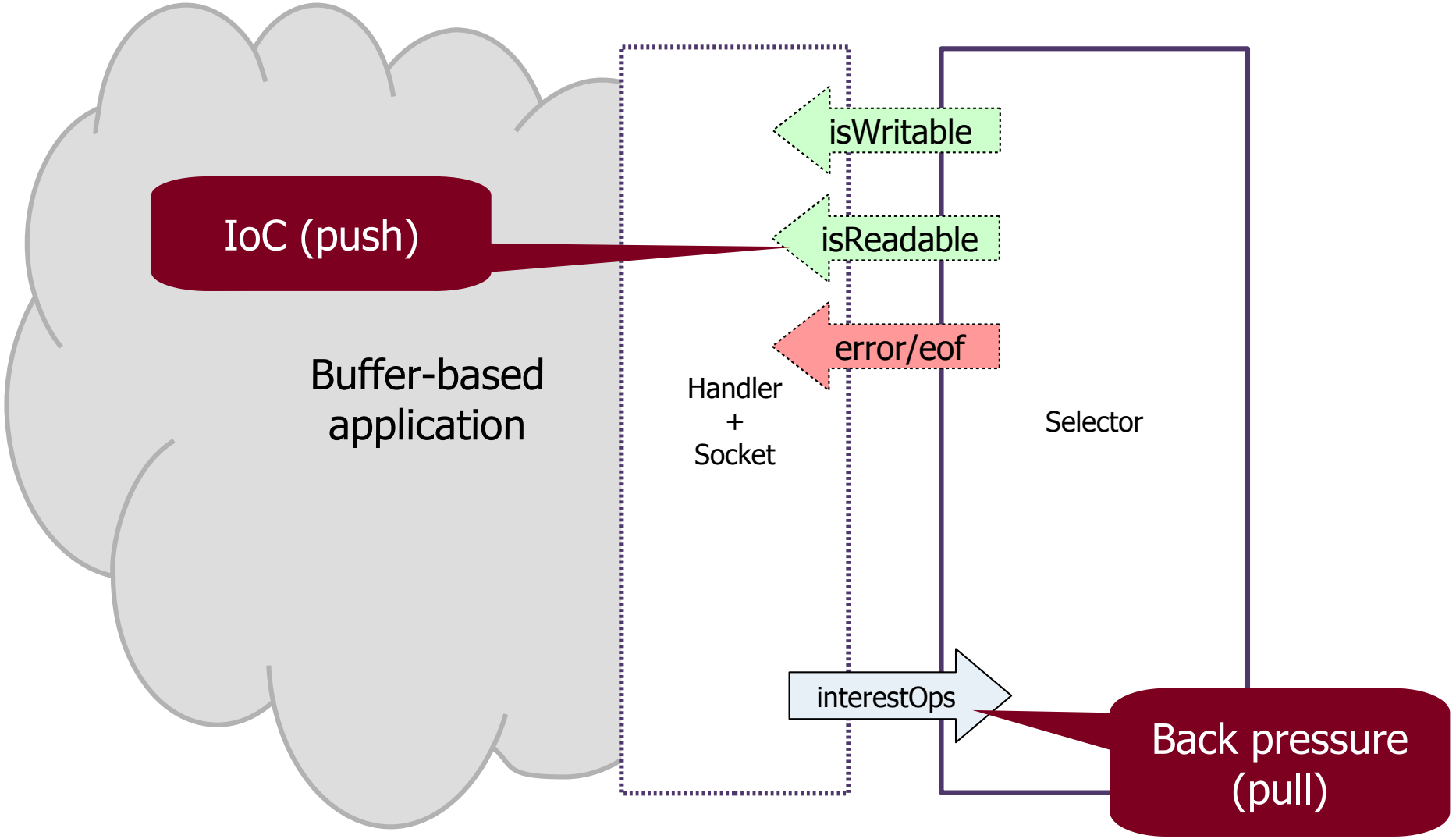
Summary

- Memory:
  - No data copying by reference aliasing
  - Reduced allocation by avoiding captive buffers
- Event-driven programs:
  - A single shallow stack
  - Minimal context switching
  - Explicit scheduling and queuing (can be purged)

