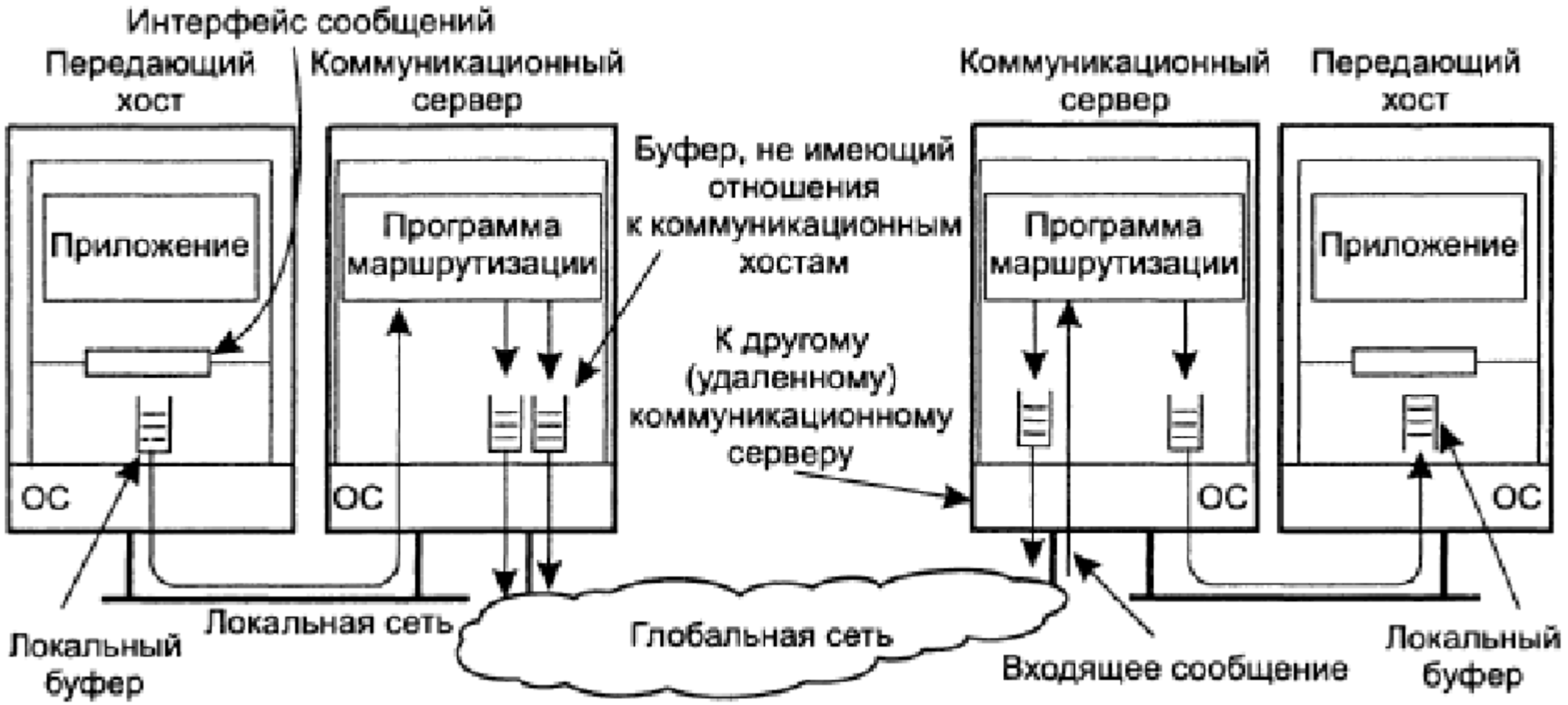
