Multiagentové systémy

Andrej Lúčny

KAI FMFI UK

lucny@fmph.uniba.sk

http://www.agentspace.org/mas

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)

MAS
s prevládajúcou nepriamou komunikáciou

Služby nepriamej komunikácie

- Space poskytuje agentov služby, ktorými môžu manipulovať s uloženými dátami
- Základné služby sú

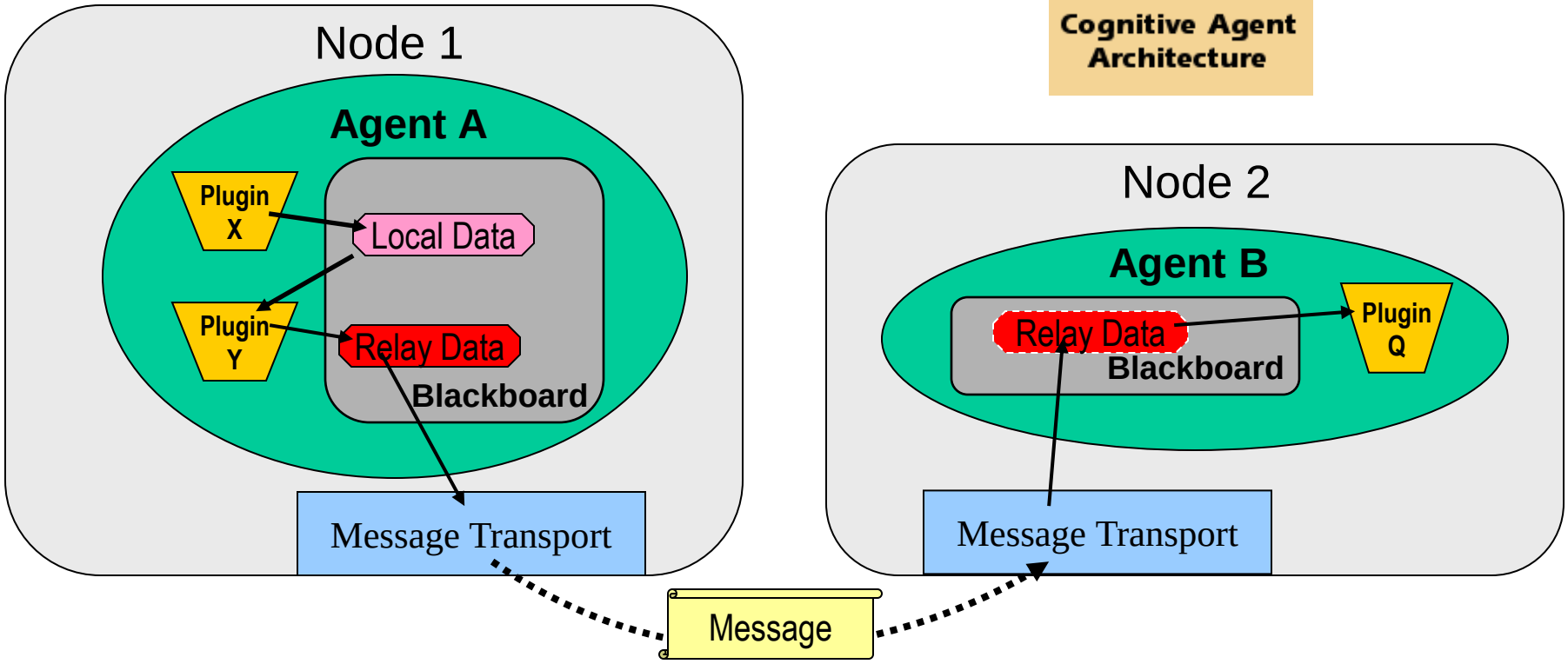
  READ

  WRITE

  DELETE
- neblokujúce a blokujúce (synchronizácia)

Príklad platformy s nepriamou komunikáciou

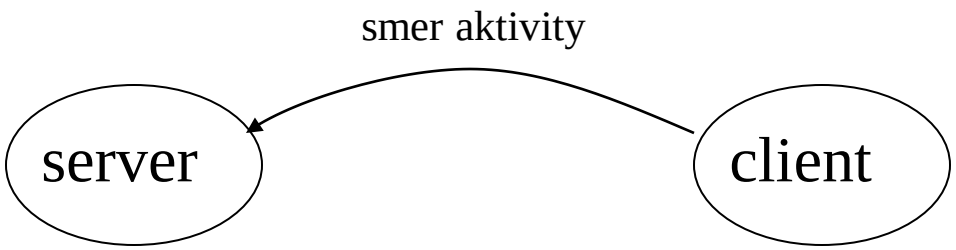
Cougaar



• MAS s nepriamou komunikáciou je distribuovaným systémom typu:

Client-Server

• označenie vzťahu dvoch procesov, kde jeden (server) poskytuje dáta či inú službu na základe požiadavky druhého (client).