Buffer-based application



Challenges

- Reuse and composition:
  - BufferedReader? ObjectInputStream?
- Complex business logic:
  - Sequential work-flows

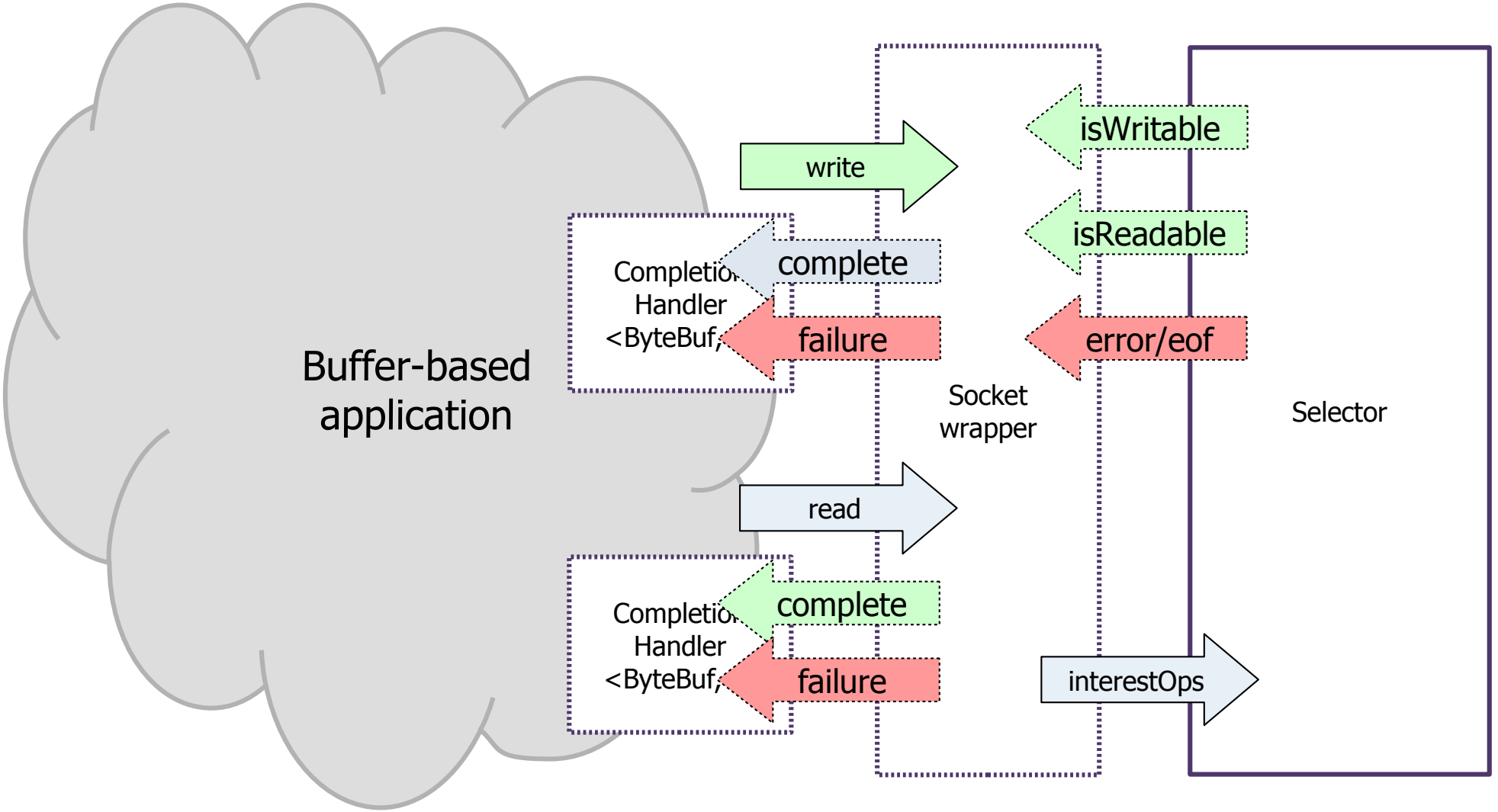
Case study

- Improve the chat server to work with lines, not buffers:
  - Simple example of serialization / reusable layer
- Validate login and password:
  - Simple example of business logic

