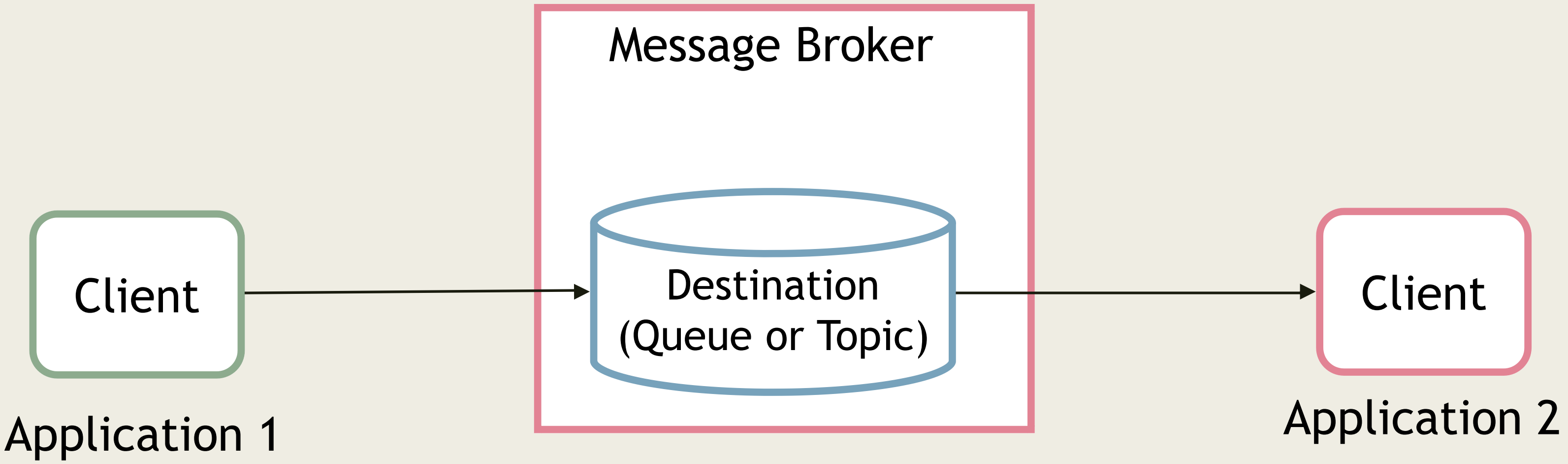
SPRING MESSAGING

Agenda

- Messaging
- JMS
- Spring JMS
- AMQP (to Do)
- Spring Messaging With RabbitMQ (to Do)

Messaging

- Messaging is a method of communication between software components or applications.
- A messaging system is a peer-to-peer facility
  - *A messaging client can send messages to, and receive messages from, any other client.*
  - *Each client connects to a messaging agent (Broker) that provides facilities for creating, sending, receiving, and reading messages.*



Advantages of Messaging

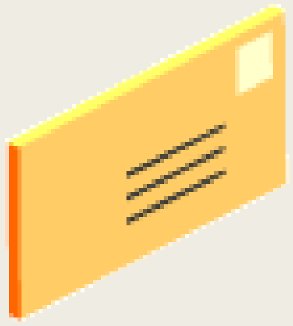
- Loosely Coupled
  - *Clients need not know each other*
- Asynchronous Communication
  - sender and the receiver do not have to be available at the same time in order to communicate
  - The sender and the receiver need to know only which message format and which destination to use.
- Reliable Delivery
  - The message will be delivered to the targeted clients only.

Messaging is NOT email

- Email is a method of communication between people or between software applications and people.
- Messaging is used for communication between software applications or software components.

