# Distribuované systémy a výpočty

X36DSV

**Jan Janeček**
**Peter Macejko**

# CORBA

## Common Object Request Broker Architecture

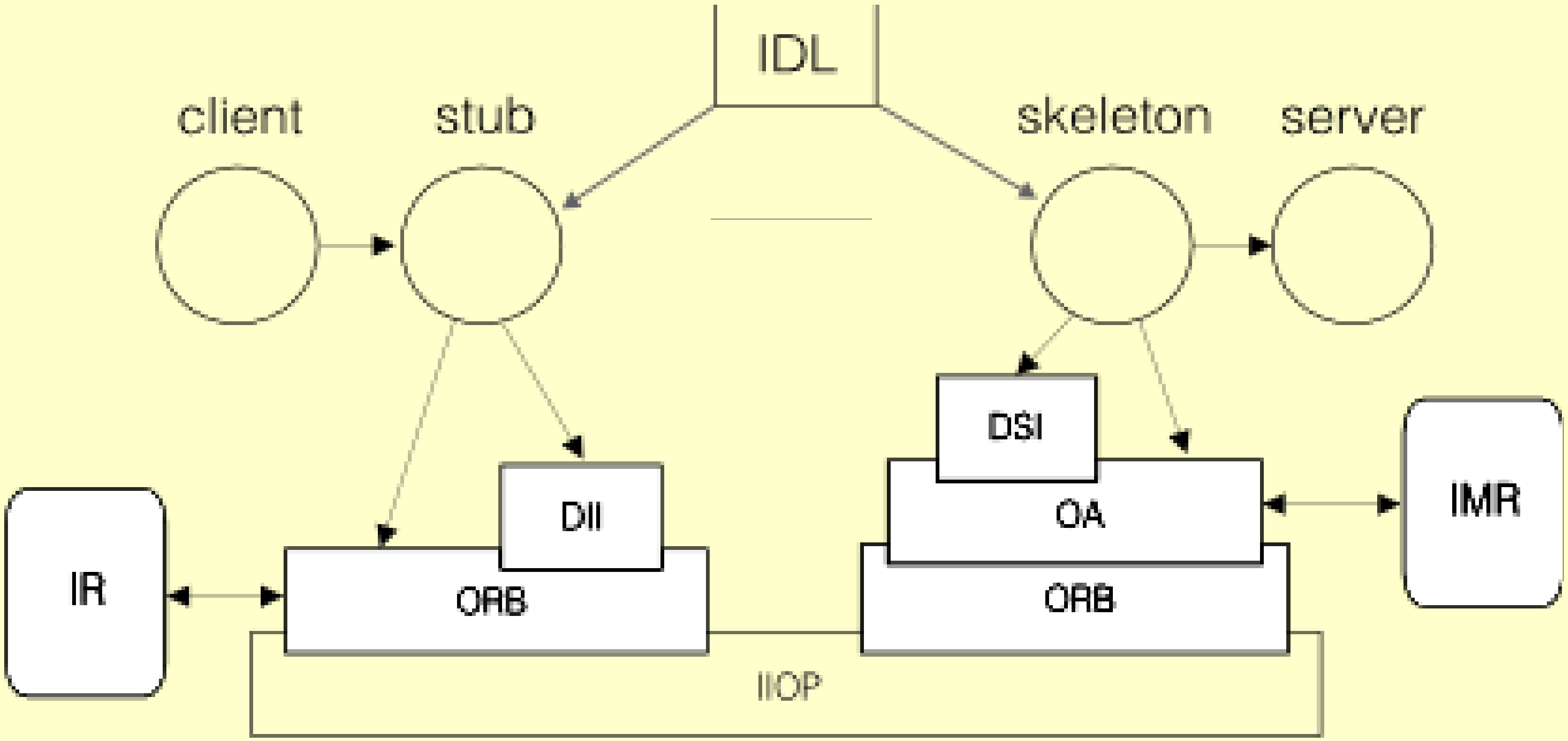
- konsorcium OMG (Object Management Group)
- standard pro podporu komunikace v DS
- pouze specifikace

## CORBA

### Common Object Request Broker Architecture

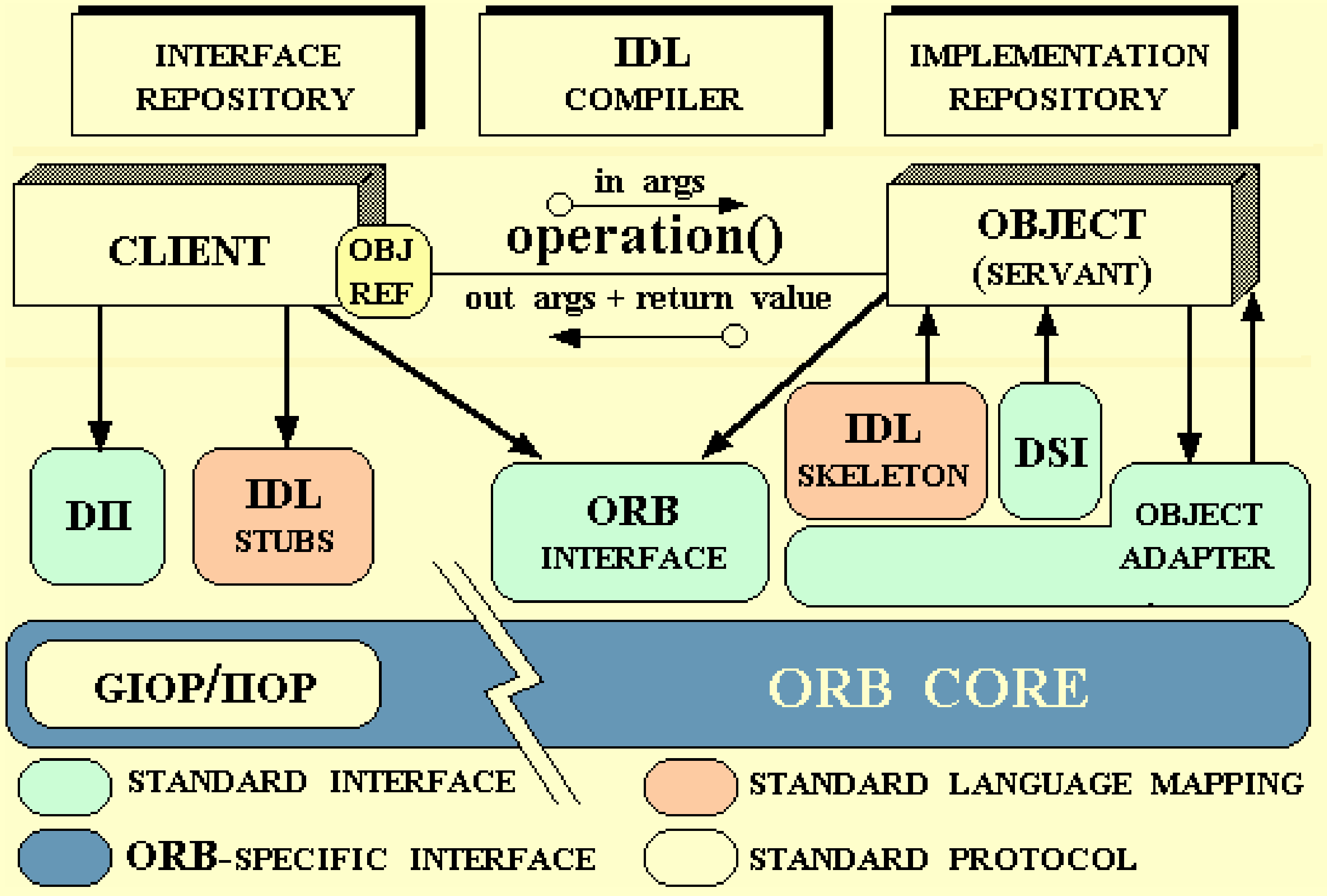
- procedurální komunikace
- programová sběrnice
- objektový přístup
- podpora složitějších forem chování serveru