Alternatives

- Asynchronous callbacks
  - CompletionHandler
- Reactive streams

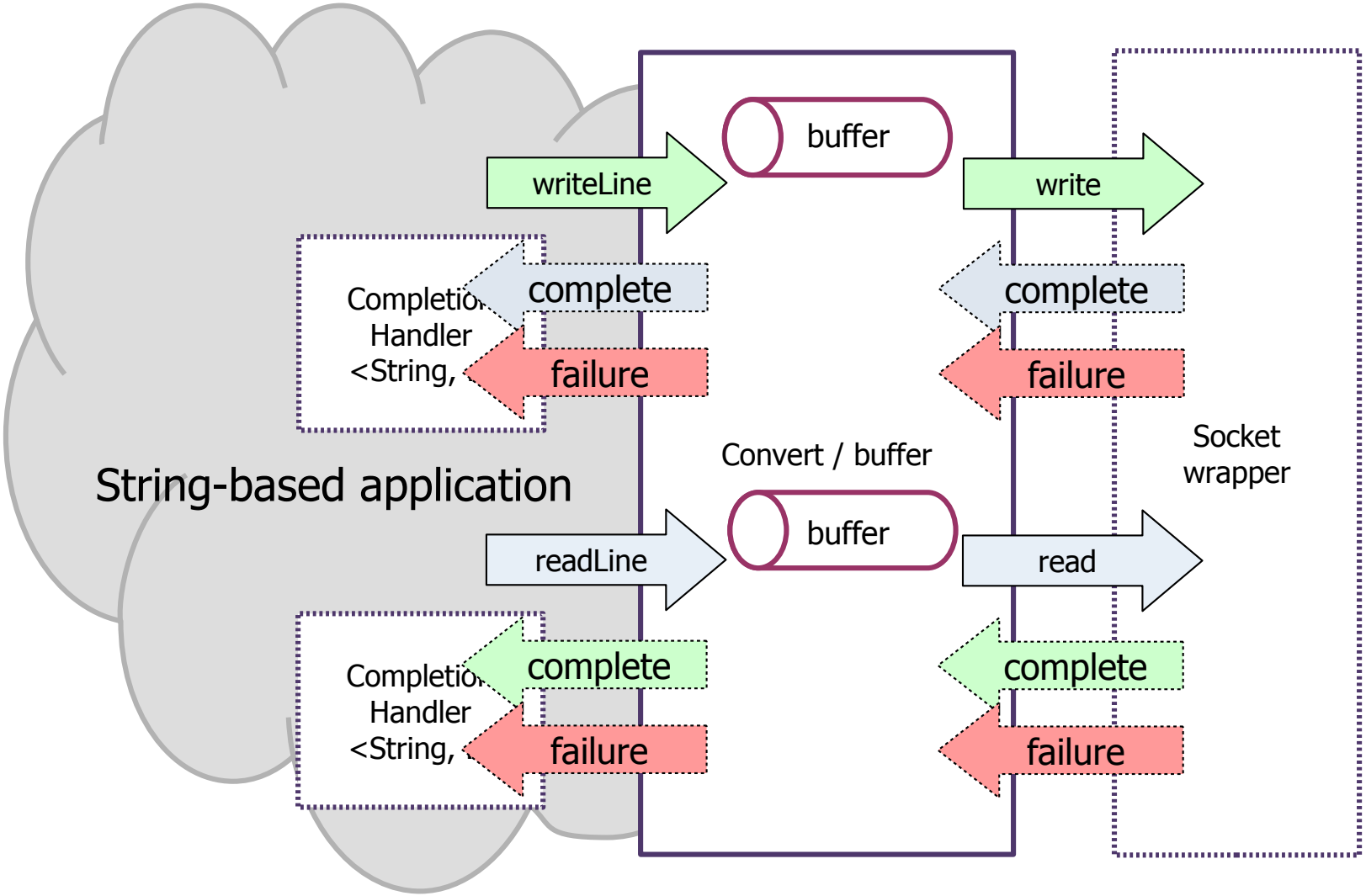
Asynchronous callbacks



Asynchronous callbacks

- Nested callbacks for sequential business logic:
  - Write "Login: "...
  - on complete: Read username…
  - on complete: Write "Password: "…
  - on complete: Read password…
  - ...

