Messaging Standards and Systems

AMQP & RabbitMQ

Gavin M. Roy
VP of Architecture
AWeber Communications
Twitter: @Crad

POSSCON
April 14, 2015

About Me

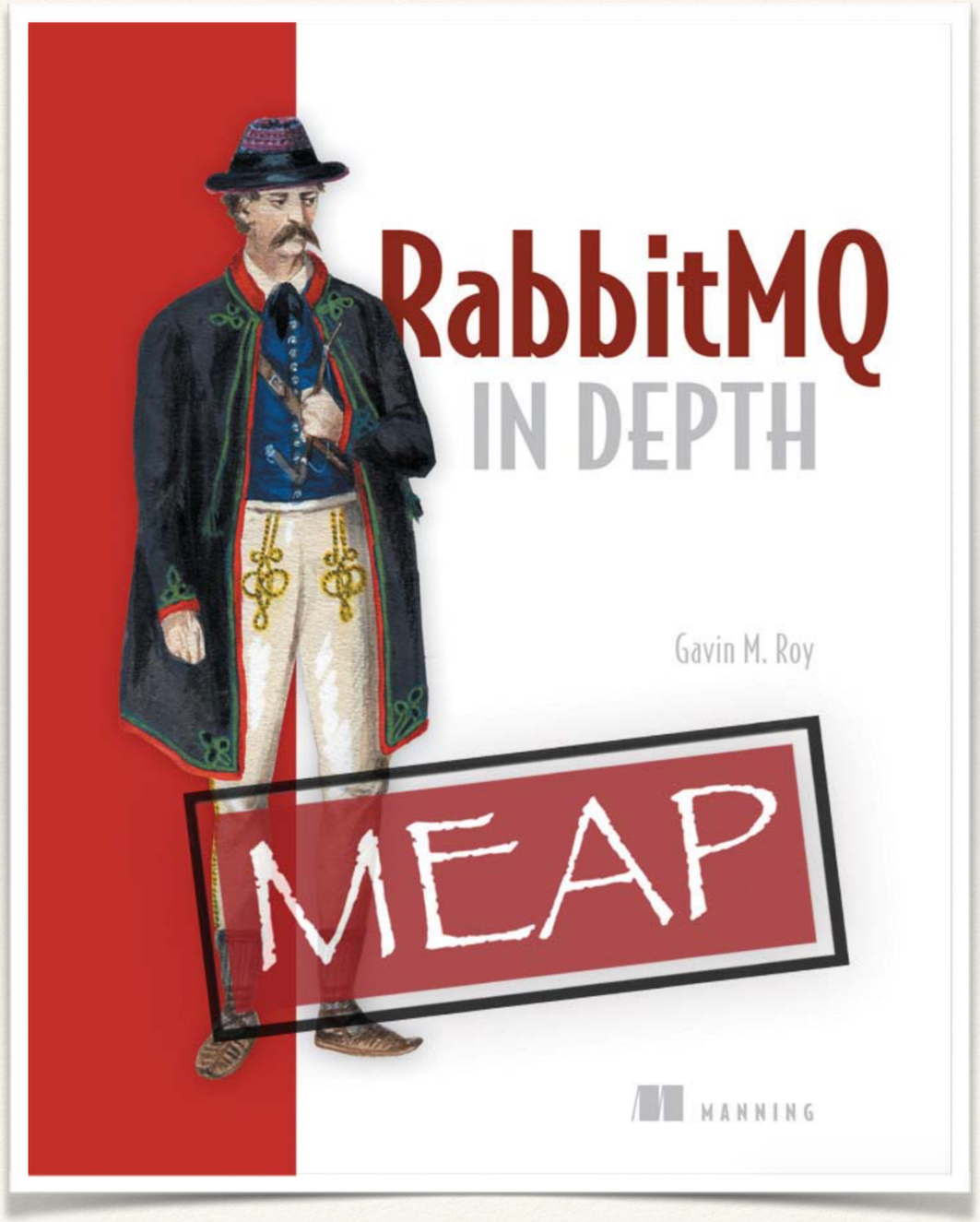
VP of Architecture
AWeber Communications

Blame me for pika, rabbitpy, pamqp, and a handful of RabbitMQ plugins

Blog: https://gavinroy.com

Github: https://github.com/gmr

Book: http://manning.com/roy



Advanced Message Queueing Protocol

- Open Standard
- Platform and Vendor Neutral
- Multiple Versions

AMQP Timeline

2003

Initial Specification

2005

Working Group formed

2006

AMQP 0-8

2008

AMQP 0-9-1

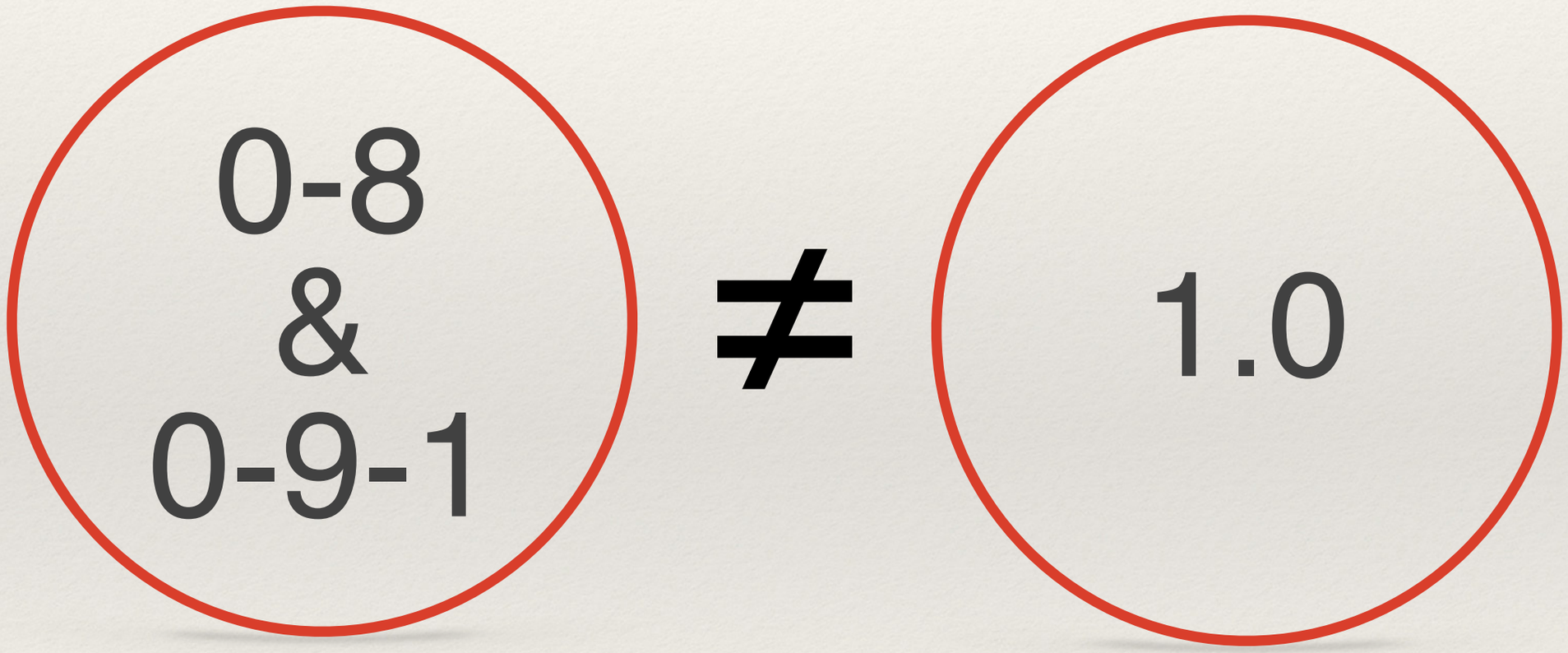
2011

OASIS AMQP 1.0

AMQP Working Group

- JPMorgan Chase
- Cisco Systems
- IONA Technologies
- iMatrix
- RedHat
- TWIST
- Bank of America
- Barclays
- Credit Suisse
- Deutsche Börse Systems
- Goldman Sachs
- HCL Technologies
- Progress Software
- IIT Software
- INETCO Systems Ltd.
- Informatica Corporation
- Microsoft Corporation
- my-Channels
- Novell
- Solace Systems
- Tervala, Inc.
- VMWare
- WSO_2

