

Multiagentové systémy

Dr. Andrej Lúčný

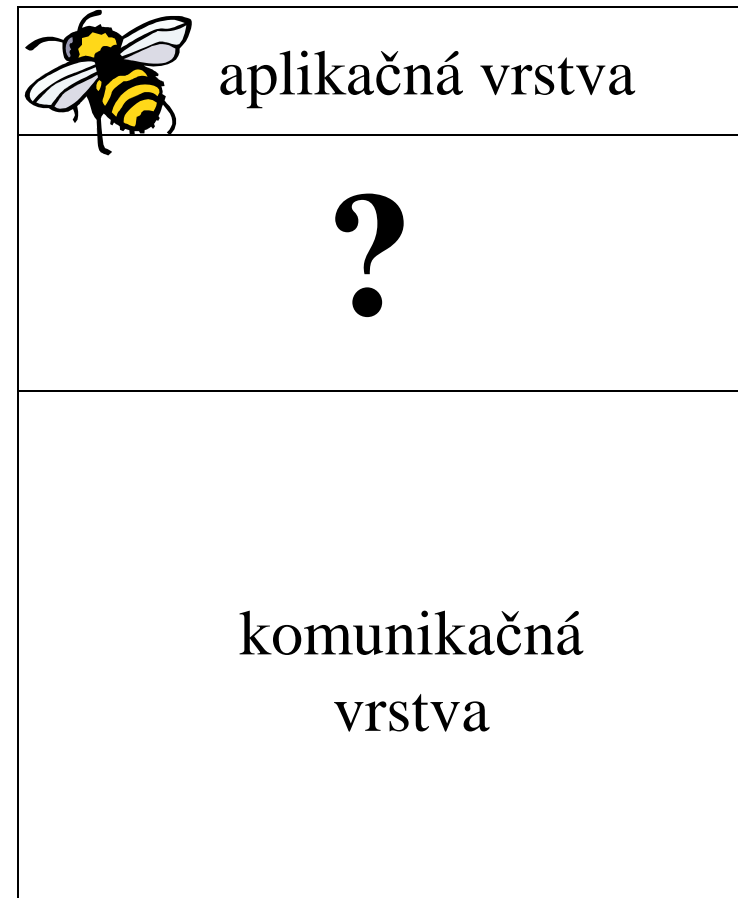
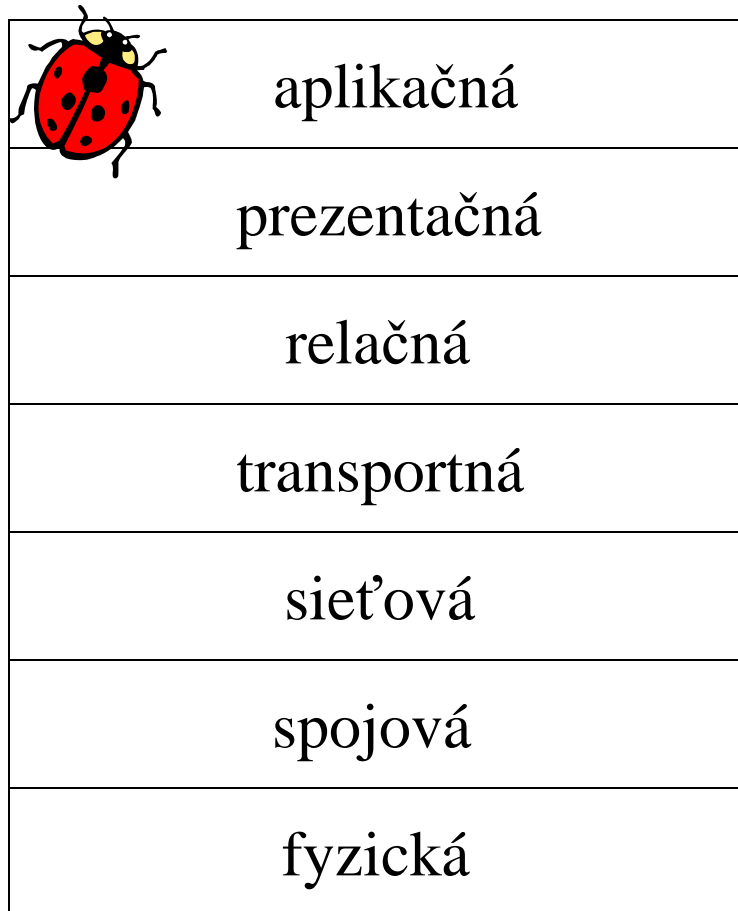
KAI FMFI UK

