AARONGROUP

www.aag.cz

Komunikace systémů pomocí zasílání zpráv

Petr Steckovič

10. 11. 2011 Hradec Králové

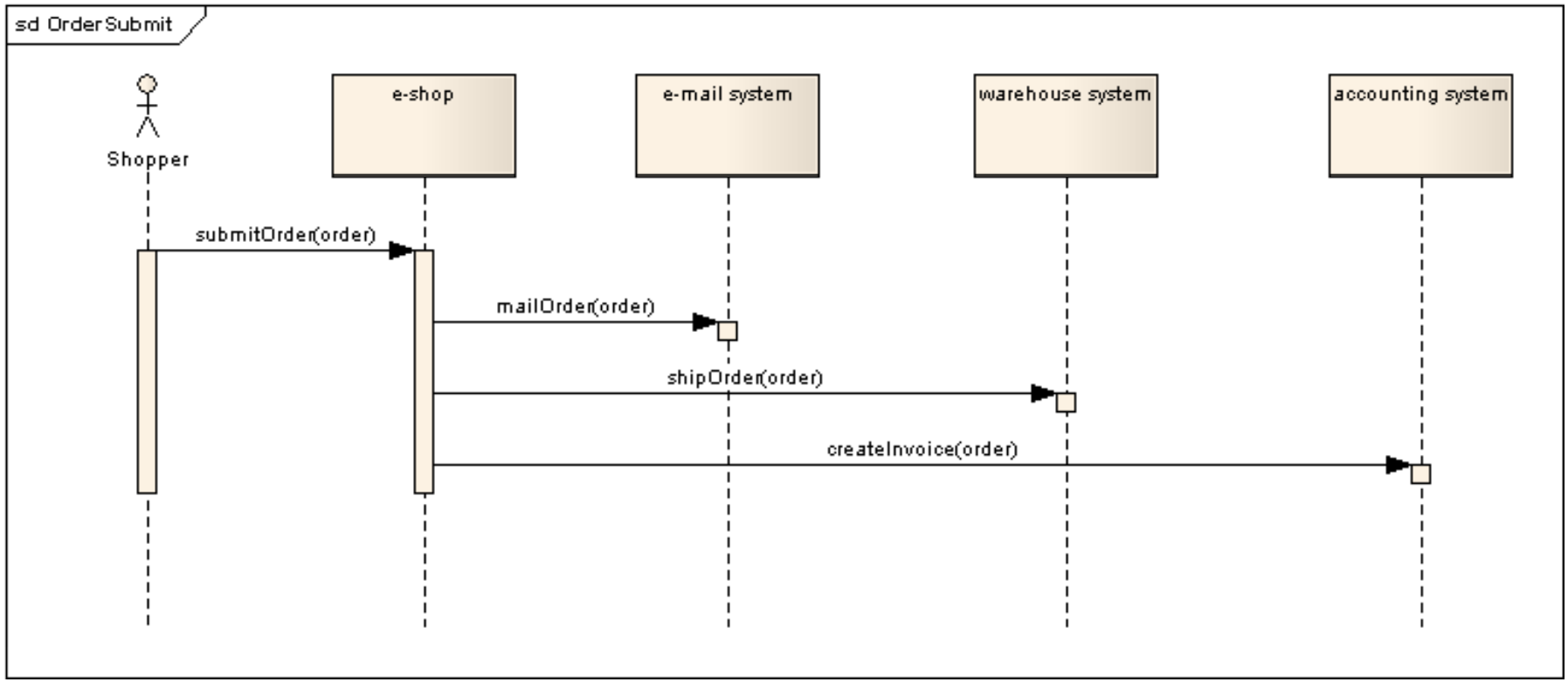
Komunikace

- Synchronní / Asynchronní
- Jednosměrná / Obousměrná
- Přímá / Zprostředkovaná
- Nejobvyklejší typ komunikace: Synchronní Obousměrná, Přímá

Úryvek ze specifikace

… po té co je objednávka založena uživatelem, je poslán e-mail s náhledem objednávky a zároveň je notifikován skladový systém, který zajistí vyskladnění zboží. Založení objednávky je dále napojeno na účetní systém přes, který bude možné vytisknout fakturu …

Založení objednávky



Rizika synchronní komunikace

- Přímá závislost volajícího na volaném
  - Problémy (částečných) výpadků, upgradů
- Příliš těsná závislost systémů
  - Nutnost znát kolaborující systémy
- Rozdílná rychlost spolupracujících systémů

