Agenti Software

Cuprins

1. Agenti Autonomi

- **Definitii (1)**

"... a hardware or (more usually) software-based computer system that enjoys the following properties:

- ***autonomy**: agents operate without the direct intervention of humans or others, and have some kind of control over their actions and internal state;*
- ***social ability**: agents interact with other agents (and possibly humans) via some kind of agent-communication language;*
- ***reactivity**: agents perceive their environment, (which may be the physical world, a user via a graphical user interface, a collection of other agents, the INTERNET, or perhaps all of these combined), and respond in a timely fashion to changes that occur in it;*
- ***pro-activeness**: agents do not simply act in response to their environment, they are able to exhibit goal-directed behavior by taking the initiative."*

[Wooldridge and Jennings]

- **Definitii (2)**

An autonomous agent is a system placed in an environment that *reacts* to this environment and *acts upon* it according to its *goal*.

[Genesereth, Ketchpel]

A software agent is an *autonomous* process capable of *reacting* to, and *initiating changes* in its environment, possibly in *collaboration* with users and other agents.

[Jennings and Woolridge]

Caracteristici (Tanenbaum, van Steen)

Caracteristica agent	Toti agentii	Descriere
Autonom	Da	Actioneaza de sine statator (in anumite limite) in numele unor utilizatori sau a altor agenti
Reactiv	Da	Raspunde la timp schimbarilor din mediul sau: evalueaza evenimentele externe si isi adapteaza comportamentul in conformitate cu aceste schimbari
Proactiv	Da	Initiaza actiuni (inclusiv iau decizii) care afecteaza mediul, in acord cu scopurile asumate
Comunicativ si cooperant	Da	Poate schimba informatii cu utilizatorii si cu alti agenti (abilitati sociale), primesc instructiuni, dau raspunsuri, coopereaza pentru indeplinirea unor scopuri comune
Negociere	Nu	Are abilitatea de a comunica pentru a stabili gradul de cooperare cu alti agenti
Mobil	Nu	Poate migra de la un site la altul
Adaptiv (invata)	Nu	Capabil sa invete. Flexibil. Nu are actiuni fixe, pre-stabilite, ci dinamice, stabilite pe baza experientei anterioare.
Orientat pe scop	Da	Actioneaza pentru un utilizator (program) si satisface cerintele acestuia

Clasificare functionala agenti

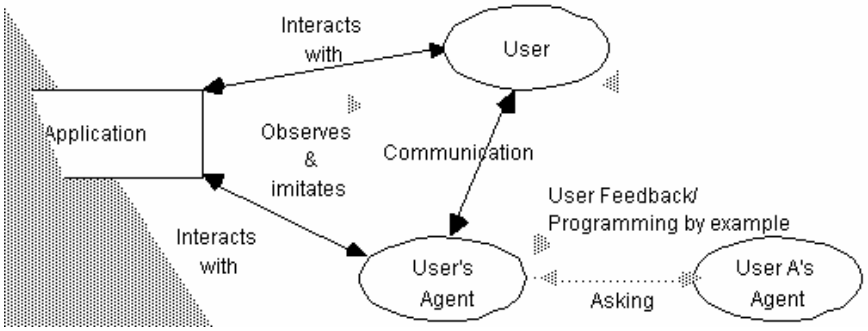
Agenti de Interfata
Ajuta utilizatorul in interactiunea cu un sistem complicat
Agenti Consultanti
Ofera servicii in sistemele de help si diagnosticare
Agenti de Filtrare
Reduc volumul informatiei livrate utilizatorului prin eliminarea informatiilor nerelevante (care un corespund profilului utilizatorului)
Agenti de Regasire
Cauta si regasesc informatii si servesc drept brokeri de informatii si documente
Agenti de Navigare
Memoreaza scurtaturi, pre-incarca info cache, marcheaza pagini de interes
Agenti de Recomandare
Fac recomandari pe baza profilelor disponibile
Agenti de Profilare
Permit construirea unor servicii personalizate profilului utilizatorilor
Agenti de Sistem
Ajuta in gestiunea sistemelor distribuite complexe
Agenti de Monitorizare
Ofera informatii asupra unor evenimente din sistemul monitorizat: actualizari, mutari, stergeri ale unor informatii

2. Agenti si mediile distribuite

In practica, functiile agentilor sunt combinatii ale functiilor elementare discutate anterior. Prezentam cateva dintre cele mai raspnadite categorii de agenti.

Asistenti personali

Asista utilizatorii in activitati de rutina.



Exemplu: Personal Assistants (MIT Media Lab)

- **Folositi pentru**:
  - planificare intalniri,
  - manipulare e-mail,
  - filtrare stiri electronice,
  - selectie carti
  - in general, activitati de rutina ale utilizatorilor
- **Caracteristica principala** - capacitatea de a invata prin
  - observarea utilizatorilor si imitarea lor
  - reactii de la utilizatori (feedback)
  - instructiuni explicite de la utilizatori
  - sfaturi primite de la alti agenti

Agenti de e-Learning (USC / ISI)

- Suporta interactiunea dintre utilizatori si sistemul de e-learning
- **Adele** consta dintr-un **agent pedagogic** si o reprezentare animata a unei persoane
- Functionalitati de interfata
  - profilare - construiesc servicii personalizate conform profilului utilizatorilor
  - monitorizare progres studenti
  - feedback, hint-uri si rationamente pentru ghidarea actiunilor studentului
  - recomandare - specifica referinte la materiale relevante
  - evaluare activitate student
- Functionalitati de informare (Internet)
  - cautare si regasire – de informatii si actiuni de brokeri de informatii si documente
  - agregare – informatii din diferite surse
- Folosit in sisteme educationale medicale pentru diagnoza si ingrijirea traumelor.

Agenti de proiectare automata

Conceputi pentru rezolvare distribuita de probleme. Sistemul foloseste

- agenti cu competente diferite
- pentru probleme mari, pentru care un singur agent nu este suficient
- sau pentru prelucrare informatiilor provenind din surse distribuite.

