

Multiagentové systémy

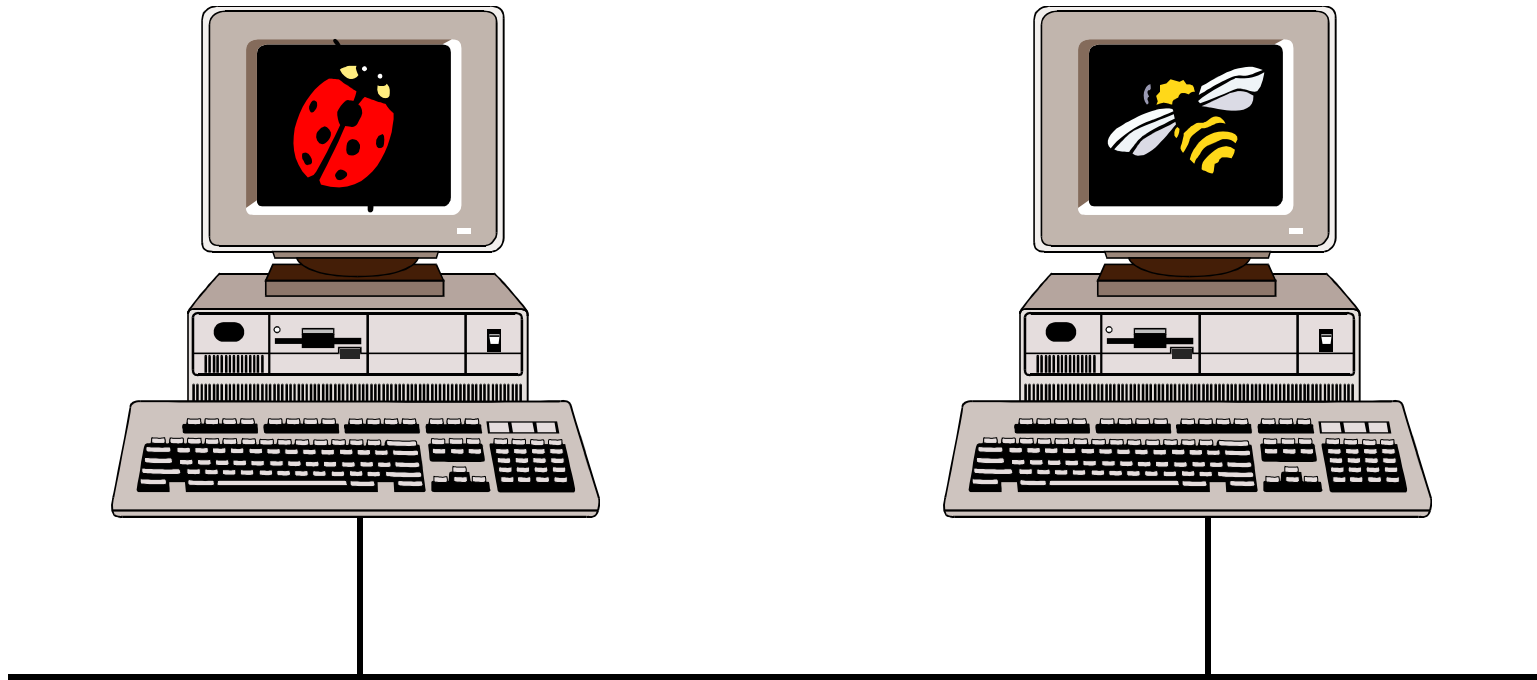
Dr. Andrej Lúčný

MicroStep-MIS

andy@microstep-mis.com

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

MAS ako middleware



LAN/WAN

Distribučovaný systém

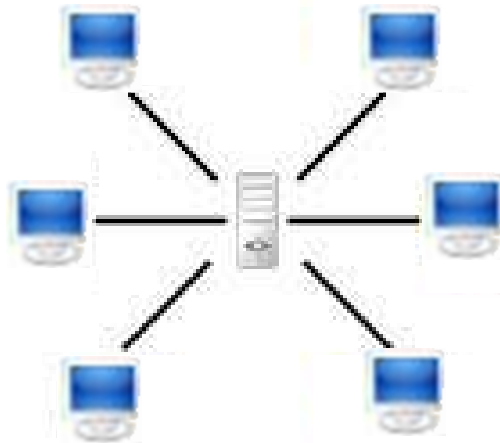
- v užšom význame: systém bežiaci na viacerých počítačoch rozmiestnených na rôznych miestach (hl'adisko nasadenia)
- V širšom význame: systém viacerých procesov, ktoré si vymieňajú dáta tak, ako distribučovaný systém v užšom význame tohto slova (hl'adisko kódovania)

Typy distribuovaných systémov

- mainframe
- client-server (two tier)
- client-server (three tier)
- client-server (n-tier)
- peer-to-peer
- SOA
- SOA / ESB

