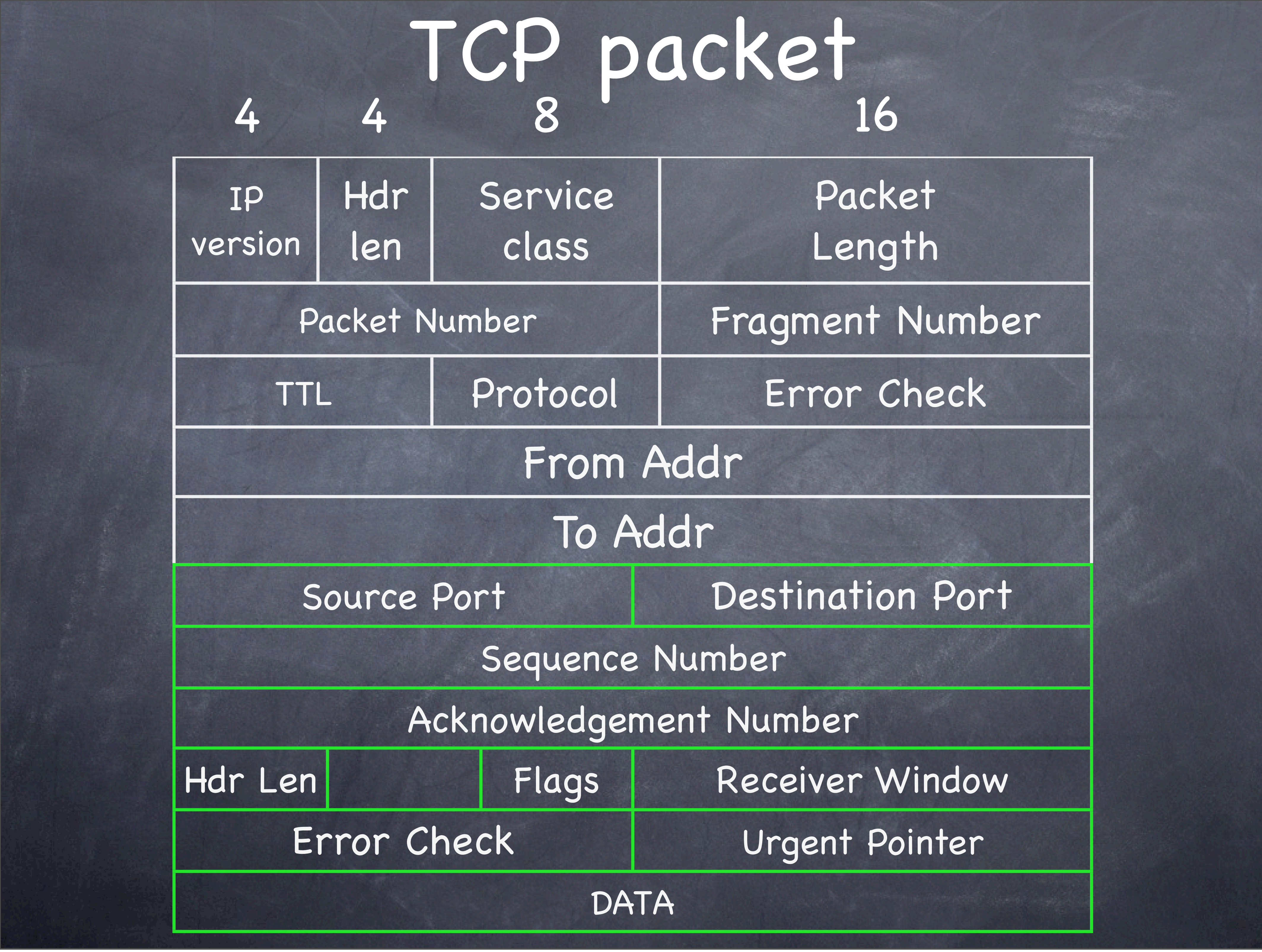
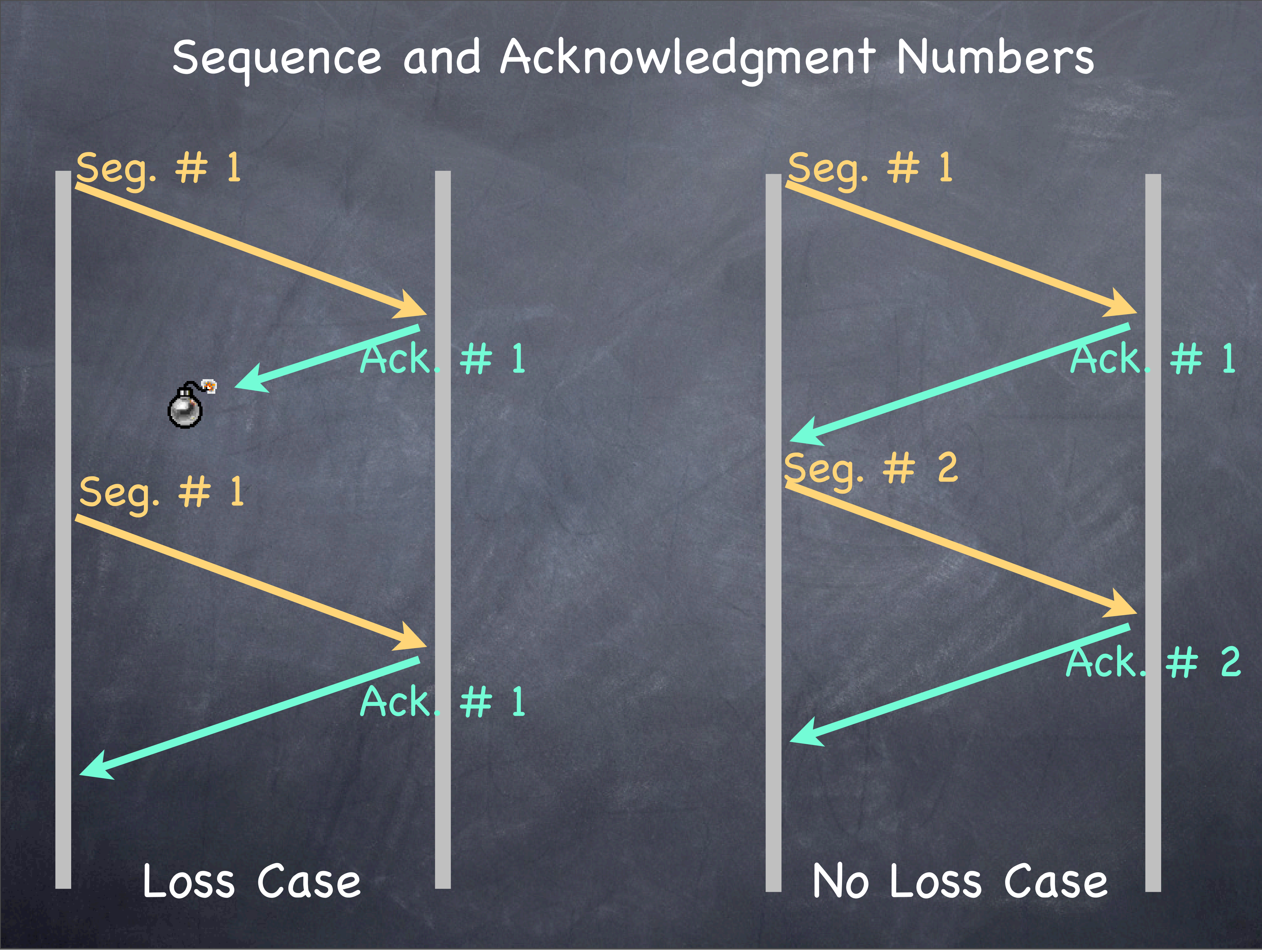
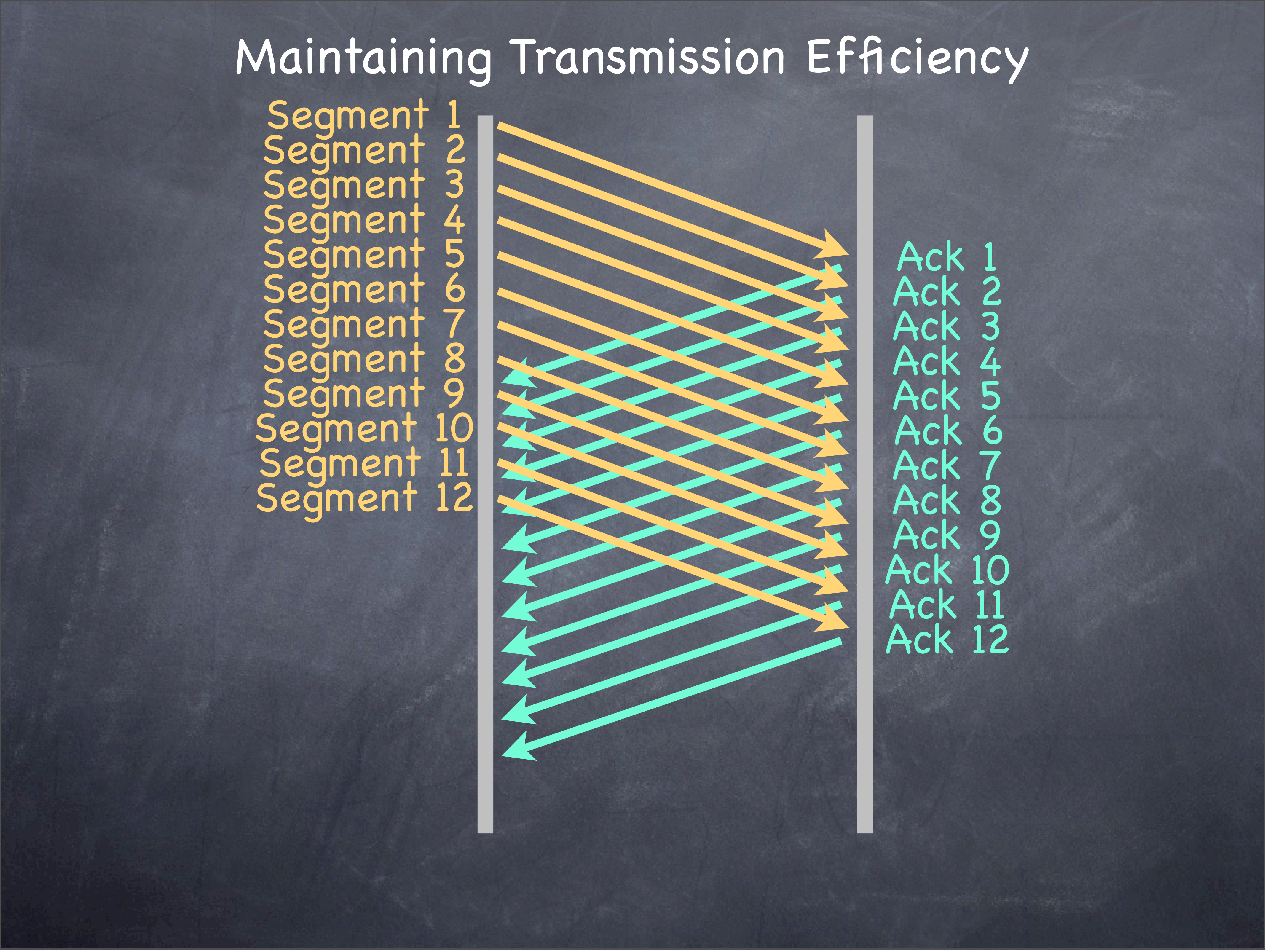