Competing AMQP Standards



AMQP 0-9-1

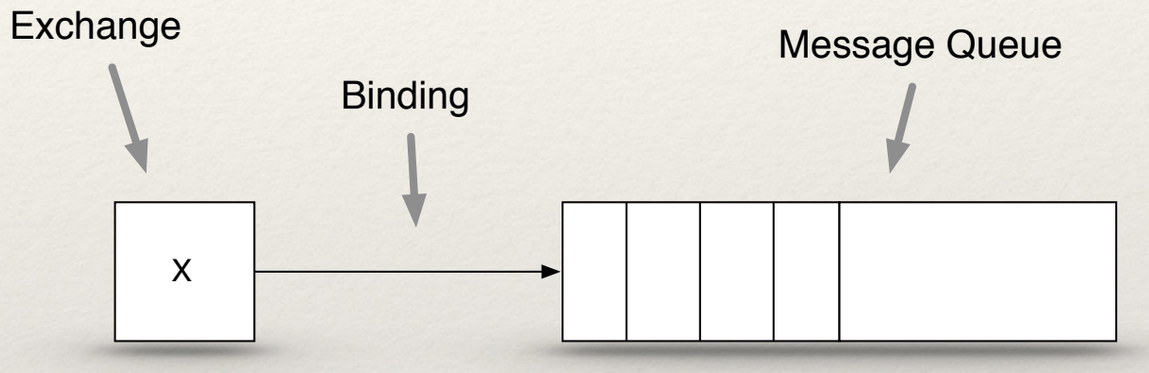
- Currently has wider support than AMQP 1.0
- Multiple Broker Implementations:
  RabbitMQ, Apache Qpid, and SwiftMQ to name a few
- Specifies a Model and Protocol

Native AMQP 0-8/0-9-1 Clients

C	Go	OCaml
Clojure	Groovy	Perl
Cobol	Haskell	PHP
Common Lisp	Java	Python
Delphi	JavaScript	Ruby
Erlang	.NET	Scala

Advanced Message Queueing Model

- **Exchange**
  Receives and route messages
- **Message Queue**
  Stores messages until they can be consumed
- **Binding**
  Defines the relationship between an Exchange and Queue and provides routing criteria



Routing Keys

- Provided when publishing a message
- Compared against binding keys by exchanges
- Ideally provide context to the message:
  - *Connote the type of the message*
  - *Categorize the content in the message*
  - *Specify the type of consumer that should receive it*

Advanced Message Queueing Protocol

- Compact, binary frame format wire protocol
- Bi-directional RPC
- Commands consist of Classes and Methods:
  - *Example Request:*

    ```
    Queue.Declare(name="foo")
    ```

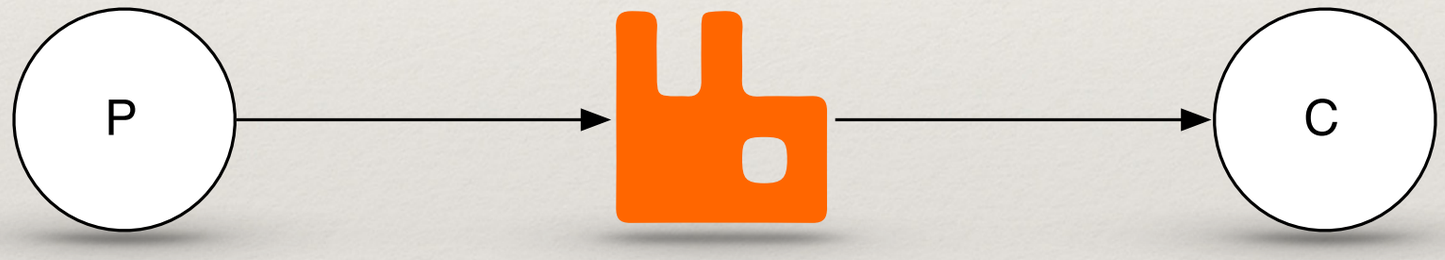
  - *Response:*

    ```
    Queue.DeclareOk(messages=0, consumers=0)
    ```

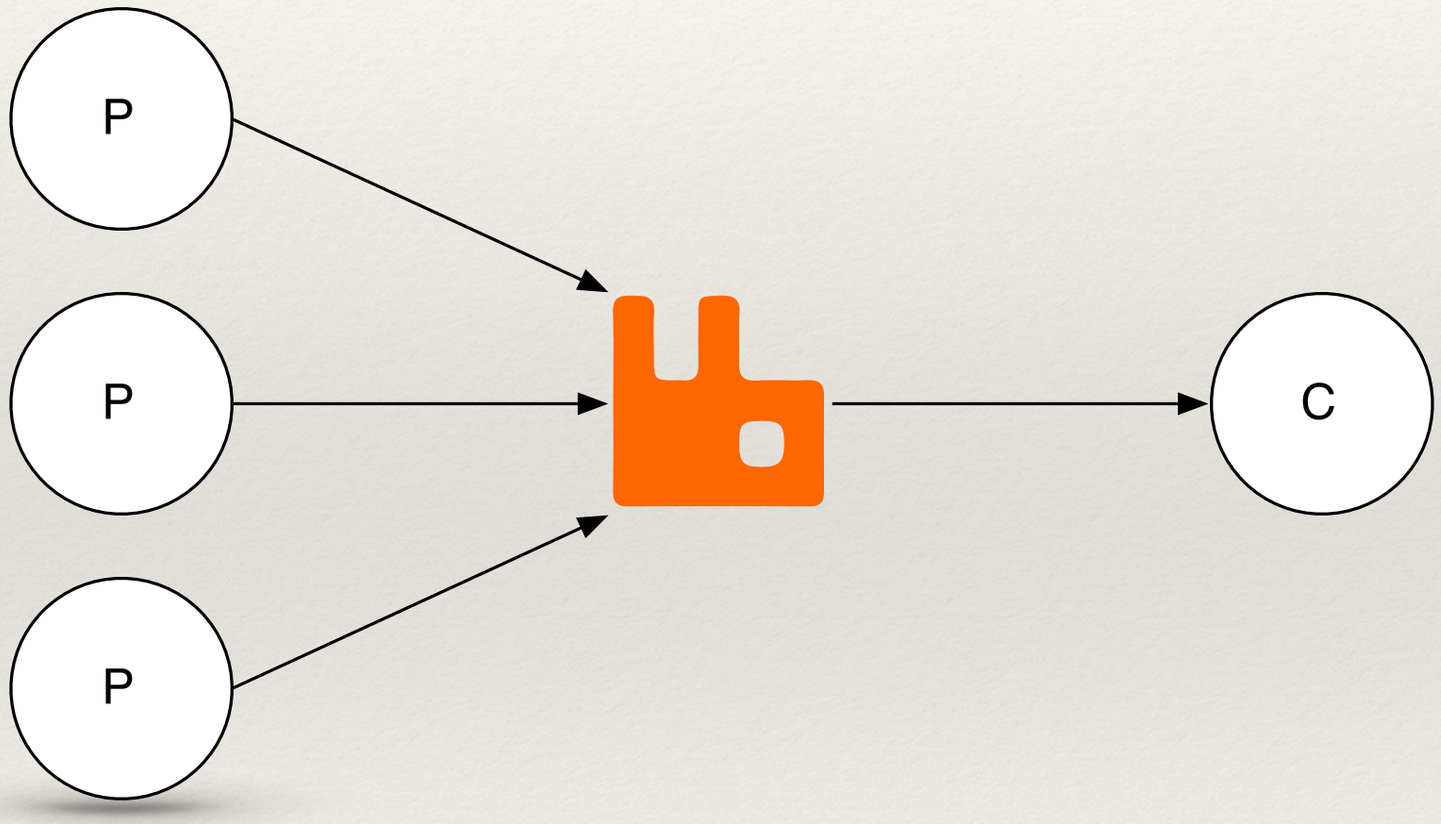
Common AMQP Terms

- **Broker**
  A server that implements AMQP
- **Producer or Publisher**
  A client application that sends messages to a broker
- **Consumer**
  A client application that reads messages from a queue