Exemplu: Automated Design Agents (Univ. of Michigan)

- ACDS (Automated Catalog-Design Service)
  - Fac proiectarea configuratiilor, inclusiv selectia partilor din catalog
  - Sistemul este realizat ca retea de agenti de diferite feluri
    - Agenti de catalog
      - fiecare agent reprezinta o componenta care poate decide sa participe la un proiect particular
    - Agent de sistem
      - Traduce proiectarea la nivel inalt in reprezentarea inteleasa de retea si o difuzeaza agentilor relevanti
    - Agenti de constrangeri
      - Asigura respectarea constrangerilor de proiectare

Agenti de Sistem

Ajuta in gestiunea sistemelor distribuite complexe, realizand:

- Executia paralela a taskurilor
- Imbunatatirea capabilitatilor de timp real
- Monitorizarea evenimentelor
- Cresterea robustetii
- Echilibrarea incarcarii (load balancing)

Exemplu: Rutarea folosind agenti

Model

- furnicile lasa o urma de feromoni in drumul spre hrana
- celelalte urmeaza calea
- comportament simplu bazat pe reactia la mediu si la partenere
- actionand in grup, pot realiza sarcini complicate, in maniera distribuita

AntNetwork routing

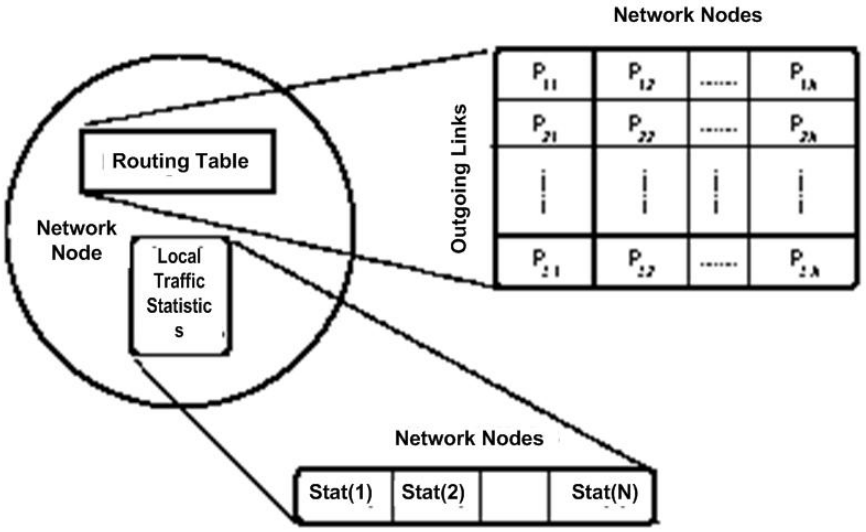
- In fiecare nod al retelei, agenti mobili sunt lansati periodic spre destinatii alese aleator
- Agentii migreaza concurent cu datele
- Agentii actioneaza independent dar comunica prin informatii lasate in nodurile vizitate
- Fiecare agent cauta calea cea mai scurta intre sursa si destinatie
- Se folosesc

- Tabele de rutare indexate prin destinatie * vecin, cu intrari probabiliste
  - Pdn = probabilitatea de alegere a lui n ca nod urmator pentru destinatia d
- Model statistic de retea
  - medii si variante ale timpilor de calatorie inregistrati de agenti corespunzătoare unor observații facute pe durata unei fereastre de timp (glisante).

Urmatoarea legatura este aleasa in functie de probabilitatea din tabele si lungimea cozii de iesire corespunzatoare:

$$p'(j) = (p(j)+ß*lj) / (1+ß*(Nk-1))$$

p(j) – prob de a selecta j ca nod urmator
lj factor euristic de corectie calculat pe baza lungimii in biti a cozii catre destinatia j (normalizata la un interval unitar)
Nk- numarul de legaturi de la nodul curent
ß – pondereaza lungimea cozii in raport cu informatia de rutare p(j) din tabel.



AntNetwork routing (2)

- In fiecare nod, agentul salveaza local
  - Identificatorul nodului precedent
  - Amprenta de timp curent
  - Timpul scurs de la lansare
  - Incarcarea nodului curent
- Odata ajunsi la destinatie, agentii sunt transmisi inapoi pe aceeasi cale dar in sens opus
- In fiecare nod agentii modifica modelul statistic de retea si continutul tabelei de rutare

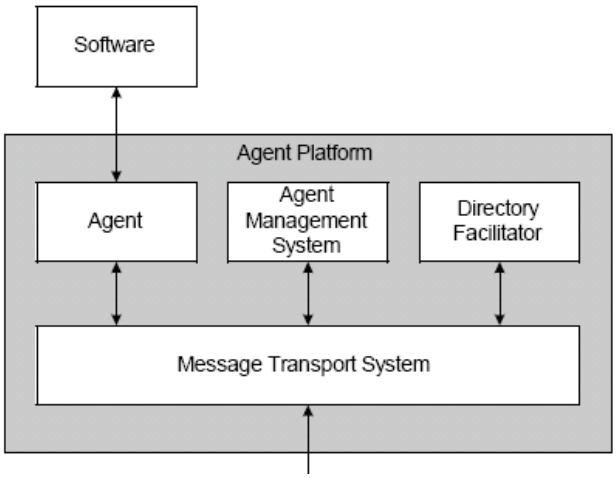
  IF (nod i a fost in calea furnicii la ducere)
  THEN $p(i) = p(i) + r\{1 – p(i)\}$
  ELSE $p(i) = p(i) + r\,P(i)$

  Miezul algoritmului este alegerea optima a lui r, descrisa in detaliu de autori in lucrarea lor [Yang 2002]. Fara a intra in detalii, precizam ca:
  r tine seama de calitatea caii spre destinatie (vecinul mai bun capata o probabilitate mai mare) pe baza info stocate la ducere
  P(i) ajusteaza corespunzator probabilitatile pentru ceilalti vecini astfel ca suma probabilitatilor pentru toti vecinii si o singura destinatie sa fie egala cu 1
- Ajunsi la sursa, agentii sunt distrusi

3. Platforme de agenti - Arhitectura FIPA

Servicii de baza

FIPA - Foundation for Intelligent Physical Agents este o organizatie internationala dedicata promovarii industriei agentilor inteligenti prin dezvoltarea specificatiilor care suporta interoperabilitatea intre agenti si aplicatii bazate pe agenti. Specificatiile se refera la gestiunea agentilor, limbajele de comunicare intre agenti, formatul mesajelor de comunicare etc. Gestiunea agentilor se refera la cadrul normativ in care agentii FIPA exista si opereaza, stabilind **modelul logic de referinta** pentru crearea, inregistrarea, localizarea, comunicarea, migrarea si retragerea agentilor. Modelul include **entitatile logice** prezentate in figura, fiecare desemnand un serviciu.



