بسم الله الرحمن الرحیم

سیستم‌های چندعاملی

درس ۲۷

عامل‌های سیار

Mobile Agents

کاظم فولادی قلعه
دانشکده مهندسی، پردیس فارابی
دانشگاه تهران

http://courses.fouladi.ir/mas

عامل سیار

MOBILE AGENT

عامل سیار *Mobile Agent*	عاملی که قادر است خودش (برنامه و حالت خودش) را در سرتاسر یک شبکه‌ی کامپیوتری انتقال دهد و اجرا در یک سایت راه دور را پیشنهاد دهد.

روی‌کردهای جاری برای محاسبات توزیع‌شده

CURRENT APPROACHES TO DISTRIBUTED COMPUTING

روی‌کردهای جاری برای محاسبات توزیع‌شده		
مدل کلاینت ـ سرور *The Client-Server Model*	**مدل کد–در–صورت–تقاضا** *Code-on-Demand*	**روی‌کرد سرویس‌های وب** *Web Services Approach*

☺ **مزایا**: امنیت
☹ **معایب**: ـ ترافیک شبکه ـ دیرکرد ـ سربار منابع

عامل‌های سیار: روی‌کرد جایگزین

MOBILE AGENTS: AN ALTERNATIVE APPROACH

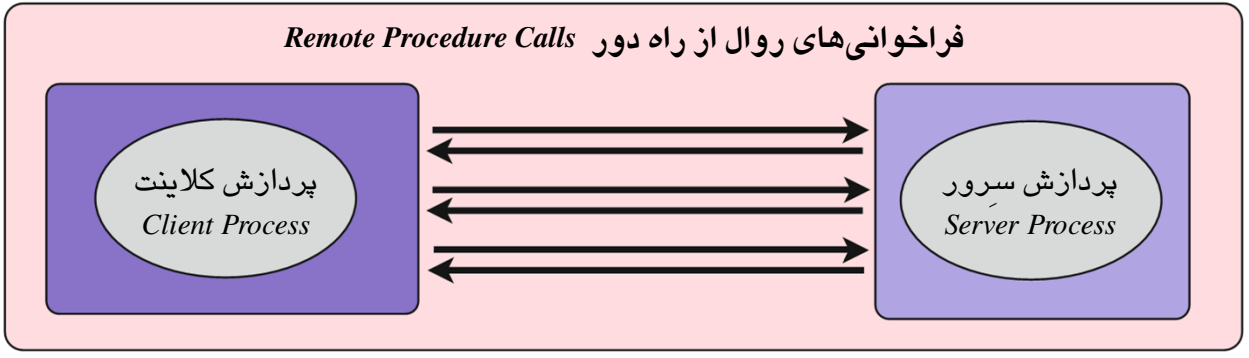
عامل سیار
Mobile Agent

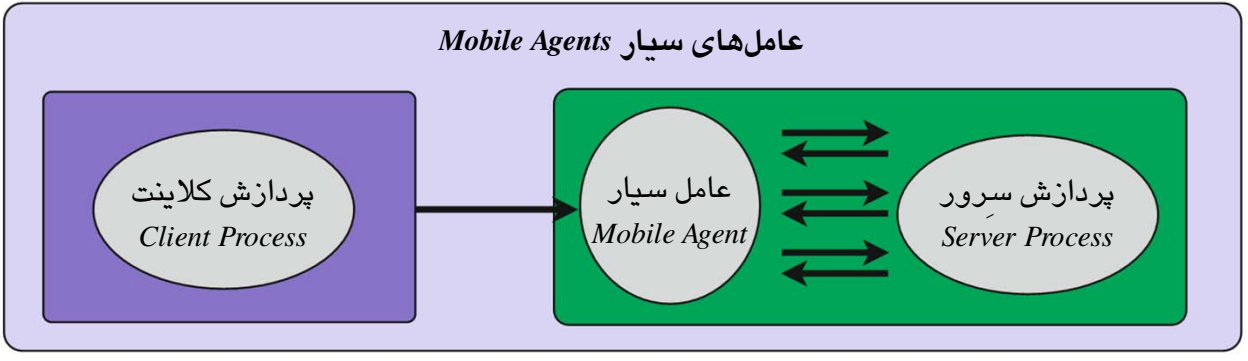
عامل سیار یک موجودیت اجرا کننده‌ی خودمختار است که
توانایی مهاجرت از یک ماشین به ماشین دیگر در یک شبکه‌ی ناهمگن را دارد
و می‌تواند اجرای خودش را از سر بگیرد.

عامل سیار می‌تواند:

- ترافیک شبکه را کاهش دهد.
- بر دیرکرد شبکه غلبه کند.
- از منابع بهتر استفاده کند.
- به‌صورت ناهمگام و خودمختار اجرا کند.
- به‌صورت پویا وفق پیدا کند.
- دارای قوام و تحمل‌پذیر نقص باشد.