Asynchronní komunikace

- Nečeká se na kompletní zpracování zprávy ☺
- Odesílatel zprávy nemusí znát (všechny) příjemce ☺ - Motivace pro použití zprostředkované komunikace
- Request / Response ☹ (odesílatel zprávy se obtížně dozví jak zpracování dopadlo)

Zprostředkovaná komunikace

Message broker

- Prostředník oddělující vysílajícího od příjemce (příjemců)
- Manager queue / topic a dalších objektů
- Queue – Producer / Consumer
- Topic – Publisher / Subscribe
- Odolnost proti výpadku
- Garantované doručení

Instalace brokeru

AARONGROUP

www.aag.cz

- Dedikovaný
- Součást JEE aplikačního serveru
- Součást aplikace
- Clustering / Load balancing / Failover
- Persistentní úložiště
  - Databáze
  - Filesystém
  - Speciální

Example - Sender

```
InitialContext ic = Utils.createInitialContext();
ConnectionFactory cf = null;
Connection connection =  null;
try {
    cf = (ConnectionFactory)ic.lookup("/ConnectionFactory");
    Queue queue = (Queue)ic.lookup(QUEUE_JNDI);

    connection = cf.createConnection();
    Session session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);
    MessageProducer producer = session.createProducer(queue);

    connection.start();

    TextMessage message = session.createTextMessage("Hello!");
    producer.send(message);
    System.out.println("Message sent!");
} finally {
    if(ic != null) {
        try {
            ic.close();
        } catch(Exception e) {
            throw e;
        }
    }

    if (connection != null) {
        connection.close();
    }
```

Example - Receiver


```
cf = (ConnectionFactory)ic.lookup("/ConnectionFactory");
Queue queue = (Queue)ic.lookup(QUEUE_JNDI);

connection = cf.createConnection();
Session session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);
MessageConsumer consumer = session.createConsumer(queue);

connection.start();

Message message = null;
System.out.println("Waiting for message ...");
message = consumer.receive(5000);
message = consumer.receiveNoWait();
message = consumer.receive();

if (message != null) {
    System.out.println("Message received '" + ((TextMessage)message).getText() + "'");
} else {
    System.out.println("No message received");
}
```

Example – Ostatní

JNDI

```
public static InitialContext createInitialContext() throws NamingException {
    Properties p = new Properties();
    p.put(Context.INITIAL_CONTEXT_FACTORY, "org.jnp.interfaces.NamingContextFactory");
    p.put(Context.URL_PKG_PREFIXES, " org.jboss.naming:org.jnp.interfaces");
    p.put(Context.PROVIDER_URL, "jnp://localhost:1099");
    return new InitialContext(p);
}
```

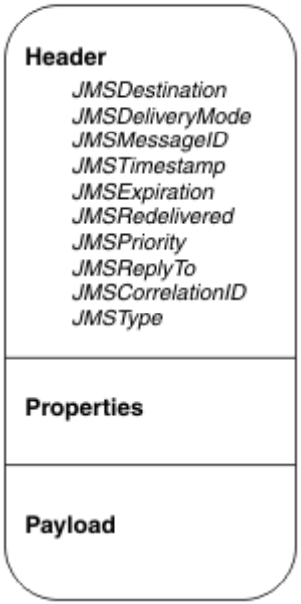
Konfigurace HornetQ

```
52
53      <queue name="BootcampQueue">
54          <entry name="/queue/BootcampQueue"/>
55      </queue>
56
57  </configuration>
58
```

AARONGROUP

Anatomie zprávy

- Header
- Properties
  - Standardní (JMSX prefix, JMSXDeliveryCount)
  - Application Specific
  - Provider (Broker) specific
- Typ těla zprávy – BytesMessage, TextMessage, StreamMessage, MapMessage, ObjectMessage



Selektory zpráv

AARONGROUP

www.aag.cz

- Logický jazyk selektorů – boolean
- Použití selektorů
  - V aplikaci
  - V brokeru (směrování / duplikaci …)
- Příklad selektoru:

COUNTRY = ‘CZ’ AND CITY_ID=‘HK’

