

Muon Workshop

17-05-2017

<http://muoncore.io/>

<http://github.com/muoncore/muon-workshop>

David Dawson

Freelance Systems Architect

London Microservices User Group Founder & Lead

Recovering Microservices Consultant

Project lead – Muon & Co

me@daviddawson.me

@davidthecoder

What we will discuss...

- 1) Why you should care about Microservices
- 2) What this costs you...
- 3) Examples of decomposition & recomposition
- 4) Muon Core
- 5) Building Event Systems
- 6) DDD Patterns and Newton
- 7) Wrap up

Why you should care about
Microservices?

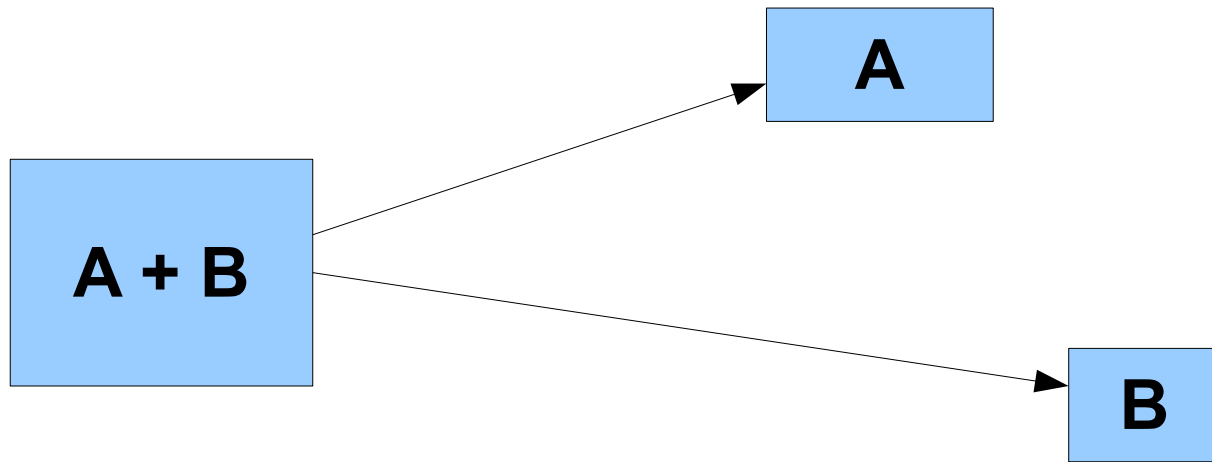
What is a
Microservice Architecture?

What is a
Microservice Architecture?

Isolation

Isolation
+
Aspiration

What this costs you



Towards the glorious world of Microservices ...

What this costs you

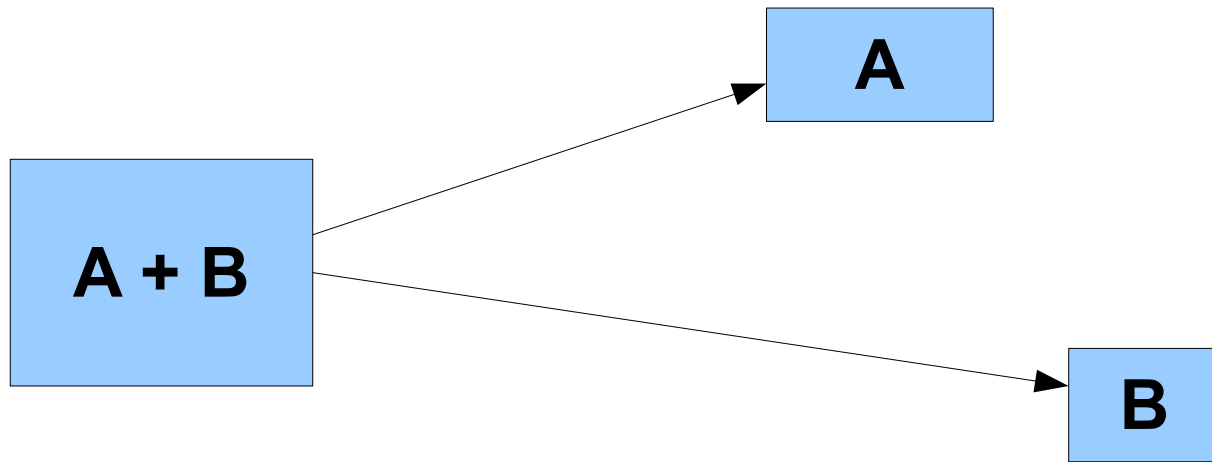
Benefit

- Component change is easy
- Can scale teams with components
- Take advantage of cloud platforms
- Polyglot

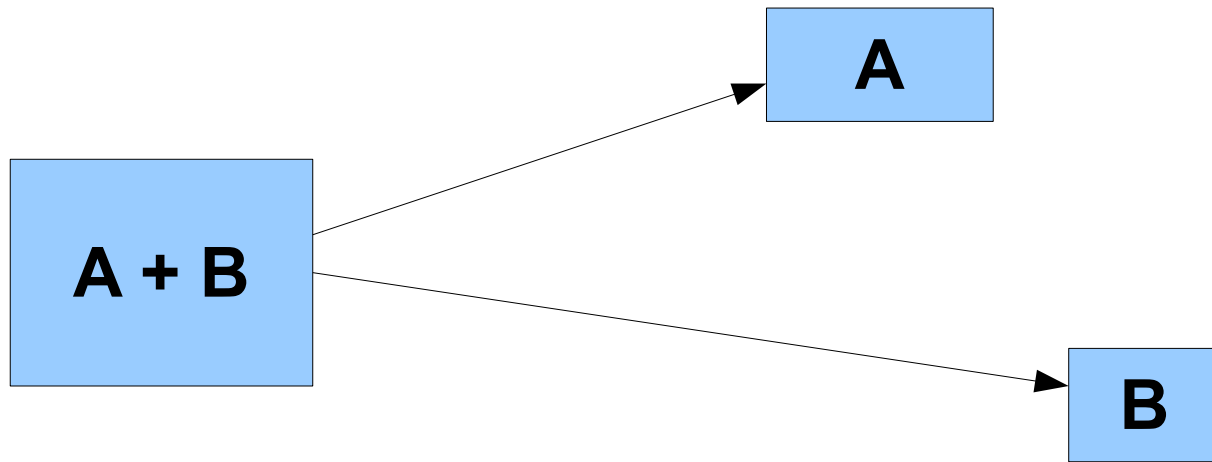
Costs

- Systemic change is harder
- Discovery is harder
- APIs are networked
- Data is not globally consistent
- Production may not be fully deterministic (ask me what I mean by this...)

Decomposition



Decomposition



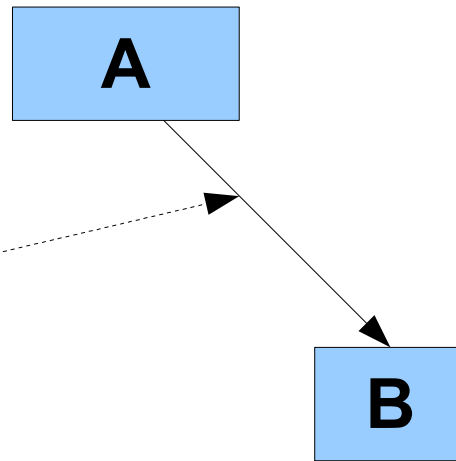
How do we decide what A and B are?

Recomposition

A

B

Recomposition



What does this look
like?

Recomposition using RPC (http anyone?)

HTTP?

AKA – Text Based RPC
Fragile

Recomposition using Messaging

Messaging?

Recomposition Using Events

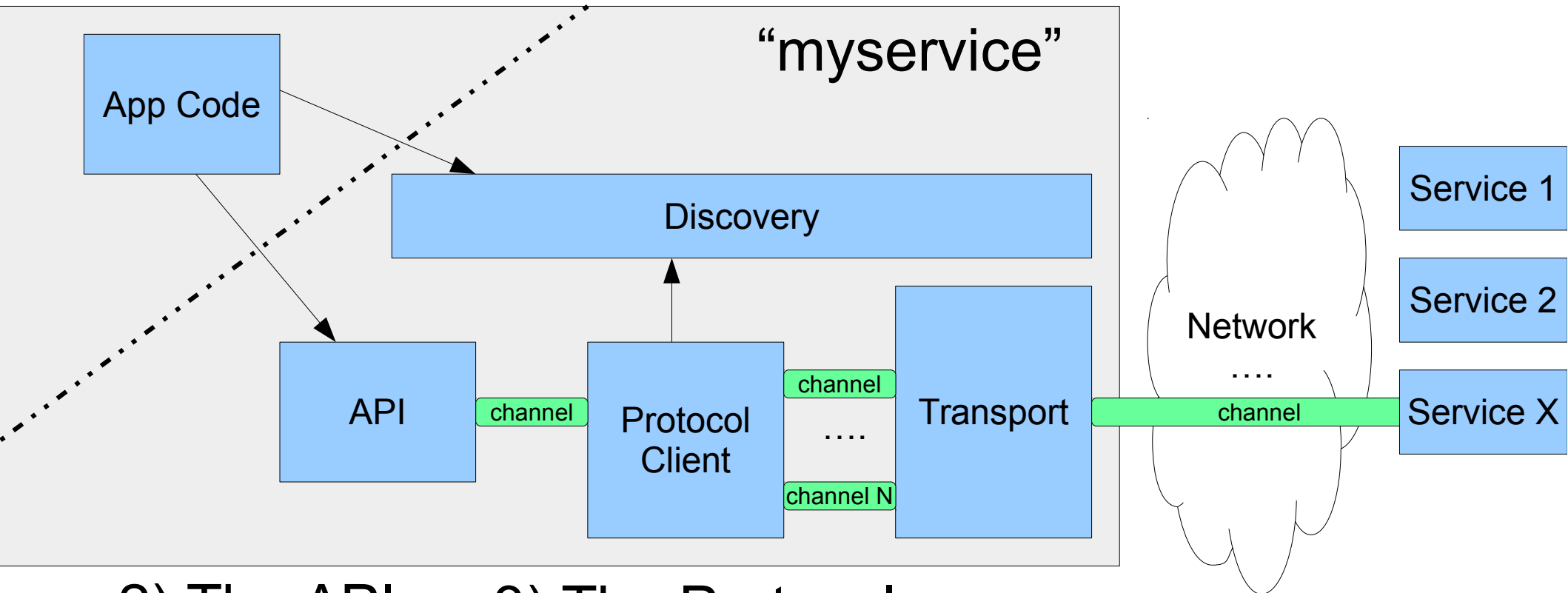
Messaging?

