Mobile Software-Agents

Workshop at Schloss Dagstuhl
Oct 13-17, 1997

organized by

Friedemann Mattern (Darmstadt), Kurt Rothermel (Stuttgart),
Fred B. Schneider (Cornell), Brent Welch (Mountain View)

Mobile agents are programs that travel through a computer communications network and have a persistent identity and local state. Agents can communicate with other mobile agents or with stationary (server) agents. Agents may travel to become co-resident with information or processing resources, to propagate data, or even to do business on behalf of their users. Possible applications include network management, service infrastructures, massively distributed information systems, and electronic commerce.

First implementations of systems supporting mobile agents already exist. General Magic's Telescript language supports sending and receiving agents from remote machines. Sun Microsystems released the Java language, which can be used in combination Sun's WWW browser. Active electronic mail over the Internet is another example.

The computational metaphor of mobile agents offers unique opportunities for structuring and implementing dynamic and open distributed systems and applications. But, numerous fundamental open technical problems must be addressed before the technology becomes usable:

- Security issues: How can the system and the infrastructure be protected against malicious agents and hosts? How can an agent or site be protected from other agents?

- Agent interaction: How do agents communicate with each other and with the environment? What are good programming constructs for specifying such interaction?

- Agent control policies: How should troops of agents be organized to accomplish a goal? How can system resources be managed and shared among a collection of agents?

- Coping with heterogeneity: What mechanisms are needed for agents to be able to migrate from site to site?

These problems all cluster around two traditional sub-fields of computing: one is programming languages, the other is operating systems. The aim of the workshop therefore was to bring together a diverse group of researchers in these fields, as well as scientists from industry who are interested in agent technology and who are committed to realize such systems. The goal was to shape a research agenda for an increasingly relevant problem area.

The organizers of the workshop want to thank the European Community for supporting this workshop by the Training and Mobility of Researchers (TMR) Program. The program, which provides financial support for travel costs and accommodation fee in Dagstuhl, enabled that selected young researchers had the possibility to attend this workshop.

Contents

1 Termination and Orphan Detection in Mobile Agent Systems

Joachim Baumann
University of Stuttgart

Orphan detection and termination in distributed systems is a well researched field for which many solutions exist. These solutions exploit well defined parent-child relationships given in distributed systems. But they are not applicable in mobile agent systems, since no similar natural relationship between agents exist. Thus new protocols have to be developed. In this talk four different approaches are presented, the 'energy concept', the 'shadow protocol', 'agent dependencies' and 'agent groups'.

The 'energy' approach is based on the idea that an agent is provided with a limited amount of energy, which can be spent in exchange for the resources used by the agent. From time to time the agent has to request additional energy from its creator. The agent is terminated as soon as the energy falls to 0. This approach to agent termination implicitly implements orphan detection, i.e. if the creator has terminated, the dependent agents are killed as soon as they have no energy left. The 'shadow' approach uses the idea of a placeholder (shadow) which is assigned by the agent system to each new agent. The shadow records the location of all dependent agents. Removing the root shadow implies that all dependent shadows and agents are terminated recursively. The 'agent dependencies' approach allows to define dependencies between agents. The idea is to allow to declare one agent dependent on another agent or a group of agents. The group dependency can be interpreted as an OR or an AND dependency. In the case of the OR dependency the agent is allowed to live as long as at least one of the agents in the group still exists, and in the case of the AND dependency the agent is declared an orphan as soon as only one of the agents in the AND group no longer exists. The 'agent groups' approach is the most flexible of the presented approaches, in that not only the existence of other agents is used to derive if an agent is an orphan, but arbitrary events. A simple example allows the definition of OR and AND groups that take the success or failure of the contained agents into account. Other, more complex groups that implement application-specific behavior, are equally simple to define. Furthermore, it is shown that these protocols can be modeled with Colored Petri Nets.

