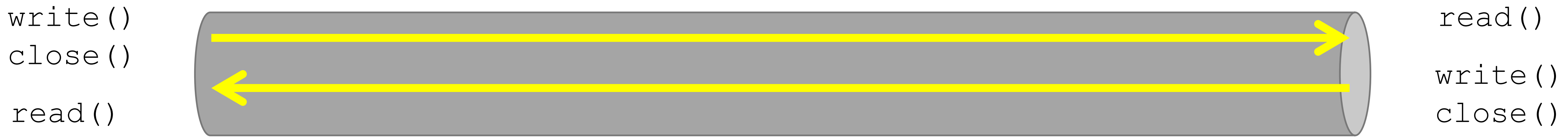
Death of a Stream

Not Controversial (I Hope): Good Case

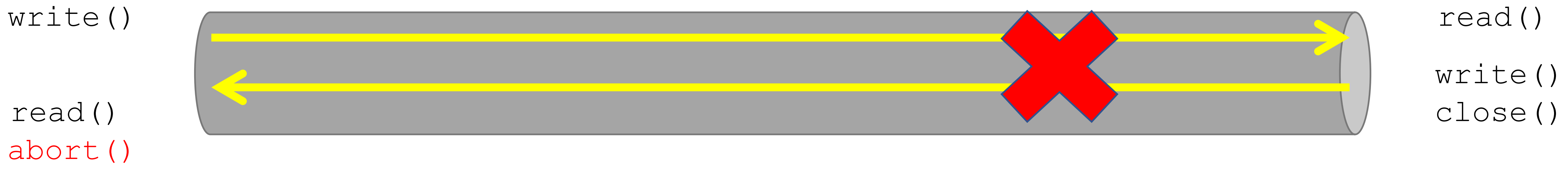


- Stream contains two channels in opposite direction
  - Each side writes data
    - STREAM frames
  - …which gets read on the other side
    - MAX_STREAM_DATA
  - …and eventually reaches an orderly end
    - FIN flag on last STREAM frame

Abrupt Closure

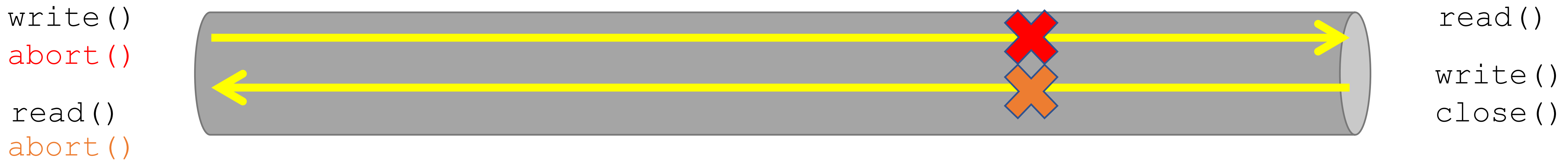
RST_STREAM, STOP_SENDING, and all things not transferred to completion

Stream Abort, <= -04



- RST_STREAM has three effects:
  - Announces that no new data will be sent nor old data retransmitted
    - Includes final offset to sync flow control
  - Announces that no new data will be read
  - Solicits matching RST_STREAM
    - Includes final offset to sync flow control

Stream Abort, >= -05



- RST_STREAM announces that no new data will be sent nor old data retransmitted
  - Includes final offset to sync flow control
- STOP_SENDING announces that no new data will be read
  - Solicits matching RST_STREAM
    - …which includes final offset to sync flow control

Various people unhappy here

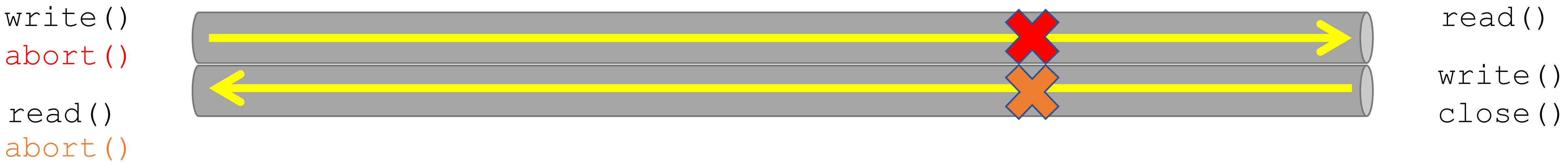
Liked Bidirectional Resets

- Bidirectional reset is a common pattern
  - Why optimize for the uncommon case?
  - Old drafts special-cased NO_ERROR for rare single-direction close
- Half-reset state feels messy
  - Shades of half-open TCP connections