Muon Core

*Make building message based
microservices easy*

Muon

1) Your code
uses an API



2) The API
creates a
Protocol

3) The Protocol
communicates
with others
using
messages

4) The transport
routes them over the
network

Protocols

- RPCish
- Reactive Streams
- Introspection
- Pipeline/ Multi process Co-ordination
- Events
- [your protocol here]

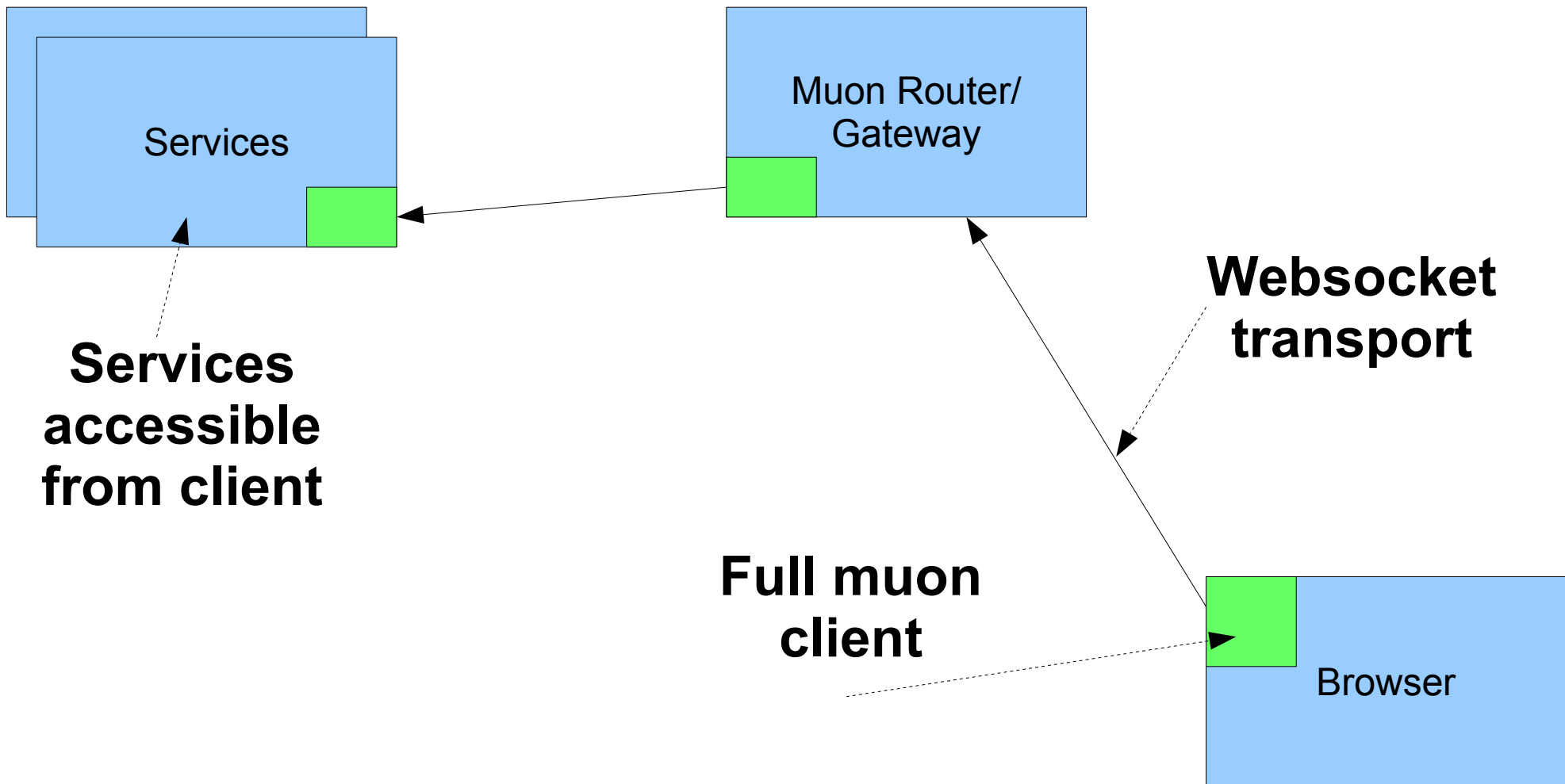
Running the demos

- <http://github.com/muoncore/muon-workshop>
- You need to get a base env
 - see <http://github.com/muoncore/muon-starter>
- Java
- NPM
- Muon CLI