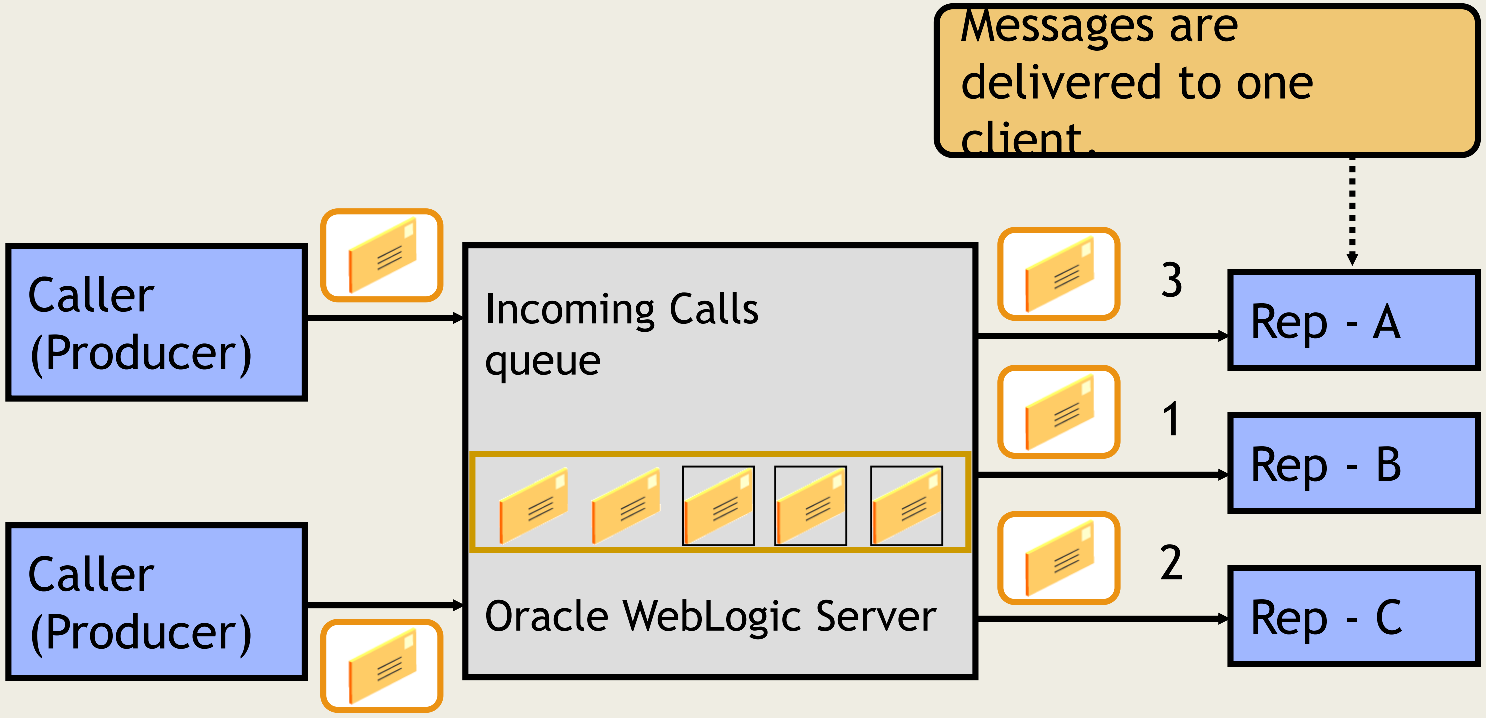
Message-Oriented Middleware

- The message-oriented architecture enables asynchronous and cross-platform integration of applications.
- Message-oriented middleware refers to an infrastructure that supports messaging.
- Typical message-oriented middleware architectures define the following elements:
  - *Message structure*
  - *The way to send and receive messages*
  - *Scaling guidelines*