مقایسه‌ی «فراخوانی‌های روال از راه دور» با «عامل‌های سیار»





چرا عامل‌های سیار؟

WHY MOBILE AGENTS?

شبکه‌های دارای پهنای باند پایین/ نرخ داده‌ی محدود

استفاده‌ی کارآمد از منابع شبکه

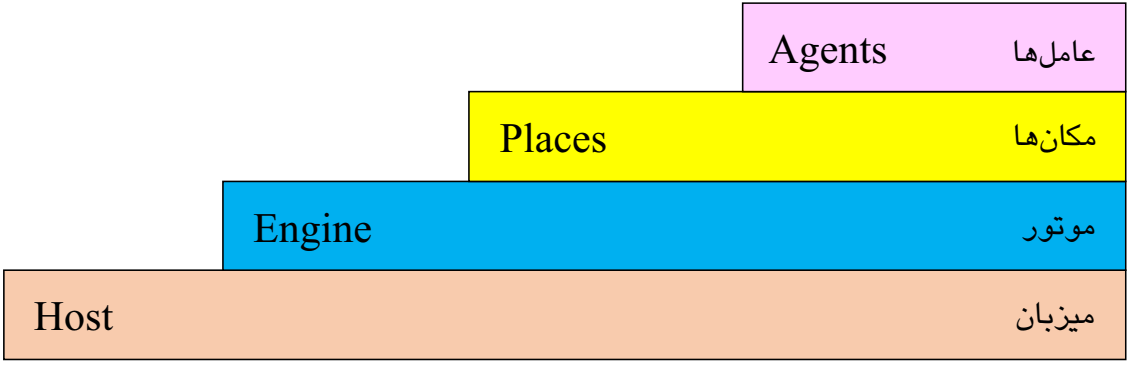
عامل‌های سیار

ملزومات زیرساختی

MOBILE AGENTS: INFRASTRUCTURE REQUIREMENTS

برای پیاده‌سازی یک عامل سیار نیاز داریم به:

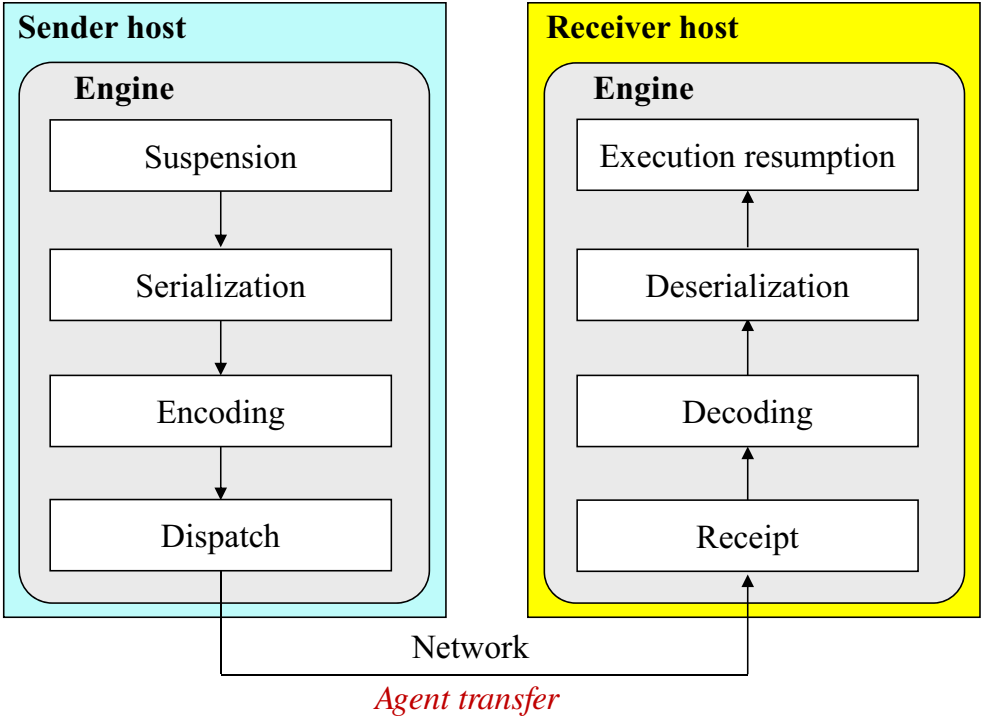
- یک زبان مناسب و قابل انتقال که کد عامل در آن نوشته خواهد شد.
- یک موتور یا مفسر برای آن زبان
- پروتکل‌های ارتباطی که اجازه می‌دهند موتورها روی ماشین‌های مختلف، عامل‌ها را مبادله کنند.



