Agenti Software

Cuprins

- Agenti Autonomi
- Agenti si mediile distribuite
- Arhitectura
- Limbaje de comunicare
- Mobilitate
- Performanta
- Implementare

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]

Agenti Autonomi

- 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
Reactiv	Da	Raspunde la timp schimbarilor din mediul sau
Proactiv	Da	Initiaza actiuni care afecteaza mediul
Comunicativ	Da	Poate schimba informatii cu utilizatorii si alti agenti (abilitati sociale)
Continuu	Nu	Are o durata de viata mare (persistent)
Mobil	Nu	Poate migra de la un sit la altul
Adaptiv	Nu	Capabil sa invete. Flexibil. Nu are actiuni fixe, pre-stabilite.
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

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

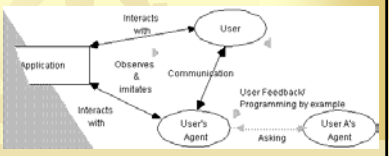
Agenti de Monitorizare
Ofera informatii asupra unor evenimente din sistemul monitorizat: actualizari, mutari, stergeri ale unor informatii

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
  - 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

- agenti cu competente diferite
- probleme mari pentru care un singur agent nu este suficient
- prelucrare informatii din surse distribuite

Exemplu: Automated Design Agents (Univ. of Michigan)

- ACDS (Automated Catalog-Design Service)
  - Fac proiectarea configuratiilor, inclusiv selectia partilor din catalog
  - Organizati 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

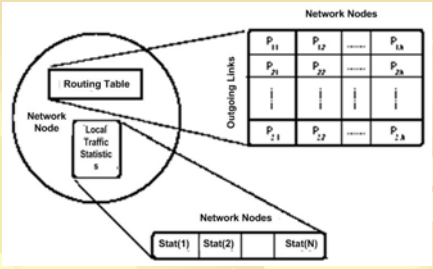
- Executia paralela a taskurilor
- Imbunatatirea capabilitatilor de timp real
- Monitorizarea evenimentelor
- Cresterea robustetii
- Echilibrarea incarcarii (load balancing)

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, folosind aceleasi tabele de rutare
- 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 ferestre glisante



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

$$p'(j) = (p(j)+\beta^* l_j) / (1+\beta^*(N_k-1))$$

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 $p(i) = p(i) + r\,P(i)$

  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

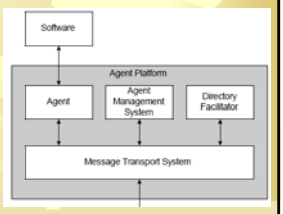
Platforme de agenti - Arhitectura FIPA

FIPA = Foundation for Intelligent Physical Agents
Gestiunea agentilor stabileste
- **modelul logic de referinta** pentru crearea, inregistrarea, localizarea, comunicarea, migrarea si retragerea agentilor
- include **entitatile logice** (serviciile) prezentate in figura

Specificatiile FIPA se refera la
- gestiunea agentilor,
- limbajele de comunicare intre agenti,
- formatul mesajelor de comunicare etc.

Entitatea de baza = **agent**

Agentii comunica folosind ACL (Agent Communication Language).
Un agent are
- un proprietar
- o identitate - Agent Identifier (AID).
- un numar de adrese de transport la care poate fi contactat.



