Java Message Service (JMS)

Message-Oriented Middleware (MOM)

- Sending/receiving messages between distributed software applications/components over heterogeneous platforms
  - Supporting asynchronous calls
- Creating a distributed communications layer
  - Insulating application developers from the details of various OSs and network interfaces
- APIs that extend across diverse platforms and networks are typically provided by MOM

Comparisons

- MOM
  - Asynchronous communication
  - Loosely-coupled components
- RPC-based/ORB-based middleware
  - Synchronous communication
  - Tightly-coupled components
- Performance?
- Application scenarios?

MOM Standards

- Historically, there was a lack of standards
- Advanced Message Queuing Protocol (AMQP)
- Data Distribution Service (DDS) by OMG
- eXtensible Messaging and Presence Protocol (XMPP)
- Java Message Service (JMS) by Java EE
  - Implemented by most MOM vendors and aims to hide the particular MOM API implementations

Java Message Service (JMS)

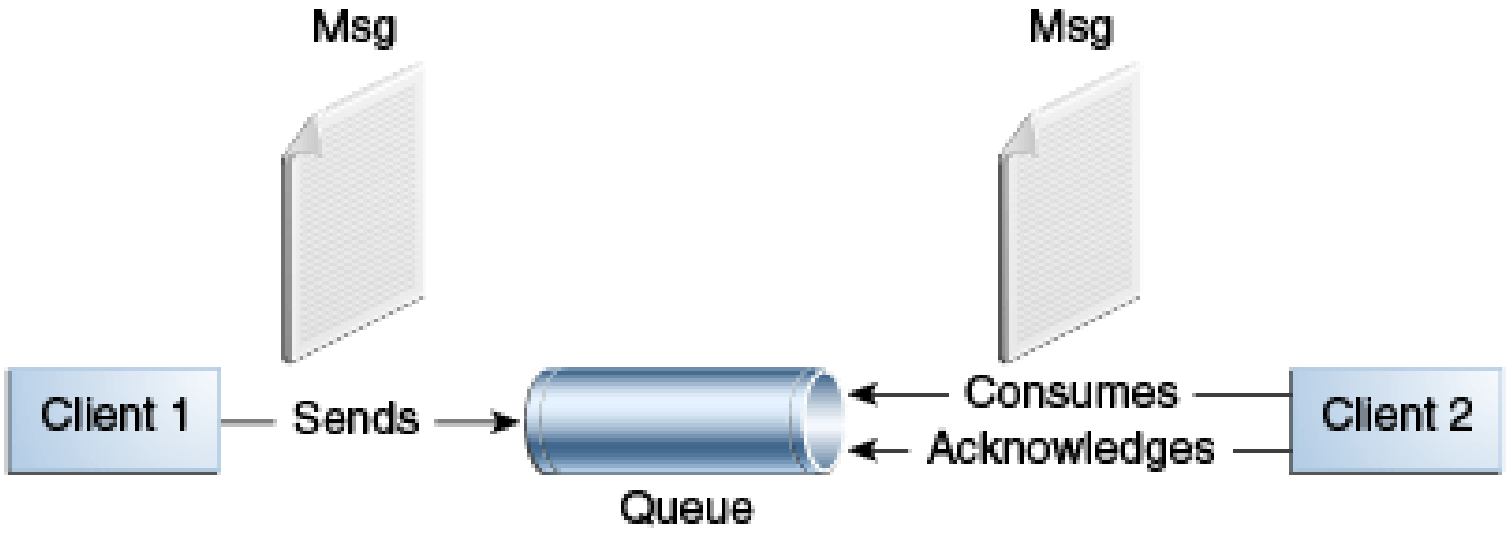
- A Java API that allows applications to create, send, receive and read messages
- Defining a common set of interfaces and associated semantics
- Striving to maximize the portability of JMS applications across JMS providers
- Enabling communication that is loosely coupled, asynchronous, and reliable

When Using JMS API

- The provider wants the components not to depend on information about other components' interfaces, so components can be easily replaced
- The provider wants the application to run whether or not all components are up and running simultaneously
- The application business model allows a component to send information to another and to continue to operate without receiving an immediate response