Štruktúra serveru

Server môže spracúvať požiadavku rôznymi spôsobmi

1. Každú zvlášť
2. Môže si pamätať stav komunikácie v odovzdávaných dátach
3. Môže si pamätať stav komunikácie vo vlastnej štruktúre

Štruktúra serveru

Server môže spracúvať požiadavku rôznymi spôsobmi

1. Každú zvlášť — jednoduché
2. Môže si pamätať stav komunikácie v odovzdávaných dátach — nebezpečné
3. Môže si pamätať stav komunikácie vo vlastnej štruktúre — univerzálne

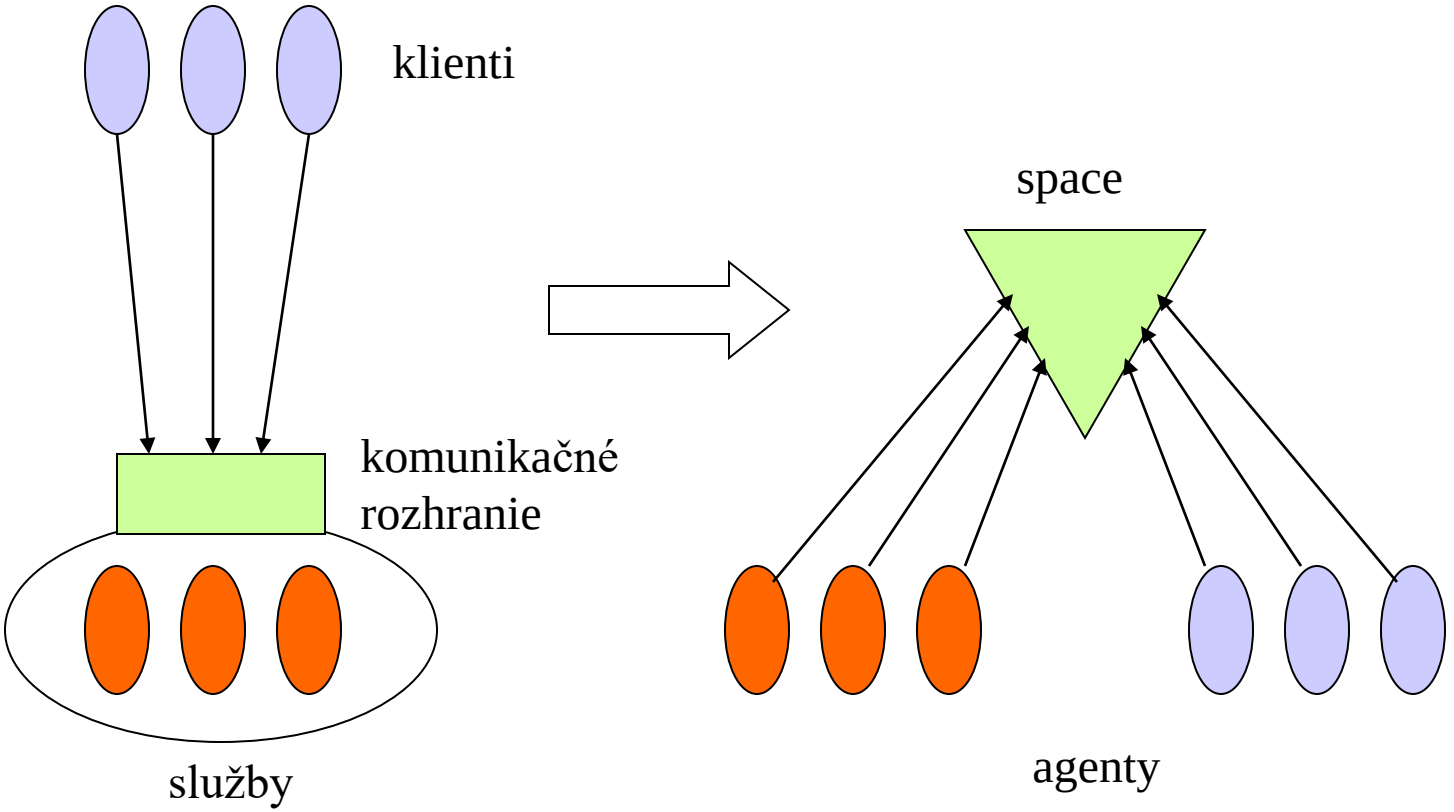
   takáto štruktúra sa nazýva **port**

MAS sú špeciálny prípad DS

V prípade, že je MAS špeciálny prípad distribuovaného systému typu client-server, tak:

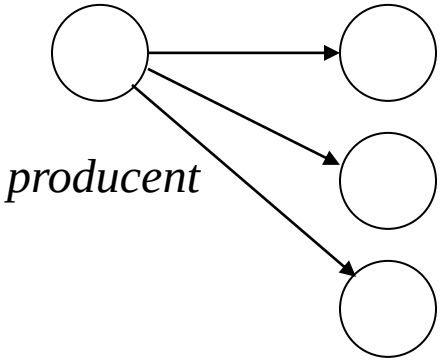
- server neobsahuje žiaden aplikačne závislý kód
- server poskytuje iba komunikačné služby
- dá sa na to teda nazerať ako na snahu o znovupoužiteľnosť servera
- client disponuje knižnicou ktorá mu poskytuje komfortné pripojenie na server