andy@microstep-mis.com

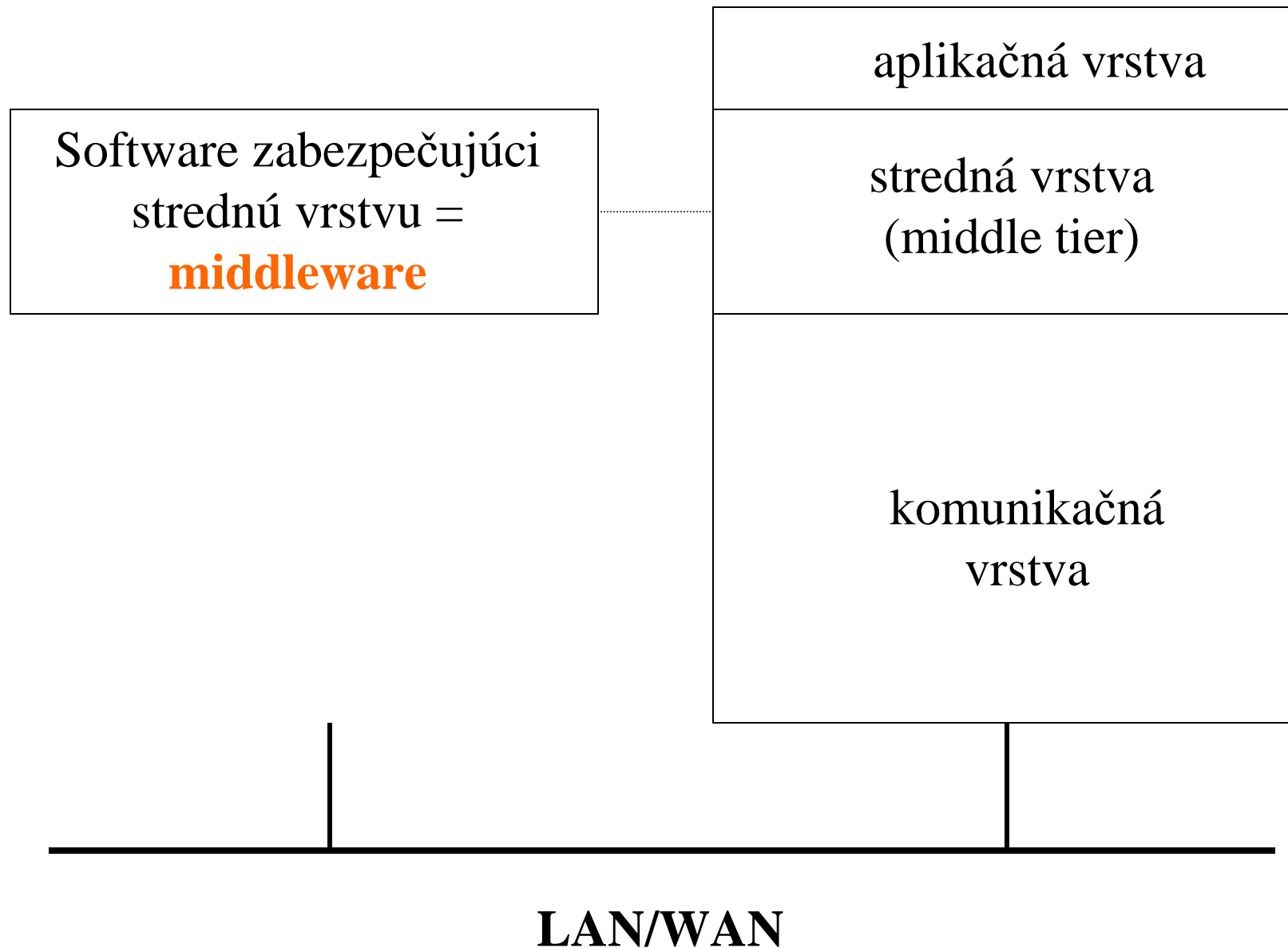
<http://www.microstep-mis.sk/~andy>

Opakovanie

- čo je OSI model ?
 - čo je frame a čo packet ?
 - čo je fyzická a logická adresa ?
 - čo je socket ?
 - čo je TCP/IP ?
 - čo sú aplikačné protokoly ?
-
- čo je middle tier ?
 - čo je middleware ?