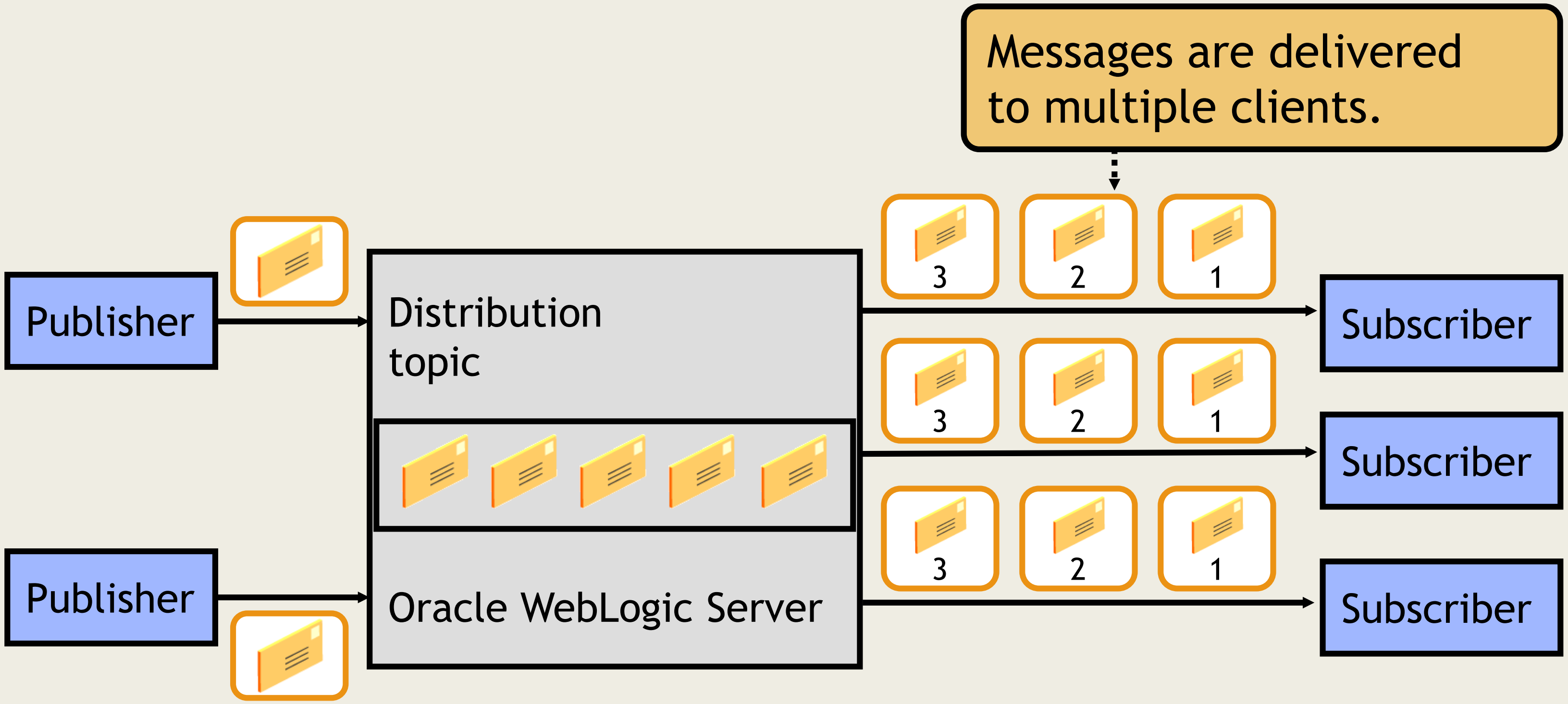
Point-To-Point Queue

- Many message producers can serialize messages to multiple receivers in a queue.



Publish/Subscribe Topics

- Publishing and subscribing to a topic decouples producers from consumers.



JMS

Java Message Service

JMS

- The Java Message Service is a Java API that allows applications to create, send, receive, and read messages.
- The JMS API defines a common set of interfaces and associated semantics that allow programs written in the Java programming language to communicate with other messaging implementations.
- The JMS API enables communication that is not only loosely coupled but also
  - *Asynchronous:*
    - A JMS provider can deliver messages to a client as they arrive; a client does not have to request messages in order to receive them.
  - *Reliable*
    - The JMS API can ensure that a message is delivered once and only once. Lower levels of reliability are available for applications that can afford to miss messages or to receive duplicate messages.
- The JMS specification was first published in August 1998. The latest version is Version 1.1, which was released in April 2002

JMS Messaging Domains

- JMS has support for
  - *Point-to-Point messaging (Queue Based Messaging)*
  - *Publish Subscribe Messaging (Topic Based Messaging)*
  - *Distributed Transaction for Messaging*