Recall the TCP packet format...

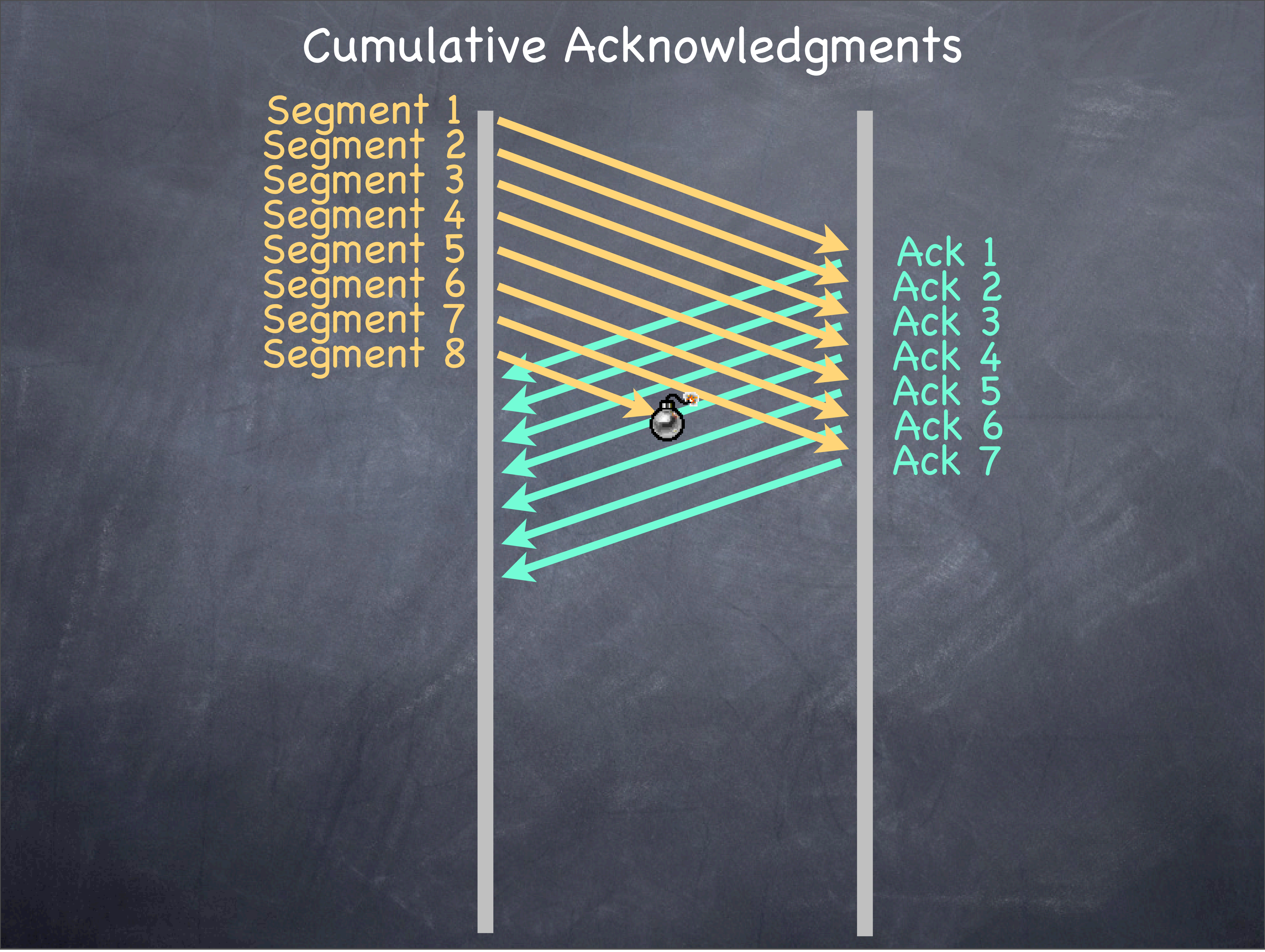


and the basic idea of using acknowledgment and retransmissions to ensure reliable delivery.