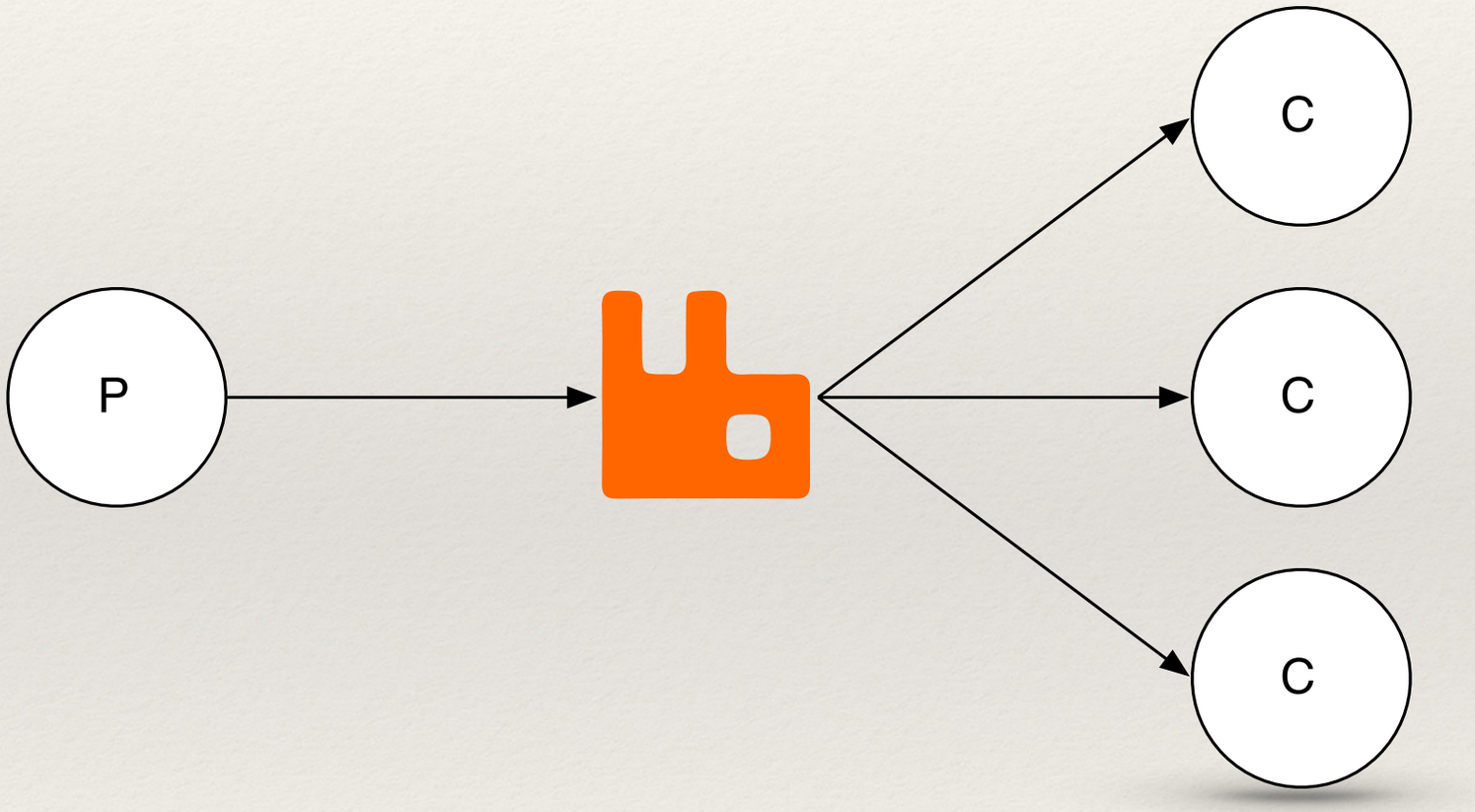
Publishers and Consumers



Multiple Publishers



Multiple Consumers



AMQP Messages

- Comprised of 3 or more frames:
  - **Method Frame**
    Basic.Publish, Basic.Deliver, etc
  - **Content Header** with body size & message properties
    timestamp, message-id, app-id, etc
  - *n* **Body** Frames with the opaque message payload

AMQP 0-9-1 Issues & Gotchas

- Ambiguous
- Authentication
- Asynchronous
- Connection Negotiation
- Exceptions

RabbitMQ

- Open Source (MPL)
- Written in Erlang/OTP
- Developed/Maintained by Pivotal
- Multi-Protocol
  AMQP 0-9-1 & 1.0, MQTT, STOMP, XMPP, HTTP, Web-STOMP & More
- Roots in AMQP 0-8/0-9-1

Who Uses It?

Agora Games

Chef

Google AdMob

Instagram

MeetMe

Mercado Libre

Mozilla

NASA

New York Times

National Science Foundation

Openstack

Rapportive

Reddit

Soundcloud

(and many more)

Why Use RabbitMQ?

- Create loosely-coupled applications
- Communicate across applications or platforms
- Tap into pre-existing message flows for new purposes
- Scale-out clustering for growth, throughput, and HA
- Federation for WAN latencies and network partitions
- Extensible plugin-in architecture