عامل‌های سیار

مهاجرت

MOBILE AGENTS: MIGRATION



عامل‌های سیار

مهاجرت: شیوه‌ها

MOBILE AGENTS: MODES OF MIGRATION

شیوه‌های مهاجرت
Modes of Migration

مدل سیر پُر حالت / قوی *The Statefull or Strong Mobility*	مدل سیر بی حالت / ضعیف *The Stateless or Weak Mobility*
Go Model	***Known Entry Point Model***
○ حالت شیئ، کد و حالت کنترلی عامل گرفته می‌شود. ○ اجرا در ماشین جدید اجازه دارد از محل دقیقی که در آن متوقف شده است، ادامه یابد.	○ حالت شیئ، کد و حالت کنترلی عامل گرفته می‌شود. ○ اجرا در ماشین جدید از یک نقطه‌ی ورودی معلوم ادامه می‌یابد.
○ برای برنامه‌نویس نهایی مناسب‌تر است، اما برای توسعه‌گر سیستم کار بیشتری می‌طلبد. ○ روتین‌هایی برای گرفتن حالت کنترل بر روی مفسرها لازم است.	○ سیستم‌های مبتنی بر جاوای تجاری از مدل «نقطه‌ی ورود معلوم» استفاده می‌کنند.

سیستم‌های عامل سیار

MOBILE AGENT SYSTEMS

سیستم‌های عامل سیار مبتنی بر جاوا *Java-Based Mobile Agent Systems*	سیستم‌های عامل سیار غیر جاوا *Non-Java Mobile Agent Systems*
o Aglets o Concordia o NOMADS	o Telescript o Agent TCL o D'Agents o Ara o TACOMA

عامل‌های سیار

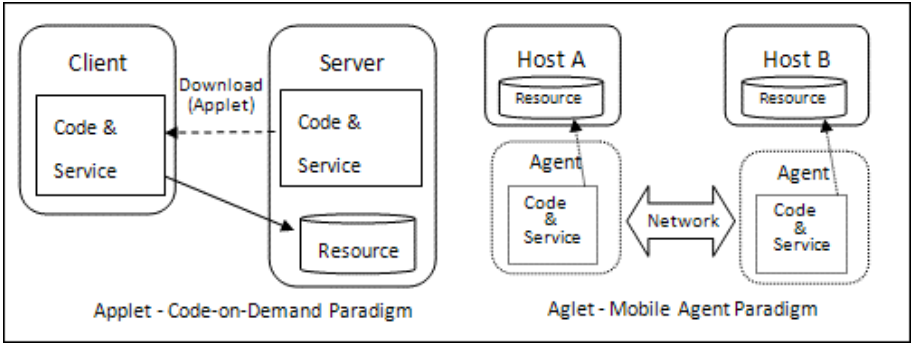
مورد مطالعاتی: اگلت‌ها

CASE STUDY: AGLETS

اگلت در آزمایشگاه پژوهشی IBM توکیو پیاده‌سازی شده است.

http://www.trl.ibm.co.jp/aglets/index.html

اگلت، حالت کنترلی عامل را در حین مهاجرت نمی‌گیرد.
این نیاز دارد تغییراتی در ماشین مجازی استاندارد جاوا ایجاد شود.
در عوض، اجرای عامل از یک «نقطه‌ی شروع معلوم» مجدداً شروع می‌شود.



عامل‌های سیار

مورد مطالعاتی: اگلت‌ها: مفاهیم پایه

BASIC CONCEPTS

Aglet: a mobile Java object that visits aglet-enabled hosts in a network.

- **Proxy:** it is a representative of an aglet.
  It serves as a shield to protect the aglet from direct access to its public methods.
- **Context**: an aglet's workplace. It corresponds to a place.
  Multiple contexts may reside on the same machine.
- **Identifier**: a unique identifier is assigned to each aglet after initialization.

عامل‌های سیار

مورد مطالعاتی: اگلت‌ها: عملیات پایه

FUNDAMENTAL AGLET OPERATIONS

- Creation
- Cloning
- Dispatching
- Retraction
- Activation and Deactivation
- Disposal

عامل‌های سیار

مورد مطالعاتی: اگلت‌ها: مدل برنامه‌نویسی

AGLET PROGRAMMING MODEL

The Aglet programming model is **event-based**

Customized *listeners* can be used to catch particular events in the life cycle of an aglet:

- Clone listener (clone creation)
- Mobility listener (dispatching, retraction, arrival)
- Persistence listener (deactivation, activation)

General process

- A mobile agent that wants to migrate calls the `dispatch` method.
- The Aglets system calls the agent's `onDispatching` method which performs application specific cleanup, kills the agent's threads, serializes the agent's code and object state, and sends the agent's code and object state to the new machine.
- On the new machine the system calls the agents `onArrival` method which performs application specific initialization .
- Finally the agent's `run` method is called in order to start the agent's execution.

عامل‌های سیار

مورد مطالعاتی: اگلت‌ها: برقراری ارتباط

COMMUNICATION BETWEEN AGLETS / ENGINES

Communication between aglets

- When an aglet wants to communicate with other aglets, it first has to obtain a proxy
- Public methods cannot be accessed directly
- Three different types of messages: now-type, future-type, oneway-types

Communication between engines

- The ATP is used in the communication layer
- The API abstracts the communication between agent systems
- Methods for creating, transferring, tracking and managing agents

عامل‌های سیار

مورد مطالعاتی: اگلت‌ها: مدل امنیتی

SECURITY MODEL

The first level of security comes from Java itself.