# CORBA - architektura

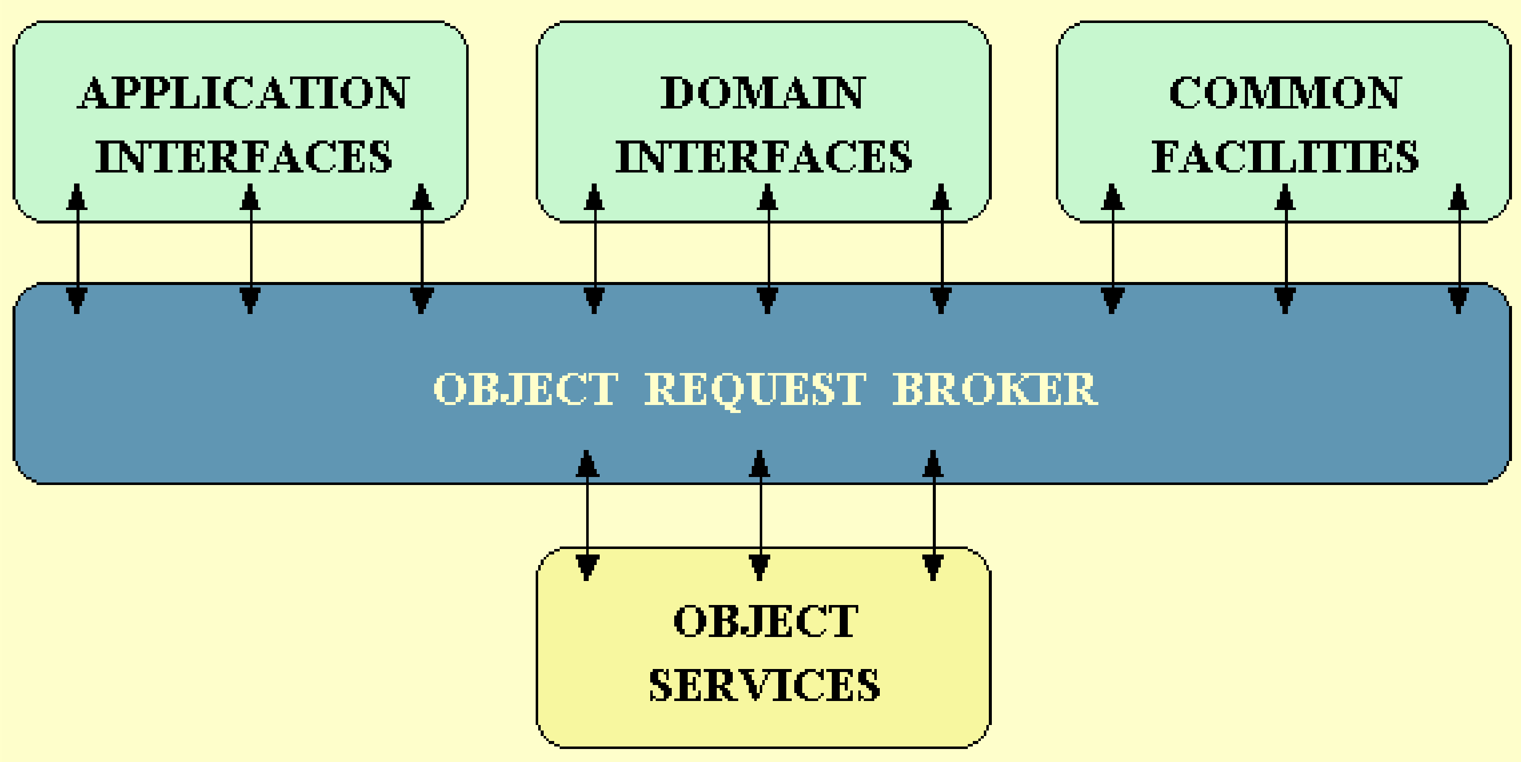


ORB – Object Request Broker
DII/DSI – Dynamic Invocation Interface / Dynamic Skeleton Interface
IR/IMR – Interface Repository / Implementation Repository
OA – Object Adapters
IIOP – Internet Inter-ORB Protocol