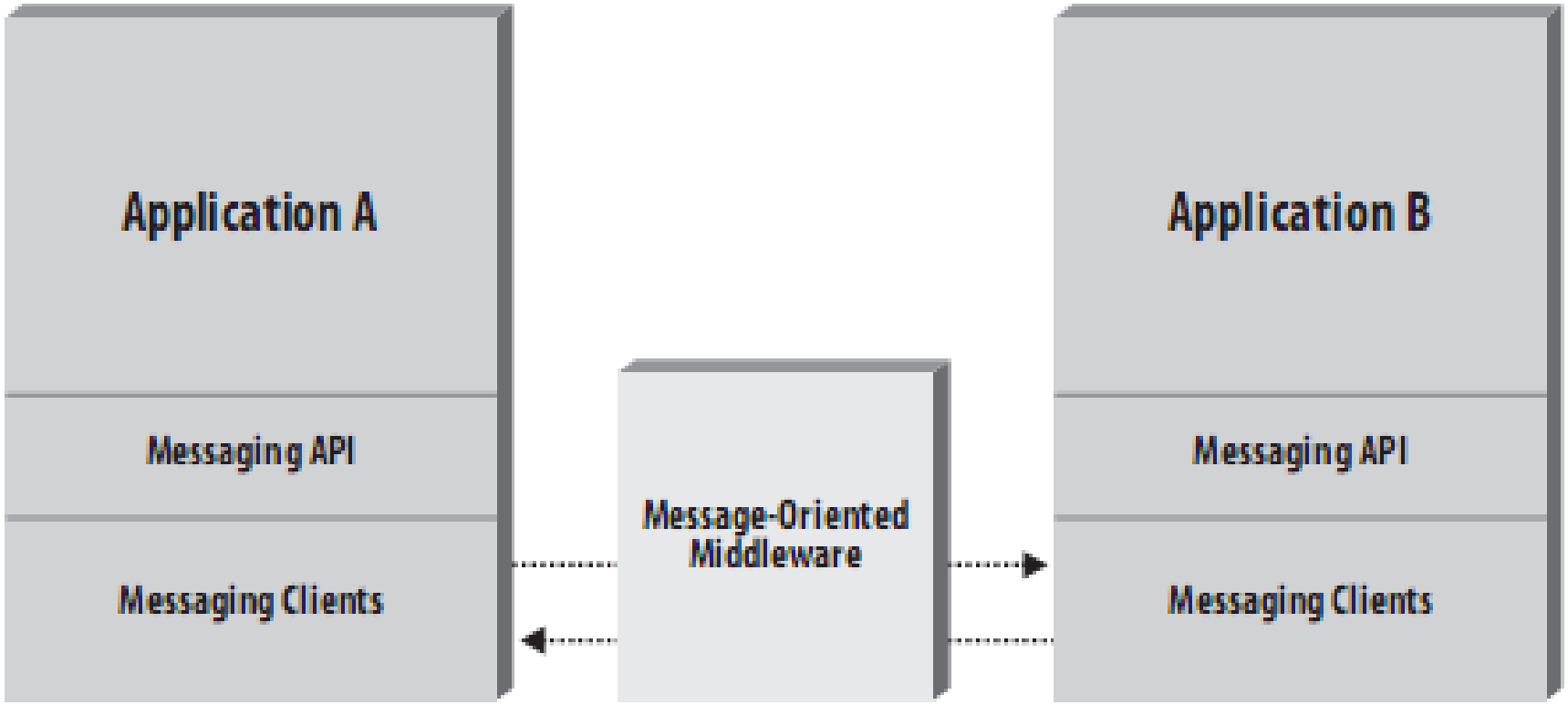
Системы обмена сообщениями