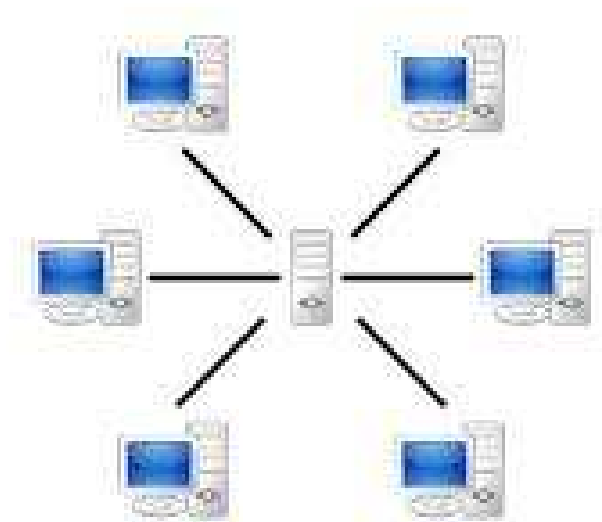
Mainframe

- bez sieťovej komunikácie, bez GUI



Client - Server

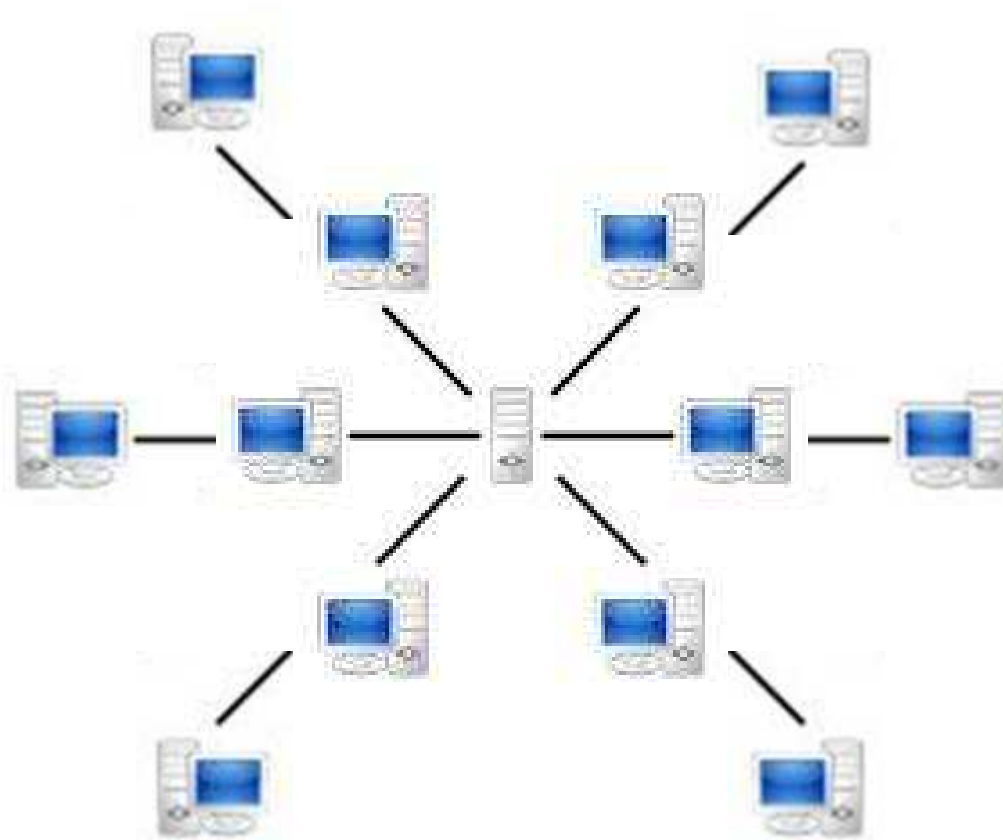
- označenie vzťahu dvoch počítačov, kde jeden (server) poskytuje dáta či inú službu na základe požiadavky druhého (client).



2-tier

Client-Server

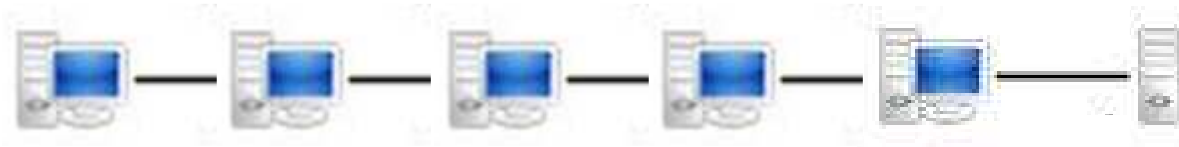
- (thin) client – application server - server



3-tier

Client-Server

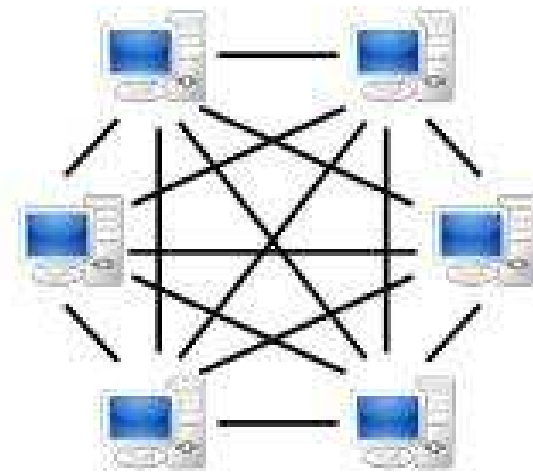
- client-server=client-server=client ... -server