- server + knižnica = middleware

Transformácia z Client-Server do Agent-Space-Agent

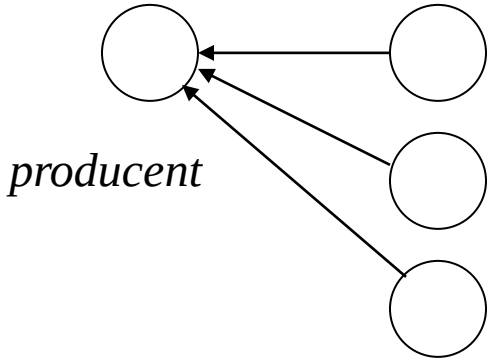


Typy dátových tokov

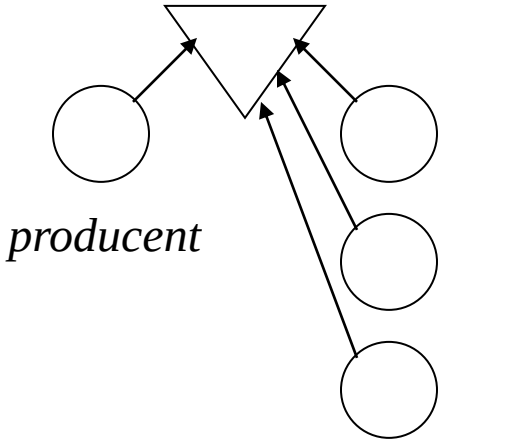
tradičný spôsob



klient-server



agent-space-agent



Vlastnosti space

- Je to server voči agentom - klientom
- aplikačná nezávislosť
- vysoká odolnosť a stabilita (efektívne algoritmy) (je to bottle-neck)
- poskytuje služby realizujúce komunikáciu medzi klientami
- pracuje s určitým marshallingom (ktorý je prípadne založený na nejakom jazyku)

Reprezentačný a komunikačný jazyk

Na základe toho, môžeme (pri nepriamej komunikácii) upresniť definíciu reprezentačného a komunikačného jazyka:

- reprezentačnému jazyku „rozumie“ agent, nie je súčasťou middleware, predstavuje tú časť dát, ktorú space „nerozbaľuje“
- komunikačnému jazyku „rozumie“ space a knižnica na komunikáciu s ním, je súčasťou middleware

