Advanced Message Queuing Protocol (AMQP)

Prof SRN Reddy, IGDTUW

Advanced Message Queuing Protocol (AMQP)

- It is an open standard for passing business messages between applications or organizations.
- It connects systems, feeds business processes with the information

AMQP connects

- Organizations – applications in different organizations
- Technologies – applications on different platforms
- Time – systems don’t need to be available simultaneously
- Space – reliably operate at a distance, or over poor networks

Key Features

- **Wire-level Protocol**
- **Standard Protocol**
- **Secure**
- **Reliable**
- **Interoperable**
- **Publish/ Subcribe**
- **Open**

Safety

- Infrastructure for a secure and trusted global transaction network
  - Consisting of business messages that are **tamper proof**
  - Supporting message durability independent of receivers being connected
  - Message delivery is able to recover to technical failure
- Supports business requirements to transport business transactions of any financial value
  - Sender and Receiver are mutually agreed upon counter parties - No possibility for injection of Spam

FIDELITY

- Well-stated message queuing and delivery semantics covering:
  - at-most-once;
  - at-least-once;
  - once-and-only-once

- Well-stated reliable failure semantics so all exceptions can be managed

'wire-level' protocol- Advantage

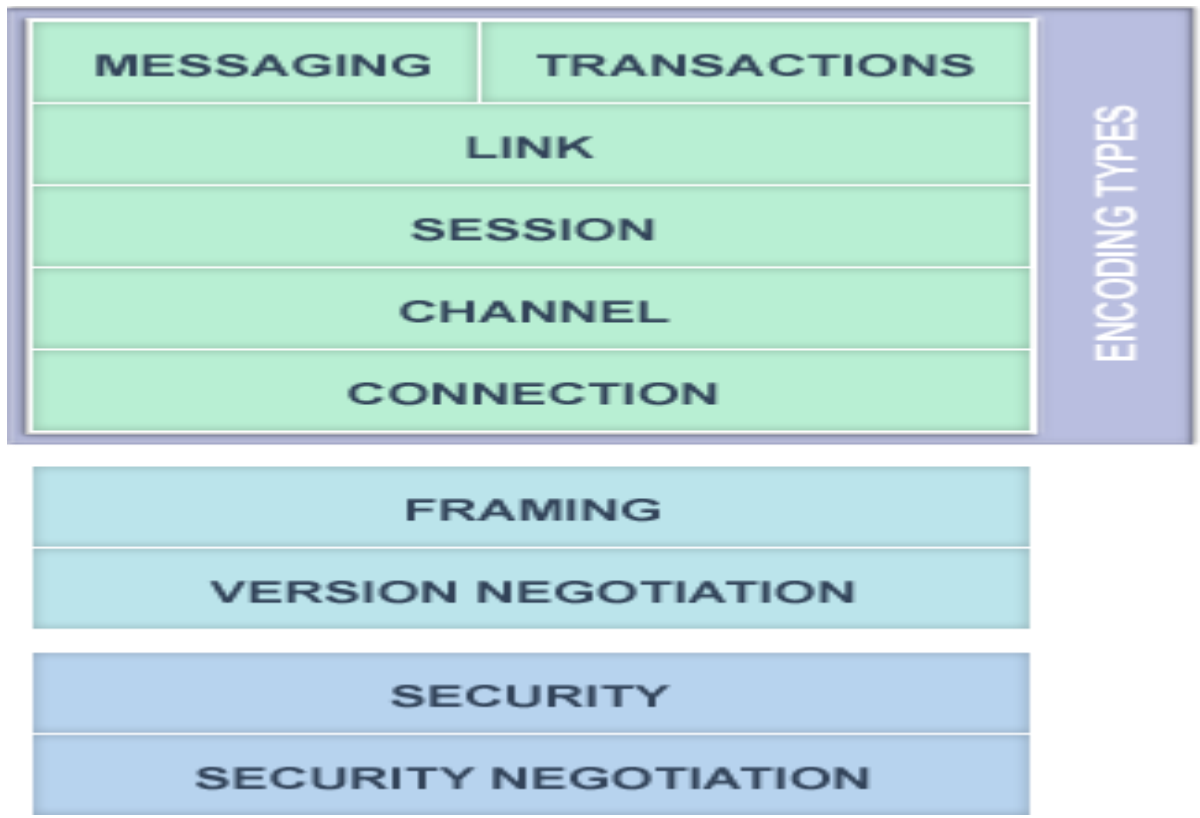
- **It is the complement of an API**. Instead of defining functions and creating libraries, you define the conversational byte sequences that pass over a network to make things happen.
- When a protocol is specified at the wire-level and published, **most technologies can use it, or be made to use it**. Compare this to an API, where the actual implementation is specific to the platform.
- JMS[Java Messaging Service] is an API. HTTP is a protocol. AMQP delivers the middleware equivalent of HTTP while leaving it up to others to provide implementations with he concept of wire-level Protocol