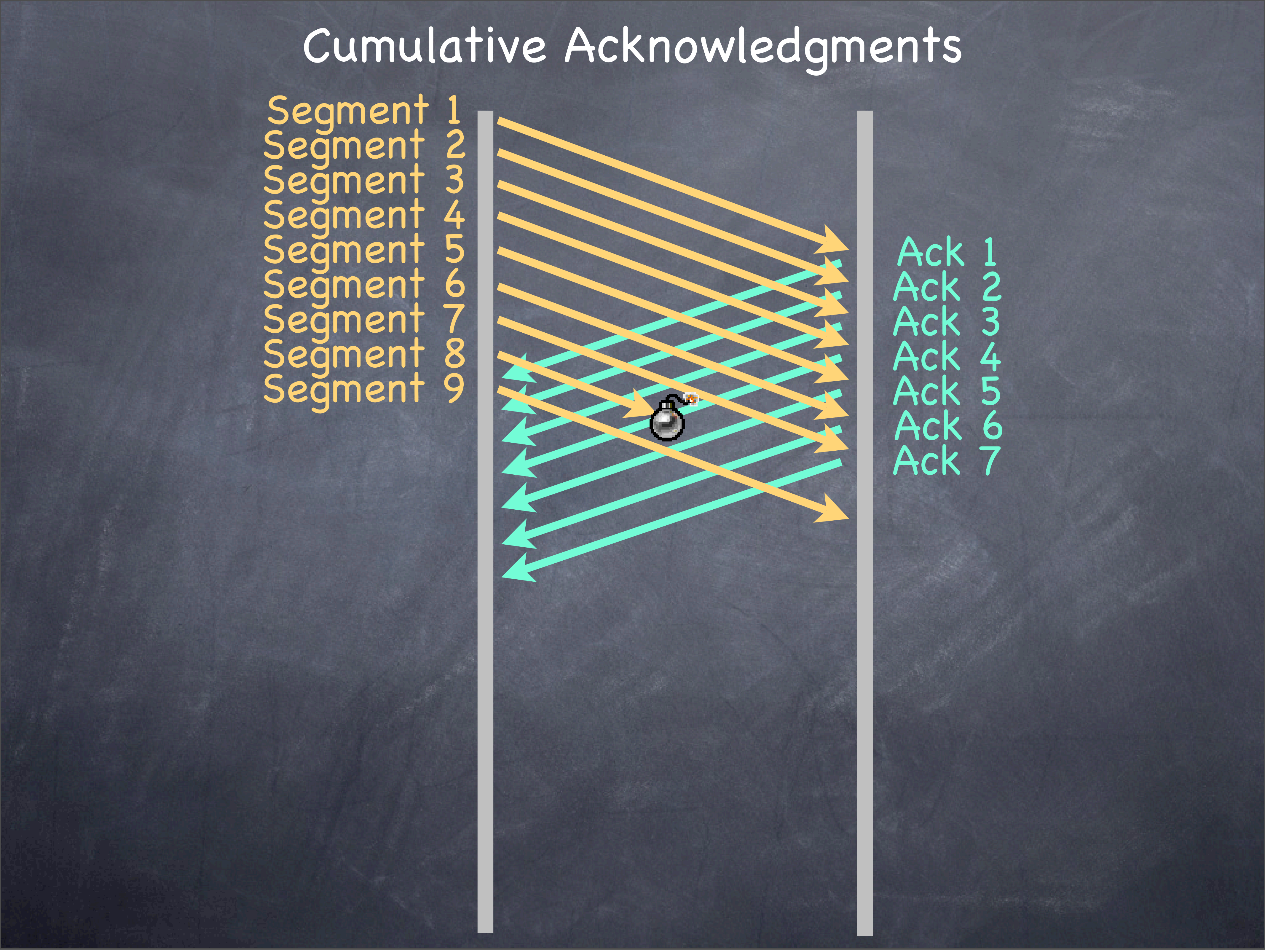


Since they knew segments would overlap on the wire, the designers of TCP did something tricky.

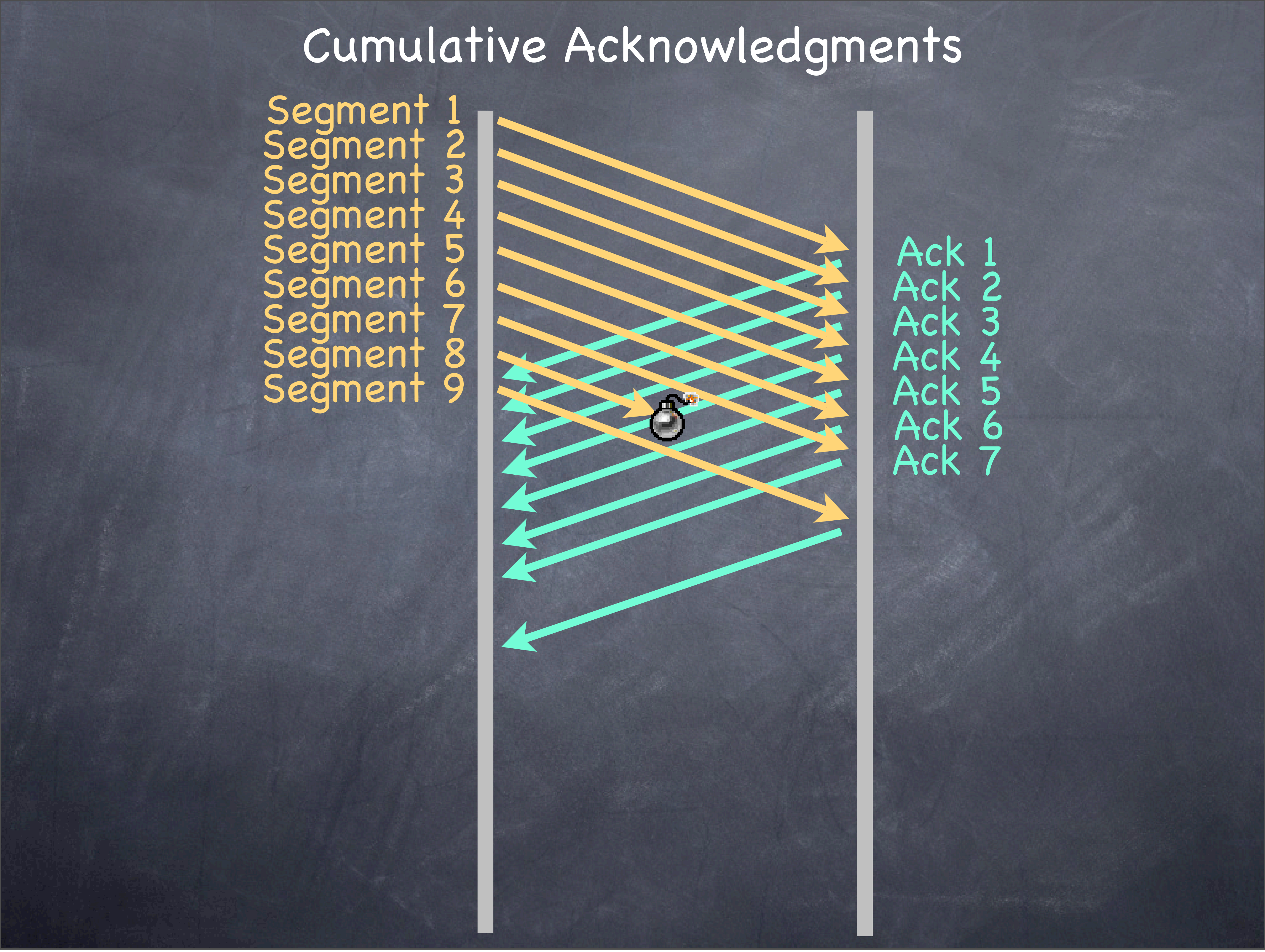
Acknowledgments are cumulative. Whenever a packet arrives, the receiver sends an acknowledgment containing the number of the last packet (actally byte) that was received "in order". That is, if the sender sees Ack 12, it knows that segments 1-12 all got through Ok.



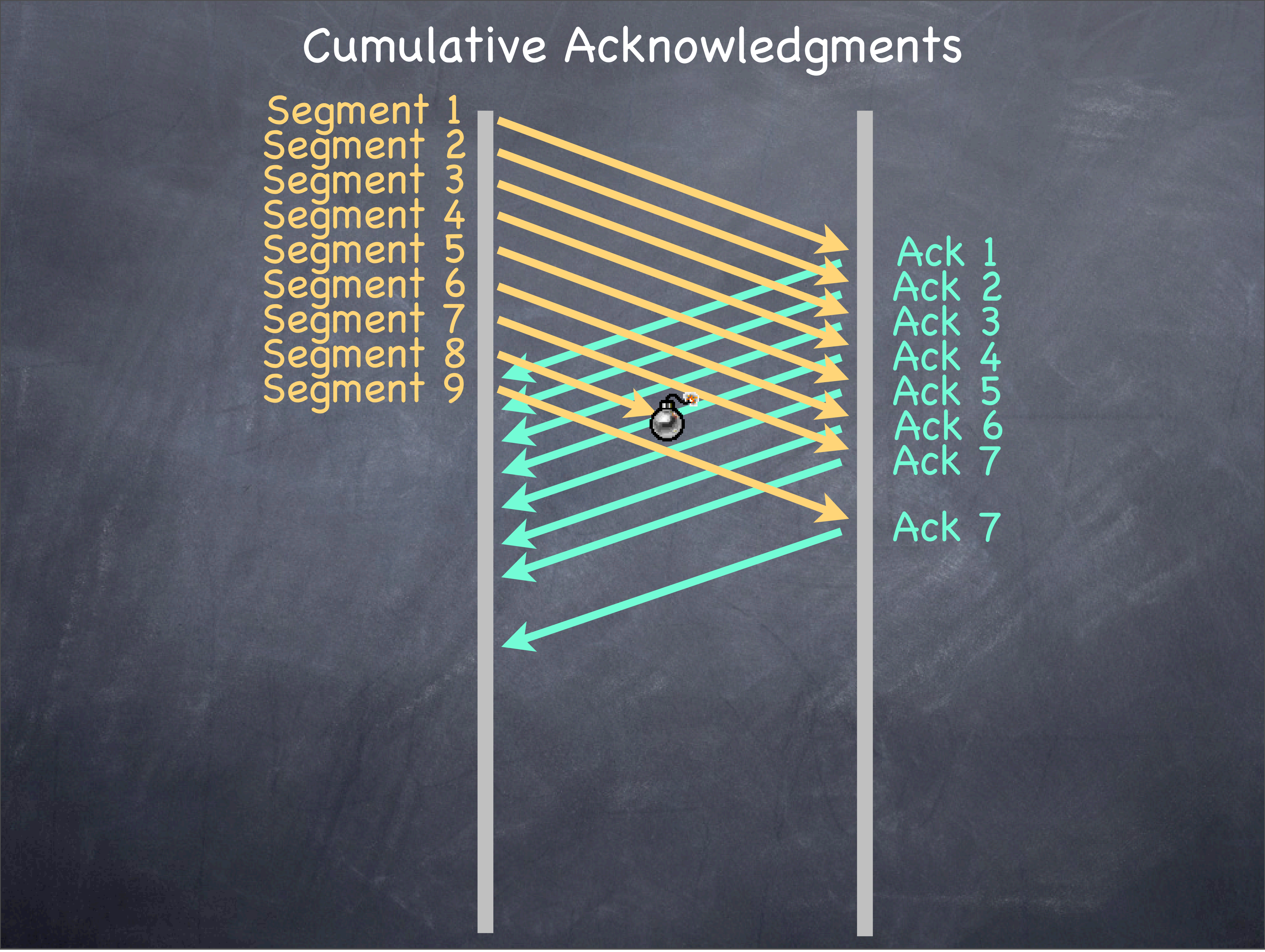
This means that if one packet of a sequence gets lost in transit, the receiver will respond to each packet that arrives after the lost packet by acknowledging the last message received before the loss. It also means that the sender may not receive an acknowledgment for every packet sent. When a lost packet is retransmitted, completing a sequence of packets, the receiver can acknowledge the receipt of the entire sequence by sending an acknowledgment for the last packet.



