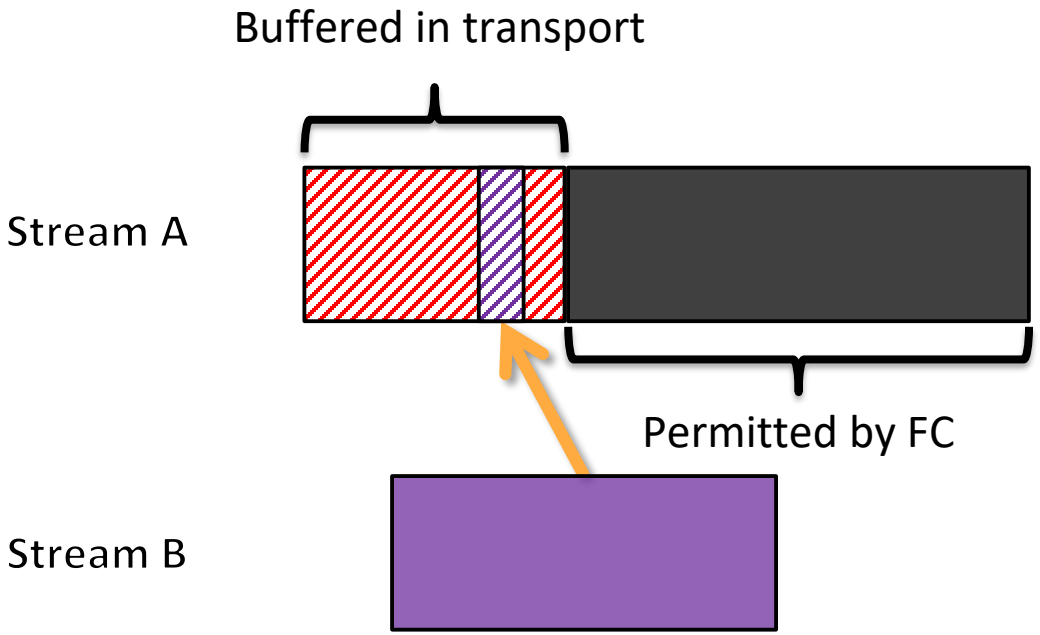
Flow Control Gotchas

QUIC Interim 1809

Flow Control is great!

…so long as all the streams and all the bytes are independent.