Want Stop Sending in Application

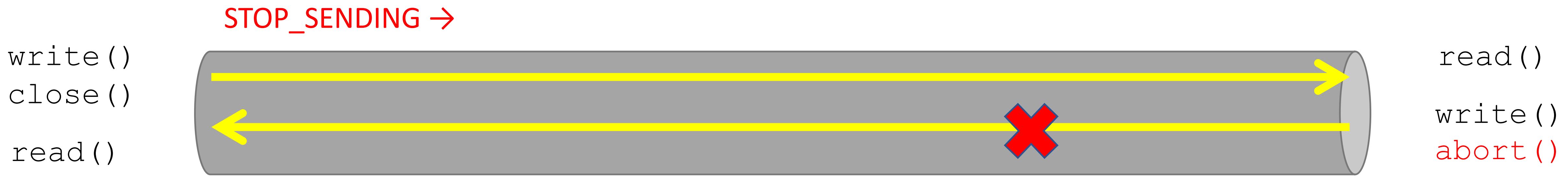
- HTTP is the only known use-case
- Only exception to "transport shouldn't be resetting streams"
  - #758, #485
  - …other than connection termination
- Only application knows which streams can't be reset safely

Toward a Unidirectional World



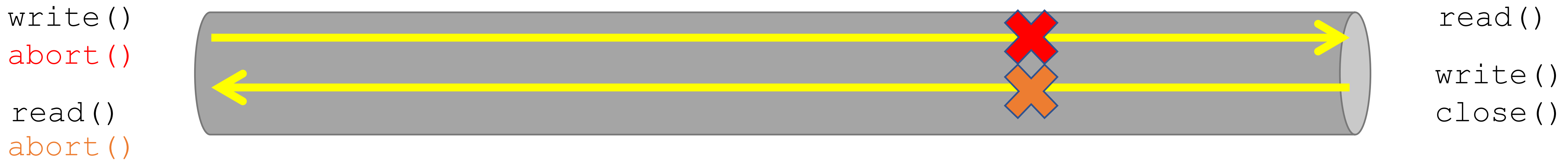
- RST_STREAM announces that no new data will be sent nor old data retransmitted
  - Includes final offset to sync flow control
- STOP_SENDING announces that no new data will be read
  - Solicits matching RST_STREAM
    - …which includes final offset to sync flow control

Transport-Clean Streams (#758,#485)



- RST_STREAM cancels the stream in one direction
  - But only when the application requests it!
- Application can define how to request closure if needed
- Possible risk: Deadlock
  - Receiver application no longer cares, stops reading
  - Transport stack stops updating flow control
  - Sender gets blocked on flow control
- Delivery of application-layer signal needs to be reliable (i.e. different stream)

Some Options

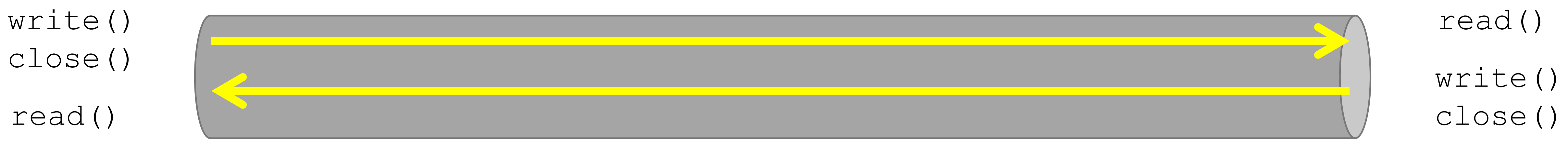


- Should we rename them to CANCEL_WRITE and CANCEL_READ?
  - Might be clearer than a unidirectional RST
- Should there be a CANCEL_BOTH?
  - Addresses the common case in a single frame
  - More complicated in unidirectional?