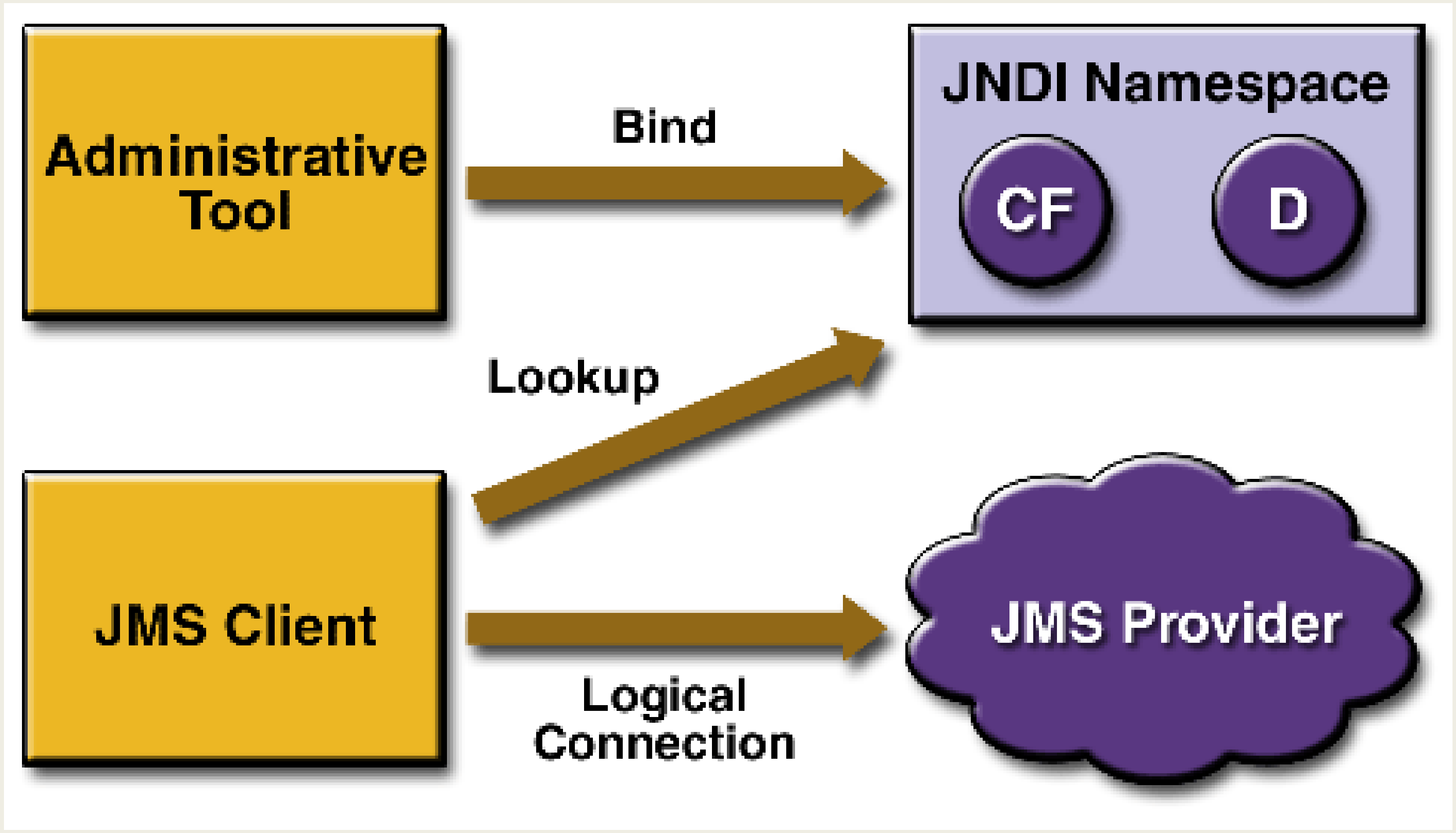
A JMS Application

- JMS Clients
  - *Java programs that send/receive messages*
- Messages
- Administered Objects
  - *preconfigured JMS objects created by an admin for the use of clients*
    - ConnectionFactory, Destination (queue or topic)
- JMS Provider (The Broker)
  - *messaging system that implements JMS and administrative functionality*

JMS Application Development

- To Create a JMS Sender Or JMS Consumer you need
  - *A ConnectionFactory*
    - To Connect to the Message Broker
  - *A Destination (Queue or Topic)*
    - To Send message to the destination
    - To receive a message from the destination

JMS Administration (In Java EE Server)



JMS Messages

- Message Header
  - *used for identifying and routing messages*
  - *contains vendor-specified values, but could also contain application-specific data*
  - *typically name/value pairs*
- Message Properties (optional)
- Message Body(optional)
  - *contains the data*
  - *five different message body types in the JMS specification*

JMS Message Types

Message Type	Contains	Some Methods
TextMessage	String	getText,setText
MapMessage	set of name/value pairs	setString,setDouble,setLong,getDouble,getString
BytesMessage	stream of uninterpreted bytes	writeBytes,readBytes
StreamMessage	stream of primitive values	writeString,writeDouble,writeLong,readString
ObjectMessage	serialize object	setObject,getObject

JMS API in a J2EE Application

- Since the J2EE1.3 , the JMS API has been an integral part of the platform
- J2EE components can use the JMS API to send messages that can be consumed asynchronously by a specialized Enterprise Java Bean
  - *message-driven bean*