LAN/WAN



Middleware

distribované objekty

služby



správy

transakcie

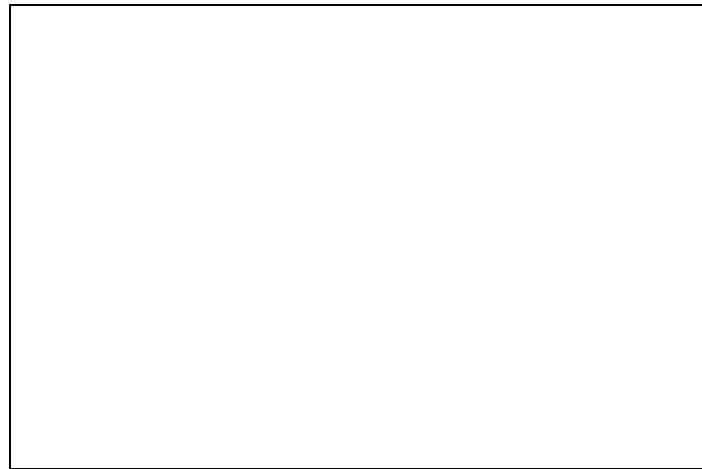
Middleware

CORBA-POA,

RPC, CORBA

RMI, COM+

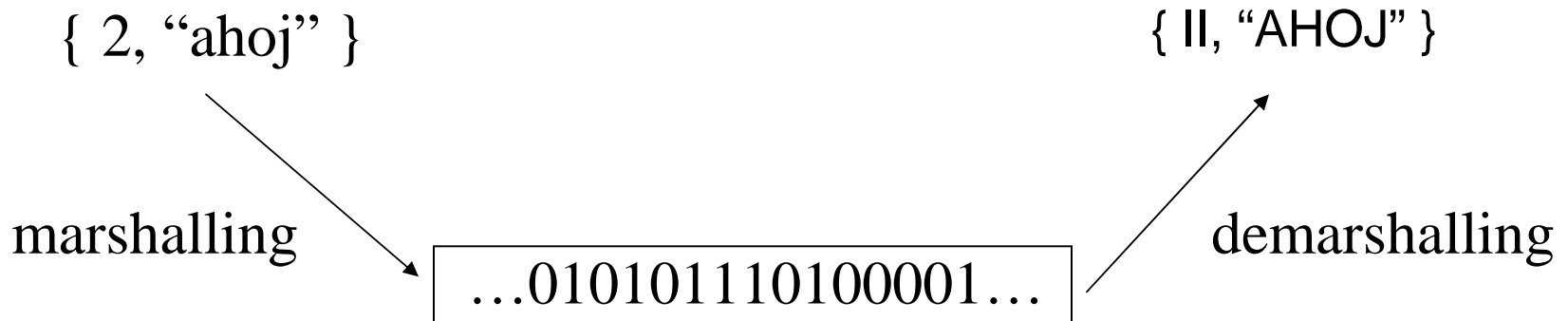
RMI-IIOP



SOAP

SQL

Marshalling / Demarshalling



- Najrozšírenejšou normou je IIOP protocol (CORBA)
- Norma budúcnosti bude založená na reprezentačnom jazyku

SOAP - příklad

```
<env:Envelope xmlns:env="http://www.w3.org/2002/12/soap-envelope">
  <env:Header>
    <n:alertcontrol xmlns:n="http://example.org/alertcontrol">
      <n:priority>1</n:priority>
      <n:expires>2001-06-22T14:00:00-05:00</n:expires>
    </n:alertcontrol>
  </env:Header>
  <env:Body>
    <m:alert xmlns:m="http://example.org/alert">
      <m:msg>Pick up Mary at school at 2pm</m:msg>
    </m:alert>
  </env:Body>
</env:Envelope>
```

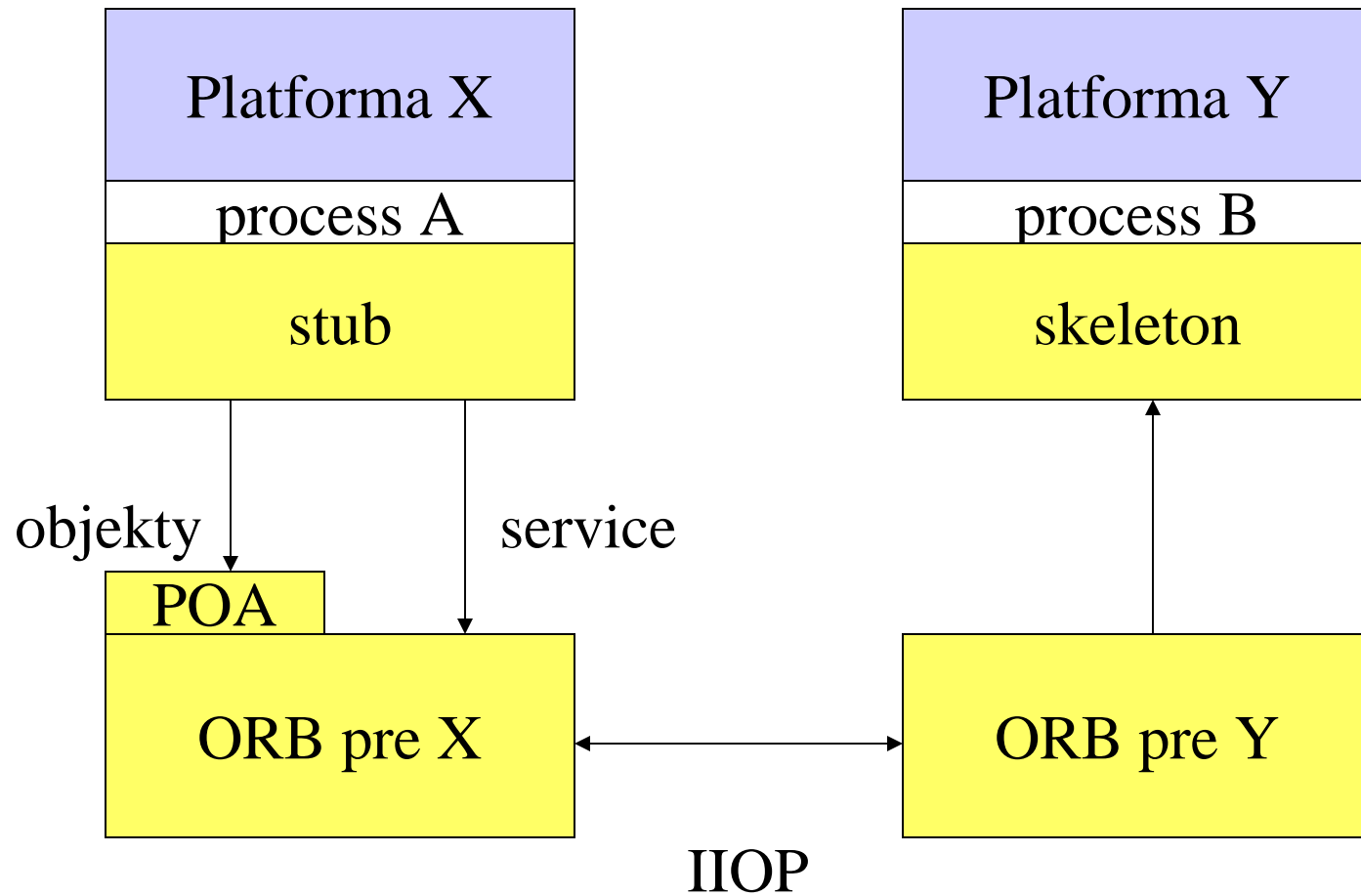
Reflekčný model

- Reflection model – umožňuje dynamicky zistiť triedu, premenné a metódy ľubovoľného objektu
- RM umožňuje napr. napísať program, ktorý k ľubovoľnému objektu vygeneruje kód v programovacom jazyku, po skompilovaní ktorého dostaneme triedu objektu
- JRM – implementácia RM v Jave