Entitatea de baza este **agent**-ul care implementeaza functionalitatea autonoma a aplicatiei. Agentii comunica folosind un limbaj de comunicare inter-agenti – ACL (Agent Communication Language). Un agent are (cel putin) un proprietar si o identitate - Agent Identifier (AID) care-l distinge neambiguu de alti agenti. Un agent poate fi inregistrat la un numar de adrese de transport la care poate fi contactat.

AID este o colectie (extensibila) de perechi <parametru: valoare> in care pot apare urmatorii parametri:

- name – forma este nume_agent@adresa_platforma
- addresses – lista de adrese de transport la care mesajul poate fi livrat; forma este cea a unui URL
- resolvers – lista de adrese de servicii de rezolvare a numelor prin functia *search*.

Exemplu de AID:

```
(agent-identifier
  :name agent-b@bar.com
  :resolvers (sequence
      (agent-identifier
        :name ams@foo.com
        :addresses (sequence iiop://foo.com/acc))))
```

Pentru a afla adresa agentului `agent-b`, se poate trimite o cerere *search* catre `ams@foo.com` aflat la adresa `iiop://foo.com/acc`.

Sistemul de management al agentilor AMS - **Agent Management System** controleaza accesul la si utilizarea unei platforme de agenti. Exista un singur AMS pe platforma. Acesta pastreaza un director de AID-uri in care sunt inscrise adresele de transport pentru agentii inregistrati in platforma gestionata (ofera un serviciu de pagini albe pentru agenti). Functiile suportate sunt realizate la platforma de referinta (home) a agentului si includ:

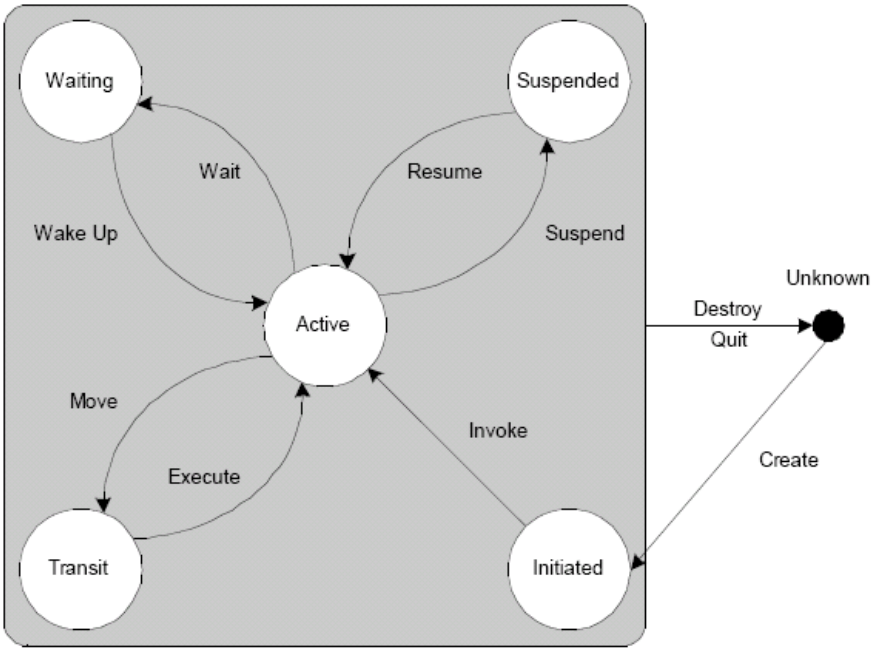
- `register`

- `deregister`
- `modify`
- `search`
- `get-description`

In plus, AMS poate instrui PA sa execute urmatoarele operatii:

- `Suspend agent,`
- `Terminate agent,`
- `Create agent,`
- `Resume agent execution,`
- `Invoke agent,`
- `Execute agent,`
- `Resource management`

in acord cu ciclul de viata al agentilor prezentat in figura urmatoare.



Serviciul de directoare **Directory Facilitator (DF)** face posibila regasirea agentilor dupa atribute (serviciu de pagini aurii). Agentii pot inregistra serviciile oferite sau pot interoga DF despre serviciile oferite de alti agenti. O intrare in Directory contine o colectie de perechi <cheie-valoare> incluzand:

- numele agentului
- adresa de transport
- servicii oferite
- cost asociat cu utilizarea agentului
- restrictii de utilizare
- etc.

O platforma poate avea mai multe DF-uri care ofera servicii agregate. Un DF are urmatoarele servicii:

- register – agentul publica un serviciu
- deregister – agentul anuleaza publicarea
- modify – agentul modifica inregistrarea
- search – un agent cauta info despre alti agenti
- subscribe – un agent subscrie pentru notificari.

Serviciul de transport MTS - **Message Transport Service** este mijlocul implicit de comunicare intre agenti pe diferite platforme.

O platforma de agenti AP - **Agent Platform** ofera infrastructura fizica in care agentii ruleaza: masini, sistem de operare, software de suport, componentele de management (DF, AMS, MTS) si agenti.
Obs.: FIPA nu standardizeaza implementarea unei platforme de agenti sau comunicarea in cadrul unei platforme, ci doar comunicarea intre agenti dintr-o platforma si agenti din afara platformei.
Software descrie programele executabile accesibile printr-un agent (pentru servicii noi, protocoale noi de comunicare, protocoale de securitate, instrumente de support al migrarii etc.)