2 System Support for Mobile Agents in Correlate

Yolande Berbers
Katholieke Universiteit Heverlee - Leuven

CORRELATE is a consumer object-oriented language (built as an extension of C++ and Java) with a flexible run time system that enables the instantiation of application specific runtime objects. This run time system uses a meta-level architecture that reflects on the application. The run time system can be used to add support for e.g. fault tolerance or load balancing for the application, or in the case of mobile agents as a location control subsystem.

3 Integration of Mobile Agent Technology in IN Environments

M. Breugst, T. Magedanz
GMD - German National Research Center for Information Technology

Existing telecommunication service architectures, such as the Intelligent Networks (IN), which are based on the traditional RPC paradigm of computing, provide services in a rather centralized way, i.e. service and management intelligence are located in only a few centralized network nodes, inherently resulting in potential communication inefficiencies and bottlenecks, and making effective on demand and customized service provision hard to accomplish. Due to their inherent autonomy and mobility, mobile agents offer new opportunities for the provisioning of services in the emerging electronic service markets. Mobile agents may for instance be used to rapidly distribute individually customized intelligent (i.e. autonomous and pro-active etc.) service components, effectively supporting on-demand service provisioning schemes. Also, by allowing the movement of both code and data, mobile agent systems allow the programmer to minimize the use of low-bandwidth, high-latency or unreliable network connections typical for mobile service environments. This lecture illustrates in which way the current intelligent network architecture can be enhanced my means of mobile agents. Since the value of mobile agent technology compared to the 'traditional' client/server technology depends strongly on the characteristics of the respective IN application, an integrated approach is chosen which is capable of supporting both modeling paradigms. The lecture outlines the conceptual approach for an agent based IN and describes service scenarios which have been realized in the context of a research project of GMD FOKUS.

4 Coordination and Mobility in PageSpace

Paolo Ciancarini
University of Bologna

The original Web did not support distributed, multiuser, interactive applications based on autonomous agents. The advent of Java has changed the situation, introducing the possibility of using the WWW as middleware for agent-based applications. However, Java mechanisms for agent-oriented programming are still in their infancy. This shortcoming is being studied, and several approaches have been proposed. However, most existing approaches are oriented to centralized coordination at servers. To overcome this deficit, we introduced PageSpace, that is both a reference architecture and a platform for distributed, coordinated Java agents.
We take a specific approach to coordinate agents in PageSpace, namely variants of the coordination language Linda that support rules and services to guide their cooperation. Coordination technology is integrated with the standard Web technology and the programming language Java.
Several kinds of agents live in the PageSpace: user interface agents, personal persistent homeagents, agents that implement applications, and the kernel agents offering coordination services on a specific platform. Within our architecture it is possible to support fault-tolerance and mobile agents as well.
Joint work with R. Tolksdorf (TU Berlin)

5 The Web-Agent Based Service Providing (WASP) Project – Almost Transparent Migration for Java-based Mobile Agents

Stefan Fünfrocken
Darmstadt University of Technology

With our project we aim at showing the usefulness of mobile agents. We developed an architecture that enables Web servers to host mobile agents. We provide this functionality to other servers by offering our Server Agent Environment (SAE) as a plug-in. On top of this architecture we will use mobile agents to implement services on Web data. A service customer has to subscribe to the service (e.g., a personalized Web newspaper service). After subscribing, the agent providing the actual service is installed at the user's Web server. This agent roams the Web to collect the information according to the preferences specified by the customer. The customer is charged according the service contract and may receive new versions of the service agent if he or she subscribed to the maintenance service. To allow the agent to access the local data of Web servers equipped with our system, we provide it with a special interface which mediates all local accesses. Our security architecture can enforce site policies for resources. This includes file access restriction, time monitoring, and network access control.
In another part of the project we develop a preprocessor which enables an agent programmer to use a 'go' statement which realizes transparent migration for Java. We transform that code in such a way that it is able to run in a standard Java environment. This transformation results in larger code because we insert code that saves the run time stack before migration and rebuilds it after migration.

