# Overview of Backpressure Models in the Project Reactor Flux Class


Douglas C. Schmidt

d.schmidt@vanderbilt.edu

www.dre.vanderbilt.edu/~schmidt

Professor of Computer Science

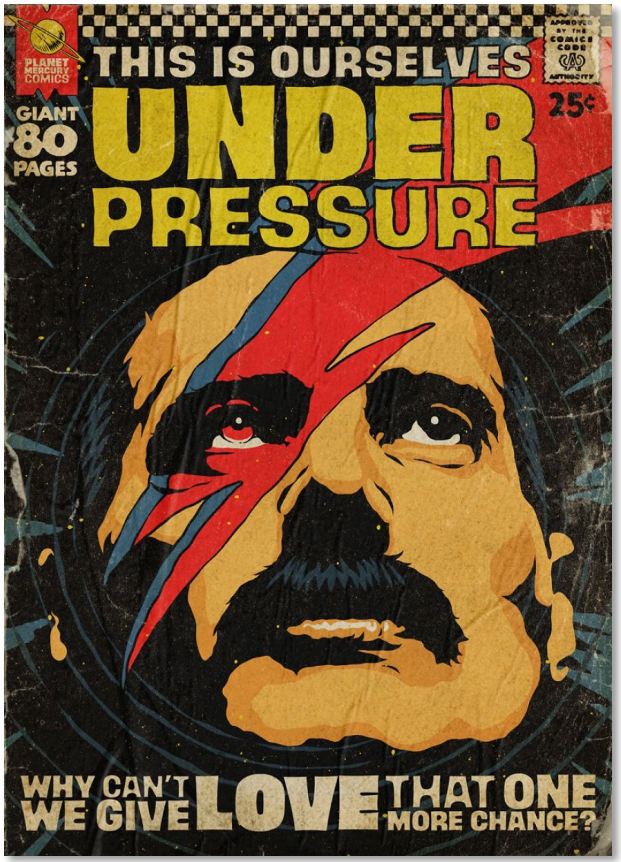
Institute for Software Integrated Systems

Vanderbilt University

Nashville, Tennessee, USA

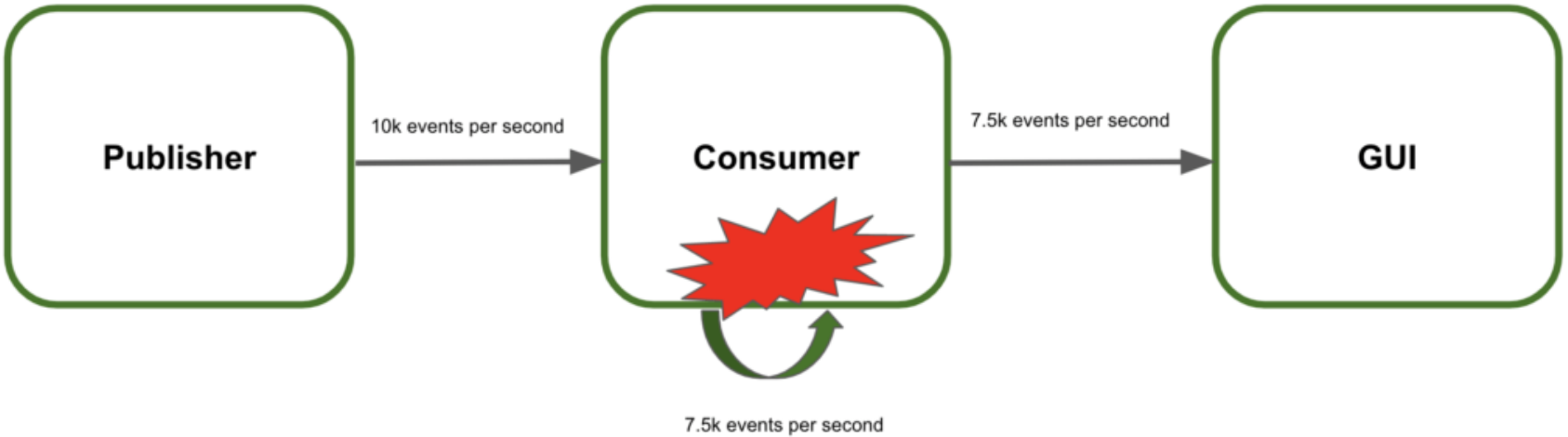

# Learning Objectives in this Part of the Lesson

- Understand key classes in the Project Reactor API
- Know how Project Reactor Flux supports backpressure