Standard Protocol

- Create a standard ratified by an official standards organization
- Qualify for an Internet Engineering Task Force (**IETF**) **Request for Comments** (**RFC**), there needs to be at least two independent implementations built from the specifications that are proven to inter-operate
- Many of the Working Group members are involved in creating implementations of AMQP

AMQP Users: 500+ commercial users

- **JPMorgan:** Sends 1 billion AMQP messages per day; used in dozens of mission critical systems worldwide.
- **National Science Foundation:** The Ocean Observatories Initiative infrastructure uses sensor nets with AMQP to bring readings ashore from ocean platforms to disseminate readings.
- **NASA:** The control plane of the Nebula Cloud Computing.
- **Red Hat: to** control its internal operations.
- **Vmware:** Makes extensive use of RabbitMQ in its virtualization products and cloud services.
- **Google:** The open-source Rocksteady project uses RabbitMQ and complex event processing to analyse user defined metrics. Its goal is to allow root cause diagnosis of breakages in real time.
- **UIDAI, Government of India:** UIDAI uses RabbitMQ to decouple sub-components of its application allowing it to scale for more than 1.2 billion users
- **OpenStack:** OpenStack is an open-source initiative that provides a massively scalable cloud operating system and uses RabbitMQ for messaging.
- **AT&T:** RabbitMQ is widely used at AT&T Interactive, the local search provider.
- **IBM:** IBM MQ Light, is a simple and scalable application development tool


MESSAGING
TRANSACTIONS
LINK
SESSION
CHANNEL
CONNECTION
ENCODING TYPES
FRAMING
VERSION NEGOTIATION
SECURITY
SECURITY NEGOTIATION