Ďalšie služby nepriamej komunikácie

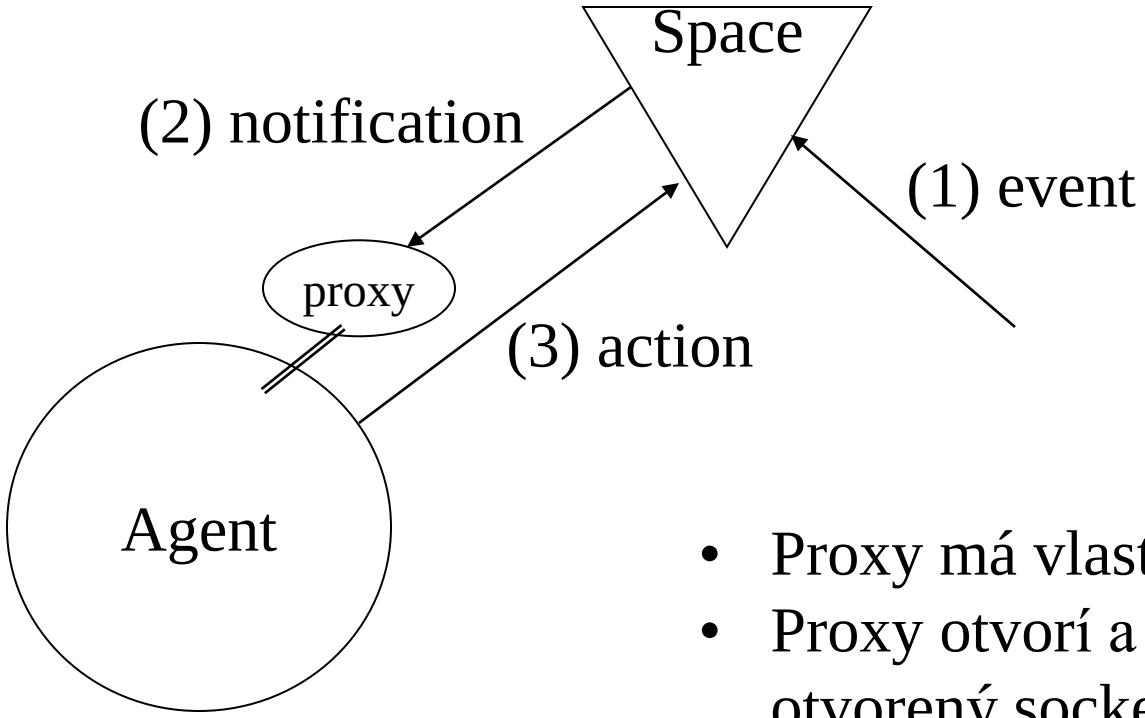
- Ďalšie služby:

  - registrácia triggrov (notifikácie)

  - hromadné manipulácie s blokmi na základe masky

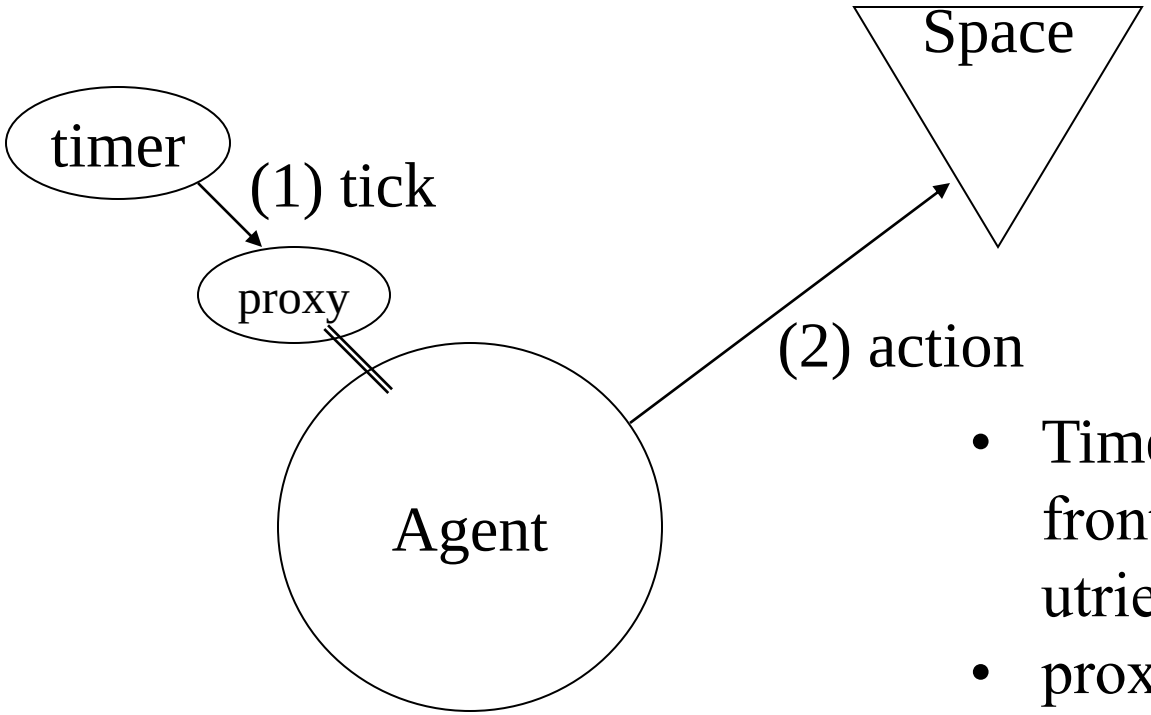
- Synchronizácia: jednotlivá služba sa vždy vykonáva bez prerušenia inou, agent má možnosť vykonať aj sériu služieb bez prerušenia