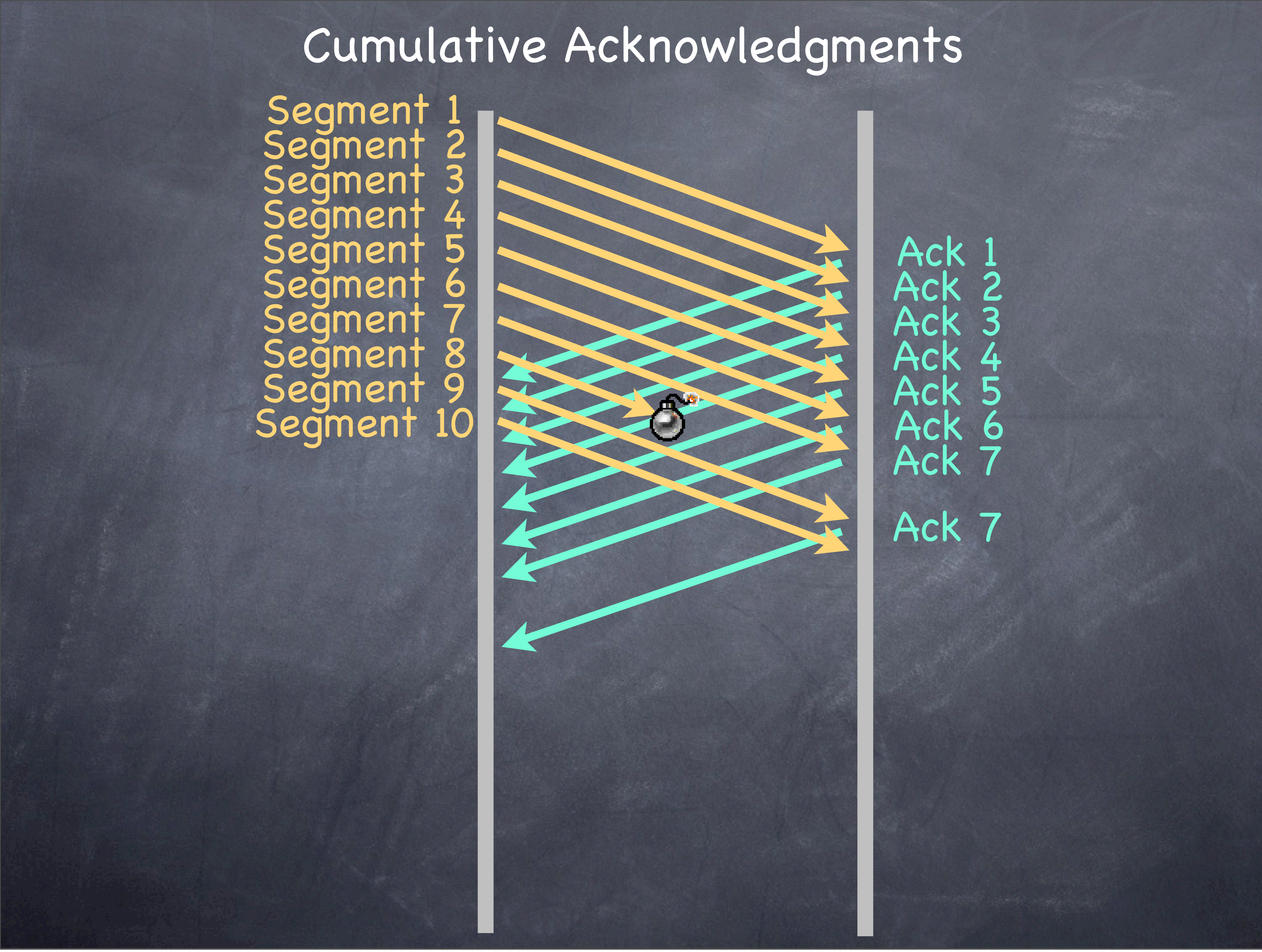
This means that if one packet of a sequence gets lost in transit, the receiver will respond to each packet that arrives after the lost packet by acknowledging the last message received before the loss. It also means that the sender may not receive an acknowledgment for every packet sent. When a lost packet is retransmitted, completing a sequence of packets, the receiver can acknowledge the receipt of the entire sequence by sending an acknowledgment for the last packet.



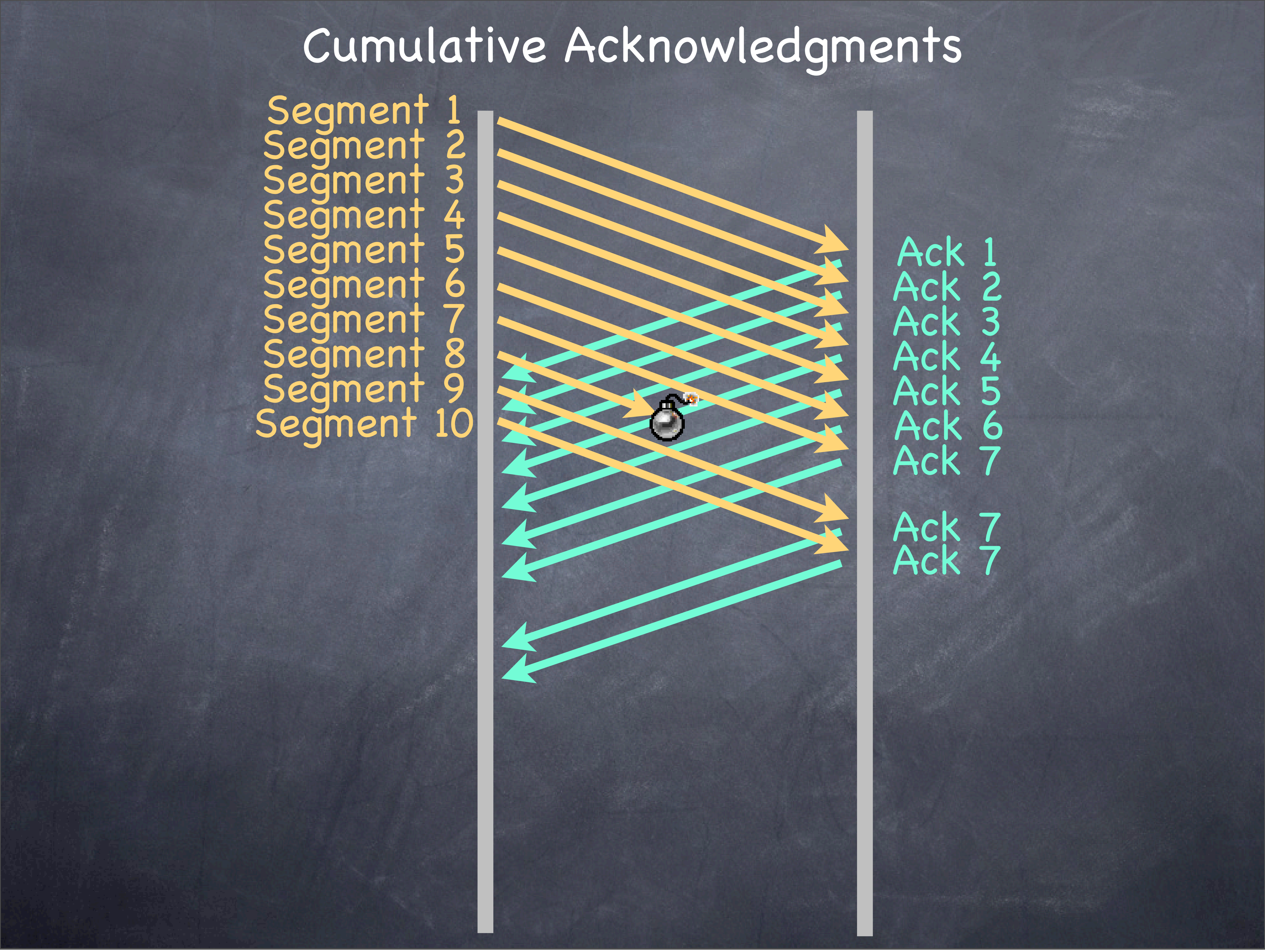
This means that if one packet of a sequence gets lost in transit, the receiver will respond to each packet that arrives after the lost packet by acknowledging the last message received before the loss. It also means that the sender may not receive an acknowledgment for every packet sent. When a lost packet is retransmitted, completing a sequence of packets, the receiver can acknowledge the receipt of the entire sequence by sending an acknowledgment for the last packet.