- Imported code fragments are subjected into a series of tests and consistency tests

As the second level the Tahiti (the visual agent manager) implements a configurable security manager that provides a fairly high degree of security for the hosting computer and its owner and it enables a number of principals to specify and enforce policies

عامل‌های سیار

امنیت

MOBILE AGENT SECURITY

امنیت عامل سیار
Mobile Agent Security

حفاظت ماشین میزبان	حفاظت عامل سیار
Protecting The Host Machine	*Protecting the Mobile Agent*

عامل‌های سیار و میزبان‌ها در برابر تهدیدهای متعدد آسیب‌پذیر هستند:

تهدیدها

تهدیدهای عامل به میزبان	تهدیدهای عامل به عامل	تهدیدهای میزبان به عامل
Agent to Host Threats	*Agent to Agent Threats*	*Host to Agent Threats*

عامل‌های سیار

امنیت: تهدیدهای عامل به میزبان

MOBILE AGENT SECURITY

تهدیدها

تهدیدهای عامل به میزبان
Agent to Host Threats

تهدیدهای عامل به عامل
Agent to Agent Threats

تهدیدهای میزبان به عامل
Host to Agent Threats

نقاب‌زنی
Masquerade

دسترسی غیرمجاز
Unauthorized Access

دستکاری پنهانی
Tampering

امتناع از سرویس
Denial of Service

عامل‌های سیار

امنیت: تهدیدهای عامل به عامل

MOBILE AGENT SECURITY

تهدیدها

تهدیدهای میزبان به عامل *Host to Agent Threats*	**تهدیدهای عامل به عامل** *Agent to Agent Threats*	تهدیدهای عامل به میزبان *Agent to Host Threats*

نقاب‌زنی
Masquerade

دستکاری پنهانی
Tampering

امتناع از سرویس
Denial of Service

انکار
Repudiation

عامل‌های سیار

امنیت: تهدیدهای میزبان به عامل

MOBILE AGENT SECURITY

تهدیدها

تهدیدهای میزبان به عامل *Host to Agent Threats*	تهدیدهای عامل به عامل *Agent to Agent Threats*	تهدیدهای عامل به میزبان *Agent to Host Threats*

نقاب‌زنی
Masquerade

دستکاری پنهانی
Tampering

استراق سمع و تحلیل ترافیک
Eavesdropping and Traffic Analysis

امتناع از سرویس
Denial of Service

عامل‌های سیار

امنیت: سرویس‌های امنیتی

SECURITY SERVICES

عامل‌های سیار

امنیت: حفاظت میزبان

MOBILE AGENTS: PROTECTING THE HOST

امنیت عامل سیار
Mobile Agent Security

حفاظت ماشین میزبان *Protecting The Host Machine*	حفاظت عامل سیار *Protecting the Mobile Agent*

روی‌کردهای مختلفی برای حفاظت میزبان پیشنهاد شده است:

تفسیر کد امن *Safe Code Interpretation*	استفاده از یک زبان مفسری
احراز هویت *Authentication*	امضاهای دیجیتال
مجوزدهی *Authorization*	به موجودیت‌ها می‌توان قابلیت‌ها یا اجازه‌های دسترسی اعطا کرد.
تخصیص منابع *Resource Allocation*	مکانیسم‌های متنوع برای تخصیص منبع: شامل مکانیسم‌های بازار
نگهداری تاریخچه‌های مسیر *Maintaining Path Histories*	برای اینکه بتوان مبدأ عامل‌های سیار را بررسی کرد.

عامل‌های سیار

امنیت: حفاظت عامل سیار

MOBILE AGENTS: PROTECTING THE HOST

امنیت عامل سیار
Mobile Agent Security

حفاظت عامل سیار *Protecting the Mobile Agent*	**حفاظت ماشین میزبان** *Protecting The Host Machine*

روی‌کردهای مختلفی برای حفاظت عامل سیار پیشنهاد شده است:

مدل قلعه *Fortress Model*	عامل‌ها در یک محیط امن اجرا می‌شوند که هیچ میزبان نامعتمدی اجازه فعالیت ندارد (بده‌بستان مزایای عامل سیار).
مهر و امضا *Sealing and Signing*	نتایج میانی برای جلوگیری از دستکاری پنهانی مهر و امضا می‌شوند.
استفاده از رمزهای مشترک *Using Shared Secrets*	استفاده از رمزهای مشترک و به‌هم قفل شده
ضبط رفتار عامل *Recording Agent Behavior*	اجرا و رفتار عامل ضبط می‌شود (ردگیری اجرا).
توابع رمزشده‌ی قابل اجرا *Executable Encrypted Functions*	توابع رمزشده‌ی قابل اجرا می‌توانند از دستکاری پنهانی جلوگیری کنند.