4. Limbaje de comunicare

- Abordari
  - Procedurale
    - Interschimb de directive procedurale (de la comenzi individuale la programe intregi) care sunt executate la receptor
  - Declarative
    - Interschimb de propozitii declarative (definitii, asertiuni, presupuneri)
    - Ex.
      - KQML - Knowledge Query and Manipulation Language
      - FIPA ACL (Foundation for Intelligent Physical Agents) -(Agent Communication Language)

Mesaje ACL (Tanenbaum)

Un mesaj FIPA ACL contine unul sau mai multe campuri, denumite **parametri**. Parametrul obligatoriu (denumit **performative**) arata scopul mesajului. Pe langa el, in mesaj sunt de regula prezenti parametri referitori la transmitator, receptor si continut. Un exemplu este urmatorul:

Camp	**Valoare**
Scop	INFORM
Transmitator	Max@http://fanclub-beatrix.royalty-spotters.nl:7239
Receptor	Elke@iiop://royalty-watcher.uk:5623
Limbaj	Prolog
Ontologie	Genealogy
Continut	female(beatrix),parent(beatrix,juliana,bernhard)

Este un mesaj informativ trimis de
agentul Max
aflat pe platforma fanclub-beatrix.royalty-spotters.nl
cu care se poate comunica prin http la portul 7239
catre agentul Elke
aflat pe platforma royalty-watcher.uk
cu care se poate comunica prin IIOP la portul 5623
limbajul mesajului este Prolog
si foloseste ontologia Genealogy
continutul spune ca beatrix este numele unei femei
ai carei parinti sunt juliana si bernhard

Tabelul urmator prezinta succint lista tuturor parametrilor FIPA ACL.

Parametru	**Categoria Parametrilor**	**Semnificatie**
performative	scopul mesajului	
sender	participant in comunicare	identitatea transmitatorului
receiver	participant in comunicare	identitatea receptorului
reply-to	participant in comunicare	agentul caruia i se va transmite mesajul (diferit de sender)
content	continutul mesajului	
language	descrierea continutului	limbajul in care este

		exprimat continutul
encoding	descrierea continutului	schema de codificare a mesajului (bazata pe XML, text etc.)
ontology	descrierea continutului	ontologia folosita pentru a da inteles simbolurilor din continut
protocol	controlul conversatiei	protocolul de interactiune folosit in conversatie
conversation-id	controlul conversatiei	identifica conversatia careia ii apartine mesajul
reply-with	controlul conversatiei	introduce o expresie care va fi folosita in raspunsul la aceasta actiune
in-reply-to	controlul conversatiei	expresia identifica mesajul la care se raspunde
reply-by	controlul conversatiei	raspunsul este asteptat nu mai tarziu de aceasta data / timp

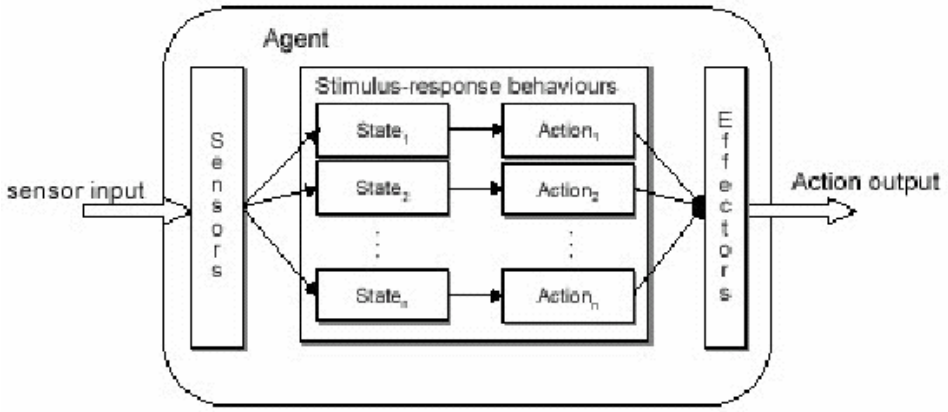
FIPA ACL

Tabelul urmator prezinta cateva din scopurile posibile ale mesajelor FIPA ACL.

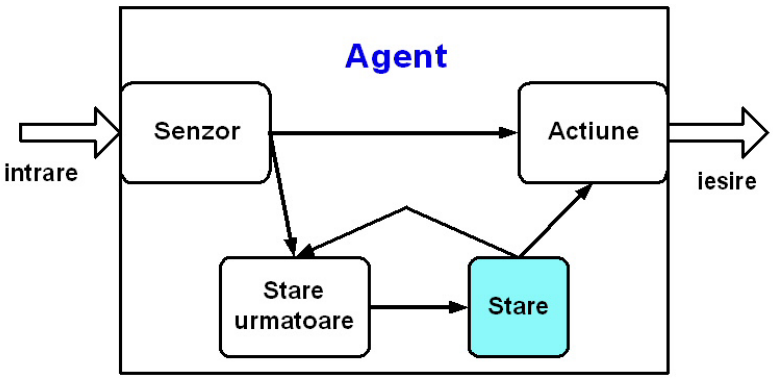
Scop mesaj	Descriere	Continut
INFORM	Informeaza ca o propozitie este adevarata	Propozitie
QUERY-IF	Query daca o propozitie este adevarata	Propozitie
QUERY-REF	Query pentru unobiect dat	Expresie
CFP	Cere o propunere	Specific propunerii
PROPOSE	Fa o propunere	Propunere
ACCEPT-PROPOSAL	Spune ca o propunere este acceptata	ID propunere
REJECT-PROPOSAL	Spune ca o propunere este rejectata	ID propunere
REQUEST	Cere executia unei actiuni	Specificare actiune
SUBSCRIBE	Subscrie la o sursa de informatie	Referinta sursa

Arhitectura agentilor reactivi

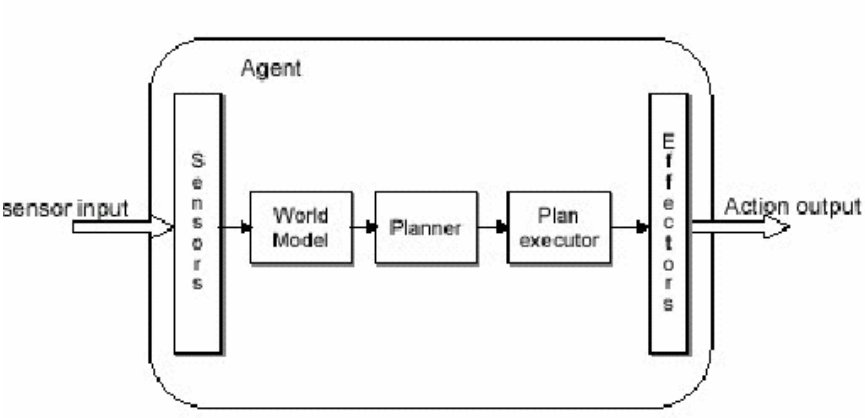
Agentii reactivi au o cuplare stransa intre perceptie si actiuni.



