

# Mobile Agents



Friedemann Mattern, ETH Zürich

*Mobile agents are a solution  
in search of a problem*

*John Ousterhout*

Note: Some slides are based on

- a tutorial given by Fritz Hohl,  
University of Stuttgart

- a presentation by George  
Cybenko and Bob Gray,  
Dartmouth College

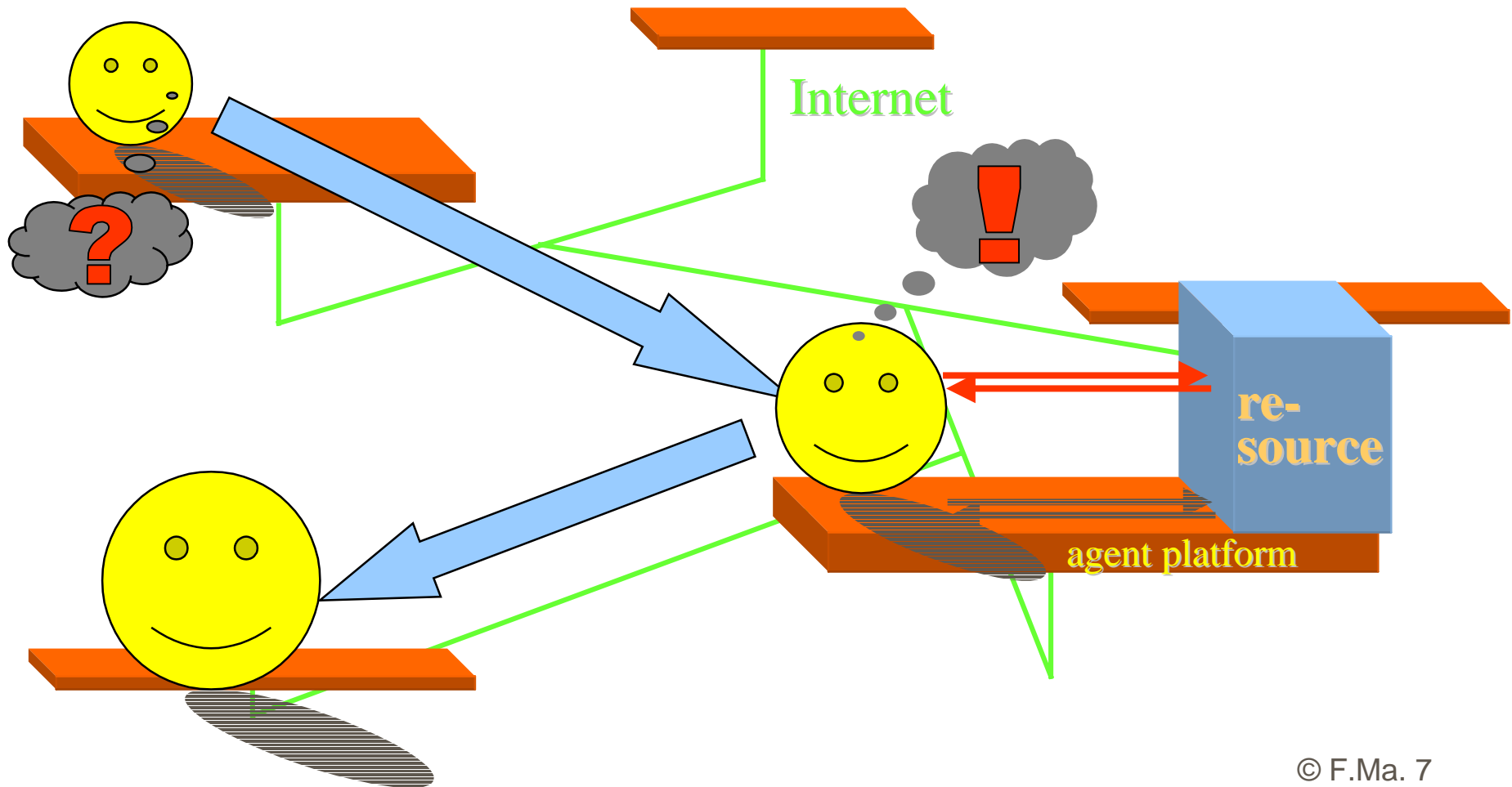
Granting the use of their slides  
is gratefully acknowledged

The slides of this presentation may be found at  
[www.inf.ethz.ch/vs/publ/selected\\_talks.html](http://www.inf.ethz.ch/vs/publ/selected_talks.html)

# Outline

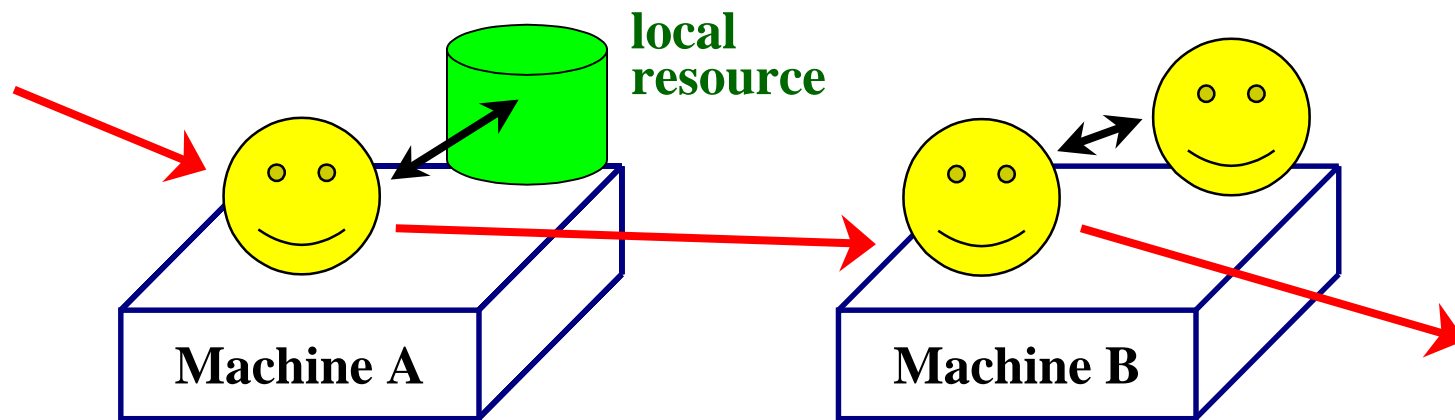
- What are agents and mobile agents?
  - mobile code
  - use of mobile agents
  - application areas
- Agent infrastructure and agent platforms
  - migration
  - communication
  - security
- Conceptual problems
- Agent systems
- Challenges and Problems

# Mobile Agents



# Mobile Agents - a First Characterization

- Program instances that are able **move** within a heterogeneous network (Internet, intranet)
  - under their **own control**
  - **interact** with local resources and other agents
- And do something useful
  - as a **delegate** of some "authority" (e.g., their creator)



# Example of a Roaming Agent: The “who” Agent (D’Agents Project)

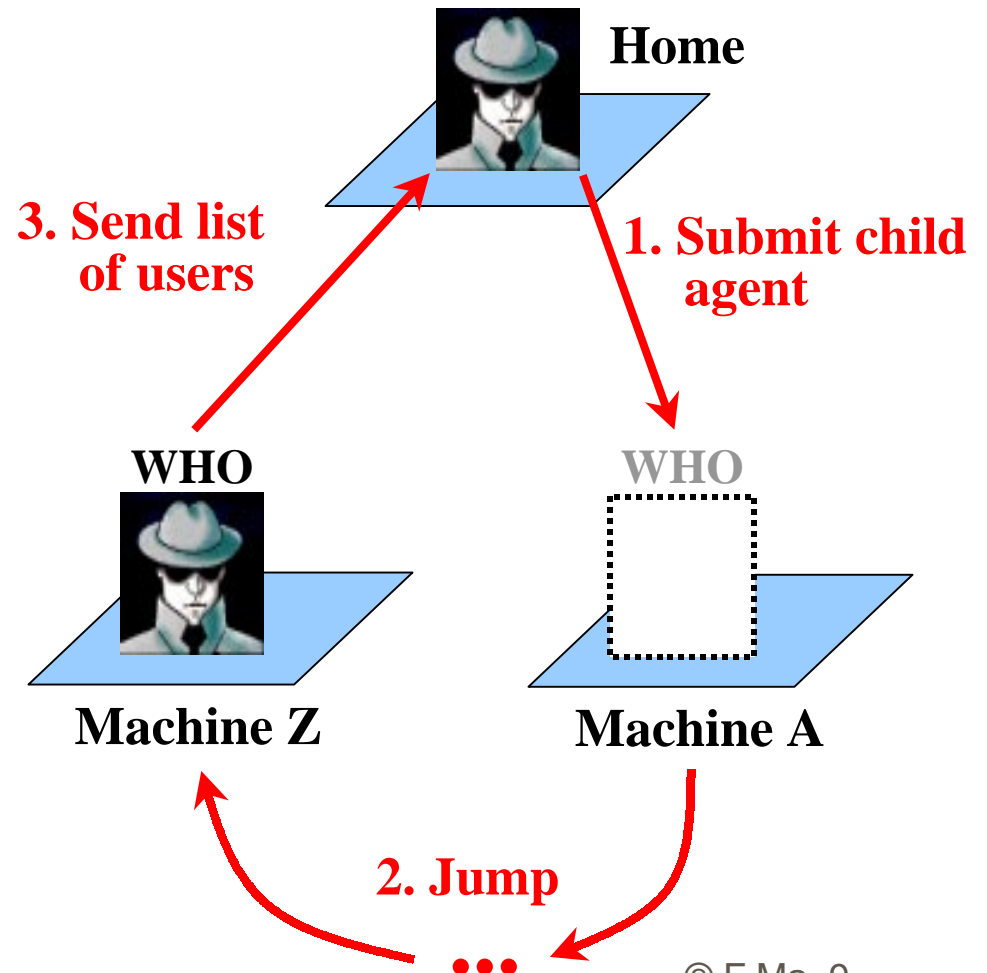
```
...                               Child Agent
set results {}

# migrate through machines

foreach machine $machines {
  agent_jump $machine
  append results \
    [exec who << {} ]
}

# send back results

agent_send \
  $agent(root) 0 $results
...
```





US005603031A

# United States Patent [19]

[11] Patent Number: **5,603,031**

White et al.

[45] Date of Patent: **Feb. 11, 1997**

A distributed computing environment in which agent processes direct their own movement through a computer network.

inities",  
nputing  
03, XP

[75] Inventors: **James E. White**, San Carlos;  
**Christopher S. Helgeson**; **Douglas A. Steedman**, both of Mountain View, all of Calif.

(List continued on next page.)

*Primary Examiner*—Kevin A. Kriess  
*Attorney, Agent, or Firm*—Skjerven, Morrill, MacPherson, Franklin & Friel; Forrest E. Gunnison

[73] Assignor: **General Magic, Inc.**, Sunnyvale, Calif.

## [57] ABSTRACT

[21] Appl. No.: **90,521**

A distributed computing environment in which agent processes direct their own movement through a computer network. Place processes provide a computing context

[22] Filed: **Jul. 8, 1993**

[51] Int. Cl.<sup>6</sup> ..... **G06F 13/00**

[52] U.S. Cl. .... **395/683**

[58] Field of Search ..... 395/650, 700

within which agent processes are interpreted. An agent process controls its movement from one place process to another within the network by using a ticket. An agent process which moves from one place process to another transports definitions of classes of which objects included in the agent process are members. An agent process which moves from one place process to a second place process avoids unnecessary transportation of objects included in the agent process by substituting equivalent objects which are found in the second place process. An agent process sends clones of the agent process to several place processes simultaneously. If two clones travel along paths which are coextensive for an initial portion thereof, a single clone is transported along the initial portion of the paths and other

## [56] References Cited

### U.S. PATENT DOCUMENTS

4,575,797	3/1986	Griner et al. . . . .	
5,079,695	1/1992	Dysatt et al. ....	395/700
5,093,914	3/1992	Coplien et al. ....	395/700
5,129,083	7/1992	Cufler et al. . . . .	
5,206,951	4/1993	Khoyi et al. ....	395/650
5,261,080	11/1993	Khoyi et al. ....	395/500

# Interest in Agents

- One observes **much interest** in agents
  - many research projects, hype & hope
  - conferences, workshops, ...
- Important potential **application areas**
  - Internet, electronic commerce,...
- But also some important **obstacles** remain
  - security, complexity, interoperability,...
  - of course, **researchers like problems!**
- Why are *you* interested in agents?
  - when did you first hear about „agents“?

Why? Maybe we will know after this talk or at the end of the week

# First Encounters with Agents

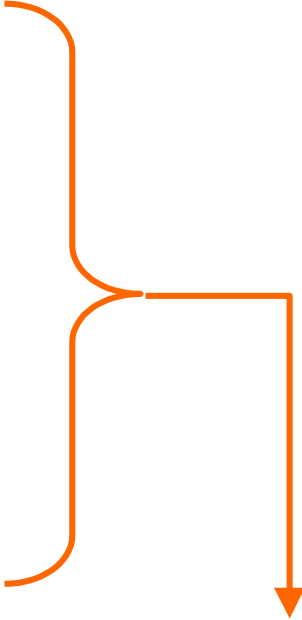
- My first encounter dates back 20 years ago
  - Master Thesis „Computing System for Societies of Agents“ (CSSA)
  - CSSA was inspired by the actor model (Hewitt, 1977)
- At that time (1981):
  - agent = reactive, computational object
  - cooperation of agents by communication
    - messages (synchronous, asynchronous)
    - remote operation invocation
  - user controls an agent through an interface agent
  - agents are coordinated by distributed control algorithms
- Many interesting research themes were generated by the agent paradigm...
- How could this idea ever generate practical interest?



# Remember 1981?



- no PCs
- no Java
- no LANs (in Universities)
- no Internet (in Europe)
- no E-mail (in Europe)
- no .com-companies

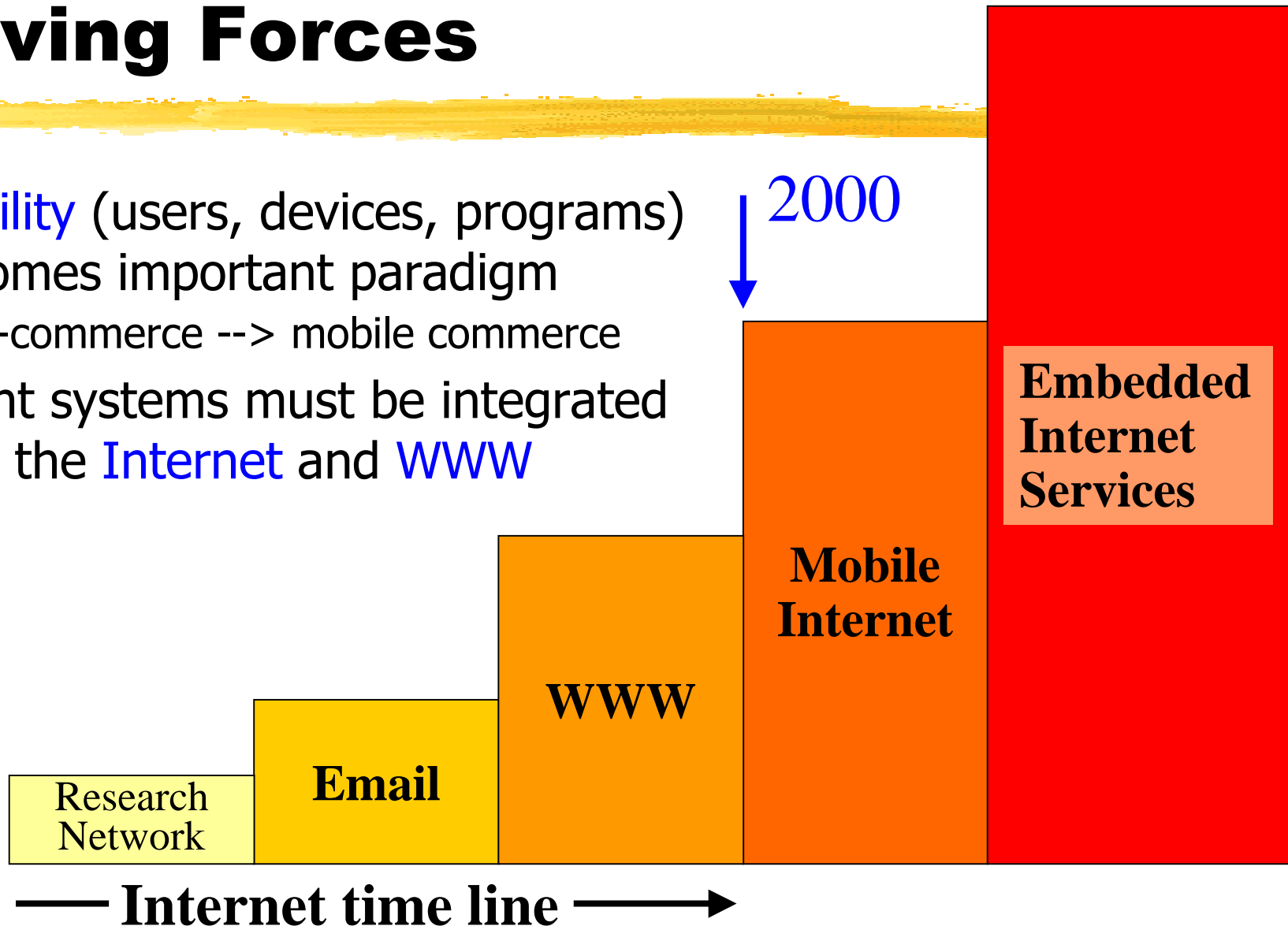


maybe all *this* is  
necessary for *that*?

- 
- no interest in agents
  - no séminaire de printemps on „Agent Technology“

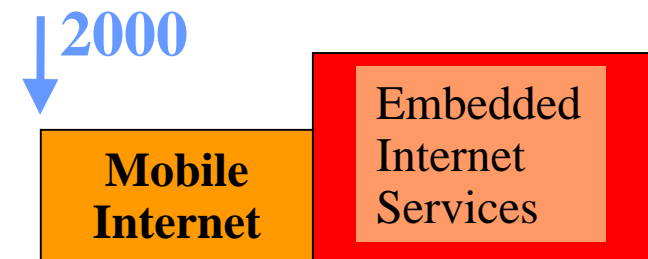
# WWW and the Internet As Driving Forces

- **Mobility** (users, devices, programs) becomes important paradigm
  - E-commerce --> mobile commerce
- Agent systems must be integrated with the **Internet** and **WWW**



# Good Times for Mobile Agents

- Internet grows and becomes ubiquitous
  - becomes **dynamic** and **mobile**
  - networked embedded systems
- New applications
  - plug & play for the masses
  - **adaptive software**, customization
- Shorter development cycles
  - **fast reaction** to dynamic market
  - remote configuration & installation
- New market strategies
  - e.g., dynamic federations of services



Mobile agents may be well suited for these modern times: they allow dynamic and flexible system structures (*e.g., function shipping, call by visit, code on demand...*)

# What is an Agent?

- No consensus on precise definition!
- Latin „agere“: to act, to drive
- Authorized to act autonomously in the name of or in place for someone
  - someone delegates a task to an agent
  - agents are representatives (of some „authority“)
- Software agent
  - as opposed to persons or physical robots
  - „a computer program acting autonomously on behalf of...” [OMG MASIF]

# General Agent Properties



## ■ Autonomous

- have control over their own actions and may operate without direct intervention of others

## ■ Interactive, communicative

- communicate with other agents, with humans, with their home base, with WWW resources...
- problem: syntax and semantics of communication?

# Intelligent Agent Properties - The AI View



- Goal oriented
- Reactive
  - they perceive their environment and respond to changes that occur in it
- Proactive
  - take initiative, not only respond to the environment
- Cooperative, social
- Intelligent
  - perform domain oriented reasoning
- Adaptive
  - they learn

Research on intelligent agents was initiated quite early, e.g. by Carl Hewitt („actor model“, 1977) or Marvin Minsky („society of mind“)

Mobility is not an issue here!

