Introduction to Messaging and AMQP

Pivotal

Agenda

- **Messaging and Asynchronous Systems**
- Introduction to AMQP
- RabbitMQ case studies

What is Messaging?

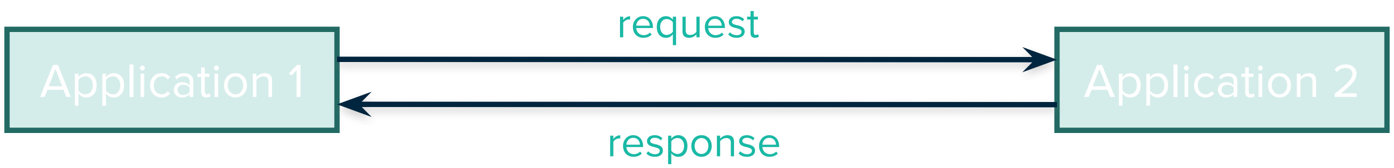
- Messaging is a way to make applications / systems communicate
- Messaging is sometimes called an "integration style”
- Messaging eases decoupling between applications
  - Applications can evolve independently.
- Messaging is often referred to as "Message Oriented Middleware" (MoM)
- Messaging server typically called a *broker*
  - Broker ensures reliable dispatching of messages

What is a Message?

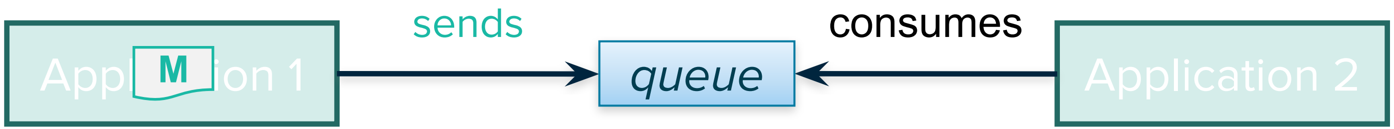
- Messages consist of a payload and multiple headers
- Payload is the actual content to exchange
  - Can be a string, a byte array (binary serialized object)
  - Often serialized with data exchange format (JSON, XML)
- Headers are metadata
  - Key/value pairs
  - Can be technology-specific or custom
  - E.g., routing (where to go, whom to answer to, etc.) …
- Messaging technologies usually come with their subtleties
  - An AMQP message can have several kinds of metadata (header, properties, delivery annotations), a body, and even a footer!

Synchronous vs. Asynchronous

- When an application wants to talk to another application, it can send a message to it …
  - Synchronously



  - Asynchronously



Pivotal

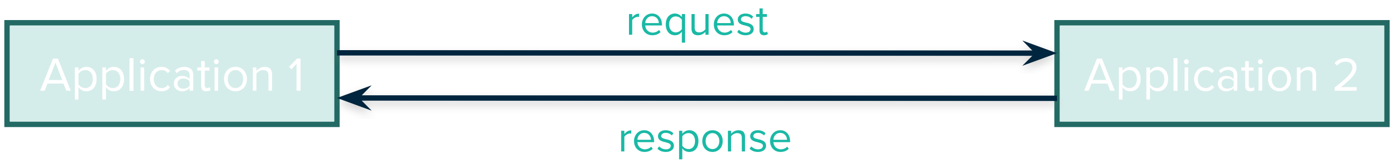
Synchronous vs. Asynchronous

- Real-world comparison:
  - Synchronous = phone
  - Asynchronous = SMS

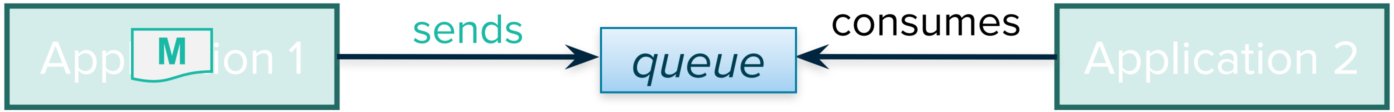
NOTE Asynchronous messaging decouples the senders and receivers, more than synchronous remote method calls.