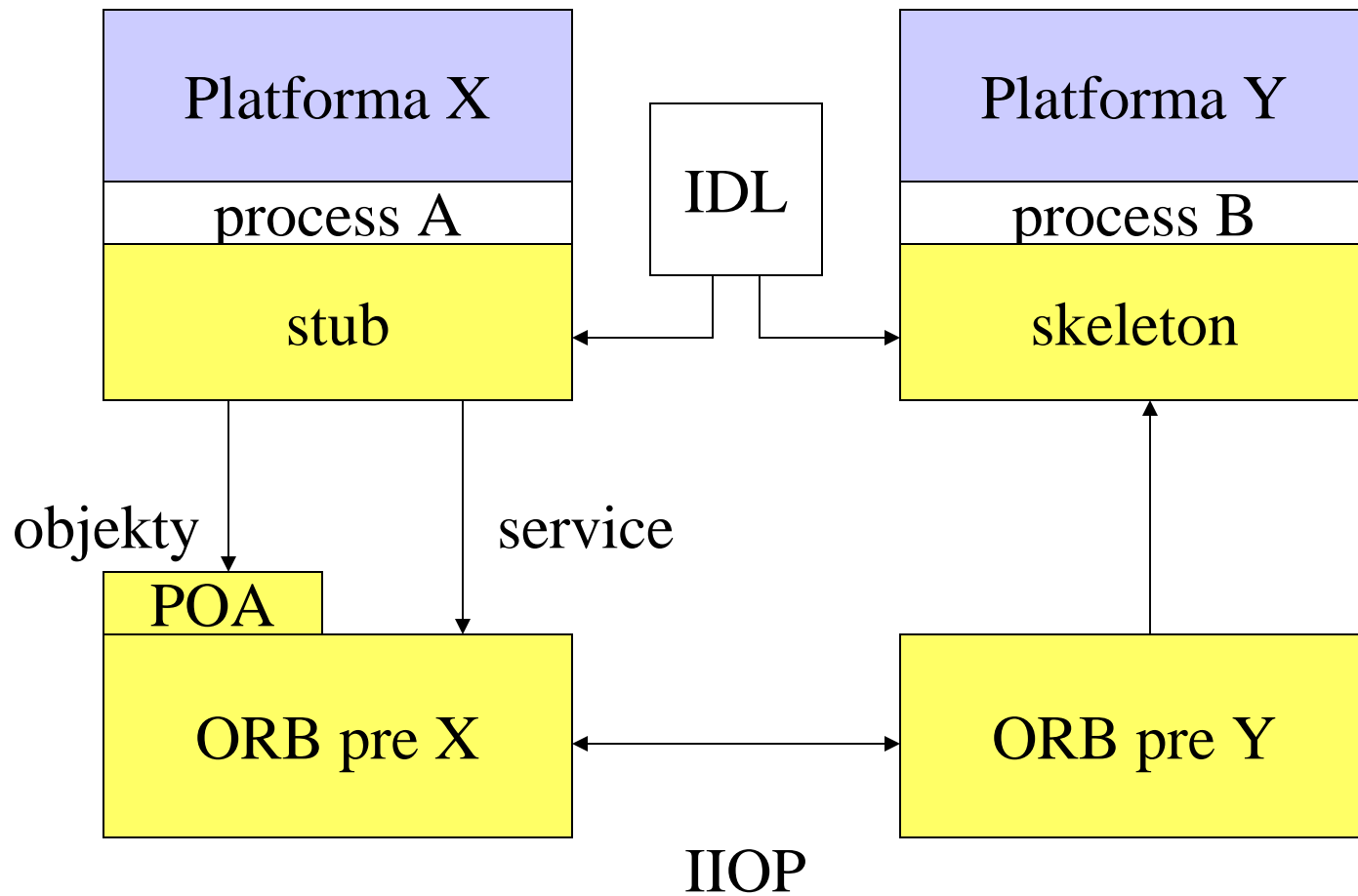
Stub / Skeleton

- Stub realizuje marshalling a demarshalling na strane volajúceho (klienta)
- Skeleton realizuje to isté u volaného (servera)
- Stub reprezentuje server u klienta
- Skeleton reprezentuje klient u servera
- Pokiaľ nemáme reflekčný model, nie je možné napísať univerzálny stub alebo skeleton = musíme ich vygenerovať pre každé použitie (máme na to nejakú utilitu)