Agenti cu stare
Decizia agentilor este influentata de istorie



Agenti deductivi
Au o reprezentare simbolica a mediului (baza de cunostinte)
Sesizeaza starea lui curenta
Stabilesc un scop (ce stare doresc sa atinga)
Alcatuiesc un plan de actiuni
Il executa



5. Mobilitatea codului
Proces complicat, justificat doar daca duce la imbunatatirea performantelor aplicatiei. Situatii tipice sunt:
- echilibrarea incarcarii resurselor unui sistem distribuit
- exploatarea paralelismului; de exemplu, cautarea realizata in paralel de mai multi agenti care migreaza la diferite servere
- reducerea comunicarii prin transferul codului (presupus de dimensiuni reduse) in locul datelor (presupuse de mare volum); un exemplu este migrarea codului de la server la client unde executia inseamna o interactiune puternica cu utilizatorul.

Ce se migreaza? Pentru a intelege modelele de mobilitate, reamintim ca un proces consta din trei segmente:
- Segmentul de Cod
- Segmentul de Resurse
  - referinte la fisiere, imprimante, echipamente, alte procese, capete conexiune, etc.

- Segment de executie
  - date private, stiva, contor program

Modele de mobilitate

- Slaba - **Weak** - migreaza segmentul de cod cu date initiale (ex. Java applets). Executia codului migrat poate avea loc:
  - in proces tinta, de exemplu executia unui applet in spatiul de adrese al browser-ului;
  - in proces separat creat de sistemul de operare.

- Puternica - **Strong** - migreaza si segmentul de executie. In acest caz, un proces poate fi oprit din executie si mutat pe o alta masina, unde executia continua din locul de suspendare. Se face diferenta intre doua solutii posibile:
  - Migrarea procesului (corespunde descrierii anterioare);
  - Clonarea procesului, caz in care clona ruleaza in paralel cu procesul parinte.

Migrarea poate fi:
- Initiata de transmitator - Sender initiated
  - Ex. trimite un agent la un server de baze de date pentru a face o interogare
- Initiata de receptor - Receiver initiated
  - Java applets

Migrarea resurselor locale

Migrarea segmentului de resurse este mai complicata decat migrarea celorlalte segmente. De exemplu, referinta la un port TCP (inclusa in segmentul de resurse) trebuie tratata altfel decat codul procesului: migrarea inseamna inchiderea portului din sistemul de origine si deschiderea unuia nou in sistemul tinta.
Migrarea resurselor este determinata de doi factori: legarea procesului la resursa si legarea resursei la masina gazda.
Sunt trei moduri de legare (binding) proces – resursa:

- Prin identificator – procesul este legat la resursa specificata de identificator; este forma de legare cea mai puternica, procesul avand nevoie de acea resursa specifica si nu de alta.
- Prin valoare – procesul are nevoie de valoarea resursei si este mutumit cu orice resursa care poate oferi aceeasi valoare; un exemplu de astfel de resursa este o biblioteeca standard.
- Prin tip – procesul are nevoie de o resursa de un anumit tip, de exemplu o imprimanta sau un monitor; orice resursa de acel tip este satisfacatoare.

Legare resursa – masina are trei forme:

- Resursa neatasata – exemplul tipic este un fisier; poate fi migrat.
- Resursa atasata – exemplul este o baza de date; migrarea se poate face dar cu un cost ridicat.
- Resursa fixa – de exemplu o resursa locala unei anumite masini specifice; nu se poate face migrarea.

Actiuni executate la migrare

In functie de aceste tipuri de legari, actiunile care trebuie realizate la migrare sunt cele prezentate in tabelul urmator.

Legare process la resursa	Legare resursa la masina: Neatasata	Atasata	Fixa
prin identificator	**MV (or GR)**	**GR (or MV)**	**GR**
prin valoare	**CP (or MV,GR)**	**GR (or CP)**	**GR**
prin tip	**RB (or MV, CP)**	**RB (or GR, CP)**	**RB (or GR)**

Notatiile folosite sunt urmatoarele:
GR – stabileste Referinta Globala (Global Reference)
MV – muta resursa (MoVe)
CP – copiaza valoarea resursei (CoPy)
RB – re-lega (ReBind) proces la resursa locala disponibila

Agenti mobili: D'Agents

D'Agents este o platforma middleware construita in jurul conceptului de agenti mobili.
Codul unei aplicatii poate fi scris in orice limbaj interpretat suportat de masina tinta. Sunt suportate Tcl, Java si Scheme. Un agent este executat de un proces care ruleaza interpretorul pentru limbajul in care agentul este scris.

Sunt suportate trei tipuri de mobilitati:
- Mobilitate slaba initiata de transmitator
- Mobilitate puternica prin migrarea procesului
- Mobilitate puternica prin clonare.