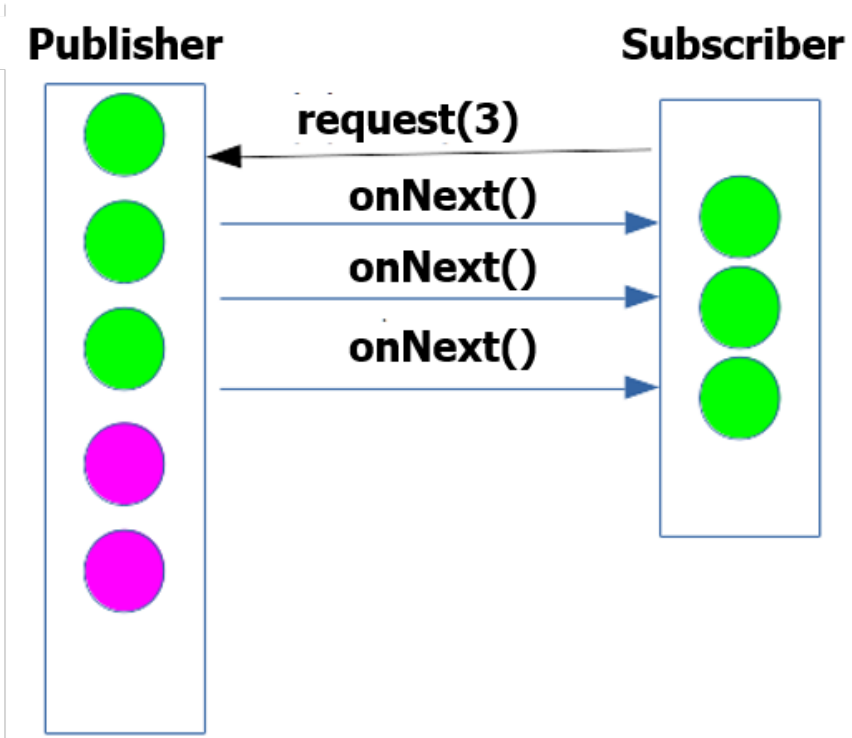
# Learning Objectives in this Part of the Lesson

- Understand key classes in the Project Reactor API
- Know how Project Reactor Flux supports backpressure, e.g.,
  - What motivates the need for back-pressure in reactive systems



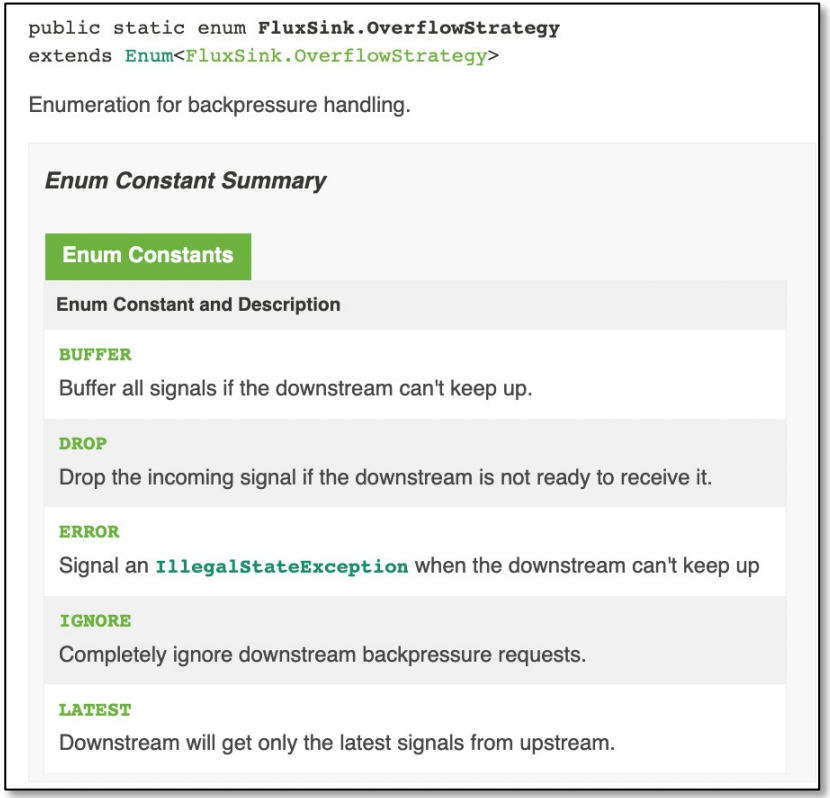
# Learning Objectives in this Part of the Lesson

- Understand key classes in the Project Reactor API
- Know how Project Reactor Flux supports backpressure, e.g.,
  - What motivates the need for back-pressure in reactive systems
  - How the Subscription.request() mechanism supports "backpressure-aware" publishers & subscribers



# Learning Objectives in this Part of the Lesson

- Understand key classes in the Project Reactor API
- Know how Project Reactor Flux supports backpressure, e.g.,
  - What motivates the need for back-pressure in reactive systems
  - How the Subscription.request() mechanism supports "backpressure-aware" publishers & subscribers
  - & overflow strategies support "back-pressure-unaware" publishers & subscribers



public static enum **FluxSink.OverflowStrategy**
extends Enum<FluxSink.OverflowStrategy>

Enumeration for backpressure handling.

