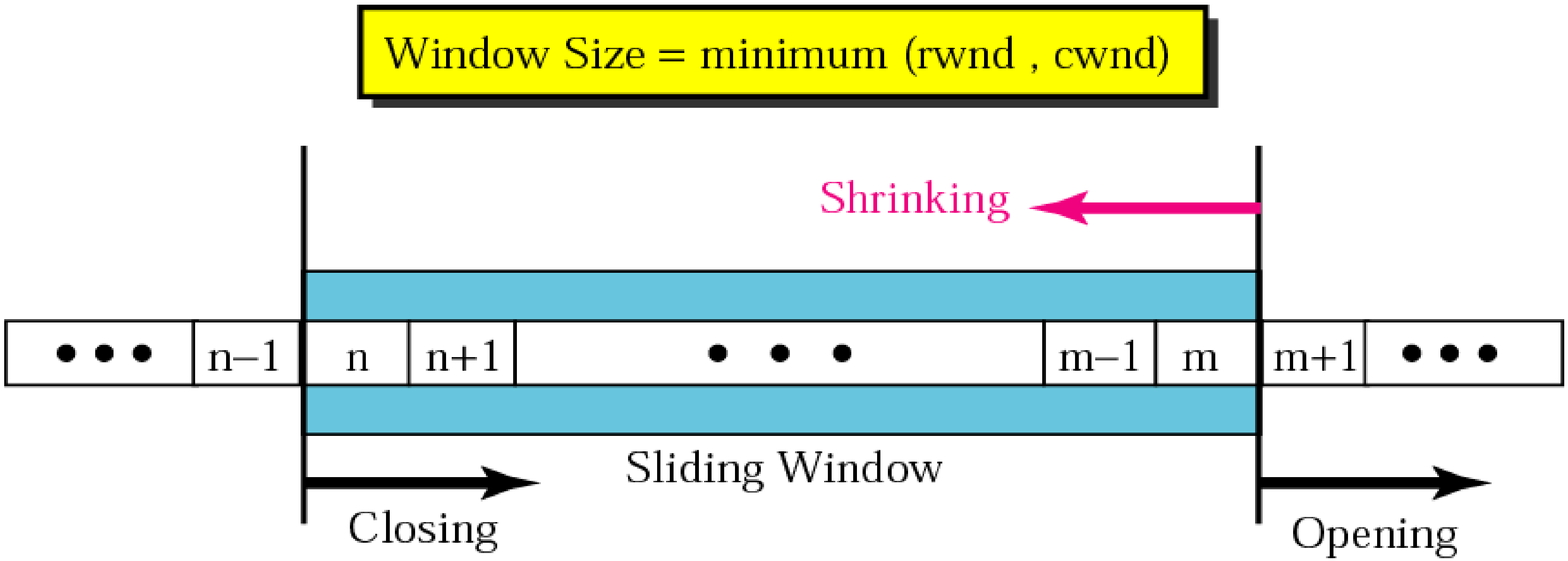
12.6 FLOW CONTROL

Flow control regulates the amount of data a source can send before receiving an acknowledgment from the destination. TCP defines a window that is imposed on the buffer of data delivered from the application program.

The topics discussed in this section include:

Sliding Window Protocol
Silly Window Syndrome

Figure 12.20 *Sliding window*



Note:

A sliding window is used to make transmission more efficient as well as to control the flow of data so that the destination does not become overwhelmed with data.

TCP's sliding windows are byte oriented.

