Event Publisher System:

We would like to create an event publisher system. This system is responsible for delivering messages reliably generated by a client to the server. The system exposes an API as follows:

```
public protocol EventPublisher {
  func publish(event: Event)
}

public struct Event {
  public let subject: String
  public let payload: String
}
```

From the API we can see that:

- Every event has a subject, determined by the client
- Every event has a payload, determined by the client
- Publishing events is an asynchronous “fire and forget” operation - the client expects the publisher to deliver the event reliably without further interaction.

Reliable publishing:
For the event publisher to be considered reliable, it needs to meet the following requirements:

- Events published to the event publisher should be sent in-order:
    - All events published from a single thread must keep their order of submission.
    - Ordering between different threads is undefined.
- All events sent to the publisher must be delivered even in case the app shuts down or crashes.
- Events should not be lost in case of network failures.
- Events are only considered delivered after a successful acknowledgement by the server [http 200].

The server endpoint:
The server receives events one by one, over an HTTP API. The API exposes a single endpoint:

Endpoint: `POST /publish/[subject]`
Content-Type: `text/plain`
Body: the event's payload
Responses:
`200 OK` - The event was successfully received and acknowledged by the server
`503 Service Unavailable` - The server was unable to accept the event at this time

Your task is to write the event publisher system, make sure it follows the requirements. You may generate events from a UI component, automated tests or any other way you see fit.

Some final highlights:

- Code organization and clean design are important
- We don't care much about input validation and performance optimization, but if you have nice ideas there - feel free to implement them.

Best of luck :)