CORBA



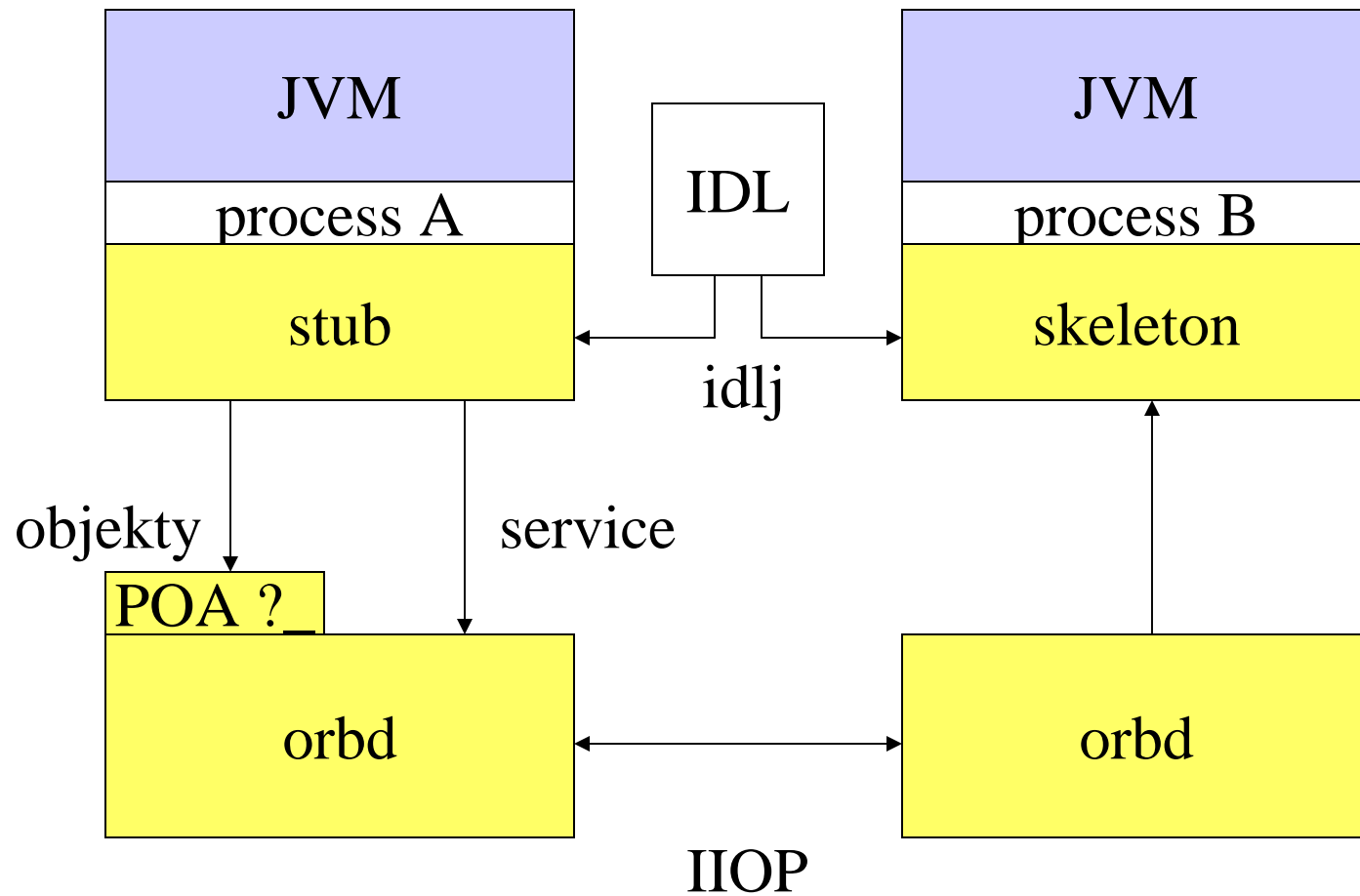
CORBA



CORBA - použitie

- Vyjadríme rozhranie v IDL
- Skompilujeme IDL na Stub a Skeleton
- Implementujeme server
- Používame rozhranie v klientovi