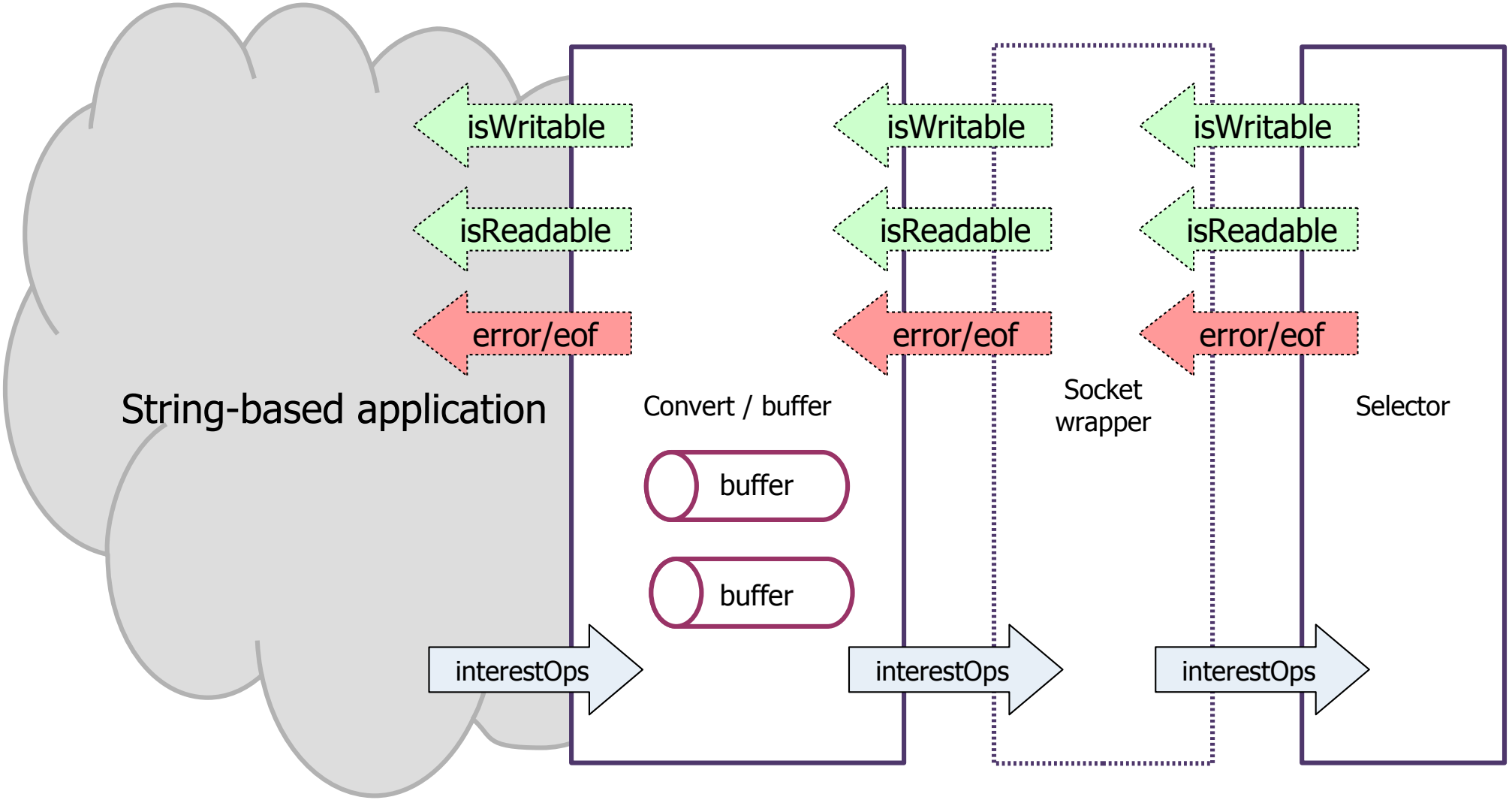
Asynchronous callbacks



Asynchronous callbacks

- Reasonable for sequential business logic
- Not ideal for data transformation and reusable layers:
  - Start and stop for each data item
  - Overhead and complexity
- The read path is awkward / reversed...