6 Agent Tcl: A Flexible and Secure Mobile Agent System

Robert S. Gray
Dartmouth College

Agent Tcl is a mobile agent sustem that is under development at Dartmouth College and that has evolved from a Tcl-only system into a system that currently supports Tcl, Java, and scheme. In our talk, we describe the base Agent Tcl system, its security mechanisms for protecting a machine against malicious agents, and several support services and applications. Then we discuss the security, fault tolerance and performance enhancements that will be necessary for Agent Tcl to realize its full potential.

7 Protecting Mobile Agents from Malicious Hosts by Using Blackbox Security

Fritz Hohl
University of Stuttgart

Mobile agents are often described as a promising technology moving towards the vision of a widely distributed scalable electronic market. In this talk, an approach to partially solve one of the most difficult aspects of security of mobile agents systems is presented, the problem of malicious hosts. After introducing the problem and discussing some existing approaches, the central idea and the components of the mechanism, which is called blackbox security, is presented. This idea consists in creating new "versions" out of an original agent at the starting node and to attach a protection interval to that version. Since a potential attacker needs at least that interval to attack the agent, i.e. to find variable values and important code lines, the agent is safe for that interval. After that "expiration date" the agent gets invalid and cannot migrate or interact anymore. The talk closes with a discussion of a counter attack and the costs of the mechanism.

8 MCK: Mobile Communication Kernel – A Framework for Intelligent Communication with Mobile Devices in a Cellular Communication Network

Leon Hurst
Trinity College

Wireless communication with mobile computing devices is known to be problematic. It suffers from a variety of problems arising from device mobility, including frequent and prolonged disconnections, very low, variable bandwidth and high latencies. Conventional communication protocols such as TCP and RPC, are not suitable for mobile communication as they react badly to these problems.
This talk deals with the development of an intelligent communication protocol, aimed specifically at the cellular wireless networks. This protocol is called the Smart Messaging Protocol (SMP) and is implemented in the Mobile Communication Kernel (MCK). The protocol spans the transport and application layers in a conventional communication stack. Messages are encoded as mobile agents which govern their own routing, recovery and filtering. As these messages pass from router to router, they respond to events regarding changes in the local environment (host and network). In reaction to these events, a message can dynamically alter its plans. Thus, all routing, recovery and filtering policies are efficiently encoded into the messages themselves, rather than into the routers. Whereas the routers are open, and provide the necessary mechanisms for messages to operate, the messages provide the policies. Hence, a separation of mechanism from policy.
The approach taken expands on the self-routing capabilities of current Mobile Agent systems such as Aglets or Telescript. We aim to provide structured support for handling the particular problems associated with wireless communication. It is argued that this offers an efficient, adaptable and robust solution to many of the problems associated with this hostile communication environment.

9 Applications of Mobile Code: Experiences with TACOMA

Dag Johansen
University of Tromsø

In this talk, we will briefly present experiences from building several agent-based distributed applications using TACOMA. Our hope is to demonstrate potential for its applicability through some real and concrete examples. We conclude that agents are complementary to more traditional structuring techniques in distributed applications. TACOMA systems run on top of UNIX, Windows 95, Windows NT, and some PDA systems (i.e. PalmPilot).
More information at URL: http://www.cs.uit.no/DOS/Tacoma/
Joint work with Fred B. Schneider and Robbert van Renesse, Cornell University, USA.

10 Security Services for Mobile Agents

Günter Karjoth
IBM Zürich Research Laboratory

We introduce a minimal set of security services a mobile agent platform shall provide to protect itself against malicious agents as well as, but to a lesser extend, to protect the agent against malicious hosts. With the eyes of a number of principals, identified with respect to their responsibilities (liabilities) and interests, we look into authentication, access control, and network security, followed by some remarks on privacy, audit, and the role of trust.

11 Java - Just What Mobile Agents Need?

Danny B. Lange
General Magic Inc.