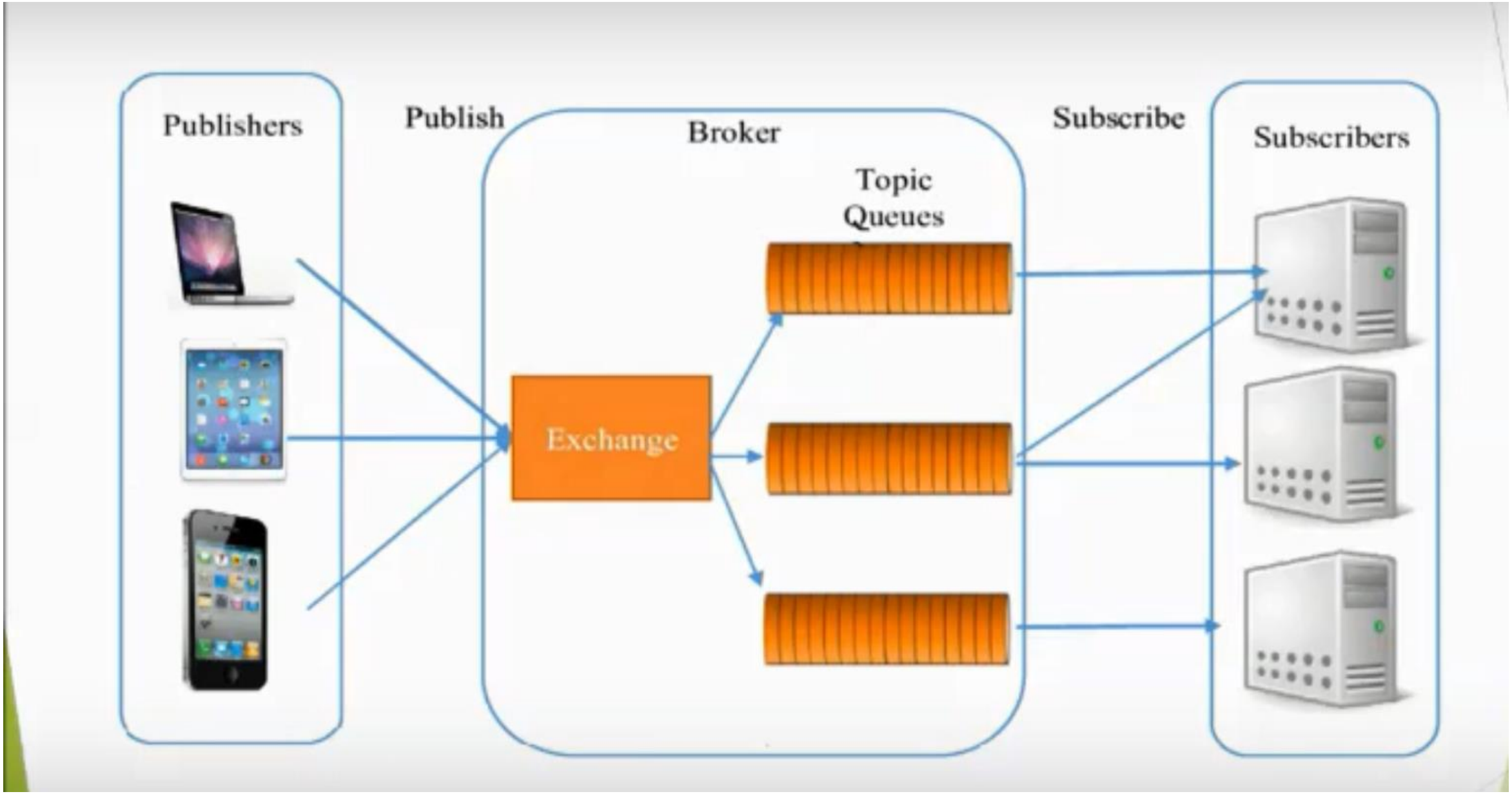
Need of AMQP over proprietary

- Realize the savings commoditization brings**[remove vendor lock-in]**
- Connect applications on different platforms; [**choose the right platform for the job]**
- Connect to business partners using a full featured open standard; **[remove technical barriers to trade]**
- Position for innovations built upon the foundation of AMQP

AMQP - Advanced Message Queuing Protocol

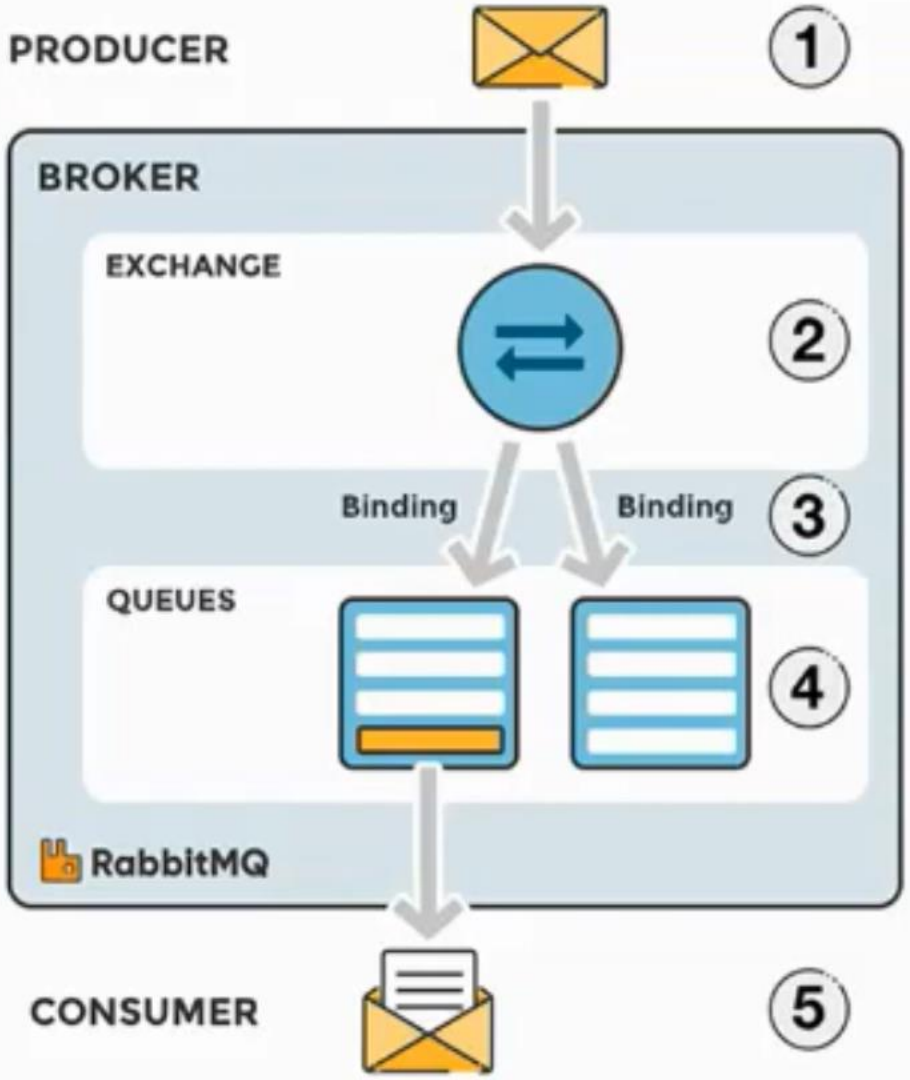
- Publisher : Client apps create messages which are given to broker
- Exchanges : This is the place where publishers deliver messages. The messages contain routing keys which are used by "exchange" module in order to route them (i.e. messages).
- Binding : This is a "link" that is set up to bind a queue to an exchange.
- Queues : Exchange queue the message.
- Consumer/Subscriber : Client apps subscribe messages from broker
- Routing Key : Routing key is a message attribute. The exchange might look at this key when deciding how to route the message to queues (depending on exchange type).

