Life is like your computer. If you're on tilt
by juggling too much, you'll crash.
RELAX, Re-BOOT & Re-FRESH!

—Someone

Message Queue implementation using RabbitMQ

A collaboration by
Maitree and Avi

AMQP

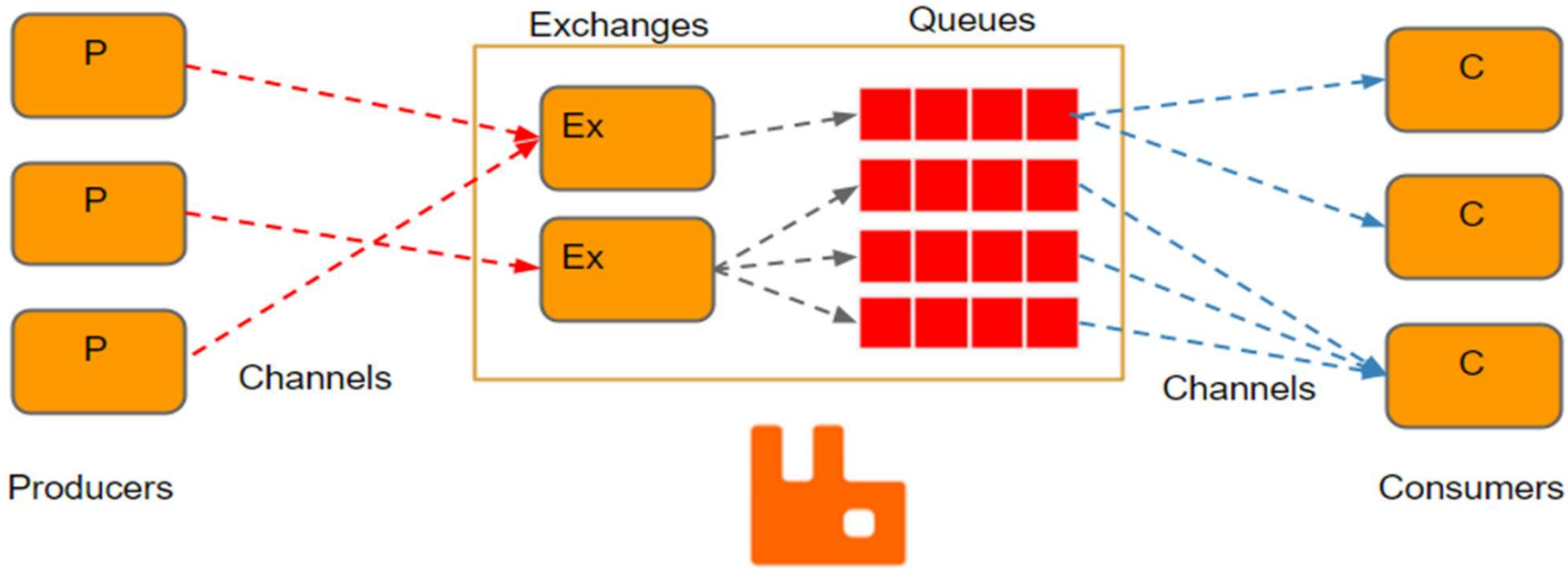
The Advanced Message Queuing Protocol is an open standard application layer protocol for message-oriented middleware. The defining features of AMQP are message orientation, queuing, routing, reliability, and security.

In addition to the Banking and Finance Industries, AMQP is also being used to connect hundreds of critical systems in Telecommunications, Defense, Manufacturing, Internet and Cloud Computing, and many additional market segments.

Java Support for AMQP

- RabbitMQ Java Client Jar
- Spring Framework Support
- Spring-AMQP
- Spring-Integration

AMQP Messaging Abstraction



RabbitMQ

RabbitMQ is an open-source message-broker software that originally implemented the Advanced Message Queuing Protocol and has since been extended with a plug-in architecture.

Rabbit

- Highly capable open source messaging server
- Community and commercial support
- Polyglot of client support
- Scale up to meet your requirements
- Support different integration / routing patterns

RabbitMQ High-Availability

- RabbitMQ supports HA features
- Built on top of clustering
- Resources (exchanges, queues) are mirrored across nodes
- Handling of automatic failovers

AMQP Message Model

- **Exchanges**
  - Accepts messages from message-producing clients
  - Route messages to appropriate queues
- **Bindings**
  - Decouples message routing from the producer
  - Rules for routing messages to the destination queue
- **Message Queues**
  - Bound to an exchange
  - Buffers messages until consumed by client

AMQP Transport Model

- **Connection**
  - A physical TCP connection between client and broker
  - A connection consists of many channels
- **Channel**
  - A virtual conduit thread between your client/broker

