Project no. 034567

Grid4All

Specific Targeted Research Project (STREP)
Thematic Priority 2: Information Society Technologies

D6.7_Telex: A Semantic Platform for Collaborative Applications

Due date of deliverable: September, 2009.

Actual submission date: 15th July 2009.

Start date of project: 1 June 2006 Duration: 37 months

Organisation name of lead contractor for this deliverable: INRIA Regal

Revision: Draft of 15th July 2009

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)

Dissemination Level

PU	Public	√
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Contents

Abbreviations and acronyms used in this document

Abbrev./acronym	Def.	Description
OR		Optimistic Replication
Grid4All		The FP6 STREP project that aims to democratise access to the Grid, through the use of peer-to-peer technologies. The co-ordinator is France Télécom; the other partners are a SME from Spain, and research labs from Greece, France, Spain, and Sweden.

Grid4All list of participants

Role	Part. #	Participant name	Part. short name	Country
CO	1	France Telecom	FT	FR
CR	2	Institut National de Recherche en Informatique en Automatique	INRIA	FR
CR	3	The Royal Institute of technology	KTH	SWE
CR	4	Swedish Institute of Computer Science	SICS	SWE
CR	5	Institute of Communication and Computer Systems	ICCS	GR
CR	6	University of Piraeus Research Center	UPRC	GR
CR	7	Universitat Politècnica de Catalunya	UPC	ES
CR	8	ANTARES Produccion & Distribution S.L.	ANTARES	ES

Executive summary

Collaborative work requires multiple users to be able to update a shared content at the same or different times. Replicated and optimistic updates meaning users are able to immediately and speculatively update a copy of the content, and will eventually see other users' updates. Concurrent updates may conflict, therefore updates remain tentative until conflicts are resolved and users agree eventually on a common state.

Currently there are a number of collaborative applications on the market. Examples include shared text editors, collaborative calendars, collaborative code repositories, and so on. Those applications generally implement ad-hoc approaches to deal with conflicts and eventual agreement, leaving much manual work to the user.

In contrast, Telex is a principled and generic solution which takes care of replication, conflict resolution and eventual agreement.

The Telex programming API allows programmers to implement collaborative applications more easily than before. Users can collaborate in a more productive way, as Telex is guided by application provided-parameters capturing application semantics and user intents.

Telex is available at gforge.inria.fr/projects/telex2 under a BSD licence.

1 Introduction

Remote and offline data sharing are increasingly important. Examples include a shared wiki, cooperative offline editing with Google Gears, enterprise platforms such as Notes or Groove, collaborative code repositories CVS or SVN, and so on.

To deal with failures, latency, and large scale, a common approach is *optimistic replication (OR)* . OR decouples data access from network access: it allows a processor to access a local replica without synchronising. A site makes progress, executing uncommitted actions, even while others are slow or unavailable. Local execution is tentative and actions may roll back later. An OR system propagates updates lazily, and ensures consistency by a global a posteriori agreement on a same state.

Implementing a collaborative application is complex. The programmer needs to manage asynchronous communication between collaborators, data replication, conflict detection and resolution, and eventual agreement. Besides, end-users should be able to make sens of the collaborative work: (i) they should not be overwhelmed or confused by remote updates and roll-back, (ii) conflict detection should be relevant, (iii) and users should be able to express preferences for conflict resolution. Additionally, the application needs to be responsive, and to provide garanties about the data persistence and convergence.

To solve these issues, current collaborative applications use ad-hoc approaches that do not guarantee correctness, and leave much manual work to the user.

In contrast, we propose an open platform called Telex that helps developers to build collaborative applications faster. Telex supports an optimistic replication model for sharing stateful data in a decentralised way over a large-scale network. Telex eases application development by taking care of the communication, replication and consistency issues. Based on a principled approach, Telex guaranties that replicas never violate safety and converge eventually. The Telex platform is designed for robustness, flexibility and performance.

2 Telex Overview

Programming an application using Telex API requires understanding some basic concepts. We start introducing some terminology.

Application A Software above Telex allowing users to collaboratively work on some shared stateful data.

Action An atomic operation performed by the application. For instance, inserting a line in a text file is an action.

Constraint A semantic relation between two actions. For instance, an action inserting a line in a text file and another updating the same line, relate by a *causal* constraint.

Conflict A specific constraint expressing an incompatibility between two actions. For example, a conflict occurs when two actions concurrently update the same line in a text file.

Document Any shared mutable content managed by an application. Actions and constraints relate to a document.

Schedule A non-conflicting sequence of actions.

Document state The current state resulting from the execution of a schedule.

Agreement A concept introduced by the optimistic execution model. The agreement is committing on a common correct state for all users.

2.1 Architecture Overview

Telex is a generic platform to ease development of collaborative applications. Telex supports optimistic sharing over a large-scale network of computers or *sites*. Telex allows the application programmer to concentrate on core functionality, delegating distribution, replication, persistence and consistency issues to Telex. The Telex programming API enables the integration of the application semantics in the process. Telex implements a classical approach for optimistic replication: updates are logged, propagated and eventually re-executed at each user's site.