Demo

1 - Muon intro

Muon.js in the Browser



Demo

2 – Muon.js

Recomposition using Events

Events

Events

Order Placed

Events

Order Placed

Payment Taken

Events

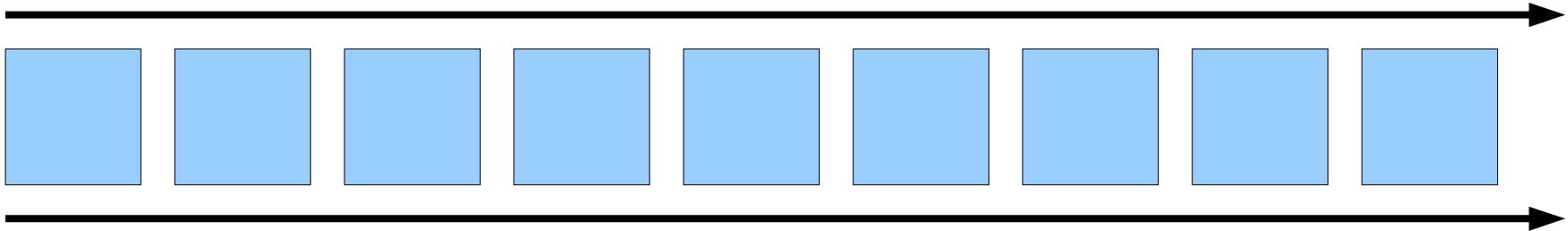
Order Placed

Payment Taken

Email Sent

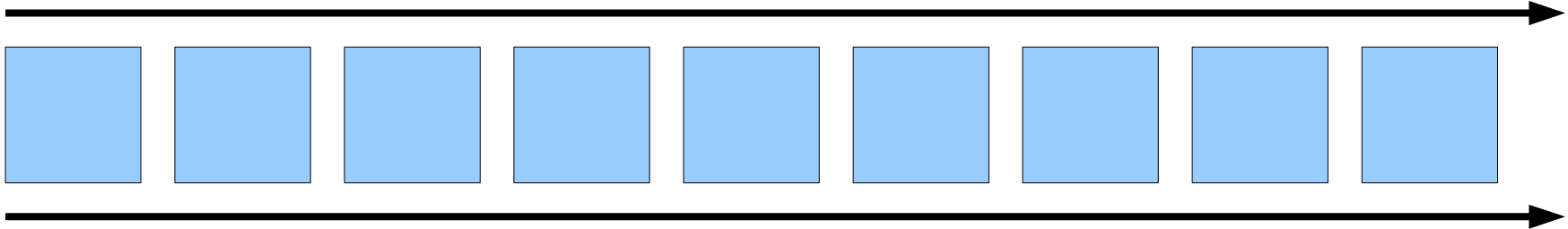
Order Stream

Events

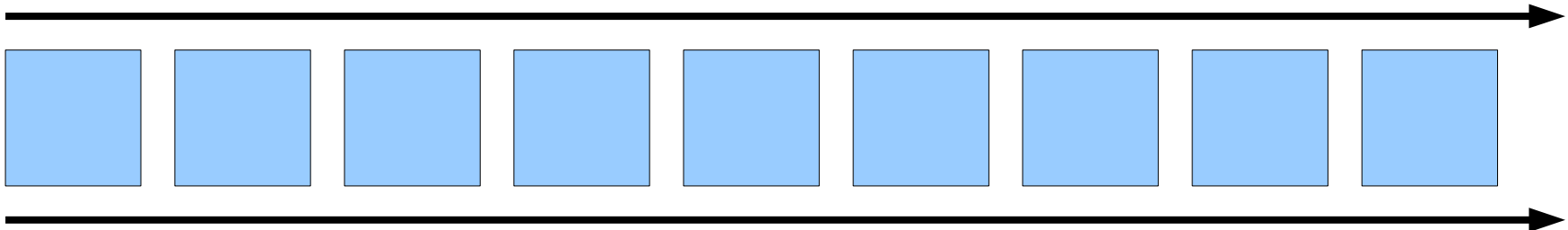


Events

Order Stream

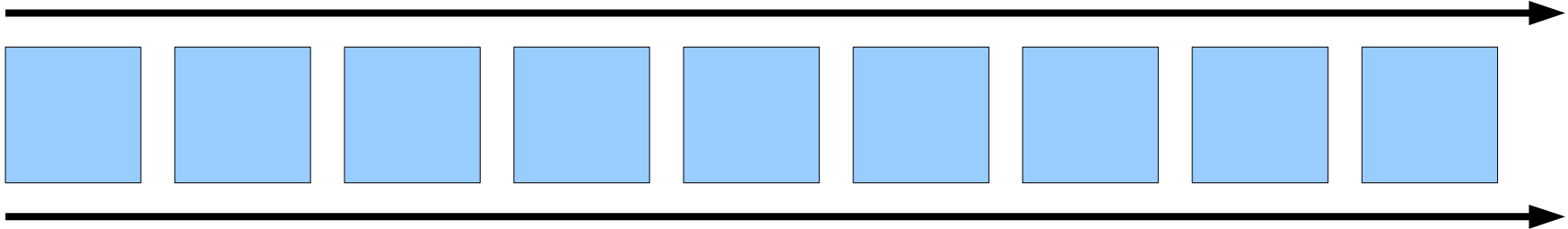


Payment Stream

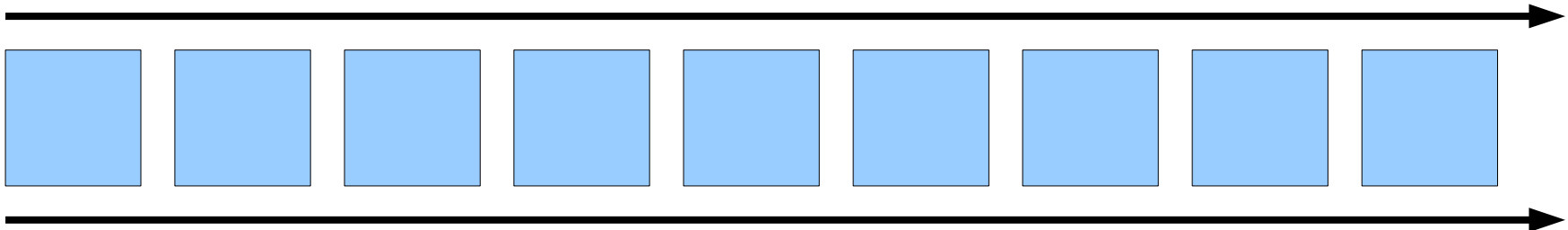


Events

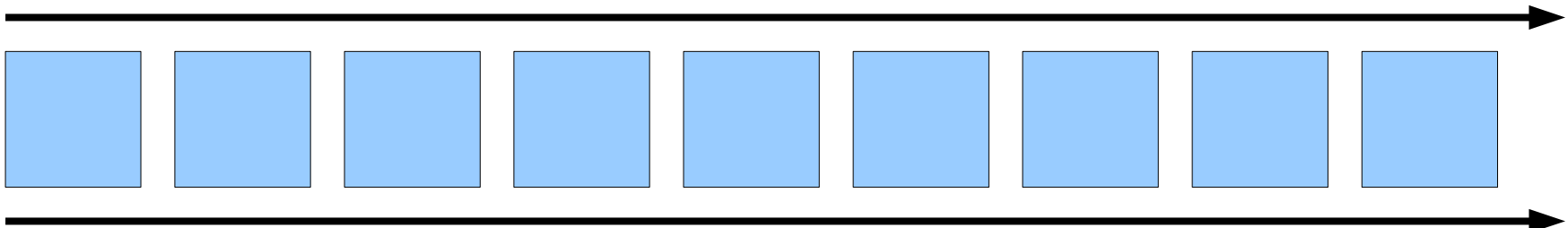