AMQP Protocol Architecture



AMQP - Standard RabbitMQ message flow

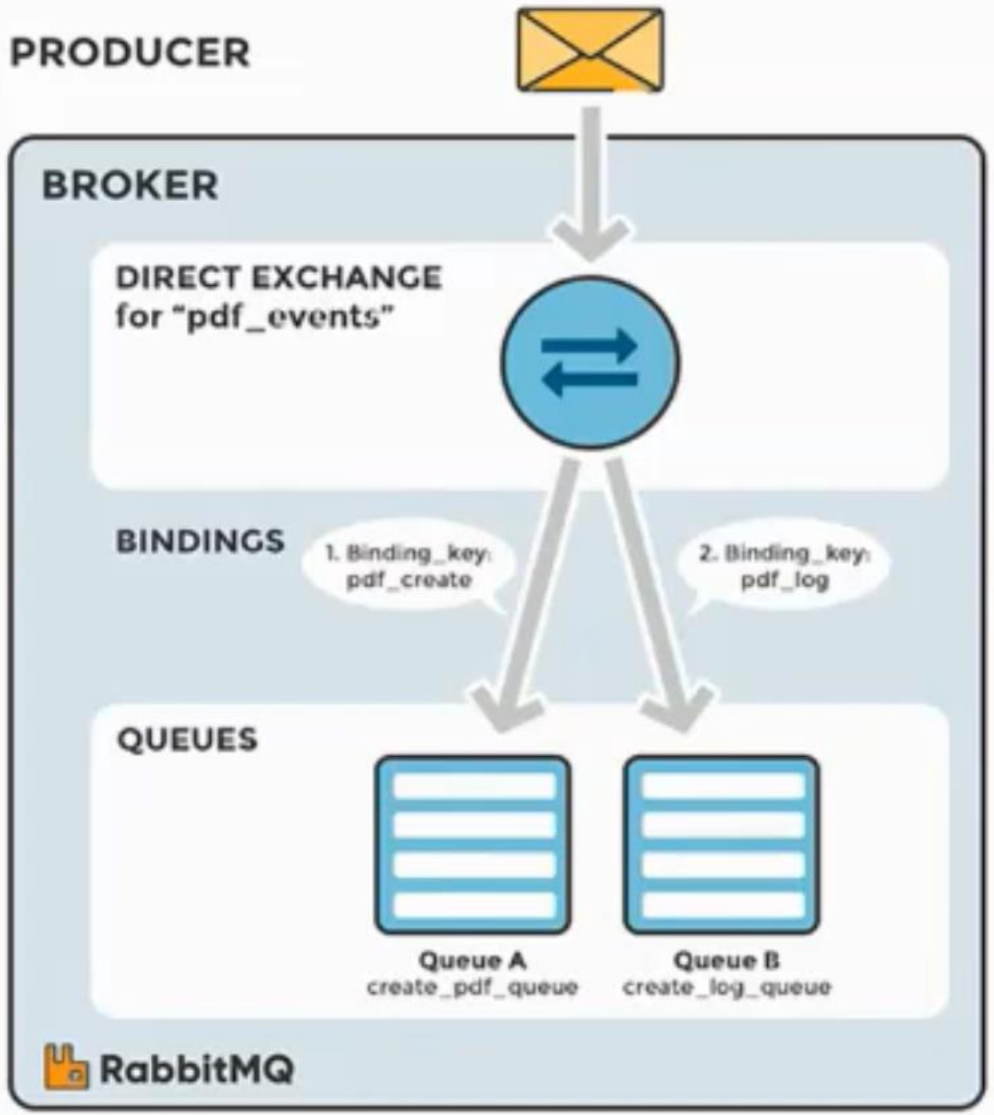
1) The producer publishes a message to the exchange.
2) The exchange receives the message and is now responsible for the routing of the message.
3) A binding has to be set up between the queue and the exchange. In this case, we have bindings to two different queues from the exchange. The exchange routes the message in to the queues.
4) The messages stay in the queue until they are handled by a consumer
5) The consumer handles the message.