Промежуточное ПО, ориентированное на сообщения



Промежуточное ПО, ориентированное на сообщения

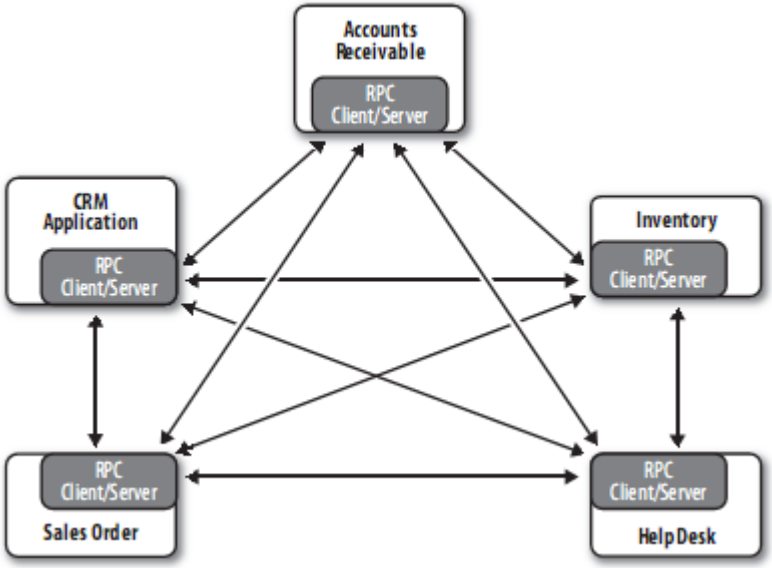
- IBM WebSphere MQ, SonicMQ, Microsoft Message Queuing (MSMQ), TIBCO Rendezvous, ...
- ActiveMQ, Glassfish MQ, RabbitMQ, ...



Основные достоинства МОМ

- Упрощение интеграции гетерогенных систем
- Уменьшение числа «узких мест» в системе
  - Несколько компонентов-обработчиков сообщений
- Улучшение масштабируемости
  - Горизонтальное масштабирование
- Увеличение эффективности работы пользователя
  - Асинхронный обмен сообщениями → больше действий с меньшим временем ожидания ответа
- Гибкость архитектуры

