Open Resource Allocation for Mobile Code

Christian F. Tschudin <tschudin@ifi.unizh.ch>
Computer Science Department, University of Zurich, Switzerland and
International Computer Science Institute (ICSI), Berkeley, USA

Abstract. Mobile code technology leads to a new type of "open systems": instead of applying openness to a standardization process we now require the running systems to become open for foreign code. The question then is how far this technical openness can go for mobile code. The less constraints we impose on hosts running mobile code, the more can the benefits of mobile code be exploited. However, there must necessarily be basic constraints regarding the utilization of resources which are always finite and most of the time will be operated near the saturation point. In this paper we argue in favor of openness even at the level of resource allocation. We link this topic to (open) market models, describe the mechanisms we developed so far for communication messengers and show how they are used to allocate resources in an open way. Finally we present experimental results of validation runs which help us testing these mechanisms.

Keywords: Mobile code, communication messengers, open resource allocation, market, computational ecosystems.

1 Introduction

An important contribution of the computer network's standardization approach in the 80ies was the concept of open systems: open access to the standardization process and open decision procedures were the basis of acceptance, leading to communication systems where components from different vendors could be intermixed. This applied to interworking of "black boxes" like bridges or routers, but also to software components inside a host where application programmer interfaces laid the ground for portable software (e.g., POSIX). The technical solutions involved in these open systems, however, are themselves not open i.e., leaving little choice on how a given service is to be implemented. This is different with open distributed systems based on mobile code where users can program the network and influence the way a service is provided. Technically spoken this means that the implementation of the network's element now become open: hosts have to accept virtually arbitrary code coming from potentially unknown users.

Such technically open distributed systems have a special execution model: distributed applications are performed by multiple threads of execution, some of them being mobile i.e., started remotely and on demand. The latter point means that hosts may be faced with a large number of uncoordinated (and not coordinable) requests for executing mobile code, which always involve local computing resources. Imagine a global mobile code network consisting of a billion nodes serving some hundred millions of users each having ten to thousand mobile agents roaming around. In such a context it is virtually impossible to do resource authorization i.e., mapping from user to agent and further down to access rights. But even if this would be feasible, it would not solve the problem to arbitrate conflicting resource requests in situations of saturation, which we assume will be the normal mode of operation of such a system.
Thus, the challenge is twofold. First we want to find methods of doing resource allocation without relying on the impractical identity of mobile agents. And second, we want resource allocation to be open, that is, to let the resource allocation be steered by mobile agents for mobile agents instead of delegating this task to the hosts. We are investigating these goals with rather small mobile agents we call communication messengers.

1.1 The Messenger Approach

At the heart of messengers is the paradigm of instruction-based communications [11]. Based on a very simple communication model, namely the exchange of pure instructions and their mandatory detached execution, we obtain a universal communication environment in which classical communication protocols need not to be known and installed at a remote site. Using instructions it is perfectly possible to advice a remote host (also called execution platform) how to perform any protocol. Accidentally, this concept is also at the center of mobile software agents as it relies on the programming of remote computers. According to our definition a messenger is a thread of control which can trigger the transfer of arbitrary code to a remote computer such that: (i) this code has a chance to be executed remotely and independently, and (ii) the resulting remote thread may trigger another code transfer.

The messenger approach extends the instruction-based communication model with design guidelines that try to keep the resulting communication architecture as universal as possible. The most important design principle is strict locality in the sense that no host should provide a service that requires the cooperation of other hosts. Cross-platform services should be implemented by messengers themselves, leaving the messenger execution environment completely protocol-neutral. Another design decision is that messengers should be anonymous, considering that agent identity (and related concepts like agent ownership etc.) is a cross-platform service which must be handled above the platform level.

The challenge with this approach is to design and build a working environment which has nice “global” properties. Because there may not be any coordination work done by the execution environments, the well-behavior of a messenger-based network fully relies on the decentralized decisions taken by independent messengers. The good global behavior thus has to “emerge” from local behavior and cannot be imposed in any way although it can be influenced by defining the border conditions that messengers have to obey. The believes underlying our research is that a set of instructions, local services and platform behaviors can be designed such that good global behavior becomes possible. Section 2 introduces and discusses the concept of open resource allocation. Section 3 describes the mobile code environment $\mathcal{M}$ and emphasizes on the aspects of memory and money for messengers. How openness is achieved in this environment is explained in section 4, which also reports on experimental results obtained. Related work is briefly discussed in section 5, just before the concluding summary.

2 Open Resource Allocation

Messengers are quite small and minimalistic mobile agents, wherefore we found it more appropriate to look at resource allocation at a very fine granularity. But even if larger units are considered, a mobile code system must provide mechanisms to allocate its resources to the roaming agents. We first elaborate on the “openness” in resource