This means that if one packet of a sequence gets lost in transit, the receiver will respond to each packet that arrives after the lost packet by acknowledging the last message received before the loss. It also means that the sender may not receive an acknowledgment for every packet sent. When a lost packet is retransmitted, completing a sequence of packets, the receiver can acknowledge the receipt of the entire sequence by sending an acknowledgment for the last packet.

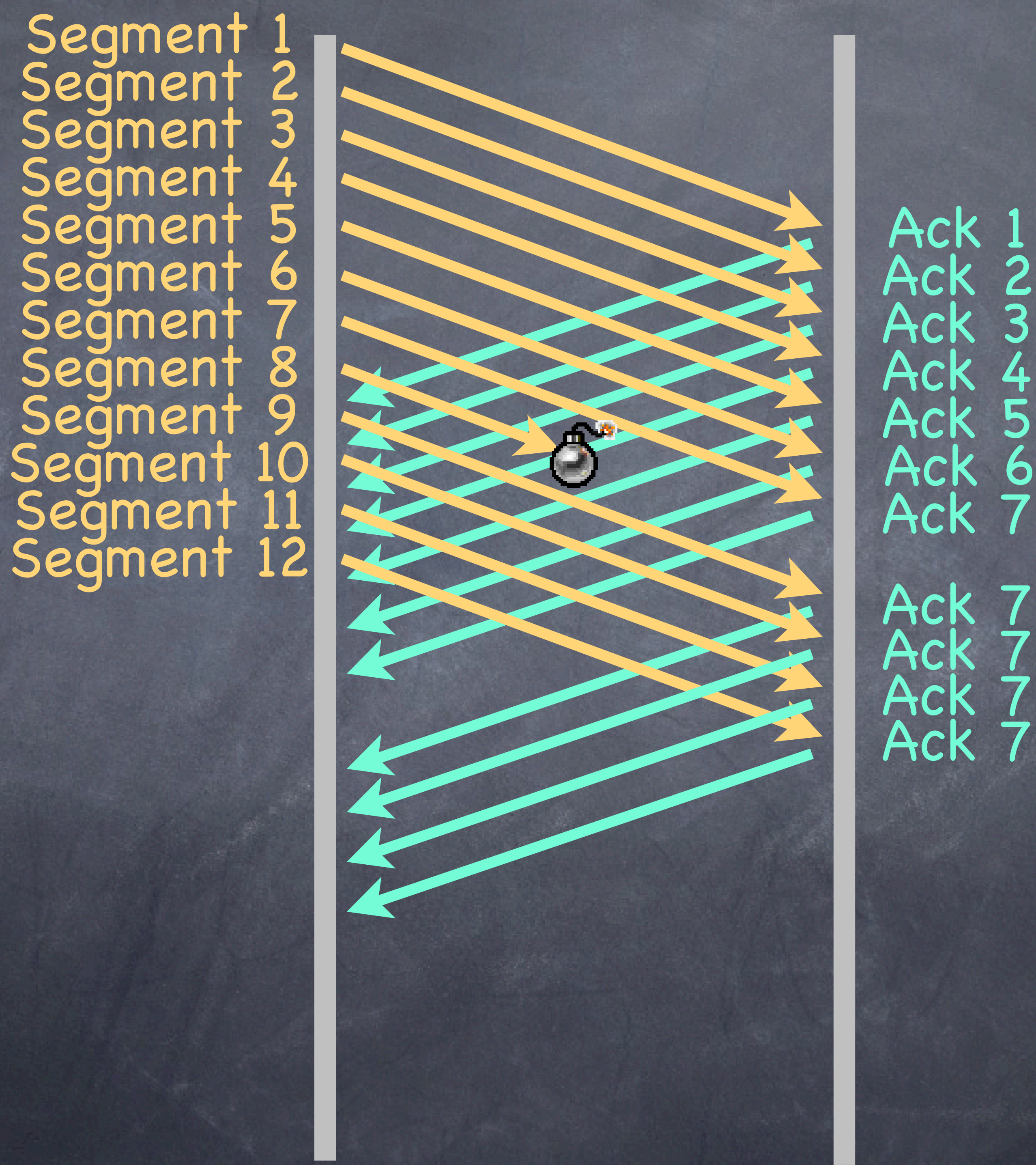


This means that if one packet of a sequence gets lost in transit, the receiver will respond to each packet that arrives after the lost packet by acknowledging the last message received before the loss. It also means that the sender may not receive an acknowledgment for every packet sent. When a lost packet is retransmitted, completing a sequence of packets, the receiver can acknowledge the receipt of the entire sequence by sending an acknowledgment for the last packet.

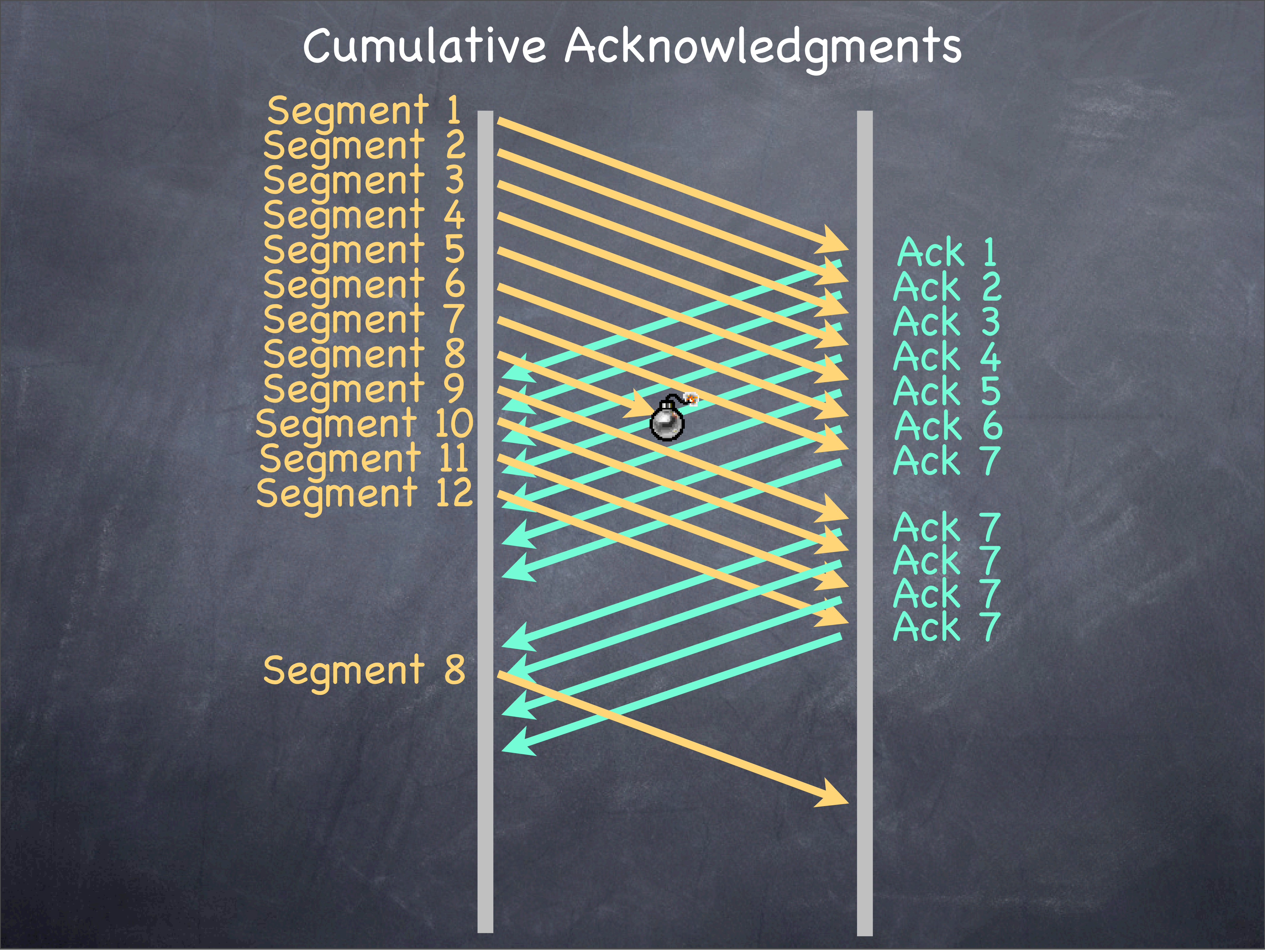


This means that if one packet of a sequence gets lost in transit, the receiver will respond to each packet that arrives after the lost packet by acknowledging the last message received before the loss. It also means that the sender may not receive an acknowledgment for every packet sent. When a lost packet is retransmitted, completing a sequence of packets, the receiver can acknowledge the receipt of the entire sequence by sending an acknowledgment for the last packet.