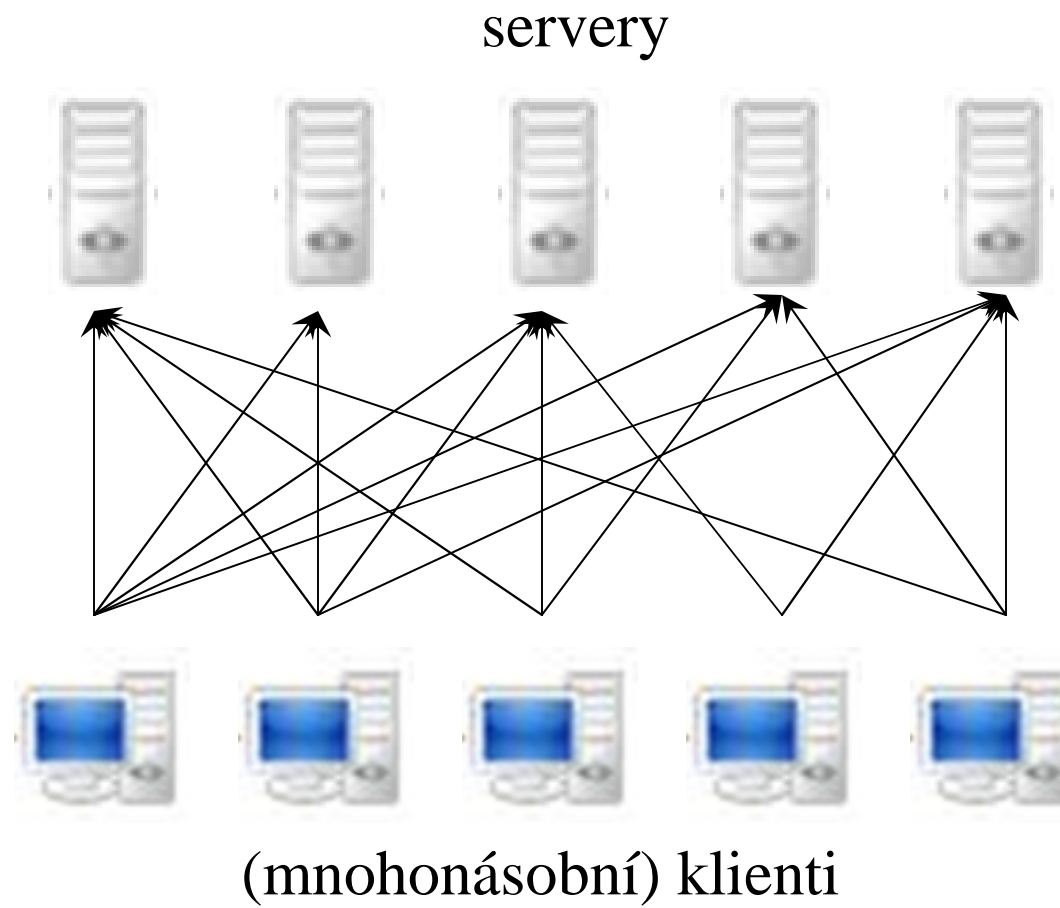
n-tier

Peer to peer

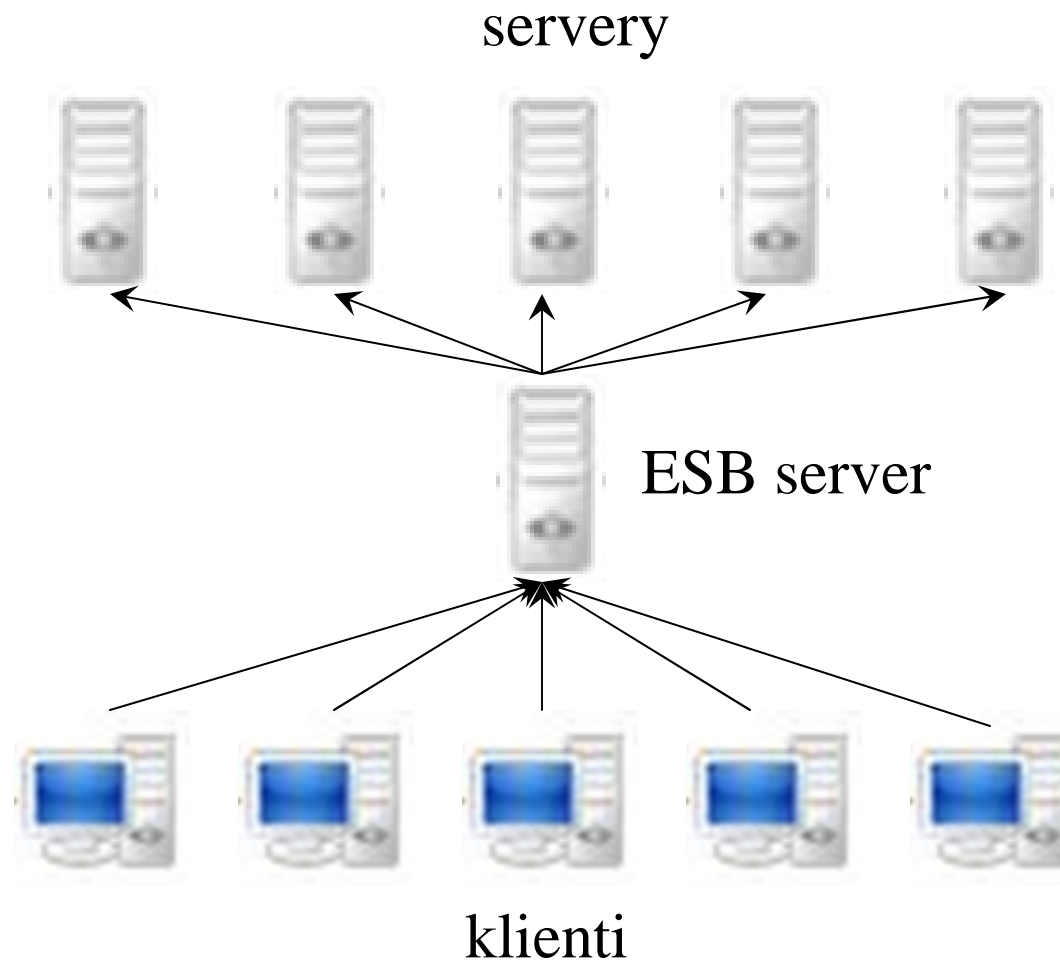
- minimálna potreba centrálnych serverov - opiera sa len o nejakú client-server službu ako je naming service
- Každý uzol je serverom pre ostatných peerov a zároveň klientom ostatných peerov