Synchronous Messaging

- Sending application must know about receiving application
  - Host, port, protocol, endpoint
- Sending application is blocked until receiving application answers
- What happens if the receiving application doesn't respond?
  - Wait?
  - Crash?
- HTTP is an example of synchronous messaging



Asynchronous Messaging



- Sending application knows only about the broker
- Sending application can "fire and forget" if it doesn't need a response
  - Request / reply also supported
- Receiving application consumes messages whenever it wants
  - Constant polling, notification, batch de-queuing
  - It consumes messages rather than receives them
- JMS and AMQP are examples of asynchronous messaging

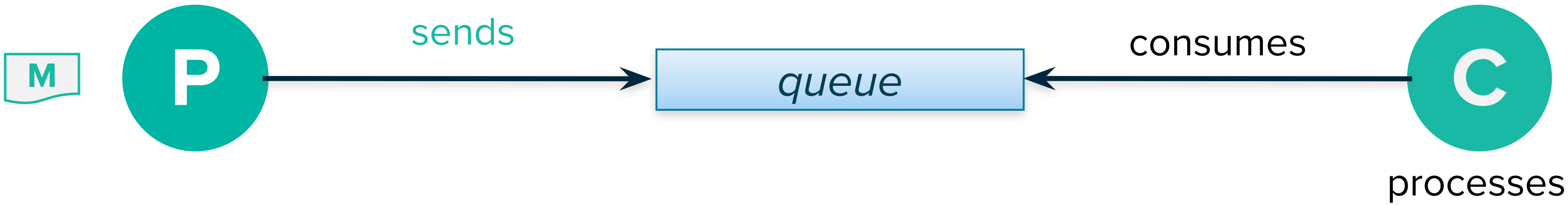
Decoupling

- The broker decouples the sender and the receiver…
  - Spatially
    - They don't need to be co-located
  - Temporally
    - No need of immediate responses
    - Processing can happen in the background
    - Receiver doesn't have to be up when message is sent
  - Logically
    - Sender and receiver don't know about each other
    - Broker can use advanced routing

Use Cases

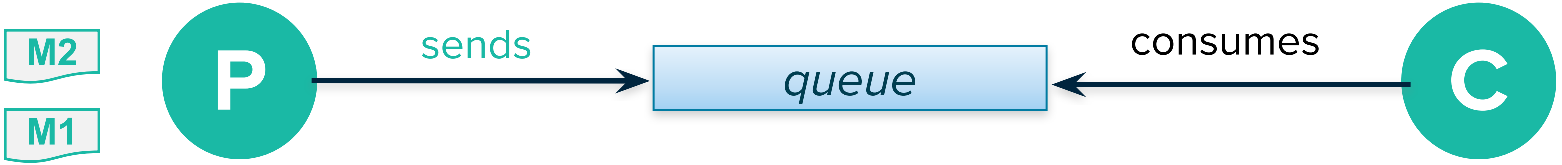
- Simple producer – consumer
  - Send message for further processing
  - E.g., a web app places an order for further processing
- Request / reply
  - Send message and wait for response
  - E.g., to throttle or scale processing on the consumer side
- Publish / subscribe
  - Send message for multiple consumption
  - E.g., order sent to inventory and billing systems