AMQP – Direct Exchange

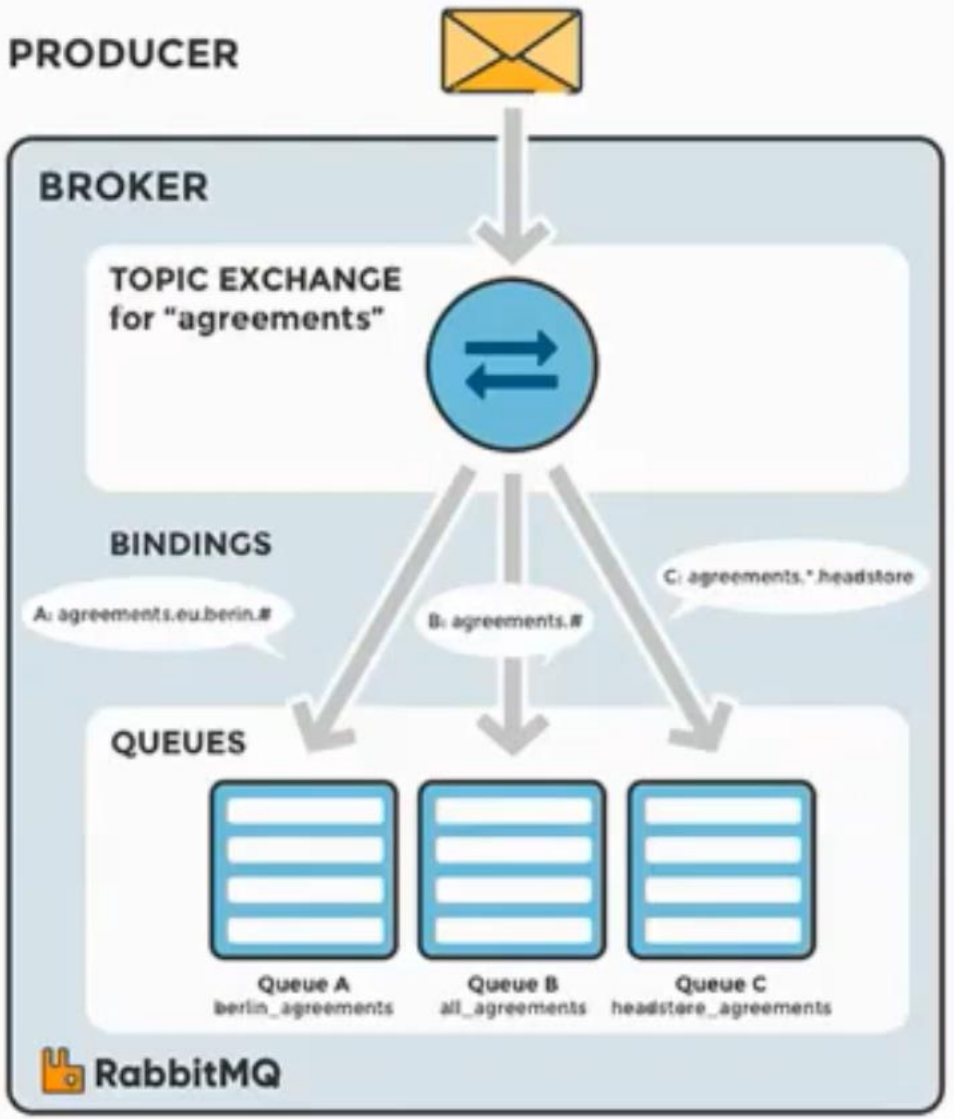
- A direct exchange delivers messages to queues based on a message routing key. The routing key is a message attribute added into the message header by the producer. The routing key can be seen as an "address" that the exchange is using to decide how to route the message. **A message goes to the queue(s) whose binding key exactly matches the routing key of the message.**

- Example: A message with routing key *pdf_log* is sent to the exchange *pdf_events*. The messages is routed to pdf_log_queue because the routing key (pdf_log) match the binding key (pdf_log).
- If the message routing key does not match any binding key, the message is discarded.