***Enum Constant Summary***

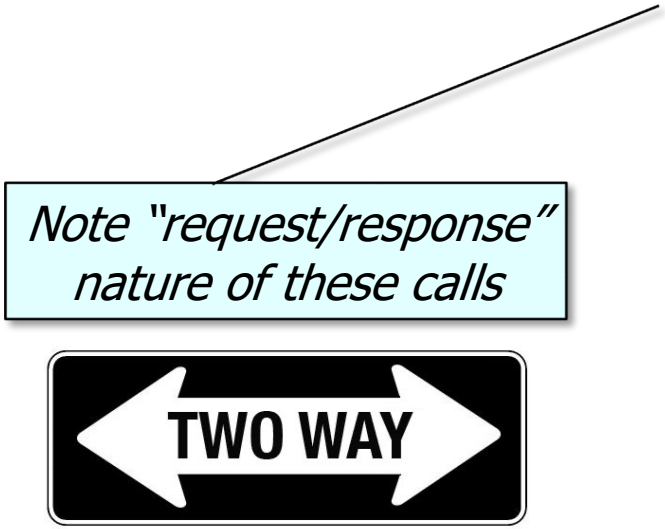
**Enum Constants**

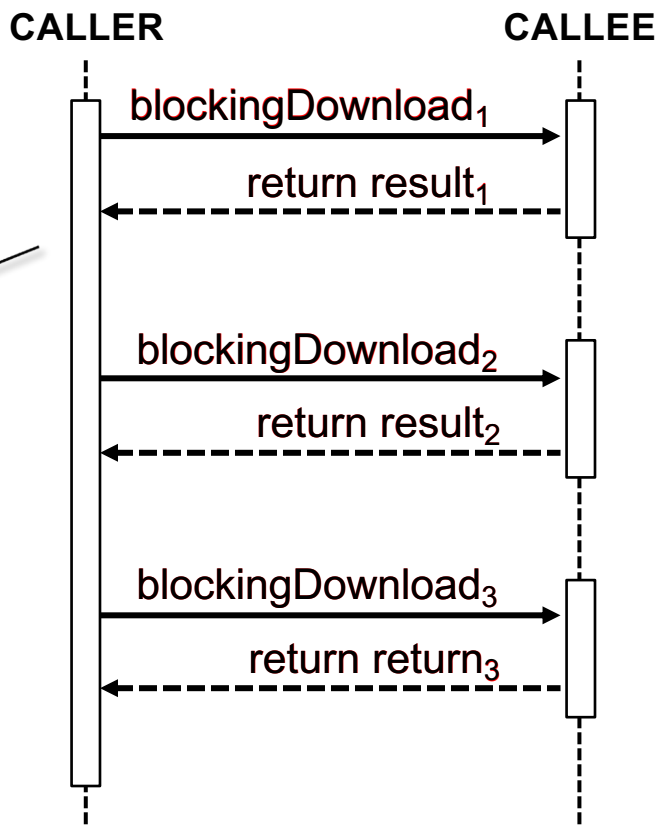
| Enum Constant and Description |
| --- |
| **BUFFER**<br>Buffer all signals if the downstream can't keep up. |
| **DROP**<br>Drop the incoming signal if the downstream is not ready to receive it. |
| **ERROR**<br>Signal an `IllegalStateException` when the downstream can't keep up |
| **IGNORE**<br>Completely ignore downstream backpressure requests. |
| **LATEST**<br>Downstream will get only the latest signals from upstream. |

# Motivation for Back pressure Mechanisms

# Motivation for Backpressure Mechanisms

- Classic client/server systems don't need backpressure mechanisms since two-way synchronous request/response interactions provide a limited form of flow-control




