Event-Driven Architecture

from a Scalability Perspective

by

√iktor Klang
Scalable Solutions

About me

- Slightly involved in Akka...
- Background in business software architecture & development
- Recently fully recovered from 9 years of Java
- Jumped on the Scala bandwagon in 2007
- Twitter: @viktorklang

Why is EDA Important for Scalability?

What building blocks can EDA consist of?

Events?

Concepts

Messaging

Publish-Subscribe

Point-to-Point

Store-forward

Request-Reply

Standards

AMQP

JMS

ØMQ (upcoming)

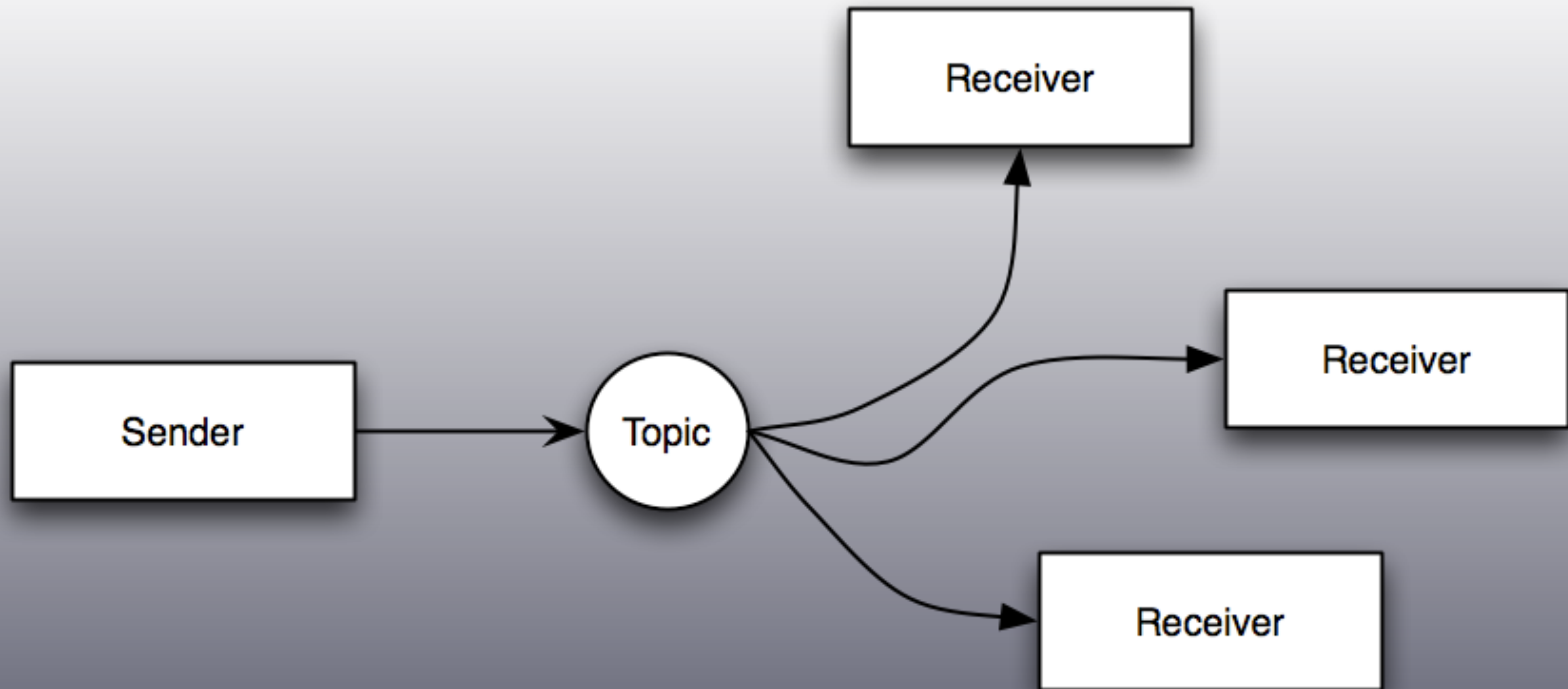
Some Products

...

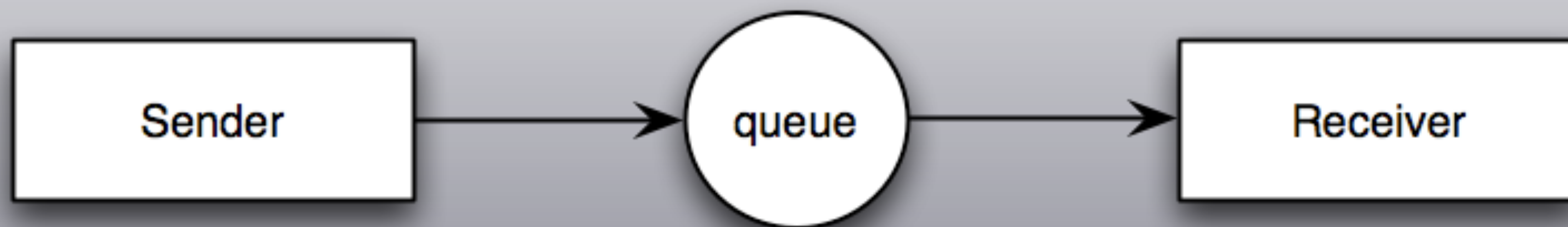
Wire Formats

Java serialization	(binary, schema, runtime)
Protobuf	(binary, schema, compiled)
Avro	(binary, schema, compiled & runtime)
Thrift	(binary, schema, compiled)
MsgPack	(binary, schema, compiled)
Protostuff	(binary, schema, compiled)
Kryo	(binary, schema-less, runtime)
BERT	(binary, schema-less, runtime)
Hessian	(binary, schema-less, compiled)
XML	(text, schema)
JSON	(text, schema-less)

