Chapter 36
Enforcing Fairness in Asynchronous Collaborative Environments

Guillaume Pierre
VU University Amsterdam, The Netherlands

Maarten van Steen
VU University Amsterdam, The Netherlands

ABSTRACT

Many large-scale distributed applications rely on collaboration, where unrelated users or organizations share their resources for everyone's benefit. However, in such environments any node may attempt to maximize its own benefit by exploiting other's resources without contributing back. Collaborative systems must therefore deploy strategies to fight free-riders, and enforce collaborative behavior. This chapter explores a family of mechanisms to enforce fairness in asynchronous collaborative environments, where simple tit-for-tat policies cannot be used. Our solutions rely on enforced neighborhood relations, where each node is restricted in the choice of other nodes to collaborate with. This creates long-term collaboration relationships, where each node must behave well with its neighbors if it wants to be able to use their resources.

INTRODUCTION

Service-Oriented Architectures offer the vision of distributed applications offering functionality to each other, such that complex applications may be realized mostly by composition of existing software components provided by independent parties. An example is the recent popularity of Web mash-ups, where any programmer can take advantage of service-oriented functionality offered through Web services. However, users of service-oriented applications do expect reasonable performance. This requirement can be translated into: how can a service provide constant performance regardless of the request load addressed to it by independent third parties? Obviously, a single server machine cannot handle arbitrary amounts of load so one must design services such that they can expand their capacity by using additional computing resources when necessary.

One way a service may obtain temporary access to extra resources when it needs them is the use of

DOI: 10.4018/978-1-61520-686-5.ch036

collaborative environments. Such environments are characterized by multiple users sharing their resources for everyone's benefit. For example, peer-to-peer file sharing applications can improve everyone's download speed of a file under the condition that those users are willing to donate their resources to upload file contents (Cohen, 2003). Similarly, a service operator may use a collaborative content delivery network, which relies on the willingness of Web server administrators to help each other if one experiences a temporary overload (Pierre and van Steen, 2006); another possible method is to use grid computing, where system administrators are willing to contribute their resources in exchange for future use of global resources (Foster and Kesselman, 1998).

One important issue in such environments is free riding, where some users try to use the shared resources without contributing an equivalent quantity of resources back to the system (Adar and Huberman, 2000). Free riding can be extremely detrimental to the performance of collaborative systems as it decreases the quantity of resources available to users as a whole. Additionally, it produces strain on the remaining good nodes in the system, which reduces the incentive to contribute positively.

An efficient mechanism to enforce collaboration is the tit-for-tat policy, as implemented for example in the BitTorrent file sharing system (Cohen, 2003). This policy dictates that, after a first altruistic interaction, resources are granted to a user under the condition that an equivalent amount of resources is simultaneously contributed back. A few properties of BitTorrent make this scheme effective and easy to apply. First, collaboration is symmetric, meaning that collaboration happens pairwise with no third party involved. Fairness enforcement can thus be realized by the two concerned peers themselves, without requiring the need for external services such as reputation systems. Second, collaboration is local in time in that the balance of respective contributions must be judged as fair by both parties at any instant of the collaboration. The system therefore does not need to maintain a memory of past interactions.

However, tit-for-tat is not a panacea for solving all fairness issues in collaborative environments. Many such environments rely on asynchronous collaboration, where the services are not provided simultaneously from A to B and from B to A. A good example is a collaborative content distribution network, where Web servers call each other for help only when they experience an overload. For such environments we need more sophisticated mechanisms.

This chapter explores a family of mechanisms to enforce fairness in asynchronous collaborative environments, based on observations from Axelrod (Axelrod, 1984). These observations state that cooperation can emerge only when:

- Nodes retain a unique identity over time
- Interactions are repeated many times between the same pairs of nodes

The intuition between these rules is that, to sustain collaboration between the well-behaving members (and to exclude free-riders), one must rely on some memory of past interactions with other nodes. In a large-scale system this implies that a given node regularly interacts with the same partners over and over again, to have the ability to gain some confidence that they will behave well in the future.

The following sections explore the application of these general principles to two classes of asynchronous collaborative environments. First, we study fairness enforcement in a collaborative content distribution network. In such a system, Web servers may request each other's help when they experience an overload. The interaction is thus necessarily asynchronous because a currently overloaded server cannot be of much help to another overloaded server. It makes more sense that underloaded servers help overloaded servers, thereby creating asynchronous collaboration. We then turn to peer-to-peer grids, where a user can