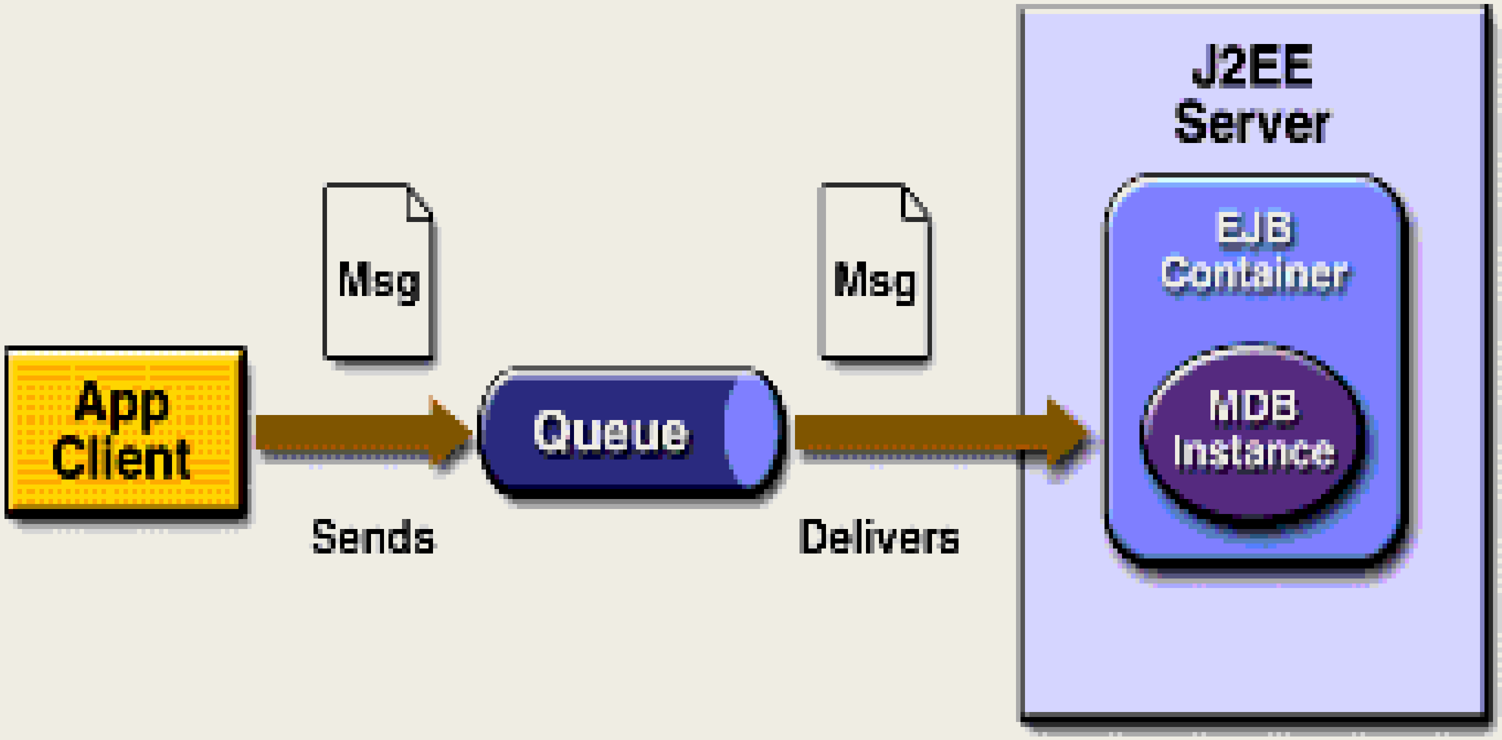
Enterprise Java Beans

- EJB is a server-side component that encapsulates the business logic of an application
- EJB simplifies the development of large, distributed applications
  - *EJB Container provides system-level services*
    - e.g. transaction management, authorization
  - *Beans have the control logic*
    - thin client applications
  - *Portable components*
    - can run on any compliant J2EE server

Message-Driven Bean

- acts as a listener for the JMS, processing messages asynchronously
- **specialized** adaptation of the JMS API used in the context of J2EE applications