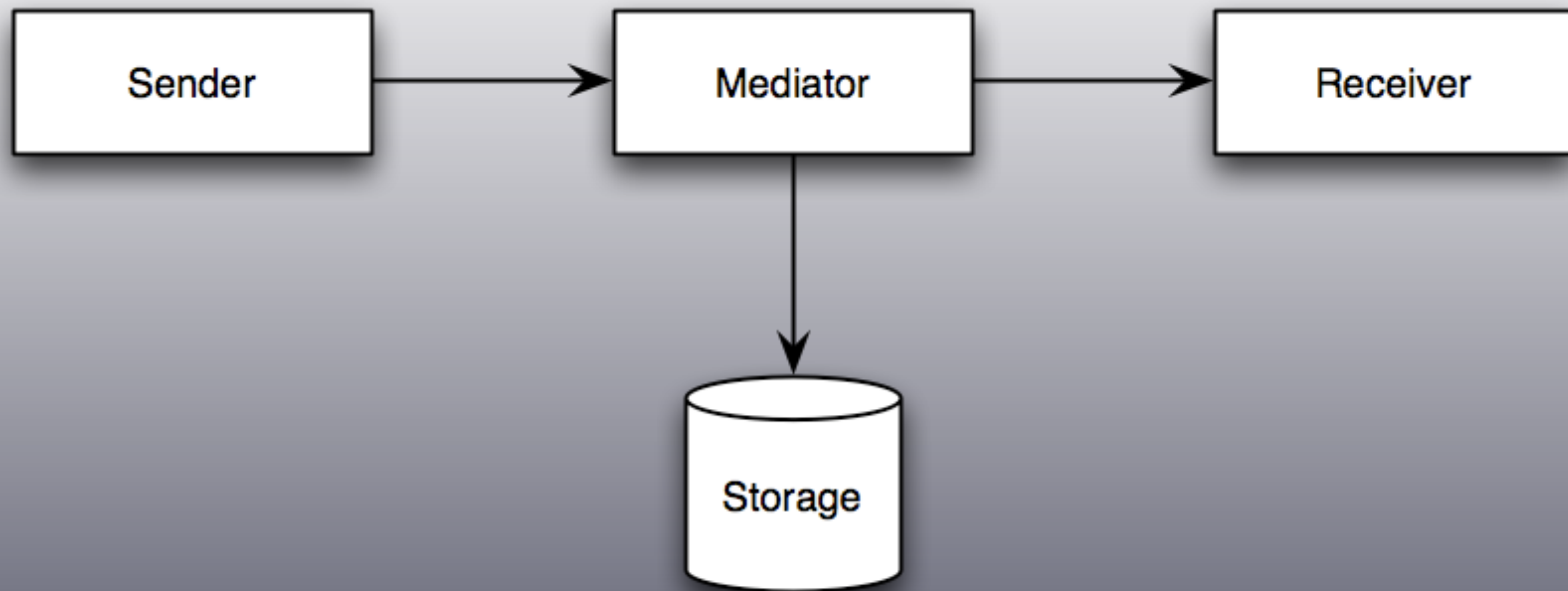
Publish-Subscribe



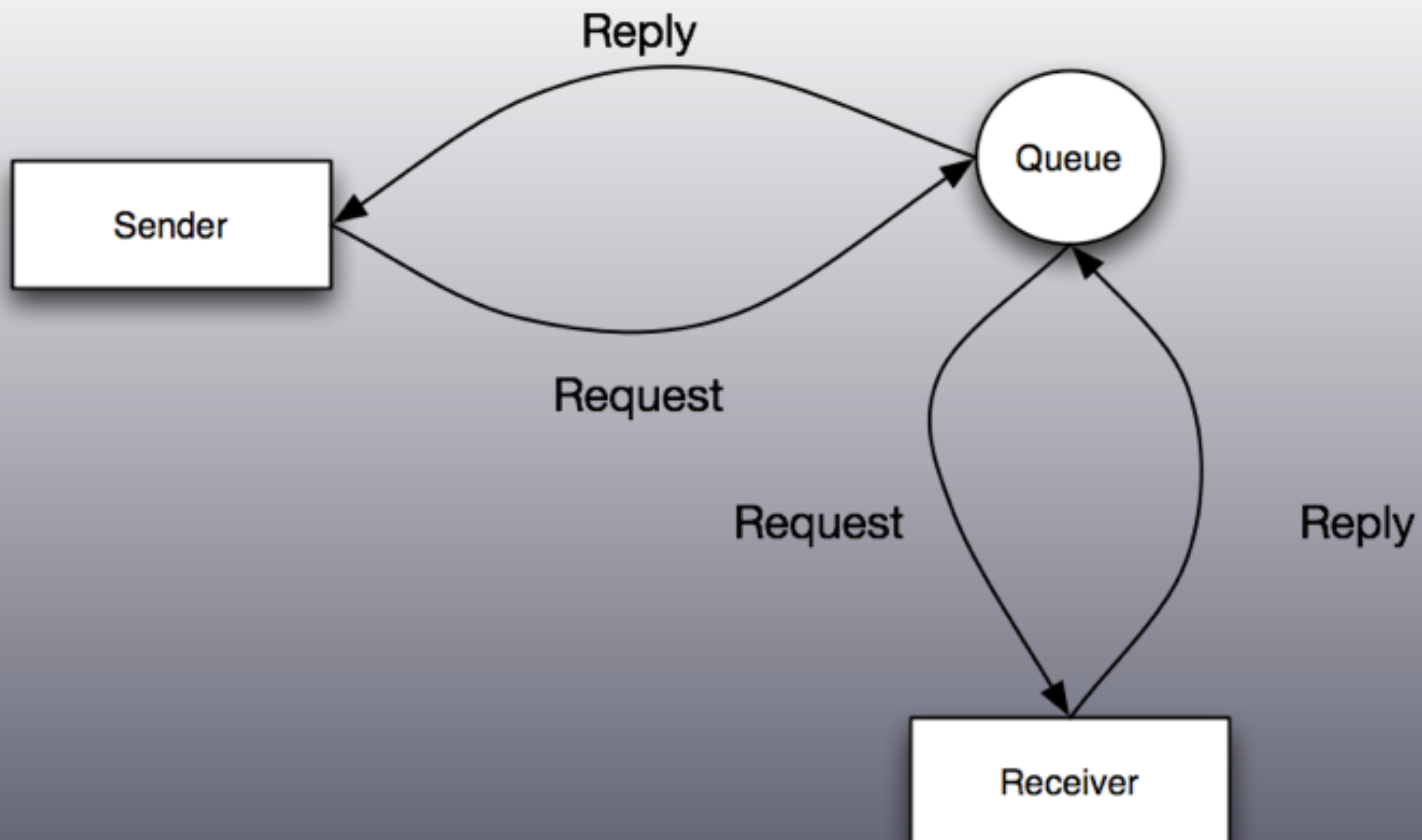
Point-to-Point



Store-Forward



Request-Reply



Queues

Async

Eventual Consistency

Elasticity

Patterns

Domain Events

“It's really become clear to me in the last couple of years that we need a new building block and that is the Domain Events”

-- Eric Evans, 2009

Domain Events