Service Oriented Architecture

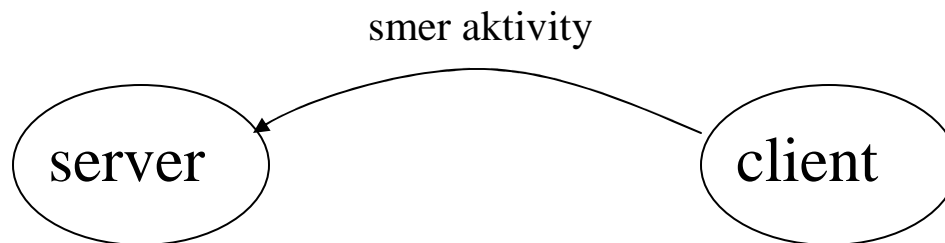


Enterprise Service Bus



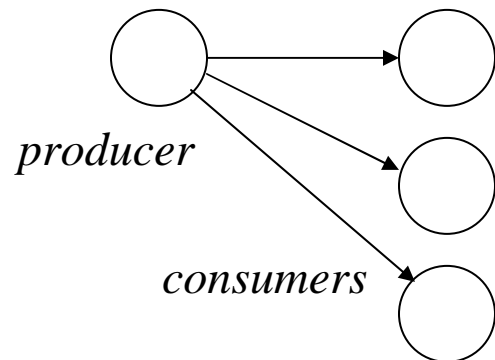
Client-Server

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

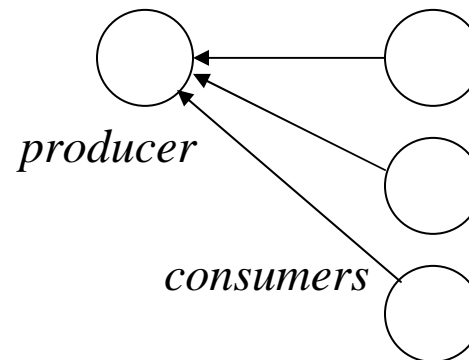


Dátový tok

- Client-server je jednou z realizácií dátového toku



traditional



client-server

→ *smer aktivity*


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**

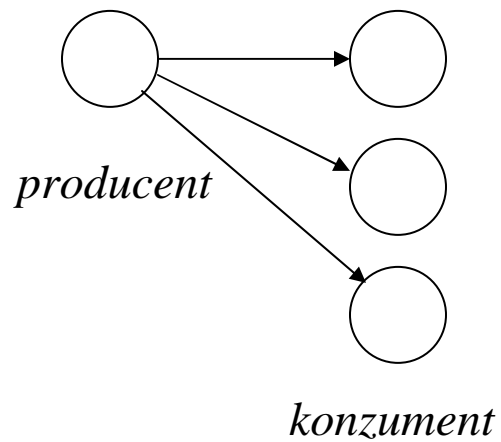
MAS sú špeciálny prípad DS

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

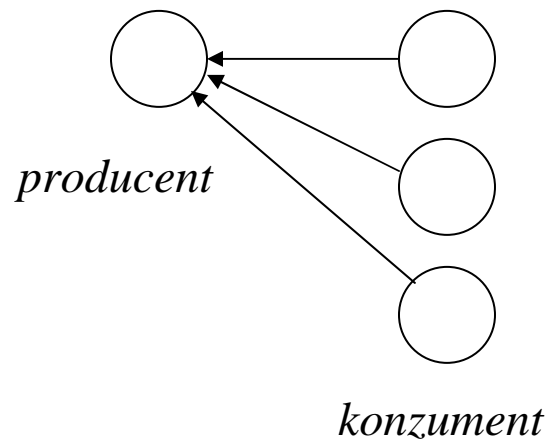
- server neobsahuje žiaden aplikačne závislý kód
- server poskytuje iba komunikačné služby
- dá sa na to teda nazerať ako na snahu o znovupoužitelnosť servera
- client disponuje knižnicou ktorá mu poskytuje komfortné pripojenie na server
- server + knižnica = middleware