Order Stream



Payment Stream

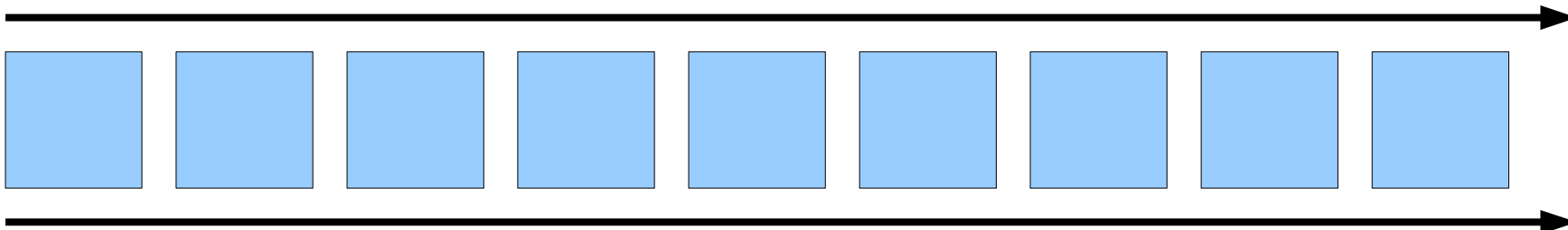


Notification Stream



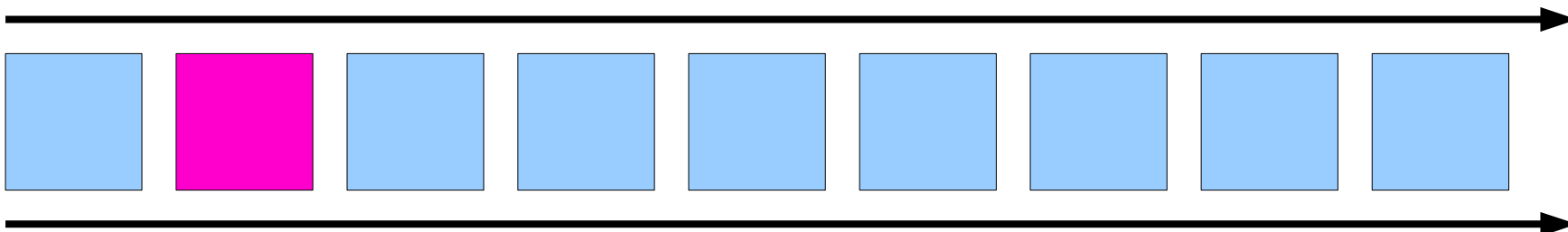
Order Stream

Events



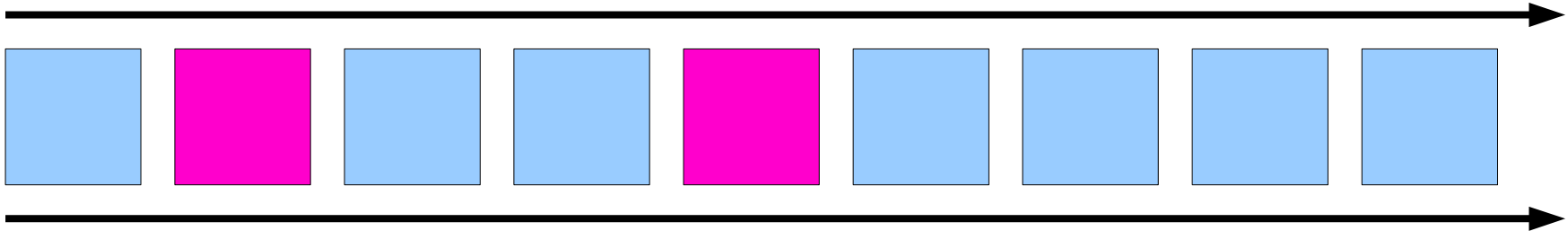
Order Stream

Events



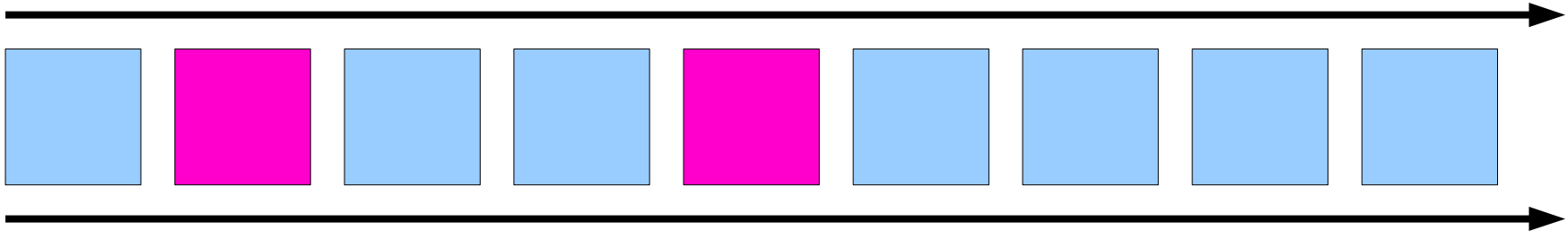
Order Stream

Events



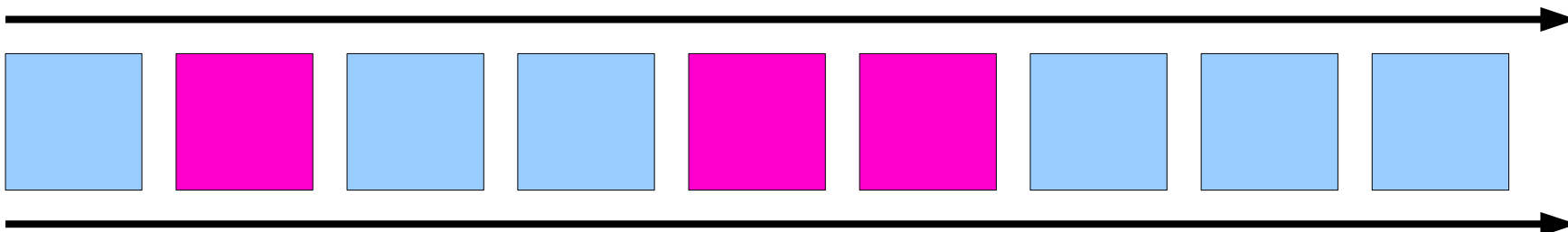
Order Stream

Events



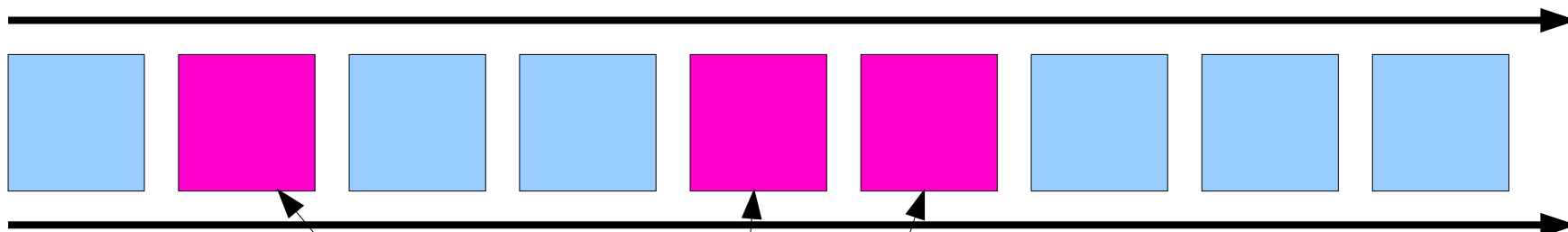
Order Stream

Events



Order Stream

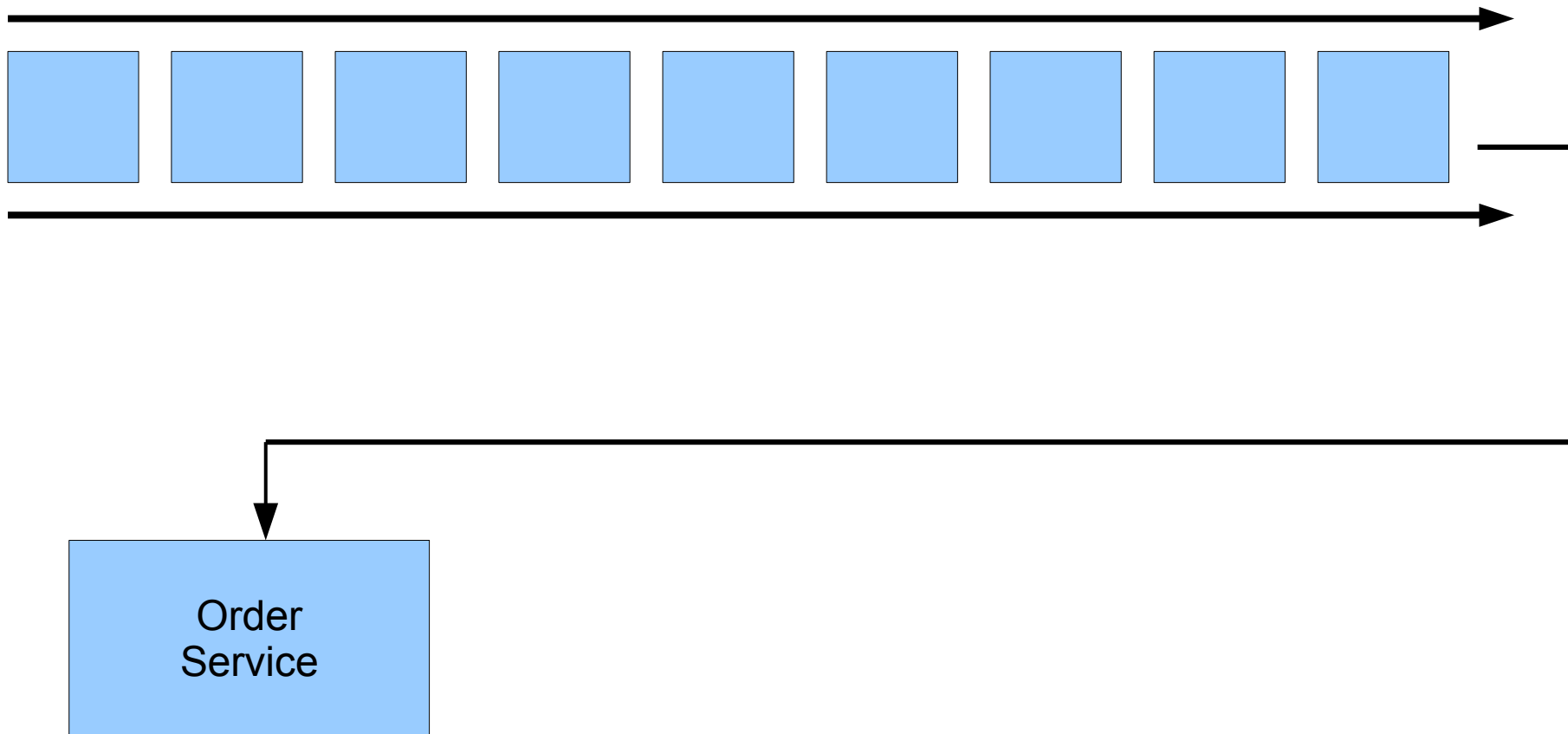
Events



An Entity

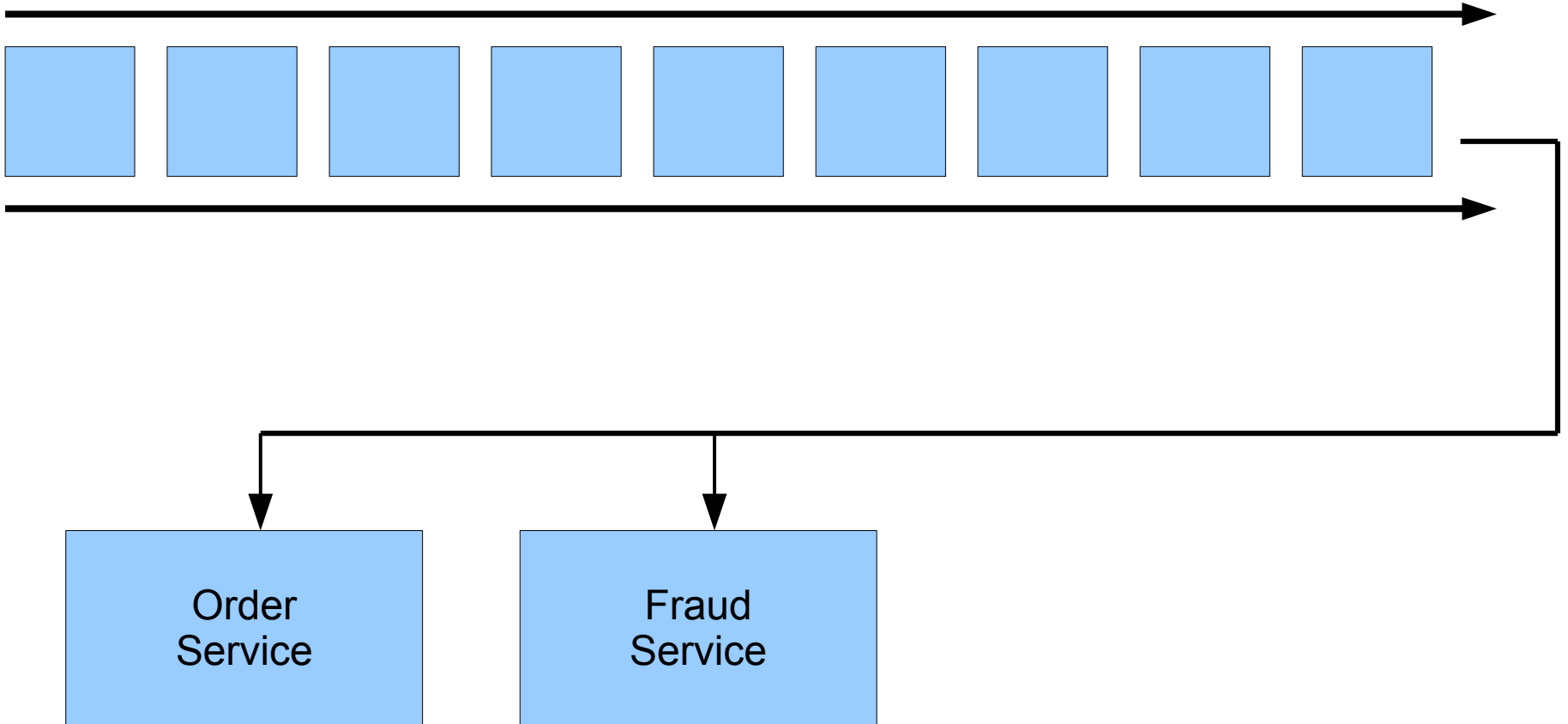