We now present a Telex life-cycle referring to Figure 1. Telex drives application progress as the controller.

1. The application opens a document using Telex open primitive. Telex imports a copy of the corresponding document.

2. A user utters a command to the application. The application computes the corresponding actions. Actions can be augmented with constraints, explained in more details in the example hereafter. The application entrust Telex with a set of actions and constraints by calling the *addFragment* primitive of Telex (Step 1 in Figure 1) . Telex logs them, stores them persistently and propagates them to each site replicating the document.

3. To verify whether an action received from a remote site causes a conflict, Telex up-calls the application's *getConstraint* interface with a pair on actions to check, (step 2 in figure 1).

4. Telex iteratively computes a schedule that: *(i)* combines actions received so far, and *(ii)* satisfies all the application-provided constraints. In the presence of conflicts, multiple alternative schedules are possible.

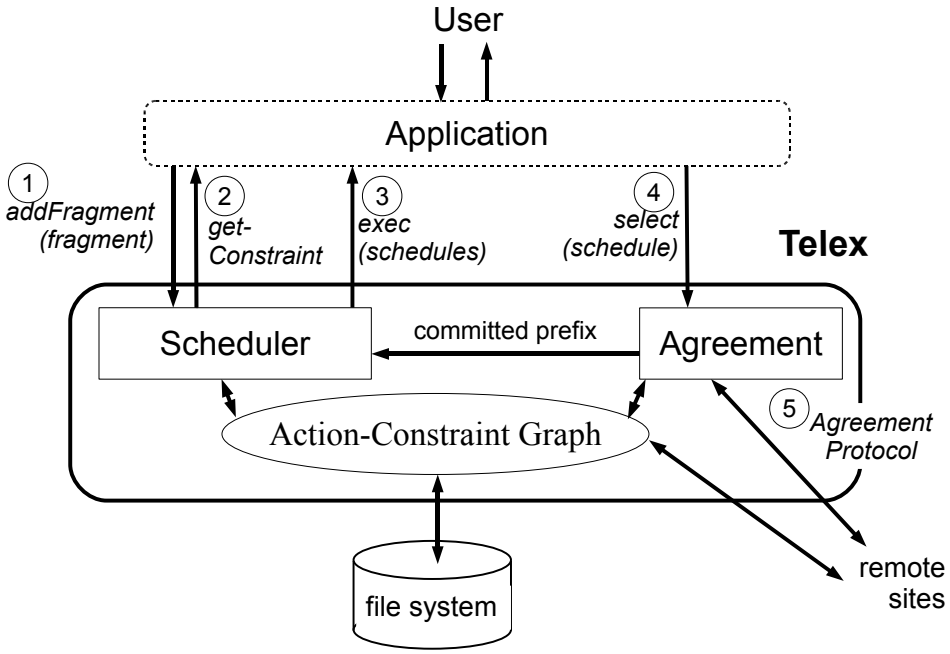

Figure 1: Telex site architecture and execution model

Telex instructs the application to execute this schedule by up-calling the application's *exec* interface, (Step 3 in Figure 1). This may be repeated several times (at the application's request) for alternate schedules.

5. If the user or the application is satisfied with the resulting state, it calls Telex's *select* interface (Step 4 in Figure 1). This encourages Telex to extend the same schedule in the future, rather than consider an alternative, and to propose this schedule as a candidate for agreement.

6. Recall that a local document state remains tentative, until the agreement. Telex sites iteratively agree on a common state (Step 5 in Figure 1). The committed state is guaranteed to be compatible with the schedules that the applications *select*ed.

3 Usage example

We examine how to programme a collaborative application using Telex. We illustrate with the example of a cooperative calendar application, Sakura. The Sakura application enables any user to share her calendar, to create and invite users to a meeting, to change its time, or to cancel it. Sakura maintains the invariant that no user is double-booked into two different meetings at the same time. In contrast to common calendar systems such as Doodle[1], Sakura over Telex ensures consistency even when a user is tentatively engaged in several meetings, i.e., the agreement protocol will not commit conflicting meetings.

[1] www.doodle.com

3.1 Designing the application

The first step is to specify the application with regard to the optimistic execution model.

The design must specify: *(i)* the data or documents that the application manages. *(ii)* the prototype of actions performed on the documents *(iii)* and a set of rule defining relations between actions expressed in term of Telex supported constraints, as we illustrate shortly.

For instance, in the Sakura application, documents are calendars and actions are: *createMtg* creates a meeting, *addUser* invites a user to a meeting, and *setDate* sets a meeting time.

Telex supports a predefined set of constraints which the programmer can combine to express a rich semantics. An illustrative rule defining constraints between actions is: “concurrent actions inviting the same user to different meetings on the same time *conflict*”. Another rule is “concurrent *setDate* actions performed on the same meeting *conflict*”.

3.2 Using Telex API for implementation

Once a programmer has designed the application, implementing over Telex is straightforward. Telex API allows the programmer to easily describe the application actions and documents. The programmer need to translate high-level user’s operations into a set of actions and constraints. Additionally, the programmer implements the *getConstraint* interface to reflect the designed relations between actions.