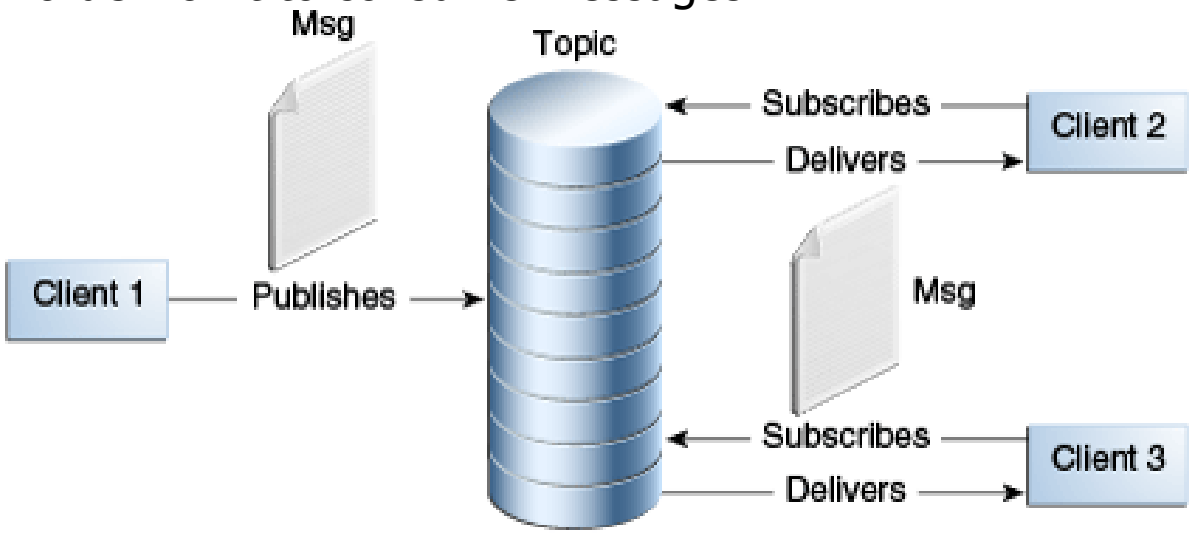
Messaging Styles

- Point-to-point messaging style
  - Each message is addressed to a specific queue
  - Queues retain all messages sent to them until the messages are consumed or expire
  - Each message has only one consumer



Messaging Styles (cont.)

- Publish/subscribe messaging style
  - Clients address messages to a topic
  - Publishers and subscribers can dynamically publish or subscribe to the topic
  - Topics retain messages only as long as it takes to distribute them to subscribers
  - Each message can have multiple consumers
  - A client that subscribes to a topic can consume only messages sent after the client has created a subscription, and the consumer must continue to be active in order for it to consume messages



Messaging Consumption

- Synchronously
  - A consumer explicitly fetches the message from the destination by calling the *receive* method
  - The *receive* method can block until a message arrives or can time out if a message does not arrive within a specified time limit
- Asynchronously
  - A client can register a *message listener* with a consumer
  - Whenever a message arrives at the destination, the JMS provider delivers the message by calling the listener's *onMessage* method