Despre Tcl – Tool Control Language

- limbaj interpretat, extensibil
  - are un set de primitive implementate in C/C++
  - peste care sunt construite alte comenzi
  - pentru agenti:
    - agent_begin, agent_end – inregistreaza / termina un agent
    - agent_submit, agent_jump - mobilitate
    - agent_send, agent_receive – schimb mesaje
    - agent_meet, agent_accept – initiaza / accepta o conexiune directa
- programele Tcl (scripts)
  - sunt textuale
  - contin structuri de control (secvente, selectii, iteratii)
  - contin variabile simple si structurate (tablouri, liste)
  - pot apela alte programe (proceduri)
  - pot fi communicate prin retea si executate in masini tinta
- Tk
  - toolkit X Window pentru interfete grafice
  - are facilitati de comunicare intre procese prin interschimb de scripturi Tcl

Mobilitate slaba (ex. in Tcl)

Exemplificare agent_submit – un agent care executa procedura factorial este migrat pe o masina tinta si pus in executie pentru o anumita valoare a parametrului. Rezultatul calculului este receptionat cu agent_recieve.

```
Proc factorial n {
    if ( $n <= 1 ) { return 1; }                    # fact(1) = 1
```

```
    expr $n * [ factorial [ expr $n – 1 ] ]     # fact (n) = n * fact(n-1)
}

set number …                                    # ce factorial sa calculeze
set machine …                                   # identific masina tinta

agent_submit $machine –procs factorial –vars number –script {factorial $number }

agent_receive …                                         # receptioneaza rezultat
```

agent_submit =
Scriptul factorial $number este trimis tintei $machine cu descrierea procedurii factorial si valoarea initiala a variabilei number.

Observatii.
Tcl pastreaza datele ca siruri de caractere.
$n este o substitutie de variabila – la executie, $x va fi inlocuit cu continutul variabilei x
[expr $n-1] este o substitutie de comanda – va fi inlocuit cu rezultatul executiei comenzii expr $n-1 adica cu valoarea lui $n - 1.

Mobilitate puternica prin migrare

Un agent viziteaza masinile specificate intr-o lista si afla utilizatorii logati la ele.

```
proc all_users machines {
    set list “”                         # creaza o lista initial goala
    foreach m $machines {               # pentru toate gazdele in setul masini
        agent_jump $m                   # salt la fiecare host
        set users [exec who]            # executa comanda who
        append list $users              # adauga rezultat la lista
    }
    return $list                        # returneaza lista completa
}

set machines …          # initializeaza set de masini la care se migreaza
set this_machine …      # setat la host care porneste agentul

# creaza un agent migrator prin transmiterea script la this_machine, de unde agentul va migra la
    toate celelalte masini din $machines

agent_submit $this_machine –procs all_users –vars machines \
                        -script {all_users $machines}
agent_receive …                         # receptie rezultat
```

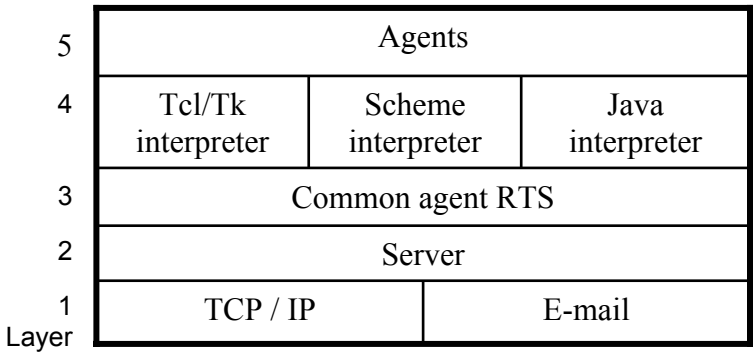
Implementare

Layer 1 - Interfata comuna cu servicii de comunicare (TCP, e-mail si altele)
Layer 2 - Management si autentificare agenti; management comunicare
Asigneaza un identificator unic agentului: <adresa server, id local unic>
Layer 3 - Suport model agenti – partea centrala a platformei de agenti
- start / stop, comunicare inter-agent, migrare

Layer 4 – Interpretare
- Mai include module pentru: securitate, interfata cu nivel inferior (nivel 3), captare stare (pentru mobilitate puternica)

Layer 5 - Agenti

- fiecare agent executat de un proces separat



- Ex. La migrare
  - Serverul creaza proces care executa intrepretorul
  - Procesul primeste starea de la care se continua agentul

Captarea si transmiterea starii

Starea unui agent, prezentata in tabelul urmator, consta din patru tabele pentru definitii de variabile globale si scripturi si din doua stive pentru starea executiei.

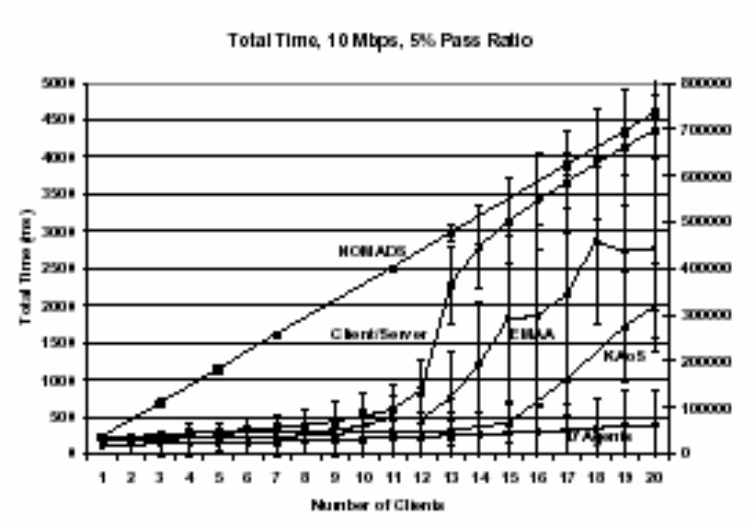
Stare Agent	**Descriere**
Variabile globale interpretor	Variabile necesare interpretorului unui agent e.g. perechi (eveniment – handler)
Variabile globale sistem	Coduri de raspuns, coduri de erori, mesaje de erori, etc.
Variabile globale program	Variabile globale definite de utilizatori in program
Definitii proceduri	Definitii de script-uri de executat de un agent
Stiva de comenzi	Stiva de comenzi aflate in executie O inregistrare contine valorile parametrilor, pointer-ul la procedura care implementeaza comanda etc.
Stiva de frame-uri de apel de procedura (call frames)	Stiva inregistrarilor de activare (una pentru fiecare comanda de apel de procedura aflata in stiva de comenzi) Un frame de apel contine un tabel de variabile locale procedurii, valori si nume ale parametrilor cu care procedura a fost apelata si o referinta la comanda de apel asociata.