Trigger



- Proxy má vlastné vlákno
- Proxy otvorí a udržiava otvorený socket
- notifikáciu dostane ako oneskorenú odpoveď

Timer



- Timer má vlastné vlákno a front časovaných úloh, utriedený podľa času
- proxy je jeho časovaná úloha
- Proxy odblokuje vlákno agenta

Referencia uložených dát

- UUID a pod.
- meno
- unifikácia mena
- unifikácia dát (v reprezentačnom jazyku)

Implementácie

- historicky vychádzajú z jazyka LINDA (1985, v paralelnom programovaní)
- štandardy sú len hypotetického resp. akademického charakteru
- väčšinou sú proprietárneho charakteru

LINDA Tupple Space

Dátová štruktúra, ktorá dokáže obsahovať n-tice termov, v princípe LISPovské zoznamy a poskytuje služby

- **out(t)** zapisuje novú n-ticu
- **in(t)** prečíta a odstráni určitú n-ticu; pokiaľ taká nie je k dispozícii, proces sa v čítaní zablokuje do doby, kedy sa objaví
- **rd(t)** robí to čo in(t), len n-ticu neodstraňuje ale ponecháva
- **inp(t)** vracia TRUE a odstráni určitú n-ticu pokiaľ taká je; inak vráti FALSE
- **rdp(t)** robí to čo inp(t), len n-ticu neodstraňuje ale ponecháva

Pritom je pri čítaní na špecifikáciu manipulovanej n-tice použitý unifikačný princíp

Java Space

- časť Java Jini balíka, ktorý je určený na posun od sietí počítačov a služieb ku sieťam služieb a vecí (dynamické siete), jeho hlavnou časťou je Java Lookup Service
- ide o middleware vybudovaný nad RMI

