MOM

MESSAGE ORIENTED MIDDLEWARE

OVERVIEW OF MESSAGE ORIENTED MIDDLEWARE TECHNOLOGIES AND CONCEPTS

Peter R. Egli
INDIGOO.COM

Contents

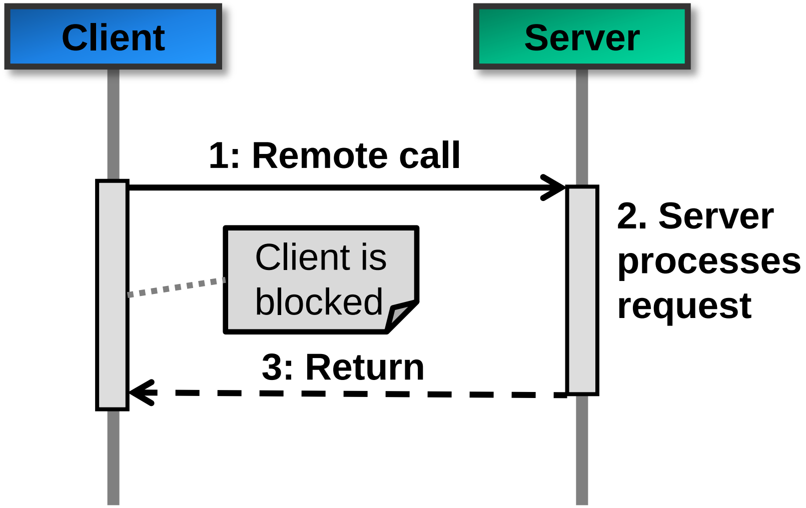
1. Synchronous versus asynchronous interaction
2. Messaging models
3. Queue types
4. Message broker - application integration pattern
5. Features of message queue systems
6. Examples of MOM middleware
7. Comparison of JMS, MSMQ and AMQP middleware
8. Comparison of MOM with Internet messaging

1. Synchronous versus asynchronous interaction (1/2)

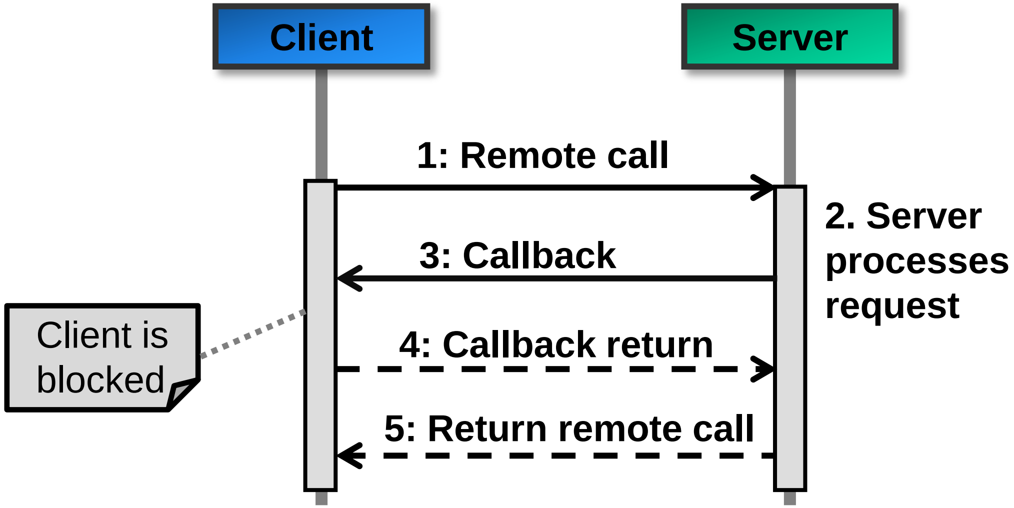
Distributed Object Technologies (DOT), RPC:

→ Synchronous operation (caller is blocked until callee returns).

Simple remote call:



Remote call with callback:



Problems / drawbacks with this model of operation:

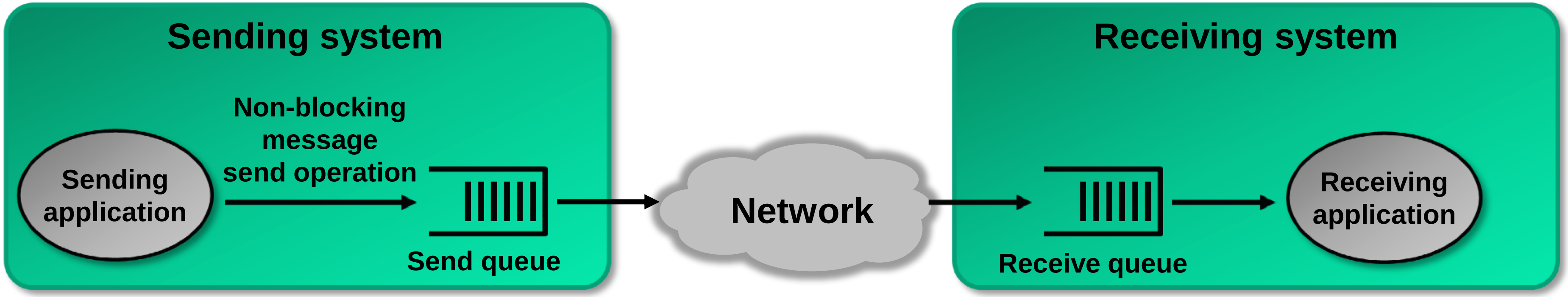
- ☹ The client is blocked until the server (object operation) call returns (tight coupling).
- ☹ Connection overhead (each call needs marshalling, entails protocol overhead for network access etc.).
- ☹ Difficult to react to failures (server may throw an exception, may not be active etc.).
- ☹ Not well-suited for nested calls (server object calls back client object which potentially calls another server object operation).