# CORBA - architektura

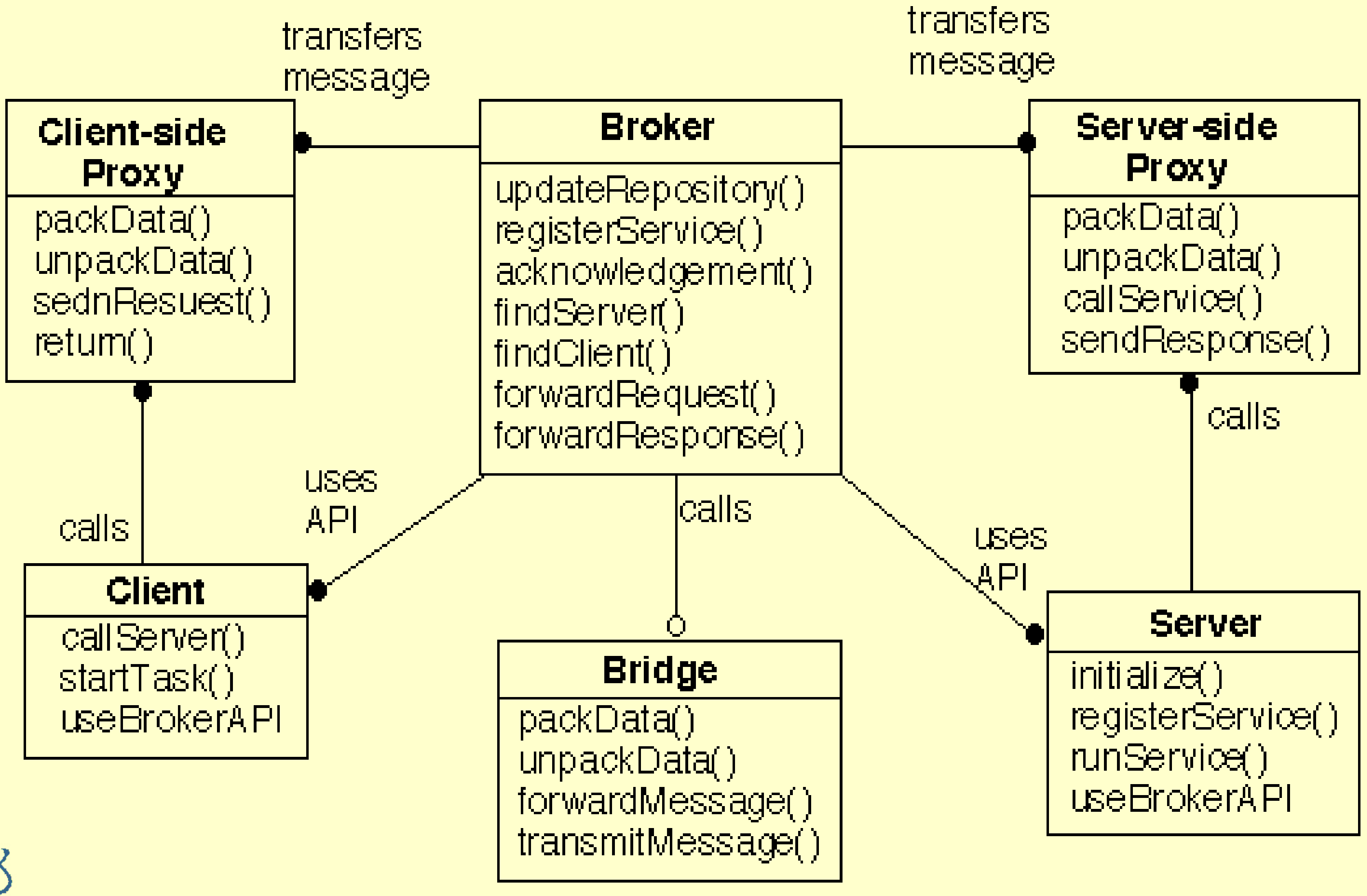


# CORBA - architektura



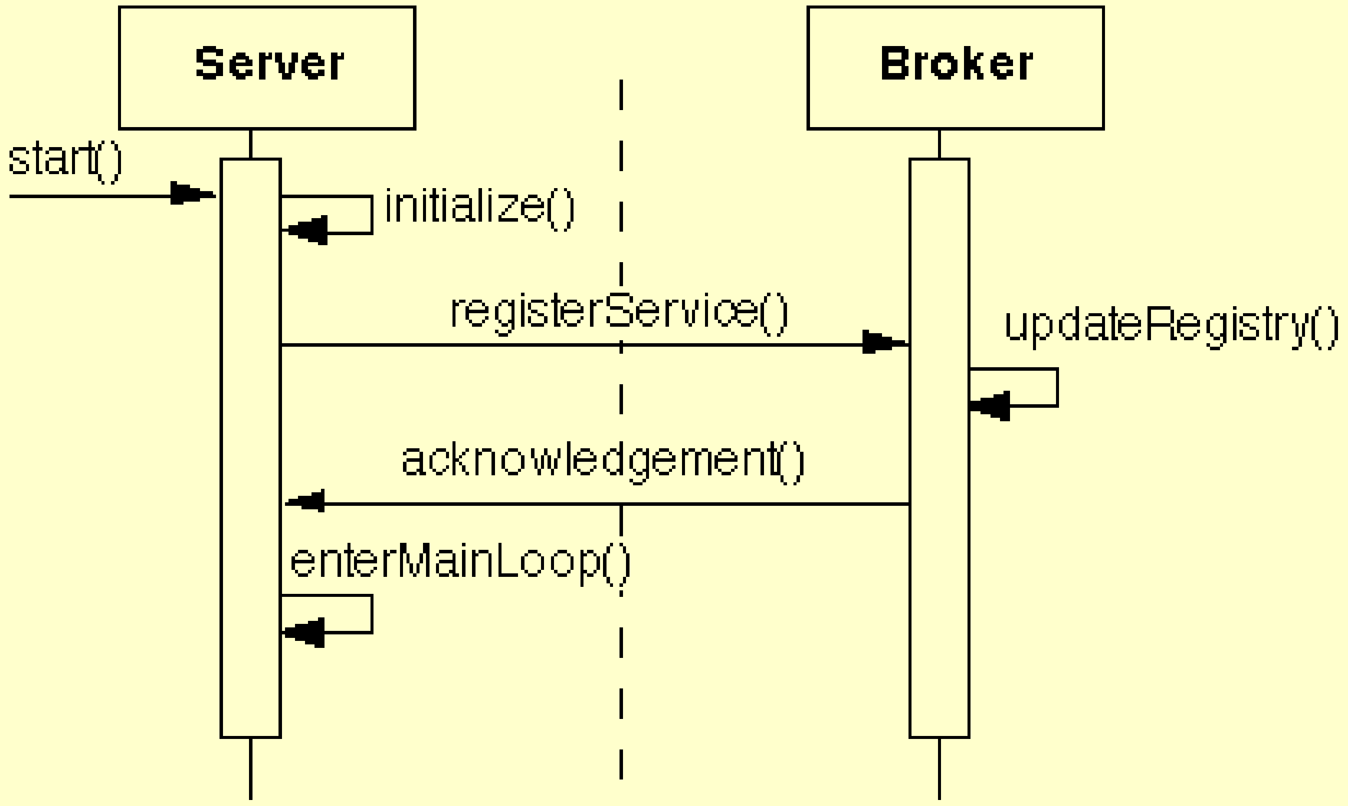
# CORBA

## broker – klient - server



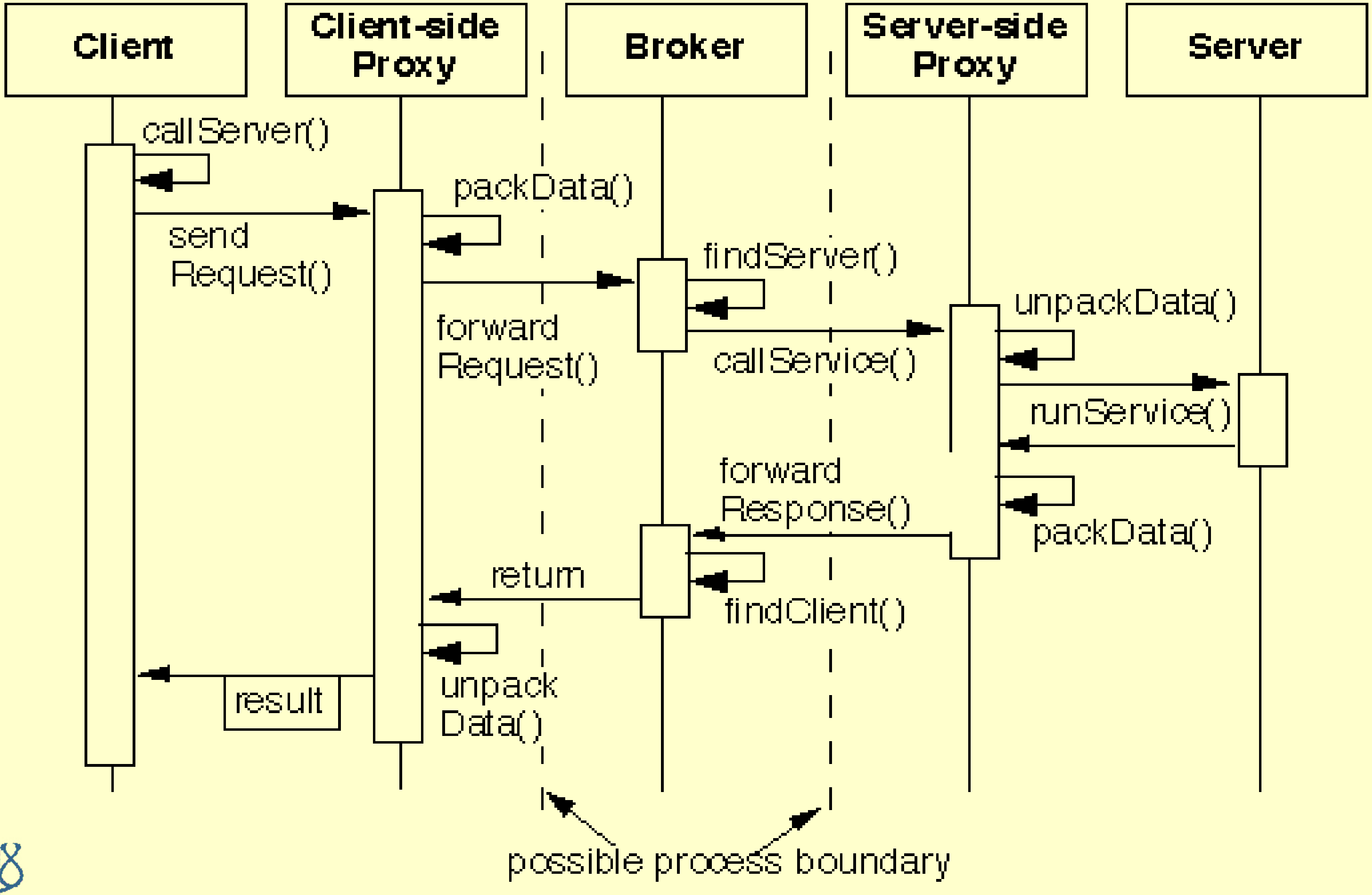
# CORBA

## registrace serveru



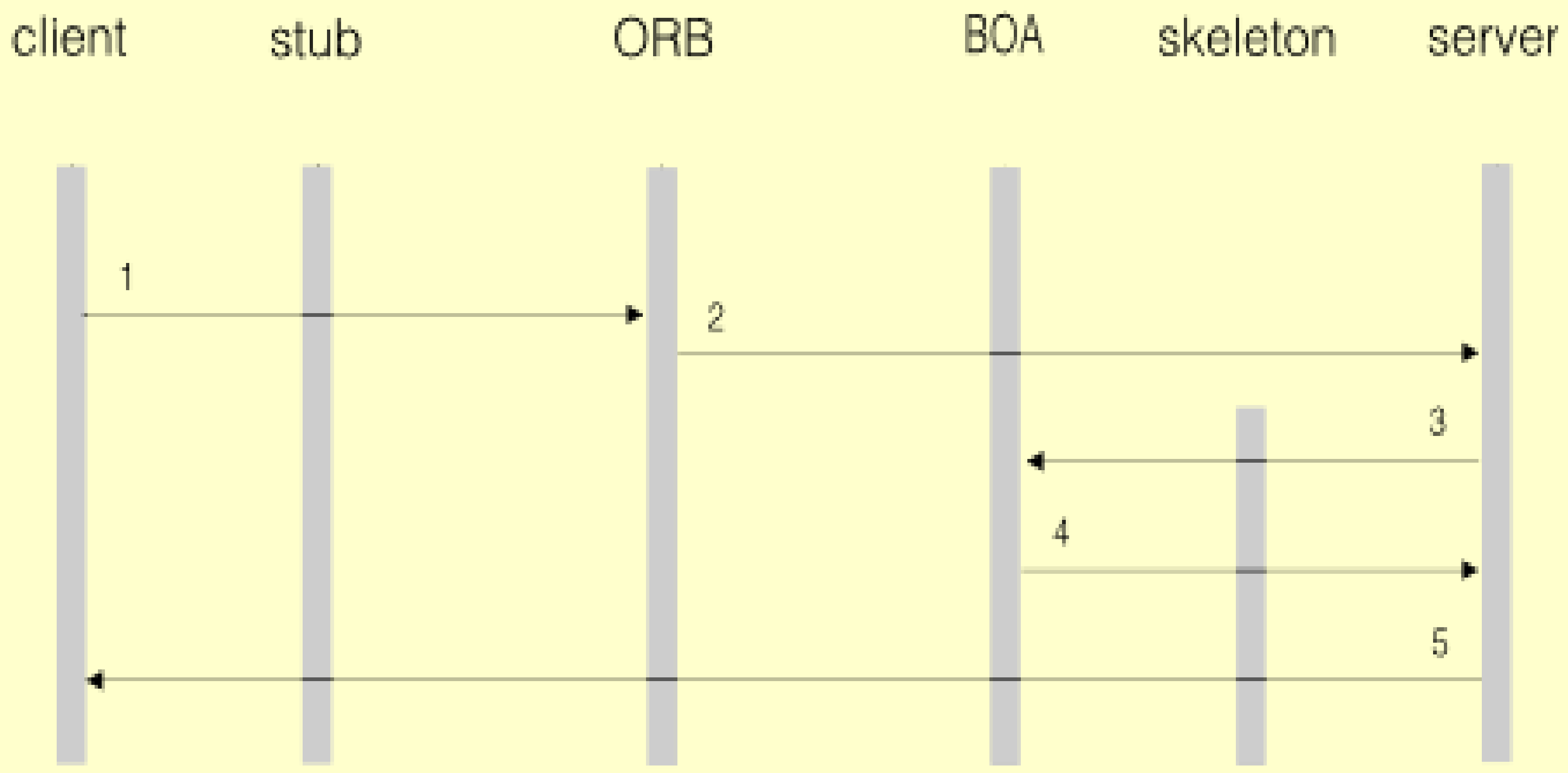
# CORBA

## komunikace klient - server



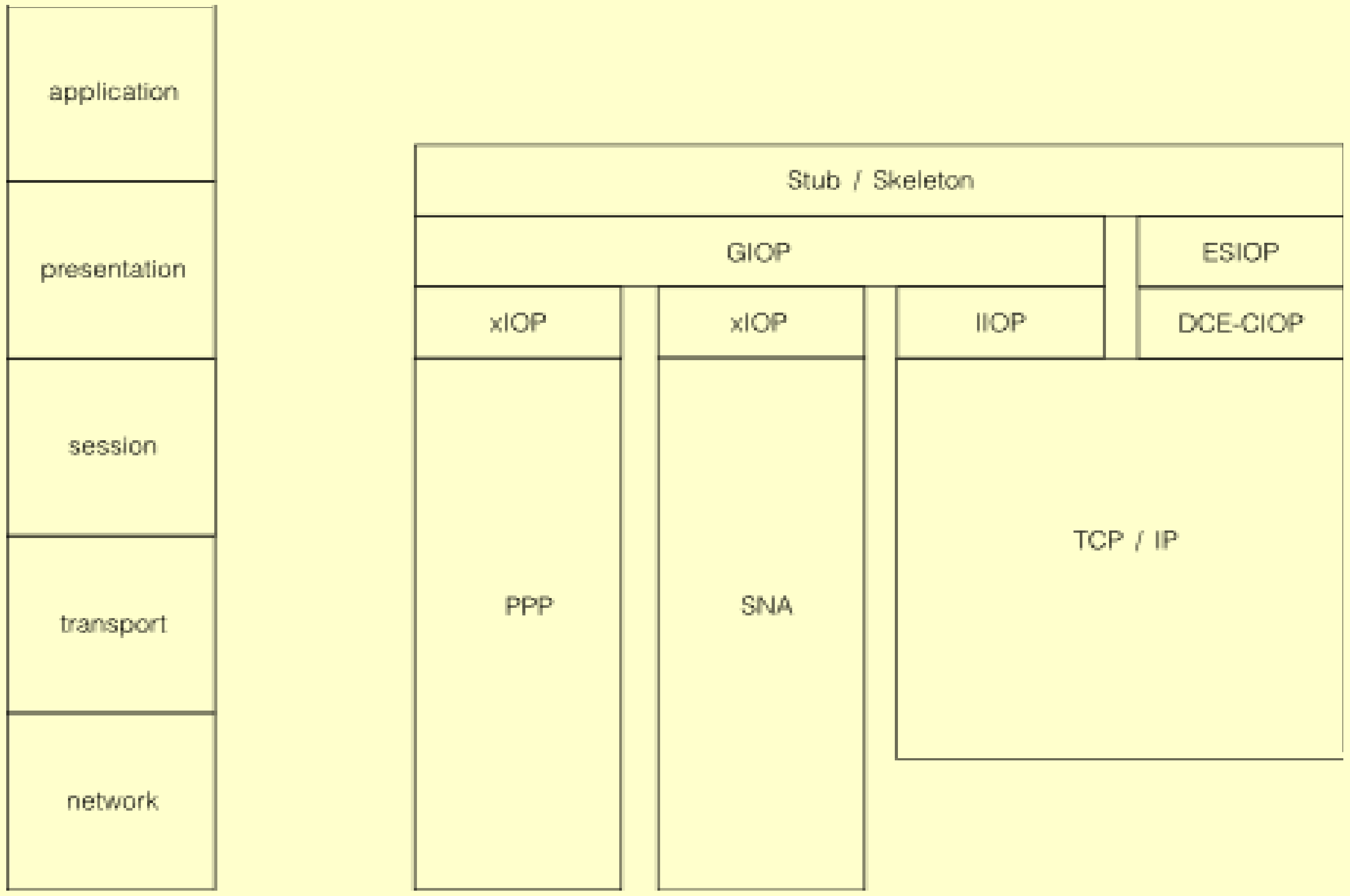
# CORBA

## komunikace



# CORBA

## komunikace



# CORBA - IDL

```
module StockObjects {

  struct Quote {
    string symbol;
    long at_time;
    double price;
    long volume;
  };

  exception Unknown{};

  interface Stock {
    Quote get_quote() raises(Unknown);
    void set_quote(in Quote stock_quote);