CORBA v Java



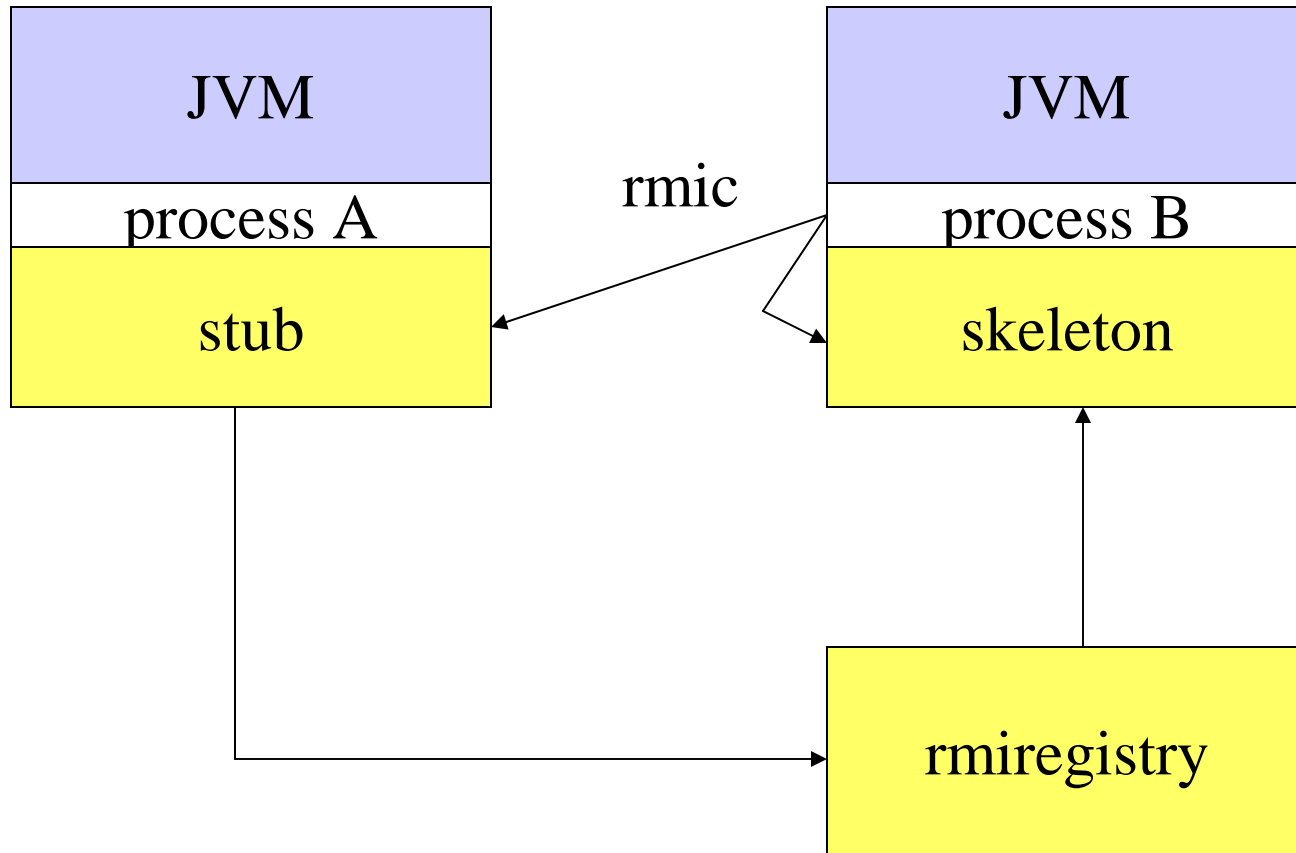
IDL

```
module HelloApp
{
  interface Hello
  {
    string sayHello();
    oneway void shutdown();
  };
};
```

CORBA

- je multiplatformná
- zavádza vlastné základné typy, ktoré mapuje na konkrétne typy na tej-ktorej platforme
- na identifikáciu klienta používa univerzálny identifikátor IOR
- ten sa dá získať napr. cez naming service

RMI



RMI - použitie

- Napíšeme rozhranie objektu, ktoré sa minimálne líši od bežných objektov
- Implementujeme časti vyplývajúce z potreby nadviazať spojenie
- Implementujeme logiku servera
- Používame rozhranie v klientovi ako keby to bol prítomný objekt