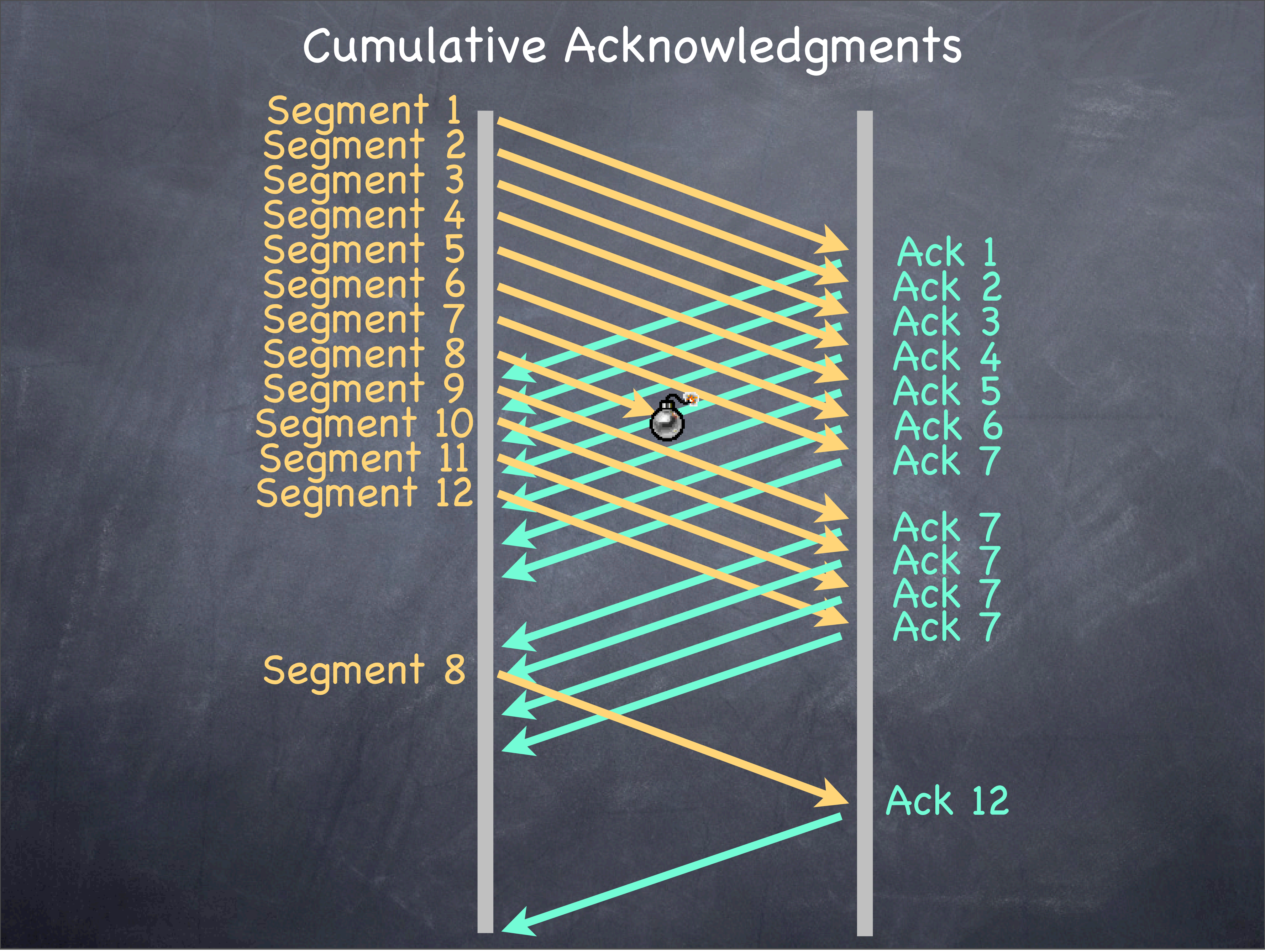
Cumulative Acknowledgments



This means that if one packet of a sequence gets lost in transit, the receiver will respond to each packet that arrives after the lost packet by acknowledging the last message received before the loss. It also means that the sender may not receive an acknowledgment for every packet sent. When a lost packet is retransmitted, completing a sequence of packets, the receiver can acknowledge the receipt of the entire sequence by sending an acknowledgment for the last packet.

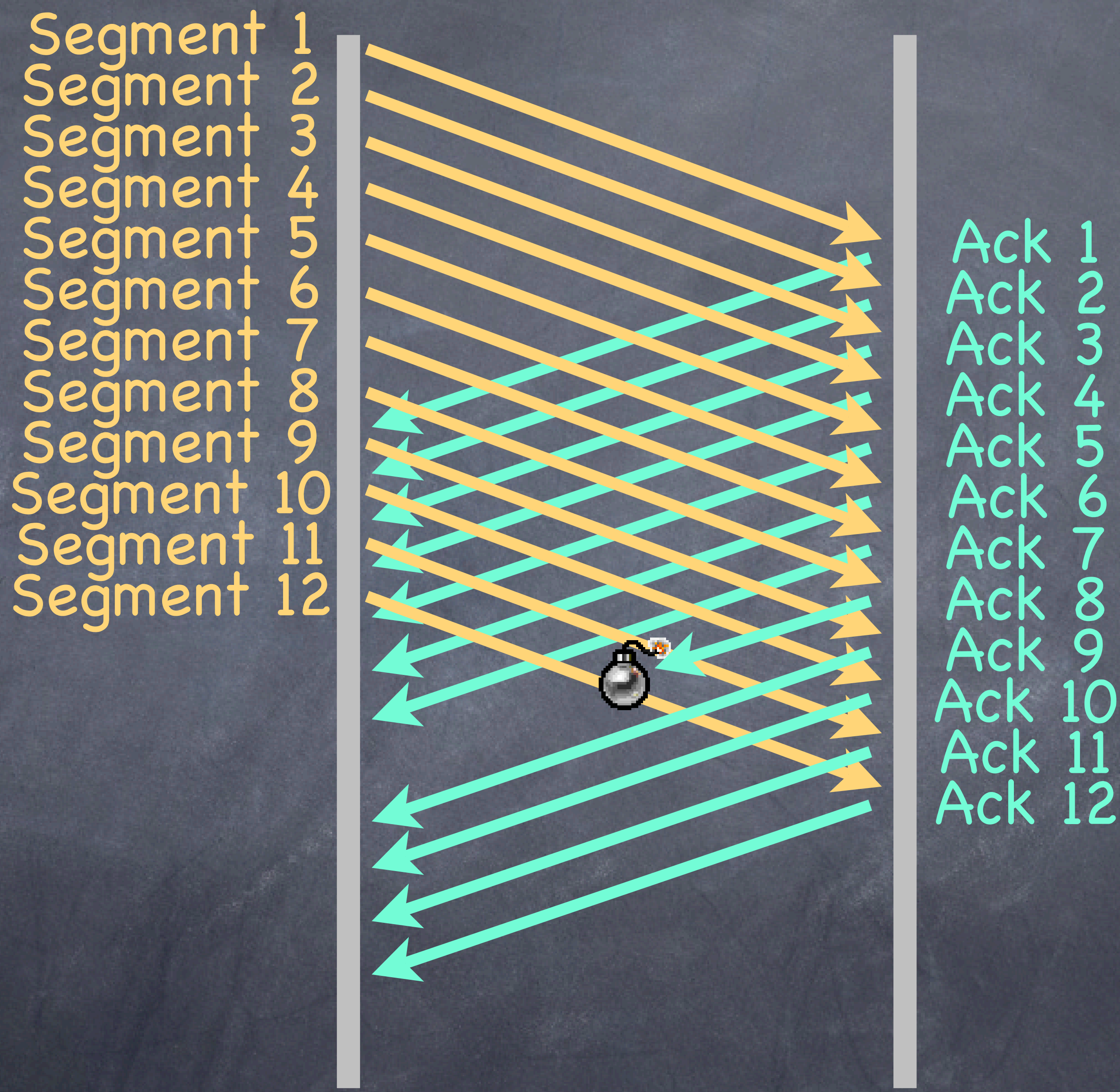


This means that if one packet of a sequence gets lost in transit, the receiver will respond to each packet that arrives after the lost packet by acknowledging the last message received before the loss. It also means that the sender may not receive an acknowledgment for every packet sent. When a lost packet is retransmitted, completing a sequence of packets, the receiver can acknowledge the receipt of the entire sequence by sending an acknowledgment for the last packet.