عامل‌های سیار

ملاحظات نرم‌افزاری

MOBILE AGENTS

وقتی می‌خواهیم ابزارهای نرم‌افزاری بسازیم که بتوانند عامل‌های سیار را پشتیبانی کنند، ملاحظات متعددی داریم:

- امنیت میزبان‌ها و عامل‌ها *(security for hosts and agents)*
- ناهمگنی میزبان‌ها *(heterogeneity of hosts)*
- پیونددهی پویا *(dynamic linking)*

عامل‌های سیار

ملاحظات نرم‌افزاری: امنیت میزبان‌ها

SECURITY OF HOSTS

نمی‌خواهیم برنامه‌های خارجی را بر روی ماشین خودمان اجرا کنیم،
چرا که ریسک‌های امنیتی زیادی وجود دارد:

اگر زبان برنامه‌سازی عامل از اشاره‌گرها پشتیبانی کند،
آن‌گاه این خطر وجود دارد که عامل‌ها فضای آدرس میزبان را خراب کنند.
(⇐ بسیاری از زبان‌های برنامه‌سازی عامل، اشاره‌گر ندارند!)

مجوزهای دسترسی مشابه یونیکس بر روی میزبان:
کتابخانه‌های امن برای دسترسی به فضای ذخیره فایل‌ها، فضای پردازش و . . .

برخی کنش‌ها (مانند ارسال ایمیل) در برخی مواقع **مضر** هستند اما در برخی دیگر **خطرناک** هستند.

عامل‌های سیار

ملاحظات نرم‌افزاری: امنیت میزبان‌ها: راه‌حل‌ها

SECURITY OF HOSTS

برخی زبان‌های عامل (مانند TELESCRIPT)
بر روی مقدار حافظه و زمان پردازشی که عامل می‌تواند به آن دسترسی پیدا کند، محدودیت می‌گذارند.

هم‌پردازنده‌های امن ***(Secure Co-Processors)*** یک راه‌حل هستند:
دارای یک پردازنده‌ی جداگانه به‌طور فیزیکی برای اجرای هر عامل که این پردازنده قرنطینه باشد.

برخی زبان‌های برنامه‌سازی عامل، اجازه می‌دهند خصوصیات امنیتی یک عامل در هنگام دریافت وارسی شود.

عامل‌های سیار

ملاحظات نرم‌افزاری: امنیت عامل‌ها

SECURITY OF AGENTS

عامل‌ها دارای حق **حریم خصوصی** هستند!

اغلب نمی‌خواهیم برنامه‌هایمان را بفرستیم:
زیرا گیرنده ممکن است مقصود آن را بفهمد و درنتیجه از نیت ما آگاه شود.

ممکن است عامل به‌گونه‌ای تغییر داده شود (خلاف)
بدون اطلاع یا موافقت مالک آن

یک عامل می‌تواند در انتقال به‌وسیله‌ی تکنیک‌های رمزگذاری متداول (مثل *PGP*) محافظت شود.

برای اطمینان از اینکه عامل دستکاری نشده است، می‌توان از واترمارک‌های دیجیتال استفاده کرد.

عامل‌های سیار

ملاحظات نرم‌افزاری: ناهمگنی میزبان‌ها

HETEROGENEITY OF HOSTS

اگر نخواهیم که عامل‌های ما فقط بر روی یک نوع ماشین اجرا شوند،
آن‌گاه باید تسهیلاتی برای اجرای یک عامل بر روی انواع متفاوت زیادی از ماشین‌ها فراهم کنیم.

برای این منظور نیاز داریم به:

زبان تفسیر شده (*interpreted language*)
زبان‌های کامپایل شده نیازمند کاهش کد به زبان ماشین هستند که به‌وضوح وابسته به سیستم است.
* این موجب کاهش کارآمدی می‌شود (احتمالاً استفاده از تکنولوژی ماشین مجازی)

پیونددهی پویا (*dynamic linking*)
کتابخانه‌هایی که به منابع محلی دسترسی دارند، بایستی واسط مشترکی را برای محیط‌های گوناگون تهیه کنند.

عامل‌های سیار

تیپولوژی

A TYPOLOGY FOR MOBILE AGENTS

عامل‌های سیار
Mobile Agents

نوع **خودمختار**	نوع **در–صورت–تقاضا**	نوع **نامه–فعال**
Autonomous	*On-demand*	*'Active-Mail'-Type*

عامل‌های سیار

تیپولوژی: نوع خودمختار

A TYPOLOGY FOR MOBILE AGENTS

عامل‌های سیار *Mobile Agents*		
نوع **خودمختار** *Autonomous*	نوع بر-صورت-تقاضا *On-demand*	نوع نامه-فعال *Active-Mail'-Type*

منظور از عامل سیار خودمختار این است که
عامل خودش قادر به این است که تصمیم بگیرد کجا و چه‌وقت برود و وقتی به آنجا رفت چه کاری انجام بدهد.
(تصمیم‌گیری بر اساس قیدهای منابع به‌خصوص، مثلاً میزان پولی که باید صرف شود.)