Dátové toky

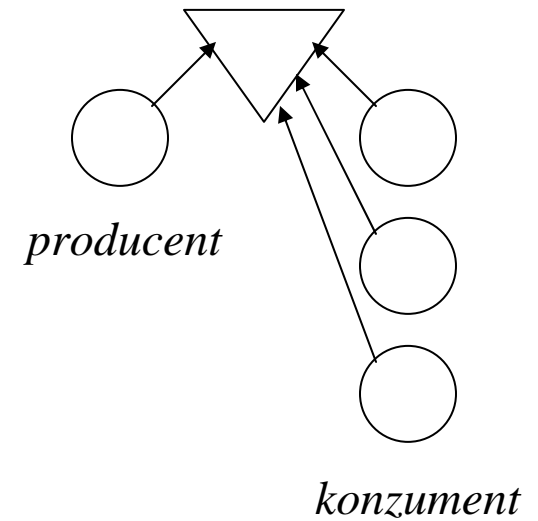
tradičný spôsob



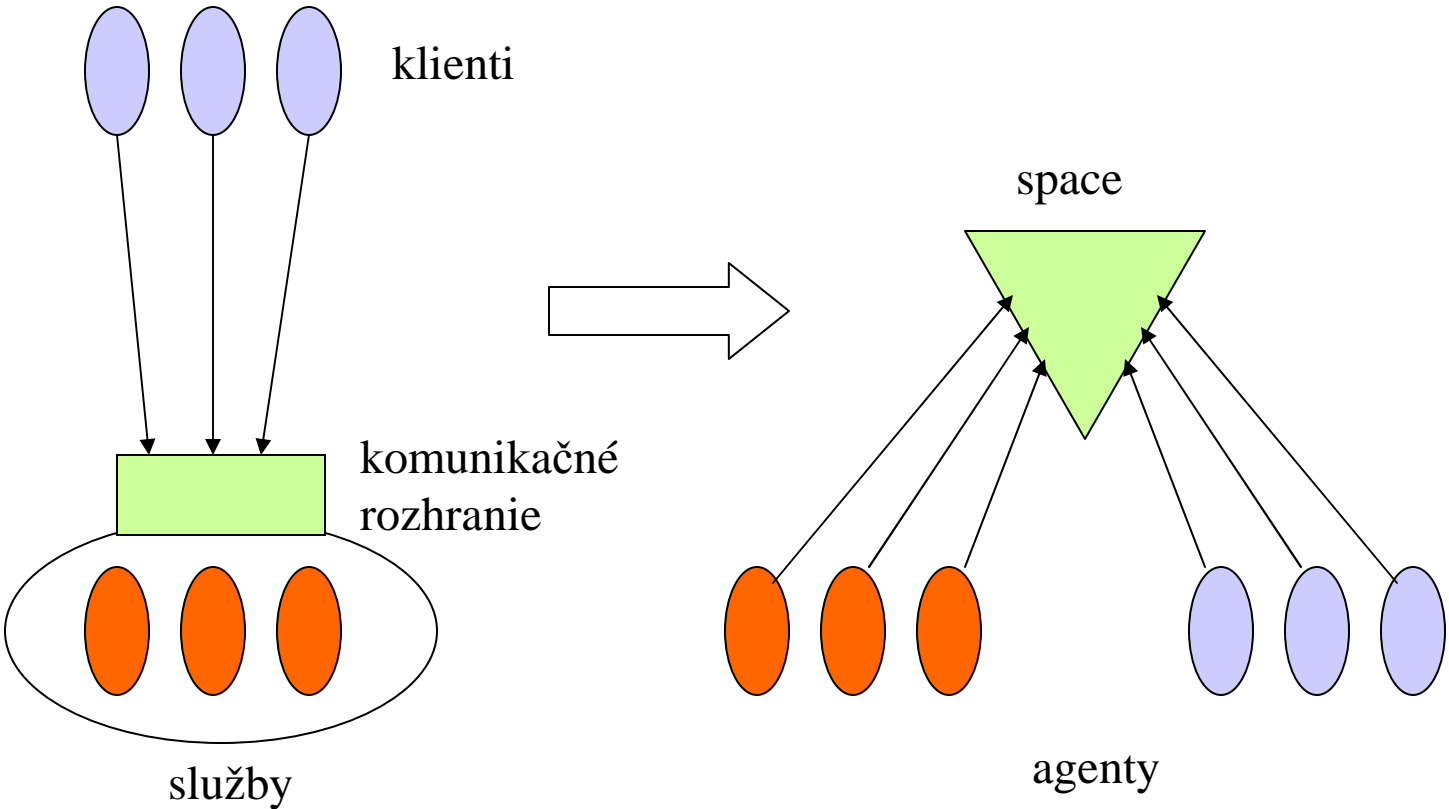
klient-server



agent-space-agent



Transformácia z Client-Server do 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 reprezentačnom 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

Služby poskytované Agentovi

Delia sa na tie, ktoré realizujú

- priamu komunikáciu
- nepriamu komunikáciu

tieto dve skupiny služieb sa líšia v prvom rade rozhraním akým agent dostáva dáta

Služby priamej komunikácie

Základnou službou je

- asynchrónne poslanie správy t.j. SEND
t.j. vyvolanie vhodnej metódy s vhodnými parametrami na strane agenta
táto môže v princípe priamo zavolať aplikačný kód (callback, typický pre aktory) alebo sa uložiť správu do nejakého interného zoznamu odkiaľ si ho vyčíta thread agenta (agentový prístup) – vysvetlíme pri VM. Agent teda potrebuje aj niečo na prijatie, nejaký RECEIVE