La migrare (agent_jump):
Starea este impachetata (captureState) si transmisa masinii tinta;
Procesul creat la tinta reface starea (restoreState) si continua cu comanda din varful stivei de comenzi.

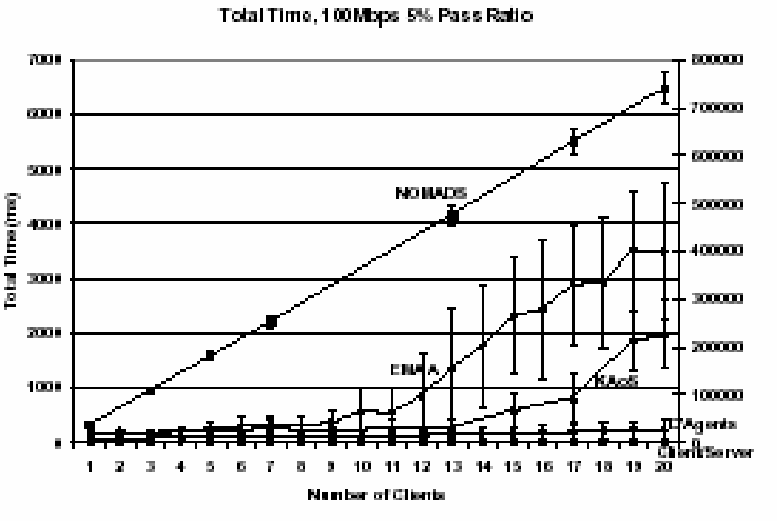
6. Performantele agentilor - Aplicatie de regasire distribuita de informatii

- Taskul filtreaza rezultatele unei interogari simple pe o colectie de documente
- Abordarea Agenti:
  - Agentii filtreaza documentele la server si apoi transfera rezultatele
  - Scrisa in Java
- Abordarea Client-server:
  - Descarca toate documentele rezultate din interogare si le filtreaza pe masina client
  - Scrisa in C++

- Platforme evaluate
  - D'Agents (Darmouth College)
  - EMAA (Lockheed-Martin Advanced Techology Laboratory)
  - KAoS (Boeing and Univ West Florida)
  - NOMADS (Univ West Florida)
  - Comparat cu abprdarea client-server
- Configuratii: mai multi clienti, un server
- Mai multe criterii, din care
  - Timp total interogare



Retea de 10 Mbps (rezultate similare la 1 Mbps)
Performante mai bune la sistemul de agenti
Client server este limitat de banda
Axa Y din dreapta este pentru NOMADS (foarte ineficient).



100 Mbps network
Client server mai bun
- Banda suficienta

- Procesarea distribuita clientilor
- Cod C++ mai eficient

Agenti mai lenti

- Executie pe un singur server
- Cod Java ineficient

7. Protectie agenti mobili

- Exemplu: agent mobil cauta un bilet de avion ieftin; pentru asta migreaza prin diferite situri ale agentiilor de voiaj.
- Atacuri posibile:
  - furt informatie credit card
  - modificare agent pentru plata unei sume mai mari
  - ocolire competitori mai ieftini
  - furt info de la proprietar, la intoarcere
- Protectie totala imposibila
- Alternative: detectie modificari (Ex: Ajanta)
  - stare read-only (semnata de proprietar) – proprietarul face un rezumat al mesajului si il cripteaza cu cheia sa privata. Tinta poate detecta modificarea starii prin calculul rezumatului si compararea lui cu rezultatul decriptarii informatiei primite
  - log-uri append-only
  - dezvaluire selectiva a informatiei – organizare a datelor in mai multe campuri, fiecare criptat cu cheia publica a tintei careia ii este destinat

Log-uri append-only

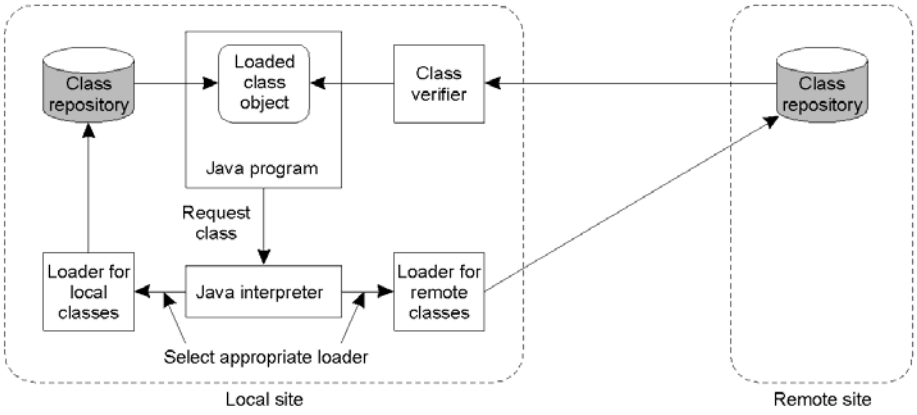
Un agent colecteaza informatii de la tintele vizitate; tintele nu pot modifica informatia acumulata deja fara ca proprietarul sa detecteze modificarea. Protocolul are urmatoarele faze:

- Initial Cinit = K^{+}owner(N) – proprietarul alege N si il cripteaza cu cheia sa publica
- Server S – primeste Cold, adauga X semnat de S si calculeaza Cnew folosind cheia publica a proprietarului
  - Cnew = K^{+}owner (Cold, sig(S,X),S)
- Inapoi la proprietar – primeste C si il decripteaza cu cheia sa privata obtinand Cnext si sig(X,S), S.
  - K^{-}owner(C) => Cnext si sig(X,S), S.
- Proprietarul verifica semnatura si foloseste X.
- Proprietarul repeta operatia pentru Cnext

Protejarea Tintei (Java sandbox)