See en.wikipedia.org/wiki/Request-response

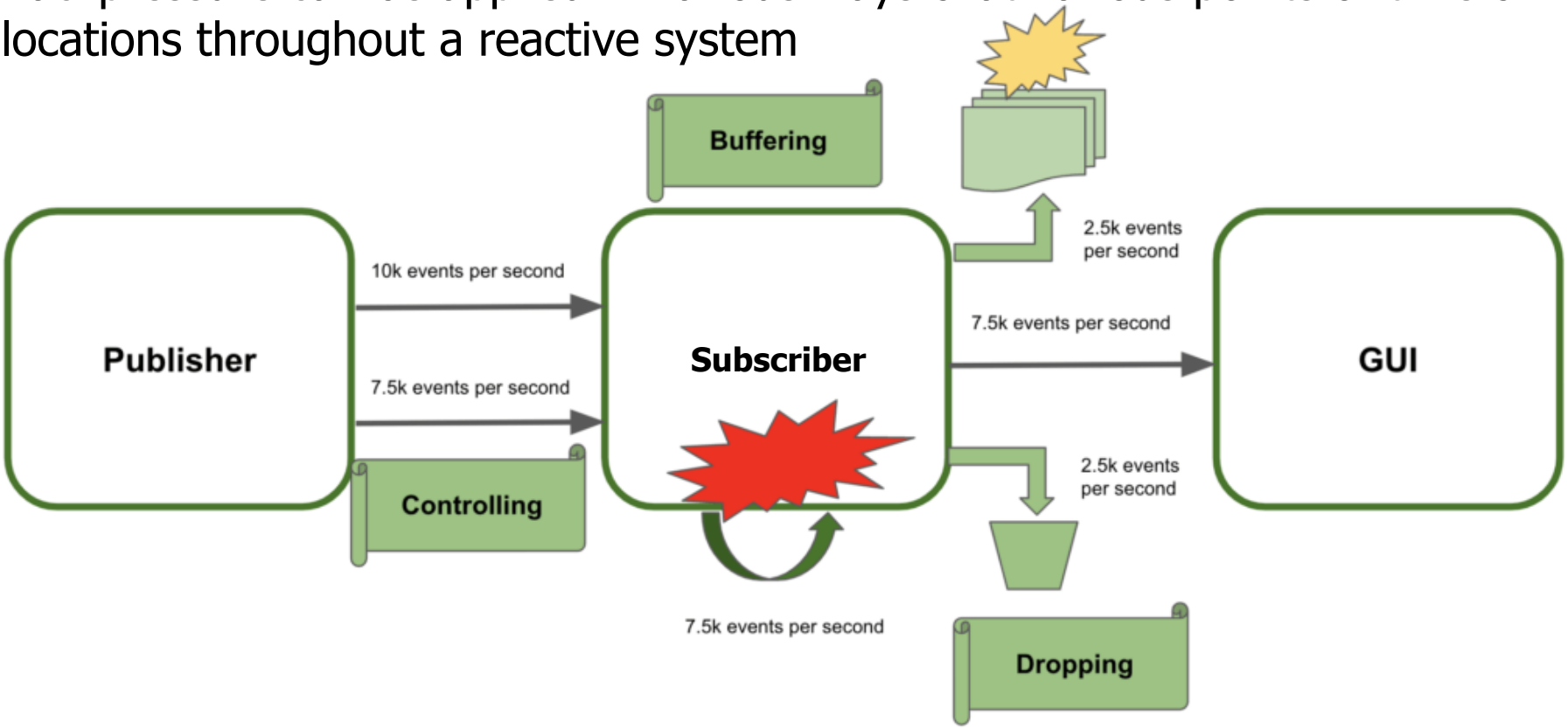
# Motivation for Backpressure Mechanisms

- Some form of backpressure is needed in reactive streams-based systems where Publisher(s) can produce events faster than Subscriber(s) are capable of consuming them


See www.baeldung.com/spring-webflux-backpressure

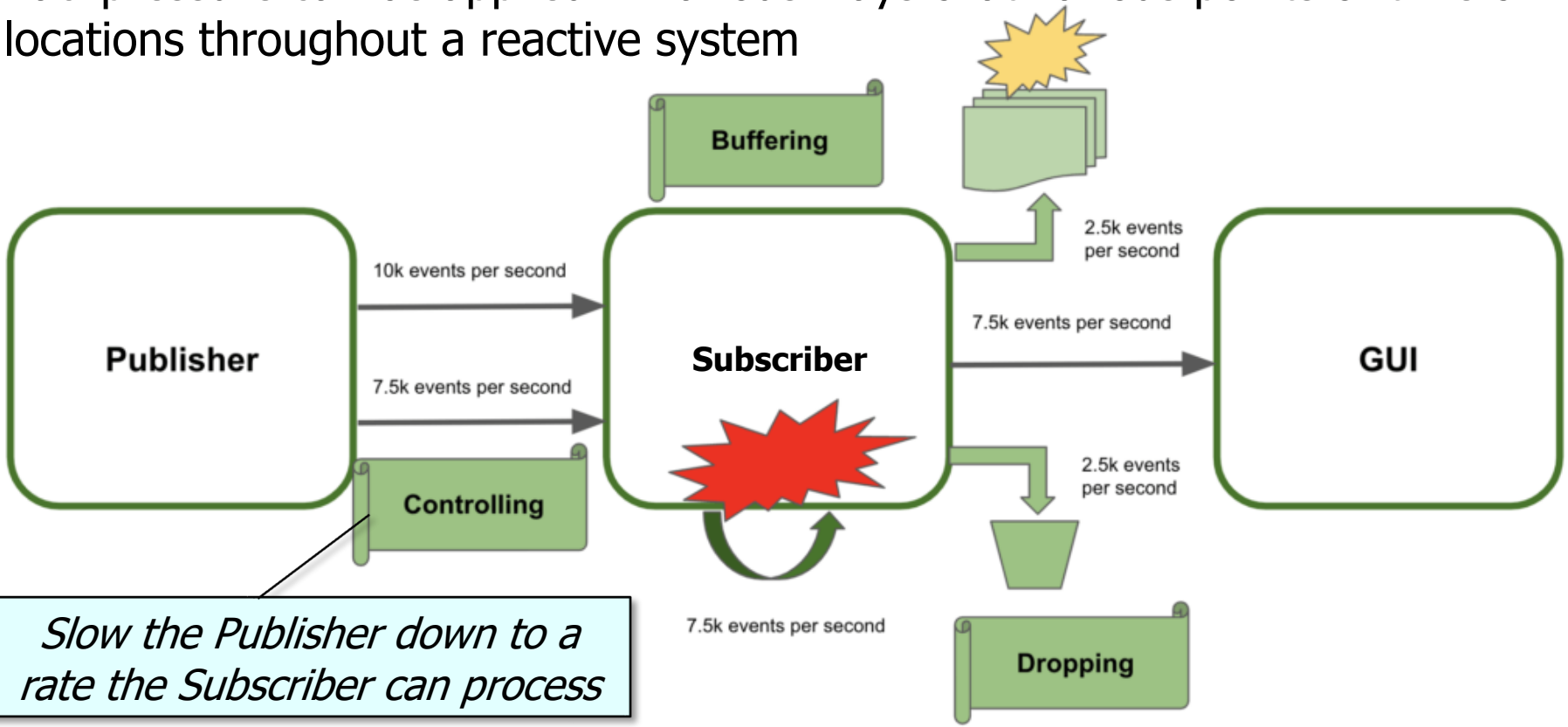
# Motivation for Backpressure Mechanisms