Consider that a user Bob selects “create meeting *MtgA* all day Monday with Lea” in the calendar menu. Sakura calls the *addFragment* primitive with four actions: *createMtg*, *addUser* (one for Bob, another for Lea), and *setDate*. The latter three actions depend on the first one, as captured by the *Causal* constraint. Sakura includes these *Causal* constraints in the *addFragment* call.

Note that in contrast to many systems, Telex does not assume that successive actions are causally related. If two actions are not linked by a constraint, then they are considered independent.

At Lea’s site, to verify whether the received *MtgA* invitation from Bob’s site causes a conflict, Telex up-calls Sakura’s *getConstraint* interface with every suspected invitations. Imagine that the suspected action invites Lea to a meeting *MtgB* on Monday. According to the rule we defined previously, Sakura response will be an *Antagonism* constraint.

4 Telex main features

Collaborative application development faster and easier Telex provides a facility of sharing by taking care of complex application-independent aspects, such as replication, conflict repair, and ensuring eventual commitment. This clear separation of concerns allows the programmer to concentrate on core functionality.

Work in disconnected mode Telex handles disconnection transparently for the application. When a user is offline, Telex continues logging his updates locally, and the user can keep working on the shared document. When he reconnects, Telex automatically sends the local updates, fetches the missing ones, and checks for possible conflicts.

Multidocument / multi application sharing Although documents are the basic sharing unit, Telex does not assumes that documents are always independent. Instead, it allows a user or an application to define a constraint between actions of two distinct documents, in order to enforce

consistency accross documents. This is possible even if these documents are processed by distinct applications as applications share the same engine. Consequently, novel cross-application scenarios are possible.

Ergonomic and more productive collaboration Telex implements several mechanisms that enhance collaboration ergonomics and productiveness:

Filtres Telex allows users to set filters defining their own view of a shared document. For instance, they should see each other updates in real time in the case of shared whiteboard. On the other hand, they can select a personalized view where other updates are ignored when editing a text document, in order not to be overwhelmed by group activity.

Smart conflict detection and resolution In order to be relevant, conflict detection and resolution takes fine grained semantics into account. All possible levels of semantics can be adressed: data semantics, application semantics and user intents.

Besides, Telex conflict resolution algorithms implement heuristics to decouple independent conflicts and computes optimal solution, minimizing the amount of lost work. Thus, Telex provides the application with alternative solutions grouped by conflict, and sorted by quality.

Social collaboration Telex enables users to express preferences for conflict resolution and agreement. For instance, they should be able to retain the most important update, or minimize the amount of lost work.

Guarantees Telex protocols are proven to be correct. They ensure that replicas never violate safety (with regards to the provided constraints) and converge eventually.

5 How Telex compares to other systems

As described previously, Telex as a generic platform is application independent, and serves as a building block for collaborative applications. Consequently, is is not relevant to compare Telex to specific applications. Note however how Telex leverages the semantics an application supports. As an illustration: thanks to Telex, the Sakura application can ensure consistency even when a user is tentatively engaged in several meetings. Another novel feature for Sakura allows a user constraints to group meetings using constraints, so that Telex manages them as a whole. We can thus consider a complex scenario where a professional consultant books a meeting over two consecutive mornings, and propose two alternative dates: Monday and Tuesday; or Thursday and Friday. If a conflicting meeting is scheduled on Monday, then Telex is able to automatically propose Thursday and Friday for the training.

Telex is somewhat similar to Google Wave. Google Wave is also a generic platform to support structured content editing. Wave uses The Operation Transformation (OT) principle to manage conflicts. OT modifies action arguments so that they all commute.

However, Telex is more expressive than Wave. Telex supports arbitrary action types and relations whereas OT is limited to commutative operations and does not consider real conflicts, such as double booking.

Level of confidentiality and dissemination

By default, each document created within Grid4All is © Grid4All Consortium Members and should be considered confidential. Corresponding legal mentions are included in the document templates and should not be removed, unless a more restricted copyright applies (e.g. at subproject level, organisation level etc.).

In the Grid4All Description of Work (DoW), and in the future yearly updates of the 18-months implementation plan, all deliverables listed in Section 7.7 have a specific dissemination level. This dissemination level shall be mentioned in the document (a specific section for this is included in the template, both on the cover page and in the footer of each page).

The dissemination level can be defined for each document using one of the following codes:

PU = Public.
PP = Restricted to other programme participants (including the EC services).
RE = Restricted to a group specified by the Consortium (including the EC services).
CO = Confidential, only for members of the Consortium (including the EC services).
INT = Internal, only for members of the Consortium (excluding the EC services).

This level typically applies to internal working documents, meeting minutes etc., and cannot be used for contractual project deliverables.

It is possible to create later a public version of (part of) a restricted document, under the condition that the owners of the restricted document agree collectively in writing to release this public version. In this case, a new document code should be given so as to distinguish between the different versions.