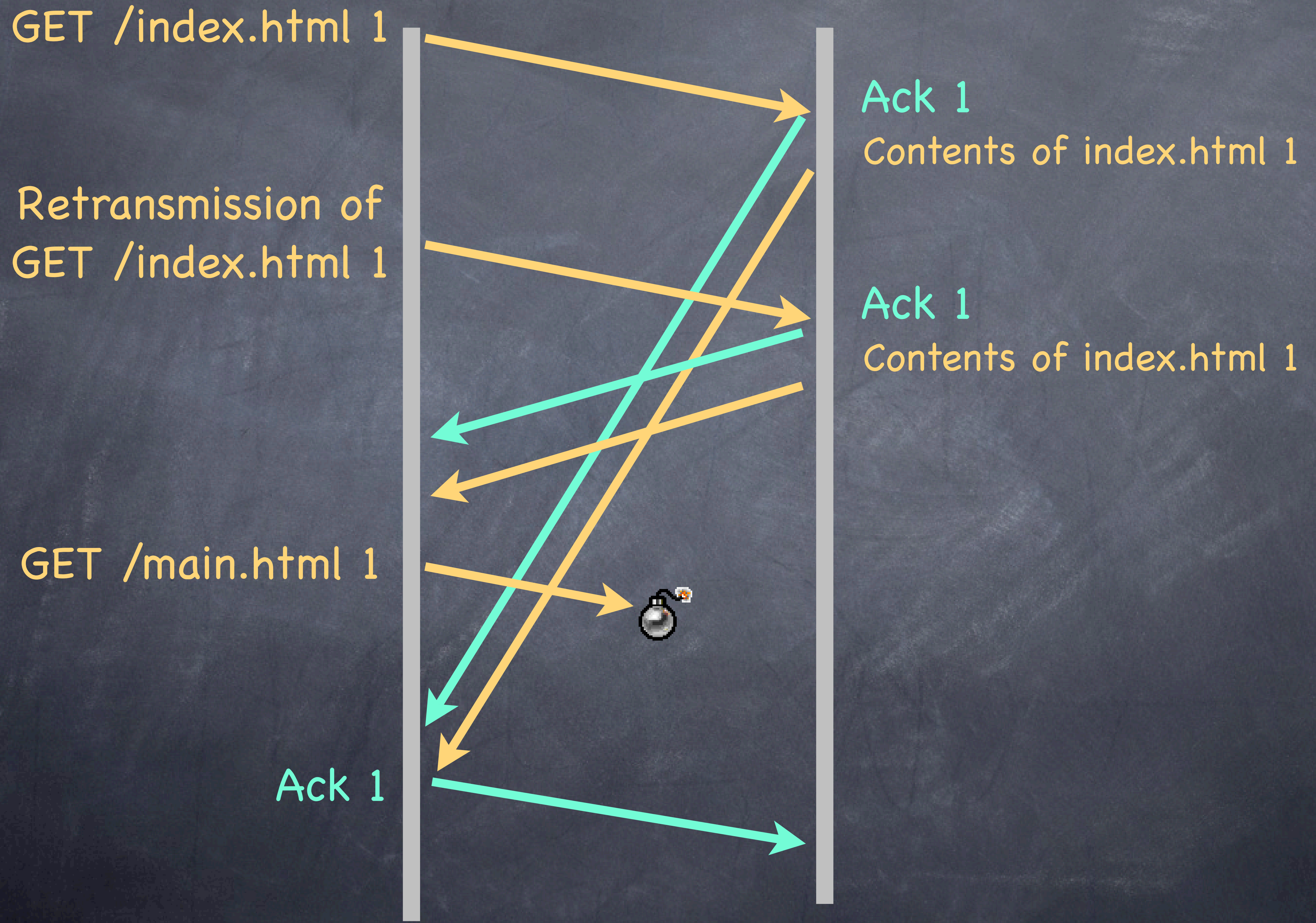
This means that if one packet of a sequence gets lost in transit, the receiver will respond to each packet that arrives after the lost packet by acknowledging the last message received before the loss. It also means that the sender may not receive an acknowledgment for every packet sent. When a lost packet is retransmitted, completing a sequence of packets, the receiver can acknowledge the receipt of the entire sequence by sending an acknowledgment for the last packet.

Cumulative Acknowledgments



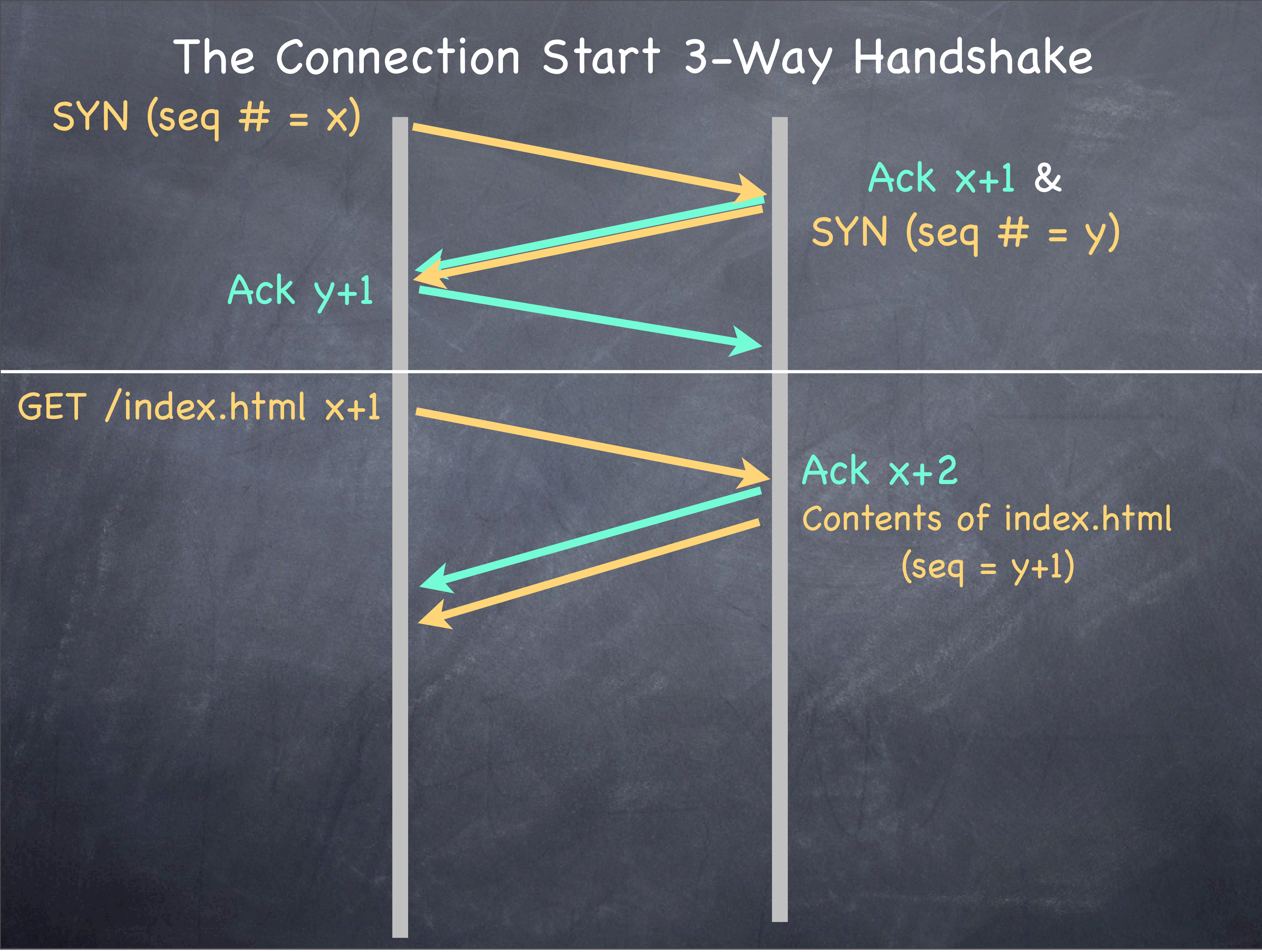
Cumulative acknowledgments also makes it unnecessary to retransmit lost acknowledgments in some cases. As long as the acknowledgment that is lost is not for the last packet in a sequence, the cummulative acknowledgment sent for a later packet will assure the sender that the earlier packet was received.

Slow Segments == Confused Conversations



If every conversation numbers packets starting with #1, it would be easy for packets from two different connections to become confused.