Služby nepriamej komunikácie

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

- Základné služby sú

READ

WRITE

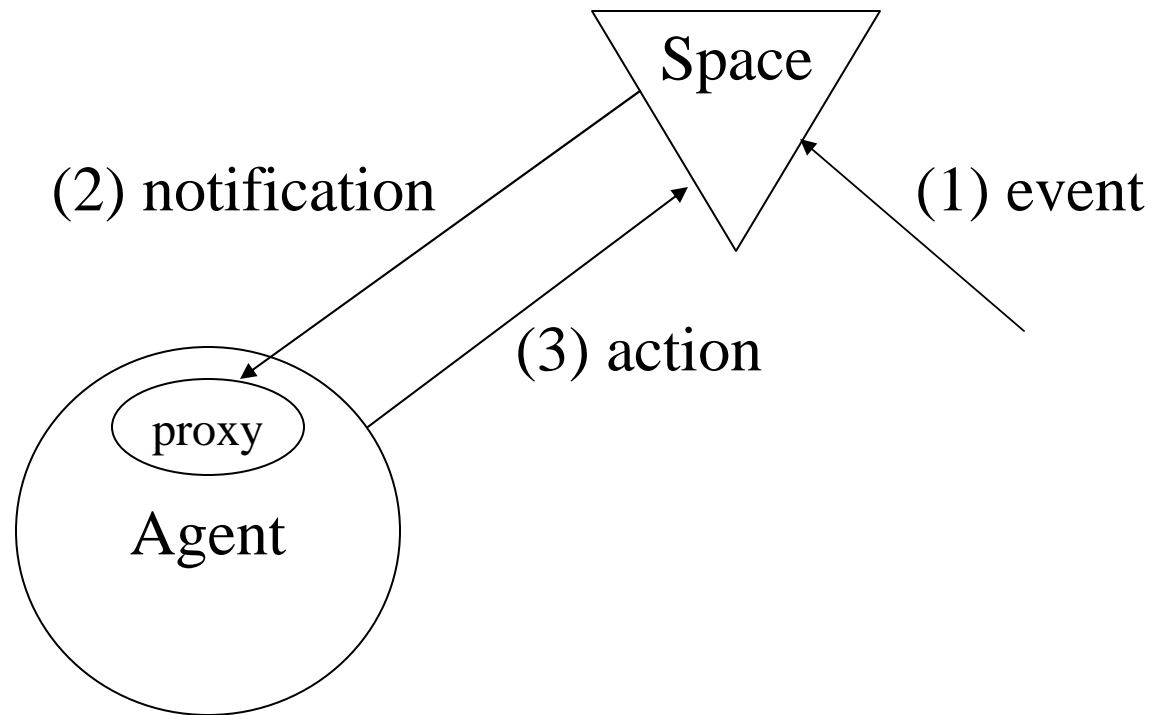
DELETE

- neblokujúce a blokujúce (synchronizácia)

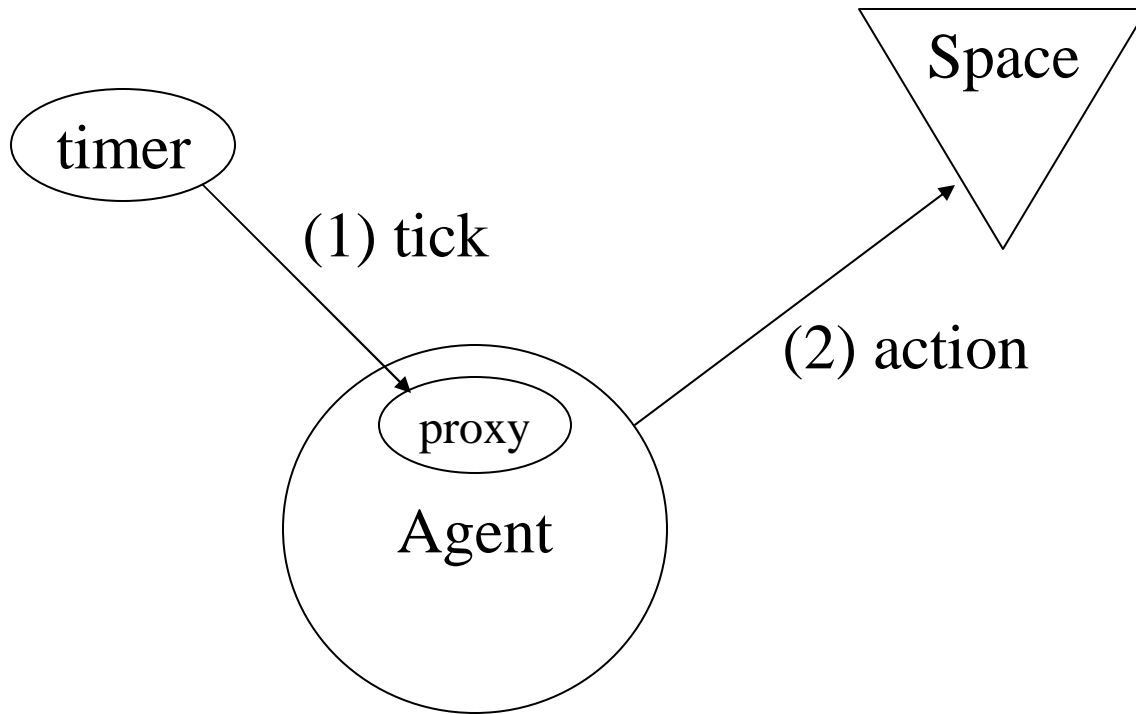
Služby nepriamej komunikácie

- 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



Timer



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 Tuple Space

Dátová štruktúra, ktorá dokáže obsahovať n-tice termov, v princípe LISPOvskej 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