RabbitMQ Extensions to AMQP

Authentication Failure

Exchange to Exchange Bindings

Delivery Confirmations

Basic.Nack

Consumer cancellations

Consumer priorities

Dead Letter Exchanges

Alternate Exchanges

Connection blocking

Message CC & BCC Routing

Queue Length Limits

Per Queue Message TTL

Per Message TTL

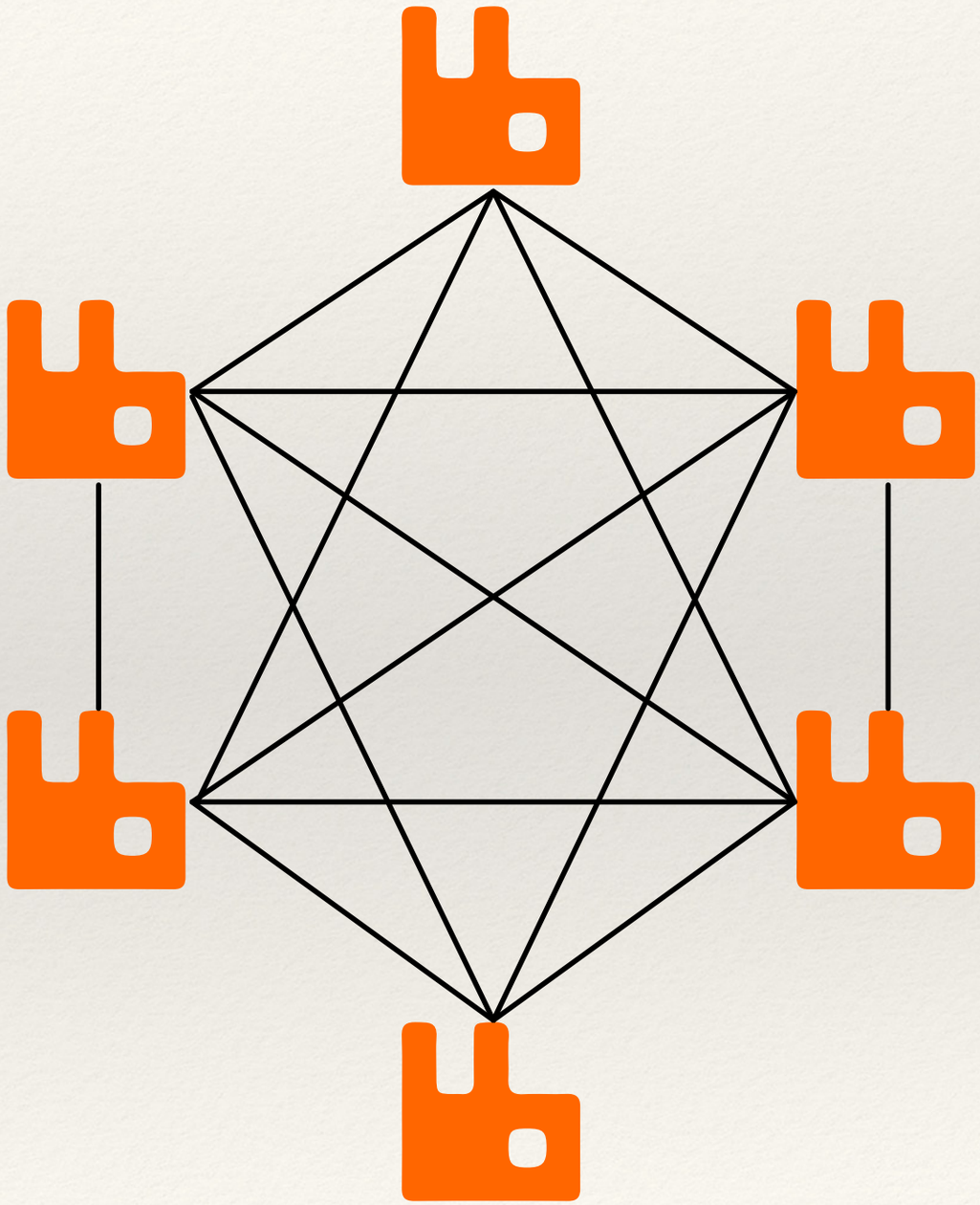
Queue TTL

Message User ID Validation

Auto-delete exchanges

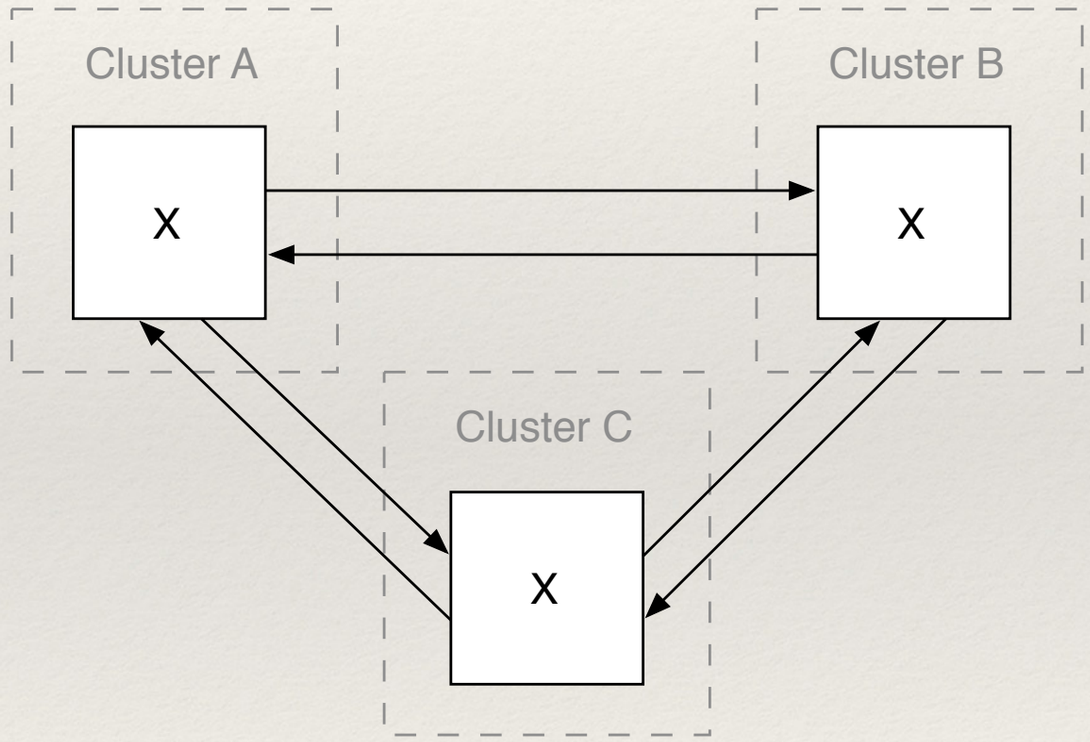
RabbitMQ Clustering

- LAN Only
- Adds highly-available queues
- Is cohesive, publish and consume from any node
- Leverages native Erlang clustering and communication
- Has multiple strategies for dealing with network partitions
- Manually configured via configuration or command line*



RabbitMQ Federation

- Provides loosely coupled inter-node/cluster RabbitMQ communication
- Connect multiple RabbitMQ clusters across data centers or the Internet
- Can be applied to both exchanges and queues



RabbitMQ Plugins

amqp1_0
Adds AMQP 1.0 support to RabbitMQ

auth-backend-amqp
AMQP RPC for Authentication

auth-backend-ldap
Authenticate via an external LDAP server

autocluster-consul
Automatically create clusters using Consul

msg_store_eleveldb_index
LevelDB storage for queue messages

presence-exchange
Publishes messages upon binding changes

rabbitmq-top
Top like view of RabbitMQ processes in the management UI

sharding
Scale out RabbitMQ with automatic queue sharding

(and many more)

Message Routing

Built-In Exchange Types

- **Direct**
  String matching on the routing key
- **Fanout**
  No routing key, messages delivered to all bound queues
- **Topic**
  Pattern matching in the routing key
- **Headers**
  No routing key, value matching in the headers property