1. Synchronous versus asynchronous interaction (2/2)

Message Oriented Middleware (MOM):

→ Asynchronous operation (caller sends a message and continue its work, „fire and forget“).
→ Store and Forward communication.
→ Sender & receiver are loosely coupled:

a. They do not need to be active at the same time.
b. The sender does not need to know the receiver location and vice versa.



Analogy:

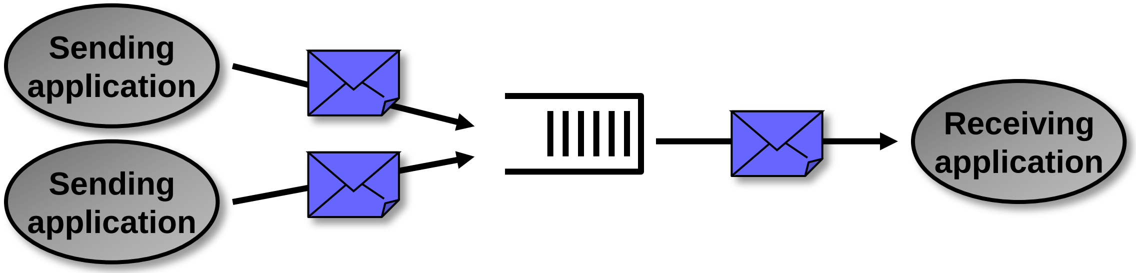
Synchronous (RPC/DOT) → ☎ Telephone

Asynchronous (MOM) → ✉ Mail

2. Messaging models

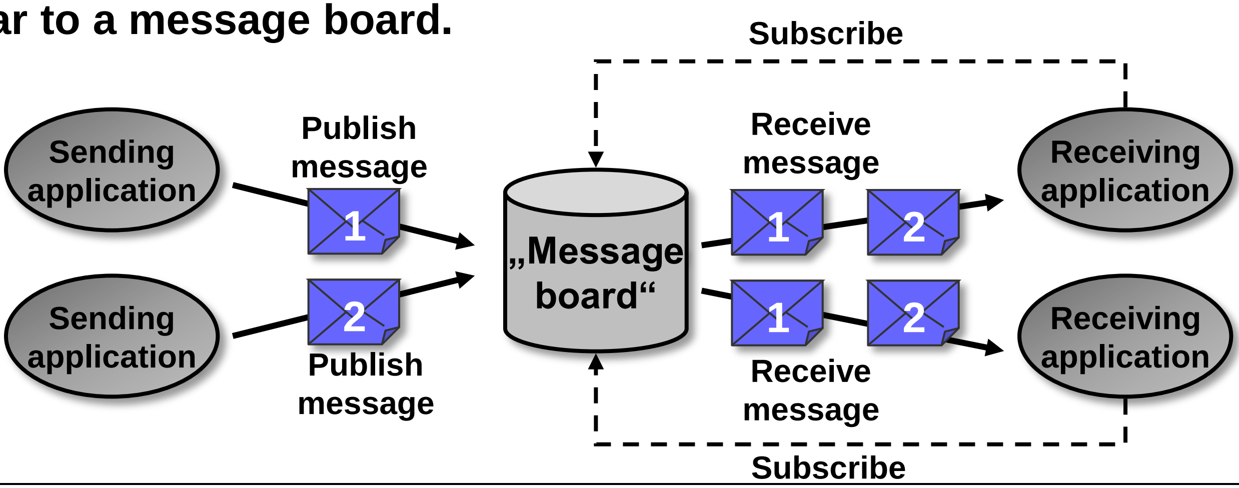
1. P2P - Point to Point:

- 1 queue per receiver (application).
- One-to-one (1 sender, 1 receiver) or many-to-one messaging (many senders, 1 receiver).



2. Publish – Subscribe („PubSub“):

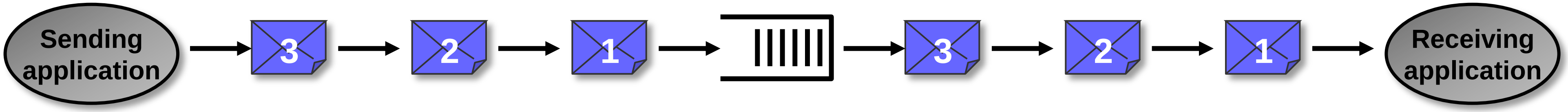
- One-to-many or many-to-many distribution of messages (same message may be received by multiple receivers if these are subscribed).
- Similar to a message board.



3. Queue types (1/2)

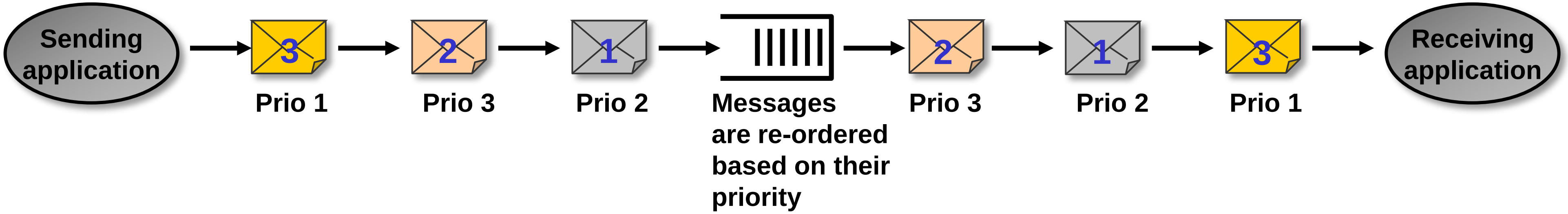
FIFO queues (first in, first out):

- All messages have the same priority level.
- Messages are delivered in the order they are sent.



Priority queues:

- Messages are buffered in FIFO queues and ordered based on priority.
- N.B.: The ordering applies to the set of messages that are in the queue at a specific point in time (= messages that are not yet received by an application).