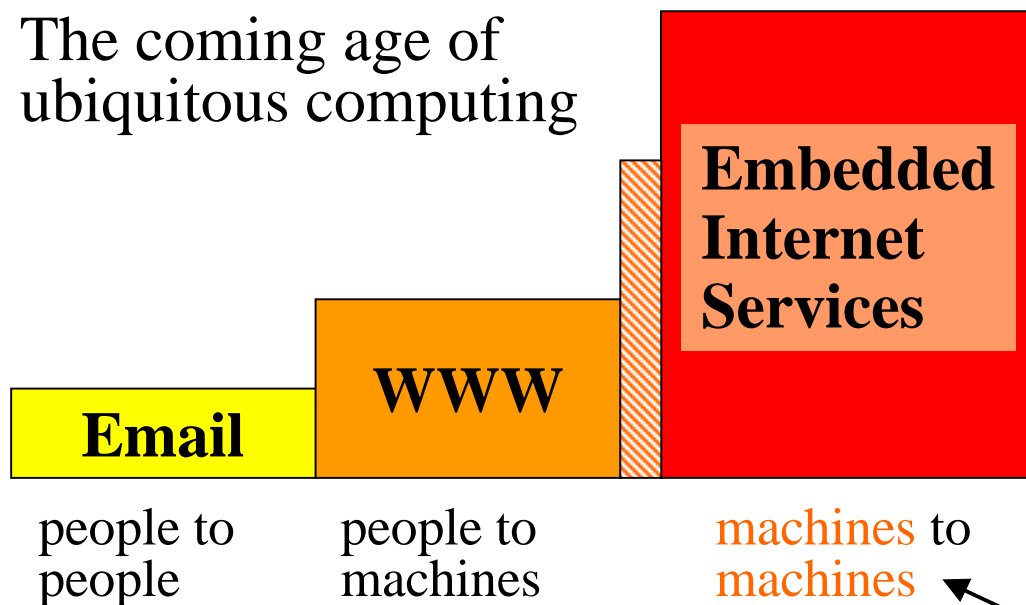
# Agents: What Are They Good for?

- Find and filter information
- Automate work
  - respond to events, such as a new version
- Customize views of information
  - e.g., email
- Make recommendations
- Perform simple cooperative tasks
  - e.g., scheduling of meetings
- ...

In the AI view, agents are often perceived as **intelligent assistants** (e.g., personal agents or „interface agents“ that learn from user actions)

# Good Times for Intelligent Agents

The coming age of  
ubiquitous computing



- Automatic processes without user intervention

- cooperating **autonomous**, decentralized entities
- goal oriented** (some useful task has to be done)
- reactive** (changing and dynamic environment)

==> Agent paradigm is quite natural here!

(already in the WWW age we have software robots and web crawlers)



# Intelligent vs. Mobile Agents

- Are these two completely different areas?
  - **intelligent** agents and multiagent systems (from the AI community)
  - **mobile** agents (from the systems community)

„The association is almost an accident because we both use the term „agent“... The groups don't necessarily talk with each other. Part of that is because the intelligent agents and multiagent people come from the AI community, and a lot of the mobile agents community come from the systems or even OS community. Thus we have different languages, different goals, and different ways of looking at these problems.

-- Dave Kotz, IEEE Concurrency, July 1999

# Mobile Agents...



- ... are not necessarily intelligent
  - but they are autonomous (to a certain degree)
  - and they are usually communicative
- ... are not always moving
  - they may also be stationary
- ... are an elaborate form of **mobile code**

# Mobile Code



- Basic idea:

- program code is moved to a remote site and executed there

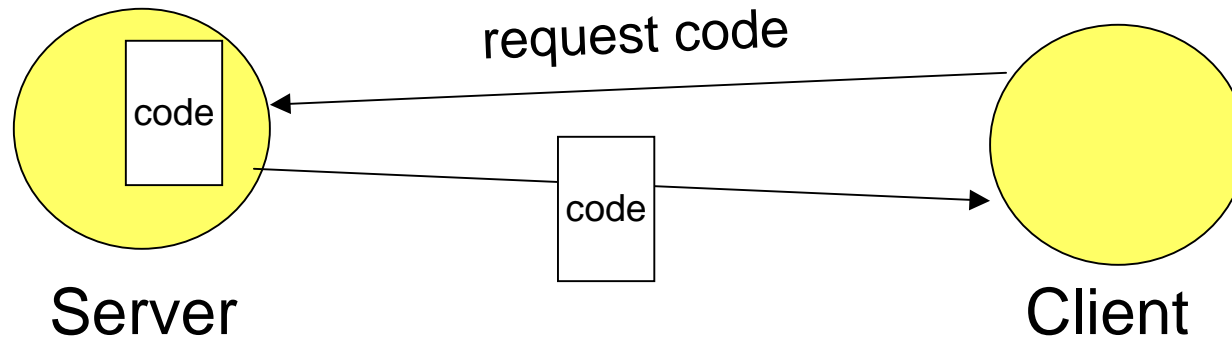
- Problems:

- heterogeneity of systems
  - machine code, resources at remote site
- security (as more parties are involved)

- Systems can be classified according to initiator of code transfer

- *pull or push*

# Mobile Code: Pull Principle



- Client asks for code and executes it at client site
- Example: Java applets

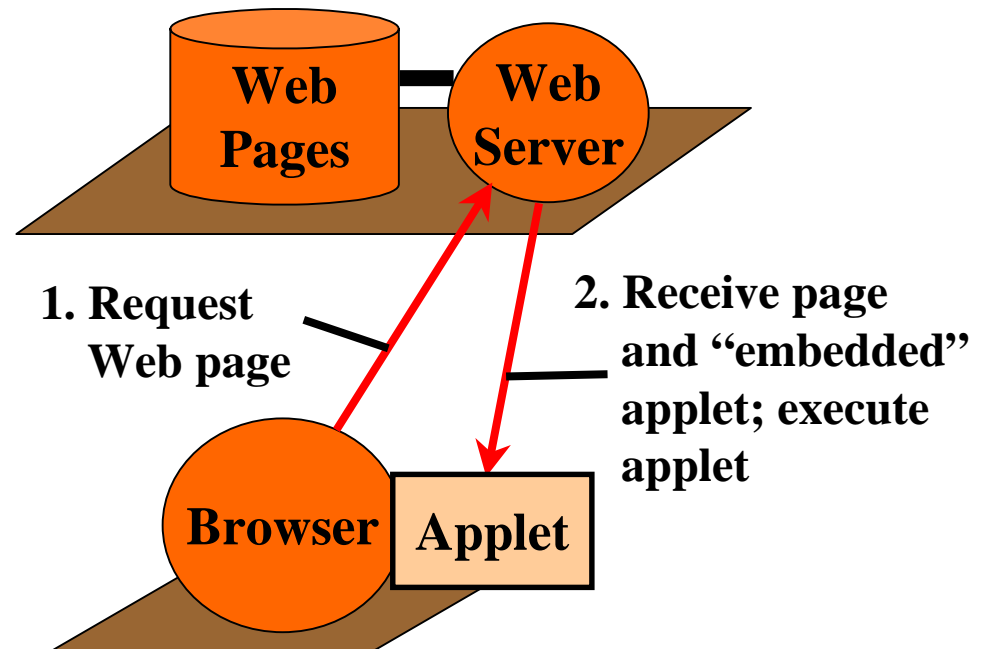
# Mobile Code: Pull Principle



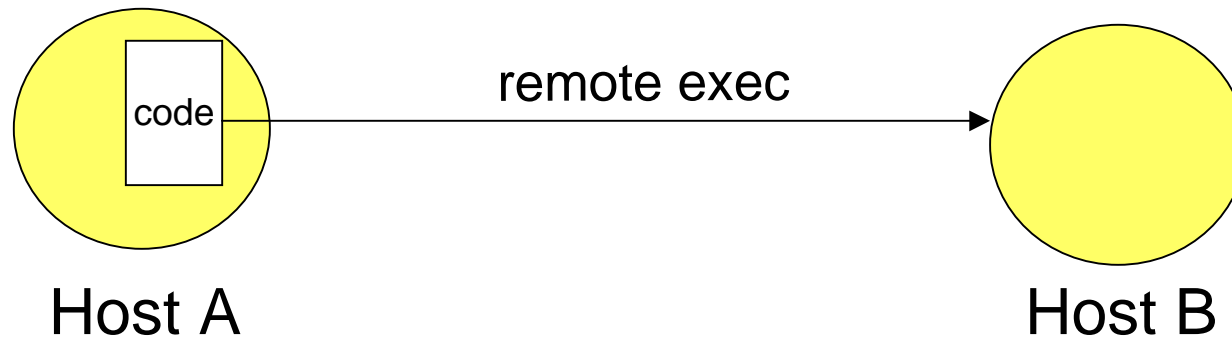
- Client asks for code and executes it at client site
- Example: Java applets

# Applets As Mobile Code

- Java & WWW
- Not yet a mobile agent
  - applet cannot move further on
  - moves only from server to client, not the other way
  - does not migrate autonomously, has to be fetched



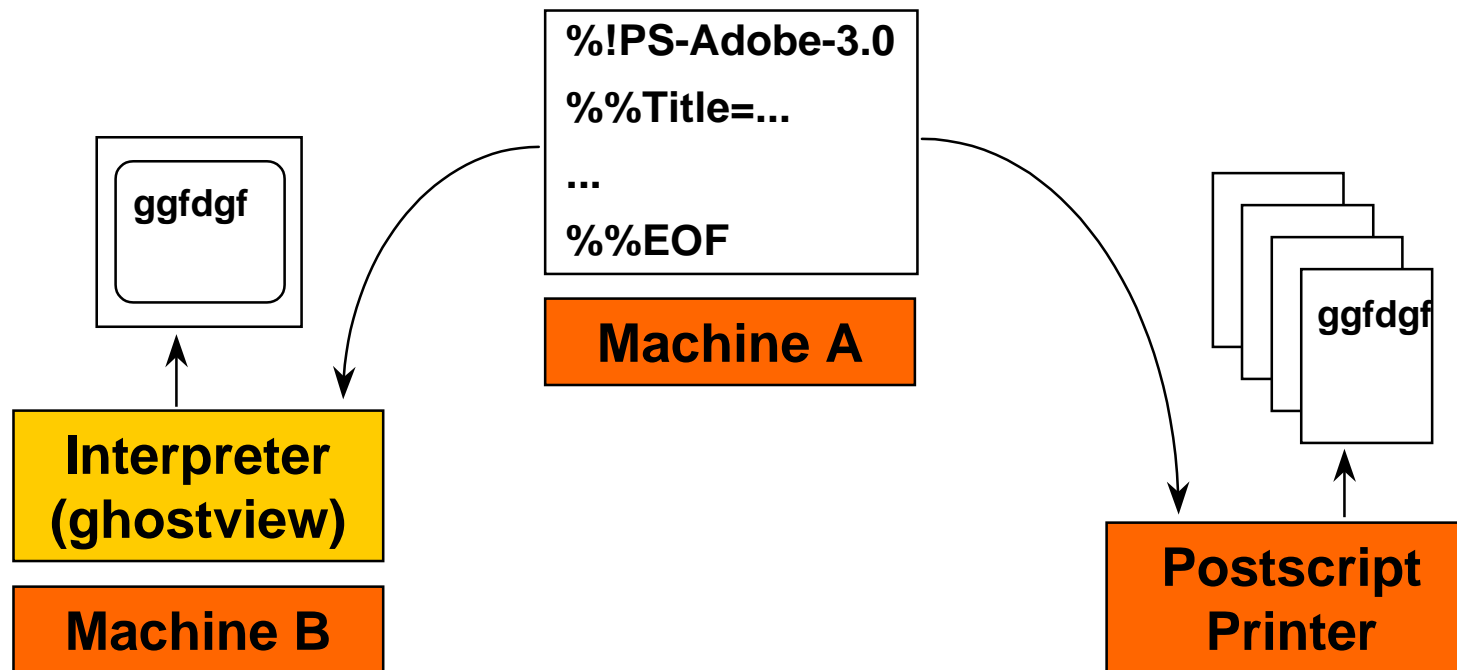
# Mobile Code: **Push** Principle



- Host sends code to another Host to execute it there ("remote execution")
- Example: postscript page description language

# Postscript - an Example of Mobile Code

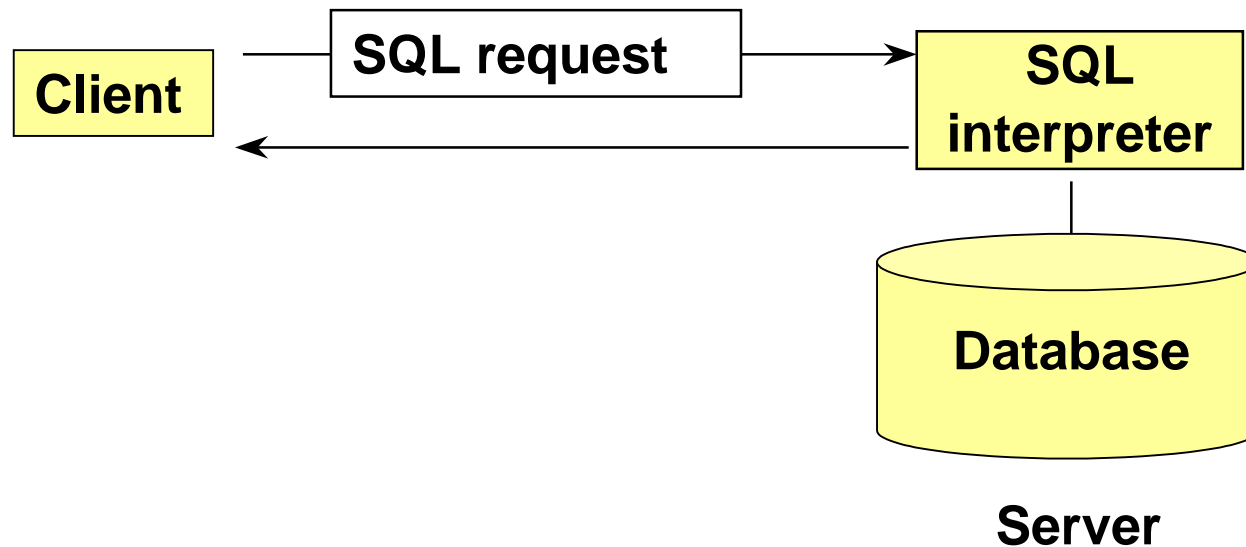
- Postscript code is executed by remote machine
  - physical or virtual machine





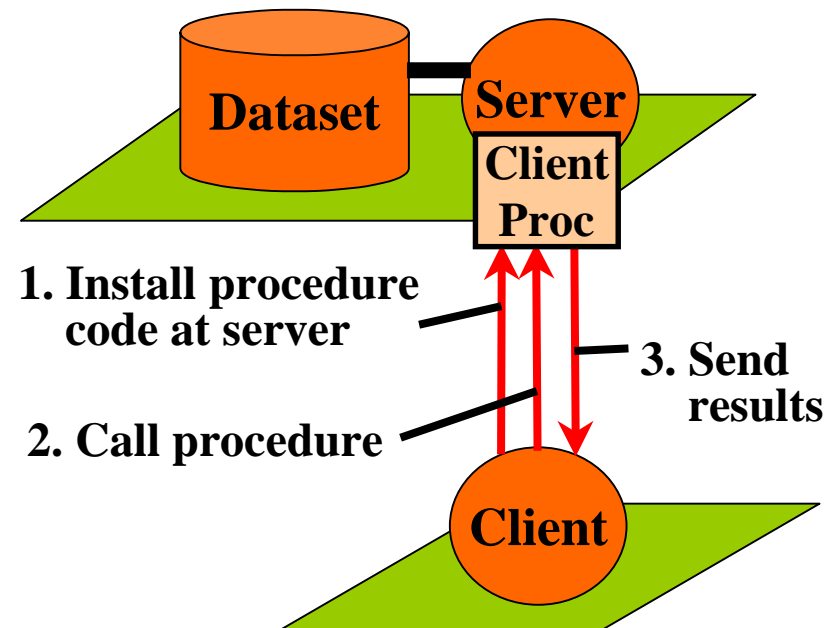
# SQL - Another Example of Mobile Code

- SQL code sent to database server
- Code executed by remote SQL interpreter



# Stored Procedures vs. Mobile Code

- Typically used with Databases
- Not a mobile agent
  - procedure cannot move further on
  - cannot spawn other procedures
  - does not actively communicate with other objects



# Active Networks - an Application of Mobile Code



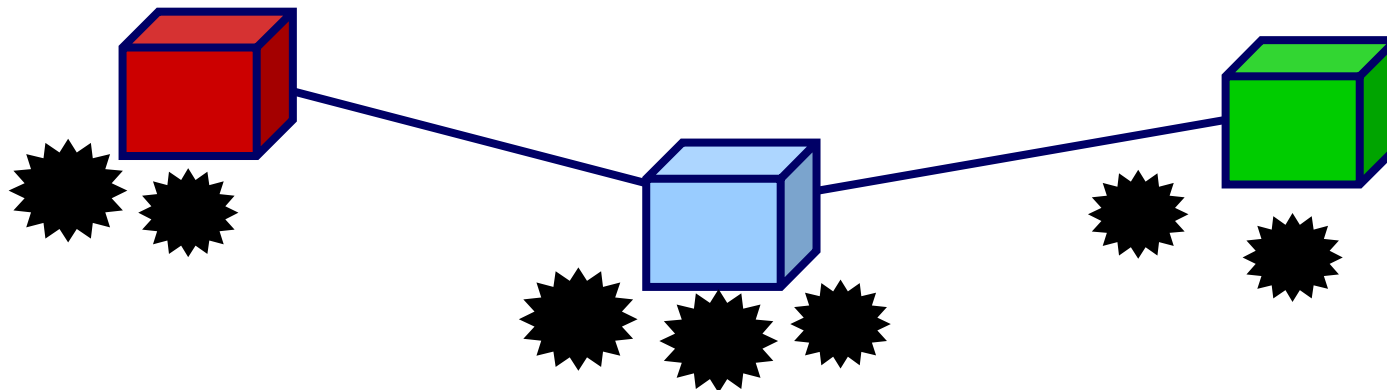
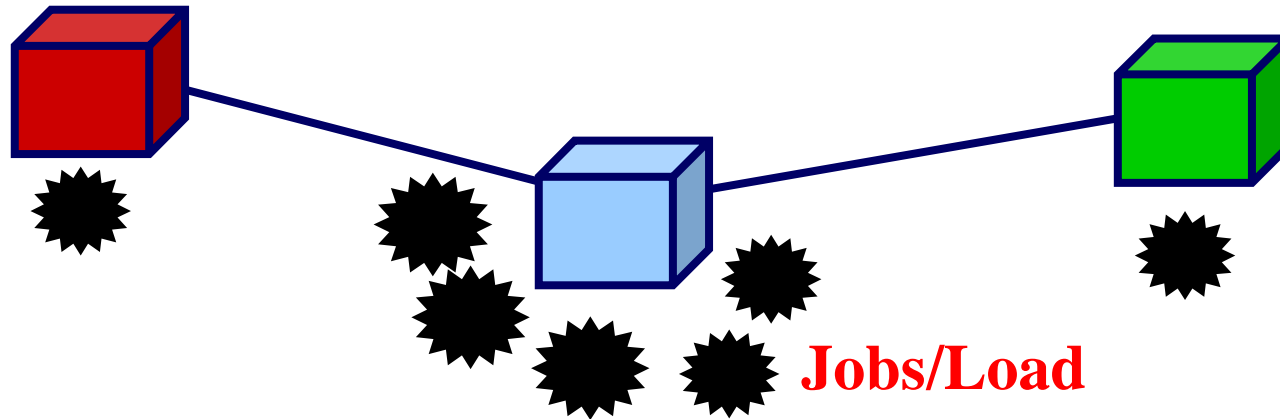
- Transporting components of a network (e.g. routers) can execute arbitrary code
- Code is provided from special (“active”) packets injected by users
- Active packets are mobile code entities!