Návrh zprávy

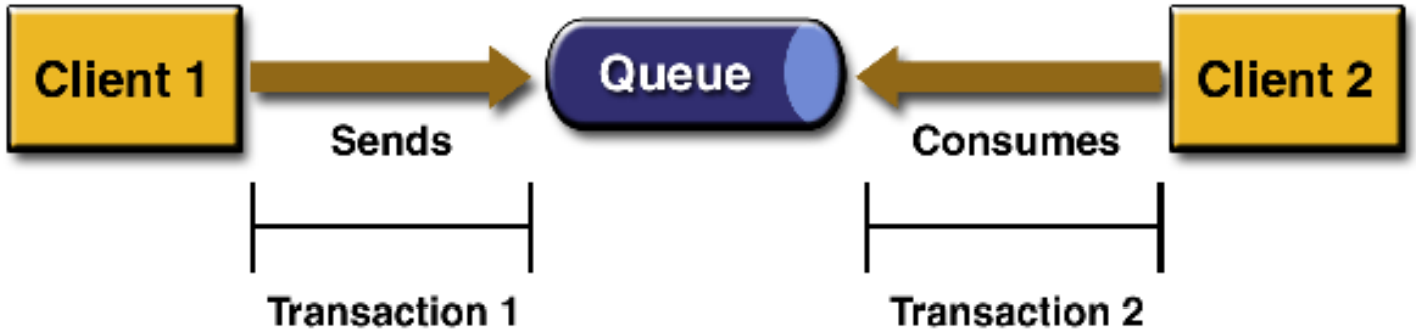
AARONGROUP

www.aag.cz

- Zpráva je obvykle rozhraní mezi systémy
  - Pozor na ObjectMessage !!!
- Vhodné aplikační property:
  - Typ zprávy (ORDER_CREATED, ORDER_CANCELLED)
  - Verze dat
  - Některá business data jako property (ORDER_ID, COUNTRY, CITY_ID)

Transakce / doručení

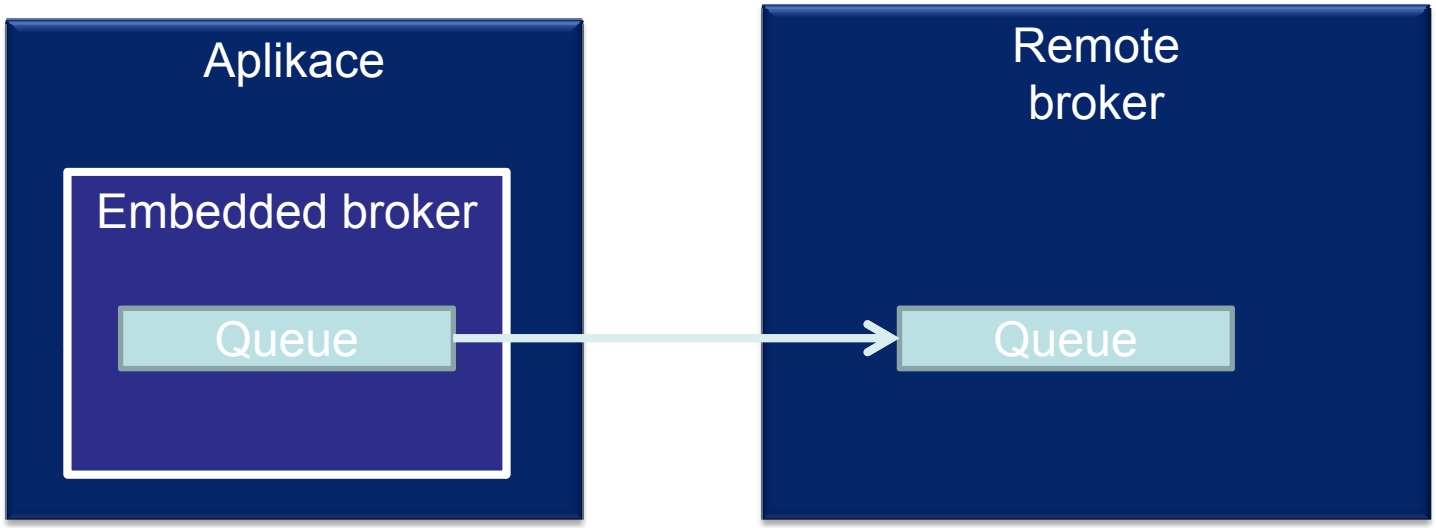
- Transakce



- Distribuované transakce
- Doručení / Redelivery
- Dead Letter Queue (DLQ popř. DMQ)
- Expiry Queue

Bridge

- Přeposílání zpráv mezi brokery
- Lokální komunikace poslání zprávy
- Odolnost proti výpadku remote brokeru



