Interdroid Versioned Databases: Decentralized Collaborative Authoring of Arbitrary Relational Data for Android Powered Mobile Devices

Emilian Miron, Nicholas Palmer, Roelof Kemp, Thilo Kielmann, and Henri Bal

VU University Amsterdam, The Netherlands
`emilian.miron@gmail.com`, `{palmer,rkemp,kielmann,bal}@cs.vu.nl`

Abstract. Complex interactions in software development have led to the creation of *version control systems* to manage source code. These systems have become increasingly flexible and support *disconnected* and *decentralized* operations.

However the authoring of other types of data has remained behind and often relies on centralized and online solutions. This is a big problem in the mobile market where applications are encouraged to use relational databases to store their information but little help is offered to allow synchronization and collaboration with peers.

The Interdroid Versioned Databases (IVDb) framework is an integrated solution for collaboratively editing and sharing of arbitrary relational data on the Android mobile platform. It is a reusable framework that addresses the storage, versioning and synchronization needs of applications, freeing developers of considerable design and programming effort.

1 Introduction

People today may own and use several computing devices including personal desktop computers, laptops, netbooks, tablets, and smartphones. Such devices are paving the way for what pervasive and ubiquitous computing advocates have long described[9]. It is certain that the way we communicate and collaborate today has significantly changed to include and take advantage of these devices.

In particular *digital collaborative authoring* of many forms has become popular. The most familiar examples of such collaborations include collaborative web applications such as Wikipedia or Google Docs, but businesses are also increasingly using collaborative workflows to design presentations, prepare documents, create software models and author source code. One can think of even more common things that are managed digitally and shared between members of a family or community including shopping and todo lists, personal financial accounting, and even medical records.

One would like to bring these kinds of collaborative applications to mobile devices. However, due to frequently changing connectivity status on these devices, it is important for applications to offer *disconnected* and *decentralized* operation. Unfortunately, writing a system which supports these modes of operation requires a large development effort for the application's data storage and synchronization layer. Because of the difficulties of distributed synchronization, applications often settle for centralized and online solutions which do not meet the needs of mobile users and introduce single points of failure.

The system presented in this paper addresses these problems by leveraging the power of version control systems to bring decentralized collaborative editing of relational databases to the mobile platform. This system is based on the popular Git version control system and the Android[1] mobile platform. The end result of this work is the Interdroid Versioned Databases (IVDb) framework which provides a platform for distributed authoring of relational data.

The IVDb framework is an integrated solution for collaboratively editing and sharing of arbitrary relational databases on Android. It is a reusable framework that addresses the general storage, versioning and synchronization needs of applications which require data storage, freeing developers of considerable design and programming effort.

The API of this system was designed for the Android platform and with developer familiarity in mind. Versioning blends naturally with the existing data access mechanisms built into the platform. The framework also includes most tools needed to bootstrap a collaborative editing application in the form of user interface activities that deal with versioning related tasks and can easily interface with pluggable data editing components written by the application developer.

The unique set of features offered by this system unlocks the door for collaborative applications using structured data stores on mobile devices.

2 Background

Git is fundamentally a content oriented filing system with version tracking. Git has the ability to track the state of a filing system based entirely on the hash of the content of the filing system. Git is also able to track multiple branches of a given history, and provides access to the history of commits as well as to individual commits in a given branch. With this structure, and feature set, Git is able to solve a portion of the problems we face when building a distributed collaborative editing system for structured data.

Android[1] is an open source software stack designed for mobile phones. Developers can write custom applications in the Java programming language by using the same platform features that the core applications use. The Android application framework provides several abstractions specifically designed for the platform. Of primary importance for this paper is an integration with the existing content provider interface.

Content providers are an abstraction of the Android application framework used for storage and retrieval of data across applications. Android uses it to expose platform data such as contacts and media information, while applications can use existing content providers or write new ones. The content providers differ from relational databases in how the tables are named. Instead of the flat names and the notion of cross product joins, the tables of content providers are presented within a URI scheme of content://. Each content provider corresponds to an authority, while the path represents either a table or a table + row identifier. In addition to the relational database operation the content provider interface also associates a MIME type for each content URI. These MIME types are used in conjunction with the intent system, in order to allow the Android framework to select viewing and editing activities for specific URIs.

3 Design

The design of the system is driven by the vision of enabling collaborative editing of structured data stores. We thus require the system to enable applications to perform the following activities: provide access to read and modify data, save and access historical versions, branch and merge, and share data with others. Finally, we require the system to integrate well with the Android platform, provide tools for easily transitioning existing applications, work well on resource constrained platforms, and operate in a complete decentralized fashion.