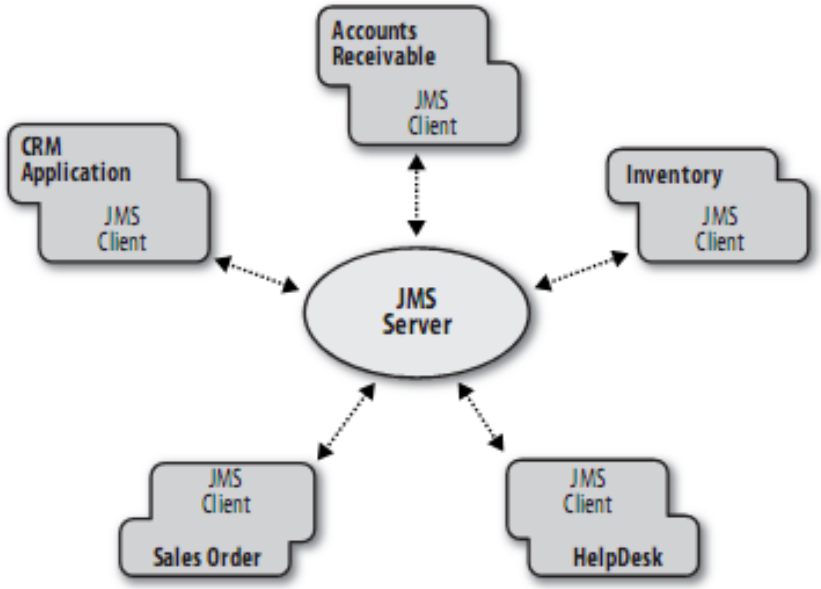
Основные достоинства MOM



RPC

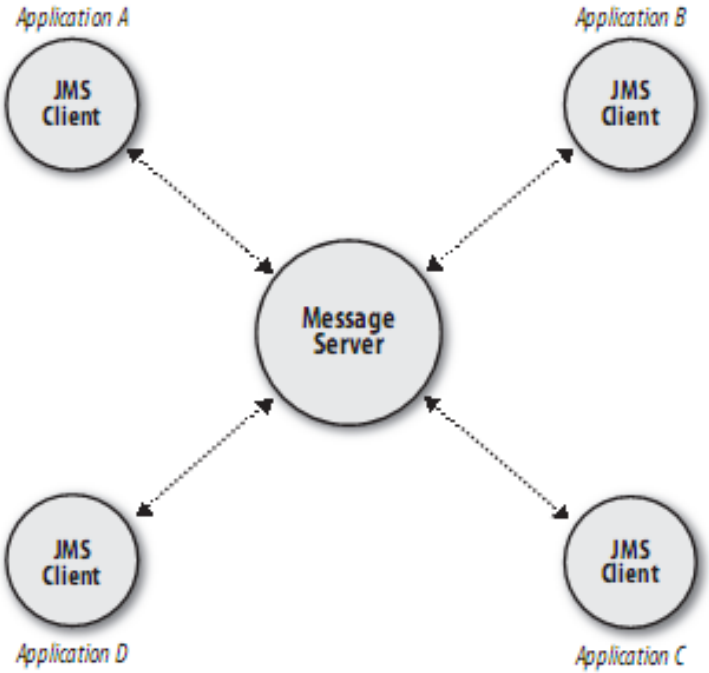
vs

MOM

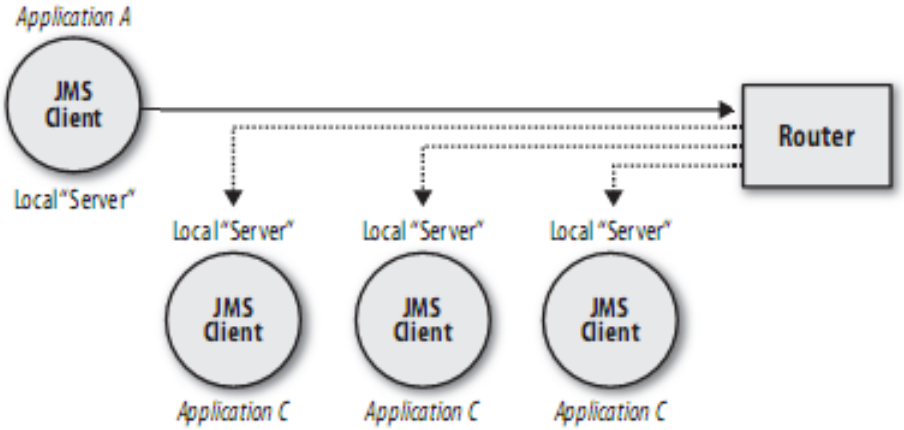


Архитектуры систем обмена сообщениями

- Централизованная



- Децентрализованная

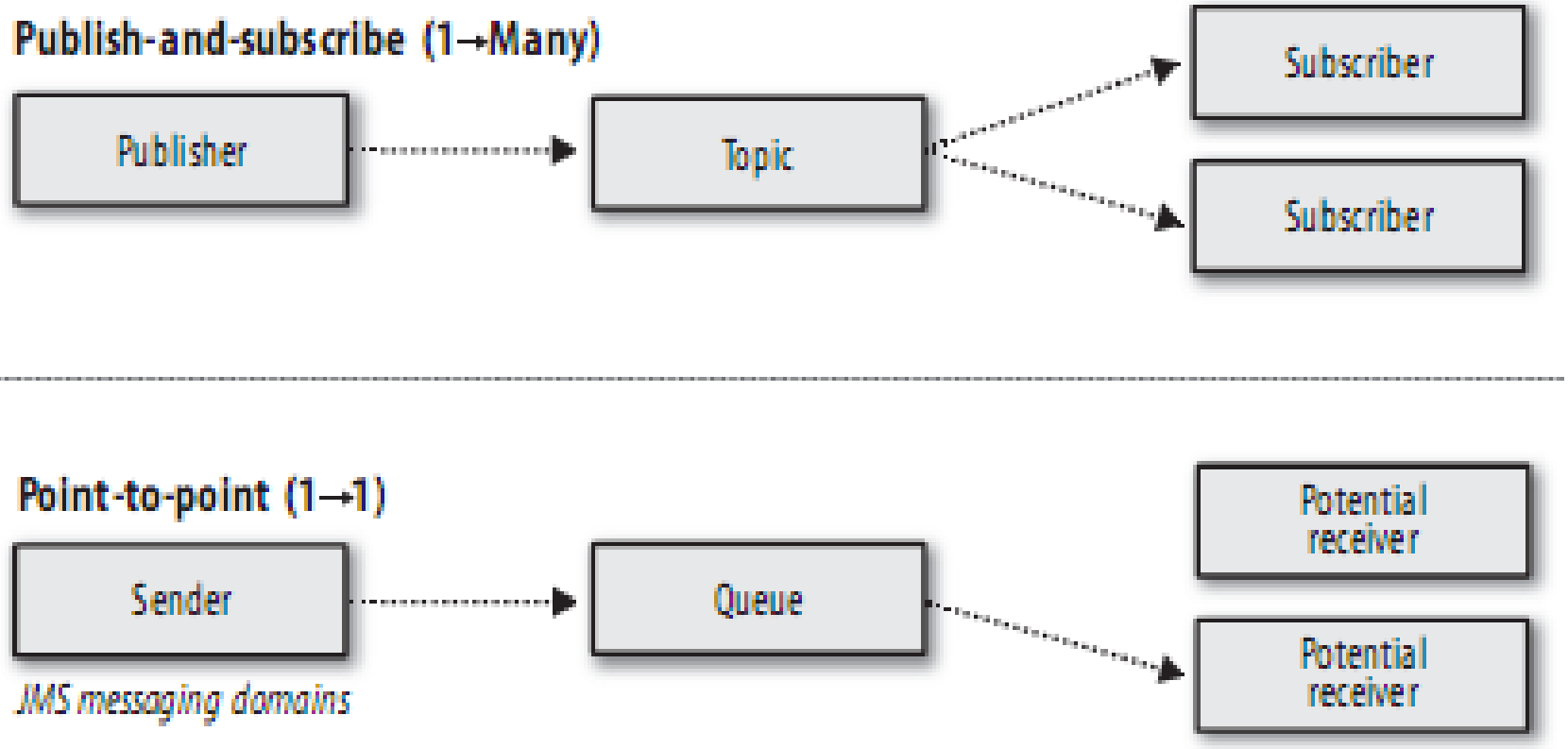


Java Message Service (JMS)

- Поставщики используют различные форматы сообщений и сетевые протоколы
- Базовая семантика отправки и приема сообщений совпадает во всех MOM
- JMS — общий API для работы с системами обмена сообщениями на платформе Java
  - Последняя версия - JMS 1.1, вышла в 2002 году
  - Аналогия с JDBC и JNDI

Модели взаимодействия в JMS

- «Точка-точка»
- «Издатель-подписчик»



JMS messaging domains

Модель взаимодействия «точка-точка»