برنامه‌نویسی این عامل‌ها عموماً در زبان ویژه‌ای انجام می‌شود که یک دستور *go* داشته باشد ...
(بهترین مثال مشخص زبان TELESCRIPT است.)

عامل‌های سیار

تیپولوژی: نوع در–صورت–تقاضا

A TYPOLOGY FOR MOBILE AGENTS

عامل‌های سیار *Mobile Agents*		
نوع خودمختار *Autonomous*	**نوع در–صورت–تقاضا** *On-demand*	نوع نامه–فعال *Active-Mail Type*

ایده‌ی عامل سیار در–صورت–تقاضا این است که
میزبان تنها لازم است زمانی یک عامل را اجرا کند که آن عامل صراحتاً تقاضا کند.

بهترین مثال مشخص برای این کارکرد، زبان JAVA است که درون یک HTML جاسازی شده است:
یک کاربر در یک مرورگر سازگار با جاوا، می‌تواند صفحات وبی را درخواست کند که حاوی **اپلت *applet*** است.
(اپلت: برنامه‌های کوچکی که در زبان جاوا پیاده‌سازی شده است).
این اپلت‌ها همزمان با سایر تصاویر صفحه، متون، فرم‌ها و . . . دانلود می‌شوند
و وقتی دانلود شدند، بر روی ماشین کاربر اجرا می‌شوند.

عامل‌های سیار

تیپولوژی: نوع نامه–فعال

A TYPOLOGY FOR MOBILE AGENTS

عامل‌های سیار *Mobile Agents*		
نوع **نامه–فعال** *'Active-Mail'-Type*	نوع در–صورت–تقاضا *On-demand*	نوع خودمختار *Autonomous*

ایده‌ی عامل سیار نامه–فعال این است که
برنامه‌ی عامل بر پشت یک ایمیل سوار شود.

بهترین مثال مشخص برای این کارکرد، گسترش MIME است که بر ایمیل سوار می‌شود
و امکان ارسال اسکریپت‌های امن TCL را فراهم می‌کند.

وقتی ایمیل دریافت شد، «عامل» آن را باز می‌کند و اسکریپت را اجرا می‌کند ...
بنابراین، نامه دیگر منفعل *(passive)* نیست، بلکه فعال *(active)* است.

عامل‌های سیار

مورد مطالعاتی: تِله اسکریپت

CASE STUDY: TELESCRIPT

تله‌اسکریپت، یک محیط مبتنی بر زبان برای ساخت سیستم‌های عامل سیار بود.

تکنولوژی تله‌اسکریپت نامی است که توسط *General Magic* به خانواده‌ای از مفاهیم و تکنیک‌ها برای پی‌ریزی محصولاتی که توسعه داده بودند، داده شده است.

دو مفهوم کلیدی در تکنولوژی تله‌اسکریپت: **عامل‌ها** و **مکان‌ها**

Places

Places are virtual locations occupied by agents. A place may correspond to a single machine, or a family of machines.

Agents

Agents are the providers and consumers of goods in the electronic marketplace applications that TELESCRIPT was developed to support.

- ***Agents are interpreted programs**, rather like TCL.*
- ***Agents are mobile** — they are able to move from one place to another, in which case their program and state are encoded and transmitted across a network to another place, where execution recommences.*
- *In order to travel across the network, **an agent uses a ticket**, which specifies the parameters of its journey:*
  - *destination;*
  - *completion time.*

عامل‌های سیار

مورد مطالعاتی: تِله اسکریپت

CASE STUDY: TELESCRIPT

عامل‌ها در تله‌اسکریپت می‌توانند با یکدیگر ارتباط برقرار کنند:

اگر این عامل‌ها مکان‌های مختلفی را اشغال کنند، آن‌گاه می‌توانند از طریق شبکه به هم متصل شوند.

اگر این عامل‌ها مکان‌های یکسانی را اشغال کنند، آن‌گاه می‌توانند یکدیگر را **ملاقات** کنند.

عامل‌های سیار

مورد مطالعاتی: تِله اسکریپت

CASE STUDY: TELESCRIPT

TELESCRIPT agents have an associated permit, which specifies:

- what the agent can do (e.g., limitations on travel);
- what resources the agent can use.

The most important resources are:

- **'money'**, measured in 'teleclicks' (which correspond to real money);
- **lifetime** (measured in seconds);
- **size** (measured in bytes).

Agents and places are executed by an engine.

- An engine is a kind of agent operating system
  — agents correspond to operating system processes.
- Just as operating systems can limit the access provided to a process (e.g., in UNIX, via access rights), so an engine limits the way an agent can access its environment.