You probably already know what Java is. The C++-like language that changed the Web overnight is offering some unique capabilities that are fuelling mobile agent systems. In this talk I will show what exactly it is that makes Java such a powerful tool for agent development. I will however also direct your attention to some short-comings in the Java language system that have implications for the design and use of Java-based mobile agent systems.

12 Talking to Mobile Agents: Requests, Results, Interaction

Anselm Lingnau
Johann-Wolfgang-Goethe-University Frankfurt

Have you talked to your agent lately? While communication among mobile agents has been reasonably well researched, leading up even to proposals for standards, the system requirements for interaction between agents and their users still offer some challenges. We identify various modes of communication—the specification of an agent's initial 'mission' is an example of user-initiated interaction, while agents may report intermediate and final results as well as errors. But agents can also get back to their owners for further instructions or an extra helping of resources such as allowable CPU time—and agents may even want to interact with other people, to ask them questions or perform services. The ffMAIN infrastructure addresses these issues through a secure mechanism based on ubiquitous WWW technology, enabling communication in a platform-independent manner even across slow, non-permanent communication links.
Slides (PostScript) for this presentation are available by anonymous FTP from `ftp://ftp.tm.informatik.uni-frankfurt.de/pub/papers/agents`.

13 Nile / Tacoma = ?

Keith Marzullo
University of California at San Diego

We've recently built a distributed environment for the execution of massive data-parallel applications in a wide-area network. This system (the NSF National Challenge NILE project) provides functions that are very similar to what mobile agent platforms provide. So, why didn't we built NILE on top of, say, TACOMA? We didn't because NILE is not only a research project; it is slated to be used by the physicists of the CLEO collaboration (an important NSF project centered at Wilson Laboratories). We have tried to keep our dependency on research-developed code to a minimum. But, this does allow us to have a benchmark with which to compare a TACOMA-based NILE. In this talk, I give an overview of how I am redesigning the current NILE system to use TACOMA. I concentrate on two functions that appear to be well-suited to TACOMA, and report on the problems that I've confronted in the re-design of these functions.

14 Mobile Software Agents: an Overview

Friedemann Mattern
Darmstadt University of Technology

In this short introductory lecture we survey the main features, challenges, and open problems of mobile agent technology.
Agents are autonomous entities that act in the name of someone. They have a state and an identity and may communicate and cooperate with other agents. An agent is called mobile if its code, and possibly also its run time state, can move to a different processor while it is being executed.
Mobile agents are attractive because in some cases they seem to represent a better communication paradigm than messages or client-server communication: Message traffic may be reduced, and resources attached to specific sites may be exploited more efficiently when the computation is moved to the data. In particular, mobile computing applications may benefit from the paradigm by launching an agent during brief connect sessions which then performs its task asynchronously. Possible application areas include electronic commerce, search and semantic information retrieval, remote diagnosis and maintenance, subscription services, information update, and cooperation services such as workflow management or

active documents.
Agent technology does not come for free: Mobile agents require open and platform independent infrastructure mechanisms such as communication protocols, migration support, and security mechanisms. Furthermore, agents and whole societies of agents must be controlled, and local resources of agent platforms must be protected. Existing prototype systems often address only part of these aspects. Current research themes are (among other things) adequate languages for mobile agents (including the semantics of mobility), inter-agent communication, security aspects, and fault tolerance.
Although agent technology seems to be very attractive, it is still unclear whether there will indeed be strong interest for agents in an open Internet-based service market.