- Backpressure can be applied in various ways & at various points of time & locations throughout a reactive system



# Motivation for Backpressure Mechanisms

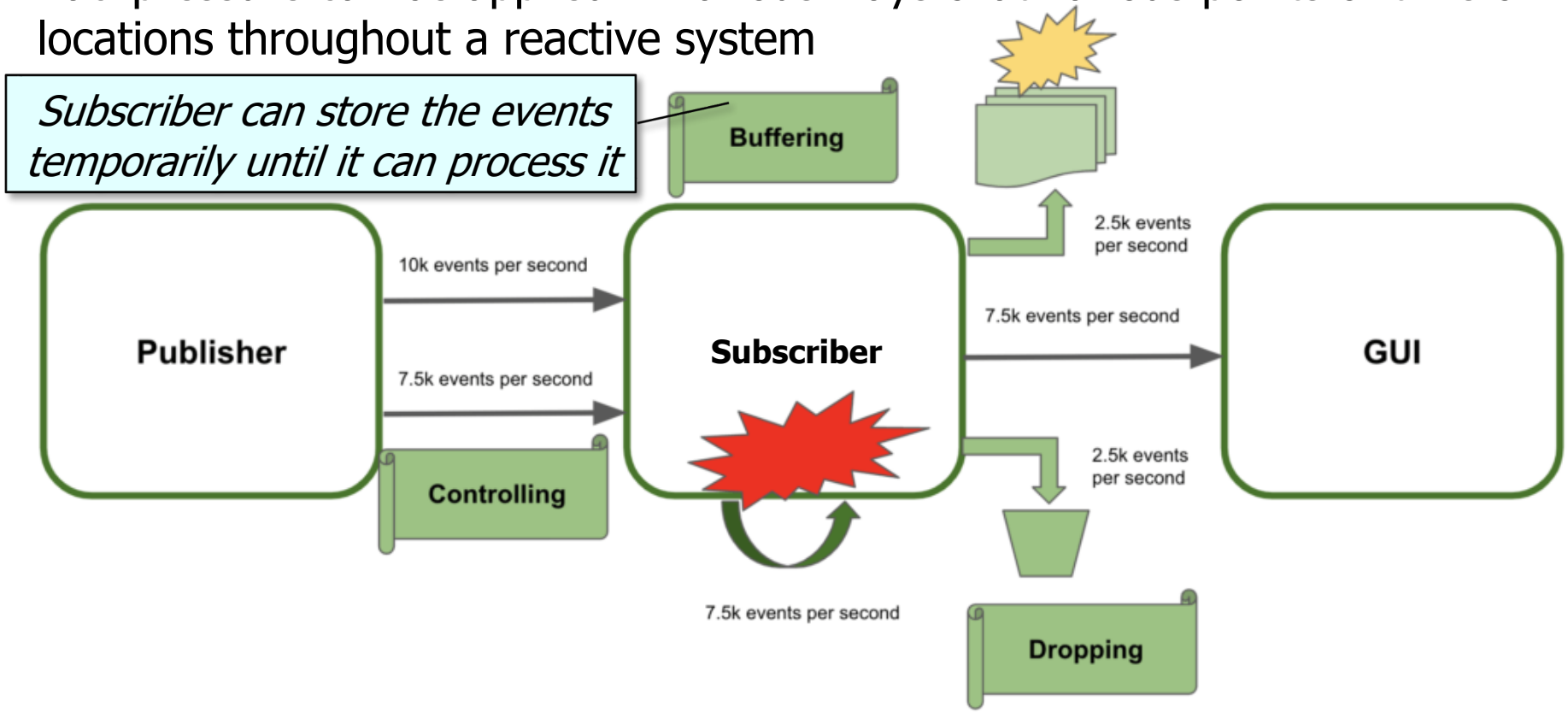
- Backpressure can be applied in various ways & at various points of time & locations throughout a reactive system