    readonly attribute string description;
  };

  interface StockFactory {
    Stock create_stock(
      in string symbol,
      in string description
    );
  };
};
```

# CORBA - příklad

Pro IONA Orbix – průmyslový CORBA standard

## Definice rozhraní

```
interface Hello
  {
        string sayHello();
  };
```

## a jeho překlad

```
idl Hello.idl;
```

# CORBA - příklad

Hello.java
- rozhraní klienta

_HelloStub.java
- stub klienta

_HelloSkeleton.java
- stub serveru

HelloPackage/
- definice typů definovaných rozhraním

_HelloImplBase.java
- základ pro implementaci serveru

. . .

# CORBA - příklad

Hello.java

- přeložené rozhraní klienta

```
public interface Hello
    extends org.omg.CORBA.Object
{
    public String sayHello() ;
    public java.lang.Object _deref() ;
}
```

# CORBA - příklad

_HelloImplBase.java

- základní objekt pro implementaci serveru

```
import IE.Iona.OrbixWeb._OrbixWeb;

public abstract class _HelloImplBase
    extends _HelloSkeleton
    implements Hello {
    public _HelloImplBase() {
        org.omg.CORBA.ORB.init().connect(this);
    }
    . . .
    public java.lang.Object _deref() {
        return this;
    }
}
```

# CORBA - příklad

## Kód metod serveru

```
public class HelloImplementation extends _HelloImplBase
  {
  public String sayHello()
    {
    return "Hello World";
    }
  }
```

a konečně samotný server . . .