JMS with EJB Example



MDB Example

```
public class MB implements MessageDrivenBean, MessageListener{
  public void ejbCreate(){}
  public void ejbRemove(){}
  public void setMessageDrivenContext(MessageDrivenContext mdc){}
  pubic void onMessage(Message m){
     //do computation on the incoming message
        try{ if (m instanceof TextMessage)
              System.out.println("MBean: message"+m.getText());
     }catch(JMSException exp){ ...}
  }
}
```

JMS WITH SPRING

JMS Simplified

AMQP

Advanced Message Queuing Protocol

Advanced Message Queuing Protocol

- Why should you care ?
  - *Many significant IT efforts include a messaging and integration component*
  - *(10%-30% of project cost)*

Limitations of exiting MOMs

- Proprietary middleware has been a source of lock-in.
  - *IBM MQ, Tibco Rendezvous , Sonic MQ.*
- Interoperability is more difficult than it need be.
  - *MQ Series and Tibco RV cannot natively interoperate with each other or other middleware products.*
- Language and platform independence is still an issue.
  - *JMS is not technology agnostic and only legitimately supports Java platforms.*

AMQP

- Conceived by JP Morgan around 2006.
- Goal was to provide a vendor-neutral protocol for managing the flow of messages across enterprise's business systems.

How does AMQP solve the problem?

- AMQP is a wire level protocol and not an API.
  - *JMS is an API.*
  - *Just like HTTP is for Internet, AMQP is for messaging.*
- When a protocol is specified at the wire-level and published, most technologies can use it.
- Compare this to an API, where the actual implementation is specific to the platform.

AMQP : Industry Support

- Cisco Systems
- Goldman Sachs
- IONA
- Novell
- Redhat
- WSO2
- Microsoft
- Credit Suisse
- Envoy
- Technologies
- iMatix
- JPMorgan Chase
- Rabbit
- Technologies
- TWIST
- 29West