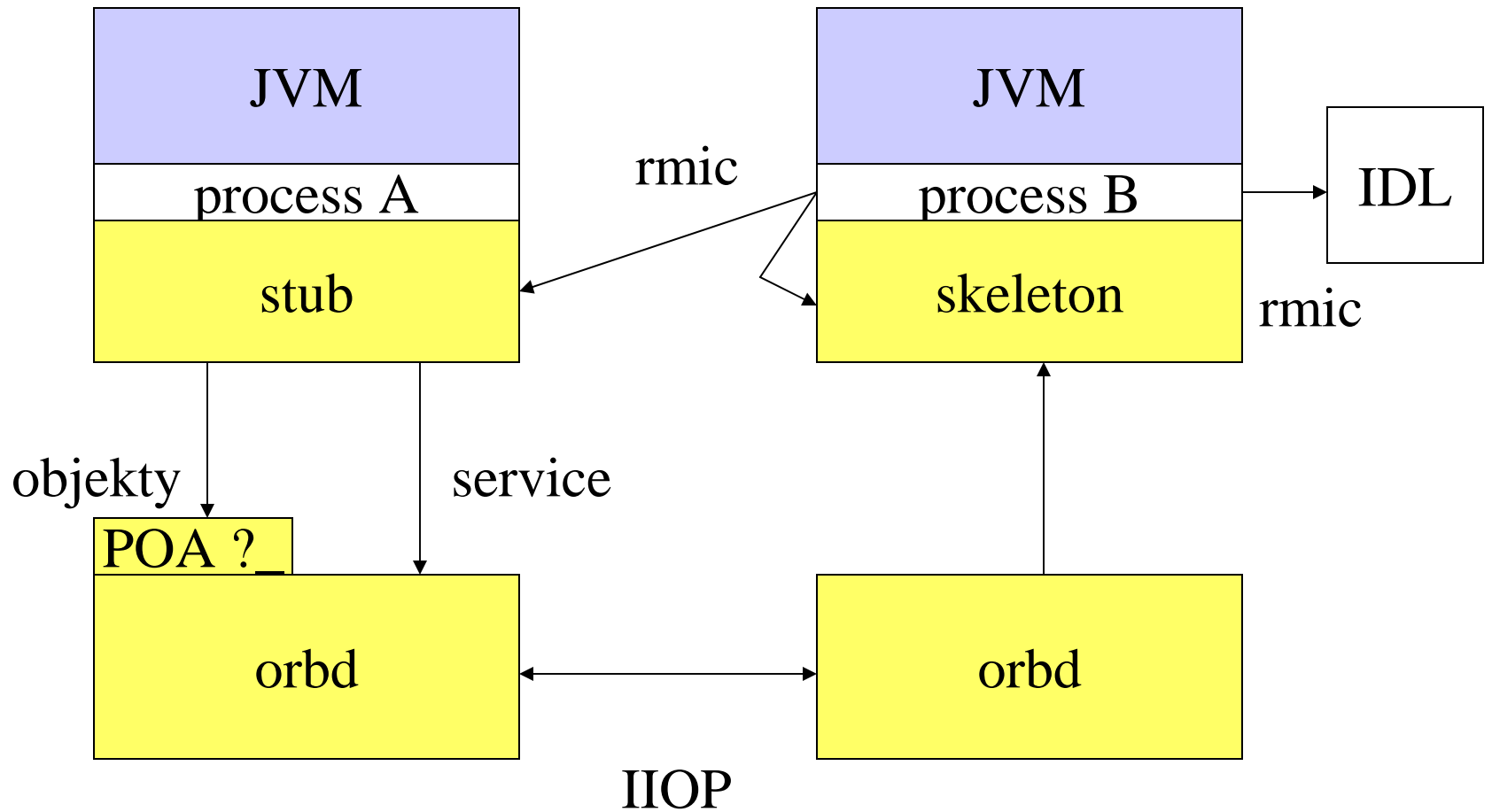
RMI

- Vďaka JRM RMI nepotrebuje IDL
- Využíva štandardný marshalling Javy (rozhranie Serializable, serialVersionUID)
- Na nájdenie servera sa používa RMI Registry (služba na báze TCP/IP)

RMI - IIOP

- Nevýhodou RMI je jeho uzavretosť pre platformu Java
- Interoperabilitu, napr. s jazykom C dokáže (teoreticky) zabezpečiť IIOP (funguje od CORBA 2.3)

RMI-IIOP



RMI-IIOP použitie

- Postupujeme ako pri RMI, ale skompilujeme iný stub a skeleton (používajúci IIOP miesto štandardného marshallingu)
- Z definície RMI remote objektu vygenerujeme IDL
- Skompilujeme IDL na CORBA stub v C
- Používame tento stub v klientovi v C

Web services

- Služby rozšiřující koncept http protokolu



GET /info/index.html HTTP/1.1

Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword, application/x-shockwave-flash, */*

Referer: http://www.swim.sk

Accept-Language: sk,en-us;q=0.5

Accept-Encoding: gzip, deflate

User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)

Host: www.swim.sk

Connection: Keep-Alive

Cache-Control: no-cache

HTTP/1.1 200 OK

Date: Sun, 11 Sep 2005 11:09:03 GMT

Server: Apache/2.0.54 (Debian GNU/Linux) mod_python/3.1.3 Python/2.3.5 PHP/4.3.10-16 mod_ssl/2.0.54 OpenSSL/0.9.7e mod_perl/1.999.21 Perl/v5.8.4

X-Powered-By: PHP/4.3.10-16

Content-Length: 2178

Connection: close

Content-Type: text/html

<html>

<meta http-equiv='Content-Type' content='text/html; charset=windows-1250'>

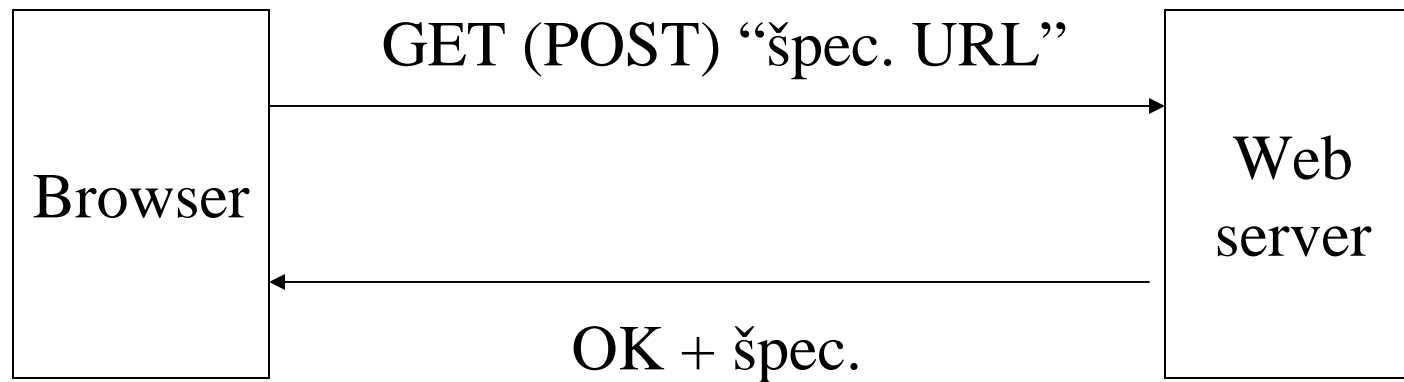
<body> ahoj </body>

</html>

HTTP

Web services

?param1=value1¶m2=value2 pre GET
alebo napr. XML pre POST



napr:
XMLHttpRequest

napr:
servlet

Web services

- Normy na webové služby: WSDL
- Platformy:

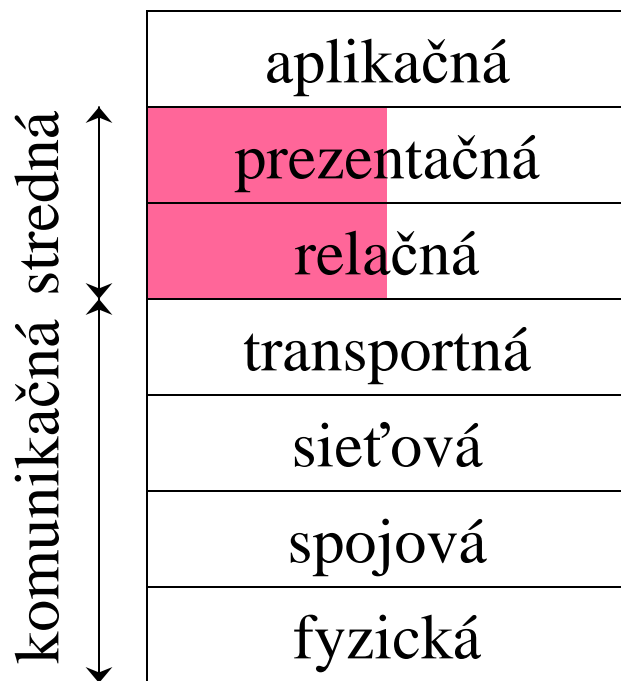
axis = soap(data encoding) + WSDL(interface encoding) +
apache (webserver) + java (SAAJ – JAX-RPC)

```
// example.wsdd
<deployment xmlns="http://xml.apache.org/axis/wsdd/"
             xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">
  <handler name="print" type="java:LogHandler"/>
    <service name="AddFunction1Service" provider="java:RPC">
      <requestFlow>
        <handler type="print"/>
      </requestFlow>
      <parameter name="className" value="AddFunction1"/>
      <parameter name="allowedMethods" value="*" />
      <beanMapping qname="myNS:Complex" xmlns:myNS="urn:BeanService"
                  languageSpecificType="java:Complex" />
    </service>
</deployment>
```

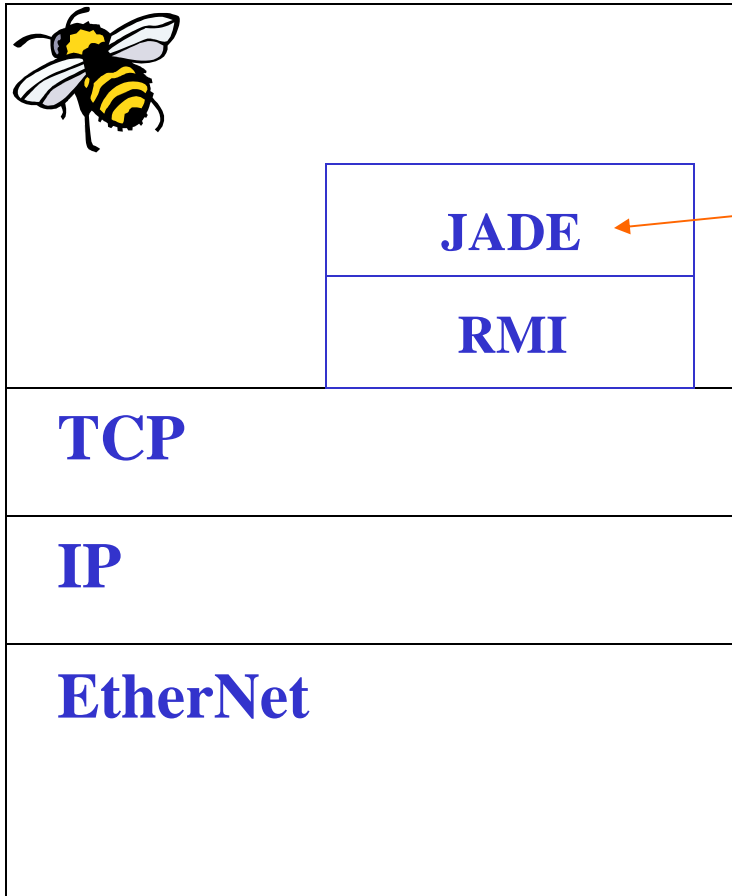
Web services - použitie

- Nakódujeme and preložíme v Jave triedu so statickou metódou
- Túto .class vložíme na webserver (Apache) do webapps/axis ako .jws
- Vo WSDL popíšeme rozhranie metódy (je na to interaktívna služba
<http://localhost:8080/axis/AddFunction.jws?wsdl>)
- Napíšeme klienta, ktorému povieme URL jws budeme používať, on dynamicky vytvorí stub a môžeme metódu zavolať.

MAS ako middleware



- **software podporujúci tvorbu MAS je vždy postavený na nejakej komunikačnej základni**
- **predstavuje špecifický druh middleware založeného na message passing-u**
- **môže sa opierať o iný middleware (často je realizovaný pomocou distribuovaných objektov)**



Implementácia
komunikačného
jazyka agentov od
FIPA

Príklad

MAS API

- Aké rozhranie poskytuje implementácia MAS pre aplikačnú úroveň ?

Sú dve možnosti

- Prevládajúca priama komunikácia (peer-to-peer) (napr. JADE)
- Prevládajúca nepriama komunikácia (stigmergic communication) (napr. Cougaar)