# CORBA - příklad

```
import IE.Iona.OrbixWeb._CORBA;
import IE.Iona.OrbixWeb.CORBA.ORB;
public class HelloServer
  {
  public static void main (String args[])   {
    org.omg.CORBA.ORB ord = org.omg.CORBA.ORB.init();
      try  {
        Hello server = new HelloImplementation();
        _CORBA.Orbix.impl_is_ready( "HelloServer" );
        System.out.println("Server going Down");
      }
        catch ( org.omg.CORBA.SystemException corbaError)  {
        System.out.println("Exception " + corbaError);
      }
    }
  }
```

# CORBA - příklad

## Překlad kódů serveru

```
Hello.java
_HelloSkeleton.java
_HelloImplBase.java
HelloImplementation.java
HelloServer.java
```

## Start name servisu

```
orbixdj -textConsole
```

## registrace serveru

```
putit HelloServer -java HelloServer
```

## spuštění serveru

```
java HelloServer
```

# CORBA - příklad

## Kód klienta

```
import IE.Iona.OrbixWeb._CORBA;
import org.omg.CORBA.ORB;
public class HelloClient  {
   public static void main(String args[])    {
      ORB.init();
      String hostname = "eli.sdsu.edu";
      String serverLabel = ":HelloServer";
      Hello server = HelloHelper.bind( serverLabel, hostname);
      System.out.println( server.sayHello() );
   }
}
```

# CORBA - příklad

## Překlad souborů klienta

```
_HelloStub.java
HelloClient.java
```

## a jeho spuštění

```
java HelloClient
```

## JMS

### Java Messaging Service

Předávání zpráv

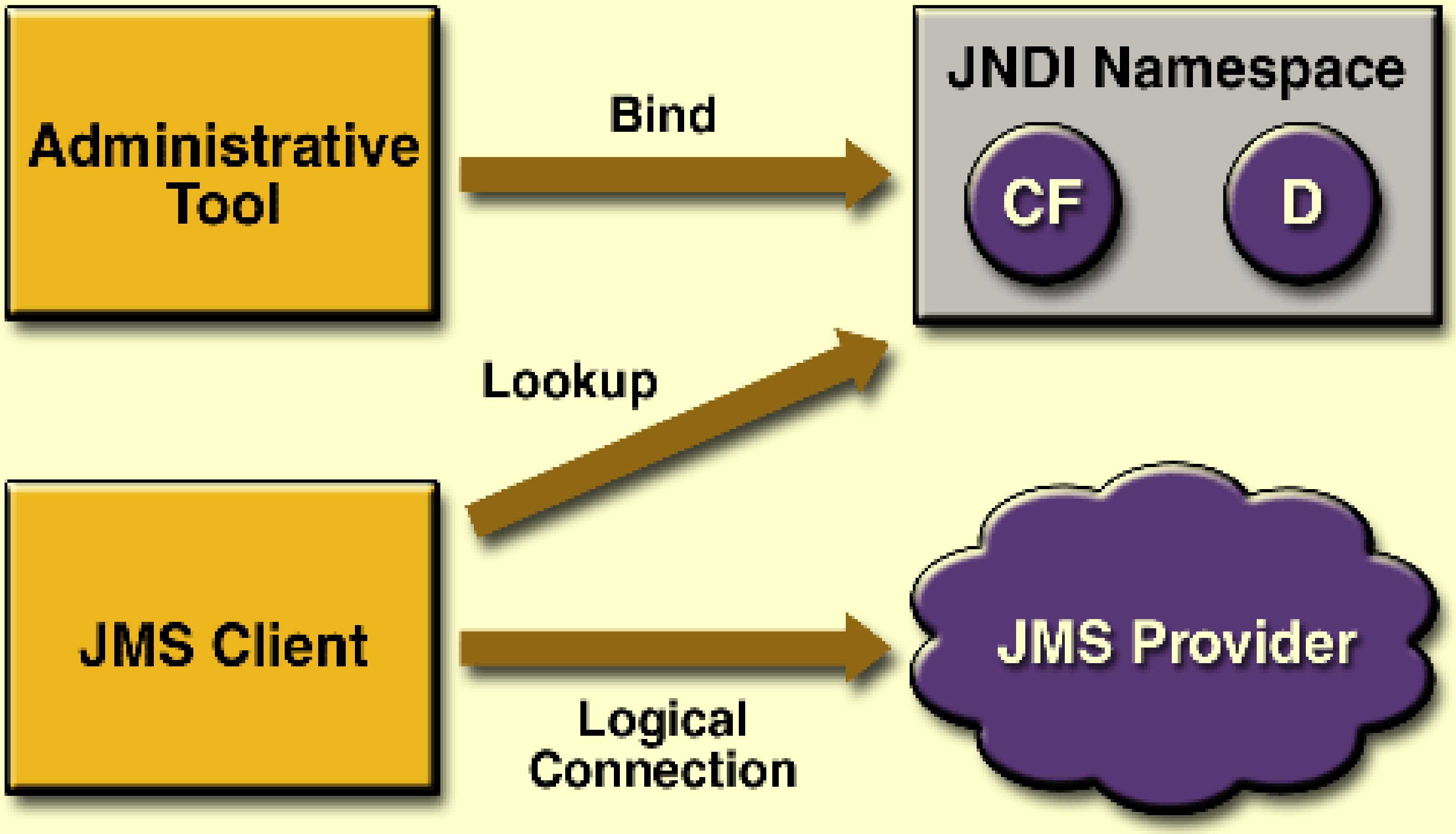
- vyšší pružnost
- vyšší složitost
- podpora Java, J2EE servers