“State transitions are an important part of our problem space and should be modeled within our domain.”

-- Greg Young, 2008

Domain Events

Uniquely identifiable

Self contained

Observable

Time relevant

Command and Query Responsibility Segregation

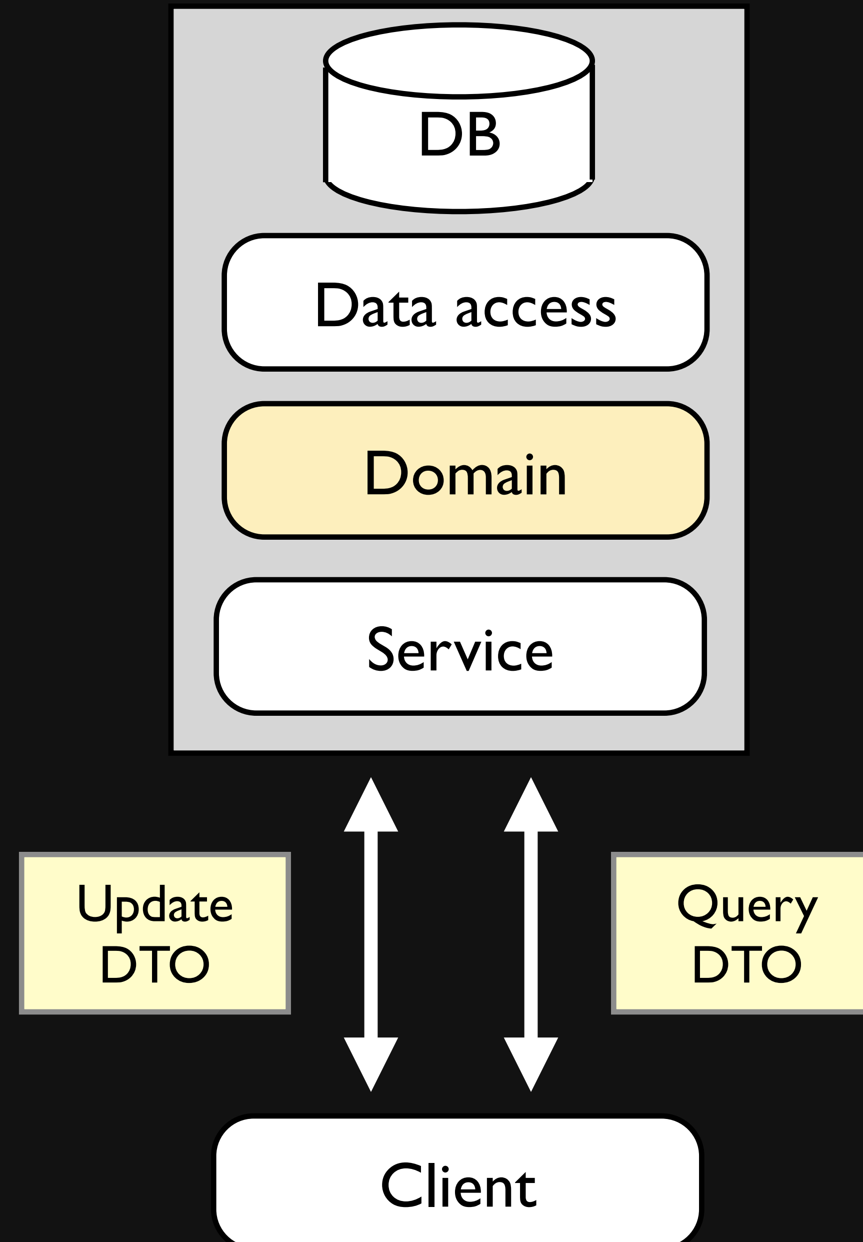
"A single model cannot be appropriate for reporting, searching and transactional behavior."