3. Queue types (2/2)

Public / private queues:

Defines different access rights:

1. **Public queue:** **All senders may send messages without access control.**
2. **Private queue:** **Sending to a private queue requires sender authentication.**

Journal queue:

The message queueing system keeps a copy of every received message for logging or monitoring purposes.

Dead-letter queue:

Queue that holds undeliverable messages (messages that time out due to time-to-live (TTL) expiry or whose queue address could not be resolved).

Bridge / connector queue:

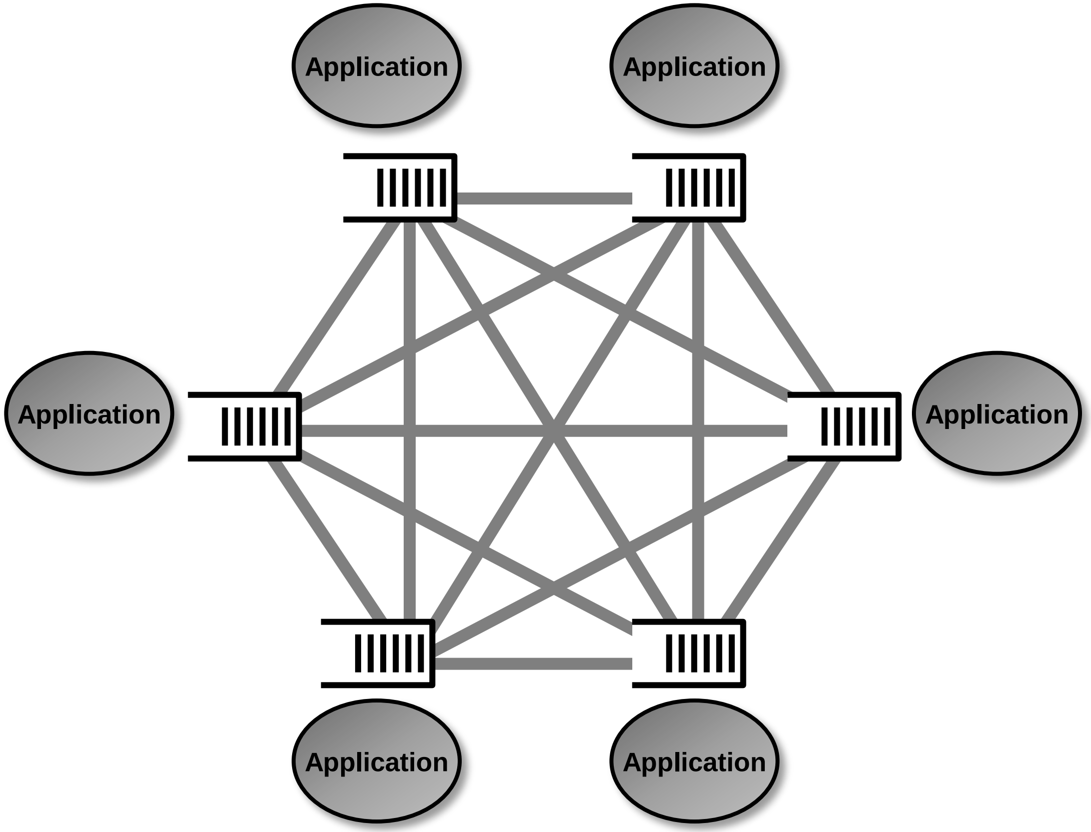
Connects different queue systems, e.g. Microsoft Message Queue (MSMQ) and Java Messaging System (JMS) based messaging systems.

4. Message broker - application integration pattern (1/4)

Message brokers distribute messages to receivers.

Case 1: No message broker:

- Requires n*(n-1)/2 „connections" between message queues. Every (endpoint) queue needs to know all other queues to send message to these queues.
- The sender must know the location (address) of the receiver.
- This model becomes complex for large numbers of queues (it does not scale).



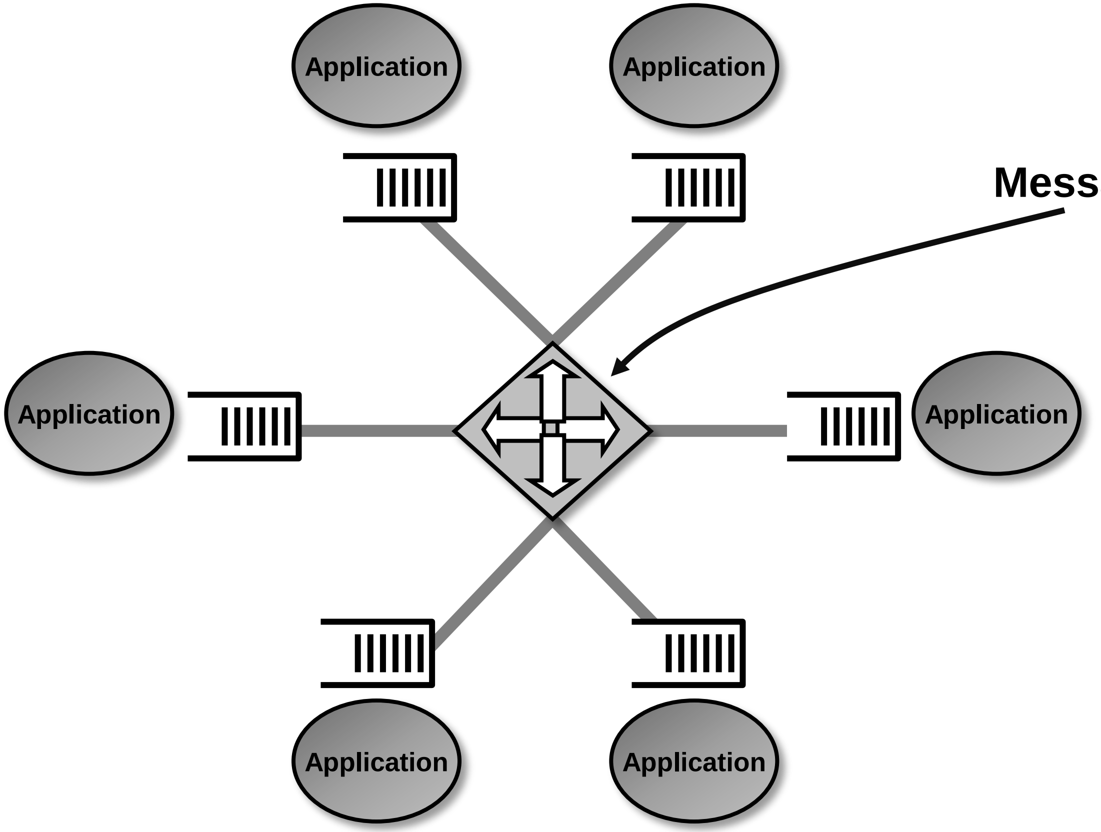
Bidirectional connections:
6 systems → 6 * 5 / 2 = 15 „connections"