# Mobile Code: Applications and Reasons



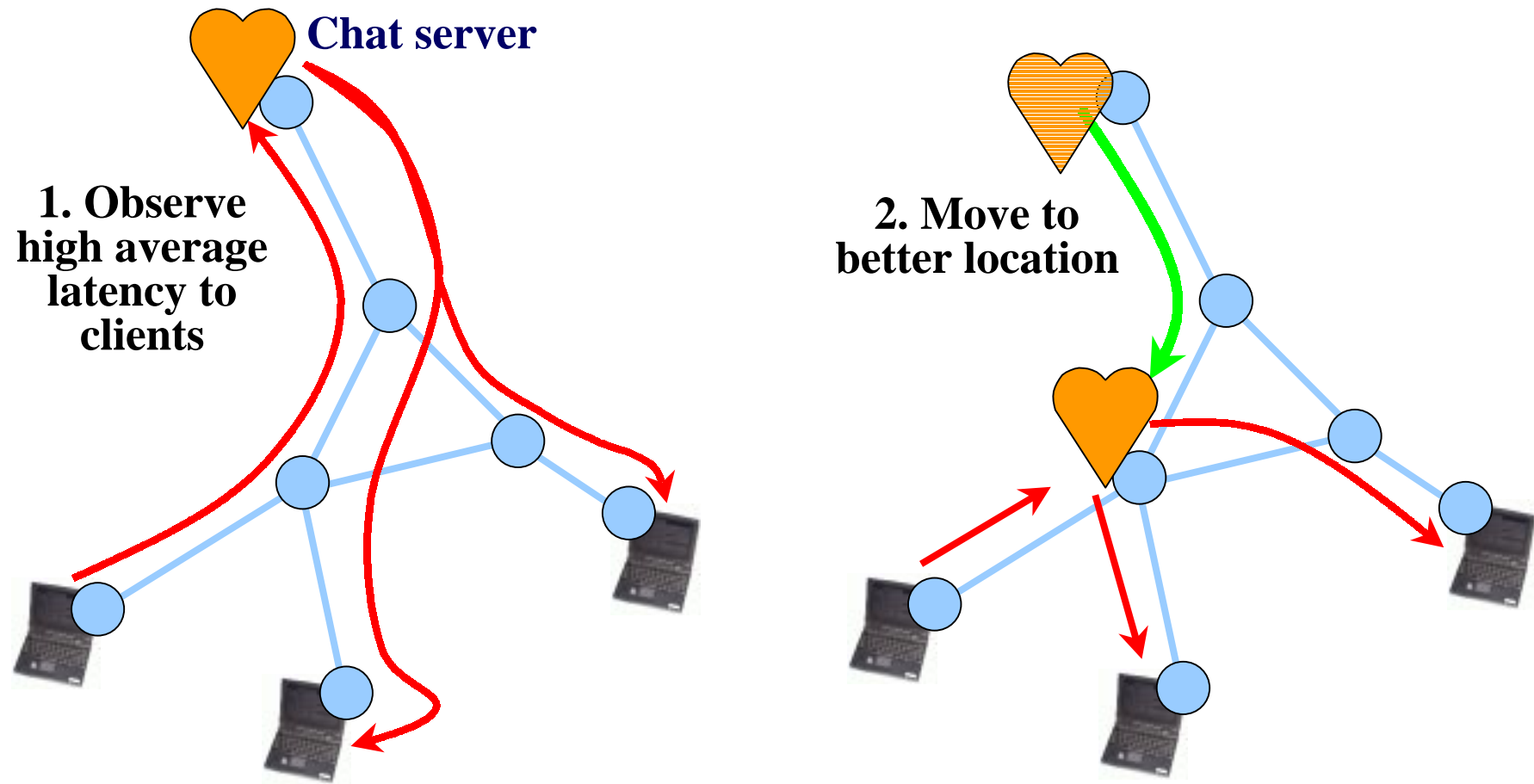
- Efficiency
  - load balancing
  - move function to faster machine or closer to the data
- Automatic software update
  - push principle
  - code may be provided by remote code server
- Dynamic extensions of services
- ...

# Load Balancing with Mobile Code



**Jobs/Load migrate in a network of machines**

# Latency Reduction with Mobile Code



# Mobile Code is Nothing New



- Submitting batch jobs on mainframes
  - *Remote Job Entry*
- *Process migration* in operating systems
  - typically on a local scale (e.g., LAN cluster)
  - main goal: load balancing
  - migration initiated by the operating system
  - challenge: location transparency, efficiency
  - more recent variant: *object* migration (e.g., Emerald, Cool)
- Active mail
- ...

# Mobile Agents and Mobile Code

- Mobile **agents** consist of
  - **code** (are therefore also mobile code entities)
- But in addition
  - **data state** (i.e. variables)
  - **execution state** (stack, program counter)
- Agents **decide on their own** when to migrate
  - what about location transparency?
  - aren't agents *location aware* by design and purpose?

All this makes **agent migration more complex** than simple code migration!



# What Makes Mobile Agents Attractive?

Compared to classical client-server programming  
(Remote Procedure Call: RPC)

- More **dynamics**
  - compared to pre-installed remote procedures
- **Asynchronous** behavior and parallelism
  - while the agent acts on behalf of an authority, the authority may perform other tasks
- Improved **real-time** abilities, fast reaction to remote events
  - acts near the source of the event (but represents a distant entity)

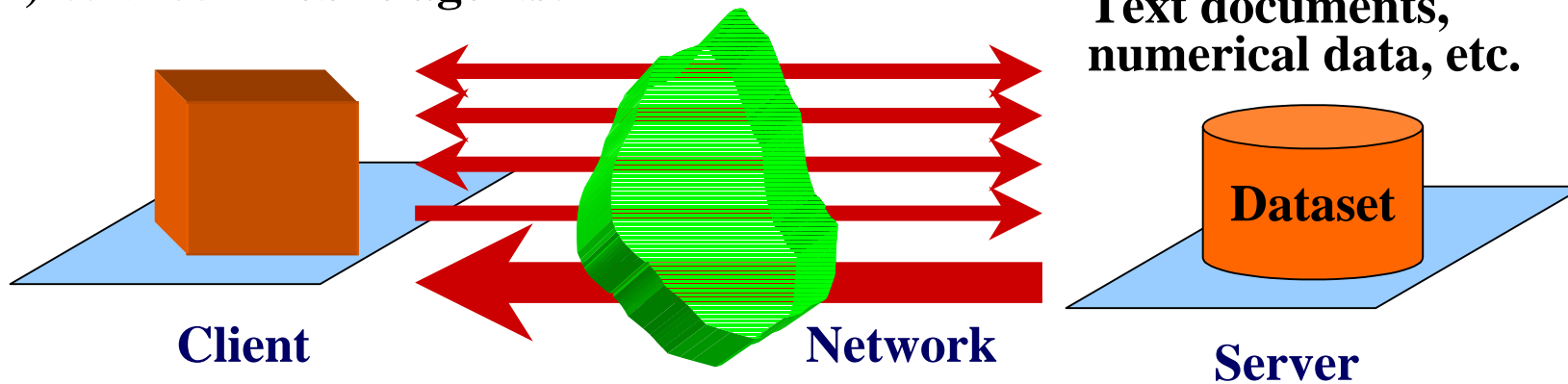
# ...Attractive?



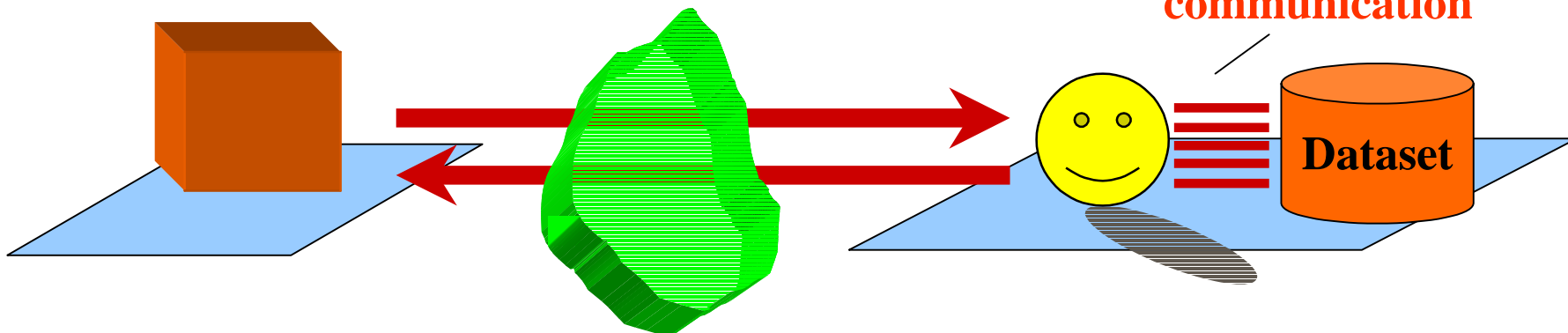
- Higher degree of **robustness** and **fault tolerance** (?)
  - agent may decide autonomously to go to a backup server
  - however: more things can go wrong with mobile agents!
- Support of **nomadic computing**
  - „migrate and disconnect“: launch agent during brief connected session
- Use of **local resources** that are not „exported“
- Reduced **communication bandwidth**
  - but: when is it better to **send data** to the program instead of **sending the program** to the source of data?

# Bandwidth Conservation

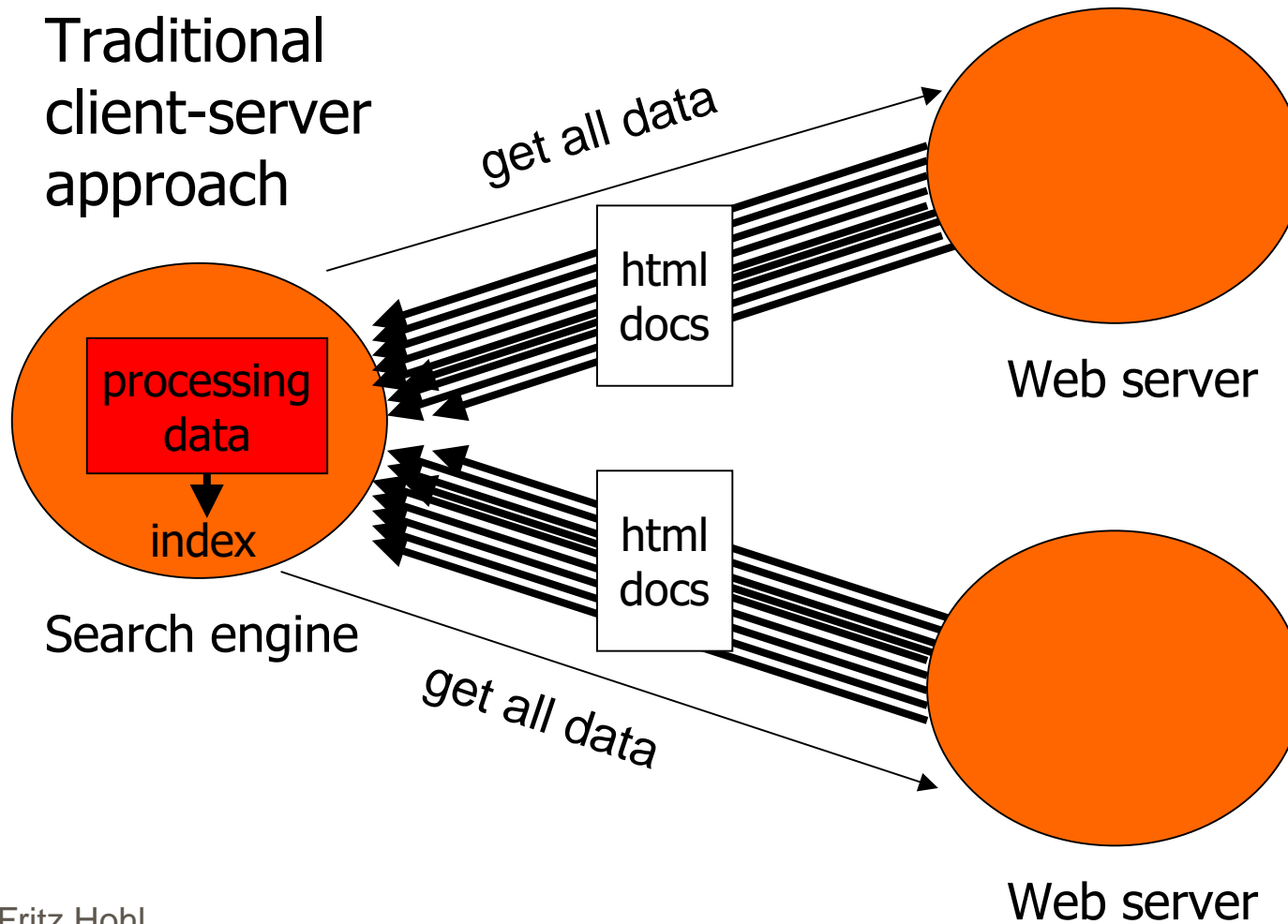
1) **Without** mobile agents:



2) **With** mobile agents:



# Example: Search Engine

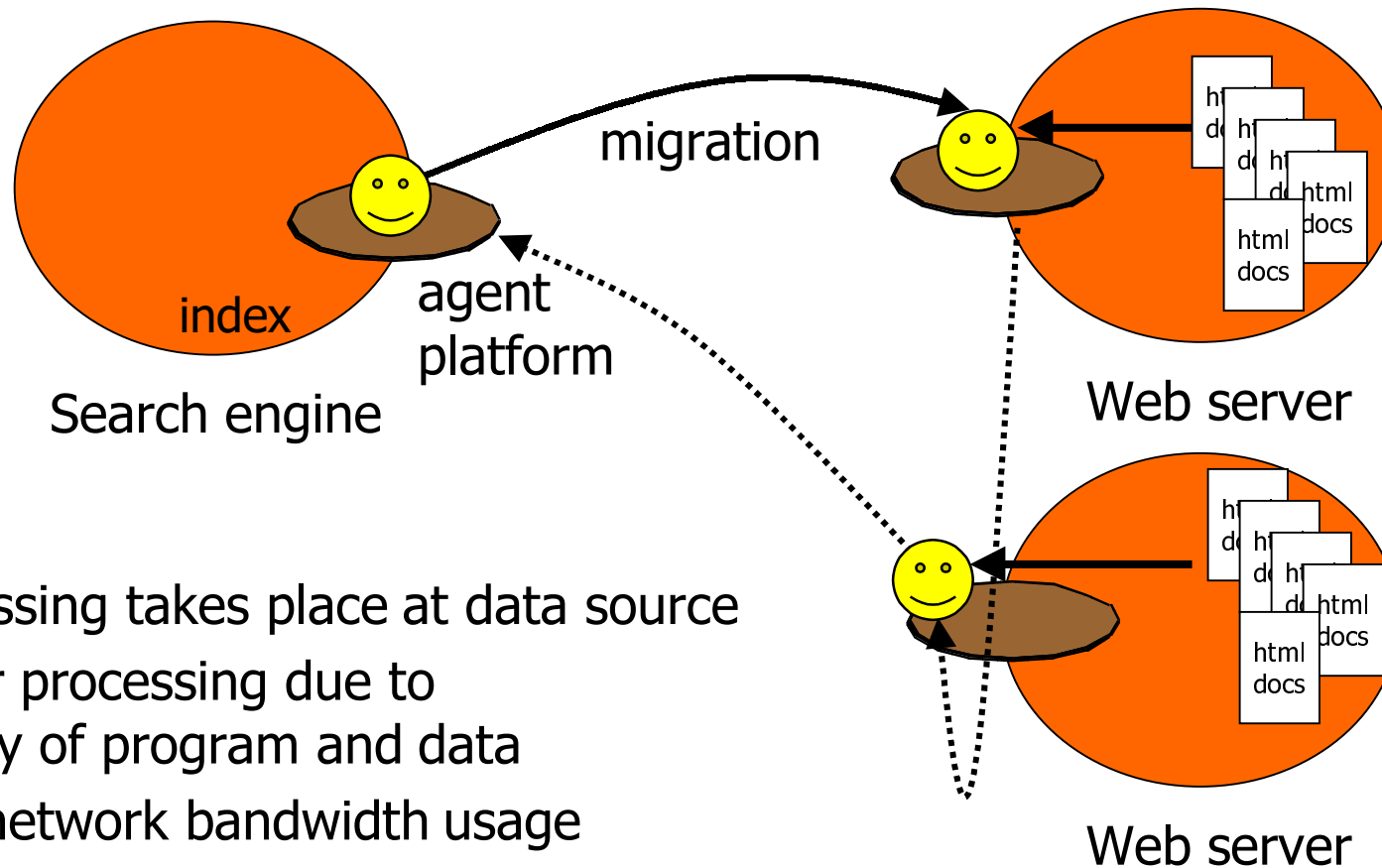


# Example: Search Engine



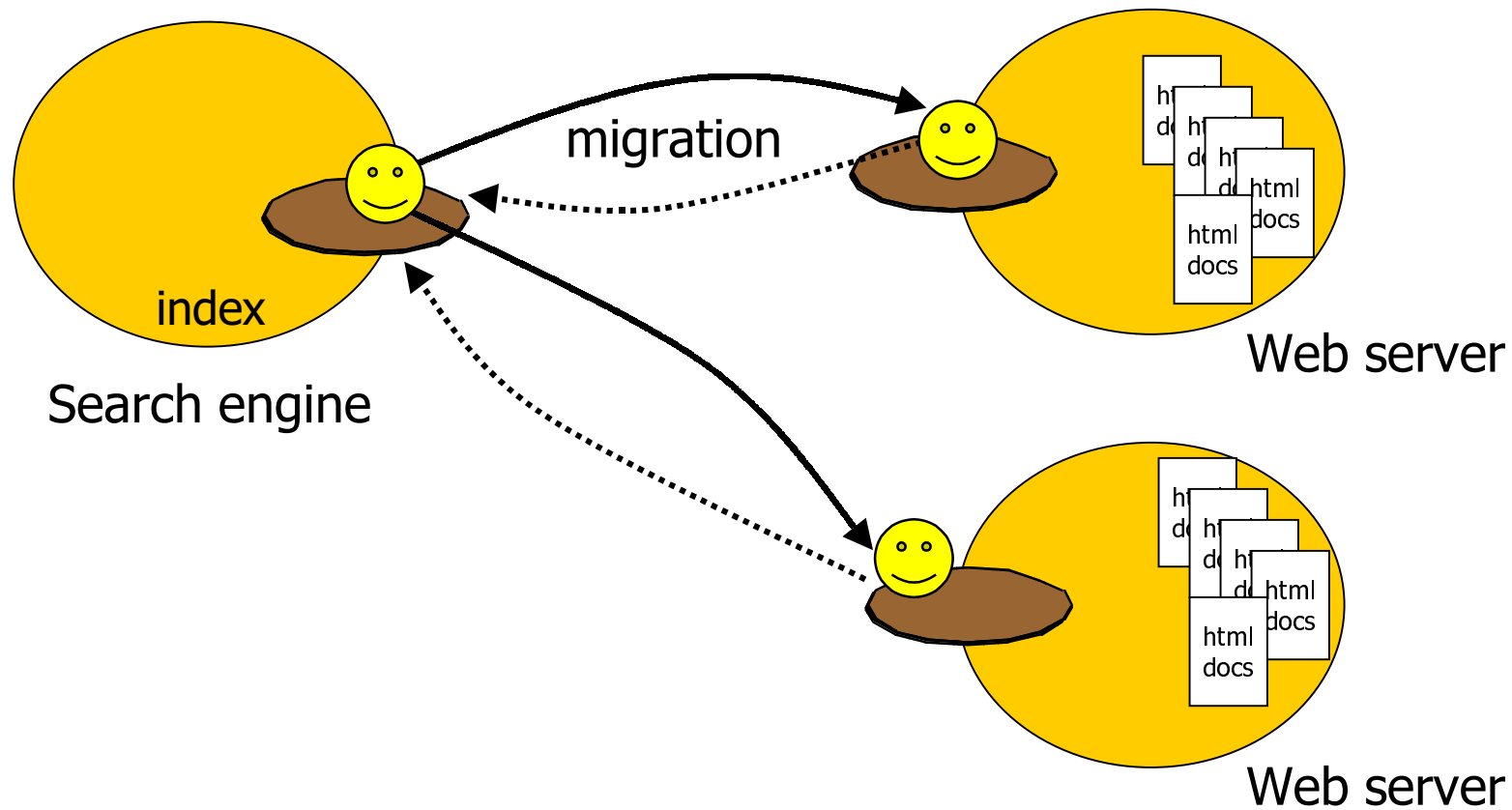
- The traditional client-server approach
  - does not scale
  - induces extensive network usage
- Bottlenecks:
  - network bandwidth
  - processing capacity of search engine