15 Electronic Contract Negotiation as an Application Niche for Mobile Agents?

Michael Merz
University of Hamburg

Requirements and constraints from application fields often reduce design options to only a small area where it is natural and maybe more efficient to follow the mobile agent approach rather than other communications/cooperation paradigms. In my presentation, I propose Electronic Contract Negotiation as an application that is very likely to gain advantage from using mobile agents. In this scenario, human users are supported in negotiating a contract (by using a negotiation protocol), after agreement has been reached, the contract is signed and executed. In order to allow users to access specific contract data, access code (GUI) is required. Further, negotiation is coordinated on a workflow basis, taking this together, we integrate code, data, and (workflow) execution state - giving a mobile agent. To my opinion, mobile agents make only sense, i.e. prove to be better than other approaches under the following conditions:

- High bandwidth costs
- 3 or more participating hosts
- Small agent size (incl. code!)
- Medium to high transaction volume (commercial transaction)

- Ubiquitous execution environment (i.e., Java)
- High level of autonomy
- Sporadic need for communication between the nodes involved

My definition for a mobil agent:
A mobile agent is an encapsulation of data, code, and execution state that is able to migrate autonomously and purposefully within computer networks during execution.

16 Mobile Objects and Agents (MOA) and Mobile Agent Facility (MAF)

Dejan Milojicic
The Open Group Research Institute

The Mobile Objects and Agents (MOA) project at the OpenGroup deals with the mobile agents system with three main goals: transparent migration while maintaining open communication channels and transparently locating agents; maintaining strict control of agent resources (open channels, number of hops, agent life time, etc.); compliance with Java Beans component model - many MOA components can be configured and instantiated at the run time. The MOA agent system has similar (but different) semantics as Telescript agents: it supports agents and places that agents visit; places can be hierarchical, they represent containers for the resources and security, and they are a basis for interagent communication.
The MOA system is being implemented in Java and it is scheduled for delivery by the end of the 1997.
As a part of the MOA project we developed a demo called "Rent-A-Soft" that we presented at the Uniforum. It demonstrates the use of mobile agents for renting and distributing software over the network. Rent-A-Soft allows software companies better to leverage market by reaching customers incapable of buying, but willing to rent certain software packages. This is applicable for games which tend to have increased price. Contribution of mobility is for encapsulating software packages and associating code for rental expiration, inventory, availability, statistics, upgrades, etc. A chain of agents can be encapsulating rented application and representing software producer, main renting company, its departments,

company that rents and its own departments, all of them being involved in the rental process.
The Mobile Agent Facility is a standard that is being standardized at OMG. The presentation dealt with the following issues:

a) What is MAF standardizing

b) What has not been proposed for standardization

c) Details on standardizing:

- mobility
- naming
- locating
- security

More details on MOA can be found at http://www.opengroup.org/RI/java/moa
More details on MAF can be found at http://www.genmagic.com/agents/MAF/

17 A Security Design for the Ara Mobile Agent Platform

Holger Peine
University of Kaiserslautern

We present the design of a security scheme which will, after some refinement, be implemented in the Ara mobile agent platform. The subjects of the scheme are agent users, agent manufacturers, and server machines, while the objects to be secured are OS objects such as files, and agent system objects such as other agents. The implementation is based on public key cryptography, in particular digital signatures, public key certificates, and the SSL protocol. Agents bear an immutable passport, signed by their user, including authorization limits and the user's certificate, as well as mutable security attributes such as an incrementally signed trail of visited places. The agent's code is also signed by the user and possibly the manufacturer. Agent user authentication is ultimately founded on OS user account protection, establishing a chain of authentication from a user's password login to his/her agent's arrival at some remote place. Machines have their own certificates, too, and all involved certificates are validated during agent migration between two machines. If both machines belong to one region under the same authority, security checks may be omitted within such a region. The rights a receiving place grants to an entering agent are computed by an application-defined function of the agent's passport, further attributes, and strength of authentication. This admission function implements the place's security policy in a flexible way. Various open issues of the presented design are named, including signing agents created away from home, and further security of the agent against the host.

18 Mobile Agents at Baan Company

Thijs Petter
Baan-T-Labs. Corp.