Stream Closure and Reliability

When is “closed” not “closed”? (#743)

Remember the Good Case?



- Stream contains two channels in opposite direction
  - Each side writes data
    - STREAM frames
  - …which gets read on the other side
    - MAX_STREAM_DATA
  - …and eventually reaches an orderly end
    - FIN flag on last STREAM frame
- Finally, the stream is closed

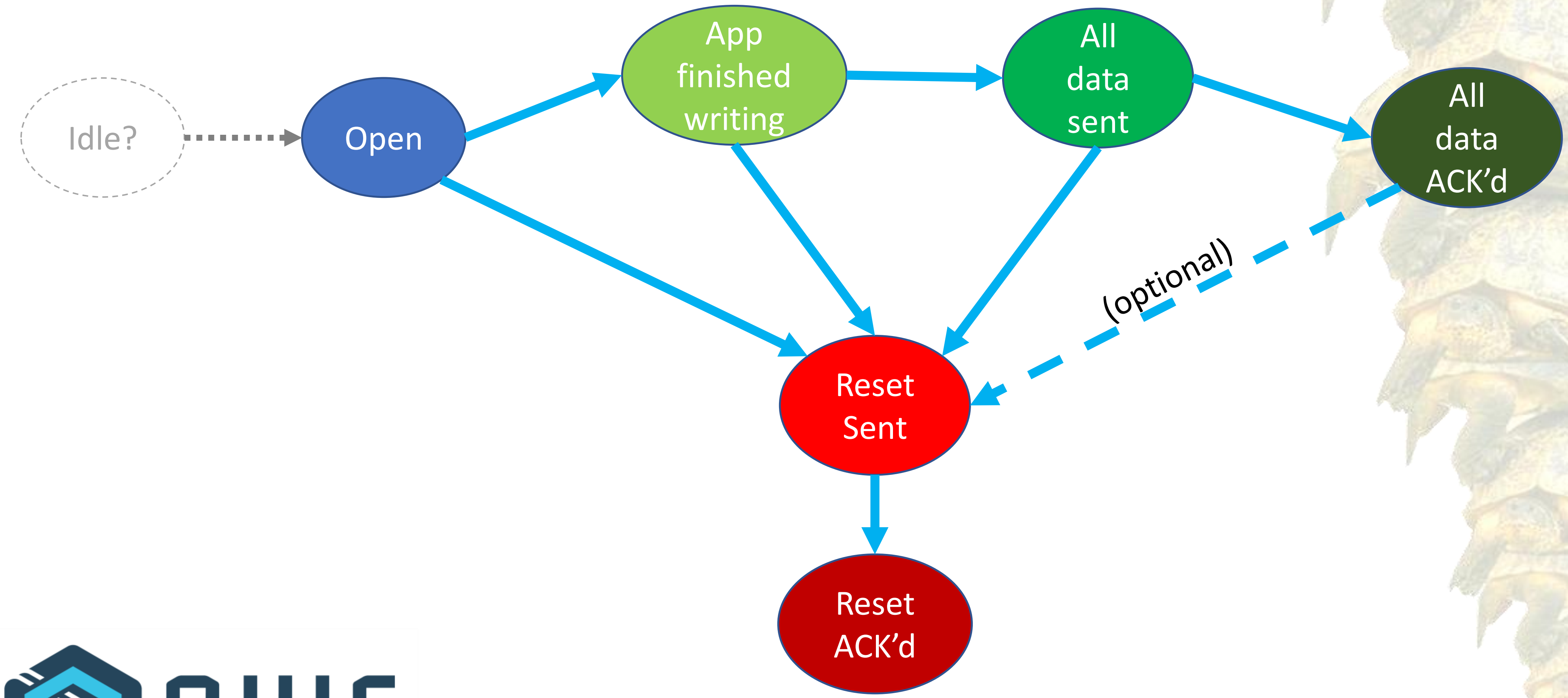
…or is it?