عامل‌های سیار

مورد مطالعاتی: تِله اسکریپت

CASE STUDY: TELESCRIPT

Engines continually monitor agent's resource consumption, and kill agents that exceed their limit.

- Engines provide (C/C++) links to other applications via application program interfaces (APIs).

Agents and places are programmed using the TELESCRIPT language:

- pure object oriented language, apparently based on SMALLTALK;
- interpreted;
- two levels — **high** (the 'visible' language), and **low** (a semi-compiled language for efficient execution);
- a '**process**' class, of which '**agent**' and '**place**' are sub-classes;
- persistent;

General Magic claim that the sophisticated built in communications services make TELESCRIPT ideal for agent applications!

عامل‌های سیار

مورد مطالعاتی: تِله اسکریپت

CASE STUDY: TELESCRIPT

Summary:

- a rich set of primitives for building distributed applications, with a fairly powerful notion of agency;
- agents are ultimately interpreted programs;
- no notion of strong agency!
- likely to have a significant impact (support from Apple, AT&T, Motorola, Philips, Sony).
- not heard of anyone who has yet actually ***used*** it!

عامل‌های سیار

مورد مطالعاتی: TCL/TK و زبان‌های اسکریپت‌نویسی

CASE STUDY: TCL/TK AND SCRIPTING LANGUAGES

زبان کنترل ابزار (آزاد) *TCL* و همراه آن *TK*
هم‌اکنون در ارتباط با سیستم‌های مبتنی بر عامل مورد اشاره قرار می‌گیرند.

TCL - Tool Command Language

TCL was primarily intended as a standard command language — lots of applications provide such languages, (databases, spreadsheets, . . .), but every time a new application is developed, a new command language must be as well.

- *TCL provides the facilities to easily implement your own command language.*

TK (framework)

TK is an X window based widget toolkit — it provides facilities for making GUI features such as buttons, labels, text and graphic windows (much like other X widget sets).

- *TK also provides powerful facilities for interprocess communication, via the exchange of TCL scripts.*

عامل‌های سیار

مورد مطالعاتی: TCL/TK و زبان‌های اسکریپت‌نویسی

CASE STUDY: TCL/TK AND SCRIPTING LANGUAGES

TCL/TK combined, make an attractive and simple to use GUI development tool; however, they have features that make them much more interesting:

- TCL is an **interpreted language**;
- TCL is **extendable** — it provides a core set of primitives, implemented in C/C++, and allows the user to build on these as required;
- TCL/TK can be **embedded** — the interpreter itself is available as C++ code, which can be embedded in an application, and can itself be extended.

عامل‌های سیار

مورد مطالعاتی: TCL/TK و زبان‌های اسکریپت‌نویسی

CASE STUDY: TCL/TK AND SCRIPTING LANGUAGES

TCL programs are called **scripts**.

TCL scripts have many of the properties that UNIX shell scripts have:

- they are plain text programs, that contain:
  - control structures (iteration, sequence, selection), and
  - data structures (e.g., variables, lists, and arrays) just like a normal programming language;
- they can be executed by a shell program
  - `tclsh` or `wish`;
- they can call up various other programs and obtain results from these programs (cf. procedure calls).

عامل‌های سیار

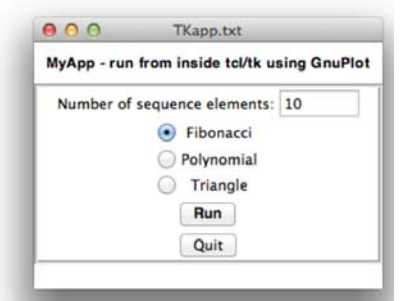
مورد مطالعاتی: TCL/TK و زبان‌های اسکریپت‌نویسی

CASE STUDY: TCL/TK AND SCRIPTING LANGUAGES

As TCL programs are interpreted, they are very much easier to prototype and debug than compiled languages like C/C++
— they also provide more powerful control constructs . . .

- . . . but this power comes at the expense of speed.
- Also, the structuring constructs provided by TCL leave something to be desired.

```
#!/usr/bin/wish
#

# Define the widgets and pack them
label .title -bd 10 -font Helvetica-Bold -text "MyApp - run from inside tcl/tk using GnuPlot"

# ###################################################################
# Create the Arguments Pane
#

frame .cmds -relief groove -borderwidth 4
label .cmds.title -text "Arguments"

frame .cmds.num
label .cmds.num.label -text "Number of sequence elements:"
entry .cmds.num.entry -width 8 -relief sunken -bd 2 -textvariable num

set num 10...
```



عامل‌های سیار

مورد مطالعاتی: TCL/TK و زبان‌های اسکریپت‌نویسی

CASE STUDY: TCL/TK AND SCRIPTING LANGUAGES

So where does the idea of an agent come in?