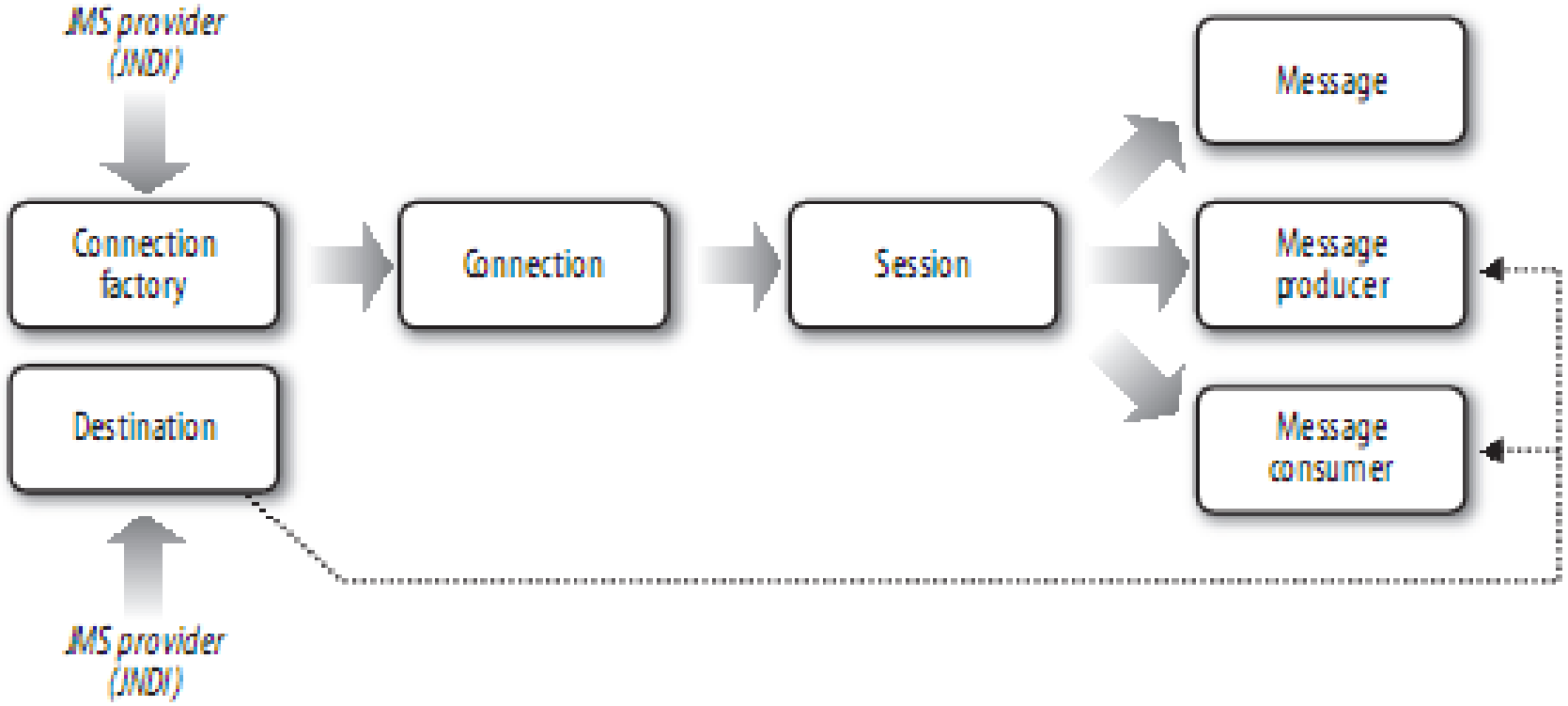
- **Очередь (Queue)**
- Синхронная и асинхронная отправка и прием
- Режим опроса
- Сообщение получает единственный обработчик
  - даже если с очередью связано несколько обработчиков
- Более тесное связывание
  - отправитель знает, как будет использовано сообщение и кто его получит
- Балансировка нагрузки

Модель взаимодействия «издатель-подписчик»

- **Тема (Topic)**
- Подписчик получает копии всех сообщений
- Новые сообщения «проталкиваются» (push)
- Более гибкое связывание
  - неизвестно, сколько подписчиков получают его сообщения и что они их обрабатывают
- Типы подписок:
  - Временные (nondurable)
  - Длительные (durable)