If every application needs to send messages to all other applications, 6 * 5 = 30 „connections" are required.

4. Message broker - application integration pattern (2/4)

Case 2: Message broker:

- The message broker serves as a central exchange of messages (hub and spoke architecture, broker routes messages to the destination queue).
- A message broker provides additional decoupling between senders and receivers.
- The broker may perform additional functions such as filtering, message transformations (e.g. enrich messages with data from a DB) and load balancing.

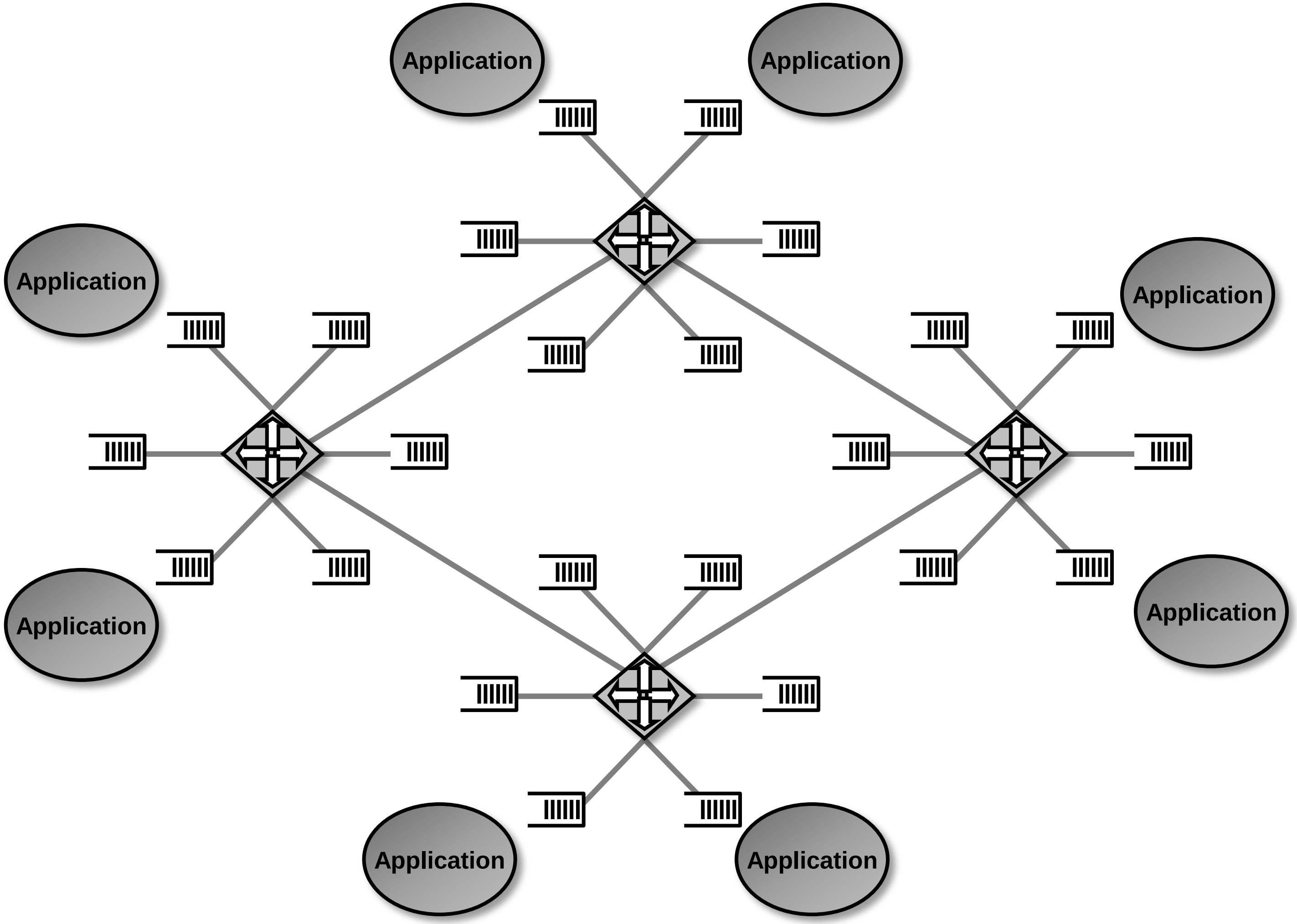


Message broker = central message exchange

4. Message broker - application integration pattern (3/4)

Case 3: Multi-hub message broker system:

- Generalization of message hub architecture (hierarchy of message hubs).