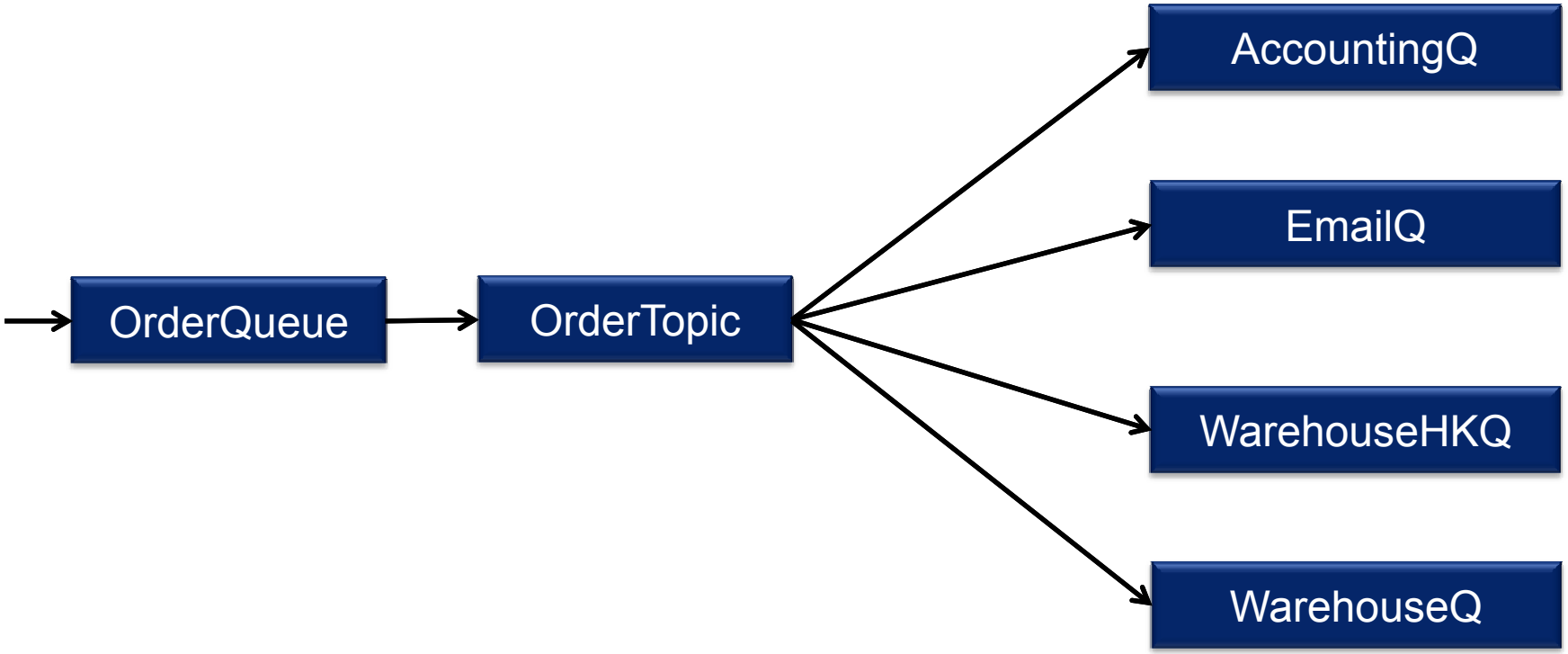
Divert

- Forwading - exclusive divert

- Duplikace – non-exclusive divert
  - možnost využít selektor

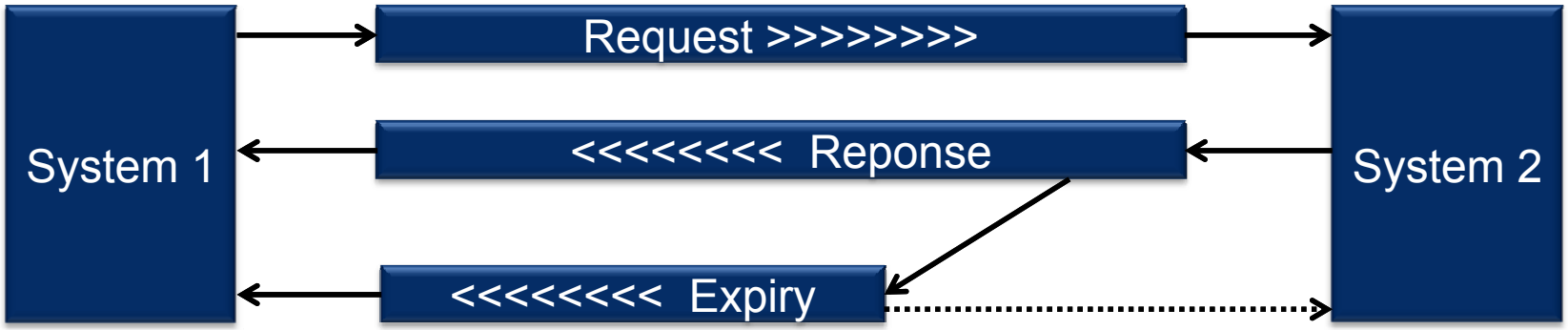
Rozposílání zpráv

Řešení specifikace prostředky brokeru



Request / Response

- Emulace synchronního volání
- Proč tak složitě?
- Obousměrná komunikace s exspirací