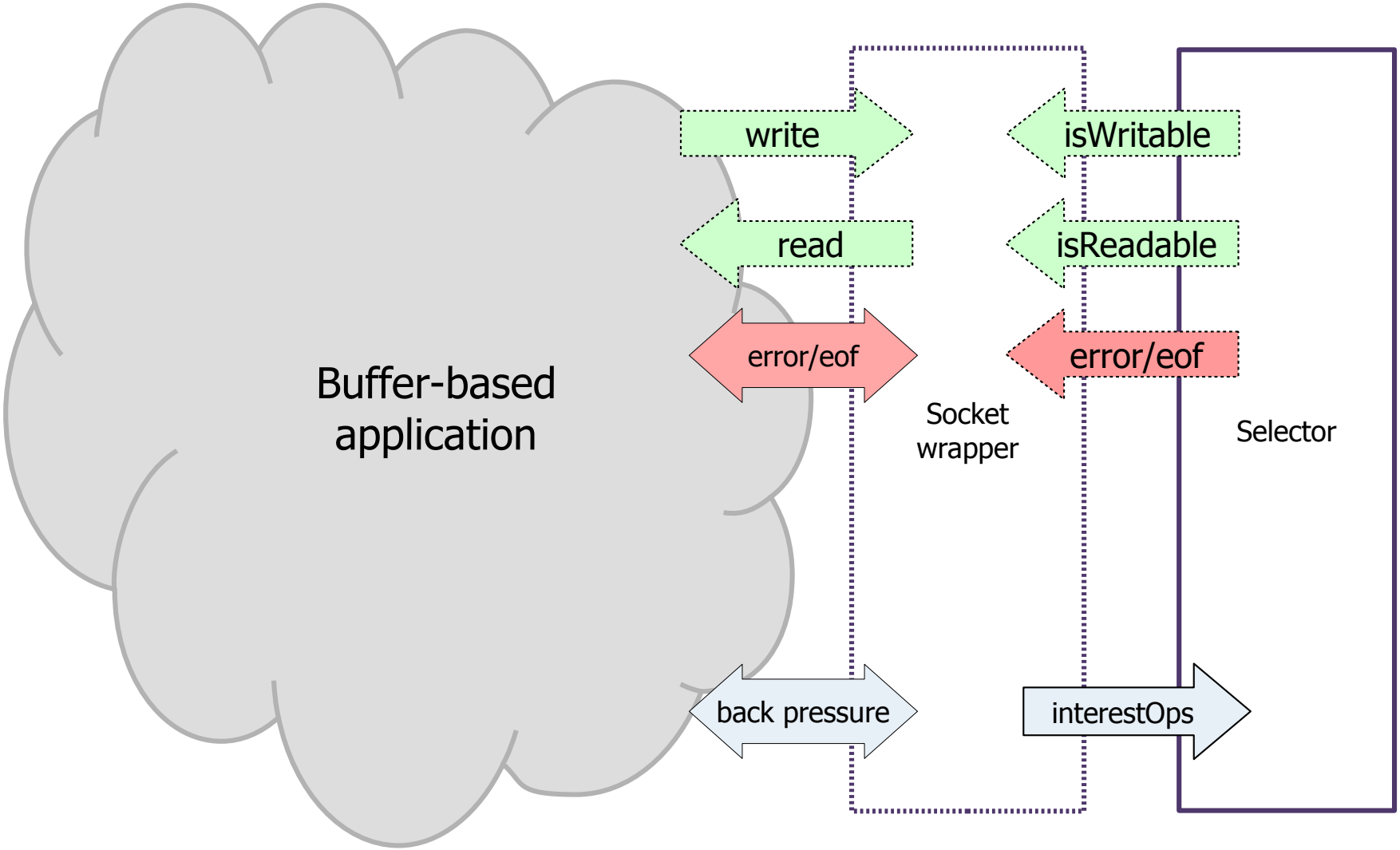
Selector interface



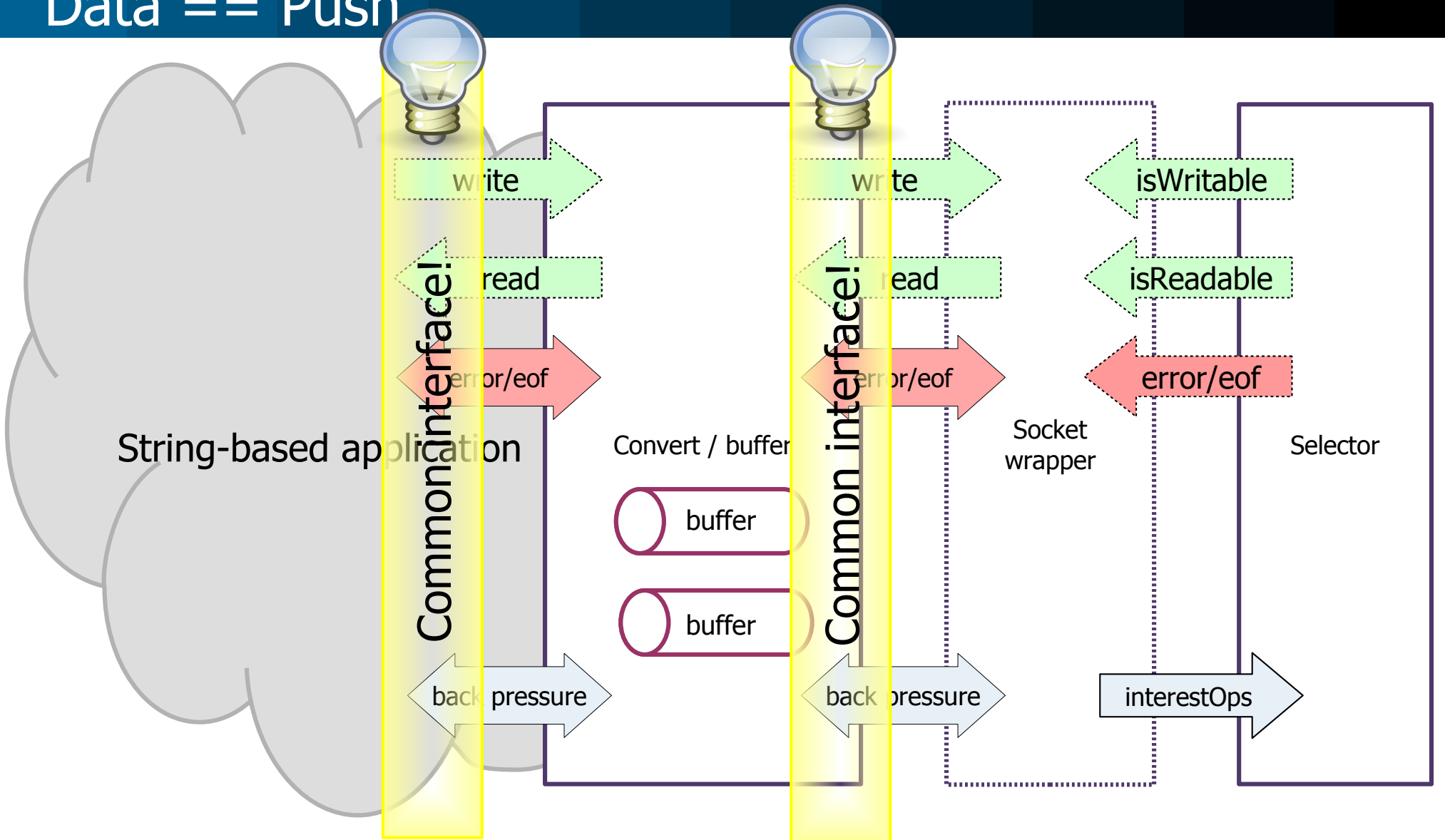
Selector interface

- Reading without explicit request:
  - Efficient / easy to use read path
- Sequential business logic requires explicit state machine
- Write path is awkward / reversed…