May not always be possible, especially for cyber-physical systems

# Motivation for Backpressure Mechanisms

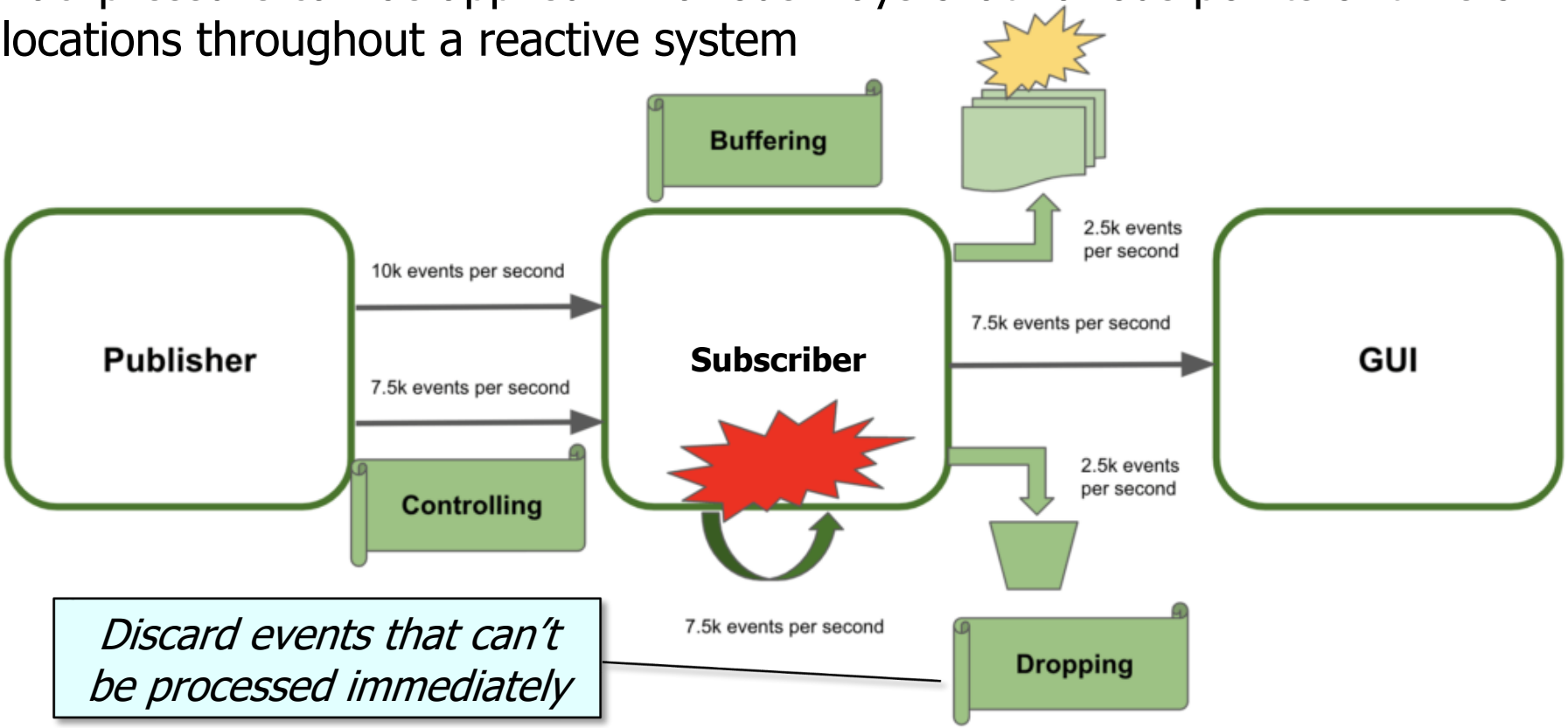
- Backpressure can be applied in various ways & at various points of time & locations throughout a reactive system




May eventually cause "out-of-memory" exceptions!

# Motivation for Backpressure Mechanisms

- Backpressure can be applied in various ways & at various points of time & locations throughout a reactive system