Implementace

- Pomocí dynamického selektoru
  - Robustní ☺
  - Bezeztrátové ☺
  - Pomalejší ☹
- Pomocí temporary queue (example)
  - Rychlé ☺
  - Konfigurace s expirací nelze ☹
  - Uzavření temporary Q ztratí její obsah ☹
  - V době zpracování requestu nemusí již žadatel existovat ☹

Req / Res - Klient


```
try {

    cf = (ConnectionFactory)ic.lookup("/ConnectionFactory");
    Queue queue = (Queue)ic.lookup(QUEUE_JNDI);

    connection = cf.createConnection();
    Session session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);
    MessageProducer producer = session.createProducer(queue);

    TemporaryQueue replyQueue = session.createTemporaryQueue(); //!!!IMPORTANT

    connection.start();

    TextMessage message = session.createTextMessage("Hello!");
    message.setJMSReplyTo(replyQueue);//!!!IMPORTANT
    producer.send(message);
    System.out.println("Message sent!");
    System.out.println("Receiving ...");

    MessageConsumer consumer = session.createConsumer(replyQueue);
    TextMessage reply =  (TextMessage)consumer.receive();
    System.out.println("Response received: " + reply.getText());

} finally {
```

Req / Res - Server


```
        cf = (ConnectionFactory)ic.lookup("/ConnectionFactory");
        Queue queue = (Queue)ic.lookup(QUEUE_JNDI);

        connection = cf.createConnection();
        Session session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);
        MessageConsumer consumer = session.createConsumer(queue);

        connection.start();

        Message request = null;
        System.out.println("Waiting for message ...");
        request = consumer.receive();

        if (request != null) {
            System.out.println("Message received '" + ((TextMessage)request).getText() + "'");
        } else {
            System.out.println("No message received");
        }

        Destination responseDestination = request.getJMSReplyTo();

        MessageProducer responseProducer = session.createProducer(responseDestination);
        Message response = session.createTextMessage("Response for message.");
        response.setJMSCorrelationID(request.getJMSCorrelationID());//!!! IMPORTANT
        responseProducer.send(response);
        System.out.println("Response sent.");
    } finally {
```

Podpora v JEE 6

- Pouze JEE Full Profile
- ConnectionFactory, Queue .. jsou zdroje AS - @Resource
- Plná podpora distribuovaných transakcí
- Podpora MDB
- MDB automaticky napojená na thread pool
- Propustnost aplikace
- Stavební kámen paralelizmu
- Zjednodušený deployment

Producer servlet JEE 6

AARONGROUP

www.aag.cz

```
@WebServlet(name="TestMessageSenderServlet", urlPatterns = {"/"})
public class TestMessageSenderServlet extends HttpServlet{
    private static final long serialVersionUID = 1L;


    @Resource(lookup="java:/ConnectionFactory")
    private ConnectionFactory connectionFactory;

    @Resource(lookup = "java:/queue/test")
    private Queue testQueue;

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException {
        String text = req.getParameter("text");
        if (text == null) {
            text = "No text";
        }

        Connection conn = null;

        try {
            conn = connectionFactory.createConnection();
            Session session = conn.createSession(false, Session.AUTO_ACKNOWLEDGE);
            MessageProducer producer = session.createProducer(testQueue);
            conn.start();

            Message message = session.createTextMessage(text);
            producer.send(message);

            res.getWriter().append("Message has been sent '" + text + "'");
        } catch (JMSException e) {
```

Consumer MDB JEE 6

```
@MessageDriven (
          name = "LoggerMDB",
          activationConfig = {
              @ActivationConfigProperty(propertyName = "destinationType", propertyValue = "javax.jms.Queue"),
              @ActivationConfigProperty(propertyName = "destination", propertyValue = "/queue/test")
          }
        )
public class EJBMessageConsumer implements MessageListener{

    @Override
    public void onMessage(Message message) {
        try {
            System.out.println("Start Consumming " + Thread.currentThread().getName());
            System.out.println("Message text: " + ((TextMessage)message).getText());
            Thread.sleep(2000);
            System.out.println("Stop consumming " + Thread.currentThread().getName());
        } catch (Exception e) {
            throw new RuntimeException("Unexpected exception",e);
        }
    }

}
```

Q&A