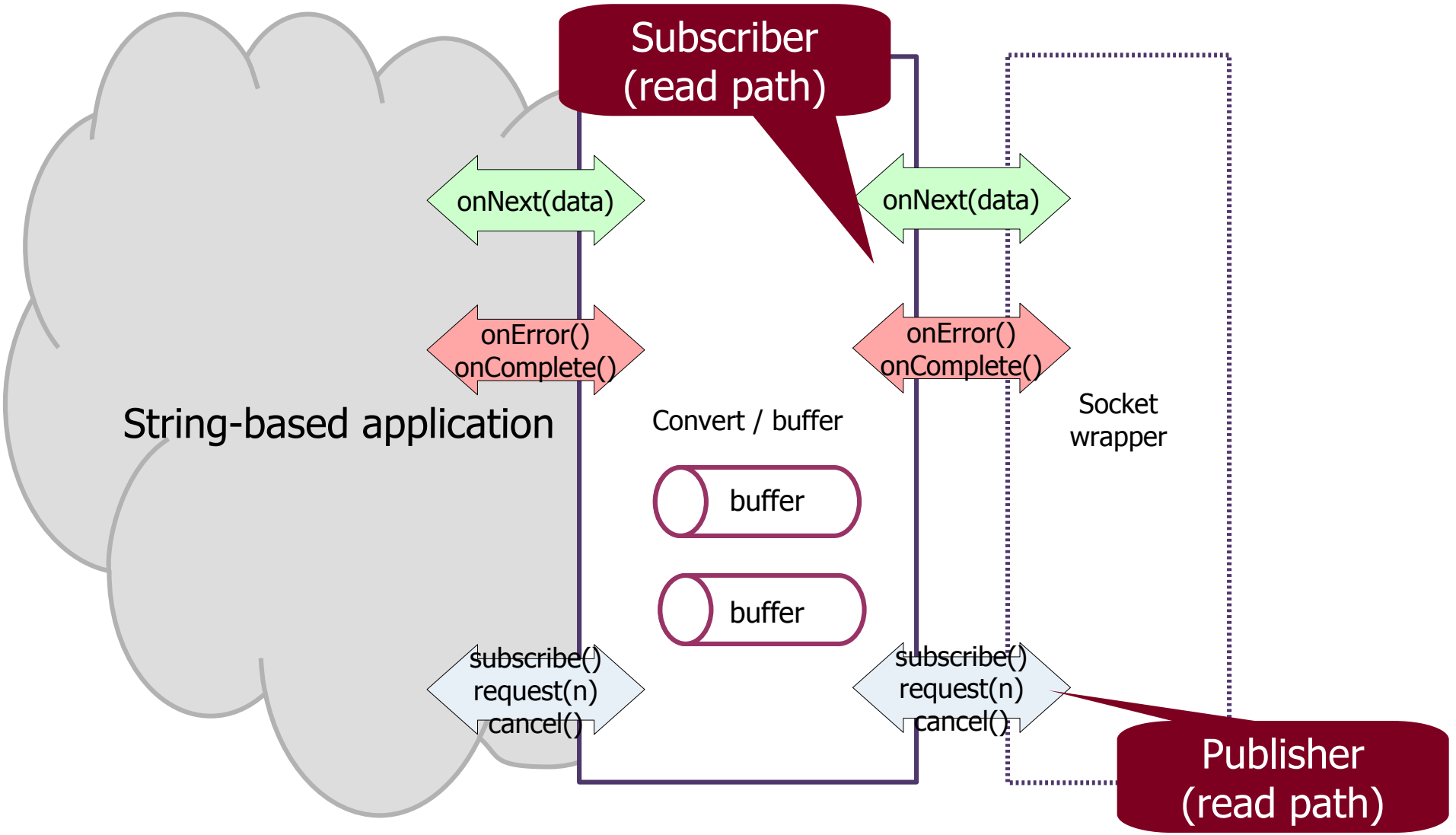
Data == Push



Data == Push



Reactive streams



Reactive streams

- Data and errors:
  - Inversion of control / push model
- Back pressure:
  - Direct / pull model
- Symmetric paths for I/O