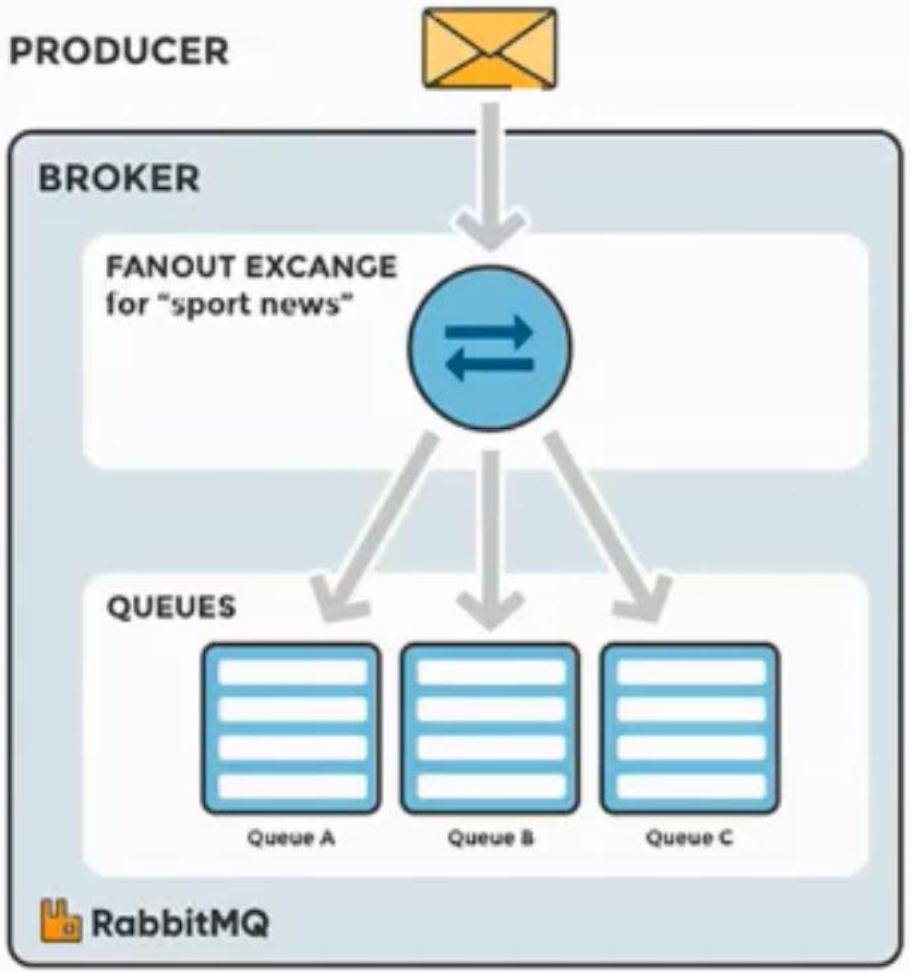
AMQP - Topic Exchange

- Topic exchanges route messages to queues based on wildcard matches between the routing key and something called the routing pattern specified by the queue binding. Messages are routed to one or many queues based on a matching between a message routing key and this pattern.
- With topic exchanges, messages are published with routing keys containing a series of words separated by a dot (e.g. "word1.word2.word3"). Queues binding to a topic exchange supply a matching pattern for the server to use when routing the message. Patterns may contain an asterisk ("*") to match a word in a specific position of the routing key, or a hash ("#") to match zero or more words. For example, a message published with a routing key of "honda.civic.navy" would match queues bound with "honda.civic.navy", "*.civic.*", "honda.#", or "#", but would not match "honda.accord.navy",
- Example: A message with routing key *agreements.eu.berlin* is sent to the exchange *agreements*. The messages are routed to the queue *berlin_agreements* because the routing pattern of "agreements.eu.berlin.#" matches any routing keys beginning with "agreements.eu.berlin". The message is also routed to the queue *all_agreements* because the routing key (agreements.eu.berlin) match the routing pattern (agreements.#).