How to Deadlock With Two Streams

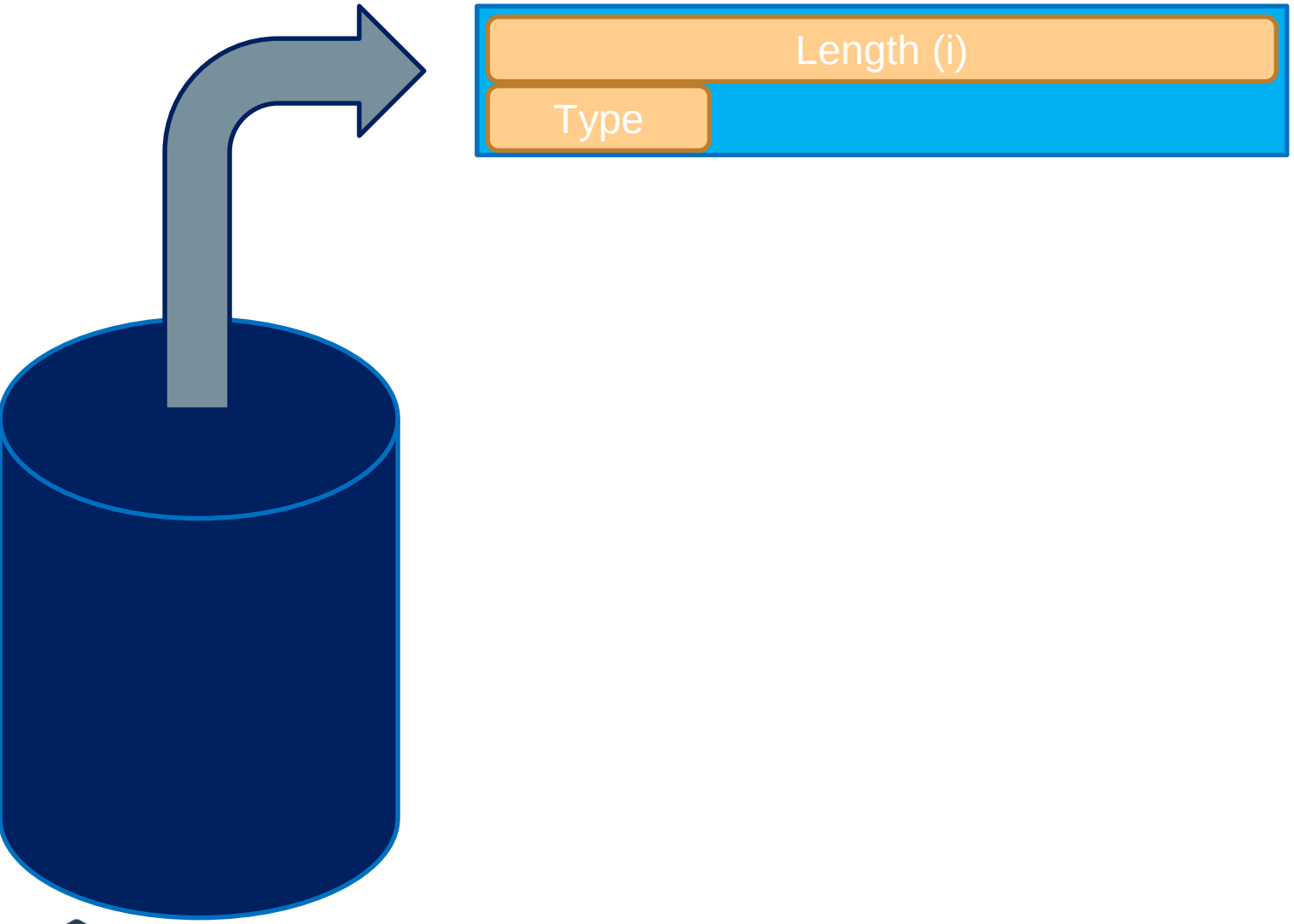


- Interpretation of Stream B depends on data from Stream A
- Flow control prevents data on Stream A from being sent
- Lack of progress on Stream B prevents new flow control credit from being issued to Stream A

Cross-Stream Dependencies and Header Compression

- QCRAM had an encoder stream for recovery only
  - Dependency on subsequently transmitted data
- QPACK uses encoder stream for all updates
  - Dependency on previously transmitted data
  - …unless flow control blocks sending in the first place