4. Message broker - application integration pattern (4/4)

Case 4: Federated message brokers:

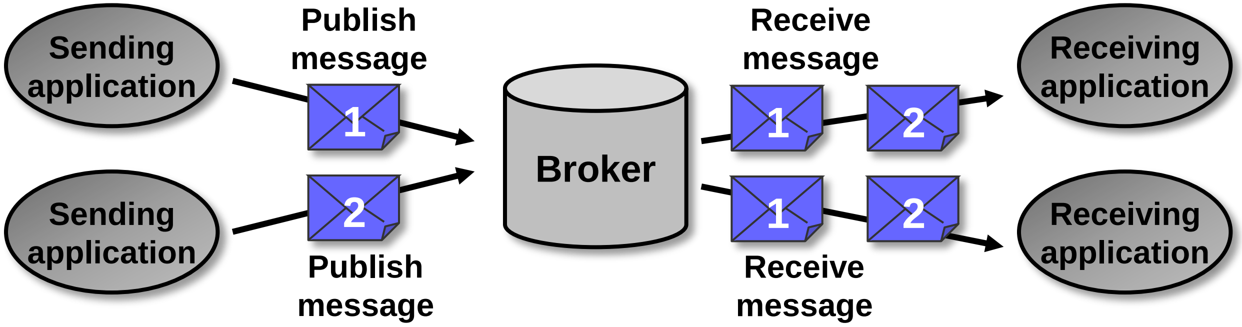
- Generalization of the multi-hub message broker pattern.
- Applications are bound to a specific message broker („home message broker“).
- Message brokers are under the responsibility of different organizations (federation).

Case 5: PubSub broker:

Rather than distributing messages to queues, the message broker routes messages to subscribers.

Thus messages may be sent to multiple receivers (multicast).

In JMS a PubSub broker is a called "topic".



5. Features of message queue systems (1/4)

Asynchronous operation:

Sending of messages is unblocking. The sender application may continue its work, the sender queue tries to deliver the message on behalf of the sender application (until successful).

Transaction support:

Sending and receiving a series of messages may be „packed" into a transaction. Either all messages are successfully sent and received or none.

In-order delivery:

Messages are queued in the order they are sent. However, messages may „overtake" messages other based on priorities.

Priority-based delivery:

Messages are queued according to a priority scheme (the receiver queue passes messages with highest priority first to the receiving application).

5. Features of message queue systems (2/4)

Message formatting:

Possibility to „wrap“ the messages into formats such as SOAP (messages are wrapped in SOAP over HTTP protocols for better Internet traversal), XML or plain text.

Notification services (triggers):

Receiver:	The queue sends notifications of new enqueued messages to the receiver.
Sender:	The queue sends notifications of the successful delivery of sent messages to the sender.

Message filtering:

The queue performs filtering based on different criteria:

- Message properties (message header fields), e.g. priority.
- Message body (e.g. SQL expression).

Message routing:

- Message forwarding through intermediate message queues (= brokers).
- Message routing may be based on different criteria (e.g. current workload on destination queues for load balancing).

5. Features of message queue systems (3/4)

Message security:

- Apply security functions like message authentication, encryption and message integrity.

Supported message transport protocols:

Messages may be transported over a range of different transport protocols.
Typical transport protocols used for message transport are:

- TCP or UDP (simplest transport protocol)
- HTTP or HTTPs (good for sending messages over the Internet)
- SMTP
- FTP
- Messaging system proprietary transport protocol

Message peek and receive:

Peeking allows a receiving application to receive a copy of a message from a queue.
The message is left in queue.
Only a receive operation actually removes the message from the queue ("pop" a message).

5. Features of message queue systems (4/4)

Delivery mode, Quality of Service (QoS):

Exactly-once (guaranteed delivery, highest QoS):

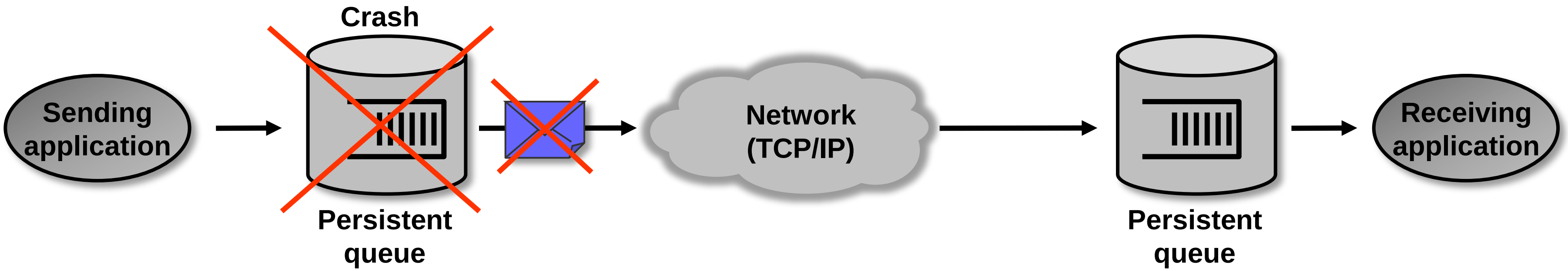
- **Persistent mode, messages survive queue crashes.**
- **The sending queue keeps the message in persistent store until it receives a positive acknowledge of the correct reception of the message by the target application.**

At-least-once (guaranteed delivery):

- **Guaranteed delivery, but duplicates of messages may be sent to the application.**
- **The sending queue keeps the message in persistent store (like exactly-once).**
- **After a crash, the sending queue does not query the receiving queue if it already has received the message, but just re-sends the message.**
- **If the sending queue crashed just prior to receiving the positive acknowledge from the receiving queue, the message is delivered twice.**

At-most-once (no delivery guarantee, lowest QoS):