The collaboration and versioning features are an extension to the Android content provider data abstraction interface. Our solution keeps track of versioning information alongside the existing non-versioned data abstraction. For instance, the branch or commit we are accessing can be embedded in either the URI passed to the content provider or via the parameters of the operation. We have chosen for the former because it is more natural to think of a particular row in a particular version as a given resource rather than as an aspect of the operations on the resource. Furthermore, this decision also reduces the effort to migrate existing applications because most often content URIs are generated using a fixed prefix and thus the system allows application programmers to alter that prefix in a single place rather than having to alter every operation. Thus, the framework makes use of a URI scheme which embeds the versioning information inside the URIs.

All URIs begin with a “content” scheme. They then specify the “authority” for the given data, which can be thought of as a namespace, and then follow with a path for the given data item or items. In the case of IVDb all repositories are served using the same authority. The first path component is therefore the

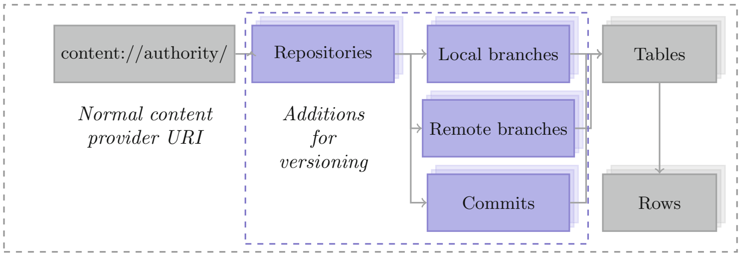

Fig. 1. Versioned Content Provider URIs

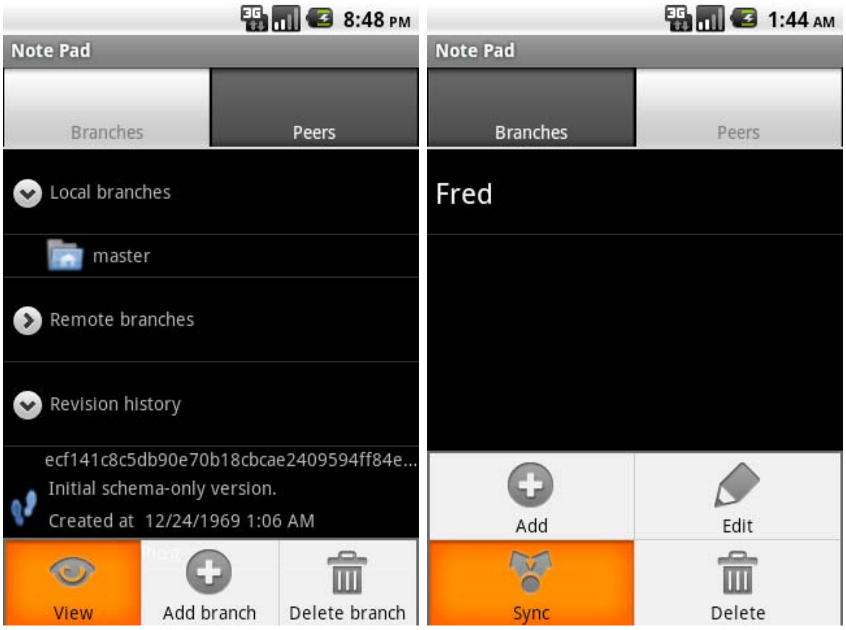

Fig. 2. Screen Shots: On the left is a shot of our Version Browser, and on the right the IVDb Sharing Manager

repository. Following this is the branch type identifier, and then the actual branch identifier. When migrating a legacy application the URI prefix can be easily changed from the content schema and authority to the default local master branch in the framework provider. Finally, the remaining path identifies the table and optional row identifier. This design is pictured in Figure 1.

4 User Interface Components

IVDb includes several reusable user interface components (activities and views) which can be used for bootstrapping a new or existing application with the versioning related user interface. With these components, application developers can focus on writing application specific data visualization user interface components. (See Figure 2.)

The activities in the framework handle the following tasks: (a) Version browser: manage local branches (add, delete), view list of versions (local branches, remote branches and historical versions), commit changes as well as (b) Sharing manager: manage remote peers, synchronize with others. The activities launch related activities based on Intents with URIs from the uniform versioned Content Provider URI scheme.

The custom application components and the activities from the framework can easily interface with each other by using Android's intent mechanism.

5 Evaluation

In order to evaluate the framework the Notepad Content Provider sample from the Android platform development kit was modified to use the framework while keeping the same data model.