Simple Producer – Consumer



Simple Producer – Consumer

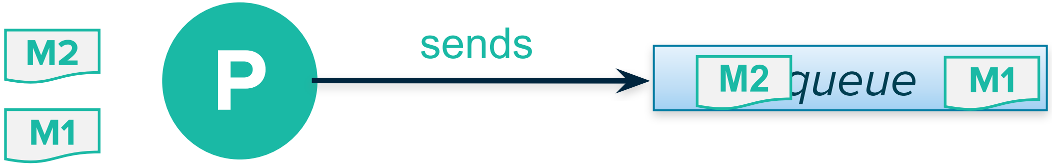
Temporal Decoupling – 1



Consumer goes down for upgrade

Simple Producer – Consumer

Temporal Decoupling – 2



Consumer goes down

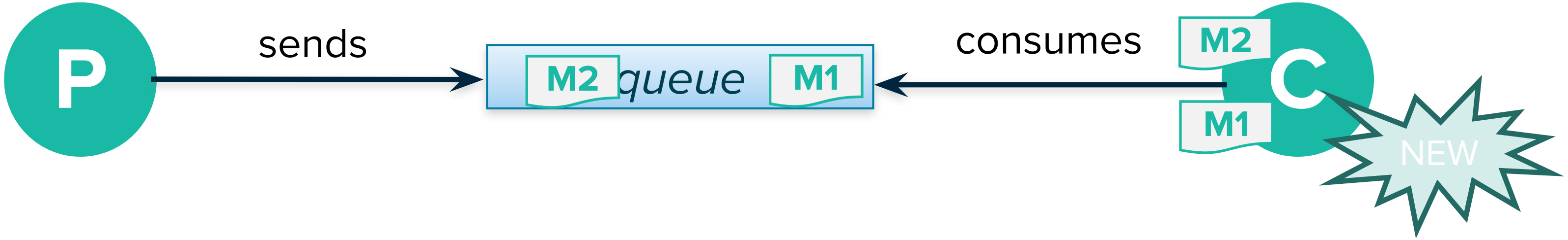
for upgrade

Simple Producer – Consumer

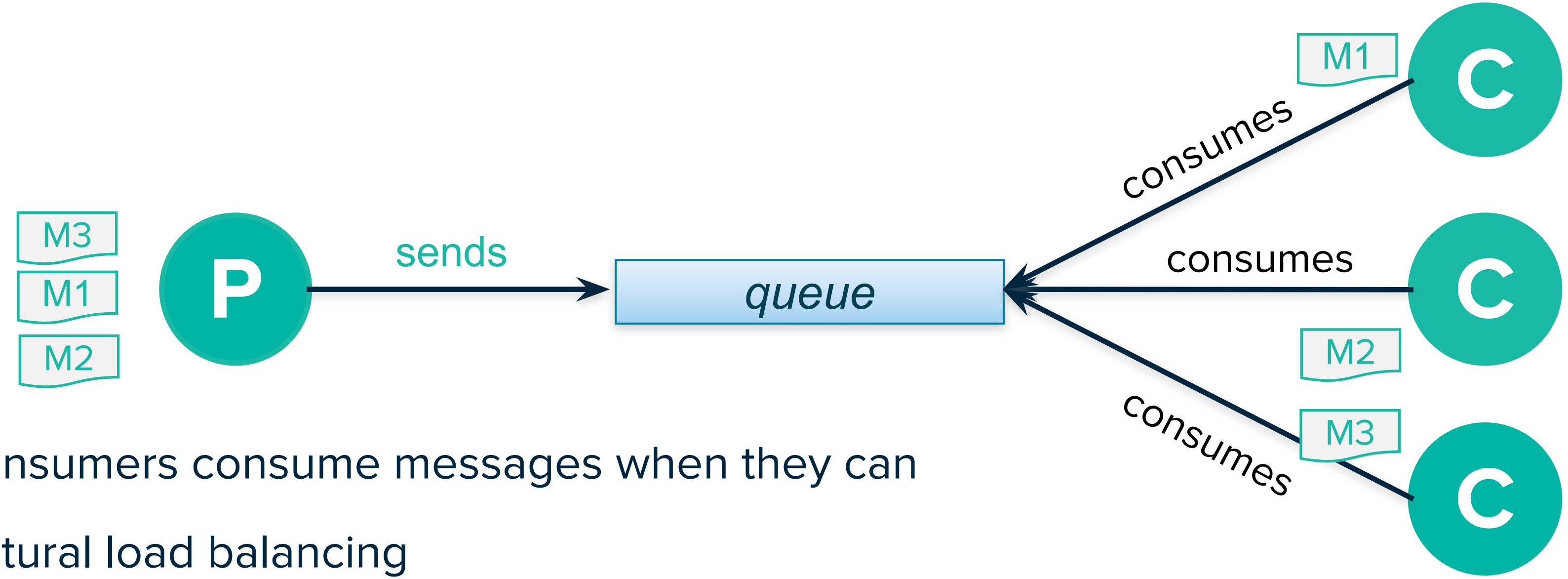
Temporal Decoupling – 3

New version

of consumer comes up

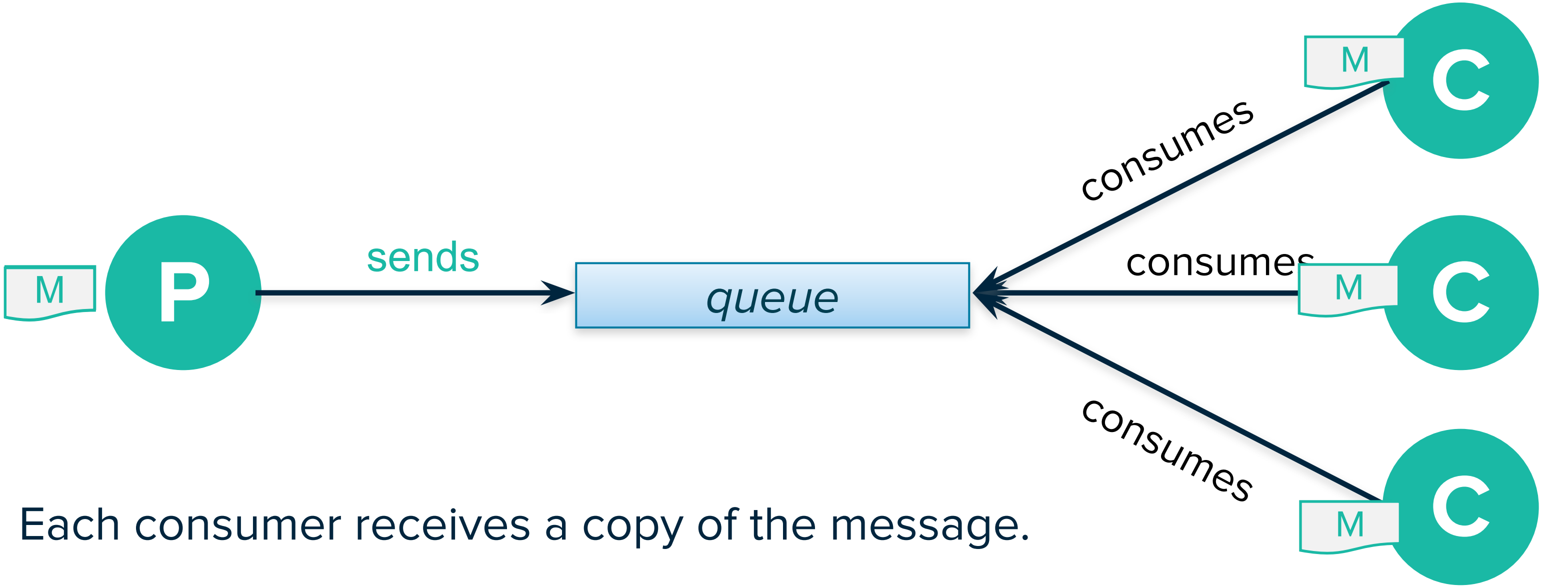


Simple Producer – Multiple Consumers



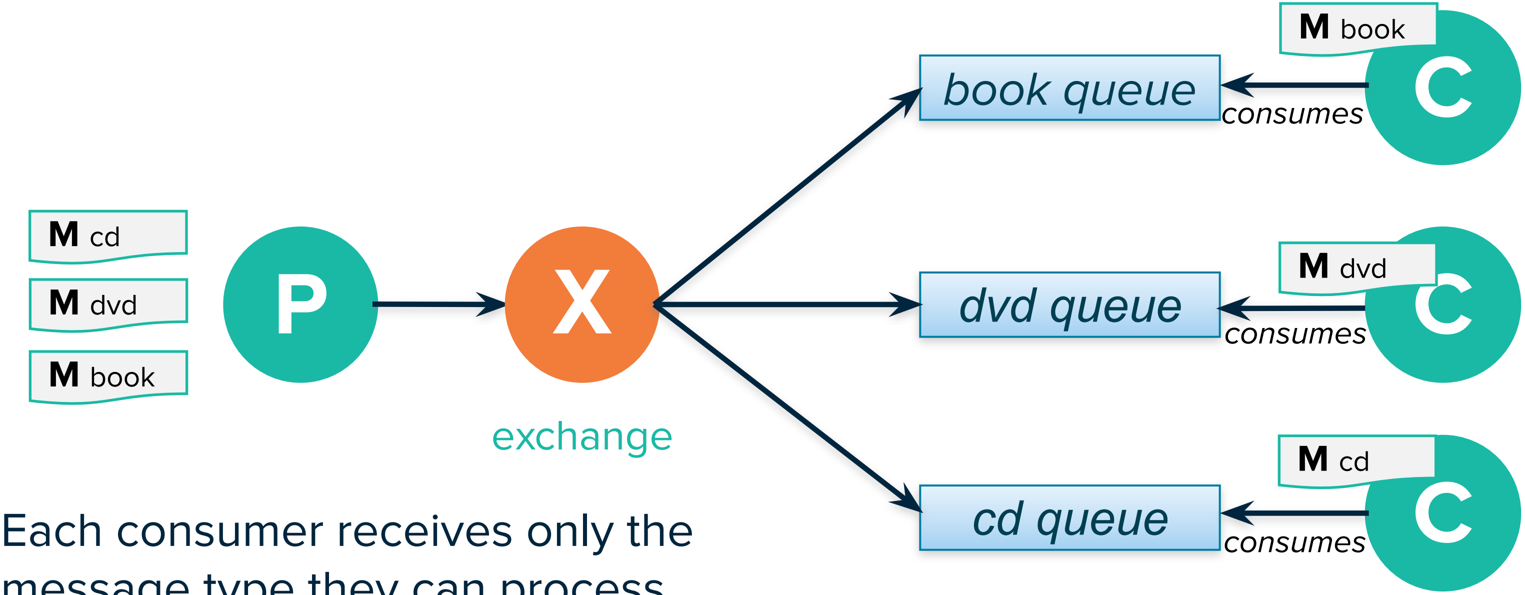
- Consumers consume messages when they can
- Natural load balancing

Publish / Subscribe



- Each consumer receives a copy of the message.

Routing



- Each consumer receives only the message type they can process.

Pivotal

Pros and Cons of Messaging

- Pros
  - Scalability
  - Loose coupling
- Cons
  - Complexity
  - Broker can be a single point of failure

Messaging in the Cloud

- Asynchronous messaging is an ideal integration tool for cloud deployments
  - Elastic
  - Scalable
  - Robust
  - Decoupled
- RabbitMQ is the preferred mechanism for integrating Pivotal Cloud Foundry applications

Pivotal
Cloud Foundry®

CF

Cloud AMQP

Agenda

- Messaging and Asynchronous Systems
- **Introduction to AMQP**
- RabbitMQ case studies

AMQP

- AMQP stands for Advanced Message Queuing Protocol
- AMQP
  - Aims to provide an open standard for messaging
  - Enables complete interoperability for messaging middleware
  - Defines the network protocol and the semantics of broker services
- AMQP is open, interoperable, and platform agnostic