Baan Company is a vendor of Enterprise Business Applications, also known as Enterprise Resource Planning (ERP) systems. These applications enable the control of information in a company. Current ERP applications tend to be monolithic. The Baan Labs department is working on a technology framework that supports development of component oriented applications that cover different domains of functionality (e.g. applications to support sales processes, manufacturing processes or inventory processes). Currently we are developing a Mobile Agent System that can support interoperability of different applications. This interoperability is defined in terms of business processes that are represented by workflow agents. A workflow example could be an Available to Promise Calculation. When a customer wants to place an order, a calculation should be made whether the product can be delivered in time. This calculation might depend on multiple companies in the supply chain. A mobile workflow agent could go to the ERP systems of suppliers and check whether or not the resources are available to manufacture and deliver the product in time. For questions with regard to workflow in ERP systems, contact Riné le Comte, rlcomte@baan.nl.
The Mobile Agent System that is currently developed at Baan Labs makes use of the Knowledge Query and Manipulation Language to provide for communication between a mobile agent and other components in the system. This communication is performed asynchronously. The communication takes place at a semantic level, in order to abstract from implementation. We make the assumption that there is no knowledge with respect to the actual implementation of a specific concept in another application. However, these concepts must have been described in an ontology. Currently the Mobile Agent System also supports the use of

synchronous communication between a mobile agent and a so-called local object (an object at the JVM where the Mobile Agent currently is present and that has published its services). This communication takes place via direct Java method calls.
Joint work with: Manfred Dalmeijer - mdalmeijer@baan.nl

19 Java Smart Cards as Secure Devices for Mobile Agents

Joachim Posegga
TZ Deutsche Telekom AG

Java-based Smart Cards are a very recent development that offers the possibility of uploading executable code into a smart card and running it. Since smart cards are usually considered to be tamper-proof, such cards can be potentially useful for solving the problem of protecting mobile agents from malicious hosts as follows: A trusted third party distributes such smart cards with an asymmetric pair of cryptographic keys to host in a network that can be visited by agents. The 'sensible' parts of those agents code is encrypted with the public key of the cards. If this part of the agent's code is supposed to be executed, it is moved into the smart card, decrypted, and run. Thus, the 'sensible' part of the code is never visible outside the card, but can only be run in a trusted device attached to the host the agents visit. Currently Java-based smart cards are clearly too limited for a 'serious' application of this principle. However, technology evolves, and future cards will offer enough resources to use them in this way.

20 On the Exactly-once Semantics for Agents

Kurt Rothermel
University of Stuttgart

Mobile agent technology has been proposed for various application areas, including electronic commerce, system management, and active messaging. Many of these applications require agents to be performed exactly once, independent of communication and node failures. In other words, once an agent has been launched, it must never be lost before execution is finished. Moreover, each 'portion' of the agent performed at the visited nodes is performed exactly once. We will describe a protocol that ensures the exactly once property of agents and additionally reduces the blocking probability of agents caused by node failures and network partitioning. The approach is to have a number of so-called observer nodes that monitor the progress of an agent. To ensure the exactly once property even in the presence of node failures and network partitioning, a voting process is integrated in 2 phase commit processing.

21 Mechanisms and Policies for Secure Mobile Code in Tacoma Too (T2)

Fred B. Schneider
Cornell University

T2 is an ML dialect for programming agents that will control an active network. This talk concerns T2's mechanisms to protect agents and hosts from attacks by agents. Agents execute in domains, which define what instructions may next be executed. The T2 runtime system supports domain enforcement and provides operations for changing domains. Domain enforcement is achieved by controlling the visibility of names through scope, through the use of sparse name space, and by controlling how names are interpreted.
What class of security policies can be enforced with these mechanisms? We show that all enforceable security policies can be enforced. The proof involves formalizing 'security policy' and 'enforceability' in a natural way. A universal language for enforceable security policies is the language of security automata, a subclass of Büchi automata. This class of automata is defined, and we discuss methods for enforcement given a policy specified as a security automaton.