Example 4

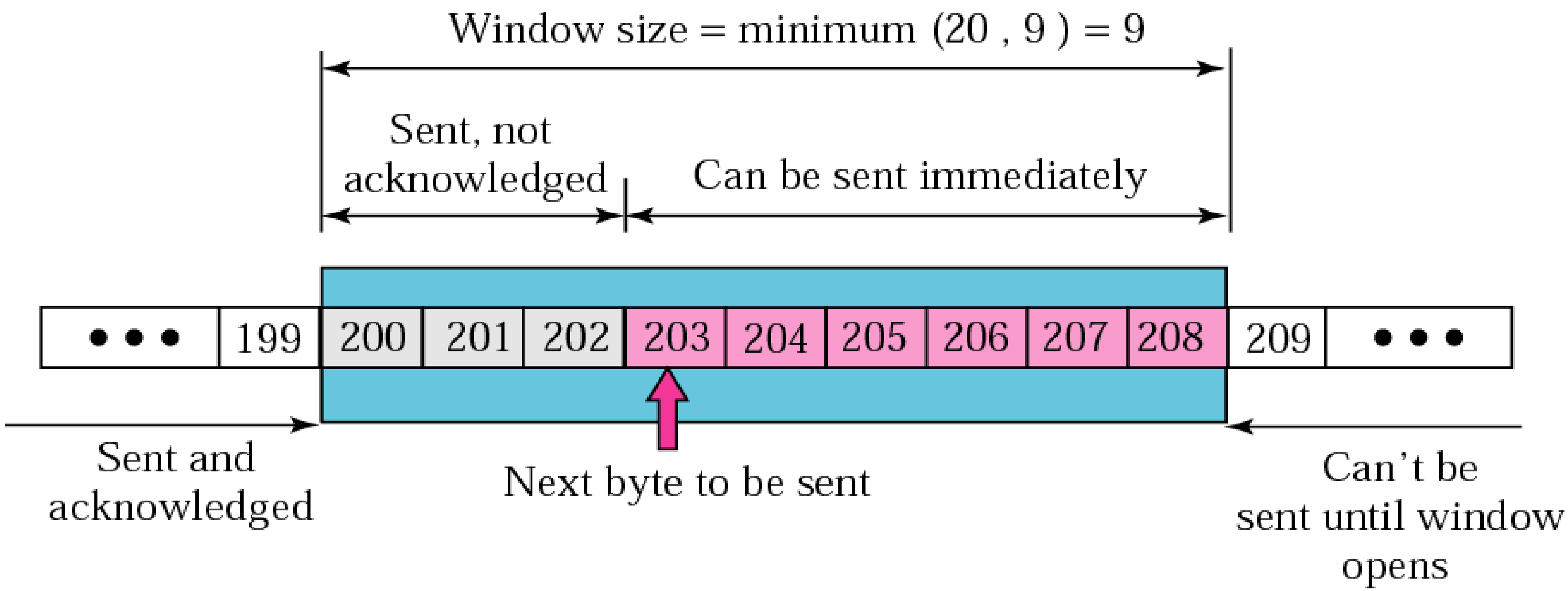
What is the size of the window for host A if the value of rwnd is 3,000 bytes and the value of cwnd is 3,500 bytes?

Solution

The size of the window is the smaller of rwnd and cwnd, which is 3,000 bytes.

Example 5

Figure 12.21 shows an unrealistic example of a sliding window. The sender has sent bytes up to 202. We assume that cwnd is 20 (in reality this value is thousands of bytes). The receiver has sent an acknowledgment number of 200 with an rwnd of 9 bytes (in reality this value is thousands of bytes). The size of the sender window is the minimum of rwnd and cwnd or 9 bytes. Bytes 200 to 202 are sent, but not acknowledged. Bytes 203 to 208 can be sent without worrying about acknowledgment. Bytes 209 and above cannot be sent.


Window size = minimum (20 , 9) = 9
Sent, not acknowledged
Can be sent immediately
• • •
199
200
201
202
203
204
205
206
207
208
209
• • •
Sent and acknowledged
Next byte to be sent
Can't be sent until window opens