# The Mobile Agent Approach



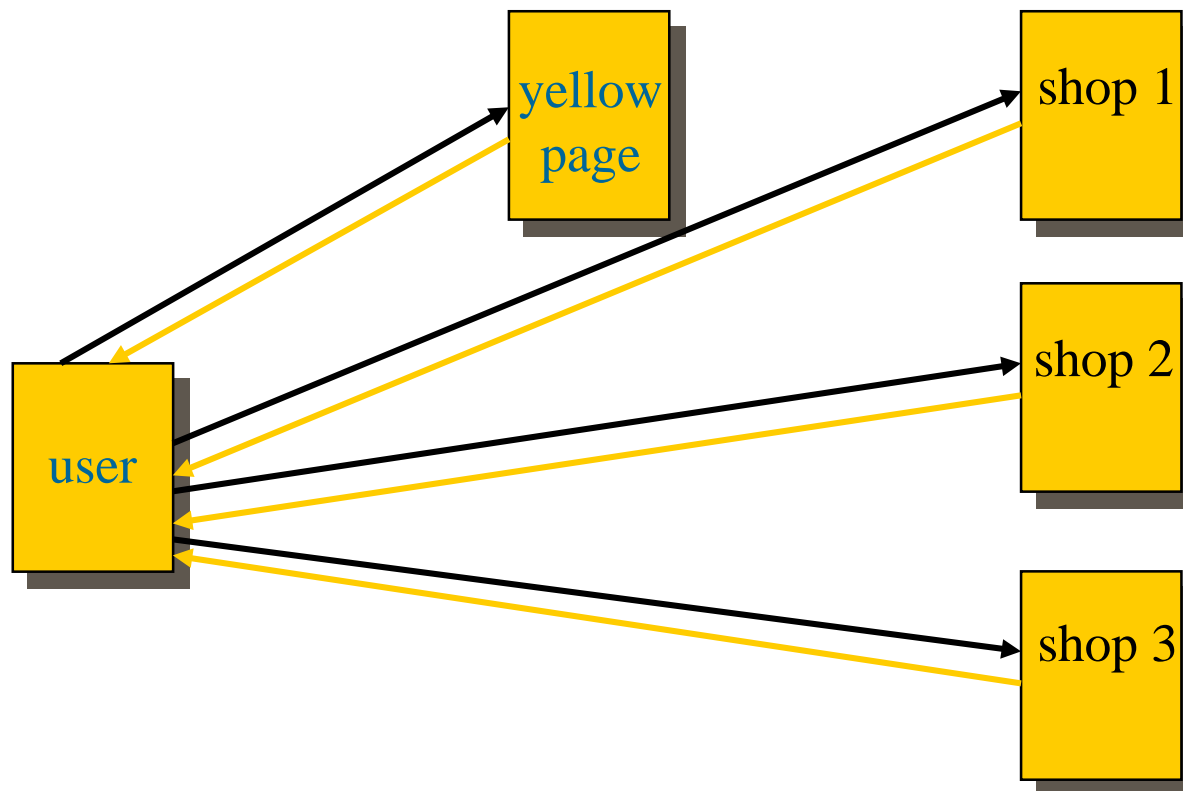
- Processing takes place at data source
- Faster processing due to locality of program and data
- Less network bandwidth usage

# Parallel Processing: Spawning Clones



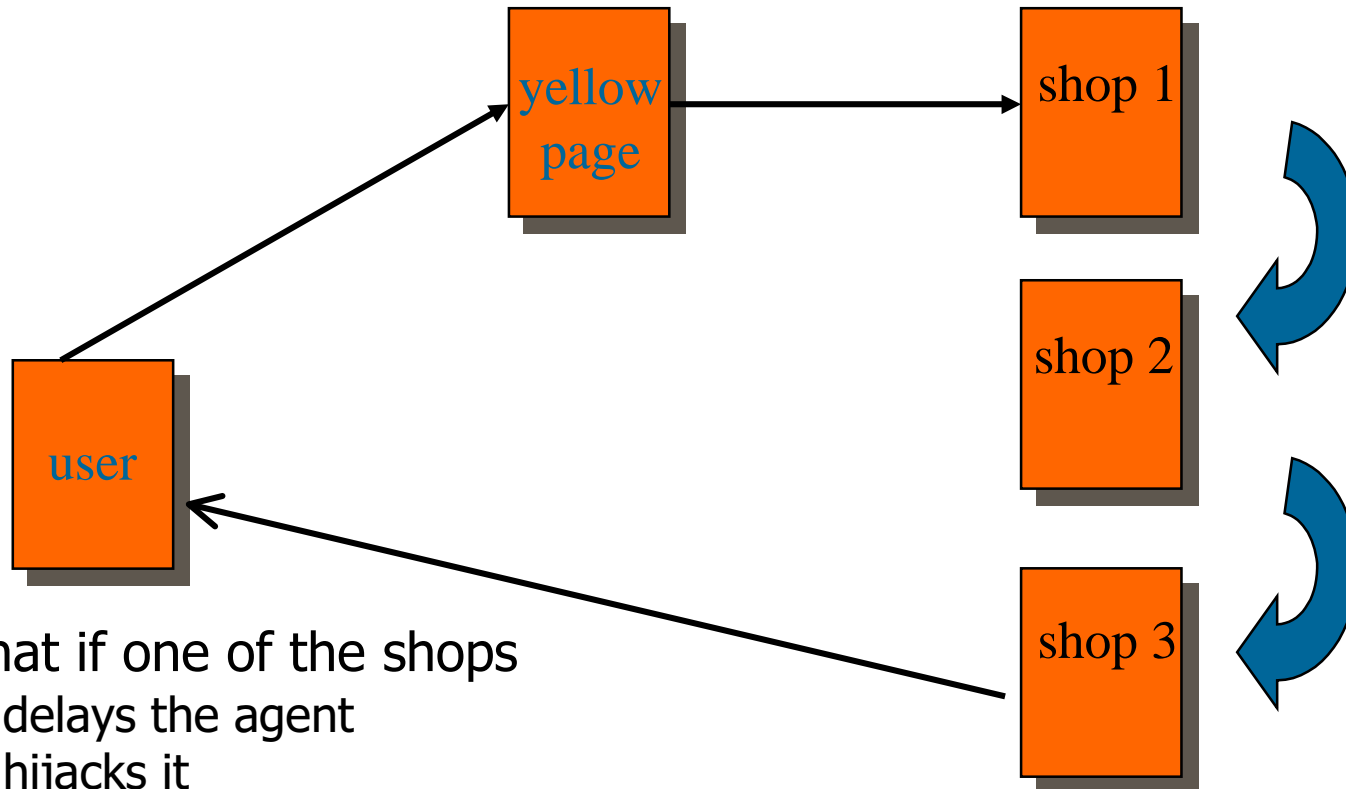
# Another Scenario: Electronic Commerce

The traditional client-server approach





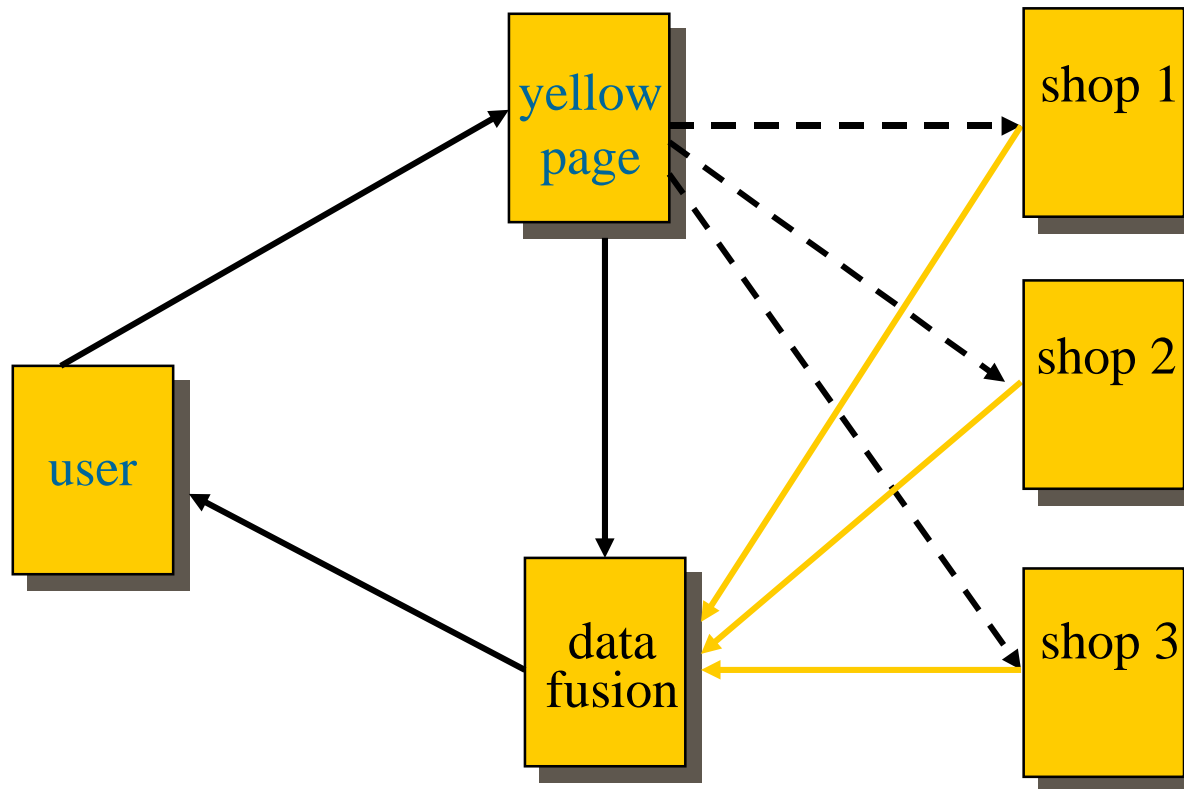
# Electronic Commerce Based on Agent Roaming



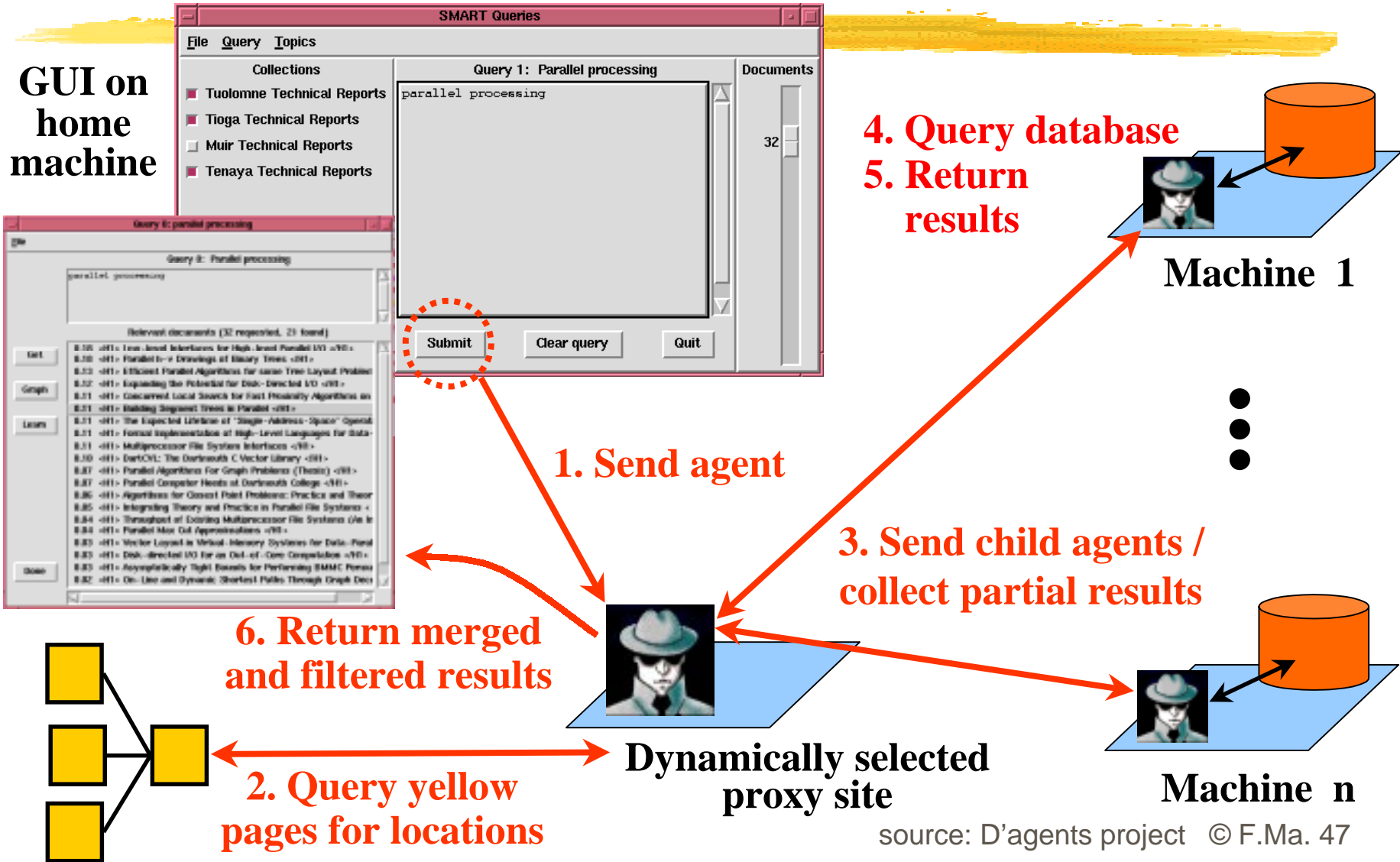
But what if one of the shops

- ▮ delays the agent
- ▮ hijacks it
- ▮ forges the information
- ▮ steals data from other shops?

# Electronic Commerce Based on Agent Cloning



# Example: Technical Reports (D'Agents Project, Dartmouth College)



# Semantic Information Compression with Agents

- Client sends an agent with a **specific search routine** to a server
- Server does remote filtering of its data base according to the search routine
- Only the filtered documents, not the whole data base is sent back to the client
- Similarly, the agent could carry an application specific **compression** algorithm or an **encryption** scheme

# Application Areas of Mobile Agents

## ■ Electronic Commerce

- agent as a seller, buyer, trader, broker...
- send personal agent on a shopping tour

## ■ Information search and retrieval

- remote filtering and compression

## ■ Mobile computing

- migrate-and-disconnect style of operation

## ■ Remote control, diagnosis, maintenance

- monitoring (e.g., network management)
- remote installation of software components

## ■ Watchdogs react locally on behalf of an authority

- my agent at Dow Jones will alert me...

# ... Application Areas

- **Workflow management**, groupware applications
  - active documents with semantic routines to process their content
  - cooperation services (e.g., scheduling of meetings)
- **Personalized Internet services**
- **Entertainment**
  - represent player on a game host
  - distributed multi-user games
- Mobile agents as a **programming paradigm**
  - higher-level abstraction
  - unifies "process" and "object" ("active objects")

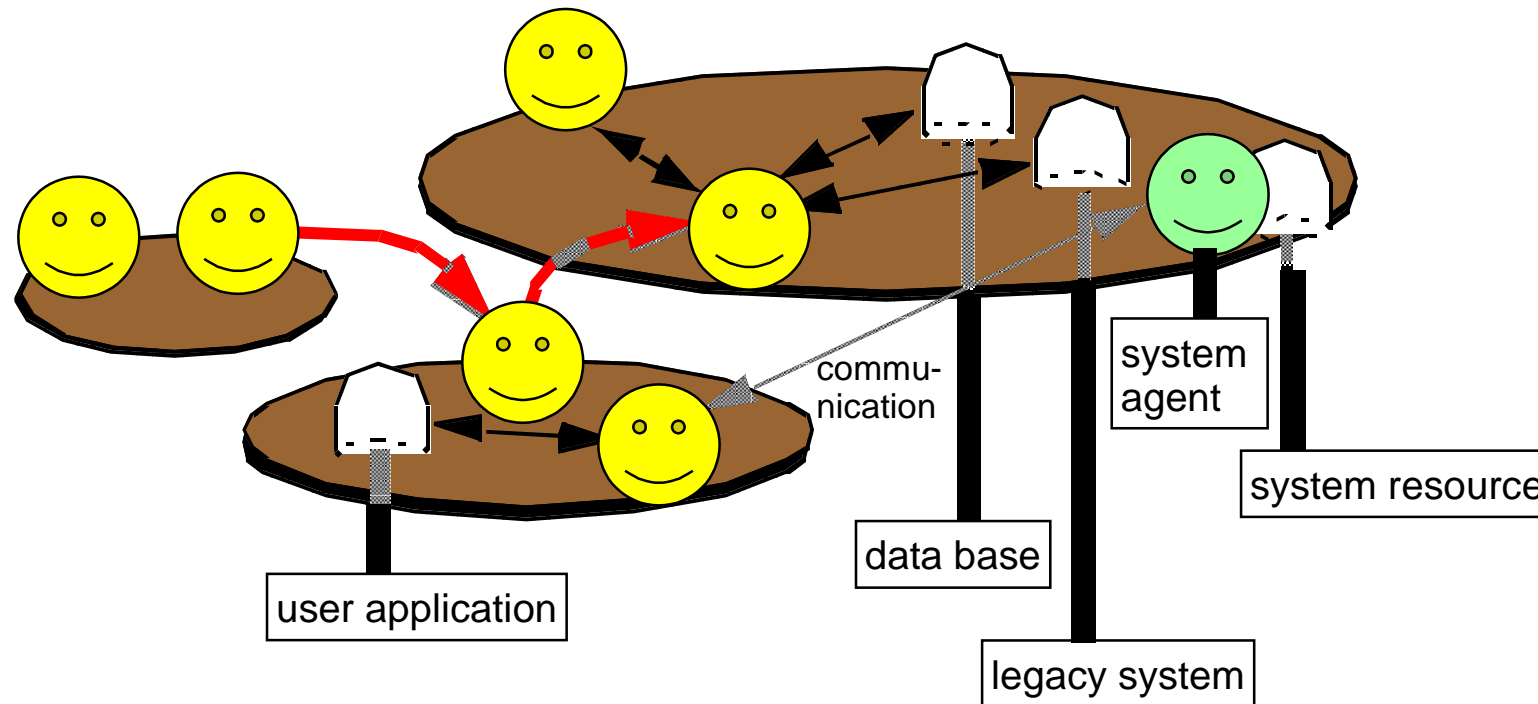
# Outline



- What are agents and mobile agents?
  - mobile code
  - use of mobile agents
  - application areas
- Agent infrastructure and agent platforms
  - migration
  - communication
  - security
- Conceptual problems
- Agent systems
- Challenges and Problems



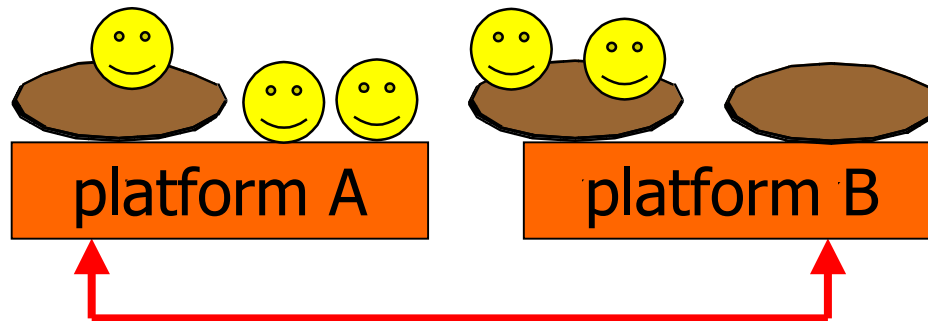
# Agent Places



- **Places** are abstractions of computer nodes
  - local system resources may be “wrapped” by **system agents**
- Allow to distinguish **locality** from **globality**
  - however: one machine may contain many places



# Mobile Agent Platform

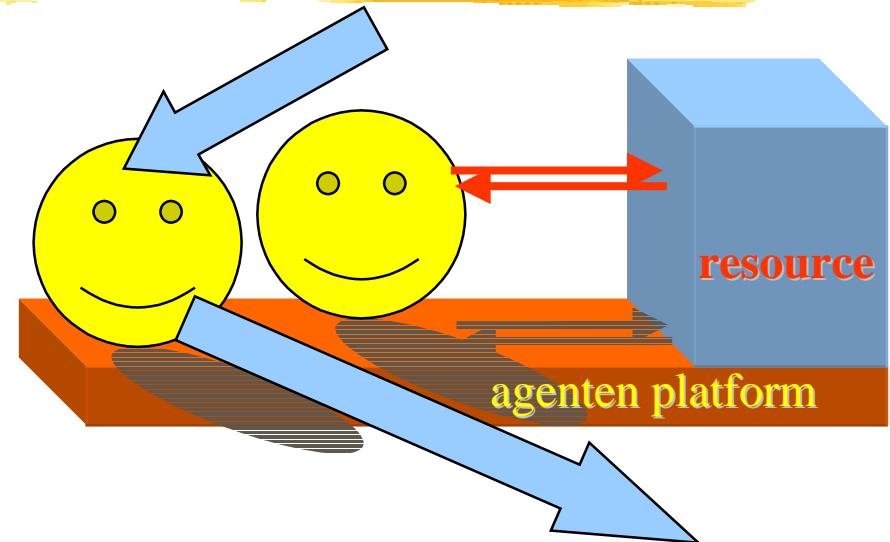


Often, there is no distinction made between places and platforms

- “Docking station” and meeting point for agents
- **Local infrastructure** similar to services of a hotel
  - launching, receiving, providing residence for agents
  - provides services, resources, run-time support to agents
  - e.g. “concierge service” (inform about locally available resources, other resident agents,...)
- Cooperation among platforms (--> **global infrastructure**)
  - forwarding agents, locating agents, communication mediation
  - requires standard **protocols** and **interfaces**

# Local Infrastructure

- Mobility support
- Communication
  - with local / remote agents
  - with „authority“ (user, owner)
  - communication paradigm?
- Take care of security
- Event delivery / management
- Exception handling
- Fault tolerance
  - transactions?
- Resource management
  - service mediation, access rights
  - fair usage (cpu, memory...)



