Available online at www.sciencedirect.com
SCIENCE DIRECT®
ELSEVIER
Electronic Notes in Theoretical Computer Science 128 (2005) 59–77
Electronic Notes in Theoretical Computer Science
www.elsevier.com/locate/entcs

Quantitative Assessment of a Peer-to-peer Cooperative Infrastructure Using Stochastic Well-Formed Nets

Carlo Bellettini[1] Lorenzo Capra[2] Mattia Monga[3]

DICo – Dipartimento di Informatica e Comunicazione
Università degli Studi di Milano, Italy

Abstract

Traditional support tools for software engineers, normally based on a client-server architecture, are unsuitable to deal with the new issues emerging from the current (and future) cooperative work scenarios (where connectivity is intrinsically transient, the number of interacting partners dynamically changes, etc.). This paper presents a quantitative assessment of a fully decentralized, peer-to-peer, cooperative infrastructure. Stochastic Well-Formed Nets (SWNs) modelling the new peer-to-peer architecture, and a traditional (client-server) one, are developed and analysed: we used SWNs for their ability to directly exploit the symmetries intrinsically present in the modelled systems, thus greatly reducing the complexity of the analysis. The main goal is to compare the impact of the two alternative protocols on the collaborative work. Together with the performance figures of interest, methodological issues concerning the choice of the most appropriate model abstraction level, the adoption of a compositional modelling approach, and the management of the model complexity are also discussed.

Keywords: SWNs, peer-to-peer, configuration management

1 Introduction

Nowadays the Internet infrastructure is so pervasive that it is common that people connect their laptop computer from a range of different locations: office, home, the hotel hosting them for a conference, or the meeting room where they

[1] Email: carlo.bellettini@unimi.it
[2] Email: capra@dico.unimi.it
[3] Email: mattia.monga@unimi.it

doi:10.1016/j.entcs.2005.01.013

are working. This is sometimes called mobile computing and it forces designers of applications to take into account two new requirements: (1) users may connect to the network from arbitrary locations (usually with different network addresses), and (2) they are not permanently connected. Thus, connectivity is intrinsically transient, and machine disconnection is not an exceptional case, but the normal way of operating.

Collaborative work over the Internet is often pursued by manipulating electronic artifacts that are shared with others by exploiting the mediation of some server. This scenario is still common and useful in several cases. However, since work activities are no more exclusive of office settings, users are often forced to work without any Internet connection. Thus, network applications that rely on servers are sometimes not desirable or even not feasible. In other words, the client/server approach is possible, and common, in all cases where a reliable and permanent network infrastructure is available to connect the participating nodes. However, in many cases, people would like to collaborate while they are supported through a much looser architecture, since they cannot or do not want to afford the cost of setting up a central server. The resulting reference architecture is a network of peers, each of which contributes to the overall logical structure in an equivalent way. Moreover, peers cannot be assumed to be always on-line, i.e., a peer is not always reachable by others. Since the network connection is intrinsically intermittent, peers may dynamically join and leave an ad-hoc community. They join it in impromptu meetings, where they synchronize their works.

We focussed our investigation on the collaborative work needed to produce software systems. Software developers typically collaborate by exchanging and sharing a number of files. Files are assigned to people according their responsibilities in the project. However, besides the person in charge of a file, several other collaborators sometimes need to view or modify it. In general, for each item we can identify the role of an *owner,* i.e., the individual who has created the artifact or who is in charge of carrying out the work on it. Moreover, there is a number of other collaborators involved in the project who need to manipulate items that are not under their direct control, i.e., artifacts they do not own. System designed to help software developers in keeping their work coherent and tracking the evolution of artifacts are called *configuration management tools.*

The underlying architecture of these tools affects the ongoing collaborative work. In fact, a server based solution forces people to keep their work consistent with the copy on the server machine. In traditional, client/server version management tools, two (or more) persons may work on the same artifact (a file): both are required to check-out the artifact from the server machine and