Law-Governed Internet Communities

Xuhui Ao, Naftaly Minsky, Thu D. Nguyen, and Victoria Ungureanu

Rutgers University, New Brunswick, NJ 08903, USA,
{ao,minsky,tdnguyen,ungurean}@cs.rutgers.edu

Abstract. We consider the problem of coordination and control of large heterogeneous groups of agents distributed over the Internet in the context of Law-Governed Interaction (LGI) [2,5]. LGI is a mode of interaction that allows a group of distributed heterogeneous agents to interact with each other with confidence that an explicitly specified policy, called the *law* of the group, is complied with by everyone in the group.
The original LGI model [5] supported only *explicit* groups, whose membership is maintained and controlled by a central server. Such a central server is necessary for applications that require each member of the group to know about the membership of the entire group. However, in the case where members do not need to know the membership of the entire group, such a central server can become an unnecessary performance bottleneck, as group size increases, as well as a single point of failure.
In this paper, we present an extension to LGI allowing it to support *implicit groups*, also called *communities*, which require no central control of any kind, and whose membership does not have to be regulated, and might not be completely known to anybody.

1 Introduction

We consider the problem of coordination and control for large heterogeneous groups of agents distributed over the Internet in the context of Law-Governed Interaction (LGI) [2,5]. LGI is a mode of interaction that allows a group of distributed heterogeneous agents to interact with each other with confidence that an explicitly specified policy, called the *law* of the group, is complied with by everyone in the group. LGI has been designed specifically to satisfy the following principles, which we consider critical for coordination in large heterogeneous systems: (1) coordination policies need to be formulated explicitly rather than being implicit in the code of the agents involved, (2) coordination policies need to be enforced, and (3) the enforcement needs to be decentralized, for scalability. LGI has been implemented in a toolkit called Moses, which has been applied to a broad range of coordination and control applications, including: on-line reconfiguration of distributed systems [6], security [4], and electronic commerce [3].

A group of agents interacting via LGI under a given law $\mathcal{L}$ is called an $\mathcal{L}$-group. LGI distinguishes between two kinds of $\mathcal{L}$-groups, called *explicit* and *implicit* groups, that differ in the manner in which a group is deployed and in the management of its membership. Explicit groups have been discussed in detail in [5]. The purpose of this paper is to introduce implicit groups, also called

A. Porto and G.-C. Roman (Eds.): COORDINATION 2000, LNCS 1906, pp. 133–147, 2000.

communities, which are more general than explicit ones, and more suitable for very large groups of heterogeneous agents operating over the Internet.

Currently, an explicit $\mathcal{L}$-group $\mathcal{G}$ is established in Moses by creating a distinguished agent called the *secretary* of $\mathcal{G}$, denoted as $\mathcal{S}_\mathcal{G}$, and defining into it the law $\mathcal{L}$, and specifying the initial membership and structure of $\mathcal{G}$. Subsequent to its initialization, $\mathcal{S}_\mathcal{G}$ serves as a "gateway" to the group by admitting new members into it, subject to law $\mathcal{L}$. $\mathcal{S}_\mathcal{G}$ also functions as a name-server for the group, helping members to find each other's location, and to verify mutual memberships in the same group.

Such a secretary is necessary whenever the entire membership of the group needs to be known, and it is appropriate for relatively small groups. This is the case, for example, for a group operating under a token-ring protocol, where the structure of the group, i.e., the placement of its members along a ring and the existence of a single token among the members of the ring, are essential to protocol. This ring structure can be defined by the secretary of the group as its initial state, and, as demonstrated in [6], can be maintained as an invariant, even if the membership of the group is allowed to change dynamically.

But such group management is neither necessary nor appropriate where no knowledge of the entire group membership is required, or available, and where the size of the group is too large to be comfortably handled by a single secretary. An everyday example for such a situation is provided by the group of all car drivers in a given city. All these drivers must obey the traffic laws of the city, but generally there is nobody that knows the names of all these drivers, or their total membership. Such conditions are becoming increasingly common in modern distributed computing, as is illustrated by the following example.

Consider a distributed set of databases servers that provides access to an heterogeneous set of clients. Suppose that for a client to consult an item in a database or to update it, it needs to lock the item first. It is possible for a single agent to maintain locks for several items (at several databases) at a time. It is well known that this activity would be *serializable* if the following kind of two-phase locking (TPL) protocol is strictly observed by all clients [10]:

> New locks cannot be acquired after the first release of a lock (until the agent has released all locks that it currently holds). That is, each transaction (representing some set of changes) is divided into two phases: a *growing phase* of locking, and a *shrinking phase* of releasing locks. A locked resource can be used during both phases.

While this protocol can be enforced by a *central coordinator* that mediates the interaction between the distributed set of servers and their clients, such coordination would be quite unscalable. Under LGI, on the other hand, this protocol can be formulated as a law $\mathcal{TPL}$ that is enforced locally at each client, allowing for scalability, provided that the set of servers and their clients is not maintained as an *explicit* $\mathcal{TPL}$-group. Because the number of clients in this case might be very large, the requirement that each of them enters the group via a single secretary would create a bottleneck and a dangerous single point of failure. Moreover,