- Can be quite complex
- Requires API or „language“  
e.g., negotiation of local resource usage

# Global Infrastructure

- Migration
  - also: exception handling when goal not reachable
- Locating agents, searching lost agents
  - more than a simple „name service“!
- Forwarding messages
- Termination of agents / groups of agents
  - orphan detection
- Brokering, service directory, yellow pages
- Connection to established system infrastructures
  - CORBA, Jini, WWW, Internet services,...
  - application frameworks and legacy systems
- Fault tolerance?
- Monitoring, management, debugging?
- Interoperability?

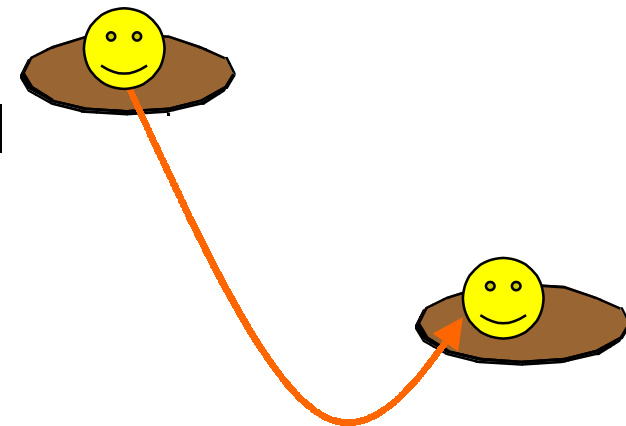
Requires  
cooperation  
among all  
platforms

Can be quite  
complex!

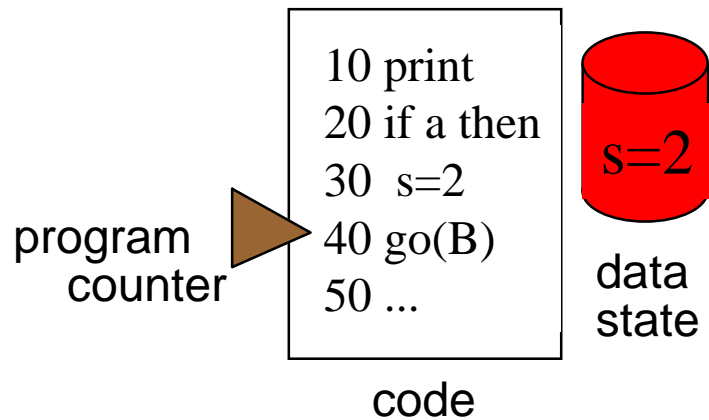
# Migration

- Transport of an agent to another place
- Autonomous
  - agent initiates migration (and decides upon goal)
- *Weak* migration
  - only program **code** is migrated
  - agent restarted at migration goal
- *Strong* migration
  - agent keeps its **context**
  - continues with next instruction

```
x := ...;  
if (x > 17) then go supercomputer;  
y := f(x) + 29;
```



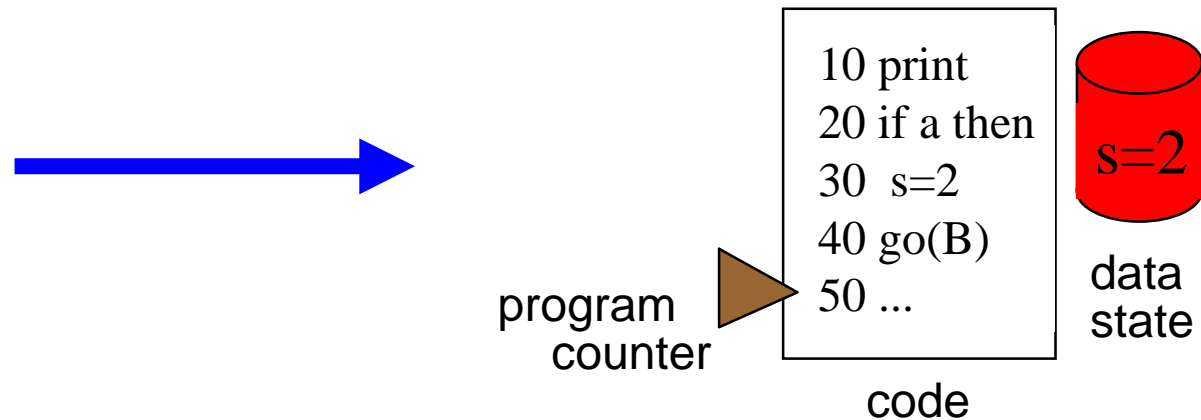
# Strong Migration



- Platform has all data associated to the agent, i.e.
  - code
  - **data state** (contents of variables)
  - **execution state**
    - | program counter
    - | operand stack
    - | heap
    - | calling stack

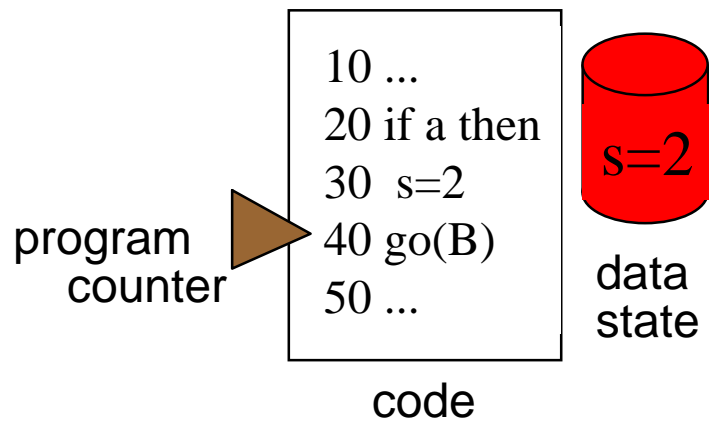
But does the platform know *how* to access all these data?

# Strong Migration



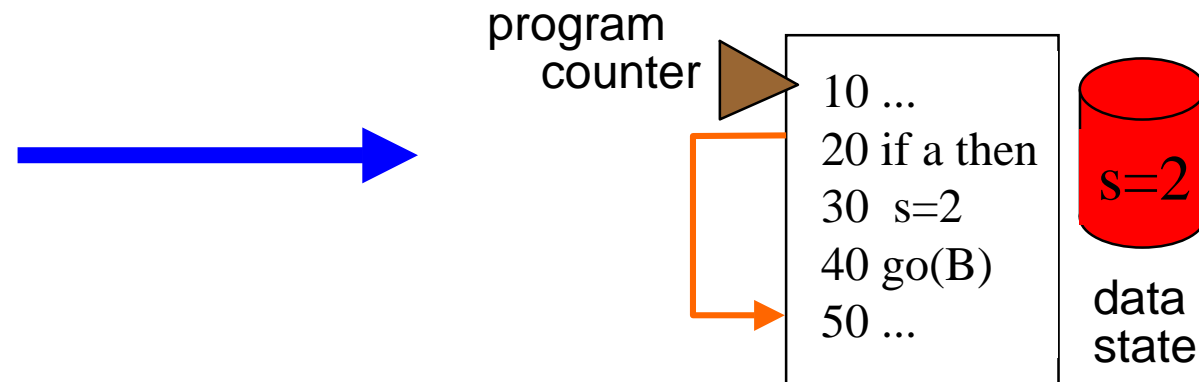
- Platform **serialises** data and **sends** it to next node
- Program continues at **next statement**
- **Transparent** to the programmer
  - question: is it transparent to the *agent*?

# Weak Migration



- Platform does **not** access **execution state** of the agent, but only
  - code
  - data state

# Weak Migration



- Program restarts at some entry-point
  - typically at the **beginning**
- Non-transparent to the programmer
  - manually save important parts from execution state
  - when restarting: **restore** relevant parts of **saved state** and **jump** to a “continuation label”
    - non-trivial for nested control structures



# Strong vs. Weak Migration

## ■ Strong migration

- easier to use (for the agent programmer)
- challenge for the implementor
  - migration in the middle of an expression evaluation when calling a function?
  - can strong migration be realized on top of Java without changing the JVM?

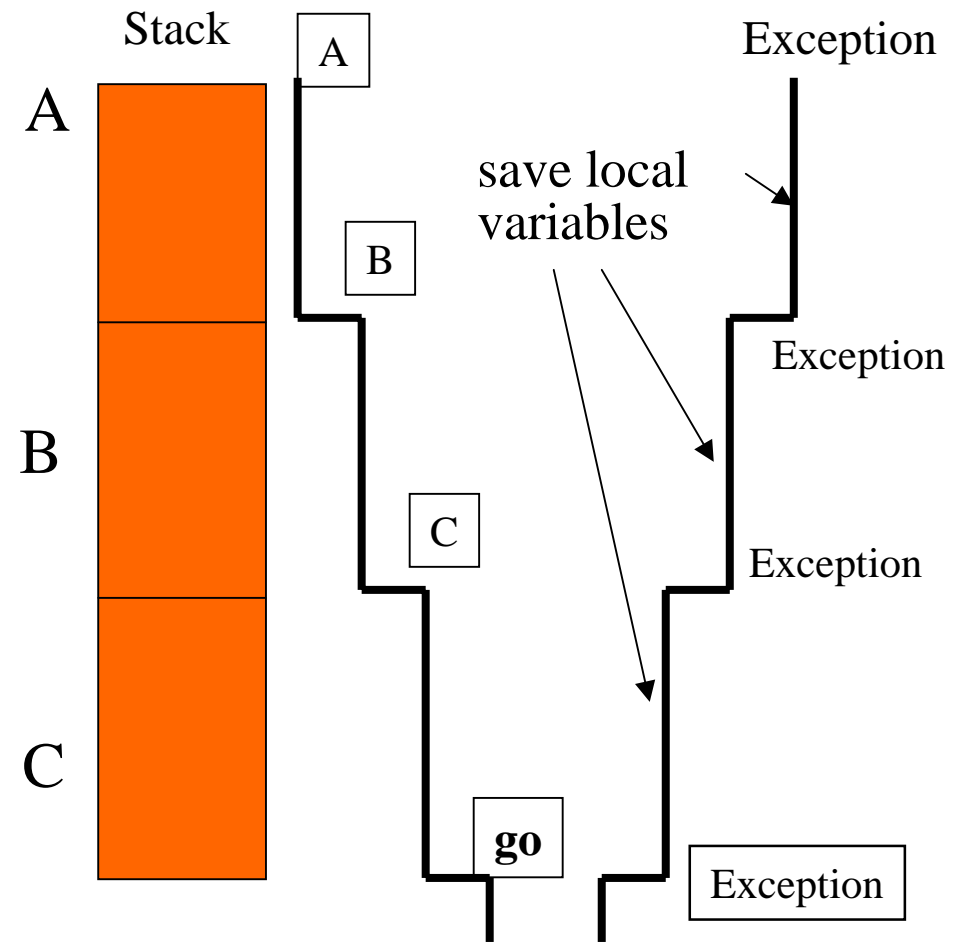
## ■ Weak migration

- easier to implement
- resource rebinding and relevant execution state restoration has to be handled explicitly
- in general more efficient due to less overhead
- probably sufficient in many cases

# Traversing the Java-Run-Time Stack before Migrating

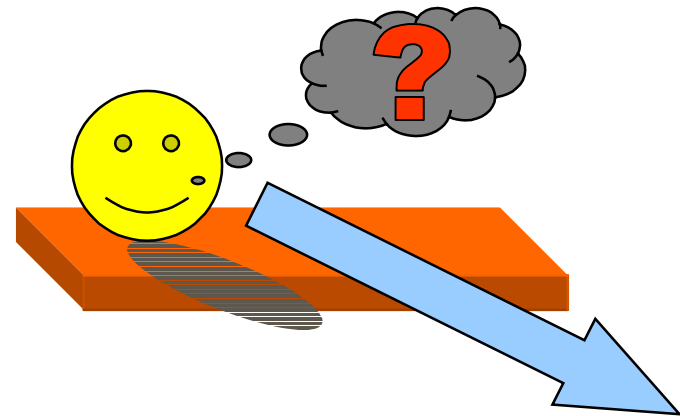
- In Java it is in principle possible to gain access to the stack without changing the JVM
  - 'go' recursively throws special exception:

```
try {  
    go(location);  
}  
catch(MigrateException mig) {  
    <save values of local variables  
    in some global save area object>  
    throw mig;  
}  
catch ...
```



# Why Migrate?

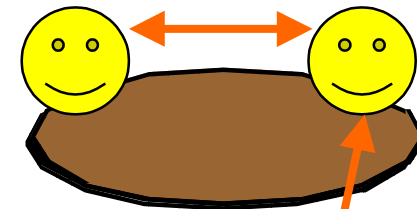
- Another machine has unique resources
- Plan dictates migration
  - e.g., roaming shopping agents
- Event initiates migration
  - e.g., agent called home
- Local platform has become suboptimal
- Local platform will be disconnected or cease to exist



# Communication Between Agents

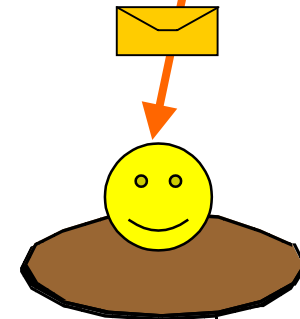
- For agents at the **same place** (“meetings”)

- procedure call
- method invocation **call by visit**
- ...



- Communication between **distant agents**

- RPC (or RMI)
- messages (mailboxes at agent places?)
- abstract global information spaces
  - e.g., tuple spaces



- **Address** of target agent?

- global name (name service required!)
- <name of platform> + <local id>

Languages for higher-level communication?  
- semantics?  
- ontologies?

# Communication with Migrating Peers?



- How to **find** the peer?
- What to do if an agent **migrates during communication**?
- Approaches: **location service, forwarding chains**

# Location Service

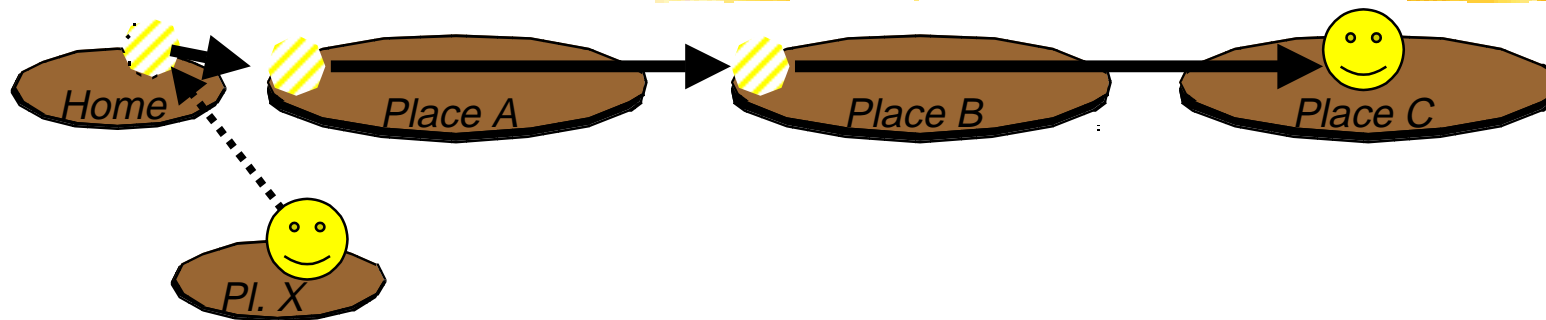
- Agent informs location service about new location when migrating
- Sender may ask location service for “current” location of receiver

- Problems:

- atomicity of location information update?
- location while in transit?



# Message Forwarding with Proxies



- Agent leaves a **path of proxies** that reference to next target place
- Communication is done via proxies
- Proxies **relay communication** to next proxy
- **Problems:**
  - possibly long forwarding chains
  - message may never catch a fast moving agent

# Outline



- What are agents and mobile agents?
  - mobile code
  - use of mobile agents
  - application areas
- Agent infrastructure and agent platforms
  - migration
  - communication
  - security
- Conceptual problems
- Agent systems
- Challenges and Problems



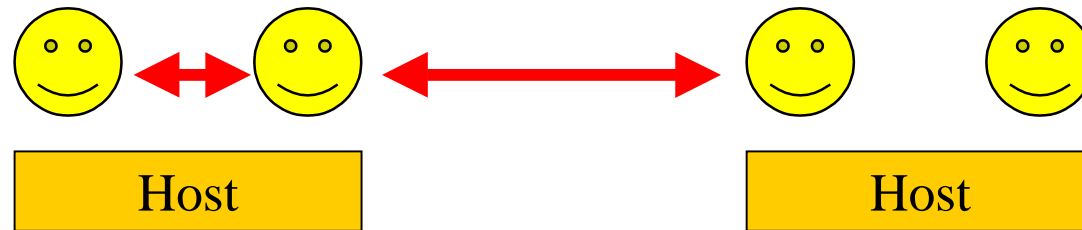


# Security is Important

- We want **open mobile agent systems**, i.e.
  - everybody can deploy mobile agents
  - everybody can operate a host
  - everybody can offer services
- **Electronic commerce** is an important target
  - money is involved
  - security is important

Security is a key factor  
for agent technology!

# Agent-Agent Security



## ■ Some attacks:

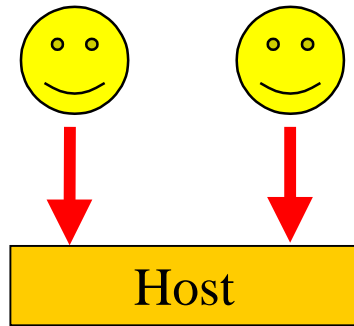
- manipulation of other agents
- masking the identity
- denial-of-service

## ■ Measures:

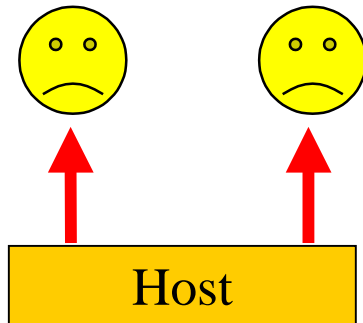
- access control
- authentication
- not a mobility problem

👉 Existing techniques can be applied!

# Host-Agent Security



- Protection of **hosts** from malicious agents
  - that's the **easier** part: well-known techniques
  - malicious agent similar to a virus



- Protection of **agents** from malicious hosts
  - **difficult**

Security needs depend on **operational issues**  
closed system, open system?

# (1) Closed System

- **Agents** can be inserted only by **trusted** users
- **Services** are offered only by **trusted** users
- **Agent system** is operated by **trusted** party
- Typical applications
  - network and system administration
  - parallelization of computations
- Security needs: system has to be **shielded** from outside access

„Trust“ is an important issue!