AMQP CONCEPTS

With RabbitMQ Implementation

Virtual Hosts

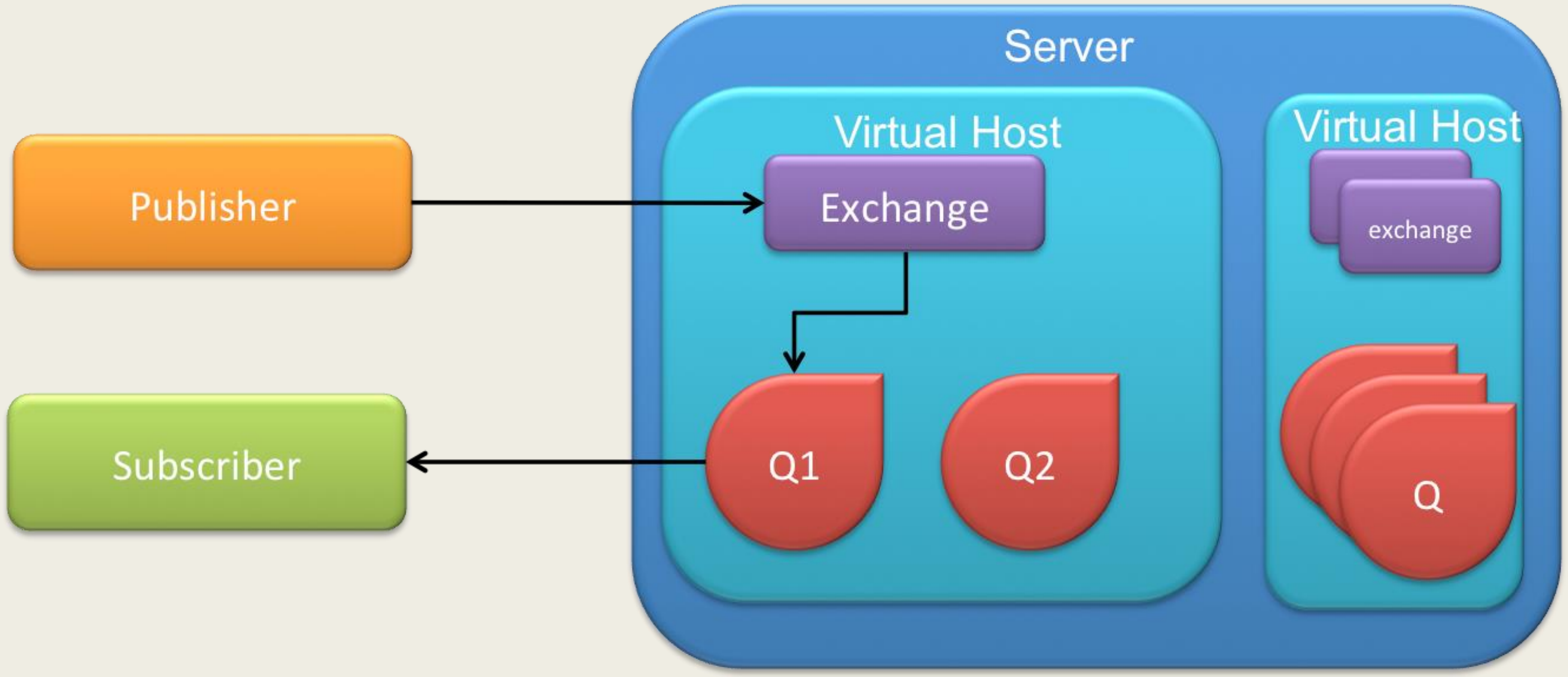


Image curtsey : https://arnon.me/2011/08/amqp/

Virtual Hosts

- Each virtual host comprises its own name space, a set of exchanges, message queues and all associated objects.

RabbitMQ Overview

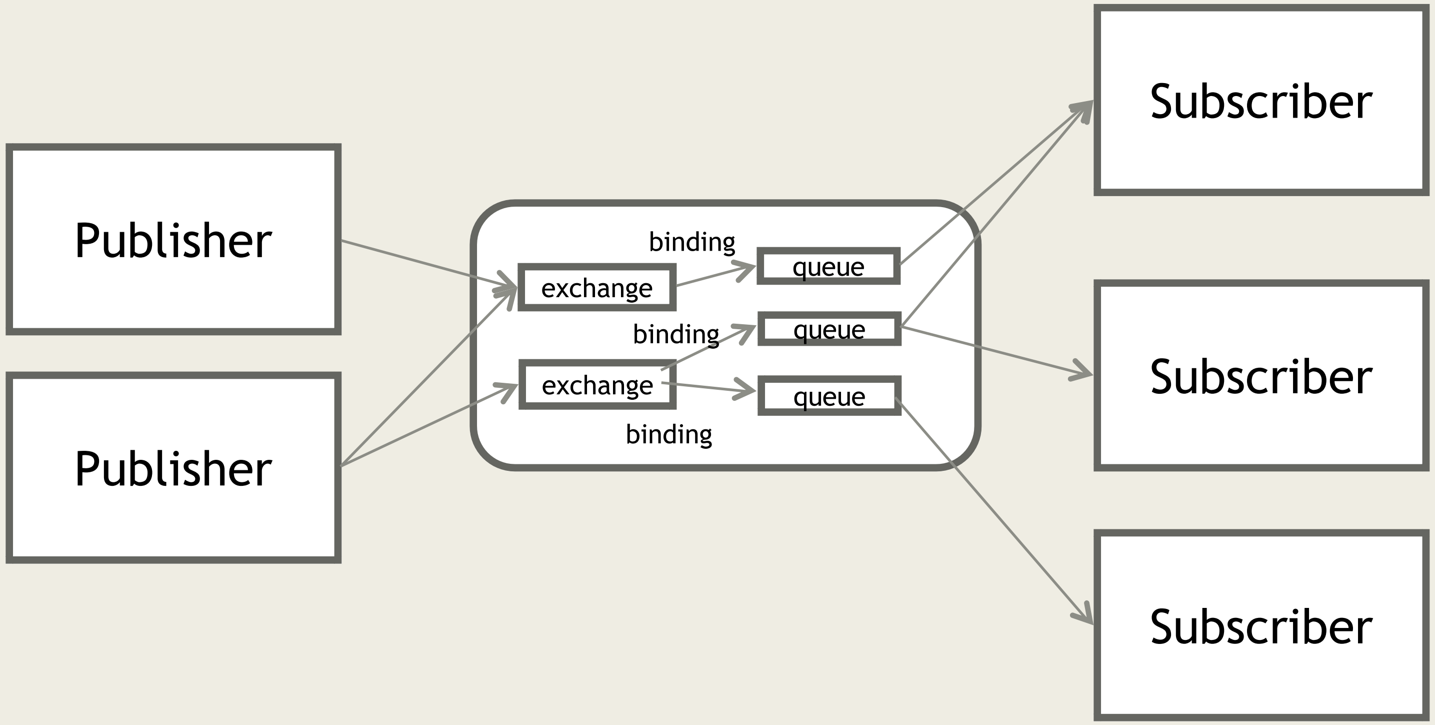
- The AMQP protocol defines:
  - ***exchanges*** *– the message broker endpoints that receive messages*
  - ***queues*** *– the message broker endpoints that store messages from exchanges and are used by subscribers for retrieval of messages*
  - ***bindings*** *– rules that bind exchanges and queues*

