TCP's Cumulative ACK mechanism

TCP delivers data reliably, in a byte-stream order - meaning that receiver application receives data in the same order in which the sender application wrote data to its TCP. TCP has to perform this job utilizing unreliable datagram services at the Network layer – IP is a connectionless, unreliable datagram service. A centerpiece of this data delivery mechanism is how acknowledgments (ACKs) are sent by the receiver in response to data received. *Note that an ACK is sent only in response to data or S or F flags; no ACK of an ACK is ever sent*.

We saw earlier that at the connection establishment stage, both TCPs exchange initial sequence numbers.

SYN,   Seq # = M

SYN, ACK, Seq # = N, Ack #=M+1

ACK,   Seq # = M+1, Ack #= N+1
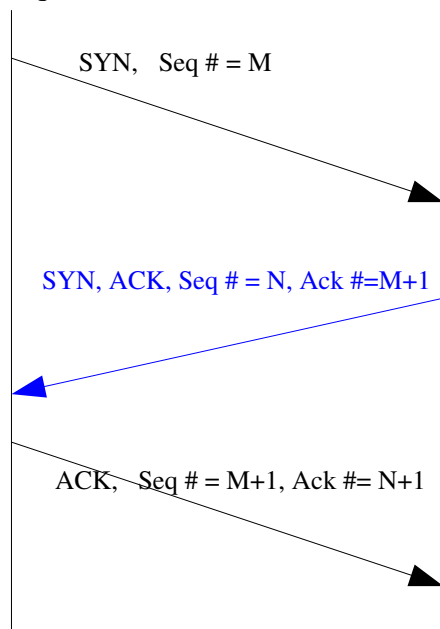
The TCP data stream from either side is numbered relative to these initial starting values. Whenever an Acknowledgment (ACK) is sent from either side, with an Acknowledgment number X, it is saying "*the next byte we expect from you is byte number X*". In other words, An Acknowledgment number of X means: "*I have received bytes up to byte number X-1, next send me byte number X*". If any other byte in the byte stream is received, TCP will discard those – only a TCP segment with sequence number X will be accepted as legitimate. This is the crux of the cumulative acknowledgment mechanism.

 Let us illustrate with an example:

A   B

Initial seq # sent by A is 123455

SYN,   Seq # = 123455

Initial seq. # by B is 321321. B is acknowledging receipt of S flag from A(which consumes one sequence number), by saying "*send me byte number 123456 next*".

SYN, ACK, Seq # = 321321, Ack #=123456

This has no data, but A is acknowledging receipt of B's S flag and ISN by saying, "*next thing I expect from you is byte # 321322*".

ACK,   Seq # = 123456, Ack #= 321322

A is sending 100 bytes of data. Usually data carrying segments have P flag set. The bytes are numbered 123456, 123457, ..., 123555. The length of the segment should tell B that 123555 is the last byte number in the segment.

A, P flags; Seq # = 123456, Ack #= 321322
100 bytes data

Acknowledgment number sent by B is 123556, which says that the next byte it expects from A is byte # 123556. This ACK informs A that all the bytes upto, but not including byte number 123556 has been delivered to B. This segment also contains 56 bytes of data from B to A. These data bytes are numbered 321322, 321323, ..., 321377

P,A flags; Seq # = 321322, Ack #=123556
56 bytes of data

A flag; Seq # = 123556, Ack #=321378
0 bytes of data

This is ACK by A of the 56 bytes from B – the next expected sequence number from B is 321378, which is A's ACK #. There is no data in this segment.

A, P flag; Seq # = 123556, Ack #=321378
120 bytes of data

This data segment carried bytes 123556, 123557,...,123675. The packet was lost in transit

Packet lost

This data segment carried bytes 123676, 123677,...,123775.

A, P flag; Seq # = 123676, Ack #=321378
100 bytes of data

Because B never received byte # 123556 from A, it totally disregards the newly arrived packet, and keeps saying "*send me byte 123556*". This is the key to the cumulative ACK mechanism – only consecutively accumulated bytes are ACK'd.

P,A flags; Seq # = 321378, Ack #=123556
56 bytes of data

At this point A knows that one of its segments carrying bits 123556 through 123675 was lost or delayed. A will have to retransmit this segment as well as the next segment of 100 bytes it sent.