The utility of the API can be quantified in lines of code saved when providing the same features using the framework versus when not using the framework. The ease of use is correlated with the utility but is more difficult to quantify as it involves how well the framework integrates with the platform and how familiar the interfaces are to developers. In general we found the modification to be extremely easy to perform. We add 55 lines of code to the user interface code in order to deal with viewing read only data which the original application did not support. However, the content provider implementation was reduced from 204 lines of code to just 70, a significant savings for this application. We anticipate applications with more complicated data types to have a higher percentage of savings in the content provider.

We feel that a small addition in lines of code to the UI in order to add explicit commit, branching, synchronization and viewing of historical versions is a very small price to pay, particularly when offset by the significant reduction in complexity of the content provider implementation. In total the codebase for the versioned application was smaller than the original application while providing more features. This shows that the framework has powerful abstractions that lower development effort while at the same time providing additional versioning features developers would otherwise not write.

6 Related Work

A great deal of work has been done on both synchronization and collaborative editing on mobile devices.

The closest system to ours with regards to data format is Bayou[6]. It is a replicated mobile database that provides eventual consistency over relational data, however, the requirement to have deterministic conflict resolution programs is admittedly very hard or even impossible without user intervention.

Byong et al[3] advocate that many collaborative writing scenarios require asynchronous disconnected operations. They developed a synchronization system for XML documents, which limits the applicability of their solution.

In Syxaw[5] the authors give a general synchronization system for files based on merging of XML data. However, their work mainly deals with optimizing bandwidth usage because of their focus on file system synchronization. Closest to IVDb is their sample collaborative XML editor called Captio which aids users by providing better merging alignment and visualization.

DocX2Go[7] makes use of optimistic replication just as IVDb does, and can work in a fully decentralized way with support. However, the XML focus of the platform, as with other related work can make it inappropriate for many applications.

Several authors emphasize the importance of differencing for the merge problem. Ronnau et al.[8] devise an algorithm that differentiates XML documents generated by the OpenOffice application in a way that notices structural differences as opposed to variability of the format. They also advocate the use of history aware merging such as the one we provide. In [4] Tancred proposes a theoretical model to XML document merging from which he devises a three-way merge algorithm.

In the Disco[2] framework the authors explore how applications can handle operation of collaborative systems in the face of disconnections. This system focuses more on handling disconnection in synchronous systems and not on data representation and versioning.

7 Conclusions

This paper presented IVDb, a framework for the development of collaborative editing applications using structured data for mobile devices. It offers developers of collaborative applications a simple way to structure and build their application, which can be used to significantly reduce development overhead, while also offering versioning features that would otherwise be very demanding to develop.

References

1. Android-Developers-Guide: What is android?, `http://developer.android.com/guide/basics/what-is-android.html`
2. Gutwin, C., Graham, T.N., Wolfe, C., Wong, N., de Alwis, B.: Gone but not forgotten: designing for disconnection in synchronous groupware. In: CSCW 2010: Proceedings of the 2010 ACM Conference on Computer Supported Cooperative Work, pp. 179–188. ACM, New York (2010)
3. Lee, B.G., Chang, K.H., Narayanan, N.H.: An integrated approach to version control management in computer supported collaborative writing. In: ACM-SE 36: Proceedings of the 36th Annual Southeast Regional Conference, pp. 34–43. ACM, New York (1998)
4. Lindholm, T.: A three-way merge for xml documents. In: DocEng 2004: Proceedings of the 2004 ACM Symposium on Document Engineering, pp. 1–10. ACM, New York (2004)
5. Lindholm, T., Kangasharju, J., Tarkoma, S.: Syxaw: Data synchronization middleware for the mobile web. Mob. Netw. Appl. 14(5), 661–676 (2009)
6. Petersen, K., Spreitzer, M., Terry, D., Theimer, M.: Bayou: Replicated database services for world-wide applications. In: Proceedings of the 7th SIGOPS European Workshop, pp. 275–280. ACM (1996)
7. Puttaswamy, K.P., Marshall, C.C., Ramasubramanian, V., Stuedi, P., Terry, D.B., Wobber, T.: Docx2go: collaborative editing of fidelity reduced documents on mobile devices. In: MobiSys 2010: Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services, pp. 345–356. ACM, New York (2010)
8. Rönnau, S., Scheffczyk, J., Borghoff, U.M.: Towards xml version control of office documents. In: DocEng 2005: Proceedings of the 2005 ACM Symposium on Document Engineering, pp. 10–19. ACM, New York (2005)
9. Satyanarayanan, M.: Pervasive computing: Vision and challenges. IEEE [see also IEEE Wireless Communications] Personal Communications 8(4), 10–17 (2001)