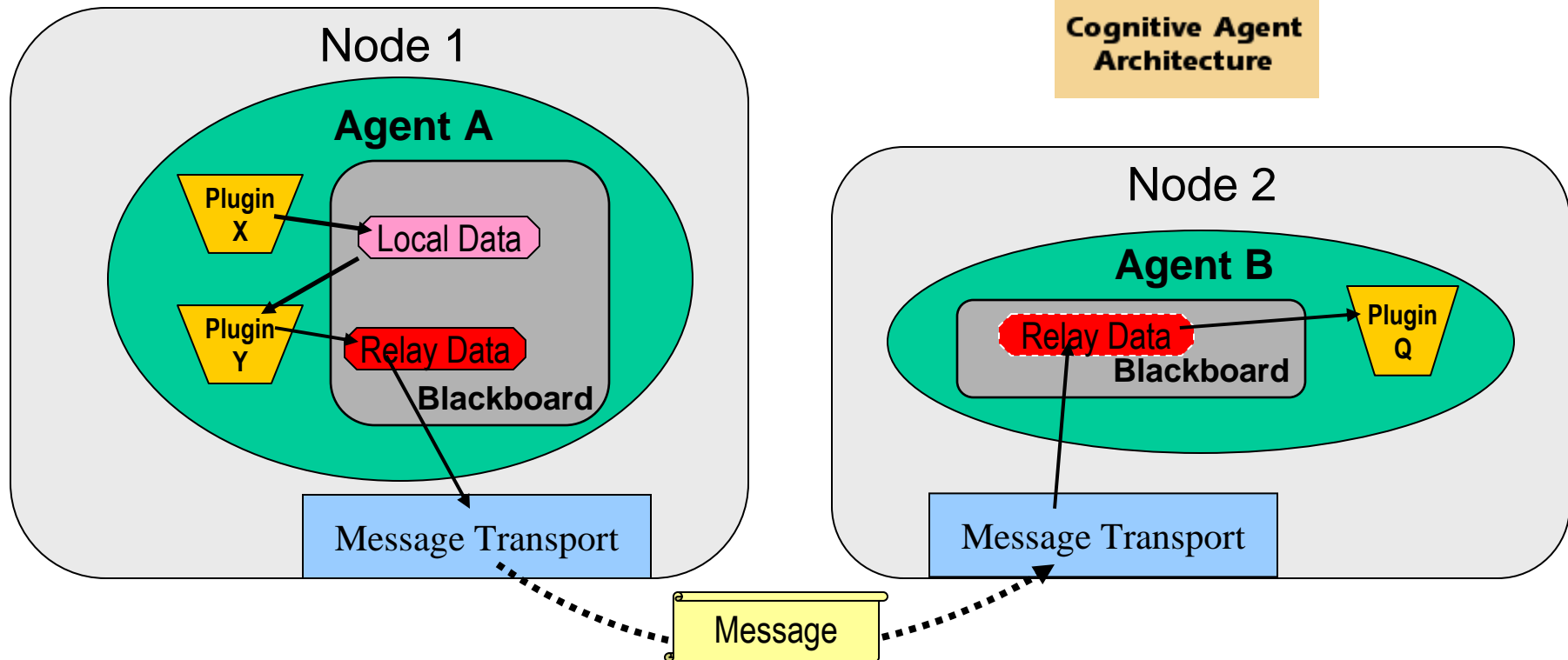
Java Space

```
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
```

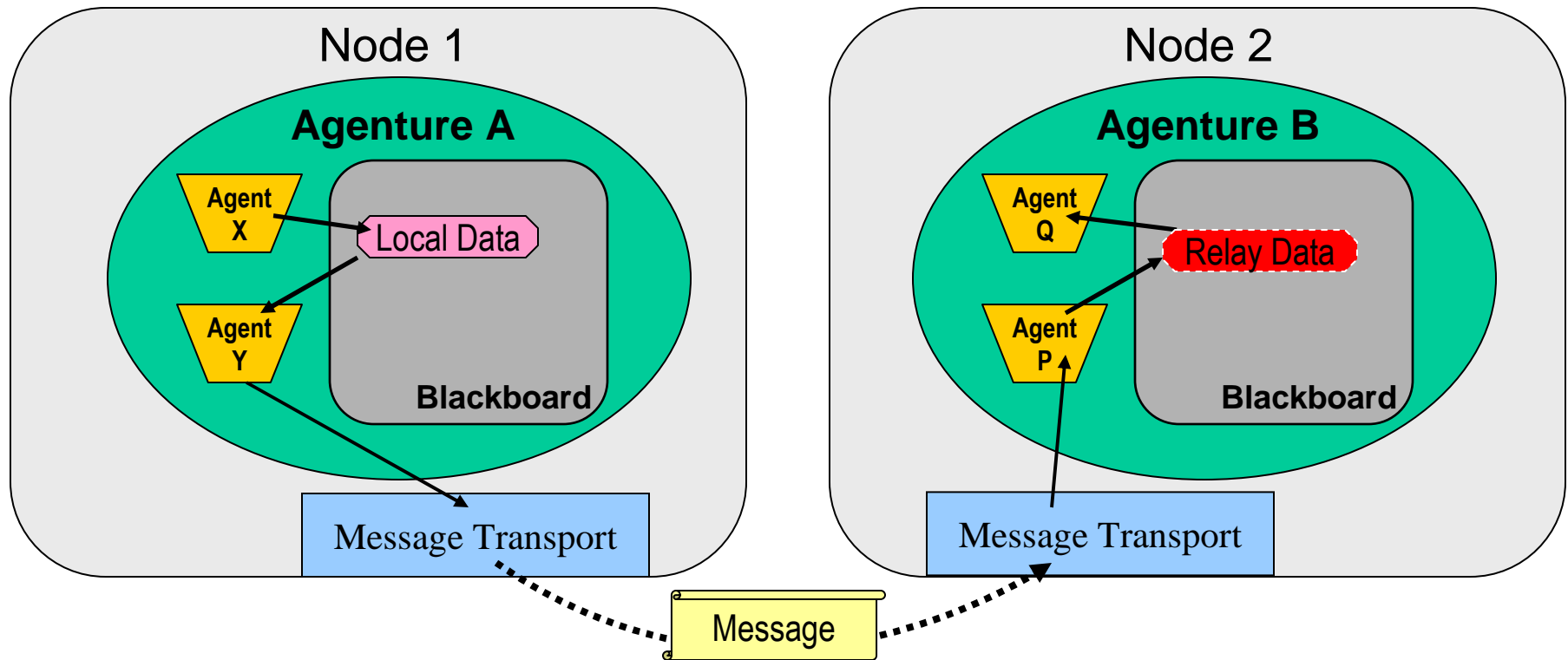
Data leasing (prenájom)