AID - Agent Identifier

AID = colectie (extensibila) de perechi <parametru: valoare>
Include parametrii:
`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))))
```

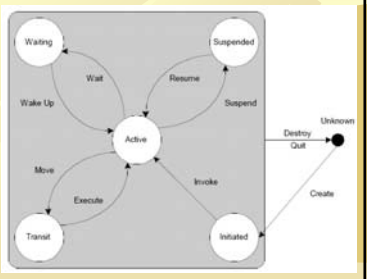
Pentru a afla adresa agentului agent-b, se poate trimite o cerere *search* catre ams@foo.com aflat la adresa iiop://foo.com/acc.

Sistemul de management al agentilor

Sistemul de management al agentilor AMS - **Agent Management System**
- controleaza accesul la si utilizarea unei platforme de agenti
- un singur AMS pe platforma
- pastreaza un director de AID-uri cu adresele de transport ale agentilor 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 cere PA sa execute urmatoarele operatii:
- Suspend agent,
- Terminate agent,
- Create agent,
- Resume agent execution,
- Invoke agent,
- Execute agent,
- Resource management



Serviciul de directoare

Serviciul de directoare = **Directory Facilitator (DF)**
- face posibila regasirea agentilor dupa atribute (serviciu de pagini aurii).

O intrare = o colectie de perechi <cheie-valoare> incluzand:
- numele agentului
- adresa de transport
- servicii oferite
- cost asociat cu utilizarea agentului
- restrictii de utilizare
- etc.

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.

O platforma poate avea mai multe DF-uri care ofera servicii agregate.

Alte componente

- 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.

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)

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)

Setul complet de parametri ai mesajelor

Parametru	Categoria Parametrilor
performative	scopul mesajului
sender	participant in comunicare
receiver	participant in comunicare
reply-to	participant in comunicare
content	continutul mesajului
language	descrierea continutului
encoding	descrierea continutului
ontology	descrierea continutului
protocol	controlul conversatiei
conversation-id	controlul conversatiei
reply-with	controlul conversatiei
in-reply-to	controlul conversatiei
reply-by	controlul conversatiei

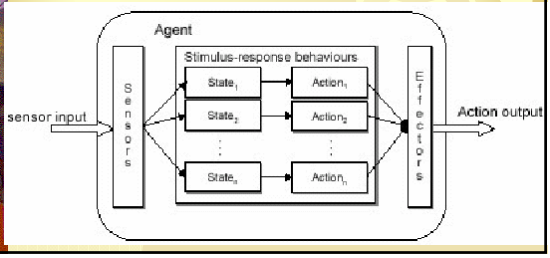
FIPA ACL

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

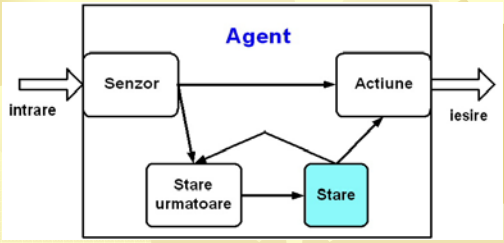
Structura agentilor reactivi

Agentii reactivi au o cuplare stransa intre perceptie si actiuni.



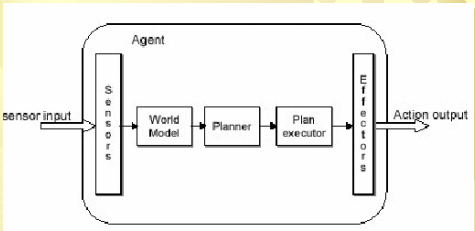
Agenti cu stare

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



Mobilitatea codului

- Proces complicat
- Justificat doar daca duce la imbunatatirea performantelor
- Situatii tipice:
  - echilibrarea incarcarii resurselor unui sistem distribuit
  - exploatarea paralelismului;
    - ex. cautarea realizata in paralel de mai multi agenti care migreaza la diferite servere
  - reducerea comunicarii
    - ex. migrarea codului de la server la client unde executia inseamna o interactiune puternica cu utilizatorul.

- Segmente de Procese
  - Cod
  - Resurse
    - referinte la fisiere, imprimante, echipamente, alte procese, capete conexiune, etc.
  - Segment de executie
    - date private, stiva, contor program
- Modele de mobilitate (1)
  - Slaba - **Weak** - migreaza segmentul de cod cu date initiale (ex. Java applets)
    - Executie in proces tinta (applets executati in spatiul de adrese al browser)
    - Executie in proces separat (SO creeaza un proces pentru executia codului migrat)

Modele de mobilitate (2)

- Puternica - **Strong** - migreaza si segmentul de executie
  - Migreaza procesul
  - Cloneaza procesul
- 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

- Legare (binding) proces - resursa
  - Prin identificator
  - Prin valoare
  - Prin tip
- Legare masina - resursa
  - Resursa neatasata
  - Resursa atasata
  - Resursa fixa

Actiuni executate la migrare

Legare resursa la masina

Legare process la resursa

	Unattached	Fastened	Fixed
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)

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

- Scris in limbaj interpretat
  - Tcl, Java, Scheme
- Mobilitati suportate
  - 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, agent_fork - 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)

Un agent care executa procedura factorial este migrat pe o masina tinta si pus in executie

```
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**.

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 in $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 si e-mail)
Layer 2 - Management si autentificare agenti; management comunicare
- Fiecare agent referit prin <adresa server, id local unic>

Layer 3 - Suport model agenti – partea centrala a platformei de agenti
- start / stop, comunicare inter-agent, migrare

Layer			
5	Agents		
4	Tcl/Tk interpreter	Scheme interpreter	Java interpreter
3	Common agent RTS		
2	Server		
1	TCP / IP	E-mail	

Implementare (2)

- **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

Layer			
5	Agents		
4	Tcl/Tk interpreter	Scheme interpreter	Java interpreter
3	Common agent RTS		
2	Server		
1	TCP / IP	E-mail	

Captarea si transmiterea starii

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 (call frames)	Stiva inregistrarilor de activare (una pentru fiecare comanda in executie) Un frame de apel contine un tabel de variabile locale procedurii, valori si nume ale parametrilor apelului si o referinta la comanda asociata.