Mechanismy

- asynchronní point-to-point
- synchronní point-to-point
- Publish / Subscribe

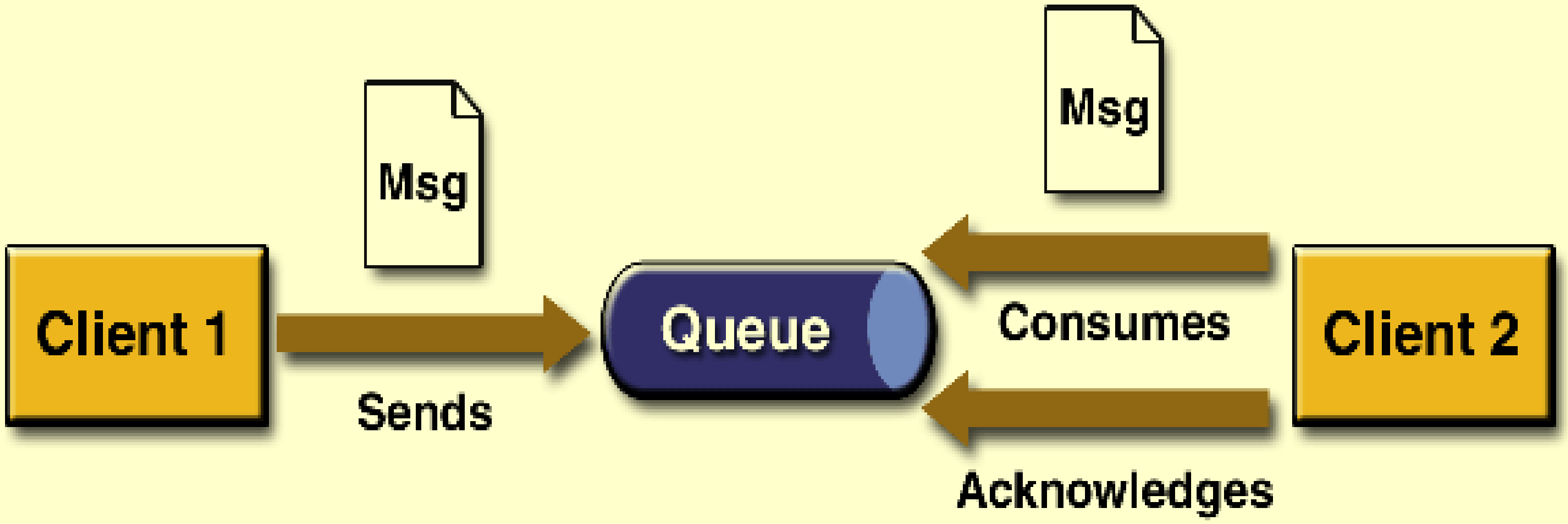
# JMS

## architektura



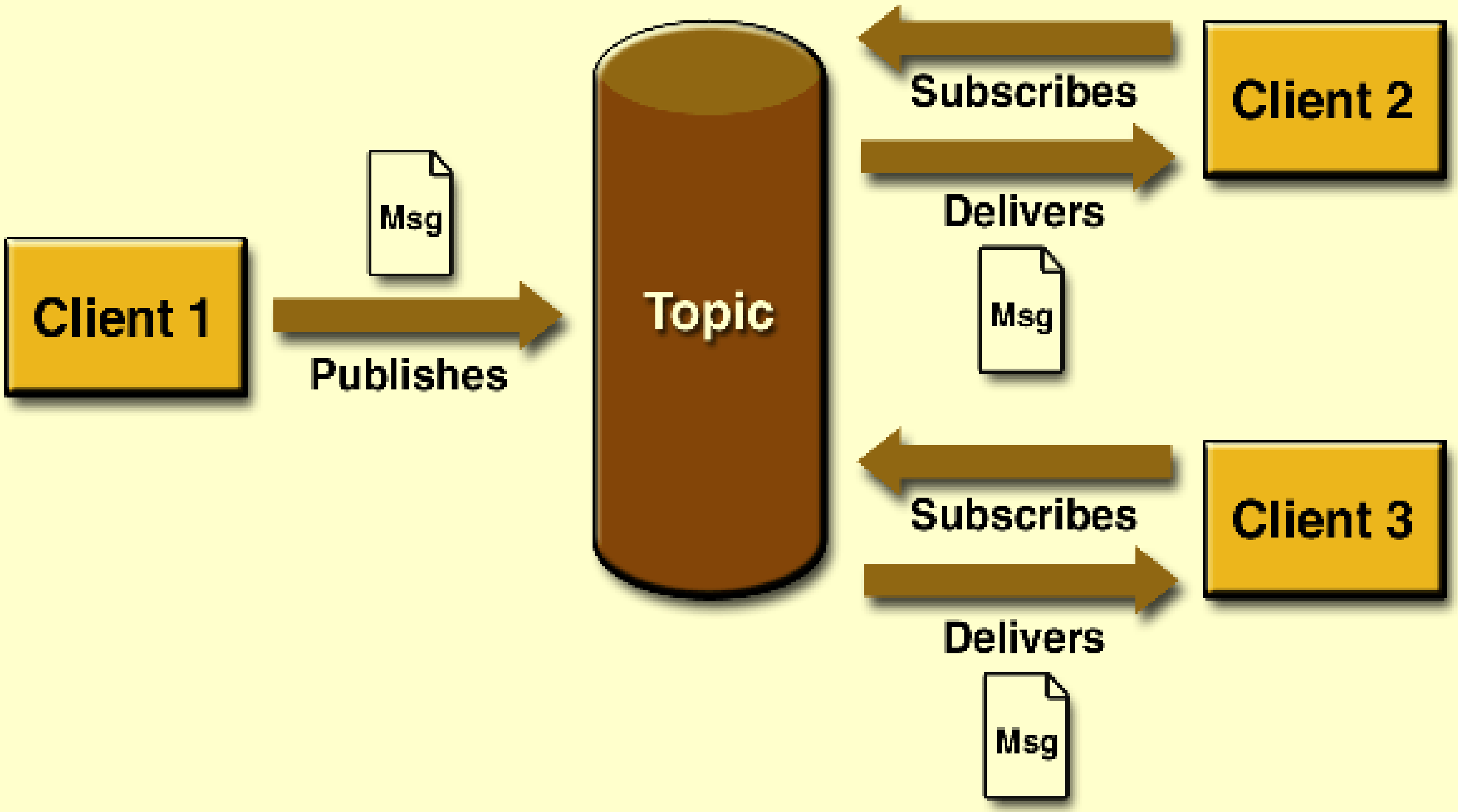
## JMS

point-to-point komunikace



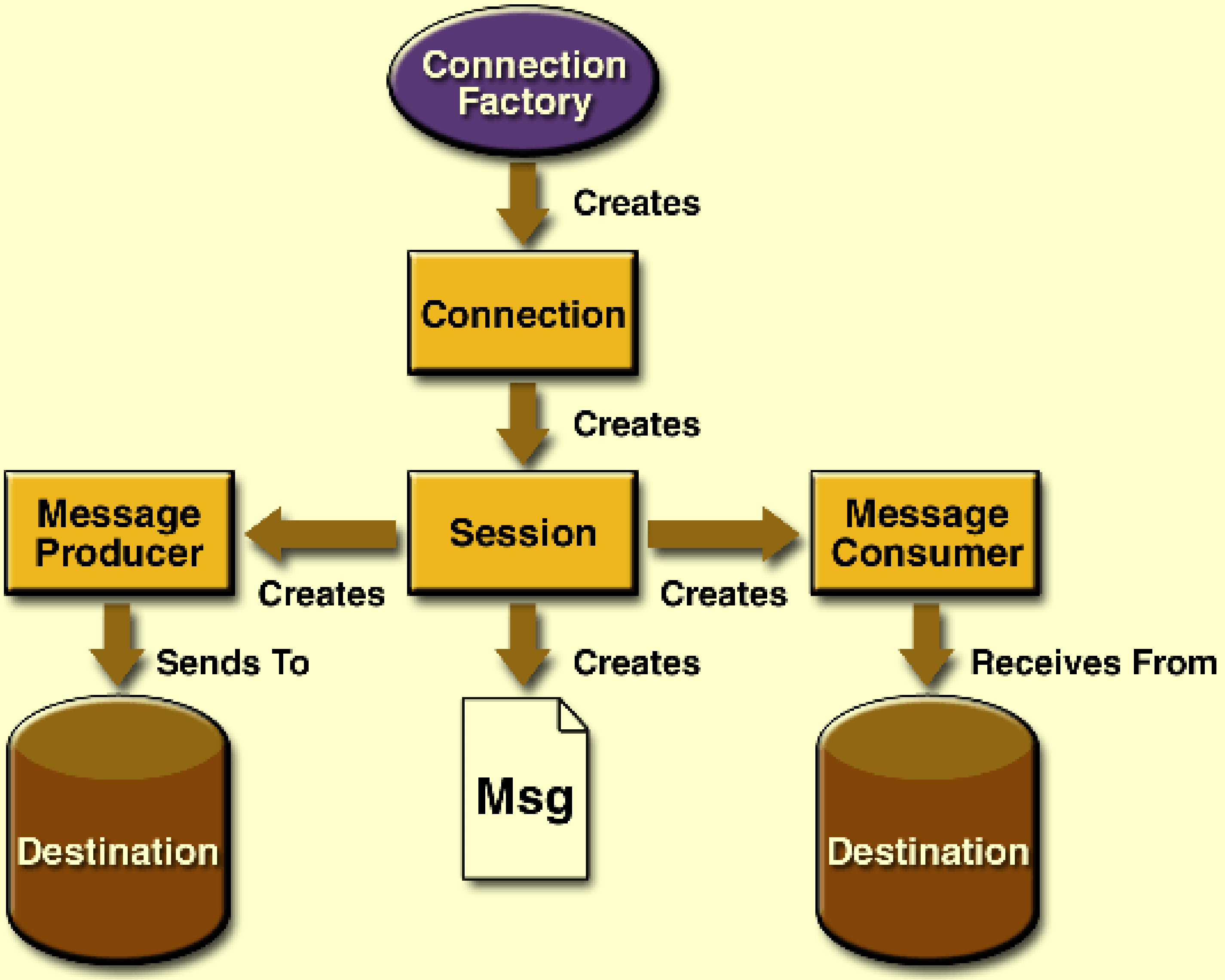
# JMS

publish/subscribe komunikace



# JMS

## Java Messaging Service API



## JMS – point-to-point sender

```
import javax.jms.*;
import javax.naming.*;

public class SimpleQueueSender {

  public static void main(String[] args) {
      String  queueName = null;
      Context  jndiContext = null;
      QueueConnectionFactory  queueConnectionFactory = null;
      QueueConnection  queueConnection = null;
      QueueSession  queueSession = null;
      Queue  queue = null;
      QueueSender  queueSender = null;
      TextMessage message = null;
      final int  NUM_MSGS;
```

# JMS – point-to-point sender

```
if ( (args.length < 1) || (args.length > 2) ) {
    System.out.println("Usage: java SimpleQueueSender " +
        "<queue-name> [<number-of-messages>]");
    System.exit(1);
}
queueName = new String(args[0]);
System.out.println("Queue name is " + queueName);
if (args.length == 2){
    NUM_MSGS = (new Integer(args[1])).intValue();
} else {
    NUM_MSGS = 1;
}
```

# JMS – point-to-point sender

```
/* Create a JNDI API InitialContext object if none exists yet.  */
try {
   jndiContext = new InitialContext();
} catch (NamingException e) {
   System.out.println("Could not create JNDI API " +
      "context: " + e.toString());
   System.exit(1);
}
/* Look up connection factory and queue. */
try {
   queueConnectionFactory = (QueueConnectionFactory)
      jndiContext.lookup("QueueConnectionFactory");
   queue = (Queue) jndiContext.lookup(queueName);
} catch (NamingException e) {
   System.out.println("JNDI API lookup failed: " +
      e.toString());
   System.exit(1);
}
```

# JMS – point-to-point sender

```
try {
    queueConnection =
        queueConnectionFactory.createQueueConnection();
    queueSession =
        queueConnection.createQueueSession(false,
            Session.AUTO_ACKNOWLEDGE);
    queueSender = queueSession.createSender(queue);
    message = queueSession.createTextMessage();
    for (int i = 0; i < NUM_MSGS; i++) {
        message.setText("This is message " + (i + 1));
        System.out.println("Sending message: " +
            message.getText());
        queueSender.send(message);
    }
```

# JMS – point-to-point sender

```
             /* Send a non-text control message indicating end */
            queueSender.send(queueSession.createMessage());
        } catch (JMSException e) {
            System.out.println("Exception occurred: " +