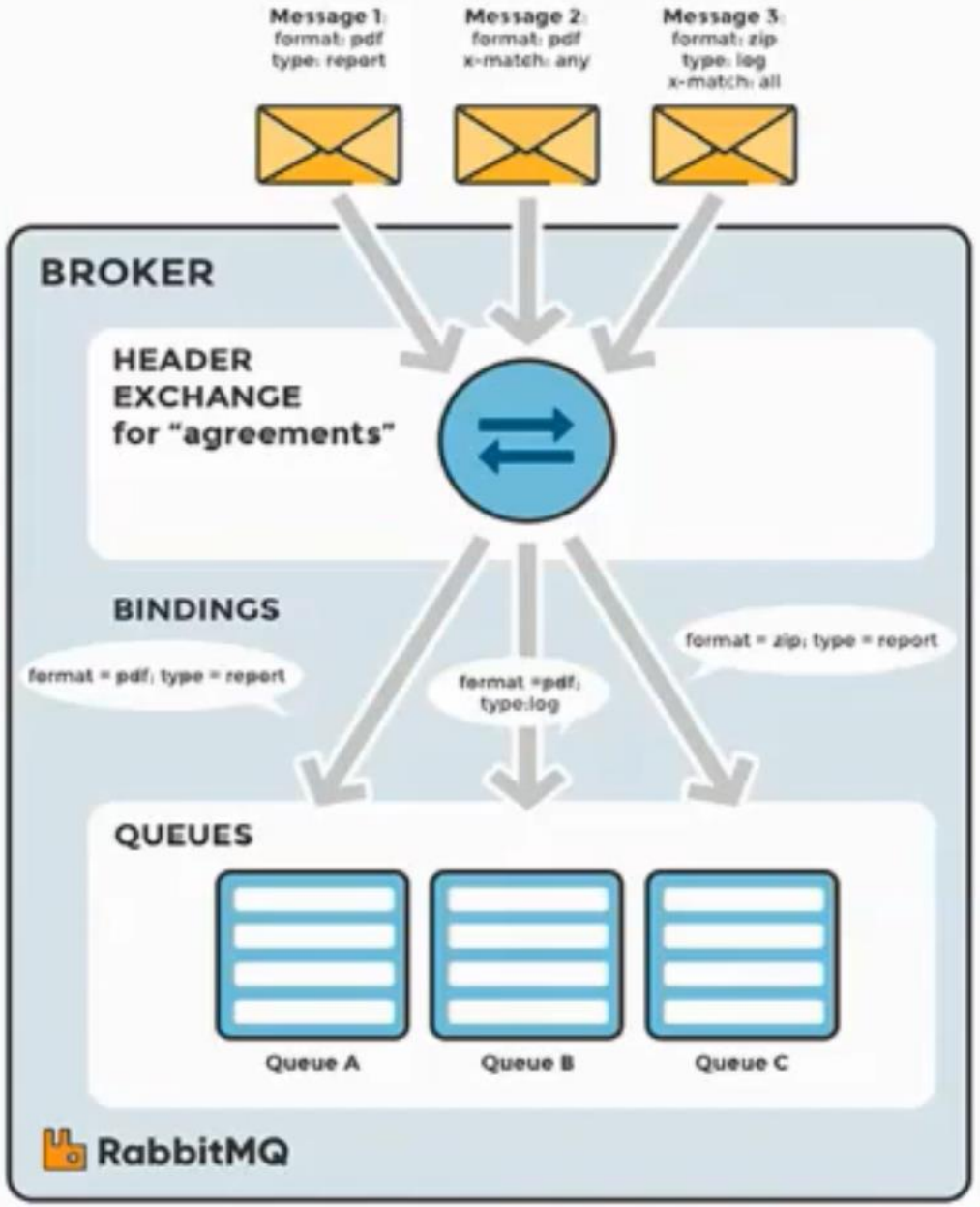
AMQP – Fanout Exchange

- The fanout copies and routes a received message to all queues that are bound to it regardless of routing keys or pattern matching as with direct and topic exchanges. Keys provided will simply be ignored.
- Fanout exchanges can be useful when the same message needs to be sent to one or more queues with consumers who may process the same message in different ways.
- Example: A message is sent to the exchange *sport_news*. The message is routed to all queues (Queue A, Queue B, Queue C) because all queues are bound to the exchange. Provided routing keys are ignored.