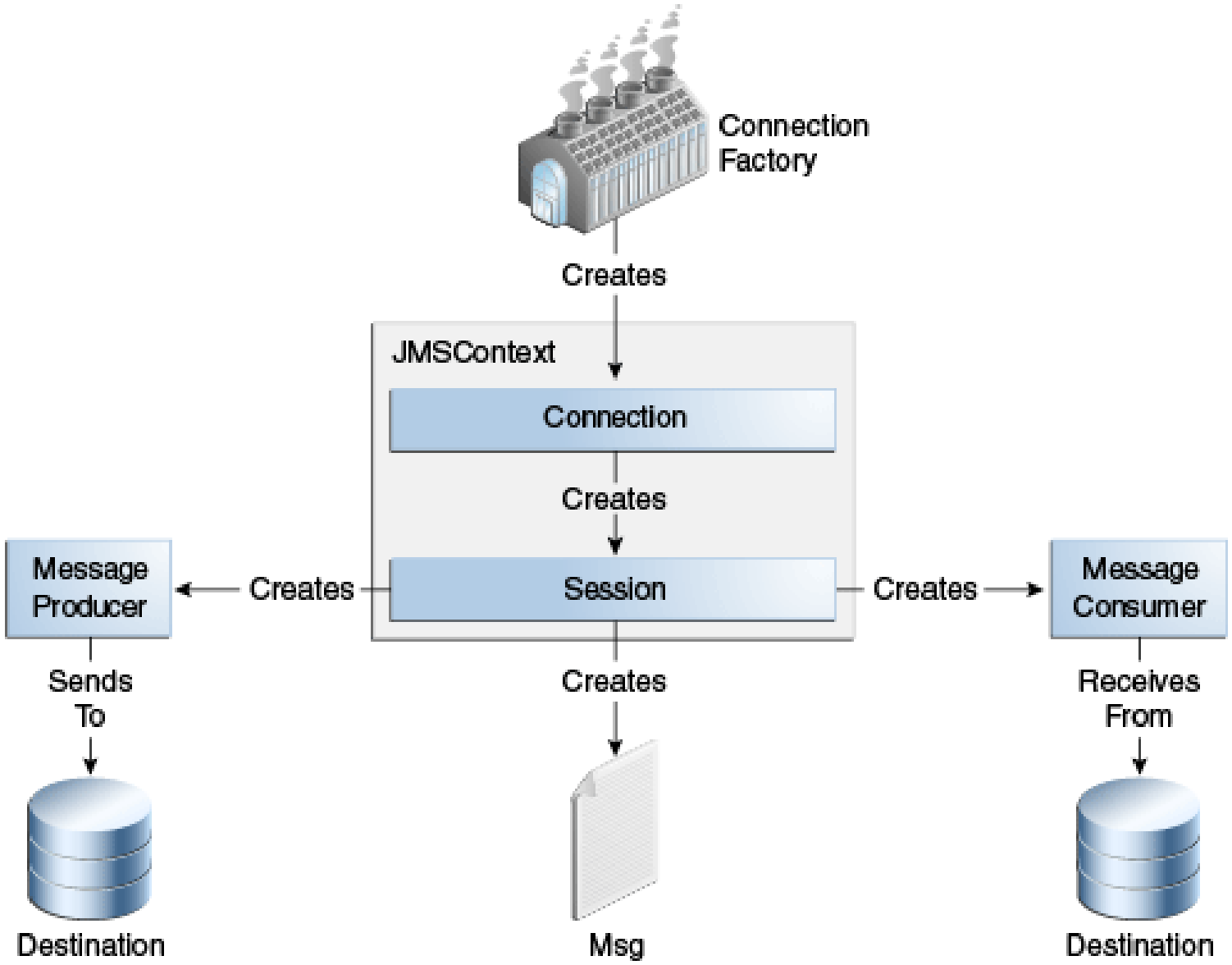
JMS Elements

- JMS Provider: an implementation of the JMS interface
- JMS Client: an application/process that produces and/or receives messages
- JMS Producer/Publisher: a JMS client that creates and sends messages
- JMS Consumer/Subscriber: a JMS client that receives messages
- JMS Message: an object that contains the data being transferred between JMS clients
- JMS Queue: a staging area that contains messages that have been sent and are waiting to be read (by only one consumer)
- JMS Topic: a distribution mechanism for publishing messages that are delivered to multiple subscribers

JMS App's Basic Building Blocks

- Administered objects: connection factories and destinations
- Connections
- Sessions
- Message producers
- Message consumers
- Messages

JMS API Programming Model



JMS Message Types

Message Type	Body Contents
TextMessage	A java.lang.String object (for example, the contents of an XML file).
MapMessage	A set of name-value pairs, with names as String objects and values as primitive types in the Java programming language. The entries can be accessed sequentially by enumerator or randomly by name. The order of the entries is undefined.
BytesMessage	A stream of uninterpreted bytes. This message type is for literally encoding a body to match an existing message format.
StreamMessage	A stream of primitive values in the Java programming language, filled and read sequentially.
ObjectMessage	A Serializable object in the Java programming language.

Lab

- Using JMS and ActiveMQ to implement distributed applications

Programming Practice

- Implement a distributed chatting application
  - Just one application (the chatting client)
  - All clients connected to the ActiveMQ instance
- Functionality
  - Login and specify the username (other existing sites receive notification)
  - Send group message
    - <TO> All
    - <MSG> morning, everyone
  - Send private message
    - <TO> Username
    - <MSG> shall we have lunch today
  - Send binary file to a specified user (using Queue)
  - Exit (other remaining sites receive notification)