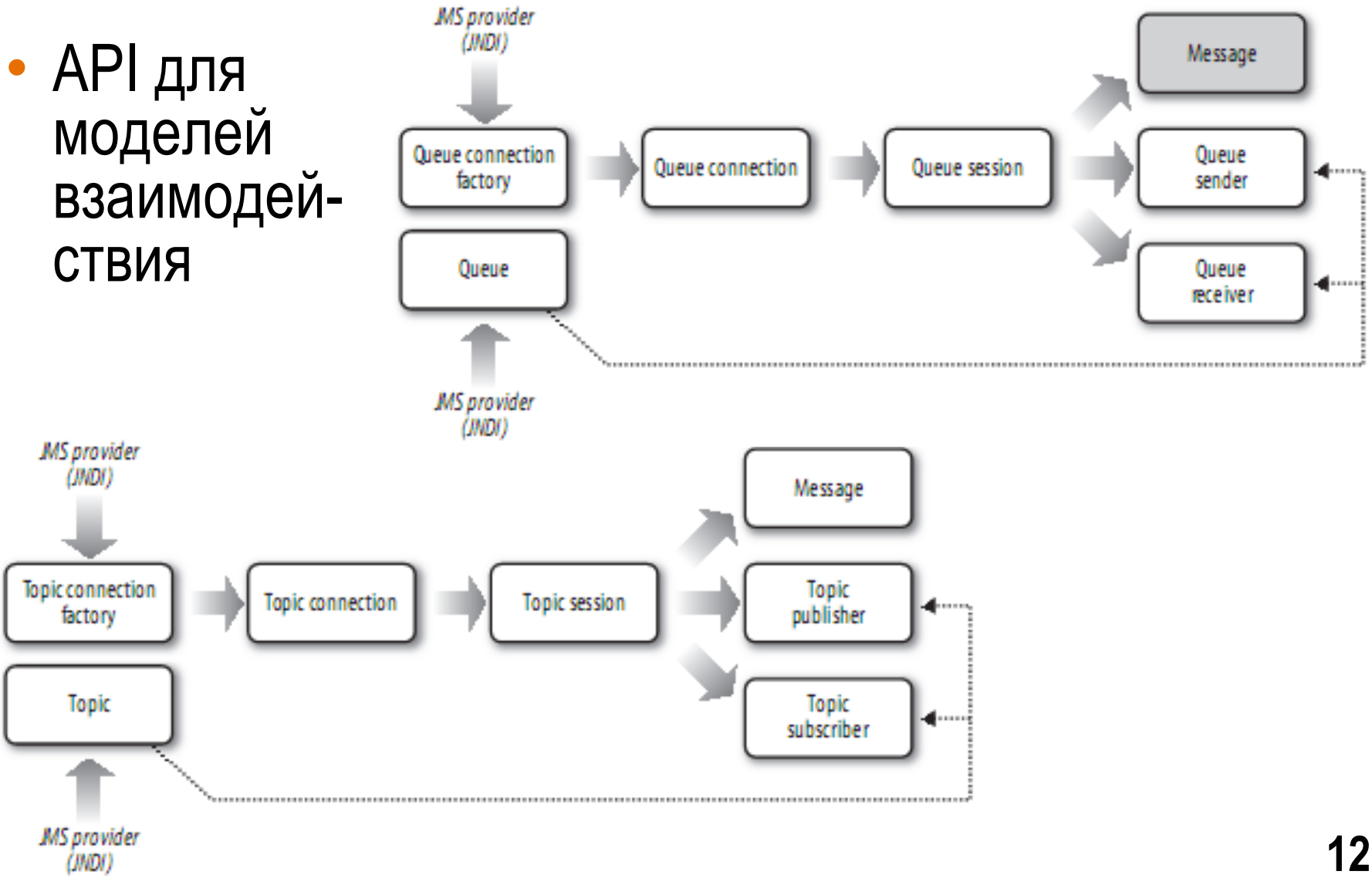
Структура JMS API

- Базовый API работает со всеми видами адресатов



Структура JMS API

- API для моделей взаимодей-ствия



Публикация сообщения в теме

- 1) Получить фабрику соединений из JNDI
  - Поиск в InitialContext либо @Resource объекта типа TopicConnectionFactory
- 2) Создать соединение
  - TopicConnection connection = conFactory.createTopicConnection();
- 3) Создать сеанс
  - TopicSession pubSession = connection.createTopicSession(false, Session.AUTO_ACKNOWLEDGE);
- 4) Получить тему из JNDI
  - Объект типа Topic

Публикация сообщения в теме

- 5) Создать издателя, связанного с темой
  - TopicPublisher publisher = pubSession.createPublisher(chatTopic);
- 6) Создать сообщение
  - TextMessage message = pubSession.createTextMessage();
  - message.setText("Hello world!");
- 7) Опубликовать сообщение
  - publisher.publish(message);