                e.toString());
        } finally {
            if (queueConnection != null) {
                try {
                    queueConnection.close();
                } catch (JMSException e) {}
            }
        }
    }
}
```

# JMS – point-to-point receiver

```
import javax.jms.*;
import javax.naming.*;
public class SimpleQueueReceiver {
   public static void main(String[] args) {
        String  queueName = null;
        Context  jndiContext = null;
        QueueConnectionFactory  queueConnectionFactory = null;
        QueueConnection  queueConnection = null;
        QueueSession  queueSession = null;
        Queue  queue = null;
        QueueReceiver  queueReceiver = null;
        TextMessage  message = null;
```

# JMS – point-to-point receiver

```
if (args.length != 1) {
    System.out.println("Usage: java " +
        "SimpleQueueReceiver <queue-name>");
    System.exit(1);
}
queueName = new String(args[0]);
System.out.println("Queue name is " + queueName);
```

# JMS – point-to-point receiver

```
/* Create a JNDI API InitialContext object if none exists. */
try {
    jndiContext = new InitialContext();
} catch (NamingException e) {
    System.out.println("Could not create JNDI API " +
        "context: " + e.toString());
    System.exit(1);
}
/* Look up connection factory and queue. */
try {
    queueConnectionFactory = (QueueConnectionFactory)
        jndiContext.lookup("QueueConnectionFactory");
    queue = (Queue) jndiContext.lookup(queueName);
} catch (NamingException e) {
    System.out.println("JNDI API lookup failed: " +
        e.toString());
    System.exit(1);
}
```

# JMS – point-to-point receiver

```
try {
    queueConnection =
        queueConnectionFactory.createQueueConnection();
    queueSession =
        queueConnection.createQueueSession(false,
            Session.AUTO_ACKNOWLEDGE);
    queueReceiver = queueSession.createReceiver(queue);
    queueConnection.start();
    while (true) {
        Message m = queueReceiver.receive(1);
        if (m != null) {
            if (m instanceof TextMessage) {
                message = (TextMessage) m;
                System.out.println("Reading message: " +
                    message.getText());
            } else {  break;  }
        }
    }
```

# JMS – point-to-point receiver

```
            } catch (JMSException e) {
                System.out.println("Exception occurred: " +
                    e.toString());
            } finally {
                if (queueConnection != null) {
                    try {
                        queueConnection.close();
                    } catch (JMSException e) {}
                }
            }
        }
    }
```