- **Non-persistent mode.**



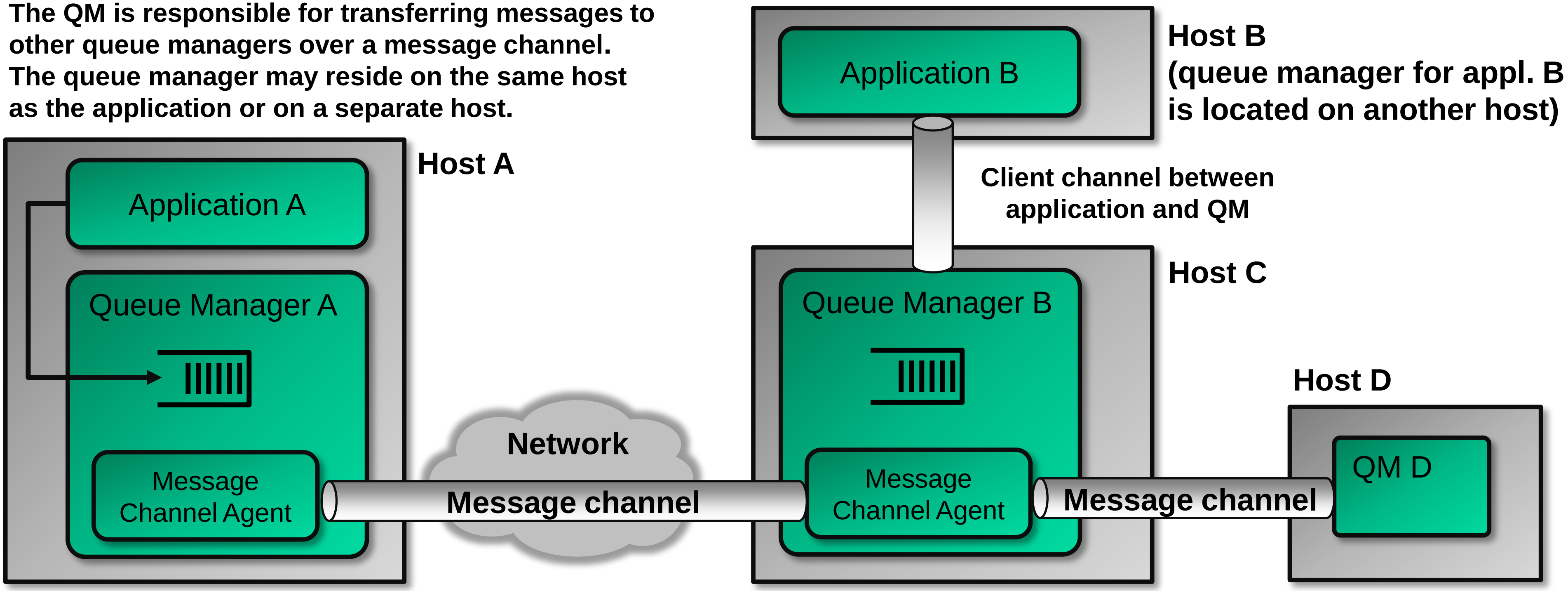
6. Examples of MOM middleware (1/7)

IBM WebSphere MQ (1/3):

- Multiplatform MOM from IBM. Available on IBM platforms, .Net, Linux etc.
- Various APIs such as JMS, XMS (JMS API for .Net, C/C++), MQI (MQ native interface).

Main elements of WebSphere MQ:

Queue manager:
Container for a message queue.
The QM is responsible for transferring messages to other queue managers over a message channel.
The queue manager may reside on the same host as the application or on a separate host.



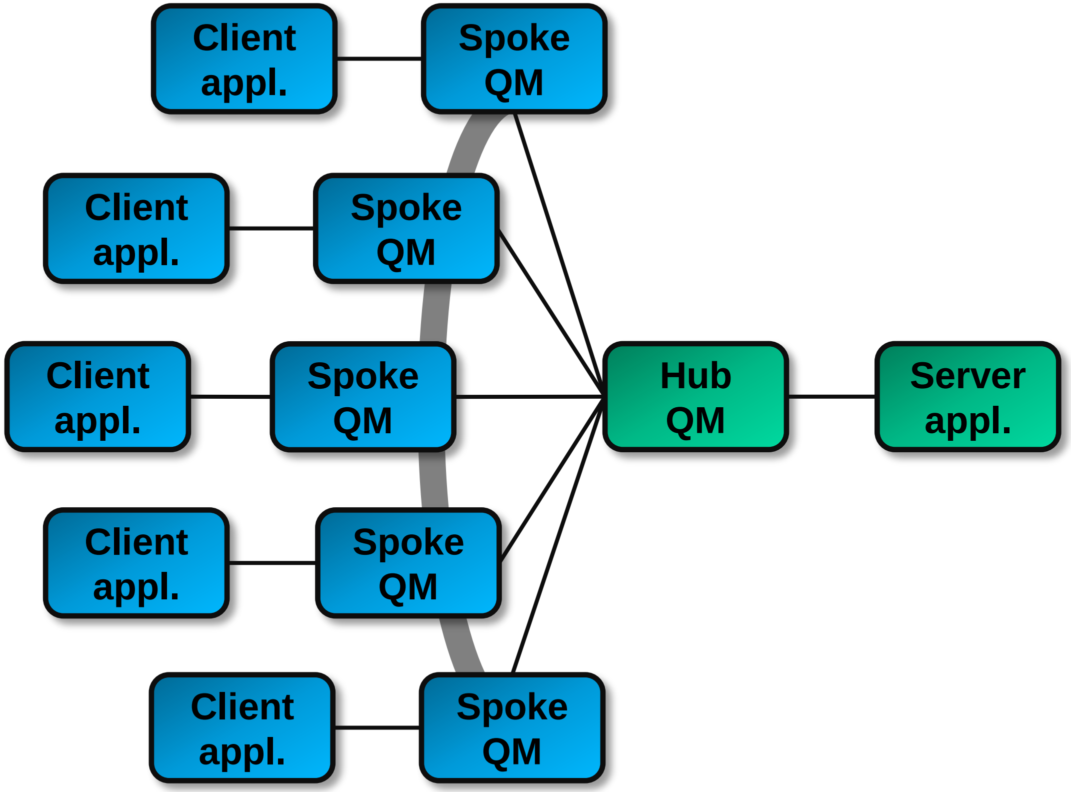
6. Examples of MOM middleware (2/7)

IBM WebSphere MQ (2/3):

The main supported topologies are:

1. Hub and spoke topology (point-to-point queues):

- **Applications subscribe to "their" queue manager.**
- **Routes to hub QM are manually defined in spoke QMs.**



6. Examples of MOM middleware (3/7)

IBM WebSphere MQ (3/3):

2. Distributed PubSub:

- **Applications subscribe to topics and send messages to multiple receivers (publish).**
- **2 Topologies: Clusters and trees.**

2.1. Cluster:

Cluster of queue manager connected by channels between QMs.
Published messages are sent to all connected queue managers of the published topic.

