PUSH Flag

A notification from the sender to the receiver to pass all the data the receiver has to the receiving application.

Some implementations of TCP automatically set the push flag if the data in the segment being sent empties the send buffer.

These same implementations ignore a received PUSH flag because they normally deliver the received data to the application as soon as possible.

In Fig 20.3 missing 3 PSH because sends did not empty the send buffer. Application has already sent all of its data to the send buffer so only last write empties the send buffer.

We cannot manually set this flag through the Sockets Application Programming Interface.
Sliding Windows

[Fig 20.4]

Sliding window example

[Fig 20.6] and [Fig 20.1]
The sliding window protocol that we observed in the previous section can be visualized as shown in Figure 20.4.

Figure 20.4 Visualization of TCP sliding window.
0.0

0.002185 (0.0022)

0.007295 (0.0051)

0.017868 (0.0106)

0.022699 (0.0048)

0.027650 (0.0050)

0.027799 (0.0001)

0.031881 (0.0041)

0.034789 (0.0029)
Figure 20.1 Transfer of 8192 bytes from svr4 to bsdi.
Figure 20.6 shows the dynamics of TCP's sliding window protocol for the data transfer in Figure 20.1.

Figure 20.6  Sliding window protocol for Figure 20.1.
Slow Start

Stevens’ examples did not use slow start up until now. All TCP implementations must now use the slow start algorithm. You must always use slow start in this class. Some examples in the book do not do this, in this class you must ALWAYS use slow start.

An Intermediate router queue may run out of space. Best not to send too much too fast. Slow start algorithm notes that the rate it should inject new packets into the network is the rate at which acknowledgements are returned by the other end.

ADD A CONGESTION WINDOW “cwnd” on the SENDER side (Note different from (buffer) window discussed before.)

- In a new connection, the congestion window is initialized to one MSS announced by other end.

- cwnd is maintained in bytes however cwnd is incremented by segment size.

- For each ACK received cwnd is increased by one segment.
- **Sender can transmit up to minimum of (the congestion window and the advertised window)**
  (Congestion window cwnd is set by sender while advertised window is set by receiver).

- Sender starts by transmitting one segment and waiting for its ACK. When that ACK is
  acknowledged, congestion window is incremented from one to two and two segments
  can be sent.

- When each of those two segments acknowledged, congestion window set to four.

- This provides an **exponential** increase.

- At some point an intermediate router will discard packets. Congestion window
  cwnd is too large.
Slow Start Example

[ Fig 20.8] MSS=512

Bulk data throughput - interaction of window size, windowed flow control, and slow start on the throughput of a TCP connection.

[Fig 20.9 & 20.10]
- At time 0 sender transmits one segment cwnd = 1, must wait for ACK.
- At times 1, 2, 3, segment moves one time unit right.
- At time 4 ACK generated.
- At time 7 ACK received at sender.
- At time 8 sender can transmit two segments, cwnd = 2 (we have round trip time = 8).
- At times 12, 13 ACK 2 and ACK 3 generated.
- At time 15, 16 ACK's RCVD and with cwnd = 4 sender transmits four segments.
- At time 24 and on can always transmit.
Figure 20.8  Example of slow start.
Figure 20.9  Times 0–15 for bulk data throughput example.
Figure 20.10 Time 16–31 for bulk data throughput example.
Bandwidth - Delay Product

In previous example sender needs to have 8 segments outstanding and unacknowledged for max throughput. Thus receivers advertised window must be at least that large so as not to limit throughput.

Capacity of pipe = bandwidth ( bits/ sec ) x round trip time ( sec ) ( also know as bandwidth-delay product)

Example:

What size should receivers advertised window be for a T1 Cross USA country phone line?

= 1, 544, 000 Bits / Sec x 0.060 Sec round trip time.
= 11,580 byte window

Congestion

The spacing of the ACKs will correspond to the bandwidth of the slowest links

[ Fig 20.13]
Figure 20.13  Congestion caused by a bigger pipe feeding a smaller pipe.
So what do mean by increase cwnd for each ACK received?

1) Stevens book says “Each time an ACK is received, the congestion window is increased by one segment”

2) RFC2001 Network Working Group W. Stevens Request for Comments: 2001 January 1997 “TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms” : Each time an ACK is received, the congestion window is increased by one segment………this provides an exponential growth, although it is not exactly exponential because the receiver may delay its ACKs, typically sending one ACK for every two segments that it receives.

3) RFC 2581 TCP Congestion Control APRIL 1999 says “During Slow start, TCP increments cwnd by at most mss bytes for each ACK received that acknowledges new data”


Delayed ACKs [RFC1122,RFC2581] allow a TCP receiver to refrain from sending an ACK for each incoming segment. However, a receiver SHOULD send an ACK for every second full-sized segment that arrives. Furthermore, a receiver MUST NOT withhold an ACK for more than [200] ms. By reducing the number of ACKs sent to the data originator the receiver is slowing the growth of the congestion window under an ACK counting system.
Conclusion: So clearly we are counting individual ACKS not the amount of data being ACKed. For this ECE4110 class always assume cwnd is incremented by only one MSS when an “ACK very other segment” is used and this ACK datagram contains an ACK for more than one segment.

Aside:

Is this definitely 100% for sure always true in the real world? Well ….

Forouzan in “TCP/IP Protocol Suite” McGraw Hill 2000 says “For each segment that is acknowledged, increase the congestion window by one maximum segment size until you reach a threshold of half the allowable window size.”

Lets stick with the ACK counting approach at the top of this slide.