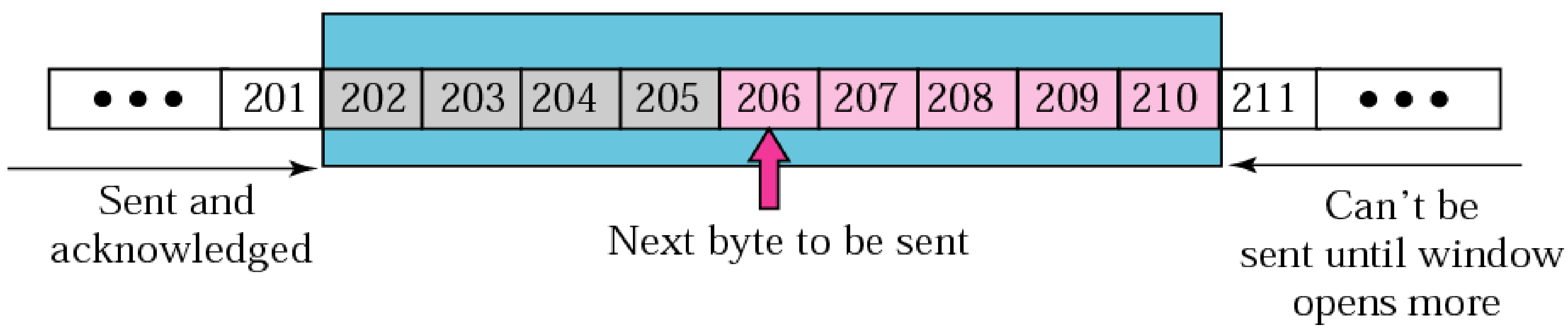
Example 6

In Figure 12.21 the server receives a packet with an acknowledgment value of 202 and an rwnd of 9. The host has already sent bytes 203, 204, and 205. The value of cwnd is still 20. Show the new window.

Solution

Figure 12.22 shows the new window. Note that this is a case in which the window closes from the left and opens from the right by an equal number of bytes; the size of the window has not been changed. The acknowledgment value, 202, declares that bytes 200 and 201 have been received and the sender needs not worry about them; the window can slide over them.

Figure 12.22 *Example 6*



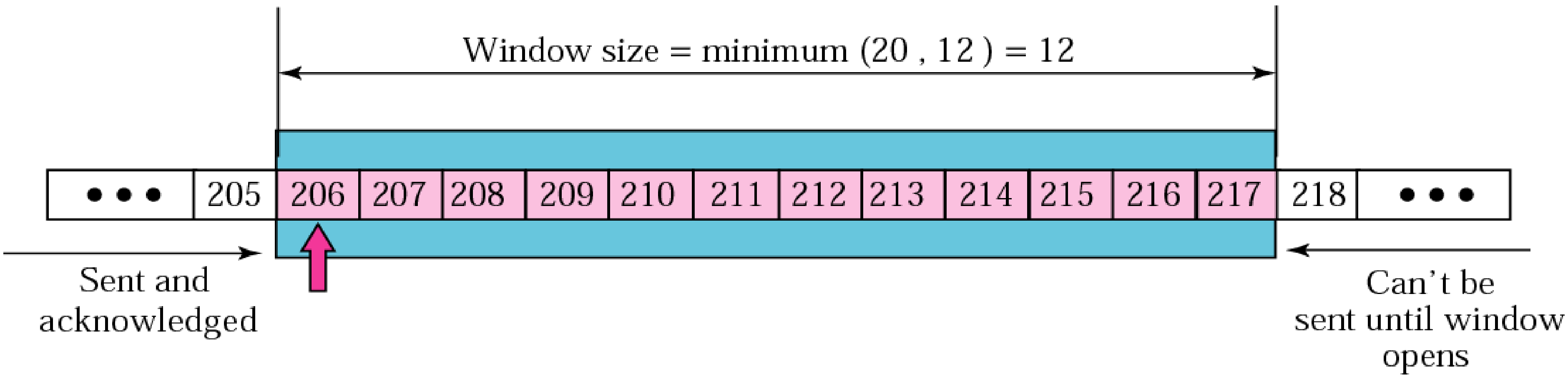
Example 7

In Figure 12.22 the sender receives a packet with an acknowledgment value of 206 and an rwnd of 12. The host has not sent any new bytes. The value of cwnd is still 20. Show the new window.

Solution

The value of rwnd is less than cwnd, so the size of the window is 12. Figure 12.23 shows the new window. Note that the window has been opened from the right by 7 and closed from the left by 4; the size of the window has increased.