22 Fast and Efficient Marshalling for Java Mobile Agents

Miguel Mira da Silva
Universidade de Evora

The time to migrate a Java agent seems to me a major concern in the agent research community. Times from seconds up to minutes have been reported as necessary to migrate a single agent, even on Ethernet-based LANs. However, other researchers have achieved far better results. Why there is such a difference? The difference is that in a type-safe, garbage-collected language like Java there is a high penalty to be paid if marshalling is made at the language level. This is the case with "pickling" in Modula-3 and "Object Serialization" in Java. Object serialization, in particular, has been used in most Java-based agent systems because it provides a simple, ready-to-use library to migrate agents between Java programs.

Marshalling at the language level is slow and inefficient because the inexistence of object identifiers forces the marshalling time to grow O(N*N) where N is the number of objects being marshalled. This problem, however, is solved when marshalling is performed below the type-safety level. Inside the Java virtual machine, C pointers can be used instead of object identifiers and efficient times O(N*log(N)) can be achieved. (Further details can be found on my PhD thesis.) Marshalling in C instead of Java is also much faster just because C is much faster – at least 2 orders of magnitude – than Java.

The main point is: fast, efficient marshalling is possible for large graphs of objects in Java. For example, a graph with 10,000 objects can be marshalled in 30 minutes or half a second, depending where it is being performed. Unfortunately, as the virtual machine has to be extended, this solution cannot be provided as a Java library. This disadvantage is somewhat minimized by the fact that most agent systems require an "agent server" to be installed on top of the Java VM anyway before migrating any agent.

23 Mobile Agents: Observations from an Outsider

David Steere
Oregon Graduate Institute of Science and Technology

As the last talk of this seminar, I take the liberty of summing up what I've heard so far, wearing that hat of an interested observer. I'll give my stab at what I consider to be the key questions asked during this seminar: 'What is a Mobile Agent?' and 'What applications domains are appropriate for mobile agents?'

24 Security for Mobile Agents

Vipin Swarup
Mitre Corporation

Currently, distributed systems employ models in which processes are statically attached to hosts. Threats, vulnerabilities, and countermeasures for these systems have been studied extensively and sophisticated security architectures have been designed. Mobile agent technology extends this model by including mobile processes, i.e., processes which can autonomously migrate to new hosts. Although numerous benefits are expected, this extension results in new security threats from malicious agents and hosts. A primary added complication is this: As an agent traverses multiple machines that are trusted to different degrees, its state can change in ways that adversely impact its functionality.
We have developed a mobile agent security architecture that extends an existing distributed system security architecture with special mechanisms that provide security in the presence of migrating stateful agents. The basic principals of this architecture are authors of programs, the programs themselves, senders of agents, the agents themselves, and interpreters that execute agents. Crucial events in an agent's life are the creation of the underlying program, creation of the agent, migration of the agent to a new execution site, remote procedure calls, and termination of the agent. These events cause complex trust relationships between principals, e.g., the trust placed by authors and senders in agents, the trust placed by an agent in the interpreters that execute it, and the trust placed by an interpreter in the agents it is executing. When an agent requests an operation

on a resource, the interpreter uses its access rules and these trust relationships to derive authorization for the request.
We have used the theory of authentication of Abadi et al to formalize the trust relationships in a generic mobile agent system and have designed our security architecture based on this work. For instance, a fundamental invariant in our system is that an interpreter "speaks for" the agents it is executing. Thus an agent must trust the interpreters that execute it. Trust is managed by controlling the principals under which the agent executes as it migrates between interpreters. Agent creation and migration can use either handoff or delegation semantics and the protocols ensure that the above invariant is maintained.
A novel aspect of our architecture is a "state appraisal" mechanism that protects against attacks via agent state modification and that enables an agent's privilege to be dependent on its current state. Checking the integrity of an agent's state is difficult since the state can change during execution and hence cannot be signed. Our agents carry a state appraisal function that checks whether the agent's state meets expected state invariants; the function returns a set of permits based on the agent's current state.