## (2) Open System



- Agents can be inserted by anyone
- Services can be offered by anyone
- Agent system can be operated by anyone
- Applications
  - any, as long as security is manageable
- Security needs: protection of agents and hosts

# (3) A Compromise: Trusted Services and Platforms



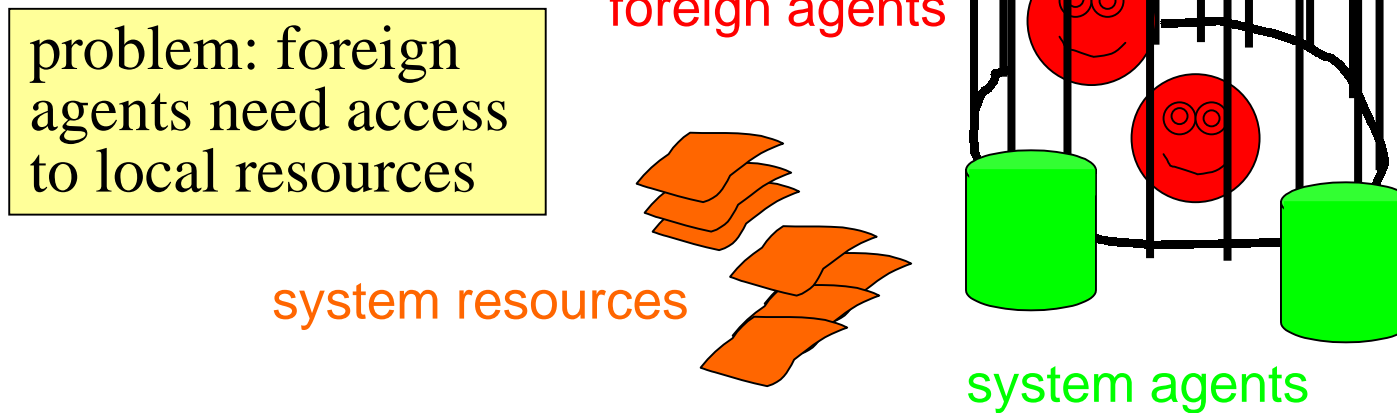
- | **Agents** can be inserted by **anyone**
- | **Services** can be offered only by **trusted** users
- | **Agent system** is operated by **trusted** party
- | Many applications, but
  - | no open service market (electronic commerce!)
  - | bad integration of legacy systems
- | Security needs
  - | **agent system** has to be **protected** from malicious agents
  - | service provision has to be controlled

# Protection of **Hosts** from Agents

- **Code signing** with cryptographic means
  - authentication: the receiver will **trust the sender**
  - belief that trusted sources behave correctly
- **Code verification**: the host may try to examine the code of the agent
  - only of limited applicability
    - what properties can be verified automatically?
  - proof carrying code?
    - proof (that some safety property is not violated) is delivered with code; receiver verifies it (which is easier than proving)
- **Access-level control**
  - security manager checks whether activities are allowed
  - sandbox model

# Protection of Hosts from Agents

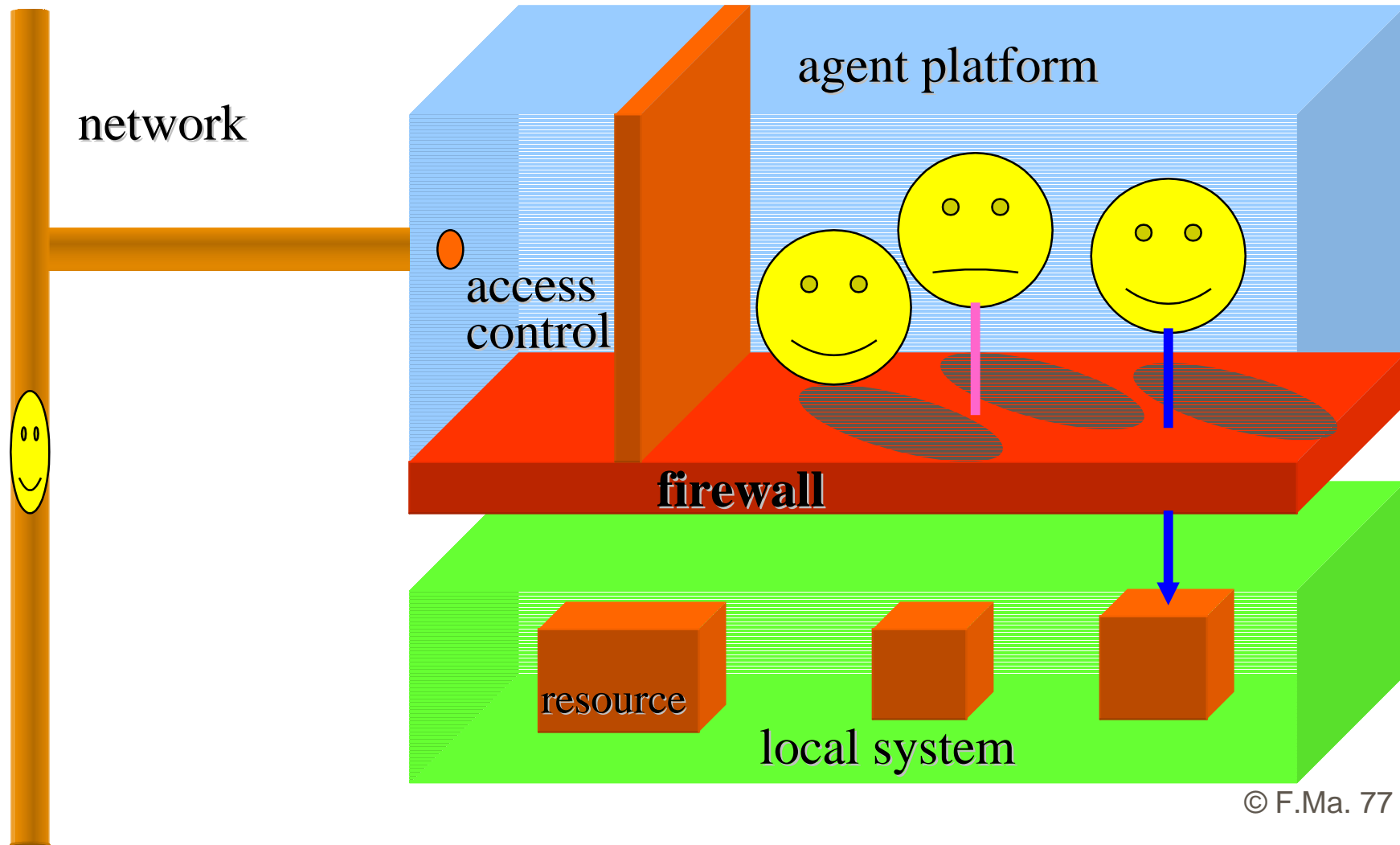
- „Cage approach“
  - e.g., Java sandbox model



- Allow only controlled access to system resources
  - negotiate consumption limits
  - record agent activities



# Access Control and Local Firewall



# Protection of **Agents** from Malicious Hosts, Example:

Agent shall **buy flowers** for its authority:

1. find all flower shops
2. migrate chain of shops and get price
3. go to cheapest shop and buy flowers

```
Address home = mx17@mypda.eac.eu
Money wallet = 20$
float maximumprice = 20$
good flowers = 10 red roses
int shoplistindex = 0
float bestprice = 20$
Address bestshop = empty
```

**prevent**  
modification  
reading  
reading

modification  
modification  
modification

# A Malicious Host Attack



Malicious host may **modify** the **code** and set bestshop to its shop:

```
if (shoplistindex >= (shoplist.length - 1)) {  
  go(bestshop);  
  buy(flowers,wallet);  
}
```



go(**MYSHOP**);



# Possible Attacks of Malicious Hosts

- Reading or modifying data or code
- Masking the identity of the host
- Incorrect execution of code
- Denial of execution or communication
- Reading/modifying communication with others
- Returning wrong results to system calls
- Wrong routing, no forwarding
- Cloning an agent
  - undecidable whether agent is a clone!

Agent cannot decide

- whether its code has been copied
- whether it has been cloned
- whether it is being executed correctly
- ...

Are there solutions?

# Protection of Agents ?

- Thesis: Since the host has complete access to the agent, it is **not possible to protect an agent from the host**
  - agent must expose its code and data to the host which supplies the means for the agent to run
- But perhaps:
  - **obfuscate** or **encrypt** the agent code so that the host doesn't know what it is doing?
    - what does this mean and when does it help?
    - prevent at least reverse engineering?
  - use **trusted hardware**?



# Tamper-Proof Environments (TPE)

- „Hardware blackbox“
  - **secure execution environment** for agents
  - **trust** (of the agent owner) towards **TPE manufacturer**
- Agent is **encrypted** with the public key of the TPE
  - TPE decrypts the agent using its private key and starts it
- Only **crucial parts** of the agent are kept in the TPE
- Probably limited to **security-sensitive domains**
  - e.g., banks, stock market

# Measures Against Some Attacks

- The “trivial” solution: visit only trusted sites
- Determine whether the agent has been tampered
  - code signing
  - maintain a safe migration history (encrypted traces, audit logs); also use it against rerouting attacks
- Partition roaming agents
  - one component for each server visited
  - each server encrypts its component with the public key of the agent’s authority
  - components are chained together (e.g., using hashes of earlier components)
  - decryption only on home base or on some other trusted site

- More difficult if there are collusions of several malicious hosts
- Some measures induce communication with other hosts: this reduces autonomy and efficiency

# Outline



- What are agents and mobile agents?
  - mobile code
  - use of mobile agents
  - application areas
- Agent infrastructure and agent platforms
  - migration
  - communication
  - security
- Conceptual problems
- Agent systems
- Challenges and Problems





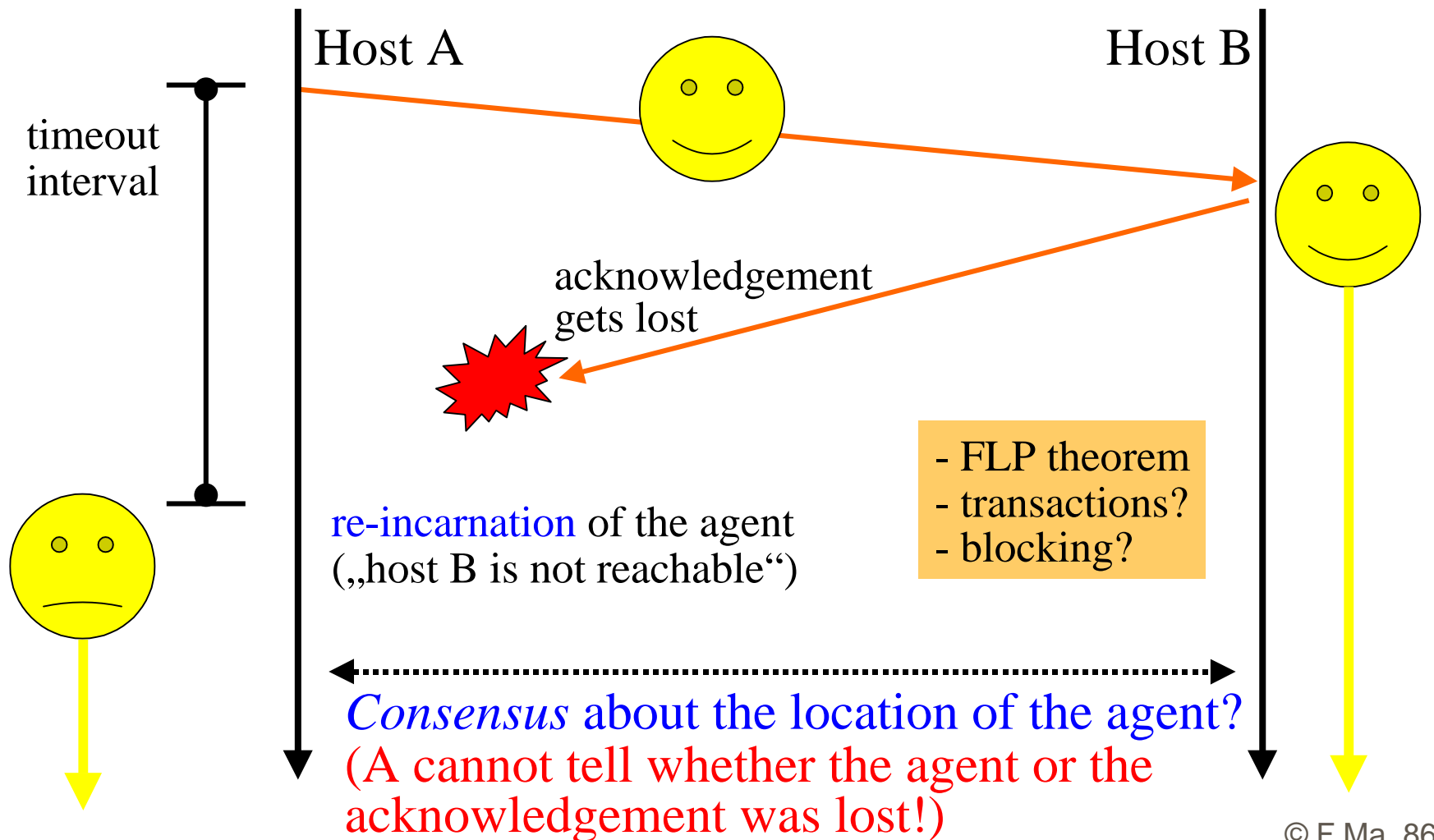
# Conceptual Problems



- Exactly once migration?
- Hiding a secret?
- Semantics of mobility?

Many serious applications (such as electronic commerce) will only be realized with mobile agents when these problems (and other important problems such as security) are well-understood and solved!

# Exactly-Once Migration?



# Can an Agent Keep a Secret?

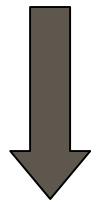
- Keep **electronic money** that cannot be copied?
- Execute a **strategy**, but **without disclosing** it to the host?
  - e.g., my mobile agent on the stock market server
  - it is the host that executes every single instruction of the agent, so the host „knows“ the agent code!
- **Mobile cryptography?**
  - encrypted programs which are equivalent to the original programs produce encrypted output (e.g.,  $A^{-1} \circ f() \circ A$ )
  - still a research issue (Sander & Tschudin, 1997), unclear whether useful in practice
  - also: blackbox security (Hohl, 1997)

# Secret Strategy?

## An Oversimplified Example

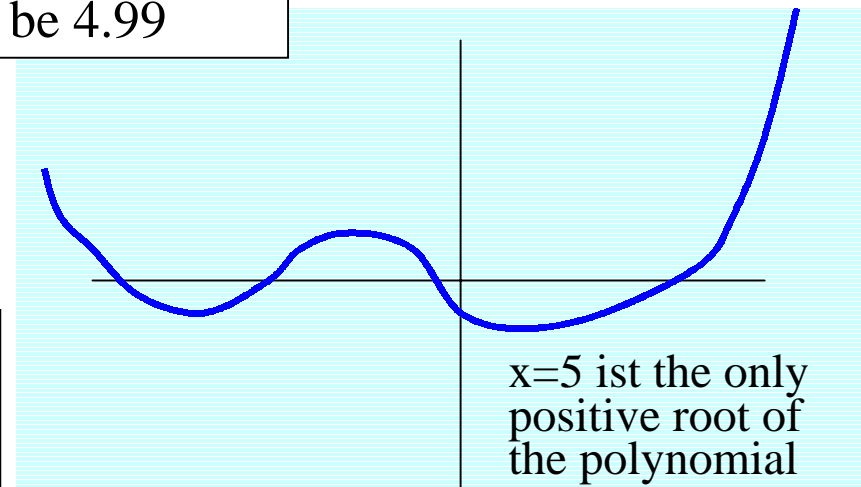
```
x = ask_for_price(...);  
if (x > 5) then go to A;  
else go to B;
```

„right“ alternative;  
my price offer  
should be 4.99



equivalence transformation  
to hide the secret strategy

```
x = ask_for_price(...);  
if (x**4+6*x**3-19*x**2-144*x)  
> 180 then go to A;
```



One-way principle:

- equivalent expression easy to construct
- but difficult to „solve“ (and to understand!)

But:

- when is that useful?
- what exactly is a „secret“?

# Semantics of Mobility?

- What is the **formal meaning** of „go to nextnode;“?
  - is it just „skip“?
  - is migration a purely nonfunctional issue?
- What exactly is the **context** of an agent?
- What parts of the context will be moved with the agent?
  - use same or only „equivalent“ local resource?
  - will `print(„hello“)` go to the local or remote display?
  - where will exceptions and error messages be delivered?
  - will events be forwarded?
- What happens in the case of an **error**?
  - e.g., migration goal not reachable
  - agent lost
  - resource not available at new location
  - local library for dynamic linking differs
  - source code no longer available with „copy by need“

All these are only  
*some* issues that  
have to be addressed!

# Outline



- What are agents and mobile agents?
  - mobile code
  - use of mobile agents
  - application areas
- Agent infrastructure and agent platforms
  - migration
  - communication
  - security
- Conceptual problems
- Agent systems
- Challenges and Problems



# Implementation Issues



- Heterogeneity of machines and systems
- Interoperability?
- Should make use of established systems, e.g.
  - WWW protocols
  - Transport protocols and Internet infrastructure
  - security infrastructure
- Complete system is a major undertaking!

# Mobile Agent Systems and Projects

- Serious development since about 1994
- List at University of Stuttgart contains (in February 2000) about 60 mobile agent systems
  - <http://mole.informatik.uni-stuttgart.de/mal/mal.html>
  - 50% Europe, 30% USA, 20% Asia
- Most are research prototypes
- Typically not interoperable with other systems

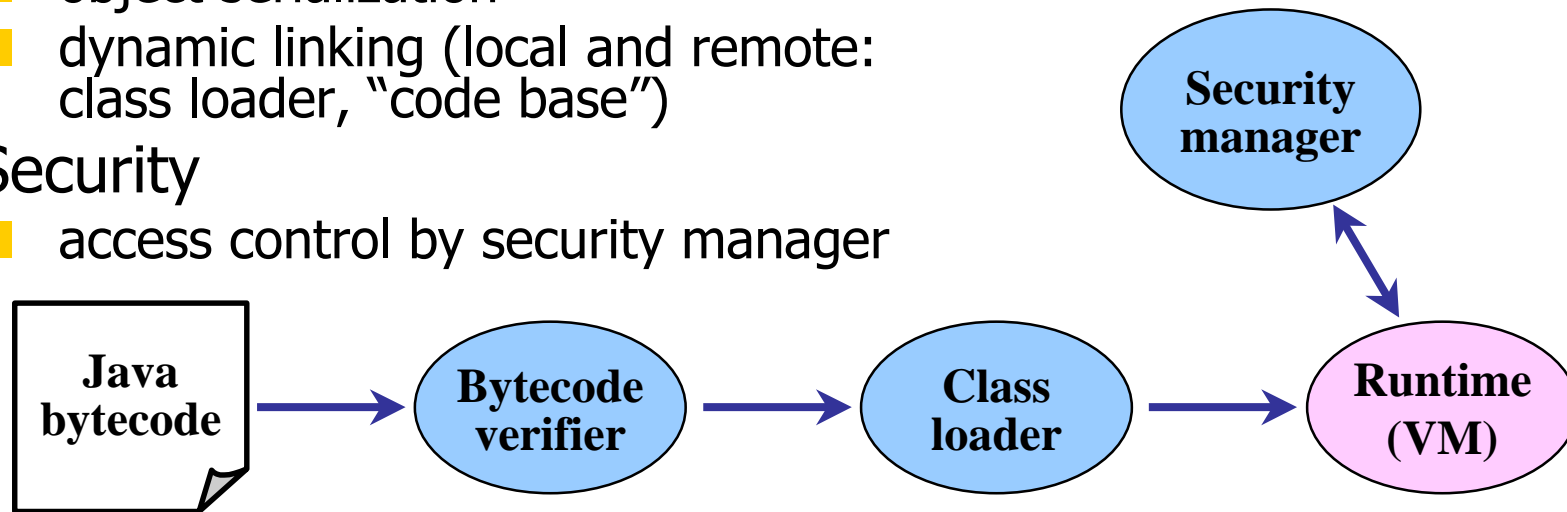


# Programming Language?

- Agents typically migrate between **heterogeneous systems**
- Agents therefore should be written in a „universal“ language
- Most **scripting languages** qualify for that
  - e.g., Tcl, Perl
  - interpreters are available for most systems
- Interpreters can easily influence the execution
  - e.g., filter system calls or trap illegal instructions
- Drawback: interpreted languages are less efficient
- **Java-Bytecode** is also available on most machines
  - interpreted by Java Virtual Machine (JVM)
  - generated by Java compiler from Java source code

# Why Java in Most Projects?

- Available on “all” machines
- Large set of standard libraries and tools
- Portability: compiled into bytecodes for a stack-based virtual machine
- Code mobility (“code on demand”)
  - object serialization
  - dynamic linking (local and remote: class loader, “code base”)
- Security
  - access control by security manager



# Telescript



General Magic <http://www.genmagic.com/>

- First real mobile agent system
  - ~ 1994
  - no longer available
- Object-oriented language similar to Java and C++
- Strong mobility
  - agents interact at places („meet“ instruction)
- Telescript is compiled into bytecodes for a RISC virtual machine
- Persistent store

