Chapter 11

Guaranteeing Correctness for Collaboration on Documents Using an Optimal Locking Protocol

Stijn Dekeyser
University of Southern Queensland, Australia

Jan Hidders
Delft University of Technology, The Netherlands

ABSTRACT

Collaboration on documents has been supported for several decades through a variety of systems and tools; recently a renewed interest is apparent through the appearance of new collaborative editors and applications. Some distributed groupware systems are plug-ins for standalone word processors while others have a purely web-based existence. Most exemplars of the new breed of systems are based on Operational Transformations, although some are using traditional version management tools and still others utilize document-level locking techniques. All existing techniques have their drawbacks, creating opportunities for new methods. The authors present a novel collaborative technique for documents which is based on transactions, schedulers, conflicts, and locks. It is not meant to replace existing techniques; rather, it can be used in specific situations where a strict form of concurrency control is required. While the approach of presentation in this article is highly formal with an emphasis on proving desirable properties such as guaranteed correctness, the work is part of a project which aims to fully implement the technique.

DOI: 10.4018/978-1-4666-2647-8.ch011
INTRODUCTION

Although some lesser known approaches exist (for example those described in Dekeyser et al., 2004; Oster et al., 2007), collaboration on documents is typically realized in one of two distinct ways. In the asynchronous setting collaborators typically check out a document from a central repository, make changes off-line over an extended time, and then check in their version of the document at which time they are typically asked to solve inconsistencies with the version currently stored in the repository. This type of collaboration has been possible since the introduction of the Source Code Control System (SCCS) in the early seventies of the previous century (Rochkind, 1975). Successors of SCCS have included RCS, CVS, and Subversion. The last of these remains in widespread use and for example provides the collaboration functionality of modern document collaboration systems such as ICE (Sefton et al., 2006).

The second approach to collaboration, found in most modern CSCW (Computer-Supported Cooperative Work) systems, involves synchronous collaboration where editors are aware of other users’ changes while working on their own content. This approach requires on-line communication and uses a replicated architecture: shared documents are replicated at local sites such that each works on their own local copy and changes are propagated to other users. Such systems are called real-time: the response for local operations is quick and the latency for remote operations is relatively low. Examples of real-time editors include CoWord (Xia et al., 2004), Google Docs, and the short-lived Google Wave experiment (which was more than just a real-time editor, but that is beyond the scope of this article). The newest such systems call themselves really real-time: local operations are applied immediately, and remote operations are applied within seconds. EtherPad (Corneli, 2010), recently acquired by Google, is an example of a really real-time editor.

In this article we present an alternative approach that aims to preserve the strengths of the existing approaches while avoiding their weaknesses. A shorter preliminary version of this work was presented in Dekeyser and Hidders (2010).

The article is organized as follows: the next section will describe the two approaches mentioned above in greater detail and will compare them to our proposed technique. Subsequently we present the theory of piecemeal over a number of sections that progressively refine the model of documents that we work with; this concludes with a proof of the optimality of our protocol. Following this we discuss some practical implications for document servers and describe how offline editing is supported. In the conclusion we summarize the article’s contribution and list expected future work.

CURRENT TECHNIQUES

We briefly discuss existing collaboration techniques and highlight some of their strengths and drawbacks. We limit this discussion to the two dominant methods introduced above.

Version Control Systems

As discussed above, Subversion is currently the most widely used versioning control system and has its foundations in SCCS. The underlying collaboration technique is well understood and robust, although not grounded on a formal model or supported by a theory of correctness. While this technique sometimes fails to prevent conflicts, it seems to have an intuitive notion of conflict as well as a limited notion of transaction. The technique makes use of a so-called diff between two text documents, at the level of individual lines. At its most basic level, a diff is represented by individual lines of text that are marked as additions, lines that are marked as deletions and lines that are not marked and that are used as a context to help