- The AMQP protocol is programmable – which means that the above entities can be created/modified/deleted by applications

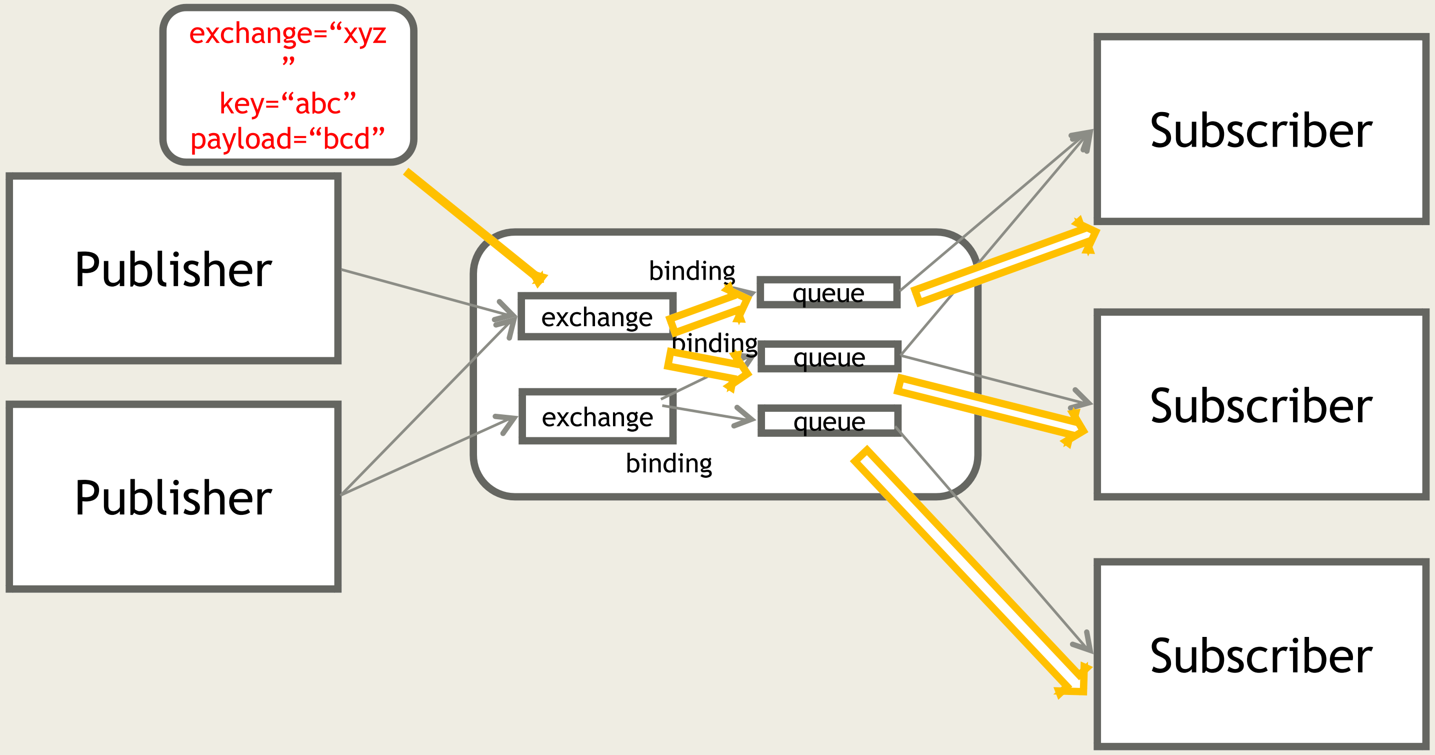
RabbitMQ Overview

- The AMQP protocol defines multiple connection channels inside a single TCP connection in order to remove the overhead of opening a large number of TCP connections to the message broker

RabbitMQ Overview



RabbitMQ Overview



RabbitMQ Overview

- Each message can be published with a **routing key**

- Each binding between an exchange and a queue has a **binding key**

- Routing of messages is determined based on matching between the routing and binding keys