# Aglets



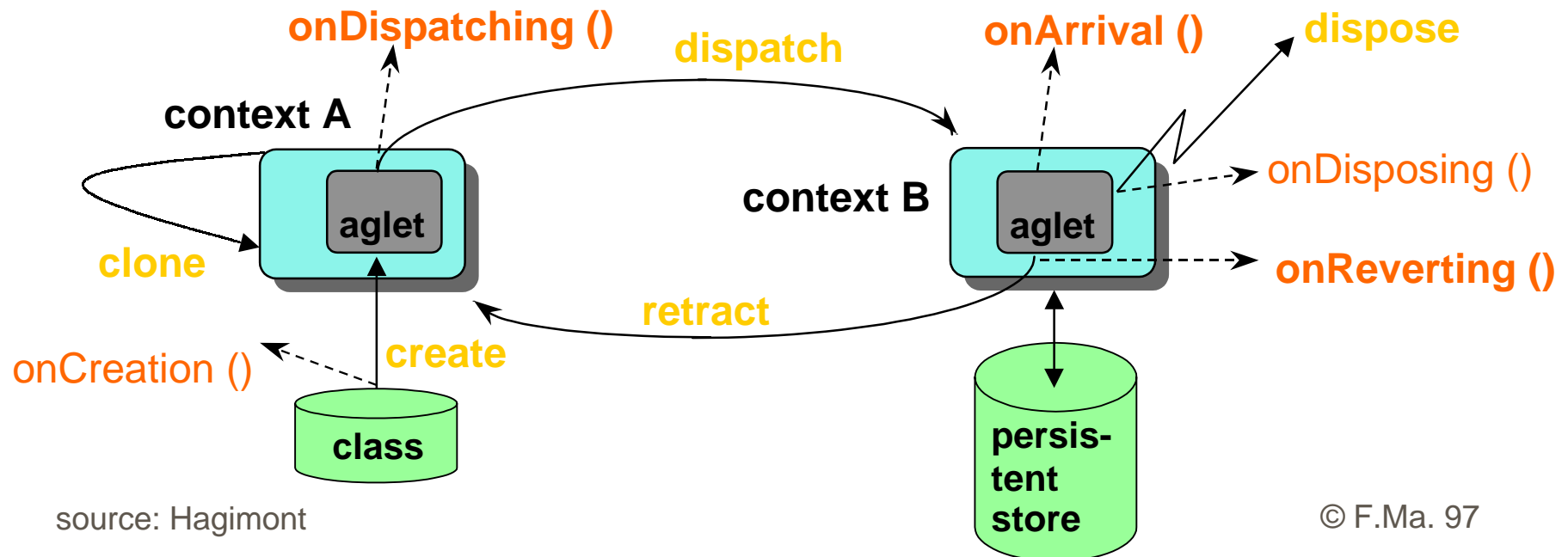
**IBM**      <http://www.trl.ibm.co.jp/aglets/>

- Java-based Aglet-API
- Mobile agent = class that extends *Aglet*
- Graph of serializable objects
- Weak mobility
- Proxies for location transparency
- Persistent store

# Aglets: Life Cycle and Event-driven Programming Model

■ Event handler for „mobility events“ associated to an aglet (*MobilityListener*)

- onArrival (MobilityEvent I)
  - onDispatching (MobilityEvent I)
  - onReverting (MobilityEvent I)
- } to be implemented by the programmer



# Aglets: Mobility Programming

```
class myListener implements MobilityListener {
    public void onDispatching (MobilityEvent I) {
        System.out.println ("I am leaving!");
    }
    public void onReverting (MobilityEvent I) {
        System.out.println ("I am going home!");
    }
    public void onArrival (MobilityEvent I) {
        System.out.println ("I have arrived!");
    }
}

public class MyAglet extends Aglet {
    public void onCreate (Object init) {
        MobilityListener listener = new myListener();
        addMobilityListener(listener);
    }
}
```

# The Aglet API



- Predefined methods (used by programmer)
  - | Aglet.dispatch (URL url)
  - | Aglet.deactivate (long time)
  - | Aglet.clone ()
  - | Aglet.getAgletContext ()
  - | ...
- Methods to be implemented by the programmer
  - | Aglet.onCreate (Object init)
  - | Aglet.run ()
  - | Aglet.HandleMessage (Message msg)
  - | Aglet.onDisposing ()
  - | ...

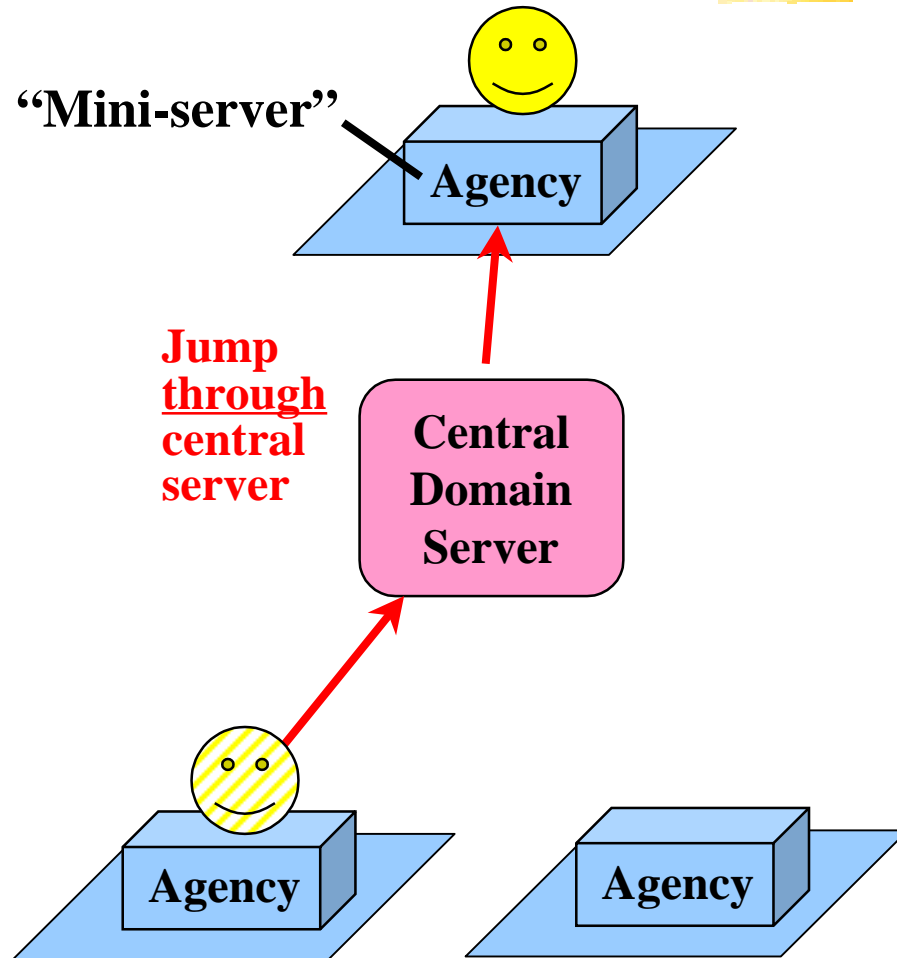
# Messages in Aglets



- Consists of two fields
  - a type (*String*)
  - an object
- May be transmitted in three different ways
  - **synchronous** call
    - `AgletProxy.sendMessage (Message msg)`
  - **asynchronous** call
    - `AgletProxy.sendAsyncMessage (Message msg)`
  - **oneway** (asynchronous)
    - `AgletProxy.sendOnewayMessage (Message msg)`



# Jumping Beans



Ad Astra Engineering

<http://www.JumpingBeans.com/>

- Java
- Weak mobility
- Central server
  - for tracking, managing and authenticating agents
  - but also failure point and bottleneck
- Persistent store

# Mole



Fritz Hohl et al., University of Stuttgart

- Java
  - Weak migration
  - Communication:
    - messages
    - RPC
    - sessions
  - Visual monitor
- Two types of agents
    - **user agents**
      - mobile
      - may not use system resources except CPU, memory, network
    - **system agents**
      - immobile (started by platform operators)
      - may access any system resource

# Voyager



VOYAGERPRO™

ObjectSpace

<http://www.objectspace.com/products/voyager/>

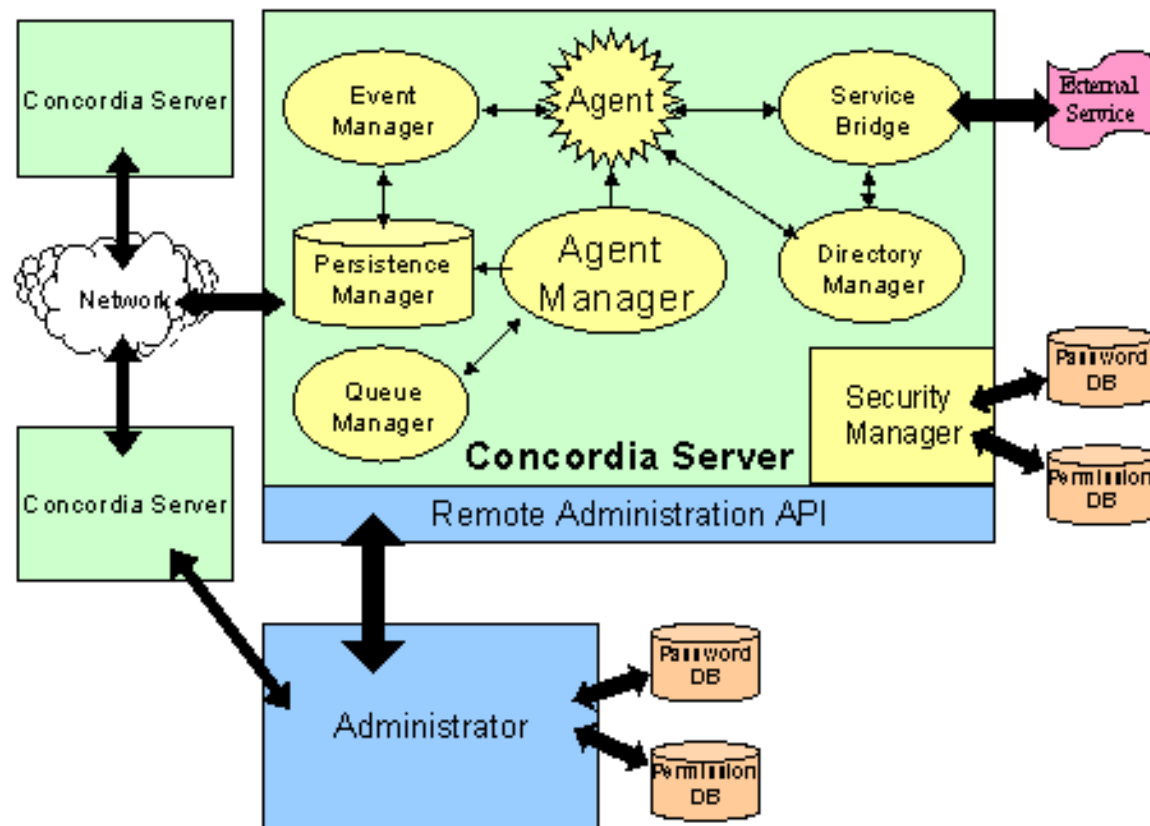
- Java
- Built on top of CORBA
- Weak mobility
- Persistent store
- Group communication (multicast)

# Concordia

## Mitsubishi America

### ■ Java

- implementation language
- agents programmed in Java and executed on server with JVM



# Concordia



## ■ Security:

- user identification
- agent authentication (credentials), security and integrity
- resources protection

## ■ Mobility:

- queue Manager (inbound and outbound) for reliable transport of agents
- Java object serialization scheme

## ■ Communication:

- TCP/IP protocol
- lightweight agent transport API (application embedded)
- inter-agent communication manager (registration, posting and notification of event)



# D'Agents

- Multiple languages

- Tcl
- Java
- Scheme

- Support services

- directory service
- tracker
- debugger (Tcl)

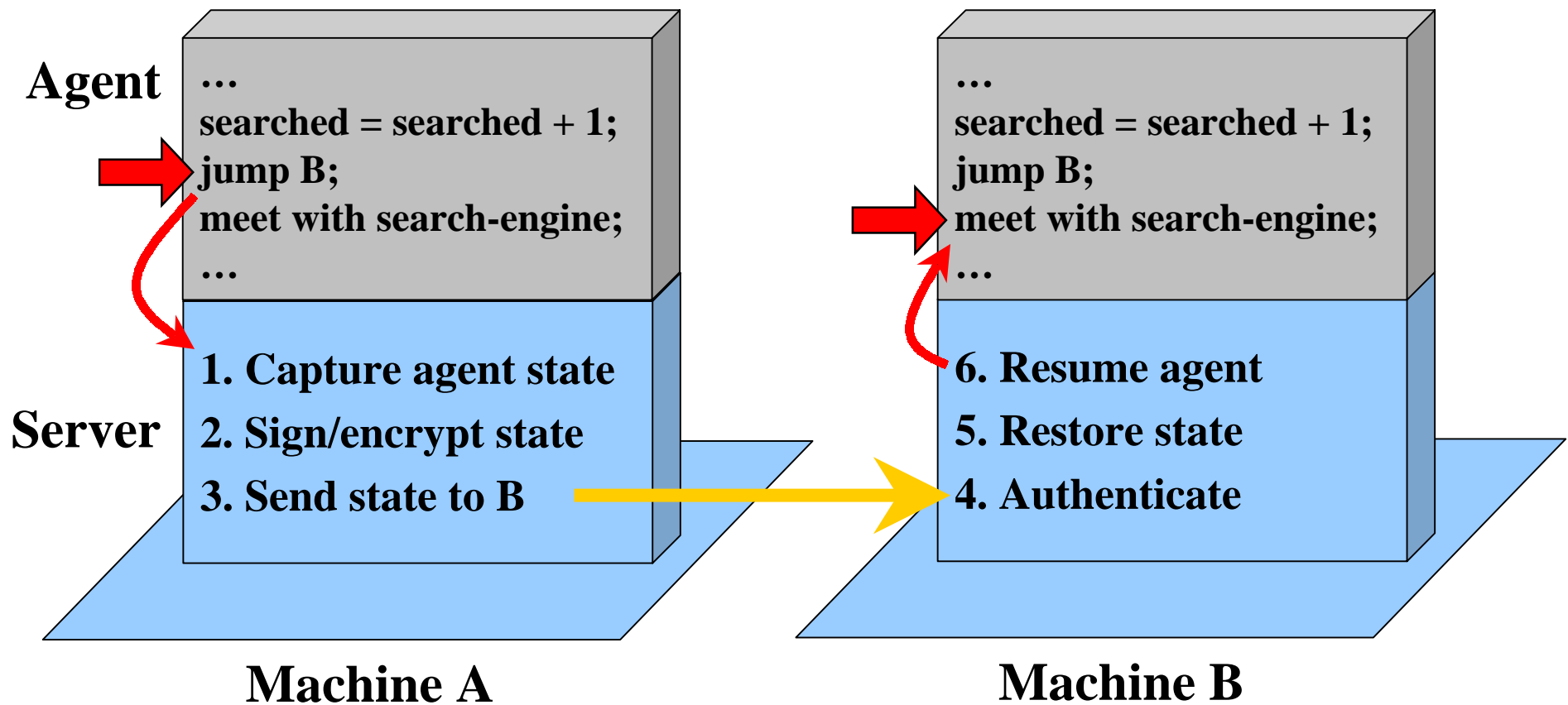
- Security

- agent protection while in transit
- no agent protection while on a machine

**Dartmouth College**

<http://agent.cs.dartmouth.edu>

# D'Agents: Strong Mobility



# D'Agents: Other Features



## **Meetings**

Second communication mechanism (direct connection between agents)

## **Select**

Wait for a message, meeting request, data arriving over a meeting, etc.

## **Status reports**

Which agents are running on machine A?  
Who owns agent X?

## **Notifications**

Message from server on agent birth or death

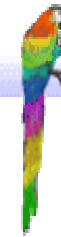
## **Event-driven programming**

Event handlers for all types of incoming communication



# Ara

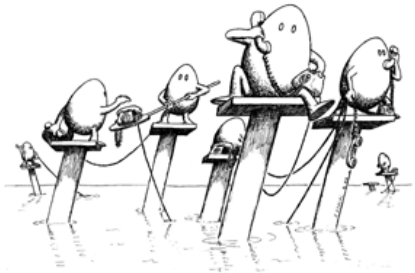
Ara



[http://www.uni-kl.de/AG-Nehmer/Projekte/Ara/index\\_e.html](http://www.uni-kl.de/AG-Nehmer/Projekte/Ara/index_e.html)

- C/C++, Java and Tcl
- **Strong mobility**
- C/C++ compiled into bytecodes for RISC virtual machine
- Server plus all agents inside one Unix process

# Obliq



**DEC Research (Compaq)**

**<http://www.research.digital.com/SRC/Obliq/Obliq.html>**

- Weak mobility
- Limited security (access checks)

# Tacoma



University of Tromsø / Cornell University

<http://www.tacoma.cs.uit.no:8080/TACOMA/>

- C, Tcl/Tk, Scheme, Python, Perl
- Weak mobility
- Single, simple abstraction: meet
  - easy to add a new language
  - less opportunity for optimization

# WASP

## (Web Agents for Service Providing)

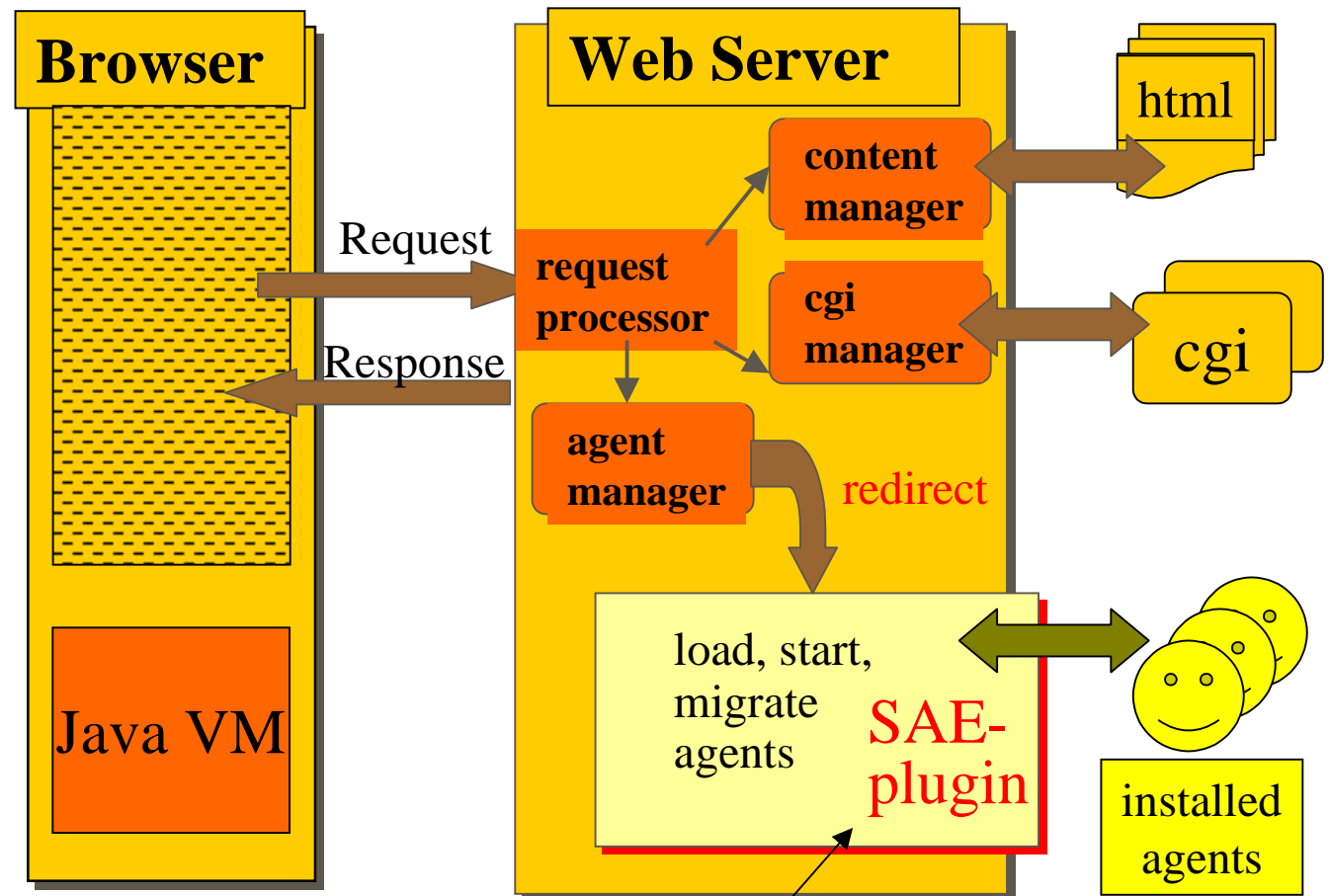


TU Darmstadt

- Java
- WWW integration
  - WWW contains all the relevant information in the Internet
  - extended WWW server as an agent platform
  - applets in WWW browsers as GUI for agents
  - HTTP for agent transfer (MIME encoded)
  - easy deployment of the agent platform using the WWW