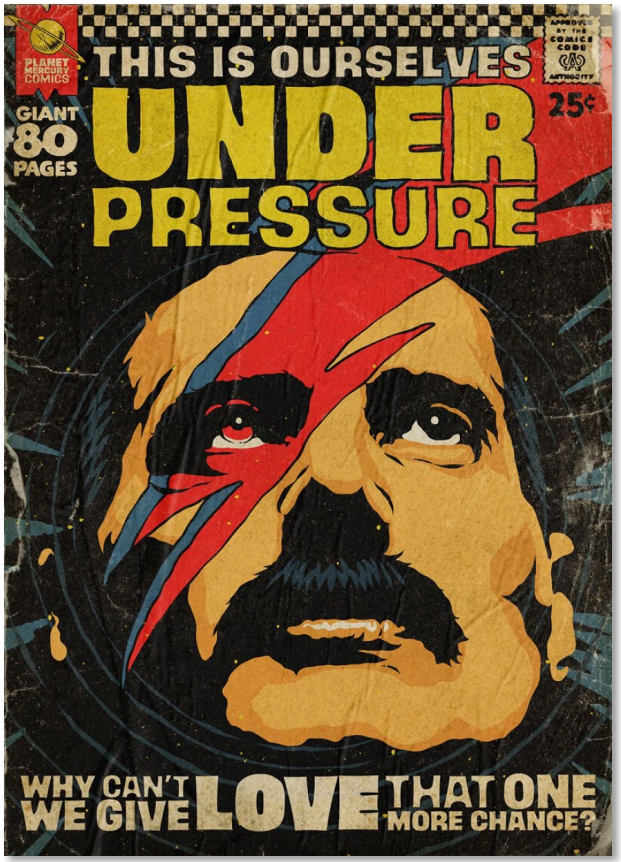



May be problematic if all events contain valuable data

# Overview of Backpressure in Project Reactor Flux

# Overview of Backpressure in Project Reactor Flux

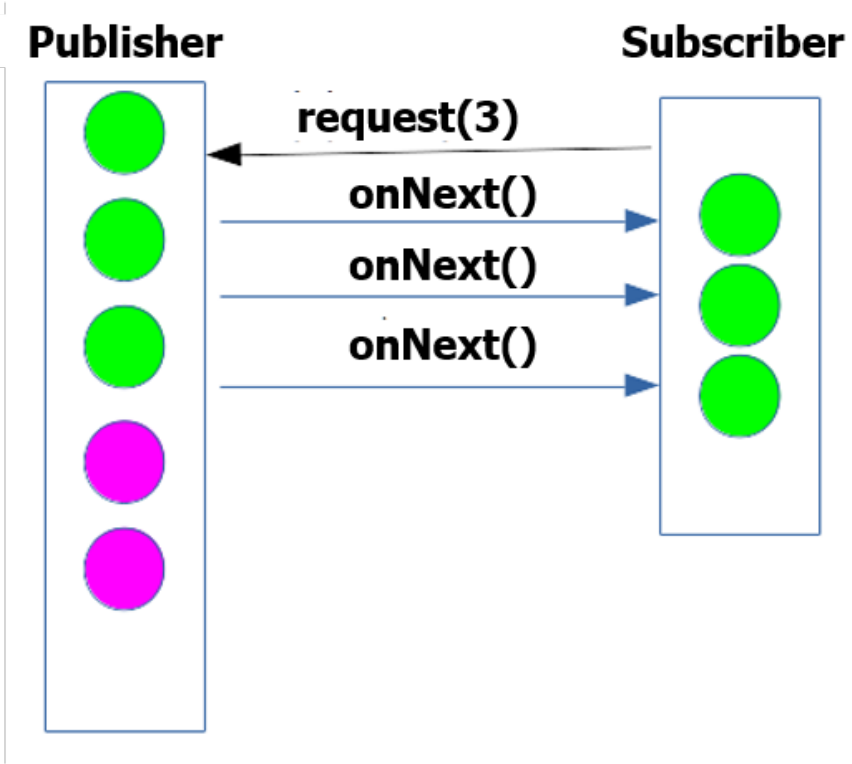
- Project Reactor Flux supports several types of backpressure




See jstobigdata.com/java/backpressure-in-project-reactor

# Overview of Backpressure in Project Reactor Flux

- Project Reactor Flux supports several types of backpressure, e.g.
  - Backpressure-aware Subscriber(s) can inform Publisher(s) how much data they can consume



# Overview of Backpressure in Project Reactor Flux

- Project Reactor Flux supports several types of backpressure, e.g.
  - Backpressure-aware Subscriber(s) can inform Publisher(s) how much data they can consume
    - The goal is to avoid overwhelming memory/processing resources
      - i.e., flow-control Publisher(s) so they don't generate events faster than Subscriber(s) can consume them

See www.appsdeveloperblog.com/implementing-backpressure-in-project-reactor

# Overview of Backpressure in Project Reactor Flux

- Project Reactor Flux supports several types of backpressure, e.g.
  - Backpressure-aware Subscriber(s) can inform Publisher(s) how much data they can consume
    - The goal is to avoid overwhelming memory/processing resources
  - Requires Publisher(s) & Subscriber(s) to interact & collaborate

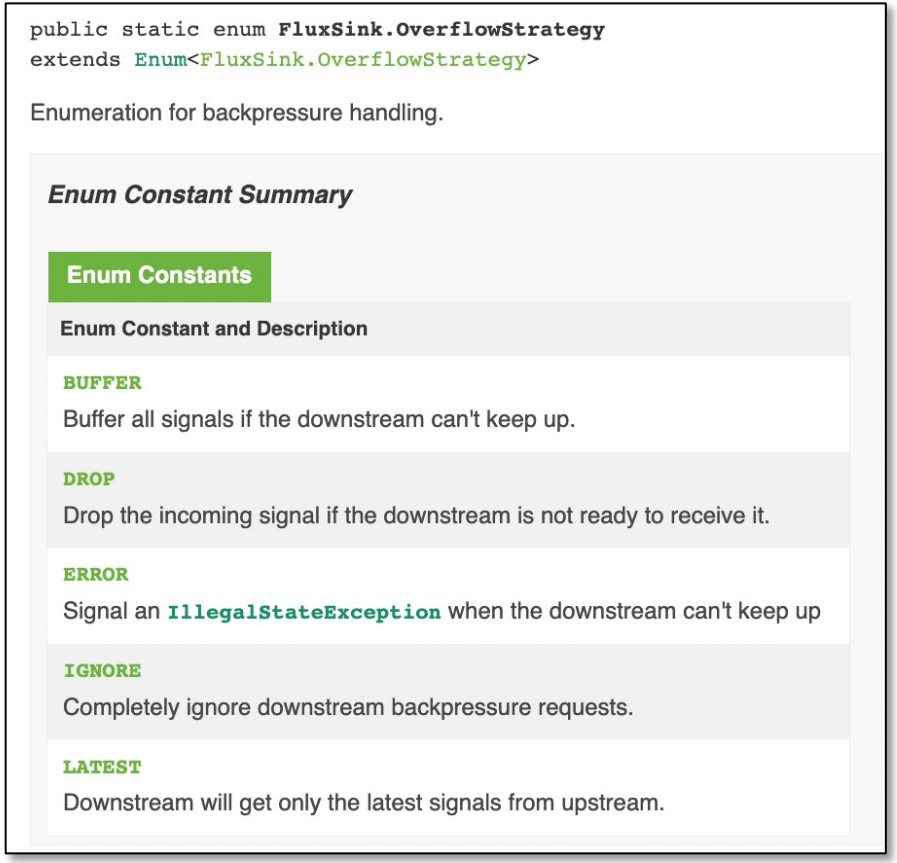
```
void onSubscribe
   (Subscription subscription){
     mSubscription =
       subscription;

     subscription
       .request(mRequestSize);
}
```