-- Greg Young, 2008

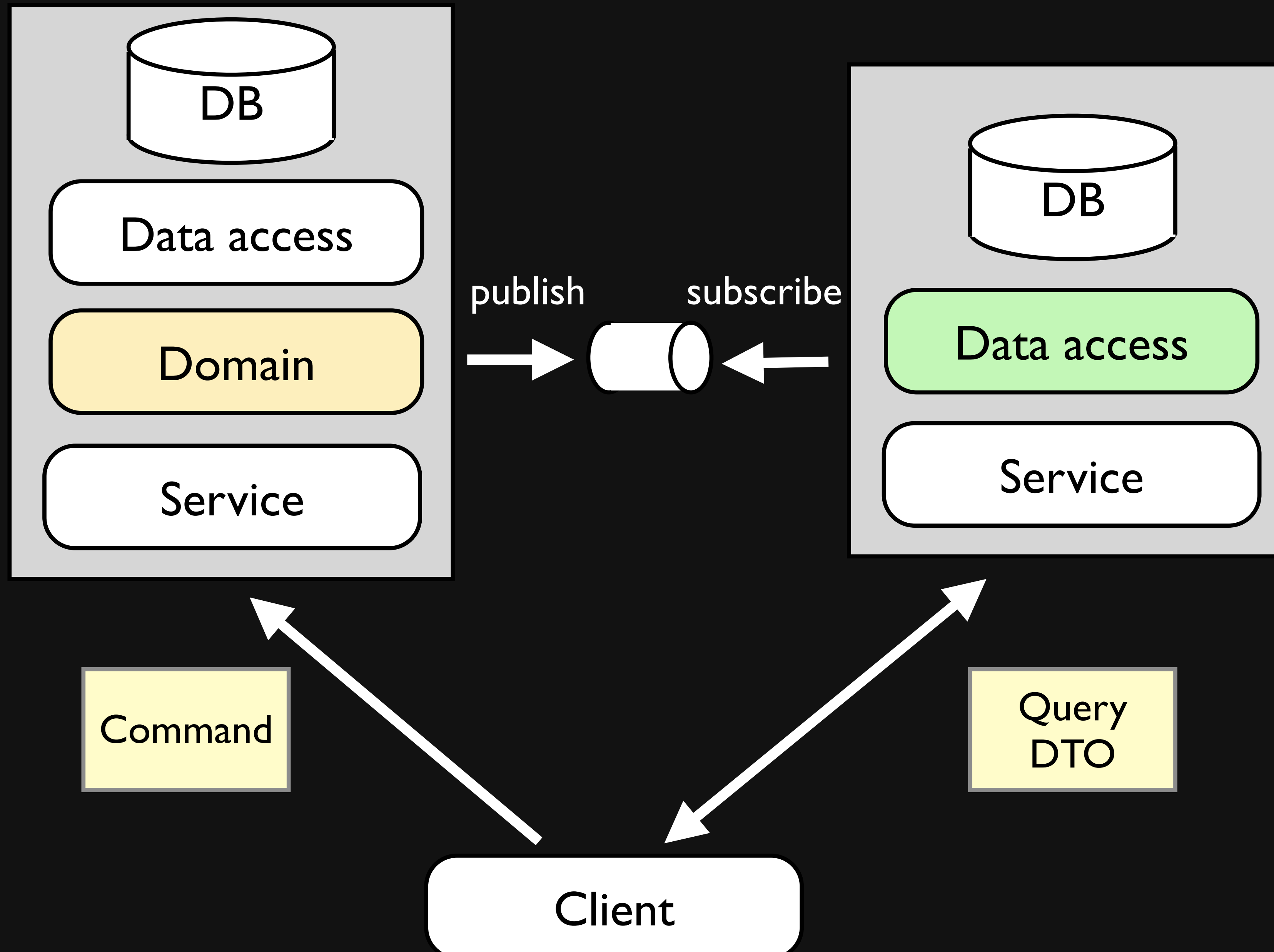
CQRS

- Aggregate roots receive Commands and publish Events
- All state changes are represented by Domain Events
- Reporting module is updated as a result of the published Events
- All Queries go directly to the Reporting, the Domain is not involved