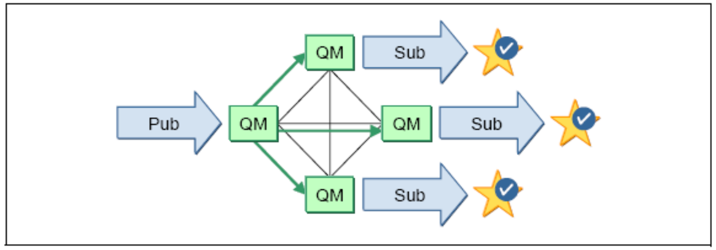


Figure 4-3 Publish/Subscribe clusters

PubSub cluster topology (source: www.redbooks.ibm.com "WebSphere MQ V7.0 Features and Enhancements")

2.2 Tree:

Trees allow reducing the number of channels between the QMs.

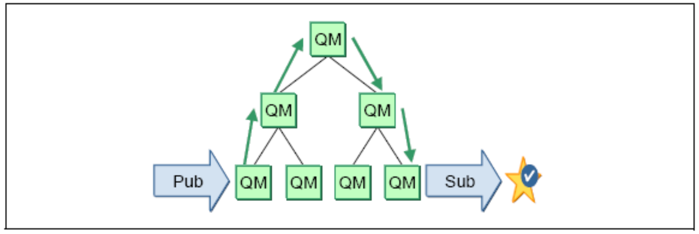


Figure 4-4 Hierarchical distributed queue managers

PubSub tree topology (source: www.redbooks.ibm.com "WebSphere MQ V7.0 Features and Enhancements")

6. Examples of MOM middleware (4/7)

Sonic Software SonicMQ:

- P2P and PubSub messaging.
- Exactly-once delivery semantics.
- Support for message broker clusters with load balancing.
- Various messaging bridges to other queueing systems (JMS, IBM WebSphere MQ, Tibco Rendezvous, FTP, Email).

Microsoft MSMQ:

- Guaranteed message delivery (message delivered even when queue is temporarily down).
- Message routing.
- Security (optional authentication and encryption).
- Priority-based messaging.
- Different message transport protocols.
- Bridge to IBM MQSeries messaging system available.

6. Examples of MOM middleware (5/7)

AMQP – Advanced Message Queueing Protocol

Why AMQP?

1. Lack of standardization:

There is little standardization in MOM products (mostly proprietary solutions).
For example, JMS is dependent on Java and does not specify a wire protocol but only an API.
Therefore different JMS providers are not directly interoperable on the wire level.

2. Need for bridges for interoperability:

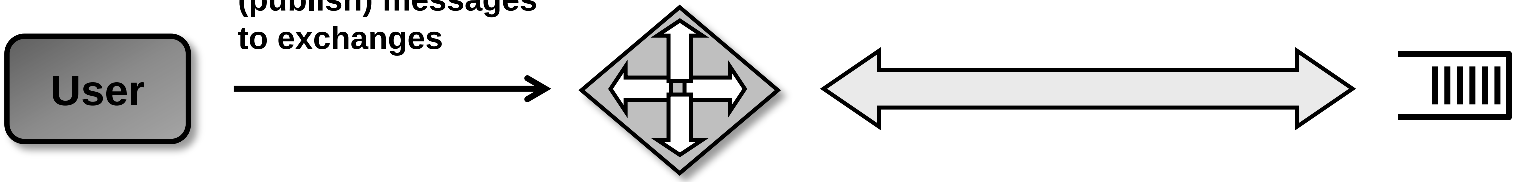
To achieve interoperability between the different queueing systems, 3rd party vendors offer bridges.
These bridges complicate the architecture / topology, increase costs and reduce performance (additional delay).

6. Examples of MOM middleware (6/7)

AMQP characteristics:

- Open protocol for business messaging, with support of industry (Cisco, Microsoft, Red Hat, Deutsche Bank, Microsoft etc.).
- Multi-platform / language messaging system.
- AMQP defines:
  a. Messaging capabilities (called AMQP model)
  b. Wire-level protocol for interoperability
- AMQP messaging patterns:
  a. Request-response (messages delivered to a specific queue)
  b. PubSub (messages delivered to a set of receiver queues)
  c. Round-robin (distribution of messages to a set of receiver based on availability)

Main components of AMQP model:



Applications send (publish) messages to exchanges

Message exchange: Exchanges receive messages and dispatch these to queues.

Binding: Defines the routing of messages to a message queue.