Java Space

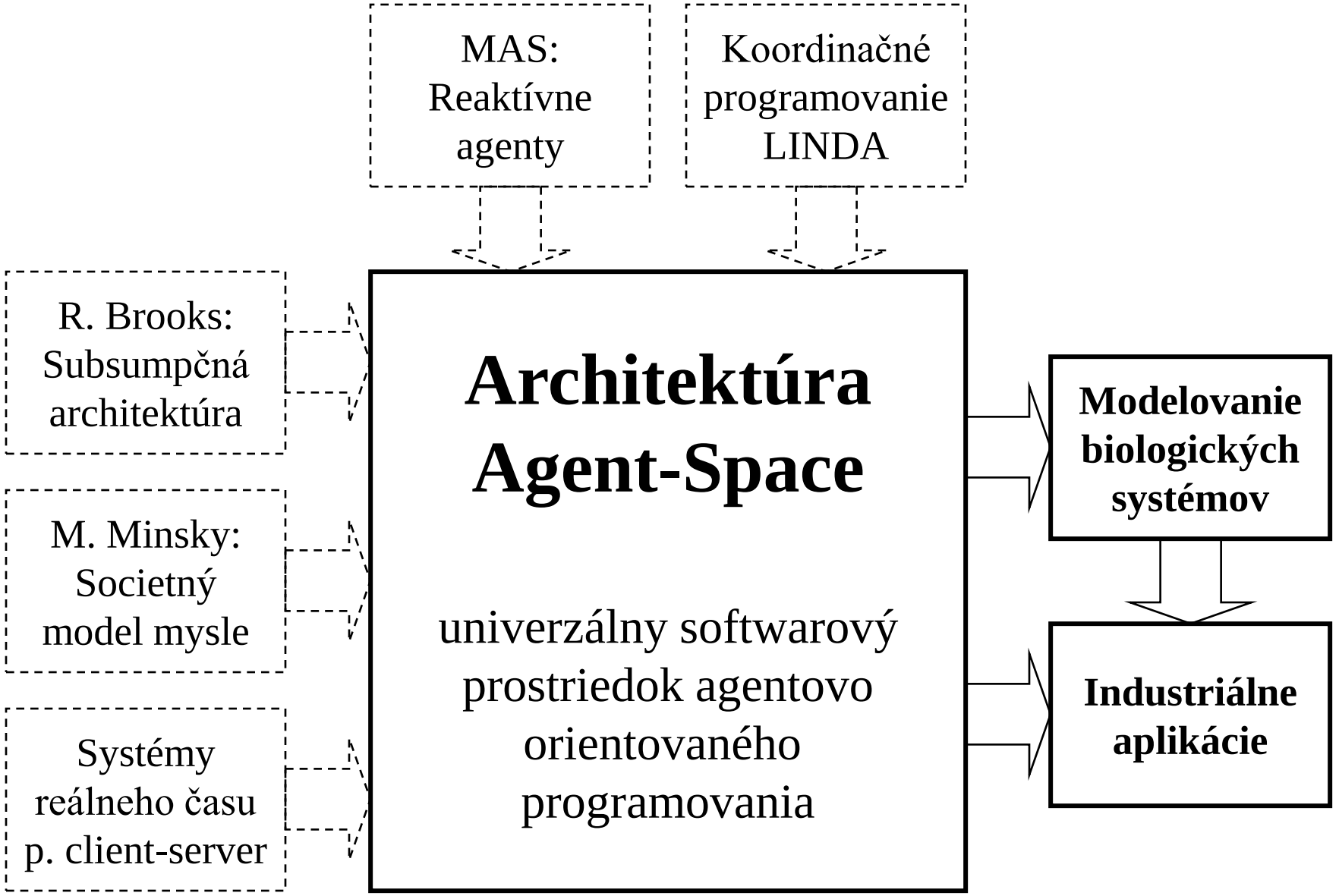
```
package net.jini.space;
import java.rmi.*
public interface JavaSpace {
   Lease write(Entry entry, Transaction txn, long lease);
   Entry read(Entry tmpl, Transaction txn, long timeout);
   Entry readIfExists(Entry tmpl, Transaction txn, long
            timeout);
   Entry take(Entry tmpl, Transaction txn, long timeout);
   Entry takeIfExists(Entry tmpl, Transaction txn, long
            timeout);
   EventRegistration notify(Entry tmpl, Transaction txn,
            RemoteEventListener ln, long lease,
            MarshalledObject handback);
   Entry snapshot(Entry e);
}
//throws clauses not shown
```

Data leasing (prenájom) – časová platnosť

- Java Space prvý krát zaviedol obmedzovanie časovej platnosti dát uložených v space
- space sa tu stará o to, aby po určitom čase boli určité dáta z neho odstránené
- to sa dá len do istej miery pasívne, a lepšie je keď space disponuje časovaným taskom, či na jeho realizáciu určeným vláknom.