- It is easy to build applications where TCL scripts are exchanged across a network, and executed on remote machines.
  - Thus TCL scripts become sort of agents.
- A key issue is safety.
  - You don't want to provide someone else's script with the full access to your computer that an ordinary scripting language (e.g., csh) provides.
- This led to Safe TCL, which provides mechanisms for limiting the access provided to a script.
  - Example: Safe TCL control the access that a script has to the UI, by placing limits on the number of times a window can be modified by a script.
- But the safety issue has not yet been fully resolved in TCL.
  - This limits its attractiveness as an agent programming environment.

عامل‌های سیار

مورد مطالعاتی: TCL/TK و زبان‌های اسکریپت‌نویسی

CASE STUDY: TCL/TK AND SCRIPTING LANGUAGES

Summary:

- TCL/TK provide a rich environment for building language-based applications, particularly GUI-based ones.
- But they are not/were not intended as agent programming environments.
- The core primitives may be used for building agent programming environments
  - the source code is free, stable, well-designed, and easily modified.

عامل‌های سیار

ملاحظات

ISSUES ON MOBILE AGENTS

Technical issues

- Current mobile agents do not bring about significant benefits
- As they are written in interpreted languages, they are slow
- Starting an execution environment and inserting an agent involves an overhead – thus higher loads in transporting and executing them locally
- To recover from failure during migration additional support is required
- Appropriate naming and location services are required
- Security
- Lack of standardization

Nontechnical issues

- Lack of a killer application
- The advantages of mobile agents are modest when applications are considered in isolation

Potential applications

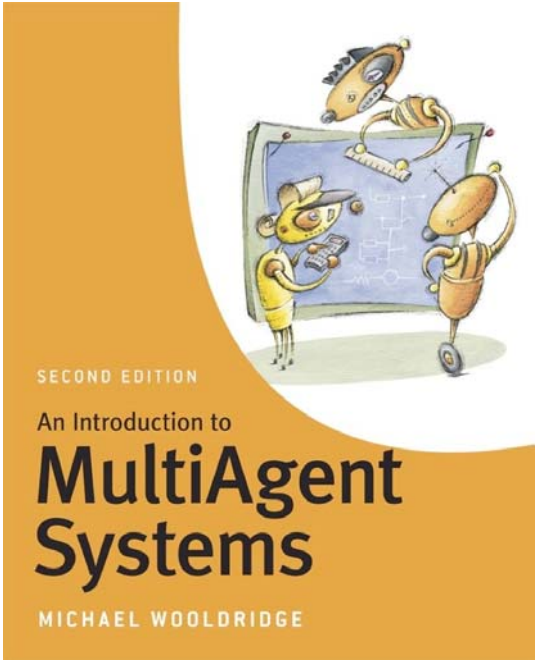
- Electronic marketplaces
- Mobile devices such as mobile phones and PDAs


Methodologies
when is an agent solution appropriate?
distribution of control
wrapping legacy
a natural metaphor
environment is open
pitfalls
overselling
getting religious or dogmatic
no clear need
generic solutions to one-off problems
believing in silver bullets
getting obsessed with AI
forgetting basic software engineering
obsessed with infrastructure
mobile agents
transmit code across network
key issue security
Telescript
Aglets
Agent Tcl
agent-oriented methodologies
AAII methodology
agent model
interaction model
agent class model
agent instance model
roles
agent class hierarchy
GAIA
abstract
roles
permissions
responsibilities
protocols
activities
liveness properties
safety properties
Agent UML
extend UML with agent features

عامل‌های سیار

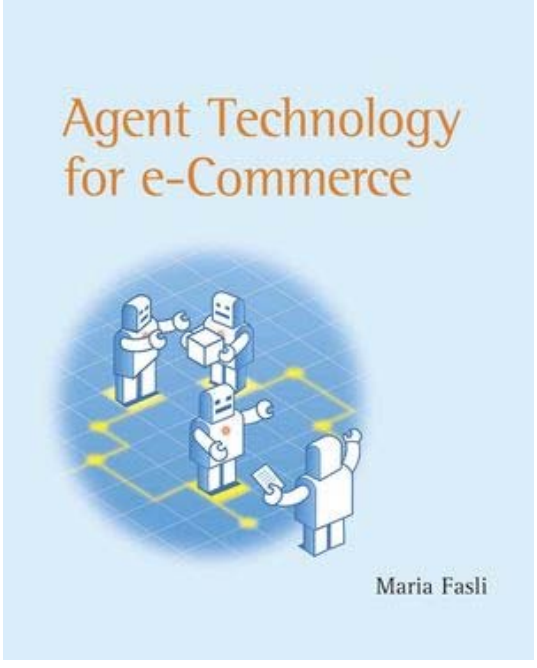
منابع

منبع اصلی



Michael Wooldridge,
An Introduction to Multiagent Systems,
Second Edition,
John Wiley & Sons, 2009.
Chapter 9

منبع کمکی



Maria Fasli,
Agent Technology For E-Commerce,
John Wiley & Sons, 2007.
Chapter 11