Order Stream

Events



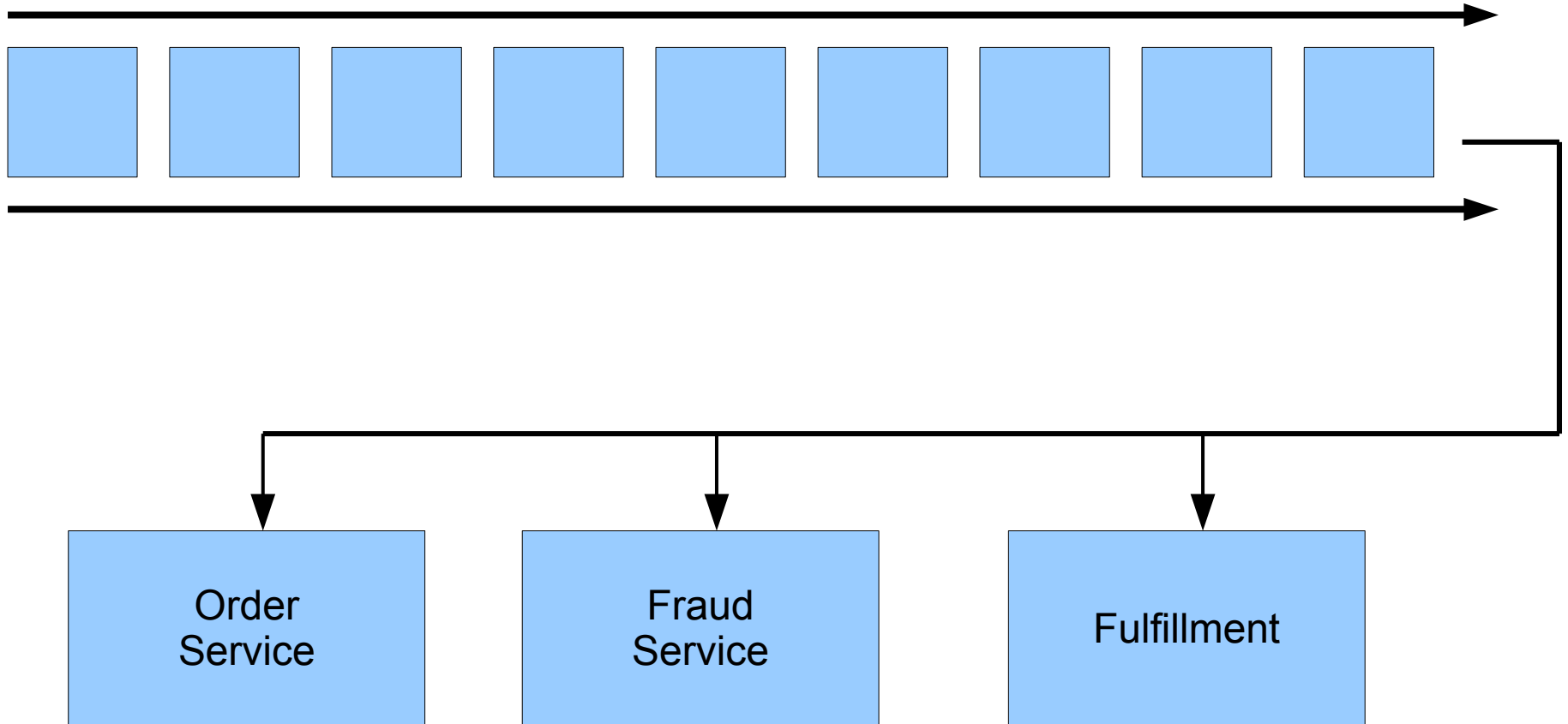
Order Stream

Events



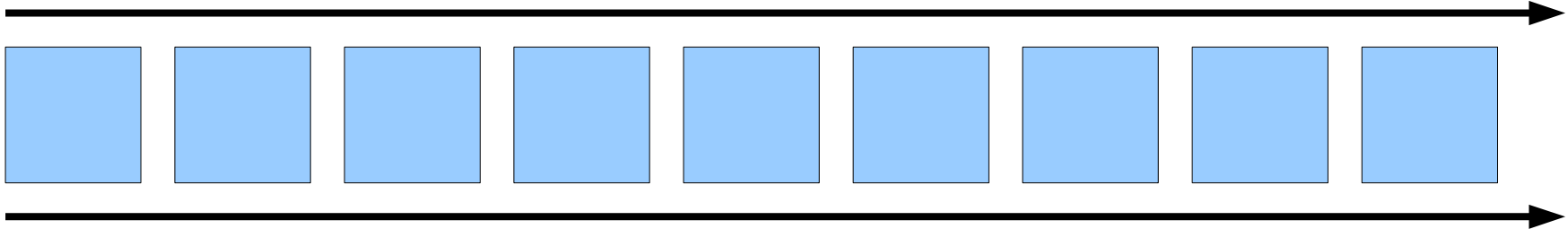
Order Stream

Events

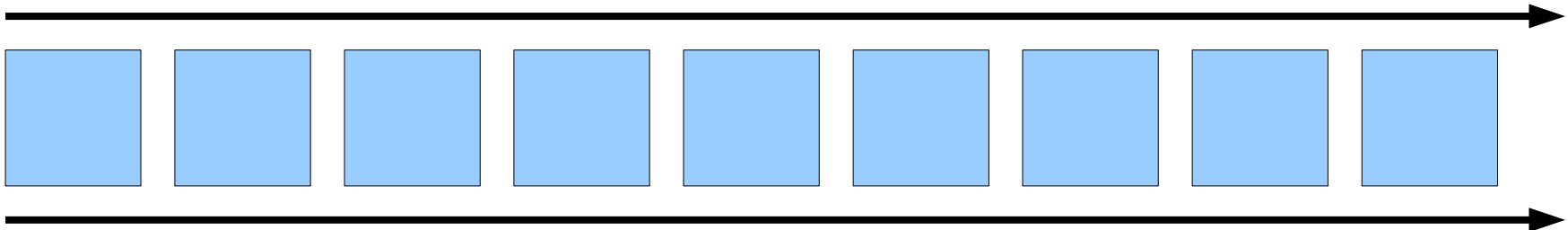


Events

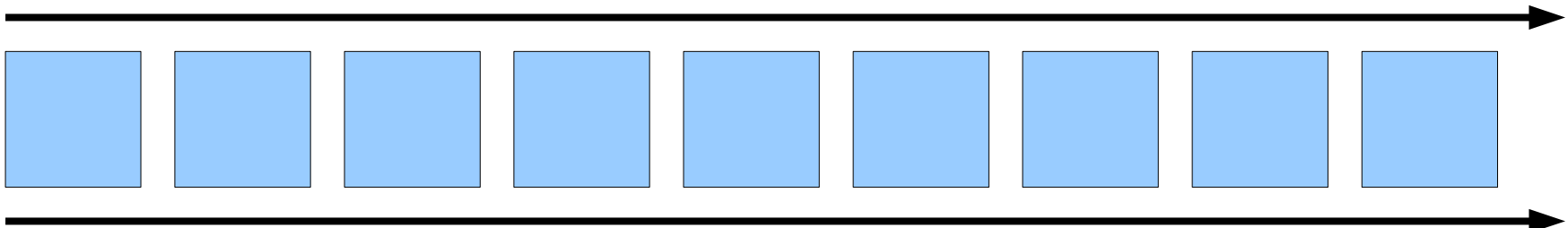
Order Stream



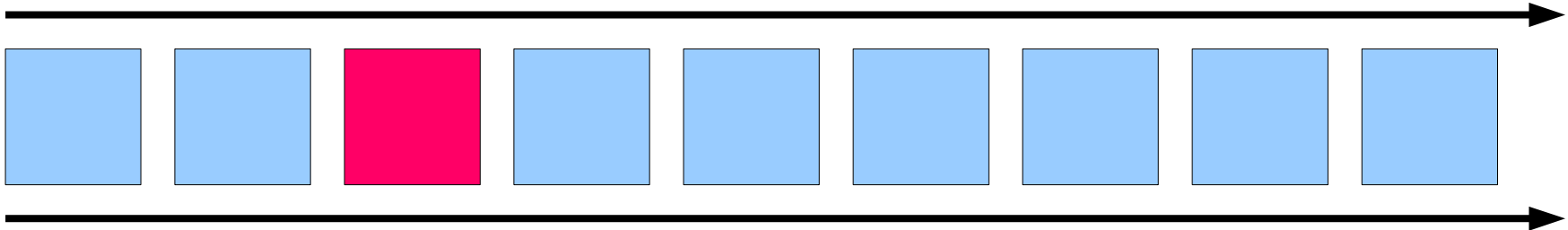
Payment Stream



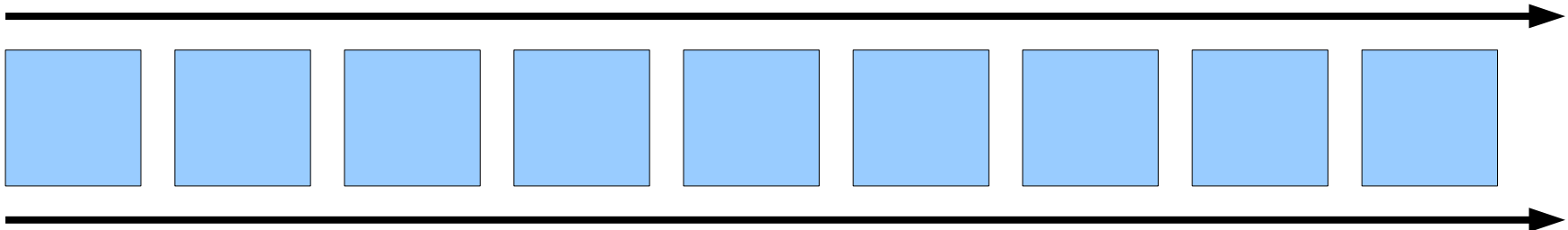
Notification Stream



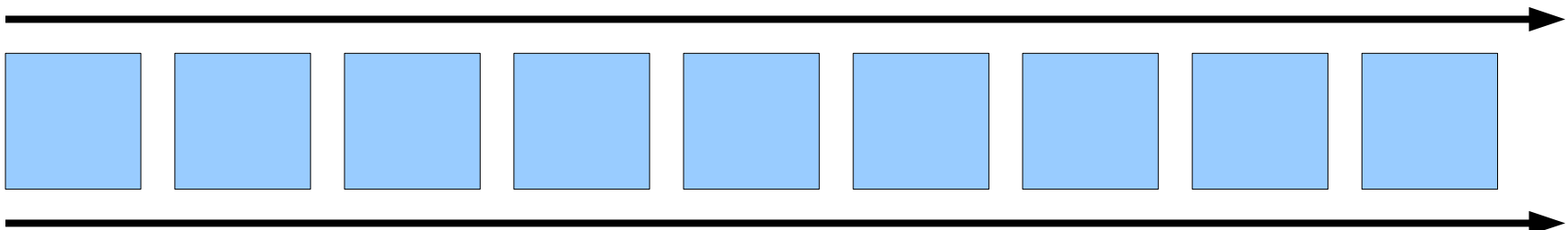
Order Stream



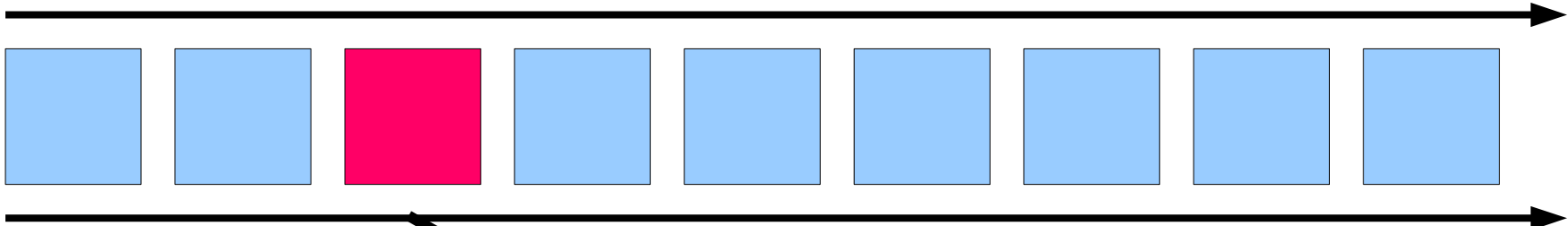
Payment Stream



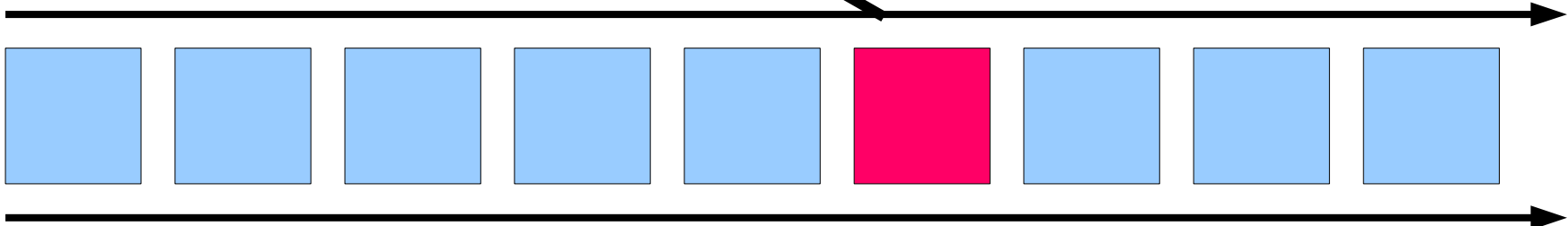