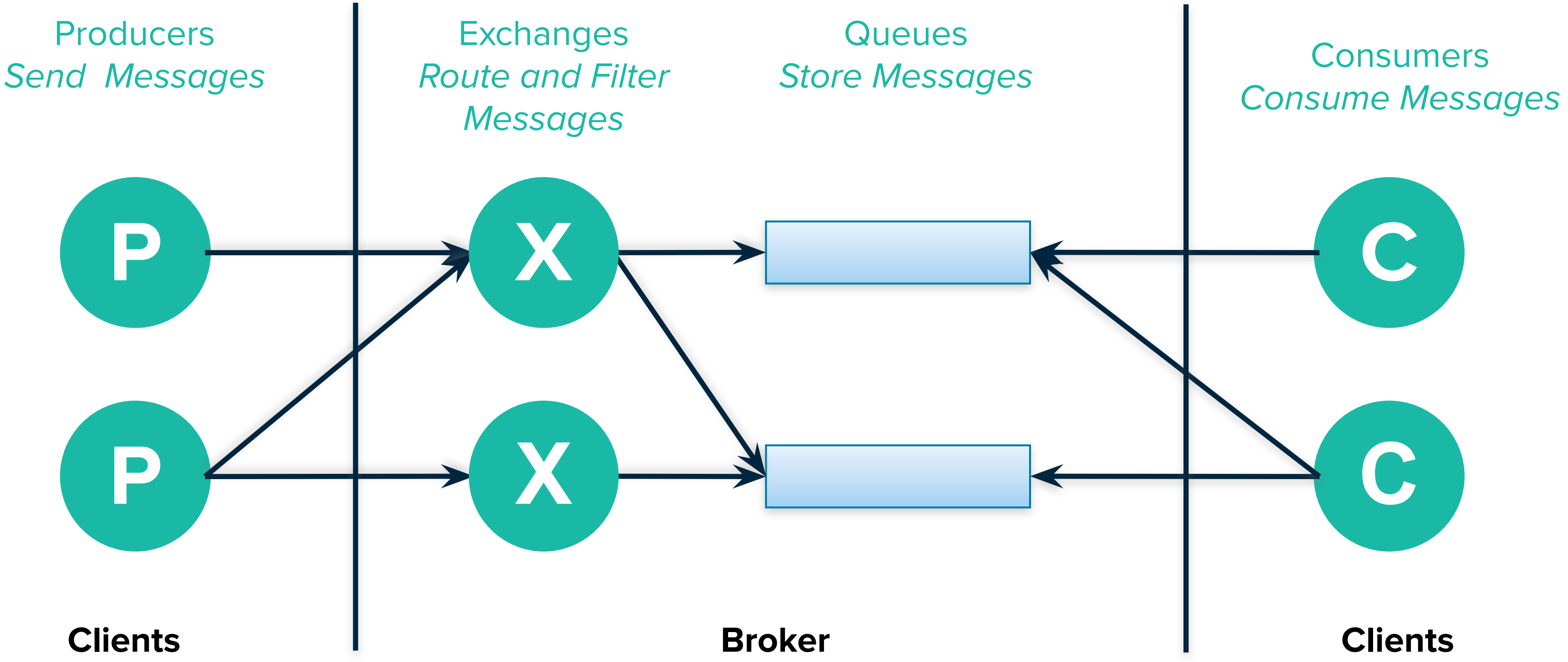
NOTE AMQP is an application protocol, like HTTP and SMTP.

History of AMQP

- Development started in 2004 by JP Morgan and iMatix
- AMQP Working Group was born when other companies joined the effort
  - See members at http://www.amqp.org/about/members
- Specification version 1.0 final in October 2011
  - Downloadable at http://www.amqp.org/resources/download
- AMQP originated in the finance industry …
- … but it addresses a large range of middleware problems

NOTE This training focuses on AMQP 0.9.1, the most popular and widespread version.

The AMQP Model (v0.9.1)



Producers
Send Messages

Exchanges
Route and Filter Messages

Queues
Store Messages

Consumers
Consume Messages

Clients

Broker

Clients

JMS / AMQP Comparison

	JMS	AMQP
Defined by	Java Community Process	AMQP Working Group
Scope	Java API	Application protocol
API	Yes	No
Interoperable	No (broker specific)	Yes
Distributed transactions	Yes	Yes*
Routing	No	Yes

NOTE * RabbitMQ implements AMQP but doesn't support distributed transactions.

Agenda

- Messaging and Asynchronous Systems
- Introduction to AMQP
- **RabbitMQ case studies**

RabbitMQ case study: New York Times

- System provides subscription services for news, video feeds, etc.
- Dozens of RabbitMQ instances
- Deployment across 6 AWS zones
- Upon launch, the system autoscaled to 500 K users
- Connection times stayed stable around 200 ms

Source http://lists.rabbitmq.com/pipermail/rabbitmq-discuss/2014-January/032920.html

RabbitMQ case study: Travis CI

- Hosted continuous integration service
- Build logs are forwarded to RabbitMQ for live display
- Messages contain an incrementing counter to identify ordering
- RabbitMQ clusters are hosted on CloudAMQP
- Travis CI handles 74 K builds per day

Source https://blog.pivotal.io/pivotal/case-studies/continuous-integration-scaling-to-74000-builds-per-day-with-travis-ci-rabbitmq

Summary

- Asynchronous messaging facilitates decoupling between systems
- Common messaging patterns:
  - Simple producer/consumer
  - Request/reply
  - Publish-subscribe
- AMQP is an open standard for messaging
  - A binary network protocol specification
  - Not just a Java interface specification like JMS!