*Subscriber(s) call the request() method on a Subscription passed by Publisher(s) to Subscriber(s) via the onSubscribe() hook method*

See github.com/douglascraigschmidt/LiveLessons/tree/master/Reactive/flux/ex4

# Overview of Backpressure in Project Reactor Flux

- Project Reactor Flux supports several types of backpressure, e.g.
  - Backpressure-aware Subscriber(s) can inform publisher(s) how much data they can consume
  - Non-backpressure-aware Subscriber(s) can apply an overflow strategy if they can't keep up with faster Publisher(s)



```
public static enum FluxSink.OverflowStrategy
extends Enum<FluxSink.OverflowStrategy>
```

Enumeration for backpressure handling.

***Enum Constant Summary***

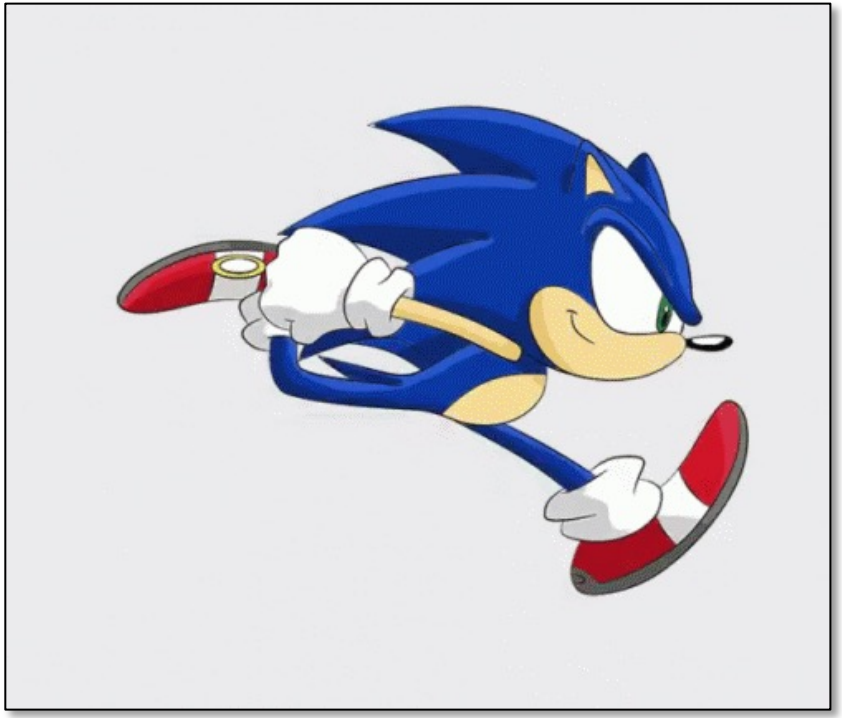
**Enum Constants**

| Enum Constant and Description |
| --- |
| **BUFFER**<br>Buffer all signals if the downstream can't keep up. |
| **DROP**<br>Drop the incoming signal if the downstream is not ready to receive it. |
| **ERROR**<br>Signal an `IllegalStateException` when the downstream can't keep up |
| **IGNORE**<br>Completely ignore downstream backpressure requests. |
| **LATEST**<br>Downstream will get only the latest signals from upstream. |

See projectreactor.io/docs/core/release/api/reactor/core/publisher/FluxSink.OverflowStrategy.html

# Overview of Backpressure in Project Reactor Flux

- Project Reactor Flux supports several types of backpressure, e.g.
  - Backpressure-aware Subscriber(s) can inform publisher(s) how much data they can consume
  - Non-backpressure-aware Subscriber(s) can apply an overflow strategy if they can't keep up with faster Publisher(s)
    - i.e., non-flow-controlled Publisher(s)



# End of Overview of Backpressure Models in the Project Reactor Flux Class