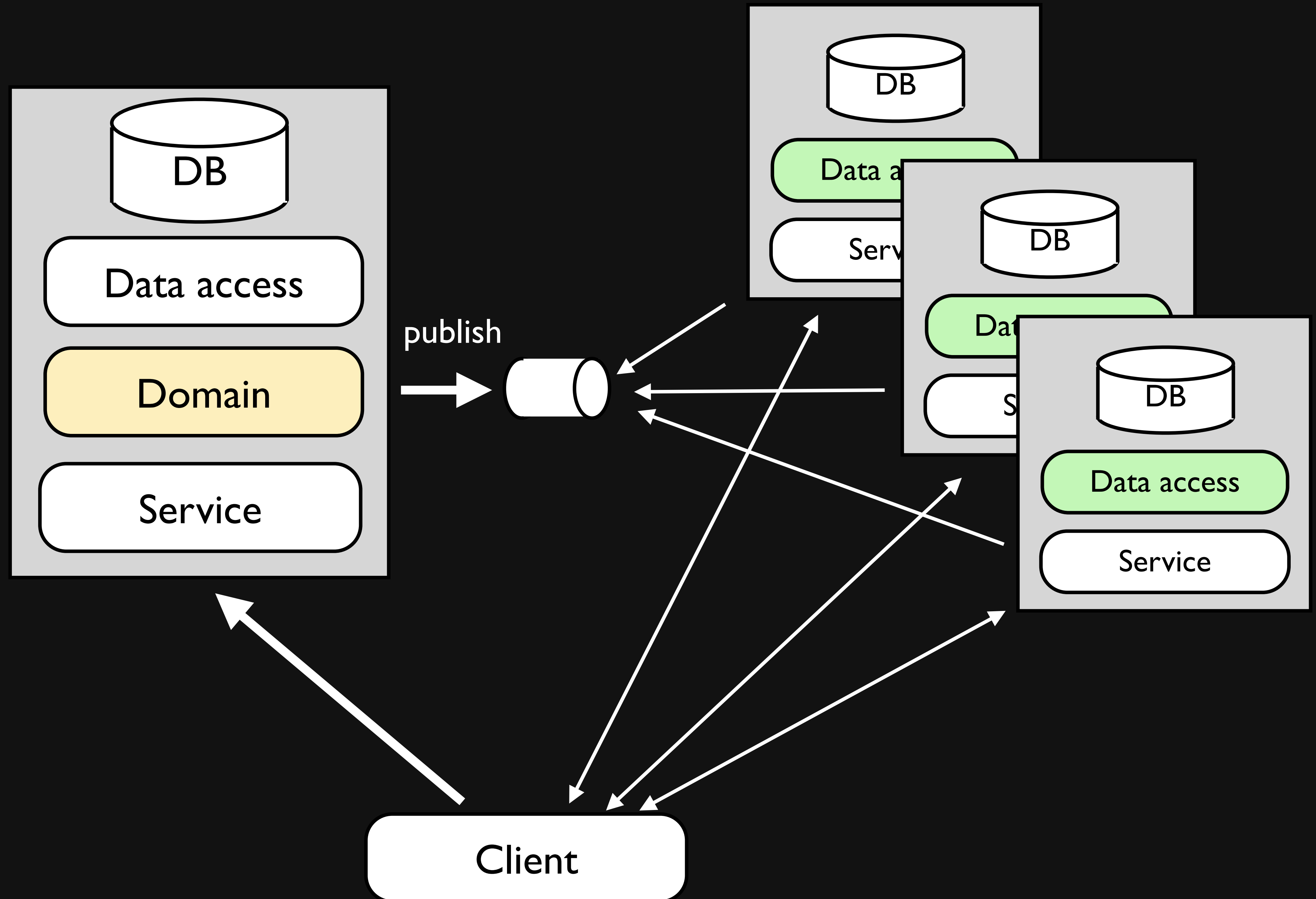
The traditional way...



The CQRS way...



The CQRS way...



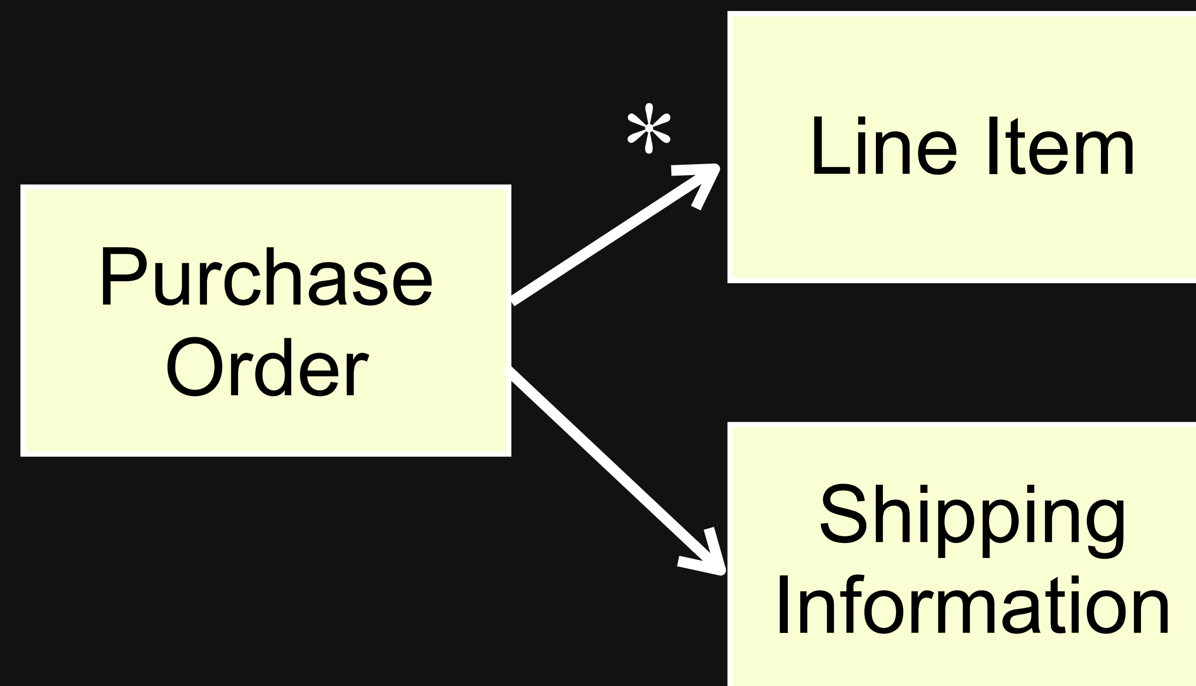
CQRS Benefits

- Separation of concern
- Fully encapsulated domain that only exposes behavior
- Queries do not use the domain model
- Easy integration with external systems
- Performance and scalability
- Testability