Topic Exchange Binding Keys

Routing Key: namespace.delimited.keys

#

Receive all messages

namespace.#

Receive all messages in namespace

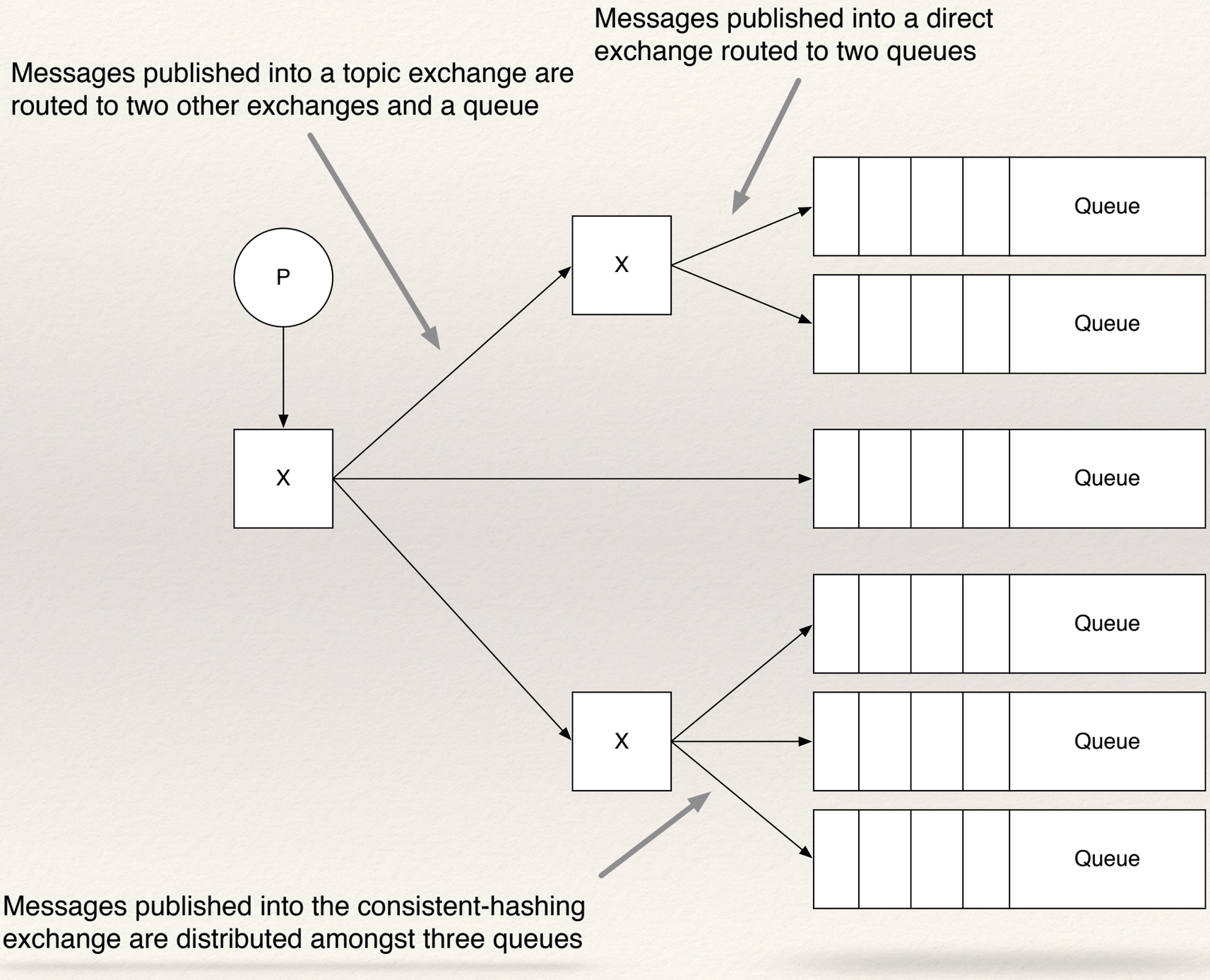
namespace.delimited.*

Receive all namespace.delimited messages

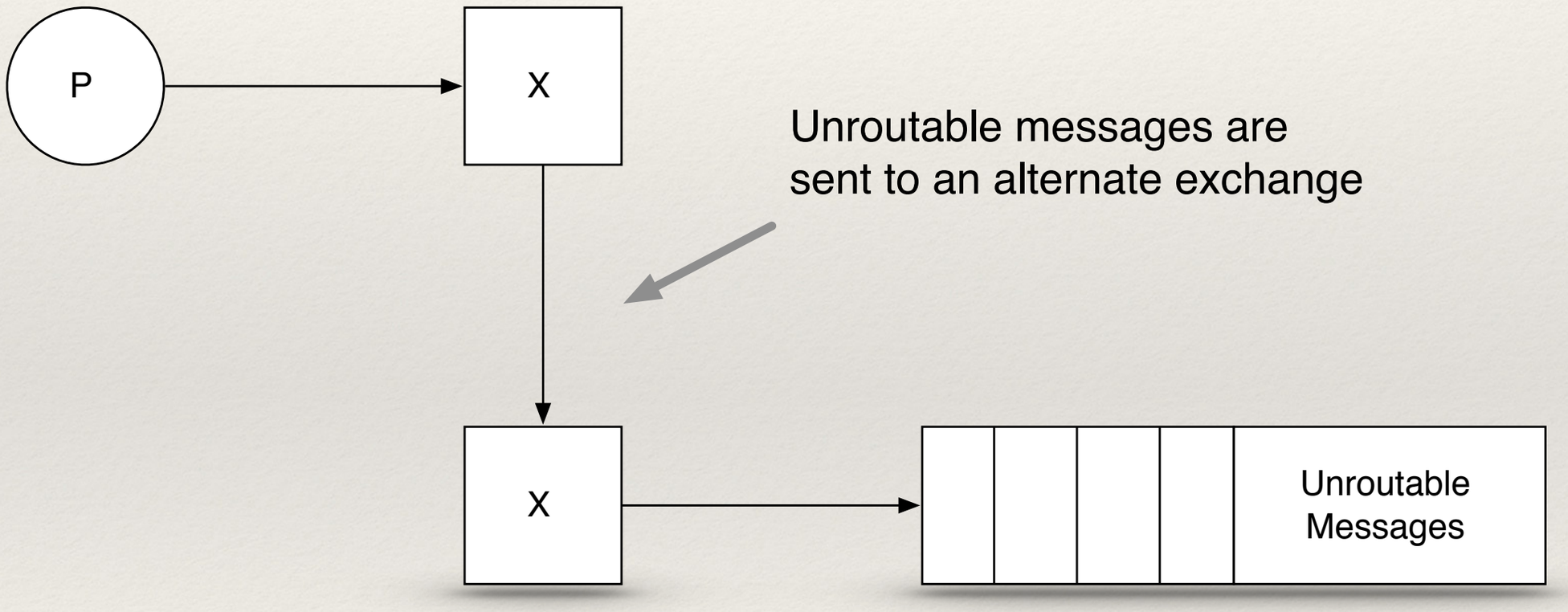
namespace.*.keys

Receive all namespace messages ending with keys

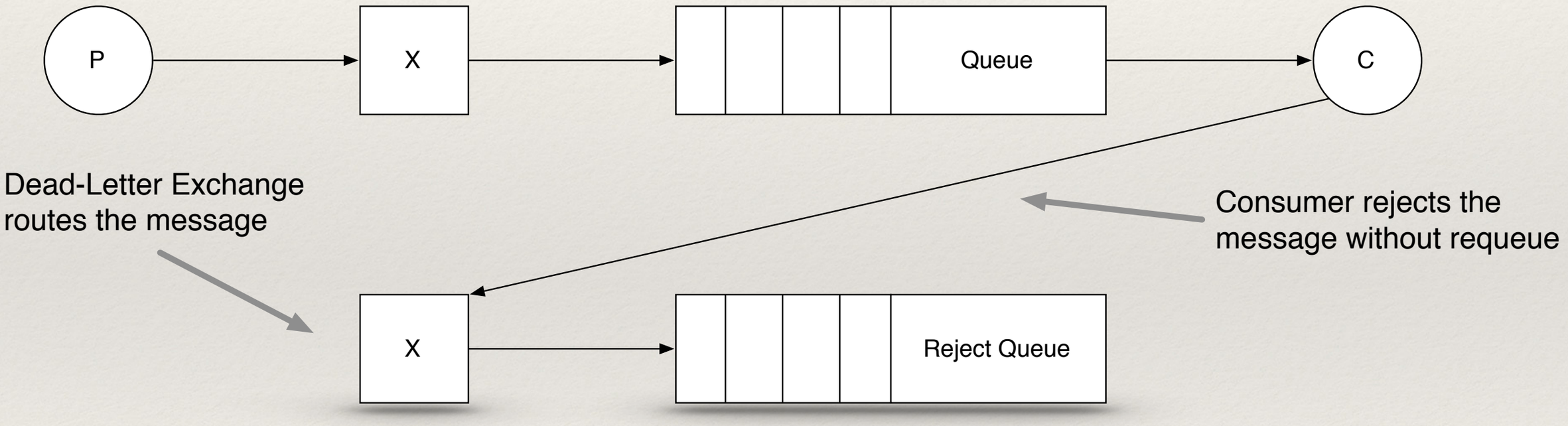
Exchange to Exchange Binding



Alternate Exchanges



Dead Letter Exchanges



Exchange Plugins

Consistent Hashing
Distribute messages via hashed value of routing key

Event
Publishes messages on AMQP events such as queue creation

Random
Distribute messages across all bound queues randomly

PostgreSQL LISTEN
Subscribes to and publishes PostgreSQL notifications

Recent History
Sends the last *n* messages to any newly bound queue

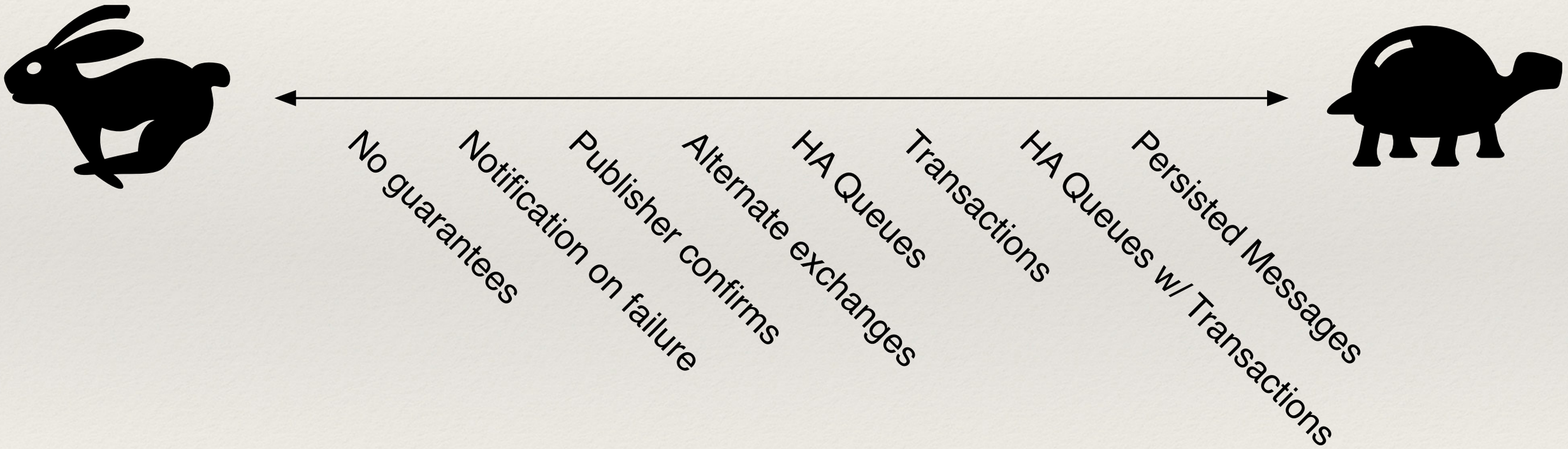
Reverse Topic
Allows for routing patterns at publish time and not via binding