How to Deadlock with One Stream

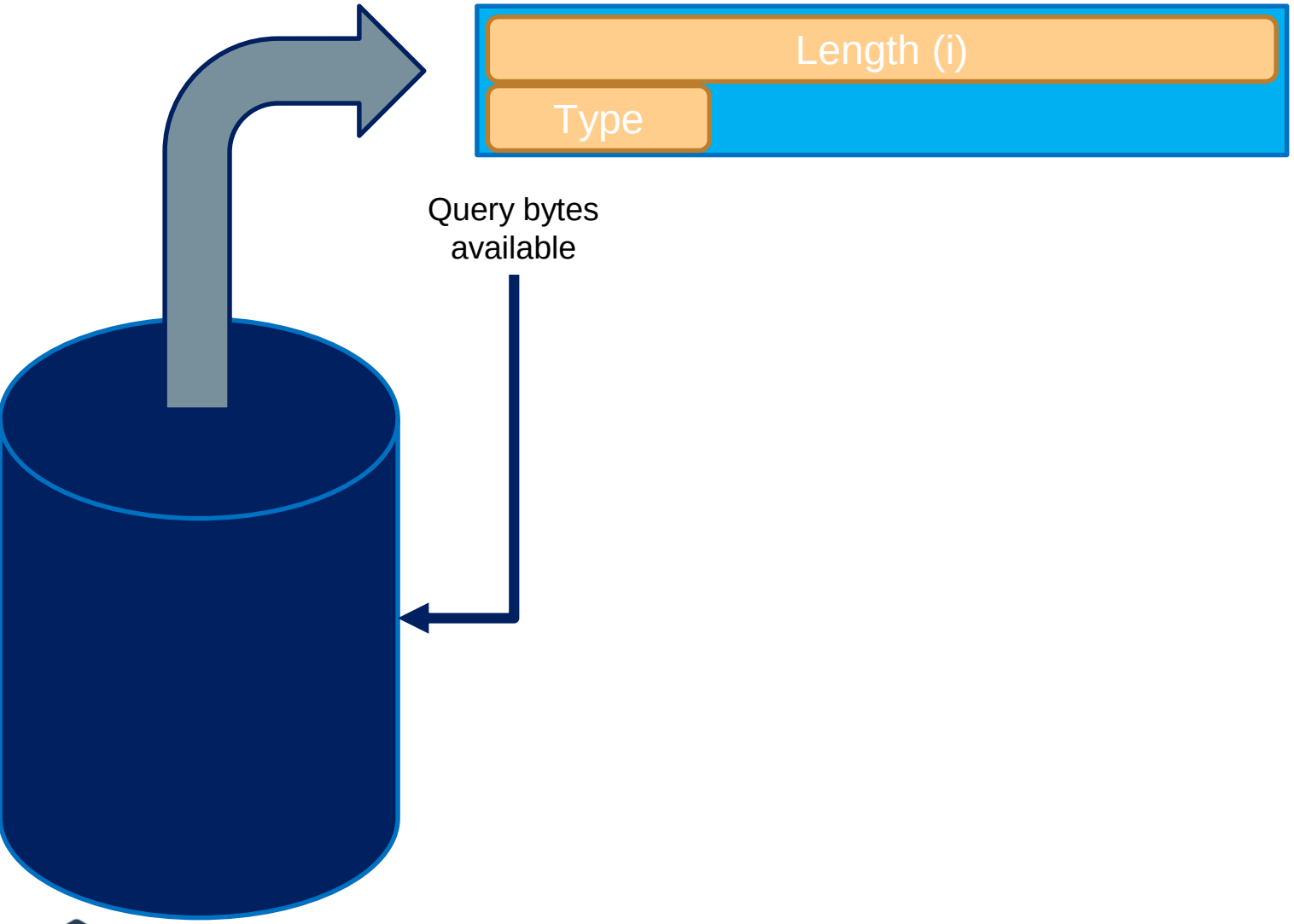


Read 5 bytes (max varint + 1)

Find Length and Type fields in next frame

QUIC

How to Deadlock with One Stream



Read 5 bytes (max varint + 1)