Notification Stream



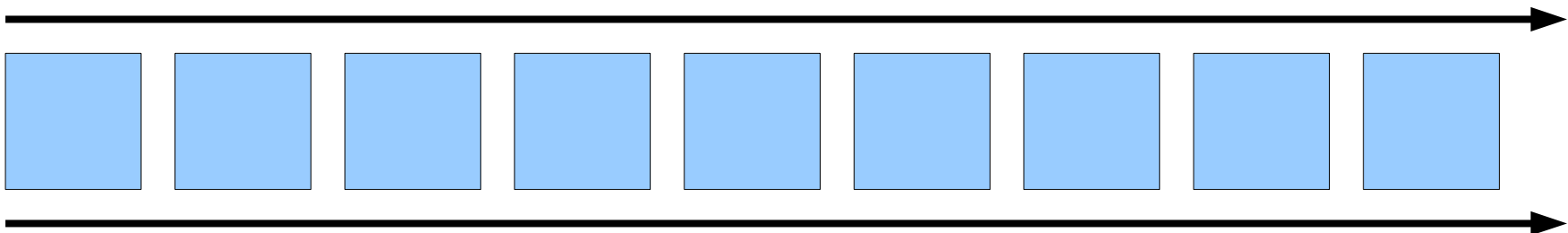
Order Stream



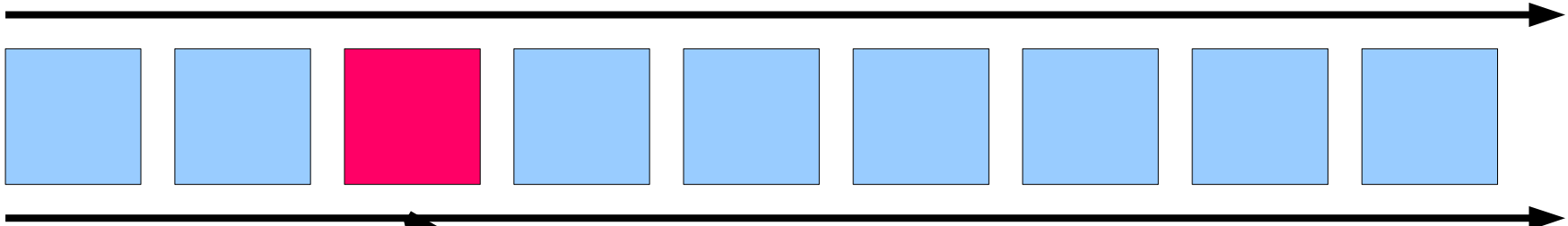
Payment Stream



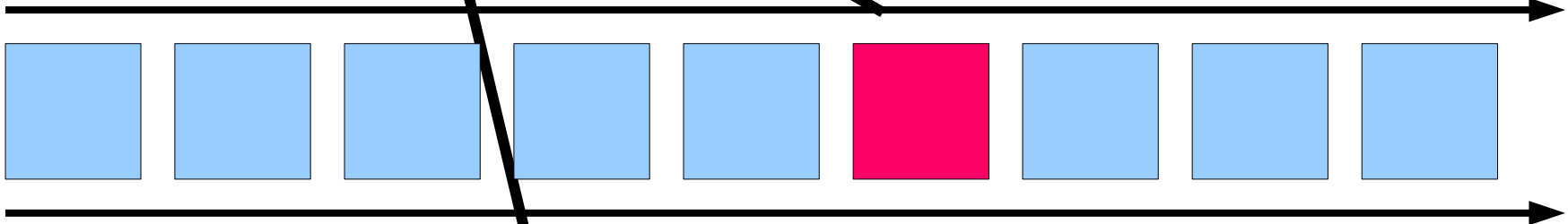
Notification Stream



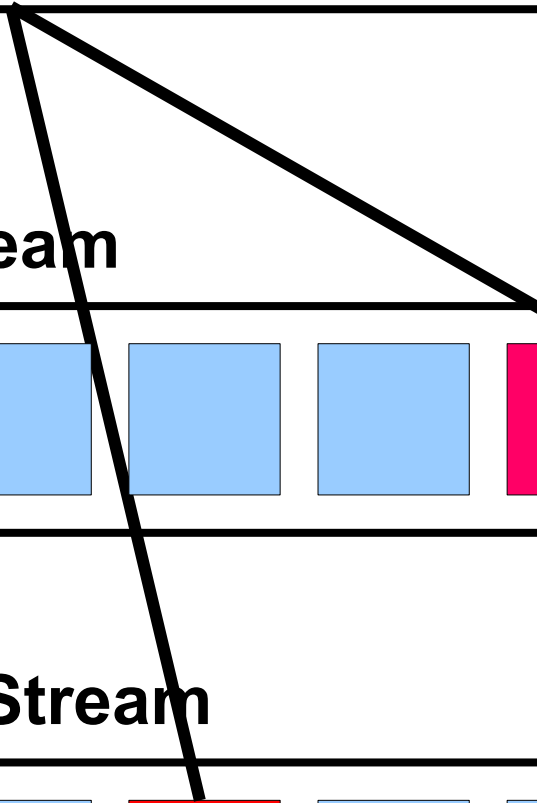
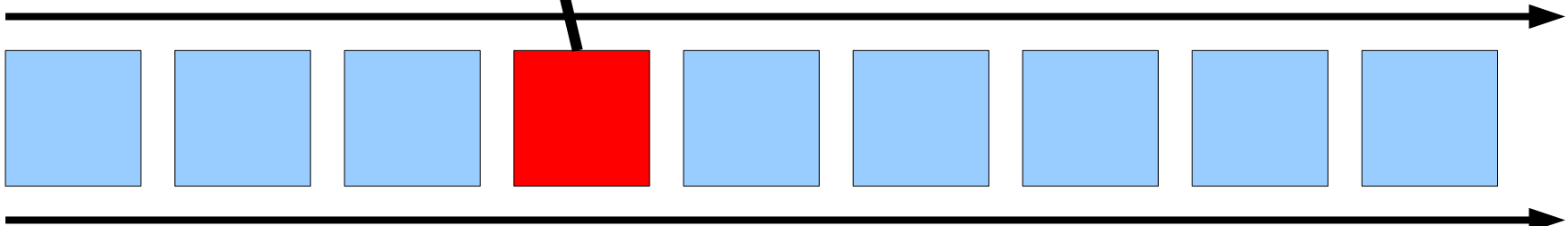
Order Stream



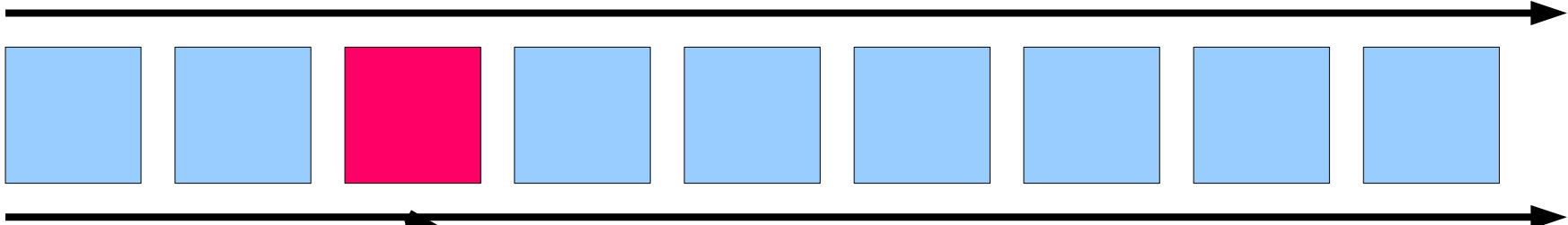
Payment Stream



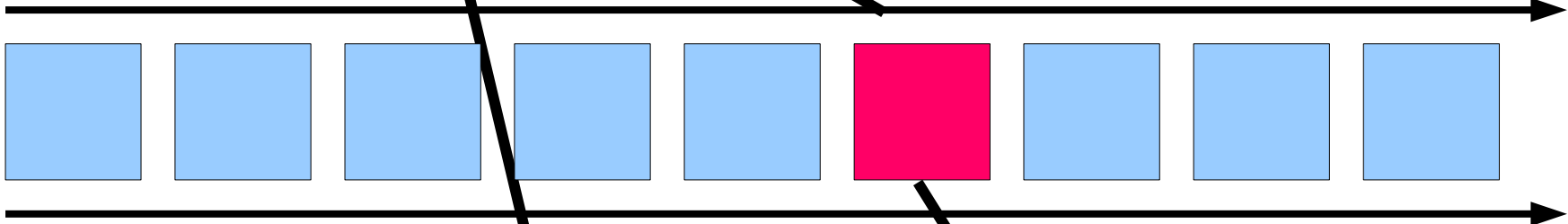
Notification Stream



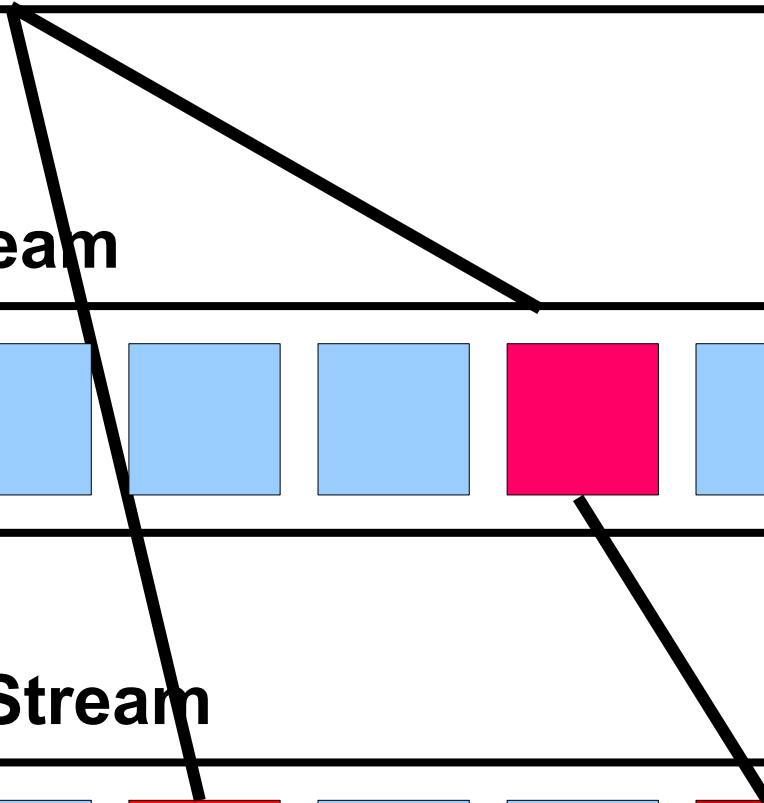
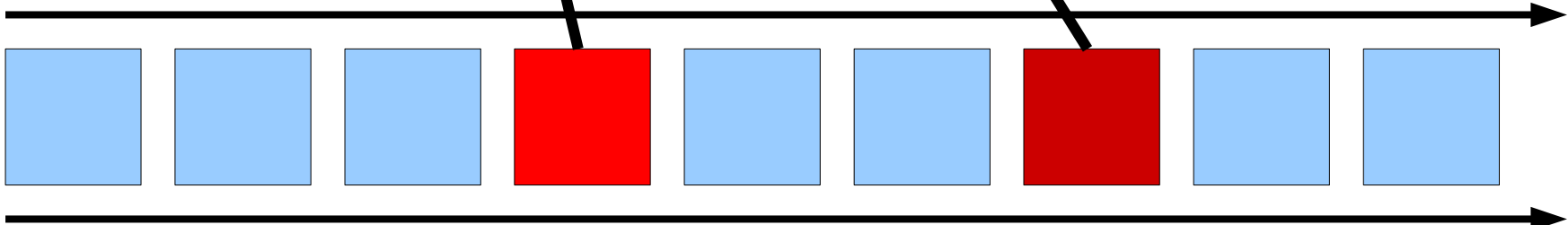
Order Stream