Riak Storage
Stores messages published through the exchange into Riak

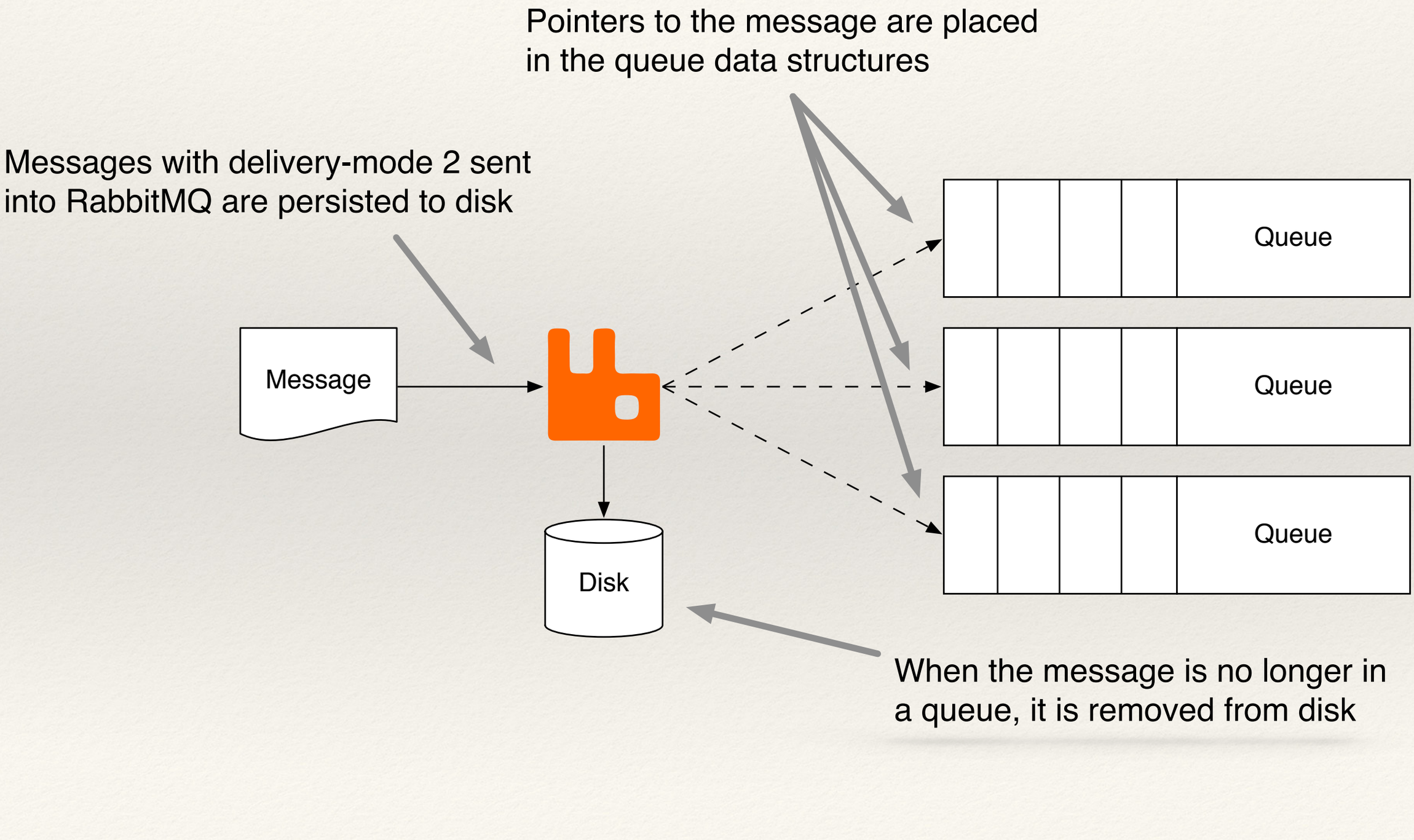
Script Exchange
Calls out to external scripts for message routing

Performance Considerations

Publishing Performance Scale



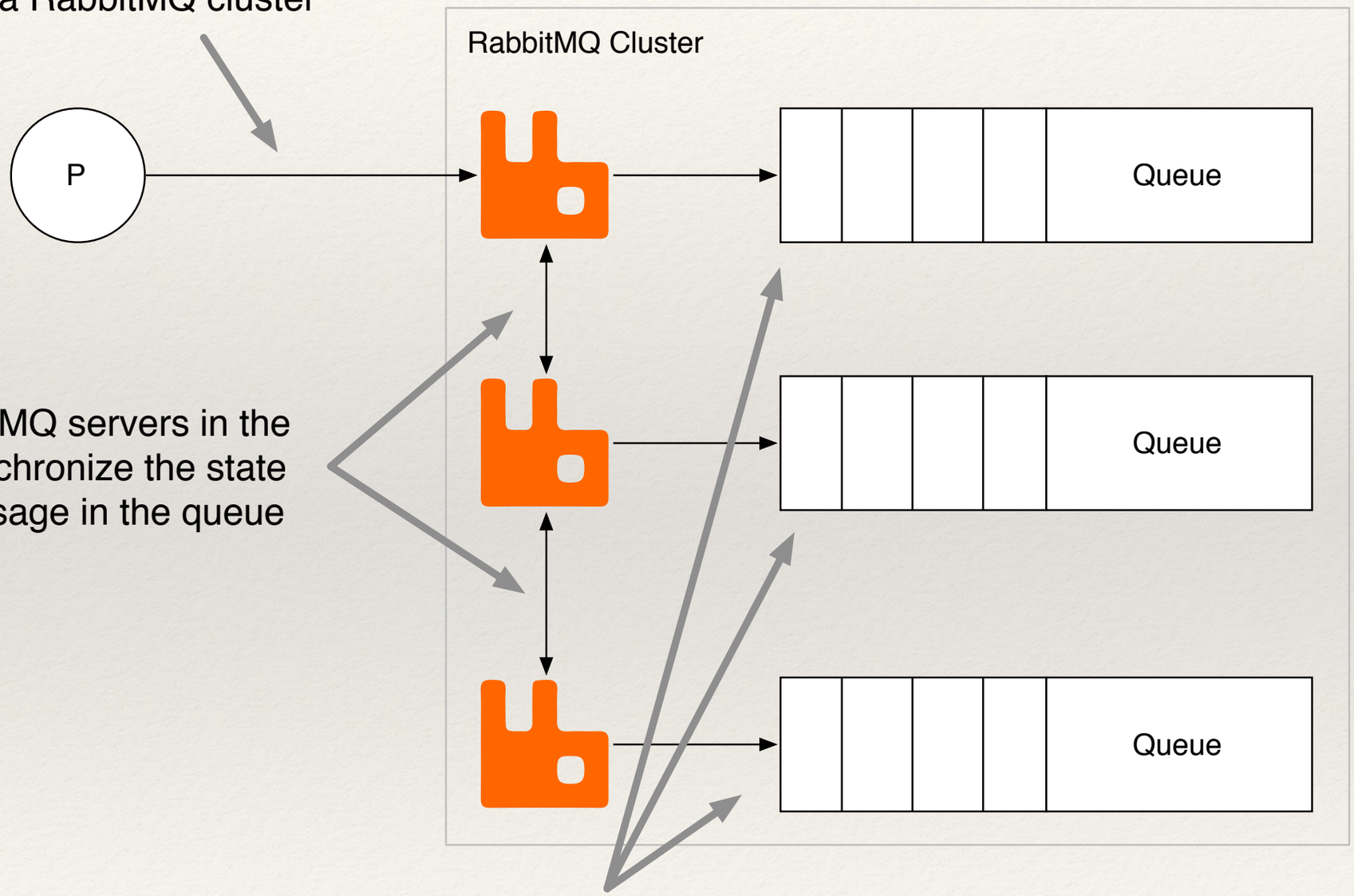
Persisted Messages IO Model



HA Queues & Performance

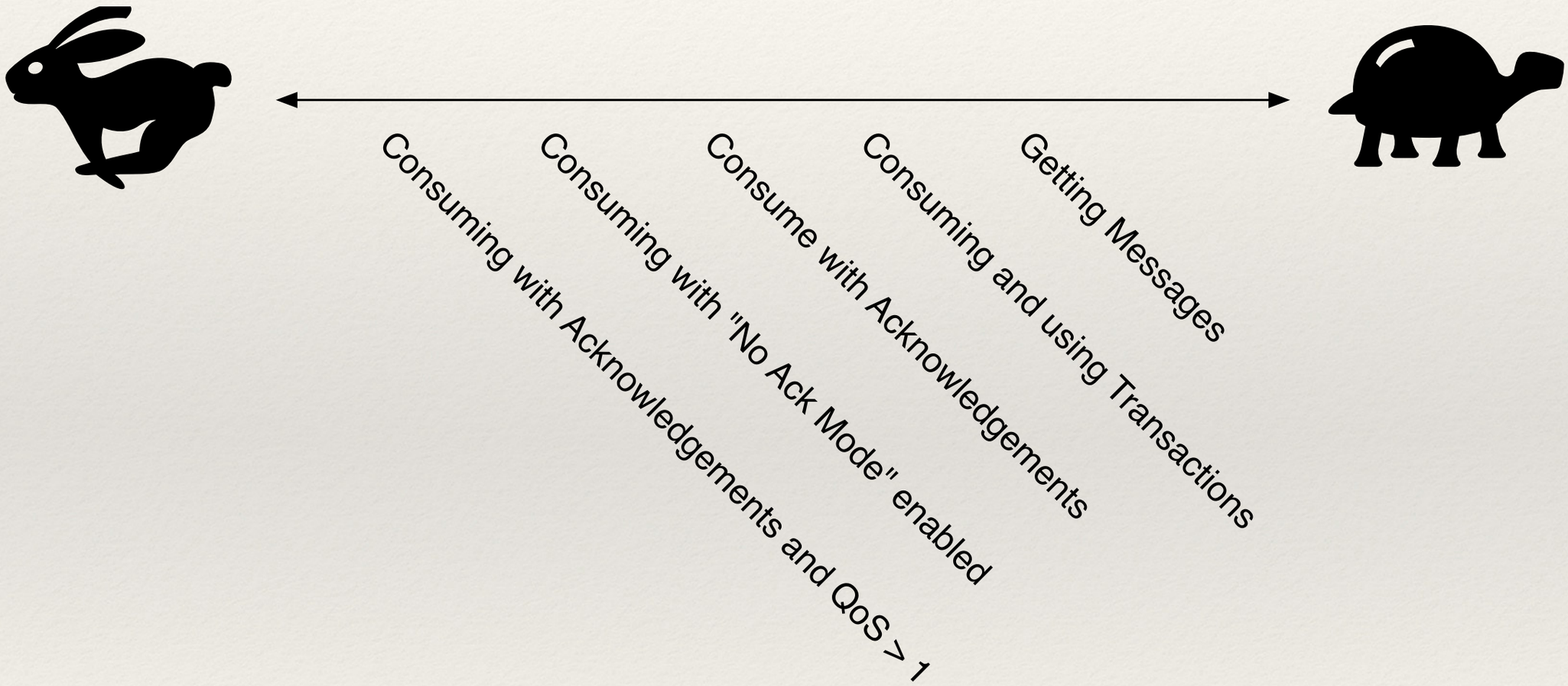
A publisher sends to any node in a RabbitMQ cluster