Подписка на тему

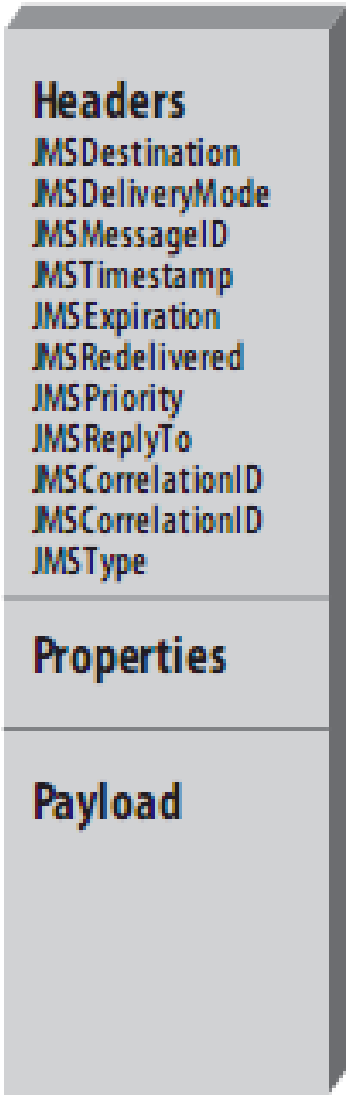
- 1) Получить фабрику соединений из JNDI
- 2) Создать соединение
- 3) Создать сеанс
- 4) Получить тему из JNDI
- 5) Создать подписчика
  - TopicSubscriber subscriber = subSession.createSubscriber(chatTopic, null, true);
    - Можно указать селектор сообщений и флаг noLocal
- 6) Установить обработчик событий
  - subscriber.setMessageListener(this);
    - Параметр — объект, реализующий интерфейс javax.jms.MessageListener

Обработка событий в подписчике

```
public void onMessage(Message message) {
        try {
                TextMessage textMessage = (TextMessage) message;
                System.out.println(textMessage.getText());
        } catch (JMSException jmse)
        { jmse.printStackTrace(); }
}
```

Сообщения

- Заголовки
- Свойства
- Полезная нагрузка



Структура сообщения — Заголовки

- Устанавливаемые автоматически JMS-провайдером при доставке сообщения
  - JMSDestination — адресат сообщения
  - JMSMessageID — уникальный идентификатор
  - ...
- Устанавливаемые программно
  - JMSReplyTo — указывает адресата, через которого можно ответить на полученное сообщение
  - JMSCorrelationID — связывает текущее сообщение с каким-либо предыдущим

Структура сообщения - Свойства

- Дополнительные заголовки
- Позволяют «прозрачно» добавить дополнительные сведения к сообщению
  - могут использоваться для фильтрации сообщений получателем
- Допустимые типы свойств:
  - String, boolean, byte, double, int, long, float
- Три категории:
  - 1) свойства конкретных приложений
  - 2) свойства расширений JMS
  - 3) свойства для конкретных JMS-провайдеров.

Структура сообщения — Полезная нагрузка

- Message — базовый для всех остальных типов сообщений, не несет полезной нагрузки
- TextMessage — строка (java.lang.String)
- ObjectMessage — сериализуемый Java-объект
- BytesMessage — байтовый массив, для обмена данными в «родном» формате приложения
- StreamMessage — поток значений встроенных типов данных Java (int, double, char и т. д.)
  - позволяет преобразовать отформатированный поток байтов в значения переменных.
- MapMessage — множество пар «имя-значение»

Message-Driven Beans

- Компоненты, управляемые сообщениями
- особый вид EJB-компонентов, предназначенный для обработки асинхронных сообщений от JMS
- не поддерживают состояние между запросами
- нет локального или удаленного представления
- нет бизнес-методов

Жизненный цикл MDB