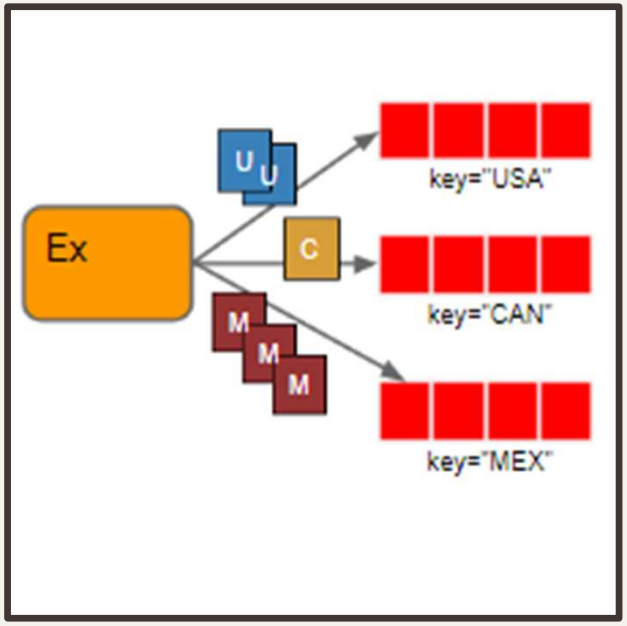
exchange is used as a routing mediator, to receive messages from producers and push them to message queues according to rules provided by the RabbitMQ exchange type.

-Exchanges support 3 routing strategies
-Delivers messages based on algorithm
-Allow for numerous routing scenarios.

Direct

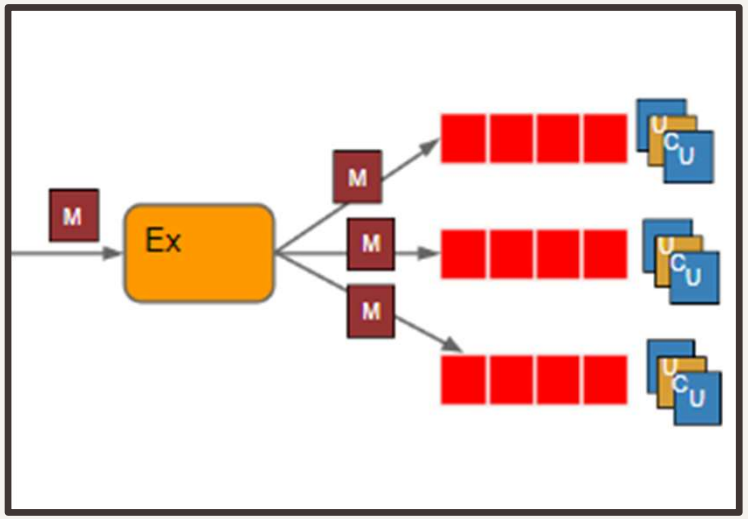
Fanout

Topic



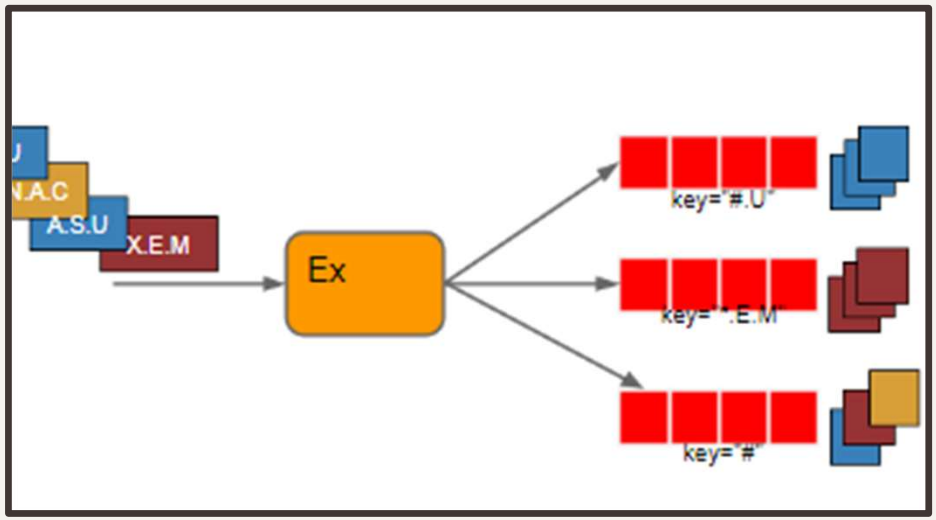
Direct

- Simple strategy
- Routed based on the binding key value in message
- Message is delivered to the queue that matches the key value.



Fanout

- Used to broadcast messages
- All bounded queues receive same messages



Topic

- Routing based on binding key patterns
- Messages can end up in one/more queues

Binding

bind method binds a queue to an exchange so that messages flow (subject to various criteria) from the exchange (the source) to the queue (the destination).

Message Queue

A message queue provides a lightweight buffer which temporarily stores messages, and endpoints that allow software components to connect to the queue in order to send and receive messages.

Channel and Connection

A connection is a TCP connection between your application and the RabbitMQ broker. A channel is a virtual connection inside a connection. In other words, a channel multiplexes a TCP connection.

RabbitMQ Implementation: Key Points

- **Leveraged Spring AMQP** for seamless integration with RabbitMQ message broker.

- **Established Asynchronous Communication:** Producers publish messages to designated queues, decoupling message delivery from processing.

- **Implemented Message Consumption:** Consumers subscribe to queues, ensuring timely retrieval and processing of messages.

List of references

- https://www.rabbitmq.com
- https://docs.google.com/presentation/u/0/d/11Fim1w8MS_Du99VmCNvQbg8R7i1zD7u_DQK6n1DVWdA/htmlpresent?pli=1
- https://en.wikipedia.org/wiki/RabbitMQ
- Write your references here
- Write your references here

Thanks

Do you have any questions?