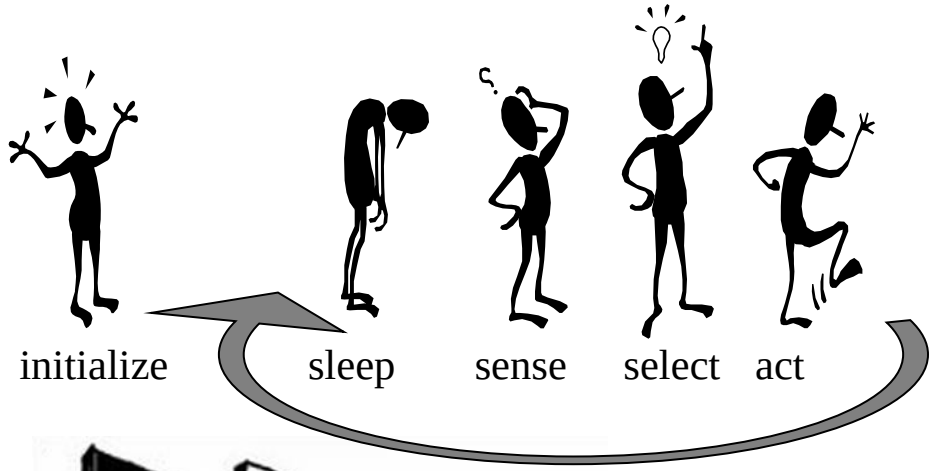
Agent-Space

- Multi-agentová architektúra navrhnutá na FMFI UK 1997-2004
- Vyjadruje tradičné myšlienky (Brooks, Minsky) novým jazykom (MAS s nepriamou komunikáciou)

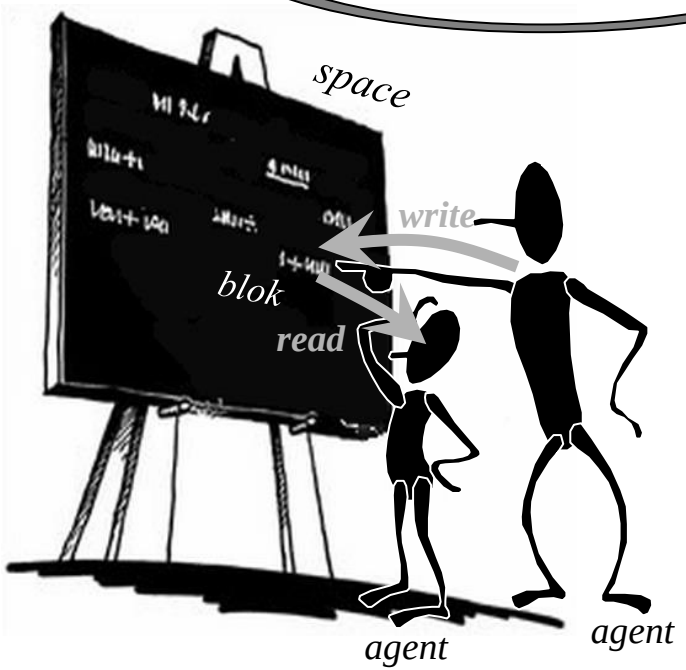

MAS:
Reaktívne
agenty
Koordinačné
programovanie
LINDA
R. Brooks:
Subsumpčná
architektúra
M. Minsky:
Societný
model mysle
Systémy
reálneho času
p. client-server
Architektúra
Agent-Space
univerzálny softwarový
prostriedok agentovo
orientovaného
programovania
Modelovanie
biologických
systémov
Industriálne
aplikácie