Figure 12.23 *Example 7*



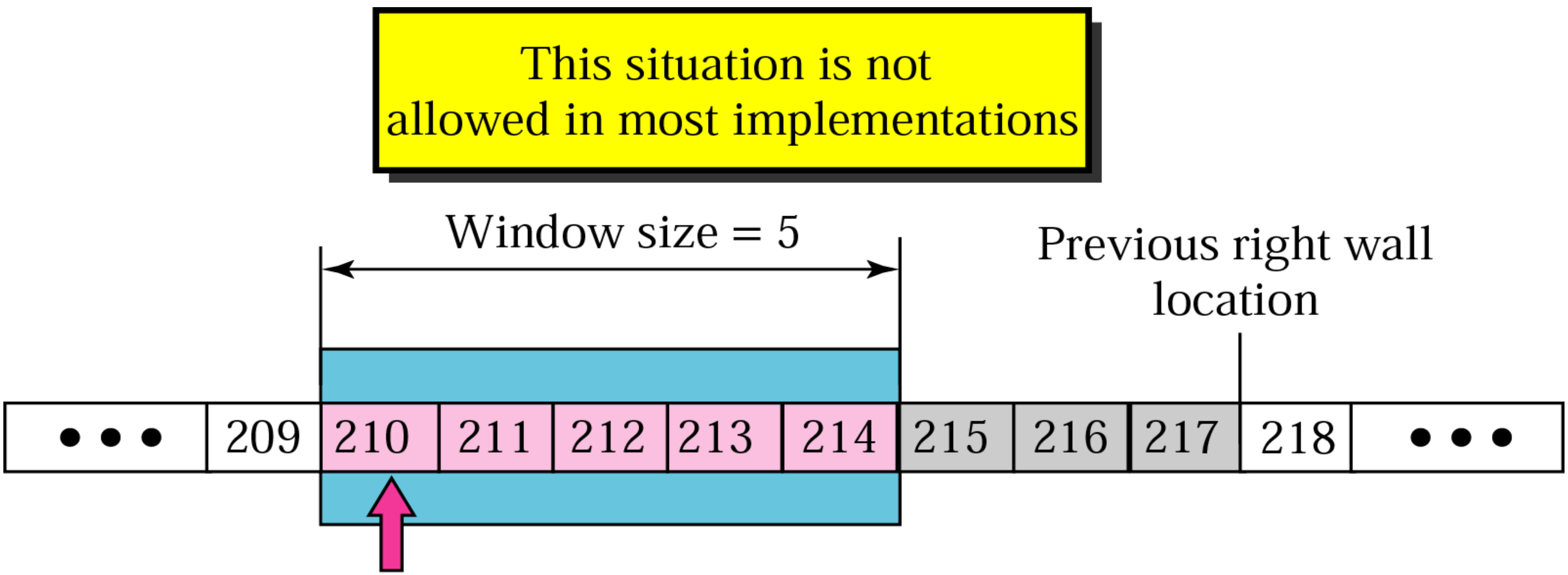
Example 8

In Figure 12.23 the host receives a packet with an acknowledgment value of 210 and an rwnd of 5. The host has sent bytes 206, 207, 208, and 209. The value of cwnd is still 20. Show the new window.

Solution

The value of rwnd is less than cwnd, so the size of the window is 5. Figure 12.24 shows the situation. Note that this is a case not allowed by most implementations. Although the sender has not sent bytes 215 to 217, the receiver does not know this.

Figure 12.24 *Example 8*



Example 9

How can the receiver avoid shrinking the window in the previous example?

Solution

The receiver needs to keep track of the last acknowledgment number and the last rwnd. If we add the acknowledgment number to rwnd we get the byte number following the right wall. If we want to prevent the right wall from moving to the left (shrinking), we must always have the following relationship.

$$\textit{new ack} + \textit{new rwnd} \geq \textit{last ack} + \textit{last rwnd}$$

or

$$\textit{new rwnd} \geq (\textit{last ack} + \textit{last rwnd}) - \textit{new ack}$$

Note:

To avoid shrinking the sender window, the receiver must wait until more space is available in its buffer.