“all data” ⬄ “RST_STREAM”

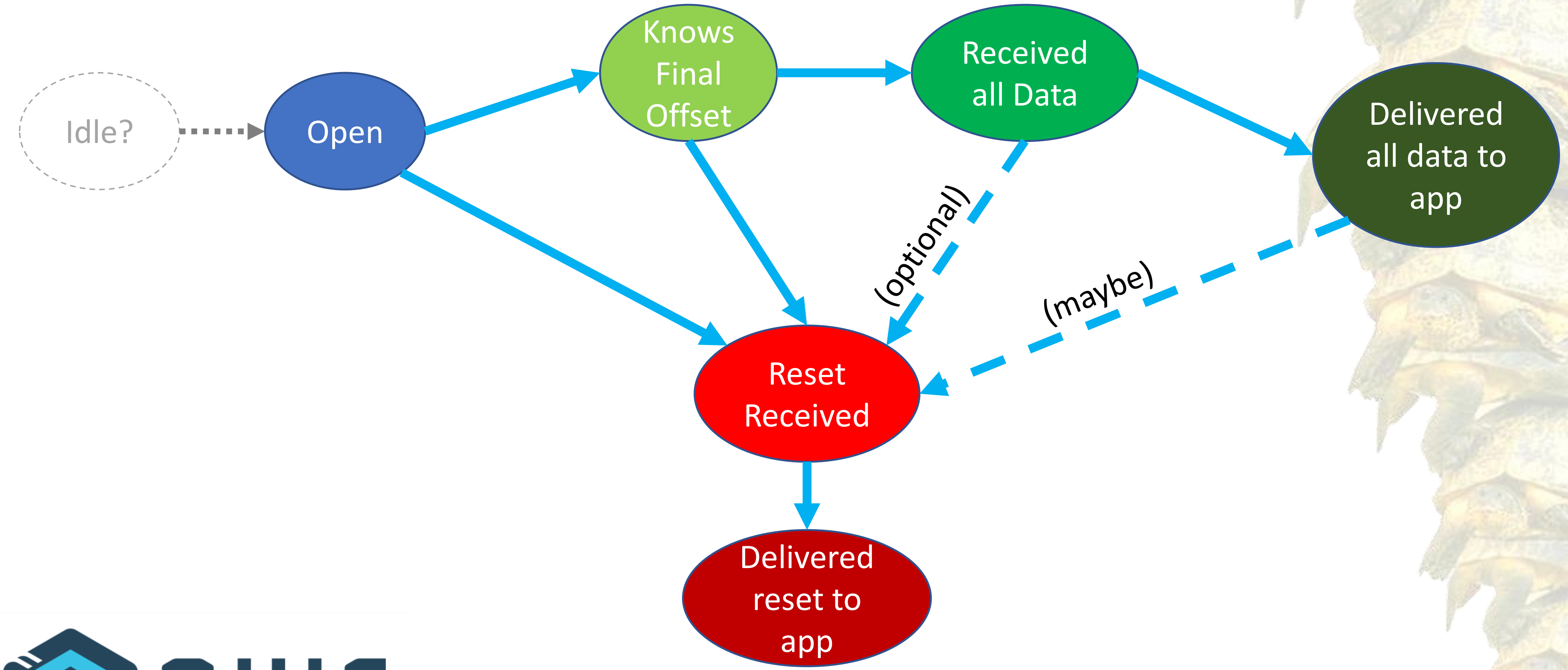
The stream is “closed” when…

- Application has delivered all data to sending transport
- Sending transport has sent packets containing all data
- Receiving transport has received packets containing all data
- Receiving application has read all data from the receiving transport
- Receiving application has generated ACKs for packets containing all data
- Sending transport has received ACKs for packets containing all data
- Receiver knows that sender knows all data has been delivered
- Sender knows that receiver knows that sender knows all data has been delivered
- Receiver knows that sender knows that receiver knows that sender knows all data has been delivered

Sender State Machine



Receiver State Machine



$$\begin{tikzcd}
\text{Idle?} \arrow[r, dotted] & \text{Open} \arrow[r] \arrow[dr] & \text{Knows Final Offset} \arrow[r] \arrow[d] & \text{Received all Data} \arrow[r] \arrow[dl, dashed, "\text{(optional)}"'] & \text{Delivered all data to app} \arrow[dll, dashed, "\text{(maybe)}"] \\
 & & \text{Reset Received} \arrow[d] & & \\
 & & \text{Delivered reset to app} & &
\end{tikzcd}$$