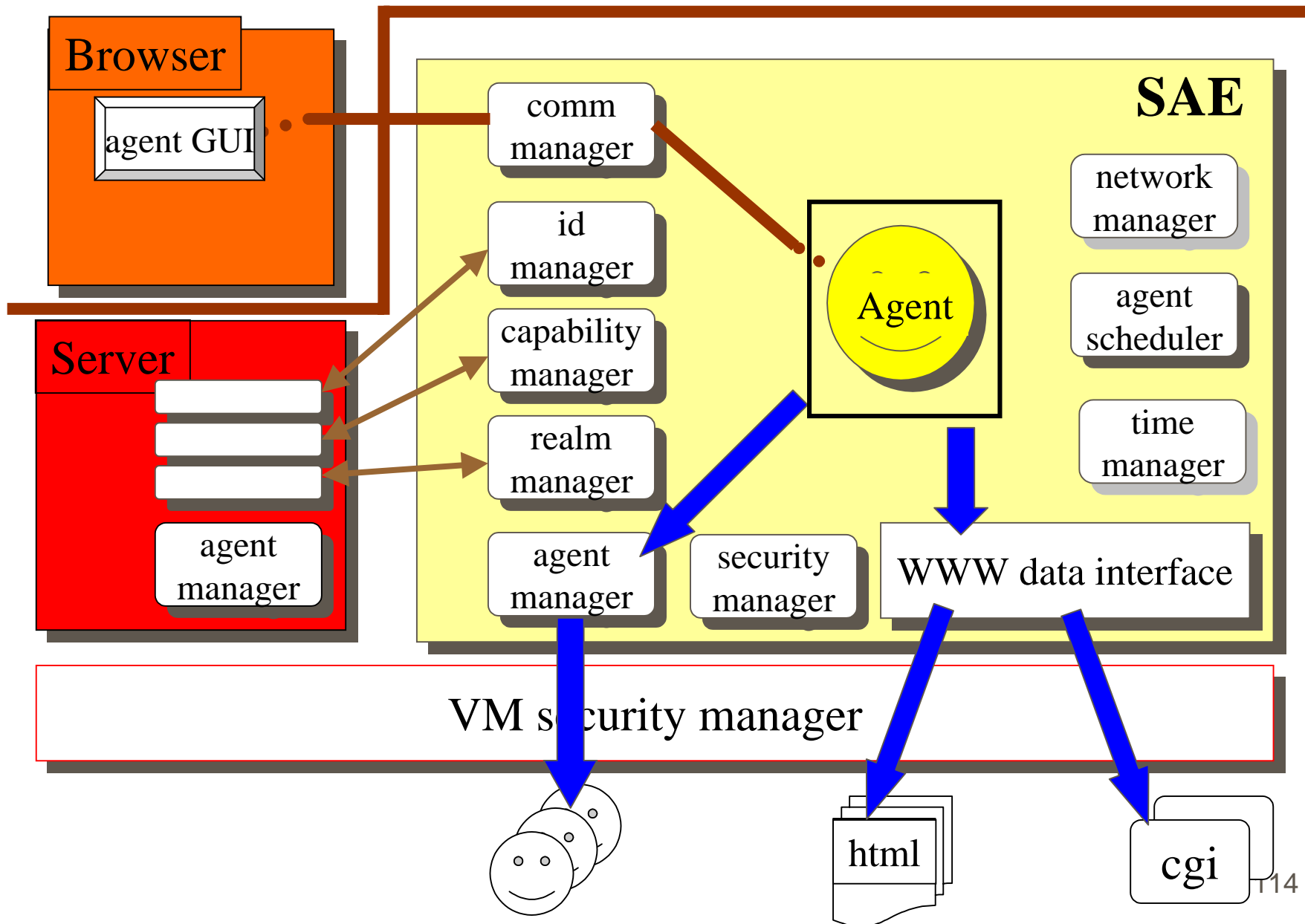
# Architecture of the WASP Server

- Agent is **addressed** via the URL of the WWW server where it was created
- Agent leaves behind **Proxy** with new address when migrating away
- **Creation** initiation of agents from an arbitrary WWW browser
- **Browser** is also the GUI of the agent



SAE = Server Agent Environment © F.Ma. 113

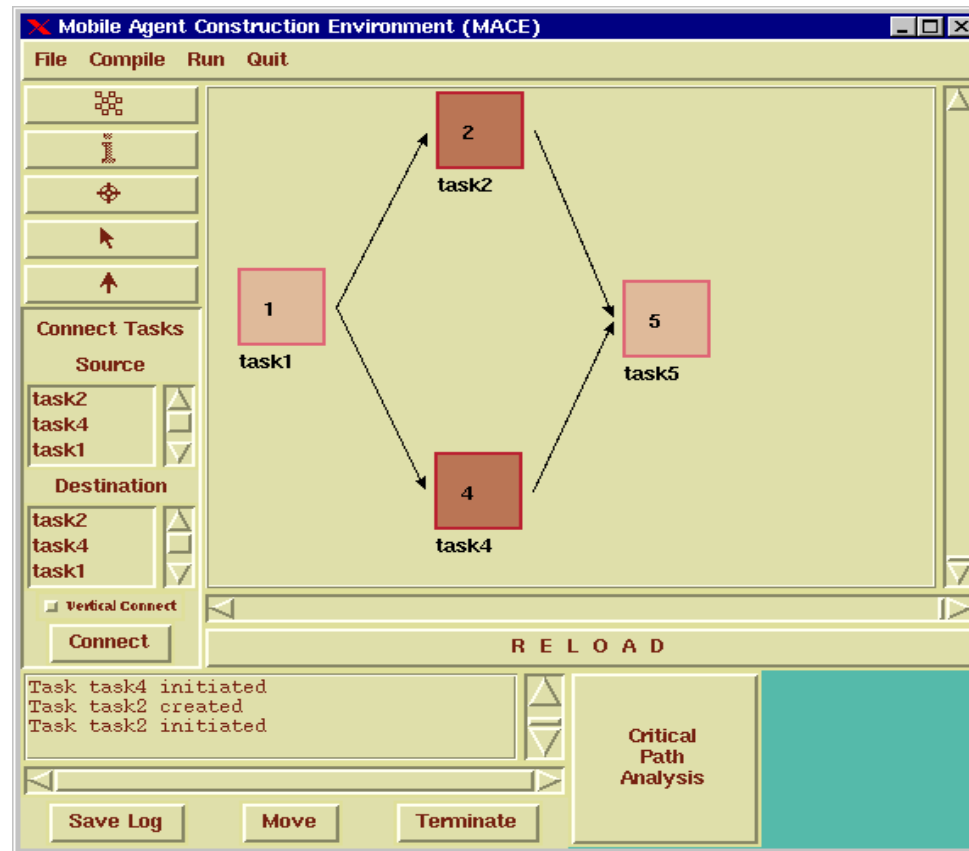
# WASP: SAE Architecture



# Programming Mobile Agents

- How can a user of an agent system **program an agent**?
  - a rarely addressed issue, although there exist **some prototypes** of interactive tools realized within some projects
- How can an agent system developer or a platform provider **monitor and control the system**?
  - specific monitoring and debugging tools are almost non-existent
  - note that monitoring and debugging distributed systems is a **difficult problem** (in theory and in practice), and mobility only adds new difficulties!

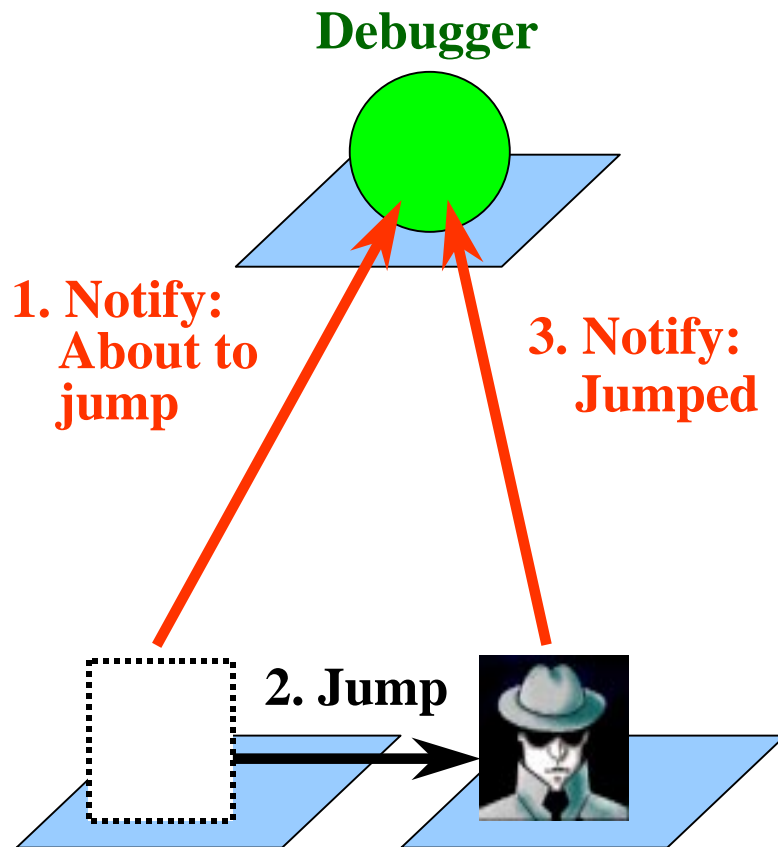
# MACE: Mobile-Agent Construction Environment



(D'agents project)



# Debugging (D'agents project)



agent\_1

Agent in file "who.tk" is running.  
Machine name: muir.cs.dartmouth.edu  
Machine IP: 129.170.192.42  
Agent name:  
Agent id: 97

1

File Buffers Breakpoints Watch Defaults Search

```
# jump from machine to machine
foreach m $machines {
    # if we do not jump successfully append an error message
    # otherwise append the user list
    if {[catch "agent_jump $m" result]} {
        append list "$m:\unable to JUMP to this machine ($result)\n\n"
    } else {
        set users [exec who]
        append list "$agent(local-server):\n$users\n\n"
    }
}
return $list
}
```

Currently viewing file: "who.tk"

step next continue kill interrupt

grint values  
evaluate  
up stack frame  
down stack frame

```
climb down stack frame -->
procedure name = who
arg 1 = machines = muir.cs.dartmouth.edu tenaya.cs.dartmouth.edu
tuolomne.cs.dartmouth.edu tioga.cs.dartmouth.edu

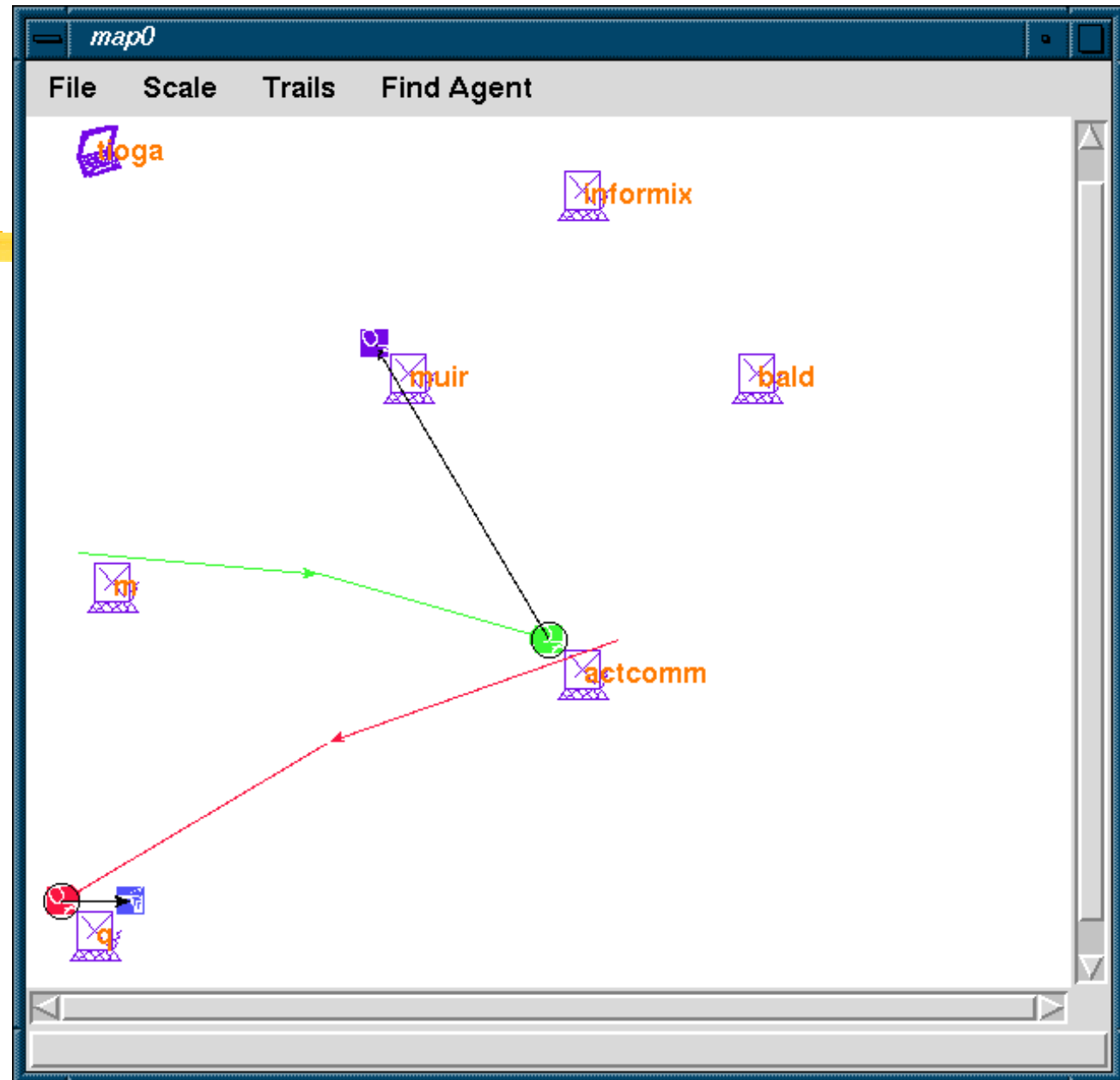
Continue running on bald.cs.dartmouth.edu 129.170.192.98 5 {}.
About to jump to muir.cs.dartmouth.edu.
Jump succeeded.
***** breakpoint number 2, run number 1 *****

Break before jump to tenaya.cs.dartmouth.edu in proc "who"
Break on line 31 of file "who.tk"
Argument values at invocation of procedure "who" -->
arg 1 = machines = muir.cs.dartmouth.edu tenaya.cs.dartmouth.edu
tuolomne.cs.dartmouth.edu tioga.cs.dartmouth.edu
```

# Tracking

## D'Agents project

- Interactively or via logs
- Current work: Merge with debugger
- Current work: Zoom in and out



# Mobile Agent Standards

- **OMG** Object Management Group **MASIF (1998)**
  - “Mobile Agent System Interoperability Facility”
  - based on CORBA (ORB, IIOP,...)
  - defines some mobile agent systems aspects
  - some areas not addressed (e.g., migration)
- **FIPA** Foundation for Intelligent Physical Agents **(1999)**
  - defines intelligent and mobile agent issues
  - proposes KQML/KIF for communication
  - only two implementations
- Standards are not yet widely accepted

# Mobile Agent Systems are Complex



- Takes time and effort to develop
- Paradigm seems to be attractive and nice
  - too many naïve pure research projects?
- Seems easy at first sight (after all, Java applets already exist) when ignoring:
  - **infrastructure** needed for real-world applications
  - interaction with the **rest of the world**
  - **fundamental problems** (fault tolerance, security...)
  - **pragmatics** (performance, e-commerce realities, simpler alternatives than mobile agents...)

# Challenges and Problems

- Security!
- Fault tolerance (transactional support,...)
- Legacy system integration
- Interoperability; wide and easy deployment
  - build upon WWW infrastructure (e.g., platforms as browser plugins)?
- Global infrastructure
  - semantic routing
  - directories, location service, brokers,... (Jini, CORBA?)
  - common ontology (= vocabulary + meaning)
- Support of small, lightweight devices

# Challenges and Problems (2)

- Agent control
  - trace and manage agents (suspend, restart, terminate)
  - orphan detection etc.
  - overpopulation by junk agents?
- Efficiency
- Go beyond prototype systems
- Market for agents?
  - services in an open Internet-based service market
  - value added service providers
- Intelligent mobile agents?
  - also: languages for agent cooperation

# Conference !

2nd Int. Symp. on **Agent Systems and Applications** + 4th Int. Symp. on **Mobile Agents**

**ASA/MA 2000**

**September 13-15, ETH Zurich**

[www.inf.ethz.ch/ASA-MA/](http://www.inf.ethz.ch/ASA-MA/)

Deadline for paper submission: **March 10, 2000**

# Conference !



2nd Int. Symp. on [Agent Systems and Applications](#) + 4th Int. Symp. on [Mobile Agents](#)

[ASA/MA 2000](#)

[September 13-15, ETH Zurich](#)

[www.inf.ethz.ch/ASA-MA/](http://www.inf.ethz.ch/ASA-MA/)

Deadline for paper submission: [March 10, 2000](#)



# Mobile Agents Literature



James White:

**Telescript Technology: The Foundation for the Electronic Marketplace.** General Magic White Paper, 1994

**Telescript Technology: Scenes from the the Electronic Marketplace.** General Magic White Paper, 1994

**Telescript Technology: An Introduction of the Language.** General Magic White Paper, 1995

*The classical papers*

David Chess, Colin Harrison, Aaron Kershenbaum: **Mobile agents: Are they a good idea?**. In Jan Vitek; Christian Tschudin (eds.): Mobile Object Systems: Towards the Programmable Internet, pages 25-45. Lecture Notes in Computer Science No. 1222. Springer-Verlag, April 1997. <http://www.research.ibm.com/massive/mobag.ps>

*Fundamental early mobile agents paper*

# Mobile Agents Literature (2)

Danny Lange, Mitsuru Oshima: **Programming and Deploying Java Mobile Agents with Aglets**, Addison-Wesley, 1998

Baumann, F. Hohl, K. Rothermel and M. Straßer (1998): **Mole - Concepts of a Mobile Agent System**, World Wide Web, Vol. 1, Nr. 3, pp. 123-137

[http://www.informatik.uni-stuttgart.de/cgi-bin/ncstrl\\_rep\\_view.pl?inf/ftp/pub/library/ncstrl.ustuttgart\\_fi/TR-1997-15/TR-1997-15.bib](http://www.informatik.uni-stuttgart.de/cgi-bin/ncstrl_rep_view.pl?inf/ftp/pub/library/ncstrl.ustuttgart_fi/TR-1997-15/TR-1997-15.bib)

## *Specific mobile agent systems*

Milojicic, Breugst, Busse, Campbell, Covaci, et al: MASIF: **The OMG Mobile Agent System Interoperability Facility**, in: Rothermel; Hohl (eds.): Proceedings of the Second International Workshop on Mobile Agents, MA'98, Springer-Verlag, 1998

<http://www.fipa.org/spec/fipa99spec0.2.htm>

## *Mobile agents and standards*

# Mobile Agents Literature (3)



Giovanni Vigna (Ed.): **Mobile Agents and Security**. Springer-Verlag, 1998

*A good collection of papers on an important subject*

Stefan Fünfroeken, Friedemann Mattern: **Mobile Agents as an Architectural Concept for Internet-based Distributed Applications**.

In: Steinmetz (Ed.): Proc. KiVS'99, pp. 32-43, Springer-Verlag, 1999

*Also introduces the WASP project*

Fritz Hohl: **Tutorial "Mobile Agents and Active Networks"** at Smartnet'99, 1999

*Some slides of this tutorial are based on Fritz Hohl's tutorial*

# Mobile Agents: Resources



## ■ General Resources

<http://www.cs.umbc.edu/agents/>

[http://www.cetus-links.org/oo\\_mobile\\_agents.html](http://www.cetus-links.org/oo_mobile_agents.html)

## ■ The Mobile Agent List

<http://mole.informatik.uni-stuttgart.de/mal/mal.html>

## ■ Mobile Agent Security Bibliography

<http://mole.informatik.uni-stuttgart.de/security.html>

## ■ Mobility Mailing List

<http://mobility.lboro.ac.uk/>

## ■ Agents Mailing List

<http://www.cs.umbc.edu/agentslist/>

**Friedemann Mattern**

ETH Zürich

[www.inf.ethz.ch/~mattern](http://www.inf.ethz.ch/~mattern)

[mattern@inf.ethz.ch](mailto:mattern@inf.ethz.ch)