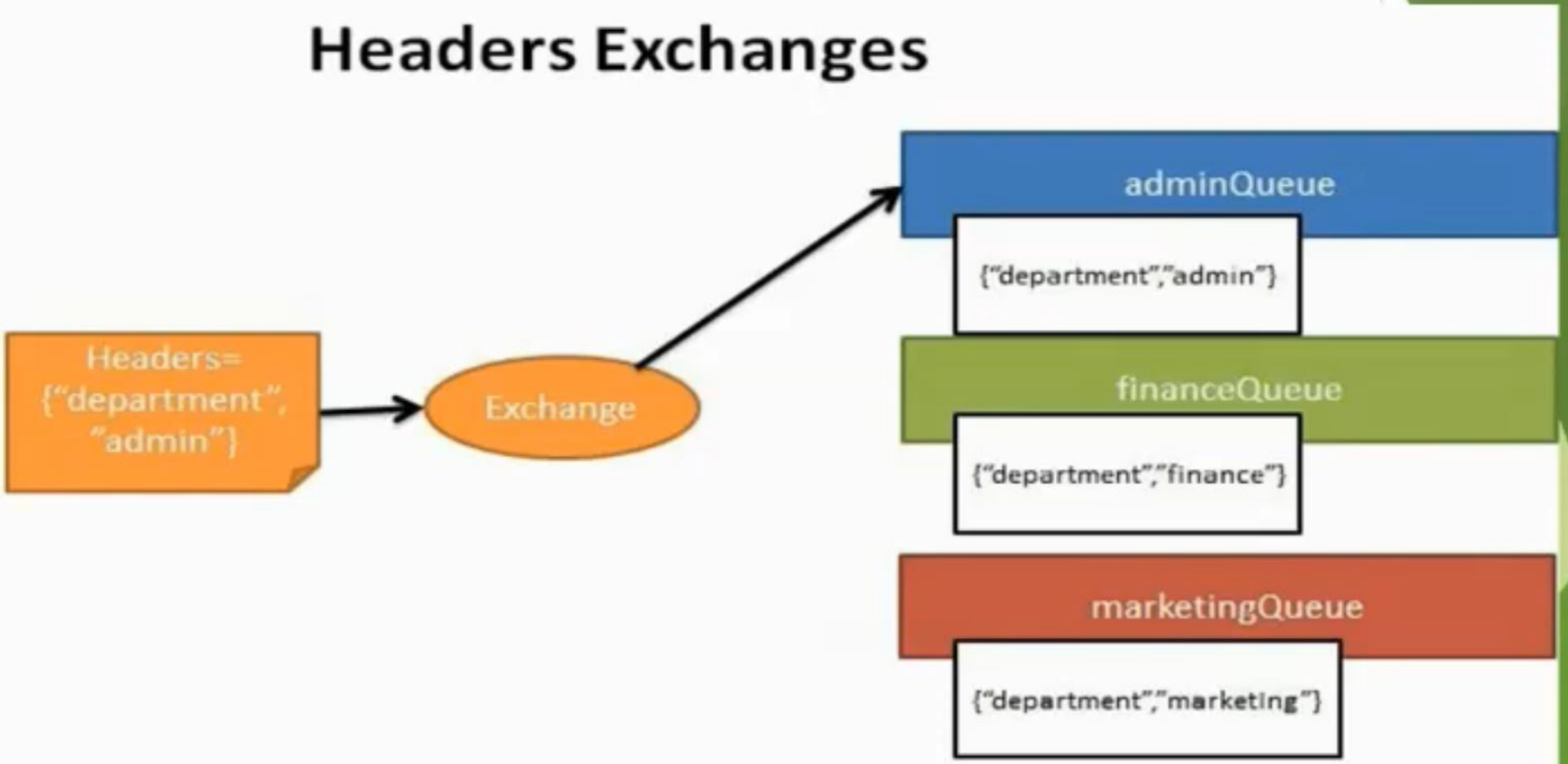


AMQP - Header Exchange

- The Header exchange type routes messages based upon a matching of message Header to the expected Header specified by the binding queue. The Header exchange type is similar to the topic exchange type in that more than one criteria can be specified as a filter, but the Header exchange differs in that its criteria is expressed in the message Header as opposed to the routing key, may occur in any order, and may be specified as matching any or all of the specified headers.
- A special argument named "x-match", which can be added in the binding between your exchange and your queue, tells if all Header must match or just one. Either any common header between the message and the binding count as a match, or all the Header referenced in the binding need to be present in the message for it to match. The "x-match" property can have two different values: "any" or "all", where "all" is the default value. A value of "all" means all header pairs (key, value) must match and a value of "any" means at least one of the header pairs must match. Header can be constructed using a wider range of data types - integer or hash for example instead of a string. The Header exchange type (used with the binding argument "any") is useful for directing messages which may contain a subset of known (unordered) criteria.



AMQP - Header Exchange



MQTT vs AMQP

MQTT is a light protocol which does less, leaving you to make up the gap with code!

- No queues! (sender and receiver must be up simultaneously)
- No persistence / durability / archival or recovery
- No JMS or WCF compatibility
- No transactions for application server or XA integration
- No flow-control or selective ACK to prevent application lock-ups
- No multiplexing for easy firewall traversal
- No Kerberos (Active Directory)

	HTTP	AMQP
Get	Y	Y
Caching Read	Y	N
Put	Y	N
Post	Y	Y
Delete	Y	N
Content filtering	N	Y
Typed headers	N	Y
Resumeable transfers	Y	Y
Transactions	N	Y
SSL/TLS	Y	Y
Kerberos	Y	Y
SASL	N	Y
Symetric Protocol	N	Y
Socket Multiplexing	N	Y
Out-of-order messaging	N	Y
Server initiated transfers	N	Y
Single packet send	Y	Y
Store-and-forward	N	Y
Publish-and-subscribe	N	Y
Defined Error Recovery	N	Y
Well defined addresses	Y	Y
Content based routing	N	Y
Credit-based flow control	N	Y

S,

References

1. https://www.amqp.org/about/what
2. https://www.youtube.com/watch?v=SXZJau292Uw[Brief Video]
3. http://www.amqp.org/sites/amqp.org/files/2014.05.01%20ISO%2019464%20AMQP-ORG_0.pdf
4. https://www.youtube.com/watch?v=wPnrb0KjTFU [video]
5. https://www.youtube.com/watch?v=ODpeIdUdClc[Imp Video]

Q & A

Thanks