The RabbitMQ servers in the cluster synchronize the state of the message in the queue



The message that was published is put in the queue and is stored on each server

Consumer Performance Scale



Operational Concerns

Trending & Monitoring

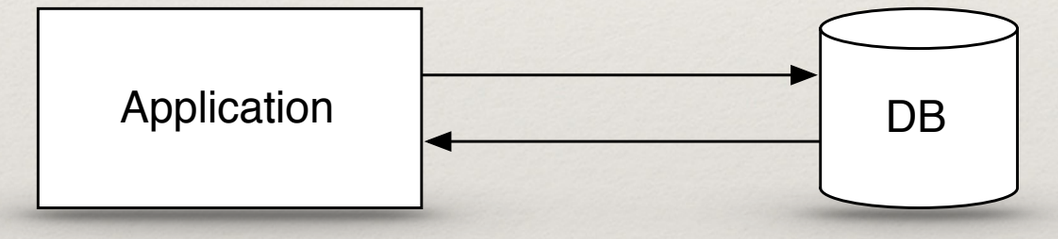
- RabbitMQ Management Plugin provides internal stats
  - *Queue depths, connection counts, throughput, memory usage, etc*
- Monitor with common tools such as Nagios or Sensu to services such as Boundary and NewRelic
- Stream based monitoring with Riemann for anomaly detection

Thoughts on Configuration

- Use configuration management!
  - Even for exchanges, queues, and bindings if possible
  - Helpful for disaster recovery
- Use RabbitMQ's Policies when possible
  - Exchanges and queues are immutable
  - Deleting and redeclaring for changes can be disruptive