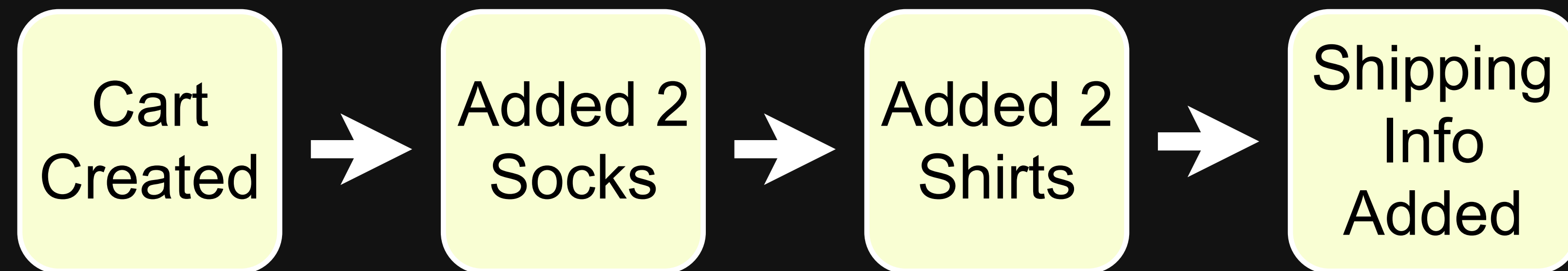
Event Sourcing

- Every state change is materialized in an Event
- All events are stored in an Event Log
- System can be reset and Event Log replayed
- Many different Listeners can be added

Storing Structure



Event Sourcing - Storing Deltas



Aggregates are tracking events as to what has changed within them

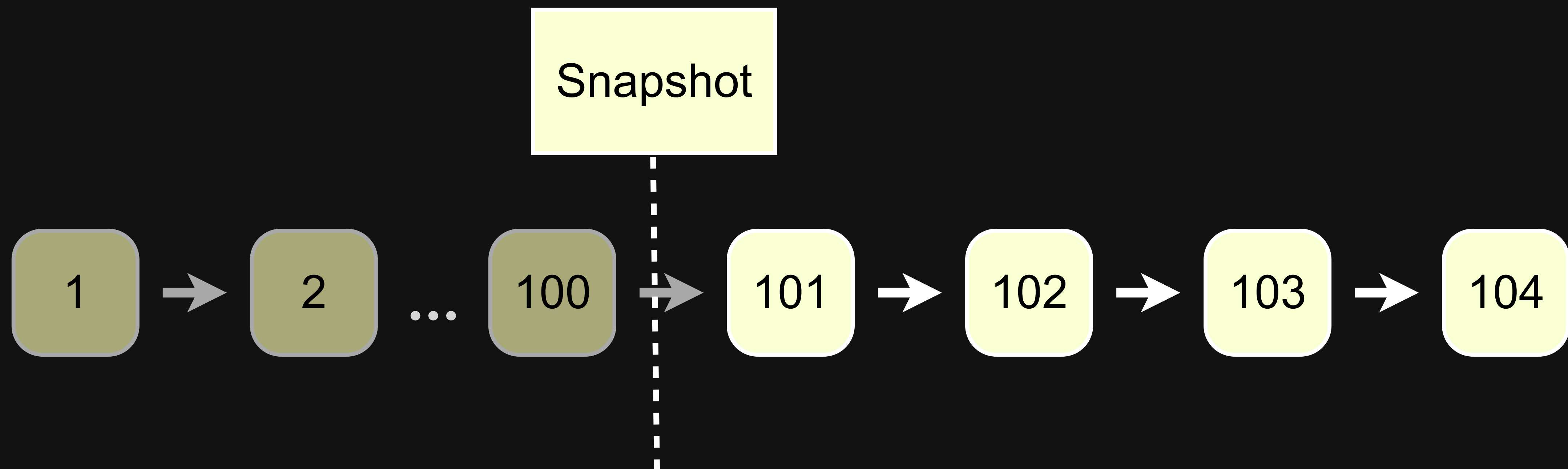
Current state is constructed by replaying all events

Data is not persisted in a structure but as a series of transactions

No ORM is needed

Event Sourcing - Replaying Events

Event Sourcing - Rolling Snapshot



Event Sourcing - Benefits

- No object-relational impedance mismatch
- Bullet-proof auditing and historical tracing
- Support future ways of looking at data
- Performance and scalability
- Testability
- Reconstruct production scenarios

Why is EDA Important for Scalability?