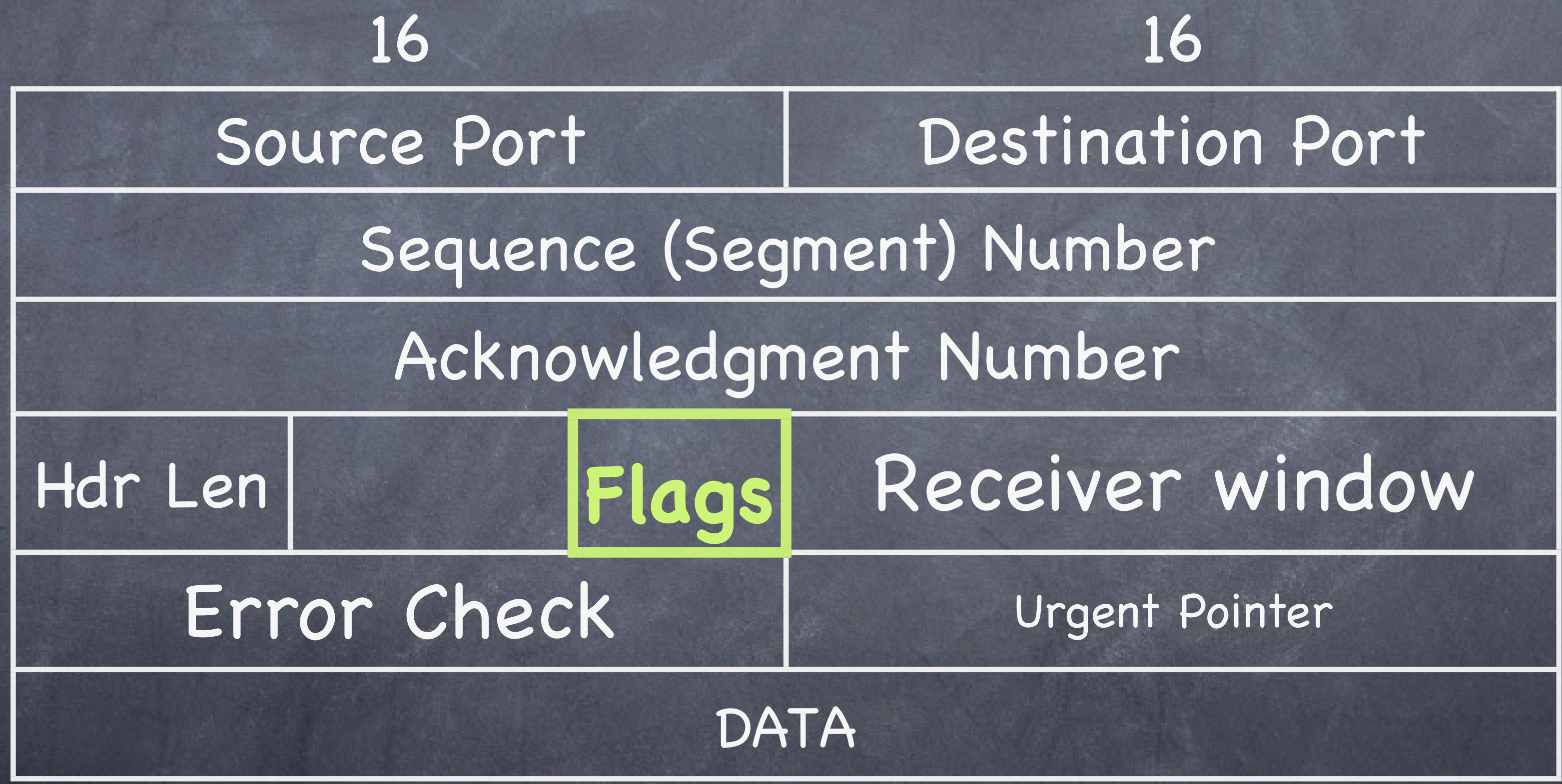
On the other hand, if every connection started with a random sequence number, how would one know that there had not been an earlier packet that was missed?



To avoid having packets from two different connections look the same (i.e., have the same sequence numbers), choose initial sequence numbers RANDOMLY and send special “start conversation” segments to tell the other side of the choice.

In an ongoing TCP connection, one cannot really distinguish the sender from the receiver. Both sides are allowed to send and receive data. Both side need to associate sequence numbers with the packets they send so that the other side can send appropriate acknowledgments. Therefore, both sides pick a random inital sequence number, both sides send this number in a special packet called a SYN, and both sides send acknowledments of the receipt

TCP Segment Format



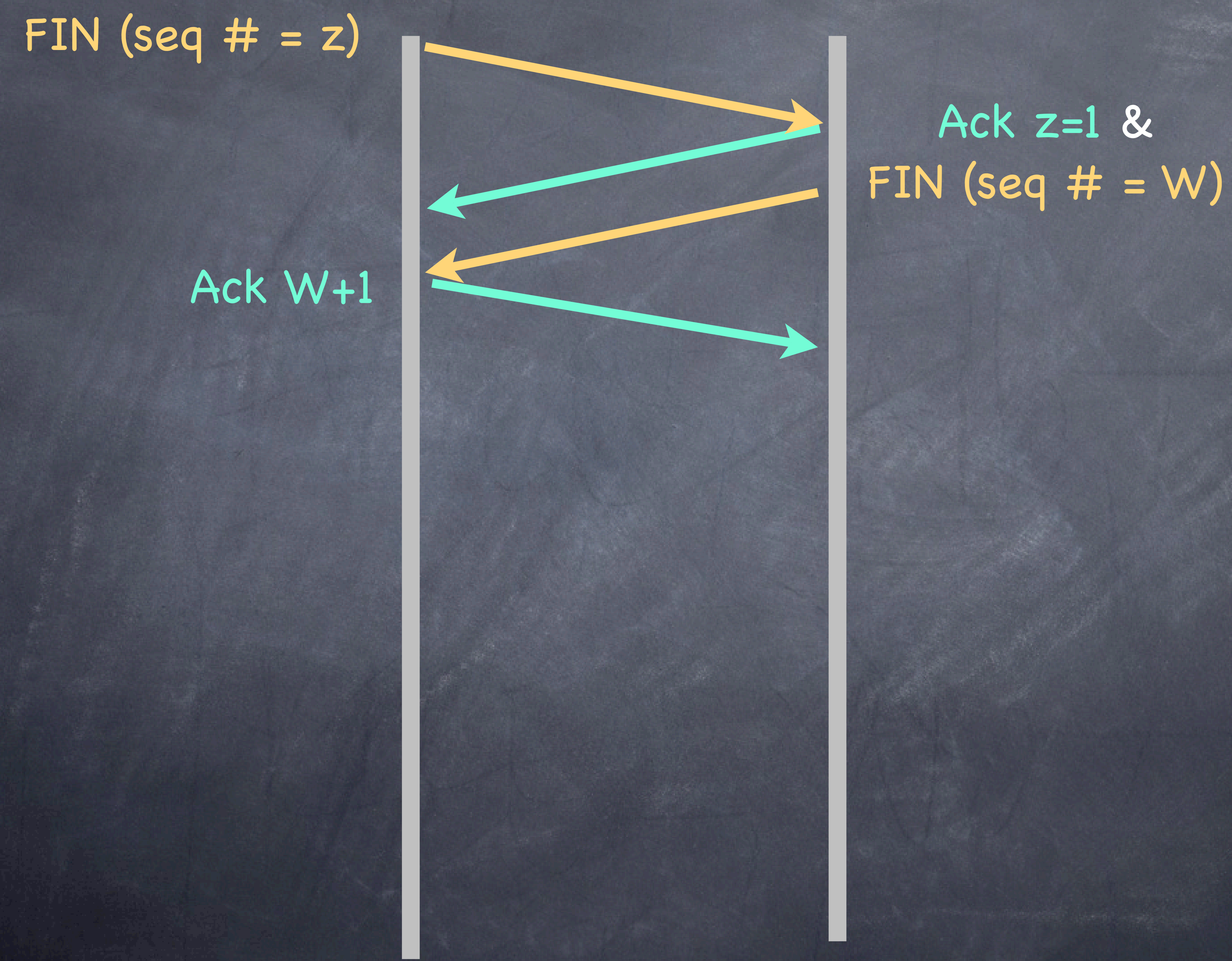
The Flags field of the TCP packet is used to distinguish special packets like the starting SYN packets.

Flags

(nothing)	Here's some data
SYN	Connect
FIN	Disconnect
ACK	I got segment #__
RESET	I'm confused and want to hang up
URG	By the way, ...
PUSH	Send data now

The Flags field is also used to indicate a) whether a packet contains an acknowledgment (they almost all do), b) whether the sender would like to terminate the connection normally (FIN), c) whether the sender would like to terminate the connection abnormally (RESET).

Connection Termination



Closing is complicated because both sides are allowed to send data. Closing one side means that that side has nothing more to send...but there might still be more data to receive! Therefore, each side has to send its own FIN packet (and acknowledge receipt of a FIN from the other side).

Let's use TCPCapture to actually see what happens at the application level and at the TCP level when two programs try to communicate using HTTP