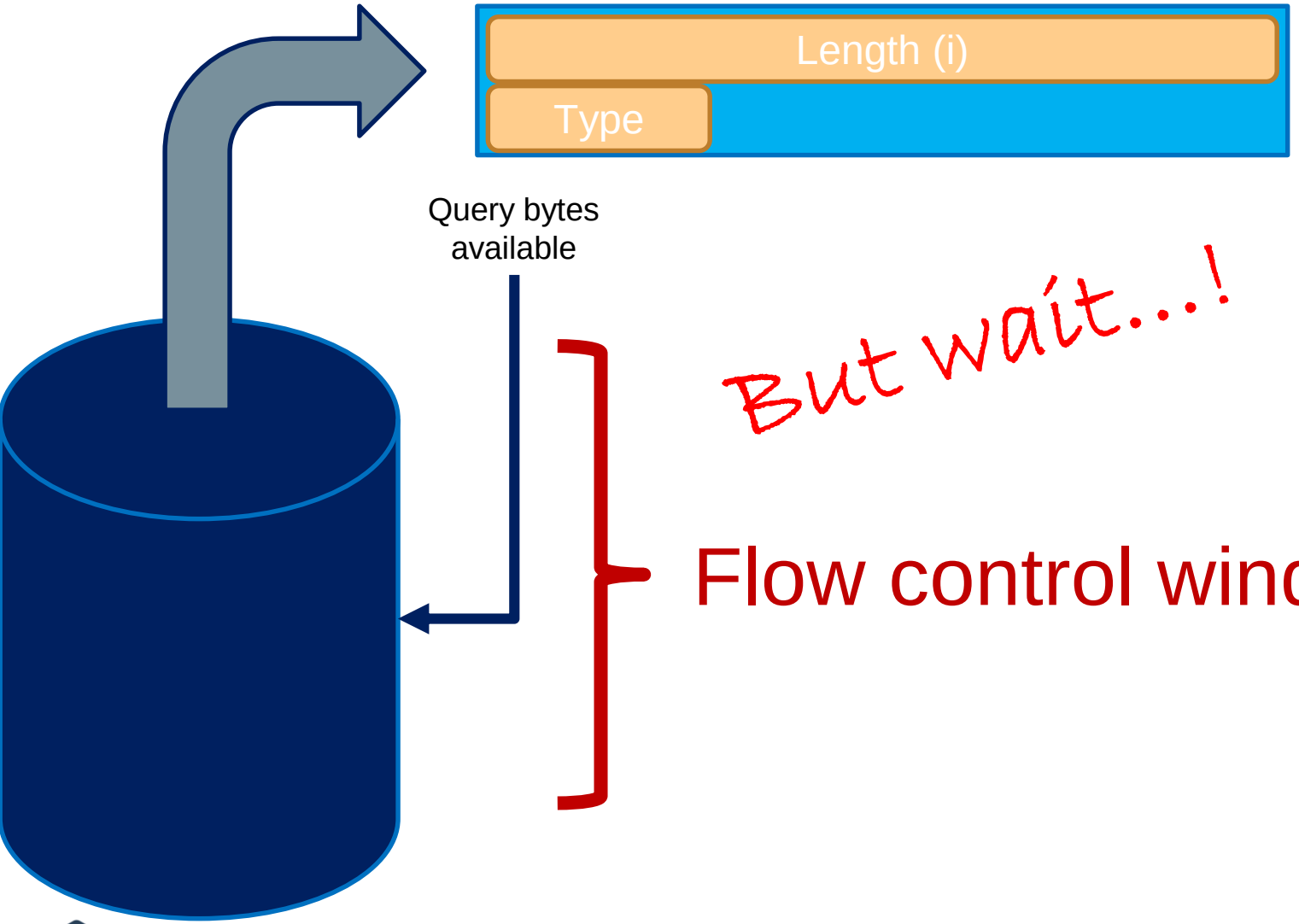
Find Length and Type fields in next frame

Are Length bytes available?

- Yes – read and parse
- No – wait

QUIC

How to Deadlock with One Stream



Read 5 bytes (max varint + 1)

Find Length and Type fields in next frame

But wait...!

Are Length bytes available?

- Yes – read and parse
- No – wait

Flow control window < Length!

QUIC

Ways to solve this

- Don’t do that!
  - Application should always keep reading; this isn’t what flow control is for!
  - …except that becomes a memory consumption DoS, and protecting from that *is* what flow control is for
- …?

Specific Changes

- Transport
  - Need a general discussion of flow control deadlocks, warning to application layer protocols
- HTTP
  - Some frames can be streamed (DATA) while others need the entire payload (HEADERS)
  - Need to coordinate maximum size of unit-processed frames and minimum size of stream flow control window
- QPACK
  - Implementations limit writing to encoder stream to immediately-available flow control window
  - Probably need text in the draft warning implementers