- Scale out and up
- Load balance
- Parallel execution
- Non-blocking
- Loosely coupled components can scale more independent of each other

https://gist.github.com/805973

for listening

extra material

Challenges

Clustering of Brokers

ActiveMQ

- Master-Slave
- Store and Forward Network of Brokers

RabbitMQ

- Cluster of Erlang nodes

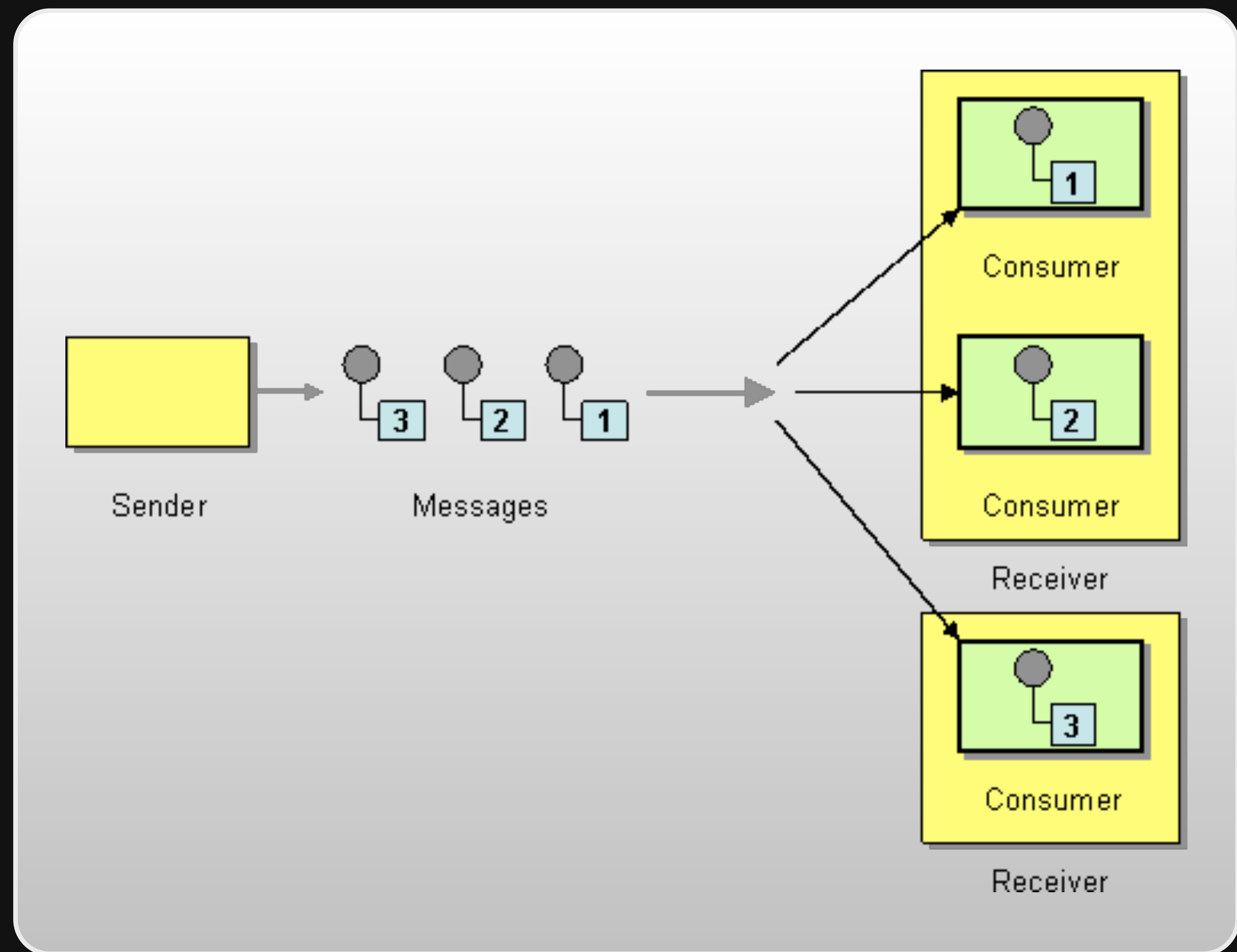
ZeroMQ

- Brokerless - point-to-point or pub-sub

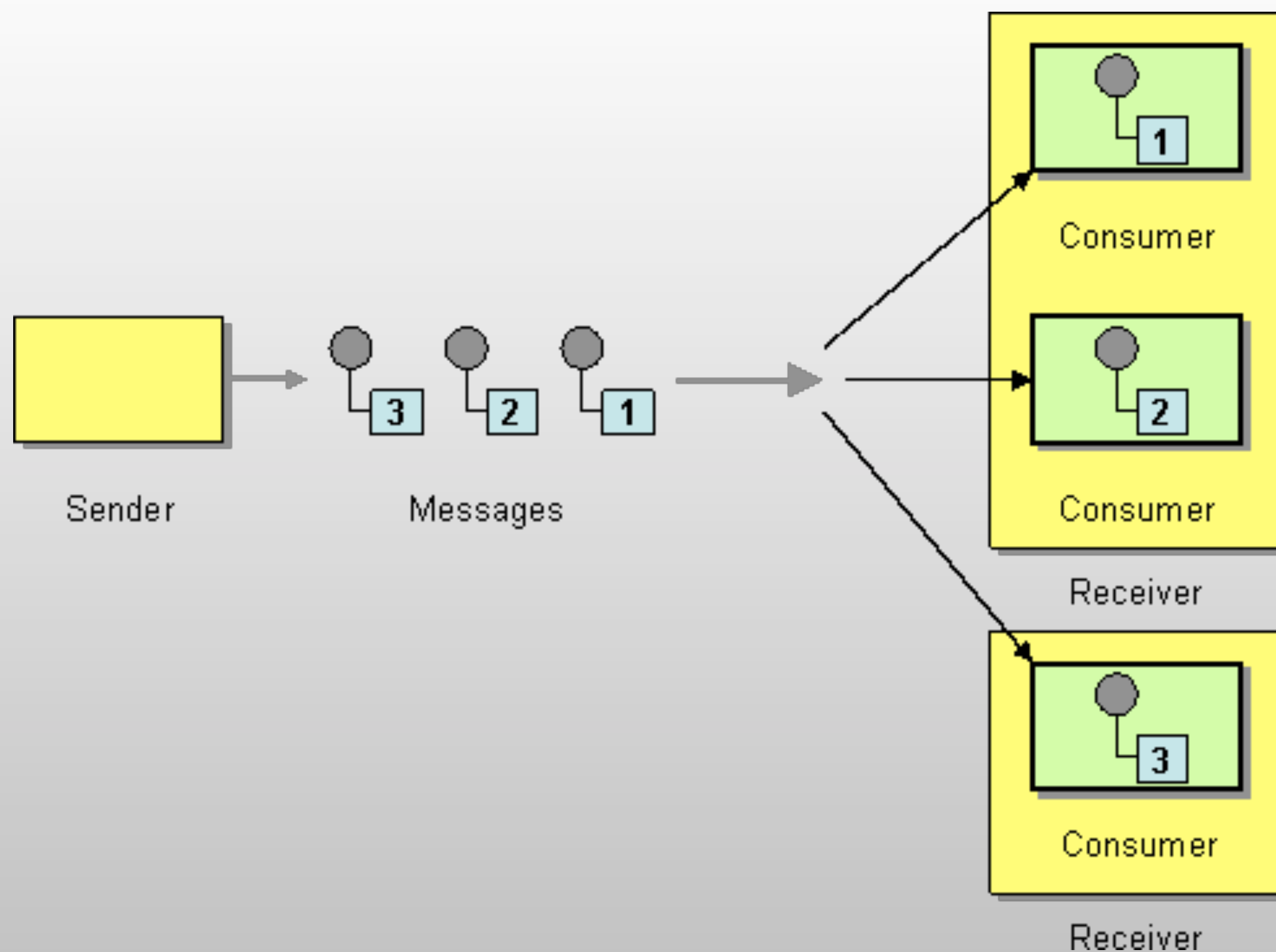
Guaranteed Delivery

Do I really need it?

Persistence increases reliability at the expense of performance



Competing Consumers



Pattern for solving:

- Load balancing
- Concurrency
- Failover

Only works with
Point-to-Point Channel