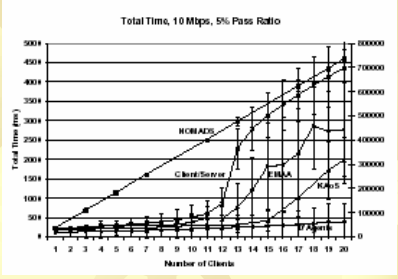
La migrare (agent_jump)
Starea este impachetata si transmisa masinii tinta
Procesul creat la tinta reface starea si ia comanda din varful stivei

Performantele agentilor

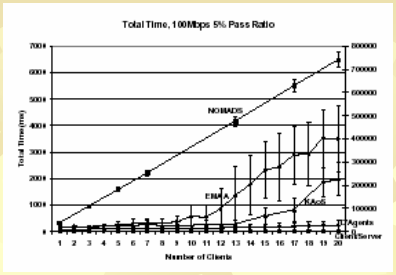
Aplicatie de regasire distribuita de informatii

- 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 (Unif 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



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

Protectie agenti mobili

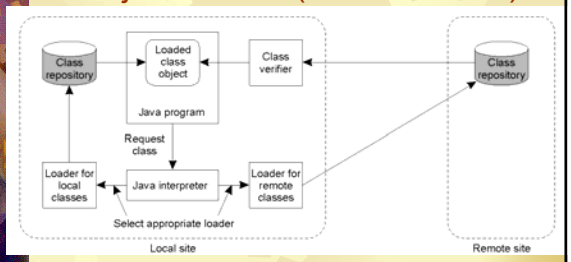
- Exemplu: agent mobil cauta tichet ieftin
- Atacuri:
  - furt informatie credit card
  - plata suma mai mare
  - ocolire competitori mai ieftini
  - furt info de la proprietar, la intoarcere
- Protectie totala imposibila
- Alternative: detectie modificari (Ex: Ajanta)
  - stare read-only (semnata de proprietar)
  - log-uri append-only
  - dezvaluire selectiva

Log-uri append-only

- Initial $C_{init} = K^{+}_{owner}(N)$
- Server S

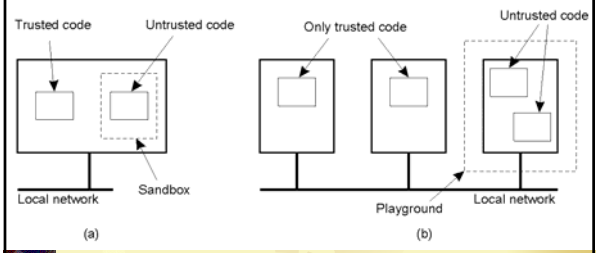
  $Cnew = K^{+}_{owner}(C_{old}, sig(S,X),S)$
- Inapoi la proprietar
- $K^{-}_{owner}(C) => C_{next}$ si sig(X,S), S.
- Proprietarul verifica semnatura si foloseste X
- Repeta operatia pentru C_{next}

Protejarea Tintei (Java sandbox)



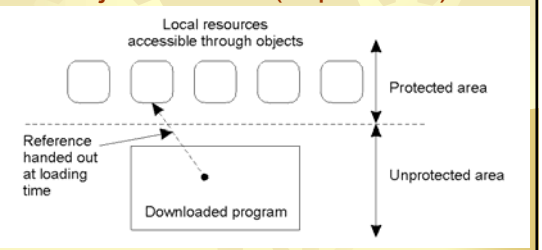
- Controlul executiei applet:
  - trusted class loaders
  - byte code verifier
  - security manager

Protejarea Tintei (playground)



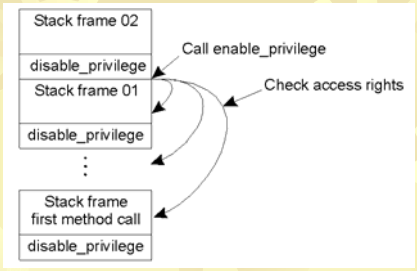
Codul descarcat
- are acces doar la resursele din Playground
- si este supus mecanismelor de protectie uzuale (similare sandbox)

Protejarea Tintei (capabilitati)



- Utilizarea referintelor la obiecte drept capabilitati
  - accesul la resusele locale este mediat de obiecte protejate
  - programul descarcat
    - primeste referinta unui obiect protejat
    - nu poate instantia un obiect propriu pentru acces

Protejarea Tintei (stack introspection)



- enable_privilege apelat automat; disable_privilege pus pe stiva
- Principiul inspectiei stivei
  - P apeleaza O1 care apeleaza O2
  - Pentru verificarea dreptului de acces se inspecteaza fiecare frame din stiva