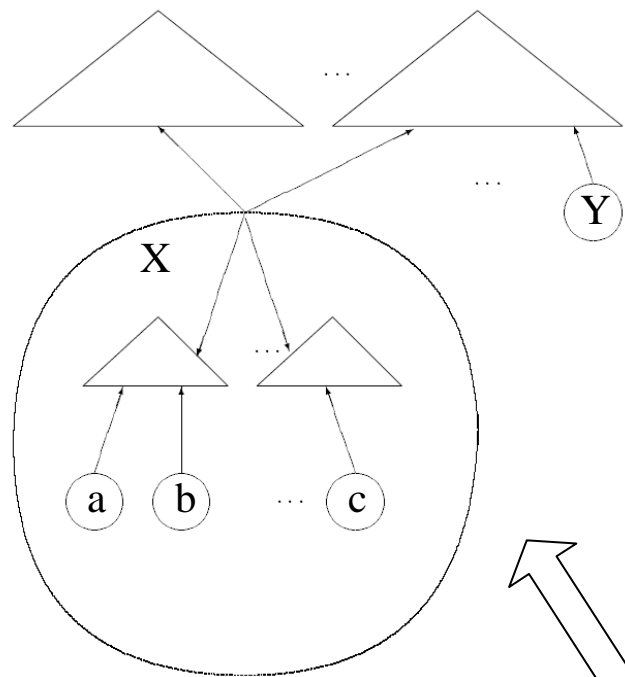
- Java Space prvý krát zaviedol obmedzovanie časovej platnosti dát uložených v space
- space sa tu stará o to, aby po určitom čase boli určité dáta z neho odstránené
- to sa dá len do istej miery pasívne, a lepšie je keď space disponuje časovaným taskom, či na jeho realizáciu určeným vláknom.

Plugins vs. agenty nižšieho rádu



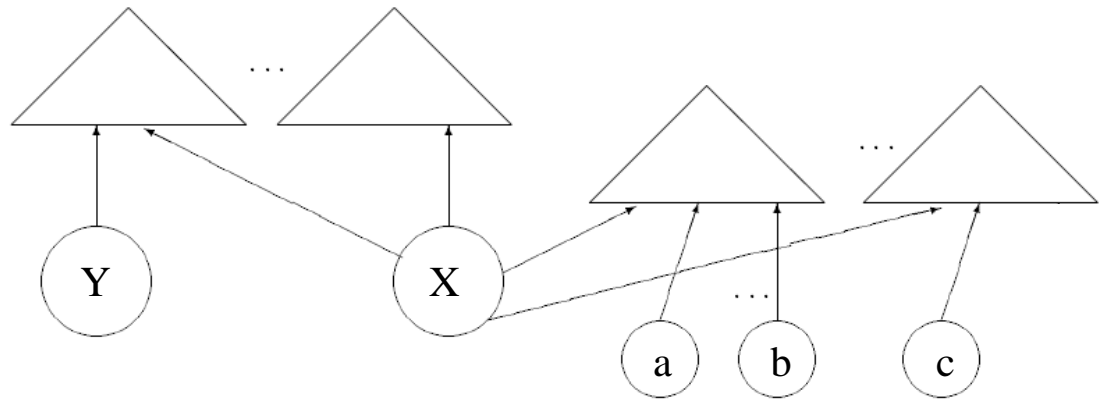
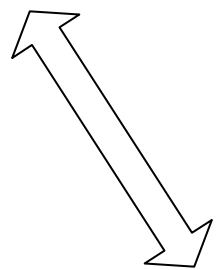
Plugins vs. agenty nižšieho rádu

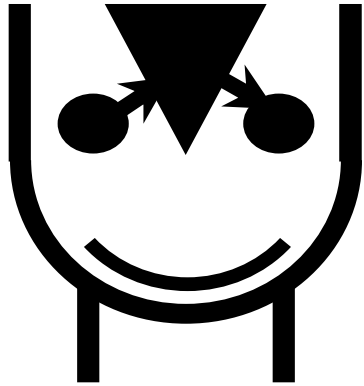




„Kelemenov princíp“

Hierarchia sa dá premeniť na enkapsuláciu a opačne





Agent - Space

- MAS architektúra opierajúca sa výlučne o nepriamu komunikáciu navrhnutá na FMFI UK
- Je založená na „Kelemenovom princípe“, t.j. neexistujú žiadne nižšie jednotky ako agenty (moduly, správania, pluginy a pod.)
- Každý agent má práve jedno vlákno
- (Obsahuje niekoľko technických detailov, ktoré umožňujú používať tzv. subsumpciu – vysvetlíme neskôr)