Challenge

- ordering
- duplicates (idempotent receiver)

Duplicate Messages

What do I need?

- Once-and-only-once
- At-least-once
- At-most-once

QOS

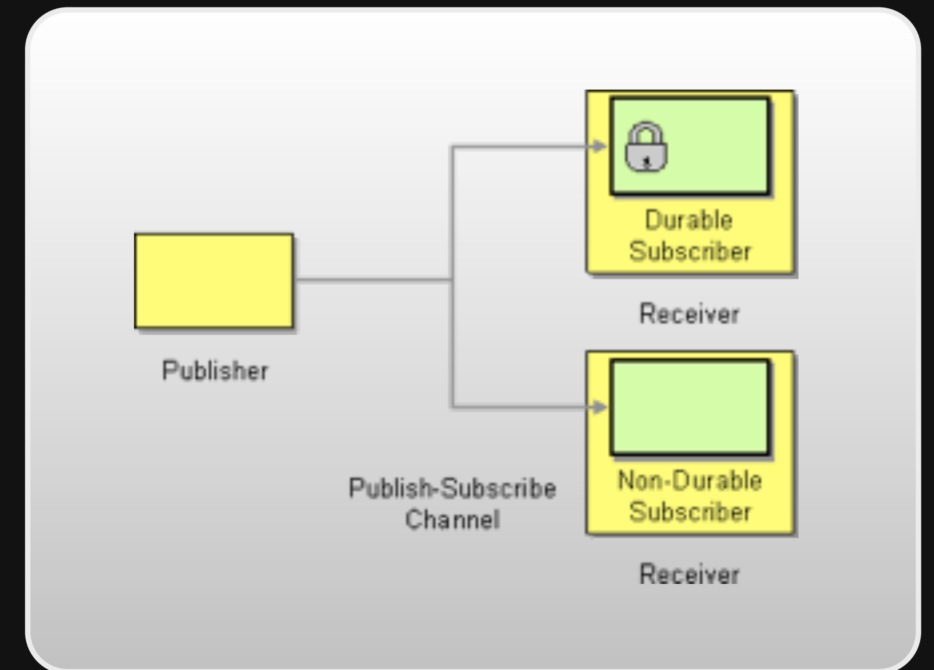
keep history of processed ids

Unique message identifier

Business semantics

How to get back on track?

Point-to-point: no problem, just make the queue persistent



Pub/sub: well, not so straight forward

Problem: only active subscribers

Solution: durable subscriber

Problem: failover and load balancing

Producer Flow Control

What to do when producers flood you with messages?

Running low on broker resources, slow consumers

Graceful degradation

- caller run (in process only)
- block
- abort
- discard

for listening