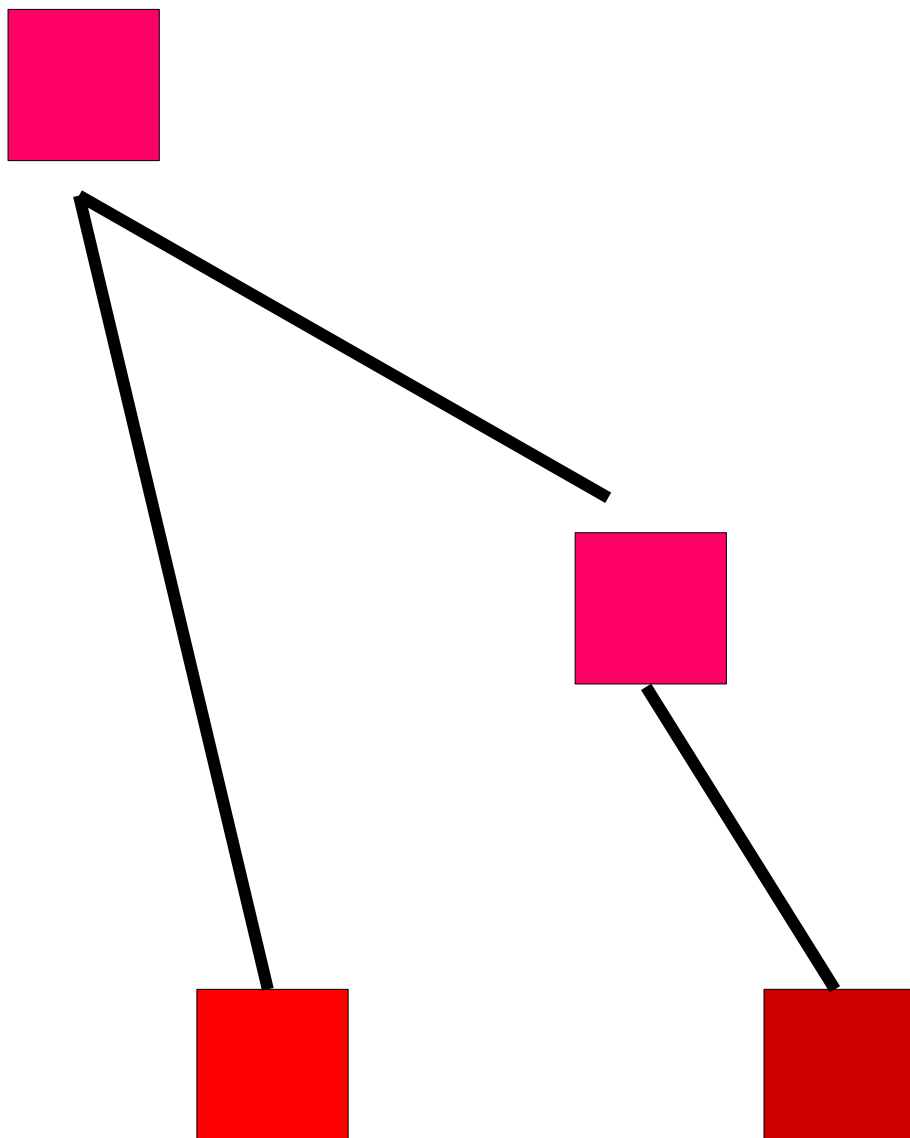


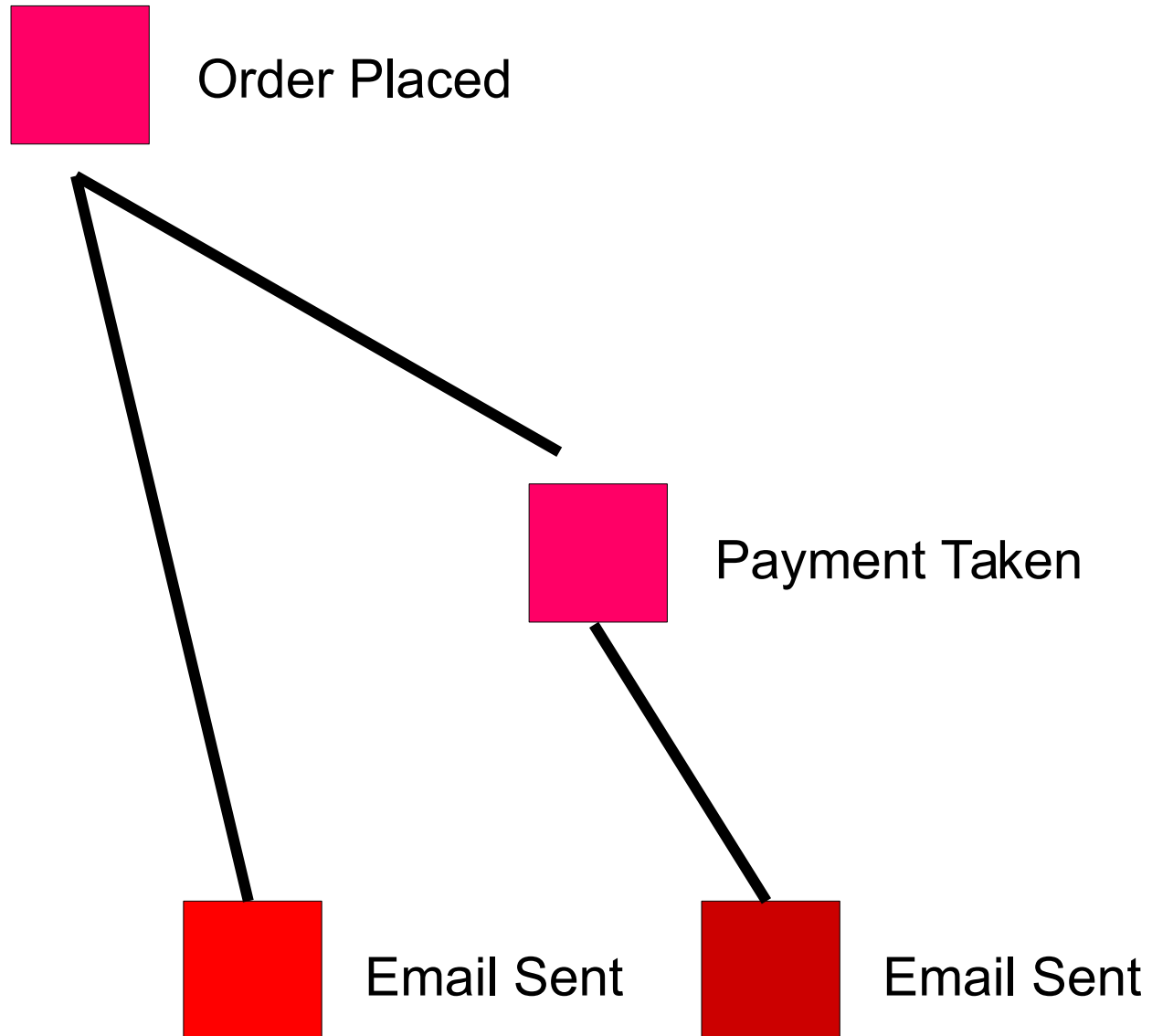
Payment Stream



Notification Stream

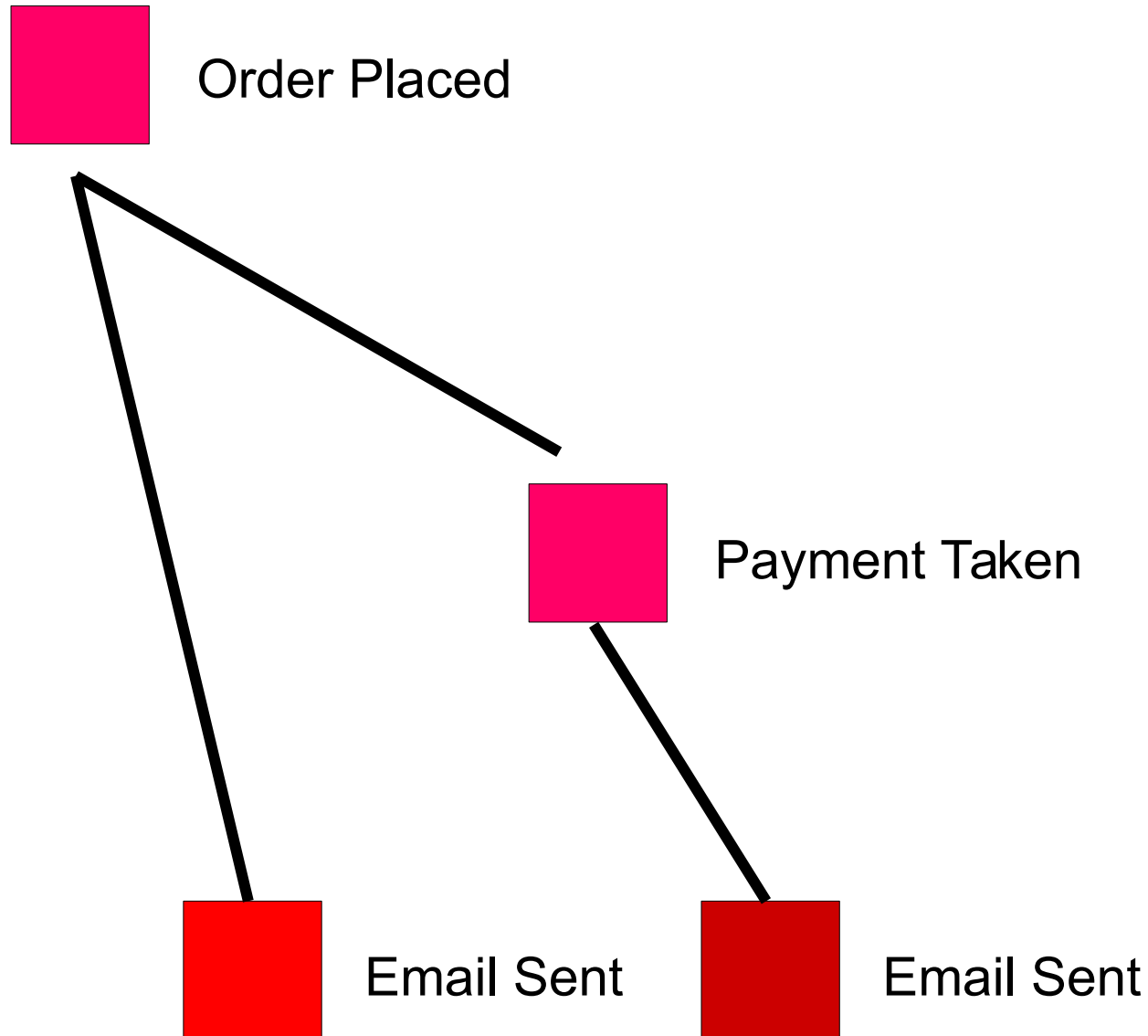




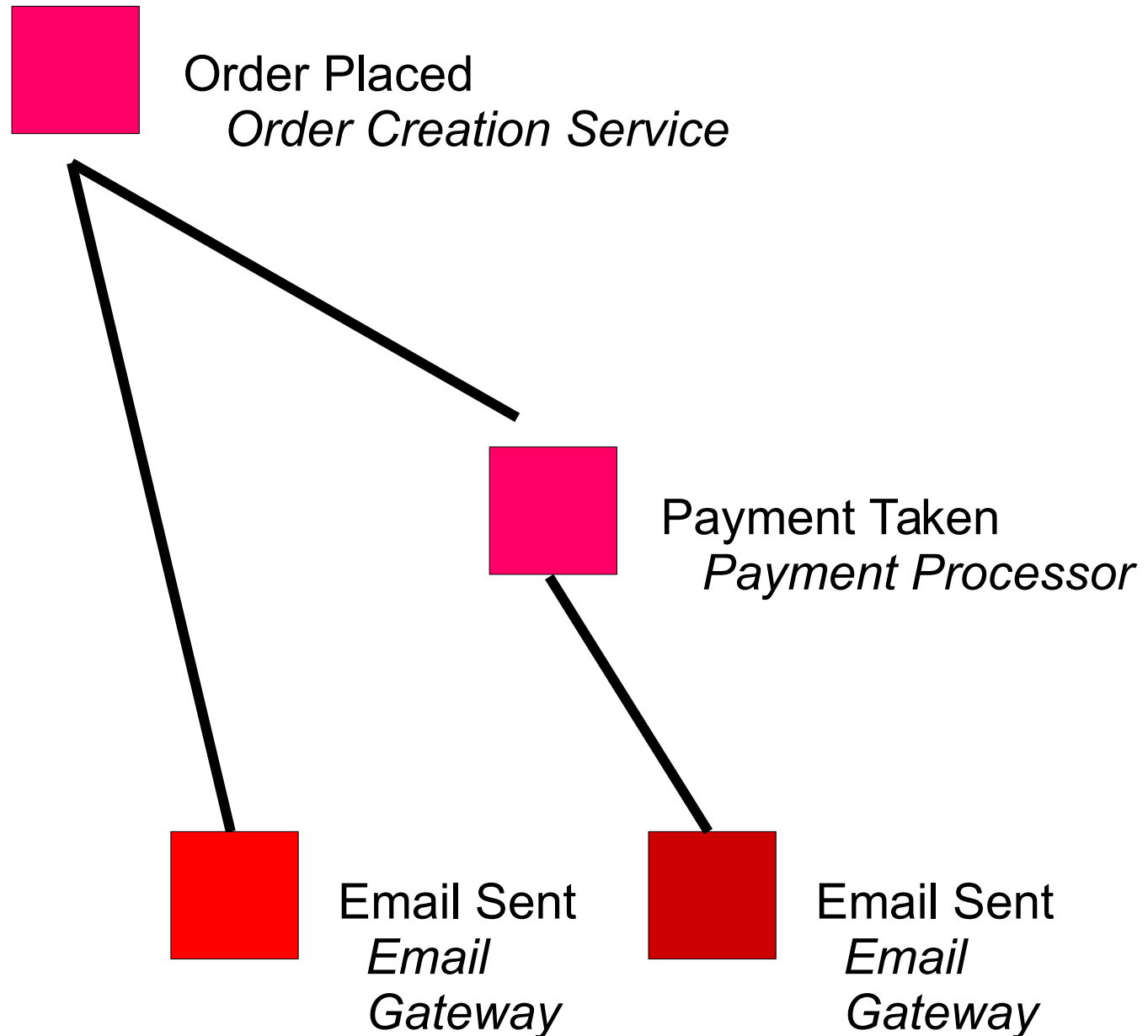


Dependencies

Dependencies

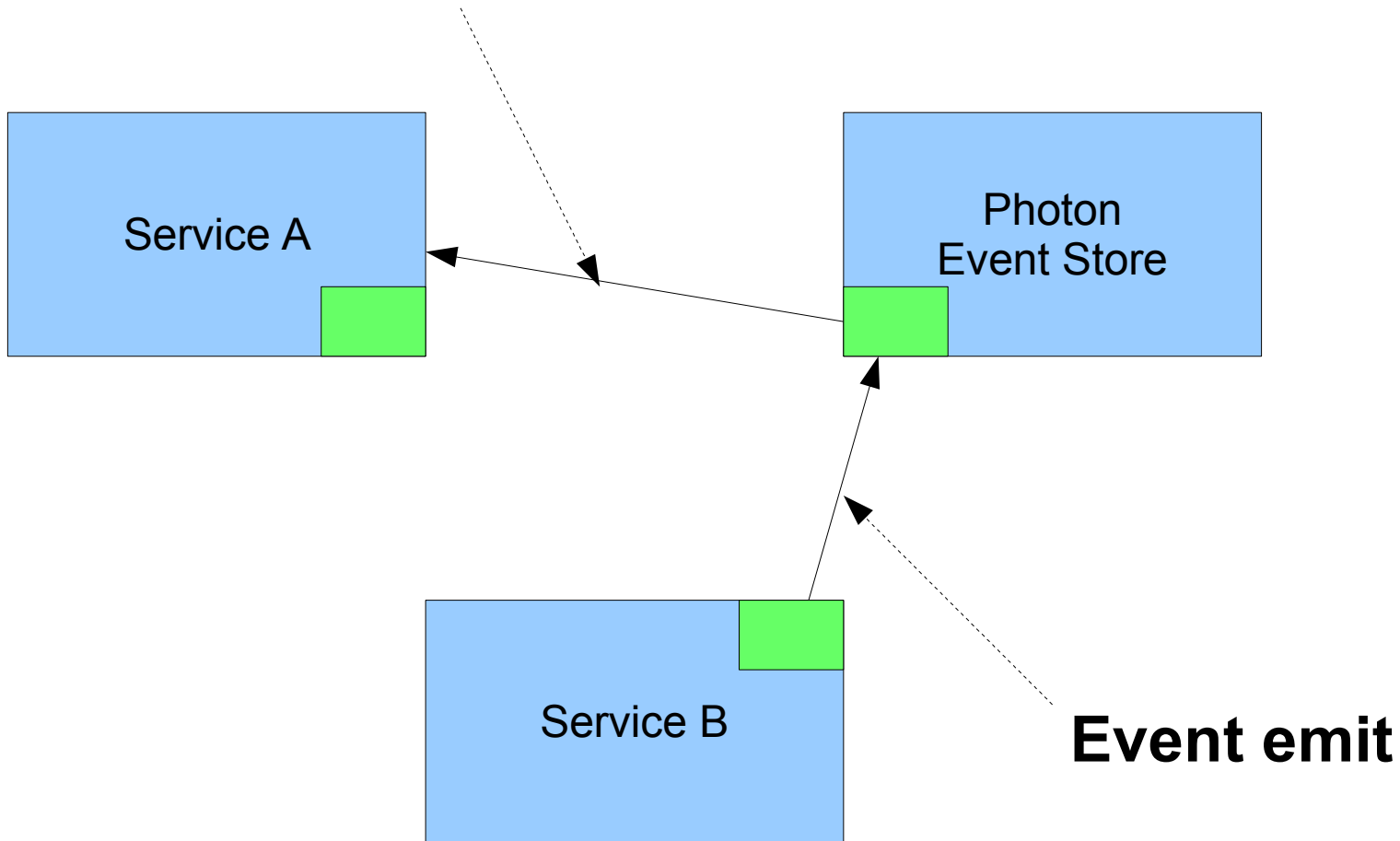


Dependencies



Event Protocols & Photon

Event replay



Demo

3 – Photon/ Events

Newton

Application framework for Domain Driven Design

Muon Event Protocols

+

Photon

+

Language framework (currently Java)

Newton

Application framework for Domain Driven Design

Materialising View from Event Streams

Complex logic using Aggregate Roots

State mutation using Commands

Long running business process using Sagas/
Process Managers

Newton

Application framework for Domain Driven Design

Materialising View from Event Streams

Complex logic using Aggregate Roots

State mutation using Commands

Long running business process using Sagas/
Process Managers

Newton

Application framework for Domain Driven Design

Materialising View from Event Streams

Complex logic using Aggregate Roots

State mutation using Commands

Long running business process using Sagas/
Process Managers

Newton

Application framework for Domain Driven Design

Materialising View from Event Streams

Complex logic using Aggregate Roots

State mutation using Commands

**Long running business process using Sagas/
Process Managers**

Demo

4 – Views in Newton

Demo

5 – Aggregate Roots

Demo

6 – Process Managers

Future plans

- Aeron low latency transport *[likely commercial]*
- Kafka transport
- Muon-Rust/ 'libMuon'. Leading to ...
 - Go, Python, Ruby ...
- Polyglot Newton
- CLI improvements
- Cloud Foundry integration *[likely commercial]*

Next steps

- Run a spike
 - Decompose within a single process?
 - eg JNDI Discovery/ current InMemTransport
- Tell me what you'd like to see
- I'm available to build/ advise
- Commercial options under development