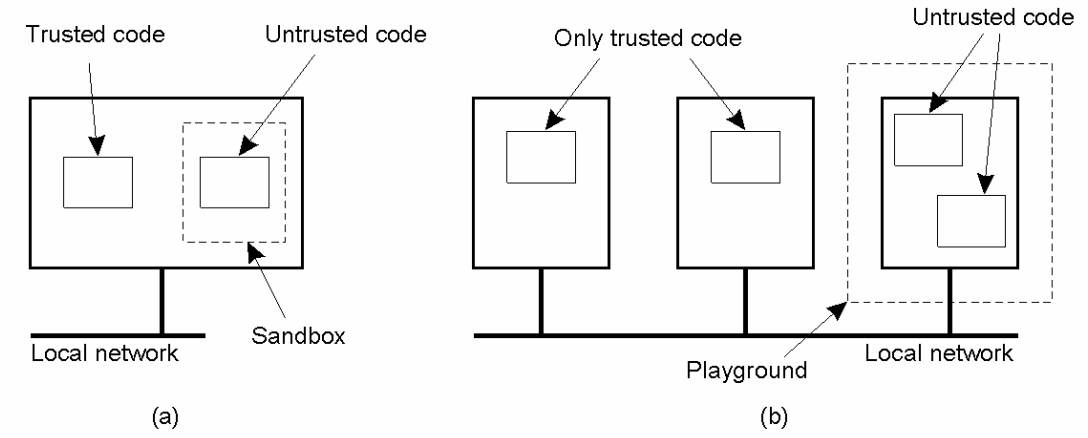
Tehnica prin care fiecare instructiune a unui applet este controlata complet.

- Controlul se bazeaza pe:
  - trusted class loaders – incarca o clasa specificata de la un server si o instaleaza in spatiul de adrese al clientului pentru ca JVM sa creeze obiecte din ea; pentru clase de la distanta se foloseste un incarcator de incredere;
  - byte code verifier – verifica daca o clasa contine instructiuni ilegale sau care pot corupe memoria sau stiva;
  - security manager – face verificari la executie, de ex. pentru operatii de intrare/iesire; de ex. nu permit operatii cu fisiere locale, doar cu cele de pe serverul din care applet-ul a fost descarcat.



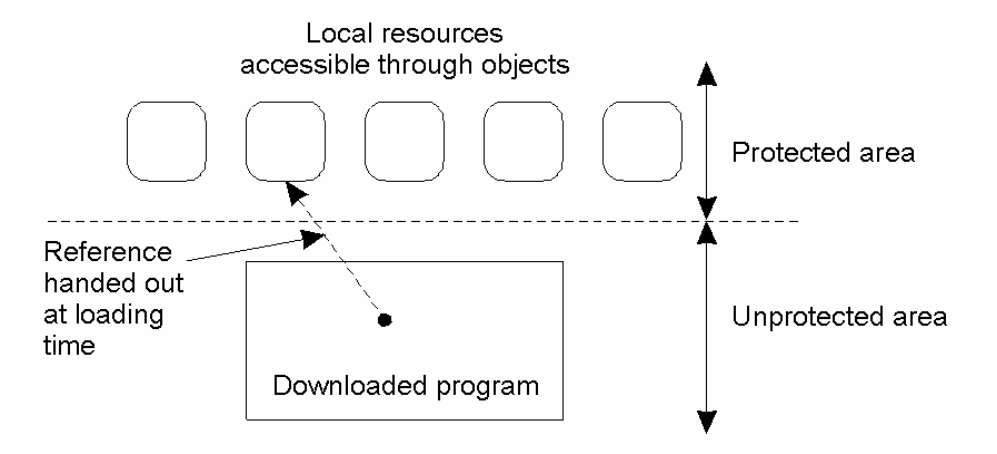
Protejarea Tintei (playground)

Playground este o masina separata, desemnata pentru cod mobil (untrusted code). Codul descarcat are acces doar la resursele din Playground si este supus mecanismelor de protectie uzuale (similare sandbox).



Protejarea Tintei (capabilitati)

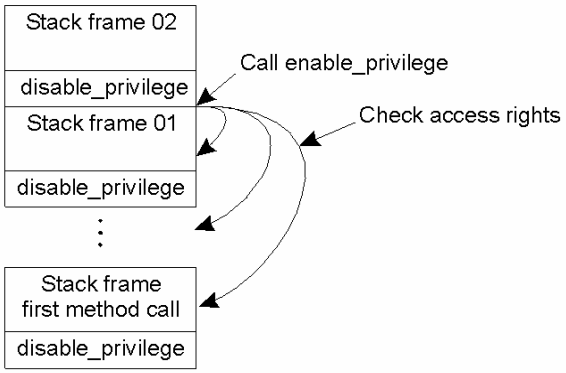
Presupune utilizarea referintelor la obiecte drept capabilitati. La descarcarea unui program, i se paseaza o referinta la un obiect specific prin care se mediaza operatiile de acces la resurse locale. Totodata sunt suprimate posibilitatile programului descarcat de a instantia propriile obiecte de acces la resursele locale.



Protejarea Tintei (stack introspection)

Principiul inspectiei stivei presupune ca la invocarea unei resurse (metode) locale, intrepretorul Java apeleaza automat o procedura enable_privilege care verifica daca accesul este permis. Daca da, se pune pe stiva un apel la o procedura disable_privilege care anuleaza privilegiile la revenirea din metoda.

Verificarea privilegiilor se face mai eficient. Presupunem ca un program P invoca obiectul O1 care invoca o metoda M a lui O2. Daca P nu are dreptul sa invoce metoda M, atunci apelul trebuie interzis, chiar daca O1 are permisiunea sa invoce O2. Verificarea se poate face daca la apelul lui M, enable_privilege inspecteaza fiecare frame din stiva incepand de la varf. Daca un frame interzice accesul la M apelul este respins.



Bibliografie

Y. Yang, A. N. Zincir-Heywood, M. I. Heywood, S. Srinivas **AGENT-BASED ROUTING ALGORITHMS ON A LAN**, In Proceedings of the 2002 IEEE Canadian Conference on Electrical & Computer Engineering, 0-7803-7514-9/02, 2002 IEEE