Architektúra Agent-Space

- Systém sa skladá z agentov

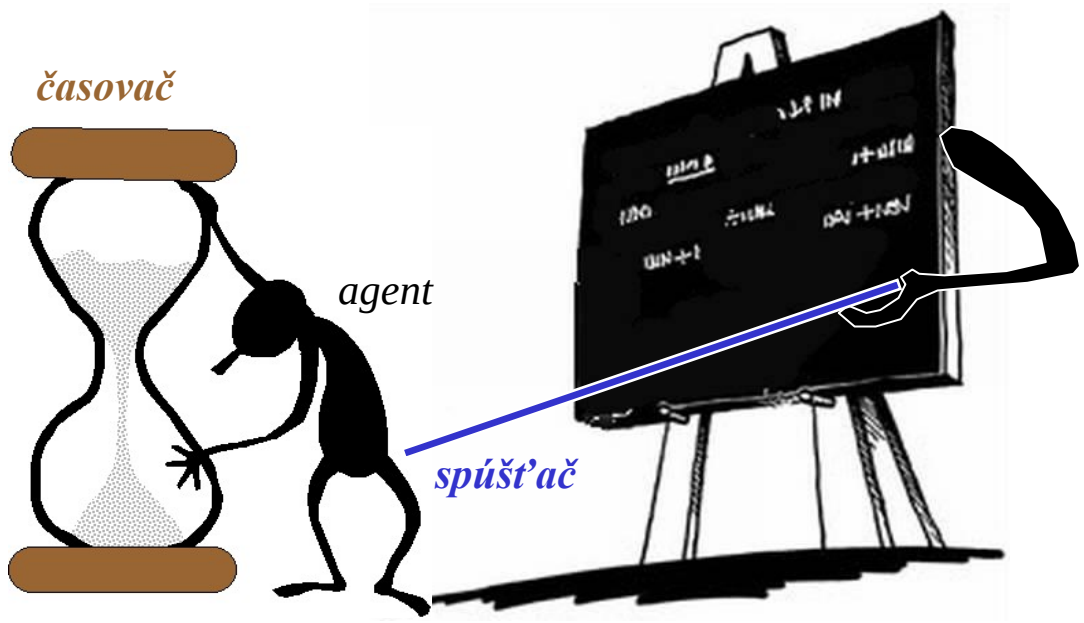


- Agenti medzi sebou komunikujú nepriamo cez Space (čiernu tabuľu)



Implementácia v C++ / Java

- Multivláknové prostredie pthreads / Thread
- Každý agent má vlastné vlákno
- Može volať metódy singletonu Space
- vlákno agenta je blokované na časovač alebo spúšťač



Ukážka kódu

```
package org.agentspace.demo;
import org.agentspace.*;

public class Agent1 extends Agent {

  int i = 0;

  public void init(String[] args) {
    attachTimer(1000);
  }

  public void senseSelectAct() {
    System.out.println("write: "+i);
    write("a",i++);
  }

}
```

```
public class Agent2 extends Agent {

  int i;

  public void init(String args[]) {
    attachTrigger("a");
  }

  public void senseSelectAct() {
    i =  (Integer) read("a",-1);
    System.out.println("read "+i);
  }

}
```

```
public class Starter {
  public static void main(String[] args) {
    new SchdProcess("space",“org.agentspace.SpaceFactory",new String[]{"DATA"});
    new SchdProcess("agent1",“org.agentspace.demo.Agent1",new String[]{});
    new SchdProcess("agent2",“org.agentspace.demo.Agent2", new String[]{});
  }
}
```

Implementácia služieb Space v distribuovaných prostrediach

- cez RMI
- cez web services
- ...

Web services

- Služby rozširujúce koncept http protokolu

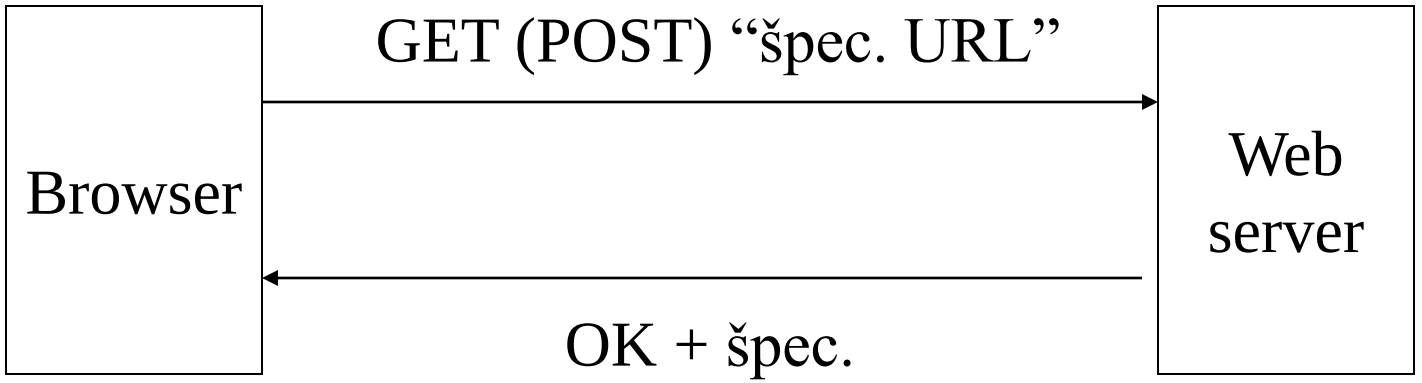
```
GET /info/index.html HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.ms-excel,
application/vnd.ms-powerpoint, application/msword, application/x-shockwave-flash, */*
Referer: http://www.swim.sk
Accept-Language: sk,en-us;q=0.5
Accept-Encoding: gzip, deflate
User-Agent: 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>
```

Web services

?param1=value1&param2=value2 pre GET
alebo napr. XML pre POST



napr:
XMLHttpRequest

napr:
servlet