25 Security and Fault Tolerance for Mobile Agent Systems

Hartmut Vogler
Darmstadt University of Technology

Mobile agents offer a new possibility for the development of applications in distributed systems. For new and useful agent-based applications, mobile agents have to be equipped with electronic commerce capabilities.
In this talk we present an enhancement of mobile agent systems for security, fault tolerance, and electronic commerce. The electronic commerce capabilities are based on the concept of First Virtual which we extended for the special requirements of mobile agents.
To achieve consistency and fault tolerance of the whole agent system, especially for the agent transfer to a new host, we use Distributed Transaction Processing (DTP) based on the different standards like the OMG Object Transaction Service, X/Open DTP and the recently published Transaction Internet Protocol (TIP) by Microsoft and Tandem. This offers the exactly-once-semantics for the agent transfer.
This system can be used on top of existing mobile agent systems, e.g. as an

enhancement of the GO-statement, and offers a high reliability due to the implementation based on standardized components.

26 Is Transparent Migration Worth It? Lessons from Distributed Operating Systems (and Other Observations)

Brent Welch
Sun Microsystems

The main point of this talk is that transparent process or agent migration may be possible, but the cost in implementation complexity and performance may be too high in comparison to a simpler remote execution model. This lesson comes from experience with the Sprite distributed operating system [Douglis90] [Nelson88] [Welch90].

Sprite provided a single system image through a shared network file system. Process migration was used to exploit CPU cycles on idle workstations. The file system provided complete Unix semantics and a high performance consistent caching system. This required stateful servers that knew how clients were using their files. We found it difficult to keep process migration working, even though we planned for it from the beginning. (We wrote the Sprite kernel from scratch.) Migration support affected many different parts of the system. Ultimately we got everything to work and had excellent system for software development. However, our distributed compilation system could have been achieved with a simpler remote execution facility. The other main user of migration were long running simulations. However, these could have used a standard checkpoint/restart system that would have provided additional robustness.

Why is migration hard? There are two aspects of process state: execution state and communication channels. Execution state is relatively straight-forward, and there are several schemes for transferring the virtual memory of a process. The main problem with execution state is its size. It can be expensive to transfer a large process. The communication channels are more difficult. In Sprite, updating the server state as clients migrated was tricky because I/O channels can be shared between processes. In other distributed operating systems that provided migration, the communication channels also added the most complexity to the system.

In contrast, remote execution is much simpler because existing mechanisms can be reused. Starting a process or agent is quite normal, as is the initial establishment of communication channels. Witness the extreme simplicity and good performance of the TACOMA system that is based on a remote execution model [Johansen97].

Other observations about mobile agents.
Scripting languages seem ideally suited for mobile code applications. Portable interpreters provide machine-independent execution platforms. Scripts are generally small and compact so they are efficient to transmit. Scripting languages provide introspection and dynamic code generation so an agent can essentially compute a new program and transmit it to a remote site.
A big problem for mobile agent technology is widespread deployment of compatible systems. Web servers seem like an ideal agent host because they are co-located with lots of interesting data. If the mobile agent community could transform web servers into agent hosts then it would rapidly accelerate progress in the field. I've built a Tcl web server [Tclhttpd] that can be used for this purpose. Another ideal host for agents is email user interfaces and web browsers. These are ideal for fostering agent to people communication.

References

[Douglis90] F. Douglis, "Transparent Process Migration for Personal Workstations", PhD Thesis, Sep. 1990. University of California, Berkeley.

[Nelson88] M. Nelson, B. Welch and J. Ousterhout, "Caching in the Sprite Network File System", ACM Transactions Computer Systems 6, 1 (Feb. 1988), 134-154.

[Welch90] B. B. Welch, "Naming, State Management, and User-Level Extensions in the Sprite Distributed File System", PhD Thesis, 1990. University of California, Berkeley.

[Johansen97] Dag Johansen, Nils P. Sudmann and Robbert van Renesse, "Performance Issues in TACOMA", 3rd. Workshop on Mobile Object Systems, 11th European Conference on Object-Oriented Programming, Jyvaskyla, Finland, 9-13 June 1997.

[Tclhttpd] httpd://sunscript.sun.com/products/tclhttpd/