Sample Usage Patterns

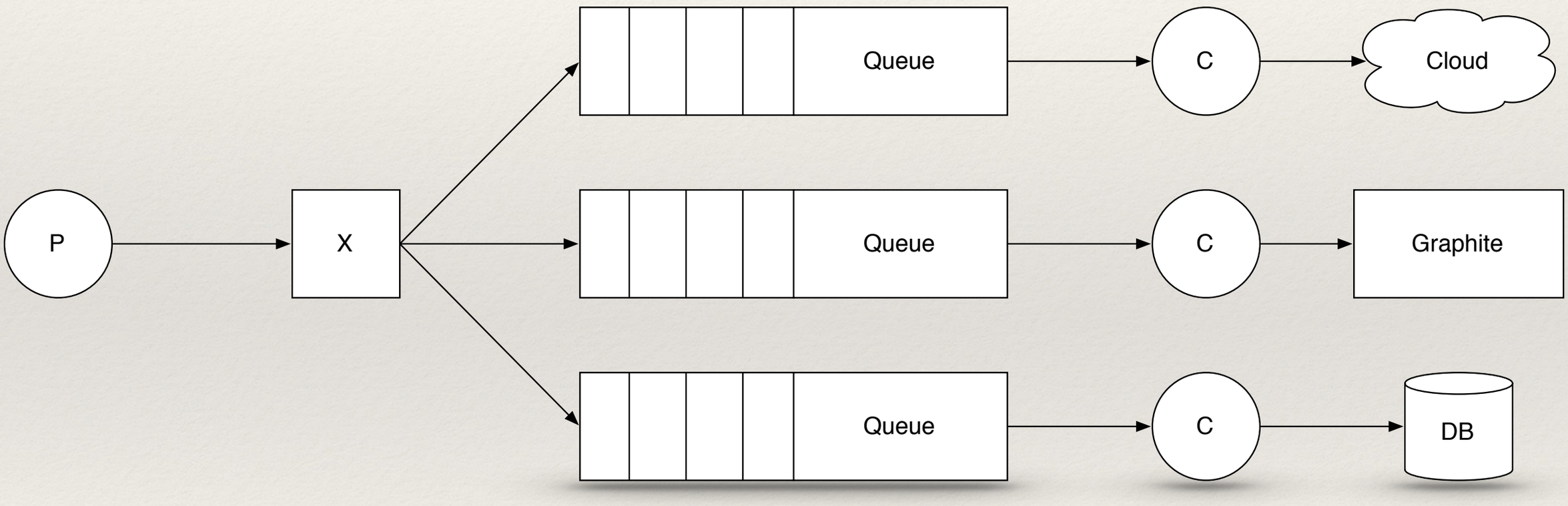
Evolve Tightly-Coupled Applications



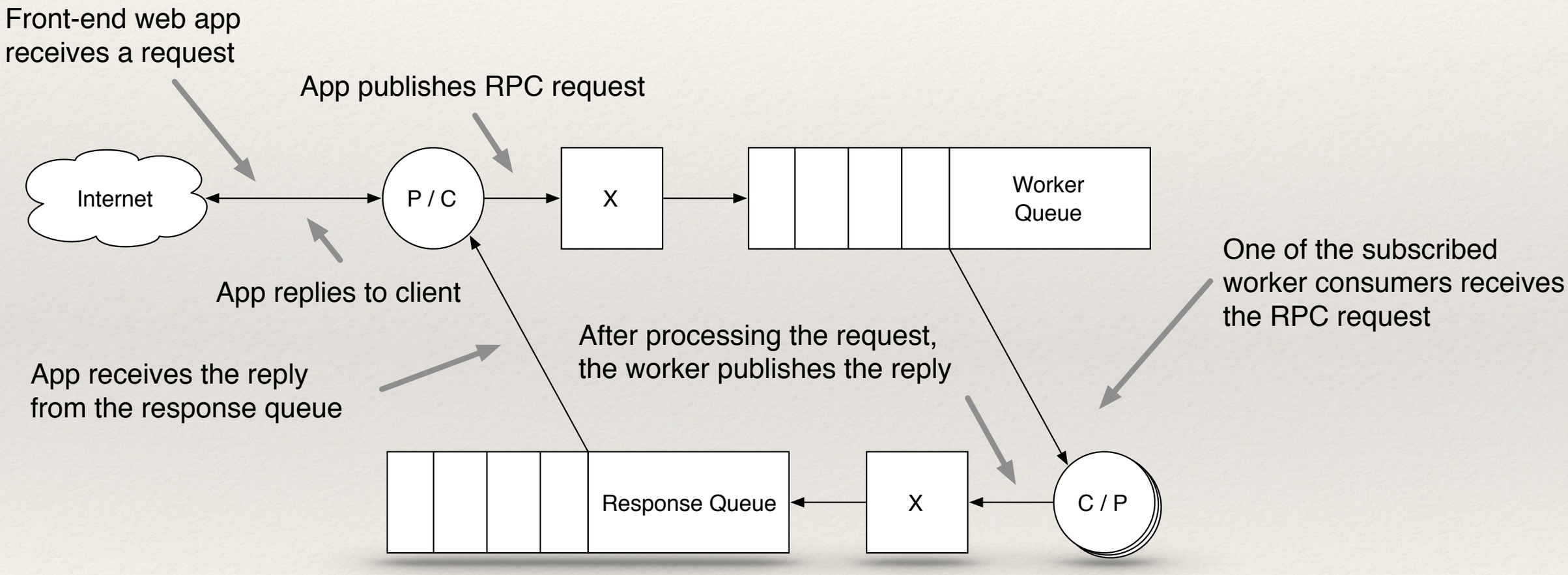
Decoupling Database Writes

Listen for Database Notifications

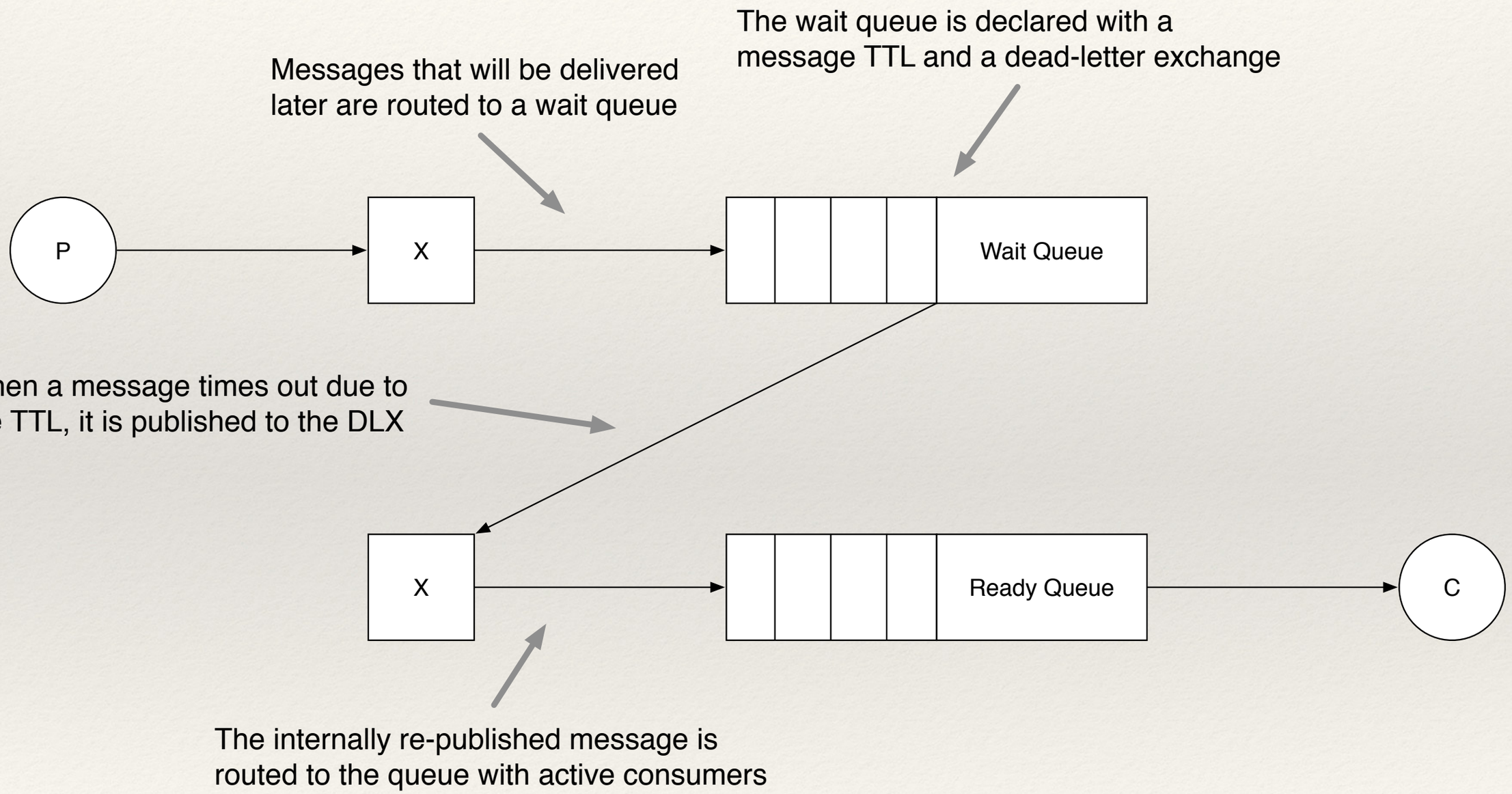
Multi-Purposed Messages



RPC

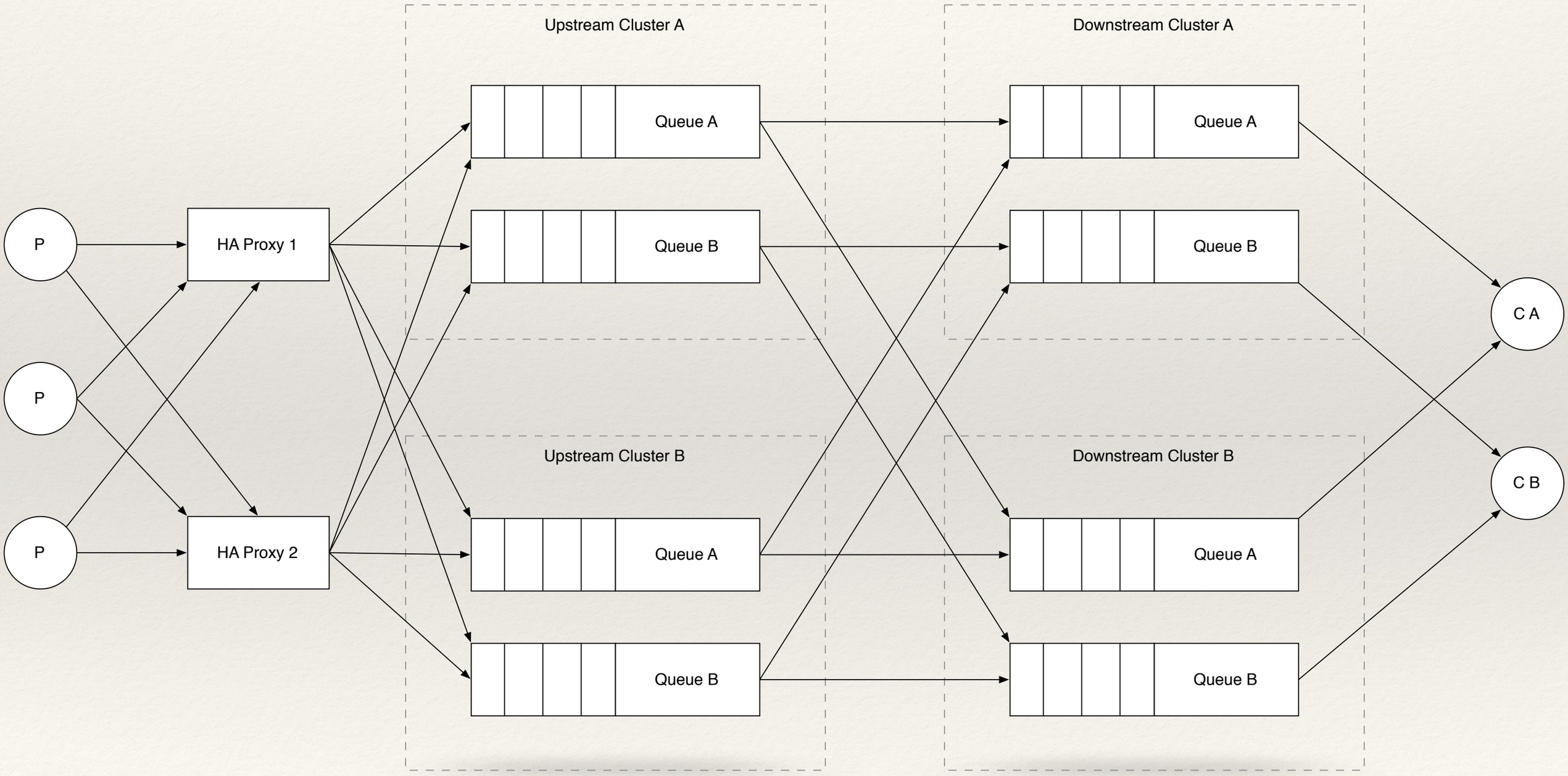

Front-end web app
receives a request
App publishes RPC request
Internet
P / C
X
Worker
Queue
One of the subscribed
worker consumers receives
the RPC request
App replies to client
After processing the request,
the worker publishes the reply
App receives the reply
from the response queue
Response Queue
X
C / P

Delayed Messages



WARNING: This pattern is a hack and can cause undesirable situations should the wait queue overwhelm the system!

HA RabbitMQ with Federation



Questions?