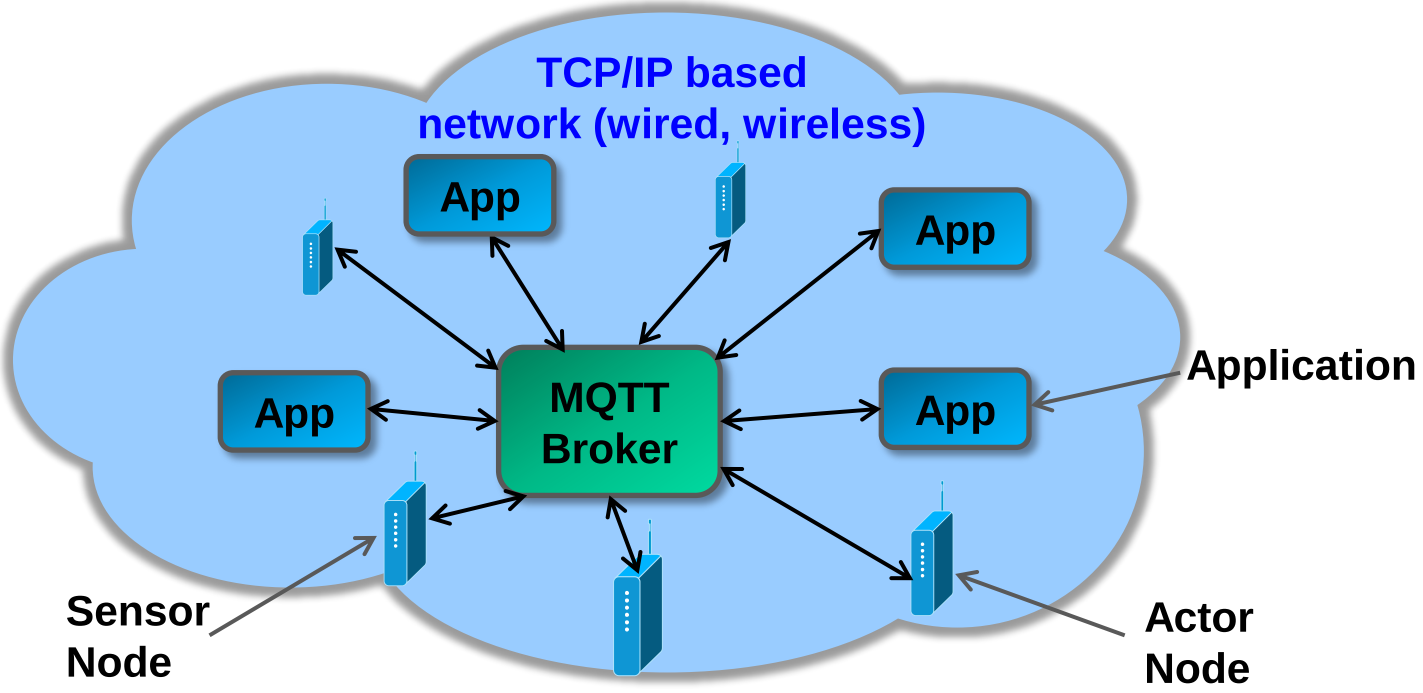
Message queue

6. Examples of MOM middleware (7/7)

MQTT – Message Queueing for Telemetry Transport:

Most MQ systems and protocols are aimed at backend and enterprise applications.
As such, these technologies are not suited for constrained devices like sensor nodes.
Such devices are typically contstrained in terms of memory, bandwidth and power.

MQTT is a message oriented protocol aimed at applications like wireless sensor networks, M2M (Mobile 2 Mobile) and ultimately Internet of Things (large number of nodes and applications loosely coupled through a messaging system).



7. Comparison of JMS, MSMQ and AMQP middleware (1/2)

Service	JMS	MSMQ	AMQP
Dead-letter queue	Yes	Yes	Yes
Journal queue	~ Persistent delivery mode	Yes (global and for each queue)	~ Persistent messages
Multicast / distribution list	Topics	Multicast, distribution lists, multiple receiver format names	* Topics (pubsub) * Fanout (send to all bound rx queues)
Message delivery QoS	* Persistent delivery mode * Different message ack types * Message priorities	Guaranteed message delivery: * MSMQ transactions * Recoverable message type * Different message ack types	Yes (exactly-once delivery semantics of a session): * Persistent / non-persistent delivery modes * Message priorities * Only message ack and no-ack modes defined
Message routing	More complicated routing schemes based on hierarchic topics and client message selection filters	Yes (requires AD / Windows domain)	Complex routing schemes possible based on routing key (contains destination matching criteria)
Security	JMS specification itself does not provide security (left to the JMS provider)	Yes (optional authentication and encryption)	Yes (SASL)
Peek queue (check if message available without receiving)	Yes (QueueBrowser object)	Yes	No
Priority based messaging	Yes (10 priority levels)	Yes (8 priority levels)	Yes (10 priority levels)

AD:	Active Directory
QoS:	Quality of Service (features for guaranteeing delivery)
SASL:	Simple Authentication and Security Layer (RFC4422)

7. Comparison of JMS, MSMQ and AMQP middleware (2/2)

Service	JMS	MSMQ	AMQP
Message transport protocol	JMS does not define a specific transport (wire) protocol (left to JMS providers)	TCP, HTTP, HTTPs, native OS, SPX	TCP & SCTP (UDP transport may be added in future versions of AMQP)
Message transactions	Yes	Yes	Yes
Receive triggers	No client triggering (left to JMS provider implementation)	Yes	Not part of the specification (AMQP provider may add this functionality)
Message formatting	None defined (JMS does not define a wire protocol)	XML, binary, ActiveX format	1 binary wire protocol defined in the specification
Acknowledgment types	3 acknowledgment types / modes defined	Yes (8 different ack types)	2 modes (message acknowledgment and no message acknowledgement)

SCTP: Stream Control Transmission Protocol (RFC4960), connection-oriented transport protocol with better characteristics than TCP
SPX: Sequenced Package Exchange (legacy Novell transport protocol)

8. Comparison of MOM with Internet messaging

Message queue systems share many properties and characteristics with Email systems.
For many of the message queue concepts there is a corresponding concept in Email systems.

MOM / Messaging	Email
MOM message	SMTP message
Message queue	Mailbox
Consumer	POP3 / IMAP4 client
Producer	SMTP client
Queue manager	MTA (Mail Transfer Agent)
Routing key	To: / Cc: address
Publish / subscribe	Mailing list
Message filter	E.g. server-side spam check
Message acknowledge	Read receipt (MS Outlook), email tracking (embedded links)
Transactional messaging	Not available
TTL	„Expires“ header field
Communication between applications or components	Communication between users